КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій
(факультет)

інформаційних технологій
(кафедра)

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Розпізнавання кібератак за допомогою нейромереж PNN»

ЧЕРНЕНЬКИЙ ОЛЕКСАНДР ЮРІЙОВИЧ
(прізвище, ім’я та по-батькові студента повністю)

Київ 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ЗАТВЕРДЖУЮ
Завідувачка кафедри ІТ
д.т.н., професор Гончаренко Т.А.

„___” _____2025 року

КВАЛІФІКАЦІЙНА РОБОТА ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

на тему:

«Web-додаток для комп’ютерного моніторингу та обробки результатів досягнень учасників спортивних заходів»

Я як здобувач вищої освіти КНУБА розумію і підтримую політику закладу з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Здобувач
Черненький Олександр Юрійович

122 «Комп’ютерні науки»
(спеціальність)
Інформаційні управляючі системи і технології
(освітня програма)
Групи КН-21
Керівник Терентьєв О.О.
(прізвище та ініціали)
Доктор технічних наук, професор
(вчене звання, науковий ступінь)
Рецензент к.т.н., доц. Баліна О.І.
(Прізвище та ініціали)
Ідентичність підтверджую

Київ, 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: автоматизації і інформаційних технологій

Кафедра: інформаційних технологій

Освітній рівень: «бакалавр» за ОПП

Спеціальність: 122 «Комп`ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
д.т.н., професор Гончаренко Т.А.

_____
(підпис)
“____”_____ 2025р.

ЗАВДАННЯ
ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

Черненький Олександр Юрійович

(прізвище, ім'я, по-батькові)

1. Тема роботи: “ Розпізнавання кібератак за допомогою нейромереж PNN” затверджена наказом ректора КНУБА №235/23/25 від “14” лютого 2025 р
2. Керівник роботи: Терентьєв Олександр Олександрович, доктор технічних наук, професор кафедри ІТППМ
3. Строк подання студентом роботи до захисту: червень 2025 року
4. Зміст пояснювальної записки за розділами:
   Р.1. Аналіз предметної області та постановка задачі
   Р.2. Проектування принципу роботи системи. Інформаційне забезпечення
   Р.3. Математична модель роботи системи
   Р.4. Розробка та тестування програмного модулю
5. Інформаційні слайди:
   С. 2. Актуальність проблеми
   С 3. Постановка задачі
   С. 4. Системи виявлення вторгнень

С. 5. Модель чорної скриньки
С 6. Схема архітектури PNN
С 7. Алгоритм розпізнавання даних
С 8. Особливості PNN
С 9. Таблиця даних KDD-99
С. 10. Алгоритм нормалізації
С. 11. Інтрерфейс програмного забезпечення
С. 12-14. Результати тестування
С. 15. Аналіз проведеного тестування

6. Календарний план виконання робіт:

Види робіт та їх зміст	Дата виконання
Р. 1. Аналіз предметної області та постановка задачі	Лютий 2025 р.
Р. 2. Проектування принципу роботи системи. Інформаційне забезпечення	Березень 2025 р.
Р. 3. Математичне забезпечення системи	Квітень 2025 р.
Р. 4. Розробка програмного модулю	Травень 2025 р.
Остаточне оформлення роботи	Червень 2025 р.
Направлення роботи на рецензування	Червень 2025 р.
Попередній захист роботи на кафедрі	Червень 2025 р.

7. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта, представника комісії	Дата	Підпис
Розділ 1	к.т.н., доц. Шабала Є. Є.		
Розділ 2	к.т.н., доц. Шабала Є. Є.		
Розділ 3	к.т.н., доц. Шабала Є. Є.		
Розділ 4	к.т.н., доц. Шабала Є. Є.		
Прийом програмного продукту	к.т.н., доц. Шабала Є. Є.		

8. Дата видачі завдання: 13 лютого 2025 року

Керівник _____ Терентьєв О.О.
(підпис) (прізвище та ініціали)

Бакалавр _____ Черненький О.Ю.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

«Розпізнавання кібератак за допомогою нейромереж PNN».

Кваліфікаційна випускова робота бакалавра за спеціальністю: 122 «Комп’ютерні науки». – Київський національний університет будівництва та архітектури. – Київ, 2025.

Робота присвячена опису та поясненню систем виявлення вторгнень, сучасним шляхам каталогізації ознак вторгнень у вигляді таблиць та інтеграції нейромережевих технологій для покращення їх роботи та створенню тестового рішення на основі нейромережевого модулю на базі архітектури PNN.

Ключові слова: ймовірнісна нейронна мережа, мережева безпека, система виявлення вторгнень.

SUMMARY

«Cyberattack detection using PNN neural network».

Bachelor's final certification work in the specialty: 122 "Computer Science". - Kyiv National University of Construction and Architecture. - Kyiv, 2025.

The work is devoted to the description and explanation of intrusion detection systems, modern ways of cataloging intrusion features in the form of tables and integration of neural network technologies to improve their performance and create a test solution based on a neural network module that utilizes PNN architecture.

Index Terms: probabilistic neural network, network security, intrusion detection system.

ЗМІСТ

ВСТУП

Розвиток мережевих технологій надав великий простір для автоматизації виробництва, обробки даних та спілкування. Та навіть з початку історії комп’ютерних мереж не були новиною різні шляхи обходу захисту для перехоплення інформації чи просто виведення усієї системи з ладу. Це досить чітка система, тому якщо вивести з ладу декілька вузлів, можна нанести великі збитки для усієї схеми з’єднань та їх вузлів.

Такі дії називають кібератаками. Кібератака – це деяка усвідомлена зловмисна спроба людини чи організації проникнути в інформаційну систему іншої людини чи організації. Часто такі дії виконуються, щоб отримати деяку вигоду. Цей вид зловмисних дій включає в себе досить широкий спектр загроз, починаючи від ручних атак в середині мережі, закінчуючи складною схемою так званих ботнетів – комплексними схемами автономної дії, здатні виводити з ладу велику кількість вузлів для підвищення вірогідності підключення нових вузлів-жертв до системи.

В останній час у сфері мережевих технологій спостерігається досить чіткі тенденції в плані захисту корпоративних систем. За даними двох останніх щорічних докладів компанії Cisco за 2023 та 2024 рік, які є у вільному доступі, спостерігається тренд підвищення масованості атак, а саме під час проведення бенчмарку на базі великого азійського та тихоокеанічного ринку було встановлено, що в середньому за хвилину в великих підприємствах фіксується шість загроз, а половина таких атак ігнорується, причому компанії несуть досить великі збитки. В докладі за 2024 рік зазначається, що більшість зареєстрованих нападів на вразливості мережі нанесли збитки більше половини мільйона доларів США кожен [1, 2]. За даними PandaLabs загрози криються в бомбардуванні систем запитами з неідентифікованих девайсів та віддалених хостів, причому беручи кількістю а не якістю. Також компанія назвала головною метою зловмисників завдати якнайбільших збитків, в той час як шпигунство відійшло на друге місце [3].

Зазначається, що найбільшими вразливостями стали мобільні пристрої, які не мають адекватного захисту проти деяких видів загроз. Виходячи з цього наразі потрібна система оцінювання рівню загроз, яка буде працювати на рівні передачі трафіку, щоб запобігти атакам на рівні мережі.

У роботі висувається пропозиція вирішення цієї проблеми за допомогою технологій машинного навчання, а саме використання ймовірнісних нейронних мереж. У результаті очікується розробити тестовий модуль високої ефективності, розроблений на засадах використання невеликої вибірки, яка оновлюватиметься на деяких проміжках часу роботи системи та швидкому доступу до результатів, що надасть змогу вчасно відреагувати на загрози.

Мета роботи: Метою є розробка та дослідження моделі розпізнавання кібератак з використанням нейромережі Probabilistic Neural Network (PNN), що дозволяє підвищити точність і швидкість виявлення шкідливої активності в комп’ютерних системах.

Об’єкт дослідження: Є процес виявлення та класифікації кібератак у комп’ютерних мережах і системах.

Предмет дослідження: Є методи та алгоритми машинного навчання, зокрема нейронні мережі типу PNN, що використовуються для аналізу трафіку та виявлення ознак кібератак.

Методи дослідження:

- Аналіз та узагальнення наукових джерел з теми кібербезпеки та машинного навчання;
- Математичне моделювання та статистичний аналіз;
- Методи обробки даних та попередньої фільтрації трафіку;
- Машинне навчання з використанням Probabilistic Neural Network;

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної області

Кількість вразливостей, які знаходять зловмисники в програмних продуктах з кожним днем стає занадто великою для того, щоб вести контроль трафіку вручну. Деколи системи огляду системи можуть не завчасно виявити проблему і внаслідок цього зловмисник може отримати доступ до місця керування або зламати сервіс.

На сьогоднішній день усе більше і більше людей мають доступ до мережі Інтернет та володіють навичками володіння комп'ютерним обладнанням та кодом для цього обладнання. Із цього виходить проблема, пов'язана з тим, що з цим числом збільшується кількість злочинного програмного забезпечення та кібернетичних атак, які здійснюються щохвилинно. Попередження таких атак відбувається через системи ведення моніторингу ресурсу та системи запису його стану – так званих логів.

Саме зараз у зв'язку з бурхливим розвитком технологій як хмарні сховища та Інтернет речей, які все ще криють у собі досить багато недосліджених вразливостей, за якими необхідно вести оперативний нагляд та додавати дану інформацію до баз даних в режимі он-лайн. Оперативно з цим не зможе впоратися жоден спеціаліст, тож потребується автоматизована система, яка буде виявляти подібні атаки та їх ознаки.

Людство досить давно знайоме з концептами штучних нейромереж. Незважаючи на те, що досить довгий час нейромережі не були настільки популярними, як зараз, через те, що перша мережа була реалізована у вигляді обладнання, а не коду. Наразі комп'ютерні системи та архітектура сучасних мов програмування дають змогу побудувати математичну систему машинного навчання на базі програмних продуктів – простих у використанні та зрозумілих для написання.

Досить великою проблемою зараз є чіткий перехід усього програмного та апаратного забезпечення на філософію дизайну, яка робить усю взаємодію користувача максимально простою та невимушеною. Проте схема роботи цього забезпечення стала складніше у декілька разів з тих часів, коли особі потрібно будо спеціальне навчання для того, щоб використовувати частину функціоналу, яку раніше надавав комп’ютер.

На даний момент існує велика кількість програм, які працюють для захисту звичайного користувача від невідомих для нього загроз. Через закритість усіх систем адміністрування від користувача для забезпечення максимального комфорту користування продуктом, нагальною стає потреба в системі, яка зможе давати чіткі дані про стан безпеки системи та зможе змінюватися з часом, при цьому не втрачаючи правильності результатів та зменшуючи кількість ресурсів для підтримання та оновлення програмного забезпечення.

1.2 Опис принципу роботи нейронних мереж

Нейронні мережі, також відомі як штучні нейронні мережі (ANN) або модельовані нейронні мережі (SNN), є підмножиною машинного навчання і лежать в основі алгоритмів глибокого навчання. Їх ім'я та структура натхнені людським мозком, імітуючи спосіб сигналізації біологічних нейронів один одному.

Штучні нейронні мережі (ANN) складаються з шарів вузлів, що містять вхідний шар, один або кілька прихованих шарів та вихідний шар. Кожен вузол, або штучний нейрон, з'єднується з іншим і має відповідну вагу та поріг. Якщо вихід будь-якого окремого вузла перевищує вказане порогове значення, цей вузол активується, надсилаючи дані на наступний рівень мережі. В іншому випадку дані не передаються на наступний рівень мережі [4].

Нейронні мережі покладаються на навчальні дані, щоб навчитися та підвищувати їх точність з часом. Однак, як тільки ці алгоритми навчання точно налаштовані на точність, вони є потужними інструментами в галузі інформатики та штучного інтелекту, що дозволяє класифікувати та кластеризувати дані з великою швидкістю [5]. Завдання з розпізнавання мови чи розпізнавання зображень можуть зайняти хвилини проти годин у порівнянні з ручним визначенням людськими експертами. Однією з найбільш відомих нейронних мереж є алгоритм пошуку Google.

Після визначення вхідного рівня призначаються ваги. Ці ваги допомагають визначити важливість будь-якої даної змінної, причому більші вагоміші вносять більший внесок у результат порівняно з іншими вхідними даними. Потім усі вхідні дані множать на їх відповідні ваги, а потім підсумовують. Потім вихід передається через функцію активації, яка визначає вихід. Якщо цей вихід перевищує заданий поріг, він «спрацьовує» (або активує) вузол, передаючи дані наступному шару в мережі. Це призводить до того, що вихід одного вузла стає входом наступного вузла. Цей процес передачі даних з одного рівня на наступний рівень визначає цю нейронну мережу як мережу прямого зв'язку.

У сучасному світі такі мережі використовуються для виконання таких задач, як наприклад кластеризація, тобто ділення результатів за вхідними ознаками, та класифікація, тобто надання певного значення або статусу кейсу залежно від отриманого у кінці результату. Особливістю роботи нейромережі є те, що незважаючи на те, що саму систему треба передчасно навчити, після виконання цього процесу програмне або апаратне рішення зможе незалежно працювати та робити свої власні висновки незалежно від стороннього оператора, що дозволяє створити автономну систему, яка буде приймати будь-які коректні значення вводу та видавати деякий результат.

Тобто, нейронна мережа є досить наглядним прикладом інформаційної системи “чорної скриньки”. Найпростішою схемою роботи такої системи для нейромережі є наведений нижче приклад (рис 1.1):

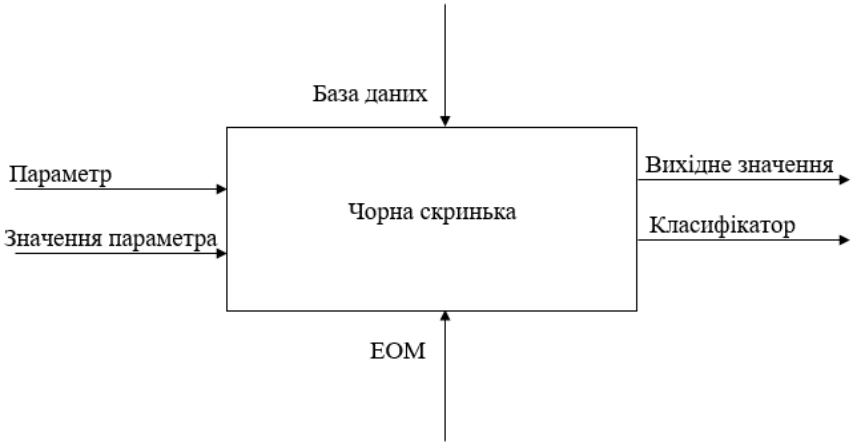


Рисунок 1.1 Модель “чорної скриньки”

На вхід чорної скриньки подають нормалізовані значення, тобто приведені до числового вигляду, часто репрезентуючи десятковий дріб у межах від значення нуля до одиниці. Ця процедура використовується для полегшення класифікації подання інформації. Тобто, з даними значеннями буде легше виразити результат у вигляді деякого відсотку. Ці дані беруться з бази даних, тобто нашої навчальної вибірки, за якими буде відбуватися процес підготовки нейромережі до функціонування.

Після результаті навчання та проходження деякого набору параметрів та їх значень отримуємо результат, часто у тому ж нормалізованому форматі або отримуємо вихід лише на один нейрон. У результаті мережа виконує одну з наведених вище задач, як класифікація чи кластеризація.

Усі нейронні мережі оперують деякою сукупність змінних параметрів, за допомогою яких система може корегувати процес отримання результату таким чином, щоб при вводі різних вибірок отримувати результати, які будуть найбільше відповідати дійсності.

