МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА ПРОГРАМУВАННЯ ТА МАТЕМАТИКИ

Пояснювальна записка

до дипломної роботи
_____ бакалавр _____
(освітньо-кваліфікаційний рівень)

на тему «Методи класифікації і прогнозування інтелектуального аналізу даних на прикладі виявлення мережевих атак»

Виконав: групи ІПЗ-15д
напряму підготовки 121 „ Інженерія програмного забезпечення ”
_____ Пацация І.М.
(підпис)

Керівник,
доцент, к. ф-м.н._____ Ковальов Ю.Г.
(підпис)

Рецензент,
к.т.н., доцент _____ Митрохін С.О.
(підпис)

СЄВЄРОДОНЕЦЬК
2019 року

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет інформаційних технологій та електроніки
Кафедра програмування та математики
Освітньо-кваліфікаційний рівень бакалавр
Спеціальність 121 „ Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМ,
д.т.н., доцент
_____Лифар В.О.
«___» _____ 2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ
ПАЦАЦИЯ ІРАКЛІЙ МУРТАЗОВИЧ

1. Тема роботи Методи класифікації і прогнозування інтелектуального аналізу даних на прикладі виявлення мережевих атак

керівник роботи доцент Ковальов Юрій Григорійович

2. Строк подання студентом роботи 06 червня 2019 р.
3. Вихідні дані до роботи

Об'єктом даної роботи є Методи класифікації і прогнозування інтелектуального аналізу даних на прикладі виявлення мережевих атак.

3.1 Літературні джерела:

Исследование средств обнаружения вторжений. [Электронный ресурс]: http://www.lghost.ru/lib/security/kurs5/theme14_chapter01.htm

Ian H. Witten, Eibe Franf «Data Mining. Practical Machine Learning Tools and Techniques» -2nd ed. – 525p.

А.Е. Лепский, А.Г. Броневич Математические методы распознавания образов: Курс лекций.-Таганрог:Изд-во ТТИ ЮФУ,2009.-155с

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

4.1 Вступ

4.2 Аналіз предметної галузі (огляд літератури), з висвітленням наступних питань:

Методи отримання даних про атаки.

Методи аналізу даних

4.3 Основна частина, в якої висвітлити:

Інформаційна модель об’єкту.
Реалізація проекта
4.4 Висновки
4.5 Перелік використаних джерел
5. Перелік графічного матеріалу немає
6. Дата видачі завдання 02 лютого 2019 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	**Примітка**
1	Одержання завдання на виконання роботи	01.02.19	
2	Укладання і погодження з керівником плану і етапів виконання роботи	20.02.19	
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	1.03.19	
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	10.03.19	
5	Проектування інфологічної моделі задачі що реалізується.	01.04.19	
6	Укладання та тестування програмного продукту	20.04.19	
7	Укладання, оформлення та погодження пояснювальної записки з керівником	15.05.19	
8	Здача готової пояснювальної записки на кафедру	06.06.19	
9	Укладання доповіді і презентації	10.06.19	

Студент _____ Пацация І.М.
(підпис)

Керівник роботи _____ Ковальов Ю.Г.
(підпис)

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студента гр. ІПЗ-15д Пацация І.М.

Науковий керівник

Доцент, к.ф-н.н. _____ Ковальов Ю.Г.

Оцінка наукового керівника: _____

Рецензент к.т.н., доцент каф. ПМ СНУ ім.В.Даля Митрохін С.О.

місто роботи, посада, ПІБ

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:

_____

Голова ЕК

_____ Лифар В.О.

підпис

РЕФЕРАТ

Текст – 100, табл. – 10, додатків – 1, літературних джерел – 50

Мета роботи полягає у порівняльне дослідження ефективності різних алгоритмів і підходів інтелектуального аналізу даних (Data Mining) при виявлення мережевих атак

Для реалізації і тестування системи виявлення DDoS-атак була використана платформа WEKA

Ключові слова: виявлення вторгнень, комп'ютерні мережі, Data Mining.

ЗМІСТ

Перелік умовних позначень, скорочень і термінів

UDP	User Datagram Protocol - протокол призначених для користувача датаграм
TCP	Transmission control protocol - протокол управління передачею
DoS	Denial of Service - відмова в обслуговуванні
IP	Internet Protocol - міжмережевий протокол
NAT	Network Address Translation - перетворення мережних адрес
Мережа	Корпоративна (комп'ютерна) мережу
СОА (В)	Система виявлення атак / вторгнень
ПО	Програмне забезпечення
ОС	Операційна система
БД	База даних
DARPA	Defense Advanced Research Projects Agency – агентство передових оборонних дослідницьких проектів
KDD-99	Knowledge Discovery in Databases – виявлення знань в базах даних
U2R	User to root - неавторизований доступ до ядра (від користувача до ядра)
R2L	Remote to Local - отримання доступу незареєстрованого користувача до комп'ютера з боку віддаленої машини
PROBE	Probing - сканування портів
PCA (МГК)	Principal Components Analysis - Метод Головних Компонент
NIPALS	Non-linear Iterative Partial Least Squares
IDS	Intrusion Detection System - система виявлення вторгнень
DPSO	Discrete Particle Swarm Optimization - дискретна оптимізація рою частинок
SVDF	Support Vector Decision Function - вирішальна функція опорного вектора

ВСТУП

Актуальність досліджень. У сучасному світі комп'ютер та приєднані до віртуального світу комп'ютерні мережі відіграють важливу роль у передачі інформації. Саме тому кіберзлочинці намагаються здійснювати вторгнення у роботу комп'ютерних систем, отримуючи доступ до важливої інформації про конкретні центри або персональну інформацію з наміром застосувати проникнення або тиск, або навіть дизасемблювати системи.

Необхідність збереження інформаційної безпеки та її ефективність у комп'ютерних мережах є досить очевидною, оскільки в технічному плані створення комп'ютерних мереж (апаратного та програмного забезпечення) без слабких місць та збоїв у безпеці практично неможливо, дослідження виявлення вторгнень в комп`ютерних системах має особливе значення. Безпека є важливою галуззю для комп'ютерів і мереж.

Hacker, Cracker та Intruder – це поняття, що характерізують порушників, які проникають в інші системи та ставлять під загрозу їхню безпеку. Запобіжні засоби, включаючи брандмауер, автентифікацію, ідентифікацію та шифрування, зазвичай недостатні для забезпечення безпеки мережі. Важливим захисним механізмом для зниження уразливості комп'ютерних мереж – системи виявлення вторгнень.

Відпрацьовування деяких моделей неправильної поведінки користувачів у системах виявлення вторгнень може зайняти багато часу. З іншого боку, деякі дані в системах виявлення вторгнень створюють перешкоди для дії на виявлення вторгнення, тому останнім часом багато дослідників зосередили на системі виявлення вторгнень на основі методів Data Mining.

Основною проблемою в системі виявлення вторгнення є точність виявлення нових атак. Тому слід використовувати системи, що самонавчаються. З іншого боку, втручання в систему слід діагностувати в

режимі реального часу. Незважаючи на те, що система виявлення вторгнення в офлайн стані також корисна для усунення слабкої безпеки мережі, використання методів інтелектуального аналізу даних може допомогти виявити приховану аномальну діяльність у мережі [1].

Технології Data Mining, завдяки можливості використання автоматизації та поліпшення продуктивності, широко використовуються для виявлення вторгнень. Однак, при використанні цих методів виявлення вторгнень, є ймовірність зіткнутися з проблемами та обмеженнями, зокрема високою складністю, нестабільністю і низькою точністю, якщо була мала кількість атак [2].

Система виявлення вторгнення – це програмне забезпечення, яке здійснює нагляд за діяльністю мережі чи системи для того, щоб знайти руйнівні дії. У зв’язку з цим, інженерами в сфері інформаційної безпеки були розроблені методи виявлення вторгнень, результати роботи котрих можна застосувати для створення бази для статистичної аналізу рівня мережевої безпеки [3]. Система виявлення вторгнення виявляє підозрілі шаблони мережевого трафіку шляхом аналізу інформаційних пакетів трафіку. Дослідження безпеки в комп'ютерних мережах показало, що фактори навколишнього середовища та умови в комп'ютерних мережах є ефективними за якістю виявлення вторгнення, метою якого є дослідження виявлення вторгнення в комп'ютерних мережах за допомогою алгоритмів Data Mining.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД

1.1 Безпека в повсякденному житті та сфері ІТ

Безпека в реальному житті є важливою проблемою для всіх людей. В епоху доісторії безпека означала принципи захисту від виродження або нападів тварин. Інші потреби, такі як безпека проти природних подій або захворювань, не було розглянуто для людей в ті часи. Слідом за цивілізаційним прогресом діапазон безпеки вийшов за рамки актуального часу і залучив ширші виміри, включаючи місце для комфортного та безпечного життя. Більшість процесів, що супроводжують нас кожного дня, можуть мати колізію з небезпекою в більшій або меншій мірі. Наприклад, коли ми йдемо подорожувати з незнайомою людиною або коли ми входимо в незнайоме місто чи країну, ми знаємо, що є певні загрози для нашої фізичної безпеки. Існуючі загрози навколо нас будуть серйозними лише тоді, коли ми знаходимось в незахищеному місці та зустрінемося з особою, яка може зловживати нашою ситуацією. Якщо ми приділятимемо достатню увагу навколишнім загрозам, нам вдасться знайти безпечне місце або знайти рішення. Наприклад, ми можемо супроводжувати людину, яка може направити нас на безпечне місце або взяти таксі. Деякі справи супроводжуються фінансовими або психологічними небезпеками, але вони не мають фізичної небезпеки. Коли ми інвестуємо (у будь-яких формах придбання землі, запасів або навіть діяльності у сфері торгівлі або роботи на ринку), ми очікуємо, що цей капітал буде повернено нам раніше. Як відомо, деякі інвестиції будуть повернуті рано чи пізно, тоді як деякі інвестиції не є такими і призводять до втрат. Наприклад, коли ми спілкуємося з новою людиною, ми сподіваємося, що це нове відношення матиме певні вигоди для нас. Ми також визнаємо ризик, що ця проблема може не мати необхідних

переваг. У деяких галузях доступ до рівня безпеки, який ми очікуємо, неможливий. Наприклад, ми завжди готові мати довге життя і здоровий організм; але статистика тривалості життя показує, що ця проблема не стосується багатьох людей. Деякі з нас гинуть у молодому віці. Деякі з нас, протягом усього життя, стикаються з різними захворюваннями, а деякі з нас живуть довгі роки і проводять здоровий спосіб життя. Ми компенсуємо нашу нездатність визначити долю зі страхуванням, щоб захистити нас від негативних фінансових наслідків, подій та хвороб. Абсолютна безпека в реальному житті або в кіберпросторі неможлива; однак адекватна відповідна безпека практично досяжна в усіх умовах навколишнього середовища.

1.1.1 Безпека як важливий елемент в інформаційних технологіях

Останнім часом, світ став ареною вражаючих змін. Зміни, що є можливими завдяки широкому контролю над інформаційними та комунікаційними технологіями (далі ІКТ), почалися з використання комп'ютера як засобу автоматизації та підвищення продуктивності, і тепер фактично змінили соціальне та індивідуальне життя людини шляхом еволюції його контролю у створенні спільного синергетичного простору. На фвплив на людську долю, що змусило істориків розділити історію людської життя на епохи. Вхід до набутого віртуального простору сучасних інформаційних та комунікаційних технологій також заснував новий період людської цивілізації, так що сучасна революція змінила манер думки, виробництва, споживання, торгівлі, управління, спілкування, війни і навіть релігійності та любові. Ця велика зміна супроводжується багатьма вимогами та наслідками, найважливішими з яких є походження сучасних концепцій безпеки в кіберпросторі. Після того, як відбулася зміна уявлення про комп'ютерні мережі від невеликих груп пристроїв до глобальних кластерів, з урахуванням зростання взаємодій та обміну інформації в комп'ютерних мережах, необхідність захисту електронних систем і створення правових

зобов'язань для користувачів виявилось важливим питанням. Інформаційна безпека у віртуальних просторах завжди розглядалася як одна з основних інфраструктур та вимог у сфері розвитку та всебічного контролю за ІКТ. Хоча абсолютна безпека недоступна ні в реальному світі, ні в кіберпросторі, створення рівню безпеки, який є адекватним і пропорційним потребам відповідно здійснюваних інвестицій, можливо майже у всіх екосистемах. Реальні особи, організації, приватні компанії та державні органи відіграють свою очікувану роль як ефективні групи цієї інтерактивної та синергетичної мережі, а також довіряють різним сторонам, котрі беруть участь в електронній біржі та, ймовірно, ніколи не бачили і не знали один одного. Гарантія безпеки інформаційних ресурсів та інфраструктурного обладнання країни, крім широкого розмаху національної безпеки, є ключем до численних нових комерційних та некомерційних шансів в Інтернеті. Актуальність полягає в тому, що проблема безпеки країни не є відсутністю технологій або продуктів безпеки, а скоріше є виробленням політики, культивуванням та відповідною ефективністю існуючих джерел і їх сумісністю таким чином, щоб вона надавала унікальну потребу в мережевій і цифровій екосистемі країни. У зв'язку з цим важливо зазначити, що архітектура інформаційної безпеки є процесом поточних процесів в архітектурі інформаційних технологій на різних рівнях, включаючи на національному та організаційних рівнях, які в цьому процесі будуть задіяні на необхідних пристроях. Також важливий аспект досвіду інших країн полягає в тому, що інформаційна безпека є багатогалузевою проблемою, і існує потреба в широкомасштабних кооперація х у цьому відношенні. Такі кооперації слід розглядати як на національному, так і на міжнародному рівні. Визначення ролей, обов'язків та відповідальності є одним з важливих моментів, які слід визначити в такому співробітництві.

Як правило, категорія безпеки інформаційних технологій стосується таких питань:

а) комп'ютерна безпека;

б) кібербезпека.

Інформаційна безпека залежить від політики уряду. Цей термін зазвичай використовується за допомогою державних установ та національних політиків у документах, законах та дослідницьких проектах, і це більш-менш є синонімом «Інтернет-безпеки». Обидві фрази стосуються аспектів безпеки мережі та принципів формування політики у таких мережах, як визначення конфіденційності, кібер-злочини, бізнес та глобальні комунікації. Різниця в цих двох термінах полягає не стільки в безпеці комп'ютерів, а в мережах та даних, що в значний мирі змішано з рутинними концепціями безпеки в кіберпросторі [4].

1.1.2 Загрози безпеки мережі

В комп'ютерних мережах будь-які зусилля для знищення, розкриття, зміни, пограбування чи отримання незаконних можливостей доступу називаються атакою. На сьогоднішній день, на відміну від минулого, підготовка атак не вимагає значних знань, а кількість нападів на інформаційну безпеку значно зросла. У комп'ютерній мережі атаки є результатом активних служб, використовуваних протоколів та відкритих портів. Експерти з інформаційної безпеки, зосередившись на трьох вищезазначених принципах, повинні створювати надійну мережу проти атак. Нападки в комп'ютерних мережах поділяються на дві категорії: активні та неактивні. Активні атаки змінюють систему чи мережу, а неактивні атаки проводять пошук для збору інформації з систем. Активні атаки впливають на доступність, інтеграцію та точність даних, тоді як неактивні атаки порушують конфіденційність [5].

Вторгнення – це спричинення низки незаконних дій, які загрожують точності та конфіденційності або доступності джерела. Вторгнення можна розділити на дві категорії: внутрішні та зовнішні. Зовнішні втручання – це ті

типи вторгнень, які здійснюються авторизованими чи неавторизованими користувачами з-за меж мережі до внутрішньої мережі, а внутрішні вторгнення здійснюються допущеними фізичними особами всередині самої мережі, і вони проникають у системи та комп'ютерні мережі через дефекти програмного забезпечення, вилучення паролів, підслуховування мережевого трафіку [6]. Комп'ютерна проникність насправді є описом дій, які ведуть до порушення методів безпеки за допомогою програмної системи. Такі дії здійснюються шляхом доступу до дійсних станів через транзит. Крім того, стан наражається на небезпеку стану, в якому хакери атакують послідовність з дійсних транзитних станів, що призводить до проникності дійсних станів. Крім того, стан проникності насправді є типом всіх непроникних станів, і взагалі проникність може визначати багато проникних станів, серед яких вибрано лише один стан [7]. Пропускна спроможність в комп'ютеризованих системах – це вразливість, яка існує в автоматичних процедурах безпеки системи, пристроях контролю управлінням, пристроях керування інтернетом, які хакери можуть призвести до переривання чутливої продуктивності системи за допомогою цієї вразливості [8]. Способи проникнення до комп'ютерних мереж включають в себе атаки, які стають можливими за рахунок трьох основних чинників: активних служб, користувацьких протоколів та відкритих портів та відповідно до невідомого характеру користувачів у комп'ютеризованих мережах, включаючи Інтернет, усі постачальники послуг стикаються з атаками [9]. Напад на комп'ютерні мережі включає в себе: Web forging атаку на IP-адресу, протокол TCP, шпигунські атаки, вилучення інформації, аплети, файли cookie, прикладні програми, підслуховування.

Web forging. У цьому методі спочатку копіюється версія веб-сайту а потім хакери маніпулюють збереженою версією без будь-яких змін у зовнішності. Хакер завантажує фальсифіковану сторінку і якось привертає увагу користувача до цього, і користувач клацає по небажаному посиланню, а хакер досягає своєї мети [10].

Атака через протокол TCP. У цьому методі хакер відключає постачальників і користувачів Інтернету, представляється постачальникам послуг як користувач і будь-який обмін інформацією відбувається між постачальником послуг і хакером. Перевагою цього способу за допомогою IP-атаки є те, що хакери атакують лише один раз, не підтримують системи безпеки пароля і цей метод є найпоширенішим видом атаки для постачальників послуг Інтернету [4].

Атака на IP-адресу. В цій атаці хакери в напрямку доступу до мережевого плану приєднуються до постачальника послуг IP за допомогою різних методів, потім опиняються між постачальником послуг і користувачами та викрадають інформацію, передаючи фальшиві пакети. Фактично, в цьому методі хакери вводять себе як одержувача для постачальника послуг і як постачальника послуг для користувача, і вони здатні переносити свої власні пакети з правильним номером як проміжним між користувачем і сервером [5].

Шпигунські атаки. В цьому методі хакери через не координовані зв'язки з протоколом TCP замінюють серійний номер відправлених пакетів постачальників послуг серійним номером наступних пакетів та копіюють їх собі. Потім хакери знову надсилають модифікований пакет, в якому вони відправляють свій серійний номер для постачальників послуг, і таким чином, у випадку незнання користувача та постачальника послуг, інформація зменшується або збільшується постачальниками послуг [11].

Підміна IP-адреси. Хакер може представити себе як хост через маршрутизацію. Хакер підміняє адресу постачальника послуг на основі адреси користувача, а потім створює нову адресу для користувача, і тим самим хакер перериває підключення користувача та підключається до постачальника послуг з підробленою адресою користувача [9].

Підробка електронної пошти. Цей онлайн-метод дуже простий та легкий, адже достатньо хакерів мають адекватну інформацію про програмування та надсилання електронної пошти [2].

Аплети. Аплети насправді є кодами Java, які можуть бути небезпечними. Сторінки браузера завантажуються безпосередньо в пам'ять, а саме коди Java автоматично застосовуються після входу на веб-сторінку браузера. Для хакерів це можливість завантажувати руйнівні коди на веб-сторінки [12].

Cookies. Файли cookie – це невеликі файли, які створюють динамічні веб-сторінки на комп'ютерах користувачів, максимальна довжина цих файлів становить 4 Кб. Така компактна інформація може бути достатньо корисною для хакерів [10].

Атаки на паролі. За допомогою цього методу хакери беруть на себе контроль над усіма необхідними розділами для пошуку паролів, включаючи конфіденційний, комерційний та навіть пароль електронної пошти [13].

Підслуховування. Хоча хакери можуть перехопити дані, встановивши програмне забезпечення Sniffer, їм не вдасться модифікувати ці дані [14].

