МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ
(підпис)
19 травня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-наукової програми «Інформатика»
на тему: «Інформаційна технологія ієрархічного машинного навчання системи виявлення кіберзагроз»
здобувачки групи ІН.м-11н Теницької Альони Олексіївни

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

_____ Альона ТЕНИЦЬКА
(підпис)

Керівник,
професор,
доктор технічних наук, професор Анатолій ДОВБИШ _____
(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
зі спеціальності 122 - Комп'ютерних наук, освітньо-наукової програми «Інформатика»
здобувача групи ІН.м-11н Теницької Альони Олексіївни

1. Тема роботи «Інформаційна технологія ієрархічного машинного навчання системи виявлення кіберзагроз»
затверджую наказом по СумДУ від «08» травня 2023 р. № 0475-VI

2. Термін здачі здобувачем кваліфікаційної роботи до 19 травня 2023 року

3. Вхідні дані до роботи _____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Огляд сучасного стану та перспектив розвитку систем виявлення кіберзагроз; 2) Аналіз методів виявлення кіберзагроз та аналітичний огляд методів машинного навчання; 3) Розробка та розгляд алгоритму системи виявлення кіберзагроз 4) Програмна реалізація системи виявлення кіберзагроз; 5) Аналіз результатів.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): *Слайди: 1) Титульний аркуш; 2) Актуальність; 3) Мета роботи, об'єкт та предмет дослідження; 4) Функціональна категорійна модель; 5) Вхідні дані; 6) Опис алгоритму машинного навчання; 7) Результат комп'ютерного моделювання; 8) Висновки.*

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ (підпис) Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Огляд сучасного стану та перспектив розвитку систем виявлення кіберзагроз*	03.04-09.04.23	
2	*Аналіз методів виявлення кіберзагроз та аналітичний огляд методів машинного навчання*	10.04-15.04.23	
3	*Розробка інформаційно-екстремальної інтелектуальної технологію машинного навчання за ієрархічною структурою даних для системи виявлення кіберзагроз*	16.04-21.04.23	
4	*Програмна реалізація системи виявлення кіберзагроз за розробленим алгоритмом*	22.04-05.05.23	
5	*Аналіз отриманих результатів*	06.05-09.05.23	
6	*Оформлення пояснювальної записки до кваліфікаційної роботи магістра*	10.05-14.05.23	

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 69 стор., 14 рис., 4 таблиці, 1 додаток, 31 літературне джерело.

Обґрунтування актуальності теми роботи – тема кваліфікаційної роботи магістра є актуальною, оскільки присвячена розв'язанню важливої практичної проблеми протидії кіберзагрозам, а саме розробці інформаційно-екстремальної інтелектуальної технології машинного навчання системи виявлення кіберзагроз за ієрархічною структурою даних.

Об'єкт дослідження — процес машинного навчання системи виявлення кіберзагроз.

Мета роботи — підвищення кіберзахищеності інфокомунікаційної системи шляхом машинного навчання системи виявлення кіберзагроз за ієрархічною структурою даних.

Предмет дослідження — функціональні категорійні моделі та метод виявлення кіберзагроз з використанням інформаційно-екстремального машинного навчання за ієрархічною структурою даних.

Результати — проведено аналіз існуючих видів, а також методів системи виявлення кіберзагроз. Розроблено алгоритм та програмне забезпечення системи виявлення кіберзагроз, застосовуючи інформаційно – екстремальне машинне навчання за ієрархічною структурою даних. Проведено навчання системи на реальних вхідних даних, взятих з відкритого репозиторію. Алгоритм розроблено за допомогою об'єктно-орієнтованої мови програмування Python.

ІЄРАРХІЧНА СТРУКТУРА ДАНИХ, ДЕКУРСИВНЕ ДЕРЕВО, ІНФОРМАЦІЙНО-ЕКСТРЕМАЛЬНА ІНТЕЛЕКТУАЛЬНА ТЕХНОЛОГІЯ, СИСТЕМА ВИЯВЛЕННЯ КІБЕРЗАГРОЗ, МАШИННЕ НАВЧАННЯ

ЗМІСТ

ВСТУП

Зростання кібератак є однією з найактуальніших проблем сьогодення, адже статистика 2022 року свідчить, що кількість успішних атак збільшилась майже на третину в порівнянні з попереднім роком. Більш того, журнал Cyber Defence пише, що більше 40% кібератак спрямовані на малий бізнес. Така статистика свідчить про те, що існуючі засоби захисту не можуть в повній мірі виявляти та блокувати нові атаки, а також їх різновиди.

У зв'язку з цим, останніми роками значну увагу приділяють саме інструментам кібербезпеки, до яких належать антивіруси, брендмауери, програми для шифрування даних, системи виявлення кіберзагроз тощо. Основні надії покладають на систему виявлення кіберзагроз, адже вона може показати найкращу ефективність в захисті мережі.

Сучасні системи виявлення кіберзагроз мають високі вимоги адже вони повинні не тільки виявляти вже існуючі атаки, а й в режимі реального часу підлаштовуватися та ідентифікувати нові модифікації, при чому маючи гарну швидкодію та невеликі обчислювальні вимоги. Існуючі системи, які працюють за сигнатурним метод та методом виявлення аномалій, а також штучні нейронні мережі (ШНМ) не є ефективними, адже для них властиві помилки першого та другого роду за умови суттєвого перетину класів розпізнавання у просторі ознак.

Новітнім та перспективним методом систем виявлення кіберзагроз є інформаційно – екстремальне машинне навчання за лінійним алгоритмом, побудовані за результатам якого вирішальні правила гарантують високу достовірністю розпізнавання кібератак, але все ж вони не є безпомилковими. Тому, для розширення потужності класу розпізнавання та забезпечення безпомилковості розпізнавання кібератак необхідно розробити алгоритм інформаційно-екстремального машинного навчання за ієрархічною структурою даних.

1 АНАЛІЗ ПРОБЛЕМИ ДОСЛІДЖЕННЯ

1.1 Поточний стан та перспективи розвитку системи виявлення кіберзагроз

Захист системи від кібератак досягається в основному за допомогою використання демілітаризованих зон, брандмауерів, шифрування даних, мереж VPN (IPsec), сегментації мережі, перевірки особистості, авторизації доступу та керування паролями. Виявлення кібератак здійснюється шляхом моніторингу мережевого трафіку, який не дозволяє виявити всі аномалії [1]. Рішення, які використовуються в IT-системах, не гарантують відсутність можливості проникнення атаки в систему. Таким чином, впроваджується її деструктивний вплив на керований процес [2]. Раннє виявлення аномалій та їх ізоляція мають важливе значення для ефективного реагування на нові небезпеки та загрози. Тому, системи виявлення кіберзагроз є одним з найперспективніших напрямків в сфері кібербезпеки.

Головною метою системи виявлення кіберзагроз (СВК, Intrusion Detection System) є ідентифікація та фільтрація потенційно шкідливих запитів (атак) в комп’ютерних мережах. СВК відстежує та аналізує мережевий трафік, який надходить або виходить із мережевих пристроїв організації, і подає тривогу, якщо спостерігається вторгнення [3].

Відповідно до особливостей застосування, існує широкий спектр класифікації систем виявлення кіберзагроз, а саме класифікація за технологіями побудови, за принципом роботи, за джерелом аудиту, за джерелом збору даних тощо. Характеристики, які вважаються найбільш важливими при виборі системи виявлення кіберзагроз, зазвичай обмежуються двома: методами виявлення атак та середовищем моніторингу [4]. У своїй роботі я зосереджуватимусь саме на цих двох аспектах.

Розглянемо види СВК за способом моніторингу системи [5]:

1. Хостова система виявлення кіберзагроз (Host-based intrusion detection system або HIDS). Вона працює на окремих пристроях із метою виявлення підозрілої чи зловмисної активності та сповіщення адміністратора про це. HIDS моніторить всі вхідні та вихідні пакети, а також періодично робить знімок системних файлів, порівнюючи його з попереднім результатом. У разі виявлення змін або видалення файлів, система надсилає сповіщення адміністратору для подальшого дослідження. HIDS є особливо корисним на важливих пристроях, де зміна конфігурації не очікується.
2. Мережева система виявлення кіберзагроз (Network intrusion detection system або NIDS). Вона встановлюється в точках мережі для аналізу трафіку з усіх пристроїв у мережі. NIDS виконує постійне спостереження за трафіком, що проходить у всій мережі, та порівнює трафік із базою відомих атак. При виявленні ненормальної поведінки або ж якогось з видів атаки, NIDS може надіслати сповіщення адміністратору.
3. Система виявлення кіберзагроз на основі протоколу (Protocol-based Intrusion Detection System або PIDS) – це система, яка спостерігає за протоколами, що використовуються в мережі, і знаходить аномалії в їх поведінці. PIDS може використовуватись для захисту веб-серверів, він розміщується на передній частині сервера і відстежує вхідний та вихідний трафік протоколів HTTP і HTTPS. Система використовує різні методи, включаючи евристичний аналіз трафіку і порівняння з зразками зловмисних атак, щоб виявляти підозрілу активність. PIDS може бути корисним при захисті мережі від шкідливих програм, що використовують певні протоколи для передачі даних, наприклад, протоколів FTP та SMTP.
4. Система виявлення кібзагроз на основі протоколу додатків (Application Protocol-based Intrusion Detection System або APIDS).

APIDS є унікальним видом системи виявлення кіберзагроз, оскільки він спеціально зорієнтований на відстеження протоколів, які використовуються конкретними програмами. Він розміщується на групі серверів та моніторить та інтерпретує зв'язок за цими протоколами, що дозволяє виявляти потенційно шкідливу активність, пов'язану з цими програмами. Наприклад, він може відстежувати протокол SQL, що використовується для зв'язку між програмним забезпеченням та базою даних на веб-сервері, і виявляти будь-які спроби вторгнення або злому.

5. Гібридна система виявлення кіберзагроз (Hybrid Intrusion Detection System). Цей вид СВК може бути реалізований за допомогою різних підходів до СВК, наприклад, поєднання NIDS та HIDS або APIDS та NIDS. Вона може бути корисною в тому, що поєднує переваги різних підходів та забезпечує більш ефективний захист від кіберзагроз. Наприклад, гібридна система може використовувати мережевий аналіз для виявлення аномальних поведінок, що вказують на можливість кібератаки, а також використовувати аналіз логів та моніторинг хостів для виявлення конкретних вразливостей та атак на окремі системи.

Резюмуючи вищевикладене, можна зазначити, що гібридна система виявлення кіберзагроз є найбільш ефективною в порівнянні з іншими видами.

Розробці систем виявлення кіберзагроз було покладено початок ще в 1990-х роках. Безліч вчених працюють в напрямку підвищення рівня безпеки СВК, до них належать як закордонні, так і вітчизняні дослідники. Серед них: Дудикевич В. Б., Гнатюк С. О., Кузнецов О. О., Бурячок В. Л., Субач І. Ю., Казмирчук С. В., Alber P., Ptaceka T., Camp O. та інші [6]. На сьогоднішній день існує безліч комерційних та безкоштовних рішень у цій галузі. Робота [7] наводить характеристики найбільш поширених та відомих СВК, які з'явилися на ринку останнім часом. До них належать Snort, OSSEC, Suricata, Zeek,

Security Onion, Open WIPS-NG, Sagan, SolarWinds Security Event Manager та Samhain. Аналізуючи характеристики цих СВК, можна зробити висновок, що не зважаючи на їх переваги та досить непогані показники роботи, вони мають свої недоліки. До головних недоліків можна віднести те, що всі вони працюють як HIDS або NIDS, а не як гібридні, що було б найкращим варіантом, враховуючи вимоги до СВК. Також кожне з цих рішень працює тільки за одним з існуючи методів виявлення атак, хоча, на сьогоднішній день підвищена увага приділяється СВК, що об’єднуються обидва з представлених методів. Більш того, наразі досі залишається невирішеною проблема досягнення безпомилковості спрацювання системи. В цілому, існуючі системи виявлення кіберзагроз мають численні недоліки та не відповідають сучасним вимогам ергономіки та ефективності безпекових рішень. Однак покращення їх ефективності є необхідним не лише для виявлення зловмисних дій на інфраструктурі захищених об'єктів інформатизації, а й для забезпечення експлуатації цих засобів у повсякденні, а також, щоб забезпечити економію інформаційних і обчислювальних ресурсів власника СВК. Тому, розробка та розгляд новітніх методів СВК є перспективним та актуальним напрямком роботи.

1.2 Аналіз методів виявлення кіберзагроз

Існують два основні методи, за допомогою яких системи виявлення кіберзагроз можуть попередити адміністраторів про ознаки загрози. До них належать метод виявлення аномалій та сигнатурний метод.

Традиційно, сигнатурний метод використовують для виявлення вже відомих загроз. Він працює за допомогою попередньо програмованого списку відомих загроз та їх індикаторів компрометації (Indicator of Compromise, IoC). IoC може бути певною поведінкою, яка загалом передує зловмисному мережевому нападу: хешами файлів, шкідливими доменами, відомими байтовими послідовностями, або навіть вмістом теми електронної пошти.

Оскільки СВК на основі сигнатур відстежує пакети, що проходять мережею, вона порівнює ці пакети з базою даних відомих IoC або сигнатур атак, щоб позначити будь-яку підозрілу поведінку.

Системи виявлення кіберзагроз, які базуються на аномаліях, можуть повідомити про підозрілу поведінку, яка є досі невідомою. Замість пошуку відомих загроз, СВК на основі методу виявлення аномалій використовує машинне навчання, щоб навчити систему виявляти нормалізовану базову лінію. Базова лінія описує те, як зазвичай працює система. Всю мережеву активність порівнюють з цією базовою лінією. Замість пошуку відомих IoC, СВК просто ідентифікує будь-яку незвичну поведінку, щоб викликати сповіщення про загрозу.

З використанням СВК на основі виявлення аномалій будь-що, що не відповідає існуючій нормалізованій базовій лінії, такі як спроба користувача увійти в систему поза стандартними робочими годинами, додавання нових пристроїв до мережі без авторизації або спроби підключення до мережі з великої кількості нових IP-адрес – викличуть потенційні попередження про загрозу. Недолік тут полягає в тому, що багато не зловмисних поведінок будуть викликати сповіщення просто за те, що вони не типові. Підвищена ймовірність помилкових спрацьовувань може вимагати додаткового часу та ресурсів для дослідження всіх сповіщень про потенційні загрози.

