

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

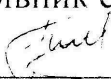
ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: «Детектор кібератак на основі дерев рішень з
реалізацією на Python»

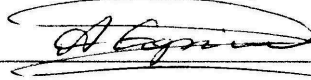
Виконав студент 4-го курсу, групи КБ-2
спеціальності 125 «Кібербезпека»


П.К. Погребніченко

Керівник старший викладач кафедри КІ та КБ


О.А. Покотило

Рецензент завідувач кафедри КІ та КБ, к.т.н., доцент


А.А. Єфіменко

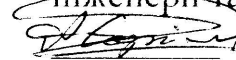
Житомир – 2022

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій Єфіменко

1 березня 2022 р.

ЗАВДАННЯ

на випускн кваліфікаційну роботу

Студент: Погребніченко Павло Костянтинович

Тема роботи: Детектор кібератак на основі дерев рішень з реалізацією на Python

Затверджена Наказом університету від «1» березня 2022 р. № 128/с

Термін здачі студентом закінченої роботи 17 червня 2022р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

Об'єктом дослідження є процес побудови моделі детектору атак, та аналіз її можливостей з використанням Python та KDD Dataset. Предметом дослідження є технологія рішення задач класифікації для детектування атак на основі набору даних з використанням Python та KDD Dataset;

Обладнання та програмне забезпечення:

1. Мова програмування Python;
2. Додаткові бібліотеки машинного навчання;
3. KDD Dataset;
4. Середовище розробки PyCharm.

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Пулеко І.В.	23.03.2022	23.03.2022
2	Пулеко І.В.	23.03.2022	23.03.2022
3	Пулеко І.В.	23.03.2022	23.03.2022

Студент  П.К. Погребніченко

Керівник  О.А. Покотило

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Видача завдання	23.03.2022	виконано
2	Розробка плану роботи	26.03.2022	виконано
3	Розробка розгорнутого плану	29.03.2022	виконано
4	Подача чернетки керівнику для перевірки	30.03.2022	виконано
5	Доопрацювання роботи та перевірка керівником	04.04.2022	виконано
6	Затвердження графічних матеріалів у керівника	07.04.2022	виконано
7	Проведення попереднього захисту	10.06.2022	виконано
8	Отримання відгуку керівника	17.06.2022	виконано
9	Допуск завідувача кафедри	21.06.2022	виконано
10	Подання роботи на рецензування	21.06.2022	виконано

Студент  П.К. Погребніченко

Керівник  О.А. Покотило

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра представляє з себе проект програмного забезпечення детектору атак на основі дерев рішень з реалізацією на Python.

Мета і завдання роботи. Метою роботи є розробка програмного забезпечення детектора атак на основі дерев рішень з використанням мови програмування Python та бази даних KDD Dataset.

Досягнення поставленої мети передбачає розв'язання наступних завдань:

- Проведення аналізу існуючих технологій машинного навчання у сфері безпеки;
- Аналізу типових складових конвеєру машинного навчання;
- Аналізу алгоритму дерева рішень та його модифікацій;
- Аналізу вхідних даних та їх попередньої обробки;
- Розробки програмного забезпечення детектору атак на основі алгоритму дерева прийняття рішень;
- Перевірки та тестування працездатності розробленого програмного продукту, оцінки якості моделі;

Об'єктом дослідження є процес побудови моделі детектору атак, та аналіз її можливостей з використанням Python та KDD Dataset.

Предметом дослідження є технологія рішення задач класифікації для детектування атак на основі набору даних з використанням Python та KDD Dataset.

Пояснювальна записка до випускної кваліфікаційної роботи містить в собі 71 сторінку, 1 таблицю та 32 рисунки. Список використаних джерел містить 29 найменувань і займає 3 сторінки.

КЛЮЧОВІ СЛОВА: штучний інтелект, машинне навчання, Python, KDD Dataset, дерево прийняття рішень, детектор атак, прогнозування.

					КБ.КР.Б – 125 – 22 – ПЗ				
Змн.	Арж.	№ докум.	Підпис	Дата	Детектор кібератак на основі дерев рішень з реалізацією на Python Пояснювальна записка	Літ.	Арк.	Аркушів	
Розроб.	Покребіченко П.К.			20.06					
Керівник	Покотило О.А.			20.06			4	75	
Реценз	Сфіменко А.А.			20.06					
Н. контр.									
Зав. каф.	Сфіменко А.А.			20.06		Житомирська політехніка. група КБ-2			

ABSTRACT

The bachelor's qualification work is an application project of an attack detector based on decision trees with implementation in Python.

The aim and objectives of the work. The aim of the work is to develop an attack detector software based on decision tree algorithm using the Python programming language and the KDD Dataset.

Achieving this goal involves solving the following tasks:

- Carrying out the analysis of existing technologies of machine learning in the field of safety;
- Analysis of typical components of the machine learning pipeline;
- Analysis of the decision tree algorithm and its modifications;
- Analysis of input data and their pre-processing;
- Development of attack detector software based on the decision tree algorithm;
- Checks and tests of serviceability of the developed software product, quality assessment of the model;

The object of research is the process of building an attack detector model, and analyzing its capabilities using Python and KDD Dataset.

The subject of the research is the technology of solving classification problems for detecting attacks based on a data set using Python and KDD Dataset.

The explanatory note to the qualification work contains 71 page, 1 table and 32 figures. The list of used sources contains 29 names and occupies 3 pages.

KEYWORDS: artificial intelligence, machine learning, Python, KDD Dataset, decision tree, attack detector, prediction.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ЗМІСТ

РЕФЕРАТ	4
ABSTRACT	5
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	7
ВСТУП.....	8
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ.....	9
1.1 Обґрунтування актуальності теми роботи	9
1.2 Огляд існуючих методів виявлення атак, що використовують алгоритми машинного навчання	12
1.3 Аналіз складових типового конвеєру машинного навчання.....	22
1.4 Аналіз алгоритму дерева рішень та його модифікацій.....	24
1.5 Постановка завдання на розробку.....	31
РОЗДІЛ 2 РОЗРОБКА ДЕТЕКТОРА АТАК НА ОСНОВІ АЛГОРИТМІВ ДЕРЕВ РІШЕНЬ З РЕАЛІЗАЦІЄЮ НА PYTHON	33
2.1 Обґрунтування вибору мови програмування.....	33
2.2 Життєвий цикл проекту машинного навчання	40
2.3 Аналіз вхідних даних та їх попередня обробка	43
2.4 Розробка програмного забезпечення детектора атак на основі алгоритмів дерев рішень	51
РОЗДІЛ 3 ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ ТА ЕФЕКТИВНОСТІ ДЕТЕКТОРА АТАК НА ОСНОВІ АЛГОРИТМІВ ДЕРЕВ РІШЕНЬ	58
3.1 Показники якості класифікації при виявленні атак	58
3.2 Оцінка якості виявлення атак розробленим програмний забезпеченням.....	61
3.3 Рекомендації по застосуванню та вдосконаленню розробленого детектора атак на основі алгоритмів дерев рішень	64
ВИСНОВКИ.....	67
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	69
ДОДАТКИ.....	72

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ASM – Міра обрання атрибутів (анг. Attribute Seeking Measure);

AUC – Площа під кривою (анг. Area under the Curve);

CART – Алгоритм класифікаційного та регресивного дерева (анг. Classification and Regression Trees);

DL – Глибоке Навчання (анг. Deep Learning);

HMM – Приховані моделі Маркова (анг. Hidden Markov Model);

IGR – Індекс Набору інформації (анг. Information Gain Ratio);

KNN – Метод найближчих сусідів (анг. K Nearest Neighbors);

LR – Метод логістичної регресії (анг. Logistic Regression Model);

NB – Метод наївних Байєса (анг. Naïve Bayes Method);

RF – Метод випадкового лісу (анг. Random Forest Method);

ROC – Крива операційної характеристики відповідей (анг. Receiver Operating Characteristics);

SL – Неглибоке Навчання (анг. Shallow Learning);

SVM – Метод опорних векторів (анг. Support Vectors Method);

MH – Машинне навчання;

ІІІ – Штучний інтелект.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Останні роки тенденція комфорту та безпеки змінюється, стає більш автоматизованою за допомогою інтернету речей (IoT). Концепції розумних будинків та мереж зв'язку між пристроями вже не є чимось незвичним, зараз це повсякденне життя для окремих людей, проте дана галузь забезпечує не лише комфорт у вашій квартирі, але й безпеку, аналіз медичних даних, таргетовану рекламу у соціальних мережах або пропонованих друзів. Для роботи усіх цих автоматизованих алгоритмів використовуються методи штучного інтелекту, та, зокрема, машинного навчання. Машинне навчання являє собою метод тренування машини задля її автоматизованої роботи у майбутньому без потреби людини. Така машина може сама приймати рішення, аналізувати дані, виявляти закономірності та робити усе їй поставлене у навчанні та після.

Актуальність теми машинного навчання у безпеці є чималою, адже велика кількість компаній та фахівців задіюють дану технологію у своїх безпекових системах. Цей підхід зменшує навантаження на фахівців, зменшує кількість помилок людського фактору та неуважності та надає людям змогу працювати з більш пріоритетними загрозами, що, у свою чергу, покращить рівень безпеки мережі чи організації та зменшить кількість вдалих атак. За вищевказаним можна сказати, що **тема роботи** направлена на розробку програмного засобу детектору атак на базі Python є **актуальною**.

Методи дослідження. В кваліфікаційній роботі було використано такі методи: прикладного програмування, двокласової класифікації, математичного моделювання та алгоритмізації коду.

Практичне значення роботи полягає у наступному:

- розроблений програмний продукт дозволяє якісно оцінювати пакети даних, аналізувати їх та прогнозувати шкідливість за допомогою мови програмування Python та KDD Dataset, що допомагає в роботі фахівцям з безпеки та системним адміністраторам;
- розроблені пропозиції використання програмного продукту у сфері кібербезпеки, мережевого та системного адміністрування.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ

1.1 Обґрунтування актуальності теми роботи

Нинішня ситуація з питань безпеки у світовій мережі та на локальних рівнях є неоднозначною. Водночас, коли одні компанії-гіганти мають найпотужніші безпекові заклади світу, інші, локальні, не мають базових правил та понять, що наражає на великі втрати та витоки інформації.

Саме у таких випадках основним питанням є системи забезпечення безпеки, та на авангарді системи виявлення та розпізнавання загроз. У попередні часи такі програми мали чіткий алгоритм роботи та набір правил, та були керовані людиною. Людська неуважність грала велику роль та мала чималий ризик, оскільки порушення цілісності безпеки може призвести до найфатальніших наслідків.

Також значною проблемою була так звана соціальна інженерія (англ. Social engineering) – навмисне введення людини в оману задля отримання потрібної зловмиснику інформації чи конфіденційних матеріалів (паролів, файлів, реквізитів, тощо).

Нині ж, окрім людського фактору, зловмисники навчилися користуватись й технологічним: підроблювати та маскувати свої атаки чи шкідливі програми під штатські процеси та офісне програмне забезпечення.

Разом з цим питанням виникла потреба мати гнучкі системи захисту, які можуть самі вивчати та аналізувати процеси, і, базуючись на дослідженнях, приймати рішення. Подібні системи є системами на базі машинного навчання.

Машинне навчання (англ. Machine Learning) – це область штучного інтелекту, яка за допомогою статистики надає можливість комп'ютерам «навчатись» з отриманих ним даних, аналізувати та робити висновки, тим самим покращуючи результат з кожним новим надходженням даних [1]. Слід зауважити, що увесь цей процес не керується явно людиною, та не потребує залучення нікого, окрім машини [3].

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

Розглянемо на прикладі ефективності роботи та часу на виконання. Оскільки люди мають нормований графік та не можуть працювати без перепочинку, процес сповільнюється та зобумовлюється багатьма факторами, від перерви на каву до поганого самопочуття фахівця. Цей фактор може призвести до враження атаківаних цілей та блокування системи, викрадення даних чи втрати контролю. У цей час машина, яка не має ніяких перешкод, одразу зреагує на загрозу, потенційну чи повноцінну, та почне приймати кроки до знешкодження з мінімальною затримкою.

З попередніх слів витікає ще проблема – швидкість реагування. На прикладі множинних атак, коли надходить велика кількість одночасних запитів з різних адрес, команда фахівців може не здужати з об'ємом оброблюваного потоку. Також у сучасному технологічному світі шкідливі програмні забезпечення мають чималу швидкість розповсюдження по системі, та можуть бути швидшими за людські дії, в той час коли машина може водночас аналізувати атаку та надсилати фахівцю звітність задля подальших дій.

Можливість адаптуватися та вивчати нові паттерни – ключовий фактор та перевага машинного навчання. Оскільки усі атаки базуються на шаблонах вразливих систем чи програм, системи безпеки можуть запам'ятовувати такі шаблони, реагуючи значно швидше у наступні рази. Таким же чином детектори мають змогу перебудовувати свою поведінку під час атаки, якщо щось йде не за планом, чи атака має у собі приховані процеси чи файли.

Разом з цим штучний інтелект значно стійкіший до монотонної праці, що повторюється на регулярній основі, наприклад складання шаблонів чи перевірка та ідентифікація даних. Машина може значно швидше провести усю монотонну працю та передати людині готовий результат, що суттєво підвищить якість роботи та знизить ризики.

Імплементація машинного навчання вже широко використовується на сучасному ринку безпеки. Багато компаній, особливо спеціалізованих на безпеці, використовують штучний інтелект як основу своєї діяльності, тож розглянемо декілька таких компаній [4].

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

1. Darktrace. Маючи понад 30 офісів по всьому світу, Darktrace допомогла тисячам компаній у різних галузях виявляти кіберзагрози та боротися з ними в режимі реального часу. Платформа штучного інтелекту Darktrace аналізує мережеві дані, щоб робити розрахунки та виявляти закономірності. Технологія машинного навчання використовує дані, щоб допомогти організаціям виявити відхилення від типової поведінки та визначити загрози.

2. Cylance. Cylance — це платформа зі штучним інтелектом, яка допомагає запобігти загрозам, перш ніж вони можуть завдати шкоди, прогножуючи та захищаючи від атак без файлів, шкідливого програмного забезпечення та виконання корисного навантаження нульового дня. Технологія Cylance працює шляхом профілювання мільярдів зразків файлів, оцінки файлів на наявність загроз, визначення наявності загрози та поміщення заражених файлів до карантину. Клієнти варіюються від малого бізнесу до підприємств і сервісних компаній у різних галузях, таких як енергетика, фінанси, уряд, охорона здоров'я та роздрібна торгівля [5].

3. Fortinet. Fortinet надає рішення безпеки для кожної частини ІТ-інфраструктури. Більшість компаній із списку Fortune 500 користуються продуктами кібербезпеки Fortinet від безпеки мереж і веб-додатків до захисту від загроз і безпечного уніфікованого доступу. Продукт компанії FortiWeb на основі штучного інтелекту — це брандмауер веб-додатків, який використовує машинне навчання та два рівні статистичних ймовірностей для точного виявлення загроз.

4. Securonix. Securonix надає різноманітні рішення щодо безпеки, від хмарних і кіберзагроз до запобігання шахрайству та вилучення даних. Використовуючи великі дані та машинне навчання, технологія компанії відстежує поведінку користувачів і облікового запису, щоб зрозуміти, що є «нормальним». Рішення Securonix потім виявляють загрози та шахрайство, відфільтровуючи не загрозливі зміни та поведінку користувачів.

5. FireEye. FireEye надає підприємствам і організаціям комплексні рішення з кібербезпеки на уніфікованій платформі, яка включає запобігання, виявлення

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

та реагування. Серед його рішень — можливості розвідки загроз, які дають командам безпеки більшу видимість атак. Технологія розвідки загроз FireEye забезпечує більший контекст і пріоритет для атак і допомагає проактивно захищатися від майбутніх загроз. Технологія компанії використовується в багатьох галузях, включаючи хмарні провайдери, уряд, фінанси, охорону здоров'я та системи промислового контролю [6].

6. SparkCognition. SparkCognition надає рішення для операцій, безпеки та автоматизації на основі ШІ для цілого ряду галузей, від кібербезпеки та авіації до фінансів та виробництва. Для організацій та підприємств, які потребують рішень для кіберзахисту, SparkCognition пропонує продукти на основі машинного навчання, які виявляють та захищають від шкідливих програм, програм-вимагачів, троянів та інших загроз.

7. Tessian. Платформа безпеки ШІ Tessian запобігає зламам, цільовому фішингу та втраті даних із шкідливих електронних листів. Компанія створює налаштовані фільтри електронної пошти, які відсікають шкідливі та підозрілі дії як у вхідних, так і у вихідних повідомленнях електронної пошти. Платформа також постачається з панеллю керування в режимі реального часу, тому служби безпеки можуть миттєво стежити за станом своєї інфраструктури.

Таким чином можна побачити широкий спектр використання машинного навчання у багатьох компаніях світу, що свідчить про неабияку актуальність даної технології.

1.2 Огляд існуючих методів виявлення атак, що використовують алгоритми машинного навчання

Методи виявлення кібератак діляться на дві категорії: на основі сигнатур і на основі невідповідності. В обох випадках використовуються методи машинного навчання.