Системи DNS. Через недостатню безпеку та проблеми, пов'язані з налаштуваннями DNS, якщо хакер може замінити IP користувача на інший, то він може відображати адресу підроблених сайтів. Зловмисник може вказати користувачеві, чи є сайт справжнім або підробленим. Потім електронні листи надсилаються на неправильну адресу. Численні небезпеки загрожують такій системі, як отруєння кешу. Цей процес містить введення неправильної інформації в кеші серверу, що відноситься до назви системи (DNS). Ми можемо надіслати підроблену відповідь, використовуючи адресу виправлення на основі інформації про заявника. Тут виняток: якщо підроблена відповідь отримується швидше, ніж основна відповідь, вона розміщується в кеші. Після цього кеш буде отруєно за допомогою підробленої інформації. Численні користувачі перебувають у небезпеці до закінчення терміну підробленої інформації [15], [16].

DOS. Хакери відправляють великий трафік на IP-адресу через руйнівний бот. Сервери не можуть відповісти.

DDOS. Хакери можуть надсилати запити до відповідної IP-адреси.

Запуск спеціальної програми та керування системою. Атакуючий може керувати програмою за допомогою спеціальної інструкції. Цей процес знищує дані, тому використовуючи сценарії загального інтерфейсу шлюзу зловмисник отримає доступ до керування системою.

Ping метод. Хакери можуть знайти IP-адреси різних сайтів за допомогою цього методу. Ми можемо визначити, чи є активною IP-адреса, відправляючи чотири пакети на зазначену адресу отримавши відповіді. У цьому процесі ми також можемо отримати різну інформацію.

Слабкі сторони у розробці мережевої системи. Операційні системи – це прикладні програми. Ті слабкі місця, які загрожують програмному забезпеченню, також загрожують операційній системі. Іншими словами, зловмисник може застосувати ці слабкі місця для управління системою [9]. Нападники починають атакувати, вибираючи найкращий спосіб атаки. Зловмисник спочатку визнає цілі, а потім використовує відповідні сценарії відповідно до слабких точок програмного забезпечення. Він може отримати таку інформацію: пароль, що містить особисту інформацію, атаки на словники. У цьому типі вторгнення хакери випробовують конкретні слова для входу в систему. Вони також перевіряють можливі комбінації символів на цьому кроці. Після пошуку архітектури мережі та заявників вони замінюють свою IP-адресу в іншому місці шляхом підробки IP-адреси.

Колізійна атака. У криптографії колізійна атака відбувається у відповідній функції. Тут метою є пошук двох необов'язкових входів і зведення отриманого повідомлення. Існує два типи атак зіткнень: класична колізійна атака та колізійна атака із вибраним префіксом. Нападник не має жодного контролю над даними в класичній колізійній атаці. Дані вибираються за допомогою алгоритму. колізійна атака із вибраним префіксом більш потужна, ніж класична. Зловмисник може обрати два різних документи, а потім додає деякі значення до всіх документів.

Атака «людина в середині». Це свого роду атака, яка називається «активним підслуховуванням». Зловмисник забезпечує незалежні зв'язки між

жертвами та розповсюджує їхні повідомлення. Напад «людина в середині» підходить для зашифрованих протоколів зв'язку та взаємодії ключів. У цьому типі атаки, коли два користувачі обмінюються загальними ключами, атакуючий поміщає себе між двома користувачами. Обидва користувачі вважають, що вони мають безпечне з'єднання при такому нападі, але атакуючий може перехоплювати весь трафік [5].

Шкідливе програмне забезпечення (Malwares). Це комп'ютерні програми, розроблені та застосовані зловмисниками для порушення роботи комп'ютера, збирання важливої інформації або доступу до приватних комп'ютерних систем. Зловмисне програмне забезпечення може з'явитися у вигляді коду, сценарію, активного вмісту та іншого програмного забезпечення. Шкідливе програмне забезпечення – це загальний термін, який використовується для різних видів проблемних програм. Це включає:

а) комп'ютерний вірус;

б) ransomware;

в) комп'ютерний хробак;

г) троянець;

ґ) руткіти;

д) «задні двері»;

е) key logger;

є) adware (рекламне програмне забезпечення) і т. д.

1.1.3 Виявлення вторгнення та його методи

Мережеві комп'ютерні системи нерідко є метою хакерів, оскільки ці системи відіграють важливу роль в передачі інформації. Методи запобігання вторгнення включають в себе автентифікацію користувачів, використання паролів, брандмауерів та захист даних шифруванням. Виявлення вторгнення

можна виділити як ще один фільтр для захисту комп'ютерних мереж. Його метою є виявлення несанкціонованого використання, неправильного використання та пошкодження комп'ютерних систем та мереж як внутрішніми користувачами, так і зовнішніми атакуючими [17].

Загалом, існує два підходи для реалізації тесту на виявлення вторгнення:

1) Виявлення зловживання. Функцією даної методики є виявлення існуючих атак та визначення шаблону для аналізу.

2) Виявлення аномалій. Функція даної методики – це виявлення нормальної роботи системи, індексація нормальної поведінки за допомогою системи аналізу та пошуку аномальної активності [18].

Системи виявлення вторгнень. Створюються як апаратні так і програмні системи і кожна з них має свої переваги та недоліки, що застосовуються внутрішніми та зовнішніми користувачами. Швидкість і точність – це переваги апаратних систем. Методи виявлення, що використовуються в системах виявлення вторгнень, поділяються на дві категорії:

1) Метод виявлення аномальної поведінки

2) Метод виявлення на основі сигнатур або виявлення зловживань [9].

Метод виявлення аномальної поведінки. У цьому методі створюється подання на нормальну поведінку. Аномалія може вказувати на втручання. Підходи, такі як нейронні мережі, техніка машинного навчання використовуються для створення поглядів на нормальну поведінку. Для виявлення ненормальної поведінки слід виявити нормальну поведінку та визначити її специфічну схему. Поведінка, що слідує за цими моделями, є нормальною, а події, що відхиляються від цих шаблонів більше, ніж звичайні статистичні дані, називаються ненормальною поведінкою. Незвичайні вторгнення важко виявляти, оскільки немає стійкої схеми для моніторингу.

Подія, яка трапляється набагато вище або нижче двох стандартних відхилень, часто вважається ненормальною. Наприклад, користувач входить і виходить двадцять разів на день замість звичайного один-два рази, або комп'ютер використовується в опівночі, хоча він не повинен бути включений після робочого часу. Кожен з цих випадків може розглядатися як ненормальна поведінка. Методи та критерії, що використовуються для виявлення аномальної поведінки, є такими:

а) виявлення порогового рівня;

б) статистичні критерії;

в) юридичні критерії;

г) методи кластеризації;

1.2 Методи отримання даних про атаки

1.2.1 Аналіз журналів реєстрації

Це метод був реалізований найпершим з усіх. Суть його полягає в аналізі журналів реєстрації (audit, log), які створюються ОС, прикладним ПО, маршрутизаторами і т.д. Всі дані записані в журналі реєстрації аналізуються СОА.

Як видно, метод є досить простим в реалізації, але в той же час має і недоліки через це: (В переліках не можна застосовувати декілька речень – виправляйте!)

а) для доказу наявності несанкціонований дій необхідно зафіксувати в журналах реєстрації великий обсяг даних для подальшої їх обробки, тобто все це негативно позначається на швидкості роботи системи;

б) для аналізу даних з журналу реєстрації необхідний фахівець;

в) всі журнали зберігають інформацію в різному вигляді, так як немає певного стандарту для ведення таких журналів;

г) аналіз даних відбувається не в реальному часі, так як він відбувається після накопичення великої кількості інформації, тобто раннє виявлення атаки неможливо і не можна перешкодити атаки, можна тільки запобігти наслідкам проведення атаки;

ґ) атаки не можуть бути виявлені на вузлах, де не ведеться журнал реєстрації.

Даний метод, як правило, використовується в комбінації з іншими методами виявлення атак, які будуть розглянуті далі. Застосування журналів реєстрації дозволяє отримувати інформацію про наявність атаки після її проведення. Це може допомогти запобігти можливі наслідки атаки, а також створити ефективні заходи щодо запобігання реалізації проведеної атаки в подальшому.

1.2.2 Аналіз «на льоту»

Сенс аналізу «на льоту» є моніторинг мережевого трафіку в реальному часу або близькому до реального часу, а також використання необхідних

алгоритмів для виявлення атак. В мережевому трафіку визначається рядок, яка вказує на несанкціоноване дію.

Перевагами виявлення атак в мережевому трафіку є:

а) всього один агент СОА може аналізувати весь сегмент мережі з великою кількістю хостів на відміну від першого методу, в якому для кожного хоста потрібен свій агент для здійснення аналізу, тобто метод може виявити атаки на всі хости, включаючи і маршрутизатори;

б) такі системи працюють в режимі реального часу і можуть виявити атаку під час її проведення та вжити заходів щодо ліквідації проведення атаки;
в) зловмисникові неможливо приховати сліди свого несанкціонованої дії, так як використовується «живий» трафік в режимі реального часу,. тобто знаючи як працювати і маніпулювати журналами реєстрації, зловмисник може приховати сліди присутності в системі, але це вже буде пізно при аналізі «на льоту», тобто в аналізованих даних можна знайти також інформацію, яка допоможе при ідентифікації зловмисника;
г) система може виявляти невдалі атаки, а також підозрілу діяльність, якщо встановити СОА попереду, із зовнішнього боку, брандмауера.

Не дивлячись на таку велику кількість переваг аналізу «на льоту», є і недоліки в даному методі. Найголовніший недолік полягає в тому, що такі системи майже не застосовні в високошвидкісних мережах.

1.2.3 Використання профілів «нормального» поведінки

Суть методу використовується профіль «нормального» поведінки полягає в порівнянні очікуваних значень профілю при нормальній роботі системи, який створюється при навчанні СОА і отриманих даних з спостереження. Найчастіше спостереження відбуваються за користувачами, системною діяльністю або мережевим трафіком.

При навчанні і використанні системи стикаються з наступними проблемами:

а) побудова профілю користувача;

б) знаходження граничних характеристик нормального поведінки користувача, що має привести до зниження ймовірності появи одного з двох випадків:

в) спостерігається поведінка суб'єкта визначається як аномальне, в той час як воно є нормальним;

г) спостерігається поведінка суб'єкта визначається як нормальне, в той час як воно є аномальним - помилка першого роду.

Використання профілів «нормального» поведінки не часто використовується в системах виявлення атак. Хоча воно знайшло своє застосування в фінансових структурах для виявлення шахрайства.

1.2.4 Використання сигнатур атак

Використання сигнатур атак часто співвідносять з аналізом «на льоту». Суть методу полягає в поданні атаки у вигляді сигнатури і пошуку даної сигнатури в мережевому трафіку або журналі реєстрації. Сигнатура може являти собою шаблон дій або рядок із символів, що характеризують атаку. Для сигнатур існують БД, в яких вони і зберігаються і надалі витягуються для аналізу.

Перевагами даного методу є простота реалізації і ефективність. Але також існують і недоліки його використання:

а) створення мови опису сигнатур;

б) як уявити атаку у вигляді сигнатур, щоб зафіксувати всі її можливі модифікації.

Найчастіше в сучасних системах комбінування методів отримання даних про атаки грунтується на виявленні атак на рівні мережі і на рівні хоста. А саме:

в) network-based, як правило, використовують аналіз «на льоту» і сигнатура;

г) host-based, в той час, використовують профілі «нормального» поведінки і аналіз журналів реєстрації.

1.3 Методи аналізу даних

Методи інтелектуального аналізу даних (Data Mining):

Найпростішим є метод регресійного аналізу, але в той же час він і найменш ефективний. Найпростіша схема даного методу є один вхідний (незалежний) параметр і один результуючий (вихідний) параметр;

Другий метод це класифікація. По-іншому його також називають метод класифікаційних дерев або дерево прийняття рішень. Даний метод являє собою алгоритм аналізу даних, при якому будується покроковий спосіб прийняття рішень відповідно до значень певних параметрів. В результаті будується дерево прийняття рішень, де кожен вузол являє собою точку прийняття рішення в залежності від вхідних параметрів. В результаті аналізу значення вхідного параметра відбувається перехід до потрібного вузла, і так далі, поки алгоритм не дійде до листа дерева, при цьому візьметься остаточне рішення і виведеться відповідь на поставлене завдання;

Третій метод це кластеризація. Метод дозволяє розбити вхідні дані на групи за певною ознакою. У цьому методі всі дані діляться на групи за певним правилом, які вказують на взаємозв'язок даних всередині кожної групи.

1.3.1 Статистичний підхід

Даний підхід має дві головні переваги:

а) використання математичного апарату статистики;

б) адаптація до поведінки суб'єкта.

Підхід починається з визначення профілів для кожного суб'єкта даної системи. Надалі відбувається порівняння використовуваного профілю з

еталоном і при будь-якому відхиленні розглядається як несанкціоноване дію. Метод є універсальним, так як при його використанні не потрібно ніяких знань про можливі атаки і вразливості системи.

а) Існують також і недоліки статистичного підходу:

б) такі системи не чутливі до порядку проходження подій;

в) важко поставити порогові значення характеристик для виявлення мережевих атак, що в подальшому буде ідентифікувати аномальну діяльність;

г) систем можуть бути перенавчитися зловмисником, щоб аномальні дії розглядалися як нормальні.

Через таких недоліків статистичний підхід застосовують з додатковими методиками.

1.3.2 Експертні системи

Суть експертних систем полягає в існуванні набору правил, які представляють собою знання експерта. Це один з найпоширеніших методів виявлення атак, в якому інформація про атаки представляється у вигляді правил. Правила, відповідно, можуть являти собою сигнатуру або послідовність дій. Якщо правило було виконано, то вважається, що дія є несанкціонованим. При такій реалізації майже відсутня помилка помилкових тривог.

Всі правила зберігаються в БД експертної системи, яка повинна містити більшість відомих атак. Для більш надійної роботи необхідно, щоб БД постійно оновлювалися, поповнюючись новими правилами про нові атаки.

Головним недоліком такого підходу є неможливість виявлення невідомих атак. При цьому, якщо модифікації відомих атак не були прописані правилами, то такі модифікації також не виявлятися.

1.3.3 Нейронні мережі

Велика кількість методів виявлення атак використовують аналіз контрольованого простору на основі правил або статистичного підходу. Контрольований простір може бути представлено журналом реєстрації або мережевим трафіком. Все це спирається на застосування правил, які були задані заздалегідь, для виявлення атак. Якщо ж атака була розбита в часі або було кілька зловмисників, то правила не зреагують. А також через різноманітність атак, навіть якщо вони постійно оновлювані правила БД експертної системи не зможуть ідентифікувати весь діапазон атак.

Проблеми, що виникли при використанні експертних систем, можуть бути вирішені використанням нейронних мереж. Нейронна мережа аналізує дані і допомагає оцінити, чи відповідають дані характеристикам, які вона навчена розпізнавати на відміну від експертних систем, які дають відповідь на основі перевірки збережених правил в БД. Достовірність оцінки нейронної мережі залежить від якості навчання даної мережі.

Для початку відбувається навчання правильної ідентифікації несанкціонованих дій на спеціально підібраною вибірці прикладів. Аналізується реакція мережі і відбувається її налаштування. Під час її експлуатації система набирається досвіду.

Головною перевагою таких мереж є те, що при виявленні несанкціонованих дій можуть вивчати характеристики атак і знаходять елементи, які не схожі на вихідні, що дає можливість виявляти модифікації атак.

1.3.4 Метод найближчих сусідів

Одним з найпростіших алгоритмів класифікації є метод найближчих сусідів. Основним принципом методу найближчих сусідів є те, що об'єкту

присвоюється той клас, який є найбільш поширеним серед сусідів даного елемента. Сусіди беруться з безлічі об'єктів тренувального набору даних і, на основі параметра k визначається, який клас найбільш численний серед них.

Однак базова версія даного алгоритму вимагає повного перебору даних для пошуку найближчих сусідів. Для адаптації обраного методу до потокових даних використовуємо техніку плаваючих вікон. В окремий момент часу алгоритм буде зберігати в пам'яті всього кілька елементів, що надійшли останніми [39].

1.3.5 Дерева Хефдінга

Дерева рішень є класичними засобами прийняття рішень, що використовують модель дерева. В математичному навчанні вони можуть бути використані як для класифікації, так і для кластеризації. Структурно дерева рішень влаштовані таким чином: внутрішні вузли дерева є деякими правилами фільтрації елементів за допомогою їх атрибутів, гілки дерева, які виходять із такого вузла - можливі результати застосування таких фільтрів, листя ж - прийняті рішення, тобто результуючі класи елементів.

Дерева рішень і взагалі класифікатори, засновані на деревах, є популярними з кількох причин. Перше - деревоподібна структура проста для інтерпретації. У дереві легко можна простежити як саме той чи інший об'єкт отримав свій клас. Також дерева відмінно підходять для візуалізації та модифікування під певний клас задач. Друге - пророкування класу елемента відбувається дуже швидко. Навчена модель здатна передбачати клас елемента за логарифмічне число швидких перевірок правил. З цієї причини такі моделі зазвичай використовуються в одній з найбільш залежних від часу областей - Web-пошуку. Третє - дерева є потужними класифікаторами, здатними моделювати нелінійні зв'язки.

Дерево Хефдінга, або Very Fast Decision Tree (VFDT) - це потокове дерево рішень зі статистичними гарантіями. Зокрема, воно використовує

кордон Хефдінга, заснований на нерівності Хефдінга, що гарантує що навчаєма модель буде давати досить точні прогнози при наявності достатньої кількості елементів для росту дерева.

1.4 Висновки до розділу

Щоб зробити процес використання комп'ютерів безпечніше, провідні компанії в галузях інформаційної безпеки розробили низку захисного програмного забезпечення. Необхідно враховувати безпеку мережі за допомогою різних платформ. Цей метод має деякі переваги для захисту від вразливостей у всіх комп'ютерних системах. Системи виявлення вторгнень (СВВ) відіграють важливу роль у виявленні аномалій та атак у мережі. Результати показують, що технології Data Mining та використання їх алгоритмів для виявлення прихованих та супутніх даних є ефективними та мають високу швидкість реагування. Крім того, основні елементи класифікації даних, високий рівень взаємодії з людьми, відсутність міток даних та ефективність відхилення службових атак за допомогою алгоритмів Data Mining можуть бути дуже корисними в системах виявлення вторгнень.

РОЗДІЛ 2
ВИБІР ТЕХНОЛОГІЙ ДЛЯ СИСТЕМ ВИЯВЛЕННЯ ВТОРГНЕНЬ В МЕРЕЖУ

Процес виявлення вторгнень є процесом оцінки підозрілих дій, які відбуваються в мережі. Іншими словами, виявлення вторгнень - це процес ідентифікації та реагування на підозрілу діяльність, спрямовану на обчислювальні або мережеві ресурси. Головне завдання СОВ полягає в автоматизації функцій щодо забезпечення інформаційної безпеки мережі і забезпеченні «прозорості» функцій інформаційної безпеки для неспеціалістів в області захисту інформації. Тому СОВ - це системи, які збирають інформацію з різних точок мережі (захищається комп'ютерної системи) і аналізують цю інформацію для виявлення не тільки спроб, але і реальних порушень захисту (вторгнень) [21-23].

Для реалізації функцій по захисту від вторгнень сучасна СОВ повинна містити такі основні елементи (рис. 2.1):

підсистему збору інформації; підсистему аналізу; модуль управління; модуль реагування.