Задля вирішення цього завдання між нейронами утворюють деякі зв’язки за допомогою проходу значень через операцію так званої функції активації. Функція активації – це деяка математична формула, за принципом якої відбувається сама робота системи. У сучасному світі використовуються разючо різні формули, залежно від поставленої задачі та коректності результатів, так як точного алгоритму вибору підходу для кожної задачі просто не існує. Тож, у результаті отримуємо, що налаштування мережі є досить ємким та складним процесом, для виконання якого потребується інтенсивне тестування різних методів реалізації та навчальних вибірок.

1.3 Визначення задач

Для початку роботи потрібно створити деяку абстрактну систему, щоб на базовому рівні зрозуміти передумови роботи даної системи. Визначення напрямків розвитку теоретичного базису ґрунтується на таких засадах:

- ефективне використання нейромережевих засобів вимагає розробки типових підходів до застосування нейромережевих моделей для розпізнавання різних видів кібератак;
- відсутність оперативності розпізнавання нових типів кібератак в основному пов'язано з тривалим нагромадженням статистичних даних, необхідних для навчання нейронної мережі. Для забезпечення оперативності можливо для навчання нейронної мережі використовувати експертні дані;
- забезпечити пристосованість нейромережевих засобів до варіативності умов застосування можливо за рахунок оптимізації виду і параметрів контролю захищеності автоматизованих інформаційних систем, що лежить в основі таких засобів;
- для адаптації нейромережевих засобів до функціонування при обмежених обчислювальних ресурсах необхідно як оптимізувати вигляд і параметри нейромережевої моделі, так і апріорно оцінювати обсяг обчислювальних ресурсів для її реалізації;

- використання нейромережевих засобів пов'язано з певним набором умов і обмежень, визначених умовами завдання оцінки і характеристиками нейромережевої моделі. Тому необхідно провести як визначення принципової доцільності застосування нейромережевих засобів, так і оцінку ефективності їх розробки;
- підвищити точність розпізнавання кібератак можливо за рахунок адаптації математичного забезпечення нейромережевих моделей до функціональних залежностей, відповідних процесів розпізнавання. Крім того, для підвищення точності розпізнавання тривалих кібератак доцільно використовувати марковский шаблон поведінки параметрів безпеки, який дозволяє частково нівелювати тимчасову складову процесу розпізнавання;
- використання нейронних мереж в високо відповідальних засобах розпізнавання кібератак вимагає теоретичної верифікації нейромережевих моделей оцінювання параметрів безпеки.

З вище вказаного можна зробити висновок, що у наш час від нейромереж, які виконують функції IDS, потребується досить висока точність визначення загроз, через велику кількість суміжних варіантів ознак трафіку, який проходить через мережу. Висока варіативність інформації, яка знаходиться в обігу, робить точне визначення причини та самого існування факту вторгнення досить складною задачею, навіть для нейромереж. В такій ситуації часто потребується мережа з композитною архітектурою, яка зможе врахувати якнайбільшу кількість виключень та хибних позитивів.

Тож, потребується створити мережу, яка зможе визначати атаки з деяким відсотком хибності, що буде перевірене тестовою вибіркою. За відсоток похибки мережі слід вважати кількість результатів, які не відповідають введеним параметрам. Це часто стається через широке варіювання складників або низьку кількість однієї з досліджуваних груп.

Так як систему, яка буде визначати усі типи атак створити неможливо, для цієї роботи буде створена нейромережа з відносно простою архітектурою,

яка повинна впоратися з виводом бінарного результату: хиба або істина, тобто чи відбулась атака, чи ні.

1.4 Класифікація систем виявлення вторгнень

Системи виявлення вторгнень мають чітку класифікацію. Існують три різні категорії систем виявлення вторгнень, які класифікуються в залежності від місця зчитування даних:

- Базовані на хостах IDS, що базується на хості, оцінює інформацію, знайдену в одній або декількох хост-системах, включаючи вміст операційних систем, файлів системи та програм.
- Мережеві IDS, які оцінюють інформацію, отриману від мережевих комунікацій, аналізуючи потік пакетів, що рухаються по мережі. Пакети захоплюються через набір датчиків.
- Оцінюючі вразливості IDS для оцінки вразливості, виявляються вразливості у внутрішніх мережах та брандмауерах

У даній роботі буде застосований тип, базований на хості-носії програми.

Основні моделі в IDS для аналізу подій для виявлення атак працюють за одним з двох принципів:

- Модель виявлення неправомірного використання: IDS виявляє вторгнення, шукаючи діяльність, яка відповідає відомим сигнатурам про вторгнення або вразливості
- Модель виявлення аномалій: IDS виявляє вторгнення, що відрізняються від встановленого шаблону для користувачів

Підходи до виявлення зловживання включають в себе:

- експертні системи, що містять набір правил, що описують атаки на мережу

- перевірка підпису, де сценарії атак перетворюються на послідовності аудиторських подій
- мережі Петрі, де відомі атаки представлені графічними мережами Петрі
- діаграми переходу держави, що представляють атаки з набором цілей та переходи з використанням випадків використання

Підходи до виявлення аномалій:

- виявлення порогового значення виявляє ненормальну активність на сервері або мережі, наприклад, ненормальну
- споживання центрального процесора для конкретного сервера або аномальна насиченість всієї мережі
- статистичні показники, дані, отримані з історичних цінностей
- заходи, засновані на правилах, правила, сформовані за допомогою експертних систем
- нелінійні алгоритми, такі як нейронні мережі та генетичні алгоритми [6]

Систему класифікації наведено нижче (рис 1.2)

Рисунок 1.2 Класифікація СВВ

Для виконання даної роботи я використаю вже існуючу базу даних, яка надасть інформацію для тестування системи та надасть декілька тестових

вхідних значень. Тож, використовуючи відомі дані, програма матиме змогу виступати і як у ролі моделі пошуку аномалій, так і у моделі пошуку аномалій, так як ми матимемо змогу до ознак нормального і злочинного трафіку.

1.5 Вибір інструментів реалізації модулю

1.5.1 Вибір архітектури нейромережі

Для роботи я вибрав ймовірнісну нейронну мережу, так звану PNN. Ймовірнісна нейронна мережа (PNN) - це нейромережа прямого зв'язку, яка широко використовується в задачах класифікації та розпізнавання образів. У алгоритмі PNN батьківська функція розподілу ймовірностей (PDF) кожного класу апроксимується вікном Парзена та непараметричною функцією. Потім, використовуючи PDF кожного класу, оцінюється ймовірність класу нових вхідних даних, а потім застосовується правило Байєса, щоб розподілити клас з найбільшою задньою ймовірністю до нових вхідних даних. За допомогою цього методу ймовірність помилкової класифікації зведена до мінімуму. Цей тип ANN був отриманий з байєсівської мережі та статистичного алгоритму, який називається дискримінантним аналізом Кернела Фішера. Він був введений Д.Ф. Шпехт у 1966 р [7]. У PNN операції організовані в багатошарову мережу прямого пересилання з чотирма шарами:

PNN часто використовується в проблемах класифікації. Коли присутній вхід, перший рівень обчислює відстань від вхідного вектора до навчальних вхідних векторів. Це створює вектор, де його елементи вказують, наскільки вхідні дані наближені до вхідних даних. Другий рівень підсумовує внесок для кожного класу вхідних даних і виробляє його чистий результат як вектор імовірностей. Нарешті, функція передачі конкурентів на виході другого рівня вибирає максимум з цих ймовірностей і створює 1 (позитивну ідентифікацію) для цього класу і 0 (негативну ідентифікацію) для нецільових класів.

Щоб класифікувати кібератаки за допомогою БШП, потрібно провести навчання нейронної мережі.

Навчання нейронної мережі - пошук такого набору вагових коефіцієнтів, при якому вхідний сигнал після проходу по мережі перетворюється в потрібний нам вихідний.

Це визначення «навчання нейронної мережі» відповідає і біологічним нейромережам. Наш мозок складається з величезної кількості пов'язаних одна з одною нейромереж, кожна з яких окремо складається з нейронів одного типу (з однаковою функцією активації). Наш мозок навчається завдяки зміні синапсів - елементів, які посилюють або послаблюють вхідний сигнал.

Якщо навчати мережу, використовуючи тільки один вхідний сигнал, то мережа просто «запам'ятає правильну відповідь», а як тільки ми подамо трохи змінений сигнал, замість правильної відповіді отримаємо нісенітницю. Ми чекаємо від мережі здатності узагальнювати якісь ознаки і вирішувати завдання на різних вхідних даних. Саме з цією метою і створюються навчальні вибірки.

Навчальна вибірка - кінцевий набір вхідних сигналів (іноді разом з правильними вихідними сигналами), за якими відбувається навчання мережі. Після навчання мережі, тобто коли мережа видає коректні результати для всіх вхідних сигналів з навчальної вибірки, її можна використовувати на практиці. Однак перш ніж відразу використовувати нейронну мережу, зазвичай виробляють оцінку якості її роботи на так званій тестової вибірці.

Тестова вибірка - кінцевий набір вхідних сигналів (іноді разом з правильними вихідними сигналами), за якими відбувається оцінка якості роботи мережі. Саме навчання нейронної мережі можна розділити на два підходи: навчання “з вчителем” і навчання “без вчителя”. У першому випадку ваги змінюються так, щоб відповіді мережі мінімально відрізнялися від уже готових правильних відповідей, а в другому випадку мережа самостійно класифікує вхідні сигнали. Метод навчання PNN – без вчителя. Для роботи з

PNN необхідно провести прохід значень через систему процесі так званого прямого проходу.

Кожен нейрон у вхідному рівні представляє змінну предиктора. У категоріальних змінних нейрони N-1 використовуються, коли існує N кількість категорій. Він стандартизує діапазон значень шляхом віднімання медіани та ділення на міжквартильний діапазон. Потім вхідні нейрони подають значення кожному з нейронів у прихованому шарі.

Шар візерунка містить по одному нейрону для кожного випадку у наборі навчальних даних. Він зберігає значення змінних предиктора для випадку разом із цільовим значенням. Прихований нейрон обчислює евклідову відстань тесту від центральної точки нейрона, а потім застосовує функцію ядра радіальної базисної функції, використовуючи значення сигми.
Підсумовуючий шар

Для PNN існує один нейрон шаблону для кожної категорії цільової змінної. Фактична цільова категорія кожного навчального випадку зберігається з кожним прихованим нейроном; зважене значення, що виходить із прихованого нейрона, надходить лише на нейрон-шаблон, який відповідає категорії прихованого нейрона. Нейрони шаблону додають значення для класу, який вони представляють.

Вихідний рівень порівнює зважені голоси для кожної цільової категорії, накопиченої в шарі шаблону, і використовує найбільший голос для прогнозування цільової категорії.

Для побудови нашого додатку буде використане прирівняння кожного результату до однієї з двох бінарних категорій, – нуль і одиниця – які будуть використані для прогнозування та подальшої роботи програми після попереднього навчання. Тобто, наприклад, усі варіанти вторгнення одного виду будуть видавати наприкінці навчання значення, рівне одиниці, незалежно від варіювання вхідних параметрів, а нормальний трафік, за допомогою якого і будуть виявлятися аномалії, буде завжди рівний нулю.

Причому треба брати до уваги спотворення результатів у процесі навчання. Якщо мережа будується з багатьох різних даних, коефіцієнти будуть мати деяку тенденцію, в залежності від кількості схожих навчальних даних.

1.5.2 Аналіз альтернативних варіантів реалізації модулю

Для того, щоб проаналізувати вибір структури PNN, слід розглянути інші варіанти реалізації модулю. Серед найбільш розповсюджених архітектур я виділив такі як багатошаровий персептрон та конволюційну (згорткову) нейронну мережу (CNN). Проаналізуємо варіанти та задачі, які вони вирішують.

Багатошаровий персептрон (БШП) - це клас прямої штучної нейронної мережі (ANN) [21]. Термін БШП використовується неоднозначно, іноді вільно для будь-якого прямого ANN, іноді строго для позначення мереж, що складаються з декількох шарів персептронів (з пороговою активацією). Багатошарові персептрони іноді розмовно називають "простими нейромережами", особливо коли вони мають один прихований шар.

БШП складається щонайменше з трьох шарів вузлів: вхідного шару, прихованого шару та вихідного шару. За винятком вхідних вузлів, кожен вузол є нейроном, який використовує нелінійну функцію активації. MLP використовує контрольовану техніку навчання, яка називається зворотним розповсюдженням для навчання. Його багатошаровість і нелінійна активація відрізняють MLP від лінійного персептрона. Він може розрізняти дані, які не можна лінійно розділити.

Згорткова нейронна мережа (англ. Convolutional neural network, CNN) - спеціальна архітектура штучних нейронних мереж, запропонована Яном Лекуном в 1988 році і націлена на ефективне розпізнавання образів, входить

до складу технологій глибокого навчання (англ. Deep learning) . Назва архітектура мережі отримала через наявність операції згортки, суть якої в тому, що кожен фрагмент зображення множиться на матрицю (ядро) згортки поелементно, а результат підсумовується і записується в аналогічну позицію вихідного зображення. Використовує деякі особливості зорової кори, в якій були відкриті так звані прості клітини, що реагують на прямі лінії під різними кутами, і складні клітини, реакція яких пов'язана з активацією певного набору простих клітин. Таким чином, ідея згорткових нейронних мереж полягає в чергуванні згорткових шарів (англ. Convolution layers) і субдіскретізіруючих шарів (англ. Subsampling layers або англ. Pooling layers, шарів підвибірки). Структура мережі - односпрямована (без зворотних зв'язків), принципово багатошарова. Для навчання використовуються стандартні методи, найчастіше метод зворотного поширення помилки. Функція активації нейронів (передавальна функція) - будь-яка, за вибором дослідника.

З вище сказаного робимо висновок, що використання інших видів нейромереж не є доцільним через погане масштабування щодо кількості нейронів на вхідному шарі. Якщо ми матимемо на меті покращити роботу такої системи за допомогою збільшення шарів або використання гібридної мережі, час для розрахунків ставатиме більше занадто швидко.

1.5.3 Вибір мови та середи реалізації

Для реалізації проекту буде застосована така мова програмування, як C# — потужна та універсальна об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET.

Основними перевагами цієї мови є досить широкий вибір документації, підтримка більшості сучасних систем та відносна простота розуміння коду. C# За задумом, C# є мовою програмування, яка найбільш безпосередньо відображає основну інфраструктуру загальної мови (CLI).

Середою реалізації користувацького інтерфейсу я вибрав вбудоване в середу розробки Visual Studio рішення для створення простих користувацьких інтерфейсів як Windows Forms. Ця утиліта підтримує вибрану мову програмування та дозволяє швидко оформити працюючий прототип деякої програми.

Windows Forms (WinForms) - це безкоштовна графічна бібліотека класів із відкритим вихідним кодом (GUI), що входить до складу Microsoft .NET Framework або Mono Framework, забезпечуючи платформу для написання розширених клієнтських програм для настільних, портативних та планшетних ПК. Хоча це розглядається як заміна попередньої та більш складної бібліотеки класів Microsoft Foundation на базі C ++, вона не пропонує порівнянної парадигми, а виступає лише платформою для рівня користувальницького інтерфейсу у багаторівневому рішенні [8].

На заході Microsoft Connect 4 грудня 2018 року Microsoft оголосила про випуск Windows Forms як проект з відкритим кодом на GitHub. Він випускається за ліцензією MIT. З цим випуском Windows Forms став доступним для проектів, націлених на платформу .NET Core. Однак структура все ще доступна лише на платформі Windows, а неповна реалізація Mono Windows Forms залишається єдиною крос-платформною реалізацією. Незважаючи на це, рішення задовольняє усі вимоги до розробки та тестування концепту програмного рішення.