Автоматизовані алгоритми машинного навчання використовують задля створення маркувань, що можуть позначати та вирізняти шкідливий код та його поведінку за останні роки.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

Почнемо з методу на основі сигнатур, або ж підписів. Такі обчислення як Генерація підпису на мережевій основі (NSG), Генерація підпису на основі довжини (LSEG), та F-Sign використовуються задля швидкого та ефектного видалення поліморфних вірусів-хробаків [7].

Метод обчислення F-Sign концентрує увагу на шкідливому коді, а саме на паразитах, які використовують «cradle flood assault» (за цими мітками можна швидко впізнати та ізолювати вірус від поширення).

SA або ж Semantic Aware – рукописне обчислення, що було розроблене для виявлення шкідливого програмного забезпечення, оснований на обсязі трафіку, що він генерує. Навіть коли трафік зашумлено, така технологія може досить чітко визначити джерело шкідливого сигналу.

Методики феномену, які використовують для детекції кібератак, часто базуються на моделі, що зображує заплановану, звичайну поведінку організаційних комунікацій, та неочікувану [2].

Існує три типи обчислень, що використовують в таких методиках: некеровані, напівнаправлені та цілком керовані. Якщо ж ви проводите поодинокі навчання, найкраще підійдуть методи групування, котрі модифікують вищезазначені обчислення. Такими методами є: k-implies, fuzzy c-means, QT та SVM [8].

Узагальнюючи, згрупований цими методами організаційний трафік позначає кожен групу даних як шкідливу або ж безпечну. У поодиноких чистих обчисленнях найважливішу групу прийнято вважати за середнє. В результаті цього правила, мережеві події, що виконуються на постійній основі, не помічаються.

Слід також зазначити методи, що використовуються в кібербезпеці, котрі входять до алгоритмів Неглибокого навчання (SL) та Глибокого навчання (DL) відповідно. Кожний з цих методів можна розділити на керовані та некеровані (рис. 1.1) [9].

Неглибоке навчання потребує так званого «експерту домену», інженера, який зможе обробити критичні задачі ідентифікації характеристик даних перед початком алгоритму навчання.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

Глибоке навчання базується на багатошаровій репрезентації введених даних, та може самостійно оброблювати усю інформацію за допомогою методу визначених представлень [10].

Оскільки поставлена задача до кваліфікаційної роботи має під собою методи алгоритму Неглибокого навчання, детальний огляд Глибокого навчання не є необхідним та не буде проводитись.

FIGURE 1. CLASSIFICATION OF ML ALGORITHMS FOR CYBER SECURITY APPLICATIONS.

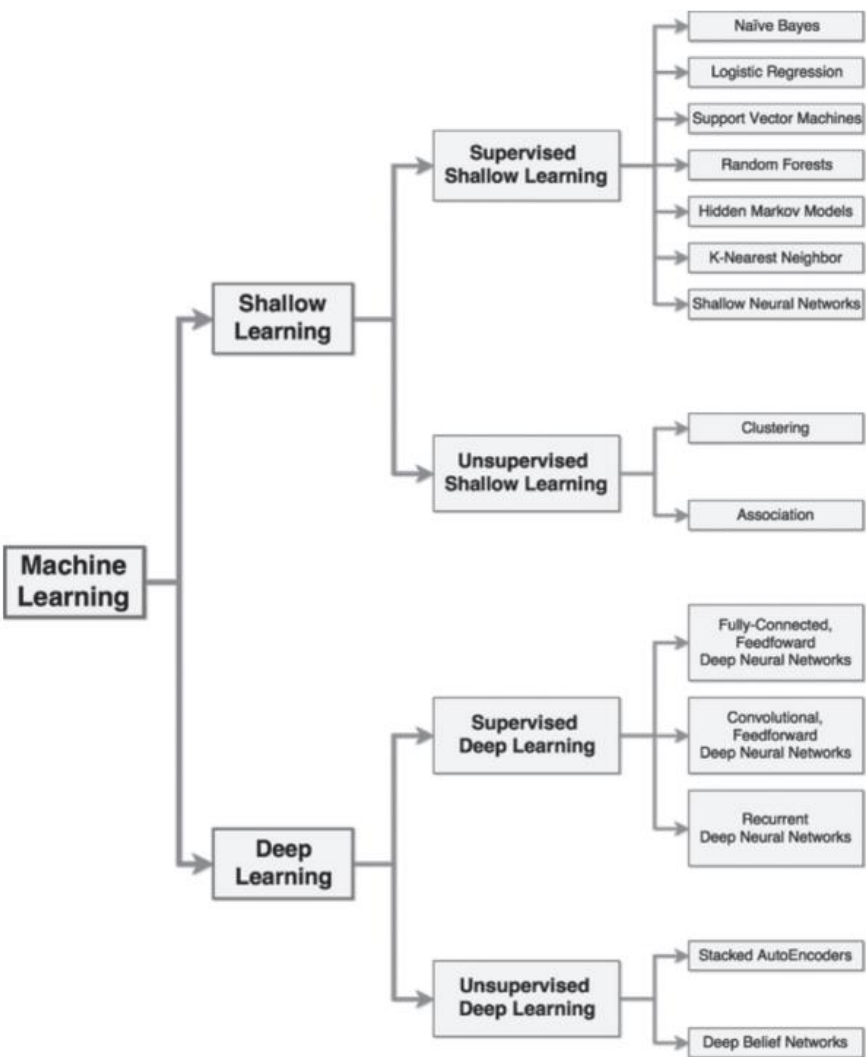


Рис.1.1 Класифікація алгоритмів та методів машинного навчання, що використовуються у сфері кібербезпеки

Перейдемо до розгляду методів алгоритму Неглибокого навчання. Почнемо з контрольованих:

Метод Наївних Байєса (NB). Цей метод є методом ймовірносних класифікаторів, котрі роблять припущення про характеристики даних заздалегідь, та початково характеризує їх як незалежні один від одного. Цей метод є масштабованим та не потребує наборів даних непомірних розмірів щоб досягти задовільних результатів.

$$P(c | x) = \frac{P(x | c)P(c)}{P(x)} \quad (1.1)$$

Likelihood
Class Prior Probability
Posterior Probability
Predictor Prior Probability

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

де $P(c | x)$ - апостеріорна ймовірність класу (c , target) даного предиктора (x , атрибуту);

$P(c)$ - апіорна ймовірність класу;

$P(x | c)$ - ймовірність предиктора даного класу;

$P(x)$ - апіорна ймовірність предиктора.

Перший крок алгоритму полягає у обчисленні апіорної ймовірності для заданих позначок класів. Потім знайти можливість для кожного атрибута класу. Потім поміщаємо ці значення у формулу Байєса і обчислюємо апостеріорну ймовірність, та дивимося, який клас має вищу ймовірність, враховуючи, що вхідні дані належать до вищого класу ймовірності.

Переваги:

1. Легко та швидко передбачити клас для набору тестових даних. Він також добре працює у прогнозуванні кількох класів.
2. Коли виконується припущення про незалежність даних, наївний класифікатор байєса працює краще в порівнянні з іншими моделями, такими як логістична регресія, і вам потрібно менше навчальних даних.
3. Він добре працює у випадку категорійних вхідних змінних порівняно з числовими змінними. Для числової змінної передбачається нормальний розподіл.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

Недоліки:

1. Якщо категоріальна змінна має категорію (у тестовому наборі даних), яка не спостерігалась у навчальному наборі даних, тоді модель присвоїть 0 (нульову) ймовірність і не зможе зробити прогноз. Це часто називають «нульовою частотою». Щоб вирішити це, ми можемо використовувати техніку згладжування. Один з найпростіших методів згладжування називається оцінкою Лапласа.

2. Наївний Байєс також відомий як поганий оцінювач, тому результати ймовірності `predict_proba` не слід сприймати надто серйозно.

Ще одним обмеженням Наївного Байєсовського методу є припущення про незалежних предикторів. У реальному житті майже неможливо отримати набір повністю незалежних предикторів.

Метод логістичної регресії (LR). Метод являє собою категоричні класифікатори, котрі приймають дискримінаційну модель. Цей метод являє собою статистичний підхід та є алгоритмом машинного навчання, який використовується для задач класифікації та заснований на концепції ймовірності. Він використовується, коли цільова змінна є категоріальною.

Він широко використовується, коли проблема класифікації є бінарною; правда чи брехня, так чи ні і т. д. Наприклад, його можна використовувати, щоб передбачити, чи є електронний лист спамом (1) чи ні (0).

Логістична регресія використовує сигмоподібну функцію(рис. 1.2), щоб повернути можливість мітки.

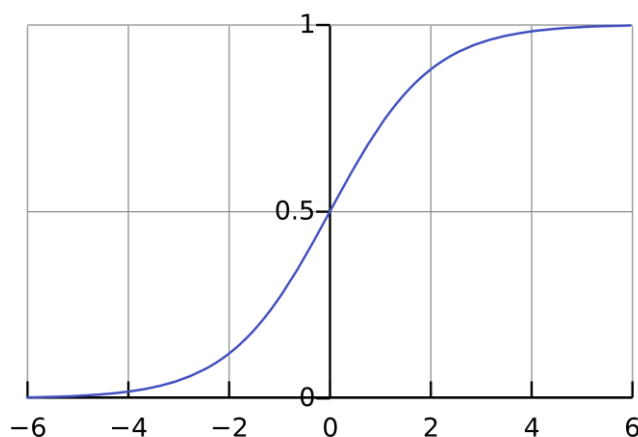


Рис.1.2 Сигмоподібна функція

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Сигмовидна функція - це математична функція, що використовується для порівняння прогнозованих значень із ймовірностями. Функція має можливість відображати будь-яке дійсне значення в інше значення в діапазоні від 0 до 1.

Переваги:

1. Логістичну регресію легше реалізувати, інтерпретувати та дуже ефективно навчати.
2. Вона не робить ніяких припущень про розподіл класів у просторі ознак.
3. Алгоритм дуже швидко класифікує невідомі записи.
4. Хороша точність для багатьох простих наборів даних, і вона добре працює, коли набір даних лінійно розділимо.

Недоліки:

1. Якщо кількість спостережень менша за кількість ознак, логістичну регресію використовувати не слід, інакше це може призвести до переоснащення.
2. Основним обмеженням логістичної регресії є припущення про лінійність між залежною змінною та незалежними змінними.
3. Нелінійні задачі не можна вирішити за допомогою логістичної регресії, тому що вона має лінійну поверхню розв'язання. Лінійно розділені дані рідко зустрічаються у реальних сценаріях.

Дерево рішень - це корисна програма машинного навчання, яку можна використовувати для вирішення задач як класифікації, так і регресії. Це потужні аналітичні моделі, які здатні обробляти дані з мінімальним часом попередньої обробки. Це інструмент підтримки, який має деревоподібну структуру та пропонує можливі наслідки та вартість рішень. Основні переваги та недоліки будуть розглянуті у наступному розділі.

Метод Опорних Векторів (SVM). Це неімовірнісні класифікатори, які відображають зразки даних у просторі з ціллю максимізації відстані між кожною категорією зразків. Вони не роблять припущення на вхідних даних. Даний метод слід використовувати у вигляді бінарних класифікаторів, оскільки вони мають не кращий результат у багатокласових класифікаціях. Також обмежена масштабованість методу може призвести до тривалого часу обробки даних.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

В алгоритмі SVM ми відображаємо кожен елемент даних як точку в n -мірному просторі (де n - кількість наявних ознак), при цьому значення кожної ознаки є значенням конкретної координати. Потім ми виконуємо класифікацію, знаходячи гіперплощину, яка дуже добре розрізняє два класи як показано на рисунку 1.3.

Опорні вектори - це просто координати індивідуального спостереження. Класифікатор SVM — це межа, яка найкраще поділяє два класи (гіперплощина/лінія).

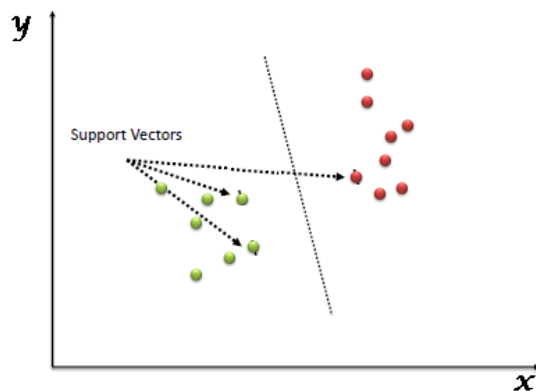


Рис.1.3 Метод опорних векторів

Метод випадкового лісу (RF). Випадковий ліс – це набір дерев рішень, котрий обчислює вихідні дані кожного дерева перед винесенням єдиної остаточної відповіді. Кожне дерево рішень являє собою умовний класифікатор: аналіз дерева проходить з верхнього вузла, та у кожному з них задана умова перевіряється на відповідність з ознаками аналізованих даних. Метод випадкового лісу ефективно підійде для великих наборів даних та виключає проблеми багатокласовості.

Переваги:

1. Машина опорних векторів дуже ефективна навіть з великими розмірними даними.
2. Якщо у вас є набір даних, де кількість функцій більше, ніж кількість рядків даних, SVM також може працювати в цьому випадку.
3. SVM можна використовувати як для регресії, так і для задачі класифікації.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

4. І останнє, але не менш важливе, SVM також може добре працювати з даними зображень.

Недоліки:

1. Нам потрібно вибрати оптимальне ядро для SVM, а це завдання складне.
2. SVM на великому наборі даних порівняно займає більше часу для навчання.
3. SVM або машина опорного вектора не є імовірнісною моделлю, тому ми не можемо пояснити класифікацію з точки зору ймовірності.
4. Важко зрозуміти та інтерпретувати модель SVM порівняно з деревом рішень, оскільки SVM є більш складним.

Приховані моделі Маркова (НММ). Моделі Маркова моделюють систему як набір станів, котрі утворюють відповіді з різними ймовірностями. Мета – визначити набір станів, котрий надав точні явні результати серед неявних. Хоча метод Маркова може проводити навчання за явними та неявними наборами даних, в кібербезпеці в основному використовуються явні набори. Також його можна використовувати для опису розвитку подій за якими спостерігаються, що залежать від внутрішніх факторів, які не піддаються безпосередньому спостереженню. Це клас ймовірнісних графічних моделей, які дозволяють нам передбачати послідовність невідомих змінних з набору змінних, що спостерігаються.

Переваги:

НММ має міцну статистичну основу з ефективними алгоритмами навчання, де навчання може відбуватися безпосередньо на основі необроблених даних послідовності. Він дозволяє послідовно обробляти штрафи за вставку і видалення у вигляді методів, що локально навчаються і може обробляти вхідні дані змінної довжини. Він також може виконувати широкий спектр операцій, включаючи множинне вирівнювання, аналіз та класифікацію даних, структурний аналіз та виявлення закономірностей. Також легко поєднувати в бібліотеки.

Недоліки:

1. Алгоритм залежить тільки від кожного стану та відповідного йому спостережуваного об'єкта.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

2. Маркування послідовності, крім зв'язку з окремими словами, також пов'язане з такими аспектами, як довжина послідовності, контекст слів та інші.

Метод К-найближчих сусідів (KNN). KNN використовується для класифікації та можуть бути використані для багатокласових проблем. Проте стадії навчання та тестування вимогливі до обчислювальних потужностей, оскільки вони класифікують кожний тестовий набір, та порівнюють його з усіма іншими навчальними.

Алгоритм К-NN — це надійний класифікатор, який часто використовується як еталон для складніших класифікаторів, таких як штучна нейронна мережа (ANN) або машина опорних векторів (SVM).

Переваги:

1. Його легко зрозуміти та просто реалізувати
2. Його можна використовувати як для задач класифікації, так задач регресії.
3. Він ідеально підходить для нелінійних даних, оскільки немає припущень щодо базових даних.
4. Він може природно обробляти випадки з декількома класами
5. Він може добре працювати з достатньою кількістю репрезентативних даних.

Недоліки:

1. Пов'язані з цим витрати на обчислення високі, оскільки в них зберігаються всі навчальні дані.
2. Потрібен великий обсяг пам'яті.
3. Прогнозування виконується повільно, якщо значення N велике.
4. Чутливий до нерелевантних функцій

Метод неглибокої нейронної мережи (SNN). Цей алгоритм базується на нейронних мережах, котрі складаються з набору опрацювання елементів (тобто нейронів), організованих у два або більше комунікативних шарів. SNN включає усебі усі типи нейронних мереж з обмеженою кількістю нейронів та

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

шарів. Не дивлячись на існування незалежних SNN, у кібербезпеці надається перевага саме цьому методу для задач класифікації.

Переваги:

1. Нейронні мережі є гнучкими і можуть використовуватися як для задач регресії, так і для класифікації. У моделі можна використовувати будь-які дані, які можна зробити чисельними, оскільки нейронна мережа — це математична модель з функціями наближення.

2. Нейронні мережі хороші для моделювання нелінійних даних з великою кількістю вхідних даних; наприклад, зображення. Він надійний у своєму багатофункціональному підході. Він працює шляхом розбиття проблеми класифікації на багаторівневу мережу простіших елементів.

3. Після тренування прогнози робляться досить швидко.

4. Нейронні мережі можна навчати з будь-якою кількістю вхідних даних і шарів.

5. Нейронні мережі найкраще працюють з великою кількістю точок даних.