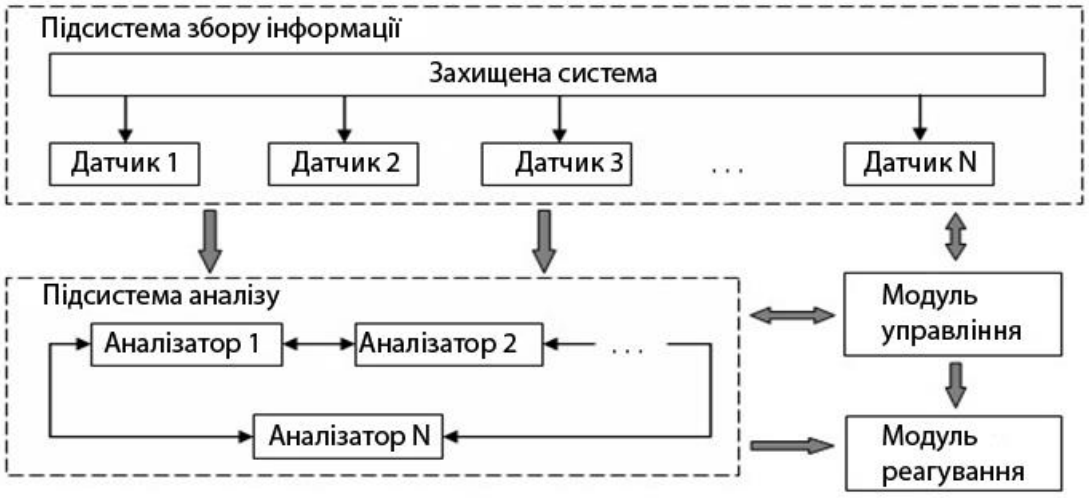

Рисунок 2.1 - Загальна структура системи виявлення вторгнень

Підсистема збору інформації акумулює дані про роботу захищається системи. Для збору інформації використовуються автономні модулі-датчики. Кількість використовуваних датчиків різна і залежить від специфіки захищається системи.

Підсистема аналізу (виявлення) здійснює пошук атак і вторгнень в захищається систему. Вона структурно складається з одного або більше модулів аналізу - аналізаторів. Кожен аналізатор виконує пошук атак або вторгнень певного типу. Вхідними даними для аналізатора є інформація з підсистеми збору інформації або від іншого аналізатора. Результат роботи підсистеми представляється у вигляді індикації про стан захищається системи та іншої необхідної інформації.

Модуль керування дозволяє управляти компонентами СОВ, а також стежити за станом захищається системи.

Модуль реагування призначений для виконання визначених дій в разі встановлення факту атаки.

Одним з найскладніших компонентів СОВ є підсистема аналізу (виявлення порушень безпеки), від властивостей якої фактично залежить безпека захищається корпоративної мережі. Ефективність цієї підсистеми в значній мірі визначається можливостями використовуваного методу аналізу даних про стан захищається системи [24]. Останнім часом в СОВ стали використовувати також технології інтелектуального аналізу даних (Data Mining), що дозволяють вирішувати слабоструктуровані і погано формалізуються завдання, до яких відносяться і завдання виявлення порушень інформаційної безпеки.

Основними цілями інтелектуального аналізу даних є пошук функціональних і логічних закономірностей в накопиченої інформації, побудова моделей і правил, які пояснюють знайдені аномалії і / або прогнозують розвиток деяких процесів, а також виявлення прихованих знань у вигляді кореляцій, тенденцій і взаємозв'язків, які аналітик не в змозі виявити й узагальнити самостійно. Технології Data Mining, на відміну від

традиційних методів обробки даних, які розширюють можливості виконувати оцінку стану спостережуваних процесів, виявляти і ранжувати причини значущих змін, прогнозувати розвиток процесів і виробляти рекомендації з підготовки можливих варіантів рішень з прогнозом їх наслідків.

Під інтелектуальним аналізом даних розуміється перетворення великих обсягів «сирих» даних в дані, які мають сенс.

Класифікувати аналізу даних можна за двома видами:

а) прямий (прогнозування);

б) непрямий (класифікація і кластеризація).

Завданням прямого аналізу є прогнозування конкретних показників. Так значення одного показника обчислюється через значення інших показників, які взаємопов'язані з ним.

Завданням же непрямого аналізу є створення груп даних в існуючому наборі або пошук в даних певної структури або схеми. Наприклад, класифікація мережевого трафіку на нормальний трафік і аномальний трафік за певним принципом.

2.1 Виявлення мережевих атак

Виявлення атак - це процес розпізнавання і реагування на підозрілу діяльність, спрямовану на мережеві або обчислювальні ресурси. Більшою мірою на ефективність виявлення атак впливають методи, використовувані для аналізу розглянутих даних.

Виявлення мережевих атак засноване на аналізі мережевого трафіку. 41 параметр - максимальна кількість інформації, яку можна було б отримати з аналізу пакетів. Ці параметри були виявлені на основі набору даних, які були отримані під час програми оцінки виявлення проникнення DARPA 1998 року разом з MIT Lincoln Labs. Набір даних був змодельований і створений у військовому середовищі, в якій локальну мережу військово-повітряних сил була піддана змодельованим атакам. Зібрані дані становили 4 ГБ стислих

даних дампа TCP, що зайняло 7 тижнів збору інформації і включали приблизно 5 мільйонів з'єднань. Розглянемо 41 параметр [25]:

duration - довжина (кількість секунд) з'єднання;

protocol_type - тип протоколу;

service - мережевий протокол на вузлі призначення;

src_bytes - кількість біт даних, що від джерела на вузол призначення;

dst_bytes - кількість біт даних, що від призначення на вузол джерела;

flag - нормальне з'єднання або є помилка в з'єднанні;

land - 1, якщо з'єднання від того ж вузла / порту або до того ж вузла / порту; інакше 0;

wrong_fragment - кількість неправильних фрагментів;

urgent - кількість термінових пакетів;

hot - кількість «гарячих» індикаторів;

num_failed_logins - число невдалих спроб входу;

logged_in - 1, якщо вдалий вхід; інакше 0;

num_compromised - число умов, що ставлять під загрозу;

root_shell - 1, якщо коренева оболонка отримана;

su_attempted - 1, якщо була команда "su root", інакше 0;

num_root - кількість доступів через root;

num_file_creations - число операцій створення файлу;

num_shells - кількість оболонок запиту;

num_access_files - кількість операцій по контролю доступу;

num_outbound_cmds - число вихідних команд в сеансі ftp;

is_hot_login - 1, якщо вхід належить до «гарячого» листу; інакше 0;

is_guest_login - 1, якщо вхід стався гостем; інакше 0;

count - кількість з'єднань з тим же вузлом, як і поточне

з'єднання за останні 2 секунди;

serror_rate - відсоток з'єднань, які мають SYN помилки;

rerror_rate - відсоток з'єднань, які мають REJ помилки;

same_srv_rate - відсоток з'єднань з тим же мережевим протоколом;

diff_srv_rate - відсоток з'єднань з різними мережевими протоколами;

srv_count - кількість з'єднань з тим же мережевим протоколом як і поточне з'єднання за останні 2 секунди;

Відноситься до з'єднання тієї ж служби: srv_serror_rate, srv_rerror_rate, srv_diff_host_rate;

class - 1, якщо є аномалія;

інакше 0;

Відноситься тепер до вузла призначення:

dst_host_count, dst_host_srv_count, dst_host_same_srv_rate,

dst_host_diff_srv_rate, dst_host_same_src_port_rate,

dst_host_srv_diff_host_rate, dst_host_serror_rate,

dst_host_srv_serror_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate.

Функції, представлені в наборі даних, можуть бути згруповані в 3 категорії:

а) основні характеристики окремих з'єднань;

б) особливості з'єднання;

в) параметри з'єднання за проміжок часу.

Отриманий під час програми набір даних був поміщений в базу даних KDD-99 (матеріали третього змагання в області вилучення знань і засобів аналізу даних, що проводиться в рамках п'ятої міжнародної конференції вилучення знань і аналізу даних) База містить аналіз типів атак і відповідних їм ознак для великого числа прецедентів.

Всі атаки, виявлені в тому наборі даних можна розділити на чотири види атак:

1. Відмова в обслуговуванні (DoS). Зловмисник перевантажує систему запитами для того, щоб система не могла обробляти легальні запити.
2. Неавторизований доступ до ядра (U2R). Використання вразливостей для несанкціонованого доступу. Наприклад, переповнення буфера.
3. Отримання доступу (R2L). Атакуючий посилає пакети в систему, використовуючи вразливості, для отримання локального доступу.
4. Сканування (PROBE). Сканування системи для отримання інформації про уразливість в ній.

Також представлено кілька класів атак в наборі даних, які представлені в таблиці 2.1 [26].

Таблиця 2.1 Набір атак для різних класів

Тип	Клас
DoS	Neptune, smurf, Pod, teardrop, land, back
Probe	Portsweep, ipsweep, satan, nmap
U2R	buffer_overflow, loadmodule, perl, rootkit
R2L	guess_passwd, ftp_write, imap, phf, multihop, warezmaster, warezclient, spy

На підставі вище описаного можна зробити висновок таблиця 2.2

Таблиця 2.2 - Переваги та недоліки основних методів виявлення порушень інформаційної безпеки

Метод	Переваги	Недоліки
Аналіз журналів реєстрації	простота реалізації	погіршення швидкості роботи для журналів великого обсягу; необхідна допомога фахівців; немає уніфікованого формату зберігання журналів; аналіз не в реальному часі; на кожен аналізований вузол необхідний свій агент
Статистичні методи	використання вже розробленого і зарекомендував себе апарату математичної статистики	не чутливі до порядку проходження подій; трудність завдання порогових значень показників подій; «Статистичні» системи можуть бути «навчені» порушниками
Аналіз «на льоту»	один агент СОВ може переглядати цілий сегмент мережі; визначення атак в реальному масштабі часу; неможливість зловмисникові приховати сліди своєї діяльності	підвищені вимоги до апаратного забезпечення (особливо в високошвидкісних мережах); неефективність роботи в комутованих мережах і мережах з канальним шифруванням
Профілі «нормального поведінки»	виявлення найменших відхилень від «нормального» поведінки	велика складність побудови профілю
Використання сигнатур	простота реалізації; визначення конкретної атаки з високою точністю і малою часткою помилкових спрацьовувань	нездатність виявлення нових атак і вторгнень; необхідність оперативного оновлення баз даних сигнатур; пропуски модифікацій відомих атак

2.2 Алгоритми зменшення розмірності

Більш зрозумілі і прозорі результати аналізу в Data Mining можуть бути отримані, якщо з безлічі вихідних змінних і значень використовувати якісь узагальнені змінні. Таким чином, виникає зменшення розмірності даних.

Одні із способів по зменшенню розмірності даних є метод головний компонент (PCA). В даному методі втрата інформативності даних мала. Метод створений Пирсоном в 1901 році. Метод застосовується в наступних областях: розпізнавання образів, придушення шуму на зображеннях, комп'ютерний зір, стиск даних, компресія зображення і відео, скорочення розмірності динамічних моделей і т.п. Суть алгоритму полягає в знаходженні власних векторів і власних значень матриці коваріаций вихідних даних. В цьому випадку власні значення будуть представляти собою головні компоненти.

Існує чотири основні версії з аналізу головних компонент [28]:

а) апроксимація даних лінійним різноманіттям меншої розмірності;

б) знаходження підпростору в ортогональної проекції, в якій розкид максимальний;

в) знаходження підпростору в ортогональної проекції, в якій середньоквадратичне відстань між точками максимально; анулювання кореляції між координатами.

Але, вирахувавши всі головні компоненти і розподіливши їх за пріоритетами, іноді необхідно ще знайти ту кількість головних векторів, яке доцільно залишити. Для оцінки числа головних компонент використовується правило зламаною тростини.

2.2.1 Апроксимація даних лінійним різноманіттям

Вхідні динячі представлені кінцевим безліччю векторів x_1, x_2, ... $x_m \epsilon R_n$. Для кожного параметра з індексом k = 0,1, ..., n - 1 серед усіх k -мірних різноманіть в R_n треба знайти таке лінійне різноманіття L_k-R_n, що сума квадратів відхилень x_i від L_k мінімальна:

$$\sum_{i=1}^{m} dist^2 (x_i \, \mathrm{L_k}) \rightarrow \min, \qquad (2.1)$$

де L_k - лінійне різноманіття,

$d(x_i,L_k)$- евклідова відстань від точки до лінійного різноманіття. Лінійне різноманіття, в даному випадку, може бути представлено як

$\mathrm{L_k} = \{a_0 + \beta_1 a_1 + \cdots + \beta_k a_k \,|\beta_i \epsilon R\}, a_0 \epsilon R^n, \{a_1, \ldots, a_k \subset R^n\}$ - ортонормованій набір векторів.

$$\mathrm{dist}^2 (x_i, L_k) = ||x_i - a_0 - \sum_{i=1}^{k} a_i (a_j, x_i - a_0)||^2 \qquad (2.2)$$

де a_j - вектори головних компонент в побудованих лінійних многовидах.

Формула (2) в координатної формі являє собою:

$$\mathrm{dist}^2 (x_i, L_k) = \sum_{l=1}^{n} (x_{il} - a_{0l} - \sum_{j=1}^{k} a_{jl} \sum_{q=1}^{n} a_{iq} (x_{iq} - a_{0q})) \qquad (2.3)$$

набір вкладених лінійних різноманіть

$L_0 \subset L_1 \subset \ldots L_{n-1}$, $L_k = \{a_0 + \beta_1 a_1 + \cdots + \beta_k a_k | \beta_i \epsilon R\}$ дає рішення задачі апроксимації. Різноманіття визначаються набором векторів $\{a_1, \ldots, a_{n-1}\}$ (векторами головних компонент) і вектором a_0.

Рішенням задачі мінімізації для L_0 є вектор a_0:

$$a_0 = \operatorname{argmin} \left(\sum_{i=1}^{n} \operatorname{dist}^2 (x_i, L_0)\right) = \operatorname{argmin} \left(\sum_{i=1}^{m} \|x_i - a_0\|^2\right) \quad (2.4)$$

В цьому випадку a_0 - вибіркове середнє:

$$a_0 = \frac{1}{m} \sum_{i=1}^{m} x_i = \overline{X} \quad (2.5)$$

Тепер вектори головних компонент можна знайти за допомогою вирішення однотипних завдань оптимізації:

1) центруємо дані, тобто віднімаємо з кожного значення його середнє:

$$x_i = x_i - \overline{X} \quad (2.6)$$

2) вираховуємо першу головну компоненту:

$$a_1 = \operatorname{argmin} \left(\sum_{i=1}^{m} \|xi - a_1(a_1, x_i)\|^2\right) \quad (2.7)$$

Якщо рішення не єдине, то вибирається одне з них;

3) Позбавляємося від першої головної компоненти, віднімаючи з даних проекцію на першу головну компоненту:

$$x_i = x_i - a_1 (a_1, x_i) \quad (2.8)$$

3) Тепер знайдемо другу головну компоненту:

$$a_2 = \operatorname{argmin} \left(\sum_{i=1}^{m} \|xi - a_2(a_2, x_i)\|^2\right) (2.9)$$

і т.п. Знаходимо все k головних компонент таким способом.

2.2.2 Пошук ортогональних проекцій з найбільшим розсіюванням

Вхідні дані представлені в вигляді централізованого набору векторів даних

$$x_i \in R^n (i = 1, \dots, m)$$

Суть методу полягає в тому, щоб знайти ортогональное перетворення в таку нову систему координат, яка б задовольняла таким умовам:

а) вибіркова дисперсія уздовж нової першої координати повинна бути максимальна. Це координата буде першою головною компонентою;

б) вибіркова дисперсія уздовж нової другої координати повинна бути максимальна, а також ортогональна першій координаті. Це координата буде другою головною компонентою;

в) вибіркова дисперсія уздовж нової k -ої координати повинна бути максимальна, а також ортогональна першим k - 1 координатами. Це координата буде k -ої головною компонентою.

Вибіркова дисперсія даних уздовж потрібного напрямку, яке унормовано вектором a_k:

$$S_m^2 [(X, a_k)] = \frac{1}{m} \sum (a_k, x_i)^2 = \frac{1}{m} \sum_{i=1}^{m} (\sum_{j=1}^{n} a_{ij} x_{ij})^2 \quad (2.10)$$

2.2.3 Пошук ортогональних проекцій з найбільшим середньоквадратичним відстанню між окулярами

Даний метод полягає в пошуку підпросторів, в проекції на які середньоквадратичне відстань між точками буде максимально:

$$\frac{1}{m(m-1)} \sum_{i,j=1}^{m}(x_i - x_j)^2 == \frac{2m^2}{m(m-1)}\left[\frac{1}{m}\sum_{i=1}^{m} x_i^2 - \left(\frac{1}{m}\sum_{i=1}^{m} x_i\right)^2\right] \quad (2.11)$$

де m - кількість векторів вхідних даних (кількість прикладів),

x_i - вектора вхідних даних.

У цій формулі ліва частина представлена середньоквадратичним відстанню між точками, а в правій частині в квадратних дужках - вибіркова дисперсія.

2.2.4 Анулювання кореляції межу координатами

Суть методу полягає в обчисленні залежності між ознаками вхідних даних, а саме розрахунку кореляції між ними. Тоді рішенням даного завдання буде знаходження лінійних комбінацій ознак, які слабо корелюється один з одним:

$$cov\,(X_i, X_j) = 0, i \neq j \quad (2.12)$$

де X_i - вектора вхідних даних.

$$cov\,(X_i, X_j) = M\,[(X_i - M\,(X_i))\,(X_j - M\,(X_j))] \quad (2.13)$$

2.3 Алгоритм оцінки числа головних компонент

Після знаходження головних компонент, треба ще визначити потрібну їх кількість, так як завданням є якраз скорочення розмірності вхідних даних. Одним з таких способів - це правило зламаною тростини. Суть даного методу полягає в тому, що нормовані власні значення ковариационной матриці (головні компоненти) порівнюються з вирахувати значеннями довжин уламків тростини одиничної довжини, зламаною в n-1 випадкової точці.

Для початку вважаються довжини уламків тростини одиничної довжини:

$$l_i = M(L_i) = \frac{1}{n}\sum_{j=i}^{n}\frac{1}{j} \qquad (2.14)$$

де L_i, i = 1, ..., n - довжини отриманих шматків тростини, які пронумеровані в

порядку убування довжини;

M- математичне очікування;

n - кількість ознак у вхідних даних.

Для визначення того, чи залишається головна компонента як значущий параметр, використовуємо правило зламаною тростини: власний вектор залишається,

якщо:

$$\frac{\lambda_1}{trC} > l_1 \;\&\; \frac{\lambda_2}{trC} > l_2 \;\&\; \ldots \frac{\lambda_k}{trC} > l_k$$

де λi - власні вектора в порядку убування їх значень,

C- ковариационная матриця,

$tr\,(C)$ - сума діагональних елементів C.

2.4 Існуючі набори значущих параметрів

На основі досліджень раніше вже були отримані набори значущих параметрів.

У статті [29] можна знайти один з наборів. В цьому випадку для знаходження параметрів використовувався алгоритм DPSO. Ідея алгоритму полягає в вирішенні задачі оптимізації за допомогою моделювання поведінки популяції частинок в просторі параметрів. Координати частинок відповідно визначають рішення задачі оптимізації. Також частка має швидкість переміщення і прискорення. У процесі виконання завдання частки проходять весь простір рішень і знаходять оптимум, до якого будуть прагнути інші частинки на наступному кроці. Кожна частинка також запам'ятовує своє краще становище. Випадкові складові можуть бути задані «божевільними» частинками, які рухаються зовсім по іншому закону на відміну від інших.

Даний метод простий у реалізації і в процесі обчислення не використовується градієнт, що спрощує роботу з алгоритмом. Такий метод може використовуватися для вирішення таких завдань, як навчання нейромереж, завдання пошуку мінімуму функцій і завдань генетичних алгоритмів.

