МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

КУЗЬМІН Євгеній Олександрович

Програмна система класифікації мережного трафіку засобами Data Mining / Software system for network traffic classification by means of Data Mining

Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

Дипломний проект

Виконав студент групи КН-42
Є.О. Кузьмін

Науковий керівник:
д.т.н., професор М.П. Комар

Дипломний проект допущено до захисту
«___»_____ 2023 р.

Завідувач кафедри
_____ М.П. Комар

Тернопіль – 2023

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____2022 р.

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

КУЗЬМІН Євгеній Олександрович
(прізвище, ім'я, по батькові)

1. Тема проекту: Програмна система класифікації мережного трафіку засобами Data Mining / Software system for network traffic classification by means of Data Mining
керівник проекту д.т.н., проф. М.П. Комар

затверджені наказом по університету від 08 грудня 2022 р. № 491.

2. Строк подання студентом закінченого проекту 01 червня 2023 р.

3. Вихідні дані до проекту: технічне завдання.

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

- опис загального підходу класифікації мережевого трафіку;
- опис методів класифікації;
- опис методів кластеризації;
- постановка задачі дослідження;
- дослідження теоретичних основ застосування технології інтелектуального збору даних;
- дослідження методів попередньої обробки даних;
- дослідження методів кластеризації даних і аналіз отриманих результатів;
- побудова класифікатора і його навчання;
- дослідження методу класифікації мережевого трафіку із застосуванням протоколу NetFlow і машинного навчання;

– проведення навчання ієрархічної класифікаційної моделі;
– проведення експериментальних досліджень.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
– схема «ручної» класифікації мережного трафіку;
– схема об'єднання однонаправлених потоків в двонаправлені.

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Н. контроль	д.т.н., професор М.П. Комар		

7. Дата видачі завдання 08 грудня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів проекту	Примітка
1	Аналіз методів і алгоритмів Data Mining для класифікації мережевого трафіку	30.12.2022	
2	Методи та засоби виявлення прихованих атак на основі технологій Data Mining	24.03.2023	
3	Практичне застосування методів класифікації мережевого трафіку з використання протоколу Netflow	12.05.2023	
4	Повне завершення та оформлення дипломного проекту	01.06.2023	

Студент_____ Є.О. Кузьмін
(підпис)

Керівник проекту_____ М.П. Комар
(підпис)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту: 77 с., 18 рис., 3 додатки, 42 джерела.

Метою дипломного проекту є підвищення точності класифікації мережевого трафіку на основі методів інтелектуального аналізу даних.

В роботі використовувалися методи системного аналізу, порівняння, логічного узагальнення результатів, методи інтелектуального аналізу даних.

Вдосконалено метод аналізу параметрів мережевого трафіку з метою його класифікації, який полягає у поєднанні декількох математичних методів в певній послідовності з одноразовим використанням попередньої обробки даних, що дозволило підвищити точність роботи класифікатора, не зменшуючи, при цьому, продуктивність роботи мережевого обладнання.

Розроблено програмне забезпечення для реалізації запропонованих методів та засобів.

Ключові слова: ПАРАМЕТРИ МЕРЕЖЕВОГО ТРАФІКУ, ВЕЛИКІ ДАНІ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, ПОПЕРЕДНЯ ОБРОБКА ДАНИХ, КЛАСТЕРНИЙ АНАЛІЗ, НЕЙРОННІ МЕРЕЖІ, ДЕРЕВА РІШЕНЬ.

ABSTRACT

The bachelor's thesis report: 77 pages, 18 figures, 3 appendices, 42 sources.

The aim of the thesis project is to increase the accuracy of classification of network traffic based on methods of data mining.

The methods of system analysis, comparison, logical generalization of results, methods of data mining were used in the work.

Improved method of analysis of network traffic parameters to classify it, which is a combination of several mathematical methods in a certain sequence with a single use of pre-processing, which increased the accuracy of the classifier without reducing the performance of network equipment.

The software for realization of the offered methods and means is developed.

Keywords: NETWORK TRAFFIC PARAMETERS, BIG DATA, DATA MINING, PRE-PROCESSING, CLUSTER ANALYSIS, NEURAL NETWORKS, DECISION TREES.

ТЕХНІЧНЕ ЗАВДАННЯ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

1.1 Програмна система класифікації мережного трафіку засобами Data Mining.

1.2 Область застосування – інтелектуальний аналіз та обробка даних.

2. ОСНОВА ДЛЯ РОЗРОБЛЕННЯ

Основою для розроблення є завдання на дипломний проект, затверджене кафедрою інформаційно-обчислювальних систем і управління факультету комп'ютерних інформаційних технологій Західноукраїнського національного університету.

3. ПРИЗНАЧЕННЯ РОЗРОБЛЕНОГО КОМПЛЕКСУ

Метою дипломного проекту є підвищення точності класифікації мережевого трафіку на основі методів інтелектуального аналізу даних.

4. ДЖЕРЕЛА РОЗРОБЛЕННЯ

Джерелами даної розробки є матеріали навчальної і реферативної літератури, технічна документація, науково-дослідні статті, журнали, Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Основні функціональні вимоги до програмної системи:

- збір даних;
- попередня обробка даних для перетворення у необхідний формат;
- навчання моделей з використанням попередньо оброблених даних;
- побудова та виконання різних експериментів з підмножиною набору даних, щоб отримати з нього значущі результати.

5.2 Вимоги до апаратних засобів:

– NetFlow.

5.3 Вимоги до програмних засобів:

– MySQL.

6. ПОРЯДОК КОНТРОЛЮ

6.1 Представлення дипломного проекту на попередній захист.

6.2 Представлення дипломного проекту на захист.

Завдання прийняв до виконання _____ Є.О. Кузьмін

(підпис) (прізвище та ініціали)

Керівник дипломного проекту_____ М.П. Комар

(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

Розвиток сучасних технологій породив необхідність автоматичного аналізу великих об’ємів різнорідних даних. Для цих цілей використовуються методи Data Mining. Метою методів Data Mining є витягування прихованих закономірностей, знань з великих об’ємів даних. Як правило, методи Data Mining застосовуються до сформованих і погоджених даних, що знаходяться в деякому сховищі даних. Результатом роботи методів є побудовані моделі даних, які потім представляються користувачеві.

Сучасні темпи росту каналів зв’язку і кількості користувачів з широкосмуговим доступом породжують проблеми, пов’язані з доступністю окремих мережевих ресурсів. Комп’ютери користувачів можуть стати частиною розподіленої мережі, яка відправляє великий об’єм трафіку на сервер жертви. В результаті буде ускладнений доступ для легітимних користувачів. Тому гостро стоїть задача розмежування легітимного і шкідливого трафіку.

Розподілені атаки на відмову в обслуговуванні (DDoS атаки) носять масовий характер, оскільки до них схильні як великі мережеві портали, так і сайти малих організацій. Атакам можуть піддатися і державні сайти, які надають електронні послуги населенню, інформують про надзвичайні ситуації і т. п. Власник мережевого ресурсу може понести економічні, репутаційні і політичні ризики. Забезпечення доступності інформації є однією із складових комплексного захисту інформації, разом з цілісністю і конфіденційністю.

Метою дипломного проекту є підвищення точності класифікації мережевого трафіку на основі методів інтелектуального аналізу даних.

Для досягнення поставленої мети необхідно виконати наступні завдання:

– опис загального підходу класифікації мережевого трафіку;
– опис методів класифікації;
– опис методів кластеризації;
– постановка задачі дослідження;
– дослідження теоретичних основ застосування технології

інтелектуального збору даних;

– дослідження методів попередньої обробки даних;

– дослідження методів кластеризації даних і аналіз отриманих результатів;

– побудова класифікатора і його навчання;

– дослідження методу класифікації мережевого трафіку із застосуванням протоколу NetFlow і машинного навчання;

– проведення навчання ієрархічної класифікаційної моделі;

– проведення експериментальних досліджень.

Об’єкт дослідження – процеси збору, попередньої обробки на аналізу параметрів мережевого трафіку.

Предмет дослідження – методи та засоби аналізу параметрів мережевого трафіку на основі Data Mining.

1 АНАЛІЗ МЕТОДІВ І АЛГОРИТМІВ DATA MINING ДЛЯ КЛАСИФІКАЦІЇ МЕРЕЖЕВОГО ТРАФІКУ

1.1 Опис загального підходу класифікації мережевого трафіку

У даній роботі розглядається проблема аналізу IP-трафіку з використанням методів Data Mining, до числа яких відносяться класифікація – розділення об’єктів на задані групи (класи) згідно з характеристиками об’єктів; регресія – пошук функції, що моделює множину об’єктів, що вивчаються, з найменшою помилкою; кластеризація – пошук незалежних груп об’єктів, кластерів і їх характеристик (групи заздалегідь не обумовлені); пошук асоціативних правил – характерних залежностей між об’єктами або подіями.

Формально методи пошуку закономірностей можна сформулювати таким чином [1].

Задача класифікації і регресії. Є множина досліджуваних об’єктів X ={x1, x2, . . ., xn}. Кожен об’єкт характеризується набором змінних (атрибутів) Xj ={a1, a2, . . ., am, y}, де ai – спостережувані змінні, значення яких відомі; y – залежна змінна, значення якої треба визначити. При цьому кожна змінна ai набуває значення з деякої множини Ai ={ai1, ai2, . . .}. Спостережувані змінні часто називаються ознаками або атрибутами. Якщо множина C ={c1, c2, . . ., ck} значень змінної y звичайно, то задача називається задачею класифікації. Якщо змінна y набуває значень на множині дійсних чисел R, то задача називається задачею регресії.

Задача кластеризації. Є множина досліджуваних об’єктів X ={x1, x2, . . ., xn}. Кожен об’єкт характеризується набором змінних Xj ={a1, a2, . . ., am}. Кожна змінна ai набуває значення з деякої множини Ai ={ai1, ai2, . . .}. Задача кластеризації полягає в побудові множини C ={c1, c2, . . ., ck}, де ci – кластер, що містить схожі об’єкти з множиною X, відносно введеної міри близькості d(xj, xp), що називається відстанню, тобто cm ={xj, xp|xj ∈ X, xp ∈ X&d(xjxp)< σ}, де σ – величина, що визначає максимальну відстань, на якій можуть знаходитися об’єкти одного кластера.

Задача пошуку асоціативних правил. Є набір початкових елементів I ={i1, i2, . . ., in}, а також набір об'єктів D ={d1, d2, . . ., dm}. Кожен об'єкт є підмножиною множини I (di ⊆ I). Відповідно до термінології, що відноситься до баз даних, di називається транзакцією, а D – базою даних. Правило – це імплікація виду X ⇒ Y, де X, Y ⊆ D і X ∩ Y = Ø. При цьому для виявлення найбільш правдоподібних правил, що відображають часто залежності, що зустрічаються, між транзакціями у базі даних, вводяться дві метрики. Підтримка набору X, що позначається як supp(X), – це пропорція набору X відносно усієї множини D. Підтримка правила supp (X ⇒ Y) = supp (X ∪ Y). Довіра до правила визначається по формулі conf(X ⇒ Y) = supp (X ∪ Y)/ supp (X). Чим більше значення підтримки і довіри, тим більш точно правило відображає залежності.

З алгоритмічної точки зору класифікація або кластеризація – це функція f : X → C, яка кожному об'єкту xi ∈ X ставить у відповідність мітку cj ∈ C. У задачі класифікації множина C визначена заздалегідь, в задачі кластеризації заздалегідь не визначена не лише множина C, але і її розмірність.

Для реалізації методів використовуються як базові статистичні алгоритми, так і генетичні алгоритми, нейронні мережі, алгоритми з області машинного навчання.

Машинне навчання є науковою галуззю, яка досліджує методи створення алгоритмів, що можуть навчатись на основі досвіду. Загалом, задача формулюється так: існує множина ситуацій та можливих реакцій на них. Є певна залежність між ситуаціями та реакціями, проте вона невідома. Відомо лише обмежена кількість прикладів у вигляді пар "ситуація–реакція", які складають навчальний набір даних. Метою є відтворення цієї залежності, тобто створення алгоритму, який може надавати достатньо точні реакції для будь-якої ситуації [2].

Алгоритми машинного навчання можуть бути класифіковані за методами навчання наступним чином: контрольоване навчання, де навчання відбувається на позначених даних і вимагається вивчення функції відображення для подальшої класифікації та диференціації вхідних даних; неконтрольоване

навчання, яке передбачає групування об'єктів без позначених даних, використовуючи міру близькості; та частково контрольоване навчання, яке відбувається на комбінації позначених і непозначених даних.

Для оцінки ефективності алгоритмів використовуються такі метрики: FP (ложно позитивний) – частка трафіку, помилково віднесена до класу X; FN (ложно негативний) – частка трафіку, що належить до класу X, але не віднесена до нього; правильність – відсоток коректно класифікованих одиниць відносно загальної кількості класифікованих одиниць, тобто (all-FP-FN)/all. Точність – співвідношення правильно класифікованих одиниць (TP) до отриманого класу: TP/(TP+FP); повнота/довіра – співвідношення правильно класифікованих одиниць до реального класу: TP/(TP+FN).

Початковими даними для дослідження є послідовності IP-пакетів, зібраних в точках спостереження. Одиницею розгляду є потік. Послідовність IP-пакетів може бути двонаправленою або однонапрямленою послідовністю пакетів між двома IP-адресами, повною TCP-сесією або однонаправленою послідовністю IP-пакетів, що визначається на основі п’яти полів заголовка: <src_ip, src_port, dst_ip, dst_port, protocol> і правил формування, по яких визначається завершення потоку (зазвичай таймаут або прапор "END" в заголовку пакету). Тут src_ip – IP-адрес джерела; src_port – порт джерела; dst_ip – IP-адрес призначення; dst_port – порт призначення, protocol – транспортний протокол. Як правило, в якості протоколів транспортного рівня використовуються TCP і UDP.

Фіксується набір змінних (атрибутів), заснованих на статистичних характеристиках, таких як розмір пакетів або інтервали між пакетами, і характеристиках, що витягуються із заголовків пакетів, таких як розмір TCP-сегментів або кількість повторних передач. Потоку ставиться у відповідність набір значень атрибутів, згідно з якими проводиться класифікація.

При описі алгоритмів і статистичних характеристик використовуються терміни теорії ймовірності і математичної статистики, визначення яких наведені в [3].

1.2 Опис методів класифікації

Методи класифікації застосовуються в інтелектуальній обробці до даних, які мають первинний деякий промаркірований список, що дозволяє виявити атрибути і властивості конкретного екземпляра, щоб побудувати бінарне дерево або дерево рішень.

Методи класифікації, можна розділити на декілька груп. За способом задання показника якості класифікації методи діляться на евристичні і оптимізаційні. За способом об'єднання - на дивизимні, агломеративні і ітеративні.

Евристичні алгоритми засновані на досвіді і інтуїції людини. Показник якості класифікації, який необхідно перетворити в екстремум, в цих алгоритмах в явному виді не заданий. Евристичні алгоритми реалізують процедури, що мають раціональний сенс з точки зору логіки людини і призводять у багатьох випадках до добрих результатів на практиці. До таких алгоритмів відносяться, наприклад, алгоритми «Граф», «Спектр», «Форель».

До оптимізаційних алгоритмів відносяться методи класифікації, в яких в явному вигляді заданий показник якості, який необхідно перетворити в екстремум (максимум або мінімум) по множині допустимого розбиття. На відміну від алгоритмів першої групи, розбиття, що отримані оптимізаційними алгоритмами класифікації, є найкращими з точки зору вибраного показника якості. Вибір конкретного показника залежить від специфіки і обмежень задачі, що вирішується, а також прийнятих пропозицій. Слід зазначити, що у багатьох випадках в евристичних алгоритмах показник якості заданий в неявному вигляді і вони можуть стати оптимізаційними, якщо вдається його формалізувати і сформулювати в явному виді [4].