Недоліки:

1. Нейронні мережі – це чорні ящики, тобто ми не можемо знати, наскільки кожна незалежна змінна впливає на залежні змінні.

2. Навчання на традиційних процесорах потребує великих обчислювальних ресурсів та часу.

3. Нейронні мережі багато в чому залежить від навчальних даних. Це призводить до проблеми перенавчання та узагальнення. Цей режим більше покладається на навчальні дані та може бути налаштований на ці дані. [11].

Щодо неконтрольованих методів, їх існує два типи:

1) Кластеризація. Це групи даних, що мають схожі характеристики. Цей метод включає в себе добре відомі підходи k-means та ієрархічну кластеризацію. Кластеризація має обмежену масштабованість, проте являє собою гнучке рішення, котре можна використати як попередній етап перед застосуванням контрольованого алгоритму або для виявлення аномалій.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

2) Асоціація. Асоціативний метод спрямований на ідентифікацію невідомих закономірностей між даними, що робить їх придатними для прогнозування. Однак вони, як правило, створюють надмірну кількість необов’язково вірних вихідних даних. Саме тому слід застосовувати цей метод під наглядом фахівця [12].

Роблячи висновок та маючи статистичний приклад (рис. 1.4), можна сказати, що дерево рішень буде найбільш точним, що вирішує питання обрання методу навчання.

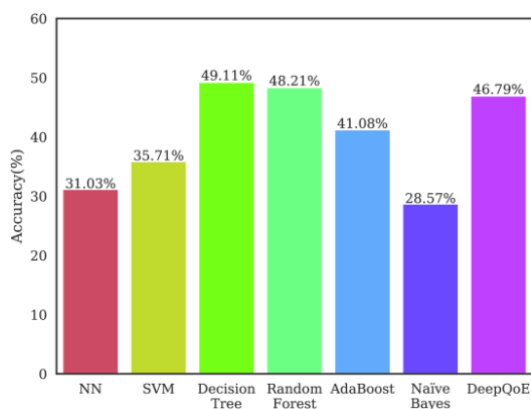


Рис.1.4 Статистичний приклад-порівняння методів машинного навчання

1.3 Аналіз складових типового конвеєру машинного навчання

Принцип роботи машинного навчання являє собою набір вхідних даних та набір відповідних вихідних даних, задля того, щоб знайти між ними співвідношення. За цим алгоритмом стоїть 6 складових елементів:

1) Набір даних. Машини потребують велику кількість даних для навчання, опрацювання та винесення рішень. Цими даними може бути будь-який неопрацьований факт, величина, звук, зображення або текст, який потрібно розпізнати та проаналізувати. Набір даних – це консолідовані дані подібного типу, що зафіксовано у різних середовищах.

Наприклад, набір даних банкнот матиме зображення цих самих банкнот, зроблених з різних ракурсів, освітлення, камер та фонами, щоб досягти максимальної точності в класифікації та ідентифікації. Коли набір даних готовий, він використовується для навчання, валідації та тестування моделі машинного навчання.

Чим більший набір даних, тим кращий навчальний потенціал для моделі та вищі шанси досягти бажаних результатів точності.

Існує правило у створенні та підборі набору даних, так зване правило “5V”. За цим правилом, якщо набір даних матиме великий обсяг, різноманітно підібрані дані, швидко буде поповнюватися, та буде містити лише правдиві значущі дані, такий набір буде найефективнішим для навчання машини.

2) Алгоритми. Алгоритм являє собою математичну або логічну програму, яка перетворює набір даних у модель. Існують різні типи алгоритмів, котрі можна вибрати залежно від типу проблеми, доступних ресурсів та характеру даних. Алгоритми машинного навчання використовують обчислювальні методи, щоб «вивчати» інформацію безпосередньо з даних, не покладаючись на заздалегідь визначене рівняння як модель.

3) Моделі. У машинному навчанні модель – це обчислювальне представлення процесів реального світу. Модель машинного навчання тренується розпізнавати певні типи паттернів шляхом циклічного повторення набору даних, використовуючи відповідні до набору алгоритми.

Наприклад, існує деяка програма, яка класифікує машини на основі їх побудови, модель тренується на наборі даних зображень з різними ознаками. Чим більше машин модель переглядає, тим кращим буде результат розпізнавання та категоризація [13].

4) Вилучення ознак. Набори даних можуть мати кілька ознак. Якщо функції в наборі даних подібні або відрізняються значною мірою, то збережені в наборі даних спостереження, вірогідніше всього, призведуть до перенавантаження моделі.

Перенавантаження моделі відбувається, коли модель вивчає сторонні деталі та шум у навчальних даних до такої міри, що це призводить до негативних наслідків у вивченні нових даних.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Задля подолання цієї проблеми необхідно упорядковувати кількість ознак в наборах даних, використовуючи методи вилучення ознак. Метод вилучення ознак спрямований на зменшення їх кількості в наборі даних шляхом створення нових ознак з вже існуючих.

5) Навчання. Навчання включає підходи, які дозволяють моделі машинного навчання визначати закономірності та приймати рішення. Існують різні способи досягнення цього результату, включаючи кероване навчання, некероване навчання, посилене навчання та ін.

6) Оцінка. Оцінка – заключний етап. Після процесу навчання ви маєте перевірити, що модель визначає саме вірні результати, наприклад, вірний тип машини за фото. Оцінка моделі проходить за тестовими даними, котрі передаються моделі, далі проводиться аналіз точності результатів прийнятих моделлю рішень [14].

1.4 Аналіз алгоритму дерева рішень та його модифікацій

Проблеми класифікації є найпоширенішим використанням дерев рішень у машинному навчанні. Це завдання контрольованого машинного навчання, у якій модель навчається класифікувати, чи є дані частиною відомого класу об'єктів. Моделі навчені призначати мітки класів для даних. Класи вивчаються моделлю у вигляді обробки помічених навчальних даних у навчальній частини життєвого циклу моделі машинного навчання .

Щоб вирішити проблему класифікації, модель повинна розуміти функції, що класифікують точку даних за різними мітками класів. Насправді проблема класифікації може виникнути у різних умовах. Приклади можуть включати класифікацію документів, програмне забезпечення для розпізнавання зображень або виявлення спаму в електронній пошті.

Дерево класифікації – це спосіб структурування моделі для класифікації об'єктів чи даних. Листя або кінцеві точки гілок у дереві класифікації - це мітки класів, точки, в яких гілки перестають розділятися. Дерево класифікації створю-

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

ється поступово, у своїй загальний набір даних розбивається більш дрібні підмножини. Він використовується, коли цільові змінні є дискретними або категоріальними, при цьому розгалуження зазвичай відбувається за допомогою двійкового розбиття. Наприклад, кожен вузол може розгалужуватися за відповіді «так» або «ні». Древа класифікації використовуються, коли цільова змінна є категоріальною або може бути присвоєна певна категорія, наприклад так чи ні. Кінцевим пунктом кожної галузі є одна категорія.

Древа рішень – це форма прогнозного моделювання, що допомагає зіставити різні рішення чи рішення із заданим результатом. Дерево рішень складається із різних вузлів. Кореневий вузол - це початок дерева рішень, яке зазвичай є весь набір даних у рамках машинного навчання. Листові вузли є кінцевою точкою гілки або кінцевим результатом серії рішень. Дерево рішень більше не розгалужуватиметься від кінцевого вузла. У деревах рішень у машинному навчанні характеристики даних є внутрішніми вузлами, а результатом є кінцевий вузол. [15].

В кожному дереві рішень є два вузли: Вузол рішень та Листовий вузол. Вузли рішень використовуються для винесення будь-яких рішень, а Листові вузли для виводу цих самих рішень, та не мають подальших гілок (рис. 1.5). Рішення або перевірка виконуються на основі особливостей даного набору даних.

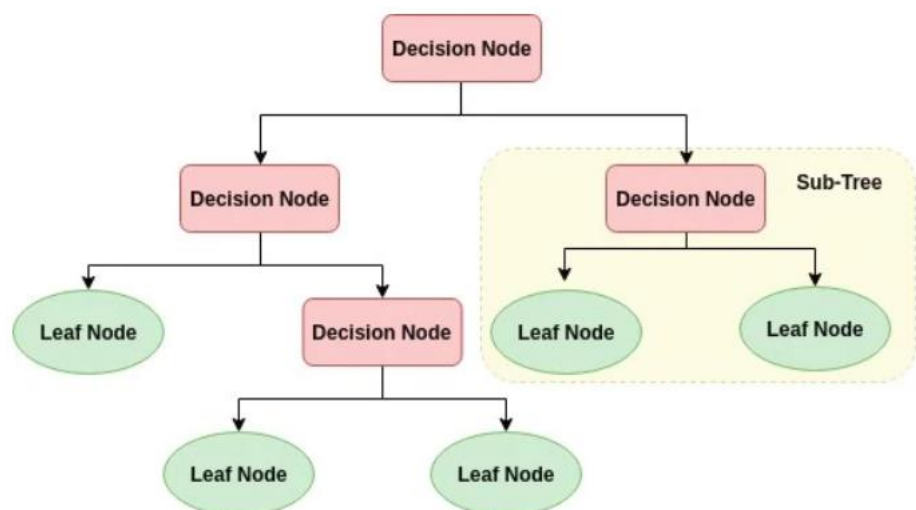


Рис.1.5 Приклад топології дерева рішень

Слід зазначити складові дерева рішень:

- Кореневий вузол (Root Node). Це початок дерева рішень. Він являє собою весь набір даних, котрий потім буде розділено на два або більше гомогенних набори.
- Листовий вузол (Leaf Node). Листові вузли – це фінальний результат, після листових вузлів дерево не може далі розподілятися.
- Розподілення (Splitting). Розподілення – це процес розділення вузлів рішень, або ж кореневого вузла на менші, під-вузли, згідно до наданих умов.
- Гілка (Branch). Дерево, яке було сформовано в результаті ділення.
- Обрізання (Pruning). Процес видалення небажаних гілок з дерева.
- Батьківський/Дочірній вузол (Parent/Child Node). Кореневий вузол дерева називається батьківським деревом, інші вузли – дочірні.

У дереві рішень прогнозування класу для заданого набору класів починається з кореневого вузла. Алгоритм порівнює значення кореневого атрибуту з атрибутом запису (реального набору даних), та, на основі порівняння, слідує за гілкою та переходить на інший вузол. Для наступного вузла алгоритм знову порівнює значення атрибутів з іншим під-вузлом та рухається далі. Цей процес продовжується, поки не досягне листового вузла дерева.

При застосуванні дерева рішень головною проблемою є знаходження найкращих атрибутів для кореневого та підвузлів. Просто випадковий вибір будь-якого вузла як кореня не може вирішити проблему. Якщо ми будемо наслідувати випадковий підхід, це може дати нам погані результати з низькою точністю. Для рішення цього питання використовується модифікуючий метод ASM (Attribute Selection Measure). За цим методом можна легко обрати кращий атрибут для вузлів [16].

Існує дві техніки для ASM:

1) Набір інформації. Набір інформації – це вимірювання змін в ентропії після розділення набору даних на основі атрибута. Він обчислює скільки інформації надає функція про клас. Відповідно до значення набору інформації, ми розби-

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

ваємо вузол і будуємо дерево рішень. Алгоритм дерева рішень завжди намагається максимізувати значення набору інформації, а вузол/атрибут, що має найвищий інформаційний приріст, розділяється першим.

2) Індекс Джині. Індекс Джині – це міра чистоти, котра використовується у створенні дерева рішень у алгоритмі CART. Атрибути, що мають низький індекс Джині, будуть важливіші, ніж атрибути з високим індексом. Він створює лише двійкові розбиття, та алгоритм CART використовує індекс Джині саме для цього [17].

Ці критерії будуть обчислювати значення кожного атрибута. Значення сортуються і атрибути розміщуються в дереві відповідно до порядку, тобто атрибут з великим значенням (у разі приросту інформації) поміщається в корінь.

При використанні приросту інформації як критерій ми припускаємо, що атрибути є категоріальними, а індексу Джині атрибути передбачаються безперервними.

Занадто велике дерево викликає великі ризики перенавантаження моделі, також і занадто мале дерево може не охопити усі важливі атрибути набору даних. Задля уникнення таких проблем використовується метод «Обрізання», котрий зменшує розміри дерева, не зменшуючи при цьому точність (рис. 1.6). Існує два типи технології. Обрізання гілок:

1) Обрізання на основі вартісної оцінки. Це комплексний метод створення набору дерев з одного початкового дерева від T_0 до T_m , де T_0 – початкове дерево, а T_m – лише корінь. На кроці розмірністю i створюється дерево, шляхом видалення під-дерева від дерева $i - 1$, та заміни видаленого елементу листовим вузлом. Після створення усіх дерев обирається найкраще за загальною точністю дерево (рис. 1.7).

2) Обрізання методом зменшення помилок. Починаючи з листків, кожен вузол замінюється найпопулярнішим класом. Якщо точність не падає, зміна зберігається. Таким чином проходиться усе дерево та змінюється його структура на

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

основі заміни (рис. 1.8-1.9). Хоча таке рішення може здаватись занадто прямолинійним, проте простота та швидкість такого методу виправдовують його застосування [18].

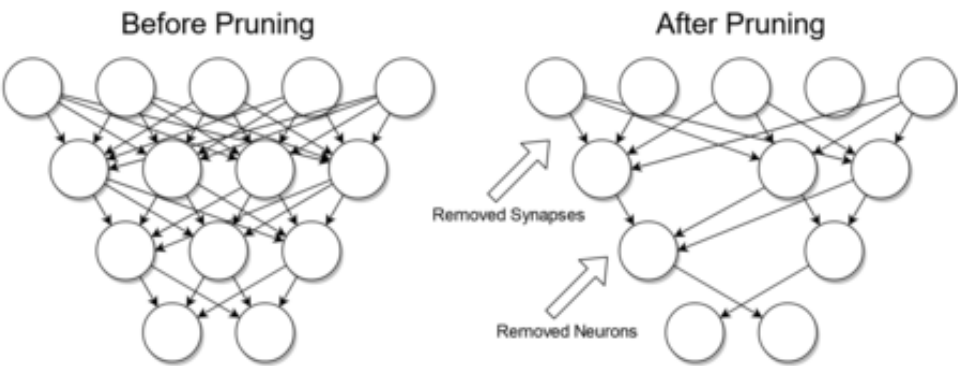


Рис.1.6 Приклад обрізання дерева рішень

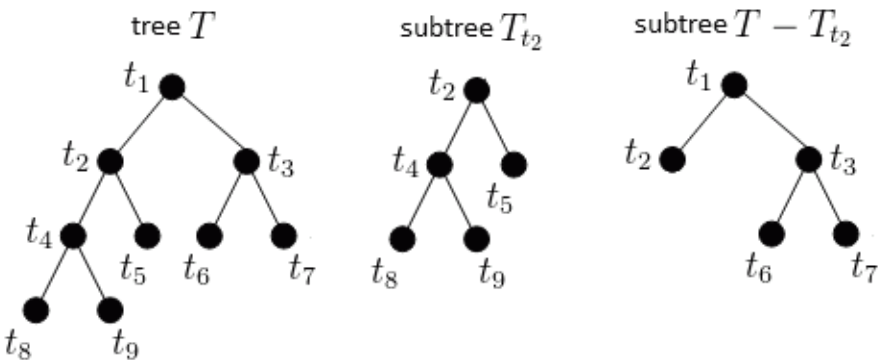


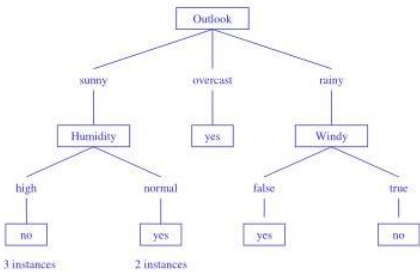
Рис.1.7 Приклад обрізання за вартісною оцінкою

Reduced-Error Pruning

Split data into growing and validation sets.

Pruning a decision node d consists of:

1. removing the subtree rooted at d .
2. making d a leaf node.
3. assigning d the most common classification of the training instances associated with d .



Accuracy of the tree on the validation set is 90%.

Рис.1.8 Приклад обрізання методом зменшення помилок

Reduced-Error Pruning

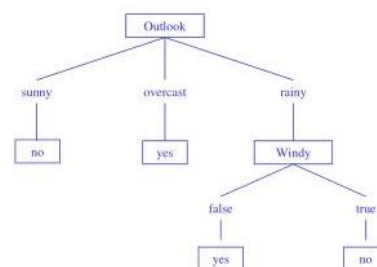
Split data into growing and validation sets.

Pruning a decision node d consists of:

1. removing the subtree rooted at d .
2. making d a leaf node.
3. assigning d the most common classification of the training instances associated with d .

Do until further pruning is harmful:

1. Evaluate impact on validation set of pruning each possible node (plus those below it).
2. Greedily remove the one that most improves validation set accuracy.



Accuracy of the tree on the validation set is 92.4%.

Рис.1.9 Приклад обрізання методом зменшення помилок

Дерева рішень - популярний підхід у машинному навчанні з поважних причин. Результуюче дерево рішень є просто для розуміння завдяки візуалізації процесу прийняття рішень. Це спрощує процес пояснення вихідних даних моделі зацікавленим сторонам, які не мають спеціальних знань у сфері аналізу даних. Зацікавлені сторони, які не є фахівцями, можуть отримати доступ та зрозуміти візуалізацію моделі та даних, тому дані доступні для різних бізнес-груп. Отже, можна ясно зрозуміти аргументацію чи логіку моделі. Пояснюваність може бути перешкодою для впровадження машинного навчання в організаціях, тому це явна перевага використання дерев рішень у машинному навчанні.