Отриманий, на основі методу DPSO, набір параметрів представлений в таблиці 2.3

Таблиця 2.3 Ключові параметри, знайдені за допомогою DPSO

Тип атаки	Параметри
DoS	5,10,24,29,33,34,38,40
Probe	2,3,23,34,36,40
U2R	3,4,6,14,17,22
R2L	3,4,10,23,33,36

Також набір значущих параметрів формується в статті [30], де використовується алгоритм SVDF. Він є різновидом методу опорних векторів. Спочатку вихідний набір даних подається на вхід класифікатора для навчання. Потім вирішальна функція класифікатора використовується для оцінки важливості параметрів. Процедура представляє собою наступне:

г) обчислюється вага від вирішальної функції вектора підтримки;

д) оцінюється важливість параметрів за абсолютним значенням ваг.

На основі даного методу отримані значущі параметри, представлені в таблиці 2.4.

Таблиця 2.4 Ключові параметри, знайдені за допомогою SVDF

Тип атаки	Параметри
DoS	1,5,6,23,24,25,26,32,36,38,39
Probe	1,2,3,4,5,6,23,24,29,32,33
U2R	1,2,3,4,5,6,12,23,24,32,33
R2L	1,3,5,6,32,33

У статті [31] набір значущих параметрів знаходиться за допомогою видалення за раз одного параметра, щоб оцінити параметр і ідентифікувати найважливіші для виявлення проникнень. Завданням було отримати 5 найбільш значущих параметрів для кожного типу атак.

Отримані результати представлені в таблиці 2.5

Таблиця 2.5 Ключові параметри з [31]

Тип атаки	Параметри
DoS	7,8,12,13,23
Probe	3,12,27,31,35
U2R	14,17,25,38,36
R2L	6,11,12,19,22

У статті [32] розглядається підхід досягає найвищої усередненої точності зі скороченням розміру вхідних даних в порівнянні з іншими.

Результати, з використанням даного підходу, представлені в таблиці 2.6 ключові параметри з [32]

Таблиця 2.6 Ключові параметри

Тип атаки	Параметри
DoS	1,2,3,4,5,6,12,23,24,31,32,37
Probe	1,2,3,4,12,16,25,27,28,29,30,40
U2R	1,2,3,10,16
R2L	1,2,3,4,5,10,22

2.5 Вибір платформи для проведення експерименту

Для виявлення мережевих атак у вже наявному наборі даних можна використовувати програмний продукт WEKA.

Інструментальне засіб WEKA - набір сучасних алгоритмів навчання машини і інструментів попередньої обробки даних. Також як і велика різноманітність алгоритмів, програма включає широкий діапазон попередньої обробки інструментів [27].

Інструментальні засоби включають методи для всіх стандартних проблем аналізу даних: регресія, класифікація, кластеризація.

Один спосіб використання WEKA полягає в тому, щоб застосувати метод вивчення до набору даних і проаналізувати його висновки, щоб дізнатися більше про дані. Метод вивчення називається класифікація. Інший спосіб повинен використовувати вивчені моделі, щоб генерувати прогнози. Третій полягає в тому, щоб приймати кілька різних висновків і порівнювати їх за продуктивністю, щоб вибрати один для подальшої роботи.

Найпростіший спосіб використовувати WEKA через графічний інтерфейс, званий Провідником. Це дає доступ до всіх його засобів.

Другий спосіб це інтерфейс The Knowledge Flow. Він дозволяє проектувати настройки конфігурації для переданої потоком обробки даних.

Третій спосіб - the Experimenter. Він розроблений щоб допомогти відповідати на основний практичне питання при застосуванні конфігурації і методів регресії: які методи і значення параметрів працюють краще.

Одне з основних переваг WEKA полягає в тому, що це не тільки самостійне додаток, але і автономний Java JAR-файл, в який можено скопіювати в папку бібліотек на сервері і використовувати в коді серверного додатка.

2.6 Знаходження значущих параметрів

Для ефективного виявлення мережевих атак. Виявлення вторгнень відбуватиметься на основі аналізу значущих параметрів в мережевому трафіку. Як було розглянуто раніше, існує велика кількість наборів значущих параметрів, які були отримані до нас. Також треба перевірити ефективність вже наявних наборів параметрів і розглянути метод по скороченню розмірності вхідних даних і знаходженню значущих параметрів. Дані методи грають важливу роль, так як усунення непотрібних (не значима) параметрів покращує точність виявлення і прискорює обчислення.

Для роботи знадобляться вхідні дані з KDD-99. Дані представлені у вигляді 42 параметрів і показані на рис 2.3.

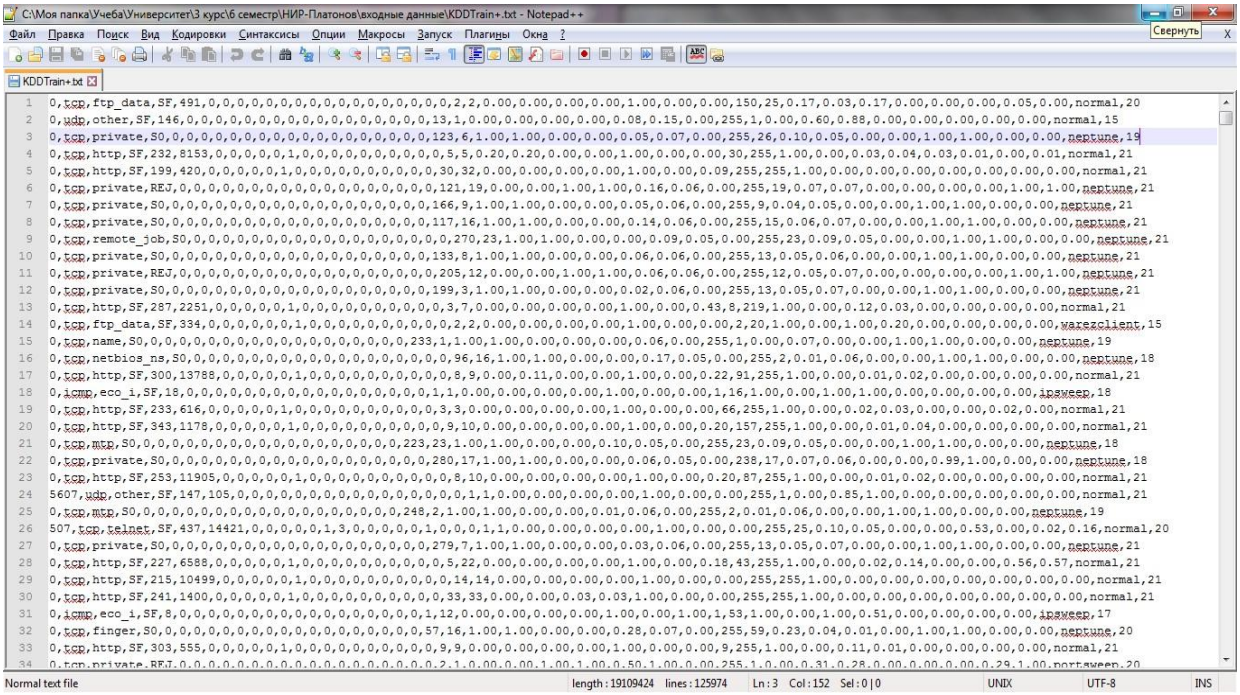

Рисунок 2.3 - Мережевий трафік представлений в 41 параметрі

Для оцінки вже існуючих параметрів і знаходження наших власних необхідно розділити весь файл на 5 частин: Do, Probe, U2R, R2L, Normal.

Такий поділ зроблено за допомогою написаної програми. У знайденому наборі даних таку кількість прикладів кожного типу атак:

е) DoS - 45927,

ж) Probe - 11656,

з) U2R - 52,

и) R2L - 995,

к) normal - 67343.

При реалізації запропонованих нами методів треба врахувати, що не всі дані представлені в числовому вигляді, тому необхідно перетворити їх, щоб можна було аналізувати.

2.6.1 Оцінка існуючих наборів значущих параметрів

Для перевірки ефективності запропонованого алгоритму необхідно порівняти отримані результати з уже наявними параметрами. Раніше вже розглядалися існуючі набори значущих параметрів, але цього не достатньо. Для порівняння потрібна кількісна оцінка набору параметрів. В якості такого порівняння взята точність методу виявлення, а саме відсоток помилки першого і другого роду.

Помилкою першого роду є ситуація, коли йдеться про те, що це не атака, а насправді атака. Таку помилку треба якомога більше мінімізувати, навіть якщо в результаті з'являться помилки другого роду, які не так критичні для вирішення завдання безпеки. Для отримання помилок першого і другого роду використовується програмний продукт WEKA.

На вхід програмного продукту подається файл формату arff, складений з отриманих раніше документів. А саме частина прикладів взята з документа містить нормальний трафік, а інша відповідно до типу

аналізованої атаки. Вид отриманого документа представлений на рис. 2.4, де на початку прописуються всі параметри, а потім відомий нам мережевий трафік у вигляді 41 параметра.

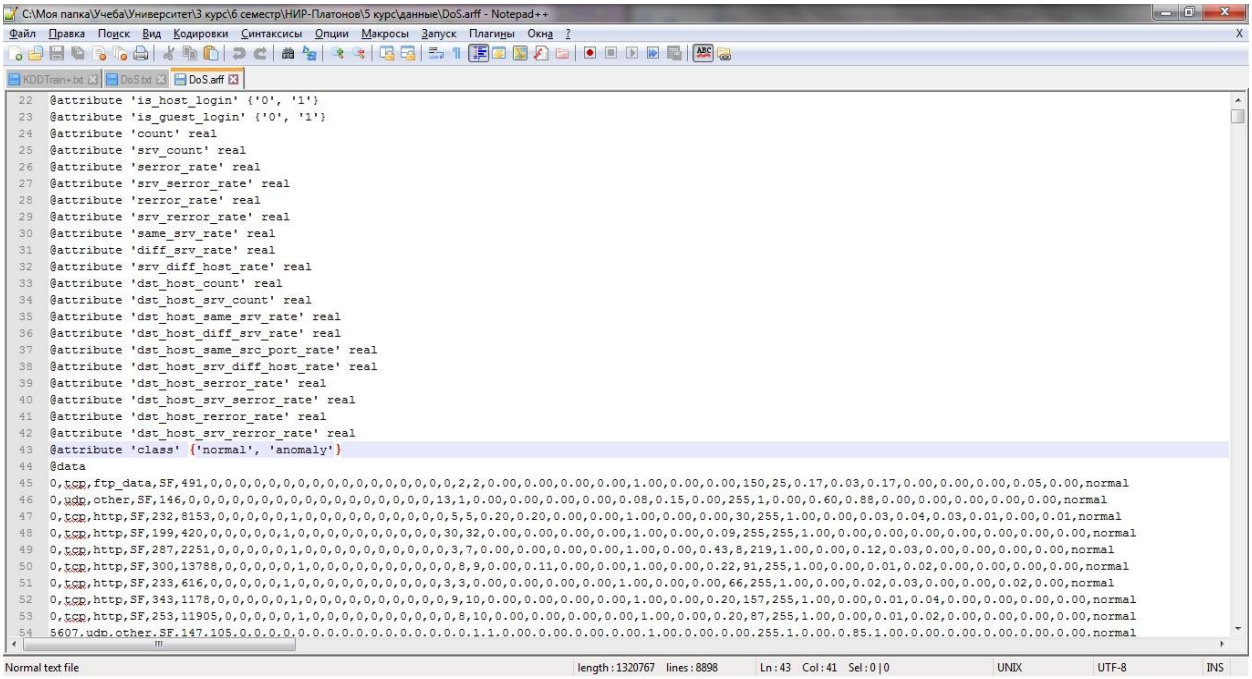

Рисунок 2.4 - Вхідні дані в форматі arff

Розглянемо алгоритми, які будуть використовуються в WEKA(Waikato Environment for Knowledge Analysis). Найголовніший метод це FilteredClusterer. Це метод виконання довільної кластеризації на даних, які були передані через довільний фільтр. Також як і кластери, структура фільтра грунтується виключно на даних тренування і тестові екземпляри будуть оброблені фільтром, не змінюючи їх структуру.

Даний метод передбачає вибір якогось алгоритму по кластеризації і набір фільтрів. У нашому випадку алгоритм кластеризації це kmeans алгоритм. А в наборі фільтрів поставлено AllFilter. У WEKA алгоритм kmeans може використовувати або Евклідова відстань (значення за

замовчуванням) або Манхеттенський відстань. Якщо Манхеттенський відстань використовується, то центроїди обчислені як покомпонентно медіана, а не середнє значення.

Під Евклідовому відстанню розуміється відстань, виражене формулою (2.15).

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (2.15)$$

точки в вимірному просторі, відстань між якими і шукається.

Інша відстань - це відстань міських кварталів (Манхеттенський відстань) [33]:

$$d(x,y) = \sum_{i=1}^{n}|x_i - y_i| \qquad (2.16)$$

Найчастіше це відстань дає такі ж результати як і Евклідова відстань, виняток лише в тому, вирахувана відстань є середнім різниць по координатах, що знижує вплив окремих великих викидів.

Алгоритм k - середній є алгоритм кластеризації на k кластерів. Кластери будуються як можна на великих відстанях один від одного. Кількість кластерів може бути вибрано на основі розв'язуваної задачі або попередніх результатах дослідженнях.

Суть алгоритму полягає в мінімізації функціоналу сумарною вибіркової дисперсії розкиду елементів щодо центрів тяжіння кластерів.

Алгоритм k-середній [34]:

л) вибираються k елементів з вибірки і задаються початковими центрами кластерів;

м) вся вибірка розбивається на k кластерів щодо обраних центрів;

н) вираховуються нові центри тяжіння в кожному кластері:

$$C_i^{(k+1)} = \frac{1}{|x_i^{(k)}|} \sum_{x \in x_i^{(k)}} x \quad (2.17)$$

о) якщо все нові центри тяжіння збігаються зі старими, то алгоритм закінчується, інакше продовжуємо розбивати вибірку по новому.

У таблиці 2.7 представлений отриманий аналіз

Таблиця 2.7 Аналіз ефективності існуючих наборів параметрів

варіанти наборів	Тип атаки: параметри	Помилка першого роду	Помилка другого роду
№ 1	DoS: 5,10,24,29,33,34,38,40	0,34%	0,35%
	Probe: 2,3,23,34,36,40	0,9%	1,3%
	U2R: 3,4,6,14,17,22	3,8%	3,5%
	R2L: 3,4,10,23,33,36	1,4%	8%
№ 2	DoS: 1,2,3,4,5,6,12,23,24,31,32,37	0,27%	0,2%
	Probe: 1,2,3,4,12,16,25,27,28,29,30,40	1,3%	2,3%
	U2R: 1,2,3,10,16	3,8%	10,3%
	R2L: 1,2,3,4,5,10,22	0,7%	4%
№ 3	DoS: 1,5,6,23,24,25,26,32,36,38,39	0,19%	0,37%
	Probe: 1,2,3,4,5,6,23,24,29,32,33	0,46%	0,5%
	U2R: 1,2,3,4,5,6,12,23,24,32,33	3,8%	8,6%
	R2L: 1,3,5,6,32,33	1%	6,2%
№ 4	DoS: 7,8,12,13,23	2,8%	11,4%
	Probe: 3,12,27,31,35	0,7%	2,9%
	U2R: 14,17,25,38,36	13,5%	5,2%
	R2L: 6,11,12,19,22	2%	9,4%

2.6.2 Реалізація алгоритмів по знаходженню значущих параметрів

Розглянемо МГК як метод по знаходженню значущих параметрів. Реалізується кілька варіантів методу, щоб в подальшому порівняти і їх ефективність.

Суть всього методу головних компонент полягає в спрощенні набору даних за рахунок скорочення даних в розмірі при найменшій втраті інформативності всього набору даних для подальшого аналізу.

Власні значення матриці коваріації знайдемо за допомогою QR-алгоритму та перетворення Хаусхолдера [35]. Кількість значущих параметрів за допомогою методу зламаною тростини.

Для анулювання кореляції між координатами необхідно побудувати матрицю ковариаций за наступним принципом:

$$C_{ij} = \frac{1}{|P|-1}\sum_{a=1}^{|p|}(x_i^a - \overline{x_\iota})\,(x_j^a - \overline{x_\jmath}) \qquad (2.18)$$

де P - безліч - наші вхідні дані, | P |
- потужність документа- кількість
рядків, x_{ia} - значення і-го
параметра в рядку а, $\bar{x}_\text{і}$ - середнє
значення і-го параметра.

В основі QR-алгоритму лежить уявлення матриці у вигляді A = QR, де Q - ортогональна матриця, а R - верхня трикутна. Перетворення Хаусхолдера здійснюється з використанням матриці Хаусхолдера, що має такий вигляд:

$$H = E - \frac{2}{v^T v} v v^T \qquad (2.19)$$

де v- довільний ненульовий вектор-стовпець,

E - одинична матриця,

vv^T - квадратна матриця того ж розміру.

Розглянемо докладніше реалізацію даного процесу

Покладемо A_0 = A і побудуємо перетворення Хаусхолдера H_1 (A_1 = $H_1 A_0$).

Матриця Хаусхолдера обчислюється згідно:

$$H_1 = E - \frac{2}{v^{1T} v^1} v^1 v^{1T} \quad (2.20)$$

$$v_1^1 = a_{11}^0 + sign(a_{11}^0)(\sum_{j=1}^{n}(a_{j1}^0)^2)^{\frac{1}{2}} \quad (2.21)$$

$$v_i^1 = a_{j1}^0, i = \overline{2,n} \quad (2.22)$$

На наступному, другому, етапі будується перетворення Хаухолдера H_2 (A_2 =$H_2 A_1$)

.

$$\boldsymbol{v_i^2} = 0 \quad (2.23)$$

$$v_2^2 = a_{22}^1 + sign(a_{22}^1)(\sum_{j=2}^{n}(a_{j2}^0)^2)^{\frac{1}{2}} \quad (2.24)$$

$$v_i^2 = a_{j1}^0, i = \overline{3,n} \quad (2.25)$$

Повторюючи процес n - 1 раз, отримаємо шукане розкладання A = QR, де

$$Q = (H_{n-1} H_{n-2} \dots H_0) = H_1 H_2 \dots H_{n-1}, R = A_{n-1} \quad (2.26)$$

Процедура QR - розкладання використовується в QR - алгоритмі обчислення власних значень. Будується наступний ітераційний процес:

Кожному матеріальному своїм значенням буде відповідати стовпець зі прагнуть до нуля поддіагональнимі елементами і як критерій збіжності ітераційного процесу для таких власних значень можна використовувати:

$$(\sum_{i=m+1}^{n}(a_{im}^{(k)})^2)^{\frac{1}{2}} \leq \varepsilon \qquad (2.27)$$

Тепер необхідно серед усіх власних значень (головних компонент) знайти ті, які є найбільш значущими в наборі вхідних даних. Для цього, як і було сказано раніше, використовується метод зламаною тростини.

2.6.3 NIPALS

Для ефективної і правильної роботи даного алгоритму необхідно провести попередню обробку даних, яку називають автомасштабованість. Дана обробка включає в себе комбінацію центрування і стандартизації набору вхідних даних.

Під стандартизацією даних розуміється приведення набору даних до певного виду для подальшого коректного порівняння їх і проведення подальшого їх аналізу. Стандартизація була зроблена ще при реалізації першого алгоритму МГК, де були перетворені всі значення змінних в чисельний вид для можливості подальшої обробки вхідного набору.

Під центруванням розуміється перетворення значень змінних до такого виду, в якому середнє арифметичне даної змінної дорівнюватиме нулю:

$$X_{ij} = X_{ij} - \bar{X}_i \quad (2.28)$$

де X- вихідна матриця вхідних даних,

$\bar{X}_i$- середнє значення –і ої змінної (j-ий стовпець).

Даний метод включає в себе розкладання матриці даних X на структурну частину і шумову частина [36]:

$$X = TP^T + E = Structure + Noise \quad (2.29)$$

де T - матриця рахунків (scores),

P - матриця навантажень (loadings),

E - залишкова матриця.

На рис. 2.5 представлено розкладання матриці вхідних даних на дві частини. В даному алгоритмі нас цікавить матриця рахунків і матриця навантажень, а саме те, що складається з суми матричного твори і залишкової матриці.

