

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:

«Алгоритми вивлення аномалій поведінки відвідувачів вебсайту»

Виконав студент групи ПЗм-23-1
КЛИМЕНКО Данило Олександрович

Керівник роботи:
ЧИЖМОТРИЯ Олексій Володимирович

Рецензент:
ЄФРЕМОВ Юрій Миколайович

Житомир – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»
Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«01» жовтня 2024 р

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КЛИМЕНКО Данило Олександрович**

Керівник роботи: **ЧИЖМОТРЕ Олексій Володимирович**

Тема роботи: **«Алгоритми виявлення аномалій поведінки відвідувачів вебсайтів»**

Затверджена Наказом закладу вищої освіти від «01» жовтня 2024 р., №520/с

Вихідні дані для роботи: дослідження веб-трафіку та виявлення характеристик аномальної поведінки; методи обробки, нормалізації та кластеризації даних для навчання; дослідження алгоритмів та підходів для створення моделей класифікації трафіку; метрики для оцінки ефективності моделей; технології та засоби для реалізації алгоритмів класифікації та машинного навчання.

Керівники / консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Керівник/консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.В.	01.10.2024	01.10.2024
2	Чижмотря О.В.	01.10.2024	01.10.2024
3	Чижмотря О.В.	01.10.2024	01.10.2024

РЕФЕРАТ

Випускна кваліфікаційна робота магістра складається з опису дослідження виявлення аномалій користувачів веб-сайтів, реалізації алгоритмів та різних підходів при створенні моделей машинного навчання з їх оцінкою, пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи складається з 82 сторінок, 56 ілюстрацій, 1 таблиці, 2 додатків та 40 літературних джерел.

Мета роботи – проведення дослідження веб-трафіку, визначення характеристик для класифікації активності з метою виявлення аномалій. Використання сучасних алгоритмів та підходів для проєктування, реалізації та оцінки моделей класифікації з використанням машинного навчання для ідентифікації різних типів запитів користувачів.

У роботі було описано цілі та поставлено задачі до досліджень та розробок. Було проаналізовано наявні алгоритми, їх особливості та визначальні характеристики. Як результат було обрано алгоритми Decision Tree, Random Forest, SVM, Voting, Stacking, LSTM, KNN, DNN. Було проведено глибокий аналіз структури веб-трафіку та визначено елементи, які впливають на аномалії. У процесі проєктування було детально проаналізовано архітектури обраних алгоритмів. Було обрано та досліджено набір даних, проведено всі необхідні з ним маніпуляції. Було реалізовано алгоритми, проведено оцінку та порівняння.

У висновку було отримано результати досліджень та розробок за час виконання кваліфікаційної роботи.

КЛЮЧОВІ СЛОВА: КЛАСИФІКАЦІЯ АНОМАЛІЙ, МАШИННЕ НАВЧАННЯ, МЕТРИКИ, АЛГОРИТМИ, НЕЙРОННІ МЕРЕЖІ, ГІПЕРОПТИМІЗАЦІЯ

ABSTRACT

The master's thesis consists of a description of the study of detecting website user anomalies, implementation of algorithms and various approaches to creating machine learning models with their evaluation, and an explanatory note. The explanatory note to the qualification work consists of 82 pages, 56 illustrations, 1 table, 2 appendices and 40 references.

The purpose of the work is to study web traffic, identify characteristics for classifying activity in order to detect anomalies. Using modern algorithms and approaches to design, implement and evaluate classification models using machine learning to identify different types of user requests.

The paper describes the goals and tasks for research and development. The existing algorithms, their features and defining characteristics were analyzed. As a result, the following algorithms were chosen: Decision Tree, Random Forest, SVM, Voting, Stacking, LSTM, KNN, DNN. An in-depth analysis of the web traffic structure was carried out and the elements that affect anomalies were identified. During the design process, the architectures of the selected algorithms were analyzed in detail. A dataset was selected and studied, and all the necessary manipulations were performed. The algorithms were implemented, evaluated, and compared.

The conclusion summarizes the results of research and development during the qualification work.

KEYWORDS: ANOMALY CLASSIFICATION, MACHINE LEARNING, METRICS, ALGORITHMS, NEURAL NETWORKS, HYPEROPTIMIZATION

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT	4
ЗМІСТ.....	5
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	10
1.1. Постановка задачі.....	10
1.2. Дослідження структури веб-трафіку та визначення вирішальних характеристик	12
1.3. Аналіз алгоритмів машинного навчання для класифікації аномалій	16
1.4. Вибір та обґрунтування інструментальних засобів розробки	24
Висновки до першого розділу.....	25
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА АЛГОРИТМІВ.....	26
2.1. Проєктування та реалізація алгоритмів та нейронних мереж	26
2.2. Дослідження датасету та методів його підготовки	34
2.3. Аналіз методів оптимізації та оцінки результатів	38
Висновки до другого розділу	43
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ОЦІНКА ТА ПОРІВНЯННЯ АЛГОРИТМІВ	45
3.1. Підготовка та нормалізація датасету.....	45
3.2. Моделювання та навчання алгоритмів виявлення аномалій	51
3.3. Оцінка та порівняння реалізованих моделей	56
Висновки до третього розділу	72
ВИСНОВКИ	73
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	75
ДОДАТКИ.....	81

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

DNN – Deep neural network

LSTM – Long Short-Term Memory

ПЗ – Програмне забезпечення

ШІ – Штучний інтелект

UDP – User Datagram Protocol

PSH – Push

URG – Urgent

KNN – K-Nearest Neighbors

TLU – Threshold Logic Unit

SVM – Support Vector Machine

OvR – One-vs-Rest

OvO – One-vs-One

ReLU – Rectified Linear Unit

FCBF – Fast Correlation Based Filter

ROC – Receiver Operating Characteristic

FPR – False Positive Rate

TPR – True Positive Rate

AUC – Area Under Curve

SMOTE - Synthetic Minority Over-sampling Technique

Hpo - Hyperparameter Optimization

ВСТУП

Актуальність теми. У сучасному світі використання інтернет ресурсів є частиною повноцінного існування. У таких умовах швидко розвиваються цифрові технології та обсяг веб-трафіку зростає з неймовірною швидкістю. Це створює передумови для поширення спроб кібератак. Виявлення аномальної активності у веб-ресурсах є важливим завданням для підвищення безпеки в інформаційних систем.

Традиційні методи виявлення аномальної поведінки, які включають у себе ручну актуалізацію правил та сигнатур, часто недостатньо ефективні в умовах динамічності у веб-середовищі. Застосування машинного навчання дозволяє автоматизувати процес аналізу великих обсягів даних запитів, виявити приховані закономірності та сформувані залежності для ідентифікації. Даний підхід допоможе швидше реагувати та адаптуватися до потенційних загроз.

Класифікація активності користувачів веб-системи дозволить проаналізувати як старі запити, так і нові. Це може надати потрібно інформацію власникам систем для аналізу спроб втручання в їх систему та сформувані подальший розвиток. Також аналізу часових рядів та закономірностей дозволить знаходити спільні закономірності між проміжками часу що підвищить ідентифікацію при спробі приховати певні дані.

Таким чином, розвиток та впровадження алгоритмів виявлення аномалій у реальному часі або в існуючих даних є необхідним для забезпечення кібербезпеки веб-ресурсів, а також для зменшення кількості хибних спрацювань традиційних алгоритмів. Це дозволить як організаціям, так і невеликим платформам зменшити ризики та збитки від кібератак, підвищуючи їхню стійкість до сучасних загроз.

Мета кваліфікаційної роботи – аналіз особливостей запитів у веб-системах, реалізація методів та алгоритмів для виявлення та класифікації активності систем з використанням сучасних методів машинного навчання. Проаналізувати наявні дані для навчання, провести їх дослідження та

нормалізацію відповідно до задач. При розробці моделей використовується мова програмування Python з використанням технологій Pandas, NumPy, Sckiti-learn, Keras, Tensorflow.

Визначена мета обумовлює такі завдання:

- аналіз та визначення основних характеристик для класифікації запитів веб-сайтів;
- дослідження, підготовка та обробка даних для навчання алгоритмів;
- дослідження існуючих методів та алгоритмів для аналізу аномалій;
- визначення типів моделей відповідно до поставленої задачі;
- розробка алгоритмів для класифікації поведінки користувачів;
- оптимізація моделей для підвищення їх точності;
- оцінка ефективності розроблених алгоритмів з використанням метрик, порівняння реалізованих методів між собою та висновки їх роботи;

Об'єктом дослідження є процес розробки моделей для класифікації поведінки відвідувачів вебсайтів за допомогою алгоритмів машинного навчання.

Предметом дослідження є алгоритми машинного навчання для виявлення різних типів поведінки користувачів вебсайтів, методи оптимізації та оцінки ефективності.

Наукова новизна роботи полягає у застосуванні методів глибокого навчання, зокрема LSTM, для класифікації аномальної поведінки користувачів вебсайтів з урахуванням часових характеристик та залежностей запитів. Також новизна включає у себе комбінування класичних методів машинного навчання (RandomForest, DecisionTree, ExtreTree, KNN) з методами ансамблю (Voting, Stacking) та глибокими нейронними мережами (DNN, LSTM) для знаходження оптимальних вирішень поставленої задачі.

Практичне значення роботи полягає у розробці ефективних алгоритмів для класифікації дій користувачів та виявленні аномальної поведінки, які можуть бути застосовані у роботах веб-систем для покращення

захищеності ресурсів, можливості аналізувати трафік та швидше адаптуватися до атак. Розроблені алгоритми можуть бути інтегровані на веб-сайти та проводити аналіз як в реальному часі, так і уже існуючих записів. Також результати проробленої роботи можуть бути використані для покращення існуючих систем захисту інформації та розробки нових інструментів для аналогічних цілей.

Апробація. За темою дослідження кваліфікаційної магістерської роботи було опубліковано 2 тези конференцій [1,2].

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Постановка задачі

Реалізація алгоритмів для виявлення аномалій у поведінці користувачів вебсайтів є важливим завданням у сфері моніторингу та захисту даних. Вебсайти, які мають великий обсяг користувацьких запитів, стикається з численними загрозами, які включають у себе спроби несанкціонованого доступу, шахрайські дії, зловмисне ПЗ для отримання доступу та інші форми. Для ефективної ідентифікації спроб таких дій необхідно впроваджувати системи, які здатні виявляти аномальну поведінку базуючись не тільки на статичних правилах, але і з можливістю виявляти приховані для людей патерни [3].

Основною метою проєкту є дослідження вирішальних характеристик веб-трафіку для ідентифікації аномальної поведінки, а також методів та алгоритмів, які допомагають класифікувати активність. Реалізовані моделі повинні ефективно класифікувати потік даних та визначати аномальну поведінку користувачів. Це дозволить виявляти загрози на ранніх етапах та забезпечувати своєчасну реакцію на потенційні ризики. Аналіз різних підходів дасть можливість оцінити ефективність тих чи інших методів, їх особливості та можливості оптимізації процесу моніторингу.

Виходячи з поставлених цілей, було сформовано такі етапи реалізації проєкту:

1. Дослідження запитів та даних, використовуючи які, можливо провести класифікацію аномалій.
 - a. Проаналізувати елементи трафіку веб-систем та їх особливості для знаходження ключових характеристик.
 - b. Визначити особливості аномальних поведінок, їх характеристики.
2. Аналіз алгоритмів та підходів для вирішення задачі класифікації, визначення оптимальних рішень та різних підходів для об'єктивних результатів.

3. Збір та підготовка даних.

- a. Пошук та формування набору даних, який задовільняє більшість вимог до досліджених характеристик.
- b. Аналіз та реалізація методів нормалізації, очищення від шуму та перетворення у зручний формат для роботи з алгоритмами.

4. Розробка та навчання моделей.

- a. На основі підготовлених даних та визначених алгоритмів реалізувати побудову моделей машинного навчання для вирішення задачі класифікації.
- b. Оцінка результатів та впровадження оптимізаційних заходів для покращення результату.
- c. Впровадження ансамблевих алгоритмів машинного навчання та їх оцінка.
- d. Побудова та навчання алгоритмів глибоких нейронних мереж з можливістю роботи з часовими характеристиками.
- e. Реалізація комбінованих підходів до побудови моделей з використанням нейронних мереж та алгоритмів.

5. Аналіз отриманих результатів та формування висновків.

- a. Проведення порівняння моделей з використанням різних метрик та підходів.
- b. Формування висновків на отриманих даних.
- c. Збереження моделей для їх можливості до подальших досліджень або використання.

Очікуваними результатами є наявні результати проведених досліджень, які демонструють ефективність впровадження та реалізації алгоритмів для класифікації поведінки користувачів вебсайтів. Також повинні бути готові моделі, які були успішно навчені та продемонстрували високі показники точності ідентифікації аномалій. Отримані моделі повинні бути у стані для продовження досліджень виявлення аномалій або використання в уже готових системах.

1.2. Дослідження структури веб-трафіку та визначення вирішальних характеристик

Ефективним із способів виявлення потенційних загроз є аналіз структури запитів та вивчення активності користувачів, що дозволяє виявляти відхилення від нормальної поведінки. Це ж і допоможе у навчання моделей та класифікації трафіку.

Веб-запити, які здійснюють користувачі, містять у собі значну кількість інформації, яка може бути використана для визначення нормальної чи аномальної поведінки. Структура цих запитів, типи HTTP запитів, параметри, заголовки та інші метадані можуть мати значний вплив на класифікацію дії. Також під час запитів важливо мати можливість збору та аналізу даних, які містяться у сесійних даних, журналах веб-серверу та cookie-файлів [3].

HTTP-запити є основним елементом, який використовують клієнти для комунікації із серверами [4]. Кожен запит містить важливу інформацію, яка може бути використана для аналізу взаємодії користувачів з вебсайтом. Розберемо структуру заголовків та склад запитів (рис. 1.1).

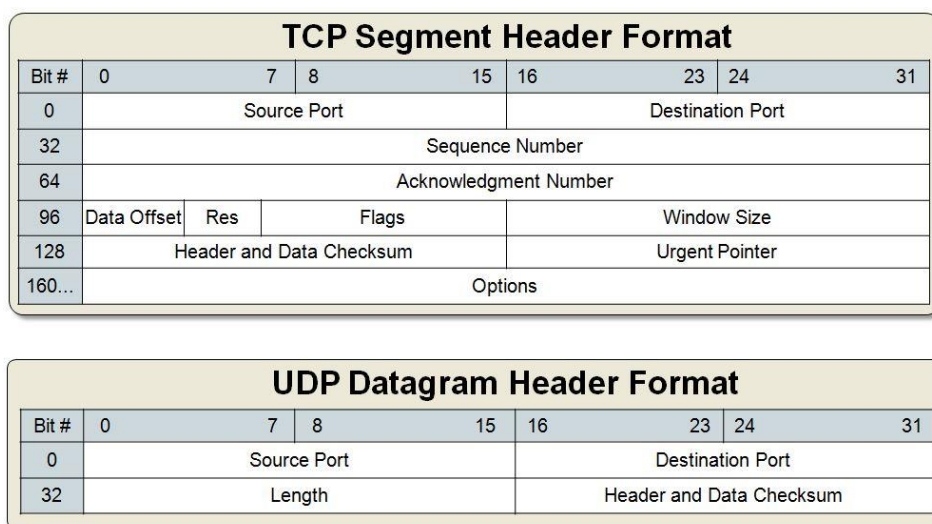


Рис. 1.1. Структура заголовків запиту

Одним із елементів запиту є протокол. Він забезпечує набір правил і стандартів для здійснення комунікації між пристроями в мережі, а саме визначає яким чином передаються дані. У дослідженні роботи використовуються протоколи для інтернет передачі даних, а саме TCP та UDP

[5]. Використання HTTP або HTTPS теж важливо для ідентифікації, хоча і в сучасних системах налаштовується переадресація на захищений протокол. Це більш категоріальна змінна, яка не може впливати на те чи буде запит аномальним чи ні, але при потребі може виступати елементом фільтрації, якщо моделі готуються для окремих протоколів.

Посилання містить у собі інформацію для ідентифікацію ресурсу направлення запиту. Саме посилання ідентифікують ресурс, до якого йшов запит. Таким чином при ідентифікації аномалії можна буде визначити що було ціллю зловмисних дій. На аномалію можуть вказувати підозрілі параметри запитів, або часті спроби доступу до конкретного посилання. Збереження даного елементу є важливим для ідентифікації та є частиною моніторингу.

Аналогічна важливість і збереження типу запиту. Якщо це RESTful API, то типи запиту можуть бути GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS. При виявленні аномалій можна звернути увагу на частоту запитів за певним типом, це може вказувати на DDoS або бутрфорс атаки.

Аналіз заголовків є надзвичайно важливим для вивчення при визначенні аномалій. Хоч вони і не містять дані, але визначають тип оброблення. Заголовки можна поділити на декілька категорій [7]:

- **Заголовки запиту** – заголовки, які клієнт надсилає разом із запитом. Містять у собі інформацію про середовище клієнта, інтерфейс користувача, авторизацію
- **Заголовки відповіді** – заголовки, які сервер надсилає клієнту разом із відповіддю. Містять у собі інформацію про тип контенту відповіді, статус, кешування.
- **Загальні заголовки** можуть бути присутні у запитах та відповідях. Містять у собі загальну мета-інформацію.

При дослідженні аномальних значень заголовки можуть вказувати на спроби обходу безпеки, наприклад фальшиві заголовки User-Agent для маскування клієнта або спроби підробити Authorization заголовки для отримання авторизованого доступу. Також про аномалію може свідчити

відсутність стандартних заголовків, що може вказувати на використання ботів або заскритованих сценаріїв. Також на аномалію може вказувати і наявність непотрібних заголовків, що може свідчити про спроби втручання в сесії.

Тіло запиту є надзвичайно важливим для визначення аномальної поведінки, адже саме в ньому зберігається інформація у складних форматах, яку сервер використовує для обробки та збереження. Для аналізу тіла, з метою забезпечення безпеки, звертають увагу на розмір, перевищення або надмірне зменшення його. Також важливим є формат, який передбачений або не передбачений системою. Саме ж наповнення тіла також може мати у собі загрози, якщо містить у собі елементи виконання коду або передаються дані, невідповідні до очікуваних.

У рамках дослідження трафіку важливо звертати увагу не лише на тіло запиту та відповіді, а і на сам потік, який може включати у себе декілька запитів [9]. Потік – це послідовність пакетів, які передаються у мережі між пристроями у межах одного безперервного з'єднання. При аналізі потоку з метою визначення важливих характеристик для ідентифікації аномалій важливо звернути увагу на такі особливості [8]:

1. Основні характеристики потоку.
 - a. Тривалість потоку.
 - b. Загальна кількість пакетів відправлення та отримання.
 - c. Загальний розмір пакетів відправлення та отримання.
2. Розмір пакетів.
 - a. Максимальний, мінімальний, середній розміри пакетів відправлення та отримання.
 - b. Відхилення розмірів від середнього значення.
3. Часові показники потоку.
 - a. Швидкість передачі байтів та пакетів при відправленні, отриманні та в загальному.
 - b. Часові інтервали між потоками: середній, максимальний, мінімальний, відхилення від середнього.

с. Часові інтервали між пакетами в обох напрямках окремо та в загальному: середній, максимальний, мінімальний, відхилення від середнього.

Також при дослідженні трафіку важливо звернути увагу на прапорці, такі як PSH або URG [6]. Ці контрольовані біти впливають на чергу оброблення всіх або окремих пакетів у TCP запитах відповідно. Зазвичай вони використовуються при обміні даних у реальному часі, що може включати відео або голосові дзвінки, а також передачі сигналів на термінове виконання певних команд. Таким чином ці дані можуть мати одне із вирішальних значень при класифікації аномальної поведінки.

Сесійні дані також є важливою складовою для відстеження взаємодії користувача з веб-сайтом. За допомогою них можна зберігати стан користувача впродовж певного часу в різних ситуаціях, таких як авторизація, кошик, налаштування браузера на сайті. Завдяки моніторингу компонентів сесії можна визначити послідовність змін, які можуть впливати на класифікацію дій користувача. Для визначення аномальної поведінки можна слідкувати за такими даними сесії: ідентифікатор сесії, тривалість, активність користувача на сторінках [10].

Аналогічно до сесій також можна слідкувати за зміною файлів cookies. Ці невеликі файли зберігають у собі інформацію про стан сесії, персональні налаштування браузера, часто використовуються для ідентифікації користувачів. Таким чином різка зміна файлів cookies між запитами може слідчити про аномальну активність.

Використовуючи сесії та файли cookies можна зберігати інформацію для моніторингу для подальшого аналізу. Для прикладу можна додати додаткові змінні, які будуть визначати поведінкові характеристики користувача. Це може бути час перебування на сайті, частота входів, певні дії з інтерфейсом вебсайту, які можуть бути вирішальними.

Аналіз структури веб-трафіку, зокрема зазначених вище елементів, можуть дати повну інформацію для ідентифікації класу активності у веб-системі.

1.3. Аналіз алгоритмів машинного навчання для класифікації аномалій

Процес виявлення аномалій за допомогою ІІІ полягає у ідентифікації моделлю незвичайних шаблонів або винятків у наборах даних, які відхиляються від нормального базового рівня [16]. За допомогою машинного навчання модель може автоматично виявляти ці аномалії, обробляючи великі обсяги даних, використовуючи мета-теги, мітки та закономірності [11].

Існує декілька та різних підходів для реалізації поставленої задачі.

Статистичні методи виявлення аномалій включають у себе виявлення точок даних, які відхиляються від очікуваних статистичних розподілів або закономірностей.

Метод класифікації за допомогою статистичних розподілів полягає у використанні параметричних та непараметричних моделей для моделювання нормальної поведінки даних. Параметричні розподіли припускають, що дані можуть бути описані конкретними параметрами, такими як середні значення, стандартні відхилення.

Одним із поширених статистичних методів параметричного розподілу є розподіл Гауса[15] у класифікаційному алгоритму Баєса [17]. Припускається, що ознаки незалежні між собою. Для кожного класу обчислюється умовна ймовірність кожної ознаки, а потім ймовірність класу для конкретного об'єкта. Клас з найвищою ймовірністю обирається як результат класифікації. Ймовірність для кожної ознаки та класу обчислюється за формулою:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \quad (1.1)$$

де μ – середнє значення (математичне сподівання);

σ – стандартне відхилення;

x - змінна, для якої визначається ймовірність.