У загальному випадку у будь-якому оптимізаційному алгоритмі класифікації можна виділити наступні елементи:

- показник якості класифікації;
- обмеження;

- механізм пошуку результуючого розбиття.

Обмеження в методах класифікації в основному торкаються типу початкових даних - множини допустимого розбиття, на якій шукається результуюче розбиття, і виду самого результуючого розбиття. Пошук результуючого розбиття здійснюється відповідно до деякого механізму оптимізації. Це може бути механізм повного або часткового перебору, випадкового перебору і т. д. Якщо механізм не забезпечує точного досягнення екстремуму показника якості, він є наближеним, а помилка оцінюється величиною відхилення значення показника якості, що досягається, від оптимуму. Якщо величина помилки незначна, алгоритм є субоптимальним (близьким до оптимального).

Конкретизація перерахованих елементів призводить до того або іншого методу класифікації. Оптимізаційні методи класифікації можуть бути засновані на кластерному аналізі.

Контрольований класифікатор naive Bayes. Так звана наївна класифікація або наївно-байєсівський підхід (naive-bayes approach) є найбільш простим варіантом методу, що використовує байєсівські мережі. При цьому підході вирішуються задача класифікації, результатом роботи методу є так звані "прозорі" моделі.

"Наївна" класифікація - досить прозорий і зрозумілий метод класифікації. "Наївною" вона називається тому, що виходить з припущення про взаємну незалежність ознак.

Властивості наївної класифікації:

1. Використання усіх змінних і визначення усіх залежностей між ними.

2. Наявність двох припущень відносно змінних:

- усі змінні є однаково важливими;

- усі змінні є статистично незалежними, тобто значення однієї змінної нічого не говорить про значення іншої.

Більшість методів класифікації зазвичай передбачають рівні ймовірності належності об'єкта до різних класів перед класифікацією, але це не завжди є

правильним підходом [5].

Припустимо, нам відомо, що певна частка даних належить до визначеного класу. Чи можемо ми врахувати цю інформацію при створенні моделі класифікації? Існує численні реальні випадки, коли такі апріорні знання можуть допомогти у класифікації об'єктів. Звичайний приклад з медицини: лікар надсилає аналізи пацієнта на подальше обстеження, він уже відносить пацієнта до певного класу. Як можна використати цю інформацію? Ми можемо застосувати її як додаткові дані при створенні класифікаційної моделі.

Байєсівські мережі як метод Data Mining мають наступні переваги:

- модель виявляє залежності між усіма змінними, що сприяє легкому оброблянню ситуацій з невідомими значеннями деяких змінних;
- легка інтерпретація байєсівських мереж та можливість проведення аналізу "що, якщо" на етапі прогнозного моделювання;
- байєсівський підхід дозволяє ефективно поєднувати закономірності, отримані з даних, з експертними знаннями, наданими у явному вигляді;
- використання байєсівських мереж допомагає уникнути проблеми перенавчання, тобто занадто складної моделі, яка є слабким місцем для багатьох методів (наприклад, дерев рішень і нейронних мереж).

Наївний байєсівський підхід має такі обмеження:

- коректне перемноження умовних ймовірностей можливе лише при статистичній незалежності всіх вхідних змінних; хоча метод часто показує хороші результати навіть при порушенні цієї умови, теоретично потрібні складніші методи, які базуються на навчанні байєсівських мереж;
- безпосередня обробка безперервних змінних неможлива, тому їх потрібно перетворити на інтервальну шкалу, щоб атрибути стали дискретними; однак такі перетворення можуть призвести до втрати важливих закономірностей;
- наївний байєсівський підхід враховує вплив лише окремих значень вхідних змінних, ігноруючи комбінований вплив пар чи трійок різних атрибутів; хоча це могло б покращити якість класифікаційної моделі з погляду прогнозної точності, кількість перевірених варіантів збільшилась би.

Байєсівська класифікація знайшла широке застосування на практиці [6].

Недавно байєсівська класифікація була впроваджена для особистої фільтрації спаму. Перший фільтр створив Пол Грем (Paul Graham). Для реалізації алгоритму вимагаються дві умови.

Перша умова полягає в наявності достатньої кількості ознак у класифікованому об'єкті. Відмінно підходять для цього всі слова користувача, крім дуже коротких або рідкісних.

Друга умова вимагає постійного перенавчання та поповнення наборів "спам" та "не спам". Ці умови працюють добре у локальних поштових клієнтах, оскільки потік "не спаму" у кінцевого користувача є досить стабільним, а якщо змінюється, то повільно.

Однак для всіх клієнтів сервера точно визначити потік "не спаму" складно, тому що те саме повідомлення може бути спамом для одного клієнта і не спамом для іншого. Словник виявляється надто великим, відсутнє чітке розподілення на спам та "не спам", внаслідок чого якість класифікації, у даному випадку рішення задачі фільтрації листів, суттєво знижується.

Алгоритм побудови асоціативних правил Apriori. На першому кроці алгоритму підраховуються 1-елементні набори (рисунок 1.1), що часто зустрічаються. Для цього необхідно пройтися по усьому набору даних і підрахувати для них підтримку, тобто скільки разів зустрічається у базі.

Наступні кроки будуть складатися з двох частин:

1) генерації наборів елементів (їх називають кандидатами), що потенційно часто зустрічаються;

2) підрахунку підтримки, для кандидатів.

Опишемо функцію генерації кандидатів. На це раз немає ніякої необхідності знову звертатися до бази даних. Для того, щоб отримати k-елементні набори, скористаємося (k-1)-елементними наборами, які були визначені на попередньому кроці і є такими, що часто зустрічаються [7].

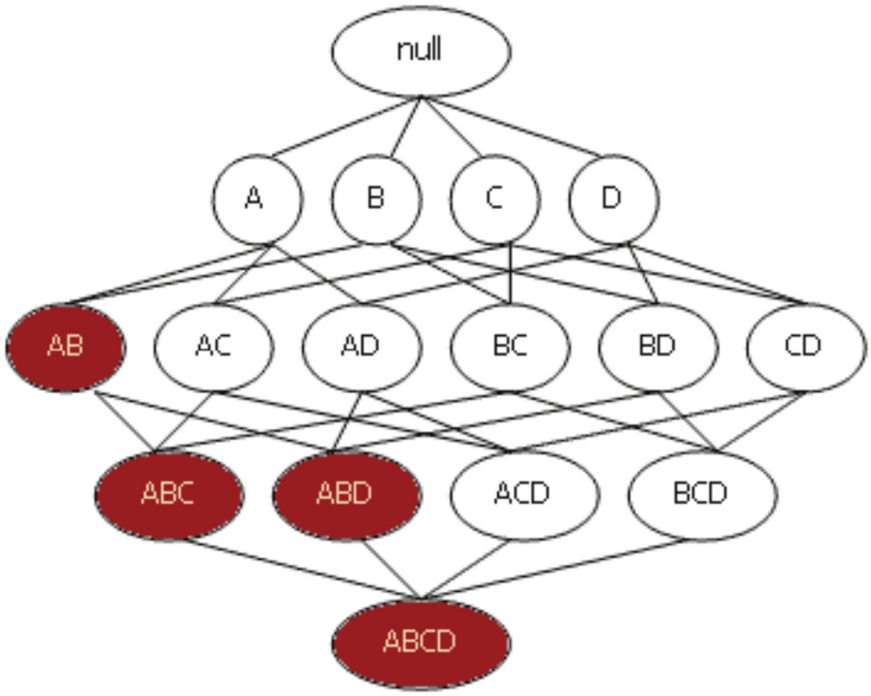

Рисунок 1.1 – Візуалізація асоціативних правил

Об’єднання. Кожен кандидат Ck формуватиметься шляхом розширення набору розміру, що часто зустрічається (k-1) додаванням елементу з іншого (k-1) - елементного набору.

Представимо алгоритм цієї функції Apriorigen у вигляді невеликого SQL-запиту.

insert into Ck;

select p.item1, p.item2, ., p.itemk - 1, q.itemk - 1;

From Fk - 1 p, Fk - 1 q;

where p.item1= q.item1, p.item2 = q.item2, ., p.itemk - 2 = q.itemk - 2, p.itemk - 1 < q.itemk - 1.

Видалення надмірних правил. На підставі властивості анти-монотонності, слід видалити усі набори с єє Ck якщо хоч би одна з його (k - 1) підмножин не є таким, що часто зустрічається.

Після створення кандидатів наступним кроком є розрахунок підтримки для кожного з них. Зрозуміло, що кількість кандидатів може бути величезною, тому потрібен ефективний метод розрахунку. Найпростіший спосіб - порівняти кожну транзакцію з кожним кандидатом, але це не найкращий варіант. Більш швидким

та ефективним є підхід, який базується на зберіганні кандидатів у хеш-дереві. Внутрішні вузли дерева містять хеш-таблиці з посиланнями на своїх нащадків, а листки - на кандидатів. Таке дерево допоможе нам швидко розраховувати підтримку кандидатів.

Хеш-дерево будується кожен раз, коли формуються кандидати. Спочатку дерево складається лише з кореня, який є листком і не має жодних наборів кандидатів. З появою нового кандидата він додається до кореня дерева, і це продовжується до тих пір, поки кількість кандидатів у кореневому листку не перевищить певний поріг. Як тільки кількість кандидатів перевищує поріг, корінь перетворюється на хеш-таблицю, тобто стає внутрішнім вузлом, і для нього створюються листкові нащадки. Приклади розподіляються по нащадках відповідно до хеш-значень елементів, що утворюють набір, і так далі. Кожен новий кандидат хешується на внутрішніх вузлах до тих пір, поки не дійде до першого листкового вузла, де він зберігатиметься, допоки кількість наборів знову не перевищить порогове значення.

Після того, як хеш-дерево з кандидатами-наборами побудоване, легко підрахувати підтримку для кожного кандидата. Щоб зробити це, потрібно "пройти" кожну транзакцію через дерево і збільшити лічильники для тих кандидатів, чиї елементи також присутні в транзакції, тобто Ck ∩∩ Ti = Ck. На кореневому рівні хеш-функція застосовується до кожного елементу транзакції. Далі, на другому рівні, хеш-функція застосовується до наступних елементів і так далі. На k-му рівні хешується k-й елемент. Це продовжується до тих пір, поки не дійдемо до листка. Якщо кандидат, що зберігається в листку, є підмножиною даної транзакції, тоді збільшуємо лічильник підтримки цього кандидата на одиницю.

Після того, як кожна транзакція з вихідного набору даних "пройшла" через дерево, можна перевірити, чи задовольняють значення підтримки кандидатів мінімальному порогу. Кандидати, які відповідають цьому критерію, вважаються часто зустрічаними. До того ж, варто запам'ятати підтримку набору, оскільки це буде корисним для виявлення правил [8]. Ті самі дії застосовуються для пошуку

(k+1)-елементних наборів і т. д.

Після того, як знайдені усі набори елементів, що часто зустрічаються, можна приступити безпосередньо до генерації правил.

Витягування правил – є менш трудомісткою задачею. Для підрахунку достовірності правила достатньо знати підтримку самого набору і множини, що лежить в умові правила. Наприклад, є набір, що часто зустрічається {A, B, C} і вимагається підрахувати достовірність для правила AB ⇒⇒ C. Підтримка самого набору нам відома, але і його множина {A, B}, що лежить в умові правила, також є такою, що часто зустрічається в силу властивості анти-монотонності, і означає його підтримка нам відома. Тоді ми легко зможемо підрахувати достовірність. Це позбавляє нас від небажаного перегляду бази транзакцій, який знадобився у тому разі якби ця підтримка була невідома.

Щоб витягнути правило з набору F, що часто зустрічається, слід знайти усі його непорожні підмножини. І для кожної підмножини s ми зможемо сформулювати правило s ⇒⇒ (F - s), якщо достовірність правила conf(s ⇒⇒ (F - s)) = supp(F)/supp(s) не менше порогу minconf.

Звернімо увагу на той факт, що чисельник в формулі залишається незмінним. Отже, якщо ми хочемо досягти мінімального значення достовірності, ми повинні максимізувати знаменник. Це можна досягти тоді, коли в умові правила міститься лише один елемент. В цьому випадку, супермножина всієї великої кількості містить меншу або рівну підтримку, що призводить до більш високого значення достовірності. Ця властивість може бути корисною при витягуванні правил. Якщо ми спочатку розглядаємо правила з одним елементом в умові та ці правила мають необхідну підтримку, то усі правила, в яких умова містить супермножину цього елементу, також матимуть достовірність вище мінімального порогу. Наприклад, якщо правило A ⇒⇒ BCDE задовольняє мінімальному порогу достовірності minconf, тоді правило AB ⇒⇒ CDE також задовольняє його. Для того, щоб витягнути всі правила, використовується рекурсивна процедура. Важливою умовою є те, що будь-яке правило, що складається з набору, що часто зустрічається, повинне містити всі елементи

цього набору. Наприклад, якщо набір містить елементи {A, B, C}, то правило A ⇒⇒ B не повинно бути розглянуте [9].

C4.5. Метод побудови дерев рішень (рисунок 1.2), який уперше був запропонований Р. Куінленом (R. Quinlan). Цей метод використовується в одному з кращих алгоритмів побудови дерев рішень C4.5.

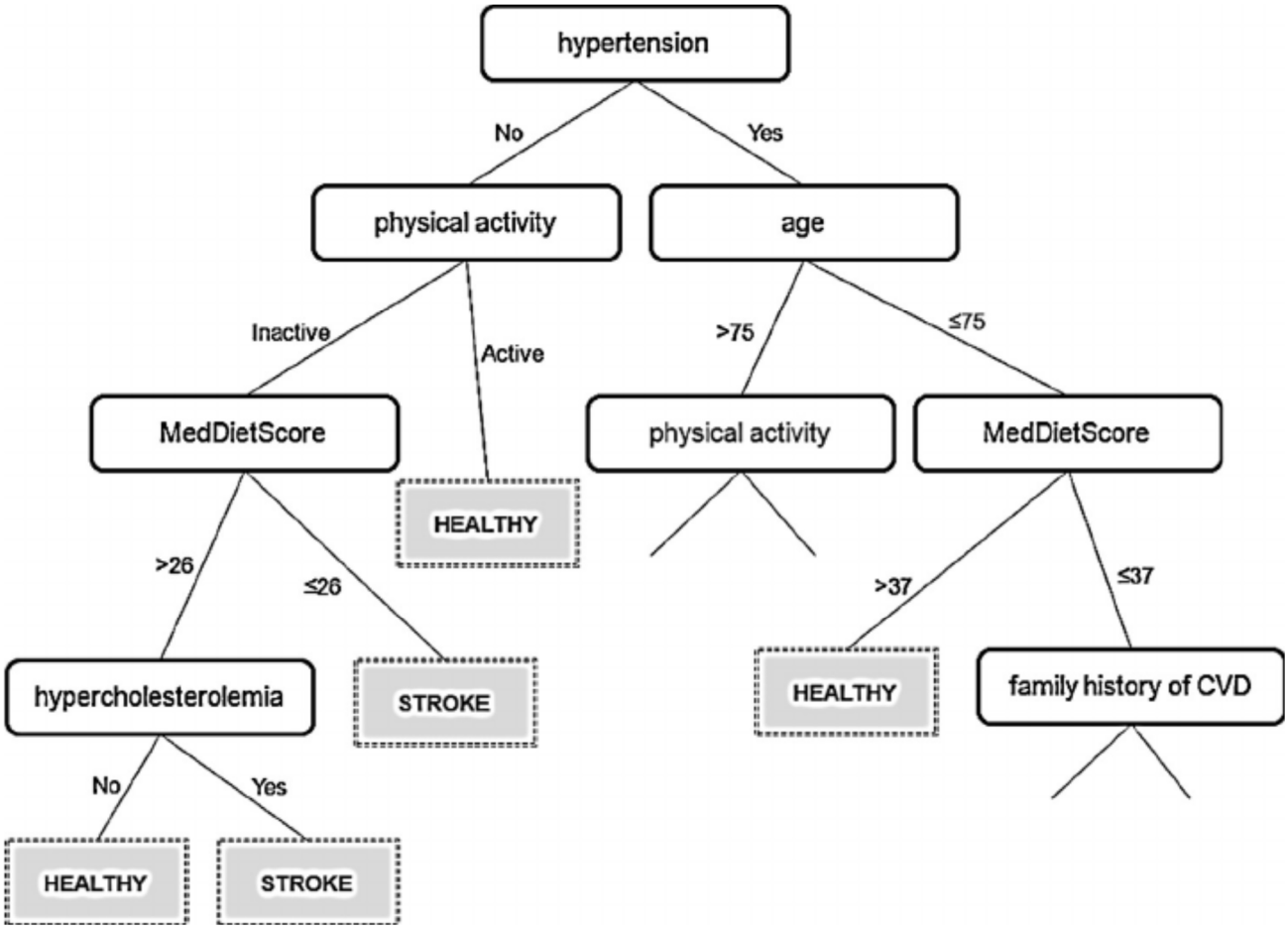

Рисунок 1.2 - Приклад дерева рішень

Перш ніж приступити до опису алгоритму побудови дерева рішень, визначимо обов’язкові вимоги до структури даних і безпосередньо до самих даних, при виконанні яких алгоритм C4.5 буде працездатний.