1.5.4 Вибір бази даних

Для навчання мережі та перевірки справності її роботи у цій роботі буде використана база даних KDD 99.

З 1999 року KDD 99 є найбільш широко використовуваним набором даних для оцінки методів виявлення аномалій [9].

Цей набір даних підготовлений побудований на основі даних, отриманих під час оцінки IDS DARPA’98. DARPA’98 - це близько 4 гігабайт

стисненого необроблені (двійкові) дані tcpdump за 7 тижнів мережевого трафіку, які можна обробити приблизно в 5 мільйонів записів підключення, кожен із приблизно 100 байтами [10]. За два тижні тестових даних налічується близько 2 мільйонів записів про підключення.

Набір навчальних даних KDD складається приблизно з 4 900 000 одинарних векторів підключення, кожен з яких містить 41 функцію і позначений як звичайний або атакований, з рівно одним конкретним типом атаки. Імітовані атаки поділяються на одну з наступних чотирьох категорій:

1) Атака відмови в обслуговуванні (DoS): це атака, при якій зловмисник робить деякий обчислювальний ресурс або ресурс пам'яті занадто зайнятим або занадто заповненим, щоб обробляти законні запити, або забороняє законним користувачам доступ до машини.

2) User to Root Attack (U2R): це клас вразливості, в якому зловмисник починає з доступу до звичайного облікового запису користувача в системі (можливо, отриманого за допомогою прослуховування паролів, атаки за словником або соціальної інженерії) і здатний скористатися деякою уразливістю, щоб отримати root-доступ до системи.

3) Remote to Local Attack (R2L): відбувається, коли зловмисник, який має можливість надсилати пакети на машину через мережу, але який не має облікового запису на цій машині, використовує певну вразливість для отримання локального доступу як користувач цієї машини.

4) Probing Attack: спроба зібрати інформацію про мережу комп’ютерів з очевидною метою обходу засобів контролю безпеки.

Від розробленого модулю буде потребуватися мати досить високу толерантність до введених даних та давати чітку відповідь на питання про те, чи відбулася атака, незалежно від її типу. Для цього можна буде прирівняти усі вище наведені вище атаки до одного класу в плані роботи нейромережевої інфраструктури та надавати інформацію про знайдену атаку вже після аналізу. Система повинна швидко попереджувати атаки, тип атаки не повинен

бути визначним фактором для переривання зв’язку зі зловмисником, таким фактором повинен бути лише сам факт наявності спроби вторгнення.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРИНЦИПУ РОБОТИ СИСТЕМИ. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Побудова роботи рішення

2.1.1 Візуальна схема роботи нейромережі

Для розуміння принципу побудови рішення у формі коду для початку треба візуально задати абстрактну математичну модель роботи даного рішення. Для виконання цієї задачі використовують схеми алгоритмів, візуалізація та діаграми прецедентів.

Для репрезентації принципу роботи нейромережі використовують спеціальні діаграми, на яких вказується кількість шарів нейронів, архітектура мережі та структура зв’язків між ними. Їх можна віднести до так званих діаграм діяльності, які показують принципи роботи об’єкта.

У процесі створення нейронної мережі, яка полягає в основі нашої системи виявлення вторгнень, буде використана PNN. Тобто мають місце бути два шари, властиві для цього типу архітектури, як шар образів та сумуючий шар, а структура зв’язків буде повнозв’язною [11]. Крім того, треба показати вхідний і вихідний шари. Окремі нейрони, які позначають колами, групуються в ряд чи стовпчик, залежно від їх належності до вказаного шару. Після чого їх з’єднують стрілками, щоб показати, процес обробки кожного вхідних значень.

На схемі необхідно зазначити можливості мережі вцілому, як кількість виходів та входів, які підтримує мережа.

Для початку побудуємо схему роботи нейрона. Штучний нейрон або просто нейрон – найменша неподільна частина нейронної мережі. Робота цього елемента базується на принципах роботи нейронів в живому організмі, хоча математичне представлення, на відміну від популярної думки, є досить тривіальним.

Модель нейрону складається з функції активації, ваги та суматора. Функція активації задає тренд для вектору, суматор використовується для зміни значення за заданою вагою, яка змінюється в процесі навчання для підлаштування до бажаного результату.

З наведеного вище, ми маємо достатньо інформації для побудови схеми нейромережі для деякої кількості вхідних ознак. У результаті отримуємо схему, яка буде мати вигляд (рис 2.1):

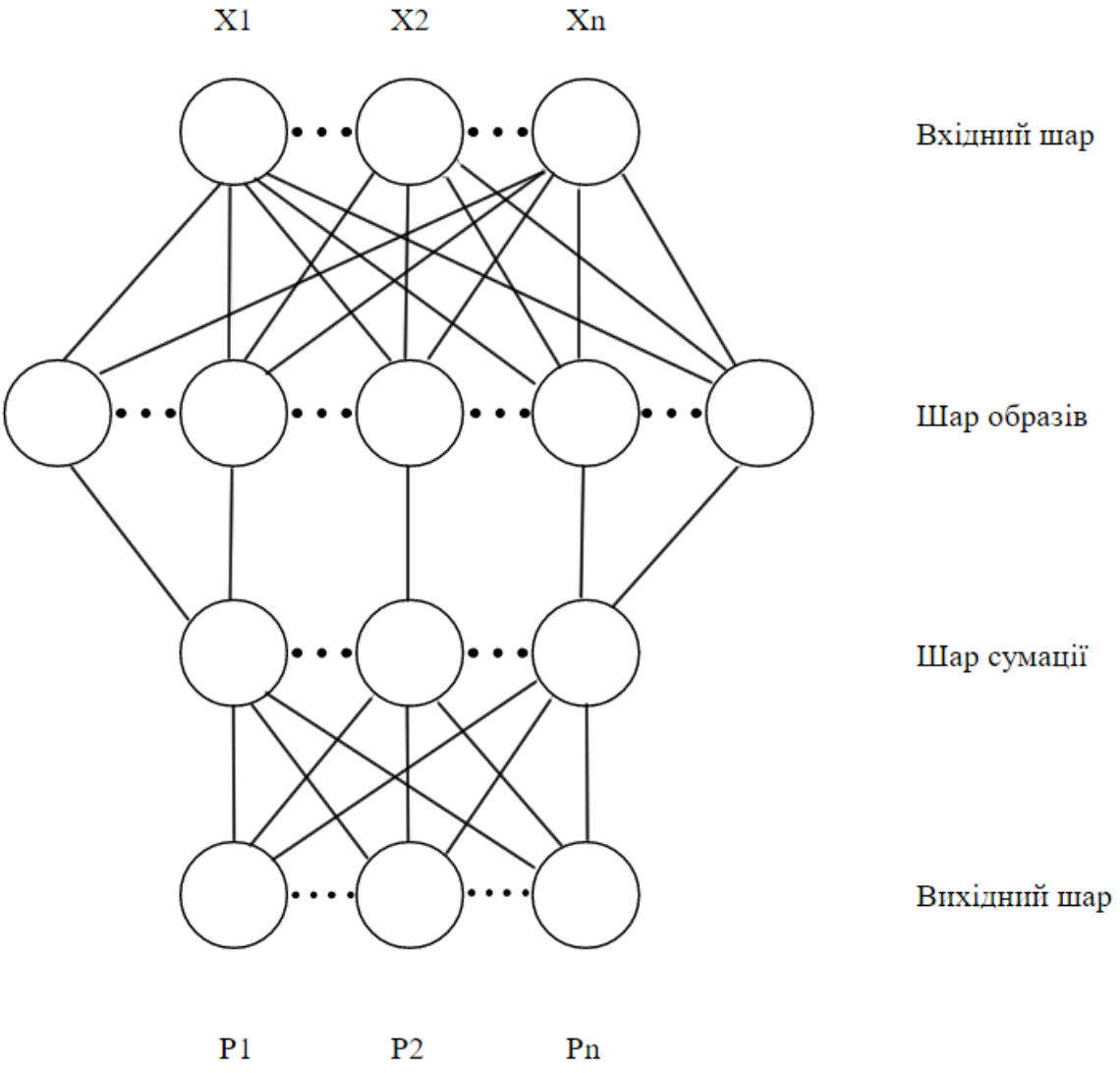


Рисунок 2.1 Схема принципу роботи нейромережі PNN

2.1.2 Блок-схема алгоритму

Блок-схема - це тип діаграми, що пояснює робочий план або деякий процес. Блок-схему також можна визначити як схематичне зображення алгоритму, поетапний підхід до вирішення завдання. Для виконання цього завдання блок-схема буде використана для відображення алгоритму роботи системи.

Блок-схема показує кроки у вигляді коробок різного типу та вказує їх порядок за допомогою стрілок Вони використовуються при проектуванні та створенні документації простих процесів або програм. Як і інші типи діаграм, вони допомагають графічно відобразити процес, тим самим допомагають його зрозуміти, та часто звертають увагу на деякі специфічні місця роботи рішення або процесу.

Існують різні типи блок-схем: кожен тип має власний набір блоків та позначень. Три найпоширеніші типи вікон у блок-схемі:

- Етап діяльності, який зазвичай позначають прямокутником.
- Рішення, яке зазвичай позначається ромбом.
- Ввід та вивід інформації, який позначають паралелограмом

Блок-схема описується як "міжфункціональна", коли діаграма розділена на різні вертикальні або горизонтальні частини, для опису управління різними організаційними підрозділами. Символ, що з’являється в певній частині, знаходиться під контролем цієї організаційної одиниці. Міжфункціональна блок-схема дозволяє автору правильно розподілити відповідальність за виконання дії чи прийняття рішення та показати відповідальність кожного організаційного підрозділу за різні частини одного процесу.

Блок-схеми відображають певні аспекти процесів і зазвичай доповнюються іншими типами діаграм. Наприклад, Каору Ісікава, визнаний фахівець в галузі управління якістю, визначив блок-схему як один із семи основних інструментів контролю якості, поряд з гістограмою, діаграмою

Парето, контрольним аркушем, діаграмою контролю, діаграмою причинно-наслідкових наслідків та діаграмою розсіювання. Подібним чином, в UML, стандартному позначенні концептуального моделювання, що використовується при розробці програмного забезпечення, діаграма діяльності, яка є типом блок-схеми, є лише одним із багатьох різних типів діаграм.

З наведених вище даних, спрощена блок-схема роботи нейромережі PNN виглядатиме так (рис. 2.2):

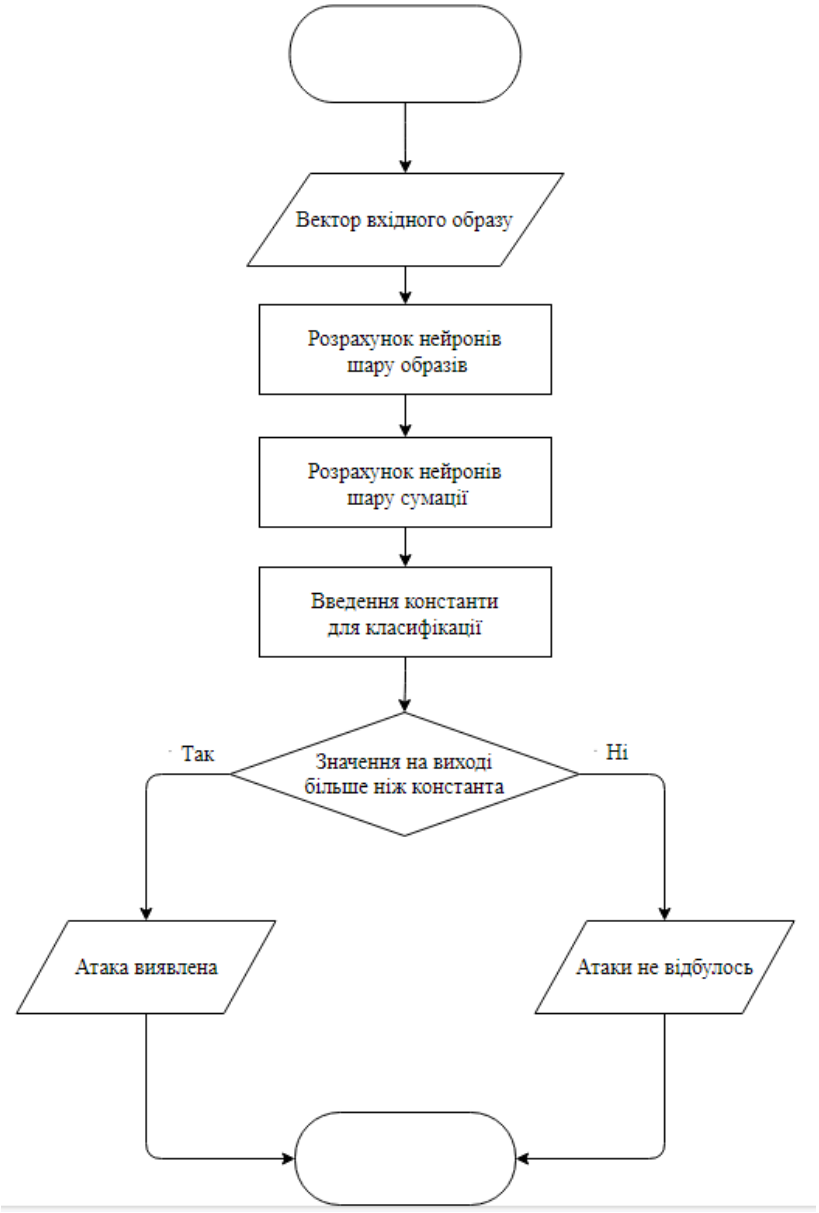


Рисунок 2.2 Алгоритм роботи нейромережі PNN

2.2 Робота з базою даних

2.2.1 Загальна інформація про KDD-99

KDD-99 - це вибірка даних, яка містить типові дані для різних типів загроз та нормального трафіку, що використовувалась на “The Third International Knowledge Discovery and Data Mining Tools Competition”. Він проводився спільно з KDD-99 на П'ятій міжнародній конференції з виявлення дати та обміну даними. Конкурсним завданням було побудувати мережевий детектор вторгнень, прогнозовану модель, здатну розрізняти «погані» з'єднання, що називаються вторгненнями або атаками, із «хорошими» нормальними з'єднаннями. Ця база даних містить стандартний набір даних, що підлягають аудиту, який включає широкий спектр вторгнень, змодельованих у середовищі мережі [12].

Хоча набору даних KDD-99 понад 15 років, він все ще широко використовується в академічних дослідженнях. Дослідити широке використання цього набору даних у дослідженнях машинного навчання та системах виявлення вторгнень. Кількість опублікованих досліджень показує, що KDD-99 є найбільш часто використовуваним набором даних в IDS та областях машинного навчання, і це фактично набір даних для цих областей дослідження. Саме тому вона була вибрана для тестування поточного проекту.

Популярність KDD-99 забезпечує легкий доступ до самого набору даних та відносна легкість обробки даних, що часто робить цю вибірку одним з небагатьох загальноприйнятих способів навчання систем виявлення вторгнень.

Незважаючи на це навіть в середовищі широкого використання таблиці, у даний момент часу існує досить велика критика KDD-99 на рахунок її сучасності. У наш час існує велика кількість різноманітних систем оцінювання ризиків та аналізу трафіку, таких як UNSW-NB15 та NSL-KDD. Проте великою перевагою саме цієї таблиці є легкість оброки та підготовки параметрів вхідних сигналів та доступність повної вибірки у відкритому доступі, тобто будь хто може завантажити свою версію документа та провести дослідження якості роботи нейромережі.