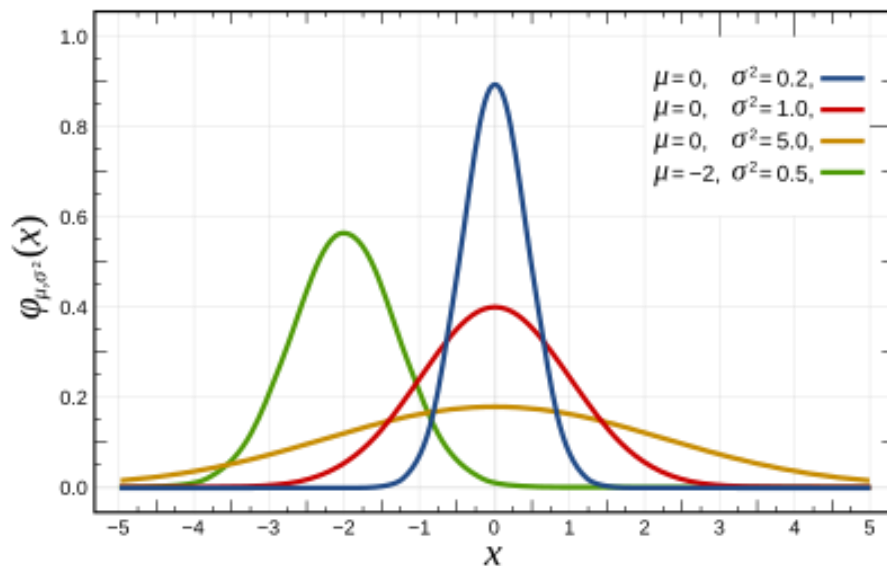


Рис. 1.2. Графік відхилень для кожного класу

Дана модель проста у моделюванні та розумінні. Доволі швидка в обчисленнях, але погано працює з даними, які не мають нормального розподілу, не підходить для складних або сильно нелінійних даних. Припущення незалежності між ознаками, що може давати хибні результати

Методи кластеризації передбачають групування схожих об'єктів даних у кластери. Ці методи допомагають виділяти нормальні об'єкти, які належать до певного кластеру, а також аномалії, які не належать до жодного кластера і віддалені від інших точок. Алгоритм K-means класифікує точки на основі мінімізації відстаней від центрів кластерів, тоді як DBSCAN орієнтований на щільність і може виявляти кластери різної форми. Існують методи, які створюють дерева кластерів для подальшого аналізу. Метою даних методів є виявлення найбільш схожих груп, при цьому зменшуючи відмінності всередині кластерів і збільшуючи їх між кластерами (рис. 1.3).

Алгоритми, що використовують це підхід, дозволяють ефективно класифікувати нові об'єкти, визначаючи їх належність до найближчих кластерів. Дані методи підходять для роботи з великою кількістю даних. Проте вони можуть бути чутливими до шуму та вихідних значень та не завжди здатні обробити складні структури даних.

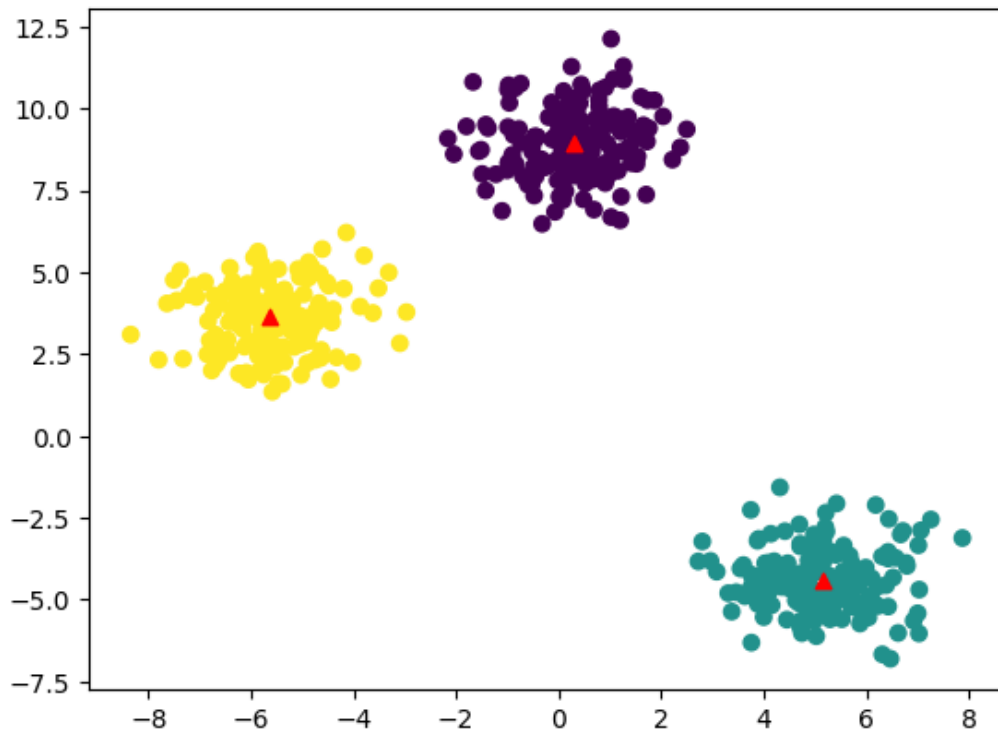


Рис. 1.3. Алгоритм кластеризації K-means

Методи класифікації використовуються для автоматично визначення типів класів даних, де кожен об'єкт має належати одному або кільком класам. Вони використовують метод навчання з учителем, де на основі навчальної вибірки моделюється функція, що здатна віднести нові об'єкти до визначених класів. Під час навчання класифікатор знаходить кордони між класами, намагаючись максимізувати відмінності між ними. Класифікатора можуть працювати з числовими та категоріальними даними, вимагаючи правильної підготовки даних. Також важливим елементом є наявність великої вибірки для уникнення перенавчання.

Однією з моделей класифікації та регресії є модель дерева рішень. Дана модель використовує структуру дерева для прийняття рішень (рис. 1.4) [18]. Кожен вузол дерева представляє перевірку на деяку умову, а гілки – результати. Листя дерева містять результат. Побудова дерева здійснюється за допомогою алгоритмів ID3 або CART, що вибирають найкращі умови розподілу на основі критеріїв інформаційного приросту або коефіцієнту Gini.

Таким чином чим менше значення індексу, тим кращий поділ на класи.

Важливою перевагою дерев рішень є їх швидкість тренування, підтримка різних типів змінних, відсутність необхідності у масштабуванні. З мінусів можна визначити нестабільність при зміні даних, схильність до перенавчання.

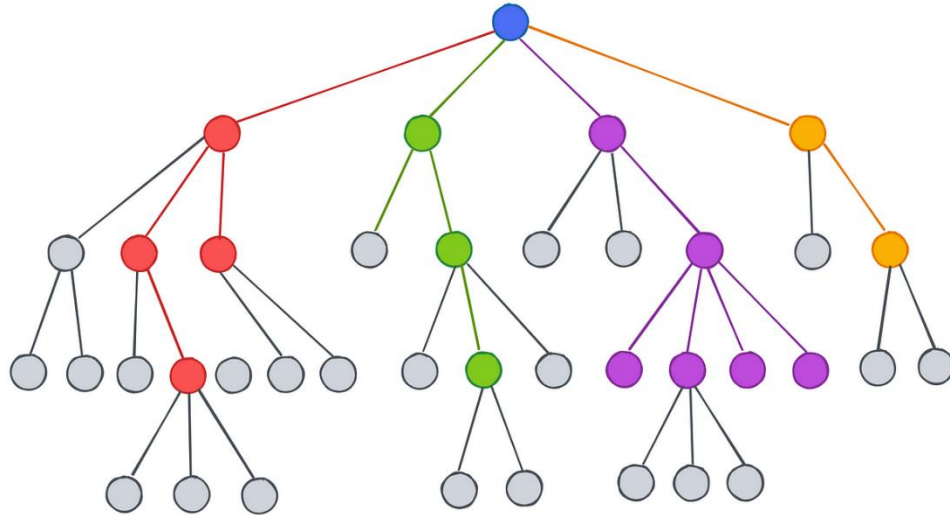


Рис. 1.4. Візуалізація дерева рішень

Інший підхід до класифікації включає у себе пошук найближчих сусідів для кожного нового об'єкта. До даного методу відноситься модель K-Nearest Neighbors [19]. Алгоритм працює за принципом обчислення відстаней між об'єктом, для якого потрібно зробити передбачення, та іншими об'єктами в наборі даних. Найближчі K сусідів використовуються для визначення класу або значення (рис. 1.5). Вибір сусідів K відбувається перед процесом навчання. Алгоритм є чутливим до шкал ознак, тому перед використанням є умовою нормалізація та стандартизація даних. Відстань між точками визначається різними метриками.

Алгоритм простий у розумінні, гнучкий у використанні, добре працює з малими та великими наборами даних. Також підходить для багатовимірних масивів. З недоліків можна визначити великі обчислювальні витрати на етапі класифікації, чутливість до масштабування.

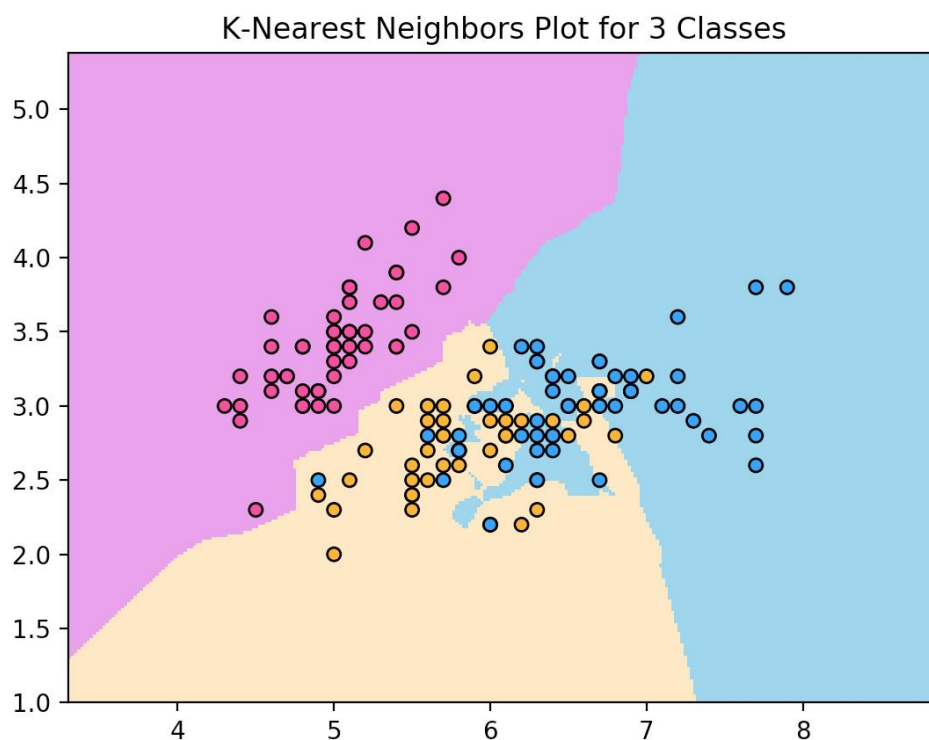


Рис. 1.5. Візуалізація KNN

Ансамблеві методи – це підхід в машинному навчання, який об’єднує кілька моделей для покращення точності та стабільності результатів [20]. Ці методи використовують ідею, що поєднання слабших моделей може дати результат, що перевищує точність окремих компонентів. Вони включають у себе техніки Bagging, Boosting, Stacking.

Bagging об’єднує результати декількох моделей, що навчаються на випадкових підмножинах даних. Однією з моделей є Random Forest, який використовує безліч дерев рішень для досягнення кращих результатів.

Boosting по черзі навчає моделі, де кожна наступна модель намагається виправити помилки попередньої. Прикладом такої моделі є Gradient Boosting Machines.

Stacking комбінує різні моделі, використовуючи мета модель для об’єднання їх прогнозів.

Перевагами ансамблевих моделей можна назвати покращену точність через зменшення можливостей помилок. Об’єднання підходів допомагає уникнути перенавчання. Можливість працювати з різними типами моделей. А також додаткові моделі зменшують вплив шуму та аномалій даних. З мінусів

можна визначити складність, адже потрібно реалізовувати більшу кількість моделей, вимога більших обчислювальних потужностей та важливості у правильному виборі гіперпараметрів.

Методи глибокого навчання є частиною машинного навчання, які використовують багат шарові нейронні мережі для аналізу великих обсягів даних. Вони здатні автоматично виявляти шаблони, класифікувати об'єкти і створювати нові дані, що нагадують вхідні. Глибоке навчання дозволяє працювати з неструктурованими даними, такими як зображення, текст або аудіо, завдяки використанню спеціалізованих архітектур. Такі мережі здатні самостійно виявляти важливі ознаки, що дозволяє отримати кращий результат ніж при використанні звичайних алгоритмів машинного навчання [**Error! Reference source not found.**].

Глибокі нейронні мережі (DNN) є потужним інструментом для вирішення складних завдань в області машинного навчання та глибокого навчання.

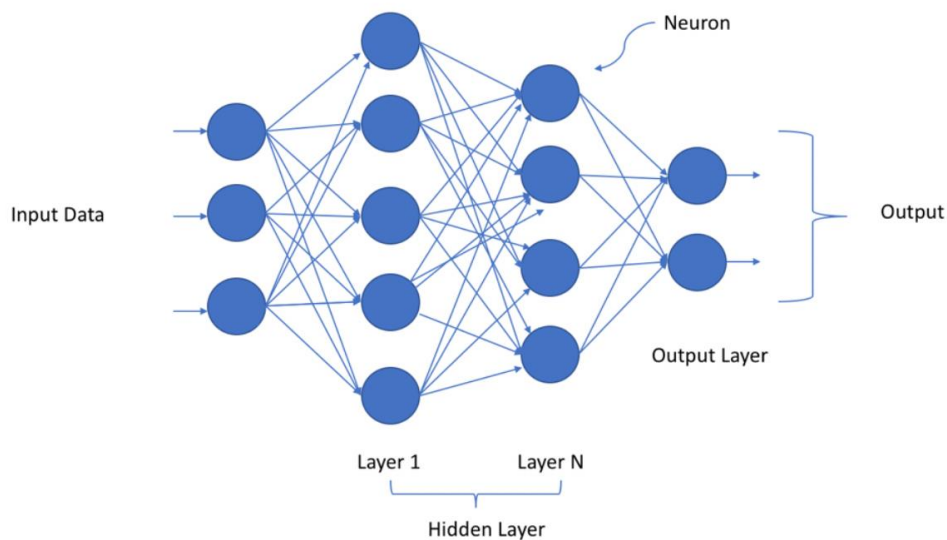


Рис. 1.6. Структура DNN

Їх особливістю є те, що вони складаються з багатьох шарів штучних нейронів, кожен з яких виконує математичні операції для трансформації вхідних даних (рис. 1.6) [14].

Для оптимізації навчання використовується функція втрат, яка мінімізується під час тренування мережі. Також для оптимізації можуть використовуватися окремі алгоритми, наприклад, Adam, для корегування ваг. Даний підхід дозволяє покращити параметри моделі, мінімізувати втрати та покращити точність.

Процес тренування DNN складається з кількох епох, під час яких мережа коригує свої ваги за допомогою методу зворотного поширення помилки [13]. У кожній епосі всі параметри мережі змінюються, поки модель не досягне найкращого рівня точності за епохи.

За потреби обробки послідовних даних, таких як часові ряди, текст, аудіо використовуються покращені нейронні мережі – рекурентні нейронні мережі. Вони мають внутрішні з'єднання, що дозволяють інформацію передаватися від одного елемента послідовності до наступного, що дає змогу моделі зберігати контекст та враховувати попередні етапи під час обробки нових даних. Одним з їх основним недоліків є ускладнене навчання на довгих послідовностях, оскільки вони не здатні ефективно зберігати інформацію про віддалені залежності.

Модель довгої та короткострокової пам'яті LSTM є покращеним варіантом рекурентних мереж, яка спеціально розроблена для обробки та прогнозування послідовних даних [22]. Їхня основна перевага полягає в здатності ефективно зберігати важливо інформацію протягом тривалого часу, що дає змогу моделі обробляти довгострокові залежності.

Основною особливістю LSTM є використання «стану клітин». Це стан дозволяє зберігати дані протягом тривалого часу без значних змін, що і забезпечує ефективне збереження даних. Кожна клітина має спеціальну структуру, яка складається з таких компонентів, воріт: вхідний, забезпечувальний, вихідний. Кожен із елементів контролює яку інформацію клітина повинна зберігати, яка забути, або яку буде передано на наступний етап або до виходу (рис. 1.7).

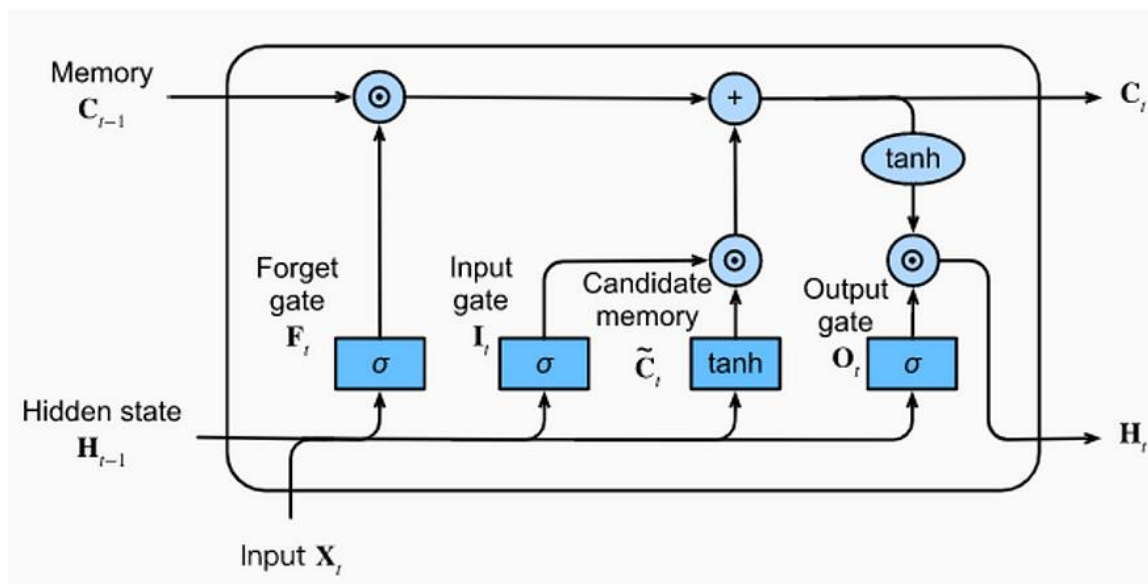


Рис. 1.7. Структура нейрону LSTM

Завдяки такій архітектурі LSTM здатна зберігати довготривалу інформацію про попередні етапи, що дозволяє ефективно працювати з послідовностями даних [23].

З основних переваг можна зазначити високу точність на складних задачах, автоматичне виділення ознак, масштабованість та гнучкість. Основними недоліками є великі вимоги обчислювальних ресурсів, довге навчання, ризик перенавчання, необхідність великої кількості даних та складність налаштування.

Виконавши дослідження особливостей моделей було вирішено реалізувати ряд алгоритмів машинного навчання, ансамблевих алгоритмів, нейронних мереж, рекурентних нейронних мереж, та комбінації нейронної мережі з алгоритмом машинного навчання. У роботі буде реалізовано такі моделі:

1. Decision Tree
2. Random Forest
3. SVM
4. Stacking
5. Voting
6. LSTM
7. DNN + KNN

1.4. Вибір та обґрунтування інструментальних засобів розробки

Для розробки та навчання моделей було проведено аналіз можливих рішень та обрано найоптимальніші інструменти для реалізації поставлених задач.

Для реалізації алгоритмів було обрано мову програмування **Python** версії 3.10. Це найбільш популярна мова у галузі машинного навчання завдяки своїй зручності, широкій течі бібліотек та активній спільноті. Використання даної мови дозволить реалізувати всі моделі. Також вона дає можливість швидко змінювати та експериментувати з моделями, гіперпараметрами, алгоритмами.

Основними бібліотеками для реалізації моделей було обрано Keras та Tensorflow.

Tensorflow є однією з провідних бібліотек для глибокого навчання, що дозволяє створювати та тренувати складні нейронні мережі з великою кількістю параметрів. Важливим показником вибору стала можливість виконання обчислень з використанням GPU, що робить його ідеальним для реалізації моделей, які було обрано [24].

Keras дозволяє спростити розробку та тестування моделей. Вона надає інтуїтивний інтерфейс, що дозволяє швидко створювати, налаштовувати та тестувати моделі [26].

Для збору та попередньої обробки даних було обрано бібліотеки **Pandas** та **NumPy**, що забезпечують ефективне опрацювання великих наборів даних та маніпуляцію з масивами.

Scikit-learn була обрана, так як дає можливість реалізації алгоритмів машинного навчання, таких як Decision Tree, Random Forest, SVM, Stacking, Voting.

Для виконання обчислень та тренування моделей обрано **vast.ai** як платформу для хмарного обчислення [25]. Цей інструмент було обрано через можливість швидко надавати доступ до потужних графічних процесорів з уже встановленим середовищем. Це надасть змогу ефективно натренувати

нейронні мережі з використанням більшого датасету, що повинно підвищити точність моделей.

Вибір інструментів був обумовлений їхньою популярністю, потужністю, а також доступністю для гнучкої та швидкої розробки моделей.

Висновки до першого розділу

У першому розділі випускної кваліфікаційної роботи було проаналізовано поставлено задачу, визначено основні вимоги та чітко поставлено задачі на цю роботу. Було проведено потрібно дослідження для подальшої реалізації алгоритмів.

Було проведено дослідження запитів, потоків даних та структури веб-трафіку. Було проведено аналіз елементів та визначено можливі характеристики, які будуть вирішувати класифікацію поведінки користувачів. Як результат було отримано список характеристик які потрібно врахувати під час дослідження датасетів для оптимального навчання.

Далі було проведено аналіз алгоритмів машинного навчання. Було визначено основні елементи архітектури, визначено переваги та недоліки, особливості побудови та їх можливості вирішення задач. Також було проаналізовано нейронні мережі глибокого навчання з можливостями роботи з часовими послідовностями. У результаті було складено список моделей, які будуть реалізовані у ході розробки.

Окремо було розглянуто, проаналізовано та вибрано інструменти, які забезпечать ефективну реалізацію поставлених цілей.

В результаті було отримано чітко сформоване бачення та розуміння реалізації алгоритмів машинного навчання та нейронних мереж для виявлення аномальної поведінки користувачів вебсайтів.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА АЛГОРИТМІВ

2.1. Проєктування та реалізація алгоритмів та нейронних мереж

Для реалізації ефективних моделей потрібно провести детальний аналіз архітектури, етапів роботи моделі, можливості оптимізації та особливості реалізації. Етапи роботи моделі включаються у себе підготовку даних, навчання, валідацію, оптимізацію та оцінювання результатів. Важливо зрештою звернути увагу на підбір гіперпараметрів та використанню методів регуляризації.

Однією з обраних моделей була модель дерева рішень. Її навчання базується на розбитті даних на підмножини через ітеративний вибір атрибутів, які найбільше зменшують невизначеність. На вході модель отримує набір даних, та намагається побудувати дерево.

На кожному етапі навчання вибирається певна ознака, яка використовується для розбиття даних. Цей вибір базується на критерії інформаційного приросту, індексу Gini, або зменшенні дисперсії. Даний критерій обирається перед навчанням та може впливати на кінцевий результат ефективності моделі. Для реалізації класифікації, де важливо мінімізувати невизначеність у вузлах, показник приросту даних має свої переваги через обчислювальну ефективність. Даний показник використовує ентропію для вимірювання невизначеності та вимірюється за формулою:

$$IG(D, X_j) = H(D) - \sum_{v \in V} \frac{|D_v|}{|D|} H(D_v), \quad (2.1)$$

Де V – усі можливі значення для X_j ;

D_v – підмножина даних, де $X_j = v$,

$H(D)$ - рівень невизначеності у даних.

Після вибору оптимальної ознаки дані розбиваються на підмножини відповідно до ознаки. Для числових ознак визначається поріг, який максимізує приріст інформації. Кожна підмножина стає вузлом у дереві і процес повторюється рекурсивно, реалізуючи деревоподібну структуру.