Ще одна перевага полягає в етапі підготовки даних для моделей машинного навчання дерева рішень. Моделі дерев рішень вимагають менше очищення даних, порівняно з іншими підходами до моделей машинного навчання. Зокрема, дерева рішень дозволяють уникнути необхідності нормалізації даних на ранній стадії процесу машинного навчання. Моделі дерев рішень можуть обробляти як категоріальні, так і числові дані, тому якісні змінні не потрібно перетворювати, як у інших методах.

До основних переваг дерев рішень у машинному навчанні належать:

- Простота розуміння навіть зацікавленими сторонами без знання технічних даних.

- Внутрішньо можна пояснити, оскільки будь-яке рішення в рамках моделі може бути пояснено в рамках моделі. Це відрізняється від алгоритмів чорної скриньки, де важко пояснити.

- Дані не потребують нормалізації, оскільки метод може обробляти як числові, так і категоріальні змінні.

- Може використовуватися для розуміння ієрархії функцій набору даних, яку можна скоротити і уточнити в майбутньому моделюванні.

Одним із основних недоліків використання дерев рішень у машинному навчанні є проблема перенавчання. Метою моделей машинного навчання є досягнення надійного рівня узагальнення, щоб модель могла точно обробляти невидимі дані після розгортання. Перенавчання – це коли модель надто близько підходить до навчальних даних, тому може стати менш точною при зіткненні з новими даними або прогнозування майбутніх результатів.

Переоснащення може бути серйозною проблемою для дерев рішень, які часто можуть бути дуже складними і занадто великими. Але з цим допомагає процес обрізки, який розглянули вище. При обрізанні видаляються гілки та вузли дерева, які не мають відношення до цілей моделі або не надають додаткової інформації. Будь-яке скорочення має бути виміряне за допомогою процесу перехресної перевірки в машинному навчанні, яке може оцінити здатність моделі функціонувати або її точність у реальному середовищі.

Деякі з основних недоліків дерев рішень у машинному навчанні, які необхідно враховувати, наведено нижче:

- Дерева рішень можуть стати дуже складними, тому вимагають скорочення та оптимізації.

- Перенавчання — поширена проблема, яка знову ж таки потребує скорочення та оптимізації гіперпараметрів.

- Невеликі зміни в навчальних даних можуть вплинути на дерево рішень, що часто призводить до створення різних дерев рішень.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

- Цей підхід може бути менш точним для завдань регресії, ніж інші методи машинного навчання, особливо безперервними числовими вихідними даними.

- Схильний до упередженості у моделях машинного навчання, якщо набір навчальних даних не є збалансованим чи репрезентативним.

1.5 Постановка завдання на розробку

Основною задачею для створення детектору атак є створення моделі машинного навчання засобами мови програмування Python та модулів до неї. Проект являтиме собою програмне відтворення моделі класифікації за алгоритмом дерева рішень. Побудова моделі складатиметься з таких пунктів:

- Імпорт бібліотек машинного навчання та KDD Dataset до середовища розробки Python. Вони є у відкритому доступі у каталозі розширень Python;

- Опрацювання вхідного набору даних. У цьому пункті слід визначити типи даних, нульові значення, форми та ін.;

- Підготовка Даних. Підготовка даних буде заключатися у відборі атрибутів, визначені масштабів, нормалізації і перетворенні, та інших діях для максимальної ефективності навчання машини;

- Забезпечення моделі навчальними та тестувальними даними. Для якісного навчання слід задати моделі обсяг навчального матеріалу. Після процесу навчання слід перевірити модель на точність винесення рішень за допомогою тестових даних;

- Процес створення та навчання моделі. Ключовий етап проекту. Оскільки алгоритм було обрано заздалегідь, потрібно провестися усі базові налаштування та створити модель у програмованому середовищі, провести оптимізацію;

- Якісна оцінка моделі. Етап «полірування» моделі. Проведення усіх тестів метрик, продуктивності та неточностей, встановлення якості моделі та визначення її готовності до штатної роботи;

- Реліз моделі як готового програмного забезпечення.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

- Надання вказівок та програмної бази для роботи з продуктом та подальша підтримка.

Висновки до 1-го розділу:

В результаті аналізу основних складових машинного навчання було чітко визначено основні принципи роботи методів та алгоритмів МН, та обрано найбільш доцільні для досягнення поставлених задач.

Проаналізовано 7 компаній, що використовують машинне навчання як основну технологію для систем захисту, за аналізом наведено актуальність машинного навчання у сфері безпеки в цілому.

Також було розглянуто принцип побудови методу класифікації та алгоритму дерева рішень. Досліджено деталі та особливості роботи з даними технологіями та проведено порівняльні характеристики.

		Погребніченко П.К			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				32
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2 РОЗРОБКА ДЕТЕКТОРА АТАК НА ОСНОВІ АЛГОРИТМІВ ДЕРЕВ РІШЕНЬ З РЕАЛІЗАЦІЄЮ НА PYTHON

2.1 Обґрунтування вибору мови програмування

З часом стає все більш очевидним, що майбутнє обчислювальних технологій буде залежати виключно від штучних інтелектів та машин, що замінять будь-яку активність людини. Саме над цим питанням багато років працює індустрія, та не менш важливим питанням є вибір мови програмування, котра найбільш підійде для застосування у цій галузі.

На нинішній час існує декілька мов, котрі активно використовують у написанні штучних інтелектів та навчанні машин, серед них:

- Python. Python може підтримувати складні сценарії та онлайн-додатки, що робить його ідеальним вибором як для початківців, так і для досвідчених розробників. Це мова програмування загального призначення, розроблена 1991 року. Через свою простоту та легкість використання він вже давно став фаворитом серед розробників. Він використовується з безліччю різних фреймворків та бібліотек.

- Мова програмування R. R зазвичай застосовують для аналізу даних та обчислень у статистичних цілях. Ця мова є розширюваною за допомогою так званих пакетів, або ж бібліотек (Gmodels, TM, RODBC), котрі допомагають у створенні проектів машинного навчання. Вони вирізняються швидкістю роботи, простотою та якісною реалізацією бізнес-логіки. R було початково створено математиками та статистиками для їх потреб, з чим мова чудово впорується. Крім того, якщо у ваших цілях є потреби побудови графіків та діаграм, ви також можете використовувати такі пакети, як: ggplot2, ggvis, googleVis, Shiny, rCharts та ін., за допомогою яких можна перетворити графіки у інтерактивні веб-програми. У порівнянні з Python, R працює повільніше та менш якісно, коли мова йде про великі обсяги даних.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

- **Scala.** Scala – справжнє спасіння, коли йде мова про великомасштабні моделі та великий набір даних. Ця мова пропонує науковцям великий набір інструментів для роботи з даними, наприклад Saddle, Scalalab або Breeze. Scala має значну підтримку паралельності, задля швидшого опрацювання великих обсягів даних. Оскільки Scala працює на JVM, тому вона є майже необмеженою у потужності завдяки Hadoop, open-сорс фреймворку розподіленої обробки, котрий може оброблювати та зберігати великі обсяги даних на базі кластерних систем. Scala є прямим конкурентом для Python, проте значно менший обсяг інструментів та бібліотек залишає питання спірним.

- **Julia.** Якщо вам потрібно рішення для високоточних машинних обчислень та аналізу, слід розглянути мову програмування Julia. Вона має подібний до Python синтаксис, та була створена саме для задач обчислень та розрахунків. Julia надає підтримку глибокого навчання завдяки TensorFlow.jl та фреймворку Mocha. Проте, через те, що ця мова досить нова, вона не підтримується багатьма бібліотеками та має замалу спільноту, порівняно з Python.

- **Java.** Java – це об'єктно-орієнтована мова програмування, котра вже багато років розвивається та широко використовується у світі у найрізноманітніших цілях, від мобільних додатків до розробки серверних структур. Вона підтримується багатьма бібліотеками для машинного навчання, такими як WEKA та Rapdiminer. Java широко використовується для обробки природних мов, алгоритмів пошуку та нейронних мереж. Це дозволяє швидко створювати великомасштабні системи з чудовою продуктивністю. Проте, якщо вам потрібні статистичні моделі та графічне відображення, Java – остання мова, котру ви хочете використовувати. Справа в тому, що для цього бракує бібліотек та розширень, а вже існуючі не можуть покрити усі питання. Python, з іншого боку, має передові інструменти, які добре підтримуються спільнотою [19].

- **C/C++.** Мови програмування, такі як C/C++, сильні, адаптовані та широко використовуються у всьому світі. І немає шляху назад, коли доходить до розробки алгоритмів машинного навчання. C++ пропонує можливість змінювати алгоритми та здійснювати розширене управління пам'яттю краще, ніж будь-яка

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

інша мова програмування. За допомогою цієї мови розробники можуть налаштувати широкий спектр параметрів продуктивності.

Підведемо підсумок у таблиці 1.1, порівнявши основні переваги та недоліки кожної мови програмування.

Таблиця 1.1

Порівняння мов програмування

Мова	Переваги	Недоліки
Python	– Висока продуктивність – Має багату різноманітність бібліотек та інструментів – Динамічна типізація	– Досить повільна – Не кращий варіант для мобільного програмування – Доступ до баз даних
R	– Графічні представлення – Плавна сумісність – Багато пакетів	– Погана безпека – Досить повільна
Scala	– Легко налагоджується – Масштабованість – Досить хороша підтримка IDE – Відмінно підходить для аналізу даних	– Обмежена кількість розробників – Складна для розуміння
Julia	– Вийняткова числова точність – Ефективний компілятор – Розподілене паралельне виконання – Велика бібліотека математичних функцій.	– Пакети дійсно містять безліч помилок, проблем з надійністю та довільних невеликих характеристик. – Не може усувати помилки інструментів.
Java	– Зручний онлайн-компілятор Java – Безпечно та зручно – Об'єктно-орієнтована – Ефективна	– Важчий для сприйняття – Займає багато місця в пам'яті – Складний код
C/C++	– Портативний – Сумісний з усім сімейством C – Об'єктно-орієнтований – Масштабований	– Проблеми з безпекою – Немає сміттєзбірника – Використовує покажчики

Кажучи про бібліотеки та фреймворки у реалізації машинного навчання та штучних інтелектів, не слід забувати, що вкрай важливим є середовище розробки, яке повинне бути структурованим та добре перевіреним, щоб розробники могли знайти найкращі рішення для написання коду. Задля скорочення часу розробки та зменшення затрат ресурсів, програмісти звертаються до ряду бібліотек та фреймворків у Python. Тому саме ця мова була обрана для реалізації поставленого завдання. Перед цим слід зазначити що таке Бібліотека. Бібліотека – це попередньо написаний код, який інші розробники можуть використовувати у своїх проектах та цілях для вирішення завдань програмування. Python, з його багатим стеком технологій, має обширний набір бібліотек для задач штучного інтелекту та машинного навчання, зокрема:

- Keras використовується для глибокого навчання. На додаток до центрального процесора комп'ютера він також використовує графічний процесор, що дозволяє виконувати швидкі розрахунки та прототипування. Це високорівнева бібліотека нейронних мереж з відкритим вихідним кодом, яка здатна працювати на Theano, TensorFlow або CNTK. Її розробив один із інженерів Google Франсуа Шолле. Вона зроблена зручною для користувача, розширюваною та модульною для прискорення експериментів із глибокими нейронними мережами.

- Scikit-learn працює з основними алгоритмами машинного навчання, такими як кластеризація, логістична та лінійна регресія, регресія та класифікація. Далі ця бібліотека буде розглянута детальніше.

- NumPy - це бібліотека Python з відкритим вихідним кодом, яка полегшує виконання складних числових операцій. Робота з програмами машинного навчання та глибокого навчання включає складні числові операції з великими наборами даних. NumPy робить реалізацію цих операцій відносно простою та ефективною в порівнянні з їх чистою реалізацією Python. По суті NumPy реалізує свою структуру даних (n-мірний масив), яка схожа на звичайний список Python. Більшість мов програмування мають лише концепцію масивів. Python реалізує списки, які працюють як масиви, але є відмінності. Одне ключове різницю між звичайними списками Python і NumPy у тому, що списки Python можуть містити

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

різні типи даних. Навпаки, масив NumPy може містити лише об'єкти одного типу даних. Інакше кажучи, структури даних однорідні. Хоча це може бути недоліком, це дозволяє виконувати операції NumPy швидше, оскільки вони можуть уникнути перетворень і обмежень при виконанні обчислень.

- SciPy може виконувати потужні маніпуляції із зображеннями, які часто використовуються на етапі попередньої обробки у кількох програмах машинного навчання. Наприклад, розмиття по Гауссу - один з найчастіше використовуваних фільтрів при роботі з програмами машинного навчання.

- Pandas - це пакет Python з відкритим вихідним кодом, який найбільш широко використовується для обробки даних/аналізу даних та задач машинного навчання. Він побудований поверх іншого пакета під назвою NumPy, який забезпечує підтримку багатовимірних масивів. Як один з найпопулярніших пакетів обробки даних, Pandas добре працює з багатьма іншими модулями обробки даних в екосистемі Python і зазвичай включається до кожного дистрибутиву Python, від тих, що поставляються з вашою операційною системою, до дистрибутивів комерційних постачальників, таких як ActivePython від ActiveState.

- Seaborn - приголомшлива бібліотека візуалізації Python, побудована поверх matplotlib. Seaborn – це бібліотека для створення статистичної графіки на Python. Він забезпечує високорівневий інтерфейс для matplotlib і тісно інтегрується зі структурами даних pandas. Функції в бібліотеці seaborn надають декларативний API, орієнтований на набори даних, що дозволяє легко перетворювати питання даних у графіку, яка може відповісти на них. Отримавши набір даних та специфікацію графіка, Seaborn автоматично зіставляє значення даних із візуальними атрибутами, такими як колір, розмір або стиль, виконує внутрішні статистичні перетворення та прикрашає графік інформативними мітками осей та легендою. Багато seaborn функцій можуть генерувати цифри з декількома панелями, які викликають порівняння між умовними підмножинами даних або між різними парами змінних набору даних. Seaborn призначено для використання протягом усього життєвого циклу наукового проекту. Створюючи повну графіку з одного виклику функції з мінімальною кількістю аргументів, seaborn полегшує швидке

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

прототипування та дослідницький аналіз даних. І, пропонуючи широкі можливості для налаштування, а також розкриваючи базові об'єкти matplotlib, його можна використовувати для створення полірованих малюнків якості публікації.

- TensorFlow використовується для маніпулювання глибоким розумінням шляхом створення, навчання та використання штучних нейронних мереж з використанням значних наборів даних. TensorFlow, створена командою Google Brain і вперше випущена для широкої публіки в 2015 році, є бібліотекою з відкритим вихідним кодом для чисельних обчислень і великомасштабного машинного навчання. TensorFlow об'єднує безліч моделей та алгоритмів машинного та глибокого навчання (також відомих як нейронні мережі) та робить їх корисними за допомогою загальних програмних метафор. Він використовує Python або JavaScript для надання зручного інтерфейсного API для створення програм при виконанні цих програм на високопродуктивному C++.

Scikit-learn має різноманітні алгоритми класифікації, регресії та кластеризації, включаючи методи опорних векторів, випадкового лісу, підвищення градієнта, k-середніх та DBSCAN. Також він чудово працює з вищевказаними бібліотеками для обчислень NumPy та SciPy. Завдяки цим бібліотекам та розширенням, котрі суспільство постійно покращує, можна досить швидко розвивати свій проєкт та не змушувати розробників знову винаходити колесо, що суттєво полегшить їм роботу та надасть час на вирішення дійсно важливих питань.

Не слід забувати й про портативність Python. Ця мова програмування є агностичною до платформ, та працює всюди, від настільного ПК до телефону. Такий фактор є дуже важливим серед розробників, адже це надає змогу опрацьовувати свої проєкти на одній машині, та використовувати їх на іншій без будь-яких (або з мінімальними) змінами. Python підтримується багатьма платформами, включаючи Windows, Linux, macOS, Android та ін. Код Python можна використовувати для скриптових команд операційних систем, що означає, що дана мова може легко працювати навіть без інтерпретатора Python. Більшість розробників використовують веб-сервіси Amazon або Google для потреб машинного навчання або штучних інтелектів, проте це коштує набагато більше та потребує набагато

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

потужніші системи, в той час, коли Python може виконувати подібні процеси навчання значно дешевше та легше.

Тепер до самого значного – популярність та підтримка спільноти. У опитуванні Developer Survey 2021, проведеному Stack Overflow, Python увійшов до п'ятірки найпопулярніших мов програмування, що означає, що ІТ компанії по всьому світу широко використовують його у найрізноманітніших цілях, і ця тенденція лише збільшується. У опитуванні розробників Python 2020 вказано, що Python зазвичай використовується у веб-розробці. На перший погляд веб-розробка є основним напрямком, та становить 26% випадків використання, проте якщо поєднати Data Science та Машинне навчання, вони складають приголомшливі 27% (рис. 2.1) [21].