Для правильної роботи алгоритму необхідні дані, що описують об'єкти предметної області, які мають бути представлені у вигляді таблиці з фіксованою кількістю атрибутів. Кожен атрибут має мати дискретне або числове значення і не повинен змінюватися від прикладу до прикладу. Крім того, кожен приклад повинен бути пов'язаний з конкретним класом, який обирається з одного з

атрибутів.

Дискретні класи. Класи мають бути дискретними, тобто мати кінцеве число значень. Кожен приклад повинен однозначно відноситися до конкретного класу. Випадки, коли приклади належать до класу з імовірнісними оцінками, виключаються. Кількість класів має бути значно меншою кількості прикладів [10].

Алгоритм побудови дерева. Нехай задана множина прикладів T, де кожен елемент цієї великої кількості описується m атрибутами. Кількість прикладів у множині T називатимемо потужністю цієї великої кількості і позначатимемо |T|.

Нехай мітка класу набуває наступних значень C1, C2, ..., Ck.

Наша мета полягає у побудові деревоподібної моделі класифікації з множини прикладів T. Процес побудови дерева починається з кореня і відбувається зверху вниз. На початковому етапі маємо порожнє дерево з коренем і множину T, яку асоціюємо з коренем. За допомогою аналізу атрибутів необхідно розбити множину T на підмножини. Кількість підмножин відповідає кількості значень вибраного атрибуту, а кожній підмножині відповідає нащадок кореня. Рекурсивно ця процедура застосовується до кожної підмножини (нащадка кореня) і т. д.

Розглянемо детальніше критерій вибору атрибуту, по якому повинно піти розгалуження. Очевидно, що в нашому розпорядженні m (по числу атрибутів) можливих варіантів, з яких ми повинні вибрати самий відповідний. Деякі алгоритми виключають повторне використання атрибуту при побудові дерева, але в нашому випадку ми таких обмежень накладати не будемо. Будь-який з атрибутів можна використати необмежену кількість разів при побудові дерева [11].

Нехай ми маємо перевірку X (в якості перевірки може бути вибраний будь-який атрибут), яка приймає n значень A1, A2 ... An. Тоді розбиття T по перевірці X дасть нам підмножини T1, T2 ... Tn, при X рівному відповідно до A1, A2. An. Єдина доступна нам інформація - та, яким чином класи розподілені у множині T і її підмножинах, що отримуються при розбитті по X. Саме цим ми і

скористаємося при визначенні критерію.

Нехай freq(Cj, S) freq(Cj, S) - кількість прикладів з деякої множини S, що відносяться до одного і того ж класу Cj. Тоді ймовірність того, що випадково вибраний приклад з множини S належатиме до класу Cj:

P=freq(Cj, S)|S|

P=freq(Cj, S)|S|

Згідно теорії інформації, кількість інформації, що міститься в повідомленні, залежить від її ймовірності:

$$\log2(1P)\ \log2\ \ (1P) \qquad (1.1)$$

Оскільки ми використовуємо логарифм з двійковою основою, то вираз (1.1) дає кількісну оцінку у бітах.

Вираз (1.2):

$$\text{Info}(T)=-\Sigma j=1k\text{freq}(Cj,\ T)|T|*\log2\text{freq}(Cj,\ T)|T|\text{Info}(T)=-\Sigma j=$$
$$1k\text{freq}(Cj,\ \ T)|T|*\log2\ \ \text{freq}(Cj,\ T)|T|, \qquad (1.2)$$

дає оцінку середньої кількості інформації, необхідної для визначення класу прикладу з множини T. У термінології теорії інформації вираз (1.2) називається ентропією множини T [12].

Ту ж оцінку, але тільки вже після розбиття множини T по X, дає наступний вираз:

$$\text{Infox}(T)=\Sigma i=1n|Ti||T|*\text{Info}(Ti)\ \text{Infox}(T)=\Sigma i=1n|Ti||T|*\text{Info}(Ti) \qquad (1.3)$$

Тоді критерієм для вибору атрибуту є наступна формула:

$$\text{Gain}(X)=\text{Info}(T)\ -\ \text{Infox}(T)\ \text{Gain}(X)=\text{Info}(T)\ -\ \text{Infox}(T) \qquad (1.4)$$

Критерій (1.4) обчислюється для усіх атрибутів. Вибирається атрибут, що

максимізував цей вираз. Цей атрибут буде перевіркою в поточному вузлі дерева, а потім по цьому атрибуту проводиться подальша побудова дерева. Тобто, у вузлі перевірятиметься значення по цьому атрибуту і подальший рух по дереву проводитиметься залежно від отриманої відповіді.

Такі ж міркування можна застосувати до отриманих підмножин T1, T2, ..., Tn і продовжити рекурсивно процес побудови дерева, до тих пір, поки у вузлі не опиняться приклади з одного класу.

Одне важливе зауваження: якщо в процесі роботи алгоритму отриманий вузол, що асоціюється з порожньою множиною (тобто жоден приклад не потрапив в цей вузол), то він позначається як листок, і в якості рішення листка вибирається клас, що найчастіше зустрічається, у безпосереднього предка цього листка.

Тут слід пояснити чому критерій (1.4) повинен максимізовуватися. З властивостей ентропії нам відомо, що максимально можливе значення ентропії досягається у тому випадку, коли усі його повідомлення рівноімовірні. У нашому випадку, ентропія (1.3) досягає свого максимуму коли частота появи класів в прикладах множини T рівноймовірна. Нам же необхідно вибрати такий атрибут, щоб при розбитті по ньому один з класів мав найбільшу ймовірність появи. Це можливо у тому випадку, коли ентропія (1.3) матиме мінімальне значення і, відповідно, критерій (1.4) досягне свого максимуму [13].

Як бути у випадку з числовими атрибутами? Зрозуміло, що слід вибрати деякий поріг, з яким повинні порівнюватися усі значення атрибуту. Нехай числовий атрибут має кінцеве число значень. Позначимо їх {v1, v2, ..., vn}. Заздалегідь відсортуємо усі значення. Тоді будь-яке значення, що лежить між vi і vi+1, ділить усі приклади на дві множини: ті, які лежать зліва від цього значення {v1, v2 . vi}, і ті, що справа {vi+1, vi+2 . vn}. В якості порогу можна вибрати середнє між значеннями vi і vi+1:

THi=vi+vi+12

THi=vi+vi+12

Таким чином, ми істотно спростили задачу знаходження порогу, і добилися

розгляду усього n-1 потенційних порогових значень TH1, TH2, ..., THn-1.

Формули (1.2), (1.3) і (1.4) послідовно застосовуються до усіх потенційних порогових значень і серед них вибирається те, яке дає максимальне значення за критерієм (1.4). Далі це значення порівнюється зі значеннями критерію (1.4), визначеним для інших атрибутів. Якщо з'ясується, що серед усіх атрибутів цей числовий атрибут має максимальне значення за критерієм (1.4), то в якості перевірки вибирається саме він.

Слід зазначити, що усі числові тести є бінарними, тобто ділять вузол дерева на дві гілки.

1.3 Опис методів кластеризації

Кластеризація є однією з найбільш важливих задач Data Mining. Нині розроблена велика кількість методів і алгоритмів кластеризації, але, на жаль, не усі вони можуть ефективно працювати з великими масивами даних, тому подальші дослідження в цьому напрямі пов'язані з подоланням цієї проблеми. Одним з широко відомих в аналітичному співтоваристві алгоритмів кластеризації, що дозволяють ефективно працювати з великими об'ємами даних, є EM-алгоритм.

Його назва походить від слів "expectation-maximization", що переводиться як "очікування-максимізація". Це пов'язано з тим, що кожна ітерація містить два кроки - обчислення математичних очікувань (expectation) і максимізацію (maximisation) [14]. Алгоритм заснований на методиці ітеративного обчислення оцінок максимальної правдоподібності, запропонованій в 1977 р. (A. P. Demster, N. M. Laird, D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm).

У основі ідеї EM-алгоритму лежить припущення, що досліджувана множина даних може бути змодельована за допомогою лінійної комбінації багатовимірних нормальних розподілів, а метою є оцінка параметрів розподілу, які максимізували логарифмічну функцію правдоподібності, що

використовується як міра якості моделі. Іншими словами, передбачається, що дані в кожному кластері підкоряються певному закону розподілу, а саме, нормальному розподілу. З урахуванням цього припущення можна визначити параметри - математичне очікування і дисперсію, які відповідають закону розподілу елементів в кластері, що якнайкраще "підходять" до спостережуваних даних (рисунок 1.3).

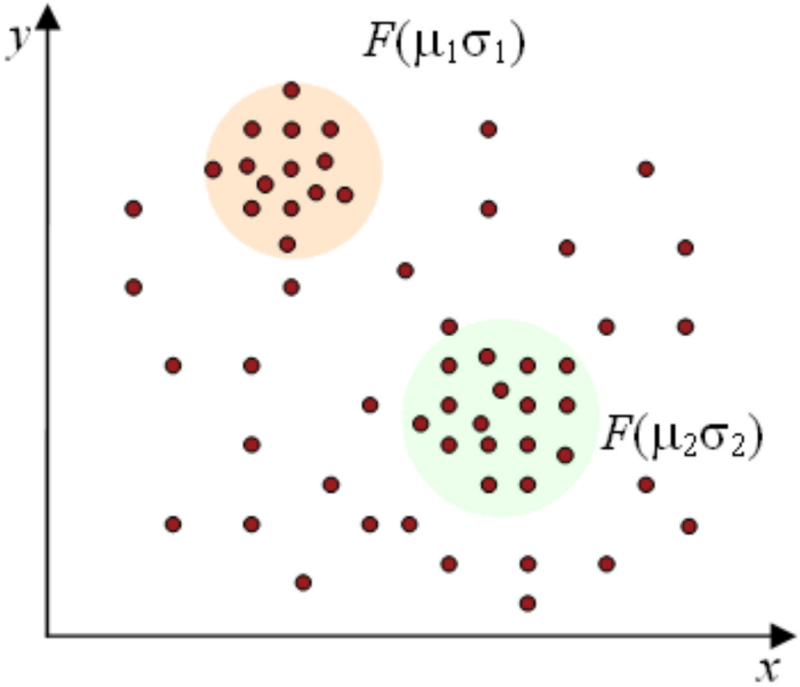

Рисунок 1.3 - Розподіл елементів в кластерах

Таким чином, ми припускаємо, що будь-яке спостереження належить до усіх кластерів, але з різною ймовірністю. Тоді задача полягатиме в "підгонці" розподілів суміші до даних, а потім у визначенні ймовірності приналежності спостереження до кожного кластера. Очевидно, що спостереження має бути віднесене до того кластера, для якого ця ймовірність вища.

Серед переваг ЕМ-алгоритму можна виділити наступні:

- потужна статистична основа;
- лінійне збільшення складності при рості об’єму даних;
- стійкість до шумів і пропусків в даних;
- можливість побудови бажаного числа кластерів;

- швидка збіжність при вдалій ініціалізації.

Проте алгоритм має і ряд недоліків. По-перше, припущення про нормальність усіх вимірів даних не завжди виконується. По-друге, при невдалій ініціалізації збіжність алгоритму може виявитися повільною. Окрім цього, алгоритм може зупинитися в локальному мінімумі і дати квазіоптимальне рішення [14].

Існують два підходи до рішення задач кластеризації: заснований на відстані і заснований на щільності. Перший підхід полягає у визначенні областей простору ознак, всередині яких точки даних розташовані ближче один до одного, ніж до точок інших областей, відносно деякої функції відстані (наприклад, Евклідової). Другий - виявляє області, які є більше "заселеними", ніж інші. Алгоритми кластеризації можуть працювати зверху вниз (ієрархічні) і від низу до верху (агломеративні). Агломаративні алгоритми, як правило, є точнішими, хоча і працюють повільніше [16].

Алгоритм EM заснований на обчисленні відстаней. Він може розглядатися як узагальнення кластеризації на основі аналізу сукупності імовірнісних розподілів. В процесі роботи алгоритму відбувається ітеративне поліпшення рішення, а зупинка здійснюється в момент, коли досягається необхідний рівень точності моделі. Мірою в даному випадку є статистична величина, що монотонно збільшується, називається логарифмічною правдоподібністю. Метою алгоритму є оцінка середніх значень CC, коваріацій RR і вагів сукупності WW для функції розподілу ймовірності, описаної вище. Параметри, оцінені алгоритмом, зберігаються в табличному вигляді.

Слід зазначити, що один з популярних алгоритмів кластеризації k-means є часткою випадком алгоритму EM, коли WW і RR постійні: wi=1kwi=1k, R=IR=I (II - одинична матриця).

Алгоритм розпочинає роботу з ініціалізації, тобто деякого наближеного рішення, яке може бути вибране випадково або задане користувачем виходячи з деяких апріорних відомостей про початкові дані. Найбільш загальним способом ініціалізації є присвоєння елементам матриці математичних очікувань

випадкових значень C←μ C←μ RandomRandom, початкова коваріаційна матриця визначається як одинична r←Ir←I, ваги кластерів задаються однаковими (wi←1kwi←1k).

Слід звернути увагу, що алгоритм може "застрягти" в локальному оптимумі і дати квазіоптимальне рішення при виборі невдалого початкового наближення. Тому одним з його недоліків слід рахувати чутливість до вибору початкового стану моделі.

Алгоритм містить два кроки: крок очікування (expectation) або E-крок і крок максимізації (maximization) або M-крок. Кожен з них повторюється до тих пір, поки зміна логарифмічної правдоподібності ΔllhΔllh не стане менше, ніж εε, або доки не буде досягнуте максимальне число ітерацій.