Завершення розбиття на вузли відбувається до моменту коли усі приклади у вузлі не належать одному класу або досягнуто мінімального розміру вузла. Після розбиття, кожен лист дерева отримує прогноз, який вираховується за формулою:

$$\hat{y} = \arg \max_{k \in (1,2,\dots,k)} \frac{n_k}{N}, \quad (2.2)$$

де $\hat{y}$ – прогнозований клас;

K – кількість класів;

n_k – кількість зразків класу у листі;

N – загальна кількість зразків у листі дерева.

Таким чином дерево рішень формує набір правил, які визначають маршрути від кореня до листових вузлів, а класифікація відбувається, слідуючи цим правилам.

Для збільшення ефективності даного методу можна використати декілька дерев рішень, що уже буде ансамблевим методом. Одним із таких методів є Random Forest. В його основі велика кількість дерев, кожне з яких працює над підмножиною даних і здійснює прогноз (рис. 2.1) [27].

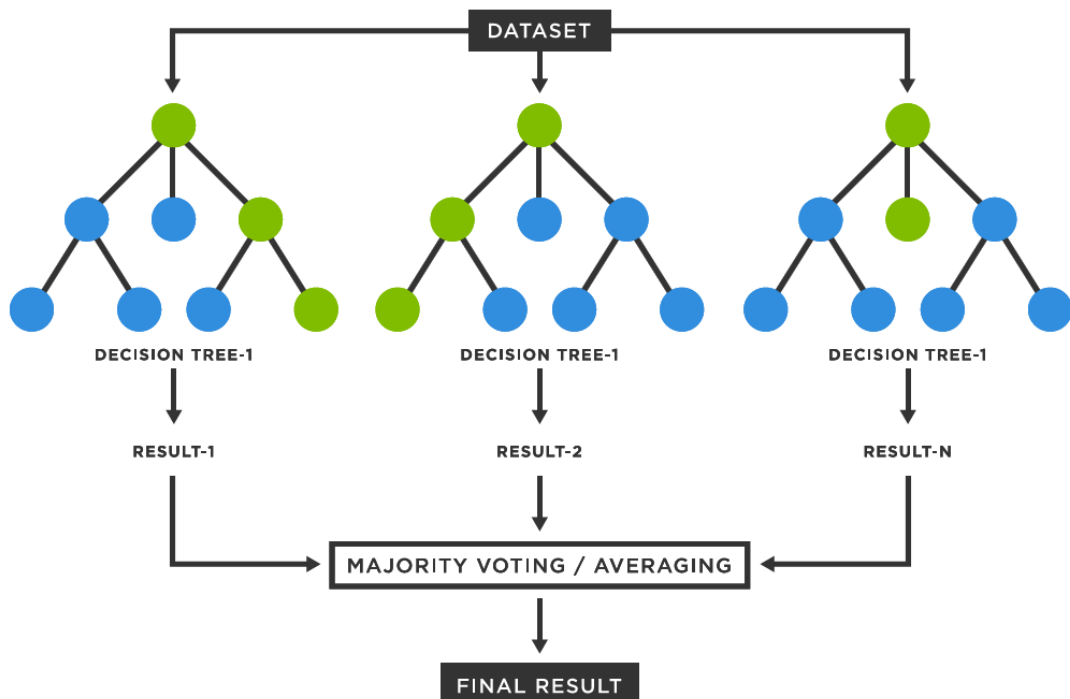


Рис. 2.1. Алгоритм роботи Random Forest

Дерева будуються за алгоритмом де кожне дерево навчається на випадковій вибірці з початкового набору даних, яка має розмір, що дорівнює розміру вихідного набору даних, але з можливістю дублювання елементів.

Для класифікації всі дерева дають свій прогноз, та клас який набирає найбільшу кількість голосів стає кінцевим.

Також було обрано реалізувати модель SVM. Цей алгоритм навчання будує гіперплощини, які розділяють дані на класи, забезпечуючи максимальний інтервал між класами. Головна ідея полягає у тому, щоб знайти таку гіперплощину, яка максимізує відстань між найближчими точками різних класів, що і є опорним вектором (рис. 2.2). Гіперплощина в SVM це субпростір, який має розмірність на одиницю менше за розмірність простору даних. Положення гіперплощини визначають опорні вектори [21].

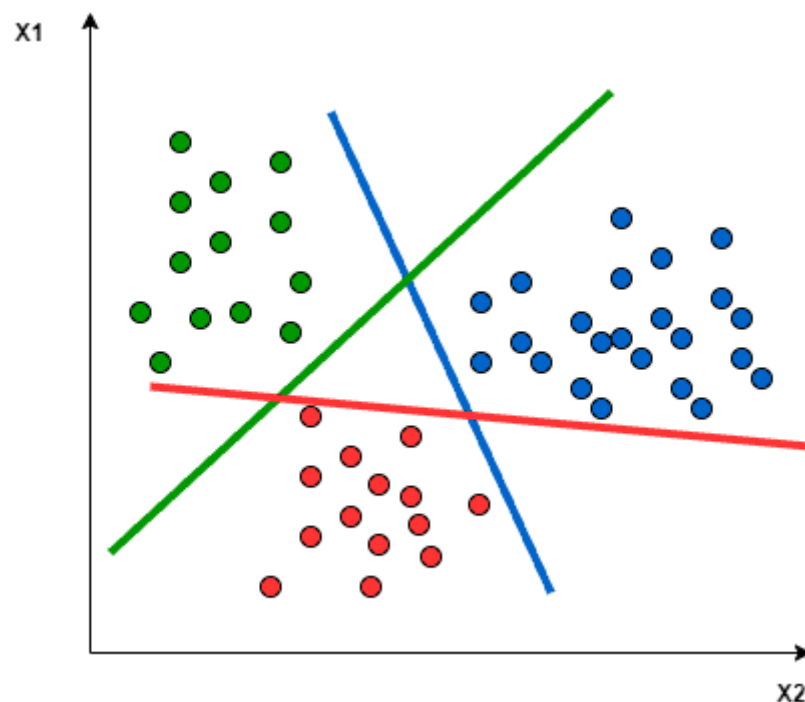


Рис. 2.2. Зображення роботи класифікації SVM

Основною задачею SVM є пошук гіперплощини так, щоб максимізувати відстань від гіперплощини до найближчих точок кожного з класів між двома класами. Математично це виглядає так:

$$\omega \cdot x + b = 0, \quad (2.3)$$

де ω – вектор ваг, нормаль до гіперплощини;

b – зсув, відстань від початку координат до гіперплощини;

x - вектор вхідних даних, ознаки.

Процес навчання полягає в оптимізації функції маржі. Таким чином завдання зводиться до пошуку параметрів ω та b , що максимізують відстань від гіперплощини до найближчих точок.

Якщо дані розділені не лінійно, то використовують ядра, що дозволяють перевести дані в більш високий простір ознак, де вони можуть стати лінійно роздільними. Для реалізації було обрано радіальне базисне ядро (RBF) [31]. Воно використовує гаусівську функцію для відстані між точками, що дозволяє ефективно обробляти складні та нелінійні ядра. Формула RBF ядра:

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right), \quad (2.4)$$

Де x, x' - два вектори, що представляють точки в просторі;

$\|x - x'\|^2$ – евклідова відстань між векторами;

σ – гіперпараметр, який регулює ширину функції ядра.

Прогнозування для нових точок робиться на основі того, на який бік гіперплощини вони потрапляють.

Існують декілька стратегій у побудові багатокласової моделі SVM: One-vs-Rest (Одним проти всіх) та One-vs-One (Один проти одного).

При використанні стратегії OvR для кожного класу будується окремий класифікатор SVM, який намагається відокремити цей клас від усіх інших. Таким чином модель створюється для кожного класу, і на основі з них обирається клас, який відповідає найбільшому значенню для кожного з SVM.

У підході SVM для кожної пари класів створюється окремий SVM, і на основі їх результатів прогнози комбінуються. Таким чином кількість моделей збільшується, але і збільшується точність в складних задачах.

Таким чином було обрано стратегію OvR.

Для об'єднання результатів розроблених моделей було вирішено реалізувати декілька підходів, які включаються у себе Voting та Stacking [30].

Для реалізації Voting класифікатора було обрано попередньо розроблені моделі, а саме: Decision Tree, Random Forest, SVM.

У процесі навчання кожен класифікатор навчається паралельно на однакових даних. Для кожного з них вибираються гіперпараметри, які оптимізуються в процесі навчання. Кожен клас генерує власне бачення для кожного тестового зразка.

Після завершення навчання класифікатора відбувається голосування за фінальний результат. Існують два типи голосування: hard та soft. Hard голосування обирає результат, який є найкращим, коли як soft видає ймовірність для кожного класу, і клас, для якого ймовірність найбільша, обирається як результат. Для реалізації було обрано метод голосування soft.

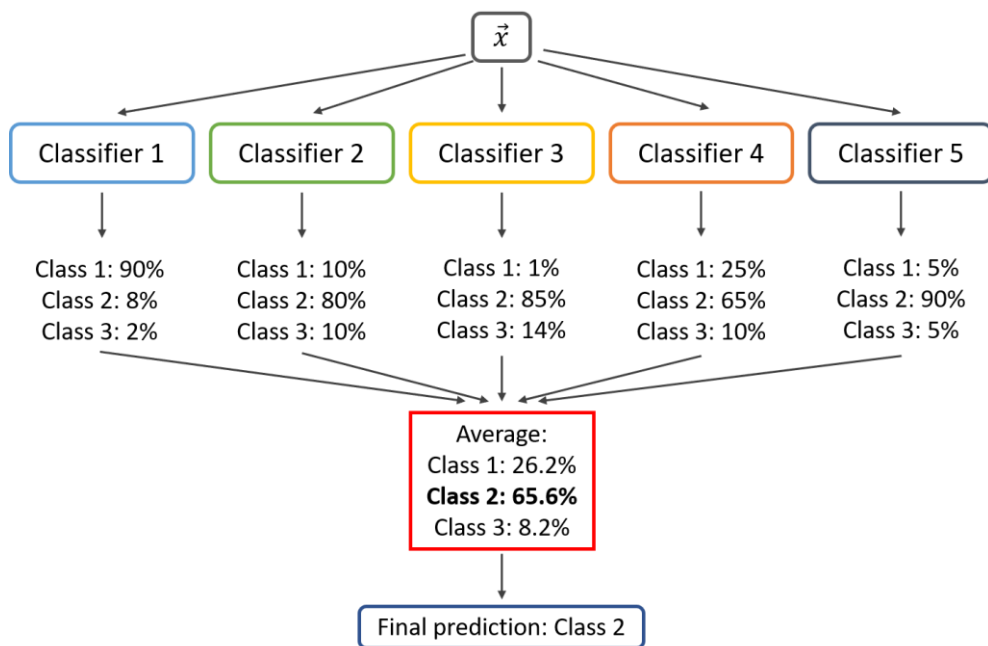


Рис. 2.3. Структура навчання Voting класифікатора

Для порівняння методів об'єднання моделей було обрано також модель Stacking. Дана модель використовує складніший метод, який полягає у використанні кількох базових моделей як вхід для мета-моделі. Ця модель навчається комбінувати прогнози базових моделей для того щоб приймати остаточне рішення.

Процес тренування полягає у тренуванні базових моделей. Після того, як базові моделі навчені, вони використовуються для прогнозування результатів

на тренувальних даних. Всі прогнози складаються у новий набір даних, що використовуються як вхід до мета-моделі.

Мета-модель навчається на основі збережених прогнозів. Складність моделі може бути меншою. У моєму випадку було обрано логістичну регресію. Головним є те, щоб вона правильно навчилася комбінувати прогнози від різних моделей. Після навчання, вона використовується для остаточного передбачення.

Реалізація нейронних мереж є складнішою, адже потрібно врахувати наявність прихованих шарів та складніший етап навчання [32].

Основною одиницею DNN є штучний нейрон, який можна описати як поріг логічної одиниці (TLU). TLU обчислює зважену суму своїх вхідних даних за допомогою формули:

$$z = \omega_1 x_1 + \omega_2 x_2 + \dots + \omega_n x_n = x^T w, \quad (2.5)$$

де $\omega_1, \omega_2, \dots, \omega_n$ – це коефіцієнти ваг;

$\omega_1, \omega_2, \dots, \omega_n$ – вхідні значення;

X – вектор вхідних даних;

W – вектор ваг.

Після цього нейрон застосовує активаційну функцію для отримання вихідного значення:

$$h_w(X) = \sigma(z), \quad (2.6)$$

де σ – активаційна функція, наприклад ReLU, що додає нелінійність у модель.

У багатошаровій нейронній мережі кожен нейрон з одного шару з'єднаний з усіма нейронами наступного шару, що створює структуру повністю з'єданого шару. У багатошаровій мережі дані проходять через кожен шар, де виконуються нові перетворення.

Для обчислення виходів багатошарової нейронної мережі для кількох екземплярів одночасно використовується матричне множення. Таким чином вихід для одного шару обчислюється за формулою:

$$h_{w,b}(X) = \sigma(XW + b), \quad (2.7)$$

де σ — це активаційна функція;

X — це матриця вхідних даних;

W — матриця ваг;

a, b — вектор зсувів.

Для прихованих шарів зазвичай використовується активаційна функція ReLU, але якщо задача багатокласова, то вихідний шар використовує функцію Softmax, яка перетворює вектори значень у ймовірності класів.

ReLU виконує перетворення для кожного вхідного значення, пропускаючи лише позитивні значення, а негативні змінює на нуль [28]. Це дозволяє більш ефективно працювати. Softmax у свою чергу перетворює лінійні виходи нейронної мережі у ймовірності для кожного класу. Ймовірності для всіх класів сумуються до 1, що дає можливість визначити їх як вірогідність належності входу до кожного з класів.

Процес навчання включає у себе обчислення кожного шару, обчислення помилки та обрахування ваг за допомогою оптимізаційних методів. У роботу було обрано метод Adam.

Для реалізації нейронної мережі було обрано створити структуру в якій є вхідний шар, два приховані шари, та вихідний шар з функцією SoftMax (рис. 2.4).

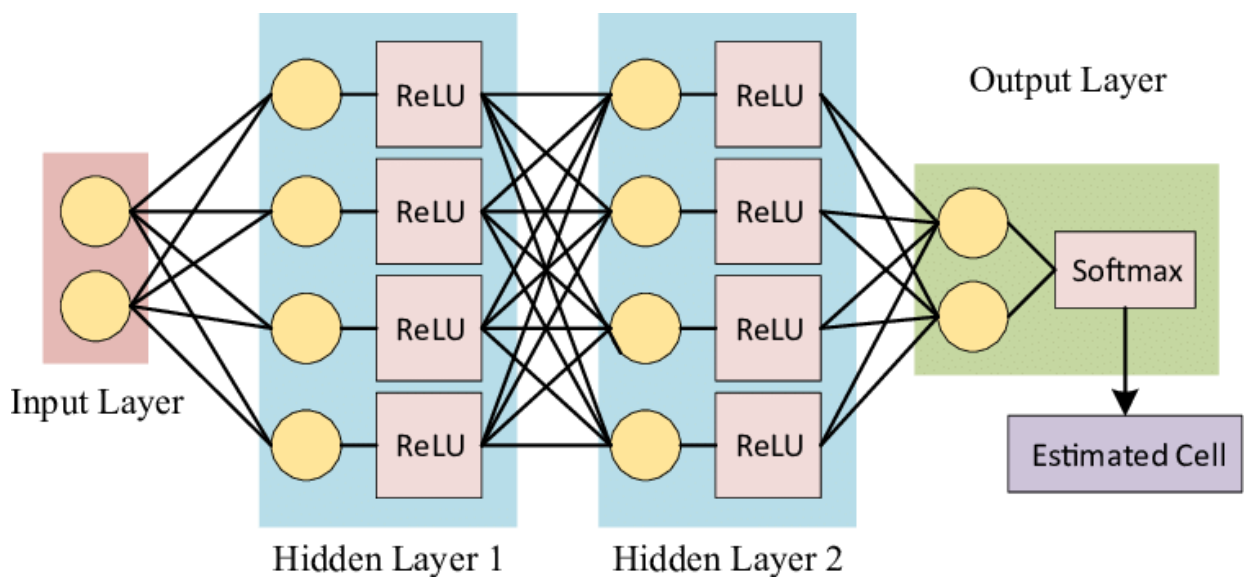


Рис. 2.4 Структура нейронної мережі

Процес навчання LSTM дещо складніший у зв'язку з можливістю зберігати інформацію.

На початку ініціалізуються всі параметри моделі, а саме зсуви для кожного з воріт та ваги воріг. Ваги зазвичай ініціалізуються випадковим чином, а зсуви встановлюються рівними нулю або малими значеннями.

Для кожного кроку часу мережа отримує вхідні дані та початкові стани прихованих шарів та клітин пам'яті. Кожен крок починається із забуття певних даних. Ворота забуття визначають яку частину старої пам'яті треба забути з використанням вагів для обчислень.

Після забуття ворота введення визначають яку нову інформацію додавати до клітин пам'яті. Це змушує клітину оновитись, комбінуючи стару та нову інформацію. На виході ворота визначають яка частина пам'яті буде передана далі.

Після проходження через мережу обчислюється помилка між прогнозованими виходами та фактичними мітками класів. Для цього використовується крос-ентропійна функція втрати, яка визначається за формулою:

$$L = -\sum y_t \log(h_t), \quad (2.8)$$

де y_t – справжнє значення;

h_t – передбачене значення для кожного кроку часу;

t – крок часу.

Після обрахування помилки використовується зворотне поширення для оновлення параметрів ваг та зсувів. Алгоритм дозволяє розрахувати градієнти помилки по кожному кроку часу. Після обчислень градієнтів застосовується метод оптимізації Adam.

Даний процес повторюється на кожному кроці часу для кожної тренувальної вибірки протягом заданої кількості епох. В процесі навчання оптимізуються ваги.

Для реалізації комбінування алгоритмів було використано DNN та KNN для створення гібридної моделі [29]. Дана модель буде комбінувати складне

модельовання даних за допомогою нейронних мереж із простотою та ефективністю методу KNN. Таким чином DNN зможе витягнути високорівневі ознаки та обрахувати ваги, а KNN зможе класифікувати на їх основі дані. Це повинно забезпечити кращий результат роботи ніж окремо кожна модель.

Після аналізу та проектування алгоритмів, які будуть розроблені, важливо обрати правильний набір даних, який буде містити у більшість ознак, які були дослідженні.

2.2. Дослідження датасету та методів його підготовки

Для проведення дослідження та розробки алгоритмів виявлення аномалій було обрано датасет CSE-CIC-IDS 2018 [33]. Цей набір даних є одним із найширших за класифікацією та кількістю даних у галузі інформаційної безпеки та мережевого трафіку.

Датасет включає у себе різноманітні типи мережесих атак. В отриманому наборі було визначено такі типи атак: DDOS attack-HOIC, DoS attacks-Hulk, Bot, SSH-Bruteforce, Infiltration, SQL Injection, DoS attacks-SlowHTTPTest, DoS attacks-GoldenEye, DoS attacks-Slowloris, DDOS attack-LOIC-UDP, Brute Force -Web, Brute Force -XSS. Це дозволяє охопити широкий спектр загроз, які зустрічаються у реальних умовах, і забезпечує можливість тестування моделей на великій кількості сценаріїв. Порівняно з іншими датасетами, даний пропонує більш сучасних перелік атак, який відповідає сучасним викликам у галузі кібербезпеки.

Набір даних сформований при імітації реальних атак на мережу. Це дає змогу забезпечити моделям ефективну роботу у реальних умовах. Датасет представлений у форматі .csv (Рис. 2.5) [34].

Датасет має 80 характеристик, з яких 77 є числовими та впливають на рішення класифікації. Це дозволить натренувати моделі, які будуть готові до ефективного використання. Основний акцент характеристик зміщений до потокових даних.

- e. Pkt Size Avg, Fwd Seg Size Avg, Bwd Seg Size Avg: Середній розмір пакета, сегментів по напрямкам.
- 3. Характеристика підпотоків.
 - a. Subflow Fwd Pkts, Subflow Fwd Byts, Subflow Bwd Pkts, Subflow Bwd Byts: кількість пакетів і байтів у підпотоках для обох напрямків.
- 4. Характеристики швидкості.
 - a. Flow Duration: тривалість потоку у мікросекундах.
 - b. Flow Byts/s: швидкість передачі байтів
 - c. Flow Pkts/s: швидкість передачі пакетів
 - d. Fwd Pkts/s, Bwd Pkts/s: швидкість передачі пакетів окремо по напрямках.
- 5. Інтервальні характеристики.
 - a. Flow IAT Mean/Std/Max/Min: середній, стандартне відхилення, максимальний та мінімальний час між пакетами в потоці.
 - b. Fwd IAT Tot/Mean/Std/Max/Min, Bwd IAT Tot/Mean/Std/Max/Min: середній, стандартне відхилення, максимальний та мінімальний час між пакетами по напрямках.
- 6. Прапори TCP.
 - a. Fwd PSH Flags, Bwd PSH Flags, Fwd URG Flags, Bwd URG Flags: показники прапорів PSH (Push) і URG (Urgent) для обох напрямків.
 - b. FIN Flag Cnt, SYN Flag Cnt, RST Flag Cnt, PSH Flag Cnt, ACK Flag Cnt, URG Flag Cnt, CWE Flag Count, ECE Flag Cnt: Лічильники різних TCP прапорів (наприклад, SYN для ініціації з'єднання, FIN для завершення).
- 7. Характеристика заголовків.
 - a. Fwd Header Len, Bwd Header Len: довжина заголовків в обох напрямках.
- 8. Активність.

- a. Active Mean/Std/Max/Min: час, коли потік був активним у мікросекундах.
- b. Idle Mean/Std/Max/Min: час, коли потік був у стані простою.

9. Характеристики TCP вікон.

- a. Init Fwd Win Byts, Init Bwd Win Byts: розмір ініціалізаційних вікон для сторін.

Аналіз датасету показує, що він містить майже всю можливу інформацію, яку можна отримати з потоків для аналізу трафіку.

Датасет має велику кількість записів тому ймовірно що буде потрібне класове балансування. Для цього можливо використати алгоритм SMOTE [35]. В своїй основі він використовується для вирішення проблем дисбалансу класів у наборах даних. Він працює шляхом створення нових синтетичних зразків для меншинного класу, що дозволяє збалансувати розподіл і покращити точність моделей машинного навчання. Алгоритм дії даного методу включає у себе вибір одного зразка класу, знаходження k найближчих сусідів з того ж класу. Для кожного найближчого генерується синтетичний зразок. Ці кроки повторюються доти, доки не буде досягнуто бажаного рівня балансу класів (рис. 2.6)

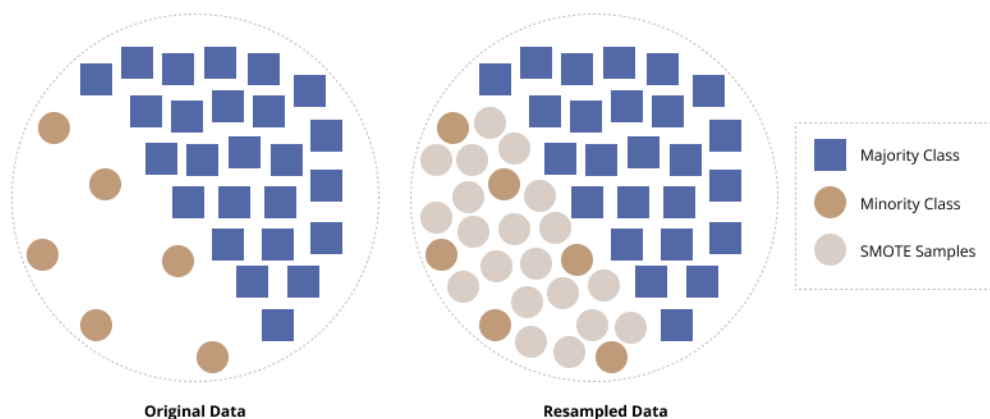


Рис. 2.6. Візуалізація роботи алгоритму SMOTE

Даний підхід дозволяє збалансувати класову нерівність у датасетах без втрати важливості даних.