What do you use Python for the most? (single answer)

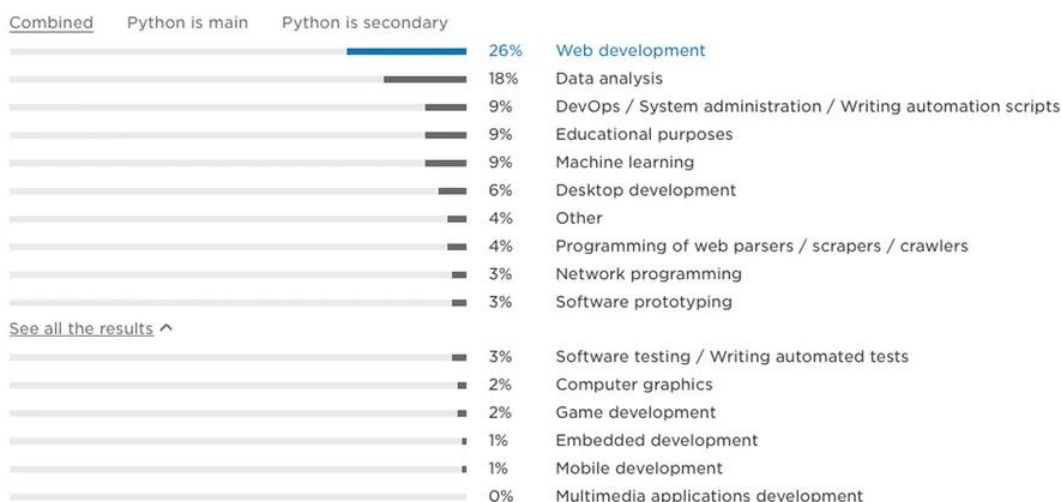


Рис.2.1 Статистика цілей використання Python

Онлайн репозиторії містять понад 140 000 створених користувачами пакетів програмного забезпечення для Python. Все ті ж бібліотеки NumPy, SciPy та Matplotlib можуть бути легко використані у проектах на основі Python. Дані пакунки використовуються для машинного навчання та допомагають розробникам помічати паттерни поведінки у великих наборах даних. Навіть Google використовує Python для аналізу веб-сторінок, всесвітньо відома студія Pixar використовує цю мову для створення своїх фільмів, а компанія Spotify – для створення алгоритмів рекомендацій пісень. В останні роки спільнота Python AI значно зросла

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

по всьому світу. Існує безліч форумів з машинного навчання та алгоритмів штучного інтелекту для Python, в котрих можна поділитись досвідом, вирішити проблеми та питання, надбати навичок та знань. Для майже будь-якого завдання, яке вам потрібно вирішити, з високою вірогідністю знайдеться відповідь на одному з таких форумів. Також навіть самі розробники Python нерідко відвідують такі форуми та відповідають на проблемні запитання. Тож, можна сміливо сказати, що у спільноті Python ви ніколи не будете самотні у своїй проблемі, та обов'язково знайдете рішення для своїх задач.

За усим вищезазначеним, було прийняте рішення, що для даної кваліфікаційної роботи мова програмування Python підійде найкращим чином.

2.2 Життєвий цикл проекту машинного навчання

Проект машинного навчання починається з розуміння бізнес-цілі. Зазвичай бізнес-аналітик працює з клієнтом і аналітиком даних, щоб перетворити бізнес-проблему на інженерний проект. Інженерний проект може мати або не мати частину машинного навчання. Ми звісно, розглядаємо інженерні проекти, в яких задіяно деяке машинне навчання.

Після того, як інженерний проект визначено, з цього починається сфера інженерії машинного навчання. У рамках більш широкого інженерного проекту машинне навчання має спочатку мати чітко визначену мету. Метою машинного навчання є специфікація того, що статистична модель отримує як вхідні дані, що вона генерує як вихідні дані, а також критерії прийнятної (або неприйнятної) поведінки моделі.

Мета машинного навчання не обов'язково збігається з бізнес-метою. Бізнес-мета – це те, чого хоче досягти організація. Наприклад, бізнес-метою Google із Gmail може бути зробити Gmail найбільш використовуваною службою електронної пошти у світі. Google може створити кілька інженерних проектів машинного навчання для досягнення цієї бізнес-цілі. Метою одного з цих проектів машинного навчання може бути розрізнення

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

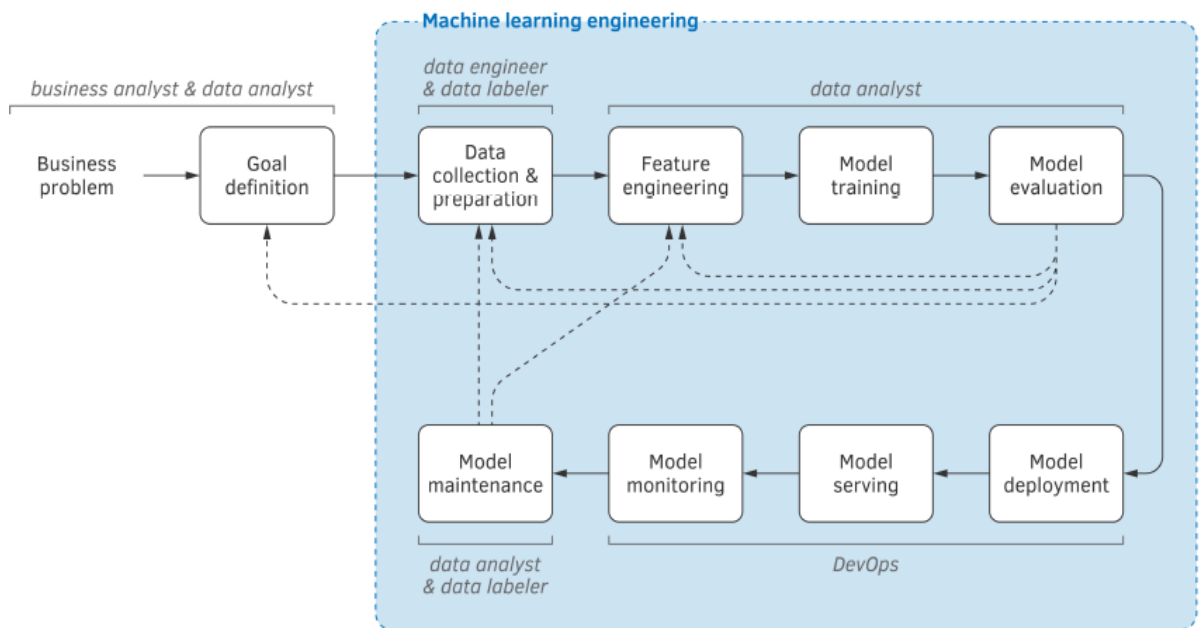


Рис.2.2 Життєвий цикл проекту машинного навчання

Загалом, життєвий цикл проекту машинного навчання, показаний на рис. 2.2 , складається з наступних етапів: 1) визначення мети, 2) збір і підготовка даних, 3) виділення ознак, 4) навчання моделі, 5) оцінка моделі, 6) розгортання моделі, 7) обслуговування моделі, 8) моніторинг (контроль роботи) моделі та 9) вдосконалення моделі.

На рис. 2.2. область інженерії машинного навчання обмежена синьою зоною. Суцільні стрілки показують типовий перебіг етапів проекту. Пунктирні стрілки вказують на те, що на деяких етапах можна прийняти рішення повернутися до процесу і або зібрати більше даних, або зібрати інші дані та переглянути ознаки і моделі (шляхом виведення з експлуатації деяких з них і створення нових).

У більш простому вигляді, коли вирішується легша задача, створюється конвеєр машинного навчання (machine learning pipeline)

Основні кроки в конвеєрі машинного навчання (рис. 2.3):

1. Визначення проблеми: визначення бізнес-проблеми.
2. Збирання даних: ідентифікуйте та зберіть набір даних.
3. Підготовка даних: обробляйте та підготуйте дані, використовуючи такі

методи, як:

- Введіть відсутні значення
 - Видаліть повторювані записи
 - Нормалізуйте значення (змінить числові значення в наборі даних для використання загальної шкали)
 - Виконайте інший тип очищення або зіставлення
 - Повне визначення ознак
 - Виключіть корельовані ознаки
 - Виконайте розробку функцій
4. Розділення даних: розділіть дані на навчальний набір, набір для перевірки та набір для тестування.
5. Навчання моделі: навчання моделі машин на основі навчального набору даних. Це ядро науки про дані. У цьому розділі ми лише розглянемо поверхню цього кроку та наступних кроків. Є й інші розділи, які більш детально охоплюють навчання моделей. Усе наведене переважно для того, щоб дати читачеві повне уявлення про повний конвеєр.

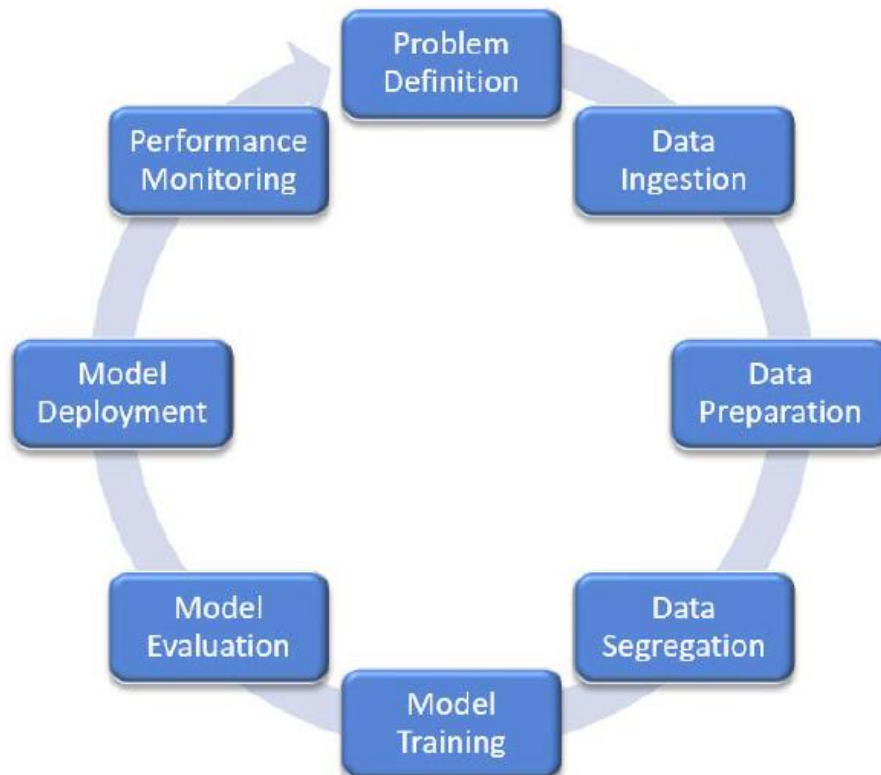


Рис.2.3 Конвеєр машинного навчання

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

6. Оцінка моделі кандидата: Виміряйте продуктивність моделей, використовуючи тестові та перевірочні підмножини даних, щоб визначити точність моделі.

7. Розгортання моделі. Після того, як модель вибрано, розгорніть її у виробництві для висновку.

8. Моніторинг продуктивності. Постійно відстежуйте продуктивність моделі, перенавчайте та відповідним чином калібруйте. Збирайте нові дані, щоб продовжувати вдосконалювати модель і запобігати її застарінню:

2.3 Аналіз вхідних даних та їх попередня обробка

Головний етап перед навчанням машини – підбір даних. На цьому етапі формується набір даних, параметрів та значень. Як правило, при обробці ми маємо справу з великими обсягами необроблених вихідних даних. Алгоритми машинного навчання розраховані на те, що, перш ніж вони зможуть розпочати процес тренування, отримані дані будуть відформатовані певним чином.

Щоб привести дані до форми, що прийнятна для алгоритмів машинного навчання, ми повинні попередньо підготувати їх і перетворити на потрібний формат. Було обрано набір даних у вигляді CSV файлу з інтернет трафіком умовної мережі. Серед цього трафіку є нормальні пакети даних та шкідливі, задані типи протоколів, прапорці та ін. Файл налічує 494 тисячі рядків потенційно шкідливих даних, на яких машина і буде навчатися. Кожний рядок розділений за своїми параметрами та зчитується напряду програмою з файлу (рис.2.4).

Слід перелічити значення усіх стовпців для розуміння даних файлу:

- duration: continuous – тривалість передачі даних;
- protocol_type: symbolic – тип протоколу;
- service: symbolic – сервіс передачі даних до отримувача;
- flag: symbolic – прапорець пакету (нормальне чи невдале підключення);
- src_bytes: continuous – кількість переданих байтів від джерела;

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

- num_outbound_cmds: continuous – кількість зовнішніх команд за ftp сесію;
- is_host_login: symbolic – перевірка логіну як хоста;
- is_guest_login: symbolic – перевірка логіну як гостя;
- count: continuous – кількість підключень до того самого хоста, що й поточне з'єднання за останні дві секунди.

Наведені нижче функції стосуються тільки підключень до одного й того самого хосту:

- srv_count: continuous – кількість підключень до того самого сервісу, що й поточне з'єднання за останні дві секунди;
- error_rate: continuous – відсоток підключень з помилками “SYN”;
- srv_error_rate: continuous – відсоток підключень з помилками “SYN”;
- error_rate: continuous – відсоток підключень з помилками “REJ”;
- srv_error_rate: continuous – відсоток підключень з помилками “REJ”;
- same_srv_rate: continuous – відсоток підключень до одного й того самого сервісу;
- diff_srv_rate: continuous – відсоток підключень до різних сервісів;
- srv_diff_host_rate: continuous – відсоток підключень до різних хостів;
- dst_host_count: continuous – кількість підключень до хостів;
- dst_host_srv_count: continuous – кількість підключень до сервісів хоста;
- dst_host_same_srv_rate: continuous – відсоток підключень до одного й того самого сервісу хоста;
- dst_host_diff_srv_rate: continuous – відсоток підключень до різних сервісів хоста;
- dst_host_same_src_port_rate: continuous – відсоток однакових підключених портів до хоста;
- dst_host_srv_diff_host_rate: continuous – відсоток підключень різних портів до хоста;
- dst_host_error_rate: continuous – відсоток помилок “SYN” до хоста;

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

- dst_host_srv_error_rate: continuous – відсоток помилок “SYN” до хоста;
- dst_host_error_rate: continuous – відсоток помилок “RAJ” до хоста;
- dst_host_srv_error_rate: continuous – відсоток помилок “RAJ” до хоста.

Ми не можемо підібрати та оцінити алгоритми машинного навчання на не-оброблених даних; натомість слід перетворити дані, щоб вони відповідали вимогам окремих алгоритмів машинного навчання. Більше того, ми повинні вибрати представлення даних, яке найкраще розкриває невідому базову структуру проблеми прогнозування для алгоритмів навчання, щоб отримати найкращу продуктивність з огляду на наші доступні ресурси у проекті прогнозного моделювання. Тому наступним кроком буде обробка вхідних даних.

Розглянемо програмну інтерпретацію набору даних, обробку цих даних та операції над ними. В першу чергу програма зчитує дані з CSV файлу за допомогою бібліотеки “pandas”.

Наступним кроком обробки буде заміна значень стовпців, а саме всі атаки ми заміняємо словом “attack”. Таким чином наша модель буде використовувати бінарну класифікацію, де значення «normal» – 0, тобто звичайне з’єднання, та «attacks» – 1, тобто виявлено атаку (рис. 2.5).

```
dataset = pd.read_csv(Path().joinpath('data', 'traffic_data_10_percent_corrected.csv'))

dataset['normal.'] = dataset['normal.'].replace(
    ['back.', 'buffer_overflow.', 'ftp_write.', 'guess_passwd.', 'imap.', 'ipsweep.', 'land.', 'loadmodule.',
     'multihop.', 'neptune.', 'nmap.', 'perl.', 'phf.', 'pod.', 'portsweep.', 'rootkit.', 'satan.', 'smurf.', 'spy.',
     'teardrop', 'warezclient.', 'warezmaster.'], 'attack')

x = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 41].values

new_data = pd.DataFrame(x)
new_data.columns = col_names
```

Рис.2.5 Обробка набору даних у програмному вигляді

Далі приводимо назви колонок екземплярів для зручості читання, за допомогою аналізу за методом словника: порівнюємо текстові значення у відповідному стовпці з назвами у словнику та замінюємо за потребою.

