**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ**
**ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
**КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

# КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Метод виявлення вразливостей web-додатків до SQL-ін'єкцій з використанням методів машинного навчання»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

*Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело*

_____ Анна ФЕДОРЕНКО
*(підпис)*

Виконав: здобувач вищої освіти групи ПДМ-63
Анна ФЕДОРЕНКО

Керівник: Віталій САВЧЕНКО
*д.т.н., професор*

Рецензент: _____
*науковий ступінь, вчене звання* *Ім'я, ПРІЗВИЩЕ*

**Київ 2025**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**

**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**Навчально-науковий інститут інформаційних технологій**

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти Магістр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення»

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2024 р.

# ЗАВДАННЯ
# НА КВАЛІФІКАЦІЙНУ РОБОТУ

Федоренко Анні Андріївні

1. Тема кваліфікаційної роботи: «Метод виявлення вразливостей web-додатків до SQL-ін'єкцій з використанням методів машинного навчання»

керівник кваліфікаційної роботи Віталій САВЧЕНКО, д.т.н., професор,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024 р. № 320.

2. Строк подання кваліфікаційної роботи «17» грудня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, дані про SQL-запити, методи та алгоритми машинного навчання, інструменти динамічного аналізу.

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

1. Дослідження методів виявлення SQL-ін’єкцій
2. Дослідження методів та алгоритмів машинного навчання для виявлення SQL-ін’єкцій
3. Розробка методу виявлення вразливостей web-додатків до SQL-ін’єкцій з використанням методів машинного навчання

5. Перелік ілюстративного матеріалу: *презентація*

1. Мета, об’єкт та предмет дослідження
2. Методи виявлення SQL-ін’єкцій
3. Порівняння алгоритмів машинного навчання для виявлення SQL-ін’єкцій
4. Математична модель
5. Математична модель (продовження)
6. Алгоритм роботи методу
7. Практичний результат
8. Практичний результат (продовження)
9. Порівняльний аналіз методів виявлення
10. Висновки
11. Публікації та апробація роботи

6. Дата видачі завдання «16» жовтня 2024 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз наявної науково-технічної літератури | 16.10-20.10.2024 | |
| 2 | Аналіз існуючих методів виявлення вразливостей | 21.10-27.10.2024 | |
| 3 | Огляд методів машинного навчання | 28.10-31.10.2024 | |
| 4 | Дослідження технологій машинного навчання для виявлення SQL-ін’єкцій | 01.11-10.11.2024 | |
| 5 | Порівння ефективності виявлення SQL-ін’єкцій алгоритмами машинного навчання | 11.11-13.11.2024 | |
| 6 | Застосування машинного навчання для виявлення вразливостей веб-додатків до SQL-ін’єкцій | 14.11-17.11.2024 | |
| 7 | Оформлення роботи: вступ, висновки, реферат | 18.11-24.11.2024 | |
| 8 | Розробка демонстраційних матеріалів | 25.11-01.12.2024 | |
| 9 | Попередній захист роботи | 02.12.2024 | |

Здобувач вищої освіти _____ Анна ФЕДОРЕНКО
*(підпис)*

Керівник
кваліфікаційної роботи _____ Віталій САВЧЕНКО
*(підпис)*

# РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 92 стор., 4 табл., 36 рис., 44 джерел.

*Метою роботи* є удосконалення процесу виявлення SQL-ін’єкцій з використанням методів машинного навчання.

*Об’єкт дослідження* – методи та засоби захисту веб-додатків від SQL-ін’єкцій.

*Предмет дослідження* – є методи машинного навчання для виявлення та запобігання SQL-ін'єкціям у web-додатках.

У роботі використано різноманітні методи, а саме методи машинного навчання, методи обробки тексту, методи динамічного аналізу та методи оцінки ефективності. Акцент був зроблений саме на методах машинного навчання.

Було проведено аналіз алгоритмів машинного навчання. Оцінено їх продуктивність та точність виявлення шкідливих запитів. Визначено найбільш ефективний алгоритм для визначення SQL-ін’єкцій.

Розроблено метод виявлення вразливості веб-додатків до SQL-ін’єкцій з використанням методів машинного навчання. Використано методи обробки тексту, такі як токенізація, паддінг і векторизація. Динамічний аналіз був реалізований через інтеграцію OWASP ZAP API, для проведення поглибленої перевірки підозрілих запитів.

Для тестування методу було розроблено веб-додаток з формую реєстрації. Метод тестувався за двома сценаріями: введення нормальних даних та введення шкідливих запитів. Метод успішно пройшов тестування та показав високу ефективність та точність виявлення SQL-ін’єкцій.

КЛЮЧОВІ СЛОВА: SQL-ІН’ЄКЦІЇ, МАШИННЕ НАВЧАННЯ, АЛГОРИТМИ МАШИННОГО НАВЧАННЯ, ДИНАМІЧНИЙ АНАЛІЗ.

## ABSTRACT

Text part of the qualification work for obtaining a master's degree: 92 pages, 4 tables, 36 figures, 44 sources.

The purpose of the work is to improve existing methods for detecting vulnerabilities of web applications to SQL injections using machine learning methods.

The object of the research is methods and means of protecting web applications from SQL injections.

The subject of the research is machine learning methods for detecting and preventing SQL injections in web applications.

The work uses various methods, namely machine learning methods, text processing methods, dynamic analysis methods and methods for assessing effectiveness. The emphasis was placed on machine learning methods.

An analysis of machine learning algorithms was conducted. Their performance and accuracy of detecting malicious queries were evaluated. The most effective algorithm for detecting SQL injections was determined.

A method for detecting vulnerabilities of web applications to SQL injections using machine learning methods was developed. Text processing methods such as tokenization, padding, and vectorization were used. Dynamic analysis was implemented through the integration of OWASP ZAP API to conduct in-depth inspection of suspicious requests.

A web application with a registration form was developed to test the method. The method was tested in two scenarios: normal data entry and malicious request entry. The method successfully passed the test and showed high efficiency and accuracy in detecting SQL injections.



KEYWORDS: SQL INJECTIONS, MACHINE LEARNING, MACHINE LEARNING ALGORITHMS, DYNAMIC ANALYSIS.

**ЗМІСТ**

# ВСТУП

На сьогоднішній день веб-додатки відіграють важливу роль у нашому житті, оскільки використовуються у різних сферах, таких як онлайн-торгівля, банківські послуги, соціальні мережі, електронна комерція. Зважаючи на їх популярність та широке застосування, потреба на забезпечення безпеки веб-додатків зростає. Збільшення кількості веб-додатків супроводжується збільшенням кількості кіберзагроз, серед яких однією з найпоширеніших є SQL-ін’єкції, які можуть призвести до доступу до конфіденційної інформації та порушення її цілісності.

Такі веб-атаки як SQL-ін’єкції існують вже десятиліттями та продовжують бути актуальними, адже стають причиною розголошення персональних даних, а також завдають негативний фінансовий вплив на бізнес і державні установи. Компанії з кібербезпеки приділяють значну увагу для пом’якшення наслідків веб-атаки, багато сучасних стратегій мають обмеження, які поточні дослідження постійно намагаються подолати.

Одним з традиційних методів пом’якшення веб-атак є статичний аналіз вхідного веб-трафіку, також відомого як виявлення за сигнатурами. Стратегія передбачає створення сигнатури, характерної для веб-атаки, коли ця сигнатура виявляється, підозрілий трафік блокується брандмауером або іншим засобом безпеки. Переваги цього методу є швидкість і те що він може бути реалізований в режимі реального часу для захисту мережевих ресурсів. Недоліками цього методу є те, що за допомогою нього можна виявити лише відомі атаки. Ще одна стратегія для пом’якшення веб-атак, розроблена спеціально для SQL-ін’єкцій. Вона полягає в аналізі структури вхідних SQL-запитів. Якщо виявляється некоректний запит, це вважається атакою SQL-ін’єкцій.

Переваги даного методу є те, що він дає добрі результати виявлення і також може виявляти нові атаки, що включають некоректні запити. Недоліками є те, що цей метод вимагає значних знань про додаток і структуру «нормальних» запитів.

*Метою роботи* є удосконалення існуючих методів виявлення вразливостей веб-додатків до SQL-ін’єкцій з використанням методів машинного навчання.

*Об’єкт дослідження* – методи та засоби захисту веб-додатків від SQL-ін’єкцій.

*Предмет дослідження* – є методи машинного навчання для виявлення та запобігання SQL-ін'єкціям у web-додатках.

*Наукове завдання* – розробка методики виявлення вразливостей веб-додатків до SQL-ін’єкцій за допомогою методів машинного навчання.

Для досягнення поставленої мети потрібно виконати наступні завдання:

- проаналізувати існуючі методи та засоби виявлення вразливостей веб-додатків до SQL-ін’єкцій;
- проаналізувати методи машинного навчання;
- порівняти методи машинного навчання для виявлення SQL-ін’єкцій;
- розробити метод виявлення вразливостей веб-додатків до SQL-ін’єкцій за допомогою методів машинного навчання.

# 1 SQL-ІН’ЄКЦІЇ ЯК ЗАГРОЗА БЕЗПЕЦІ WEB-ДОДАТКІВ

## 1.1 Поняття та механізм виконання SQL-ін’єкцій

SQL-ін’єкція — це кібератака, під час якої зловмисник впроваджує SQL код в поля для вводу даних з метою виконання цього коду. За допомогою такої техніки впровадження коду можна отримувати доступ та маніпулювати базою даних, змінювати або видаляти дані, також можна виконувати адміністративні операції. Такі дії становлять значну загрозу безпеці та цілісності цільової системи [1].

Такий тип атаки як SQL-ін’єкції є однією з найпоширеніших типів вразливостей у веб-додатках. Вони продовжують становити значну загрозу інформаційній безпеці та залишають проблемою протягом багатьох років [1].

Як правило шкідливий код SQL вставляють у поля введення програми, так як ці поля введення є частиною взаємодії з користувачем, як-от форма входу, рядки пошуку чи інша форма подання даних. Коли програма обробляє ці вхідні дані без відповідної перевірки чи дезінфекції, введений SQL-код стає частиною SQL-запиту програми, змінюючи його заплановану поведінку [1].

Під час SQL-атаки фрагмент шкідливого коду вставляється в параметри запиту, що змушує сервер виконувати запити, що призводить до витоку даних та пошкодження бази даних. Як показано на Рисунку 1.1, зловмисник надсилає шкідливі оператори у застосунок за допомогою методів ін’єкція введення користувача, ін’єкція cookie та ін’єкція змінної сервера [2].

На рисунку 1.1 зображено схему атаки SQL-ін’єкціями.