У той же час, СВК на основі сигнатур не може виявляти атаки нульового дня, або нові атаки, які відсутні в базі існуючих загроз. Цей суттєвий недолік робить СВК, яка працює за цим методом не ефективною, враховуючи швидкість генерації новітніх атак сьогодення, незважаючи, навіть на високу швидкість обробки та більшу точність для відомих атак.

Ці два методи виявлення кібератак мають свої переваги та недоліки, які загалом добре доповнюють один одного і найкраще використовуються в тандемі, тому основна уваги приділяється саме розробці гібриду цих методів.

1.3 Аналітичний огляд методів машинного навчання

Виявлення кіберзагроз є складною задачею, оскільки кіберзлочинці постійно шукають нові способи обходу систем захисту. Для ефективного виявлення вторгнень необхідно розробляти нові методи та алгоритми, які будуть адаптивними до змінних умов та можуть виявляти незвичні та приховані атаки. Наприклад, наразі для вирішення цієї задачі широко застосовуються методи машинного навчання. Серед них нечітка логіка [11], нейронні мережі [8], методи аналізу даних (Data Mining) [12], генетичні алгоритми [10], метод опорних векторів [9] та інформаційно-екстремальна інтелектуальна технологія аналізу даних.

Нечітка логіка була запропонована у 1965 році Лотфі А. Заде. Вона заснована на теорії можливостей та забезпечує робочий простір для обчислень за допомогою слів і допомагає керувати невизначеністю під час проектування експертних систем. Нечітка логіка стала невід'ємною частиною машинного навчання, оскільки може працювати з неточними та невизначеними ситуаціями. Цей алгоритм машинного навчання широко використовується для виявлення вторгнень, адже будує модель виявлення. Нечітка логіка – це надмножина булевої логіки, яка була розширена для обробки концепції істинних значень між повністю істинними та повністю хибними. У той час як булева логіка допускає лише значення true або false, нечітка логіка дозволяє будь-яке значення між ними [13].

Архітектура нечіткої логіки складається з чотирьох основних частин:

- правила – містить всі правила та умови if-then, які пропонуються експертами для управління системою прийняття рішень. Останнє оновлення у теорії нечіткої логіки надає різні ефективні методи проектування та налаштування нечітких контролерів;
- фазифікація – перетворює вхідні дані або чіткі числа на нечіткі множини. Можливо вимірювати чіткі вхідні дані за допомогою

датчиків та передавати їх у систему управління для подальшої обробки;

- логічний висновок – визначає ступінь відповідності між нечітким введенням і правилами. В залежності від вхідного поля він вирішує правила, які мають бути виконані. Комбінуючи виконані правила, визначається дія керування;
- дефазифікація – перетворює нечіткі набори на чітке значення.

Переваги застосування нечіткої логіки:

- гнучкість та можливість легко адаптуватися до різних умов;
- легкість реалізації;
- можливість одночасного оброблення вхідних даних і прийняття точних рішень;
- легкість прийняття рішення, адже система нагадує людські міркування;
- не вимогливість до обчислювальних ресурсів системи.

Недоліки застосування нечіткої логіки:

- вимога до частоти оновлення системи керування;
- повна залежність від людського фактору: досвіду та інтелекту;
- невисока ефективність, адже система працює на нечітких входах;
- обмежене використання через доведеність неточності прийняття рішень.

Нейронні мережі також широко застосовуються для систем виявлення кіберзагроз. Вони пропонують потенціал для вирішення ряду проблем, з якими стикаються інші підходи до виявлення кібератак. Даний алгоритм машинного навчання був запропонований як альтернатива компоненту статистичного аналізу систем виявлення аномалій [14]. Вони призначені для розпізнавання шаблонів даних і навчання на них, що робить їх підходящими для виявлення аномалій і визначення потенційних загроз безпеці.

У СВК із використанням нейронних мереж мережа навчається з використанням великого набору позначених даних, таких як журнали мережевого трафіку або журнали активності системи, для вивчення нормальних моделей поведінки в мережі. Після того, як мережу було навчено, її можна використовувати для виявлення відхилень від цієї нормальної поведінки, які можуть вказувати на атаку або спробу вторгнення.

Машинне навчання нейронних мереж можливе за одним з цих методів:

- контрольоване навчання або навчання з вчителем – включає навчання нейронної мережі на позначених даних, де кожен зразок позначається як нормальний або аномальний. Потім мережа використовує це навчання для виявлення майбутніх аномалій;
- неконтрольоване навчання або навчання без вчителя – навчання нейронної мережі на даних без будь-яких попередньо визначених категорій або міток. Потім мережа визначає шаблони в даних і об’єднує подібні дані, що дозволяє ідентифікувати аномальну активність;
- глибоке навчання – передбачає використання глибоких нейронних мереж із багатьма рівнями для вивчення складних закономірностей і зв’язків у даних, що дозволяє точніше виявляти аномалії.

До переваг нейронних мереж належить:

- можливість самонавчання та самостійної генерації вхідних даних;
- багатозадачність, а точніше можливість виконувати декілька завдань паралельно;
- вміння працювати з неповними даними;
- здатність виявити несправність та створити вихід у тому випадку, коли нейронна мережа несправна;
- ефективна робота з великими масивами даних;
- відсутність вимоги до наявності окремої бази даних, адже вхідні дані зберігаються у власній мережі.

Серед недоліків нейронних мереж можна виділити:

- «чорна скринька»;
- ефективне навчання нейронної мережі вимагає великої кількості даних;
- вимога до значних обчислювальних ресурсів;
- апаратна залежність;
- складність реалізації;
- невідома тривалість.

Генетичний алгоритм – це тип еволюційного алгоритму, натхненний процесом природного відбору. Його можна використовувати в системах виявлення кіберзагроз для оптимізації вибору та налаштування функцій, що використовуються для виявлення. Цей метод ідентифікує та розраховує відмінності між поведінкою несанкціонованого з'єднання та нормального з'єднання за допомогою запропонованої відповідної (цільової) функції [15].

В СВК за допомогою генетичних алгоритмів генерується сукупність варіантів рішень, де кожне рішення-кандидат представляє можливу конфігурацію функцій для виявлення кіберзагроз. Потім ці рішення оцінюються за допомогою функції відповідності, яка вимірює їхню ефективність у виявленні відомих атак. Наступним кроком генетичний алгоритм проходить через серію поколінь, де найкращі рішення з кожного покоління відбираються для розведення та створення наступного покоління рішень-кандидатів. Цей процес розведення передбачає поєднання характеристик вибраних рішень для створення нових рішень-кандидатів із потенційно кращою продуктивністю. Операції мутації та кросинговеру також використовуються для введення варіативності в рішення-кандидати та дослідження нових областей простору рішень. Операція мутації випадковим чином змінює деякі функції рішення, тоді як операція кроссоверу поєднує функції двох або більше батьківських рішень. Генетичний алгоритм продовжується, доки не буде знайдено задовільний розв'язок на основі функції

відповідності. Це рішення представляє оптимальний набір функцій для виявлення вторгнень в IDS.

До переваг генетичного алгоритму можна віднести:

- зрозуміла концепція;
- підходять для паралельної обробки, що дозволяє їм використовувати переваги сучасного апаратного забезпечення;
- стохастичність;
- підтримує багатоцільову оптимізацію;
- різноманітність.

Серед недоліків генетичного алгоритму виділяють:

- повільна конвергенція;
- складність налаштування параметрів;
- залежність від представлення, адже вибір поганого представлення може привести до поганих результатів;
- високі обчислювальні вимоги;
- реалізація вимагає багато часу.

Метод опорних векторів – це, в основному, керований метод машинного навчання, призначений для двійкової класифікації [16]. Він аналізує дані та розпізнає шаблони, які використовуються для ефективної класифікації та проведення регресійного аналізу. У контексті виявлення кіберзагроз метод опорних векторів можна використовувати для ідентифікації зловмисного або аномального мережевого трафіку.

Метод опорних векторів працює шляхом знаходження оптимальної гіперплощини, яка розділяє точки даних на різні класи. Він намагається знайти гіперплощину, яка максимізує запас між класами. Маржа — це відстань між гіперплощиною та найближчими точками даних кожного класу. Алгоритм має на меті знайти гіперплощину, яка максимізує маржу, водночас правильно класифікуючи всі точки даних.

До переваг методу опорних векторів відносять:

- здатність оброблять дані великого розміру та нелінійні межі рішень;
- добре працюють з напівструктурованими, та навіть неструктурованими даними;
- добре масштабуються;
- мають узагальнення на практиці.

Серед недоліків методу опорних векторів виділяють:

- вимогливість до обчислювальних ресурсів при роботі з великими масивами даних;
- чутливість до вибору функції ядра;
- складність налаштування параметрів ядра, що використовується для класифікації.

Методи аналізу даних (Data Mining) – це процес виявлення закономірностей, тенденцій і аномалій у великих наборах даних. Методи аналізу даних можуть бути використані в системах виявлення кіберзагроз з метою аналізу даних мережевого трафіку, системних журналів та інших відповідних джерел для виявлення можливих загроз безпеці.

Data Mining в СВК включає кілька кроків: підготовку даних, дослідження даних, моделювання та оцінку. Етап підготовки даних передбачає збір і попередню обробку даних з різних джерел, включаючи журнали мережевого трафіку, системні журнали та журнали подій безпеки. Етап дослідження даних передбачає дослідження даних для виявлення шаблонів і аномалій, які можуть вказувати на загрози безпеці. Це можна зробити за допомогою різних методів інтелектуального аналізу даних, таких як кластеризація, аналіз правил асоціації та класифікація. Етап моделювання передбачає розробку моделі, яка може точно класифікувати мережевий трафік як нормальний або ненормальний. Це можна зробити за допомогою різних контрольованих і неконтрольованих алгоритмів машинного навчання, таких як дерева рішень, нейронні мережі та метод опорних векторів. Коли модель розроблена, її можна використовувати для виявлення потенційних загроз

безпеці в режимі реального часу шляхом аналізу даних мережевого трафіку під час їх створення. Етап оцінювання включає перевірку ефективності моделі інтелектуального аналізу даних за допомогою різних показників, таких як точність, запам’ятовування та оцінка F1.

Переваги Data Mining:

- здатність виявляти раніше не відомі загрози;
- здатність аналізувати великі обсяги даних у режимі реального часу;
- здатність ідентифікувати шаблони та аномалії, які можуть бути пропущені традиційними інструментами безпеки.

До недоліків можна віднести:

- можливість отримання хибно-позитивних і хибно-негативних результатів;
- необхідність ретельної підготовки й обробки даних;
- вимогливість до обчислювальних ресурсів;
- ризик переналаштування моделі інтелектуального аналізу даних для навчальних даних.

Інформаційно-інтелектуальна екстремальна технологія є сучасною технологією машинного навчання, розробленою під керівництвом професора А. С. Довбиша в Сумському державному університеті. Дана технологія набула застосування у різних галузях людської діяльності, до якої відносять промисловість [17, 18], медицину [19, 20], економіку [21, 22], освіту [23] та військову діяльність [24, 25]. Це свідчить про її ефективність та можливість успішного використання для різних цілей.

В сфері кібербезпеки застосування цієї технології розглядалося ще в роботах [7, 26]. Ці роботи доводять що IEI-технологія є ефективною і у обраній сфері, але все ж потребую підвищення безпомилковості прийняття рішень. Застосування IEI-технології за ієрархічною структурою даних дозволить розширити потужність класу розпізнавання та забезпечити безпомилковість розпізнання кіберзагроз.

2 ОПИС МЕТОДУ ДОСЛІДЖЕННЯ

2.1 Науково-методологічні основи інформаційно-екстремальної інтелектуальної технології аналізу даних

Машинне навчання в рамках ІЕІ-технології розвивається навколо ідеї трансформації вхідного нечіткого розбиття простору ознак в чітке розбиття класів еквівалентності. Це досягається за допомогою ітераційного процесу оптимізації параметрів функціонування ІС. Одночасно з цим здійснюється оптимізація параметрів функціонування системи контрольних допусків та пошук глобального максимуму багатоекстремальної функції статистичного інформаційного критерію. Це дозволяє визначати та відновлювати оптимальні роздільні гіперповерхоні в радіальному базисі бінарного простору ознак розпізнавання, що забезпечує ефективне розбиття простору ознак на класи еквівалентності. Методи ІЕІ-технології відрізняються від інших тим, що вони дозволяють перетворювати вхідний нечіткий розподіл реалізацій образів в чіткий саме під час оптимізації системи контрольних допусків. Тому, це дає змогу побудувати безпомилкові вирішальні правила та змінювати значення ознак розпізнавання [27].

У межах ІЕІ-технології вдається успішно поєднувати процес нормалізації образів та безпосередній етап навчання, що забезпечує створення вирішальних правил з високою точністю та ефективністю. Нормалізація образів дозволяє виправити їх апріорну деформацію відносно еталонного образу, що дозволяє відновити правильність їх подання та покращити результати подальшого навчання. Завдяки такому поєднанню методів можливо побудувати вирішальні правила, що працюють з високою точністю та ефективністю.

Допустимо, маємо $\{X_m^0 | m = \overline{1, M}\}$ – заданий алфавіт класів розпізнавання. При використанні гіпотези нечіткої компактності реалізацій образу, розбиття $\tilde{R}^{|M|}$ простору ознак на класи розпізнавання є нечітким і

відповідає умовам, що вказані нижче (2.1). Водночас, елементи розбиття $\tilde{R}^{|M|}$ також є нечіткими [27].

$$
\begin{aligned}
&1)\ \left(\forall X_m^o \in \widetilde{\Re}^{|M|}\right)[X_m^o \neq \emptyset];\\
&2)\ \left(\exists X_k^o \in \widetilde{\Re}^{|M|}\right)\left(\exists X_l^o \in \widetilde{\Re}^{|M|}\right)[X_k^o \neq X_l^o \to X_k^o \cap X_l^o \neq \emptyset];\\
&3)\ \left(\forall X_k^o \in \widetilde{\Re}^{|M|}\right)\left(\forall X_l^o \in \widetilde{\Re}^{|M|}\right)[X_k^o \neq X_l^o \to KerX_k^o \cap KerX_l^o = \emptyset];\\
&4)\ \bigcup_{X_m^o \in \widetilde{\Re}} X_m^o \subseteq \Omega_{\text{Б}};\ k \neq l; k,l,m = \overline{1,M}
\end{aligned}
\tag{2.1}
$$

У бінарному просторі ознак Ω гіперпаралелепіпед відображає форму оптимального контейнера класу, який розпізнається. Щоб зручніше побудувати такий контейнер, дозволено ввести таку фігуру, як «псевдогіперсфера», що містить всі вершини гіперпаралелепіпеда. Такі дії дозволяють надалі аналізувати параметри оптимізації контейнера, до яких відносять еталонний вектор та радіус псевдосферичного контейнера. Останній визначається у просторі Хемінга за формулою [27]:

$$d_m = \sum_{i=1}^{N}(x_{m,i} \oplus \lambda_i), \tag{2.2}$$

де $x_{m,i}-$ i – та координата еталонного вектора x_m; λ_i- i – та координата вектора λ, вершина якого відноситься до контейнера $K_m^o \in X_m^o$.