2.3. Аналіз методів оптимізації та оцінки результатів

Оптимізація роботи моделей машинного навчання може бути досягнута шляхом відбору найважливіших ознак, що зменшує складність моделі, покращує її продуктивність та знижує ризик перенавчання. Даний підхід було вирішено застосувати для реалізації моделей машинного навчання.

Одним із методів, який активно використовується для оцінки важливостей ознак, є взаємна інформація. Даний показник демонструє кількість інформації, яку одна або набір ознак надають про класову мітку. Вона вимірює ступінь взаємозв'язку між ознакою та класом. І чим вища її значущість, тим більше інформації ознака несе для класифікації. Взаємна інформація для вхідних ознак та міток класів визначається за формулою:

$$MI(X, Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \left(\frac{p(x, y)}{P(X)p(y)} \right), \quad (2.9)$$

де X – вхідні ознаки;

Y – мітки класів;

$p(x, y)$ – спільна ймовірність подій x та y ;

$p(x)$ – маргінальна ймовірність для ознаки X ;

$p(y)$ – маргінальна ймовірність для мітки класу Y .

Після обчислень взаємної інформації для кожної ознаки їх потрібно відсортувати для простішого відбору. Далі потрібно нормалізувати важливості ознак, тобто поділити важливість кожної ознаки на загальну суму для отримання відносної важливості.

Після чого відбираються ознаки які задовільняють відсоток загальної важливості. Було обрано 95% як знак добре змодельованої моделі. Після відбору ознак, вони використовуються для подальшого навчання. Це допомагає досягнути кращих результатів у задачах класифікації.

Приклад коду для оцінки важливостей ознак:

```
# оцінка взаємної інформації між ознаками та мітками класу.  
importances = mutual_info_classif(X_train, y_train)  
print(importances)  
# обчислення суми значущостей усіх ознак та сортування по ній по спаданню
```

```

f_list = sorted(zip(map(lambda x: round(x, 4), importances), cols),
reverse=True)
Sum = 0
fs = []
for i in range(0, len(f_list)):
    Sum = Sum + f_list[i][0]
    fs.append(f_list[i][1])
# Фільтрування найважливіших ознак на основі значущості для отримання 95%
f_list2 = sorted(zip(map(lambda x: round(x, 4), importances/Sum), cols),
reverse=True)
Sum2 = 0
fs = []
for i in range(0, len(f_list2)):
    Sum2 = Sum2 + f_list2[i][0]
    fs.append(f_list2[i][1])
    if Sum2 >= 0.95:
        break
X_fs = df[fs].values

```

Для оптимізації гіперпараметрів було використано Баєсівську оптимізацію [36]. Даний метод глобальної оптимізації застосовує ймовірності моделі для пошуку оптимальних значень гіперпараметрів. Це дозволяє значно зменшити кількість випробуваних комбінацій параметрів і знаходити оптимальні значення ефективніше.

Баєсівська оптимізація намагається мінімізувати функцію втрат. У випадку поставленої задачі це функція, яка оцінює ефективність моделі на основі заданих гіперпараметрів. Для зменшення кількості випробувань використовується ймовірнісна модель, що дозволяє прогнозувати які комбінації гіперпараметрів ймовірно приведуть до найкращого результату. Поступово оновлюючи модель, даний метод вибирає наступні точки для оцінки на основі попередніх результатів. Це зменшує кількість спроб і дозволяє зосередитися на найбільш оптимальних комбінаціях параметрів.

Даний підхід дозволить швидко знайти оптимальні гіперпараметри при будь-якій їх кількості та підвищити точність моделі.

Для оптимізації гіперпараметрів нейронних мереж було використано Keras Tuner [40]. Даний підхід дозволяє автоматизовано оптимізувати гіперпараметри для моделей. У поставлених умовах було використано RandomSearch, який є одним із методів пошуку оптимальних гіперпараметрів. Даний метод передбачає вибір випадкових значень гіперпараметрів з визначного простору пошуку. Це дозволяє не перебирати всі можливі

варіанти, що значно знижує обчислювальну складність порівняно з іншими методами. Процес оптимізації включає у себе межі значень для кожного гіперпараметру, тренування моделі для кожної комбінації, використання колбеків зупинки, якщо подель поичнає перенавчатися.

Оцінка ефективності моделей є важливим етапом їх створення [37]. Для оцінки було використано метрики які включають у себе дані точності (accuracy), втрат (loss), точність відносно позитивних результатів (precision), повнота (recall), оцінка F1, матрицю плутанини класів та ROC-криві з AUC даними.

Точність моделі (accuracy) вимірює загальний відсоток правильних передбачень серед усіх можливих. Є ключовим фактором оцінки моделі. Даний параметр обраховується за формулою:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \quad (2.10)$$

де TP (True Positives) — правильно класифіковані позитивні;

TN (True Negatives) — правильно класифіковані негативні;

FP (False Positives) — хибні позитивні;

FN (False Negatives) — хибні негативні.

Точність відносно позитивних прогнозів (precision) вимірює наскільки модель правильно класифікує позитивні приклади серед усіх прикладів, які модель бачить як позитивні. Обраховується за формулою:

$$Precision = \frac{TP}{TP + FP}, \quad (2.11)$$

де TP (True Positives) — правильно класифіковані позитивні;

FP (False Positives) — хибні позитивні.

Точність відносно позитивних прогнозів

Повнота (recall) вимірює, яка частка всіх фактичних позитивних прикладів була правильно класифікована моделлю як позитивна. Дана метрика обраховується з формулою:

$$Recall = \frac{TP}{TP + FN}, \quad (2.12)$$

де *TP* (True Positives) — правильно класифіковані позитивні;

FN (False Negatives) — хибні негативні.

Оцінка *F1* дозволяє обчислити єдину метрику, яка поєднує точність та повноту. Вона дає значення, яке є найбільшим, коли обидві метрики дають однакове значення. Обраховується за формулою:

$$F1 = 2 \frac{Precision \cdot Recall}{Precision + Recall}, \quad (2.13)$$

де *Precision* — точність;

Recall — повнота.

Високе значення *F1* свідчить про гарний баланс моделі. Це одна із важливих метрик, адже видно балансоване значення, яке описує можливості моделі.

Для демонстрації результатів класифікації використовується також матриця плутанини класів (рис. 2.7). Це таблиця, яка оцінює якість класифікації моделі. Вона дозволяє побачити як часто модель правильно чи хибно класифікує об'єкти кожного класу [38].

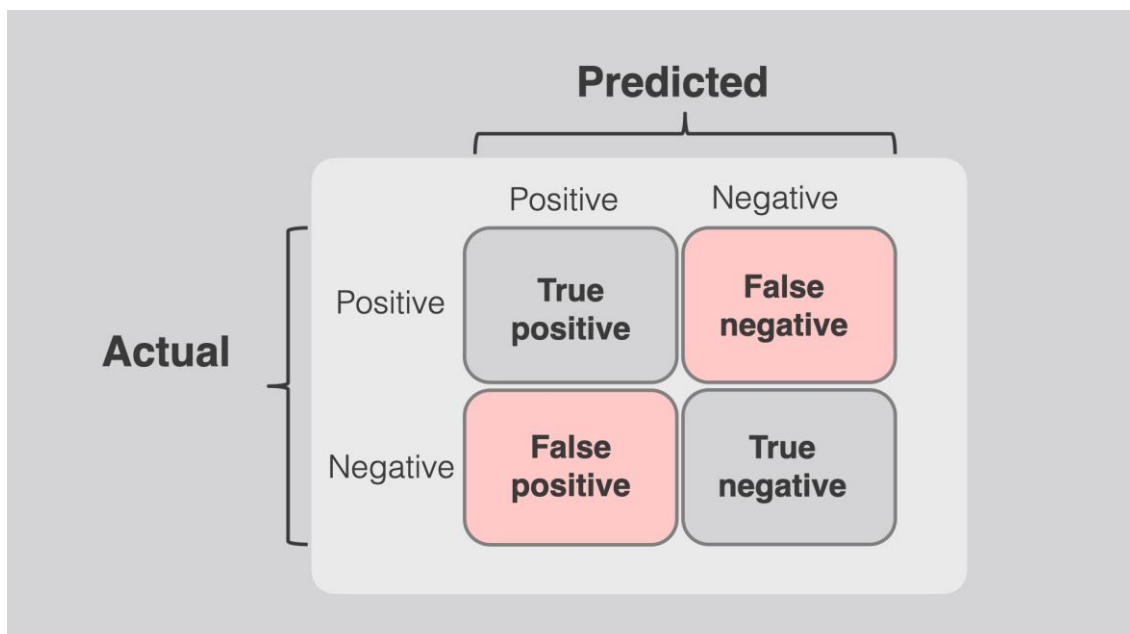


Рис. 2.7. Структура матриці плутанини

Для відображення залежності між чутливістю та хибно позитивними спрацюванням використовується ROC-крива. Для її побудови для кожного

класу обраховується частка хибно позитивних та істинно позитивних передбачень.

Частка хибнопо зитивних обчислюється за формулою:

$$FPR = \frac{FP}{FP + TN}, \quad (2.14)$$

де FP (False Positives) — хибно позитивні;

TN (True Negatives) — істинно негативні.

Частка істинно позитивних обчислюється за формулою:

$$TPR = \frac{TP}{TP + FN}, \quad (2.15)$$

де TP (True Positives) — істинно позитивні;

FN (False Negatives) — хибно негативні.

Для побудови рок кривою на вісь X відкладається FPR , а на вісь Y відкладається TPR (рис. 2.8). Дана крива дозволяє оцінити якість моделі, чи є модель збалансованою, або вона має проблеми з окремими класами [39].

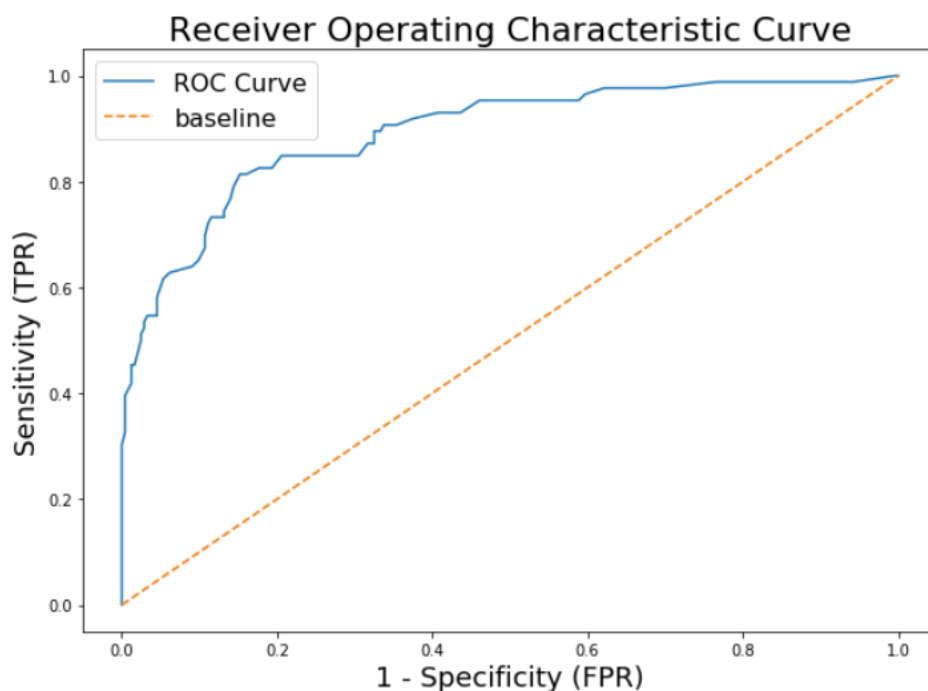


Рис. 2.8. ROC-крива

AUC – це площа під ROC-кривою. Його значення показує на скільки добре модель розріняє класи у числовому діапазоні від 0 до 1, де 1 – ідеальна модель. Формула для розрахунку AUC:

$$AUC = \int_0^1 TPR(FPR)d(FPR), \quad (2.16)$$

де $TPR(FPR)$ — функція залежності чутливості (TPR) від помилкових позитивів (FPR).

$d(FPR)$ — зміна FPR уздовж осі X.

AUC використовується для порівняння моделей та відображення їх ефективності.

Після аналізу архітектури, проєктуванню моделей, аналізу даних та метрик можна перейти до навчання моделей та їх оцінці.

Висновки до другого розділу

У другому розділі кваліфікаційної роботи було проведено аналіз архітектури алгоритмів, визначено основні етапи розробки, проведено всі потрібні кроки для успішної реалізації.

У підрозділі було проведено детальний аналіз архітектур обраних алгоритмів. Було визначено основні кроки реалізації, що потрібно врахувати та сформовано план моделювання. Було детально проаналізовано особливості роботи моделей машинного навчання Decision Tree, Random Forest, SVM, Voting, Stacking. Було визначено особливості реалізації кожної та можливі проблеми. Також було спроектовано нейронні моделі глибокого навчання DNN та LSTM, що включало у себе аналіз всіх елементів, проєктування структури моделі. Для створення гібридної моделі було обрано комбінацію нейронної мережі та машинного алгоритму KNN.

Далі було проведено аналіз джерел та обрано набір даних, який відповідає більшості поставлених вимог. За результатом пошуку було обрано датасет CSE-CIC-IDS 2018 як найоптимальніше рішення, яке забезпечить моделювання ефективних моделей. Було проаналізовано методи балансування даних, розроблено алгоритм з використанням K-means, який забезпечить створення нормального датасету.

Було проведено аналіз методів та способі оптимізації моделей, які включають у себе аналіз оцінки важливостей ознак, оптимізацію гіперпараметрів різними способами. Також було проаналізовано потрібні метрики для оцінки та порівняння моделей.

У результаті було виконано всі потрібні етапи для початку моделювання та оцінки моделей та завантаження готових моделей для роботи у реальних умовах.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ОЦІНКА ТА ПОРІВНЯННЯ АЛГОРИТМІВ

3.1. Підготовка та нормалізація датасету

Для навчання моделей потрібно правильно підготувати набір даних. Даний процес включає у себе завантаження датасету, очищення, балансування класів, якщо існує дисбаланс, нормалізацію даних та їх перетворення у відповідний формат для подальшого використання.

Завантаживши та проаналізувавши датасет (рис. 3.1), було визначено, що датасет має більше 8 млн записів що є надмірною кількістю для тренування, адже займе занадто багато часу.

	Dst Port	Protocol	Timestamp	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	Fwd Seg Size Min	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
0	443	6.0	02/03/2018 08:47:38	141385.0	9.0	7.0	553.0	3773.0	202.0	0.0	...	20.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Benign
1	49684	6.0	02/03/2018 08:47:38	281.0	2.0	1.0	38.0	0.0	38.0	0.0	...	20.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Benign
2	443	6.0	02/03/2018 08:47:40	279824.0	11.0	15.0	1086.0	10527.0	385.0	0.0	...	20.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Benign
3	443	6.0	02/03/2018 08:47:40	132.0	2.0	0.0	0.0	0.0	0.0	0.0	...	20.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Benign
4	443	6.0	02/03/2018 08:47:41	274016.0	9.0	13.0	1285.0	6141.0	517.0	0.0	...	20.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Benign
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8284190	23	6.0	28/02/2018 11:59:12	3.0	1.0	1.0	0.0	0.0	0.0	0.0	...	24.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Infiltration
8284191	425	6.0	28/02/2018 10:50:04	2.0	1.0	1.0	0.0	0.0	0.0	0.0	...	24.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Infiltration
8284192	445	6.0	28/02/2018 12:52:55	732728.0	2.0	2.0	0.0	0.0	0.0	0.0	...	32.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Benign
8284193	23	6.0	28/02/2018 11:10:50	22.0	1.0	1.0	0.0	0.0	0.0	0.0	...	24.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Infiltration
8284194	443	6.0	28/02/2018 11:12:18	2.0	1.0	1.0	0.0	0.0	0.0	0.0	...	24.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Infiltration

8284195 rows x 80 columns

Рис. 3.1. Огляд датасету

Також це підтвердилось великим класовим дисбалансом, де крайні класи мають більше 6 млн та 87 записів (рис. 3.2).

Label	
Benign	6112151
DDOS attack-HOIC	686012
DoS attacks-Hulk	461912
Bot	286191
FTP-BruteForce	193360
SSH-Bruteforce	187589
Infiltration	161934
DoS attacks-SlowHTTPTest	139890
DoS attacks-GoldenEye	41508
DoS attacks-Slowloris	10990
DDOS attack-LOIC-UDP	1730
Brute Force -Web	611
Brute Force -XSS	230
SQL Injection	87
Name: count, dtype: int64	

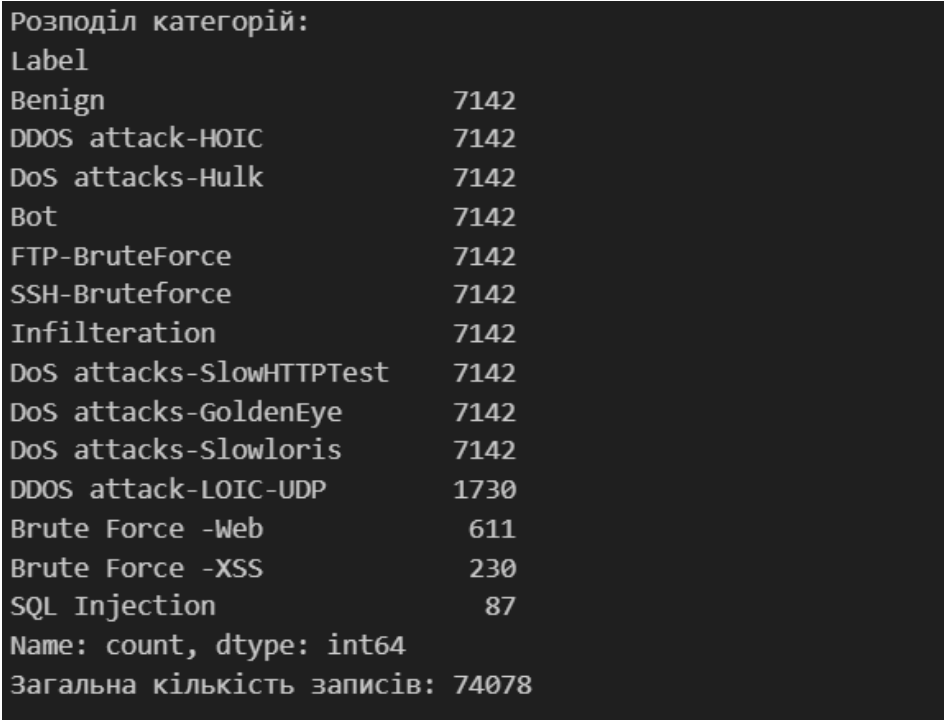
Рис. 3.2. Аналіз класового балансу

Для зменшення класового дизбалансу було вирішено поставити крайню верхню межу кількості записів для кожного класу, щоб фінальний датасет не перевищував кількість у 100000.

Лістинг коду для зменшення датасету:

```
# Встановлення цільового максимуму записів для великих класів
target_samples_per_class = int(100000 / len(df['Label'].unique()))
class_counts = df['Label'].value_counts()
balanced_dfs = []
for category, count in class_counts.items():
    class_df = df[df['Label'] == category]
    if count > target_samples_per_class:
        reduced_class_df = class_df.iloc[:target_samples_per_class]
    else:
        reduced_class_df = class_df
    balanced_dfs.append(reduced_class_df)
df = pd.concat(balanced_dfs)
```

Як результат було отримано більш збалансований та менший за розмір набір даних (рис. 3.3).



Розподіл категорій:

Label	
Benign	7142
DDOS attack-NOIC	7142
DoS attacks-Hulk	7142
Bot	7142
FTP-BruteForce	7142
SSH-Bruteforce	7142
Infiltration	7142
DoS attacks-SlowHTTPTest	7142
DoS attacks-GoldenEye	7142
DoS attacks-Slowloris	7142
DDOS attack-LOIC-UDP	1730
Brute Force -Web	611
Brute Force -XSS	230
SQL Injection	87
Name: count, dtype: int64	
Загальна кількість записів: 74078	

Рис. 3.3. Аналіз класів після зменшення

Класови дизбаланс все ще присутній, але у меншій мірі. Він буде виправлений після нормалізації датасету перед збереженням.

Було проведено чищення, трансформування типів даних. Для моделей машинного навчання, які не працюють з часовими рядами, часова змінна видаляється для оптимальної класифікації. А для LSTM зберігається та трансформується у потрібний формат.

Код лістингу очищення для моделей без досліджень часових рядів:

```
df.drop(['Dst Port', 'Timestamp', 'Protocol'], axis=1, inplace=True) # видалення
непотрібних значень
df.dropna(inplace=True) # видалення нульових значень
df = df[~(df.isin([np.nan, np.inf, -np.inf]).any(axis=1))] # видалення
нескінчених даних
```

Коді лістингу трансформації даних для моделі LSTM:

```
df["Timestamp"] = pd.to_datetime(df["Timestamp"], format='mixed')
df = df.dropna(subset=["Timestamp"])
df["Timestamp"] = pd.to_datetime(df["Timestamp"]).apply(lambda x:
int(x.timestamp()))
df.sort_values(by='Timestamp', inplace=True)
df.drop(['Dst Port', 'Protocol'], axis=1, inplace=True) # LSTM
df.dropna(inplace=True) # видалення нульових значень
df = df[~(df.isin([np.nan, np.inf, -np.inf]).any(axis=1))] # видалення
нескінчених даних
```

У результаті було отримано очищену структуру датасету (рис. 3.4).

```
Index: 73946 entries, 0 to 4528223
Data columns (total 77 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Flow Duration                        73946 non-null  float64
1   Tot Fwd Pkts                        73946 non-null  float64
2   Tot Bwd Pkts                        73946 non-null  float64
3   TotLen Fwd Pkts                     73946 non-null  float64
4   TotLen Bwd Pkts                     73946 non-null  float64
5   Fwd Pkt Len Max                     73946 non-null  float64
6   Fwd Pkt Len Min                     73946 non-null  float64
7   Fwd Pkt Len Mean                    73946 non-null  float64
8   Fwd Pkt Len Std                     73946 non-null  float64
9   Bwd Pkt Len Max                     73946 non-null  float64
10  Bwd Pkt Len Min                     73946 non-null  float64
11  Bwd Pkt Len Mean                    73946 non-null  float64
12  Bwd Pkt Len Std                     73946 non-null  float64
13  Flow Byts/s                         73946 non-null  float64
14  Flow Pkts/s                         73946 non-null  float64
15  Flow IAT Mean                       73946 non-null  float64
16  Flow IAT Std                        73946 non-null  float64
17  Flow IAT Max                        73946 non-null  float64
18  Flow IAT Min                        73946 non-null  float64
19  Fwd IAT Tot                         73946 non-null  float64
...
75  Idle Min                           73946 non-null  float64
76  Label                              73946 non-null  object
dtypes: float64(76), object(1)
```

Рис. 3.4. Очищена структура датасету

Для роботи з категоріями не підходить тип. Потрібні числові значення, тому було закодовано класи.