Тож з усього вище сказаного можна сказати, що схема цієї таблиці підходить для тестування нейронних мереж та вирішення задач, пов’язаних з системами виявлення вторгнень. Також вона зарекомендувала себе як надійний інструмент, який широко використовується в сфері інформаційних технологій і донині, незважаючи на деякі

2.2.2 Аналіз навчальної вибірки

Розглянемо склад навчальної вибірки. Використовувана база даних складається з вхідних ознак і ідентифікаторів, які відповідають певним типам атак, так як і звичайному трафіку. Всього у даній вибірці є 41 вхідний параметр і 1 вихідний параметр. Ознаки, створені для бази даних, включають в себе наведені параметри:

1. Duration - довжина (кількість секунд) з'єднання;
2. Protocol type - тип протоколу, наприклад tcp, udp тощо;
3. Service - послуга мережі в пункті призначення, наприклад, http, telnet тощо;
4. Src_byte - кількість байтів даних від джерела до місця призначення;
5. Dst_byte - кількість байтів даних від місця призначення до джерела;
6. Flag - нормальний стан або стан помилки з'єднання;
7. Land - 1, якщо з'єднання знаходиться з / до того ж хоста / порту; 0 інакше;
8. Wrong_fragment - кількість `` неправильних " фрагментів;

9. Urgent - кількість термінових пакетів;
10. Hot - кількість "гарячих" показників;
11. Num_failed_logins - кількість невдалих спроб входу;
12. Logged_in - 1 при успішному вході; 0 інакше;
13. Num_compromised - кількість `` компрометованих " умов;
14. Root-shell - 1, якщо отримана коренева оболонка; 0 інакше;
15. Su_attempted - 1, якщо спроба команди `` su root "; 0 інакше;
16. Num_root - кількість звернень `` root ";
17. Num_file_creations - кількість операцій зі створення файлів;
18. Num_shells - кількість підказок оболонки;
19. Num_access_shells - кількість операцій над файлами контролю доступу;
20. Num_outbound_cmds - кількість вихідних команд у сеансі ftp;
21. Is_hot_login - 1, якщо логін належить до списку "гарячих"; 0 інакше;
22. Is_guest_login - 1, якщо логін - гостьовий реєстратор; 0 інакше;
23. Count - кількість підключень до того ж хоста, що і поточне з'єднання за останні дві секунди;
24. Serror_rate - відсоток з'єднань, які мають помилки `` SYN ";
25. Rerror_rate - відсоток з'єднань, які мають помилки `REJ ';
26. Same_srv_rate - відсоток підключень до тієї ж послуги;
27. Diff_sr _rate - відсоток підключень до різних служб;
28. Srv_count - кількість підключень до тієї ж послуги, що і поточне з'єднання за останні дві секунди;
29. Srv_serror_rate - відсоток з'єднань, які мають помилки `` SYN ";
30. Srv_rerror_rate - відсоток з'єднань, які мають помилки `REJ ';
31. Srv_diff_host_rate - відсоток підключень до різних хостів;
32. Dst_host_count - кількість підключень, що мають один і той же порт призначення;
33. Dst_host_srv_count - кількість підключень, що мають один і той же порт призначення та використовують ту саму послугу;

34. Dst_host_same_srv_count - відсоток з'єднань, що мають один і той же порт призначення та використовують ту саму послугу;
35. Dst_host_diff_srv_count - відсоток різних послуг на поточному хості;
36. Dst_host_same_src_port_rate - відсоток підключень до поточного хоста, який має той самий порт src;
37. Dst_host_srv_diff_host_rate - відсоток підключень до тієї ж послуги, що надходять від різних хостів;
38. Dst_host_serror_rate - відсоток підключень до поточного хоста, які мають помилку S0;
39. Dst_host_srv_serror_rate - відсоток підключень до поточного хоста та вказаної служби, які мають помилку S0;
40. Dst_host_rerror_rate - підключень до поточного хоста, які мають помилку RST;
41. Dst_host_srv_rerror_rate - відсоток підключень до поточного хоста та вказаної служби, що мають помилку RST [13].

Останнім значенням в вибірці визначений тип атаки або непричетність вказаних ознак до будь-яких загроз, визначений як нормальний трафік. Для нормальної роботи системи визначення вторгень потрібно відібрати схожі за цими параметрами значення, щоб уникнути так званих “хибних позитивів” – випадків, коли результат під час тестування відповідності вихідного параметра вхідним даним продукт видає хибну відповідь.

Для подолання цієї загрози потрібно виконати налаштування вибірки для роботи системи, яка буде на ній навчатися. Правила для підготовки навчальної вибірки включають в себе:

- Групування схожих навчальних даних
- Задання чітких відмінностей різних варіантів виходу та повна дискретизація даних
- Нормалізування даних для роботи

- Для бінарних систем важливе подання однакової кількості навчальних ознак двох різних очікуваних вихідних даних

У результаті повинна бути отримана чиста вибірка з коректними даними, за допомогою яких можна буде якнайточніше навчити систему. Робота з базами даних є найголовнішою частиною розвитку сфери нейронних мереж через те, що на корпоративному рівні вони працюють за принципом “чорної скриньки”, і єдина інформація, яка впливатиме на хід навчання та розпізнавання, за умови справності роботи коду, є сама база даних.

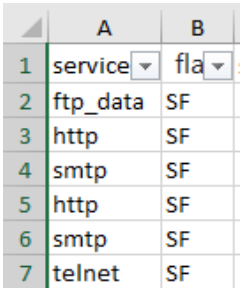
2.2.3 Нормалізація даних

Для подання даних для навчання потрібно деяким чином обробити вхідну інформацію, щоб система, яка працює з числовими значеннями могла приймати інформацію. Цей процес називається нормалізацією.

Нормалізація даних - це коригування значень відповідно до деякими функціями перетворення, з метою зробити їх більш зручними для порівняння. Нормалізація даних потрібне для випадків, коли несумісність одиниць вимірювань змінних може відбитися на результатах і рекомендується, коли підсумкові звіти можуть бути поліпшені, якщо висловити результати в певних зрозумілих або сумісних одиницях. Наприклад, час реакції, записаний в мілісекундах, легше інтерпретувати, ніж число тактів процесора, які були отримані дані експерименту.

Тобто, можна сказати, що цей процес створений для того, щоб привести усі значення до спільного знаменника. Для цього потрібно буде виконати наступні дії.

Для початку роботи над нормалізаціям даних ми маємо змінити текстову інформацію на чисельну. Для цього надамо однаковим текстовим значенням відповідні чисельні значення. У цьому випадку необхідно провести цю операцію з рядками сервісу, прапорця та типу трафіку (рис 2.3).



	A	B
1	service	fla
2	ftp_data	SF
3	http	SF
4	smtp	SF
5	http	SF
6	smtp	SF
7	telnet	SF

Рисунок 2.3 Текстові дані в вибірці

По-друге необхідно буде кодування нашої відкоректованої вибірки. Під кодуванням мається на увазі заміна усіх символьних значень на чисельні. Для цього потрібно в документі використати інструменти пошуку для таблиці та профільтрувати усі стовбці в яких є будь-які символьні позначення (рис 2.4).

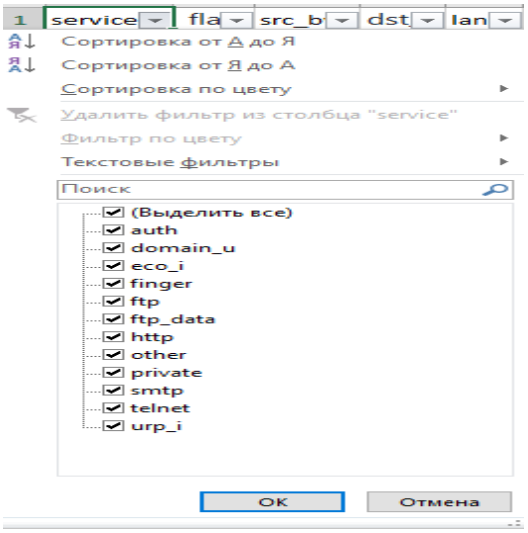


Рисунок 2.4 Вікно фільтрації з усіма тесктовими значеннями

Далі треба за допомогою заміни всі значення, які ми знайшли, записати у першому рядку, а у другому записати число на котре ми бажаємо його замінити. Для кожного нового текстового значення тут буде використаний інкремент попереднього позначення (рис 2.5).

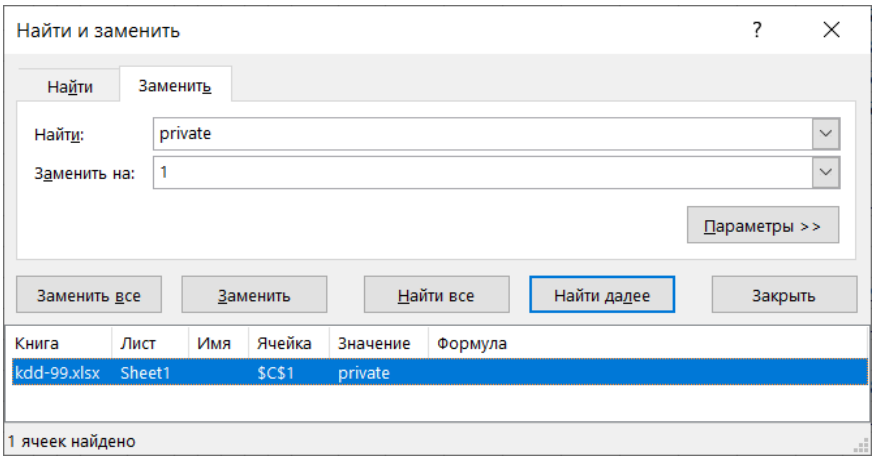


Рисунок 2.5 Заміна символів у KDD-99

Результат такого кодування можна переглянути за допомогою інструменту фільтрації в вибраному програмному забезпеченні для перегляду таблиці. В цьому випадку у пошуковому рядку повинні бути відображені тільки цифрові значення (рис 2.6).

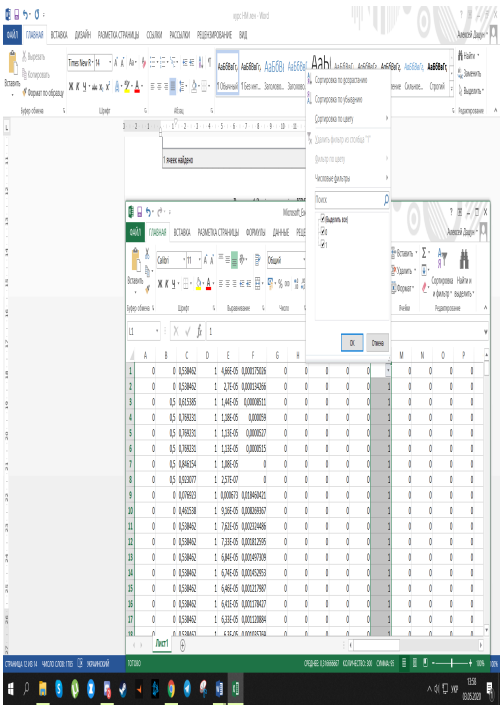


Рисунок 2.6 Перевірка стовпця на правильність значень

Як можемо переконатися, вибірка готова до останнього виду маніпуляції даними – ранжування усіх унікальних ознак значеннями від нуля

до одного. Для цього потрібно нашу, вже закодовану вибірку, а точніше кожне її значення, підставити під формулу нормалізації (2.1):

$$X^* = \frac{X - \min(X)}{\max(X) - \min(X)} \quad (2.1)$$

Де:

X* - це вже нормалізоване значення;

X- значення яке потребує нормалізації;

min(X)- мінімальне значення атрибута;

max(X)- максимальне значення атрибута;

Основний закон, який повинен моделюватись мережею повинен добре прослідживатись в навчальних даних, а не затінюватись в них несуттєвими закономірностями. Для цього навчальні дані перед використанням в НМ проходять попередню обробку. Під цією обробкою розуміється нормалізація даних, їх фільтрація та перекодування. Вважається, що в багатьох практичних сферах НМ дозволяють досягти помилки узагальнення 90% при одночасній помилці апроксимації ≈98-100%.

Взагалі нормалізація даних – це одна з операцій перетворення ознак, на етапі підготовки даних. Для нормалізація даних ми використовуємо підхід на основі мінімум та максимум.

Метою нормалізація є:

- Зменшення обсягу для зберігання даних.
- Підвищення ефективності роботи БД.

В результаті нормалізація даних ми повинні отримати числа на проміжку від 0 до 1. А для того, щоб у подальшому правильно використати нашу формулу нам потрібно записати мінімальні та максимальні значення кожного стовбця у новий рядок (рис 2.7).

12	3	88382	834163	0	3	0	7	0	1	611	1	2	684	2	1	3	0	0	1	511	511	1	1	1	1	1	1	1	255	255	1	0,93	1	1	0,9	1	1	1	2	21
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	1

Рисунок 2.7 Приклад перетворених значень

Після цього нам треба використовуючи функціонал програми для перегляду таблиць, якщо бути більш точним її здібності до розрахунку елементів по формулі, створити ще одну таблицю кожен елемент якої буде замінений результатом формули записаної вище. Досить важливим є момент вводу правильних формул, так як якість ймовірнісної нейронної мережі перш за все описується якістю введених результатів.

Для цього нам потрібно обрати пусту клітинку нашого файлу, після чого у рядок для формули записати такі дані (рис 2.8):

```
=ABS((A1-$A$254)/($A$253-$A$254))
```

Рисунок 2.8 Формула для нормалізація даних в середі Microsoft Excel

У результаті була отримана методика обробки даних для подачі їх на вхід нейронної мережі. Алгоритм проведених розрахунків матиме такий вигляд (рис. 2.9):