На рисунку 2.6 показано назви колонок, а на рисунку 2.7 виведено результат співставлення колонок і стовпців.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

```
# Load the dataset
col_names = [
    'duration',
    'protocol_type',
    'service',
    'flag',
    'src_bytes',
    'dst_bytes',
    'land',
    'wrong_fragment',
    'urgent',
    'hot',
    'num_failed_logins',
    'logged_in',
    'num_compromised',
    'root_shell',
    'su_attempted',
    'num_root',
    'num_file_creations',
    'num_shells',
    'num_access_files',
    'num_outbound_cmds',
    'is_host_login',
    'is_guest_login',
```

Рис.2.6 Фрагмент запису найменувань колонок для відображення

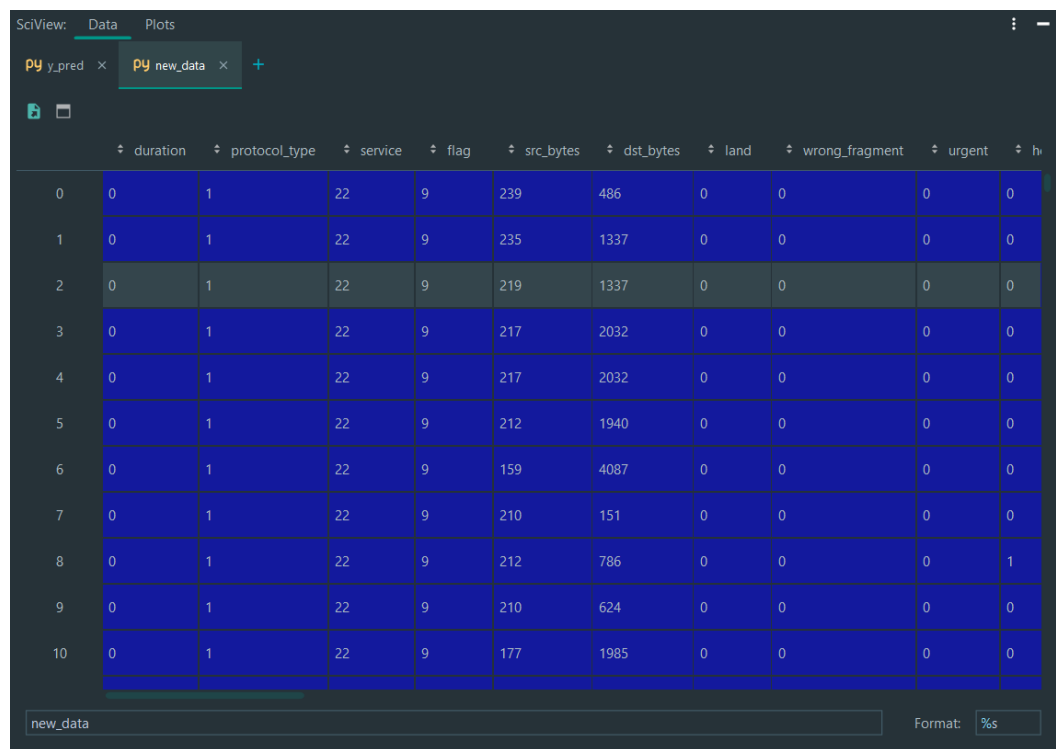


Рис.2.7 Відображення фрагменту таблиці даних після впорядкування

Далі слід категоризувати дані, підготувати свої необроблені дані, використовуючи перетворення даних, перш ніж підганяти модель машинного навчання.

Це потрібно для того, щоб ми найкращим чином представили структуру нашої проблеми прогнозного моделювання для алгоритмів навчання.

Застосування перетворень даних, таких як масштабування або кодування категоріальних змінних, є простим, якщо всі вхідні змінні мають один і той самий тип. Це може бути непросто, якщо у нас є набір даних зі змішаними типами і ви хочете вибірково застосувати перетворення даних до деяких, але не до всіх вхідних об'єктів. Така проблема присутня у нашому наборі даних. Але бібліотека машинного навчання Python scikit-learn надає метод ColumnTransformer, який дозволяє вибірково застосовувати перетворення даних до різних стовпців у нашому наборі даних, що стає нам у нагоді [22].

Отже, скориставшись цим методом перетворимо наші дані в категоріальні, щоб уникнути непотрібних значень, та залучити лише необхідні (рис. 2.8).

```
# encoding categorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer

labelencoder_x_1 = LabelEncoder()
labelencoder_x_2 = LabelEncoder()
labelencoder_x_3 = LabelEncoder()
x[:, 1] = labelencoder_x_1.fit_transform(x[:, 1])
x[:, 2] = labelencoder_x_2.fit_transform(x[:, 2])
x[:, 3] = labelencoder_x_3.fit_transform(x[:, 3])
onehotencoder_1 = ColumnTransformer([('oh_enc', OneHotEncoder(), [1])], remainder='passthrough')
x = onehotencoder_1.fit_transform(x)
onehotencoder_2 = ColumnTransformer([('oh_enc', OneHotEncoder(), [4])], remainder='passthrough')
x = onehotencoder_2.fit_transform(x)
onehotencoder_3 = ColumnTransformer([('oh_enc', OneHotEncoder(), [70])], remainder='passthrough')
x = onehotencoder_3.fit_transform(x)
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)
```

Рис.2.8 Категоризація даних

Після першого перетворення можемо побачити, що наші категоріальні дані закодовані. Якщо поглянути на другий стовпець, то тип протоколу тепер не текстове значення (рис.2.9).

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

	0	1	2	3	4	5	6	7
0	0.0	1.0	0.0	0	22	9	239	486
1	0.0	1.0	0.0	0	22	9	235	1337
2	0.0	1.0	0.0	0	22	9	219	1337
3	0.0	1.0	0.0	0	22	9	217	2032
4	0.0	1.0	0.0	0	22	9	217	2032
5	0.0	1.0	0.0	0	22	9	212	1940

Рис.2.9 Категоризація даних

Перед початком роботи з алгоритмом залишається лише два етапи: розділення набору даних на навчальні та тестувальні та нормалізація даних.

Розподілимо набір даних за співвідношенням 30 до 70, де 30% – дані для тестування, та, відповідно, 70% – для навчання (рис. 2.10).

```
# splitting the dataset into the training set and test set
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
```

Рис.2.10 Розподілення даних

Після цього слід нормалізувати дані, будемо це робити за допомогою тієї ж бібліотеки “sklearn” та функції StandardScaler. Слід розглянути методи нормалізації перед застосуванням у програмному вигляді [23]:

Нормалізація за абсолютним максимумом. За цим методом кожний елемент ділять на свій абсолютний максимум, щоб отримати результат між -1 та 1.

$$x_{scaled} = \frac{x}{\max(|x|)} \quad (2.1)$$

Нормалізація за методом мінімаксингу. Для отримання результату, кожний елемент віднімають від мінімального та розділяють на різницю максимального та мінімального елементів. Таким чином відповідь обчислення потрапляє у відрізок між 0 та 1.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (2.2)$$

Нормалізація за методом Z-оцінки. Цей метод також часто називають стандартизацією. В його основі лежить віднімання від елементу середнього значення відповідного йому та розділення на значення стандартного відхилення. Зазвичай, у цього метода немає чітких границь, проте в 99% даних відповідь лежить у множині від -3 до 3.

$$x_{std} = \frac{x - \mu}{\sigma} \quad (2.3)$$

Robust scaling метод. Основна ідея цього методу полягає у відніманні медіани та діленні на міжквартильний розмах. Міжквартильний розмах – це різниця між третім та першим квартилем, що являє собою центральні 50% даних, де $Q_1(x)$ – перший квартиль, $Q_2(x)$ – медіана, а $Q_3(x)$ – третій квартиль.

$$x_{rs} = \frac{x_i - Q_2(x)}{Q_3(x) - Q_1(x)} \quad (2.4)$$

Ми використовуємо функцію StandardScaler, котра базується на методі Z-оцінки. Ця функція обраховує середнє відхилення популяції, де сума квадратів ділиться на N (кількість значень у популяції).

$$\sigma = \sqrt{\frac{\sum_{i=1}^N (y_i - \mu)^2}{N}} \quad (2.5)$$

Після створення об'єкту StandardScaler обчислюються параметри трансформації, використовуючи метод .fit(). Потім викликається метод .transform(), щоб застосувати стандартизацію до фрейму даних. Метод .transform використовує параметри згенеровані у методі .fit(). Для спрощення цих операцій було використано метод .fit_transform(), котрий об'єднує два попередні методи в один. Щодо для тестувальних даних – використовуємо метод .transform() (рис.2.11) [23].

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

```

from sklearn.preprocessing import StandardScaler

sc_x = StandardScaler()
x_train = sc_x.fit_transform(x_train)
x_test = sc_x.transform(x_test)

```

Рис.2.11 Нормалізація даних

Після нормалізації дані були приведені до єдиного вигляду, як ми можемо побачити на рисунку 2.12:

	÷ 0	÷ 1	÷ 2	÷ 3	÷ 4	÷ 5	÷ 6	÷ 7	÷ 8	÷ 9	÷ 10
0	-0.00380	-0.24042	-0.03325	-0.00510	-0.04313	-0.46246	-0.01102	-0.00741	-0.00450	0.55308	-0.01453
1	-0.00380	-0.24042	-0.03325	-0.00510	-0.04313	-0.46246	-0.01102	-0.00741	-0.00450	0.55308	-0.01453
2	-0.00380	-0.24042	-0.03325	-0.00510	-0.04313	-0.46246	-0.01102	-0.00741	-0.00450	0.55308	-0.01453
3	-0.00380	-0.24042	-0.03325	-0.00510	-0.04313	2.16233	-0.01102	-0.00741	-0.00450	-1.80805	-0.01453
4	-0.00380	-0.24042	-0.03325	-0.00510	-0.04313	2.16233	-0.01102	-0.00741	-0.00450	-1.80805	-0.01453
5	-0.00380	4.15932	-0.03325	-0.00510	-0.04313	-0.46246	-0.01102	-0.00741	-0.00450	-1.80805	-0.01453
6	-0.00380	-0.24042	-0.03325	-0.00510	-0.04313	-0.46246	-0.01102	-0.00741	-0.00450	0.55308	-0.01453
7	-0.00380	4.15932	-0.03325	-0.00510	-0.04313	-0.46246	-0.01102	-0.00741	-0.00450	-1.80805	-0.01453

Рис.2.12 Нормалізовані дані

2.4 Розробка програмного забезпечення детектора атак на основі алгоритмів дерев рішень

Після підготовки вхідних даних, їх розбиття на навчальні та тестувальні та нормалізації, можна розпочати етап опрацювання та навчання машини. Нами було обрано алгоритм дерева рішень, адже він показує найбільшу точність серед наведених раніше алгоритмів неглибокого навчання. Також цей алгоритм є досить універсальним, та використовується у рішеннях статистики, аналізу даних, та, звичайно, машинного навчання. Модель буде навчатись за заздалегідь переданими даними, а саме за навчальним набором, що складає 30% від усього набору. Далі буде проводитись аналіз, пошук закономірностей та прогнозування результати. Після цього проводиться тестування на коректність прогнозів.

Імплементація дерева рішень буде проводитись за допомогою бібліотеки sklearn, котра має у собі широкий набір рішень для машинного навчання (рис. 2.13).

```
# Fitting Decision Tree to the Training set
from sklearn import tree

# fit model no training data
classifier = tree.DecisionTreeClassifier()
classifier.fit(x_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(x_test)
```

Рис.2.13 Імпорт алгоритму дерева рішень та піднесення даних

За допомогою цієї функції буде здійснюватися основна робота дерева рішень, весь аналіз та прогнозування.

Клас DecisionTreeClassifier() – це клас, який може проводити багатокласову класифікацію набору даних. Як і інші класифікатори, DecisionTreeClassifier() бере на вході два масиви: масив X, в котрому знаходяться навчальні зразки даних, та масив Y, в котрому знаходяться класові лейбли для цих навчальних даних. У випадку, коли існує декілька класів з однаково високою ймовірністю, класифікатор спрогнозує тей клас, котрий має найменший індекс серед усіх. За альтернативою, замість виведення відповіді у вигляді конкретного класу, можна спрогнозувати ймовірності кожного класу, тобто частку від навчальних даних в листовому вузлі. Як вже було згадано, цей класифікатор може працювати у бінарній (де лейбли підпадають у множину $[-1;1]$) та багатокласовій (де лейбли у множині $[0, \dots, K-1]$) класифікації. За цим проводиться прогнозування даних у тестовому наборі за допомогою класу predict() [24].

Далі створимо матрицю невідповідностей, котра допоможе нам наочно переглянути точність алгоритму (рис. 2.14). В даній матриці кожний рядок являє собою зразок прогнозованого класу, а стовпець – зразок справжнього, або ж навпаки. Матриця допомагає бачити невідповідності між цими класами, напри-

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

клад, коли алгоритм помічає прогнзований клас як справжній. Окрім цього можна бачити не лише помилки прогнозів матриці, а й природу цих помилок, їх типи та походження. За алгоритмом обрахування матриці, кількість вірних та невірних прогнозів суммується з кількістю справжніх величин та розділяється на кожний клас.

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
```

Рис.2.14 Створення матриці невідповідностей

У бібліотеці sklearn матриця невідповідностей є окремим класом, та позначається командою `confusion_matrix()`. Діагональні елементи матриці представляють кількість точок, для яких передбачена мітка дорівнює істинній, тоді як недіагональні елементи – це ті, які були помилково позначені класифікатором. Чим вищі показники діагоналі матриці, тим краще, що вказує на велику кількість вірних рішень [25].

Після матриці невідповідностей перевіримо результати прогнозування машини за допомогою k-вкладеного перехресного затвердження (k-fold cross-validation). Перехресне затвердження – це методика, яка оцінює модель та її прогнози, щоб зрозуміти наскільки точно модель працюватиме на практиці. Метод є досить популярним, тому що його загальні результати зазвичай менш упереджені чи оптимістичні, у порівнянні з іншими. Основною метою є випробування здатностей моделі передбачувати нові дані, які не використовувалися при її визначенні, щоб простежити проблеми перенавчання та вибіркового упередження, і щоб зрозуміти як ця модель буде узагальнюватись на незалежному наборі даних (тобто невідомому для неї). В цьому методі є єдиний параметр k, котрий позначає кількість груп, на які буде поділено набір даних. Коли значення k обрано, його можливо замінити на обране значення, що позначатиме кількість вкладень у затвердженні. Загальний процес роботи методу такий [26]:

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

1. Випадково перемішати набір даних.
2. Розділити набір на k груп.
3. Для кожної унікальної групи:
 - Прийняти групу як тестувальні дані.
 - Всі інші групи прийняти за навчальні дані.
 - Налаштувати модель на навчальних даних та оцінити на тестових.
 - Отримати оцінку та видалити модель.
4. Підсумувати навички моделі, використовуючи отримані оцінки.

Важливо, що кожне спостереження у вибірці даних призначається індивідуальній групі, та залишається у цій групі протягом усієї процедури. Це означає, що кожний зразок може бути використаний у наборі лише 1 раз, та використаний у навчанні моделі $k - 1$ разів. Також важливо, що будь-яка підготовка даних перед підгонкою моделі проводилася на навчальному наборі затвердження, замість основного набору. Це також відноситься до будь-яких налаштувань гіперпараметрів, адже помилка у впровадженні даних операцій може призвести до витoku даних та оптимістичних оцінок навичок моделі. Результати k -вкладеного перехресного затвердження часто підсумовуються з середнім показником навичок моделі. Також, гарною практикою вважається додання величини дисперсії оцінок навичок, наприклад стандартного відхилення чи стандартної помилки.

Щодо визначення величини k , вона має бути обрана з високою точністю до набору даних. Погано обрана величина k може призвести до неправильно представленої ідеї навичок моделі, наприклад, до оцінки з високою дисперсією, яка може сильно змінюватись на основі даних, що використовуються, або до високої упередженості, що виллється у переоцінку навичок моделі. Або ж, якщо обране значення k не поділяє набір даних рівномірно, то одна з груп міститиме решту прикладів. Краще розділяти набори даних на кількість k груп однакового розміру, щоб вибірка оцінок навичок моделі була еквівалентною.

Існує три розповсюджених методи визначення величини k :

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

- Репрезентативний. Величина k обирається таким чином, щоб кожна навчальна/тестувальна група була достатньо великою, щоб стати статистично наглядною для більшого набору даних.

- Метод $k = 10$. За цим неформальним методом значення k фіксується на 10. За експериментами фахівців, це значення, як правило, призводить до оцінки навичок моделі з низьким зміщенням та помірною дисперсією. Воно дуже погигрене в області машинного навчання у випадках, коли важко оцінити та обрати значення для вашого набору даних.

- Метод $k = n$. Значення k фіксується до n , де n – це розмір набору даних, таким чином з’являється можливість для кожного тестового зразка даних використовуватись в отриманому наборі даних. Цей підхід ще називають перехресним затвердженням без виключення (leave-one-out cross-validation).

```
# Applying k-Fold Cross Validation
from sklearn.model_selection import cross_val_score

accuracies = cross_val_score(estimator=classifier, X=x_train, y=y_train,
                              cv=10)

# %%
print("CV Accuracy: %0.4f (+/- %0.4f)" % (accuracies.mean(), accuracies.
                                         std() * 2))
```

Рис.2.15 Застосування k -вкладеного перехресного затвердження для оцінки моделі

Існують також й інші методи затверджування, такі як:

- Розподіл на навчання/тестування. Величина k обирається від екстримального значення, наприклад 2 (але ніколи не 1), так, щоб було створено один навчальний/тестувальний набір даних для оцінки моделі.

- LOOCV. Також обирається екстримальне значення k за кількістю спостережень у наборі даних, щоб кожне спостереження мало шанс залишитись поза набором даних.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

- Стратифікований. Розподіл даних проводиться таким чином, щоб кожне вкладення мало однакові пропорції та кількість спостережень з категоріальною величиною.

- Повторюваний. За цим методом процедура k-вкладеного перехресного затвердження повторюється n разів, та набір даних перемішується перед кожним повторенням, що призводить до іншого розділення набору.

- Вкладений. В цьому методі виконується k-вкладене перехресне затвердження в кожній вкладинці перехресного затвердження. Частіше всього задля налаштування гіперпараметрів під час оцінки моделі. Цей метод також називають подвійним перехресним вкладенням [27].

Після основних алгоритмів опрацювання даних та валідації, виведемо усі показники, матрицю невідповідностей та ROC/AUC криву задля розуміння ефективності навчання машини (рис. 2.16–2.17).