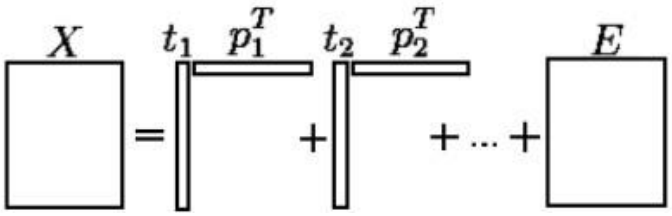

Рисунок 2.5 - Розкладання матриці вхідних даних

Максимальна кількість підсумовування матричного твори буде дорівнює кількості змінних у вхідному наборі даних. Тоді залишкова матриця буде дорівнює 0, так як при цьому не загубиться ніякої інформації. У такому варіанті немає сенсу, так як це просто розкладання

матриці, яка не допоможе скоротити розмірність з мінімальною втратою інформативності. Отже, необхідно щоб кількість таких творів було якомога менше кількості початкових змінних. При цьому інформативність структурної частини повинна бути якомога більше.

Залишкова матриця не є частиною моделі. Це буде частина матриці вхідних даних, що не пояснюється моделлю. Потому залишкова матриця не має бути «великий», що б означало, що ми видалили багато інформації. Це дасть нам погану нову модель, так як вона не буде добре описувати стару. Зрештою залишкова матриця буде показувати відмінність між новою і старою моделлю даних.

На основі залишкової матриці розраховувати залишки по об'єктах (рядках матриці) і по змінним (стовпцями матриці). Розглянемо залишки по об'єктах, тоді для кожного рядка розрахуємо:

$$e_i^2 = \sum_{j=1}^{n} e_{ij}^2 \qquad (2.30)$$

де і - кількість змінних (кількість стовпців).

Для порівняння нової та старої моделі краще використовувати загальне залишкове відмінність для всіх об'єктів одночасно:

$$e_{tot}^2 = \sum_{j=1}^{n} e_{ij}^2 \qquad (2.31)$$

Чим менше ця різниця, тим ближче моделі один до одного.
Алгоритм представляє з себе наступне:

п) для початку проводиться центрування матриці вхідних даних;

р) задаємо $E_{(0)} = X$;

с) t беремо як стовпець з X, $threshold$ = 0.00001 - для перевірки збіжності; Цикл (від 1 до кількості змінних):

1. Візьмемо проекцію X, щоб знайти відповідне навантаження:

$$p = \frac{(E^T{}_{(i=1)}t)}{t^T t} \qquad (2.32)$$

2. Нормалізуємо вектор навантажень:

$$p = p(p^T p)^{-0.5} \qquad (2.33)$$

3. Візьмемо проекцію, щоб знайти вектор рахунків p:

$$p = \frac{(E_{(i=1)}p)}{p^T p} \qquad (2.34)$$

4. Перевірити збіжність. Якщо різниця між значенням $Tnew = t^{\mathrm{T}}\mathrm{t}$ і T_{old} (з минулого ітерації) більше ніж $threshold$, то повертаємося на крок 1.
5. Видалити компоненту з $\mathrm{E}_{(\mathrm{i\text{-}1})}$:

$$\mathrm{E}_{(\mathrm{i})} = \mathrm{E}_{(\mathrm{i}-1)} - (\mathrm{tp}^{\mathrm{T}}) \qquad (2.35)$$

Для вирішення поставленого нами завдання, необхідно трохи змінити алгоритм.

У вихідному алгоритмі вибирається параметр, який буде видалений з вхідних даних і матриця перетвориться так, щоб при цьому вона не втратила інформативності. Далі вибирається наступний параметр і відбувається те ж саме.

У нашому випадку, на кожному етапі розглядається значимість кожного параметра щодо всіх вхідних даних і віддаляється той, у якого

найменша інформативність. В результаті виходить набір із значущих параметрів в порядку видалення. Кількість таких значущих параметрів виділялося на основі практичних досліджень в WEKA.

2.6.4 Пошук ортогональних проекцій з найбільшим розсіюванням

Суть методу полягає в пошуку ортогонального перетворення матриці вхідних даних в нову систему координат, що задовольняє певні умови.

Для початку дані центруються. Для знаходження проекції на головні компоненти скористаємося наступною формулою [37]:

$$Y = AX \qquad (2.36)$$

де Y - проекція вихідних даних на головні компоненти;

A - матриця перетворення;

X - вихідна матриця даних.

Побудова матриці відбувається наступним чином:

т) будується ковариационная матриця для вихідних даних;

у) вираховуються власні значення і власні вектори ковариационной матриці. Власні значення будемо шукати за допомогою перетворення Хаусхолдера і QR-алгоритму, як робилося в першому методі. Отримавши власні значення, знайдемо власні вектори за методом зворотної ітерації [38]:

$$(X - \lambda E)x = b$$

де λ - відповідне власне значення, вирахувана раніше;

E - одинична матриця;

x - власний вектор, який ми хочемо отримати;

b - вектор, обраний на удачу.

знайдемо x, вирішуючи наступне рівняння:

$$x = (X - \lambda E)^{-1} b$$

$$C^{-1} = \frac{1}{|C|} C^T \qquad (2.37)$$

де $|C|$ - визначник матриці;

C^T - транспонована матриця.

Отримавши матрицю перетворень A, Розрахуємо матрицю проекцій на головні компоненти. Власні вектори симетричною матриці ортогональні один одному. Ортогональні перетворення відповідають ортогональним матрицями. матриця перетворенняAє ортогональною. отже, A ортогональне перетворення. Стовпці і рядки ортогональної матриці ортогональні. Отже, нова матриця задовольняє заданим умовам і можна шукати вибіркову дисперсію.

В отриманій матриці знайдемо максимальну вибіркову дисперсію, яка буде першою головною компонентою, друга максимальна дисперсія буде другою головною компонентою і т.д.

Кількість значущих параметрів оцінюється за правилом Кайзера, де головна компонента залишається, якщо власне число більше середньої вибіркової дисперсії за координатами.

2.6.5 Оцінка отриманого набору значущих параметрів

За допомогою методів, розглянутих раніше, були отримані набори значущих параметрів. Щоб порівняти їх ефективність, розрахуємо помилки першого і другого роду і порівняємо з уже існуючими значеннями.

За методом анулювання кореляції між координатами виходять наступні результати таблиця 2.8.

Таблиця 2.8 Результати аналізу набору даних за методом анулювання кореляції між координатами

Тип атаки: параметри	Помилка першого роду	Помилка другого роду
DoS: 1,2,3,4,5,6,7,25,35	0,12%	0,48%
Probe: 3,4,5,6,10,11	0,83%	0,45%
U2R: 1,2,3,4,5,6,9	-	5,2%
R2L: 1,3,4,5,6,10	0,8%	3,5%

Для алгоритму NIPALS в спочатку представленому вигляді результати погані, та як він відкидає тільки ті параметри, які явно не є значущими, так як їх значення у всіх стану приймають одне і теж значення. Для прикладу, наведемо отримані результати даного алгоритму: З R2L - прибираються параметри 7,8,20 і 21; з U2R - 7,8,15,20,21,22,26,31 і 38; з Probe -7,8,9,14,15,18,19,20 і 21 і т.д.

Змінивши трохи алгоритм, при якому ми за один прохід матриці прибираємо тільки одну компоненту з найменшою різницею збіжності, вийшли результати краще. Результат видавався у вигляді списку компонент в порядку віддалення їх з матриці через найменшу інформативності серед інших.

Кількість значущих змінних підбиралось вручну, так як визначити його не можна.

Таблиця 2.9 Результати аналізу набору даних за методом NIPALS

Тип атаки: параметри	Помилка першого роду	Помилка другого роду
DoS: 12,13,25,26,27,30,34,35,36,38,39,40,41	1,6%	2,7%
Probe: 2,27,28,29,30,31,34,35,36,37,38,40,41	0,95%	1,4%
U2R: 2,9,11,12,19,29,34,36,37,41	9,6%	8,6%
R2L: 2,9,12,13,14,16,17,18,19,29,37	0,5%	12%

В результаті розбору алгоритму зроблений висновок, що даний метод не підходить для знаходження значущих параметрів, так як він просто скорочує розмір вхідних даних за рахунок перетворення матриці і видалення з неї компонент в порядку яке буде задано.

При використанні методу пошук ортогональних проекцій з найбільшим розсіюванням вийшов набір даних, представлений в таблиці 2.10.

Таблиця 2.10 Результати аналізу набору даних за методом пошуку ортогональних проекцій з найбільшим розсіюванням

Тип атаки: параметри	Помилка першого роду	Помилка другого роду
DoS: 1,2,3,4,5,6,7,8,10,13	0,34%	0,9%
Probe: 6,7,8,13,14,17,18,19,20	1,2%	16%
U2R: 1,2,3,4,5,6,14	-	5,1%
R2L: 1,2,3,4,5,6,19	0,6%	4%

Для перевірки виконаної роботи необхідно порівняти помилки першого і другого роду існуючих наборів і знайдених нами і представимо їх у таблиці 2.11.

Таблиця 2.11 - Результати аналізу набору даних за методом пошуку ортогональних проекцій з найбільшим розсіюванням

R2L	U2R	Probe	DoS	Тип атаки	
1,4%	3,8%	0,9%	0,34%	Помилка першого роду	Класс1
8%	3,5%	1,3%	0,35%	Помилка другого роду	
0,7%	3,8%	1,3%	0,27%	Помилка першого роду	Класс2
4%	10,3%	2,3%	0,2%	Помилка другого роду	
1%	3,8%	0,46%	0,19%	Помилка першого роду	Класс3
6,2%	8,6%	0,5%	0,37%	Помилка другого роду	
2%	13,5%	0,7%	2,8%	Помилка першого роду	Класс4
9,4%	5,2%	2,9%	11,4%	Помилка другого роду	
0,8%	-	0,83%	0,12%	Помилка першого роду	Анулювання кореляції
3,5%	5,2%	0,45%	0,48%	Помилка другого роду	
0,5%	9,6%	0,95%	1,6%	Помилка першого роду	NIPALS
12%	8,6%	1,4%	2,7%	Помилка другого роду	
0,6%	-	1,2%	0,34%	Помилка першого роду	Ортогональні проекції з найбільшим розсіювання
4%	5,1%	16%	0,9%	Помилка другого роду	

З представлених результатів можна зробити наступні висновки. Для виявлення DoS атак краще використовувати параметри, які були отримані за

допомогою методу анулювання кореляції між координатами. Для виявлення Probe атак кращий результат дали параметри з інших робіт, але в підсумковому наборі за методом головних компонент представлений набір параметрів з анулювання кореляції між координатами. Для виявлення U2R атак - метод пошуку ортогональних проекцій з найбільшим розсіюванням. Для виявлення R2L - також метод пошуку ортогональних проекцій з найбільшим розсіюванням.

2.7 Опис алгоритму Дерева Хефдінга

Алгоритм навчання простий. Кожен лист зберігає статистику по частині потоку, це досягається підрахунком двох кращих атрибутів, відповідних критерію розбиття. Нехай ΔG - різниця значень функції, що представляє критерій розбиття, від цих двох атрибутів. Нехай ε - величина, що залежить від визначеного користувачем довірчого значення δ, яке зменшується з ростом числа оброблених елементів. Якщо $\Delta G > \varepsilon$, то кращих атрибут з двох розглянутих є найкращим для розбиття вузла. Кордон Хефдінга гарантує що цей вибір правильний з ймовірністю більше ніж 1 - δ .

Псевдокод процедури навчання елемента наведено на рис. 2.6 [40].

Спочатку для чергового елемента знаходиться потрібний лист (рядок 1). Далі в цьому листі оновлюється статистика (рядок 2). У нашому випадку статистикою є розподіл класів для кожного значення кожного атрибута. На практиці зазвичай заводять лічильники n_{ijk} для кожного атрибута i, значення j та класу k. Алгоритм так само інкрементує кількість примірників ((n_l), розглянутих в цьому вузлі, грунтуючись на функції ваги екземпляра.

Algorithm 1 HoeffdingTreeInduction(X, HT, δ)

Require: X, a labeled training instance.
Require: HT, the current decision tree.
1: Use HT to sort X into a leaf l
2: Update sufficient statistic in l
3: Increment n_l, the number of instances seen at l
4: **if** n_l mod $n_{min} = 0$ **and** not all instances seen at l belong to the same class **then**
5: For each attribute, compute $\overline{G}_l(X_i)$
6: Let X_a be the attribute with highest $\overline{G}_l$
7: Let X_b be the attribute with second highest $\overline{G}_l$
8: Compute the Hoeffding bound $\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n_l}}$
9: **if** $X_a \neq X_\emptyset$ **and** $(\overline{G}_l(X_a) - \overline{G}_l(X_b) > \epsilon$ **or** $\epsilon < \tau)$ **then**
10: Replace l with an internal node branching on X_a
11: **for all** branches of the split **do**
12: Add a new leaf with derived sufficient statistic from the split node
13: **end for**
14: **end if**
15: **end if**

Рисунок 2.6 - Псевдокод процедури навчання елемента VFDT

Окремо взятий екземпляр слабо змінює розподіл класів, тому алгоритм намагається збільшувати дерево тільки після певного числа переглянутих в даному вузлі елементів (n_{min}). Також алгоритм не викликає зростання дерева, якщо всі екземпляри, які досягли даного листа, належать одному і тому ж класу. (Рядок 4).

Для того щоб збільшити дерево, алгоритм намагається знайти відповідний атрибут для розбиття аркуша. Алгоритм підраховує для кожного атрибута критерій розбиття $\overline{G_l}(X_i)$ (Рядок 5). Цей критерій може бути функцією теоретичної інформатики, наприклад, ентропія або критерій інформативності, який підраховується, використовуючи лічильники n_{ijk}. Алгоритм так само обчислює значення критерію для випадку, коли не потрібно проводити ніякого розбиття $X_\emptyset$.

Далі алгоритм вибирає два кращих атрибута X_a та X_b, грунтуючись на значеннях критерію (рядки 6 і 7), обчислює різницю значень критерію розбиття $\Delta\bar{G}_l = \bar{G}_l(X_a) - \bar{G}_l(X_b)$ і приймає рішення про необхідність розбиття шляхом порівняння $\Delta\bar{G}_l$ з кордоном Хефдінга з поточним довірчим параметром δ. Якщо різниця значень критерію розбиття більше значення кордону Хефдінга, то атрибут X_a є найкращим атрибутом для розбиття з часткою ймовірності 1 - δ.

Рядок 9 відображає підсумкову умову розбиття аркуша. Якщо кращий атрибут є штучним атрибутом, доданим для випадку відсутності розбиття ($X_\emptyset$), алгоритм не виконує розбиття. Алгоритм так само використовує дискретний механізм τ, для випадку, якщо різниця значення критерію поділу між X_a та X_b занадто мала. Якщо значення кордону Хефдінга стає менше ніж τ ($\Delta\bar{G}_l < \varepsilon < \tau$), тоді поточний кращий атрибут вибирається незалежно від значення функції $\Delta\bar{G}_l$.

Якщо алгоритм розбиває вузол, він замінює лист l внутрішнім вузлом. Також він створює гілки, що ведуть до нових листів, базуючись на кращому атрибуті, і ініціалізує ці листи, використовуючи розподіл класів, отриманий з використанням кращого атрибута. (Рядки 10-13).

2.7.1 Варіативні дерева Хефдінга

В процесі тренування моделі потокових даних важливо розуміти обмеження пам'яті і обробляти дані так швидко, як це можливо. Дерева Хефдінга досягають цього шляхом накопичення достатньої статистики в вузлах до точки, де вона може бути використана для прийняття рішення про розбивку. Рішення це буде практично таким же надійним, як рішення, виконане алгоритмом, який переглянув весь набір даних.

Наступним кроком поліпшення базового алгоритму в машинному навчанні є реалізація ансамблі дерев Хефдінга використовуючи такі

техніки, як Bagging і Boosting. Зокрема, такий підхід дозволяє проблеми властиві жадібним алгоритмам. Навчальні дерева мають обмежену видимість (кращий результат одного тесту залежить від дітей, до яких звернеться алгоритм в процесі тесту) і такий підхід може призвести до погіршення якості моделі. Це пов'язано з вибором конкретного атрибута для поділу, в той час як оцінка інших атрибутів теж може бути придатна для розбиття, однак не використовується. Атрибут обраний на конкретному кроці алгоритму може бути кращим рішенням на даний момент, однак бути зовсім не репрезентативним в довгостроковій перспективі. Отже, потрібні інші варіанти алгоритму, що додають деревам більше інформації і варіативності в ухваленні рішення.

Одним із способів збільшення якості моделі є використання ансамблю класифікаторів, кожен з яких обробляє дані за своїми правилами (обробляє тільки деяку інформацію або ж робить це в іншій послідовності). Однак, такий підхід показав порівняно невеликий приріст в точності класифікації, в той час як споживання пам'яті моделлю значно збільшилося.

Іншим підходом є використання варіаційних (варіативних) дерев. Варіаційні дерева є чимось середнім між одним деревом і ансамблями. Вони здатні видавати додаткову корисну модель без споживання занадто великої кількості ресурсів.

Варіаційні дерева складаються з єдиної структури, яка ефективно представляє безліч дерев. Конкретний екземпляр даних може переміщатися в ході роботи алгоритму вниз по дереву, на кожному кроці переходячи в декількох напрямках або в гілку основного дерева, або в спеціальні додаткові гілки. Клас тестового екземпляра визначається на основі всіх досягнутих елементом листя.

На рис. 2.7 [41] наведено приклад декількох верхніх рівнів варіаційного дерева. Дерево схоже по структурі зі звичайним вирішальним

деревом, однак в ньому присутні додаткові вузли, позначені на рисунку прямокутниками

.

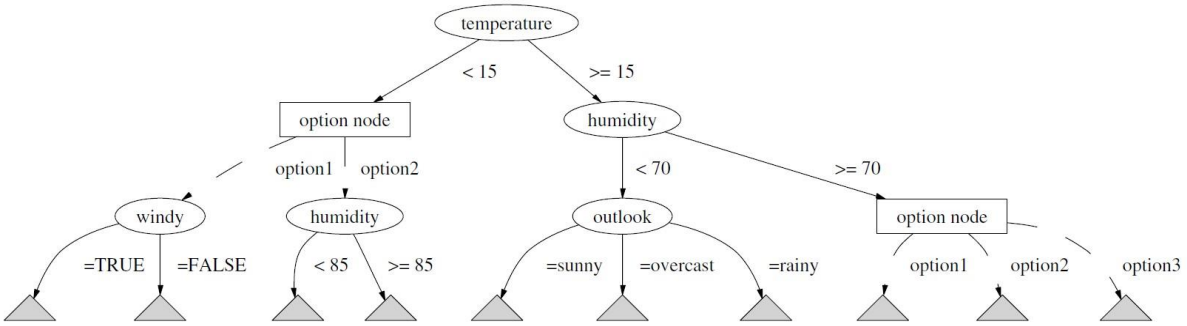

Рисунок 2.7 - Структура варіаційного дерева [12]

На рисунку 2.8 [41] описаний псевдокод процедури побудови варіаційного дерева Хефдінга. Цей алгоритм є модифікацією алгоритму дерева Хефдінга, що використовує структуру варіаційних дерев.

Рядки 19-34 є модифікацією базового алгоритму побудови дерева Хефдінга, яка додає можливість оперувати з додатковими вузлами. Інша відмінність в алгоритмах ми можемо бачити в рядках 3 і 4, де примірнику ставиться у відповідність не один вузол, а декілька. Як і в минулій версії, критерієм виміру G для всіх тестових екземплярів є коефіцієнт втрат.