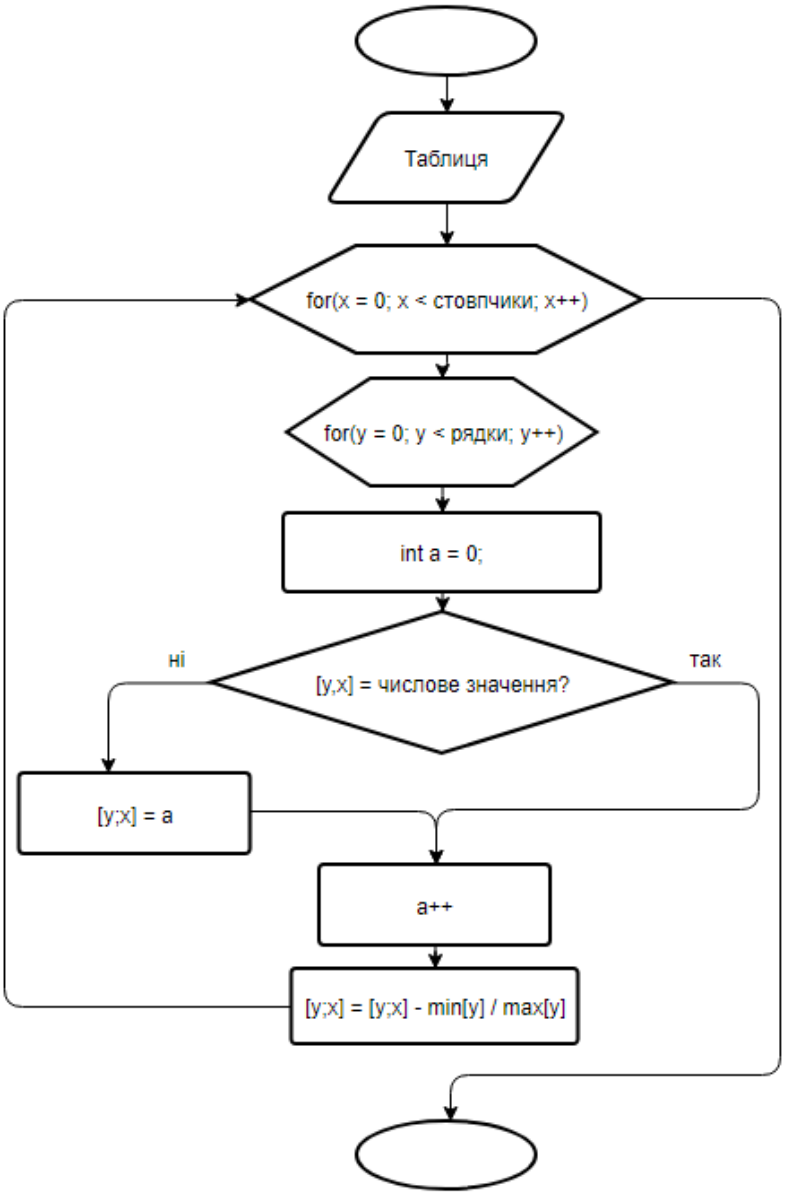
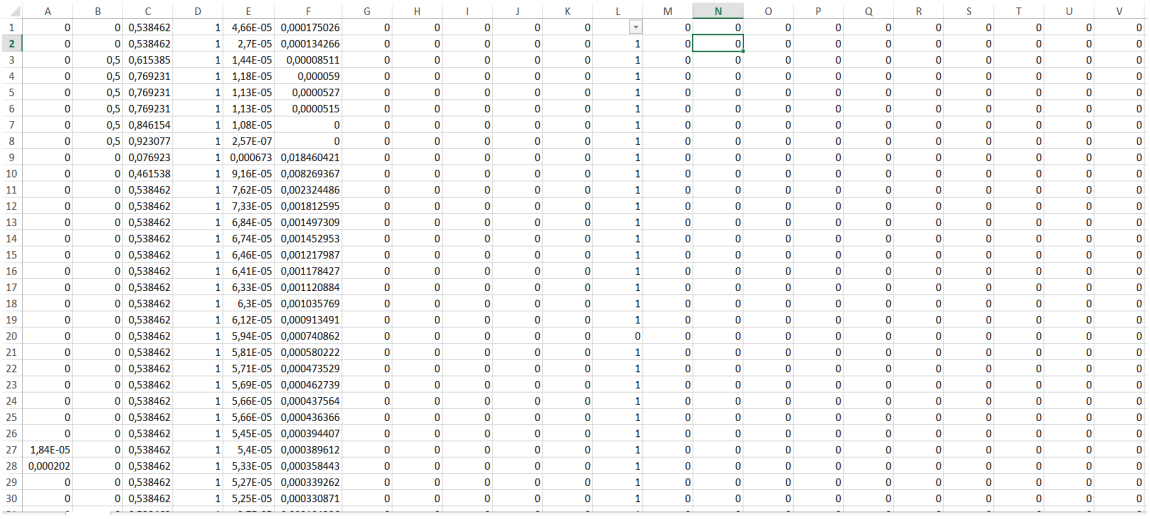


Рисунок 2.9 Алгоритм нормалізації даних для KDD-99

Після проходження всіх етапів перетворення KDD-99 до її нормалізованої копії ми повинні отримати таблицю, у якій всі елементи будуть знаходитись на проміжку від 0 до 1 (рис 2.10).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	0	0	0,538462	1	4,66E-05	0,000175026	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
2	0	0	0,538462	1	2,7E-05	0,000134266	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
3	0	0,5	0,615385	1	1,44E-05	0,00008511	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
4	0	0,5	0,769231	1	1,18E-05	0,000059	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
5	0	0,5	0,769231	1	1,13E-05	0,0000527	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
6	0	0,5	0,769231	1	1,13E-05	0,0000515	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
7	0	0,5	0,846154	1	1,08E-05	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
8	0	0,5	0,923077	1	2,57E-07	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0,076923	1	0,000673	0,018460421	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0,461538	1	9,16E-05	0,008269367	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
11	0	0	0,538462	1	7,62E-05	0,002324486	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
12	0	0	0,538462	1	7,33E-05	0,001812595	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
13	0	0	0,538462	1	6,84E-05	0,001497309	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
14	0	0	0,538462	1	6,74E-05	0,001452953	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
15	0	0	0,538462	1	6,46E-05	0,001217987	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
16	0	0	0,538462	1	6,41E-05	0,001178427	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
17	0	0	0,538462	1	6,33E-05	0,001120884	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
18	0	0	0,538462	1	6,3E-05	0,001035769	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
19	0	0	0,538462	1	6,12E-05	0,000913491	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
20	0	0	0,538462	1	5,94E-05	0,000740862	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0,538462	1	5,81E-05	0,000580222	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
22	0	0	0,538462	1	5,71E-05	0,000473529	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
23	0	0	0,538462	1	5,69E-05	0,000462739	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
24	0	0	0,538462	1	5,66E-05	0,000437564	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
25	0	0	0,538462	1	5,66E-05	0,000436366	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
26	0	0	0,538462	1	5,45E-05	0,000394407	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
27	1,84E-05	0	0,538462	1	5,4E-05	0,000389612	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
28	0,000202	0	0,538462	1	5,33E-05	0,000358443	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
29	0	0	0,538462	1	5,27E-05	0,000339262	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
30	0	0	0,538462	1	5,25E-05	0,000330871	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Рисунок 2.10 Нормалізована навчальна вибірка

У результаті буде отримана вибірка, ознаки якої можна буде подавати на вхід мережі.

2.3 Опис функціоналу Windows Forms

Windows Forms - це технологія інтерфейсу користувача .NET та набір керованих бібліотек, що спрощує загальні завдання програми, такі як читання та запис у файлову систему. Коли ви використовуєте середовище розробки, таке як Visual Studio, ви можете створювати програми для смарт-клієнта Windows Forms, які відображають інформацію, вимагають введення даних у користувачів та спілкуються з віддаленими комп’ютерами через мережу.

У Windows Forms форма - це візуальна середа, на якій ви відображаєте інформацію користувачеві. Коли користувач робить щось із формою або одним із її елементів керування, дія генерує подію. І так додаток реагує на ці події кодом і обробляє події, коли вони відбуваються.

Windows Forms містить різноманітні елементи керування, які можна додати до форм: елементи керування, які відображають текстові поля, кнопки, випадаючі поля, перемикачі та навіть веб-сторінки. Якщо існуючий елемент керування не відповідає вашим потребам, Windows Forms також

підтримує створення власних налаштовуваних елементів керування за допомогою класу UserControl.

Windows Forms має розширені елементи керування інтерфейсом, які імітують функції таких висококласних програм, як Microsoft Office. Використовуючи елементи керування ToolStrip та MenuStrip, ви можете створювати панелі інструментів та меню, які містять текст та зображення, відображати підменю та розміщувати інші елементи керування, такі як текстові поля та комбіновані поля.

За допомогою конструктора перетягування Windows Forms у Visual Studio ви можете легко створювати програми Windows Forms. Просто виберіть елементи керування курсором і розмістіть їх у потрібному місці у формі. Дизайнер надає такі інструменти, як лінії сітки та лінії прив’язки, щоб усунути клопоти з вирівнювання елементів управління. За допомогою елементів керування FlowLayoutPanel, TableLayoutPanel та SplitContainer можна створити розширені макети форм за менший час.

Нарешті, якщо вам потрібно створити власні власні елементи інтерфейсу, простір імен System.Drawing містить великий вибір класів для візуалізації ліній, кіл та інших фігур безпосередньо у формі.

Існує дві реалізації Windows Forms:

- Реалізація з відкритим кодом, розміщена на GitHub. Ця версія працює на .NET 5 та .NET Core 3.1. Візуальний конструктор Windows Forms вимагає, як мінімум, попереднього перегляду Visual Studio 2019 версії 16.8.
- Реалізація .NET Framework 4, яка підтримується Visual Studio 2019 та Visual Studio 2017. .NET Framework 4 - це лише версія Windows .NET і вважається компонентом операційної системи Windows. Ця версія Windows Forms поширюється з .NET Framework.

Розгляд функціоналу показав, що доступний інструментарій та легкість освоєння візуальної середи забезпечать робочий інтерфейс для потреб

вирішення даної задачі. Для виконання проекту буде використана версія, яка підтримується на базі програмного забезпечення для написання та тестування коду Microsoft Visual Studio, яка має деякі переваги, такі як пряма інтеграція з вище вказаною середою для програмування.

РОЗДІЛ 3. МАТЕМАТИЧНА МОДЕЛЬ РОБОТИ СИСТЕМИ

3.1 Функції активації та їх характеристики

Функція активації, яку прийнято позначати як a (x), визначає вихідне значення нейрона в залежності від результату зваженої суми входів і порогового значення. Розглянемо нейрон, у якого зважена сума входів (3.1):

$$z = \Sigma iwixi + b, \quad (3.1)$$

де wi і xi - вага і вхідний значення i-ого входу, а b - зсув. Отриманий результат передається в функцію активації, яка вирішує розглядати цей нейрон як активований, або його можна ігнорувати. У наш час існує широка класифікація різних функцій активації.

Ступінчаста функція (англ. Binary step function) є пороговою функцією активації. Тобто якщо z більше або менше деякого значення, то нейрон стає активованим. Така функція відмінно працює для бінарної класифікації. Але вона не працює, коли для класифікації потрібна більша кількість нейронів і кількість можливих класів більше двох, тобто руйнується згадана на початку умова. Цю функцію прийнято застосовувати у випадку, коли змінна виходу має тип даних булеан.

Лінійна функція являє собою пряму лінію, тобто має вигляд на кшталт цієї формули (3.2):

$$a\,(x) = \Sigma icixi, \quad (3.2)$$

а це значить, що результат цієї функції активації пропорційний переданому аргументу. На відміну від попередньої функції, вона дозволяє отримати діапазон значень на виході, а не тільки бінарні 0 і 1, що вирішує проблему класифікації з великою кількістю класів. Але у лінійної функції є дві основні проблеми:

- Неможливість використання методу зворотного поширення помилки. Так як в основі цього методу навчання лежить градієнтний спуск, а для того

щоб його знайти, потрібно взяти похідну, яка для даної функції активації - константа і не залежить від вхідних значень. Тобто при оновленні ваг не можна сказати чи покращується емпіричний ризик на поточному кроці чи ні. Розглянемо нейронну мережу з декількома шарами з цією функцією активації. Так як для кожного шару вихідне значення лінійно, то вони утворюють лінійну комбінацію, результатом якої є лінійна функція. Тобто фінальна функція активації на останньому шарі залежить тільки від вхідних значень на першому шарі. А це означає, що будь-яку кількість шарів може бути замінено всього одним шаром, і, отже, немає сенсу створювати багатошарову мережу.

- Головна відмінність лінійної функції від інших в тому, що її область визначення не обмежена: (-∞; + ∞).

Отже, її потрібно використовувати, коли вихідне значення нейрона має ∈R, а не обмеженому інтервалу.

Сігмоїдальна функція є гладкою монотонно зростаючою нелінійної функцією (3.3):

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.3}$$

Так як ця функція нелінійна, то її можна використовувати в нейронних мережах з великою кількістю шарів, а також навчати ці мережі методом зворотного поширення помилки. Сигмоїда обмежена двома горизонтальними асимптотами y = 1 і y = 0, що дає нормалізацію вихідного значення кожного нейрона. Крім того, для сигмоїдної функції характерний гладкий градієнт, який запобігає "стрибки" при підрахунку вихідного значення. Крім того, у цій функції є ще одна перевага, для значень x> 2 і x <-2, y "притискається" до однієї з асимптот, що дозволяє робити чіткі прогнози класів. Незважаючи на безліч сильних сторін сигмоїдної функції, у неї є значний недолік. Похідна такої функції вкрай мала у всіх точках, крім порівняно невеликого проміжку. Це сильно ускладнює процес поліпшення ваг за допомогою градієнтного

спуску. Більш того, ця проблема посилюється в разі, якщо модель містить багато шарів. Дана проблема називається проблемою зникаючого градієнта. Що стосується використання у сигмоїді функції, то її перевага над іншими - в нормалізації вихідного значення. Іноді, це буває вкрай необхідно. Наприклад, коли підсумкове значення шару повинно представляти ймовірність випадкової величини. Крім того, цю функцію зручно застосовувати при вирішенні задачі класифікації, завдяки властивості "притискання" до асимптотам.

Функція гіперболічного тангенса має вигляд (3.4):

$$tanh\,(z) = \frac{2}{1 + e^{-2z}}. \tag{3.4}$$

Ця функція є скоригованою версією сигмоїдної функції, тобто вона зберігає ті ж переваги і недоліки, але вже для діапазону значень (-1; 1). Зазвичай, вона виконує роль сигмоїди у випадках, коли немає необхідності в нормалізації. Це відбувається через те, що область визначення даної функції активації центрована щодо нуля, що знімає обмеження при підрахунку градієнта для переміщення в певному напрямку. Крім того, похідна гіперболічного тангенса значно вище поблизу нуля, даючи велику амплітуду градієнтному спуску, а отже і більш швидку збіжність.

Функція ReLU - Rectified Linear Unit - це найбільш часто використовувана функція активації при глибокому навчанні. Ця функція повертає 0, якщо приймає негативний аргумент, в разі ж позитивний аргумент, функція повертає саме число. Тобто вона може бути записана як показано в формулі (3.5):

$$f\,(z) = max\,(0,z). \tag{3.5}$$

На перший погляд може здатися, що вона лінійна і має ті ж проблеми що і лінійна функція, але це не так і її можна використовувати в нейронних мережах з великою кількістю шарів.

Функція ReLU володіє декількома перевагами над сигмоїдою і гіперболічним тангенсом:

- Дуже швидко і просто знаходиться похідна. Для від'ємних значень - 0, для позитивних - 1.
- Розрідженість активації. У мережах з дуже великою кількістю нейронів використання сігмоїдної функції або гіперболічного тангенса як активаційний функції тягне активацію багатьох нейронів, що може позначитися на продуктивності навчання моделі [14].

Якщо ж використовувати ReLU, то кількість включаються нейронів стане менше, в силу характеристик функції, і сама мережа стане легше. У цієї функції є один недолік, який має назву проблемою вмираючого ReLU. Так як частина похідної функції дорівнює нулю, то і градієнт для неї буде нульовим, а то це означає, що ваги не будуть змінюватися під час спуску і нейронна мережа перестане вчитися. Функцію активації ReLU слід використовувати, якщо немає особливих вимог для вихідного значення нейрона, на кшталт необмеженої області визначення. Але якщо після навчання моделі результати вийшли не оптимальні, то варто перейти до інших функцій, які можуть дати кращий результат.

Однією з проблем стандартного ReLU є затухаючий, а саме нульовий, градієнт при негативних значеннях. При використанні звичайного ReLU деякі нейрони вмирають, а відстежити вмирання нейронів не просто. Щоб вирішити цю проблему іноді використовується підхід ReLU з «витоком» (leak) - графік функції активації на негативних значеннях ніяк не горизонтальну пряму, а похилу, з маленьким кутовим коефіцієнтом (близько 0,01). Тобто вона може бути записана як показано в формулі (3.6):

$$f(x)\begin{cases}0.01x, & \text{якщо } x < 0\\ x, & \text{у інших випадках}\end{cases} \tag{3.6}$$

Таке невелике від'ємне значення допомагає домогтися ненульового градієнта при негативних значеннях. Однак, функція Leaky ReLU має деякі

недоліки: Складніше вважати похідну, в порівнянні зі стандартним підходом (так як значення вже не дорівнюють нулю), що уповільнює роботу кожної епохи. Кутовий коефіцієнт прямої також є гіперпараметром, який треба налаштовувати. На практиці, результат не завжди сильно поліпшується щодо ReLU. Варто відзначити, що крім проблеми вмираючих нейронів, у ReLU є і інша - проблема затухання. При занадто великій кількості шарів градієнт буде приймати дуже маленьке значення, поступово зменшуючись до нуля. У наслідку цього змінна суми або вагів досягає таких значень, що більшість типів даних в сучасних мовах програмування не в змозі коректно відобразити настільки мале значення. Через це нейронна мережа працює нестабільно і неправильно. Leaky ReLU (LReLU) вирішує першу проблему, але в по-справжньому глибоких мережах проблема загасання градієнта все ще зустрічається і при використанні цього підходу. На практиці LReLU використовується не так часто. Практичний результат використання LReLU замість ReLU відрізняється не занадто сильно.