Логарифмічна правдоподібність обчислюється як:

llh=Σi=1nln(sumpi)

llh=Σi=1nln (sumpi)

Змінні δδ, RR, PP є матрицями, що зберігають відстані Махаланобіса, коваріації і ймовірність членства в кластері для кожної з nn точок. C, C,, R, R, і W, W, є тимчасовими матрицями, що використовуються тільки для обчислень. ||W||=1||W||=1, тобто Σi=1kwi=1Σi=1kwi=1. Позначення виду pipi, використане в псевдокоді, означає k-розмерний вектор приналежності ii-го спостереження до кожного з kk кластерів. Відповідно, xixi - нормована ймовірність приналежності до кожного з k кластерів. Стовпець CjCj матриці CC матриці є оцінка математичного очікування по jj-у кластеру, RR - діагональна матриця, тобто Rij=0Rij=0 для усіх i≠ji≠j. Із статистичної точки зору це означає, що коваріації є незалежними.

Діагональність є ключовим припущенням, яке робить алгоритм масштабованим. В цьому випадку детермінант матриці і його звернення може бути обчислений за час O(p), а алгоритм має складність O(kpn). У разі недіагональної матриці складність алгоритму складе O(kp2n), тобто квадратично зростатиме зі збільшенням розмірності даних.

Для оптимізації використовуваного об’єму пам’яті, алгоритм може

працювати в двох режимах. У першому завантажується тільки частина доступних даних і на їх основі робиться спроба побудови моделі. Якщо вона увінчалася успіхом, то алгоритм завершує роботу, інакше завантажується наступна порція даних і т. д., поки не будуть отримані прийнятні результати. У другому режимі завантажуються відразу усі наявні дані. Як правило, останній варіант забезпечує точнішу підгонку моделі, але пред'являє жорсткіші вимоги до об'єму доступної оперативної пам'яті.

K-Means. Алгоритм розділової кластеризації, заснований на розбитті множини елементів векторного простору на заздалегідь визначене число кластерів k (рисунок 1.4).

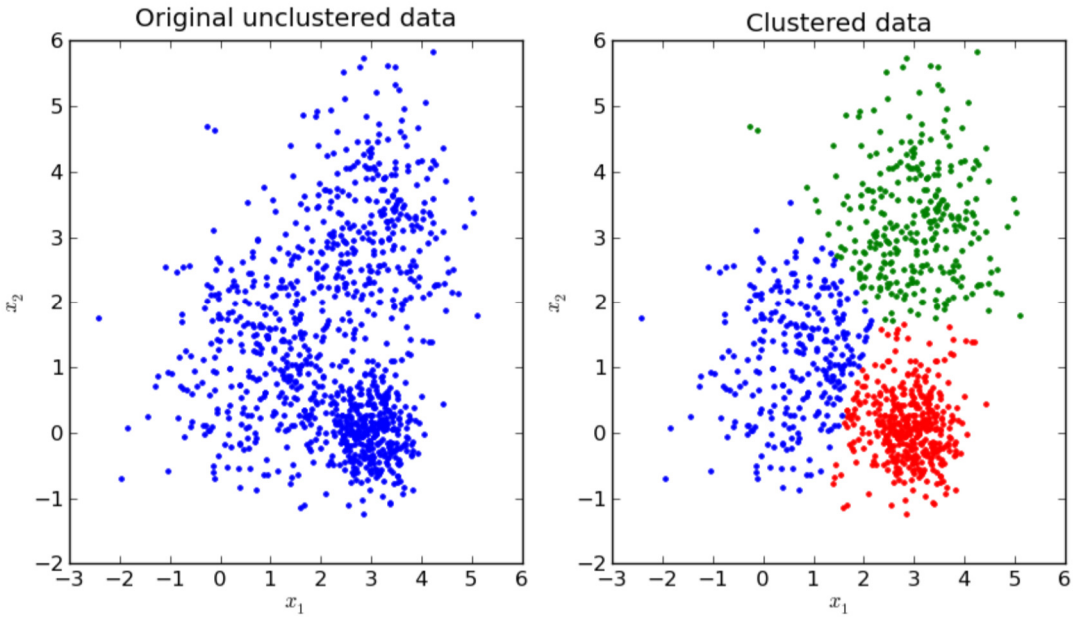

Рисунок 1.4 - Приклад кластеризації даних методом k-means

Алгоритм є ітераційною процедурою, в якій виконуються наступні кроки.

1. Вибирається число кластерів k;

2. З початкової множини даних випадковим чином вибираються k записів, які служитимуть початковими центрами кластерів;

3. Для кожного запису початкової вибірки визначається найближчий до неї центр кластера. При цьому записи, «притягнуті» певним центром, утворюють початкові кластери;

4. Обчислюються центроїди - центри тяжіння кластерів. Кожен центроїд - це вектор, елементи якого є середніми значеннями ознак, вичисленими по усіх записах кластера. Потім центр кластера зміщується в його центроїд [17].

Потім 3-й і 4-й кроки ітеративно повторюються. Очевидно, що на кожній ітерації відбувається зміна меж кластерів і зміщення їх центрів. В результаті мінімізується відстань між елементами всередині кластерів.

Зупинка алгоритму проводиться тоді, коли межі кластерів і розташування центроїдів не перестануть змінюватися від ітерації до ітерації, тобто на кожній ітерації в кожному кластері залишатиметься один і той же набір записів. На практиці алгоритм зазвичай знаходить набір стабільних кластерів за декілька десятків ітерацій.

Перевагою алгоритму є швидкість і простота реалізації. До його недоліків можна віднести невизначеність вибору початкових центрів кластерів, а також те, що число кластерів має бути задане спочатку, що може зажадати деякої апріорної інформації про початкові дані [18].

1.4 Постановка задачі дослідження

У роботі розглянуті різні постановки задачі класифікації: класифікація трафіку згідно з категоріями, додатками або виявлення трафіку категорії P2P як такої, що найважче ідентифікується.

Об’єктами класифікації є потоки: однонапрямлені, двонаправлені, повні TCP-сесії, IP-трафік між хостами. Для обчислення атрибутів, що відповідають потокам, використовується інформація про заголовки пакетів і час їх прибуття. "Абсолютна" істина про приналежність потоку до категорії трафіку будується на основі повного вмісту пакетів.

При дослідженні алгоритмів вивчені факти, що впливають на якість класифікації:

1) важливість вибору набору несуперечливих і ненадмірних атрибутів і залежність набору від зразків даних;

2) необхідність вибору різних наборів атрибутів і асоціативних правил для TCP - і UDP-потоків, незважаючи на те що обидва типи потоків можуть одночасно використовуватися одним і тим же додатком, як, наприклад, у разі P2P;

3) чутливість до вибору параметрів розподілу значень атрибутів всередині класу;

4) застосовність алгоритму до однонапрямлених потоків трафіку;

5) вплив розмірів блоків зразків даних.

Розглянуті особливості контрольованої і частково контрольованої класифікації і алгоритмів аналізу даних стосовно класифікації трафіку. Залежність від реалізації застосувань можна ослабити, якщо частіше проводити перенавчання. Проте в цьому випадку виникає проблема складності перевірки: для невеликих об’ємів даних простіше встановити "абсолютну" істину відносно представлених класів застосувань, але недостовірні параметри ефективності, а великі об’єми важко проаналізувати.

Проведення порівняння результатів класифікації, представлених в різних роботах, істотно ускладнене внаслідок використання різних заходів ефективності алгоритмів, обчислення ефективності відносно різних структур даних (байти, пакети, потоки) і відмінності досліджуваних блоків даних.

Для роботи алгоритмів Data Mining необхідно підготувати початкові дані, вибрати атрибути для аналізу і параметри роботи алгоритмів. Початковими даними є захоплені мережеві пакети, атрибутами - поля пакетів на IP, TCP і HTTP рівнях.

З метою поліпшення якості даних, слід провести первинний факторіальний аналіз, щоб виявити незначні параметри. До незначних параметрів відносяться атрибути, що описують час і місце проведення експерименту. З точки зору предметної області DDoS атак, IP адреса відправника є незначним параметром, оскільки часто вона є підміненою. Крім того, IP адреса жертви так само виключається. Ці параметри є інформаційними - не дають загальне розуміння проблеми, але непридатні для аналізу: вони специфічні для конкретних умов, не

є загальним випадком. Якщо застосовувати один з методів Data Mining до вищеперелічених атрибутів, то вони переважатимуть, тим самим приховуючи внутрішні характеристики трафіку.

Дерева рішень мають швидкий процес навчання. На побудову класифікаційних моделей вимагається значно менше часу, ніж, наприклад, на навчання нейронних мереж.

При автоматичному визначенні кількості кластерів g-means, кластерний аналіз займає на порядок більше часу, в порівнянні з деревами рішень. При фіксованій кількості кластерів k-means аналіз проходить швидше, проте це вимагає від аналітика зразкового розуміння кількості кінцевих кластерів, що в реальних умовах складно зробити.

Нейронні мережі займають проміжне положення - оптимальний час і результат досягаються при зниженні в два рази на кожному шарі кількості нейронів, в порівнянні з початковою кількістю параметрів. Передбачалося використання нейронних мереж, проте використання для машинного навчання нейронної мережі викликає подальші складнощі, оскільки вона є чорним ящиком, який буде складно розкласти для прикладної програмної реалізації. Таким чином, використання дерев рішень дозволить чітко виділяти правила і застосовувати їх в конкретній реалізації в режимі реального часу.

Отже, метою кваліфікаційної роботи є підвищення точності класифікації мережевого трафіку на основі методів інтелектуального аналізу даних.

Висновки до даного розділу:

1. У даній роботі розглядається проблема аналізу IP-трафіку з використанням методів Data Mining, до числа яких відносяться класифікація – розділення об’єктів на задані групи (класи) згідно з характеристиками об’єктів; регресія – пошук функції, що моделює множину об’єктів, що вивчаються, з найменшою помилкою; кластеризація – пошук незалежних груп об’єктів, кластерів і їх характеристик (групи заздалегідь не обумовлені); пошук асоціативних правил – характерних залежностей між об’єктами або подіями.

2. Для роботи алгоритмів Data Mining необхідно підготувати початкові

дані, вибрати атрибути для аналізу і параметри роботи алгоритмів. З метою поліпшення якості даних, слід провести первинний факторіальний аналіз, щоб виявити незначні параметри.

3. При дослідженні алгоритмів вивчені факти, що впливають на якість класифікації:

1) важливість вибору набору несуперечливих і ненадмірних атрибутів і залежність набору від зразків даних;

2) необхідність вибору різних наборів атрибутів і асоціативних правил для TCP- і UDP-потоків, незважаючи на те що обидва типи потоків можуть одночасно використовуватися одним і тим же додатком, як, наприклад, у разі P2P;

3) чутливість до вибору параметрів розподілу значень атрибутів всередині класу;

4) застосовність алгоритму до однонапрямлених потоків трафіку;

5) вплив розмірів блоків зразків даних.

2 МЕТОДИ ТА ЗАСОБИ ВИЯВЛЕННЯ ПРИХОВАНИХ АТАК НА ОСНОВІ ТЕХНОЛОГІЙ DATA MINING

2.1 Інтелектуальний збір даних

У телекомунікаційних компаніях (ТК) з використанням БД збираються щодня великі об’єми інформації. З такого об’єму даних необхідно витягнути цінну інформацію, що стосується роботи компанії і її клієнтів. Для цієї мети добре підходить технологія Data Mining - як мультидисциплінарна область, що виникла і розвивається на базі таких наук як прикладна статистика, класифікація і кластеризація, розпізнавання образів, штучний інтелект, теорія баз даних та ін. Тому тут добре проглядається інтеграція теорії прикладної статистики аналізу даних з очищенням даних, навчанням і візуалізацією результатів.

Стосовно обробки інформації ТК, головна проблема полягає в тому, що часто заздалегідь невідомо, як групувати ці дані, як знаходити зв’язки між подіями і т. д. Для часткового вирішення цієї проблеми використовуються методи класифікації і кластеризації Data Mining.

За допомогою їх можна об’єднувати дані за будь-якими вибраними ознаками. Наприклад, знаходити взаємозв’язок події від навантаження в мережі або дня тижня і т. п.

В якості прикладу накопичення великих об’ємів даних розглянемо телекомунікаційні компанії, які щодня збирають інформацію про трафік користувачів. Ставиться наступне завдання: із застосуванням відомих методів інтелектуального аналізу усього об’єму інформації, отриманого за один місяць роботи, виявити приховані атаки сервера телекомунікаційної компанії. Під прихованими атаками ми маємо на увазі умисне або ненавмисне генерування однотипного і такого, що не має сенсу трафіку, який навантажує мережеве устаткування, тим самим заважаючи проходженню нормального трафіку через мережеві вузли [19].

Вирішення задачі включатиме наступні етапи:

1) збір трафіку за певний період, під час якого ймовірно були здійснені

«приховані атаки»;

2) за допомогою методів кластерного аналізу необхідно виявити аномалії (нестандартна поведінка клієнтів);

3) проаналізувати отримані результати і вибрати ознаки, по яких можна надалі виявляти такі аномалії;

4) промаркувати вектори даних і навчити машину для подальшого процесу перевірки трафіку в майбутньому.

Однією з телекомунікаційних компаній був наданий великий масив даних (135 Гб у форматі .txt) про трафік, який клієнти генерують щодня. Він містить в собі в числі інших, наступну інформацію: ID аккаунта, IP адресу джерела, IP адресу одержувача, розмір пакету даних, номер порту джерела, номера порту одержувача, день і час події в Unix форматі. В цілому приблизно 45 мільйонів рядків наступного вигляду (рисунок 2.1):

timestamp	1456634876
account_id	0
source	192.168.101.3
destination	178.218.82.79
t_class	2210
packets	1
bytes	258
sport	53
dport	57577
date	Sun Feb 28 08:47:56 2020

Рисунок 2.1 - Приклад рядка наданих даних

Вибір таких атрибутів як: ID аккаунт, об’єм трафіку за добу в гігабайтах, день тижня, порядковий номер дня, тип дня (вихідний - 0, робочий - 1), кількість запитів за добу в першу чергу обгрунтований поставленим завданням. Об’єм трафіку за добу, це важливий показник активності для клієнта

телекомунікаційної компанії, чим менше він його згенерує, тим менше потрібно буде платити провайдерові за канал. За об'ємом трафіку видно, яку ширину смуги допустимого каналу використовує клієнт, відносно того, який він придбав. День тижня, порядковий номер дня і тип дня дозволяють в процесі проведення ручного аналізу визначати «допустимість» поведінки того або іншого клієнта, оскільки робочий день і вихідний сильно відрізняються по своїй активності користувачів. Кількість запитів за добу, це той показник, який дозволить нам визначити, чи адекватну кількість запитів робить клієнт, відносно часу використовуваного каналу [20].

Це і є класична задача Data Mining, коли з дуже великого об'єму даних вимагається витягнути найпотрібнішу і ціннішу інформацію. Провести аналіз таких даних вручну не представляється можливим, тому було прийнято рішення перетворити дані в потрібний формат і застосувати до рішення поставленої задачі статистичні методи кластеризації і класифікації. Крім того, для аналізу таких об'ємів інформації також неможливо застосувати готові статистичні пакети. Відомий метод класифікації на основі кореляційного аналізу також тут не може бути застосований внаслідок того, що вийде кореляційна матриця дуже великого порядку. Алгоритм k-means вибраний як найпростіший і добре описаний спосіб виявлення викидів в наявних початкових даних.

Нижче приведена послідовність дій, необхідних для вирішення поставленої задачі.

2.2 Попередня обробка даних

Для виявлення співвідношення об'єму трафіку і активності клієнта в цілому вирішено перетворити дані в наступний формат: ID аккаунта, об'єм трафіку за добу в гігабайтах, день тижня, порядковий номер дня, тип дня (вихідний - 0, робочий - 1), кількість запитів за добу - всього 6 впорядкованих ознак кожного вектору даних. Таким чином, признаковий простір для вирішення поставленої задачі включає 6 вищеперерахованих ознак.