Надалі кодова відстань (2.2), для спрощення, буде позначатися як $d_m = d(x_m \oplus \lambda)$.

Використання IEI-технології дозволяє ефективно відновлювати оптимальний контейнер у радіальному базисі, приміром, K_m^o за допомогою послідовної цілеспрямованої трансформації в гіперсферичний габарит, де радіус d_m збільшується з кожним кором навчання відносно рекурентної процедури [27]:

$$d_m(k) = [d_m(k-1) + h \big| d_m(k) \in G_m^d], \quad (2.3)$$

де k – змінна числа збільшень радіуса контейнера K_m^o; h – крок збільшення радіуса; G_m^d– область допустимих значень радіуса d_m.

Припустимо, що між класами X_k^o та X_l^o існує мінімальна міжцентрова відстань $d(x_k \oplus x_l)$, тобто вони є "найближчими сусідами". Для запобігання "вбирання" ядра одного класу іншим, за IEI-технологією, умови (2.1) доповнюються наступним предикатним виразом:

$$\left(\forall X_k^o \in \widetilde{\Re}^{|M|}\right)\left(\forall X_l^o \in \widetilde{\Re}^{|M|}\right)[X_k^o \neq X_l^o \rightarrow \left(d_k^* < d(x_k \oplus x_l)\right) * \\ * (d_l^* < d(x_k \oplus x_l))], \quad (2.4)$$

де d_k^*, d_l^*– оптимальні радіуси контейнерів K_k^o і K_l^o відповідно.

Алгоритм навчання за IEI-технологією є ітеративним та багатоциклічним, він спрямований на оптимізацію структурованих просторово-часових параметрів функціонування інтелектуальних систем. Процедура навчання включає пошук глобального максимуму усередненого значення коефіцієнта функціональної ефективності за алфавітом $\{X_m^o\}$.

Умовою успішного використання IEI-технології для навчання інтелектуальних систем є обґрунтування гіпотези компактності. Це означає, що головна ідея цієї технології полягає у нормалізації вхідного математичного опису інтелектуальної системи шляхом трансформації апріорних габаритів розкиду образів. Ця трансформація здійснюється з метою максимального захоплення образів відповідних класів контейнерами, які будуються в радіальному базисі під час навчання. За допомогою IEI-технології, оптимальні контейнери забезпечують максимальну різноманітність між сусідніми класами. Ця різноманітність вимірюється максимальним значенням

інформаційного коефіцієнту ефективності навчання в робочій області, що дозволяє забезпечити ефективну функціональність ІС. Оптимальні геометричні параметри контейнерів, отримані в результаті навчання дозволяють ефективно приймати рішення в режимі екзамену за допомогою простого детермінованого вирішального правила, що є важливим фактором в реалізації алгоритмів прийняття рішень в режимі реального часу. За такої умови повна достовірність класифікатора наближається до максимальної асимптотичної, що окреслюється ефективністю процесу навчання. Для досягнення максимальної асимптотичної достовірності розпізнавання необхідно забезпечити однакові характеристики статистичної стійкості та статистичної однорідності навчальних та екзаменаційних матриць. Ця умова може бути виконана за умови навчання ІС у процесі функціонально-статистичних випробувань. За допомогою ІЕІ-технології ми можемо досягти цілеспрямованої оптимізації просторово-часових параметрів функціонування ІС шляхом аналізу та визначення тенденції зміни асимптотичних точнісних характеристик процесу навчання. Це дає можливість збільшувати точність класифікації та зменшувати час, потрібний для прийняття рішень, що є важливим для реалізації алгоритмів в реальному часі [27].

2.2 Інформаційний критерій оптимізації параметрів машинного навчання

Оцінка функціональної ефективності процесу навчання є ключовим фокусом при синтезі інформаційних систем. Вона визначає найвищу достовірність прийнятих рішень під час екзамену. В ІЕІ-технології можуть використовуватися різні критерії як КФЕ, які мають властивості інформаційних мір, такі як [27]:

- інформаційна міра є невід'ємною та монотонно зростаючою функцією від імовірності;
- чисельність інформації для $p_i = 1$ та $p_i = 0$ рівна нулю;

– при значенні ймовірності $p_i = \frac{1}{m}$ – інформаційна міра рівна екстремуму.

Оптимальним вибором інформаційних мір, необхідних для оцінки функціональної ефективності СППР, яка навчається, є статистичні логарифмічні критерії, здатні працювати з відносно малими навчальними вибірками. Такі критерії можуть включати ентропійні міри та інформаційну міру Кульбака, які широко використовуються для цієї мети [27].

Для прийняття рішень в двохальтернативній системі оцінок (M = 2) і рівноймовірних гіпотез можна скористатися ентропійним критерієм, який дозволяє оцінити найбільш важкий у статистичному сенсі випадок прийняття рішень. При цьому може бути виражений наступним чином:

$$E_m^{(k)} = 1 + \frac{1}{2}\left(\frac{\alpha_m^{(k)}(d)}{\alpha_m^{(k)}(d) + D_{2,m}^{(k)}(d)} log_2 \frac{\alpha_m^{(k)}(d)}{\alpha_m^{(k)}(d) + D_{2,m}^{(k)}(d)} + \right.$$
$$+ \frac{\beta_m^{(k)}(d)}{D_{1,m}^{(k)}(d) + \beta_m^{(k)}(d)} log_2 \frac{\beta_m^{(k)}(d)}{D_{1,m}^{(k)}(d) + \beta_m^{(k)}(d)} + \qquad (2.5)$$
$$+ \frac{D_{1,m}(d)}{D_{1,m}^{(k)}(d) + \beta_m^{(k)}(d)} log_2 \frac{D_{1,m}(d)}{D_{1,m}^{(k)}(d) + \beta_m^{(k)}(d)} +$$
$$\left. + \frac{D_{2,m}^{(k)}(d)}{\alpha_m^{(k)}(d) + D_{2,m}^{(k)}(d)} log_2 \frac{D_{2,m}^{(k)}(d)}{\alpha_m^{(k)}(d) + D_{2,m}^{(k)}(d)}\right),$$

де d – міра, що визначає радіуси гіперсферичних контейнерів, які побудовані в радіальному базисі простору Хемінга;

$\alpha_m^{(k)}(d)$ – помилка першого роду на k-му кроці навчання;

$\beta_m^{(k)}(d)$ – помилка другого роду;

$D_{1,m}^{(k)}(d)$ – перша достовірність;

$D_{2,m}^{(k)}(d)$ – друга достовірність.

У процесі застосування ІЕІ-технології для розпізнавання образів, критерій (2.5) є ключовим елементом. У зв'язку з тим, що точнісні характеристики – це функції відстані геометричних центрів контейнерів класів, що розпізнаються від вершин еталонних векторів, то цей критерій є нелінійним та з взаємно-неоднозначним функціоналом. Тому його визначення вимагає знаходження допустимої області під час навчання [27].

Розглянемо нову модифікацію міри Кульбака, що може бути представлена як відношення правдоподібності Λ помножене на міру відхилень відповідних розподілів імовірностей.

В роботі [28] досліджується логарифмічне відношення між повною ймовірністю $P_{t,m}^{(k)}$ правильного прийняття рішень щодо належності реалізацій класів X_m^o та X_c^o контейнеру $K_{m,k}^o \in X_m^o$ та повною ймовірністю помилкового прийняття рішень. Тому, для двохальтернативної системи оцінок рішень це відношення має вигляд [27]:

$$\Lambda = log_2 \frac{P_{t,m}^{(k)}}{P_{f,m}^{(k)}} = log_2 \frac{p(\mu_m)p(\gamma_{1,k}/\mu_m)+p(\mu_c)p(\gamma_{2,k}/\mu_c)}{p(\mu_m)p(\gamma_{2,k}/\mu_m)+p(\mu_c)p(\gamma_{1,k}/\mu_c)}, \quad (2.6)$$

де $p(\mu_m)$ – ймовірність появи реалізацій класу X_m^o;

$p(\mu_c)$ – ймовірність появи реалізацій сусіднього класу X_c^o;

$\gamma_{1,k}$ – припущення про належність контейнера $K_{m,k}^o$ реалізаціям класу X_m^o;

$\gamma_{2,k}$ – альтернативна гіпотеза.

Після врахування (2.6), якщо допустити згідно з принципом Лапласа-Бернуллі, що $p(\mu_m) = p(\mu_c) = 0{,}5$, та з переозначенням апріорних умовних ймовірностей відповідно до точних характеристик, загальна міра Кульбака може бути записана наступним чином:

$$E_{Km}^{(k)} = log_2 \frac{P_{t,m}^{(k)}}{P_{f,m}^{(k)}} * [P_{t,m}^{(k)} - P_{f,m}^{(k)}] =$$

$$= \begin{vmatrix} P_{t,m}^{(k)} = 0{,}5D_{1,m}^{(k)}(d) + 0{,}5D_{2,m}^{(k)}(d) \\ P_{f,m}^{(k)} = 0{,}5\alpha_m^{(k)}(d) + 0{,}5\beta_m^{(k)}(d) \end{vmatrix} = 0{,}5\, log_2 \left(\frac{D_{1,m}^{(k)}(d) + D_{2,m}^{(k)}(d)}{\alpha_m^{(k)}(d) + \beta_m^{(k)}(d)} \right) *$$

$$* \left[(D_{1,m}^{(k)}(d) + D_{2,m}^{(k)}(d)) - (\alpha_m^{(k)}(d) + \beta_m^{(k)}(d)) \right] = \quad (2.7)$$

$$= log_2 \left(\frac{2 - (\alpha_m^{(k)}(d) + \beta_m^{(k)}(d))}{\alpha_m^{(k)}(d) + \beta_m^{(k)}(d)} \right) * \left[1 - (\alpha_m^{(k)}(d) + \beta_m^{(k)}(d)) \right]$$

Нормований вигляд критерію (2.7) можна подати наступний чином:

$$E_{K,m}^{(k)} = \frac{E_{Km}^{(k)}}{E_{Kmax}^{(k)}}, \quad (2.8)$$

де $E_{Kmax}^{(k)}$ – значення критерію, коли $\alpha_m^{(k)}(d) = \beta_m^{(k)}(d) = 0$ та $D_{1,m}^{(k)}(d) = D_{2,m}^{(k)}(d) = 1$ для (2.7).

При застосуванні ІЕІ-технології для оптимізації параметрів функціонування у процесі навчання, нормування критеріїв оптимізації є не обов'язковим заходом. У такому випадку метою є знаходження таких значень параметрів навчання, які максимізують КФЕ в його визначеній робочій області. Однак нормування критеріїв оптимізації може бути корисним при оцінці ступеня близькості реальної інтелектуальної системи до потенційної та у випадку порівняння аналізів результатів досліджень [27].

2.3 Базовий алгоритм інформаційно-екстремального машинного навчання

Під час оптимізації просторово-часових параметрів функціонування СВК, базовий алгоритм інформаційно-екстремального машинного навчання реалізується в середині внутрішнього циклу алгоритму навчання. Базовий

алгоритм навчання визначає оптимальні значення параметрів навчання. Також під час реалізації вказаного алгоритму обчислюється сам інформаційний критерій та визначається його глобальний максимум у робочій області [27].

При застосуванні базового алгоритму навчання на вхід подається така інформація:

- масив дійсних реалізацій образу $\{y_m^{(j)} | m = \overline{1, M}; j = \overline{1, n}\}$;
- $\{\delta_{K,i}\}$ – система полів контрольних допусків;
- рівні селекції $\{\rho_m\}$ координат еталонних векторів, що рівні 0,5 для класів розпізнавання.

Проведемо аналіз етапів реалізації базового алгоритму [27]:

1. Генеруємо бінарну матрицю $||x_{m,i}^{(j)}||$ за правилом:

$$x_1^{(j)} = \begin{cases} 1, \text{ if } y_{1,i} - \delta \le y_{1,i}^{(j)} \le y_{1,i} + \delta, \\ 0, \text{ if else.} \end{cases} \tag{2.9}$$

2. Формуємо еталонні двійкові вектора $\{x_{m,i} | m = \overline{1, M}, i = \overline{1, N}\}$ за правилом:

$$x_{m,i} = \begin{cases} 1, & if \quad \frac{1}{n} \sum_{j=1}^{n} x_{m,i}^{(j)} > \rho_m; \\ 0, & ifelse, \end{cases} \tag{2.10}$$

де ρ_m – рівень селекції координат вектора $x_m \in X_m^o$.

3. Знаходимо пари «найближчих» сусідів $\Re_m^{|2|} = \langle x_m, x_l \rangle$ на основі еталонних векторів за наступним алгоритмом:

1. Структуруємо множину еталонних векторів, розпочинаючи з вектора x_1 , який належить базовому класу X_1^o та визначає найбільшу функціональну ефективність;
2. Будуємо матрицю кодових відстаней між еталонними векторами;

3. Знаходимо найменший елемент в кожному рядку матриці кодових відстаней, який належить стовпчику вектора, що найбільш близький до вектора, що визначає рядок. У випадку, коли є кілька однакових мінімальних елементів – слід обрати будь-який з них, оскільки вони мають однакове значення;
4. Створюємо структуровану множину елементів попарного розбиття $\{\Re_m^{|2|} | m = \overline{1, M}\}$, що задає план навчання.

4. Оптимізуємо кодову відстань d_m за рекурентною процедурою, враховуючи, що $E_m(0) = 0$:

$$d_m(k) = [d_m(k-1) + h | d_m(k) \in G_m^d] \tag{2.11}$$

5. Закінчуємо процедуру, у випадку коли знайдено максимум критерію в робочій області:

$$E_m^* = \max_{\{d\}} E_m, \tag{2.12}$$

де $\{d\} = \{d_1, \dots, d_k, \dots, d_{max}\} \epsilon [0; d(x_m \oplus x_l) - 1]$ є множиною радіусів гіперсфер, центр яких задається вершиною еталонного вектора $x_m \in X_m^o$.