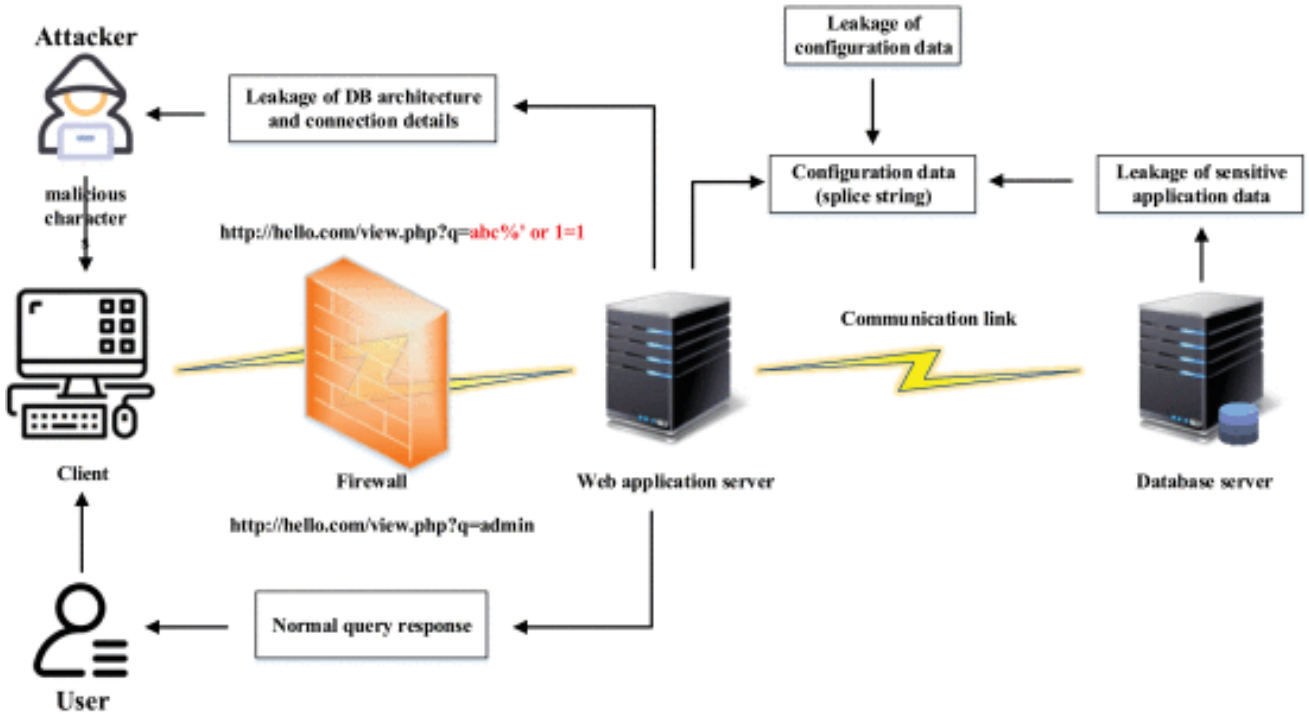



Рис. 1.1 Схема атаки SQL-ін’єкціями [2].

SQL-ін’єкція стала однією з найбільш часто використовуваних атак, через те що вона потребує лише створення структурованих операторів запиту без розробки додаткових шкідливих програм. Методи атаки часто змінюються з розвитком технологій веб-сайтів. Структура мови SQL змінна і підтримує різні методи кодування, тому деякі традиційні методи виявлення, такі як механізм чорного списку та виявлення на основі правил, не забезпечують кращий захисний ефект [1].

Припустимо, у веб-додатку є форма авторизації де користувач вводить своє ім’я та пароль. Дані, введені користувачем використовуються для створення SQL-запиту для перевірки авторизації:

```
SELECT * FROM users WHERE username = '[input_login]' AND password = '[input_password]'
```

Якщо додаток не перевіряє та не обробляє вхідні дані належним чином, зловмисник може використати це для впровадження SQL-ін’єкції [1]. Наприклад, якщо у поле «ім’я користувача» ввести наступне:

```
' OR '1'='1
```

Тоді SQL-запит буде виглядати наступним чином:

```
SELECT * FROM users WHERE username = '' OR '1'='1' AND password =
'[input_password]'
```

Через додавання оператора OR речення WHERE повертає перше id з users таблиці незалежно від того, які є username і password. Зазвичай першим користувачем id бази даних є адміністратор, і таким чином зловмисник не тільки обійде авторизацію, але й отримує права адміністратора [1].

SQL-ін'єкції також можуть бути націлені на поля пошуку в веб-додатках. При введені пошукового запиту, програма обробляє введені дані та створює запит бо бази даних для отримання відповідних результатів. Змінивши SQL-запит до бази даних, зловмисник може отримати конфіденційну інформацію з бази даних [1].

Успішні атаки SQL-ін'єкцій можуть мати серйозні наслідки як для веб-додатків так і для організацій що за ними стоять. Використання вразливостей при перевірці вхідних даних може призвести до значної шкоди, поставивши під загрозу конфіденційні дані, конфіденційність та цілісність сайту [1].

Наслідки SQL-інєкцій можуть бути серйозними і мати наступні наслідки :

1. Порушення даних: SQL-ін'єкції можуть призвести до несанкціонованого доступу до бази даних, що дозволяє зловмиснику викрасти конфіденційну інформацію користувача, таку як їхні імена, паролі, адреси електронної пошти та особисті дані. Що може призвести до витоку даних з подальшими наслідками, які включають крадіжку особистої інформації, фінансове шахрайство та шкоду репутації для постраждалої організації [1].
2. Несанкціонований доступ і підвищення привілеїв: шляхом маніпулювання запитами, зловмисник може отримати несанкціонований доступ до областей обмеженого доступу або підвищити свої привілеї в програмі. Це може надати їм можливість виконувати адміністративні дії або навіть отримати повний контроль над системою [1].

3. Пошкодження бази даних: отримавши доступ до бази даних зловмисник може виконати шкідливі операції, такі як видалення або зміни таблиць, що в результаті може призвести до недоступності або пошкодження даних [1].
4. Виконання вторгнення: SQL-ін’єкції зловмисники можуть використовувати для виконання додаткових атак, таких як віддалене виконання коду, увімкнення зловмисного програмного забезпечення або вторгнення в інші системи, що взаємодіють з базою даних [1].
5. Втрата довіри користувачів: атаки SQL-ін’єкцій може призвести до розкриття конфіденційних даних, що зберігаються в базах даних, наприклад фінансові записи, медичну інформацію або інтелектуальну власність. Що може призвести до втрати довіри до ресурсів та його власника [1].

SQL-ін’єкції залишаються постійною та поширеною загрозою безпеці та цілісності веб-додатків. Розуміння ключових висновків щодо SQL-ін’єкцій та впровадження ефективних заходів є важливим для забезпечення безпеки конфіденційних даних та надійності програм [1].

### 1.2 Види SQL-ін’єкцій та їх вплив на безпеку web-додатків

За допомогою SQL-ін’єкцій зловмисник може обійти автентифікацію, отримати доступ до бази даних, змінювати та видаляти в ній дані. У деяких випадках використовується для виконання команд в операційній системі та потенційно дозволити перейти до більш шкідлтвих атак усередині мережі. Класичні SQL-ін’єкції мають такі основні типи: In-band (внутрішньосмугові), SQLi, Blind (сліпі) SQLi, Out-of-band (позасмугові) SQLi та комбіновані SQLi [3].

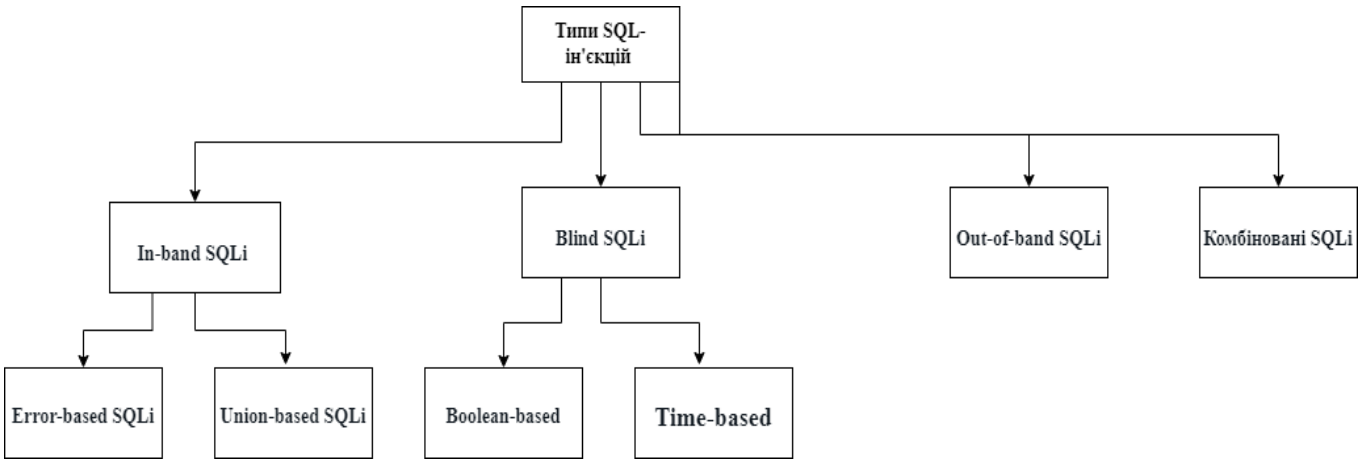



Рис. 1.2 Типи SQL-ін’єкцій.

In-band SQLi (внутрішньосмугові) — це тип впровадження SQL коду, коли зловмисник отримує результат як пряму відповідь, використовуючи той самий канал зв’язку. In-band SQLi впровадження також ще відоме як класичне впровадження SQL коду. Два найпоширеніших типа внутрішньосмугового впровадження SQL є SQLi на основі помилок та SQLi на основі об’єднання [4].

1. SQLi на основі помилок (Error-based SQLi) — ця техніка базується на повідомленнях про помилки, які видає сервер бази даних, для того щоб отримати інформацію про структуру бази даних. У деяких випадках достатньо лише SQLi на основі помилок, щоб отримати доступ до усієї базт даних. Однак такі помилки корисні під час розробки вебзастосунків, проте на робочих сайтах їх слід вимикати або записувати у файл із обмеженим доступом [4].
2. SQLi на основі об’єднання (Union-based SQLi) — ця техніка використовує оператор SQL UNION для об’єднання результатів операторів SELECT в один результат, який потім повертається як частина відповіді HTTP [4].