Для такого перетворення була написана програма на мові C#, яка відкривала по черзі .txt файли на читання, складала трафік за добу кожного клієнта і зчитувала кількість звернень цього клієнта до сервера. Після перетворень було отримано 60473 рядків інформації, і, якщо поділити це число на кількість днів в місяці, то ми отримаємо кількість активних абонентів компанії. Кожен рядок, це інформація про клієнта за день. Нижче на рисунку 2.2 наведений приклад даних для п’яти клієнтів компанії з 6-ма впорядкованими ознаками:

```
0,30.0108,1,1,1,13883303
1,0.007,1,1,1,4573
10,0.0341,1,1,1,3333
29,0.7152,1,1,1,38790
66,0.0716,1,1,1,4737
```

Рисунок 2.2 - Приклад даних після перетворення

Алгоритм роботи цієї програми розбивається на декілька циклічних етапів:

1. Відкриття текстових файлів з інформацією, поки не пройшли по усіх файлах.

2. Порядкове зчитування інформації, поки не кінець файлу.

3. Розбір рядка на складові: id аккаунта, розмір пакету і т. д.

4. Підсумовування об’єму трафіку.

5. Підрахунок кількості звернень до сервера.

6. Перетворення даних у формат CSV і збереження отриманих даних у файл.

Схема алгоритму роботи програми показана на рисунку 2.3. Після первинної обробки вимагається розбити ці дані на групи і проаналізувати отриманий результат [21]. При цьому апріорі передбачається, що серед клієнтів компанії за двома ознаками (2-ю і 6-ю) можуть бути не стандартні. Для цього був вибраний метод кластеризації k-середніх.

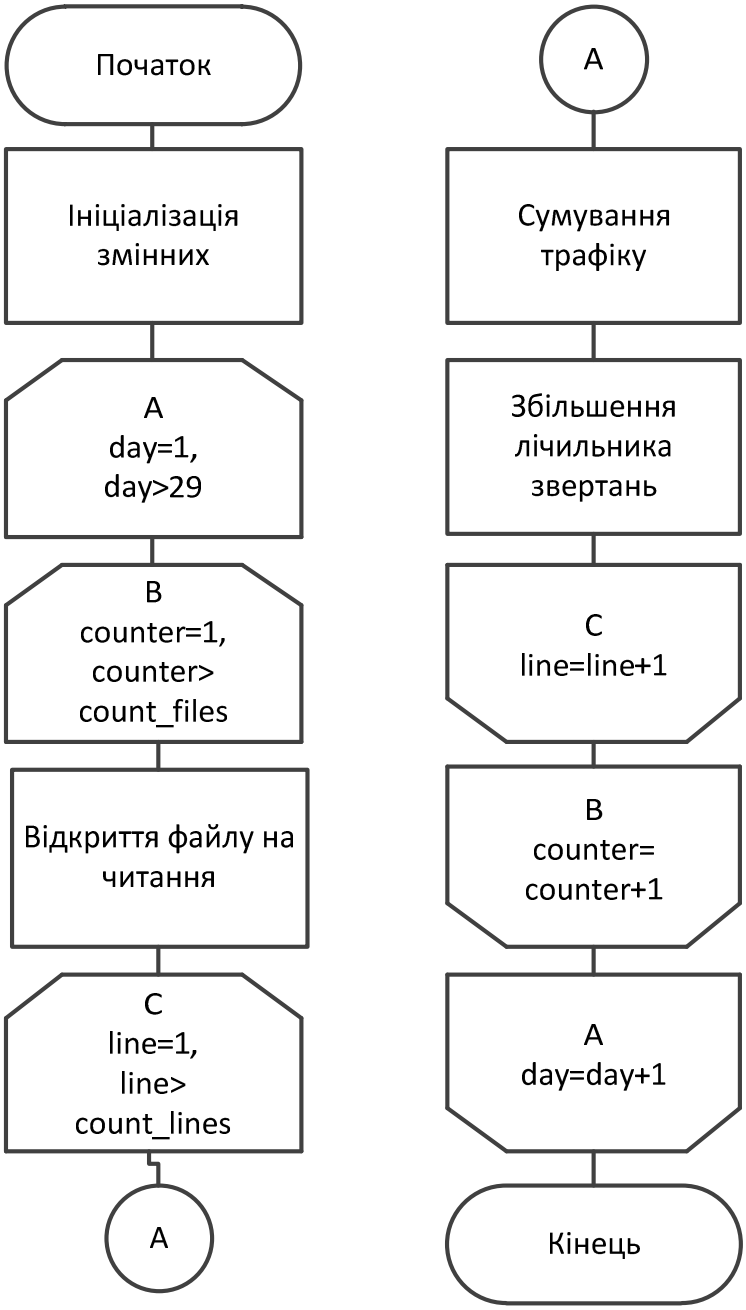

Рисунок 2.3 - Схема алгоритму роботи програми

2.3 Кластеризація даних і аналіз отриманих результатів

Точність розбиття отриманих векторів по кластерах залежить від спочатку вибраних центроїдів. Нами вибраний метод ініціалізації центроїдів під назвою k-means++. Це поліпшена версія алгоритму кластеризації k-середніх. Суть поліпшення полягає в знаходженні більш «хороших» початкових значень центроїдів кластерів. Алгоритм методу включає 5 основних кроків.

Ініціалізація:

1. Вибрати перший центроїд випадковим чином (серед усіх точок).

2. Для кожної точки знайти значення квадрата відстані до найближчого центроїда (з тих, які вже вибрані) $(dx)^2$.

3. Вибрати з цих точок наступний центроїд так, щоб ймовірність вибору точки була пропорційна обчисленому для неї квадрату відстані.

4. Це можна зробити таким чином. На кроці 2 треба паралельно з розрахунком $(dx)^2$ підраховувати суму $Sum(dx^2)$. Після накопичення суми знайти значення Rnd=random(0.0,1.0)*Sum. Генератор Rnd випадковим чином вкаже на число з інтервалу [0; Sum), і нам залишається тільки визначити, якій точці це відповідає.

Для цього треба знову почати підраховувати суму $S(dx^2)$ до тих пір, поки сума не перевищить Rnd. Як тільки це станеться, підсумовування зупиняється, і ми можемо взяти поточну точку в якості центроїда. При виборі кожного наступного центроїда спеціально стежити за тим, щоб він не співпав з однією із вже вибраних в якості центроїдів точок не треба, оскільки ймовірність повторного вибору деякої точки дорівнює 0.

5. Повторювати кроки 2 і 3 до тих пір, поки не будуть знайдені усі необхідні центроїди.

Далі виконується основний алгоритм k-середніх. Це найбільш популярний метод кластеризації, який був винайдений ще в 1950-х роках математиком Гуго Штейнгаузом і майже одночасно Стюартом Ллойдом.

Дія алгоритму така, що він прагне мінімізувати сумарне квадратичне

відхилення точок кластерів від центрів цих кластерів:

$$V = \sum_{i=1}^{k} \sum_{x_i \in S_i} (x_j - \mu_i)^2 \rightarrow min, \quad (2.1)$$

де k – число кластерів,

S_i – отримані кластери,

i=1,2,..,k

μ_i – центри мас векторів x_j, що належать S_i.

Для проведення кластерного аналізу була написана програма на мові Python, з підключеною бібліотекою sklearn, яка містить в собі реалізації більшості математичних методів інтелектуального аналізу даних. Укрупнена схема алгоритму програми наведена на рисунку 2.4.

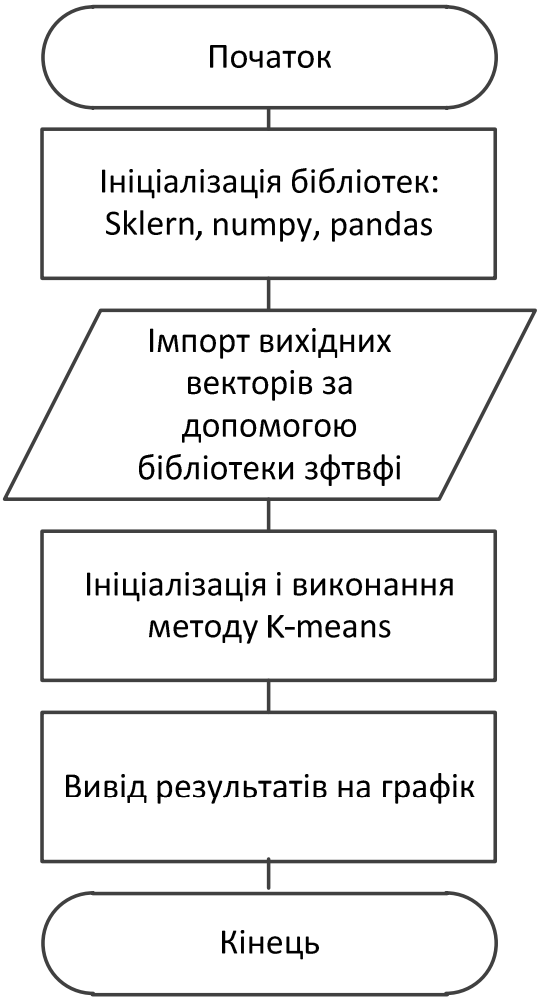

Рисунок 2.4 - Схема алгоритму програми кластеризації

Застосувавши [22] поєднаний метод k-середніх з методом ініціалізації центроїдів k-means++, і розбивши дані на 2 кластери, отримаємо наступні результати:

1-й кластер включає 60427 векторів;

2-й кластер - 46 векторів.

На рисунку 2.5 можна подивитися графічний розподіл векторів (записів) по кластерах.

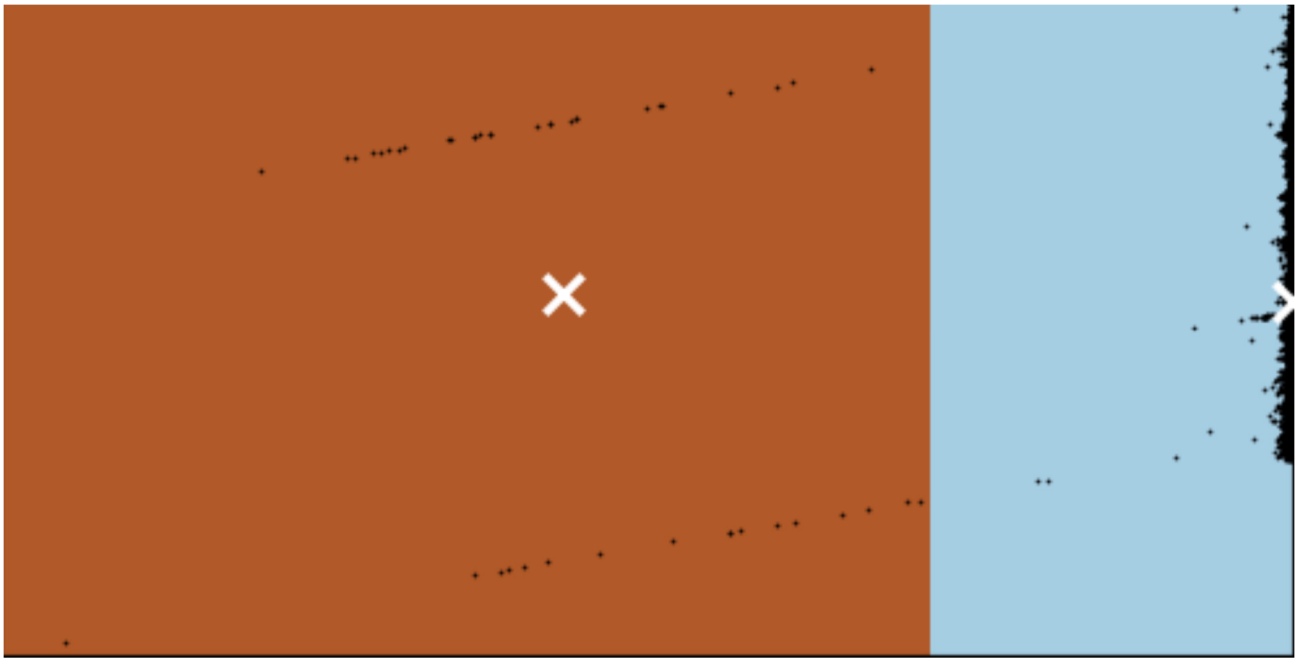

Рисунок 2.5 - Кластери, отримані методом k-середніх

Тут по осі X вказана кількість відповідей від сервера, а по осі Y - account id. Білими хрестиками відображені центроїди кластерів.

Як можна інтерпретувати отримані дані? Перший кластер (справа) містить інформацію про стандартні записи, які не виділяються з маси. Ця стандартна поведінка клієнтів протягом усього місяця. Другий кластер (ліворуч) містить інформацію про нестандартні записи, що виділилися з маси, містять приховані атаки на компанію. Ця ненормальна поведінка для клієнтів, швидше за все на стороні клієнтів з такими ознаками можна виявити вірусну активність, або ще якесь «погане» ПЗ, що викликають дуже велике число звернень до сервера ТК.

2.4 Побудова класифікатора і його навчання

Для того, щоб програмне забезпечення надалі само автоматично визначало приналежність того або іншого запису до виділених вже кластерів, сформованих вище, необхідно застосувати метод класифікації. Для цього розглянемо метод опорних векторів.

Метод опорних векторів спочатку відноситься до бінарних класифікаторів. Він належить до сімейства лінійних класифікаторів.

Ідею методу простіше пояснити на простому прикладі: дані точки на площині і їх потрібно розбити на два класи. Тоді в якості розділяючої прямої між класами виберемо ту, відстань від якої до кожного класу максимальна. У загальному випадку замість розділяючої прямої буде гіперплощина з максимальною відстанню до кожного класу. Метод опорних векторів будує класифікуючу функцію F у вигляді $F(x) = sing(\langle w, x\rangle + b)$, де $\langle w, x\rangle$ скалярний добуток векторів, w – нормальний вектор до розділяючої гіперплощини, а b – проміжний параметр.

Так як гіперплощина може бути задана у виді $\langle w, x\rangle + b$ для деяких w і b, то необхідно визначити такі w і b, які максимізували відстань до кожного класу.

Для застосування методу класифікації до даних кожен запис був вручну помічений приналежністю до конкретного класу [23]. Вважатимемо ці дані навчальною вибіркою, оскільки дані промарковані відповідно до ручної класифікації, а не застосування алгоритмів машинного навчання. Маркування векторів проводилося відповідно до суб'єктивного погляду авторів на «приховані атаки». Тепер дані виглядають таким чином (рисунок 2.6). Введено нове поле target, яке повинне зберігати в собі інформацію про приналежність кожного запису до того або іншого класу.

account_id	0
traffic	30.0108
count	13883303
dayOfWeek	1
n_day	1
type_day	1
month	2
target	1

Рисунок 2.6 - Вигляд даних

Тут поле target - це класифікатор: 1 - відносить дані до першого кластера, 0 - до другого.