Передбачення класу примірника робиться комбінуванням пророкувань всього листя, що відносяться до нього. Найчастіше вибір класу проводиться голосуванням або зваженим голосуванням.

```
1: Let HOT be an option tree with a single leaf (the root)
2: for all training examples do
3:    Sort example into option nodes L using HOT
4:    for all option nodes l of the set L do
5:       Update sufficient statistics in l
6:       Increment n_l, the number of examples seen at l
7:       if n_l mod n_min = 0 and examples seen at l not all of same class then
8:          if l has no children then
9:             Compute G̅_l(X_i) for each attribute
10:            Let X_a be attribute with highest G̅_l
11:            Let X_b be attribute with second-highest G̅_l
12:            Compute Hoeffding bound ε = sqrt(R²ln(1/δ) / 2n_l)
13:            if G̅_l(X_a) − G̅_l(X_b) > ε or ε < τ then
14:               Add a node below l that splits on X_a
15:               for all branches of the split do
16:                  Add a new option leaf with initialized sufficient statistics
17:               end for
18:            end if
19:         else
20:            if l.optionCount < maxOptions then
21:               Compute G̅_l(X_i) for existing splits and (non-used) attributes
22:               Let S be existing child split with highest G̅_l
23:               Let X be (non-used) attribute with highest G̅_l
24:               Compute Hoeffding bound ε = sqrt(R²ln(1/δ') / 2n_l)
25:               if G̅_l(X) − G̅_l(S) > ε then
26:                  Add an additional child option to l that splits on X
27:                  for all branches of the split do
28:                     Add a new option leaf with initialized sufficient statistics
29:                  end for
30:               end if
31:            else
32:               Remove attribute statistics stored at l
33:            end if
34:         end if
35:      end if
36:   end for
37: end for
```

Рисунок 2.8 - Псевдокод процедури побудови варіаційного дерева

2.7.2 Обмеження зростання дерева

Варіаційні дерева мають тенденцію швидко збільшуватися в розмірах, якщо це не контролювати. Bernhard Pfahringer в своїй роботі [42] використовує стратегію обмеження кількості варіантів для одного примірника. У цій же роботі пропонується оптимальне число варіантів для дерева, отримане в ході експериментів. На рисунку 2.9 видно, що точність алгоритму перестає сильно зростати після досягнення межі в 5 варіантів для екземпляра.

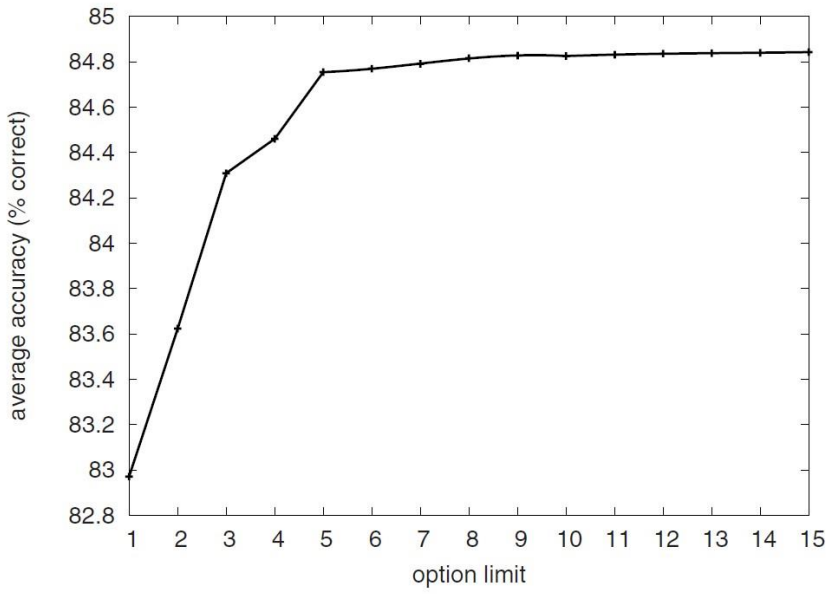

Рисунок 2.9 - Співвідношення обмеження кількості варіантів для екземпляра до точності прогнозів алгоритму

Щоб створити нову структуру в додатковому вузлі, ми будемо вибирати для нього тільки ті атрибути, які не були відібрані в базовому вузлі. Такий підхід дозволить зменшити прагнення дерева рости і подолати проблему створення безлічі однакових за своєю суттю вузлів на одному кроці.

Останнім кроком управління ростом дерева є визначення конкурентоспроможності додаткових варіантів у порівнянні з існуючими поділами. Параметр δ контролює допустиму та помилку алгоритму. Чим менше це значення, тим довше дерево буде чекати перед тим як рости. Також введемо другий параметр δ' для вирішення того, коли додавати додатковий вузол нижче того вузла, який вже був розбитий. Новий вузол може бути доданий, якщо кращий невикористаний атрибут виглядає репрезентативніше, ніж поточний кращий додатковий вузол з точки зору критерію втрат і границі Хефдінга з довірчою величиною δ'.

2.8 Висновки до розділу

Визначено завдання, що виникають в процесі аналізу даних, і деякі підходи до їх вирішення.

Так само в даному розділі для проведення експериментів був обраний програмний продукт WEKA

З представлених результатів можна зробити наступні висновки. Для виявлення DoS атак краще використовувати параметри, які були отримані за допомогою методу анулювання кореляції між координатами. Для виявлення Probe атак кращий результат дали параметри з інших робіт, але в підсумковому наборі за методом головних компонент представлений набір параметрів з анулювання кореляції між координатами. Для виявлення U2R атак - метод пошуку ортогональних проекцій з найбільшим розсіюванням. Для виявлення R2L - також метод пошуку ортогональних проекцій з найбільшим розсіюванням.

Для подальшої реалізації проекту були запропоновано досліджувати метод класифікації найближчих сусідів і дерева рішень Хефдінга

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОЕКТА

В якості платформи реалізації алгоритмів вибран Massive Online Analysis (MOA). Даний фреймворк є програмним середовищем з відкритим вихідним кодом, призначеним для реалізації, застосування та запуску експериментів інтелектуального аналізу даних, що надходять з потоку.

MOA пов'язана з бібліотекою інтелектуального аналізу WEKA. Дана платформа реалізована засобами мови Java

Повний вихідний код проекту без урахування коду класифікаторів представлений в додатку А.

3.1 Структура проекту

Структурно проект розділений на 3 пакети вихідних кодів.

Перший пакет, BachelorClassifiers, містить в собі реалізації обраних алгоритмів класифікації.

Другий пакет, BachelorEvaluation, містить в собі реалізацію класу, що оцінює ефективність роботи алгоритмів.

Третій пакет є головним і містить в собі реалізацію тестової програми (TestClass), яка застосовує класи з інших пакетів для оцінювання ефективності вирішення задачі класифікації на оброблюваному наборі даних.

Структура проекту наведена на рис 3.1.

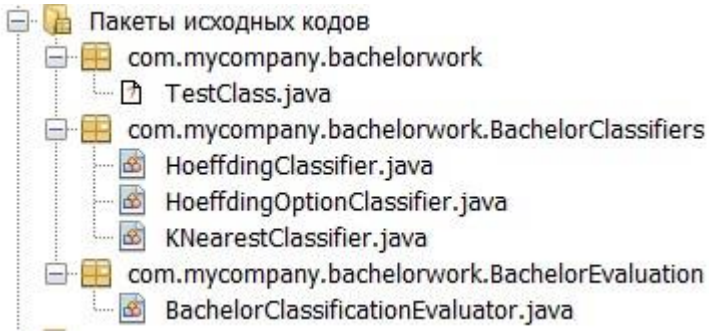

Рисунок 3.1 - Структура проекту

3.2 Реалізація обраних для дослідження алгоритмів

При реалізації класифікаторів засобами платформи MOA будемо керуватися наступним алгоритмом:

а) створити клас, який реалізує інтерфейс AbstarctClassifier бібліотеки moa.classifiers, і описати в ньому особливі структури даних, властиві конкретному алгоритму;

б) визначити параметри алгоритму, доступні користувачеві. Для того щоб система автоматично перехоплювала ці параметри, необхідно, щоб; вони були public членами класу, який наслідує тип moa.options.Option.

в) далі необхідно перевизначити метод resetLearningImpl. Даний метод викликається при ініціалізації моделі. У ньому ініціалізуються всі необхідні для навчання моделі структури;

г) наступним кроком буде перевизначення методу trainOnInstanceImpl (Instance inst). Це головний метод навчального алгоритму, що викликається для кожного тренувального примірника;

ґ) і нарешті необхідно перевизначити метод getVotesForInstance (Instance inst). Даний метод використовується для передбачення класу надійшов елемента.

3.2.1 Метод найближчих сусідів

На першому кроці реалізації визначимо структуру класифікатора KNearestClassifier.

Структурно класифікатор містить в собі змінну window типу weka.core.Instances, що містить наше плаваюче вікно.

Далі визначимо власні параметри алгоритму. У разі методу найближчих сусідів, такими параметрами є:

а) kOption - параметр k. кількість найближчих сусідів, які використовуються для прийняття рішення про клас елемента;

б) limitOption - максимальне число елементів потоку, що зберігаються класифікатором в змінній window;

в) nearestNeighbourSearchOption - спосіб пошуку найближчих сусідів серед примірників плаваючого вікна. Тут будемо використовувати 2 наданих MOA способи пошуку найближчих сусідів: Лінійний пошук і пошук з побудовою кмерного дерева;

Тренувальний метод даного алгоритму простий. На кожному кроці тренування одержаний примірник додається в плаваюче вікно. Якщо розмір вікна перевищив допустимий розмір вікна, найбільш старий елемент вікна видаляється.

Метод getVotesForInstance містить в собі застосування обраного методу пошуку найближчих сусідів, а також підрахунок кількості примірників різних класів і повертає результат голосування по класах.

3.2.2 Дерево Хефдінга

Для початку опишемо і реалізуємо структуру дерева, яку будемо використовувати. Структура дерева приведена на рис. 3.2.

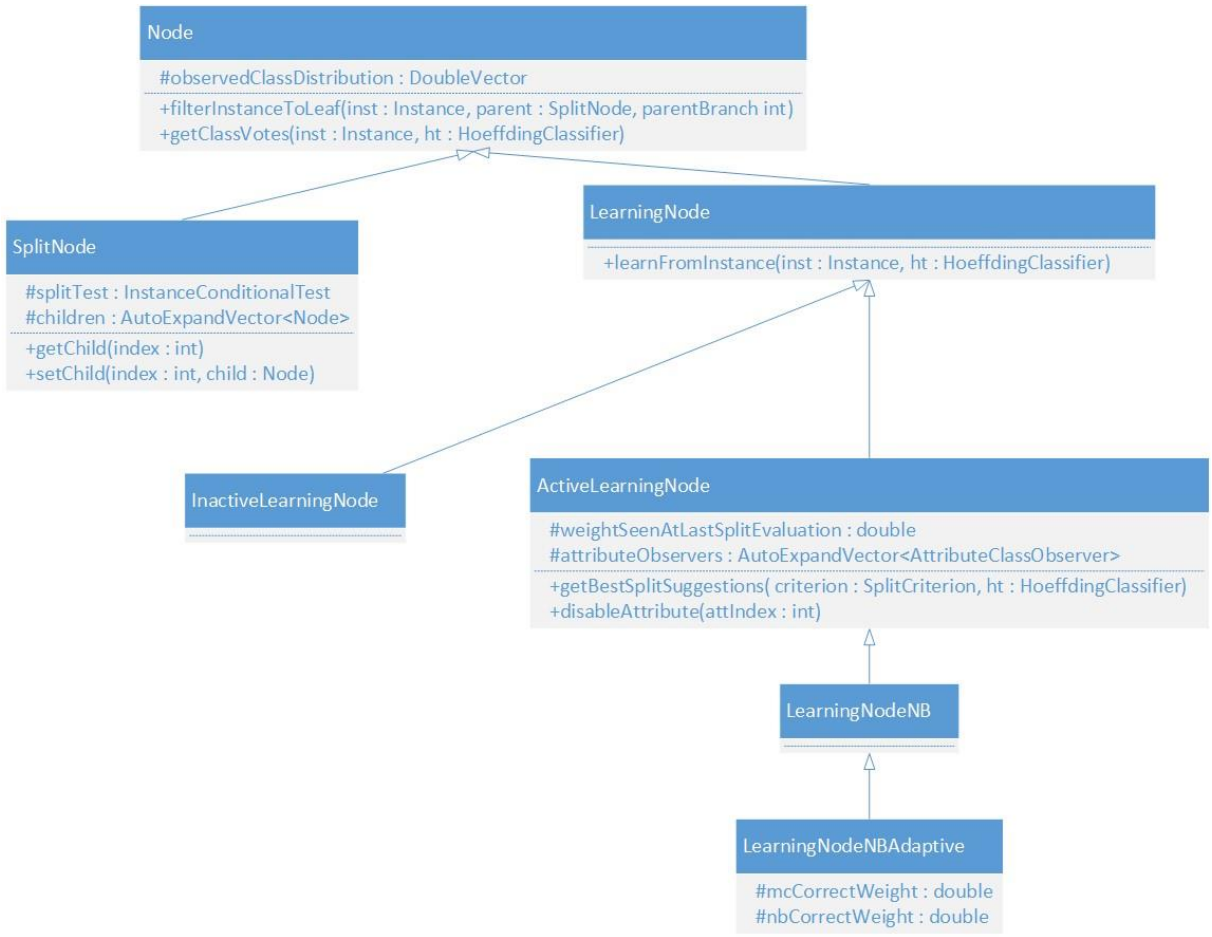

Рисунок 3.2 - UML діаграма дерева Хефдінга

Структурно класифікатор містить посилання на корінь дерева типу Node, яке містить поточний вузол дерева.

Далі визначимо власні параметри алгоритму. У разі дерев Хефдінга такими параметрами є:

а) gracePeriodOption - кількість примірників, оброблених в листі, після якого можна дробити лист;

б) splitConfidenceOption - довірче значення δ, описане вище;

в) tieThresholdOption - значення величини τ, що використовується в прийнятті рішення про поділ;

г) binarySplitsOption - флаг, який встановлюється, якщо необхідно тільки бінарне дерево;

г) removePoorAttsOption - флаг, який встановлюється, якщо перестати вважати статистику по атрибутам, що не задовольняє умові Хефдінга, з метою економії пам'яті;

д) noPrePruneOption - флаг, який встановлюється, якщо розглядаємо рішення «не розбивати вузол» як ще одне рішення розбиття;

е) leafpredictionOption - варіант передбачення класу примірника в вузлах на основі статистики. Приймає три значення: MC - перемагає переважний клас. NB - рішення приймається на основі байєсівської ймовірності, NBAdaptive - лист вибирає той спосіб, який менше помилявся під час навчання;

є) nbTresholdOption - кількість елементів, яку має переглянути вузол, щоб почати застосовувати байесівський спосіб вирішення.

Тренувальний метод реалізуємо, керуючись псевдокодом. Для кожного елемента знаходимо відповідний йому лист, якщо він може дробитися, запускаємо процедуру перевірки умов дроблення і поділяємо. З метою економії пам'яті забороняємо вузлам нижче даного вважати статистику по атрибутах, які не задовольняють умові кордону Хефдінга. Якщо під час процедури вирішення про дроблення переміг варіант «Не дробити вузол далі», деактивуємо лист, забороняючи йому створювати нащадків надалі.

Метод отримання результату передбачення для тестового екземпляра (getVotesForInstance) виконує пошук листа для екземпляра, а потім отримує результат передбачення способом, обраним параметром leafpredictionOption.

3.2.3 Варіативне дерево Хефдінга

Структура дерева, використовуваного оновленим алгоритмом відрізняється від базової. Це пов'язано з тим, що тепер внутрішні вузли

мають посилання на додаткові вузли, а також кількість додаткових вузлів в них і методи оновлення цього числа в нащадках.

Схема варіаційного дерева Хефдінга приведена на рис. 3.3.

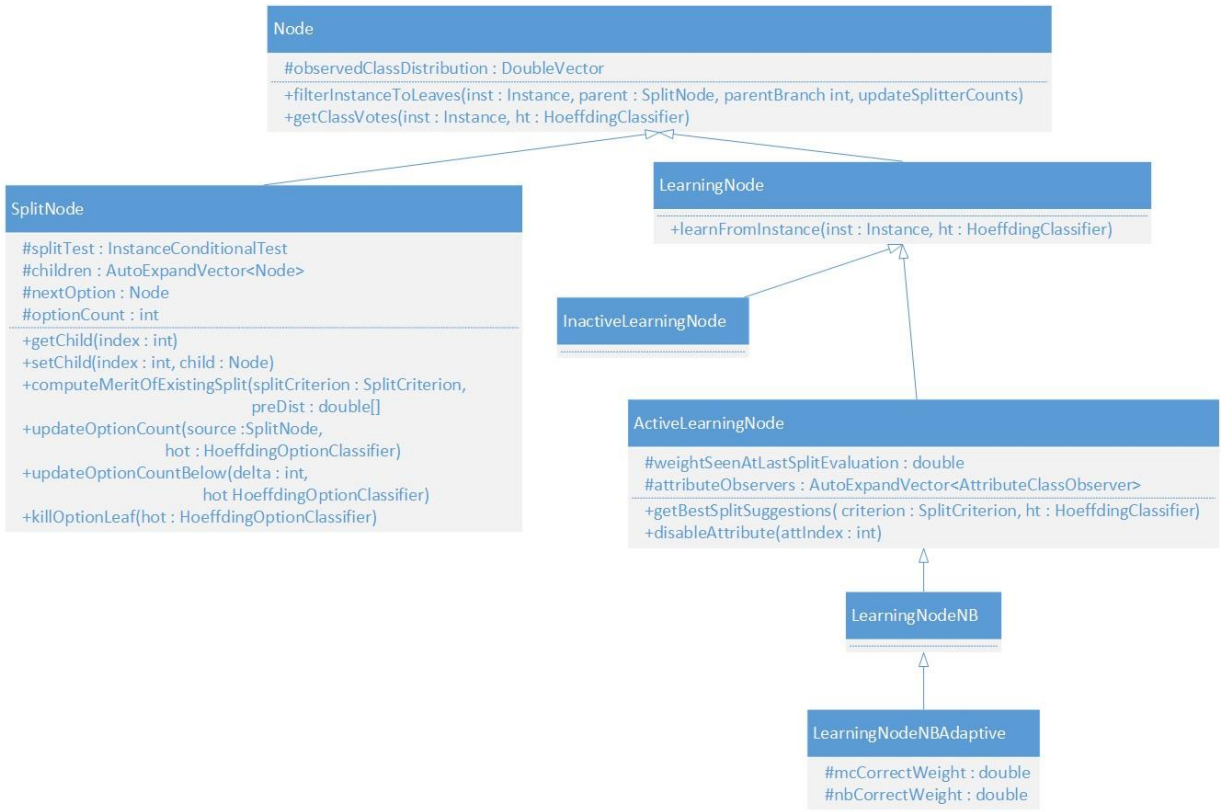

Рисунок 3.3 - UML-діаграма варіаційного дерева Хефдінга

Також необхідно додати кілька користувальницьких параметрів:

а) secondarySplitConfidenceOption - довірче значення δ ', описане вище;

б) maxOptionPathsOption - максимальне число альтернативних варіантів для одного вузла. Значення за замовчуванням 5.

Тренувальний метод реалізуємо, керуючись псевдокодом. Для кожного елемента знаходимо відповідні йому листи, якщо він може дробитися, запускаємо процедуру перевірки умов дроблення і поділяємо. Якщо під час процедури вирішення про дроблення переміг варіант «Не дробити вузол далі», деактивуємо лист, забороняючи йому створювати нащадків надалі. До процедури перевірки умов і дроблення додамо код пов'язаний з додатковими

вузлами. При кожній операції дроблення, зберігаємо початковий лист як альтернативний якщо не досягнута межа альтернативних вузлів.

3.3 Висновки до розділу

В даному розділі були розроблені структура проекту.