Однак в разі використання Leaky потрібно додатково налаштовувати гіперпараметр (рівень нахилу при негативних значеннях), що вимагає певних зусиль. Ще однією проблемою є те, що результат LReLU не завжди краще ніж при використанні звичайного ReLU, тому найчастіше такий підхід використовують як альтернатива. Досить часто на практиці використовується PReLU (Parametric ReLU), який дозволяє домогтися більш значних поліпшень в порівнянні з ReLU і LReLU. Також, в разі параметричної модифікації ReLU, кут нахилу не є гіперпараметром і налаштовується нейромережею.

Виходячи з вище сказаного можна зробити висновки для того, що потрібно для реалізації цього модулю підійде сигмоїдна функція активації через те, що вона досить проста в реалізації та в неї нема серйозних проблем, які виникають при її використанні [15].

3.2 Зв’язки між нейронами. Вхідний та вихідний шари

У сфері вивчення нейромережевих технологій досить важливу роль займають шляхи введення та виведення інформації. Як було показано вище, для початку роботи системи необхідно буди визначити спосіб, яким буде проходити обмін та прохід інформації.

Зв’язки між вище згаданими нейронами будуються через проходження значень через кожен окремий вузол, вибудовуючи дерева проходу. Система приймає до уваги усі значення, подані на вихід, та робить висновки залежно від того, як побудована нейронна мережа. Чим більше нейронів пройшов сигнал до виходу, тим більша точність визначеного результату. Логіка цього процесу працює таким чином: чим більша кількість факторів, яку може змінити система, тим меншою ставатиме похибка. У випадку PNN вагів як таких не існує, тож система може видати результат досить швидко та точно. Також треба брати до уваги кількість вхідних ознак, так як чим більше вхідних ознак, тим більша вірогідність того, що вхідні вектори будуть відрізнятися достатньо, аби уникнути хибних позитивів.

У нашому випадку кількість вхідних параметрів досить висока та ознаки атаки та нормального трафіку відрізняються достатньо разюче, щоб уникнути грубих помилок під час роботи мережі для визначення результатів.

3.3 Особливості роботи PNN

Однією з найголовніших особливостей принципу роботи ймовірнісної мережі є те, що вона не має чітко виділеного етапу навчання, як наприклад багатошаровий персептрон, який вміщує в собі велику кількість вагів. Розпізнавання образів відбувається за допомогою прорахунку належності кожного вхідного вектора до заданих класів. Цей процес відбувається під час проходу значень через функцію активації, яка простою мовою порівнює значення та видає більші значення при більшому сходженні вхідного

значення з шуканим. Цей процес відбувається за так званою логістичною функцією [16].

PNN належить до так званих нейронних мереж з прямим зв’язком. Це означає, що прохід сигналу йде в одному напрямку та ніколи не звертається до попередніх етапів, тобто не утворює коло. З цього витікає, що для безперервної роботи цієї системи потрібен буде певний механізм перезапуску процесу обробки даних [17]. Під час цього процесу сигнал проходить через вище згадані шари роботи: вхідний, образів, сумування та вихідний.

На етапі шару образів відбувається прохід вхідних даних та модифікація вхідних даних функцією активації. В залежності від схожості даних на очікувані, або точніше належності певному класу, на етапі сумації ми порівнюємо, до якого класу вихідні значення з попереднього шару ми отримаємо. В результаті з останнього нейрона в вихідному шарі ми отримуємо вихід з шару сумації, який містить у собі інформацію про клас, якому належить вхідний вектор ознак [18].

Серед ознак роботи ймовірнісних нейронних мереж виділяють такі переваги та недоліки. Перевагами ймовірнісних нейронних мереж є [19]:

- Мережі PNN набагато швидші, ніж багатошарові мережі перцептронів.
- Мережі PNN можуть бути точнішими, ніж багатошарові мережі персептронів.
- Мережі PNN відносно нечутливі до хибних даних.
- Мережі PNN генерують точні прогнозовані цільові показники ймовірності.

Серед слабких місць PNN:

- Мережі PNN можуть потребувати більше місця для роботи.
- Мережі PNN повільніше класифікують нові групи, через те, що процес диференціації результатів відбувається як частина процесу роботи системи, а не як інтерпретація вихідних даних

Після аналізу вище наведених даних про задачі, структуру і шляхи вішення поставлених задач можна починати розробку тестового модуля.

РОЗДІЛ 4. РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЮ

4.1 Інтерфейс програмного забезпечення

Для початку роботи над рішенням в плані коду потрібно проробити функціональну сторону рішення з боку користувача. Потрібно створити користувацький інтерфейс для простоти інтерпретації даних.

Візуальна оболонка програми матиме включати в себе способи введення інформації, вікно для таблиці та статистичну інформацію після проведення обробки даних для базового рівня користування. Під час процесу розробки вікна рішення були враховані потреби користувача. У результаті був розроблений подальший інтерфейс (рис. 4.1).

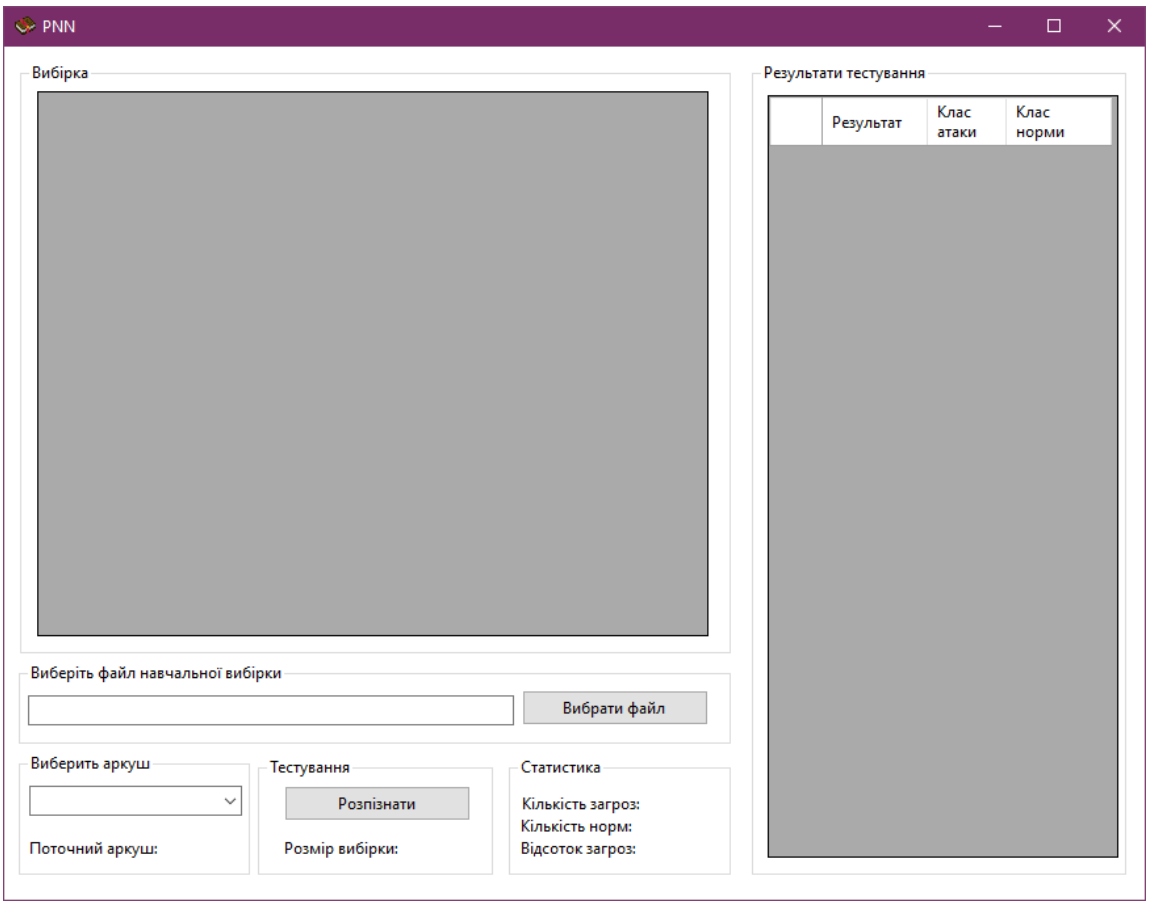


Рисунок 4.1 Користувацький інтерфейс програми

Отриманий користувацький інтерфейс вміщує в себе такі елементи:

- Вікно “вибірка”, в якому відображатимуться дані таблиці, які були завантажені з поля вибору файлу навчальної вибірки.
- Вікно вибору файлу навчальної вибірки, в якому знаходиться поле для вводу шляху до оброблюваного файлу та клавіша для швидкого доступу до таблиці за допомогою вбудованих засобів файлової системи операційної системи.
- Вікно вибору аркушу, яке містить у собі викидне меню з аркушами з завантаженого з вікна вибору файлу навчальної вибірки таблиці, що дозволяє змінювати сторінки одного файлу не виходячи з програми. Також містить поле, яке показує назву аркушу.
- Вікно “тестування” містить кнопку для початку тестування та поле з виводом кількості вхідних векторів, які були подані на початок нейронної мережі.
- Вікно “статистика” містить у собі поля кількості небезпечних з’єднань та нормального трафіку, а також поле для показу процентного відношення загрозливих з’єднань.
- Вікно з результатами тестування, в якому відображаються результати тестування, який виводить чи відбулась атака та виводить нормалізовану відносно отриманих даних активність класів атаки та нормального трафіку для порівняння.

Для початку використання програми користувач має взяти нормалізовану таблицю за форматами KDD-99 та подати на вхід нейронної мережі аркуш за яким проводитиметься навчання. У результаті буде завантажена таблиця в вікні вибірки. Після чого користувач повинен натиснути на кнопку розпізнавання та отримати вікно з результатами проходу даних через мережу.

4.2 Розробка програмного продукту

У процесі розробки програмного продукту були створені елементи управління, за допомогою яких буде налаштована робота програмного забезпечення. У результаті були створені такі елементи системи (рис. 4.2).

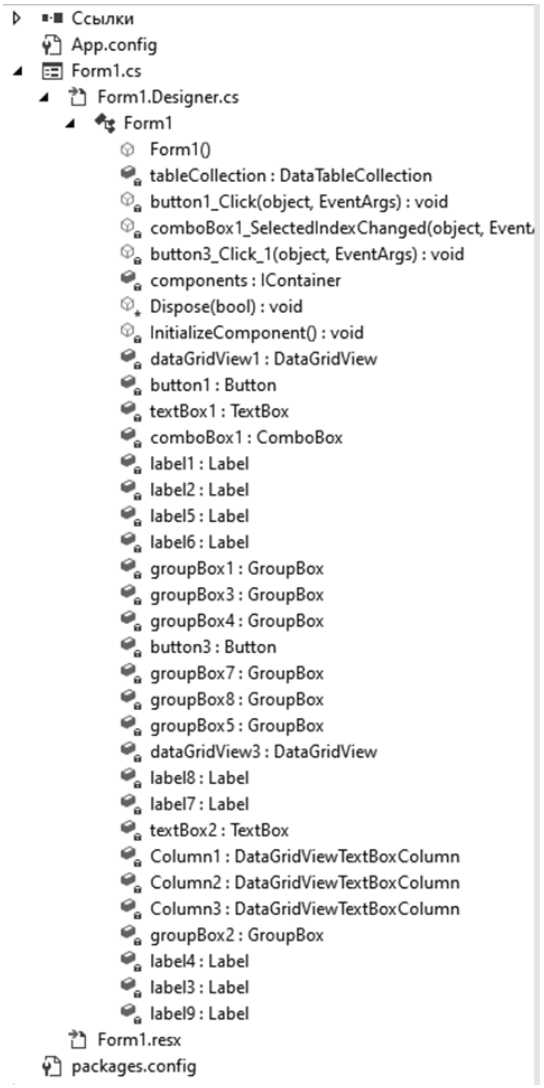


Рисунок 4.2 Елементи програмного інтерфейсу

В інтерфейсі програмного продукту були використані вбудовані в Windows Forms елементи DataGridView. Ці елементи здатні форматувати дані у вигляді таблиці. Вивід навчальної вибірки і статистики був побудований на базі бібліотеки, ExcelDataReader, яка містить інструменти, здатні працювати з таблицями зовнішнього формату Excel [20]. За допомогою цього класу

ExcelReaderFactory в першу таблицю виводяться дані з завантаженої вибірки. Подальші маніпуляції відбуватимуться на базі саме цього елементу інтерфейсу, радше ніж самою таблицею. Це було зроблено для швидкої роботи з файлами, так як в реальній системі потрібно буде всього лише завантажити навчальну вибірку для коректної роботи системи. Очікується, що такий продукт буде отримувати нову вибірку через деякі проміжки часу, щоб працювати згідно з сучасними тенденціями в сфері кібербезпеки, тому після натискання кнопки вибору файлу вибірка зберігається в внутрішній пам’яті додатку.

Наступним завданням постає реалізація функції активації, яка буде забезпечувати роботу процесу обробки даних. У даному випадку була вибрана логістична функція. Для її налаштування було створено функцію, яка буде поштучно приймати на вхід дані з вибірки та диференціювати вихідні дані за допомогою належності певному класу, про що буде свідчити сума у шарі сумації (рис. 4.3).

```
step = Math.Pow(-up, 2) / Math.Pow(delta, 2);
sum += Math.Exp(step);
```

Рисунок 4.3 Реалізація функції активації

Також, був налаштований цикл з розміром кроку з ширину таблиці значень для того, щоб приймати результати вибірки незалежно від кількості елементів в одному ряді. Тобто, через функцію активації проходитимуть усі значення, які подаються на вхід, тож створений код логічно відповідає схемі ймовірнісної нейронної мережі для цього завдання.

Після проходу значень через шар образів отримані значення вносяться до масиву, де потім потрапляють на реалізацію шару сумації (рис. 4.4). Саме тут відбуватиметься перевірка того, чи є наведений трафік загрозливим чи ні.

```
double h = 0;
A = (y[0] + y[1]) / 2;
for (int i = 2; i < number2; i++)
{
    h += y[i];
}
B = h / number2 - 2;
```

Рисунок 4.4 Реалізація шару сумації

В залежності від того, яке значення буде більше, А чи В, на виході буде показана інформація в зрозумілому для оператора вигляді: якщо А то атаки не виявлено, та якщо В – атака відбулась (рис 4.5).

```
if (A <= B)
{
    dataGridView3.Rows.Add();
    dataGridView3[0, a].Value = "Виявлено атаку";
    numA++;
}
else
{
    numB++;
    dataGridView3.Rows.Add();
    dataGridView3[0, a].Value = "Атаки не виявлено";
}
```

Рисунок 4.5 Вихід нейронної мережі

Ця інформація проходить через цикл для виводу значень для кожного рядку. Після цього у таблиці результатів виводяться нормалізовані значення з вихідного шару та класифікатор атаки. Наприкінці цього процесу потрібно створити статистику для загальної оцінки (рис 4.6).

```
label3.Text = "Кількість загроз: " + Convert.ToString(numA);
label4.Text = "Кількість норми: " + Convert.ToString(numB);
int en = dataGridView1.Rows.Count;
double danger = 100 * Convert.ToDouble(numA) / Convert.ToDouble(en);
label9.Text = "Відсоток загроз: " + Convert.ToString(Math.Round(danger, 2)) + "%";
```

Рисунок 4.6 Вивід статистичних даних

У результаті оператор матиме достатньо ресурсів для оцінки рівня загроз та перевірки правильності роботи системи. Вихідний код форми представлений в додатку А.