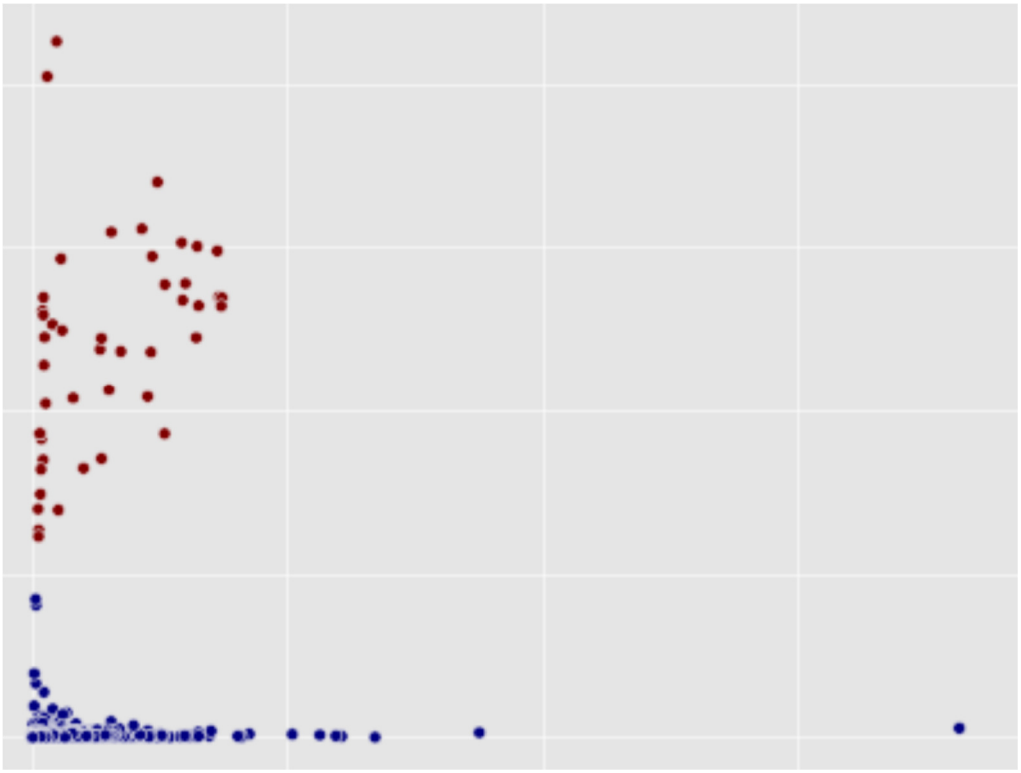

Рисунок 2.7 - Графічний вигляд класифікованих даних

На рисунку 2.7 відображені класифіковані дані по полю target: червоним кольором позначені вектори, що належать до кластера 1, синім кольором - до кластера 2.

Написана програма на мові Python, з використанням тієї ж бібліотеки sklearn, яка класифікує нові дані за допомогою методу опорних векторів і виводить інформацію про приналежність цих даних до того або іншого класу. Узагальнена схема алгоритму програми наведена на рисунку 2.8.

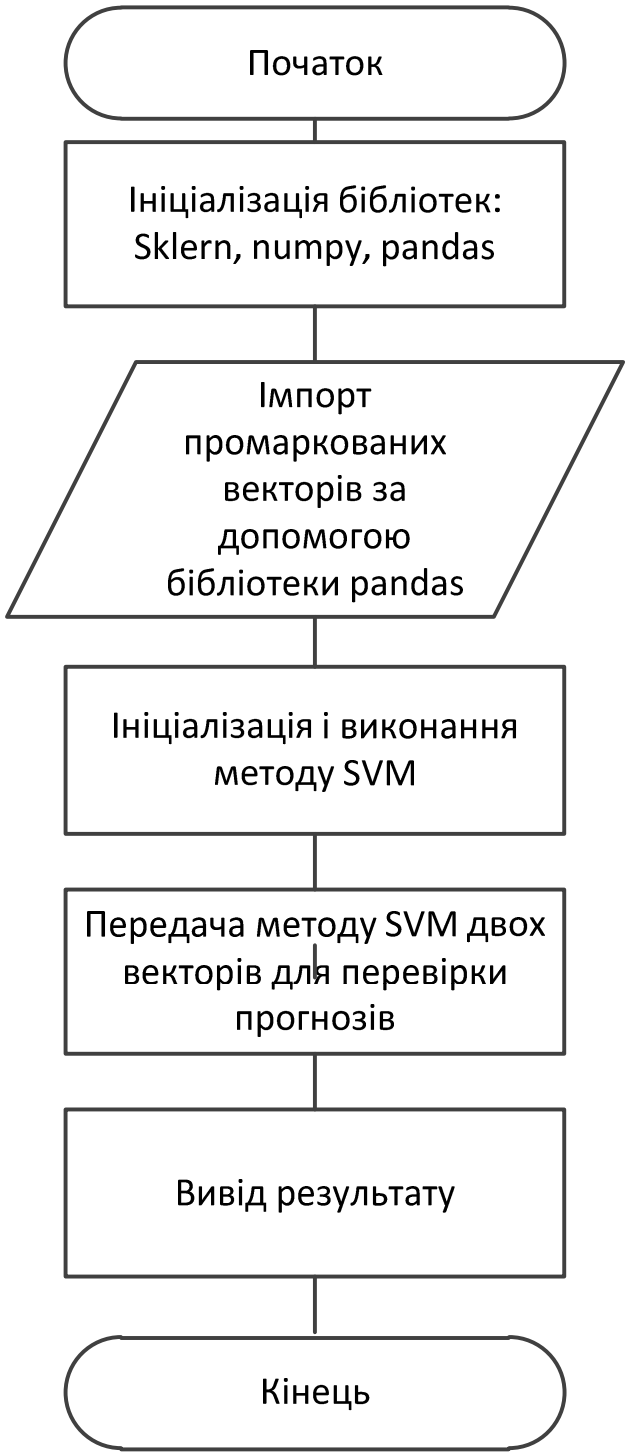

Рисунок 2.8 - Схема алгоритму програми класифікації даних

На основі досвіду авторів по знаходженню прихованих атак в ручному режимі, з історії взяті записи, що відповідають «прихованій атаці» і нормальній поведінці. Приймемо ці дані за контрольну вибірку. Узято всього два записи, для того, щоб читач наочно бачив значення атрибутів, які передаються машині, а не сухі дані про те, скільки записів вийшло що належать тому або іншому класу [24].

Для переконливості перевіримо приналежність до кластерів наступні записи (розмір трафіку в гігабайтах, кількість відповідей від сервера, день тижня, день місяця, тип дня, місяць):

1. 30.0108,13883303,1,1,2,1.

2. 0.007,4573,1,1,2,1,0.

В результаті програма дає відповідь: 1-й запис належить до першого кластера, 2-а - до другого кластера, що повністю підтверджує правильність роботи класифікатора. За властивостями першого запису видно, що було виконано дуже багато запитів до сервера, тому віднесемо це до «прихованої атаки». Другий запис еквівалентний нормальній поведінці клієнта.

Висновки до другого розділу:

1. Проведено повний спектр аналізу даних телекомунікаційної компанії. Спочатку компанією були надані сирі дані за лютий місяць, що містять службову інформацію про IP-потоках. Для проведення повного аналізу написана програма на мові C#, що агрегує дані по днях, таким чином були виділені ознаки, по яких надалі здійснюватиметься класифікація за відомими методами.

2. Проведено кластерний аналіз за допомогою програми на мові Python, виявлені два кластери: середньостатистичні клієнти і «особливі», які представляють загрозу телекомунікаційної компанії. Інформація відображена на графіках. За цими даними було проведено навчання вибірки великого об’єму (60427 векторів) з введенням поля «target».

3. Застосовано метод опорних векторів для класифікації нових даних, що поступають в реальному часі. Навчений класифікатор може бути далі застосовуватися для аналізу нових даних телекомунікаційної компанії.

3 ПРАКТИЧНЕ ЗАСТОСУВАННЯ МЕТОДІВ КЛАСИФІКАЦІЇ МЕРЕЖЕВОГО ТРАФІКУ З ВИКОРИСТАННЯ ПРОТОКОЛУ NETFLOW

3.1 Класифікація мережевого трафіку із застосуванням протоколу NetFlow і машинного навчання

Клієнти, особливо ті, хто не має досвіду в IT-бізнесі, часто задають питання щодо різниці між послугами для корпоративних клієнтів і приватних абонентів. Як відрізнити однакові послуги для різних замовників, і чому вартість однакових параметрів послуг може відрізнятись так суттєво? Відмінність полягає в тому, що корпоративні та масові послуги ґрунтуються на принципово різних підходах, незважаючи на те, що на перший погляд вони можуть бути подібними. Корпоративний замовник, заплативши більше, отримує гарантії доступності та якості послуг, тоді як для масового ринку надається стандартний набір послуг за принципом "as is" [26].

Квінтесенцією першого підходу, його наочним втіленням можна вважати Угоду про Рівень Обслуговування (Service Level Agreement - SLA). Цей документ є невід'ємною частиною будь-якого контракту на обслуговування бізнесклієнтів і сприймається як стандарт дефакто в корпоративному сегменті. Він дозволяє формалізувати і зробити предметом обговорення рівень обслуговування, який постачальник пропонує своєму клієнтові. Наявність гарантованих рівнів обслуговування є фундаментальним чинником ціноутворення корпоративних рішень. Однією з головних проблем телекомунікаційної компанії в цій області є надання звіту про якість послуг, що надаються, перед замовником.

Одна справа суб'єктивне бачення замовника про «недостатню» швидкість, а інше звіти про увесь трафік клієнта, з усією вичерпною інформацією. Для цього спробуємо розв'язати проблему за допомогою класифікації трафіку і отриманні про нього детальної інформації.

Визначення типу трафіку на зорі мережі Інтернет не складало жодної проблеми. Кожному протоколу або групі застосувань відповідав свій порт, через який ідентифікація трафіку проводилася без особливих проблем. З розвитком

мережі Інтернет такий спосіб ідентифікації перестав працювати, оскільки вихідний порт і порт призначення вже не були прив'язані до певного протоколу або групи застосувань. Більше того, у більшості випадків порти міняються динамічно протягом одного сеансу. На зміну такому методу ідентифікації прийшов метод глибокого аналізу пакетів (DPI).

Deep Packet Inspection – технологія накопичення статистичних даних, перевірки і фільтрації мережевих пакетів по їх вмісту. На відміну від брандмауерів, Deep Packet Inspection аналізує не лише заголовки пакетів, але і повний вміст трафіку на рівнях моделі OSI з другого і вище. Deep Packet Inspection здатна виявляти і блокувати віруси, фільтрувати інформацію, що не задовольняє заданим критеріям. Ця технологія має один істотний недолік – вона вимагає великих апаратних ресурсів. Оскільки вимагається перехопити пакет, проаналізувати його по моделі OSI з другого рівня і вище, прийняти рішення про його подальшу долю, а найголовніше витримати норму швидкості, яку надає провайдер. Якщо устаткування не справлятиметься – пакети проходитимуть без аналізу або вибудовуватимуться в чергу, що спричинить зниження швидкості. Витонченіший варіант і менш витратний по апаратному ресурсу був запропонований зовсім нещодавно – машинне навчання за непрямими ознаками мережевого трафіку. Для цього можна застосувати протокол NetFlow.

NetFlow – відкритий протокол, розроблений Cisco для моніторингу трафіку в мережі. Netflow надає можливість аналізу мережевого трафіку на рівні сеансів, роблячи запис про кожну транзакцію TCP/IP. Кожна транзакція містить в собі наступні дані про IP-потоки: дата і час початку потоку, дата і час кінця потоку, тривалість потоку за часом в секундах, протокол передачі даних, IP джерела, IP одержувача, кількість переданих пакетів, кількість переданих байт, порт джерела, порт одержувача, тип обслуговування (ToS), кількість пакетів в секунду, кількість біт в секунду, кількість байт на пакет. Об'єднавши однонапрямлені потоки в двонаправлені виходить вектор непрямих ознак: TX тривалість потоку, RX тривалість потоку, TX к-ть пакетів, RX к-ть пакетів, TX к-ть байт, RX к-ть байт, протокол, порт джерела, порт одержувача, тип

обслуговування, TX кількість пакетів в секунду, RX кількість пакетів в секунду, TX кількість біт в секунду, RX кількість біт в секунду, TX кількість байт на пакет, RX кількість байт на пакет. Де TX – передача в один бік (запит), а RX передача в зворотній (відповідь). Таким чином виходить 17 ознак (рисунок 3.1), по яких можна відрізнити один протокол від іншого. Далі треба проаналізувати трафік і поставити кожному двонаправленому потоку свій маркер, по якому можна буде навчити машину і проводити автоматичну класифікацію.

N потока	ToS	DstPort	TX_Bytes	TX_Duration	TX_Packets	TX_bps	TX_pps	TX_bpp
1	0	53	192	1.7	3	903	1	64
2	0	12798	258	0	2	0	0	129
N потока	Protocol	SrcPort	RX_Bytes	RX_Duration	RX_Packets	RX_bps	RX_pps	RX_bpp
1	UDP	2174	903	1.7	3	4249	1	301
2	UDP	7014	738	0	2	0	0	369

Рисунок 3.1 - Приклад NetFlow даних у вигляді векторів

Для аналізу трафіку повертаємося до вже згаданої технології DPI. Існує бібліотека nDPI [27], яка дозволяє аналізувати трафік в реальному часі, а також з файлів, які містять захоплені пакети (PCAP файли). Після аналізу у вихідному файлі міститься інформація про трафік, з позначкою до якого протоколу він відноситься. Бібліотека nDPI на момент написання цього тексту містить інформацію про 185 протоколів. Наступним кроком буде конвертація цього ж трафіку у формат протоколу NetFlow. Це робиться при використанні програми-конвертора softflowd і колектора ncapd. Робота йде за принципом передачі-прийому. Трафік з PCAP файлу передається у вигляді UDP пакетів на колектор, який зберігає дані у бінарному вигляді. За таким же принципом збирається реальний трафік з мережевого устаткування. Далі з бінарного вигляду в текстовий дані конвертуються за допомогою програми nfdump. У результаті виходить трафік, розбитий на однонапрямлені потоки. Для подальшої обробки написана програма на мові C#.

Програма зчитує дані з вихідних файлів після аналізу nDPI і nfdump.

Створює з однонапрямлених потоків двонаправлені. Стикує отримані результати з протоколом, який видала бібліотека nDPI. Зрештою виходять вектори промаркованих даних, по яких можна навчати машину.

3.2 Навчання ієрархічної класифікаційної моделі

Для того, щоб абстрагуватися від процесу розробки математичних методів, які вже і так розроблені, використовується Accord .NET framework. Це повноцінне середовище для обробки статистичних даних. Містить в собі реалізації усіх популярних методів машинного навчання.

Серед запропонованих бібліотекою методів класифікації вибраний C4.5. Це алгоритм для побудови дерев рішень, розроблений Джоном Квінланом (англ. John Ross Quinlan). C4.5 є вдосконаленою версією алгоритму ID3 того ж автора. Перш ніж приступити до опису алгоритму побудови дерева рішень, визначимо обов'язкові вимоги до структури даних і безпосередньо до самих даних, при виконанні яких алгоритм C4.5 буде працездатний:

1. Опис атрибутів. Дані, необхідні для роботи алгоритму, мають бути представлені у вигляді плоскої таблиці. Уся інформація про об'єкти (далі приклади) з предметної області повинна описуватися у вигляді кінцевого набору ознак (далі атрибути). Кожен атрибут повинен мати дискретне або числове значення. Самі атрибути не повинні мінятися від прикладу до прикладу, і кількість атрибутів має бути фіксованою для усіх прикладів.

2. Визначені класи. Кожен приклад має бути асоційований з конкретним класом, тобто один з атрибутів має бути вибраний в якості мітки класу.

3. Дискретні класи. Класи мають бути дискретними, тобто мати кінцеве число значень. Кожен приклад повинен однозначно відноситися до конкретного класу. Випадки, коли приклади належать до класу з ймовірнісними оцінками, виключаються. Кількість класів має бути значно меншою кількості прикладів.