Ще один вид SQLi — це сліпа SQL-ін’єкція (Blind SQLi). Ця техніка, на відмінно від внутрішньосмугового SQL, займає більше часу щоб використати його.

Під час використання сліпої SQL-ін’єкції зловмисник не може побачити результат атаки у тому ж каналі, за що вона і отримали свою назву “сліпа SQL-ін’єкція”. Однак надсилаючи шкідливі запити, зловмисник може відтворити структуру бази даних, спостерігаючи за відповідями вебзастосунку та аналізуючи поведінку серверу бази даних. Є два типи основних типа такої техніки: SQLi на основі логічних значень і SQLi на основі часу [5].

1. Сліпі SQL-ін’єкції на основі логічних значень (Boolean-based) — ця техніка полягає у відправлені SQL-запиту до бази даних, що змушує застосунок повертати різний результат залежно від того, чи повертає запит значенні TRUE або FALSE. Залежно від результату вміст відповіді HTTP зміниться або залишиться незмінним. Це дозволить зробити висновок, чи використане корисне навантаження повернуло істину чи хибність, навіть якщо з бази даних дані не повернуто. Цей тип атаки зазвичай повільний, оскільки потрібно перерахувати базу даних символ за символом [5].
2. Сліпі SQL-ін’єкції на основі часу (Time-based) — ця техніка базується на надсиланні SQL-запиту до бази даних, що змушує її чекати певний час, перш ніж відповісти. Час очікування вкаже на те чи є результат запиту TRUE або FALSE [5].

Позасмуговий (Out-of-band) SQLi — цей тип ін’єкцій є не дуже поширеним тому, що залежить від певних функцій, увімкнених на сервері бази даних, який використовує вебзастосунок. Позасмугова SQL-ін’єкція відбувається тоді, коли не можливо використати той самий канал для запуску атаки та збору результатів. Цей метод пропонує альтернативну сліпим ін’єкціям на основі часу, особливо якщо сервера не стабільні. Ця техніка спирається на здатність сервера бази даних здійснювати DNS- або HTTP- запитів для передачі даних зловмиснику [6].

Комбіновані SQLi — це поєднання двох або більше атак, що атакують веб-застосунки та викликають серйозні наслідки. Комбіновані атаки з’явились як наслідок швидкого розвитку методів запобігання та виявлення різних SQL-ін’єкцій.

Комбінована атака походить від поєднання атаки SQL-ін’єкціями та інших атак на веб-сайти, що можна класифікувати наступним чином [7]:

SQLi + DDoS атака — такий вид комбінованої атаки використовується для призупинення роботи сервера, виснаження ресурсів, щоб користувач не міг отримати доступ до сервера. Є різні команди, які можуть бути використані в SQL-ін’єкціях для проведення DDoS-атаки, наприклад такі як encode, compress, join тощо. Наукових досліджень щодо запобігань такого виду атаки проведено дуже мало, оскільки це складний вид атаки з точки зору безпеки. Для здійснення такого типу атаки виконують кілька основних етапів: пошук вразливості, підготовка до її використання та застосування складного коду для атаки. Чим більше стовпців і рядків у базі даних, тим легше виконати DDoS-атаку за допомогою SQL [7];

SQLi + DNS Hijacking — витік даних за використання сліпої SQL-ін’єкції зазвичай відбувається повільно. Тому вигадали атаку через DNS, яка значно швидше та менш помітна, ніж сліпа SQL-ін’єкція. Основна мета такої комбінованої атаки — впровадити SQL-запит у DNS—запит, перехопити його та направити в інтернет. Термін викрадання DNS (DNS Hijacking) стосується модифікації DNS-записів або зловживання адміністративним доступом до реєстру доменів. Після того як викрадання DNS-записів буде здійснено, наступним етапом буде SQL-ін’єкція з використанням DNS-запиту [7];

SQLi + XSS — SQL-ін’єкції в цьому типу атаки є лише засобом підготовки до неї, решту роботи виконується за допомогою XSS (міжсайтового скриптингу). Такі атаки відомі як атаки третьої хвилі, оскільки вони не є типовими старими методами, а використовують команди для приховування від пристроїв моніторингу мережі [7];

SQLi + недостатня автентифікація — цей тип атаки пов'язаний із недостатньою автентифікацією, коли користувач або адміністратор сайту не має достатнього досвіду. Параметри безпеки не ініціалізовані належним чином, через що застосунок не може ідентифікувати місцезнаходження користувача, служби або застосунку. Цю

вразливість використовують для здійснення атаки за допомогою SQL-ін’єкцій. Цей тип атаки є відносно простим у виконанні у порівняні з іншими видами атак [7].

Також значну загрозу становлять SQL-ін’єкції що автоматизовані за допомогою ботів. Вони можуть виконувати тисячі спроб впровадження SQL одночасно на кількох вебзастосунках. Тестуючи різні поля введення та намагатись виконувати різні типи ін’єкцій на високих швидкостях, та використовувати передові методи, щоб уникнути виявлення, наприклад змінювати IP-адресу, використовувати проксі та шифрування. Після успішного впровадження шкідливого коду, вони автоматизують процес вилучення конфіденційних даних із бази даних. Боти можуть постійно шукати вразливості та адаптуватись до змін у структурі програми або заходах безпеки. Для боротьби з керованими ботами, запроваджують обмеження швидкості, для того щоб контролювати обсяги запитів з окремих IP-адрес [8].

Одні з основних причин, які роблять систему вразливою до SQL-ін’єкцій це:

пряме вбудування вводу користувача в SQL-запит — коли вхідні дані безпосередньо підставляються в SQL-запит, цей запит можна модифікувати, додавши власні команди [9];

недостатня валідація вхідних даних — якщо вхідні дані не перевіряються належним чином на наявність шкідливих символів, можна легко ввести спеціальні символи, які будуть інтерпретовані як частина SQL-коду [9];

використання динамічних SQL-запитів — динамічні запити дозволяють будувати SQL-запити на основі вхідних даних, але вони також створюють ризики SQL-ін’єкцій, якщо дані не обробляються належним чином [9];

відсутність екранування спеціальних символів — спеціальні символи такі як лапки (`), крапка з комою (;) та інші, мають особливе значення в SQL. Якщо вони не екрануються, їх можна використати для маніпуляції SQL-запитом [9];

використання небезпечних функцій — такі функції в базі даних, як eval () та execute (), дозволяють виконувати довільний код, ці функції можуть призвести до SQL-ін’єкцій, якщо їх використовувати з неперевіреними вхідними даними [9].

## 1.3 Методи та засоби виявлення SQL-ін’єкцій

На сьогоднішній день існує багато методів та засобів виявлення SQL-ін’єкцій. Вони використовують різні механізми кодування та спеціальні інструменти для виявлення та запобігання SQL атак на базу даних. Для запобігання сучасним атакам SQL-ін’єкцій завжди рекомендується використовувати підготовлені вирази, оскільки вони фіксовані та не можуть бути зміненні користувачем веб-застосунку.

Одним з традиційних методів виявлення SQL-ін’єкцій є статичний аналіз коду. Статичний аналіз, також відомий аналіз вихідного коду, зазвичай проводиться в рамках перевірки коду і реалізується на стадії впровадження життєвого циклу розробки безпеки. Цей процес зазвичай включає використання інструментів статичного аналізу коду, які спрямовані на виявлення потенційних вразливостей в неактивному вихідному коді за допомогою методів, таких як аналіз забруднення та аналіз потоку даних та лексичний аналіз [10].

Існують різні методи аналізу статичного вихідного коду на потенційні вразливості, які можна об’єднати в одне рішення. Ці методи часто походять від технологій компілятора [10].

1. Аналіз потоку даних – аналіз потоку даних використовується для збору під час виконання інформації про дані в програмному забезпеченні, коли вона перебуває в статичному стані. В аналізі потоку даних використовуються три загальні терміни: основний блок (код), аналіз потоку керування (потік даних) і шлях потоку керування (шлях, яким проходять дані) [10].
2. Аналіз забруднення – намагається виявити змінні, які були «зіпсовані» керованим користувачем введенням, і відстежує їх до можливих вразливих функцій [10].
3. Лексичний аналіз – перетворює синтаксис вихідного коду на токени інформації, намагаючись абстрагувати вихідний код і полегшити маніпулювання ним [10].

Такий метод добре масштабується та якщо йдеться про автоматичний пошук вразливостей, таких як переповнений буфер, SQL-ін’єкції та інші, даний метод дуже корисний [10].

Обмеженнями такого методу є те, що інструменти статичного аналізу коду часто дають хибнопозитивні результати, коли інструмент повідомляє про можливу вразливість, якої насправді немає. Також він може давати хибнонегативні результати, коли є вразливість, але інструмент не повідомляє про них [10].

Ще одним методом є динамічний аналіз коду. Динамічний аналіз коду – також відомий як динамічне тестування безпеки додатків (DAST), призначений для перевірки запущеної програми на наявність вразливостей, які потенційно можуть бути використані в майбутньому. Динамічний аналіз коду застосовується, коли програма в основному завершена та готова до виконання. Цей метод використовує шкідливі дані для імітації реалістичних атак на програму та спостереження за її відповідями [11].

Однією з переваг динамічного аналізу коду є те, що він може імітувати поведінку програми в реалістичному середовищі розгортання. Що дозволяє виявити проблеми конфігурації та інші вразливості, які можуть бути видимі лише тоді, коли код активний [11].

Третім методом виявлення є виявлення на основі сигнатур. Система ідентифікації на основі сигнатур виявляють вторгнення, спостерігаючи за подіями та ідентифікуючи шаблони, які відповідають сигнатурам відомих атак. Система виявлення на основі сигнатур, призначена для швидкого виявлення шаблонів у мережевому трафіку, які вказують на зловмисну активність або неавторизований доступ [12].

Виявлення на основі сигнатур є одним з найбільш прямих і добре налагоджених методів виявлення зловмисної діяльності. Цей метод перевіряє мережевий трафік, порівнює його з відомими сигнатурами та генерує сповіщення, коли збігається. Однак даний метод не в змозі виявляти моделі або індикатори нових загроз, які ще не відомі [12].

Четвертим методом виявлення SQL-ін’єкцій є фільтрація за допомогою регулярних виразів. Регулярні вирази є потужним і простим способом опису набору рядків. З цієї причини регулярні вирази часто обирають як мову вводу для текстових обробних застосунків. Тривіальний регулярний вираз для виявлення SQL-ін’єкцій полягає у відстеженні специфічних для SQL метасимволів, таких як одинарна лапка (‘) або подвійний дефіс (--) та інші. Цей метод ґрунтується на регулярних виразах що будуються для різних полів введення, а потім система перевіряє шаблони атак проти цих регулярних виразів. Якщо будь-яке правило не відповідає, то шаблони атаки надсилає до системи виявлення. Потім система аналізує шаблон атаки за допомогою внутрішньої бази даних для класифікації вразливості [13].