Отже, базовий алгоритм навчання являє собою ітераційний процес, який має на меті знайти глобальний максимум інформаційного критерію в робочій області:

$$d_m^* = arg \max_{\{d\}} E_m^* \tag{2.13}$$

3 ІНФОРМАЦІЙНЕ, АЛГОРИТМІЧНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ АДАПТИВНОЇ СИСТЕМИ ВИЯВЛЕННЯ КІБЕРЗАГРОЗ

3.1 Вхідний математичний опис системи виявлення кіберзагроз

Для створення математичного опису інформаційно-екстремального навчання системи виявлення кіберзагроз було використано дані, що були зібрані з функціонуючих пристроїв та знаходяться у відкритому репозиторії даних «Machine Learning Repository» для машинного навчання [29]. Таким чином, використання реальних даних трафіку дозволяє створювати більш точні та ефективні моделі системи виявлення кіберзагроз.

Вхідний опис системи виявлення кіберзагроз ґрунтувався на 4 видах трафіку, кожен з яких мав 115 характеристик, які описані нижче:

Таблиця 3.1 – Характеристика трафіків

№¤	Характеристика·¤	№¤	Характеристика·¤	№¤	Характеристика·¤
1¤	MI_dir_L5_weight¤	40¤	HH_L3_std¤	78¤	HH_jit_L0.01_weight¤
2¤	MI_dir_L5_mean¤	41¤	HH_L3_magnitude¤	79¤	HH_jit_L0.01_mean¤
3¤	MI_dir_L5_variance¤	42¤	HH_L3_radius¤	80¤	HH_jit_L0.01_variance¤
4¤	MI_dir_L3_weight¤	43¤	HH_L3_covariance¤	81¤	HpHp_L5_weight¤
5¤	MI_dir_L3_mean¤	44¤	HH_L3_pcc¤	82¤	HpHp_L5_mean¤
6¤	MI_dir_L3_variance¤	45¤	HH_L1_weight¤	83¤	HpHp_L5_std¤
7¤	MI_dir_L1_weight¤	46¤	HH_L1_mean¤	84¤	HpHp_L5_magnitude¤
8¤	MI_dir_L1_mean¤	47¤	HH_L1_std¤	85¤	HpHp_L5_radius¤
9¤	MI_dir_L1_variance¤	48¤	HH_L1_magnitude¤	86¤	HpHp_L5_covariance¤
10¤	MI_dir_L0.1_weight¤	49¤	HH_L1_radius¤	87¤	HpHp_L5_pcc¤
11¤	MI_dir_L0.1_mean¤	50¤	HH_L1_covariance¤	88¤	HpHp_L3_weight¤
12¤	MI_dir_L0.1_variance¤	51¤	HH_L1_pcc¤	89¤	HpHp_L3_mean¤

Продовження таблиці 3.1

№¤	Характеристика·¤	№¤	Характеристика·¤	№¤	Характеристика·¤
13¤	MI_dir_L0.01_weight¤	52¤	HH_L0.1_weight¤	90¤	HpHp_L3_std¤
14¤	MI_dir_L0.01_mean¤	53¤	HH_L0.1_mean¤	91¤	HpHp_L3_magnitude¤
15¤	MI_dir_L0.01_variance¤	54¤	HH_L0.1_std¤	92¤	HpHp_L3_radius¤
16¤	H_L5_weight¤	55¤	HH_L0.1_magnitude¤	93¤	HpHp_L3_covariance¤
17¤	H_L5_mean¤	56¤	HH_L0.1_radius¤	94¤	HpHp_L3_pcc¤
18¤	H_L5_variance¤	57¤	HH_L0.1_covariance¤	95¤	HpHp_L1_weight¤
19¤	H_L3_weight¤	58¤	HH_L0.1_pcc¤	96¤	HpHp_L1_mean¤
20¤	H_L3_mean¤	59¤	HH_L0.01_weight¤	97¤	HpHp_L1_std¤
21¤	H_L3_variance¤	60¤	HH_L0.01_mean¤	98¤	HpHp_L1_magnitude¤
22¤	H_L1_weight¤	61¤	HH_L0.01_std¤	99¤	HpHp_L1_radius¤
23¤	H_L1_mean¤	62¤	HH_L0.01_magnitude¤	100¤	HpHp_L1_covariance¤
24¤	H_L1_variance¤	63¤	HH_L0.01_radius¤	101¤	HpHp_L1_pcc¤
25¤	H_L0.1_weight¤	64¤	HH_L0.01_covarian	102¤	HpHp_L0.1_weight¤
26¤	H_L0.1_mean¤	65¤	HH_L0.01_pcc¤	103¤	HpHp_L0.1_mean¤
27¤	H_L0.1_variance¤	66¤	HH_jit_L5_weight¤	104¤	HpHp_L0.1_std¤
28¤	H_L0.01_weight¤	67¤	HH_jit_L5_mean¤	105¤	HpHp_L0.1_magnitude¤
29¤	H_L0.01_mean¤	68¤	HH_jit_L5_variance¤	106¤	HpHp_L0.1_radius¤
30¤	H_L0.01_variance¤	69¤	HH_jit_L3_weight¤	107¤	HpHp_L0.1_covariance¤
31¤	HH_L5_weight¤	70¤	HH_jit_L3_mean¤	108¤	HpHp_L0.1_pcc¤
32¤	HH_L5_mean¤	71¤	HH_jit_L3_variance¤	109¤	HpHp_L0.01_weight¤
33¤	HH_L5_std¤	72¤	HH_jit_L1_weight¤	110¤	HpHp_L0.01_mean¤
35¤	HH_L5_magnitude¤	73¤	HH_jit_L1_mean¤	111¤	HpHp_L0.01_std¤

Продовження таблиці 3.1

№¤	Характеристика·¤	№¤	Характеристика·¤	№¤	Характеристика·¤
35¤	HH_L5_radius¤	74¤	HH_jit_L1_variance¤	112¤	HpHp_L0.01_magnitude¤
36¤	HH_L5_covariance¤	75¤	HH_jit_L0.1_weight¤	113¤	HpHp_L0.01_radius¤
37¤	HH_L5_pcc¤	76¤	HH_jit_L0.1_mean¤	114¤	HpHp_L0.01_covariance¤
38¤	HH_L3_weight¤	77¤	HH_jit_L0.1_variance¤	115¤	HpHp_L0.01_pcc¤
39¤	HH_L3_mean¤	¤	¤	¤	¤

Для дослідження, окрім ознак (характеристик), було використано 100 реалізацій. Таким чином вхідна навчальна матриця мала 4 стани мережі (один нормальний та 3 заражені), кожен з яких мав розмірність 100x115.

3.2 Категорійна функціональна модель інформаційно-екстремального машинного навчання системи виявлення кіберзагроз за ієрархічною структурою даних

Математична оптимізаційна модель структури класів розпізнавання представлена як орієнтований граф, що використовується в ієрархічному машинному навчанні. Категорійна модель базується на вхідному математичному описі, який представлено у вигляді структури [30]:

$$\Delta_B = \langle T, G, \Omega, Z, Y, X; f_1, f_2 \rangle, \tag{3.1}$$

де T – множина моментів часу, в які формуються вектори-реалізації класів розпізнавання;

G – фактори, які впливають на функціонування системи;

Ω – простір діагностичних ознак;

Z – простір станів технічної системи, що визначає алфавіт класів розпізнавання;

Y – вхідна навчальна матриця для заданих класів розпізнавання;

X – бінарна матриця;

f_1 – оператор, який відповідає за формування навчальної матриці;

f_2 – оператор, який відповідає за перетворення навчальної матриці в бінарну.

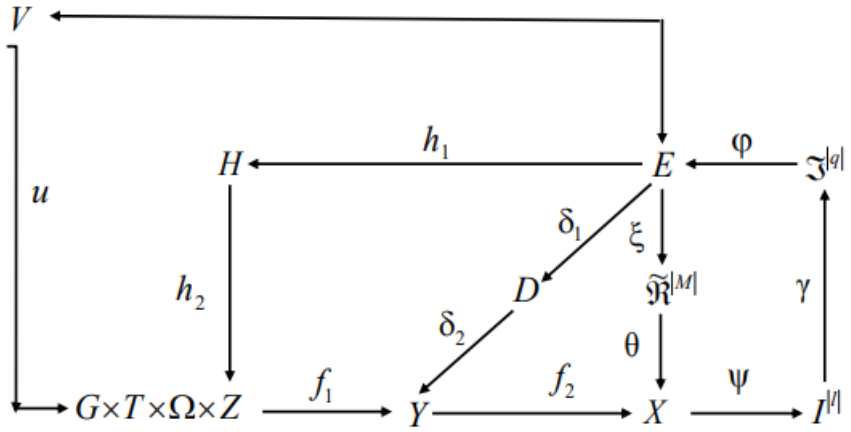


Рисунок 3.1 – Категорійна функціональна модель

Рис. 3.1 відображає категорійну функціональну модель інформаційно – екстремального машинного навчання системи виявлення кіберзагроз за ієрархічною структурою даних.

Відповідно цій моделі декартів добуток множин $G \times T \times \Omega \times Z$ створює універсум випробувань, який слугує джерелом інформації для кожного кроку машинного навчання. Оператор ξ відновлює контейнери класів розпізнавання в радіальному базисі простору ознак згідно з інформаційним критерієм E. Ці контейнери в загальному випадку утворюють нечітке розбиття $\tilde{\Re}^{|M|}$. Оператор θ використовується для перетворення побудованого розбиття $\tilde{\Re}^{|M|}$ на двійкові вектори – реалізації бінарної матриці X. В свою чергу оператор ψ використовується для перевірки основної статистичної гіпотези про належність реалізацій відповідному класу розпізнавання. Після проведення

статистичної перевірки гіпотез, утворюється множина статистичних гіпотез $I^{|l|}$. Оператор γ визначає множину точнісних характеристик $\Im^{|q|}$, де $q = l^2$. Оператор φ виконує розрахунок множини E значень інформаційного критерію для оптимізації параметрів машинного навчання. В категорійній моделі система контрольних допусків (терм – множина D) використовується для замикання контуру оптимізації контрольних допусків на ознаки розпізнавання. Ця множина дозволяє використовувати контрольні допуски як рівні квантування ознак розпізнавання при формуванні бінарної матриці. Використання бінарної навчальної матриці дозволяє здійснювати квантування за рівнем діагностичних ознак, що дозволяє адаптувати математичний опис вхідних даних до максимальної достовірності діагностичних рішень. Також, категорійна модель містить контур, який відповідає за оптимізацію ієрархічної структури даних, що реалізується через множину H. Ця множина включає варіанти розташування навчальних матриць відповідних класів, що розпізнаються в ієрархічній структурі. У випадку, якщо не досягнуто високу функціональну ефективність машинного навчання – за принципом відкладених рішень категорійна модель визначає можливість переходу до іншої, більш складної структури вирішальних правил. Задля цього категорійна модель передбачає множину V, що знаходиться в зовнішньому контурі оптимізації. Ця множина описує набір геометричних форм контейнерів класів розпізнання, що поновлюються під час машинного навчання в радіальному базисі простору діагностичних ознак. Оператор $u: V \to G \times T \times \Omega \times Z$ регламентує процес машинного навчання системи виявлення кіберзагроз [31].

3.3 Алгоритм інформаційно-екстремального машинного навчання системи виявлення кіберзагроз за ієрархічною структурою даних

Алгоритм інформаційно – екстремального машинного навчання системи виявлення кіберазагроз за ієрархічною структурою даних представлено як

ітераційну процедуру, що здійснює пошук глобального максимуму інформаційного критерію, усередненого за алфавітом $\{X_m^0\}$, в робочій області його визначення [30]:

$$h^* = \arg\max_{G_h}\left\{\max_{G_\delta}\left\{\max_{G_E}\bar{E}\right\}\right\}, \tag{3.2}$$

де G_h - область допустимих значень.

Для побудови ієрархічної структури слід сформувати варіаційний ряд. Розглянемо більш детальний алгоритм побудови варіаційного ряду для системи виявлення кіберзагроз:

1. Застосовуємо базовий алгоритм інформаційно-екстремального машинного навчання та знаходимо максимум інформаційного критерію (2.7) у відповідному δ з робочої області;
2. Обчислюємо бінарну матрицю (2.9) для кожного класу, використовуючи δ, знайдене на попередньому кроці;
3. Знаходимо еталонні двійкові вектори (2.10) для бінарних матриць;
4. Обчислюємо відстань кожного еталонного вектора від нульового вектора:

$$d(x_0 \oplus x_l) \tag{3.3}$$

5. Розміщуємо еталонні вектори кожного з класів на одній прямій, сортуючи їх в порядку зростання за значенням відстані, обрахованій на попередньому кроці. Протиставляємо кожному еталонному вектору відповідний клас розпізнавання, отримуючи варіаційний ряд.

Після побудови варіаційного ряду можемо будувати декурсивне бінарне дерево за алгоритмом:

1. Вектор впорядкованих класів розпізнавання розбиваємо на 2 групи, що будуть визначати відповідні 2 гілки декурсивного дерева;
2. За вершини першого ярусу обираються сусідні навчальні матриці, що знаходяться на межі кожної з груп;
3. Артибут з батьківської страти переноситься у страту з нащадками, тобто на нижчій ярус;
4. Кожна страта з нащадками містить також навчальну матрицю, що є найближчим сусідом в своїй групі до іншого атрибута страти;
5. Формування дерева продовжується доти, доки всі класи розпізнавання не будуть внесені до декурсивного дерева.