Для реалізації класифікаторів засобами платформи MOA було запропоновано алгоритм події:

Реалізовано алгоритми методу найближчих сусідів, дерева Хефдінга а також варіативного дерева Хефдінга

Побудовано UML діаграма дерева Хефдінга і UML-діаграма варіаційного дерева Хефдінга

РОЗДІЛ 4
ТЕСТУВАННЯ І АНАЛІЗ РЕЗУЛЬТАТІВ

4.1 Підготовка вхідних даних

В якості набору даних для тестування будемо використовувати матеріали третього змагання в області вилучення знань і засобів аналізу даних, що проводиться в рамках п'ятої міжнародної конференції вилучення знань і аналізу даних (KDD-99) [42]. Основним завданням змагання було розробити детектор вторгнень в мережеву інфраструктуру, прогностичну модель, здатну розрізняти «погані» підключення, звані вторгненнями або атаками і «хороші» звичайні підключення. Даний набір даних містить дані аудиту, що містять широкий спектр симульованих атак у військовій мережевій інфраструктурі.

4.1.1 Опис завдання

Необхідно розробити програму, здатну виявляти вторгнення з боку неавторизованих користувачів і, можливо, співробітників власної компанії, в мережеву інфраструктуру з метою її захисту. Завдання розробки детектора вторгнень - то завдання проектування прогностичної моделі (наприклад, класифікатора) здатної розрізняти «погані» підключення, звані вторгненнями і «хороші» звичайні підключення.

Запропонований набір даних згенерований Lincoln Labs [43] в ході дев'ятиденного експерименту зі збору необроблених даних TCP в локальній мережі, що симулює типову локальну мережу, яка використовується військово-повітряними силами США. Дана мережа використовувалася як справжня мережа ВПС, проте піддавалася безлічі атак.

Необроблені дані входять до складу чотиригігабайтного стисненого TCP дампа семи тижнів мережевого трафіку. Вони представлені у вигляді п'яти мільйонів записів підключень. Аналогічно два тижні тестових даних представлені у вигляді двох мільйонів рядків підключень.

Підключення - це послідовність TCP пакетів, що починається і закінчується в певні моменти часу, між якими дані, використовуючи відомий протокол, переміщаються до і від IP-адреси призначення до цільової IP-адреси. Кожне підключення віднесено до класу нормальних підключень або ж до класу атак, з міткою про тип атаки.

Атаки розділені на 4 основні категорії (див.п.2.1):

Важливою частиною завдання є те, що тестові дані мають відмінний розподіл ймовірності атак в порівнянні з тренувальними, а також мають особливі типи атак, що не зустрічаються в тренувальному наборі. Даний підхід робить завдання більш реалістичним. Деякі експерти в області мережевої безпеки вважають, що більшість нових атак - це варіанти вже відомих атак і розуміння основ виявлення атак може бути досить для нових варіантів атак.

4.1.2 Структура набору даних

Існує набір високорівневих атрибутів, що дозволяє відрізнити нормальні підключення від атак. Існує кілька категорій таких атрибутів.

Перша категорія атрибутів відноситься до підрахунку статистики підключень, що відбулися за останні дві секунди, що мають такий же хост призначення або той же сервіс, що і зараз приєднані. Дана категорія атрибутів носить назву «атрибути трафіку, засновані на часі».

Деякі атаки зондування сканують хости (або порти) використовуючи більші проміжки часу, ніж дві секунди, наприклад, один раз в хвилину. Тому, рядки підключень були так само відсортовані по хосту призначення, і деякі атрибути були побудовані на основі вікна в 100 підключень замість

тимчасового вікна. Даний набір дозволив визначити набір так званих атрибутів трафіку, заснованих на хостах.

На відміну від більшості DOS-атак і атак зондування, атаки типів R2L і U2R носять випадковий характер і не мають яскраво вираженої тимчасової прояви. Це відбувається у зв'язку з тим, що DOS і зондувальні атаки тягнуть за собою множинні підключення до деяких хостів в короткий проміжок часу. Однак R2L і U2R атаки носять одиничний характер і пов'язані зі зміною частини даних в пакетах. З метою їх детектування, в набір даних були включені контент-атрибути, такі як кількість невдалих спроб виходу, що допомагають виявити підозрілу активність в середовищі.

Повний список атрибутів наведено в таблицях 4.1, 4.2 і 4.3.

Таблиця 4.1 - Базові атрибути TCP підключень

Ім'я атрибута	Опис	Характер даних
Duration	Тривалість (в секундах) підключення	Безперервний
Protocol_type	Тип використовуваного протоколу (наприклад, tcp, udp)	Дискретний
Service	Мережевий сервіс призначення (наприклад, http, telnet)	Дискретний
Src_bytes	Кількість байт від джерела до призначення	Безперервний
Dst_bytes	Кількість байт від призначення до джерела	Безперервний
Flag	Статус підключення (нормальне або з помилкою)	Дискретний
Land	1 якщо підключення використовує один і той же хост і порт, інакше - 0	Дискретний
Wrong_fragment	Кількість битих фрагментів	Безперервний
Urgent	Кількість термінових пакетів	Безперервний

Таблиця 4.2 - Контентні атрибути підключення

Ім'я атрибута	Опис	Характер даних
Hot	Кількість пакетів, зазначених маркером hot	Безперервний
Num_failed_logins	Кількість невдалих спроб входу	Безперервний
Logged_in	1 в разі успішного входу, 0 в зворотньому	Дискретний
Num_compromised	Число скомпрометованих умов	Безперервний
Root_shell	1, якщо отриманий доступ до суперкористувача	Дискретний
Su_attempted	1, якщо була спроба команди «su root»	Дискретний
Num_root	Кількість доступів як суперкористувач	Безперервний
Num_file_creations	Кількість створень файлів	Безперервний
Num_shells	Кількість разів, коли отримано доступ до командного рядку	Безперервний
Num_access_files	Кількість операцій, що мають привілей змінювати файли	Безперервний
Num_outbound_cmds	Кількість вихідних команд в ftp сесії	Безперервний
Is_hot_login	1 якщо сесія входу відноситься до списку «гарячих»	Дискретний
Is_guest_login	1 якщо сесія гостьова	Дискретний

Таблиця 4.3 - Атрибути трафіку, пораховані використовуючи двохсекундне тимчасове вікно.

Ім'я атрибута	Опис	Характер даних
Count	Кількість підключень до такого ж хосту в останні 2 секунди	Безперервний
Наступні атрибути відносяться до підключень з однаковими хостами		
Serror_rate	% підключень, що мають SYN помилки	Безперервний
Rerror_rate	% підключень, що мають REJ помилки	Безперервний
Same_srv_rate	% підключень, до того ж сервісу	Безперервний
Diff_srv_rate	% підключень до інших сервісів	Безперервний
Srv_count	Кількість підключень до того ж сервісу за останні 2 сеунди	Безперервний
Наступні атрибути відносяться до підключень з однаковими сервісами		
Srv_serror_rate	% підключень мають SYN помилки	Безперервний
Srv_rerror_rate	% підключень, що мають REJ помилки	Безперервний
Srv_diff_host_rate	% підключень до інших хостам	Безперервний

4.1.3 Процедура підготовки даних

Для того, щоб наша програма могла інтерпретувати дані, переведемо їх в формат arff.

Для генерації потоку даних з нашого файлу використовуємо клас бібліотеки MOA с назвою ArffFileStream.

4.2 Тестування використовуваних алгоритмів

Як згадувалося вище, процес тестування складається з періодичного вимірювання точності алгоритмів в процесі навчання. Як інтервал тестування виберемо 10 000 тренувальних примірників. В процесі тестування будемо підраховувати середній час тренування в мілісекундах, точність (accuracy)

передбачення алгоритму, а також Kappa-статистику, отриману шляхом тестування моделі на всьому тестовому наборі даних. В процесі тестування також будемо підраховувати середній час тестування в мілісекундах.

4.2.1 Метод найближчих сусідів

Метод найближчих сусідів був запущений з наступними параметрами:

а) кількість елементів у вікні для запам'ятовування – 1000;

б) кількість найближчих сусідів, що відбирається для голосування – 5;

Результати тимчасових оцінок алгоритму наведені в таблиці 4.4

Таблиця 4.4 - Результати тимчасових оцінок методу найближчих сусідів.

Параметр	Оцінка, мс
Середній час тренування 10000 примірників	104,13
Середній час тестування 500000 примірників	231542,89

Результат підрахунку точності і K-статистики в ході роботи алгоритму наведено на рис. 4.1 і 4.2 відповідно.

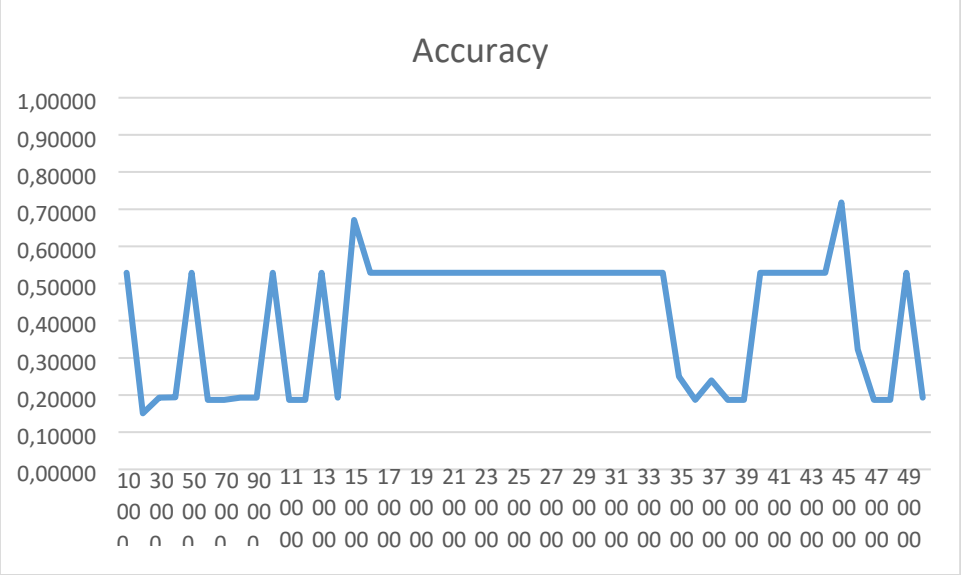

Рисунок 4.1 - Точність методу найближчих сусідів в залежності від обсягу тренувальних даних, отримана в ході експерименту

Рисунок 4.2 - K-статистика методу найближчих сусідів в залежності від обсягу тренувальних даних, отримана в ході експерименту

4.2.2 Дерева Хефдінга

Алгоритм VFDT був запущений з наступними параметрами:

а) gracePeriodOption - 200;

б) splitConfidenceOption - 0.0000001;

в) tieThresholdOption - 0.05;

г) leafpredictionOption - Використано усі 3 варіанти передбачення у листях;

ґ) nbTresholdOption – 0.

Результати тимчасових оцінок алгоритму для різних варіантів прийняття рішення про класифікацію у вузлі приведена в таблиці 4.5.

Таблиця 4.5 - Результати тимчасових оцінок алгоритму VFDT

Опція прийняття рішення в листі	Параметр	Оцінка, мс
Majority class	Середній час тренування 10000 примірників	87,68
	Середній час тестування 500000 примірників	1477,74
Naïve Bayes	Середній час тренування 10000 примірників	91,47
	Середній час тестування 500000 примірників	5875,26
Naïve Bayes Adaptive	Середній час тренування 10000 примірників	161,20
	Середній час тестування 500000 примірників	5842,91

Результат підрахунку точності і K-статистики в ході роботи алгоритму для всіх трьох опцій передбачення значень в листях наведено на рис. 4.3, 4.4, 4.5, 5.6, 4.7, 4.8 відповідно.

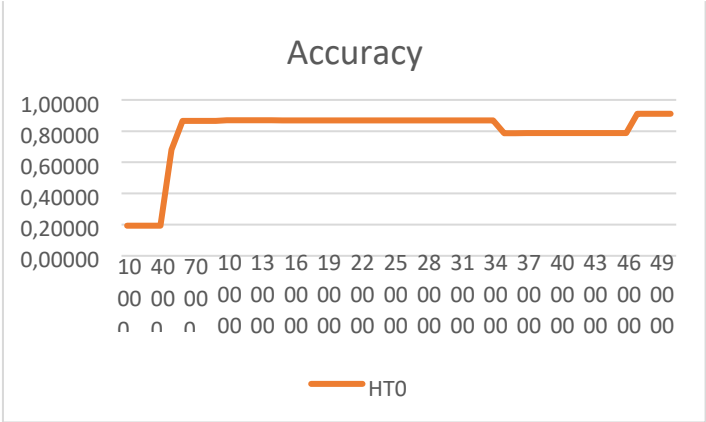

Рисунок 4.3 - Точність алгоритму VFDT з опцією Majority class в залежності від обсягу тренувальних даних, отримана в ході експерименту

Рисунок 4.4 - Карра статистика алгоритму VFDT з опцією Majority class в залежності від обсягу тренувальних даних, отримана в ході експерименту

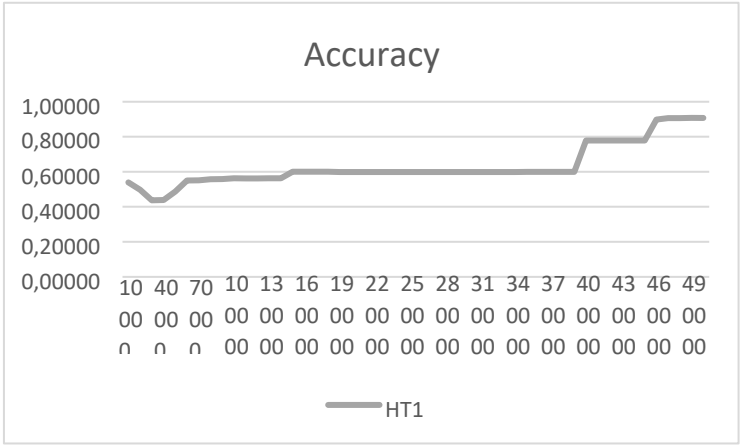

Рисунок 4.5 - Точність алгоритму VFDT з опцією Naive Bayes в залежності від обсягу тренувальних даних, отримана в ході експерименту

Рисунок 4.6 - Карра статистика VFDT з опцією Naive Bayes в залежності від обсягу тренувальних даних, отримана в ході експерименту

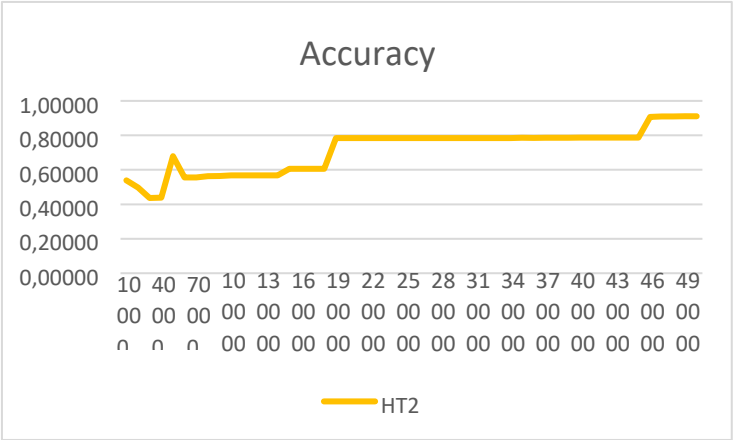

Рисунок 4.7 - Точність алгоритму VFDT з опцією Naive Bayes Adaptive в залежності від обсягу тренувальних даних, отримана в ході експерименту

Рисунок 4.8 - K-статистика VFDT з опцією Naive Bayes Adaptive в залежності від обсягу тренувальних даних, отримана в ході експерименту

4.2.3 Варіативні дерева Хефдінга

Алгоритм варіативного дерева Хефдінга був запущений з наступними параметрами:

а) gracePeriodOption - 200;

б) splitConfidenceOption - 0.0000001;

в) tieThresholdOption - 0.05;

г) leafpredictionOption - Використано усі 3 можливих варіанти передбачення в листях;

ґ) nbTresholdOption - 0;

д) secondarySplitConfidenceOption -0.1;

е) maxOptionPathsOption - 5.

Результати тимчасових оцінок алгоритму для різних варіантів прийняття рішення про класифікацію в вузлі приведені в таблиці 4.6.

Таблиця 4.6 - Результати тимчасових оцінок варіативних дерев Хефдінга

Опція прийняття рішення в листі	Параметр	Оцінка, мс
Majority class	Середній час тренування 10000 примірників	192,10
	Середній час тестування 500000 примірників	1987,37
Naïve Bayes	Середній час тренування 10000 примірників	188,28
	Середній час тестування 500000 примірників	7158,46
Naïve Bayes Adaptive	Середній час тренування 10000 примірників	762,48
	Середній час тестування 500000 примірників	7032,08

Результат підрахунку точності і Kappa-статистики в ході роботи алгоритму для всіх трьох опцій передбачення значень в листі наведено на рис. 4.9,4.10, 4.11, 4.12, 4.13, 4.14 відповідно.

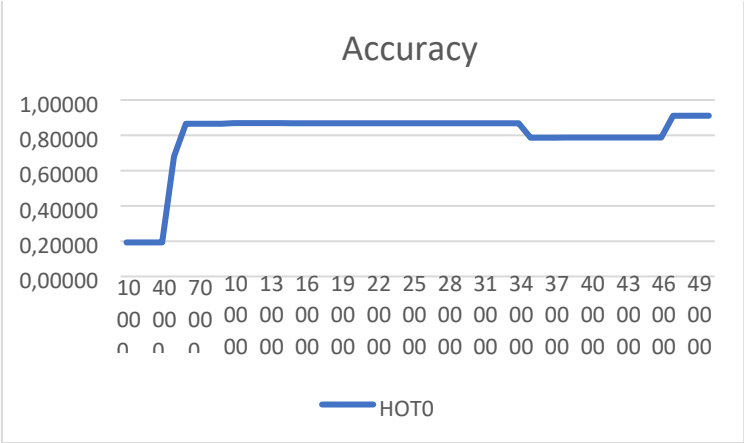

Рисунок 4.9 - Точність алгоритму HOT з опцією Majority class в залежності від обсягу тренувальних даних, отримана в ході експерименту

Рисунок 4.10 - Kappa статистика алгоритму HOT з опцією Majority class в залежності від обсягу тренувальних даних, отримана в ході експерименту

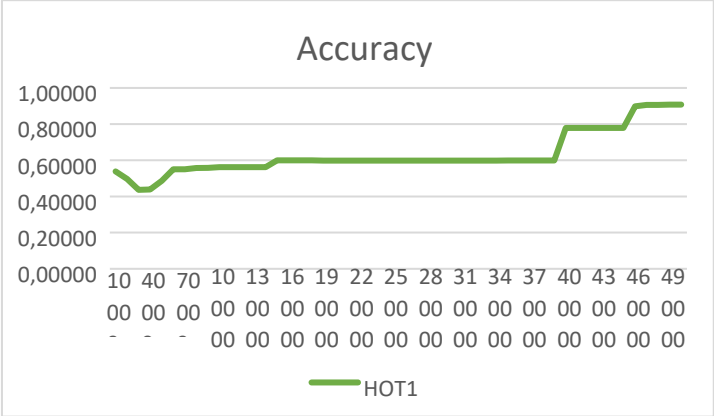

Рисунок 4.11 - Точність алгоритму HOT з опцією Naive Bayes в залежності від обсягу тренувальних даних, отримана в ході експерименту

Рисунок 4.12 - Карра статистика алгоритму HOT з опцією Naive Bayes в залежності від обсягу тренувальних даних, отримана в ході експерименту

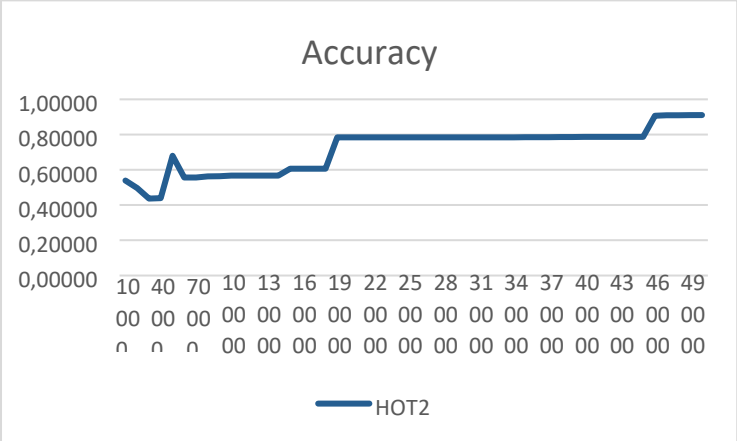

Рисунок 4.13 - Точність алгоритму HOT з опцією Naive Bayes Adaptive в залежності від обсягу тренувальних даних, отримана в ході експерименту

Рисунок 4.14 - Карра статистика HOT з опцією Naive Bayes Adaptive в залежності від обсягу тренувальних даних, отримана в ході експерименту

4.3 Висновки до розділу

В результаті проведення експерименту було встановлено, що для вирішення задачі класифікації даних запропонованої KDD-99 метод найближчих сусідів не підходить. Це пов'язано з тим, що модель пам'ятає тільки 1000 останніх переглянутих примірників, що є причиною того, що точність алгоритму в конкретний момент часу залежить від цих 1000 елементів і не підходить для детектування аномалій в мережевому трафіку, тому що в даній задачі частота народження помилок вкрай мала. Можливим вирішенням цієї проблеми може стати збільшення розміру плаваючого вікна, проте це збільшить і без того сильне навантаження на оперативну пам'ять і процесорний час. Даний алгоритм показав найгірший час тестування серед інших алгоритмів. Це пов'язано з тим, що на етапі тестування необхідно вважати відстань до цільового примірника на 1000 зберігаючихся у вікні елементів, потім вибирати найкращі. Отже, даний алгоритм не підходить для вирішення задачі класифікації даних в потоці, тому що не задовольняє вимозі швидких і точних прогнозів.