Лістинг кодування класів:

```
labels = df['Label'] # збереження всіх категорій
# Перетворення колонки класів у потрібний формат для подальшого тренування
le = LabelEncoder()
le.fit(df['Label'])
mapping = dict(zip(le.classes_, le.transform(le.classes_)))
```

```
df["Label"] = le.transform(df["Label"])
```

Результат кодування (рис. 3.5).

```
{'Benign': 0,  
'Bot': 1,  
'Brute Force -Web': 2,  
'Brute Force -XSS': 3,  
'DDOS attack-HOIC': 4,  
'DDOS attack-LOIC-UDP': 5,  
'DoS attacks-GoldenEye': 6,  
'DoS attacks-Hulk': 7,  
'DoS attacks-SlowHTTPTest': 8,  
'DoS attacks-Slowloris': 9,  
'FTP-BruteForce': 10,  
'Infiltration': 11,  
'SQL Injection': 12,  
'SSH-Bruteforce': 13}
```

Рис. 3.5. Результат кодування класів

Після даних дій можна зберігати датасет для подальшої реалізації LSTM, так як там потрібно виконати окремі дії із часовими значеннями. Для реалізації інших моделей було нормалізовано значення використовуючи функцію стандартизації даних. Особливістю функцію є стандартизація даних, роблячи їх центрованими навколо нуля та із стандартним відхиленням. Такий підхід робить дані придатними для алгоритмів машинного навчання, які чутливі до масштабування, а також пришвидшує навчання завдяки уніфікації ознак.

Лістинг стандартизації ознак:

```
df_without_labels = df.drop('Label', axis=1, inplace=False) # видалення  
категорій для правильної нормалізації  
df_columns = df_without_labels.columns.tolist()  
df_s_scaled = StandardScaler().fit_transform(df_without_labels) # нормалізація  
значень standart  
df_s_scaled = pd.DataFrame(data = df_s_scaled, columns = df_columns) #  
приведення до формату датафрейму  
df_s_scaled['Label'] = df['Label'].reset_index(drop=True)
```

Результатом є датасет який сформований до навчання (рис. 3.6). Але також ще можлив покращити ситуації із класовим дазбалансом, використовуючи раніше створену функцію з використанням K-means.

	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	Fwd Pkt Len Mean	Fwd Pkt Len Std	Bwd Pkt Len Max	...	Fwd Seg Size Min	Active Mean	Active Std
0	-0.399349	-0.150239	0.108883	-0.150588	0.149373	-0.016223	-0.177331	0.106840	0.086918	2.303956	...	-0.779506	-0.121311	-0.086362
1	-0.403415	-0.150622	-0.239345	-0.151469	-0.074846	-0.680335	-0.177331	-0.573416	-0.562353	-0.772922	...	-0.779506	-0.121311	-0.086362
2	-0.395360	-0.150129	0.573187	-0.149675	0.550746	0.724830	-0.177331	0.704371	0.534911	2.303956	...	-0.779506	-0.121311	-0.086362
3	-0.403419	-0.150622	-0.297383	-0.151534	-0.074846	-0.834215	-0.177331	-0.877928	-0.849934	-0.772922	...	-0.779506	-0.121311	-0.086362
4	-0.395527	-0.150239	0.457111	-0.149335	0.290097	1.259360	-0.177331	1.410367	1.118155	2.303956	...	-0.779506	-0.121311	-0.086362
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
73941	-0.259121	-0.150458	-0.123269	-0.150346	0.038244	1.976115	-0.177331	1.346615	2.471813	3.237557	...	-0.779506	-0.121311	-0.086362
73942	-0.259001	-0.150458	-0.123269	-0.150329	0.045970	2.016610	-0.177331	1.378669	2.519677	3.511526	...	-0.779506	-0.121311	-0.086362
73943	-0.232406	-0.149582	0.341035	-0.142749	0.933993	1.919423	-0.177331	3.038759	2.367810	3.300781	...	-0.779506	-0.121311	-0.086362
73944	-0.205732	-0.149198	0.573187	-0.139048	1.329363	1.919423	-0.177331	3.297098	2.322827	4.539962	...	-0.779506	-0.121311	-0.086362
73945	-0.403417	-0.150622	-0.239345	-0.151534	-0.074846	-0.834215	-0.177331	-0.877928	-0.849934	-0.772922	...	-0.779506	-0.121311	-0.086362

73946 rows x 77 columns

Рис. 3.6. Датасет після нормалізації

Для виправлення дизбалансу було використано алгоритм SMOTE, який забезпечує створення схожих зразків для вирішення класового дизбалансу.

Лістинг коду:

```
X = df_s_scaled.drop('Label', axis=1) # Ознаки
y = df_s_scaled['Label'] # Мітки
smote = SMOTE(sampling_strategy='auto') # Автоматичне визначення співвідношення класів
X_res, y_res = smote.fit_resample(X, y)
df_balanced = pd.DataFrame(data=np.column_stack([X_res, y_res]),
columns=df_s_scaled.columns)
print(df_balanced['Label'].value_counts())
```

```
Label
0.0    7142
4.0    7142
7.0    7142
1.0    7142
10.0   7142
13.0   7142
11.0   7142
8.0    7142
6.0    7142
9.0    7142
5.0    7142
2.0    7142
3.0    7142
12.0   7142
Name: count, dtype: int64
```

Рис. 3.7. Структура класів після SMOTE

Було покращено ситуацію із класами (рис. 3.7). Дизбаланс все ще присутній, але він уже всі класи мають мінімальну необхідну кількість значень для навчання.

Також було проведено аналіз датасету та його найважливіших ознак. Для цього було використано метод вибору ознак SelectKBest, який використовує статистичну функцію для оцінки значущості кожної ознаки відносно цільової змінної.

Таким чином можна визначити які характеристики є значущими при класифікації.

Лістинг визначення найкращих ознак:

```
selector = SelectKBest(score_func=f_classif, k=20) # Вибір топ 20 ознак
X = balanced_df
y = balanced_df['Label']
X = X.drop(columns = ['Label'])
selector.fit(X, y)
selected_features = X.columns[selector.get_support()]
print(f"shape of selected features: {selected_features.shape}")
X_selected = X[selected_features]
scores      =      pd.DataFrame(selector.scores_[selector.get_support()],
index=selected_features, columns=['Score'])
scores.sort_values('Score', ascending=False)
correlation_matrix = X_selected.corr()
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Теплова карта кореляції ознак')
plt.savefig('results/corr_plot.png')
plt.show()
```

Як результат було отримано список із значущих елементів з якого можна сформувати теплову карту (рис. 3.8) для відображення. За допомогою теплової карти можна візуально оцінити взаємозв'язки між різними ознаками в наборі даних.

Кожна комірка на цій карті відповідає парі ознак та значення з кольором вказує на силу і напрямок кореляції між цими ознаками.

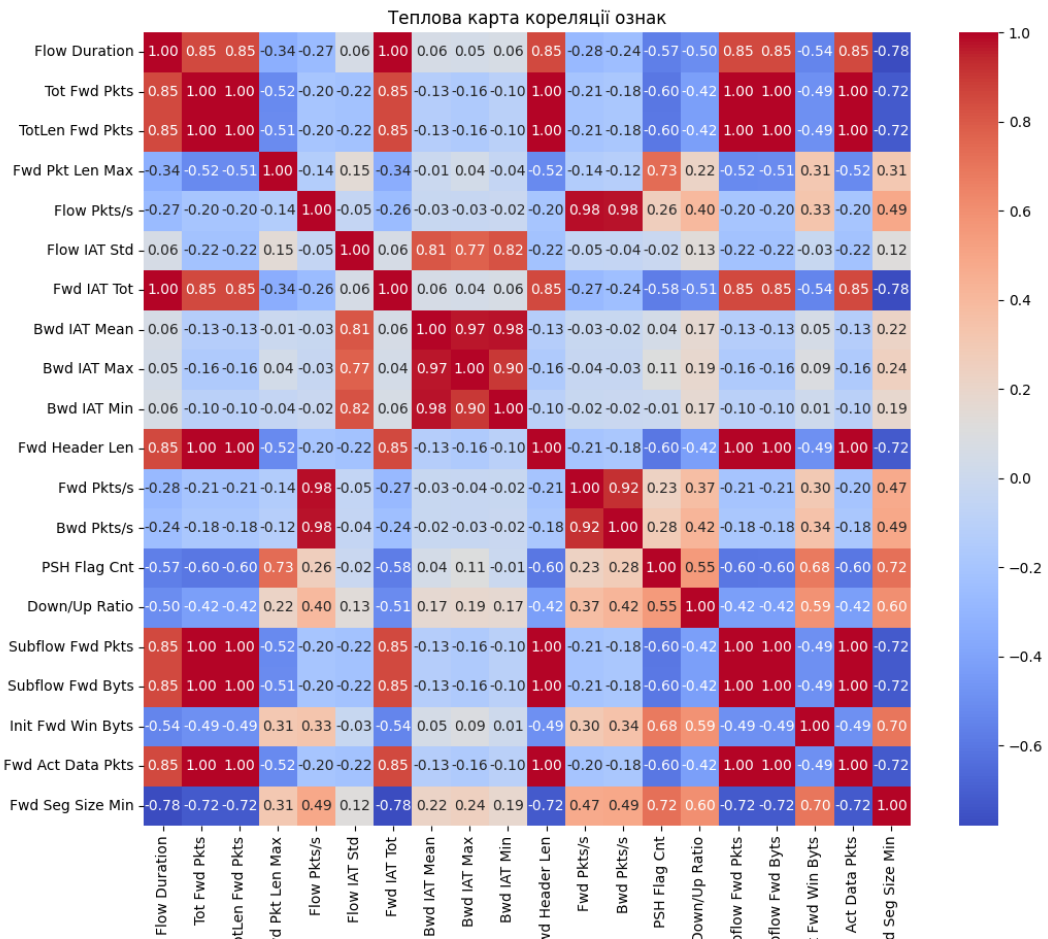


Рис. 3.8. Теплова карта кореляції топ 20 ознак

Датасети сформовані, тому можна приступати до навчання моделей.

3.2. Моделювання та навчання алгоритмів виявлення аномалій

Маючи всі необхідні дані можна перейти до тренування моделей. Цей процес включає у себе налаштування параметрів моделі, сам процес навчання, реалізація спроб оптимізації зміною параметрів, використанням різних підходів та алгоритмів для покращення результату. Для виконання дослідження було виконано реалізацію декількох типів моделей, які включаються у себе дерева рішень, ансамблеві методи, векторні методи, нейронні мережі та їх модифікації а також створення гібридних моделей.

Для оптимізації роботи алгоритмів було вирішено виконати процедуру відбору ознак. Основна мета – визначити ознаки в наборі даних, які мають найбільших вплив на цільову змінну та зосередити процес навчання саме на них. Лістинг коду:

```
from sklearn.feature_selection import mutual_info_classif
```

оцінка взаємної інформації між ознаками та мітками класу.

```
importances = mutual_info_classif(X_train, y_train)
```

```
print(importances)
```

```
# обчислення суми значущостей усіх ознак та сортування по ній по спаданню
f_list = sorted(zip(map(lambda x: round(x, 4), importances), cols),
reverse=True)
Sum = 0
fs = []
for i in range(0, len(f_list)):
    Sum = Sum + f_list[i][0]
    fs.append(f_list[i][1])
# Фільтрування найважливіших ознак на основі значущості для отримання 95%
f_list2 = sorted(zip(map(lambda x: round(x, 4), importances/Sum), cols),
reverse=True)
Sum2 = 0
fs = []
for i in range(0, len(f_list2)):
    Sum2 = Sum2 + f_list2[i][0]
    fs.append(f_list2[i][1])
    if Sum2 >= 0.95:
        break
print(f"Кількість ознак: {len(fs)}")
X_fs = df[fs].values
```

Даний підхід дозволив зменшити кількість ознак з отриманням 95% збереження ваг для класифікації. В результаті було зменшено кількість характеристик до 48.

Далі було побудовано моделі машинного навчання. Для оптимізації моделей було вирішено використати баєсівську оптимізацію гіперпараметрів.

Лістинг побудови моделі дерева рішень після визначення гіперпараметрів:

```
dt_hpo = DecisionTreeClassifier(
    min_samples_leaf=int(best['min_samples_leaf']),
    max_depth=int(best['max_depth']),
    min_samples_split=int(best['min_samples_split']),
    max_features=int(best['max_features']),
    criterion=criterion_mapping[best['criterion']],
    class_weight='balanced'
)
timed_dt_hpo = Timer(dt_hpo)
timed_dt_hpo.fit(X_train, y_train)
dt_score_hp = timed_dt_hpo.score(X_test, y_test)
y_predict = timed_dt_hpo.predict(X_test)
```

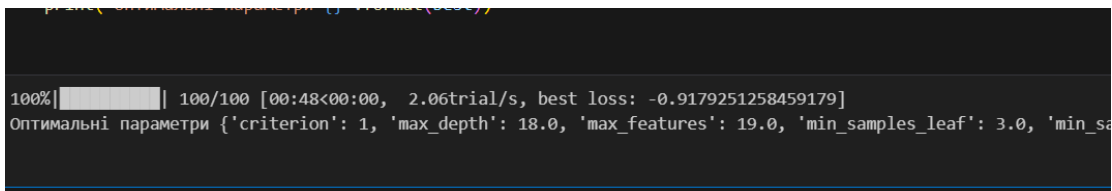


Рис. 3.9. Процес визначення гіперпараметрів

Модель LSTM розроблена спеціально для роботи з послідовними даними, які можуть включати у себе часові ряди. Саме тому, для збереження послідовності запитів було створений окремий датасет. Структура LSTM зберігає інформацію, тому потрібно сформувати певні послідовності та часові ряди для забезпечення кращого навчання.

Для створення послідовностей було реалізовано функцію:

```
def create_sequences(data, time_steps=1):
    sequences = []
    for i in range(len(data) - time_steps):
        seq = data[i:i + time_steps]
        sequences.append(seq)
    return np.array(sequences)
```

Так як на попередньому етапі для даної моделі не було нормалізовано даних, тому це відбудеться зараз. Для надання даних для визначення послідовностей було відсортовано набір даних, після чого створено окремі часові мітки для покращення машинного навчання. Для кращого опису періодичних процесів часових даних було додатково створено синусоїдні та косинусоїдні види часових міток. Використання цих ознак дозволить краще виявляти сезонність та періодичність аномалій.

Лістинг перетворення часових міток:

```
# Перетворення Timestamp у datetime
df['Timestamp'] = pd.to_datetime(df['Timestamp'])
# додавання часових міток
df['hour'] = df['Timestamp'].dt.hour
df['minute'] = df['Timestamp'].dt.minute
df['dayofweek'] = df['Timestamp'].dt.dayofweek
df['dayofyear'] = df['Timestamp'].dt.dayofyear
df['month'] = df['Timestamp'].dt.month
# Перетворення циклічних характеристик
df['hour_sin'] = np.sin(2 * np.pi * df['hour'] / 24)
df['hour_cos'] = np.cos(2 * np.pi * df['hour'] / 24)
df['minute_sin'] = np.sin(2 * np.pi * df['minute'] / 60)
df['minute_cos'] = np.cos(2 * np.pi * df['minute'] / 60)
df['dayofweek_sin'] = np.sin(2 * np.pi * df['dayofweek'] / 7)
df['dayofweek_cos'] = np.cos(2 * np.pi * df['dayofweek'] / 7)
df['month_sin'] = np.sin(2 * np.pi * df['month'] / 12)
df['month_cos'] = np.cos(2 * np.pi * df['month'] / 12)
```

Після формування часових даних було масштабовано дані та створено послідовності:

```
# Створення послідовностей
time_steps = 5 # Кількість попередніх записів для використання
X = create_sequences(X_raw, time_steps)
y = create_sequences(y_raw, time_steps)
y = y[:, -1]
```

Дані готові до тренування тому було розділено на тренувальні та тестові дані та обчислено ваги класів для зваження під час тренування, що допоможе для виправлення дисбалансу, який присутній.

Лістинг коду:

```
# Розділення на тренувальні та тестові дані
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7,
test_size=0.3, random_state=0, stratify=y)
# One-hot encoding для міток класів
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
# Обчислення ваг класів для зваження під час тренування для виправлення
дисбалансу
class_weights = compute_class_weight(class_weight='balanced',
classes=np.unique(y_train.argmax(axis=1)), y=y_train.argmax(axis=1))
class_weight_dict = dict(zip(np.unique(y_train.argmax(axis=1)), class_weights))
```

Для оптимізації моделі було додано алгоритм Adam, який адаптивно регулює швидкість навчання для кожного параметра моделі. При побудові моделі було вирішено зробити два LSTM шари.

Перший шар з 128 нейронами вивчає базові залежності даних, а другий шар вивчає більш складні залежності на основі виходів першого шару. Послідовності передаються з першого шару до другого. Після кожного шару додається шар нормалізації для покращення стійкості навчання та шар Dropout, який випадково відключає частину нейронів під час тренування для запобігання перенавчання.

Також було додано повнов'язний прихований шар для нелінійної трансформації даних. З використанням функції Softmax на виході буде отримано ймовірність приналежності до кожного класу.

Для запобігання перенавчання також було додано механізм ранньої зупинки, який дозволить припинити процес тренування, якщо якість моделі не покращується тривалий час. Після створення та навчання (рис. 3.10) моделі було отримано структуру шарів (рис. 3.11).

Для покращення моделі було прийнято рішення провести пошук кращих гіперпараметрів, що може забезпечити вищу ефективність роботи моделі. Для цього було використано об'єкт RandomSearch з бібліотеки Keras Tuner.

```

809/809 — 11s 14ms/step - accuracy: 0.9359 - loss: 0.1757 - precision: 0.9415 - recall: 0.9295
Epoch 7/100
809/809 — 11s 13ms/step - accuracy: 0.9366 - loss: 0.1638 - precision: 0.9408 - recall: 0.9320
Epoch 8/100
809/809 — 11s 13ms/step - accuracy: 0.9381 - loss: 0.1532 - precision: 0.9418 - recall: 0.9335
Epoch 9/100
809/809 — 11s 14ms/step - accuracy: 0.9392 - loss: 0.1541 - precision: 0.9420 - recall: 0.9358
Epoch 10/100
809/809 — 11s 14ms/step - accuracy: 0.9400 - loss: 0.1505 - precision: 0.9429 - recall: 0.9365
Epoch 11/100
809/809 — 10s 13ms/step - accuracy: 0.9427 - loss: 0.1412 - precision: 0.9451 - recall: 0.9396
Epoch 12/100
809/809 — 8s 9ms/step - accuracy: 0.9464 - loss: 0.1277 - precision: 0.9481 - recall: 0.9445 -
Epoch 13/100
...
Epoch 61/100
809/809 — 12s 15ms/step - accuracy: 0.9729 - loss: 0.0573 - precision: 0.9732 - recall: 0.9728
Epoch 62/100

```

Рис. 3.10. Процес навчання моделі LSTM

Layer (type)	Output Shape	Param #
lstm_10 (LSTM)	(None, 5, 128)	111,616
batch_normalization_20 (BatchNormalization)	(None, 5, 128)	512
dropout_15 (Dropout)	(None, 5, 128)	0
lstm_11 (LSTM)	(None, 64)	49,408
batch_normalization_21 (BatchNormalization)	(None, 64)	256
dropout_16 (Dropout)	(None, 64)	0
dense_10 (Dense)	(None, 64)	4,160
batch_normalization_22 (BatchNormalization)	(None, 64)	256
activation_10 (Activation)	(None, 64)	0
dropout_17 (Dropout)	(None, 64)	0
dense_11 (Dense)	(None, 14)	910
batch_normalization_23 (BatchNormalization)	(None, 14)	56
activation_11 (Activation)	(None, 14)	0

Рис. 3.11. Будова шарів LSTM

Гібридна модель поєднує у собі переваги глибокого навчання нейронної мережі та методів машинного навчання. Даний підхід повинен об'єднати 2 абсолютно різних алгоритми та дати кращий результат класифікації.

Процес побудови гібридної моделі включає у себе побудову нейронної глибокої мережі, навчання її. З нейронної мережі видаляється останній шар

класифікації. Після чого отримана модель використовується для обчислення нових ознак, базуючись на отриманих ознаках попередньої.

При побудові нейронної мережі було створено модель з чотирьох шарів, в якій перший вхідний, два прихованих та четвертий вихідний гар класифікатора. В результаті було отримано таку структуру (рис. 3.12).

Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 128)	9,856
batch_normalization_4 (BatchNormalization)	(None, 128)	512
activation_4 (Activation)	(None, 128)	0
dropout_3 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 64)	8,256
batch_normalization_5 (BatchNormalization)	(None, 64)	256
activation_5 (Activation)	(None, 64)	0
dropout_4 (Dropout)	(None, 64)	0
dense_6 (Dense)	(None, 32)	2,080
batch_normalization_6 (BatchNormalization)	(None, 32)	128
activation_6 (Activation)	(None, 32)	0
dropout_5 (Dropout)	(None, 32)	0
dense_7 (Dense)	(None, 14)	462
batch_normalization_7 (BatchNormalization)	(None, 14)	56
activation_7 (Activation)	(None, 14)	0

Рис. 3.12. Будова шарів DNN

Після навчання було видалено останній шар та збережено ознаки.

Лістинг коду:

```
# Витягування ознак з нейронної мережі для навчання KNN
feature_extractor = Sequential(dnn_model.layers[:-1])
X_train_dnn_features = feature_extractor.predict(X_train)
X_test_dnn_features = feature_extractor.predict(X_test)
```

Далі вкористовуючи ці ознаки було побудовано модель KNN. Таким чином було отримано гібридну модель.

Реалізувавши то натренувавши всі моделі було отримано достатньо інформації для проведення оцінки та порівняння результатів.

3.3. Оцінка та порівняння реалізованих моделей

Для оцінки ефективності моделей використовувалися стандартні метрики, такі як точність (ассигасу), точність відносно позитивних (precision),

повнота (recall) та F1-score. Ці метрики допомагають оцінити як загальну ефективність моделей, так і їхню здатність правильно класифікувати позитивні та негативні випадки.

Крім того, було розглянуто побудову ROC-кривих та обчислення площі під кривою, що дозволяє оцінити здатність моделей відокремлювати класи. ROC-криві надають візуальне уявлення про якість класифікації для різних порогових значень. Обчислення AUC є важливим, оскільки ця метрика дозволяє порівнювати моделі незалежно від вибору порогу.

Також було створено матриці плутанини які надають детальне уявлення про розподіл правильних та помилкових класифікацій. Аналіз матриць плутанини дозволяє зрозуміти, у яких випадках моделі роблять помилки і як ці помилки розподіляються серед класів.

Decision Tree

У результаті навчання моделі Decision Tree Було отримано такий результат по основним метрикам (рис. 3.13 - 3.14).