На рис. 3.2 зображено приклад ієрархічної структури даних у вигляді декурсивного дерева:

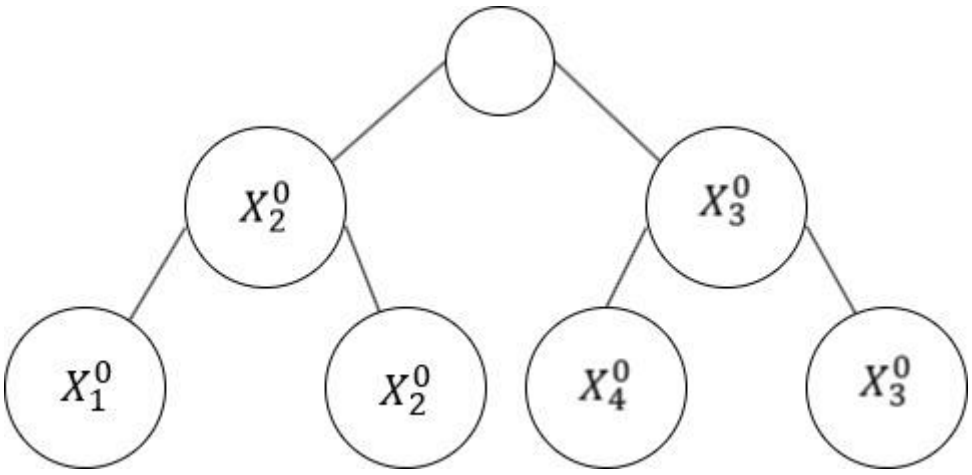


Рисунок 3.2 – Приклад декурсивного дерева

Розглянемо ключові кроки алгоритму інформаційно-екстремального машинного навчання системи виявлення кіберзагроз за ієрархічною структурою даних [30]:

1. Обнуляємо лічильник кроків навчання: $r := 0$;
2. Задаємо лічильник варіантів ієрархічних структур: $r := r + 1$;
3. Обнуляємо лічильник ярусів: $h := 0$;
4. Задаємо лічильник ярусів: $h := h + 1$;

5. Обнуляємо лічильник страт яруса: $s \coloneqq 0$;
6. Задаємо лічильник страт: $s \coloneqq s + 1$;
7. Для кожної s – ї страти h – го ярусу виконується інформаційно-екстремальний алгоритм навчання з паралельною оптимізацією контрольних допусків, що обчислює максимальне значення інформаційного критерію $\bar{E}^*_{r,h,s}$, усереднене по всім стратам ярусу;
8. У випадку, коли $s \leq S_h$, де S_h – кількість страт h – го ярусу, то виконається пункт під номером 6, а якщо ні – пункт 9;
9. У випадку, коли $h \leq h_{max}$, де h_{max} – кількість ярусів r – го кроку навчання, то виконається пункт під номером 4, а якщо ні – пункт 10;
10. Обчислюємо максимальне значення інформаційного критерію $\bar{E}^*_{r,h}$, усереднене по всім ярусам;
11. У випадку, коли $r \leq r_{max}$, де r_{max} – кількість кроків навчання, то виконається пункт під номером 2, а якщо ні – пункт 12;
12. Визначаємо оптимальну ієрархічну структуру: $h^*_r = arg \max_{\{r\}} \bar{E}^*_{r,h}$;
13. Зупинка алгоритму.

Отже, алгоритм інформаційно-екстремального машинного навчання системи виявлення кіберзагроз за ієрархічною структурою даних полягає в процедурі пошуку глобального максимуму інформаційного критерію для кожної страти ієрархічної структури та вибору оптимального з них.

3.4 Короткий опис програмного забезпечення

Для програмної реалізації інформаційно-екстремального машинного навчання системи виявлення кіберзагроз за ієрархічною структурою даних було використано інтерпретовану об’єктно-орієнтовану мову програмування Python. Python часто використовується для створення веб-сайтів і програмного забезпечення, автоматизації завдань і аналізу даних. Ця мовою є загального призначення, тобто її можна використовувати для створення різноманітних

програм і вона не спеціалізується на конкретних проблемах. Python став основним продуктом науки про дані, дозволяючи аналітикам даних та іншим професіоналам використовувати цю мову для проведення складних статистичних обчислень, створення візуалізацій даних, створення алгоритмів машинного навчання, обробки та аналізу даних, а також виконання інших завдань, пов'язаних із даними. Тобто, ця мова вирізняється швидкодією при обробці великих обсягів даних, тому і була обрана для написання алгоритму.

Для функціонування системи виявлення кіберзагроз було написано основні функції, які описані в табл. 3.2:

Таблиця 3.2 – Функції, які забезпечують функціонування СВК

Функція	Призначення
txt_to_array	Завантаження даних з файлів формату .txt
load_input_data	Формування вхідного математичного опису
mean_vector	Обчислення середнього значення кожного стовпця для базового класу
binary_matrix	Пошук бінарної матриці
etalon_vector	Обчислення еталонного вектору
distance_between_vectors	Пошук відстані між двома еталонними векторами
near_class	Пошук найближчого сусіда до заданого класу
code_distance	Обчислення кодової відстані
or_kulbaka	Обчислення критерію Кульбака
inf_criterion	Обчислення інформаційного критерію
precise_characteristics	Визначення точнісних характеристик
delta_calculation	Обчислення оптимального значення δ

Продовження таблиці 3.2

Функція	Призначення
radius_calculation	Обчислення оптимальних радіусів для кожного класу
variation_range	Побудова варіаційного ряду
algorithm_for_each_strata	Обчислення оптимального значення δ та радіусів для кожної страти
hierarchical_structure	Машинне навчання з використанням декурсивного дерева
main	Запус алгоритму

Перелік основних змінних, які використовуються під час функціонування СВК наведено в табл. 3.3.

Таблиця 3.3 – Основні змінні, які забезпечують функціонування СВК

Змінна	Характеристика
Y	Навчальна матриця
delta	Значення δ
bin_matrices	Бінарні матриці
etalon_vectors	Еталонні вектори
radius	Оптимальні радіуси
near_clas	Найближчий сусід до обраного класу
distance	Відстань між класом та його найближчим сусідом
code_dist	Кодова відстань
d1	Перша достовірність – ймовірність, що свої реалізації розпізнаються як свої

Продовження таблиці 3.3

Змінна	Характеристика
d2	Друга достовірність – ймовірність, що чужі реалізації розпізнаються як чужі
alpha	Помилка першого роду – ймовірність, що свої реалізації не розпізнаються як свої
beta	Помилка другого роду – ймовірність, що чужі реалізації розпізнаються як свої
inf_criterions	Значення інформаційного критерію
found_variation_range	Варіаційний ряд
optimal_delta	Оптимальне значення δ
max_inf_criterion	Значення інформаційного критерію при оптимальному значенні δ
tree_pairs	Страти декурсивного дерева
delta_in_work, radius_in_work	Дельта/радіус, які належать робочій області
delta_out_of_work, radius_out_of_work	Дельта/радіус, які лежать поза робочою областю
crit_in_work	Значення інформаційного критерію, яке лежить в межах робочої області
crit_out_of_work	Значення інформаційного критерію, яке лежить поза межами робочої області

Повний лістинг коду наведений в Додатку А.

3.5 Результати комп’ютерного моделювання

У процесі реалізації інформаційно-екстремального машинного навчання системи виявлення кіберзагроз за ієрархічною структурою даних було використано такі 4 класи: X_1^0 – нормальний трафік, X_2^0, X_3^0, X_4^0 – заражені трафіки.

Для побудови варіаційного ряду було проведено паралельну оптимізацію контрольний допусків, результат якої представлено на рис. 3.3. На цьому рисунку штриховкою позначено робочу область визначення функції критерію. Аналіз рисунку показує, що глобальний максимум критерію оптимізації досягається на ділянці типу плато. У цьому випадку оптимальний параметр δ визначався як мінімальний із зазначених. Тому, оптимальне δ становить 72 при значенні критерію $\bar{E} = 4{,}095$.

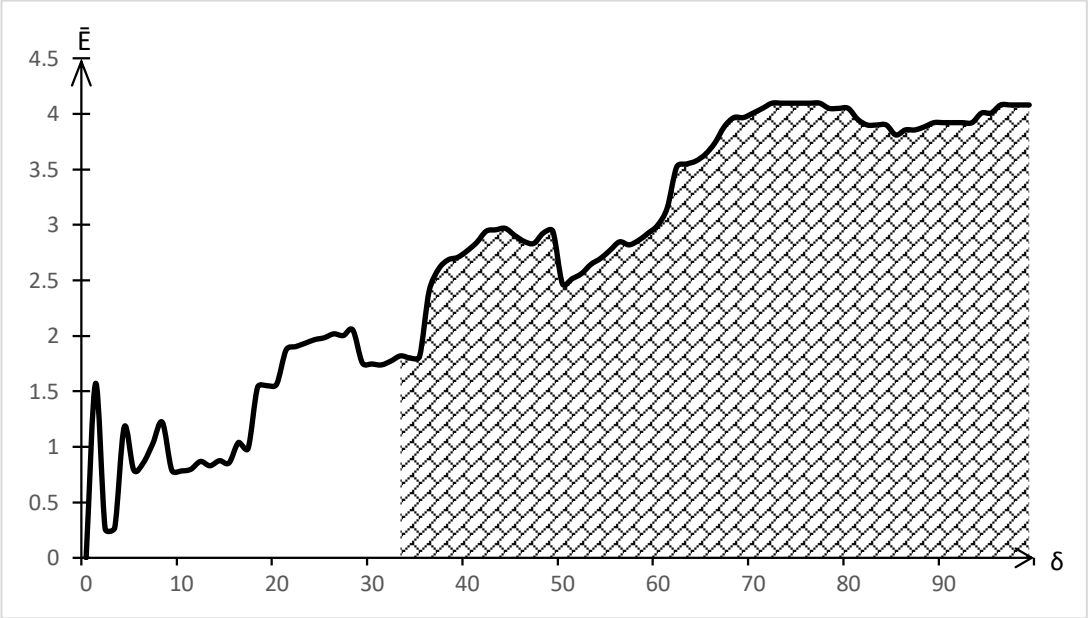


Рисунок 3.3 – Графік залежності інформаційного критерію від δ для 4-х класів

За обчисленим вище параметром δ було знайдено бінарні матриці та еталонні вектори кожного з класів. Це дало змогу побудувати варіаційний ряд, який представлений на рис. 3.4:

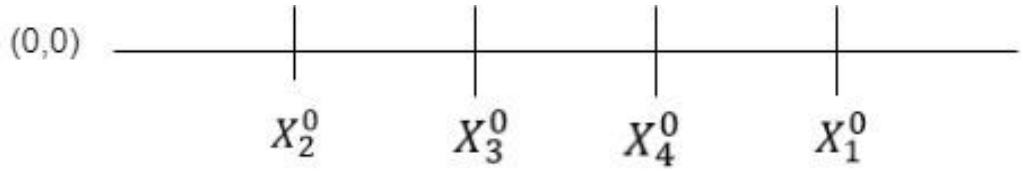


Рисунок 3.4 – Варіаційний ряд

Побудова варіаційного ряду дала змогу визначити та побудувати декурсивне дерево (рис. 3.5) та визначити 3 страти, для яких окремо обчислювалися оптимальні параметри.

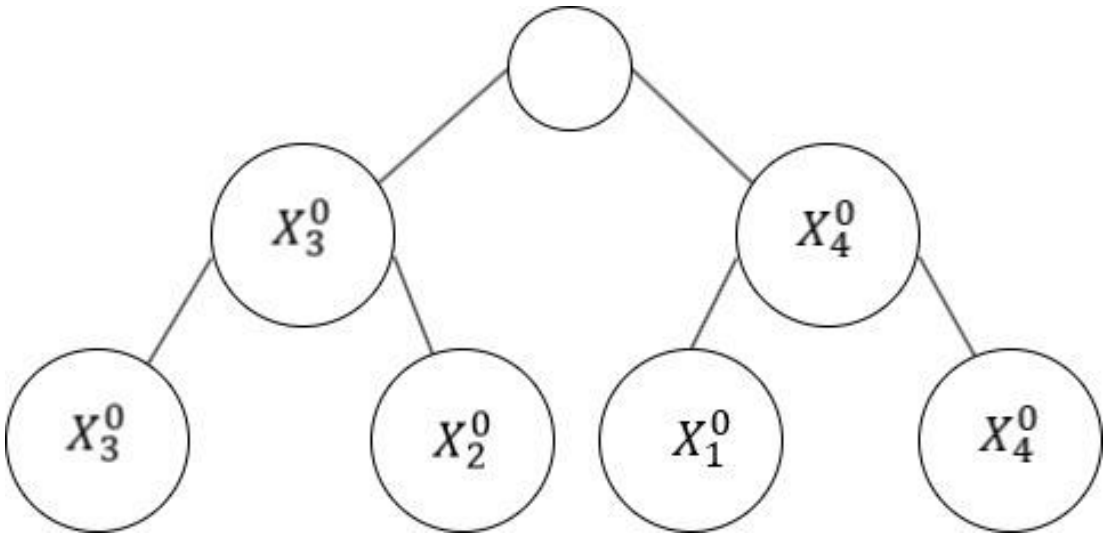


Рисунок 3.5 – Декурсивне дерево

На рис. 3.6 показано графік залежності усередненого інформаційного критерію для страти першого (верхнього за дендрографічною класифікацією) ярусу декурсивного дерева, тобто для класів X_3^0 та X_4^0. Для цієї страти значення критерію $\bar{E} = 4{,}392$ при $\delta = 6$.