У таблиці 1.1 наведенні переваги та недоліки методів виявлення SQL-ін’єкцій.

Таблиця 1.1

Порівняльний аналіз методів виявлення SQL-ін’єкцій

| Метод виявлення | Опис методу | Переваги | Недоліки |
|---|---|---|---|
| Статичний аналіз коду | Аналіз вихідного коду без його виконання. Шукає потенційно вразливі конструкції та шаблони коду. | Висока точність виявлення, можливість виявлення вразливостей на ранніх стадіях розробки. | Не може виявити всі типи вразливостей, особливо ті, що з'являються під час виконання. Вимагає значних обчислювальних ресурсів. |
| Метод на основі сигнатур | Пошук відомих шаблонів атак у вхідних даних або мережевому трафіку | Швидкий і ефективний для виявлення відомих атак. | Не може виявити нові, невідомі атаки. Вимагає постійного оновлення бази сигнатур. |
| Динамічний аналіз коду | Аналіз програми під час її виконання шляхом введення різних даних та моніторингу реакції програми | Виявляє вразливості, які неможливо виявити статичним аналізом. Може виявити реальні умови експлуатації вразливостей. | Вимагає великої кількості тестових даних. Може бути трудомістким для великих додатків. |

Продовження таблиці 1.1

Порівняльний аналіз методів виявлення SQL-ін’єкцій

| Метод виявлення | Опис методу | Переваги | Недоліки |
|---|---|---|---|
| Метод виявлення на основі регулярних виразів | Визначення наявності певних послідовностей символів у вхідних даних, які можуть свідчити про спробу SQL-ін’єкції | Гнучкий і настроюваний метод. Може виявити різноманітні типи атак. | Може генерувати хибнопозитиві результатів. |

Ручна перевірка на вразливість до SQL-ін’єкцій – це процес ручного використання вразливості шляхом впровадження коду SQL у поля введення програми. Така перевірка є першим кроком у виявленні вразливості у веб-додатках. Вона потребує уважного аналізу веб-ресурсу та використання специфічних технік для виявлення вразливості. Для оцінки рядка запиту, рядок розбивається на послідовність токенів і перевіряється, чи містять вони лише довірені дані. Якщо всі токени проходять тести, запит вважається безпечним і передається  для оцінки розміру результату запиту [14].

Зміст цієї перевірки полягає у введені різноманітних значень у поля вводу у додатку та спостереженні за результатами запитів до бази даних. Якщо результати відповідають очікуванню, то веб-ресурс не є вразливим до SQL-ін’єкцій, але якщо результати не відповідають очікуваним, можна припустити що ресурс має вразливості до даного виду атаки [14].

Основні кроки які можна вжити для перевірки на вразливість:

1. Взаємодія з формами: ретельно перевірити усі можливі поля введення, де користувач може ввести дані. Особливо звернути увагу на поля, які взаємодіють з базою даних, такі як поля пошуку, паролі, логіни [14].
2. Введення спеціальних символів:  ввести в поля вводу різні спеціальні символи (наприклад, «’», «”», «,», «/*», «*/») і проаналізувати реакцію

системи. Якщо введення спеціальних символів змінює поведінку додатку або викликає помилки бази даних, це може бути ознакою вразливості [14].

3. Введення логічних виразів: ввести логічні вирази, такі як «OR 1=1» або «OR 1=0». Якщо в результаті введення «OR 1=1» повертаються усі записи з бази даних, а «OR 1=0» не повертає нічого, це свідчить про вразливість [14].
4. Введення субзапитів: ввести в поля вводу субзапити наприклад «'; SELECT * FROM users WHERE 'a' = 'a'» та проаналізувати поведінку додатку [14].
5. Тестування додаткових сценаріїв: використати конструкції UNION-атаки, наприклад, «UNION SELECT username, password FROM users» та перевірити чи відображається дані з інших таблиць або поля [14].
6. Перевірка обробки вхідних даних: перевірити як додаток обробляє правильні і не правильні вхідні дані. Спробувати ввести не правильні дані або спеціальні символи, щоб побачити, як додаток реагує на них. Якщо додаток екранує неправильні дані, це може свідчити про наявність вразливості [14].
7. Проаналізувати повідомлення про помилки: перевірити повідомлення про помилки, якщо такі виникають. Вони можуть містити інформацію про структуру бази даних або синтаксис SQL [14].

Ручна перевірка на вразливості до SQL-ін'єкцій дозволяє виявити вразливості, які може бути важко виявити автоматизованими інструментами. Також вона дозволяє здійснювати глибокий аналіз коду та інфраструктури додатка. Однак цей метод має свої обмеження, цей метод є часозатратний та працезатратний, особливо для великих проектів. Також він піддається людським помилкам і може не виявити всі можливі вразливості [14].

Автоматизовані сканери виявлення вразливостей забезпечують як автоматизовану оцінку вразливостей, так і автентифіковані перевірки безпеки, націлюючись на все: від кінцевих пристроїв до вразливих ділянок веб-додатків. Вони ефективно ідентифікують слабкі місця у системі, використовуючи широку базу даних

вразливостей. Основні проблеми які вони вирішують, це потенційні прогалини в безпеці, забезпечуючи проактивний захист [15].

Існує багато автоматизованих сканерів виявлення вразливостей, які можуть допомогти виявити потенційні SQL-ін’єкції в веб-додатках.

Ось декілька популярних сканерів:

1. Nessus – адаптований до різних середовищ та надає детальне звітування. Надає глибоке розуміння потенційних ризиків, здатний інтегруватися з багатьма платформами [15].
2. OpenVas – інструмент з відкритим кодом, призначений для сканування вразливостей і керування ними. Містить повну та регулярну оновлювану базу даних тестів мережевої вразливості. Також може інтегруватися з іншими інструментами з відкритим вихідним кодом [15].
3. SQLMap – інструмент виявлення SQL-ін’єкцій з відкритим кодом, виявляє вразливості до SQL-ін’єкцій у базах даних та веб-додатках [15].
4. Invicti – хмарна платформа, яка забезпечує комплексне тестування безпеки веб-додатків, включаючи виявлення ін’єкцій SQL. Використовує комбінацію ручного тестування та автоматичного сканування для виявлення вразливостей [15].
5. Burp Scanner – інструмент тестування безпеки веб-додатків, який включає можливості виявлення впровадження SQL. Може виявляти різні вразливості впровадження SQL, включаючи сліпу та часову SQL-ін’єкцію [15].
6. Acunetix – комплексний інструмент тестування безпеки веб-додатків. Може виявляти різні вразливості SQL-ін’єкцій, включаючи сліпу SQL-ін’єкцію [15].

При виборі інструмента виявлення SQL-ін’єкцій, слід врахувати багато факторів, таких як функції, простота використання. Проте, варто зазначити, що автоматизовані сканери не завжди здатні знайти всі можливі вразливості, і ручна перевірка та аудит безпеки також є необхідним для забезпечення повної безпеки веб-додатків [15].

У таблиці 1.2 наведено переваги та недоліки розглянутих сканерів.

Таблиця 1.2

Порівняльний аналіз автоматизованих сканерів вразливостей

| Автоматизований сканер | Переваги | Недоліки |
|---|---|---|
| Nessus | Широкий спектр можливостей сканування<br>Інтегрується з багатьма сторонніми системами | Може бути складний для початківців |
| OpenVas | Можливість налаштування та адаптації<br>Велика бібліотека тестів мережевих вразливостей | Інтеграція з системами може потребувати налаштування вручну |
| SQLMap | Може автоматично виявляти та використовувати SQL-ін’єкції<br>Виконує широкий спектр тестів | Може генерувати помилкові спрацьовування<br>Може бути заблокований брандмауерами веб-додатків |
| Invicti | Комплексне тестування безпеки веб-додатків<br>Хмарна платформа<br>Детальні звіти | Обмежені можливості налаштування |
| Burp Scanner | Повна база даних корисних навантажень атак<br>Може виявляти різні типи вразливостей впровадження SQL | Потрібне добре розуміння тестування безпеки веб-додатків |
| Acunetix | Комплексне тестування безпеки веб-додатків<br>Надає докладний звіт та пропозиції щодо виправлення | Потрібне добре розуміння тестування безпеки веб-додатків |

Сканери проводять швидкий та масштабний аналіз додатків, за допомогою різних технік сканування здатні виявляти потенційні вразливості. Однак у них є і свої обмеження, вони не завжди здатні виявити складні вразливості або вразливості, які вимагають ручної перевірки. Також вони можуть надавати хибні результати [15].

Одним із найефективніших способів звести до мінімуму шанси на успішне впровадження SQL є використання брандмауера веб-додатків (WAF). WAF впроваджується на стороні сервера для захисту веб-додатків та веб-сайтів [16].

Оскільки WAF працює на 7 рівні мережевої моделі OSI, вони краще розуміють, які дані надходять, ніж традиційні брандмауер що працює на 3 рівні моделі OSI. Він відстежує і фільтрує вхідні HTTP-запити GET і POST, блокуючи пакети даних, які вони вважають шкідливими, ідентифікує та блокує відомий шкідливий синтаксис SQL і аналізує вхідний синтаксис SQL, щоб оцінити його потенційну шкоду. Ефективність цього інструменту залежить від їх конфігурації, конкретних правил, які вони використовують, і їх здатності розпізнавати потенційні спроби SQL-ін’єкцій [16].

Однак є багато ситуацій, у яких WAF можуть не захистити від SQL-ін’єкцій:

невідомі шаблони та сигнатури атак – цей інструмент зосереджений на виявленні в відомих шаблонів і сигнатур атак [16];

неправильна конфігурація – якщо інструмент налаштований неправильно та використовує лише загальні правила, він може не забезпечувати достатнього захисту [16];

дані немережевого трафіку – WAF не може вирішувати проблеми, що виникають через власний код програми або джерело даних, потенційно залишаючи вразливі місця [16];

синтаксис JSON – WAF можуть не розпізнати синтаксис JSON у корисних навантаженнях SQL-ін’єкцій як потенційно шкідливий, дозволяючи зловмисникам обійти захист WAF [16].

## Висновки

SQL-ін’єкції становлять серйозну загрозу для web-додатків та бази даних, оскільки з їх допомогою можна отримати доступ до конфіденційної інформації. Через недостатнє валідацію введених даних або неправильне використання SQL-запитів, введенні дані не обробляються належним чином та стають частиною запиту до бази даних, що дозволяє змінювати її структуру.