4.3 Тестування розробленого модулю

Для перевірки справності роботи системи буде проведена перевірка результатів тестового модулю. Перевірка буде відбуватися в три етапи, які будуть відрізнятися кількістю введених даних з навчальної вибірки. Для реалістичної постановки задачі буде використана велика кількість ознак нормального трафіку, в той час як атак буде абсолютна меншість. Такі умови створюються для того, щоб показати реальний стан речей – під час роботи мережі більша частина трафіку все ще не буде представляти ніяких загроз.

Для початку тестування потрібно буде вибрати файл з тестовою вибіркою. В інтерфейсі був реалізований спосіб вводу інформації через провідник системи, тому можна знайти потрібний файл за допомогою файлового менеджера операційної системи. Нормалізована вибірка подається на вхід системи за допомогою поля вибору шляху до файлу.

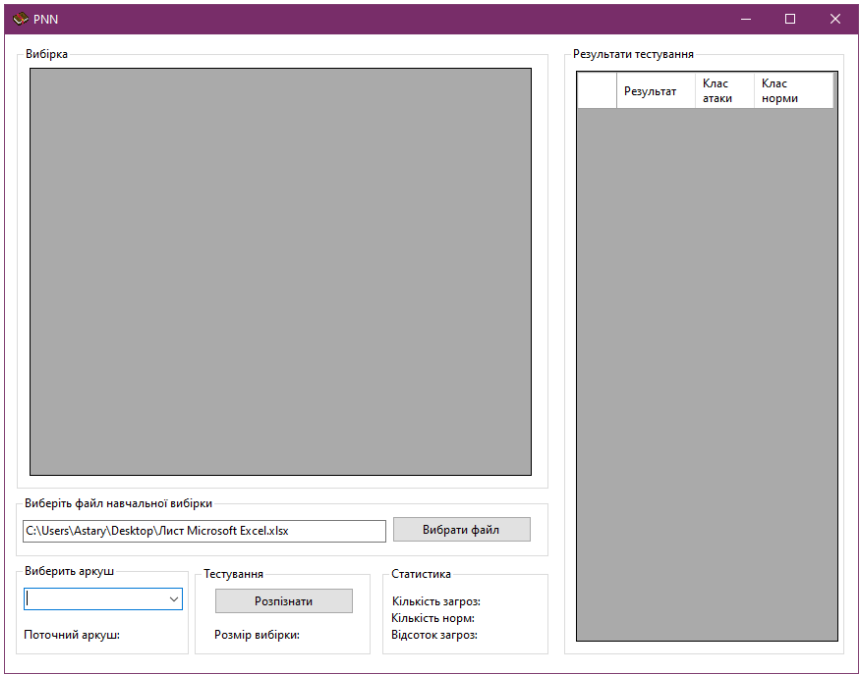


Рисунок 4.7 Вибір файлу

Після вибору потрібного файлу потрібно виділити потрібний аркуш з таблиці. Так як формат таблиць Excel може містити в собі декілька різних аркушів, потрібно зазначити з якого саме аркуша потрібно зчитувати інформацію. Для цього потрібно скористатися полем вибору аркуша (рис. 4.8). Для першого досліду буде використана вибірка на 300 елементів.

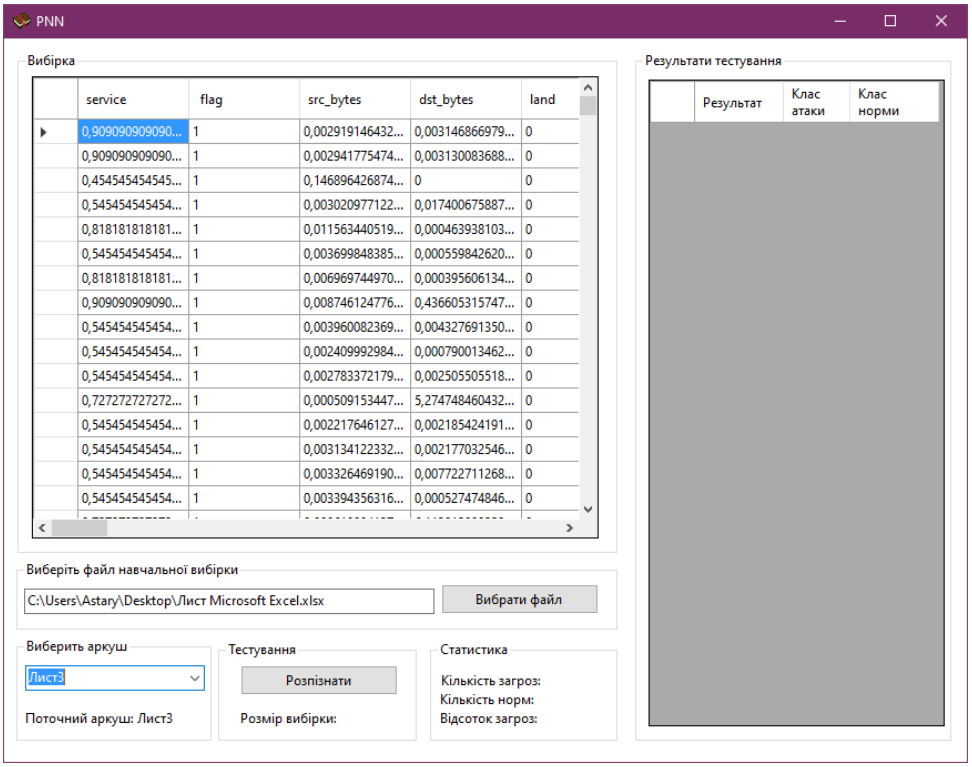


Рисунок 4.8 Вибір аркуша для зчитування

Після вибору аркуша для навчання можна розпочинати розпізнавання. Без заданих вхідних даних користувач побачить повідомлення про те, що не були задані всі необхідні параметри для того, щоб нейронна мережа змогла почати роботу (рис. 4.9). Також це вікно з’явиться у випадку, якщо таблиця або вибраний її аркуш був порожнім або не мав даних для зчитування. У результаті користувач знатиме, чому система не працює. Після специфікації користувачем потрібної вибірки натискається кнопка розпізнавання для того, щоб провести тестування.

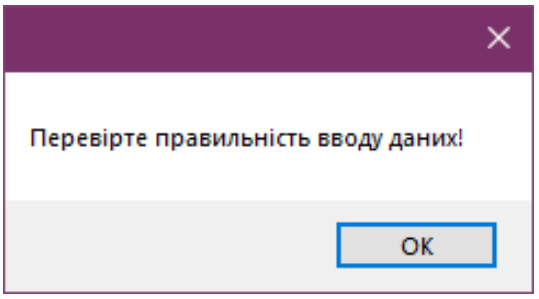


Рисунок 4.9 Вікно помилки зчитування

Після введення першого аркуша та натискання кнопки розпізнавання система прочитала 500 рядків, як і було введено в документі. Програма коректно розпізнала образи та почала роботу. Результати тестування вивелися в окремій таблиці справа (рис 4.10).

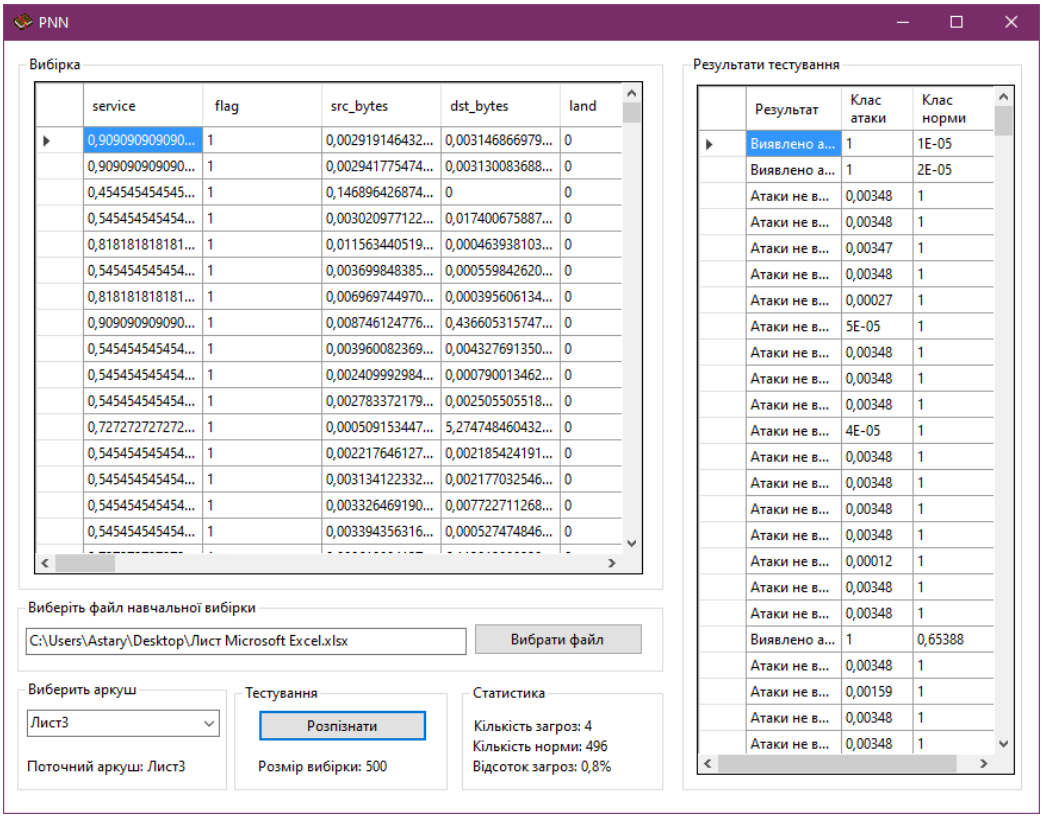


Рисунок 4.10 Результат першого тестування

Можна побачити, що система знайшла чотири загрози, в той час як насправді ознаки загрози описували лише два перших рядки. Це пов’язано з

тим, що навіть нормальний трафік може видавати так звані “хибні позитиви”. Це означає, що ознаки нормального підключення були сприйняті системою як загроза. Саме для цього і існує окремий вивід результатів ознак класів.

Для справності роботи системи необхідно виключення таких випадків. В той час, як перерване нормальне з’єднання можна провести знову повторним запитом з іншими ознаками, зловмисне підключення завжди матиме ознаки аномальної активності. Завдяки цьому виключається загроза, так як зловмисні запити будуть блокуватися кожен раз. Також, можна побачити, що різниця класу атаки до класу нормального трафіку у перших двох рядках, які описують ознаки атаки, екстремально висока, в той час як у випадку хибних позитивів різниця є не такою помітною. Тобто можна легко інтерпретувати вивід програми та робити поправки в навчальній вибірці, щоб досягти оптимального результату.

У той самий час проявляється одна із слабких частин роботи ймовірнісної нейронної мережі – погана скаляція відносно кількості вхідних векторів. Як було згадано вище, це зумовлено тим, що для того, щоб надати якісний результат і забезпечити відносну простоту реалізації, система складається з навчальної вибірки, тобто не може працювати без якогось вводу, від якого треба відштовхуватися. PNN досить гарно показує себе для створення швидких та якісних систем для нескладних завдань.

Для вирішення цієї проблеми мною були запропоновані такі заходи для підтримки працездатності системи:

- Оперативне оновлення вибірки релевантними даними
- Виключення повторів з еталонної вибірки
- Використання навчальної вибірки як словника для нових рядків. Тобто коли потрібно буде проаналізувати трафік, його ознаки, взяті з мережевого датчика або програмного забезпечення для роботи з пакетами, таким, як наприклад Wireshark, будуть доповнені до існуючої

вибірки та буде проведене розпізнавання. Після того, як з'єднання буде розірване, рядок буде виключений з таблиці.

За допомогою цих заходів можна буде легко інтерпретувати вхідні сигнали з мінімальними затримками та доступом до оперативної інформації без того, щоб використовувати більш громіздкі види нейронних мереж.

Тестування було продовжене зі зменшеною кількістю вхідних векторів, залишаючи таку ж кількість ознак загрозливого з'єднання (рис 4.11).

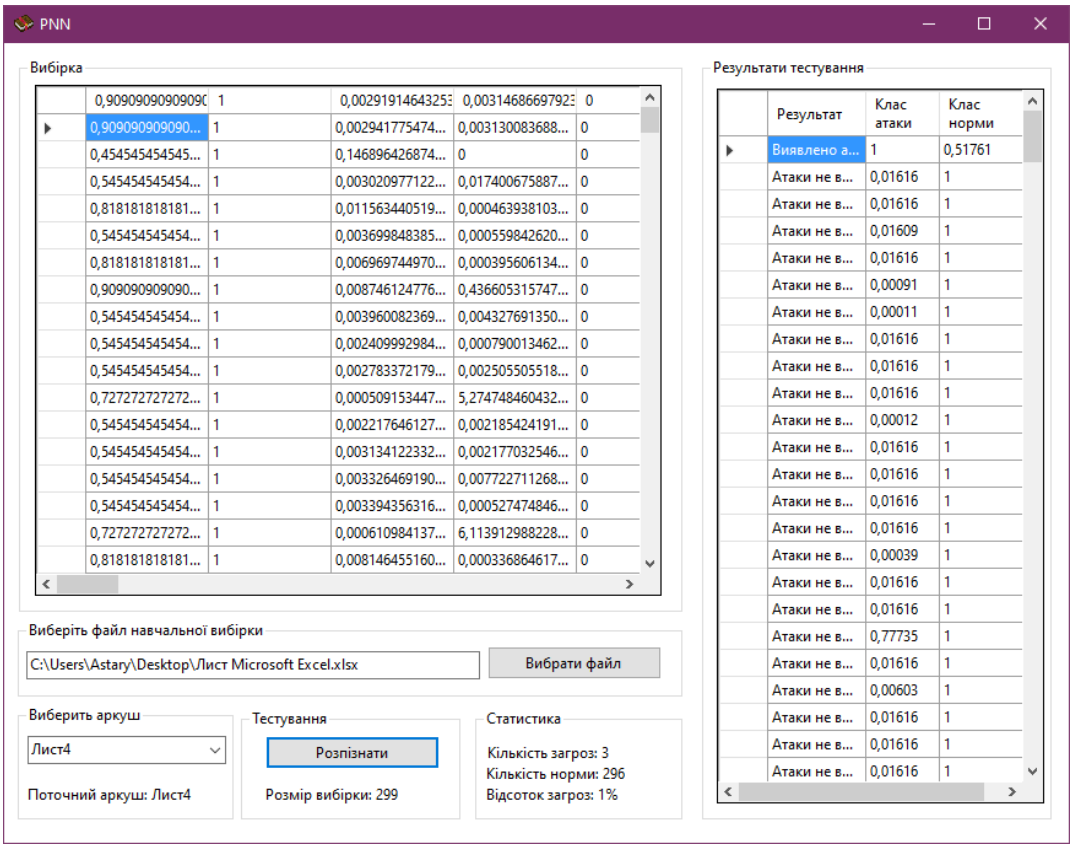


Рисунок 4.11 Результат другого тестування

У результаті були отримані схожі вихідні дані. Час виконання помітно скоротився, а точність оцінки залишилася на такому ж рівні. Жоден рядок з загрозою не був ідентифікований як нормальний трафік, тобто система все ще працює справно і надає захист від цієї загрози.