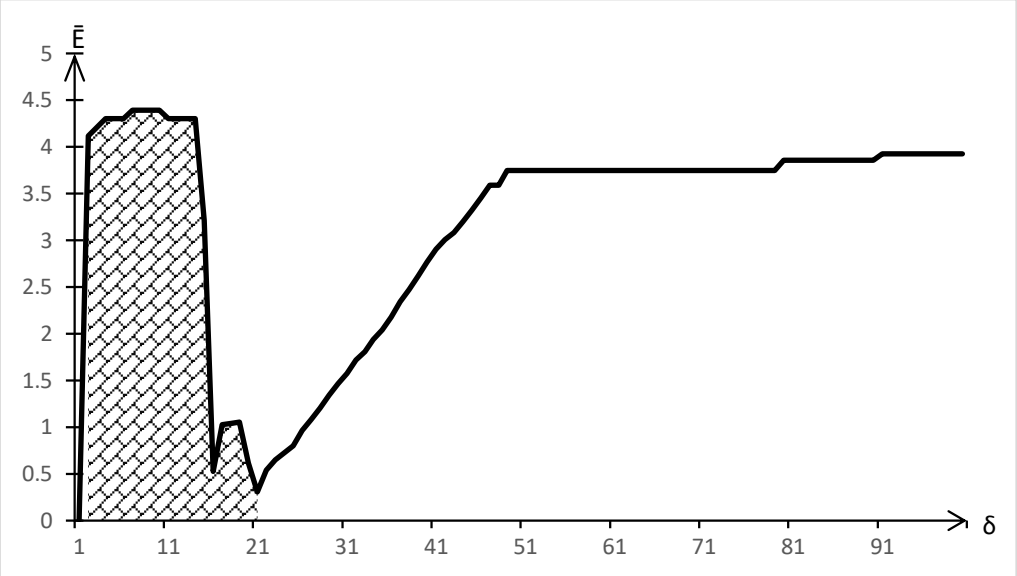
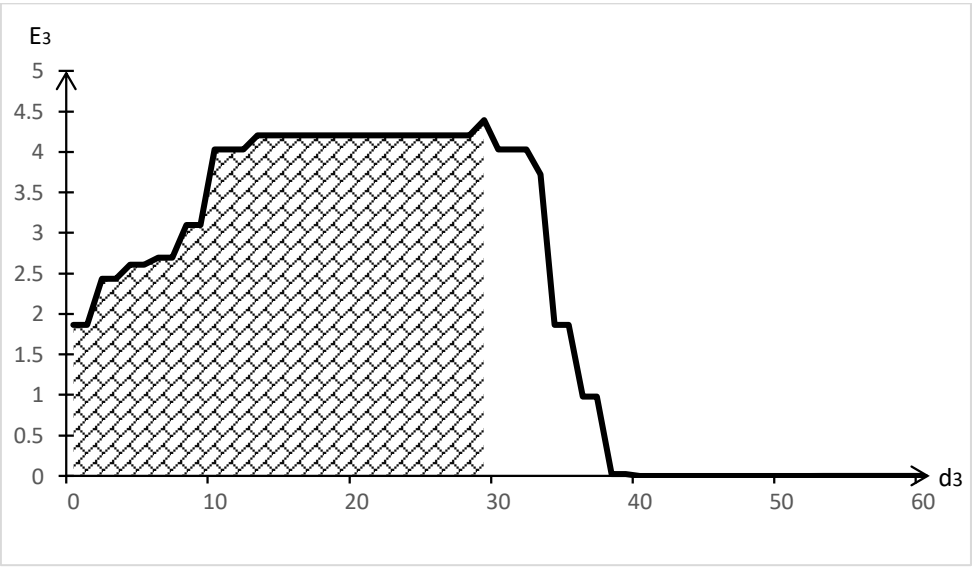
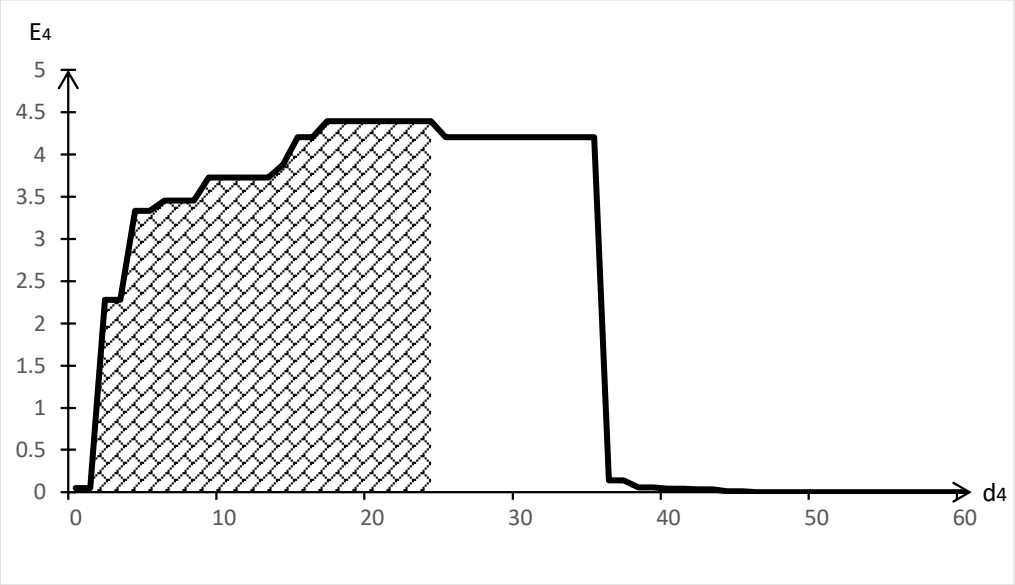


Рисунок 3.6 – Графік залежності інформаційного критерію від параметра поля контрольних допусків для страти першого ярусу декурсивного дерева

Оскільки вирішальні правила будуються в рамках геометричного підходу, то для них необхідно знання отриманих в процесі машинного навчання оптимальних геометричних параметрів контейнерів класів розпізнавання. На рисунку 3.7 показано графіки залежності інформаційного критерію від радіусів контейнерів класів розпізнавання для страти першого ярусу декурсивного дерева.



а



б

Рисунок 3.7 – Графік залежності інформаційного критерію від радіусів контейнерів класів розпізнавання для першого ярусу, де а – клас X_3^0, б – X_4^0

На рис. 3.8 показано графік залежності усередненого інформаційного критерію для першої страти другого ярусу декурсивного дерева, тобто для класів X_2^0 та X_3^0. Для цієї страти значення критерію $\bar{E} = 4{,}392$ при $\delta = 13$.

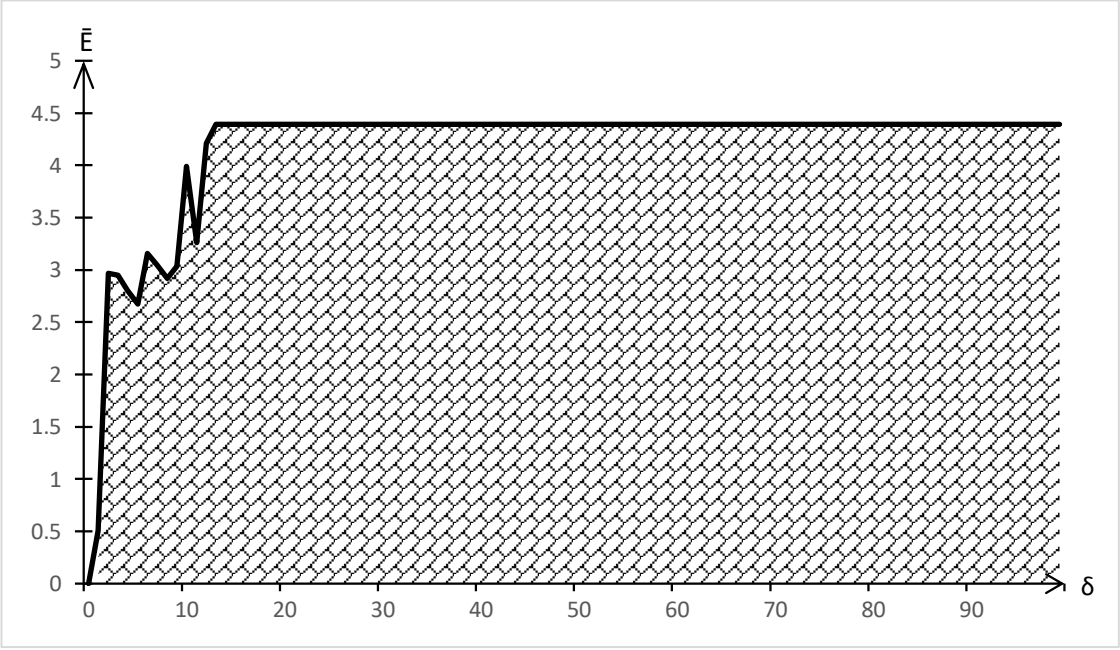
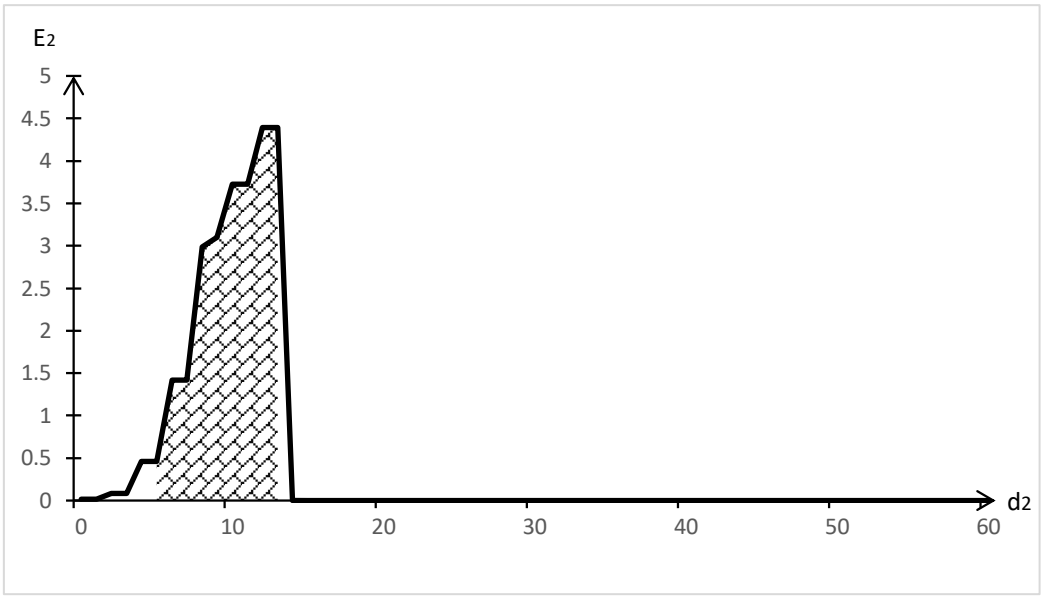
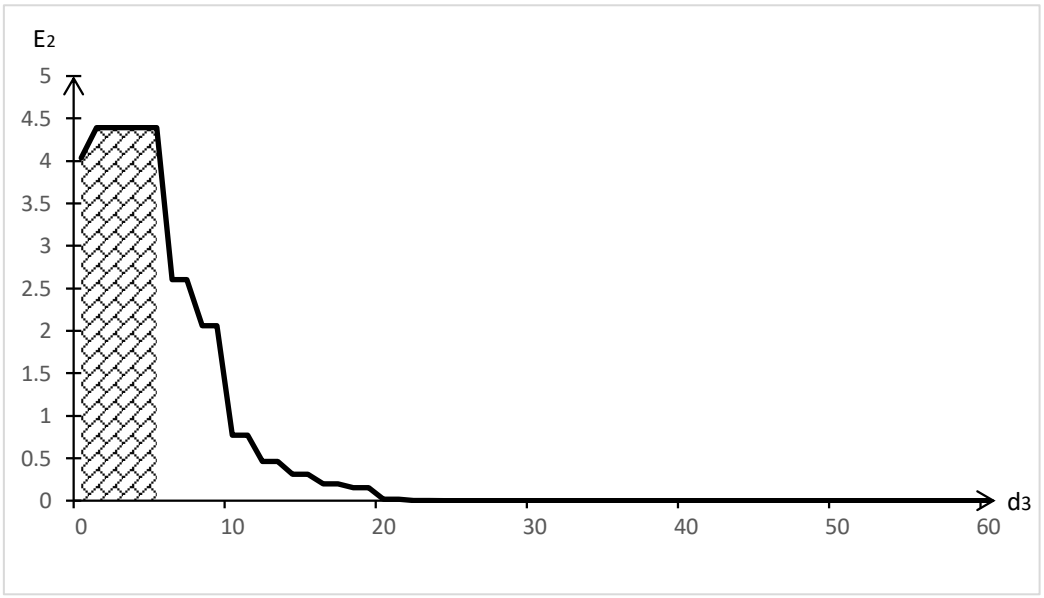


Рисунок 3.8 – Графік залежності інформаційного критерію від параметра поля контрольних допусків для першої страти другого ярусу

На рисунку 3.9 показано графіки залежності інформаційного критерію від радіусів контейнерів класів розпізнавання для першої страти другого ярусу декурсивного дерева.



а



б

Рисунок 3.9 – Графік залежності інформаційного критерію від радіусів контейнерів класів розпізнавання для першої страти другого ярусу, де а – клас X_2^0, б – X_3^0

На рис. 3.10 показано графік залежності усередненого інформаційного критерію для другої страти другого ярусу декурсивного дерева, тобто для класів X_1^0 та X_4^0. Для цієї страти значення критерію $\bar{E} = 4{,}132$ при $\delta = 7$.