Аналізуючи види SQL-ін’єкцій, можна зробити висновок, що вони можуть здійснюватися у різних формах та через різні механізми взаємодії з web-додатками. Наслідки від такого виду атаки, можуть бути руйнівними як для компанії, так і для клієнтів. Одними з найнебезпечніших атак, є атаки спрямовані на отримання доступу адміністраторських привілеїв у базі даних, оскільки вони дозволяють здійснювати критичні зміни в системі. Невід’ємною частиною забезпечення інформаційної безпеки є захист від SQL-ін’єкцій. Основні методи захисту включають: використання підготовлених запитів, правильна валідація вхідних даних, обмеження прав доступу до бази даних, шифрування даних, впровадження систем моніторингу та виявлення вторгнень. Комплексне застосування цих методів дозволяє мінімізувати ризик атак.

Загалом, SQL-ін’єкції є одною з найбільш поширених і небезпечних загроз безпеці веб-додатків, що можуть призвести до втрати конфіденційної інформації та фінансових збитків.

# 2 ДОСЛІДЖЕННЯ МЕТОДІВ ТА АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ SQL-ІН'ЄКЦІЙ

## 2.1 Огляд методів машинного навчання

Машинне навчання (ML) — це тип штучного інтелекту, який дозволяє машинам навчатись на основі даних без явного програмування. Це робиться шляхом оптимізації параметрів моделі за допомогою обчислень, щоб поведінка моделі відображала дані або досвід. Алгоритми навчання постійно оновлює значення параметрів у міпу навчання, дозволяючи моделі машинного навчання навчатись та роботи прогнози або рішення на основі науки про дані. Алгоритми машинного навчання знайшли застосування у багатьох сферах, таких як медицина, фільтрація електронної пошти, розпізнавання мови, виявлення шахрайства, захист від зловмисного програмного забезпечення, автоматизація бізнес-процесів, комп'ютерний зір, проєктування безпілотних апаратів, тощо, де розробка звичайних алгоритмів для виконання необхідних завдань є складною або неможливою [17].

Мета машинного навчання — передбачати результати на основі вхідних даних. Чим більш різноманітні вхідні дані, тим легше машині виявити закономірність та тим точнішим буде результат. Щоб навчити машину, потрібно три складові: дані, ознаки та алгоритми[18, с.12].

Першим кроком у машинному навчанні є збір відповідних даних, які можуть надходити з різних джерел, таких як бази даних, датчики або Інтернет. Після збору даних необхідно їх попередньо обробити, щоб гарантувати їх якість і придатність для аналізу. Наступний етап — навчання моделі машинного навчання, яка вчиться робити прогнози чи приймати рішення на основі вхідних даних, що впливають на її продуктивність. Коли модель навчена, її потрібно оцінити для визначення

ефективності і відповідності критеріям. Після успішного навчання і оцінки, модель можна розгорнути для використання у реальних додатках машинного навчання [17].

Виділяють чотири основні типи машинного навчання: класичне навчання, навчання з підкріпленням, ансамблі та нейромережі і глибоке навчання. Класичне навчання поділяють на навчання «з учителем» та навчання «без учителя»[18, с.14].

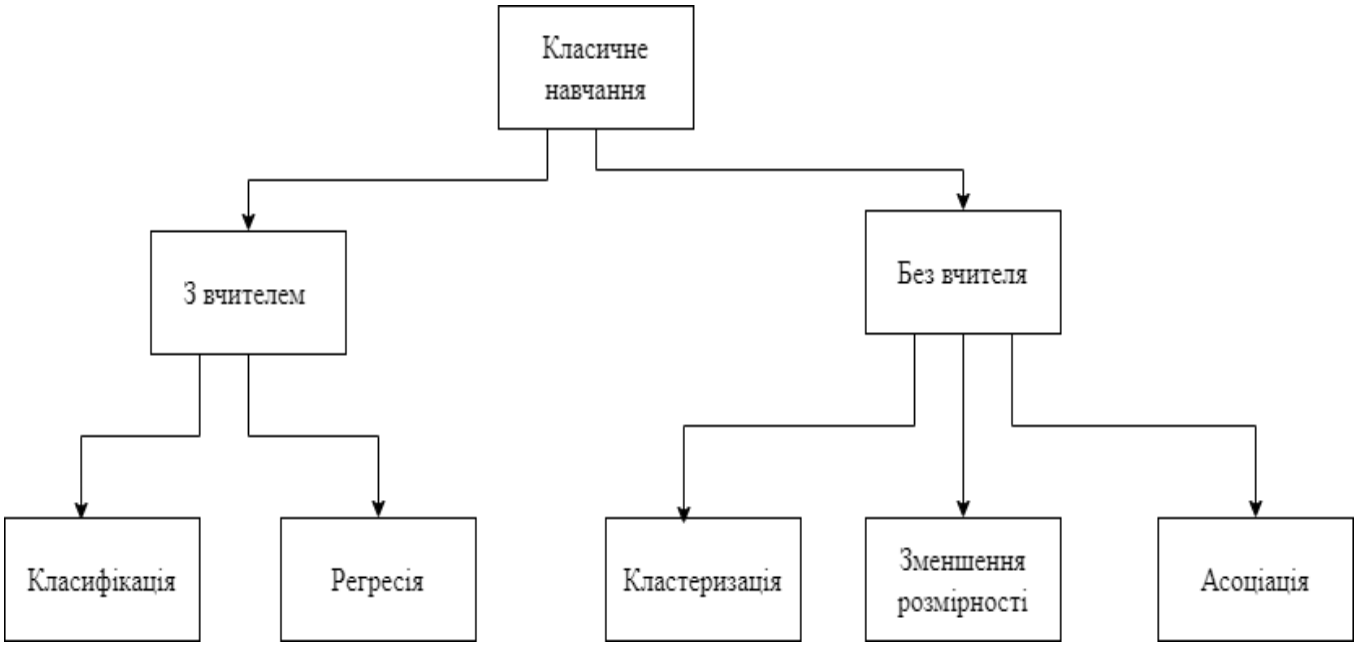



Рис. 2.1 Типи класичного навчання

Контрольоване навчання — це підхід до машинного навчання на основі розмічених даних. У таких даних кожному вхідному прикладу відповідає правильна мітка, що дозволяє моделі навчатися прогнозувати результати для нових невідомих прикладів. Алгоритм вимірює свою точність за допомогою функції втрат, коригуючи, доки помилка не буде достатньо мінімізована [19].

Контрольоване навчання поділяється на дві основні категорії: регресія та класифікація [19].

Регресія — це статистичний підхід, який використовується для аналізу зв’язку між залежною змінною та однією чи декількома незалежними змінами. Метою є визначення найбільш підходящої функції, яка характеризує між цими змінними. Цей метод машинного навчання, використовується для прогнозування значення залежної змінної для нових, невідомих даних. Він моделює зв'язок між вхідними характеристиками та цільовою змінною, дозволяючи оцінювати або прогнозувати числові значення [20].

Існує три типи регресії:

проста регресія — використовується для прогнозування безперервної залежної змінної на основі однієї незалежної змінної. Просту лінійну регресію застосовують, коли є лише одна незалежна змінна [20];

множинна регресія — використовується для прогнозування постійної залежної змінної на основі кількох незалежних змінних. Такий вид регресії слід використовувати, коли є кілька незалежних змінних [20];

нелінійна регресія — зв'язок між залежною змінною та незалежною змінною є нелінійним. Забезпечує гнучкість у моделювання широкого діапазону функціональних форм [20].

Регресію використовують для вирішення наступних задача: прогнозування цін та тенденцій, виявлення факторів ризику та прийняття рішень [20].

Класифікація — це процес класифікації даних або об’єктів у заздалегідь визначені класи або категорії на основі їх характеристик або атрибутів. Основна ідея класифікації полягає в тому, щоб навчити модель на позначеному наборі даних, де вхідні дані пов’язані з відповідними вихідними мітками, щоб вивчити шаблони та зв’язки між вхідними даними та вихідними мітками. Коли модель навчена, її можна використовувати для прогнозування вихідних міток для нових невідомих даних [21].

Існує два основних типи класифікації:

бінарна класифікація — у двійковій класифікації метою є класифікація вхідних даних за одним із двох класів або категорій [21];

багатокласова класифікація — у такому типу класифікації метою є класифікація вхідних даних за одним із кількох класів або категорій [21].

На рисунку 2.2 зображено схема процесу машинного навчання .