```
# the performance of the classification model
print("Test Accuracy is : %0.4f" % ((cm[0, 0] + cm[1, 1]) /
                                     (cm[0, 0] + cm[0, 1] + cm[1, 0] + cm[1, 1])))
recall = cm[1, 1] / (cm[0, 1] + cm[1, 1])
print("Recall is : %0.4f" % (recall))
print("False Positive rate: %0.4f" % (cm[1, 0] / (cm[0, 0] + cm[1, 0])))
precision = cm[1, 1] / (cm[1, 0] + cm[1, 1])
print("Precision is: %0.4f" % (precision))
print("F-measure is: %0.4f" % (2 * ((precision * recall) /
                                     (precision + recall))))

from math import log

print("Entropy is: %0.4f" % (-precision * log(precision)))
print("\nConfusion Matrix:\n", cm)

# Building the confusion matrix visual
ax = sns.heatmap(cm / np.sum(cm), annot=True,
                  fmt='.2%', cmap='Blues')
ax.set_title('Percentage Confusion Matrix\n')
ax.set_xlabel('Predicted Category')
ax.set_ylabel('Actual Category ')
sns.set(rc={'figure.figsize': (10, 7)})
plt.show()
```

Рис.2.16 Виведення результатів ефективності навчання моделі

```
# Building the AUC/ROC Curve

from sklearn.metrics import roc_curve, roc_auc_score

false_positive_rate1, true_positive_rate1, threshold1 = roc_curve(y_test, y_pred)
print('Roc/Auc Score for DecisionTree: ', roc_auc_score(y_test, y_pred))
plt.subplots(1, figsize=(10, 10))
plt.title('Receiver Operating Characteristic - DecisionTree')
plt.plot(false_positive_rate1, true_positive_rate1)
plt.plot([0, 1], ls=":", linewidth="8", color="lightblue")
plt.plot([0, 0], [1, 0], color="lightcoral"), plt.plot([1, 1], color="lightcoral")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.savefig("ROC.png")
plt.show()
```

Рис.2.17 Побудова ROC/AUC кривої

Повний лістинг програмного коду надано у [додатку А](#).

Висновки до 2-го розділу:

В результаті пошуку мов програмування, що підійдуть для поставлених задач, було проведено порівняльний аналіз та обрано цільову мову, а саме: Python. Після цього проаналізовано мову Python та її підхід до машинного навчання, досліджено принцип побудови конвеєру машинного навчання. Досліджено бібліотеки та додатки для машинного навчання, роботи з даними та графічною інтерпретацією. Також було розібрано методики обрахування, що відбуваються під час навчання та прогнозування машиною відповідей.

Після теоретичного дослідження було обрано набір даних для опрацювання. Досліджено його вміст, проаналізовано поля та стовпці, підготовано для роботи.

Розроблено програмний продукт машинного навчання детектору кібератак, наведено та розібрано усі етапи розробки. Також розроблено методи графічного відображення результатів успішності, точності навчання та прогнозування машини.

РОЗДІЛ 3 ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ ТА ЕФЕКТИВНОСТІ ДЕТЕКТОРА АТАК НА ОСНОВІ АЛГОРИТМІВ ДЕРЕВ РІШЕНЬ

3.1 Показники якості класифікації при виявленні атак

Результатом більшості задач машинного навчання є метрики, котрі використовуються для розуміння точності роботи, прогнозування та в цілому поведінки машини. Ці метрики можуть бути різними для різних типів задач, проте у всіх є обов'язковий набір показників, котрі й відображають усю ефективність роботи машини. Серед цих метрик існують такі:

- Точність (Accuracy). Під точністю класифікації ми, зазвичай, маємо на увазі термін відношення кількості правильних прогнозів до загальної кількості вхідних вибірок.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \quad (3.1)$$

У випадках бінарної класифікації, точність можна обрахувати за обчисленням позитивних та негативних показників:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \quad (3.2)$$

де TP = істинно-позитивні, TN = істинно-негативні, FP = хибно-позитивні і FN = хибно-негативні.

Проте слід приділяти велику увагу до наборів даних та їх рівноцінності. Наприклад, в навчальному наборі є 98% зразків класу А та 2% зразків класу В. Тоді наша модель може легко отримати 98% точності навчання, передбачивши кожную вибірку в класі А. Проте, коли буде проводитись тестування у наборі, де 60% зразків класу А, та 40% зразків класу В, точність вже впаде до 60%. Таким чином складається хибне уявлення про точність класифікації та моделі, що може призвести до фатальних помилок. У сфері безпеки така помилка може коштувати як мільйонів долларів, так і мільйонів життів.

- Влучність (Precision). Влучність має на меті відповідати на питання «Яка кількість позитивних відповідей є дійсно коректною?».

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3.3)$$

Влучність допомагає, коли існує велика вартість хибно-позитивних результатів. Наприклад, проблема пов'язана з несправним біометричним валідатором. Коли у моделі низька влучність, то багатьом буде дозволено доступ, хоча так бути не повинно, і такий результат буде мати багато помилкових рішень. Це може призвести до великих втрат та несанкціонованого доступу до різного класу приміщень.

- Повнота (Recall). Повнота ж відповідає на питання «Яка кількість позитивних відповідей була коректно визначена?».

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3.4)$$

Дана метрика буде у нагоді, коли існує велика кількість хибно-негативних результатів. Наприклад, якщо потрібно розпізнати надходячу ракету, хибно-негативний результат призведе до руйнівних наслідків. Коли в моделі велика кількість хибно-негативних результатів – це головне, чого вам треба позбутись.

- F1 Оцінка (F1 Score). Оцінка F1 – це загальна величина двох попередніх показників: влучності та повноти.

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (3.5)$$

Хороший результат Оцінки F1 означає, що у моделі низький рівень помилково-позитивних та помилково-негативних рішень, тобто, вона правильно визначає реальні загрози та не сповіщає за хибними. Оцінка F1 вважається ідеальною, коли вона дорівнює 1, тоді як 0 – найгірший результат.

- Матриця невідповідностей. Як вже було зазначено у попередньому розділі, матриця невідповідностей – це N x N матриця, де N – кількість класів, що прогнозується. Матриця порівнює фактичні цільові значення із передбаченими моделлю машинного навчання. Це дає нам цілісне уявлення про те, наскільки добре працює наша модель класифікації і яких помилок вона допускає. Оскільки у нас бінарна класифікація, то наша матриця буде розміром 2 x 2. Цільова змінна

має два значення: позитивне чи негативне. Стовпці представляють фактичні значення цільової змінної. Рядки представляють передбачені значення цільової змінної (рис. 3.1) [28].

Confusion Matrix		Target			
		Positive	Negative		
Model	Positive	a	b	Positive Predictive Value	$a/(a+b)$
	Negative	c	d	Negative Predictive Value	$d/(c+d)$
		Sensitivity	Specificity	Accuracy = $(a+d)/(a+b+c+d)$	
		$a/(a+c)$	$d/(b+d)$		

Рис.3.1 Приклад обрахувань бінарної матриці невідповідностей

- AUC-ROC крива. ROC (Receiver Operator Characteristic) крива є метрикою оцінки для задач бінарної класифікації. Це крива ймовірностей, котра співставляє істино-позитивні відповіді з хибно-позитивними на різних порогових значеннях, та відокремлює «сигнал» від «шуму». AUC (Area Under the Curve) – це міра здатності класифікатора розрізняти класи, вона використовується як підсумок до кривої ROC. Чим більший показник AUC, тим краща здатність моделі відрізняти позитивні та негативні класи. Як і в F1 Оцінці, 1 означає найкращий результат, та 0 як найгірший. Модель з нулевим показником AUC буде показувати хибні результати як істинні, а істинні як хибні. Коли AUC лежить в межах від 0.5 до 1, це середній показник, при якому модель зможе коректно працювати, адже кількість вірних прогнозів буде більшою за кількість хибних. Коли ж $AUC = 0.5$, машина не зможе адекватно прогнозувати результати, вони будуть або випадковими на кожний клас, або однаковими для всіх. ROC крива загалом використовується у питаннях бінарних класифікацій, у багатокласових питаннях існують лише модифікації класифікаторів на заміну (OneVsAll та подібні).

- Ентропія визначається як випадковість або міра безладу в наборі інформації, що обробляє машина. Іншими словами, ентропія – це показник машинного навчання, який вимірює непередбачуваність або сторонні додатки у системі. Ентропія завжди лежить у межах між 0 та 1, проте, залежно від кількості класів цей показник может бути й більше 1 (рис. 3.2) [29].

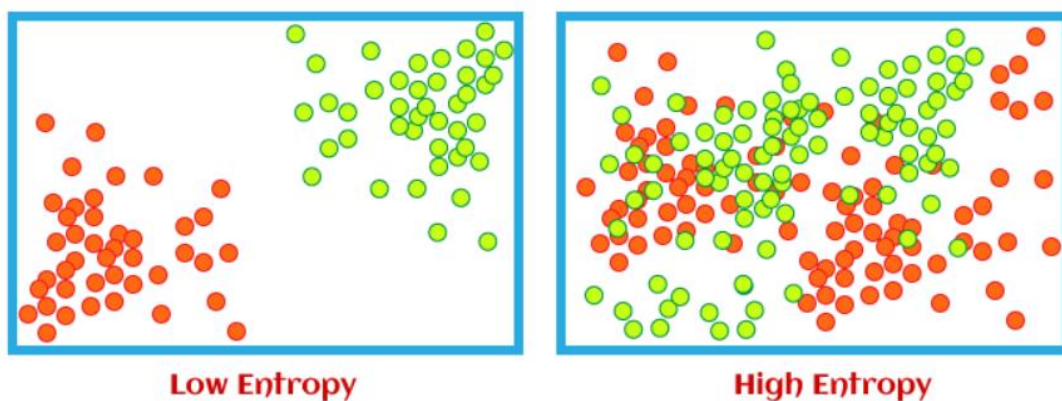


Рис.3.2 Приклад ентропії

Математично ентропія обраховується так:

$$E = - \sum_{i=1}^N P_i \log_2 P_i \quad (3.6)$$

де N – кількість класів у наборі даних;

P_i – ймовірність випадково обрати елемент у класі i .

3.2 Оцінка якості виявлення атак розробленим програмний забезпеченням

Після розробки моделі та розгляду усіх метрик, можна перейти до програмно-реалізованої оцінки якості моделі. У створеному програмному продукті вивід даних забезпечено до консолі, з відповідними позначками та підписами. Матриця невідповідностей подана як у графічному вигляді, так і у чисельному.

```
CV Accuracy: 0.9997 (+/- 0.0002)
Test Accuracy is : 0.9998
Recall is : 0.9995
False Positive rate: 0.0002
Precision is: 0.9993
F-measure is: 0.9994
Entropy is: 0.0007