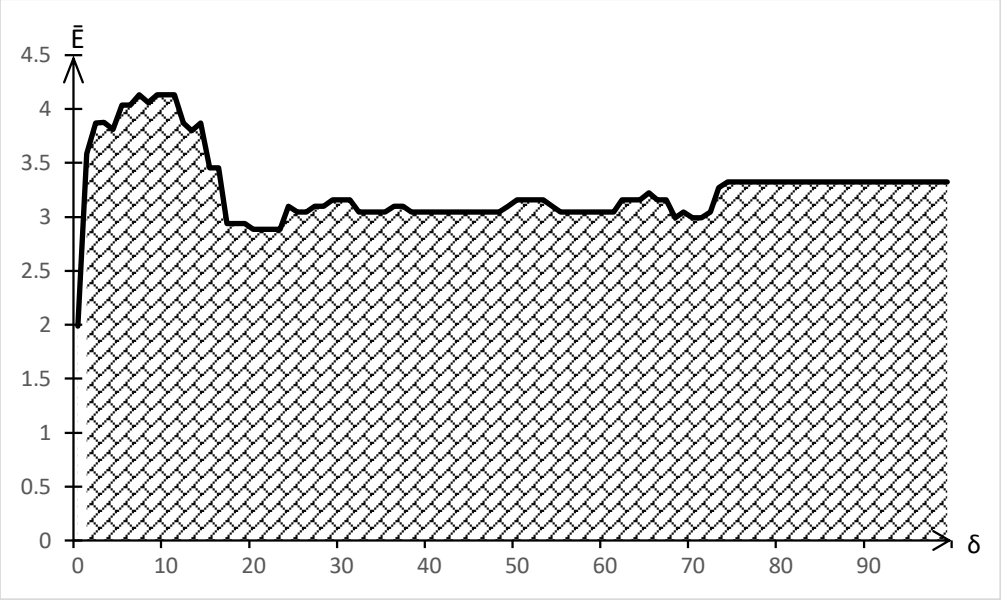
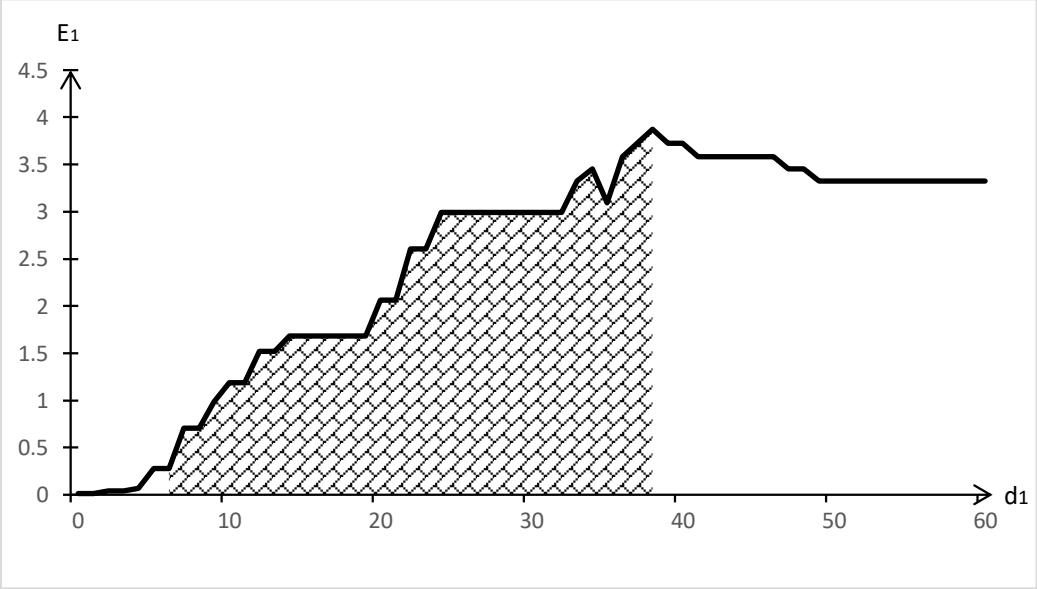
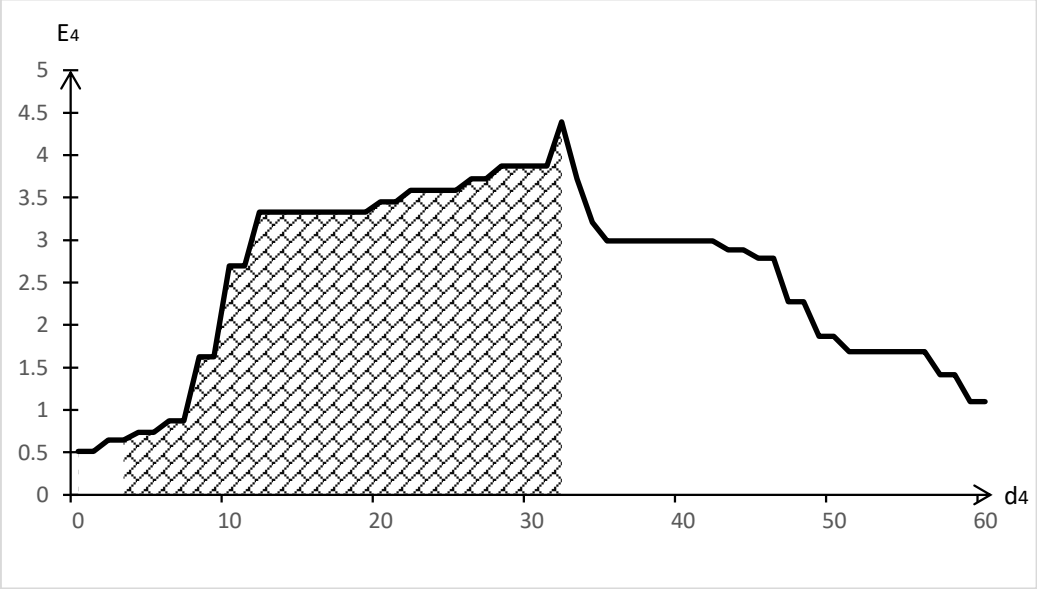


Рисунок 3.10 – Графік залежності інформаційного критерію від параметра поля контрольних допусків для другої страти другого ярусу

На рисунку 3.11 показано графіки залежності інформаційного критерію від радіусів контейнерів класів розпізнавання для другої страти другого ярусу декурсивного дерева.



а



б

Рисунок 3.11 – Графік залежності інформаційного критерію від радіусів контейнерів класів розпізнавання для першої страти другого ярусу, де а – клас X_1^0, б – X_4^0

У таблиці 3.4 представлено результати інформаційно-екстремального машинного навчання системи виявлення кіберзагроз за ієрархічною структурою даних.

Таблиця 3.4 – Результати

№ h.r	δ	$\bar{E}$	d_1	E_{d_1}	d_2	E_{d_2}
1.1	6	4.392	$d_{X_3^0} = 29$	$E_{d_{X_3^0}} = 4.392$	$d_{X_4^0} = 24$	$E_{d_{X_4^0}} = 4.392$
2.1	13	4.392	$d_{X_2^0} = 13$	$E_{d_{X_2^0}} = 4.392$	$d_{X_3^0} = 5$	$E_{d_{X_3^0}} = 4.392$
2.2	7	4.132	$d_{X_4^0} = 32$	$E_{d_{X_4^0}} = 4.392$	$d_{X_1^0} = 38$	$E_{d_{X_1^0}} = 3.873$

Порівнюючи значення радіусів контейнерів класів розпізнавання для страт першого та другого ярусів, за оптимальне слід приймати менше значення. Тобто, для класу X_3^0 за оптимальний радіус слід вважати $d_{X_3^0} = 5$, а для X_4^0 – $d_{X_4^0} = 24$.

Отже, в результаті вдалося побудувати безпомилкові вирішальні правила, адже кожна страта мала свої оптимальний параметр δ та глобальний максимум інформаційного критерії, що забезпечило високу достовірність розпізнавання в порівнянні з лінійною структурою.

ВИСНОВКИ

1. У кваліфікаційній роботі магістра розглянуто сучасний стан систем виявлення кіберзагроз. Проаналізовано класифікацію СВК, а саме розглянуто п’ять основних видів систем за способом моніторингу. Зроблено висновок, що найкращим варіантом СВК є гібридна система, яка буде поєднувати переваги чотирьох інших. Також, проведено детальний аналіз методів виявлення атак, де розглянуто переваги та недоліки сигнатурного методу та методу виявлення аномалій. До того ж, проведено детальний аналітичний огляд методів машинного навчання, де було доведено, що інформаційно – екстремальне машинне навчання є перспективною альтернативою штучним нейронним мережам.

2. Під час реалізації системи виявлення кіберзагроз застосовано інформаційно-екстремальну інтелектуальну технологію машинного навчання за ієрархічною структурою даних. У процесі інформаційного синтезу описано категорійну модель, яка виступає загальною архітектурою навчання за ієрархічною структурою. Після цього проведено розробку методу побудови варіаційного ряду, на основі якого будується декурсивне дерево. Вхідний математичний опис складався з характеристики реальних трафіків, які були взяті з відкритого репозиторію даних.

3. Програмна реалізація проведена за допомогою інтерпретованої об'єктно-орієнтованої мови програмування Python. Отримані вирішальні правила показали, що розроблена система виявлення кіберзагроз підтверджує високу достовірність розпізнавання. Тому, ці правила можуть бути використані для тестування СВК у режимі екзамену, з можливістю перенавчання.

СПИСОК ЛІТЕРАТУРИ

1. Cardenas A. Challenges for Securing Cyber Physical Systems / A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, S. Sastry // In Proceedings of the Workshop on Future Directions in Cyber-Physical Systems Security, Newark, NJ, USA, 22–24 July 2009.
2. Ding D. A Survey on Security Control and Attack Detection for Industrial Cyber-Physical Systems / D. Ding, Q.L. Han, Y. Xiang, X. Ge, X.M. Zhang // Neurocomputing, 2018. – Vol. 275. – p. 1674–1683.
3. Javaid A. A Deep Learning Approach for Network Intrusion Detection System / A. Javaid, Y. Ahmad, N. Quamar, S. Weiqing, A. Mansoor // EAI Endorsed Trans. Security Safety, 2016. – Vol. 3. – Nr. 9. – S. e2.
4. Павлов І.М. Аналіз таксономії систем виявлення атак у контексті сучасного рівня розвитку інформаційних систем / І.М. Павлов, С.В. Толюпа, В.І. Ніщенко // Сучасний захист інформації. – 2014. – Вип. 4. – С.44 – 52.
5. Intrusion Detection System (IDS). [Electronic resource]. – Regime of access: https://www.geeksforgeeks.org/intrusion-detection-system-ids/
6. Толюпа С. Модель системи протидії вторгненням в інформаційних системах / С. Толюпа, І. Пархоменко, С. Штаненко // Інфокомунікаційні технології та електронна інженерія. – 2021. – Вип. № 1 (1). – С. 39–50.
7. Теницька А.О. Система виявлення кібератак. Інформаційна технологія функціонування системи виявлення атак в режимі моніторингу [Текст]: робота на здобуття кваліфікаційного ступеня бакалавра; спец.: 125 – кібербезпека / А.О. Теницька; наук. кер. А.С. Довбиш // Суми: СумДУ, 2021. – 65 с.
8. Cannady J. Artificial Neural Networks for Misuse Detection / J. Cannady // Proceedings of the ’98 National Information System Security Conference (NISSC’98), 1998. – pp. 443-456.

9. Shon T. SVM Approach with A Genetic Algorithm for Network Intrusion Detection / T. Shon, J. Seo, J. Moon // Lecture Notes in Computer Science, Springer Berlin, Heidelberg, 2005. – Vol. 3733. – pp. 224-233.
10. Yu Y. An Ensemble Approach to Intrusion Detection Based on Improved Multi-Objective Genetic Algorithm / Y. Yu, H. Huang // Journal of Software, 2007. – Vol.18. – No.6. – pp.1369-1378.
11. Luo J. Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection / J. Luo, S. M. Bridges // International Journal of Intelligent Systems, 2000. – Vol. 15. – No. 8. – pp. 687-704.
12. Honig, A. Adaptive Model Generation: An Architecture for the Deployment of Data Mining-Based Intrusion Detection Systems / A. Honig, A. Howard, E. Eskin, S. J. Stolfo // Applications of Data Mining in Computer Security, Kluwer Academic Publishers, Boston, MA, 2002. – pp. 154-191.
13. Jamshir M. FUZZY LOGIC FOR INTRUSION DETECTION SYSTEM / M. Jamshir, M. Ajeesha // International Journal of Emerging Technologies and Innovative Research, December, 2020. – Vol.7. – Issue 12. – pp.877-882.
14. Kesavulu Reddy E. Neural Networks for Intrusion Detection and Its Applications / E. Kesavulu Reddy // Proceedings of the World Congress on Engineering, London, U.K, 2013. – Vol. II. – pp.1210-1214.
15. Suhaimi H. Network intrusion detection system by using genetic algorithm / H. Suhaimi, S. Suliman, M. Ismail, A. Harun, R. Mohamad // Indonesian Journal of Electrical Engineering and Computer Science, 2016. – Vol.16. – pp. 1593-1599.
16. Jayshree J. Intrusion Detection System using Support Vector Machine / J. Jayshree, R. Leena // IJAIS Proceedings on International Conference and workshop on Advanced Computing, 2013. – ICWAC (3).25-30.

17. Довбиш А.С. Оптимізація параметрів навчання системи підтримки прийняття рішень для керування виробництвом композитних матеріалів / А.С. Довбиш, В.О. Боровик, Н.І. Андрієнко // Вісник СумДУ. Серія “Технічні науки”. – 2012. – №3. – С. 10-15.
18. Проценко С.І. Інформаційна технологія розпізнавання електронограм на просвічуючому електронному мікроскопі [Текст]: робота на здобуття кваліфікаційного ступеня магістра; спец.: 122 - комп`ютерні науки (інформатика) / С.І. Проценко; наук. керівник А.С. Довбиш // Суми: СумДУ, 2021. – 76 с.
19. Руденко М.С. Розпізнавання онкологічних захворювань [Текст] / М.С. Руденко, С.С. Мартиненко // Інформатика, математика, механіка (ІММ-2009) : матеріали та програма IV міжвузівської науково-технічної конференції викладачів, співробітників, аспірантів і студентів, Суми : СумДУ, 21-24 квітня 2009. – С. 45.
20. Довбиш А. С. Система підтримки прийняття рішень для визначення схеми лікування гострої кишкової інфекції [Текст] / А.С. Довбиш, Г.А. Стадник, К.С. Полов’ян // Вісник Сумського державного університету. Серія Технічні науки. – 2012. – №1. – С. 25-31.
21. Довбиш А.С. Підхід до класифікаційного прогнозування валютного курсу [Текст] / А.С. Довбиш, А.В. Чала // Інтелектуальні системи в промисловості і освіті : тези доповідей Третьої міжнародної науково-практичної конференції, м. Суми, 2-4 листопада 2011. — Т.2. — С. 26 - 30.
22. Довбиш А.С. Оптимізація інвестиційного портфеля за методом функціонально-статистичних випробувань [Текст] / А.С. Довбиш, В.А. Тронь // Вісник Сумського державного університету. Серія Технічні науки. — 2006. — №4(88). — С. 43-53.
23. Міщенко А. Є. Інформаційна технологія адаптації навчального контенту випускової кафедри до вимог ринку праці : робота на

здобуття кваліфікаційного ступеня магістра : спец. 122 – комп'ютерні науки / наук. кер. А. С. Довбиш. Суми : Сумський державний університет, 2022. 67 с.

24. Бортова система безпілотного літального апарату для автономного розпізнавання наземних малогабаритних об'єктів [Текст] : звіт про НДР (проміжний) / кер. А. С. Довбиш. — Суми : СумДУ, 2020. — 96 с.
25. Савченко Т.Р. Інформаційна технологія розпізнавання об'єктів. Машинне навчання бортової системи розпізнавання наземних об'єктів [Текст]: робота на здобуття кваліфікаційного ступеня бакалавра; спец.: 122 - комп`ютерні науки (інформатика) / Т.Р. Савченко; наук. кер. А.С. Довбиш, Суми: СумДУ, 2021. – 67 с.
26. Dovbysh A. Information-extreme machine learning of a cyber attack detection system / A. Dovbysh, V. Liubchak, I. Shelehov, J. Simonovskiy, A. Tenytska // Radioelectronic and Computer Systems, 2022. – vol. 3. – pp. 121-131.
27. Довбиш А.С. Основи проектування інтелектуальних систем : навч. посіб. / А.С. Довбиш // Суми. Сумський Державний Університет. – Суми : СумДУ, 2009. – 170 с.
28. Удод В. О. Лекції з теорії ймовірностей та математичної статистики / В. О. Удод // Суми: Сумська обласна друкарня, 1999.– 188 с.
29. Machine Learning Repository. [Electronic resource]. – Regime of access:https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT.
30. Dovbysh A. Optimization of hierarchical data structure of intelligent system of functional diagnosis of technical condition of complex machines / A. Dovbysh, V. Zimovets, M. Bibyk // Bulletin of National Technical University KhPI Series System Analysis Control and Information Technologies, 2018. – 1320. – pp. 42-49. 10.20998/2079-0023.2018.44.08.