Для третього експерименту було зменшено кількість вхідних векторів ще, вибірка досягла розміру ста вхідних векторів. Результат показаний на нижче наведеному зображенні (рис 4.12).

В результаті проведення третього випробування було підвищення часу виконання розпізнавання. Спостерігається трохи більший рівень похибки, проте він все ще залишається досить низьким.

Після проведених тестувань можна зробити висновок, що побудоване програмне рішення працює правильно та надає такі результати, які були очікувані від проходу даних через мережу.

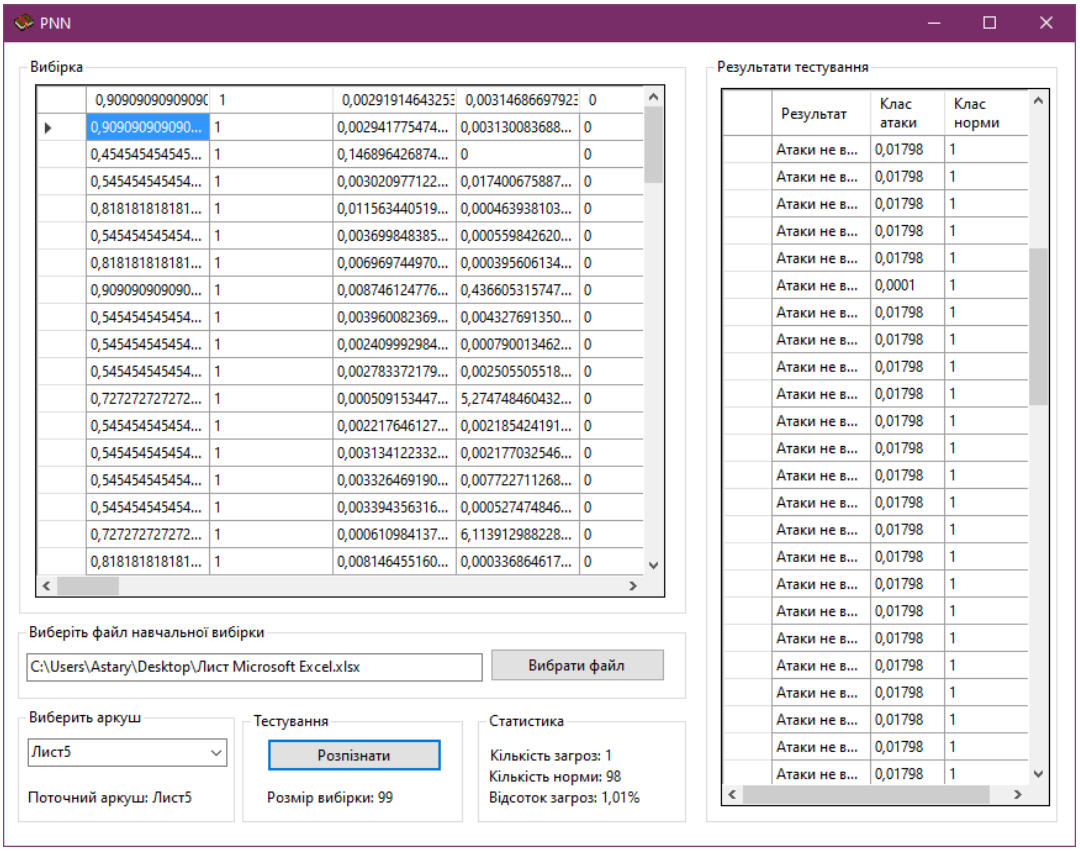


Рисунок 4.12 Результат третього тестування

Програмний продукт показав себе досить точним інструментом для розпізнавання загроз, навіть з високим рівнем так званого шуму, тобто повторення однакових значень, що могло сильно вдарити по точності

конкурентних архітектур, як багатошарового перцептрона, який мав би занадто високий вплив з одного боку класу.

З програмної точки зору, Windows Forms показав себе як досить зрозумілий інструмент для використання для розробки додатків. В результаті використання деяких компонентів були зроблені висновки для роботи рішення та висунуті способи їх вирішення.

4.4 **Аналіз результатів тестування**

Після проведеного тестування, були підбиті підсумки. Було проведено три випробовування, у результаті яких були знайдені деякі тенденції. У результаті першого тестування було, тобто час зростає нелінійно і проблема PNN з роботою з високим обсягом даних показала себе досить яскраво. Проте навіть не зважаючи на це, час для наведених вибірок все ще залишається досить непоганим результатом. Для прикладу, класичні методи пошуку, такі як, наприклад, бінарний пошук, використовують системи порівнянь для того, щоб знайти результат, який буде співпадати з наведеним. Цей процес займатиме більше часу та буде потребувати якнайповнішої репрезентації усіх можливих видів загроз, проте зріст часу буде не таким разючим, як у PNN.

Для представлення даних досліджень були записані дані виконання тестувань, як час, відсоток помилок та обсяг вибірки.

Результати тестування представлені в таблиці 4.1.

Таблиця 4.1. Результати тестування

	Кількість входів, шт.	Відсоток помилок,%	Час оцінювання, с
Тестування №1	500	0.4	3.25
Тестування №2	300	0.3	1.76
Тестування №3	100	1	0.4

Проте ймовірнісна нейронна мережа не опирається на великий список загроз, а аналізує ознаки, незважаючи на їх типи. Це забезпечує їй високу адаптованість у випадках, коли система надає схожі на нормальний трафік аномальні сигнали. У той час, як класична система пошуку ідентифікує трафік як аномальний, тобто той, якого нема в базі даних, ймовірнісна нейронна мережа здатна показати вірогідність того, що трафік належить до одного з наведених класів. Це робить систему легкою в обслуговуванні, так як вона не потребує знати все, щоб робити правильні висновки в більшості випадків.

З наведених даних можна прослідкувати тенденцію швидкого зросту часу виконання обчислювання. Відсоток помилок визначався відношенням неправильно ідентифікованого трафіку до кількості усіх інших вхідних векторів. Також, можна побачити, що навіть під час використання зменшеної кількості векторів, якість досліджень все ще залишалася досить високою для системи.

В процесі дослідження було встановлено, що найбільший вплив на час виконання має створення масиву даних для виводу результатів для таблиці виводу даних. Елементи форми як daraGridView використовують більший обсяг пам’яті з кожним новим рядком, що може сповільнити роботу системи. Час обробки без виводу інформації графічно падає більш ніж в п’ять разів, так як обмеженнями стають вже мова програмування та потужність обладнання, а не інструменти форми.

Для аналізу кореляції часу обчислення з точністю був побудований графік на базі таблиці тестування (рис 4.13).

Як можна побачити на графіку, через деякий час кількість елементів вибірки не дає ніяких переваг для обчислення. Тож, можна зробити висновок, що навіть з відносно невеликою таблицею значень система теоретично здатна надавати більш менш точні результати.

Після аналізу даних можна сказати, що нейронна мережа впоралася з завданням та надала гарні результати не зважаючи на велику кількість представників одного класу.

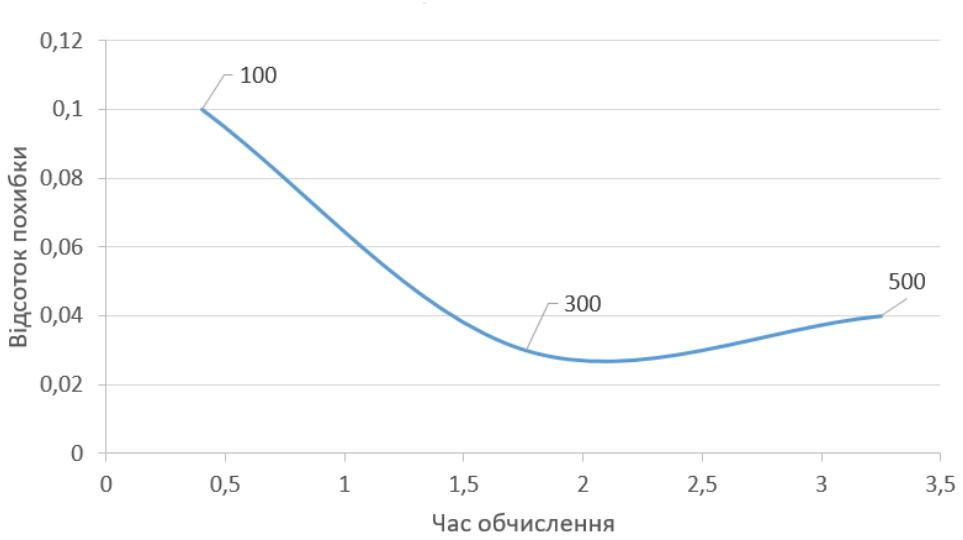


Рисунок 4.13 Графік похибки відносно часу

Для вирішення питань оптимізації додатку потрібно буде заздалегідь оновлювати таблиці та зменшити інформацію для подачі, виводячи лише нечіткі результати або результати атак. За допомогою цих методів програма здатна буде працювати оптимально.

У результаті з тестового модулю буде створена швидка альтернатива класичним методам пошуку.

ВИСНОВОК

В результаті проведеного дослідження було виявлено, що проблематика задачі є досить актуальною через високі рівні зросту кількості атак за останні три роки. Були проведені:

1. Аналіз предметної області;
2. Оцінка тенденцій рівню загроз;
3. Розробка алгоритмів роботи системи;
4. Обробка даних вибірки;
5. Розробка та тестування програмного модулю.

Програмна розробка працює досить швидко та не володіє складною структурою, що дозволяє використовувати її для невеликих мереж для фільтрування вхідних сигналів та перевірки правильності таблиць даних. Система володіє досить високими показниками точності оцінки даних, завдяки чому можна легко досліджувати тенденції нормального трафіку відносно ознак загроз.

Розроблений модуль володіє підтримкою вводу даних з легких для роботи форматів даних Excel та володіє простим для розуміння користувацьким інтерфейсом, який надає всю інформацію, яка класифікуватиме дані.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Cisco 2025 Cybersecurity Readiness Index. URL: https://newsroom.cisco.com/c/dam/r/newsroom/en/us/interactive/cybersecurity-readiness-index/2025/documents/2025_Cisco_Cybersecurity_Readiness_Index.pdf.
2. Executive summary cisco 2024 Global Threat Analysis Report. URL: https://newsroom.cisco.com/c/dam/r/newsroom/en/us/interactive/cybersecurity-readiness-index/documents/Cisco_Cybersecurity_Readiness_Index_FINAL.pdf .
3. PandaLabs 2024 annual report. URL: https://disclosure.bursamalaysia.com/FileAccess/apbursaweb/download?id=241394&name=EA_DS_ATTACHMENTS
4. СУББОТІН, Сергій Олександрович. Нейронні мережі: теорія та практика. 2020. URL: https://eir.zp.edu.ua/server/api/core/bitstreams/2abb401b-9ee6-4afc-a92a-2de5c332d12f/content.
5. ДОБРОВСЬКА, Людмила Миколаївна; ДОБРОВСЬКА, Ірина Арбіківна. Теорія та практика нейронних мереж. 2015. URL: https://ela.kpi.ua/server/api/core/bitstreams/c4dbb86a-d617-44df-9b7a-5b6e9f91cebf/content.
6. Vacca J. R. Computer and information security handbook. Amsterdam; Boston : Elsevier, 2009.
7. Specht D. F. Generation of Polynomial Discriminant Functions for Pattern Recognition. IEEE Transactions on Electronic Computers. 1967. EC-16, № 3. С. 308–319. URL: https://doi.org/10.1109/pgec.1967.264667.
8. What is windows forms - windows forms .NET. Developer tools, technical documentation and coding examples | Microsoft Docs. URL:

https://docs.microsoft.com/enus/dotnet/desktop/winforms/overview/?view=net desktop-5.0#:~:text=Windows%20Forms%20is%20a%20UI,easy%20to%20 build%20desktop%20apps.

9. A detailed analysis of the KDD CUP 99 data set / M. Tavallaee та ін. 2009 IEEE symposium on computational intelligence for security and defense applications (CISDA), м. Ottawa, ON, Canada, 8—10 лип. 2009 р. 2009. URL: https://doi.org/10.1109/cisda.2009.5356528.
10. Özgür A., Erdem H. A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015. PeerJ preprints. 2016. C. 1. URL: https://peerj.com/preprints/1954/.
11. Baestaens D. E. Neural network solutions for trading in financial markets. London : Financial Times, 1994. 245 c.
12. Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives / A. Divekar та ін. 2018 IEEE 3rd international conference on computing, communication and security (ICCCS), м. Kathmandu, 25—27 жовт. 2018 р. 2018. URL: https://doi.org/10.1109/cccs.2018.8586840.
13. Chaudhuri S., Madigan D., Fayyad U. Kdd-99. ACM SIGKDD explorations newsletter. 2000. Т. 1, № 2. С. 49—51. URL: https://doi.org/10.1145/846183.846194.
14. Vinod Nair, Geoffrey E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines // 27th International Conference on International Conference on Machine Learning. — USA: Omnipress, 2010. — С. 807–814. — (ICML'10).
15. MENON, Anil, et al. Characterization of a class of sigmoid functions with applications to neural networks. Neural networks, 1996, 9.5: 819-835. URL: https://tinyurl.com/j4664uvh.
16. Kyurkchiev N., Markov S. Sigmoidal functions: some computational and modelling aspects. Biomath communications. 2015. Т. 1, № 2. URL: https://doi.org/10.11145/j.bmc.2015.03.081.

17. Mitchell T. M. Machine learning. New York : McGraw-Hill, 1997. 414 с.
18. Probabilistic logics and probabilistic networks / J. Williamson та ін. Springer, 2013. 155 с.
19. Probabilistic and General Regression Neural Networks. DTREG. URL: http://www.dtreg.com/pnn.
20. ExcelDataReader / ExcelDataReader. GitHub. URL: https://github.com/ExcelDataReader/ExcelDataReader.
21. POPESCU, Marius-Constantin, et al. Multilayer perceptron and neural networks. WSEAS Transactions on Circuits and Systems, 2009, 8.7: 579-588. URL: https://tinyurl.com/34ahuvah

ДОДАТОК А

ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ КОЛОІОС
04071, Україна, місто Київ, вулиця Оболонська, будинок, 29,
e-mail: hello@aegas.io, Ідентифікаційний код 43015277

Від 01.06.2025

ДОВІДКА

Про впровадження у робочий процес

результатів бакалаврського дослідження

ЧЕРНЕНЬКОГО ОЛЕКСАНДРА ЮРІЙОВИЧА

на тему:

“Розпізнавання кібератак за допомогою нейромереж PNN”

Цим підтверджується, що Черненький Олександр Юрійович, у рамках виконання завдань на підприємстві проявив високий рівень технічної підготовки та ініціативності. Зокрема, ним було успішно впроваджено систему розпізнавання кібератак на основі Probabilistic Neural Network (PNN) у внутрішній процес моніторингу подій інформаційної безпеки.

Розроблена модель дозволила:
значно підвищити точність виявлення аномальної активності у корпоративній мережі;
скоротити час реагування на потенційні інциденти;
автоматизувати частину процесів первинної класифікації загроз.

Інтеграція PNN-моделі була реалізована як пілотний проєкт у тестовому середовищі SIEM-системи компанії, з подальшим успішним перенесенням у продуктивне середовище.

Засновник ТОВ “КОЛОІОС” ІВАНОВА