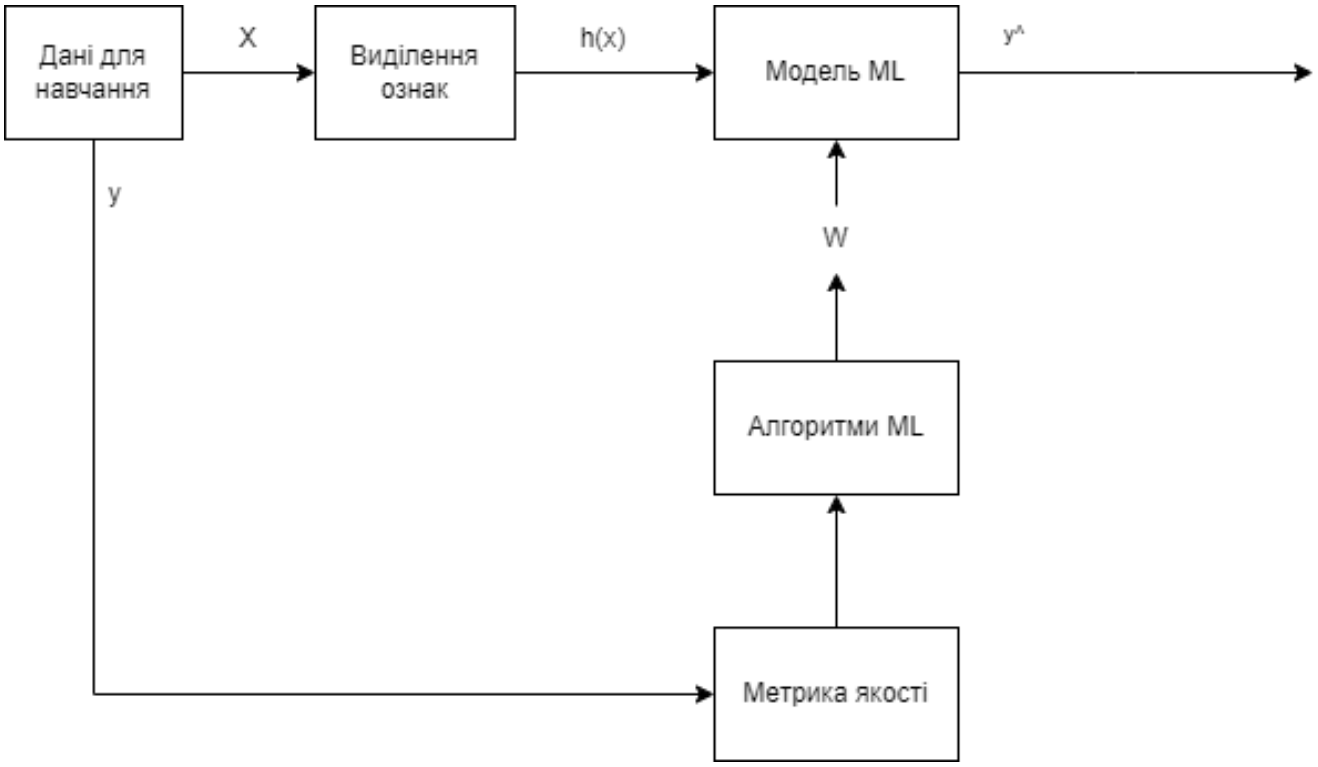



Рис. 2.2 Схема процесу машинного навчання

Дані для навчання позначені як $X$ (вхідні ознаки) і $Y$ (цільові мітки). Блок виділення ознак відповідає за обробку вихідних даних та перетворення їх у формат, зручний для моделі машинного навчання. Функція $h(x)$ представляє цей процес перетворення ознак. Модель машинного навчання приймає перетворенні ознаки і намагається передбачити результат $\hat{y}$ на основі вхідних даних. Алгоритми машинного навчання використовують метрику якості для навчання моделі та налаштовують ваги $\hat{w}$ моделі для оптимізації її роботи. Блок метрика якості, оцінює наскільки добре модель виконує завдання. Метрика порівнює прогнозовані результати $\hat{y}$ з реальними $y$ і відправляє зворотній зв'язок до алгоритму, щоб поліпшити модель [19].

Під час навчання моделі дані зазвичай поділяються у співвідношенні 80:20, тобто 80% — навчальних даних, та 20% — тестових [19].

Переваги контрольованого навчання полягають в його здатності точно передбачати закономірність та приймати рішення на основі даних, включають створення складних моделей для точного прогнозу на основі нових даних, навчальні дані з мітками мають вирішальне значення для того, щоб моделі контрольованого навчання могли ефективно вивчати взаємозв'язки «вихід-вхід» [19].

Однак не зважаючи на переваги цього методу, існують помітні недоліки контрольованого навчання. Зміщення в навчальних даних може призвести до несправедливих прогнозів у контрольованих алгоритмах навчання. Також такий вид навчання значною мірою залежить від позначених навчальних даних, отримання яких може бути трудомістким [19].

Ефективність контрольованого машинного навчання полягає в його здатності узагальнювати навчальні дані в нові, невідомі дані, що робить його безцінним для різноманітних застосунків, від розпізнавання зображень до фінансового прогнозування [19].

Неконтрольоване навчання — аналізує та кластеризує непомічені набори даних за допомогою алгоритмів машинного навчання. Ці алгоритми знаходять приховані закономірності та дані без будь-якого втручання людини. На відмінно від контрольованого навчання, де дані з міткою, алгоритми неконтрольованого навчання мають завдання знаходити закономірність та зв'язки в даних без попереднього знання значення даних. Навчальна модель має лише значення вхідних параметрів і самостійно виявляє групи або шаблони [22].

Неконтрольоване навчання аналізує немарковані дані для виявлення закономірності і зв'язків. Оскільки дані не мають попередньо визначених категорій чи результатів, алгоритм самостійно шукає ці шаблони та зв'язки. Хоч це може бути складним завданням, він здатний виявити інформацію, яка була б недоступною в позначеному наборі даних [22].

Існує три типи алгоритмів неконтрольованого навчання: кластеризація, асоціації та зменшення розмірності [22].

Кластеризація або кластерний аналіз — це групування точок даних на основі їх схожості одна з одною. Метою кластеризації є формування груп однорідних точок даних із різнорідного набору даних. Цей метод оцінює подібність на основі показників, таких як, евклідова відстань, косинусова подібність, манхеттенська відстань, потім на основі цих показників групує точки з найвищим показником подібності [23].

Існує два типи кластеризації:

жорстока кластеризація — у цьому типі кожна точка даних повністю чи ні належить до кластеру [23];

м’яка кластеризація — у цьому типі кожна точка даних оцінюється вірогідність того, що ця точка є цим кластером [23].

На поверхневому рівні кластеризація допомагає аналізувати неструктуровані дані. Елементи що впливають на формування кластерів — побудова графіків, найкоротші відстані і щільність точки. Кластеризація — це процес визначення взаємозв’язків об’єктів на основі міри подібності. Показники подібності легше знайти в менших наборах ознак, але зі збільшенням кількості ознак це стає складнішим. Залежно від типу алгоритму кластеризації, використовуються різні методи для групування даних. Методи кластиризації включають [23]:

1. Кластеризація на основі центроїди (методи поділу).
2. Кластеризація на основі щільності (методи на основі моделі).
3. Кластеризація на основі підключення (ієрархічна кластеризація).
4. Кластиризація на основі розподілу.

Загалом ця техніка використовується для групування даних на основі різних шаблонів, таких як подібності чи відмінності. Ці алгоритми застосовуються для обробки сирих даних, некласифікованих об’єктів даних у групи [23].

Асоціація — це метод неконтрольованого навчання, що використовує для пошуку зв’язків між змінними у великій базі даних. Він призначений для виявлення строгих правил, виявлених у базах даних, використовуючи деякі показники цікавості [24].

Навчання асоціаціями базується на концепції правил, які є наслідком форми (2.1):

$$x \rightarrow y, \tag{2.1}$$

де *X* і *Y* — непересічні набори елементів [24].

Показники, що використовуються в асоціативному навчанн і [24]:

підтримка — ймовірність того, що транзакція містить *X* і *Y*;

впевненість — це міра надійності висновку, зробленого за правилом;

підйом — це відношення спостережуваної підтримки до очікуваної, якби *X* і *Y* були незалежними. Значення підвищення більше 1вказує на те, що присутність X збільшує ймовірність того, що Y також буде присутній в транзакції.

Асоціаційне навчання застосовується в різних областях, зокрема: роздрібна торгівля, охорона здоров’я, інтелектуальний аналіз використання веб-сайтів, фінанси [24].

Хоча асоціативне навчання є потужним інструментом, воно також стикається з кількома проблемами. Велика кількість правил, що може створити асоціативне навчання, можуть бути некорисними або зайвими. Створені правила є статистичними і не обов’язково передбачають причино-наслідковий зв'язок. Також вибір правильної підтримки та порогових значень впевненості може бути складними без знання домену [24].

Зменшення розмірності — це процес зменшення кількості ознак у наборі даних із збереженням якомога більшої кількості інформації. Це можна зробити з метою зменшити складність моделі, покращити продуктивність алгоритму навчання або

полегшити візуалізацію даних. Є кілька методів зменшення розмірності даних, аналіз головних компонентів, сингулярне розкладання і лінійний дискримінантний аналіз [25].

Проблема високої розмірності є поширеною проблемою в машинному навчанні, тому що зі збільшенням кількості функцій, зростає і складність моделі, тому важче знайти хороше рішення. Зменшення розмірності може допомогти пом’якшити ці проблеми шляхом зменшення складності моделі та покращенням її ефективності. Існує два основних підходи для вирішення цієї проблеми: вибір ознак та виділення ознак [25].

вибір ознак — вибір ознак передбачає вибір підмножини оригінальних функцій, які найбільш відповідають проблемі, що розглядається. Мета цього підходу полягає в тому, щоб зменшити розмірність даних, зберігаючи найважливіші характеристики. Є кілька методів вибору ознак, серед яких методи фільтрації, обгортки та вбудовані методи. Методи фільтрації ранжують ознаки за їхньою відповідністю цільовій змінній, методи обгортки використовують продуктивність моделі для вибору ознак, а вбудовані методи поєднують вибір ознак із процесом навчання моделі [25];

виділення ознак — цей метод передбачає створення нових функцій шляхом поєднання або перетворення оригінальних функцій. Мета цього метода полягає у створення набору функцій, який фіксує суть вихідних даних у просторі нижчих розмірів [25].

Метод зменшення розмірності допомагає видали зайві функції, якщо такі є. Сприяє стисненню даних, що зменшує простір для зберігання. Також цей метод може допомогти зменшити складність даних та запобігти переобладнанню. Зменшення розмірності може допомогти підвищити продуктивність моделей машинного навчання шляхом зменшення складності даних. Цей метод можна використовувати як етап попередньої обробки перед застосуванням алгоритмів машинного навчання [25].

Однак застосування такого методу може призвести до втрати деякої кількості даних. Зменшені розміри може бути важко інтерпретувати та важко зрозуміти зв'язок

між оригінальними характеристиками та зменшеними розмірами. Також зменшення розмірності може призвести до втрати корисної інформації [25].

Проблеми неконтрольованого навчання полягають у тому, що важко оцінити ефективність алгоритмів без попередньо визначених міток або категорій. Алгоритми неконтрольованого навчання чутливі до якості вхідних даних. Неповні дані можуть призвести до неточних результатів [25].

Переваги даного методу полягають у тому, що алгоритми неконтрольованого навчання можуть ідентифікувати шаблони та зв’язки в даних, які можуть бути неочевидними для людей. Їх можна використовувати для різних завдань таких як: кластеризація, зменшення розмірності та виявлення аномалій. Неконтрольоване навчання допомагає досліджувати нові дані й отримувати інформацію, яку важко здобути іншими методами [25].

Неконтрольоване навчання застосовують у різних сферах та для розв’язування різноманітних задач таких як: сегментація клієнтів, виявлення шахрайства, системи рекомендацій, обробка природної мови та для аналізу зображення [26].