Confusion Matrix:
[[118821    14]
 [    22 29349]]
AUC: 0.9995665757118547
```

Рис.3.3 Виведення метрик машини

За даними на рисунку 3.3 можна побачити основні критерії оцінки якості моделі машинного навчання: Точність, Повнота, Влучність, F1 Оцінка, Матриця неповності та площа AUC. Також врахована точність перехресного затвердження та Ентропія.

Точність перехресного затвердження допоможе зрозуміти наскільки добре модель адаптується до незалежних даних. За показниками, що ми бачимо на зображенні, модель має чудову здібність до адаптації.

Точність тестування дорівнює 0.9997, що вказує на майже стовідсоткову кількість правильних прогнозів. Враховуючи те, що модель навчалася на 30% даних, а тестування займало 70%, можна сказати, що влучність також не підведе.

Повнота дорівнює 0.9995, це кількість позитивних відповідей, що були визначені коректно, то ж, модель у переважній більшості випадків буде чітко прогнозувати ймовірні загрози.

FPR (False Positive Rate), або ж коефіцієнт хибно-позитивних відповідей. Це показник точності, котрий вказує на кількість хибно винесених прогнозів. Даний коефіцієнт дорівнює 0.0002, що вказує на вкрай малу вірогідність винесення хибно-позитивного результату.

Влучність складає 0.9993 з 1, цей результат є майже досконалим, що вже було заочно затверджено на основі точності. Слідуючи з цього, можна не перейматись через невірне винесення рішень після зміни набору даних чи перемішування елементів.

Оцінка F1 має величину 0.9994, та, як вже було вказано у попередньому пункті, така величина є майже безпомилковою для винесення рішень, та має низький рівень помилково-позитивних та помилково-негативних прогнозів.

Міра Ентропії дорівнює 0.0007, що вказує на неабияку впорядкованість даних та малу кількість випадкових рішень.

Щодо матриці невідповідностей, можемо бачити на її головній діагоналі велику кількість прогнозованих відповідей, тому можемо зробити висновки про якісне навчання та роботу машини.

Також розглянемо ROC криву та площу під нею AUC на рисунку 3.4.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

Крива AUC - ROC показує вимірювання продуктивності для завдань класифікації при різних порогових значеннях. ROC є кривою ймовірності, а AUC являє собою ступінь або міру роздільності. Він каже, як модель здатна розрізняти класи. Чим вище AUC, тим краще модель передбачає клас як 0 і клас 1 як 1.

Відмінна модель має AUC, близьку до 1, що означає добрий ступінь роздільності. Оскільки наш показник AUC дорівнює 0.99 це показує чіткість роботи нашої моделі.

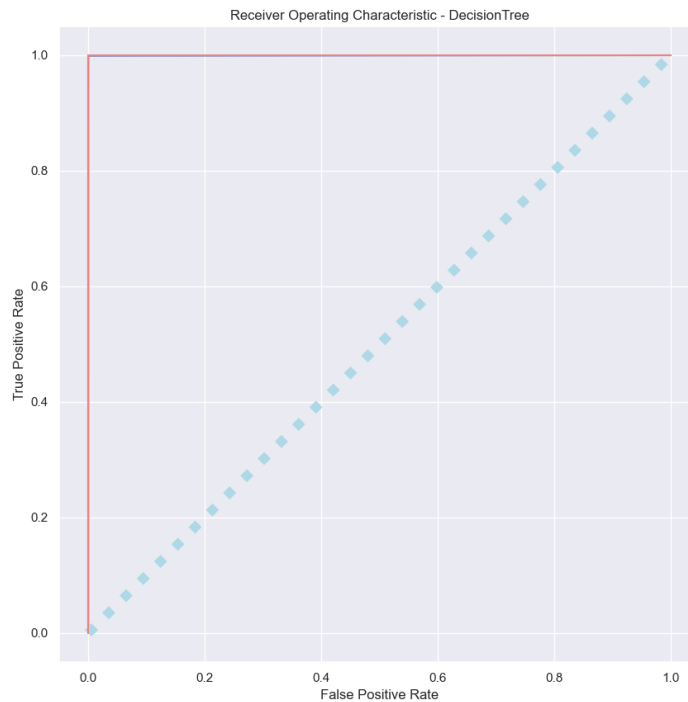


Рис.3.4 ROC крива

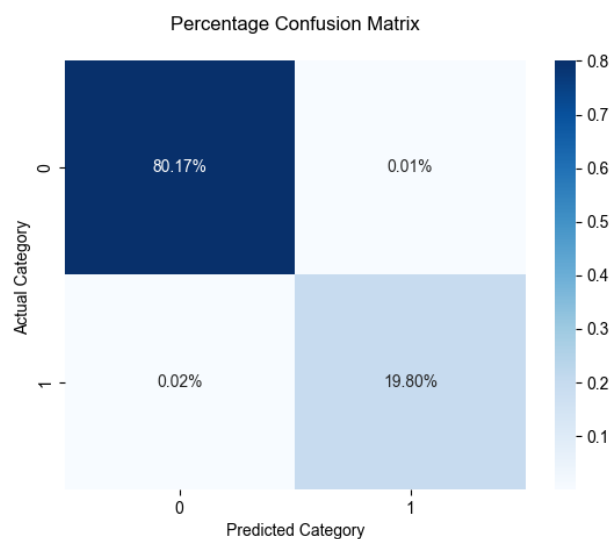


Рис.3.5 Матриця невідповідностей

На рисунку 3.6 ми можемо побачити спрогнозовані класи для виявлення атак.

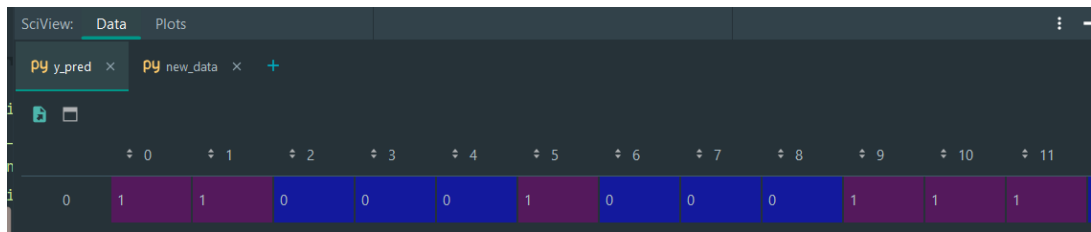


Рис.3.6 Приклад фрагменту прогнозів машини після навчання та тестування

За усіма вищевказаними метриками та дослідженнями можна зробити висновки про чудову якість виявлення атак програмним забезпеченням, його можливість перенавчатись та незалежно приймати рішення. З усіма поставленими для кваліфікаційної роботи задачами продукт впорається відмінно.

3.3 Рекомендації по застосуванню та вдосконаленню розробленого детектора атак на основі алгоритмів дерев рішень

У кваліфікаційній бакалаврській роботі розроблено програмний продукт машинного навчання для детектування атак на мережевому рівні. Дане програмне забезпечення слід використовувати за структурою, наданою у навчальному наборі даних та записаному в даному звіті. Програмний продукт може бути використаний як виконуваний (.exe) файл, або ж за допомогою програмних оболонок, таких як PyChart, задля змінення чи вдосконалення роботи машини, тестування та перегляду результатів. Зчитування відповідей оператором є досить зрозумілим за наданими скріншотами, тому в користувача не має виникнути питань щодо розуміння якості роботи машини.

Щодо вдосконалення методів роботи детектору атак, цей потенціал стрімко зростає з розвиненням машинних технологій. Першим, та, мабуть, найголовнішим змінням буде переведення машини на DeepLearning модель, в котрій застосовуються нейронні мережі. Під час глибокого навчання комп'ютерна модель вчиться виконувати завдання класифікації безпосередньо з зображень, тексту чи звуку. Моделі глибокого навчання можуть досягати надзвичайної точності, іноді пере-

вищуючи людські здібності. Більшість методів глибокого навчання використовують архітектури нейронних мереж, тому моделі глибокого навчання часто називають глибокими нейронними мережами. Використовуючи стандартний підхід машинного навчання, виникає можлива потреба власноруч обирати елементи чи класи, щоб навчити модель коректно. Завдяки ж робочому процесу глибокого навчання, відповідні функції виконуються автоматично. Крім того, моделі глибокого навчання використовують наскрізне навчання – коли мережа отримує необроблені дані та завдання для виконання, наприклад, задачу класифікації, та навчається вирішувати цю задачу автоматично. Ще одна ключова відмінність полягає в тому, що алгоритми глибокого навчання масштабуються за даними, тоді як машинне навчання зводиться, тобто перестає встигати за набором даних. Проте, глибоке навчання потребує великих обчислювальних потужностей та ще більшу кількість даних для обробки та навчання, що слід зауважити при переході до цього методу.

Крім зміни методу навчання також можна розглянути зміну сервісу, на якому можна побудувати дану модель. В сучасному світі все більше технічних пристроїв працюють на базі конструктора Microsoft Azure. Ця технологія має великі досягнення і в галузі машинного навчання, наприклад, у розпізнаванні людей чи машин, номерних знаків або предметів. Azure має зручну графічну оболонку у веб сервісі, в котрій наявні модулі та елементи для навчання, що суттєво зменшує потреби у навичках програмування. Даний сервіс цілком забезпечує усіма необхідними модулями та наборами даних для успішного та якісного навчання моделей за будь-якими алгоритмами. Крім того усі метрики та графіки, такі як ROC крива чи матриця невідповідностей, підключені автоматично до конструктора та будуть виводитись разом з перевіркою моделі. Єдиний, проте чималий, недолік Azure в тому, що усі послуги конструктора платні, та потребують «кредитів» для роботи, котрі можуть сягати сотень доларів.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

Висновки до 3-го розділу:

Було проаналізовано усі характеристики успішності навчання та тренування моделі, розібрано їх математичний вміст, досліджено додаткові метрики. Проведено перевірку успішності навчання, тренування та можливостей прогнозування моделі, проаналізовано рейтинги успішності.

Наведено розгорнуту відповідь щодо кожної метрики успішності, надано висновки за загальним показником. Додано графічні відображення точності прогнозування відповідей, у вигляді матриці неточностей та ROC кривої.

Надано рекомендації щодо застосування продукту та можливих покращень майбутніх версій. Проаналізовано можливі покращуючі технології та підходи.

		Погребніченко П.К			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

ВИСНОВКИ

В даній кваліфікаційній бакалаврській роботі було вирішено завдання з розробки програмного забезпечення детектору атак на основі мови програмування Python з застосуванням KDD Data Set.

Обґрунтування актуальності теми дало змогу оцінити вагу машинного навчання в окремих галузях людського життя, особливо у сфері безпеки та контролю.

Після проведення аналізу важливості та актуальності машинного навчання у галузі безпеки, було обрано методи неглибокого керованого машинного навчання, а саме метод класифікації, для вирішення задачі кваліфікаційної роботи. Для більш наочного розуміння вибору методу, було проведено аналіз інших методів навчання, також це допомогло з обранням алгоритму навчання моделі, таким чином було обрано алгоритм дерева прийняття рішень.

Для вирішення задачі детектору атак було обрано мову програмування Python та середовищу програмування PyCharm. Мова програмування Python має велику кількість бібліотек та розширень для машинного навчання та обробки даних, окрім того зручні бібліотеки для візуалізації даних та метрик, таких як матриця невідповідностей. Саме середовище розробки PyCharm має усі необхідні інструменти для розуміння коду, підтримки читабельності та простоти, також має зручну консоль для виводу результату роботи та аналізу.

Після ознайомлення з мовою програмування та середовищем розробки, було виділено етапи розробки, навчання та тестування моделі: аналіз вхідних даних, підготовка вхідних даних для навчання моделі, розробка моделі та її навчання, тестування моделі та її оцінка.

Програмний підхід до розробленої моделі включає в себе наступні кроки: розбиття набору даних на словник, визначення категоріальних даних та їх кодування, розбиття набору даних на навчальні та тестувальні, нормалізація даних, застосування алгоритму дерева рішень, прогнозування даних, створення матриці невідповідностей на базі прогнозів, застосування k-вкладеного перехресного затвердження, виведення метрик якості навчання моделі та візуальне відображення матриці невідповідностей.

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

Оцінка та аналіз моделі показали відмінні результати за обраним методом та алгоритмом, та, за аналізом у першому розділі, дерево рішень є кращим з алгоритмів неглибокого машинного навчання, тому можна зробити висновки про майже досконалу роботу машини.

Було розглянуто доцільність використання даної моделі у сфері безпеки. Основними перевагами є можливість швидко оброблювати великі обсяги даних, миттєво прогнозувати та аналізувати дані у реальному часі. Використання моделі детектору атак на основі Python надає змогу своєчасно помічати шкідливий трафік та реагувати на нього, що допоможе фахівцям з безпеки приймати рішення щодо дій у їх мережах.

		Погребніченко П.К			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Флах. П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных/ пер. с англ. А. А. Слинкина. - М.: ДМК Пресс, 2015. - 400 с.
2. Мерков А. Б. Распознавание образов. Введение в методы статистического обучения. 2011. 256 с.
3. Ed Burns. Machine learning. URL: <https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML>
4. Al Perlman. The Growing Role of Machine Learning in Cybersecurity, 2022. URL: <https://www.securityroundtable.org/the-growing-role-of-machine-learning-in-cybersecurity/>
5. Alyssa Schroer. 27 companies merging AI and cybersecurity to keep us safe and sound, 2022. URL: <https://builtin.com/artificial-intelligence/artificial-intelligence-cybersecurity>
6. Stephen Cooper. 10 AI Cybersecurity Firms to watch in 2022, 2021. URL: <https://www.comparitech.com/blog/information-security/leading-ai-cybersecurity-companies/>
7. Abi Tyas Tunggal. What is a Cyber Attack? Common Attack Techniques and Targets, 2022. URL: <https://www.upguard.com/blog/cyber-attack>
8. Machine Learning Enabled Attacks: What You Need to Know. URL: <https://prowritersins.com/cyber-insurance-blog/machine-learning-enabled-attacks/>
9. Iqbal H. Sarker, A. S. M. Kayes, Shahriar Badsha, Hamed Alqahtani, Paul Watters and Alex Ng. Cybersecurity data science: an overview from machine learning perspective, 2020. URL: <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00318-5>
10. Arne Wolfewicz. Deep Learning vs. Machine Learning – What’s The Difference? 2022. URL: <https://levity.ai/blog/difference-machine-learning-deep-learning>

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

11. Katrina Wakefield. A guide to the types of machine learning algorithms and their applications. URL: https://www.sas.com/en_gb/insights/articles/analytics/machine-learning-algorithms.html
12. Pethuru Raj, Preetha Evangeline. Advances in Computers. Machine learning and deep learning algorithms on the Industrial Internet of Things (IIoT), 2020, p. 321-338.
13. Archana Oberoi. Back to Basics: 5 Crucial Components of Machine Learning, 2020. ?. URL: <https://insights.daffodilsw.com/blog/back-to-basics-5-crucial-components-of-machine-learning>
14. James Shah. Six Elements Of Machine Learning - A Beginner's Guide, 2020. URL: <https://dev.to/jamesshah/six-elements-of-machine-learning-a-beginner-s-guide-4i35>
15. Anshul Saini. Decision Tree Algorithm – A Complete Guide, 2021. URL: <https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/>
16. Decision Tree Classification Algorithm. URL: <https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm>
17. A Distance-Based Attribute Selection Measure for Decision Tree Induction. URL: <https://link.springer.com/article/10.1023/A:1022694001379>
18. Nagesh Singh Chauhan. Decision Tree Algorithm, Explained. URL: <https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html>
19. Monomita Chakraborty. What are the best programming languages for artificial intelligence, 2021. URL: <https://www.analyticsinsight.net/what-are-the-best-programming-languages-for-artificial-intelligence/>
20. John Selawsky. Top five programming languages for AI and machine learning you should learn this year, 2022. URL: <https://www.itproportal.com/features/top-five-programming-languages-for-ai-and-machine-learning-you-should-learn-this-year/>
21. Angela Beklemysheva. Why Use Python for AI and Machine Learning? URL: <https://steelkiwi.com/blog/python-for-ai-and-machine-learning/>

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

22. Jason Brownlee. How to Use the ColumnTransformer for Data Preparation, 2019. . URL: <https://machinelearningmastery.com/columntransformer-for-numerical-and-categorical-data/>
23. Amanda Iglesias Moreno. Data normalization with Pandas and Scikit-Learn, 2020. URL: <https://towardsdatascience.com/data-normalization-with-pandas-and-scikit-learn-7c1cc6ed6475>
24. Philip Wilkinson. Introduction to decision tree classifiers from scikit-learn, 2020. URL: <https://towardsdatascience.com/introduction-to-decision-tree-classifiers-from-scikit-learn-32cd5d23f4d>
25. Samarth Agrawal. Understanding the Confusion Matrix from Scikit learn, 2021. URL: <https://towardsdatascience.com/understanding-the-confusion-matrix-from-scikit-learn-c51d88929c79>
26. Jason Brownlee. How to Configure k-Fold Cross-Validation, 2020. URL: <https://machinelearningmastery.com/how-to-configure-k-fold-cross-validation/>
27. Jason Brownlee. Nested Cross-Validation for Machine Learning with Python, 2020. URL: <https://machinelearningmastery.com/nested-cross-validation-for-machine-learning-with-python/>
28. Andrew Long. Understanding Data Science Classification Metrics in Scikit-Learn in Python, 2018. URL: <https://towardsdatascience.com/understanding-data-science-classification-metrics-in-scikit-learn-in-python-3bc336865019>
29. Evaluation Metrics With Python Codes, 2022. URL: <https://www.analyticsvidhya.com/blog/2022/01/evaluation-metrics-with-python-codes/>

		Погребніченко П.К			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				71
Змн.	Арк.	№ докум.	Підпис	Дата		



ДОДАТКИ

		Погребніченко П.К			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг програмного коду

```

from pathlib import Path
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
col_names = [
    'duration',
    'protocol_type',
    'service',
    'flag',
    'src_bytes',
    'dst_bytes',
    'land',
    'wrong_fragment',
    'urgent',
    'hot',
    'num_failed_logins',
    'logged_in',
    'num_compromised',
    'root_shell',
    'su_attempted',
    'num_root',
    'num_file_creations',
    'num_shells',
    'num_access_files',
    'num_outbound_cmds',
    'is_host_login',
    'is_guest_login',
    'count',
    'srv_count',
    'serror_rate',
    'srv_serror_rate',
    'rerror_rate',
    'srv_rerror_rate',
    'same_srv_rate',
    'diff_srv_rate',
    'srv_diff_host_rate',
    'dst_host_count',
    'dst_host_srv_count',
    'dst_host_same_srv_rate',
    'dst_host_diff_srv_rate',
    'dst_host_same_src_port_rate',
    'dst_host_srv_diff_host_rate',
    'dst_host_serror_rate',
    'dst_host_srv_serror_rate',
    'dst_host_rerror_rate',
    'dst_host_srv_rerror_rate'
]

dataset = pd.read_csv(Path().joinpath('data', 'traffic_data_10_percent_corrected.csv'))

dataset['normal.'] = dataset['normal.'].replace(
    ['back.', 'buffer_overflow.', 'ftp_write.', 'guess_passwd.', 'imap.',
    'ipsweep.', 'land.', 'loadmodule.',
    'multihop.', 'neptune.', 'nmap.', 'perl.', 'phf.', 'pod.', 'portsweep.',
    'rootkit.', 'satan.', 'smurf.', 'spy.',
    'teardrop.', 'warezclient.', 'warezmaster.'], 'attack')

```

		Погребніченко П.К			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

```

x = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 41].values

new_data = pd.DataFrame(x)
new_data.columns = col_names

# encoding categorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer

labelencoder_x_1 = LabelEncoder()
labelencoder_x_2 = LabelEncoder()
labelencoder_x_3 = LabelEncoder()
x[:, 1] = labelencoder_x_1.fit_transform(x[:, 1])
x[:, 2] = labelencoder_x_2.fit_transform(x[:, 2])
x[:, 3] = labelencoder_x_3.fit_transform(x[:, 3])
onehotencoder_1 = ColumnTransformer([('oh_enc', OneHotEncoder(), [1])], remainder='passthrough')
x = onehotencoder_1.fit_transform(x)
onehotencoder_2 = ColumnTransformer([('oh_enc', OneHotEncoder(), [4])], remainder='passthrough')
x = onehotencoder_2.fit_transform(x)
onehotencoder_3 = ColumnTransformer([('oh_enc', OneHotEncoder(), [70])], remainder='passthrough')
x = onehotencoder_3.fit_transform(x)
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)

# splitting the dataset into the training set and test set
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
# feature scaling
from sklearn.preprocessing import StandardScaler

sc_x = StandardScaler()
x_train = sc_x.fit_transform(x_train)
x_test = sc_x.transform(x_test)
# Fitting Decision Tree to the Training set
from sklearn import tree

# fit model no training data
classifier = tree.DecisionTreeClassifier()
classifier.fit(x_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(x_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

# Applying k-Fold Cross Validation
from sklearn.model_selection import cross_val_score

accuracies = cross_val_score(estimator=classifier, X=x_train, y=y_train, cv=10)

# %%
print("CV Accuracy: %0.4f (+/- %0.4f)" % (accuracies.mean(), accuracies.std() * 2))

# the performance of the classification model
print("Test Accuracy is : %0.4f" % ((cm[0, 0] + cm[1, 1]) / (cm[0, 0] + cm[0, 1] + cm[1, 0] + cm[1, 1])))

```

		Погребніченко П.К.			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

```

1]))))
recall = cm[1, 1] / (cm[0, 1] + cm[1, 1])
print("Recall is : %0.4f" % (recall))
print("False Positive rate: %0.4f" % (cm[1, 0] / (cm[0, 0] + cm[1, 0])))
precision = cm[1, 1] / (cm[1, 0] + cm[1, 1])
print("Precision is: %0.4f" % (precision))
print("F-measure is: %0.4f" % (2 * ((precision * recall) /
                                   (precision + recall))))

from math import log

print("Entropy is: %0.4f" % (-precision * log(precision)))
print("\nConfusion Matrix:\n", cm)

# Building the confusion matrix visual

ax = sns.heatmap(cm / np.sum(cm), annot=True,
                 fmt='.2%', cmap='Blues')
ax.set_title('Percentage Confusion Matrix\n')
ax.set_xlabel('Predicted Category')
ax.set_ylabel('Actual Category ')
sns.set(rc={'figure.figsize': (10, 7)})
plt.savefig("ConfusionMatrix.png")
plt.show()

# Building the AUC/ROC Curve

from sklearn.metrics import roc_curve, roc_auc_score

false_positive_rate1, true_positive_rate1, threshold1 = roc_curve(y_test,
y_pred)
print('Roc/Auc Score for DecisionTree: ', roc_auc_score(y_test, y_pred))
plt.subplots(1, figsize=(10, 10))
plt.title('Receiver Operating Characteristic - DecisionTree')
plt.plot(false_positive_rate1, true_positive_rate1)
plt.plot([0, 1], ls=":", linewidth="8", color="lightblue")
plt.plot([0, 0], [1, 0], color="lightcoral"), plt.plot([1, 1],
color="lightcoral")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.savefig("ROC.png")
plt.show()

```

		Погребніченко П.К			КБ.КР.Б. – 125 – 22 – ПЗ	Арк.
		Покотило О.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		75