У нашому випадку враховані усі три пункти. Атрибути описані за допомогою сімнадцяти параметрів, що недвозначно описують кожен вектор.

Визначені класи за допомогою «ручної» класифікації (DPI). Класи дискретні, один і той же потік не може належати до декількох видів трафіку.

Алгоритм побудови дерева. Нехай задана множина прикладів T, де кожен елемент цієї великої кількості описується m атрибутами. Кількість прикладів у множині T називатимемо потужністю цієї великої кількості і позначатимемо |T|. Нехай мітка класу набуває наступних значень C1, C2, ..., Ck.

Наше завдання полягатиме в побудові ієрархічної класифікаційної моделі у вигляді дерева з множини прикладів T. Процес побудови дерева відбуватиметься зверху вниз. Спочатку створюється корінь дерева, потім нащадки кореня і т. д.

На першому кроці ми маємо порожнє дерево (є тільки корінь) і початкову множину T (асоційована з коренем). Вимагається розбити початкову множину на підмножини. Це можна зробити, вибравши один з атрибутів в якості перевірки. Тоді в результаті розбиття виходять n (по числу значень атрибуту) підмножин і, відповідно, створюються n нащадків кореня, кожному з яких поставлена у відповідність своя підмножина, отримана при розбитті множини T. Потім ця процедура рекурсивно застосовується до усіх підмножин (нащадків кореня) і т. д.

Навчимо «машину» на даних, які отримані в процесі «ручної» класифікації за допомогою DPI [28]. Для цього в програму обробки потоків підключимо залежності бібліотеки Accord.MachineLearning DecisionTrees і DecisionTrees.Learning. Створимо два масиви double[][] inputs і int[] outputs. Перший міститиме атрибути векторів, а другий його маркер у відповідних індексах. Задамо назви атрибутам за допомогою DecisionVariable[] variables. Створимо об’єкт класу C45Learning за допомогою конструктора і передамо йому дані про атрибути. Далі навчимо за допомогою методу Learn. Збережемо навчений об’єкт за допомогою бінарної серіалізації для подальшого використання.

Отже, ми маємо дерево рішень і хочемо використати його для розпізнавання нового об’єкту. Обхід дерева рішень розпочинається з кореня

дерева. На кожному внутрішньому вузлі перевіряється значення об’єкту Y по атрибуту, який відповідає перевірці в цьому вузлі, і, залежно від отриманої відповіді, знаходиться відповідне розгалуження, і по цій дузі рухаємося до вузла, що знаходить на рівень нижче і т. д. Обхід дерева закінчується, як тільки зустрінеться вузол рішення, який і дає назву класу об’єкту Y.

У рамках вивчення і аналізу мережевого трафіку по NetFlow створена програма, яка дозволяє виводити атрибути по кожному потоку. Додамо в цю програму наш створений серіалізованний файл, підключимо Accord .NET Framework. Використовуючи метод Compute класифікуємо дані, за раніше отриманими даними.

В результаті дослідження вийшов набір програмних продуктів, що дозволяють збирати NetFlow дані, класифікувати їх за допомогою DPI, а в подальшому використати цю інформацію для навчання машини методів C4.5, і в майбутньому використати ML для класифікації трафіку без допомоги DPI. Таким чином розв’язана проблема швидкості роботи DPI на не дорогому устаткуванні. У цьому підході є і мінуси - класифікація трафіку здійснюється пост-факт, а не в реальному часі. Тобто визначивши шкідливу активність або небажане застосування, обмежити його дію не буде можливості. Такий підхід хороший для збору і аналізу статистики поведінки користувачів в мережі, для її подальшого поліпшення. Головна проблема, а саме надання звітів клієнтів у рамках угоди SLA вирішена, і тепер має вагомий аргумент у вигляді повної інформації про мережевий трафік, не лише в цифрах, але і в списку сервісів, що використовуються клієнтами, що наочніше показує якість послуг, що надаються.

3.3 Проведення експериментальних досліджень

У рамках проведеного дослідження були досліджені і розглянуті усі популярні методи кластеризації і класифікації даних. Основна проблема цих методів - це «сирість» даних на початковому етапі, що не дозволяє більшості математичних методів розкритися повною мірою [29].

Щоб розв’язати цю проблему на прикладі обробки даних мережевого трафіку, була зроблена спроба провести попередню обробку даних існуючими не математичними методами, а після цього застосувати математику. Вибір впав на технологію DPI (Deep Packet Inspection), яка дозволяла ідентифікувати приналежність трафіку до того або іншого протоколу. Автори технології заявляють точність визначення 98%, що є дуже добрим результатом в порівнянні з математичними методами.

Але і попередня обробка технологією DPI не панацея, оскільки наприклад трафік YouTube або будь-якого іншого відео сервісу ділиться на два типи: HTTP і Video. А DPI їх не розрізняє, проте властивості, , що описують IP-потік, відмінно дозволяють їх розрізнити, та ще і в автоматичному режимі - методами кластеризації. Був застосований метод K-Means і увесь трафік був розділений на підмножини.

Тільки після усіх цих маніпуляцій був застосований метод класифікації C4.5, який дозволив побудувати таке дерево рішень, яке дало результат до 98% точності визначення трафіку. Швидкість роботи визначення нового набору трафіку займає мілісекунди у відмінності від технології DPI, яка перевіряє усі дані по моделі OSI. Так само використання класифікації даних на етапі пост-факт, а не в реальному часі дозволяє не заважати проходженню IP пакетів через мережеве устаткування.

У рамках даного дослідження, для наочного перегляду результату аналізу було розроблено програмне забезпечення і налагоджено відповідне устаткування для збору і аналізу трафіку.

Налагоджений і введений в експлуатацію колектор NetFlow, що збирає дані про трафік з бордерів в телекомунікаційній компанії (рисунок 3.2).

Дані обробляються і додаються у базу даних MySQL, для спрощеної роботи надалі (рисунок 3.3).

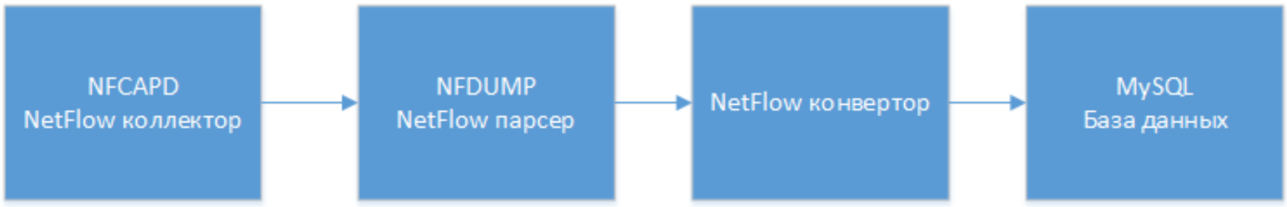

Рисунок 3.2 - Принцип роботи апаратного і програмного комплексу

Результат #1 (6×16 882)

inet_ntoa(Srcaddr)	total	sum(TX_Bytes)	sum(RX_Bytes)	sum(TX_Packets)	sum(RX_Packets)
178.218.89.162	256 154	45 263 795	166 834	532 138	2 168
178.218.93.252	1 395	1 739 419	12 312 756	7 189	10 409
178.218.82.109	1 361	26 330 456	709 160	19 659	11 506
178.218.82.207	1 175	1 147 835	10 101 845	12 941	13 151
172.16.128.128	1 069	484 533	9 721 481	6 220	8 682
178.218.82.61	1 067	750 196	415 151	8 880	14 449
178.218.81.181	1 041	1 398 876	6 342 653	6 555	6 406
178.218.89.176	934	316 471	2 577 168	4 609	4 947
172.16.129.148	898	251 465	223 830	2 291	1 566
178.218.80.121	877	214 919	563 466	3 662	3 790
178.218.94.44	844	539 637	2 190 232	3 223	3 426
172.16.128.210	797	199 209	293 585	2 022	1 368
178.218.81.161	731	3 063 924	1 641 675	4 708	3 506
178.218.94.26	576	116 206	135 436	1 398	936
172.16.128.26	562	402 503	1 781 328	5 595	7 865

Рисунок 3.3 - Приклад даних у БД MySQL

Встановлено програмне забезпечення, що реалізовує технологію DPI-nDPI. Бібліотека з відкритим початковим кодом. Зібрано більше за 10Тб трафіку для його класифікації і проведений повний цикл дослідження із застосуванням кластеризації і навчання методом C4.5 [30].

Інтерфейс розробленої програми містить повний набір перемикачів і налаштувань для отримання звіту з будь-якими вимогами оператора (рисунок 3.4).

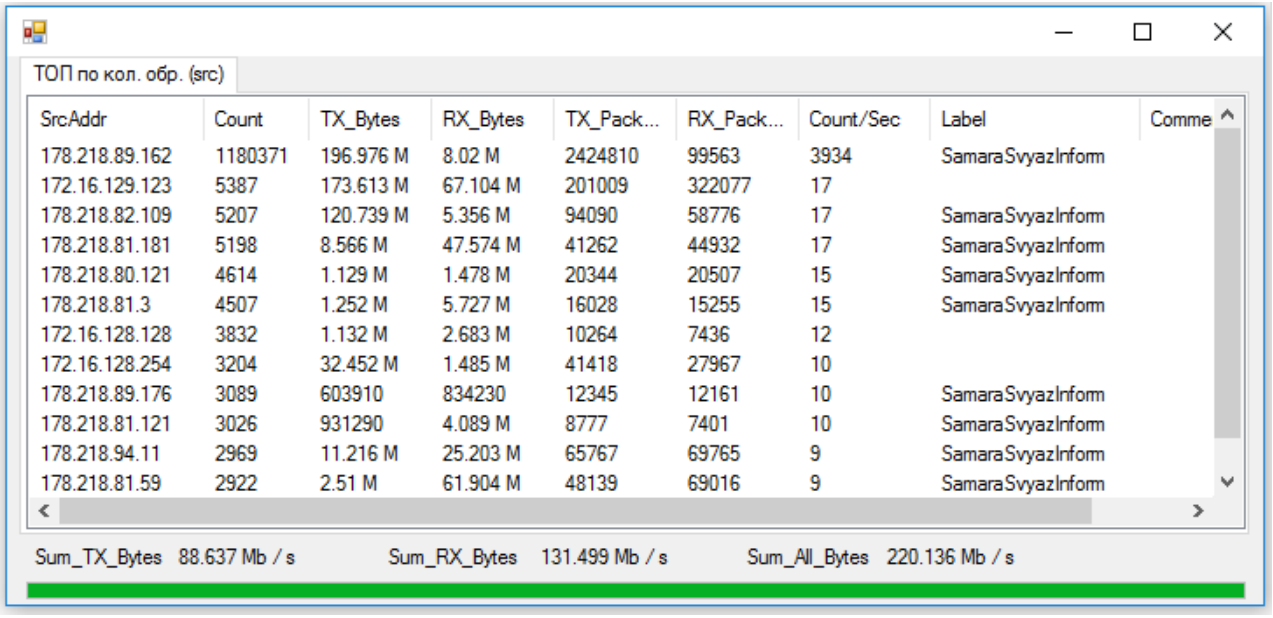

Рисунок 3.4 - Головне вікно програми

При натисненні на запис відкривається вікно з детальною інформацією про те, які звернення виконуються з цього хоста (рисунок 3.5).

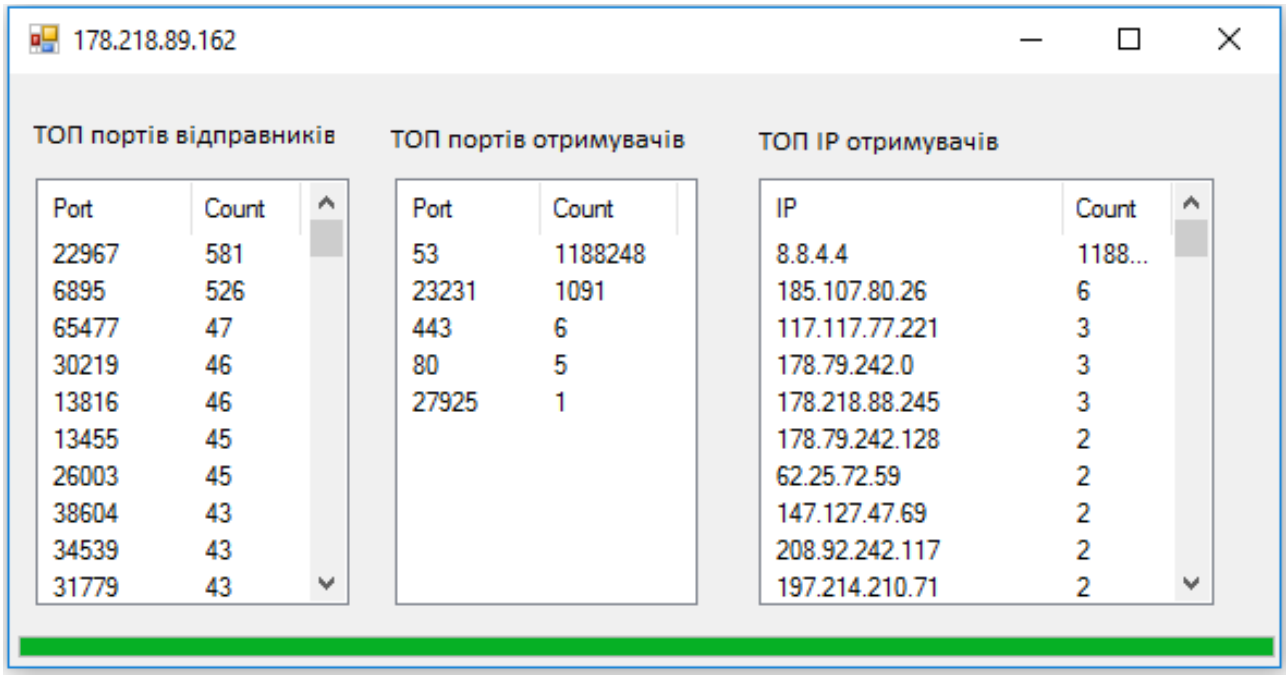

Рисунок 3.5 - Приклад вікна з детальною інформацією

Також можна отримати більш детальну інформацію (рисунок 3.6).

Marker	DateTime	SrcAddr	DstAddr	SrcPort	DstPort	Protocol	TX_Packets	RX_Packets	TX_Bytes
SSL	2023.05.17 05:17:53	178.218.81.4	195.101.32.54	51816	81	TCP	2	2	80
Office365	2023.05.17 05:17:53	178.218.81.4	178.218.131.36	49319	7547	TCP	2	2	80
Skype	2023.05.17 05:17:53	178.218.81.4	178.218.202.237	52070	7547	TCP	2	2	80
Office365	2023.05.17 05:17:53	178.218.81.4	178.218.239.88	40098	23	TCP	2	2	80
Office365	2023.05.17 05:17:53	178.218.81.4	178.218.95.191	26899	7547	TCP	2	2	80
Office365	2023.05.17 05:17:53	178.218.81.4	178.218.236.95	15056	23	TCP	2	2	80
Skype	2023.05.17 05:17:53	178.218.81.4	1.34.53.50	30920	13841	UDP	5	6	388
SSL	2023.05.17 05:17:53	178.218.81.4	174.103.154.174	43776	7547	TCP	7	7	575
SSL	2023.05.17 05:17:53	178.218.81.4	178.218.83.22	46777	23	TCP	2	0	80
Office365	2023.05.17 05:17:53	178.218.81.4	178.218.235.1	41657	23	TCP	2	2	80

Рисунок 3.6 - Детальна інформація

У запропонованому варіанті класифікації є можливості розвиватися далі. Тисячі математиків і програмістів у світі зайняті проблемою нейронних мереж, які є просунутими варіантами методів класифікації [31].