31. Інтелектуальний протез кінцівки, що самонавчається [Текст] : звіт про НДР (остаточний) / кер. А. С. Довбиш. — Суми : СумДУ, 2019. — 101 с.

ДОДАТОК А

Повний лістинг коду:

```
# ----------------------------------------------------------
# Інформаційно-екстремальне машинне навчання системи виявлення кіберзагроз за ієрархічною структурою даних
#
# (C) 2023 Теницька Альона, Суми, Україна
# email tenickajaalena@gmail.com
# ----------------------------------------------------------

import numpy as np
from math import log


def txt_to_array(clas):
    """Завантаження даних з файлів формату .txt """
    with open(clas, 'r') as f:
        values = [[float(num) for num in line.split()] for line in f]
    return values


def load_input_data(input_data):
    """Формування вхідного математичного опису

    Ключові аргументи:
    input_data -- масив, який містить назви файлів вхідних даних
    Y -- навчальна матриця
    """
    Y = [txt_to_array(clas) for clas in input_data]
```

```
        return Y


    def mean_vector(array):
        """Обчислення середнього значення для кожного стовпця базового класу

        Ключові аргументи:
        array -- базовий клас
        mean_vector -- вектор, що містить середнє значення по стовпчиках для базового класу
        """
        mean_vector = [sum(i) / len(i) for i in zip(*array)]
        return mean_vector


    def binary_matrix(classes, mean_vector, delta):
        """Пошук бінарної матриці

        Ключові аргументи:
        classes -- клас навчальної матриці
        mean_vector -- вектор, що містить середнє значення по стовпчиках для базового класу
        delta -- значення дельти
        vd, nd -- верхній та нижній допуски
        binary_matrix -- бінарна матриця
        """
        binary_matrix, vd, nd = [], [], []
```

```
    for i in range(len(mean_vector)):
        vd.append(mean_vector[i] + delta)
        nd.append(mean_vector[i] - delta)

    for i in range(len(classes)):
        binary_matrix.append([])
        for j in range(len(classes[0])):
            if nd[j] <= classes[i][j] <= vd[j]:
                binary_matrix[i].append(1)
            else:
                binary_matrix[i].append(0)

    return binary_matrix



def etalon_vector(bin_matrix):
    """Пошук еталонного вектору

    Ключові аргументи:
    bin_matrix -- бінарна матриця класу
    ro -- рівень селекції
    et_vector -- еталонний вектор відповідної бінарної матриці класу
    """
    ro = 0.5
    m_vector = mean_vector(bin_matrix)
    et_vector = []

    for i in range(len(m_vector)):
        if ro <= round(m_vector[i]):
```

```
                et_vector.append(1)
            else:
                et_vector.append(0)

        return et_vector


    def distance_between_vectors(first_vector, second_vector):
        """Пошук відстані між еталонними векторами

        Ключові аргументи:
        first_vector, second_vector -- еталонні вектори класів, між якими шукається відстань
        ds -- значення відстані між векторами
        """
        ds = 0
        for i in range(len(first_vector)):
            if first_vector[i] != second_vector[i]:
                ds += 1
        return ds


    def near_class(etalon_vectors):
        """Пошук найближчого сусіда

        Ключові аргументи:
        etalon_vectors -- еталонні вектори навчальної матриці
        near_class -- номер сусіднього класу
        distances -- відстань між класом і найближчим сусідом
```

```
        """
        near_class = []
        distances = []

        for i in range(len(etalon_vectors)):
            distances.append(len(etalon_vectors[0]) + 1)
            near_class.append(len(etalon_vectors[0]) + 1)

        for i in range(len(etalon_vectors)):
            for j in range(len(etalon_vectors)):
                if (i != j):
                    dist = distance_between_vectors(etalon_vectors[i],
etalon_vectors[j])
                    if (dist < distances[i]):
                        distances[i] = dist
                        near_class[i] = j

        return near_class, distances


    def code_distance(etalon_vector, binary_matrix):
        """Пошук кодової відстані між

        Ключові аргументи:
        binary_matrix -- бінарна матриця класу
        etalon_vector -- еталонний вектор відповідної бінарної матриці класу
        code_dist -- кількість кодових відстаней від геометричних центрів
контейнерів класів до їх реалізації
        """
```

```
        code_dist = []

        for i in binary_matrix:
            code_dist.append(distance_between_vectors(etalon_vector, i))
        code_dist = np.transpose(code_dist)

        return code_dist


    def or_kulbaka(alpha, beta):
        """Обчислення Критерію Кульбака

        Ключові аргументи:
        alpha -- помилка першого роду
        beta -- помилка другого роду
        kulbak -- значення критерію Кульбака
        """
        kulbak = (log((2 - (alpha + beta) + 0.1) / (alpha + beta + 0.1)) / log(2)) * (1
- (alpha + beta))
        return kulbak


    def inf_criterion(alpha, beta):
        """Обчислення інформаційного критерію (для даного випадку -
критерію Кульбака)

        Ключові аргументи:
        alpha -- помилка першого роду
        beta -- помилка другого роду
```

```
        inf_crit -- значення інформаційного критерію"""
        inf_crit = or_kulbaka(alpha, beta)
        return inf_crit


    def precise_characteristics(etalon_vectors, binary_matrices, near_class, distance):
        """Пошук точнісних характеристик

        Ключові аргументи:
        binary_matrices -- бінарні матриці навчальної матриці
        etalon_vectors -- еталонні вектори навчальної матриці
        near_class -- найближчий сусід кожного класу
        distance -- відстань між класом та його найближчим сусідом
        d1 -- перша достовірність
        d2 -- друга достовірність
        alpha -- помилка першого роду
        beta -- помилка другого роду
        inf_criterions -- значення інформаційного критерію для кожного класу
        """
        inf_criterions = []

        for clas in range(len(etalon_vectors)):
            inf_criterions.append([])
            for radius in range(61):
                d1 = list(map(lambda i: 1 if i <= radius else 0, code_distance(etalon_vectors[clas], binary_matrices[clas])))
                d1 = sum(d1) / len(d1)
                alpha = 1 - d1
```

```
                beta = list(map(lambda i: 1 if i <= radius else 0, code_distance(etalon_vectors[clas], binary_matrices[near_class[clas]])))
                beta = sum(beta) / len(beta)
                d2 = 1 - beta

                criterion_value = inf_criterion(alpha, beta)

                work_area = (d1 > 0.5) and (d2 > 0.5) and (beta < 0.5) and (radius < distance[clas])
                inf_criterions[clas].append([criterion_value, work_area])

        return inf_criterions


    def delta_calculation(Y):
        """Пошук оптимальної дельти

        Ключові аргументи:
        Y -- навчальна матриця
        optimal_delta -- значення оптимальної дельти
        max_inf_criterion -- значення інформаційного критерію для оптимальної дельти

        """

        optimal_delta, max_inf_criterion = 0, 0
        delta_in_work, crit_in_work, delta_out_of_work, crit_out_of_work = [], [], [], []
```

```
print(f'-----------------------------------------------------')
print(f'Значення дельта  Значення критерію     В робочій області')

for delta in range(100):

    bin_matrices, etalon_vectors = [], []
    mean_vect_for_base_clas = mean_vector(Y[0])

    for i in range(len(Y)):
        one_class_bin_matrix = binary_matrix(Y[i], mean_vect_for_base_clas, delta)
        bin_matrices.append(one_class_bin_matrix)
        one_class_etalon_v = etalon_vector(one_class_bin_matrix)
        etalon_vectors.append(one_class_etalon_v)

    near_clas, distance = near_class(etalon_vectors)

    inf_criterions = [precise_characteristics(etalon_vectors, bin_matrices, near_clas, distance) for _ in range(len(etalon_vectors))]

    sum_crit_in_work_area, sum_crit = [], []

    for i in range(len(inf_criterions)):
        for j in range(len(inf_criterions[i])):
            a = -10
            b = -10
            for value, key in inf_criterions[i][j]:
                if key and b < value:
```

```
                    b = value
                if a < value:
                    a = value
            sum_crit_in_work_area.append(b)
            sum_crit.append(a)

        current_value = sum(sum_crit_in_work_area) / len(sum_crit_in_work_area)

        if (current_value > max_inf_criterion):
            optimal_delta, max_inf_criterion = delta, current_value

        if current_value > 0:
            delta_in_work.append(delta)
            crit_in_work.append(sum(sum_crit) / len(sum_crit))
            print(f'  {delta_in_work[-1]}  {crit_in_work[-1]}  True')
        else:
            delta_out_of_work.append(delta)
            crit_out_of_work.append(sum(sum_crit) / len(sum_crit))
            print(f'  {delta_out_of_work[-1]}  {crit_out_of_work[-1]}  False')

    print(f"Оптимальна дельта: {optimal_delta}")
    print(f"Значення критерію для даної дельти: {max_inf_criterion}")

    return optimal_delta


def radius_calculation(etalon_vectors, bin_matrices):
```

```
        """Пошук оптимального радіуса

        Ключові аргументи:
        binary_matrices -- бінарні матриці навчальної матриці
        etalon_vectors -- еталонні вектори навчальної матриці
        radius -- оптимальний радіус для класу
        """
        near_clas, distance = near_class(etalon_vectors)

        inf_criterions = []
        inf_criterions.append(precise_characteristics(etalon_vectors,
bin_matrices, near_clas, distance))
        radius = []
        print(f'-----------------------------------------------------')
        print(f'Пошук радіусів:')
        print(f'Радіус  Значення критерію     В робочій області')
        for i in range(len(inf_criterions[0])):
            res = inf_criterions[0][i]
            index = -1
            value = -1
            radius_in_work, crit_in_work, radius_out_of_work, crit_out_of_work =
[], [], [], []

            for j in range(len(res)):
                if (res[j][1] and res[j][0] >= value):
                    value = res[j][0]
                    index = j
                    radius_in_work.append(value)
                    crit_in_work.append(index)
```

```
                print(f'{crit_in_work[-1]}     {radius_in_work[-1]}     True')
            else:
                radius_out_of_work.append(res[j][0])
                crit_out_of_work.append(j)
                print(f'   {crit_out_of_work[-1]}          {radius_out_of_work[-   1]}
False')
        print(f'-----------------------------------------------------')
        radius.append(index)
    return radius

def take_second(elem):
    """Допоміжна функція для сортування варіаційного ряду """
    return elem[1]

def variation_range(etalon_vectors, Y):
    """Пошук варіаційного ряду

    Ключові аргументи:
    Y -- навчальна матриця
    etalon_vectors -- еталонні вектори навчальної матриці
    dist_between_zero_vector -- відстань від нульового вектора
    vector -- варіаційний ряд
    sorted_class_values -- сортована навчальна матриця відносно
варіаційного ряду
    """
    dist_between_zero_vector = []

    for i in range(len(etalon_vectors)):
        [dist_between_zero_vector.append([i for _ in range(2)])]
```

```
        for i in range(len(etalon_vectors)):
            dist_between_zero_vector[i][1] = etalon_vectors[i].count(1);

        sorted_list = sorted(dist_between_zero_vector, key=take_second)
        vector = []
        for i in range(len(sorted_list)):
            vector.append(sorted_list[i][0])

        print("Варіаційний ряд:", vector)

        sorted_class_values = [Y[i] for i in vector]

        return [vector, sorted_class_values]


    def algorithm_for_each_strata(Y, clas1, clas2):
        """Функція пошуку дельта та радіусу для кожної страти декурсивного
дерева

        Ключові аргументи:
        Y -- навчальна матриця
        delta -- оптимальна дельта
        radius -- оптимальні радіуси
        """
        print(f"Пошук дельта для класів {clas1} та {clas2}:")
        delta = delta_calculation(Y)

        bin_matrices, etalon_vectors = [], []
```

```
        mean_vect_for_base_clas = mean_vector(Y[0])

        for one_class in Y:
            one_class_bin_matrix = binary_matrix(one_class,
mean_vect_for_base_clas, delta)
            bin_matrices.append(one_class_bin_matrix)
            one_class_etalon_v = etalon_vector(one_class_bin_matrix)
            etalon_vectors.append(one_class_etalon_v)

        radius = radius_calculation(etalon_vectors, bin_matrices)
        print(f"Радіуси для класів {clas1} та {clas2}: {radius}")

        return [delta, radius]



    def hierarchical_structure(classes, vector):
        """Машинне навчання для кожної страти декурсивного дерева

        Ключові аргументи:
        classes -- класи для навчання
        vector -- варіаційний ряд
        result_d -- оптимальна дельта кожної страти
        result_r -- оптимальні радіуси кожного класу кожної страти
        """

        tree_pairs, result_d, result_r = [], [], []
        for i in range(len(classes)-1):
            tree_pairs = [classes[i], classes[i+1]]
```

```
            result_d, result_r = algorithm_for_each_strata(tree_pairs, vector[i], vector[i+1])

print("**********************************************************************")
            return result_d, result_r


        def main():
            input_data = ("N.txt", "B1.txt", "B2.txt", "B3.txt")

            Y = load_input_data(input_data)

            print("Пошук оптимальної дельта за паралельною оптимізацією контрольних допусків:")
            delta = delta_calculation(Y)

            # Обчислення бінарної матриці та еталонних векторів, відносно оптимальної дельта
            bin_matrices, etalon_vectors = [], []
            mean_vect_for_base_clas = mean_vector(Y[0])

            for one_class in Y:
                one_class_bin_matrix = binary_matrix(one_class, mean_vect_for_base_clas, delta)
                bin_matrices.append(one_class_bin_matrix)
                one_class_etalon_v = etalon_vector(one_class_bin_matrix)
                etalon_vectors.append(one_class_etalon_v)
```

```
    # Визначення варіаційного ряду
    found_variation_range, new_Y = variation_range(etalon_vectors, Y)

    # Пошук оптимальних значень для кожної страти декурсивного дерева
    hierarchical_structure(new_Y, found_variation_range)


if __name__ == "__main__":
    main()
```