Загальна точність: 0.917991799179918				
Час навчання: 0.72 секунд				
Точність відносно позитивних: 0.9229015382146222				
Повнота: 0.917991799179918				
F1: 0.9163652265050887				
	precision	recall	f1-score	support
0.0	0.77	0.75	0.76	2143
1.0	1.00	1.00	1.00	2143
2.0	0.97	0.98	0.97	2142
3.0	1.00	0.99	0.99	2143
4.0	1.00	1.00	1.00	2142
5.0	1.00	1.00	1.00	2143
6.0	1.00	1.00	1.00	2142
7.0	1.00	1.00	1.00	2143
8.0	0.82	0.51	0.63	2142
9.0	1.00	1.00	1.00	2143
10.0	0.64	0.89	0.75	2143
11.0	0.75	0.77	0.76	2142
12.0	0.98	0.97	0.98	2143
13.0	1.00	1.00	1.00	2143
accuracy			0.92	29997
macro avg	0.92	0.92	0.92	29997
weighted avg	0.92	0.92	0.92	29997

Рис. 3.13. Навчання Decision Tree

Загальна точність: 0.9143581024769144				
Час навчання: 1.49 секунд				
Точність відносно позитивних: 0.9191719727410735				
Повнота: 0.9143581024769144				
F1: 0.9127279430988434				
	precision	recall	f1-score	support
0.0	0.73	0.73	0.73	2143
1.0	1.00	1.00	1.00	2143
2.0	0.97	0.99	0.98	2142
3.0	1.00	0.99	0.99	2143
4.0	1.00	1.00	1.00	2142
5.0	1.00	1.00	1.00	2143
6.0	1.00	1.00	1.00	2142
7.0	1.00	1.00	1.00	2143
8.0	0.82	0.51	0.63	2142
9.0	1.00	1.00	1.00	2143
10.0	0.64	0.89	0.75	2143
11.0	0.73	0.73	0.73	2142
12.0	0.99	0.98	0.98	2143
13.0	1.00	1.00	1.00	2143
accuracy			0.91	29997
macro avg	0.92	0.91	0.91	29997
weighted avg	0.92	0.91	0.91	29997

Рис. 3.14. Навчання Decision Tree Про

З отриманих даних можна визначити що модель має непогану точність, а саме 0.914, а всі інші метрики доволі близько до неї, що говорить про балансний процес тренування: точність: 0.914, точність відносно позитивних: 0.919, повнота 0.914, f1-score: 0.913. Після проведеної баєсівської оптимізації

було отримано незначні покращення показників. Отримані дані: Точність: 0.918, точність відносно позитивних: 0.922, повнота 0.918, f1-score: 0.916. Покращення доволі незначні, що ставить під сумнів ефективність використання оптимізації гіперпараметрів при балансних показниках точності. Збільшення показників відбулось лише на 0.4%, що є доволі непомітним результатом.

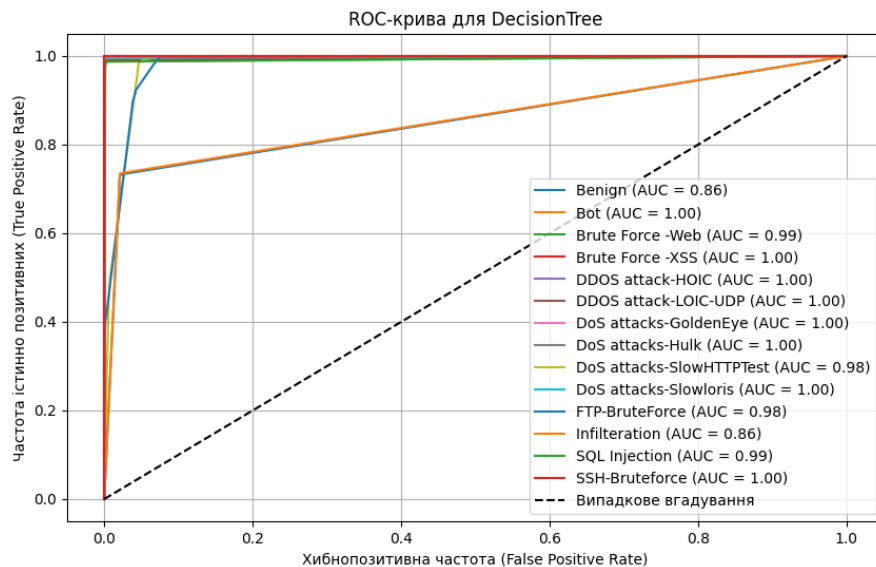


Рис. 3.15. ROC-крива Decision Tree

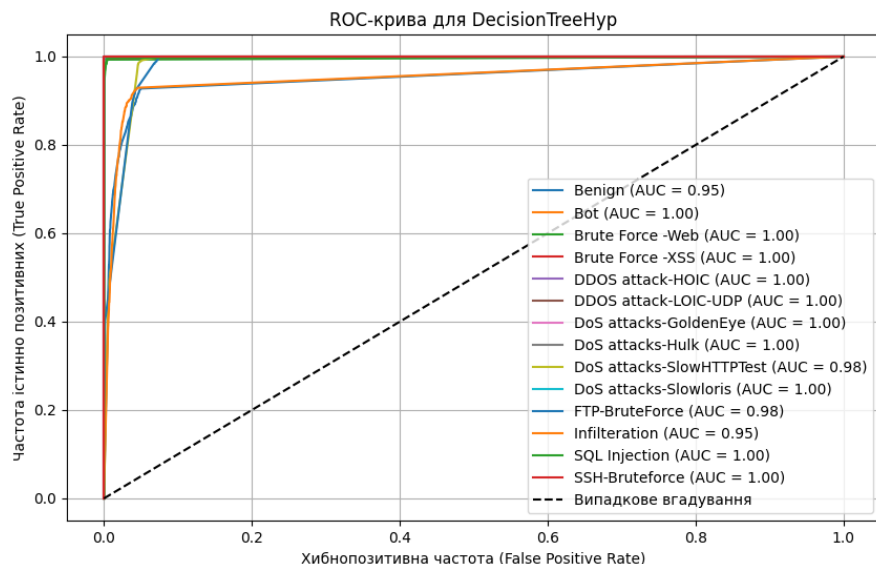


Рис. 3.16. ROC-крива Decision Tree Hyp

Також було оцінено ROC-криві (рис. 3.15-3.16). З аналізу графіків ROC-кривих видно що для окремих класів покращилися значення, що може показувати на наявні позитивні результати після гіпероптимізації. Матриця плутанини в обох випадках залишається майже незмінною (рис 3.17).

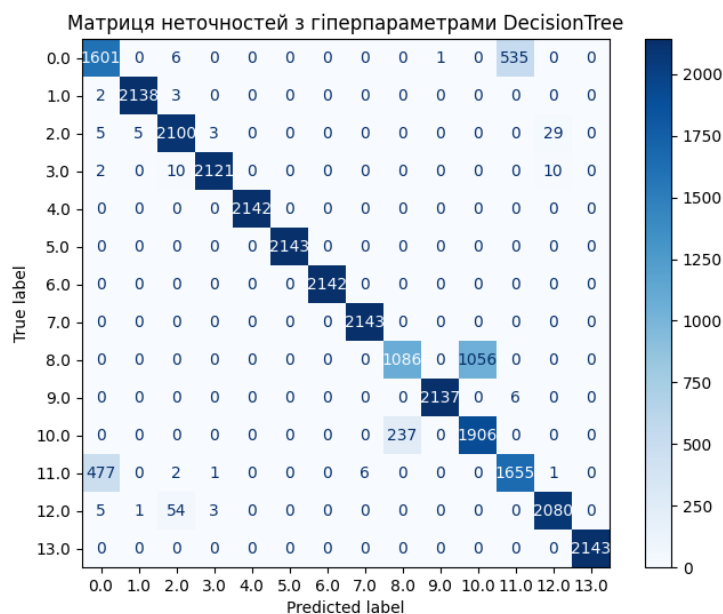


Рис. 3.17. Матриця плутанини Decision Tree

Random Forest

У результаті навчання моделі Decision Tree Було отримано такий результат по основним метрикам (рис. 3.18-3.19).

З отриманих даних можна визначити такі метрики: точність: 0.921, точність відносно позитивних: 0.926, повнота 0.921, f1-score: 0.92.

Загальна точність: 0.9215254858819215
 Час навчання: 14.19 секунд
 Точність відносно позитивних: 0.9262298218730504
 Повнота: 0.9215254858819215
 F1: 0.9199246399513646

	precision	recall	f1-score	support
0.0	0.79	0.77	0.78	2143
1.0	0.99	1.00	1.00	2143
2.0	0.97	0.98	0.98	2142
3.0	0.99	0.99	0.99	2143
4.0	1.00	1.00	1.00	2142
5.0	1.00	1.00	1.00	2143
6.0	1.00	1.00	1.00	2142
7.0	1.00	1.00	1.00	2143
8.0	0.82	0.51	0.63	2142
9.0	1.00	0.99	1.00	2143
10.0	0.64	0.89	0.75	2143
11.0	0.78	0.79	0.78	2142
12.0	0.98	0.98	0.98	2143
13.0	1.00	1.00	1.00	2143
accuracy			0.92	29997
macro avg	0.93	0.92	0.92	29997
weighted avg	0.93	0.92	0.92	29997

Рис. 3.18. Навчання Random Forest

Загальна точність: 0.9235923592359236
 Час навчання: 40.62 секунд
 Точність відносно позитивних: 0.9282572505137976
 Повнота: 0.9235923592359236
 F1: 0.9219985598224318

	precision	recall	f1-score	support
0.0	0.81	0.77	0.79	2143
1.0	1.00	1.00	1.00	2143
2.0	0.98	0.98	0.98	2142
3.0	1.00	0.99	0.99	2143
4.0	1.00	1.00	1.00	2142
5.0	1.00	1.00	1.00	2143
6.0	1.00	1.00	1.00	2142
7.0	1.00	1.00	1.00	2143
8.0	0.82	0.51	0.63	2142
9.0	1.00	1.00	1.00	2143
10.0	0.64	0.88	0.75	2143
11.0	0.78	0.82	0.80	2142
12.0	0.98	0.98	0.98	2143
13.0	1.00	1.00	1.00	2143
accuracy			0.92	29997
macro avg	0.93	0.92	0.92	29997
weighted avg	0.93	0.92	0.92	29997

Рис. 3.19. Навчання Random Forest Про

Після проведеної баєсівської оптимізації було отримано незначні покращення показників. Отримані дані: Точність: 0.923, точність відносно

позитивних: 0.928, повнота: 0.923, f1-score: 0.922. Покращення доволі знову не показують помітних результатів. Покращення відбулось на 0.2%, що є аналогічно значенням похибки.

Побудова ROC-кривих (рис. 3.20) показала гарну збалансованість моделі відносно класів. Гіперпараметризація не вплинула на значення кривою або гіперплощини.

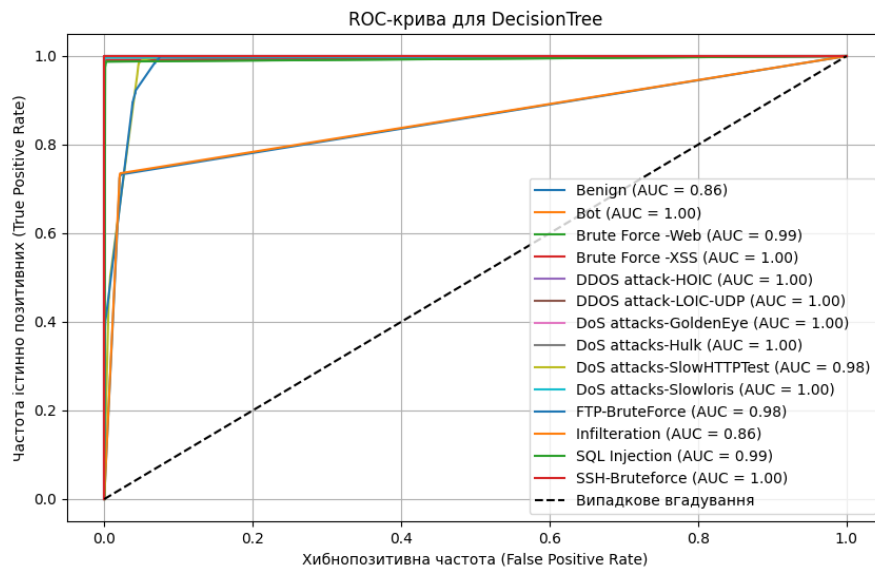


Рис. 3.20. ROC-крива Random Forest

Матриця плутанини в обох випадках залишається майже незмінною (рис 3.21). Модель демонструє гарні результати у визначенні класів. Лише у декількох випадках відхилення становить більше норми.

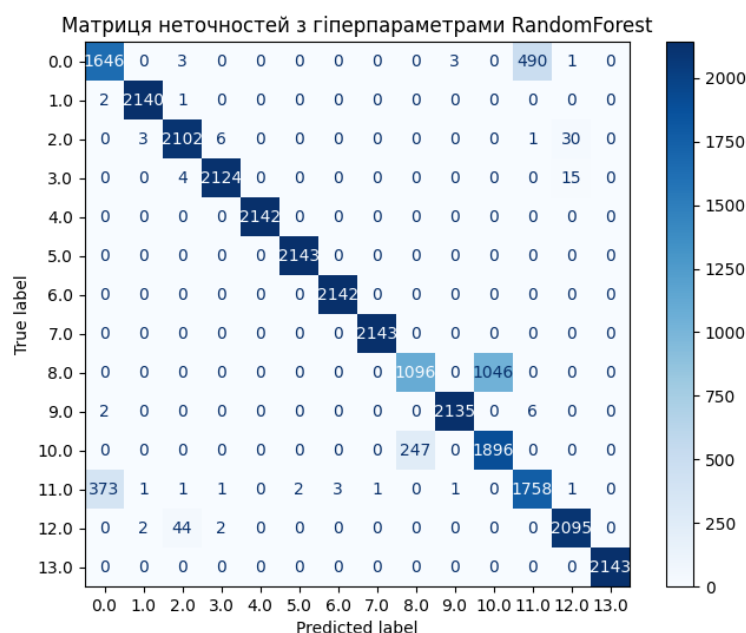


Рис. 3.21. Матриця плутанини Random Forest

SVM

Серед усіх алгоритмів модель SVM навчалась найдовше. Провести гіперпараметризацію даної моделі не вийшло через її вимогливість до ресурсів при великих наборах даних з урахування широкої класифікації.

Загальна точність: 0.8077141047438077				
Час навчання: 199.60 секунд				
Точність відносно позитивних: 0.8604541103508795				
Повнота: 0.8077141047438077				
F1: 0.8036819999305973				
	precision	recall	f1-score	support
0.0	0.71	0.61	0.66	2143
1.0	0.46	1.00	0.63	2143
2.0	0.90	0.67	0.77	2142
3.0	1.00	0.48	0.65	2143
4.0	0.98	1.00	0.99	2142
5.0	1.00	0.99	1.00	2143
6.0	1.00	1.00	1.00	2142
7.0	0.98	1.00	0.99	2143
8.0	0.60	0.99	0.75	2142
9.0	0.99	0.97	0.98	2143
10.0	0.98	0.33	0.49	2143
11.0	0.72	0.58	0.64	2142
12.0	0.75	0.68	0.71	2143
13.0	1.00	1.00	1.00	2143
accuracy			0.81	29997
macro avg	0.86	0.81	0.80	29997
weighted avg	0.86	0.81	0.80	29997

Рис. 3.22. Навчання SVM

З отриманих даних можна визначити такі метрики: точність 0.807, точність відносно позитивних: 0.860, повнота 0.807, f1-score: 0.803. Модель показала дещо гірший результат точності відносно інших моделей. Побудова ROC-кривої показала що модель все ще добре збалансована, хоча і значення гіперплощин дещо нижчі за інші моделі.

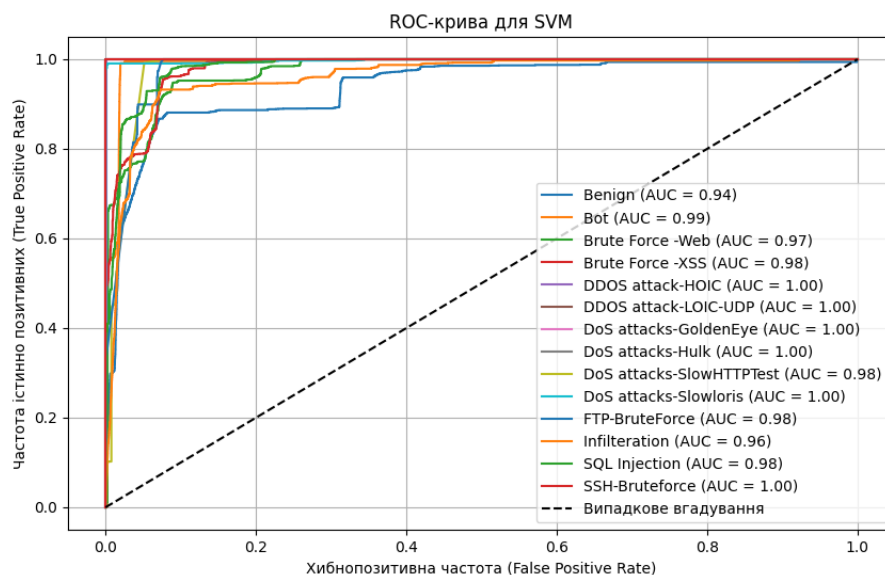


Рис. 3.23. ROC-крива SVM

Матриця неточностей показує наявність певних проблем у моделі з класами, хоча це і не відноситься до всіх (рис. 3.24).

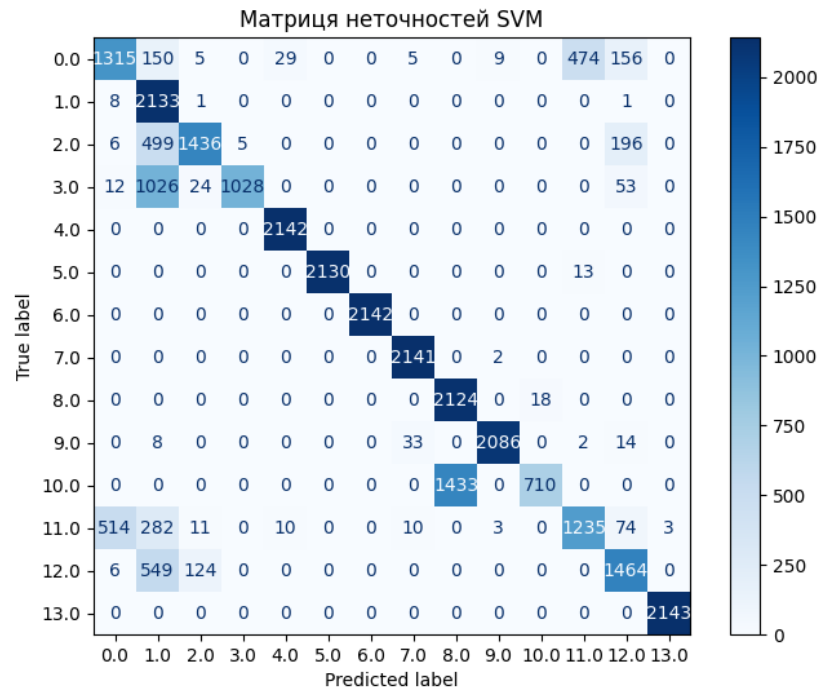


Рис. 3.24. Матриця неточностей SVM

Voting

У результаті навчання ансамблевої моделі Voting було отримано такий результат по основним метрикам (рис. 3.25). З отриманих даних можна визначити такі метрики: точність: 0.919, точність відносно позитивних: 0.934, повнота 0.919, f1-score: 0.915.

```

Загальна точність: 0.9195919591959196
Час навчання: 241.97 секунд
Точність відносно позитивних: 0.9338770612241625
Повнота: 0.9195919591959196
F1: 0.9154186151569546

```

	precision	recall	f1-score	support
0.0	0.81	0.73	0.77	2143
1.0	0.99	1.00	1.00	2143
2.0	0.99	0.96	0.97	2142
3.0	1.00	0.99	0.99	2143
4.0	1.00	1.00	1.00	2142
5.0	1.00	1.00	1.00	2143
6.0	1.00	1.00	1.00	2142
7.0	1.00	1.00	1.00	2143
8.0	0.62	0.98	0.76	2142
9.0	1.00	1.00	1.00	2143
10.0	0.96	0.40	0.56	2143
11.0	0.75	0.83	0.79	2142
12.0	0.96	0.99	0.98	2143
13.0	1.00	1.00	1.00	2143
accuracy			0.92	29997
macro avg	0.93	0.92	0.92	29997
weighted avg	0.93	0.92	0.92	29997

Рис. 3.25. Метрики Voting

Так як було обрано тип вирішення soft, то фінальні передбачення базуються на середніх ймовірностей і будуть нижчі за найкращі моделі. Незважаючи на це, модель показує високі результати.

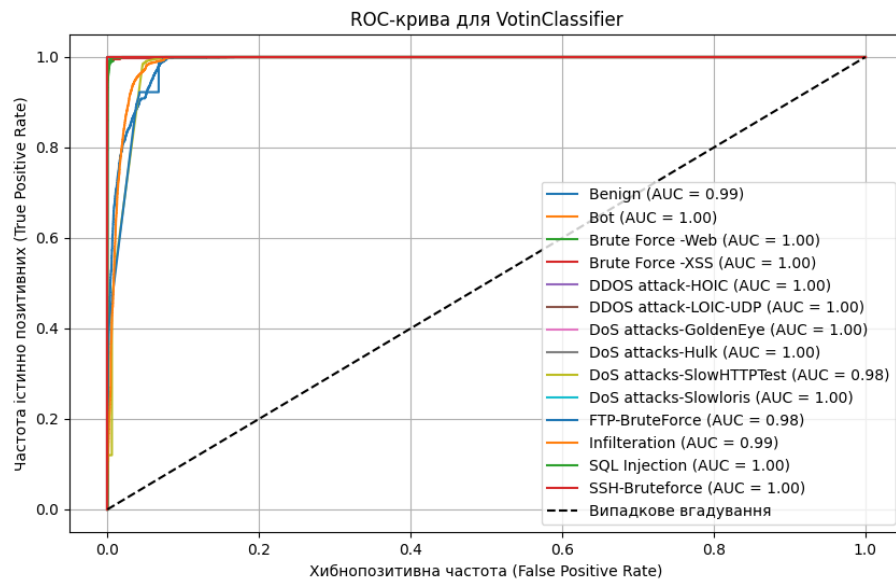


Рис. 3.26. ROC-крива Voting

Побудована ROC-крива демонструє збалансованість моделі (рис. 3.27). Значення гіперплощин не опускаються нижче 0.98.

Матриця неточностей показує наявні проблеми для трьох класів, що є не критичним, але при використанні може давати більше хибнопозитивні передбачення (рис 3.27).

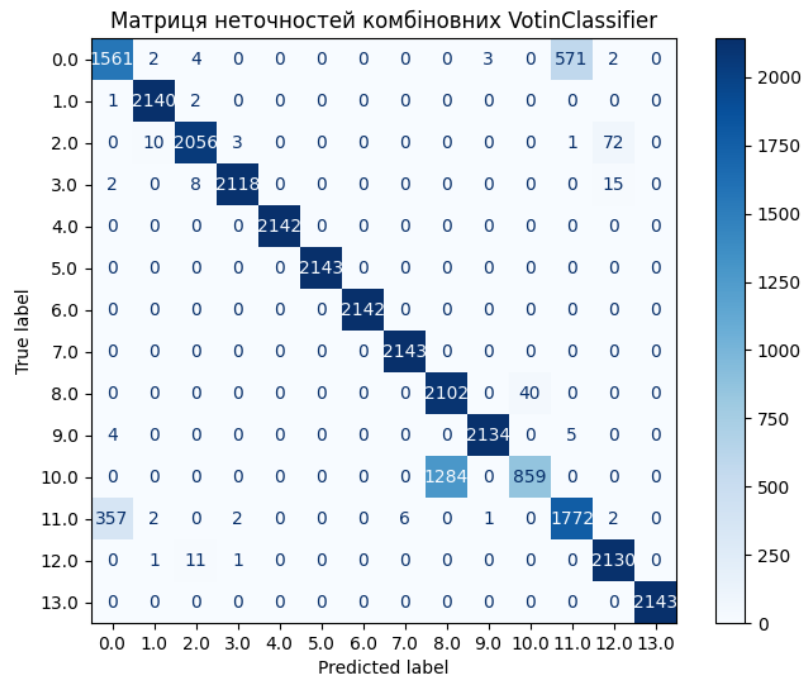


Рис. 3.27. Матриця неточностей Voting

Stacking

У результаті навчання ансамблевої моделі Stacking було отримано такий результат по основним метрикам (рис. 3.28).