Штучна нейронна мережа (ШНМ) є математичною моделлю, яка побудована за принципом функціонування біологічних нейронних мереж, які складаються з нервових клітин. Ці моделі були розроблені для того, щоб змоделювати процеси, що відбуваються в мозку, і в подальшому були застосовані в різних практичних задачах, таких як прогнозування, розпізнавання образів та управління.

ШНМ складається з взаємодіючих між собою простих процесорів, які називають штучними нейронами. Кожен штучний нейрон працює тільки зі сигналами, які отримує і надсилає іншим нейронам. Незважаючи на те, що окремі нейрони досить прості, вони здатні виконувати складні задачі завдяки взаємодії великої кількості нейронів у мережі [32-33].

З точки зору машинного навчання, нейронна мережа є окремим випадком методів розпізнавання образів, аналізу дискримінанта, методів кластеризації і т. п.

З математичної точки зору, навчання нейронних мереж – це багатопараметрична задача нелінійної оптимізації [34].

З точки зору кібернетики, нейронна мережа використовується в задачах

адаптивного управління і як алгоритми для робототехніки [35].

З точки зору розвитку обчислювальної техніки і програмування, нейронна мережа – спосіб вирішення проблеми ефективного паралелізму.

А з точки зору штучного інтелекту, ШНМ є основою філософської течії коннективізму і основним напрямом в структурному підході по вивченню можливості побудови (моделювання) природного інтелекту за допомогою комп'ютерних алгоритмів.

Нейронні мережі не програмуються в звичному значенні цього слова, вони навчаються [36-41]. Можливість навчання – одна з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно навчання полягає в знаходженні коефіцієнтів зв'язків між нейронами. В процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними даними і вихідними, а також виконувати узагальнення. Це означає, що у разі успішного навчання мережа зможе повернути вірний результат на підставі даних, які були відсутні в навчальній вибірці, а також неповних і/або «зашумленних», частково спотворених даних [36-41].

Самонавчання запропонованого методу класифікації на прикладі нейронних мереж дозволить йому підвищити свій рівень в десятки разів.

В даному розділі отримано наступні висновки:

1. Захоплення мережевого трафіку проводиться утилітою tshark в pcap-файли на робочому мережевому інтерфейсі. Збирається трафік з одного з мережевих вузлів телекомунікаційної компанії. Далі дані обробляються утилітами DPI, для ручної класифікації трафіку. За допомогою написаної програми отримані маркери співвідносяться з даними трафіку і зберігаються у базу даних, заздалегідь об'єднуючись в двонаправлені потоки. Це основа навчальної вибірки, за допомогою якої в майбутньому здійснюватиметься класифікація трафіку.

2. Аналіз підготовлених даних проводився в декількох системах, щоб понизити вплив конкретної реалізації алгоритму: python, sklearn. Паралельно використовувалася Data Mining утиліта Weka. Послідовно застосовувалися

методи кластерного аналізу, нейронні мережі, дерева рішень, реалізовані вказаними програмними засобами.

3. Методи оцінювалися не лише за якістю отримуваного результату, але і за часом виконання. Скорочення часу на навчання дозволяє досягти аналізу мережевого трафіку в режимі, близькому до реального часу.

ВИСНОВКИ

1. У даній роботі розглядається проблема аналізу IP-трафіку з використанням методів Data Mining, до числа яких відносяться класифікація – розділення об’єктів на задані групи (класи) згідно з характеристиками об’єктів; регресія – пошук функції, що моделює множину об’єктів, що вивчаються, з найменшою помилкою; кластеризація – пошук незалежних груп об’єктів, кластерів і їх характеристик (групи заздалегідь не обумовлені); пошук асоціативних правил – характерних залежностей між об’єктами або подіями.

2. В цілому, незважаючи на наявність різних обмежень і припущень (про незалежність атрибутів, про нормальність розподілу числових атрибутів всередині класу, про правильність порядку прибуття пакетів, про мінімальний вплив мережевого оточення на затримки між пакетами), можна зробити висновок, що дані методи є досить ефективними.

3. Проведено повний спектр аналізу даних телекомунікаційної компанії. Спочатку компанією були надані сирі дані за лютий місяць, що містять службову інформацію про IP-потоках. Для проведення повного аналізу написана програма на мові C#, що агрегує дані по днях, таким чином були виділені ознаки, по яких надалі здійснюватиметься класифікація за відомими методами.

4. Проведено кластерний аналіз за допомогою програми на мові Python, виявлені два кластери: середньостатистичні клієнти і «особливі», які представляють загрозу телекомунікаційної компанії. Інформація відображена на графіках. За цими даними було проведено навчання вибірки великого об’єму (60427 векторів) з введенням поля «target».

5. Застосовано метод опорних векторів для класифікації нових даних, що поступають в реальному часі. Навчений класифікатор може бути далі застосовуватися для аналізу нових даних телекомунікаційної компанії.

6. Наукова новизна полягає у використанні декількох математичних методів в певній послідовності з використанням попередньої обробки даних, що дозволила збільшити якість роботи класифікатора до 98%. Використання

технології DPI при попередній обробці даних дозволяє отримати точність визначення трафіку до 98%, і ця технологія використовується один раз, а не постійно, що добре позначається на швидкості каналу мережевого устаткування.

7. Використання методу кластеризації K-means на етапі попередньої обробки даних дозволило відрізнити трафік одного протоколу на декілька підвидів, наприклад, на HTTP, Video і SSL, що так само позитивно позначилося на якості роботи класифікатора. Нарешті, використання сучасного і популярного класифікатора C4.5 дозволило консолідувати усі зусилля і отримати максимально ефективну «навчену машину».

8. Захоплення мережевого трафіку проводиться утилітою tshark в pcap-файли на робочому мережевому інтерфейсі. Збирається трафік з одного з мережевих вузлів телекомунікаційної компанії. Далі дані обробляються утилітами DPI, для ручної класифікації трафіку. За допомогою написаної програми отримані маркери співвідносяться з даними трафіку і зберігаються у базу даних, заздалегідь об’єднуючись в двонаправлені потоки. Це основа навчальної вибірки, за допомогою якої в майбутньому здійснюватиметься класифікація трафіку.

9. Аналіз підготовлених даних проводився в декількох системах, щоб понизити вплив конкретної реалізації алгоритму: python, sklearn. Паралельно використовувалася Data Mining утиліта Weka. Послідовно застосовувалися методи кластерного аналізу, нейронні мережі, дерева рішень, реалізовані вказаними програмними засобами.

10. Методи оцінювалися не лише за якістю отримуваного результату, але і за часом виконання. Скорочення часу на навчання дозволяє досягти аналізу мережевого трафіку в режимі, близькому до реального часу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Data Mining: Practical Machine Learning Tools and Techniques by Ian H. Witten, Eibe Frank, and Mark A. Hall.

2. DeGroot, M. H., & Schervish, M. J. (2011). Probability and Statistics. Pearson.

3. Crotti, M. Application protocol fingerprinting for traffic classification [Текст] / Crotti, M., Dusi M., Este A – 346 с.

4. Haddad, R. A. A class of fast Gaussian binomial filters for speech and image processing [Текст] / Haddad R. A., Akansu A. N. - IEEE Trans. Acoust., Speech Signal Proc. 1991. V. 39. P. 723–727.

5. Dedinski, I. Cross-layer peer-to-peer traffic identification and optimization based on active networking [Текст] / Dedinski I., De Meer H., Han L. - Proc. of the 7th Intern. workshop on active networks (IWAN 2005), Sophia-Antipolis (France). Nov. 21–23, 2005. Berlin; Heidelberg: Springer-Verlag, 2009. P. 13–27.

6. Chui, Ch. K. An introduction to the wavelets [Текст] / Chui, Ch. K. - N. Y.: Acad. Press, 1992.

7. Moore, A. Internet traffic classification using Bayesian analysis techniques [Текст] / Moore A. W., Zuev D. - ACM SIGMETRICS 2005, Banff, Alberta (Canada), June 2005. N. Y.: ACM, 2005. P. 50–60.

8. Yu, L. Feature selection for high-dimensional data: A fast correlation-based filter solution [Текст] / Yu L., Liu H. - Proc. of the 20th Intern. conf. on machine learning (ICML 2003), Washington, 2003. Palo Alto: AAAI Press, 2003. P. 856–863.

9. Duffield, N. G. Entropy of ATM traffic streams [Текст] / Duffield N. G., Lewis J. T., O'Connell N. - IEEE J. Select. Areas Commun. 1995. V. 13, iss. 6. P. 981–990.

10. Moore, A. Discriminators for use in flow-based classification [Текст] / Moore A. W., Zuev D. - Tech. report / Intel Res. Cambridge, 2005.

11. Wand, M. P. Kernel smoothing [Текст] / M. P. Wand, M. C. Jones. L.: Chapman and Hall/CRC, 1995.

12. John, G., Estimating continuous distributions in Bayesian classifiers [Текст] / John G., Langley P. - UAI'95: Proc. of the 11th conf. on uncertainty in artificial intelligence, Quebec (Canada), 1995. San Francisco: Morgan Kaufmann, 1995. P. 338–345.

13. Guyon, I. An introduction to variable and feature selection [Текст]/ Guyon I., Elisseeff A - J. Machine Learn. Res. 2003. V. 3. P. 1157–1182.

14. Moore, A. W. Discrete content-based classification — a data set [Текст]/ Moore A. W. - Tech. report / Intel Res. Cambridge, 2005.

15. Lin, Y-D. Application classification using packet size distribution and port association [Текст] / Lin Y-D., Lu Ch-N., Lai Y-Ch. - J. Network Computer Appl. 2009. V. 32. P. 1023–1030.

16. Hu, Y. Profiling and identification of P2P traffic [Текст] / Hu Y., Chiu D-M., Lui J. C. S. - Comput. Networks. 2009. V. 53. P. 849–863.

17. Agrawal, R. Fast algorithms for mining association rules [Текст] / Agrawal R. - Proc. of the 20th VLDB conf., Santiago de Chile (Chile), Sept. 12–15, 1994. San Francisco: Morgan Kaufmann, 1994. P. 487–499.

18. Paxson, V. Bro A system for detecting network intruders in real-time [Текст] / Paxson V. - Comput. Networks. 1999. V. 31, N 23/24. P. 2435–2463.

19. Li, W. Efficient application identification and the temporal and spatial stability of classification schema [Текст] / Li W., Canini M., Moore A. W., Bolla R. - Comput. Networks. 2009. V. 53, N 6. P. 790–809.

20. Williams, N. A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification [Текст] / Williams N., Zander S., Armitage G. - SIGCOMM Comput. Commun. Rev. 2006. V. 36, iss. 5. P. 5–16.

21. Lim, T.-S. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms [Текст]/ Lim T.-S., Loh W.-Y., Shih Y.-S. - Machine Learn. 2000. V. 40, iss. 3. P. 203–229.

22. Canini, M. GTVS: Boosting the collection of application traffic ground truth [Текст] / Canini M., Li W., Moore A. W., Bolla R. - Lecture Notes Comput. Sci. 2009.

V. 5537. P. 54–63.

23. Moore, A. W. Toward the accurate identification of network applications [Текст] / Moore A. W., Papagiannaki K. - Lecture Notes Comput. Sci. 2005. V. 3431. P. 41–54.

24. Щербакова, Н. Г. Аналіз IP-трафіка методами Data Mining [Текст] // Н. Г. Щербакова - Пробл. інформатики. 2012. № 4. С. 30–46.

25. Durrett, R. (2010). Probability: Theory and Examples. Cambridge University Press.

26. McGregor, A. Flow clustering using machine learning techniques [Текст] / McGregor A., Hall M., Lorier P., Brunskill J. - Proc. Lecture Notes in Computer Science. V. 3015, Antibes Juan-les-Pins (France), Apr. 2004. N. Y.: Springer, 2004. P. 205–214.

27. Zander, S. Self-learning IP traffic classification based on statistical flow characteristics [Текст] / Zander S., Nguyen T., Armitage G. - Passive and active measurement workshop (PAM 2005): Proc. Lecture Notes in Computer Science. V. 3431, Boston, USA, March — Apr. 2005. N. Y.: Springer, 2005. P. 325–328.

28. Zander, S. Automated traffic classification and application identification using machine learning [Текст] / Zander S., Nguyen T., Armitage G. - 30th Annual IEEE conf. on local computer networks (LCN 2005): Proc.

29. Cheeseman, P. Bayesian classification (AutoClass): theory and results [Текст] / Cheeseman P., Stutz J. - Advances in knowledge discovery and data mining. Palo Alto (CA, USA): AAAI/MIT Press, 1996. P. 61–68.

30. Erman, J. Traffic classification using clustering algorithms [Текст] / Erman J., Arlitt M., Mahanti A. - Special interest group on data communication (SIGCOMM) 2006 workshops: Proc. of the 2006 SIGCOMM workshop on Mining network data, Pisa (Italy), Sept. 11–15, 2006. N. Y.: ACM, 2006. P. 281–286.

31. Ester, M. A density-based algorithm for discovering clusters in large spatial databases with noise [Текст] / Ester M., Kriegel H., Sander J., Xu X. - Proc. of

the 2nd Intern. conf. on knowledge discovery and data mining (KDD-96), Portland (USA), 1996. Palo Alto: AAAI/MIT Press, 1996. P. 226–231.

32. Witten, I. H. Data mining: practical machine learning tools and techniques [Текст] / I. H.Witten, E. Frank. San Francisco: Morgan Kaufmann, 2005. P. 560.

33. Paxson, V. Empirically-derived analytic models of wide-area TCP connections [Текст] / V. Paxson - IEEE/ACM Trans. Network. 1994. V. 2, N 4. P. 316–336.

34. Erman, J. Identifying and discriminating between web and peer-to-peer traffic in the network core [Текст] / Erman J., Mahanti A., Arlitt M., Williamson C. - Proc. of the 16th Intern. conf. on world wide web (WWW) 2007, Banff (Alberta, Canada), May 8–12, 2007. N. Y.: ACM, 2007. P. 883–892.

35. Sen, S. Accurate, scalable in-network identification of P2P traffic using application signatures [Текст] / Sen S., Spatscheck O., Wang D. - Proc. of the 13th Intern. world wide web conf., New York (USA), May 2004. N. Y.: ACM, 2004. P. 512–521.

36. Berry, M. J. A., & Linoff, G. S. (2004). Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management. John Wiley & Sons.

37. Haukin S. Neural networks: a comprehensive foundation / S. Haukin. – Prentice Hall. – 1999. – 842 p.

38. Golovko V. A. Learning Technique for Deep Belief Neural Networks / V. Golovko, A. Kroshchanka, U. Rubanau, S. Jankowski // in book Neural Networks and Artificial Intelligence. – Springer, 2014. – Vol. 440. Communication in Computer and Information Science. – P. 136-146.

39. Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural Computation, 18(7), 1527-1554.

40. Backpropagation applied to handwritten zip code recognition / Y. LeCun [and oth.] // Neural computation. – 1989. – 1(4). – P. 541–551.

41. Golovko, V. A simple shallow convolutional neural network for accurate handwritten digits classification / V. Golovko, E. Mikhno, A. Brich // Proceedings of the 13–th on Pattern recognition and Information Processing (PRIP'2016, 3–5 October 2016). Minsk : BSU, 2016. – P. 209 – 212.

42. Комар М.П., Саченко А.О., Васильків

Н.М., Гладій Г.М., Коваль В.С. Методичні рекомендації до виконання кваліфікаційної роботи з освітньопрофесійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. - Тернопіль: ЗУНУ, 2021. - 56 с.