Алгоритм VFDT показав більш високі показники точності і Kappa-статистики, при цьому зі збільшенням числа елементів, вивчених моделлю дані показники, ростуть. Однак зростання даних показників зупиняється на певній позначці (0,91 для точності і 0,90 для Kappa-статистики). Дана особливість обумовлена специфікою тестових даних, що містять класи, відсутні в тренувальному наборі даних.

Тимчасові показники для алгоритму VFDT наступні: середній час тренування -, середній час тестування -. Звідси можна зробити висновок, що даний алгоритм істотно менше навантажує процесор і здатний видавати передбачення практично миттєво, що робить його відмінним алгоритмом потокової класифікації.

Алгоритм Варіативних дерев Хефдінга показав схожі статистичні результати в порівнянні з алгоритмом, що використовує стандартні дерева Хефдінга, однак тимчасові показники даного алгоритму істотно нижче

базової версії. Отже, можна зробити висновок, що збільшення складності побудови дерева в конкретної задачі тільки додало накладних витрат і дало несуттєвий приріст в продуктивності. Проте, даний алгоритм, як і раніше є хорошим розв'язанням даного завдання, здатним видавати передбачення високої точності практично миттєво.

Висновки

У даній роботі проаналізовано методи пошуку значущих параметрів для виявлення мережевих атак.

Для реалізації і тестування системи виявлення DDoS-атак була використана платформа WEKA. Експерименти були виконані с чотирма типами атак: DoS:, Probe, U2R, R2L.

Для виявлення DoS атак краще використовувати параметри, які були отримані за допомогою методу анулювання кореляції між координатами. Для виявлення Probe атак кращий результат дали параметри за методом головних компонент представлений набір параметрів з анулювання кореляції між координатами. Для виявлення U2R атак - метод пошуку ортогональних проекцій з найбільшим розсіюванням. Для виявлення R2L - також метод пошуку ортогональних проекцій з найбільшим розсіюванням.

Для вирішення задачі класифікації даних запропонованої KDD-99 найкраще результат показав алгоритм варіативних дерев Хефдінга, однак требо відзначити, що збільшення складності побудови дерева додає накладних витрат і несуттєвий приріст в продуктивності. Проте, даний алгоритм є здатним видавати передбачення високої точності практично миттєво.

В підсумку треба відзначити, що на даний момент Data Mining є дуже популярним напрямком аналізу і прогнозування з метою поліпшення виробництва. Етапи такого аналізу дозволяють не тільки зробити прогнози і прийняти рішення, але також класифікувати, нормувати і підвищити якість даних використовуваних в IT-системах.

Процес інтелектуального аналізу даних, яким є і Data Mining, досить трудомісткий і тривалий. До 80% цього процесу може зайняти етап попередньої обробки даних. Тому очевидна нагальність даної проблеми, пошуку оптимальних і необхідних методів обробки даних перед їх безпосереднім аналізом.

На даний момент, існує достатня кількість інструментів з проведення всіх етапів Data Mining, в тому числі і попередньої обробки даних. Однак область інтелектуального аналізу даних, залишається відкритою для нововведень, оскільки відносно нова.

Список використаних джерел

1. Gupta, D., Singhal, S., Malik, S., & Singh, A. (2016, May). Network intrusion detection system using various data mining techniques. In Research Advances in Integrated Navigation Systems (RAINS), International Conference on (pp. 1-6). IEEE.

2. Gaidhane, R., Vaidya, C., & Raghuwanshi, M. (2014). Survey: Learning Techniques for Intrusion Detection System (IDS).

3. Stolfo, S. J., Malkin, T., Keromytis, A. D., Misra, V., Locasto, M., & Parekh, J. (2015). U.S. Patent Application No. 14/846,188.

4. Elhag, S., Fernández, A., Bawakid, A., Alshomrani, S., & Herrera, F. (2015). On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on Intrusion Detection Systems. Expert Systems with Applications, 42(1), 193-202.

5. Saraswati, A., Hagenbuchner, M., & Zhou, Z. Q. (2016, December). High Resolution SOM Approach to Improving Anomaly Detection in Intrusion Detection Systems. In Australasian Joint Conference on Artificial Intelligence (pp. 191-199). Springer International Publishing.

6. Costa, K. A., Pereira, L. A., Nakamura, R. Y., Pereira, C. R., Papa, J. P., & Falcão, A. X. (2015). A nature-inspired approach to speed up optimum-path forest clustering and its application to intrusion detection in computer networks. Information Sciences, 294, 95-108.

7. Gautam, S. K., & Om, H. (2017). Comparative Analysis of Classification Techniques in Network Based Intrusion Detection Systems. In Proceedings of the First International Conference on Intelligent Computing and Communication (pp. 591-601). Springer Singapore.

8. Ashfaq, R. A. R., Wang, X. Z., Huang, J. Z., Abbas, H., & He, Y. L. (2017). Fuzziness based semi-supervised learning approach for intrusion detection system. Information Sciences, 378, 484-497.

9. Joffe, R. L. (2016). U.S. Patent No. 9,356,942. Washington, DC: U.S. Patent and Trademark Office.

10. Devi, R., Jha, R. K., Gupta, A., Jain, S., & Kumar, P. (2017). Implementation of Intrusion Detection System using Adaptive Neuro-Fuzzy Inference System for 5G wireless communication network. AEU-International Journal of Electronics and Communications, 74, 94-106.

11. Adebowale, A., Olutayo, A., Sunday, I., Ogbonna, A. C., & Oluwabukola, O. (2017). Scalable Unsupervised Ensemble Algorithm For Effective Insider Threat Detection. Universal Journal of computer and Technology (UJCT), 1(3), 76-83.

12. Levitt, Karl, and Gihan Dias. «Towards Detecting Intrusions in a Networked Environment.» (2017).

13.Petersen, R. (2015). Data Mining for Network Intrusion Detection: A comparison of data mining algorithms and an analysis of relevant features for detecting cyber-attacks.

14. Kevric, J., Jukic, S., & Subasi, A. (2016). An effective combining classifier approach using tree algorithms for network intrusion detection. Neural Computing and Applications, 1-8.

15. Eesa, A. S., Orman, Z., & Brifcani, A. M. A. (2015). A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems. Expert Systems with Applications, 42(5), 2670-2679.

16. Kim, G., Lee, S., & Kim, S. (2014). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. Expert Systems with Applications, 41(4), 1690-1700.

17. Dash, S., Mishra, R. K., Das, R. K., & Panda, M. (2017). Comparison of AIS based Data Mining Algorithms for Intrusion Detection. International Journal of Computer Science and Information Security, 15(1), 619.

18. Gai, K., Qiu, M., Tao, L., & Zhu, Y. (2016). Intrusion detection techniques for mobile cloud computing in heterogeneous 5G. Security and Communication Networks, 9(16), 3049-3058.

19. Aggarwal C. Data Streams: Models and Algorithms. Boston: Springer, 2017. Вып.

20. Bifet A., Kirkby R. Data stream mining. A practical appoach. Waikato: The university of Waikato, 2017. Вып. 1.

21. Шаньгин В.Ф. Информационная безопасность компьютерных систем и сетей. Москва, ИД «ФОРУМ»: ИНФРА-М, 2008, 416 с.

22. Мельников В.В. Защита информации в компьютерных системах. Москва, Финансы и статистика, 1997, 368 с.

23. Allen J., Christie A., Fithen W. et al. State of Practice of intrusion detection technologies. Technical Report CMU/SEI-99-TR-028. Carnegie Mellon Software Engineering Institute, 2000.

24. Исследование средств обнаружения вторжений. [Электронный ресурс]: http://www.lghost.ru/lib/security/kurs5/theme14_chapter01.htm

25. LJCSNS International Journal of Computer Science and Network Security, VOL.9

No.4, April 2009 «Adaptive Framework for Network Intrusion Detection by Using Genetic-Based Machine Learning Algorithm» [Электронный ресурс]: http://search.ijcsns.org

26. H. Günes Kayacık, A. Nur Zincir-Heywood, Malcolm I. Heywood. Selecting Features for Intrusion Detection:A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets. Dalhousie University, Faculty of Computer Science - 6c.

27. Ian H. Witten, Eibe Franf «Data Mining. Practical Machine Learning Tools and Techniques» -2nd ed. – 525p.

28. Метод главных компонент. [Электронный ресурс]: http://www.machinelearning.ru/wiki/index.php

29. A.Zaind, M. Maarof, S.Shamsnddin, A. Abraham,"Ensamble of One-class Classifier for Ntwork Intrusion Detections". [Электронный ресурс]:www.softcomputing.net/ias08_1.pdf

30. S.Mukkamala, A.h.Sung," Identifying Significant Features for Network Forensic Analysis using Artificial Intelligent Techniques", International Journal of Digital Evidence, Vol 1, Issue 4, Winter 2003.

31. S.Mukkamala, A.h.Sung, A. Abraham, Modeling Intrusion Detection System using Linear Genetic Programmimg Approaches", LNCS 3029, Springer Hiedelberg, pp 633-642,2004.

32. T.S.Chou, K.K.Yen, J.LNO," Network Intrusion Detection Design Using Feature Selection of Soft Computing Paradigms", International Journal Of Computational Intelligence, 2008.

33. Кластерный анализ [Электронный ресурс]: http://www.statsoft.ru/home/textbook/modules/stcluan.html

34. А.Е. Лепский, А.Г. Броневич Математические методы распознавания образов: Курс лекций.-Таганрог:Изд-во ТТИ ЮФУ,2009.-155с. [Электронный ресурс]: http://window.edu.ru/resource/800/73800/files/lect_Lepskiy_Bronevich_pass.pdf

35. Численные методы решения задач на собственные значения и собственные векторы матриц [Электронный ресурс]: http://www.uchites.ru/files/nummethod_book_chapter1-2.pdf

36. Principal Component Analysis (PCA) & NIPALS algorithm, Henning Risvik, May 10, 2007 [Электронный ресурс]: http://folk.uio.no/henninri/pca_module/pca_nipals.pdf

37. Сотников П. И. Обзор методов обработки сигнала электроэнцефалограммы в интерфейсах мозг-компьютер, Эл. Научно-технический журнал: Инженерный вестник., 2014.

38. Алгебраические проблемы собственных значений. Научная библиотека. [Электронный ресурс]: http://stu.sernam.ru/book_dig_m.php?id=79

39. Ueno K. и др. Anytime Classification Using the Nearest Neighbor Algorithm with Applications to Stream Mining // Sixth International Conference on Data Mining (ICDM'06). 2006.

40. Domingos P., Hulten G. Mining high-speed data streams // Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '00. 2000.

41. Yusuf B., Reddy P. Mining Data Streams using Option Trees // International Journal of Computer Network and Information Security. 2012. Т. 4. № 8. С. 49-54.

42. Pfahringer B., Holmes G., Kirkby R. New Options for Hoeffding Trees // AI 2007: Advances in Artificial Intelligence. С. 90-99

43. KDD Cup 1999 Data [Электронный ресурс]. URL: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.

44. MIT Lincoln Laboratory: Cyber Systems & Technology: DARPA Intrusion Detection [Электронный ресурс]. URL: http://www.ll.mit.edu/ideval/

45. Aggarwal C. Data Streams: Models and Algorithms. Boston: Springer, 2017. Вып.

46. Bifet A., Kirkby R. Data stream mining. A practical appoach. Waikato: The university of Waikato, 2017. Вып. 1.

47. Bifet A., Kirkby R. Massive online analysis manual. Waikato: The University of Waikato, 2017. Вып. 1.

48. Rajeev T., Santosh K. A Quick Review of Data Stream Mining Algorithms // Imperial Journal of Interdisciplinary Research. 2016. Т. 2. № 7. С. 870-873.

49. Mohammed H., Soliman A. Data stream mining // Data Mining and Knowledge Discovery Handbook / под ред. M. Oded, R. Lior. New York: Springer, 2010. Вып. 1. С. 231-235.

50.Mining data streams // Mining of Massive Datasets / под ред. J. Leskivec, A. Ullman, D. Jeffrey. Cambridge: Cambridge University Press, 2017. Вып. 2. С. 131162.

ДОДАТОК А

Лістинг коду тестового класу

```
public class TestClass {

  private static void exists(String fileName) throws FileNotFoundException {
    File file = new
File(fileName);     if
(!file.exists()){
      throw new FileNotFoundException(file.getName());

}

}

  public static void write(String fileName, String text) {
    // визначаємо файл
    File file = new
File(fileName);     try {
      // перевіряємо, що якщо файл не існує то
створюємо його
if(!file.exists()){         file.createNewFile();
      }
      //PrintWriter забезпечить можливості запису в файл
      PrintWriter out = new
PrintWriter(file.getAbsoluteFile());       try {
        // Записуємо текст у файл
out.print(text);
      } finally {
        // Після чого ми повинні
закрити файл
        // Інакше файл не запишеться
out.close();
      }
    } catch(IOException e) {
throw new
RuntimeException(e);

}

}

  public static String read(String fileName) throws FileNotFoundException {
    // Цей спец. об'єкт для побудов
StringBuilder sb = new
StringBuilder();
exists(fileName);     //Определяем
файл
    File file = new
File(fileName);     try {
      // Об'єкт для читання файлу в
буфер
```

```
      BufferedReader in = new
BufferedReader(new FileReader(
file.getAbsoluteFile()));       try {
        // У циклі через підрядник
зчитуємо файл   String s;
        while ((s = in.readLine()) !=
null) {           sb.append(s);
sb.append("\n");
        }
      } finally {
        // Також не забуваємо закрити
файл   in.close();
      }
    } catch(IOException e) {
throw new
RuntimeException(e);
    }
    //Повертаємо отриманий текст з
файлу
    return sb.toString();
  }
  public static void update(String nameFile, String newText) throws
FileNotFoundException {
exists(nameFile);
    StringBuilder sb = new
StringBuilder();     String oldFile
= read(nameFile);
sb.append(oldFile);
sb.append(newText);
    write(nameFile, sb.toString());
  }
  public static void main(String[] args) throws
FileNotFoundException {     String str="Finished!";     long
SumTrainingTime=0;     long SumTestTime =0;

//*************************************************************************
********
**************************************
    // Основний тестовий метод

//*************************************************************************
********
**************************************
    String filename1 = "D:\\Bachelor-work\\resultOptionHoeffding2Acc.txt";
    String filename2 = "D:\\Bachelor-
work\\resultOptionHoeffding2Kappa.txt";     /*
    String filename1 = "D:\\Bachelor-work\\resultOptionHoeffding1Acc.txt";
    String filename2 = "D:\\Bachelor-work\\resultOptionHoeffding1Kappa.txt";
    String filename1 = "D:\\Bachelor-work\\resultOptionHoeffding0Acc.txt";
    String filename2 = "D:\\Bachelor-
work\\resultOptionHoeffding0Kappa.txt";
    String filename1 = "D:\\Bachelor-work\\resultHoeffding2Acc.txt";
    String filename2 = "D:\\Bachelor-
work\\resultHoeffding2Kappa.txt";     String filename1 =
"D:\\Bachelor-work\\resultHoeffding1Acc.txt";
    String filename2 = "D:\\Bachelor-work\\resultHoeffding1Kappa.txt";
    String filename1 = "D:\\Bachelor-work\\resultHoeffding0Acc.txt";
    String filename2 = "D:\\Bachelor-
work\\resultHoeffding0Kappa.txt";
```

```
    String filename1 = "D:\\Bachelor-
work\\resultKNearestAcc.txt";     String filename2 =
"D:\\Bachelor-work\\resultKNearestKappa.txt"
    */

write(filename1,"");
write(filename2,"");
int numInstances
=10000;     int i =
0;
    Classifier  learner2= new HoeffdingOptionClassifier();
    // Classifier  learner2= new KNearestClassifier();
    //Classifier  learner2= new HoeffdingClassifier();

((HoeffdingOptionClassifier)learner2).leafpredictionOption.setChosenIndex(2);
System.out.println("Hoeffding 2");
    BachelorClassificationEvaluator                    evaluator          =          new
BachelorClassificationEvaluator();
    ArffFileStream      strm  =     new
ArffFileStream("D:\\Bachelorwork\\DATA\\KDDCup9910p.arff",42);
    ArffFileStream      TestStream  =     new
ArffFileStream("D:\\Bachelorwork\\corrected\\corrected.arff",42);
strm.prepareForUse();

learner2.setModelContext(strm.getHeader
());     learner2.prepareForUse();
int numberSamplesCorrect=0;     int
numberSamples=0;     int
numberTrainSamples=0;       while
(strm.hasMoreInstances()) {     i++;


    long start = System.nanoTime();
    System.out.println("number of instances for training " + numInstances);
    System.out.print(i+") ");


    while(strm.hasMoreInstances() && numberTrainSamples<numInstances){
        Instance trainInst =
strm.nextInstance();
learner2.trainOnInstance(trainInst);
numberTrainSamples++;       }
      long end =
System.nanoTime();
SumTrainingTime+=end-start;
System.out.println("test");
numberTrainSamples =0;
evaluator.reset(39);
TestStream.restart();
start = System.nanoTime();
       while (TestStream.hasMoreInstances()){
        Instance testInst = TestStream.nextInstance();
if (learner2.correctlyClassifies(testInst)){
numberSamplesCorrect++;
        }
        numberSamples++;
        evaluator.addResult(testInst,
learner2.getVotesForInstance(testInst));
      }
      end = System.nanoTime();
      SumTestTime+=end-start;
```

```
      Measurement[] x =evaluator.getPerformanceMeasurements();
      double accuracy = 100.0*(double) numberSamplesCorrect/(double)
numberSamples;
update(filename1,evaluator.getFractionCorrectlyClassified() + "\n");
      update(filename2,evaluator.getKappaStatistic()+ "\n");
      System.out.println(numberSamples + " instances processed with " +
accuracy + "% accuracy " );
      System.out.println("вес " +
evaluator.TotalweightObserved);       numberSamples=0;
numberSamplesCorrect=0;
    }
    System.out.println(str);
    System.out.println("Avg training time :" + SumTrainingTime/i);
    System.out.println("Avg test time :" + SumTestTime/i);
```