З отриманих даних можна визначити такі метрики: точність: 0.924, точність відносно позитивних: 0.929, повнота 0.924, f1-score: 0.922.

Дана модель поки демонструє один із найвищих показників точності відносно інших, хоча і різниця не є великою.

Загальна точність: 0.923992399239924				
Час навчання: 1162.52 секунд				
Точність відносно позитивних: 0.9287539568835889				
Повнота: 0.923992399239924				
F1: 0.922438823129907				
	precision	recall	f1-score	support
0.0	0.81	0.77	0.79	2143
1.0	1.00	1.00	1.00	2143
2.0	0.98	0.98	0.98	2142
3.0	1.00	0.99	0.99	2143
4.0	1.00	1.00	1.00	2142
5.0	1.00	1.00	1.00	2143
6.0	1.00	1.00	1.00	2142
7.0	1.00	1.00	1.00	2143
8.0	0.82	0.51	0.63	2142
9.0	1.00	0.99	1.00	2143
10.0	0.64	0.88	0.75	2143
11.0	0.78	0.82	0.80	2142
12.0	0.98	0.98	0.98	2143
13.0	1.00	1.00	1.00	2143
accuracy			0.92	29997
macro avg	0.93	0.92	0.92	29997
weighted avg	0.93	0.92	0.92	29997

Рис. 3.28. Метрики Stacking

Так як було обрано тип вирішення soft, то фінальні передбачення базуються на середніх ймовірностей і будуть нижчі за найкращі моделі. Незважаючи на це, модель показує високі результати.

Побудова ROC-кривою показала збалансованість моделі. Результати аналогічно з попередньою моделлю Voting (рис. 3.29).

Матриця неточностей показує більшу кількість хибнопозитивних передбачень відносно Voting моделі, незважаючи на точність (рис 3.30).

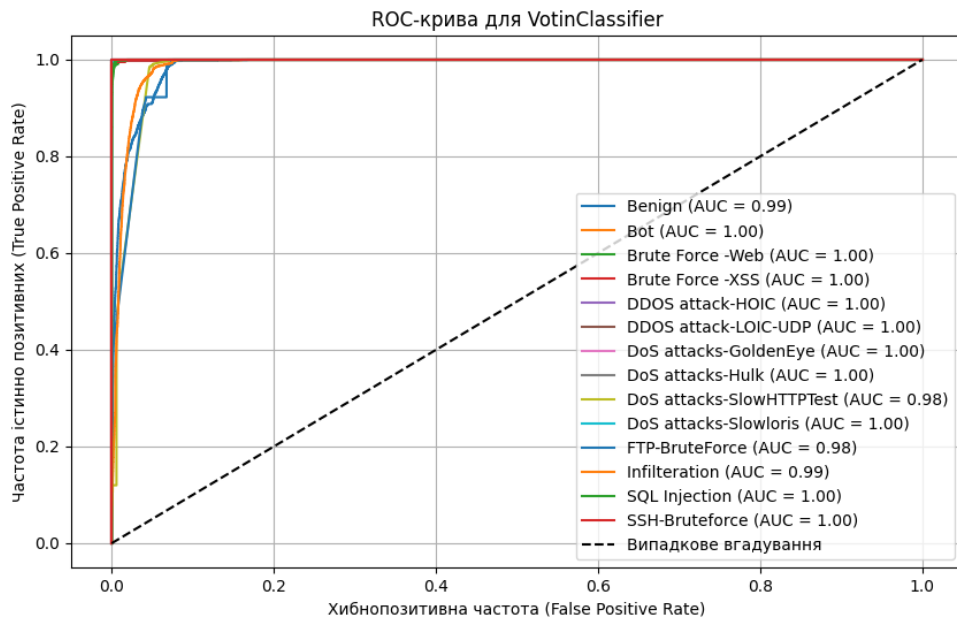


Рис. 3.29. ROC-крива Stacking

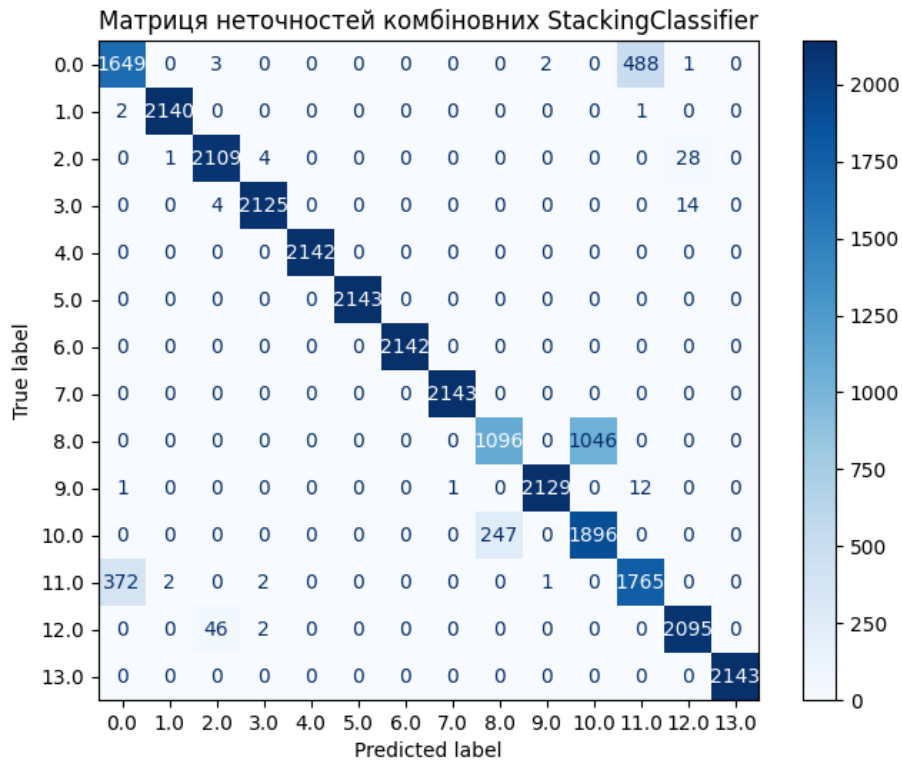


Рис. 3.30. Матриця неточностей Stacking

Попередньо порівнявши моделі можна визначити які на даному етапі мають кращі показники (рис. 3.31).

На даному етапі найкращі показники мають моделі Random Forest та Stacking. Матриці неточностей мають схожі патерни проблем, що може свідчити про не ідеально сформований набір даних.



Рис. 3.31. Порівняння точності

LSTM

У результаті навчання та оптимізації моделі LSTM було отримано такі результати (рис. 3.32-3.33).

	precision	recall	f1-score	support
0	0.88	0.85	0.87	2132
1	1.00	1.00	1.00	2143
2	0.99	0.98	0.99	183
3	0.99	0.99	0.99	69
4	1.00	1.00	1.00	2143
5	1.00	1.00	1.00	519
6	1.00	1.00	1.00	2142
7	1.00	1.00	1.00	2143
8	0.88	0.92	0.90	2143
9	1.00	1.00	1.00	2143
10	0.91	0.87	0.89	2142
11	0.86	0.89	0.87	2112
12	0.89	0.96	0.93	26
13	1.00	1.00	1.00	2143
accuracy			0.95	22183
macro avg	0.96	0.96	0.96	22183
weighted avg	0.95	0.95	0.95	22183

Рис. 3.32. Навчання LSTM

	precision	recall	f1-score	support
0	0.82	0.61	0.70	2132
1	0.93	0.98	0.95	2143
2	0.94	0.96	0.95	183
3	0.83	0.94	0.88	69
4	1.00	1.00	1.00	2143
5	1.00	0.99	1.00	519
6	1.00	1.00	1.00	2142
7	1.00	1.00	1.00	2143
8	0.85	0.96	0.90	2143
9	1.00	1.00	1.00	2143
10	0.95	0.83	0.89	2142
11	0.73	0.87	0.80	2112
12	0.31	0.81	0.45	26
13	1.00	1.00	1.00	2143
accuracy			0.93	22183
macro avg	0.88	0.93	0.89	22183
weighted avg	0.93	0.93	0.93	22183

Рис. 3.33. Навчання LSTM Про

З отриманих даних можна визначити такі метрики моделі: точність: 0.954, точність відносно позитивних: 0.954, повнота: 0.954, f1-score: 0.95.

Після оптимізації гіперпараметрів результати зменшились: точність: 0.926, точність відносно позитивних: 0.935, повнота: 0.915, f1-score: 0.93.

Погіршення показників моделі може бути пов'язано із перенавчаннями, під час оптимізації модель могла стати занадто складною або процес оптимізації параметрів вибрав параметри, які не підходять для даної задачі.

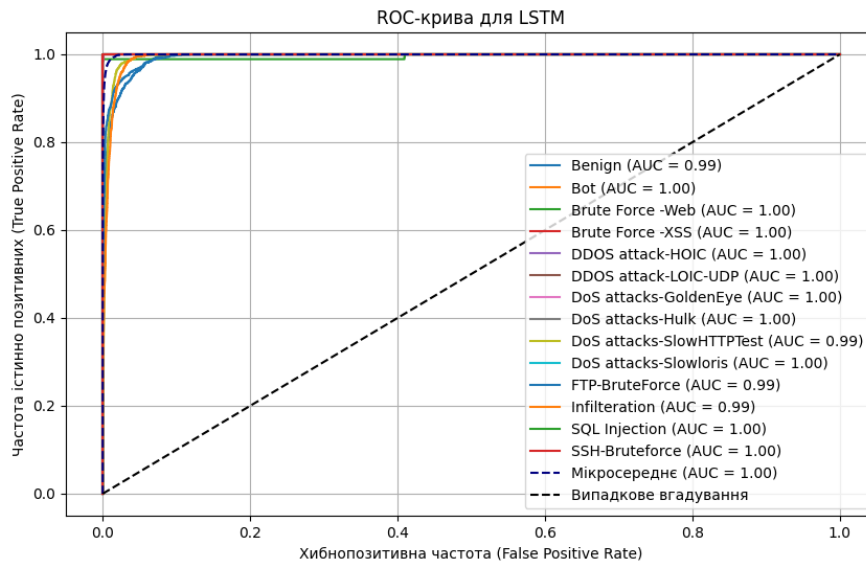


Рис. 3.34. ROC-крива LSTM

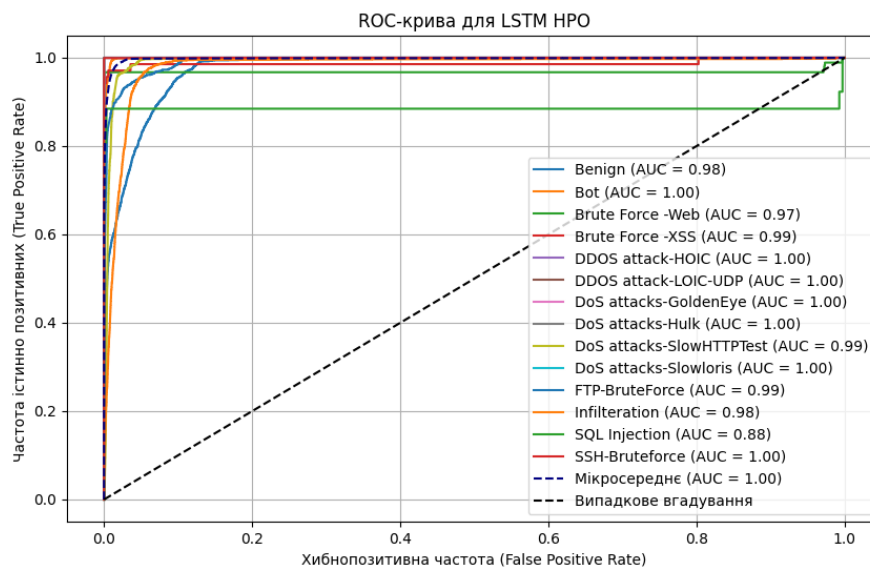


Рис. 3.35. ROC-крива LSTM Нро

Аналіз ROC кривих також дає аналогічно результати де оптимізація погіршує якість моделі (рис. 3.34-3.35). Незважаючи на це звичайна модель має стабільні, майже ідеальні, значення для всіх класів.

При аналізі нейронних мереж є можливість аналізувати процес навчання та визначити його особливості. Графіки навчання моделі задані у додатку Б.

Проаналізувавши матрицю неточностей можна спостерігати за присутнім класовим дисбалансом для декількох класів та можливих неточностей у передбаченні (рис. 3.36)

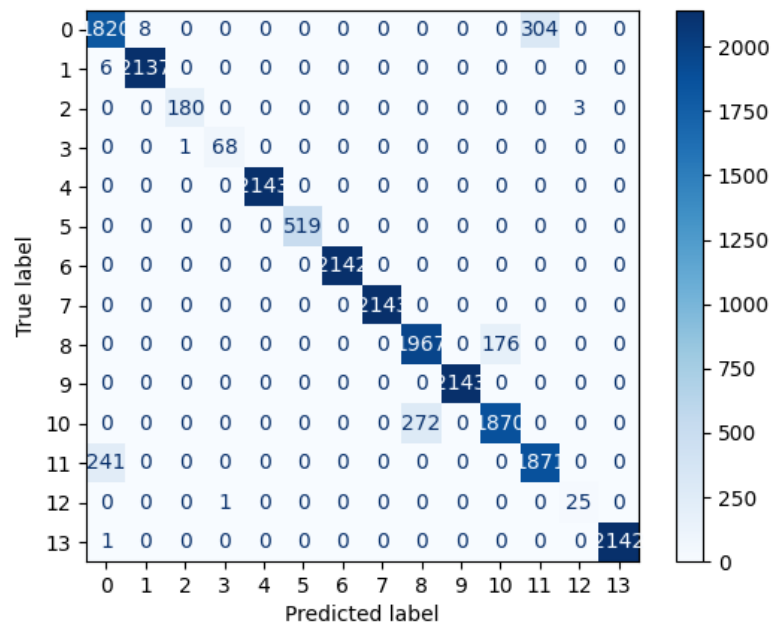


Рис. 3.36. Матриця неточностей LSTM

DNN + KNN

У результаті навчання моделі DNN+KNN було отримано такий результат по основним метрикам (рис. 3.37).

Модель DNN продемонструвала такі характеристики: точність: 0.866, точність відносно позитивних: 0.871, повнота: 0.862, f1-score: 0.843.

```

Загальна точність: 0.8965896589658966
Час навчання: 1.61 секунд
Точність відносно позитивних: 0.9082527911520242
Повнота: 0.8965896589658966
F1: 0.8929996033372729

```

	precision	recall	f1-score	support
0	0.80	0.71	0.75	2143
1	0.96	0.97	0.97	2143
2	0.91	0.85	0.88	2142
3	0.86	0.88	0.87	2143
4	1.00	1.00	1.00	2142
5	1.00	1.00	1.00	2143
6	1.00	1.00	1.00	2142
7	1.00	1.00	1.00	2143
8	0.62	0.96	0.75	2142
9	1.00	0.99	1.00	2143
10	0.91	0.42	0.57	2143
11	0.74	0.82	0.78	2142
12	0.92	0.93	0.92	2143
13	1.00	1.00	1.00	2143
accuracy			0.90	29997
macro avg	0.91	0.90	0.89	29997
weighted avg	0.91	0.90	0.89	29997

Рис. 3.37. Метрики DNN+KNN

Після витягування ознак та тренування моделі було отримано такий результат гібридної моделі: точність: 0.896, точність відносно позитивних: 0.908, повнота 0.896, f1-score: 0.892.

Після оптимізації гіперпараметрів було отримано такі результати (рис. 3.39).

Загальна точність: 0.9022235556889022				
Час навчання: 11.00 секунд				
Точність відносно позитивних: 0.9161362734501564				
Повнота: 0.9022235556889022				
F1: 0.8981340682849062				
	precision	recall	f1-score	support
0	0.80	0.73	0.76	2143
1	0.98	0.98	0.98	2143
2	0.92	0.89	0.91	2142
3	0.89	0.90	0.90	2143
4	1.00	1.00	1.00	2142
5	1.00	1.00	1.00	2143
6	1.00	1.00	1.00	2142
7	1.00	1.00	1.00	2143
8	0.62	0.98	0.76	2142
9	1.00	1.00	1.00	2143
10	0.95	0.40	0.57	2143
11	0.75	0.82	0.78	2142
12	0.92	0.94	0.93	2143
13	1.00	1.00	1.00	2143
accuracy			0.90	29997
macro avg	0.92	0.90	0.90	29997
weighted avg	0.92	0.90	0.90	29997

Рис. 3.38. Метрики DNN+KNN Про

Показники: точність: 0.902, точність відносно позитивних: 0.916, повнота 0.902, f1-score: 0.898.

Очікування від моделі були більшими. Одним із можливих рішень може бути оптимізація саме нейронної мережі для кращих результатів.

Матриця неточностей демонструє можливі похибки у передбаченні до класів (рис. 3.39).

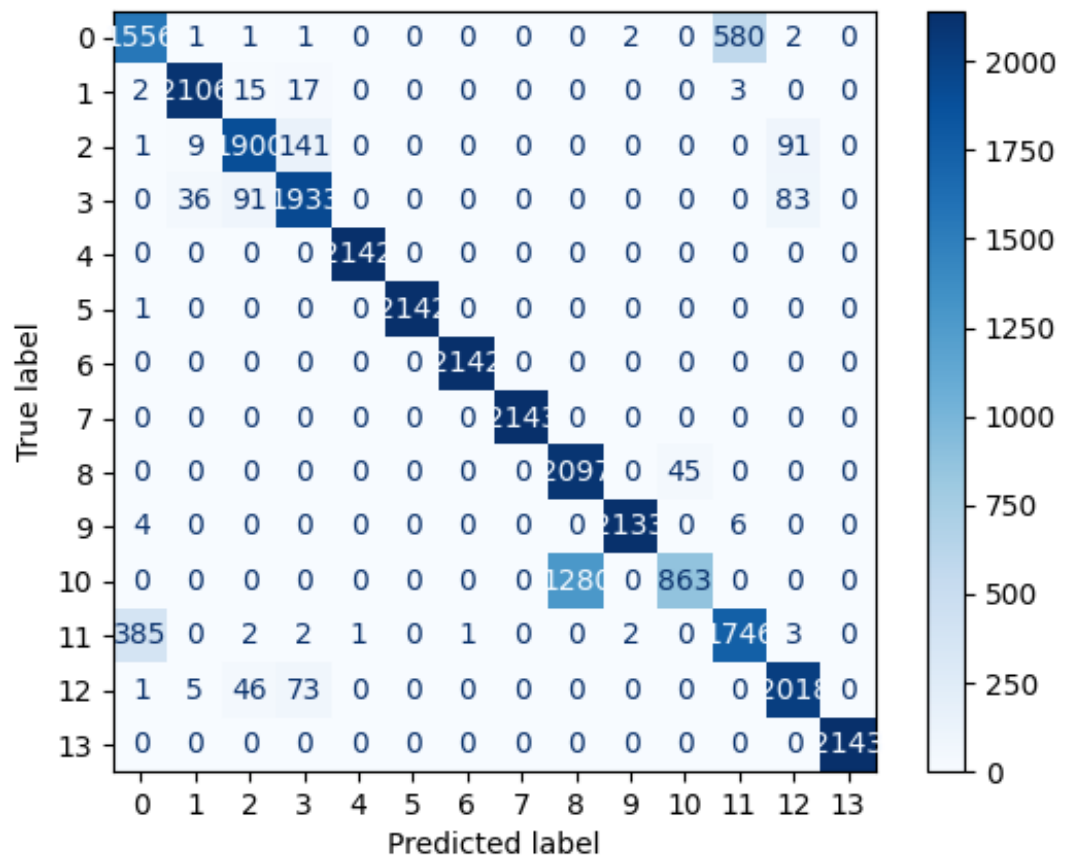


Рис. 3.39. Матриця неточностей DNN+KNN

Також було проаналізовано ROC-криві які показують як KNN покращує DNN (рис. 3.40-3.41).

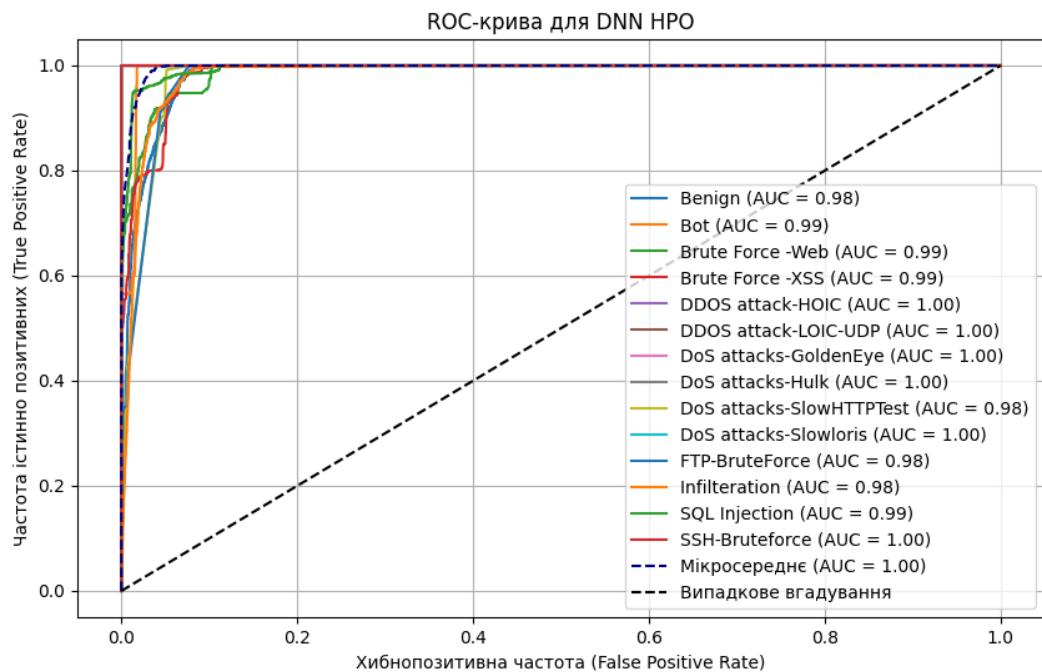


Рис. 3.40. ROC-крива DNN

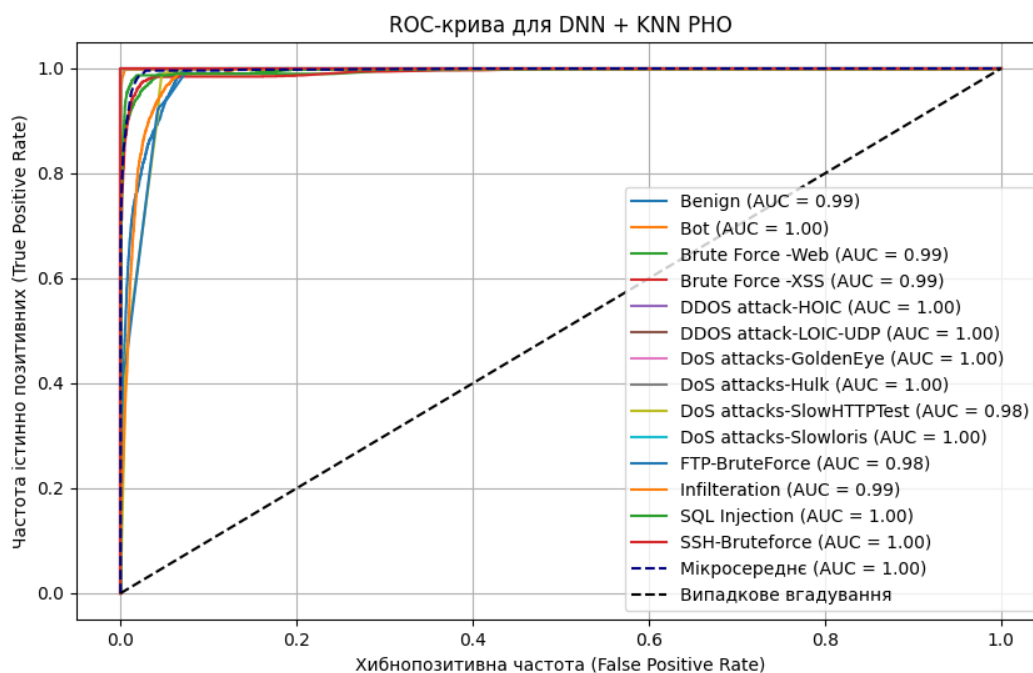


Рис. 3.41. ROC-крива DNN + KNN

Проаналізувавши розроблені алгоритми машинного навчання було визначено, що більшість показує гарні результати точності. Всі моделі є добре збалансованими. Для порівняння було створено таблицю з усіма метриками.