Таблиця 2.1

Порівняння методів машинного навчання.

| Критерії | Контрольоване навчання | Неконтрольоване навчання |
|---|---|---|
| Мета | Прогнозування або класифікація з використанням міток | Виявлення структур або патернів без міток |
| Тип задачі | Класифікація, регресія | Кластеризація, зменшення розмірності, виявлення аномалій |
| Вхідні дані | Мічені дані | Дані без міток |
| Застосування | Прогнозування майбутніх результатів, розпізнавання | Виявлення прихованих закономірностей |
| Точність результатів | Висока при наявності якісних мічених даних | Менш точне для передбачення конкретних результатів |
| Оцінка якості | Легко оцінюється за допомогою метрик (точність, F1) | Важче оцінювати якість, оскільки відсутні мітки. |

Продовження таблиці 2.1

Порівняння методів машинного навчання.

| Критерії | Контрольоване навчання | Неконтрольоване навчання |
|---|---|---|
| Переваги | Чітке прогнозування або класифікація | Виявлення прихованих патернів в даних |
| Недоліки | Необхідність у великій кількості мічених даних | Важче інтерпретувати, результати можуть бути неточними. |

Для розробки методу виявлення вразливості веб-додатків до SQL-ін’єкцій, найкраще підходить контрольоване машинне навчання. Оскільки, потрібно чітко класифікувати запити на безпечні та небезпечні, для чого потрібні мічені дані. Контрольоване навчання дозволить створити модель з високою точністю та передбачуваністю на основі мічених даних.

Неконтрольоване навчання найкраще підходить для виявлення прихованих патернів, однак для розроблювального методу не є оптимальним підходом, оскільки не має міток для надання коректного результату.

## 2.2 Аналіз алгоритмів машинного навчання

Алгоритм машинного навчання — це набір правил або процесів, які використовуються системою штучного інтелекту для виконання завдань, найчастіше для виявлення нових ідей і шаблонів даних або для прогнозування вихідних значень із заданого набору вхідних змінних [27].

Система навчання алгоритму машинного навчання поділяється на три основні частини:

1. Процес прийняття рішень — алгоритми машинного навчання, за допомогою якого моделі аналізують дані, визначають шаблони та приймають оптимальні

рішення на основі заданих цілей і критеріїв. Алгоритм вироблятиме оцінку шаблону в даних на основі вхідних даних, які можуть бути з мітками або без міток. [27].

2. Функція помилок — використовується для вимірювання відмінності між прогнозованими значеннями моделі та фактичним значенням у тренувальних даних. Допомагає алгоритму оптимізувати свої параметри для досягнення максимальної точності [27].
3. Процес оптимізації моделі — процес налаштування параметрів моделі для досягнення максимальної точності, мінімізації функції помилок або підвищення ефективності виконання задачі. Алгоритм повторюватиме процес оцінки та оптимізації, автономно оновлюючи ваги, доки не буде досягнуто порогу точності [27].

Одним із різновидів алгоритмів машинного навчання, є лінійна регресія. Лінійна регресія — це тип контрольованого машинного навчання, який обчислює лінійний зв'язок між залежною змінною та однією або декількома незалежними ознаками шляхом підгонки лінійного рівняння до даних спостереження [28].

Лінійна модель аналізує набір точок даних з відомими вхідними та вихідними значеннями і знаходить лінію, яка найкраще підходить до цих точок. Ця лінія, відома як «лінія регресії», служить моделлю для прогнозування. Використовуючи цю лінію, можна оцінити або передбачити вихідне значення (Y) для даного вхідного значення (X) [28].

Лінійна регресія в основному використовується для прогнозування моделювання, а не для категоризації. Аналізуючи нахил і перетин лінії регресії, можна отримати уявлення про зв'язок між змінними та роботи прогнози на основі цього розуміння [28].

Існує два основних типи лінійної регресії: проста лінійна регресія та множинна лінійна регресія.

Проста лінійна регресія полягає в тому, що присутня лише одна незалежна змінна, і модель має знайти її лінійний зв'язок із залежною змінною. У простій лінійній регресії є дві основні змінні: залежна змінна Y та незалежна змінна X [11]. Рівняння лінійної регресії таке (2.2):

$$y = mx + c, \tag{2.2}$$

де $y$ — залежна змінна;

$x$ — незалежна змінна;

$m$ — кут нахилу;

$c$ — точка перетину $y$ [11].

У множинні лінійній регресії є більш ніж одна незалежна змінна для моделі, щоб знайти зв'язок. Множинна лінійна регресія розширює концепцію простої лінійної регресії, щоб врахувати більше ніж одну змінну, дозволяючи проаналізувати, як багато факторів впливають на залежну змінну [11]. Рівняння множинної лінійної регресії виглядає наступним чином (2.3):

$$y = b_0 + b_1 X_1 + b_2 x_2 + \cdots + b_n x_n, \tag{2.3}$$

де y — залежна змінна;

$x_1, x_{2,\ldots}, x_n$ — незалежні змінні;

$b_1, b_2, \ldots, b_n$ — коефіцієнти, що описують зв'язок між кожною незалежною змінною та залежною змінною [11].

Лінійна регресія працює шляхом знаходження найкращої лінії через набір точок даних. Цей процес передбачає наступні етапи [28]:

1. На першому етапі обирається відповідне лінійне рівняння для опису зв’язку між залежною та незалежною змінними [28].

2. На другому етапі використовується метод, який відомий як Звичайні найменші квадрати (OLS), щоб мінімізувати суму квадратів різниць між спостережуваними значеннями та значеннями, передбаченими моделлю. Це досягається шляхом коригування нахилу та перетину лінії, щоб знайти оптимальне прилягання. Метою методу є зменшення помилки або різниці між прогнозованими та фактичними значеннями. Цей процес налаштування є ключовою частиною керованого машинного навчання, де модель навчається на навчальних даних [28].
3. На третьому етапі якість відповідності оцінюється за допомогою таких показників, R-квадрат, який вимірює частку дисперсії залежної змінної, яку можна передбачити на основі незалежних змінних [28].

Лінійну модель застосовують у різних сферах, у таких, як економіка та фінанси для прогнозування цін на ринку, та аналіз попиту та пропозиції, оцінка ризиків. У медицині лінійну регресію використовують для передбачення розвитку хвороб, аналізу ефективності лікування. Соціальні науки використовують лінійну модель, для аналізу поведінки споживачів, дослідження впливу соціальних факторів на різні явища [28].

Логістична регресія — алгоритм контрольованого машинного навчання, який використовується для задач класифікації. Логістична регресія оцінює ймовірність належності зразка до певного класу. На практиці його зазвичай використовують для групування вхідних даних у дві категорії: основний клас та не основний клас [29].

Логістична регресія використовує сигмоїдну функцію, щоб відобразити прогнози та їхню ймовірність. Сигмоїдна функція відноситься до S-подібної кривої, яка перетворює будь-яке дійсне значення в діапазон від 0 до 1. Якщо результат сигмоїдної функції перевищує попередньо визначене порогове значення на графіку, модель передбачає, що примірник належить до цього класу. Якщо оцінена ймовірність менша за попередньо визначений поріг, модель передбачає, що примірник не належить до цього класу [29].

Сигмоїдна функція називається функцією активації логістичної регресії та визначається як (2.4):

$$f(x) = \frac{1}{1+e^{-x}} \ . \tag{2.4}$$

Логістична регресія визначається як (2.5):

$$y = \frac{e^{(b_0+b_1x)}}{1+e^{(b_0+b_1x)}}, \tag{2.5}$$

де $x$ — вхідне значення або вхідна змінна,

$y$ — прогнозований результат;

$b_0$ — зміщення або коефіцієнт перетину;

$b_1$ — коефіцієнт для вхідного значення [29].

Властивості логістичної регресії включають:

– залежна змінна логістичної регресії підпорядковується розподілу Бернулі [29];
– прогнозування базується на методі максимальної правдоподібності [29];
– придатність моделі оцінюється за допомогою коефіцієнта конкордантності [29].

Логістична регресія поділяється на бінарну, мультиномінальну та порядкову. Кожен тип відрізняється виконанням та теорією [29].

Бінарна логістична регресія передбачає зв'язок між незалежними та двійковими залежними змінними. Прикладами виходу цього типу можуть бути успіх/невдача, 0/1, true/false [29].

В мультиномінальній логістичній регресії категоріальна залежна змінна має два або більше дискретних результатів. Цей тип регресії має більше двох можливих результатів [29].

Порядкова логістична регресія застосовується, коли залежна змінна знаходиться в упорядкованому стані. Залежна змінна визначає порядок із двома або більше категоріями чи рівнями [29].

Отже, логістична регресія є ефективним методом для задач класифікації, який оцінює ймовірність належності зразка до певного класу. Цей алгоритм є простим в реалізації, легко інтерпретується та підходить для аналізу лінійних залежностей між змінними. Однак її обмеження включають чутливість до викидів і низьку ефективність у складних нелінійних задачах [29].

Наївний Байєс — ефективний метод класифікації, що базується на теоремі Байєса. Його називають “наївним”, через те, що він передбачає, що всі ознаки є незалежними одна від одної, що на практиці рідко відповідає дійсності. Однак, модель часто демонструє високу продуктивність у реальних задачах, особливо за умови великого обсягу даних [30].

Теорема Баєса визначає ймовірність події, яка відбудеться, враховуючи ймовірність іншої події, яка вже відбулась. Математично теорема Баєса формулюється як таке рівняння (2.6):

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \tag{2.6}$$

де A і B — події;

$P(B) \neq 0$ [30].

Існує три типи наївної моделі Баєса:

наївний баєсівський класифікатор Гаусса — у Гауссовому наївному Баєсові безперервні значення, пов’язані з кожною ознакою, вважаються такими, що розподілені за гауссовим розподілом. Гауссовий розподіл також називається

нормальним розподілом. Цей підхід використовується для класифікації, де дані оцінюються під припущенням нормального розподілу для більшої точності [30];

багаточлен наївного Баєса — вектори ознак представляють частоти, з якими певні події були згенеровані мультиномінальним розподілом. Це модель події, яка зазвичай використовується для класифікації документів [30];

наївний Баєс Бернуллі — у багатовимірній моделі подій Бернуллі ознаки є незалежними булевими змінними, що описують вхідні дані. Ця модель популярна для завдань класифікації документів, де використовуються ознаки на основі наявності термів, тобто, чи присутнє слово в документі чи ні [30].

Класифікатор Баєса простий у реалізації та ефективний з точки зору обчислень та у випадках з великою кількістю ознак. Добре працює за наявності категорійних ознак та навіть з обмеженими тренувальними даними. Незважаючи на свої надто спрощені припущення, наївний класифікатор Баєса досить добре працює в багатьох реальних ситуаціях, відомих як класифікація документів і фільтрація спаму. Йому потрібна невелика кількість навчальних даних для оцінки необхідних параметрів. Ефективність, швидкість та здатність працювати з обмеженими даними робить наївний класифікатор Баєса цінним в реальних сценаріях, компенсуючи його наївне припущення про незалежність [30].