Таблиця 3.1

Порівняльна таблиця базових метрик моделей

Модель	Точність (Accuracy)	Точність відносно позитивних (Precision)	Повнота (Recall)	F1-score
Decision Tree	0.914	0.919	0.914	0.913
Decision Tree Hpo	0.918	0.923	0.918	0.916
Random Forest	0.921	0.926	0.921	0.92
Random Forest Hpo	0.924	0.928	0.923	0.922
SVM	0.808	0.86	0.808	0.804
Voting	0.916	0.934	0.916	0.915
Stacking	0.924	0.929	0.924	0.922
DNN+KNN	0.896	0.908	0.896	0.893
DNN+KNN Hpo	0.902	0.916	0.902	0.898
LSTM	0.954	0.954	0.954	0.955
LSTM Hpo	0.926	0.935	0.915	0.93

У результаті проведеного аналізу різних моделей було виявлено, що LSTM без гіперпараметризації показала найкращі результати за всіма метриками. Моделі Voting, Stacking та Random Forest Нро також продемонстрували високі показники, підтверджуючи ефективність ансамблевих методів. Гіперпараметризація загалом покращила результати моделей, окрім LSTM, де показники погіршилися. SVM показала найнижчі результати, що свідчить про потребу додаткової оптимізації для даного завдання. Загальний висновок підтверджує, що використання ансамблевих методів та нейронних мереж є найбільш ефективним для класифікації та виявлення аномалій у веб-трафіку.

Висновки до третього розділу

У третьому розділі випускної кваліфікаційно роботи було проведено роботу з реалізації алгоритмів класифікації для виявлення аномальної поведінки користувачів.

Спочатку було розібрано структуру отриманого датасету, проведено всі необхідні маніпуляції для формування даних, які відповідають вимогам для навчання моделей. Було сформовано два датасети, які містили у собі нормалізовані дані та дані з часовими характеристиками.

Потім було описано моделювання та навчання моделей. Були продемонстровані підходи до побудови та навчання моделей. Також було продемонстровано способи оптимізації для отримання кращого результату.

Як результат було проведено детальний аналіз ефективності моделей машинного навчання для класифікації аномалій у веб-трафіку. Використовувалися стандартні метрики (точність, точність відносно позитивних, повнота, F1-score), ROC-криві та матриці плутанини для візуалізації результатів. Було розглянуто як початкові моделі, так і моделі після гіперпараметризації для оцінки впливу оптимізації параметрів. Ансамблеві методи (Voting і Stacking) показали високу ефективність.

ВИСНОВКИ

Під час виконання кваліфікаційної магістерської роботи було детально досліджено особливості аномалій у мережі інтернет та їх виявлення за допомогою алгоритмів машинного навчання та нейронних мереж. Результатом роботи стало сформовані дані та готові до використання навчені та оптимізовані моделі під різні вимоги. Основними завданнями було проведення досліджень характеристик для виявлення аномалій та їх класифікації, аналіз архітектурних рішень для реалізації моделей машинного навчання та їх реалізація з подальшою оцінкою та порівнянням.

У першому розділі було проаналізовано завдання, визначено основні вимоги та поставлені чіткі цілі для роботи. Було досліджено запити, потоки даних та структуру веб-трафіку, що дозволило визначити характеристики для класифікації поведінки користувачів. Розгляд алгоритмів машинного навчання включав вивчення їх переваг, недоліків та можливостей вирішення завдань, а також розгляд нейронних мереж глибокого навчання. На основі цього було складено список моделей для реалізації. Також вибрано і проаналізовано інструменти для ефективного виконання цілей.

У другому розділі було детально проаналізовано архітектури алгоритмів, визначено основні етапи розробки та виконано всі необхідні кроки для успішної реалізації. Було досліджено архітектури моделей машинного навчання, зокрема Decision Tree, Random Forest, SVM, Voting і Stacking, а також нейронні мережі глибокого навчання DNN та LSTM. Для створення гібридної моделі обрано комбінацію нейронної мережі та алгоритму KNN. Було обрано набір даних CSE-CIC-IDS 2018 для моделювання ефективних моделей. Було проаналізовано методи оптимізації моделей та необхідні метрики для оцінки і порівняння результатів.

У другому розділі було детально проаналізовано архітектури алгоритмів, визначено основні етапи розробки та виконано всі необхідні кроки для успішної реалізації. Було досліджено архітектури моделей машинного

навчання, зокрема Decision Tree, Random Forest, SVM, Voting і Stacking, а також нейронні мережі глибокого навчання DNN та LSTM. Для створення гібридної моделі обрано комбінацію нейронної мережі та алгоритму KNN. Було обрано набір даних CSE-CIC-IDS 2018 для моделювання ефективних моделей. Було проаналізовано методи оптимізації моделей та необхідні метрики для оцінки і порівняння результатів.

У третьому розділі було здійснено реалізацію алгоритмів класифікації для виявлення аномальної поведінки користувачів. Спочатку проаналізовано структуру датасету та виконано всі необхідні маніпуляції для підготовки даних до навчання. Було сформовано два датасети, один з нормалізованими даними, інший з часовими характеристиками. Далі описано процес моделювання та навчання моделей, включаючи демонстрацію підходів до їх побудови та оптимізації. Виконано детальний аналіз ефективності моделей машинного навчання для класифікації аномалій у веб-трафіку, використовуючи стандартні метрики, ROC-криві та матриці плутанини для візуалізації результатів. Розглянуто початкові моделі та моделі після гіперпараметризації для оцінки впливу оптимізації параметрів.

Як результат, у випускній кваліфікаційній роботі було здійснено всебічний аналіз та реалізацію алгоритмів класифікації для виявлення аномальної поведінки користувачів вебсайтів. Система виявлення аномалій має великий потенціал для подальшого вдосконалення та адаптації, забезпечуючи інструмент для моніторингу та захисту веб-трафіку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Клименко Д.О., Чижмотря О.В. Особливості інтеграції алгоритмів машинного навчання для забезпечення вебсайтів. Комп'ютерні технології: інновації, проблеми, рішення: тези VII Всеукраїнської науково-технічної конференції, м. Житомир, 02-03 грудня 2024 р. – Житомир: Державний університет «Житомирська політехніка», 2024
2. Клименко Д.О., Чижмотря О.В. Аналіз поведінкових характеристик для виявлення аномальної активності на вебсайтах. Комп'ютерні технології: інновації, проблеми, рішення: тези VII Всеукраїнської науково-технічної конференції, м. Житомир, 02-03 грудня 2024 р. – Житомир: Державний університет «Житомирська політехніка», 2024
3. Data Anomaly Detection in the Internet of Things [Електронний ресурс] // The Science and Information (SAI) Organization. – Режим доступу: https://thesai.org/Downloads/Volume14No9/Paper_1-Data_Anomaly_Detection_in_the_Internet_of_Things.pdf
4. HTTP messages - HTTP | MDN [Електронний ресурс] // MDN Web Docs. – Режим доступу: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages>
5. Chiradeep. Differences between TCP and UDP [Електронний ресурс] / Chiradeep // Spiceworks Inc. – Режим доступу: <https://www.spiceworks.com/tech/networking/articles/tcp-vs-udp/>
6. Understanding TCP flags: site24x7 [Електронний ресурс] // Site24x7. – Режим доступу: <https://www.site24x7.com/learn/linux/tcp-flags.html>
7. HTTP headers - HTTP | MDN [Електронний ресурс] // MDN Web Docs. – Режим доступу: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>.
8. Song, W., Beshley, M., Przystupa, K., Beshley, H., Kochan, O., Pryslupskyi, A., Pieniak, D., & Su, J. (2020). A Software Deep Packet Inspection System for Network Traffic Analysis and Anomaly Detection. *Sensors*, 20(6), 1637 [Електронний ресурс] / Song, W., Beshley, M., Przystupa, K., Beshley, H.,

Kochan, O., Pryslupskyi, A., Pieniak, D., & Su, J. // MDPI. – 2020 - Режим доступа: <https://doi.org/10.3390/s20061637>

9. RFC 793: Transmission Control Protocol [Электронный ресурс] // IETF Datatracker. – Режим доступа: <https://datatracker.ietf.org/doc/html/rfc793>.

10. What are sessions? How do they work? | baeldung on computer science [Электронный ресурс] // Baeldung on Computer Science. – Режим доступа: <https://www.baeldung.com/cs/web-sessions>.

11. R. Vinayakumar, K. P. Soman and P. Poornachandran, "Applying deep learning approaches for network traffic prediction," *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Udupi, India, 2017, pp. 2353-2358, doi: 10.1109/ICACCI.2017.8126198 [Электронный ресурс] / R. Vinayakumar, K. P. Soman and P. Poornachandran // ResearchGate – 2017. – Режим доступа до ресурсу: https://www.researchgate.net/publication/321502679_Applying_deep_learning_approaches_for_network_traffic_prediction

12. Edosa Osa, Patience E. Orukpe, Usiholo Iruansi, “Design and implementation of a deep neural network approach for intrusion detection systems”, e-Prime - Advances in Electrical Engineering, Electronics and Energy, Volume 7, 2024, ISSN 2772-6711 [Электронный ресурс] / Edosa Osa, Patience E. Orukpe, Usiholo Iruansi // ScienceDirect – 2024 – Режим доступа: <https://doi.org/10.1016/j.prime.2024.100434>

13. Xingwei Zhang, Xiaolong Zheng, Desheng Dash Wu, “Attacking DNN-based Intrusion Detection Models”, IFAC-PapersOnLine, Volume 53, Issue 5, 2020, Pages 415-419, ISSN 2405-8963 [Электронный ресурс] / Xingwei Zhang, Xiaolong Zheng, Desheng Dash Wu // ScienceDirect – 2020 — Режим доступа: <https://doi.org/10.1016/j.ifacol.2021.04.118>

14. Venkat Venkatasubramanian, N. Sanjeevrajan, Manasi Khandekar, Abhishek Sivaram, Collin Szczepanski, “Jaynes machine: The universal microstructure of deep neural networks”, Computers & Chemical Engineering, Volume 192, 2024, ISSN 0098-1354 [Электронный ресурс] / Venkat

Venkatasubramanian, N. Sanjeevrajan, Manasi Khandekar, Abhishek Sivaram, Collin Szczepanski // ScienceDirect – 2024 — Режим доступа: <https://doi.org/10.1016/j.compchemeng.2024.108908>

15. Chen,S. (2024).The Gaussian distribution: Derivation method and its applications in selected examples.Theoretical and Natural Science,38,196-202. [Электронный ресурс] / Chen,S. / ResearchGate – 2024. – Режим доступа до ресурсу:

https://www.researchgate.net/publication/381680192_The_Gaussian_distribution_Derivation_method_and_its_applications_in_selected_examples

16. Anomaly Detection Using AI & Machine Learning | Nile [Электронный ресурс] // Nile. – Режим доступа: <https://nilesecure.com/ai-networking/anomaly-detection-ai>

17. Naive bayes classifiers - geeksforgeeks [Электронный ресурс] // GeeksforGeeks. – Режим доступа: <https://www.geeksforgeeks.org/naive-bayes-classifiers/>

18. All about decision trees | Abhishek Jain [Электронный ресурс] // Medium – Режим доступа: <https://medium.com/@abhishekjainindore24/all-about-decision-trees-80ea55e37fef>

19. IBM. What is the k-nearest neighbors algorithm? | IBM [Электронный ресурс] / IBM // IBM - United States. – Режим доступа: <https://www.ibm.com/think/topics/knn>.

20. Lutins E. Ensemble methods in machine learning: what are they and why use them? [Электронный ресурс] / Evan Lutins // Medium. – Режим доступа: <https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f>.

21. Multiclass classification using support vector machines | baeldung on computer science [Электронный ресурс] // Baeldung on Computer Science. – Режим доступа: <https://www.baeldung.com/cs/svm-multiclass-classification>.

22. An Intuitive Explanation of LSTM | Ottavio Calzone [Электронный ресурс] // Medium – Режим доступа: <https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c>

23. Predicting Sequential Data using LSTM: An Introduction | George V Jose [Электронный ресурс] // Medium – Режим доступа: <https://towardsdatascience.com/time-series-forecasting-with-recurrent-neural-networks-74674e289816>

24. Use a GPU | TensorFlow Core [Электронный ресурс] // TensorFlow. – Режим доступа: <https://www.tensorflow.org/guide/gpu>.

25. QuickStart | vast.ai [Электронный ресурс] // Vast AI. – Режим доступа: <https://vast.ai/docs/overview/quickstart>.

26. Keras documentation: Keras 3 API documentation [Электронный ресурс] // Keras: Deep Learning for humans. – Режим доступа: <https://keras.io/api/>.

27. Random forest: a complete guide for machine learning | built in [Электронный ресурс] // Built In. – Режим доступа: <https://builtin.com/data-science/random-forest-algorithm>.

28. Patel M. Understanding the rectified linear unit (relu): a key activation function in neural networks [Электронный ресурс] / Meet Patel // Medium. – Режим доступа: <https://medium.com/@meetkp/understanding-the-rectified-linear-unit-relu-a-key-activation-function-in-neural-networks-28108fba8f07>.

29. Hybrid deep learning approach to improve classification of low-volume high-dimensional data - BMC Bioinformatics [Электронный ресурс] // BioMed Central. – Режим доступа: <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-023-05557-w>.

30. GeeksforGeeks. Voting Classifier - GeeksforGeeks [Электронный ресурс] / GeeksforGeeks // GeeksforGeeks. – Режим доступа: <https://www.geeksforgeeks.org/voting-classifier/>.

31. Eskandar S. Introduction to RBF SVM: a powerful machine learning algorithm for non-linear data [Электронный ресурс] / Sahel Eskandar // Medium. – Режим доступа: <https://medium.com/@eskandar.sahel/introduction-to-rbf-svm-a-powerful-machine-learning-algorithm-for-non-linear-data-1d1cfb55a1a>.

32. DeepAI. Hidden layer [Электронный ресурс] / DeepAI // DeepAI. – Режим доступа: <https://deepai.org/machine-learning-glossary-and-terms/hidden-layer-machine-learning>.

33. IDS 2018 | datasets | research | canadian institute for cybersecurity | UNB [Электронный ресурс] // University of New Brunswick | UNB. – Режим доступа: <https://www.unb.ca/cic/datasets/ids-2018.html>. – Назва з екрана.

34. IDS 2018 Intrusion CSVs [Электронный ресурс] // Kaggle – Режим доступа: <https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv>

35. Viadinugroho R. A. A. Imbalanced classification in python: SMOTE-ENN method [Электронный ресурс] / Raden Aurelius Andhika Viadinugroho // Medium. – Режим доступа: <https://towardsdatascience.com/imbalanced-classification-in-python-smote-enn-method-db5db06b8d50>.

36. Wang W. Bayesian optimization concept explained in layman terms [Электронный ресурс] / Wei Wang // Medium. – Режим доступа: <https://towardsdatascience.com/bayesian-optimization-concept-explained-in-layman-terms-1d2bcdeaf12f>

37. Performance metrics in machine learning [complete guide] - neptune.ai [Электронный ресурс] // neptune.ai. – Режим доступа: <https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide>

38. Nisar R. (Visually) Interpreting the confusion-matrix: [Электронный ресурс] / Riddhi Nisar // Medium. – Режим доступа: <https://medium.com/analytics-vidhya/visually-interpreting-the-confusion-matrix-787a70b65678>

39. Narkhede S. Understanding AUC - ROC curve [Электронный ресурс] / Sarang Narkhede // Medium. – Режим

доступу: <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>

40. Automatic hyperparameter optimization with keras tuner | digitalocean
[Электронный ресурс] // DigitalOcean | Cloud Infrastructure for Developers. –
Режим
доступу: <https://www.digitalocean.com/community/tutorials/hyperparameter-optimization-with-keras-tuner>.

ДОДАТКИ

Лістинг алгоритму обробки набору даних

```

# Встановлення цільового максимуму записів для великих класів
target_samples_per_class = int(100000 / len(df['Label'].unique()))
class_counts = df['Label'].value_counts()
balanced_dfs = []
for category, count in class_counts.items():
    class_df = df[df['Label'] == category]
    if count > target_samples_per_class:
        reduced_class_df = class_df.iloc[:target_samples_per_class]
    else:
        reduced_class_df = class_df
    balanced_dfs.append(reduced_class_df)
df = pd.concat(balanced_dfs)
df["Timestamp"] = pd.to_datetime(df["Timestamp"], format='mixed') #якщо LSTM
df = df.dropna(subset=["Timestamp"])
df["Timestamp"] = pd.to_datetime(df["Timestamp"]).apply(lambda x:
int(x.timestamp()))
df.sort_values(by='Timestamp', inplace=True)
df.drop(['Dst Port', 'Timestamp', 'Protocol'], axis=1, inplace=True)
# df.drop(['Dst Port', 'Protocol'],axis=1, inplace=True) # LSTM
df.dropna(inplace=True) # видалення нульових значень
df = df[~(df.isin([np.nan, np.inf, -np.inf]).any(axis=1))]
le = LabelEncoder()
le.fit(df['Label'])
mapping = dict(zip(le.classes_, le.transform(le.classes_)))
df["Label"] = le.transform(df["Label"])
df_without_labels = df.drop('Label', axis=1, inplace=False)
df_columns = df_without_labels.columns.tolist()
df_without_labels.head()
df_s_scaled = StandardScaler().fit_transform(df_without_labels)
df_s_scaled = pd.DataFrame(data = df_s_scaled, columns = df_columns)
df_s_scaled.head()
df_s_scaled['Label'] = df['Label'].reset_index(drop=True)
df_s_scaled.head()
X = df_s_scaled.drop('Label', axis=1)
y = df_s_scaled['Label']
smote = SMOTE(sampling_strategy='auto')
X_res, y_res = smote.fit_resample(X, y)
df_balanced = pd.DataFrame(data=np.column_stack([X_res, y_res]),
columns=df_s_scaled.columns)

```

Графіки навчання нейронної мережі LSTM

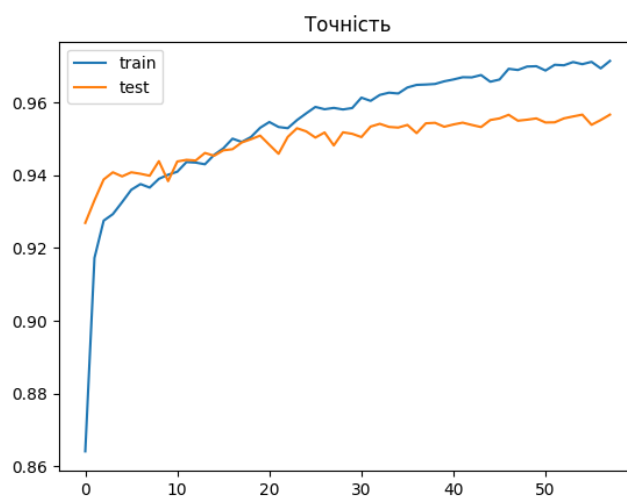


Рис. Б.1. Графік зміни точності



Рис. Б.2. Графік зміни точності відносно хибнопозитивних

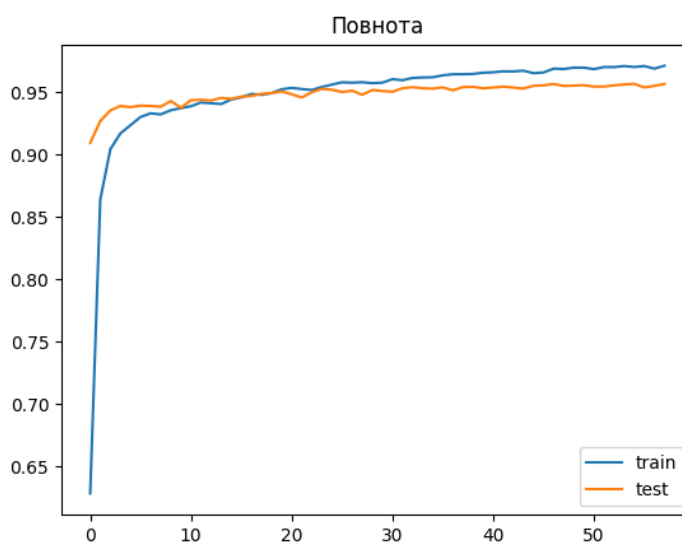


Рис. Б.3. Графік зміни повноти

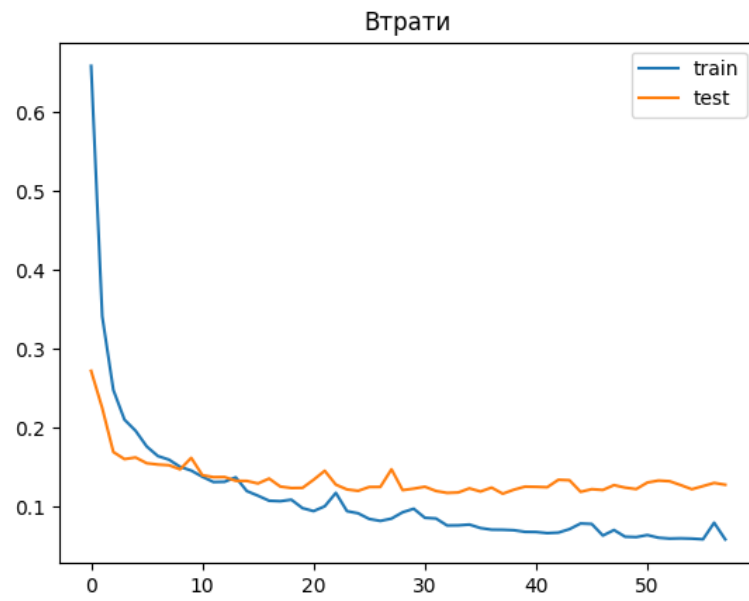


Рис. Б.4. Графік зміни втрат