Дерево рішень — алгоритм машинного навчання, який використовується для задач класифікації та регресії. Це ієрархічна структура, де рішення приймається послідовно, починаючи з кореня і розгалужуючись до листових вузлів. Кожен вузол представляє перевірку певної умови на одній із ознак, а кожен листовий вузол відповідає результату [31].

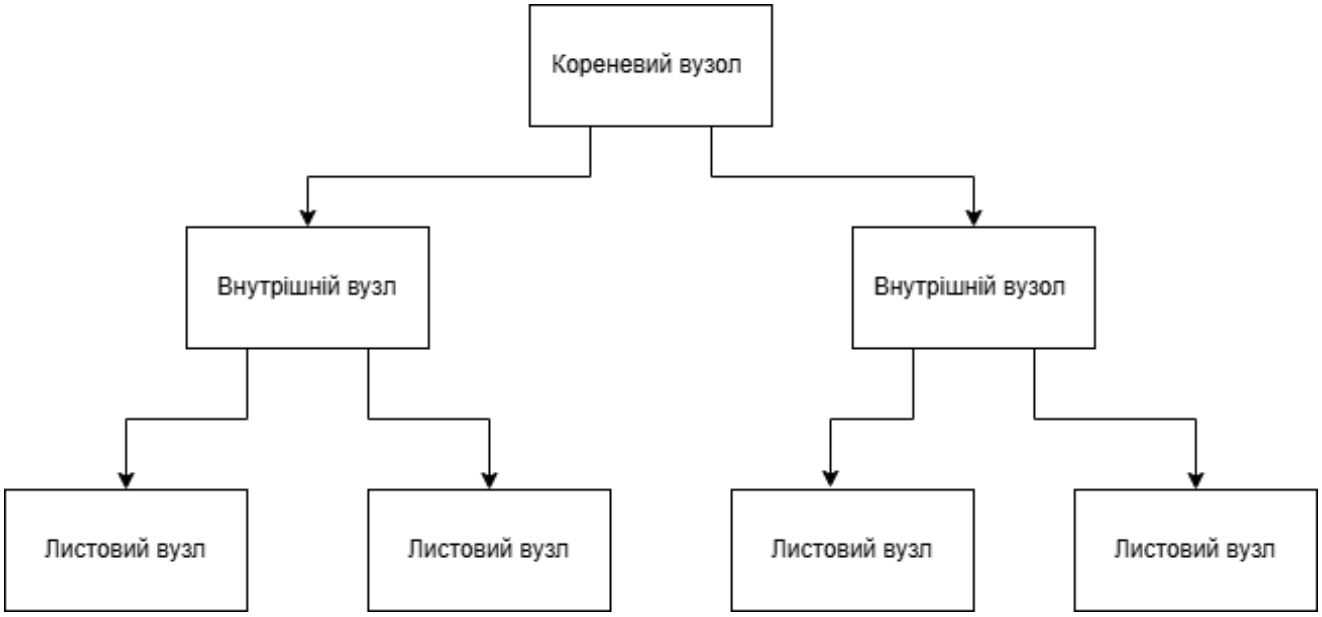



Рис. 2.3 Дерево рішень

На рисунку 2.3 зображено структуру дерева рішень, яка складається з кореневого вузла, внутрішніх вузлів та листових вузлів. Кореневий вузол починає процес поділу, внутрішні вузли представляють умови для поділу даних, а листові вузли містять результати.

Процес навчання дерева рішень використовує стратегію «розділяй або володарюй», виконуючи пошук для визначення оптимальних точок розбиття у дереві. Процес розбиття повторюється зверху вниз, поки всі або більшість записів не буде класифіковано під певними класовими мітками. Чи будуть усі точки даних класифіковані як однорідні набори, багато в чому залежить від складності дерева рішень. Менші дерева легше досягають чистих листових вузлів, тобто точок даних в одному класі. Однак, коли дерево зростає, утримувати цю чистоту стає важко, що часто призводить до фрагментації даних та перенавчання. Щоб зменшити складність і запобігти перенавчанню зазвичай застосовують обрізку, яка видаляє гілки з низькою важливістю ознак. Придатність моделі потім оцінюють за допомогою перехресної перевірки. Ще один спосіб зберегти точність дерев рішень — створення ансамблю за

допомогою алгоритму випадкового лісу, що забезпечує точніші результати, особливо коли окремі дерева не корелюють між собою [31].

Показники, що використовуються в алгоритмі дерево рішень для вибору оптимальних точок розділення даних у вузлах дерева, наступні: індекс Джині, ентропія та інформаційний приріст [31].

Індекс Джині вимірює ймовірність неправильної класифікації нового екземпляра, якщо його було випадково класифіковано відповідно до розподілу класу у наборі даних. Формула індексу Джині (2.7):

$$Gini = 1 - \sum_{i=1}^{n}(p_i)^2 \ , \tag{2.7}$$

де $p_i$ — ймовірність належності елемента до класу $i$;

$k$ — кількість класів [31].

Ентропія вимірює кількість невизначеності або домішки в наборі даних. Вона показує, наскільки різноманітними є класи у вибірці. Формула ентропії (2.8):

$$Entropy = \sum_{i=1}^{n} p_i \log_2(p_i), \tag{2.8}$$

де $p_i$ — ймовірність класифікації екземпляра в певний клас [31].

Інформаційний приріст вимірює зменшення ентропії або індекс Джині, як набір даних розділених на атрибути. Формула інформаційного приросту (2.9):

$$IG(T, X) = Entropy(T) - \sum_{i=1}^{k} \frac{|T_i|}{|T|} * Entropy(T_i), \tag{2.9}$$

де $T$ — поточний вузол;

$T_i$ — вузол після розділення;

$k$ — кількість вузлів після розділення [31].

Ієрархічна структура дерева рішень, дозволяє чітко визначити ознаки що найбільше впливають на прийняття рішень. Алгоритм здатний працювати з різними

типами даних, включаючи дискретні й безперервні значення, які можна поділити на категорії за допомогою встановлення порогів. Цей алгоритм використовують як для задач класифікації, так і для регресії, через що він є універсальним і менш залежним від взаємозв'язків між ознаками [31].

Дерева рішень, є ключовим інструментом в машинному навчанні, вони моделюють і прогнозують результати на основі вхідних даних за допомогою деревоподібної структури [31].

К-найближчий сусід (KNN) — це алгоритм контрольованого навчання, що використовується для завдань класифікації та прогнозного моделювання. Назва алгоритму відображає його підхід до класифікації результату на основі його близкості до інших точок даних на графіку. Цей алгоритм є одним із найпопулярніших та найпростіших класифікаторів регресії, які на сьогодні використовуються в машинному навчанні [32].

Метою алгоритму k-найближчого сусіда полягає в прогнозуванні класу або значення нового зразку на основі його схожості з найближчими сусідами в наборі даних. Для класифікації алгоритм визначає клас нового зразка через голосування класів k-найближчих сусідів. У випадку регресії алгоритм прогнозує значення, обчислюючи середнє значення числових результатів k-найближчих сусідів. KNN є ефективним методом, що базується на ідеї, що схожі точки даних мають подібні властивості [32].

Найбільш часто використовувана міра відстані, яка обмежена дійсними векторами — евклідова відстань. Вона вимірює пряму лінію між точкою запиту та іншою точкою, що вимірюється. Евклідова відстань визначається наступною формулою (2.10):

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}, \tag{2.10}$$

де *x* та *y* — два n-вимірні вектори (точки в n-вимірному просторі);

$x_i$ і $y_i$ — координати точок x та y за *i*-ю ознакою [33].

Ще один популярний показник відстані, який вимірює абсолютне значення між двома точками — Манхеттенська відстань. Її ще називають відстанню таксі або відстанню міського кварталу, оскільки її зазвичай візуалізують за допомогою сітки, що ілюструє, як можна пересуватись з однієї адреси на іншу вулицями міста [33]. Визначається наступною формулою (2.11):

$$d(x, y) = (\sum_{i=1}^{n} |x_i - y_i|), \quad (2.11)$$

де *x* та *y* — два n-вимірні вектори (точки в n-вимірному просторі);

$x_i$ і $y_i$ — координати точок x та y за *i*-ю ознакою [33].

Відстань Мінковського — це узагальнена евклідової та манхеттенської відстаней [33]. Визначається за формулою (2.12):

$$d(x, y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}, \quad (2.12)$$

де *x* та *y* — два n-вимірні вектори (точки в n-вимірному просторі);

$x_i$ і $y_i$ — координати точок x та y за *i*-ю ознакою;

$p$ — параметр, що визначає тип відстані [33].

При різних значеннях *p* відстань Мінковського набуває різних форм. Призначені $p$ =1, вона перетворюється на манхеттенську відстань, при $p$ = 2 на евклідову відстань. Якщо $p \to \infty$, вона стає максимальною відстанню або відстанню Чебишева [33].

Значення k є дуже важливим для алгоритму KNN, оскільки воно визначає кількість сусідів в алгоритмі. Значення k найкраще обирати на основі вхідних даних. Якщо у даних багато шуму або аномальних значень, варто обрати більше значення k, для того щоб зменшити вплив цих аномалій на результат класифікації. Непарне значення k допомагає уникнути ситуації, коли декілька класів мають однакову кількість голосів, що ускладнює визначення класифікації. Для експериментального

вибору оптимального значення k, можна використати метод крос-валідації, щоб забезпечити найкращі результати класифікації [33].

Як і будь-який алгоритм машинного навчання, KNN має свої сильні і слабкі сторони. Враховуючи простоту й точність алгоритму, його легко реалізувати. В разі додаванню нових навчальних даних, алгоритм налаштовується з урахуванням будь-яких нових даних, оскільки всі навчальні дані зберігаються в пам'яті. Гіперпараметри які KNN вимагає, це k та метрики відстані, що є не багато порівняно з іншими алгоритмами машинного навчання [33].

Однак, KNN має слабкі сторони, такі як погане масштабування. Він займає більше пам'яті порівняно з іншими класифікаторами. Також він погано працює з вхідними даними великої розмірності. Це інколи, ще називають феноменом піку, коли після того, як алгоритм досягає оптимальної кількості ознак, додаткові ознаки збільшують кількість помилок класифікації [33].

Машина опорних векторів (SVM) — це алгоритм контрольованого машинного навчання, який класифікує дані, знаходячи оптимальну лінію або гіперплощину, що максимізує між кожним класом у N-вимірному просторі [34].

Алгоритм SVM може виконувати як лінійне, так і не лінійне завдання класифікації, через що його широко використовують в машинному навчанні. Однак, якщо дані не є лінійно роздільними, використовуються ядрові функції для перетворення даних у простір вищої вимірності, що дозволяє здійснити лінійне розділення [34].

Здатність SVM знаходити гіперплощину, яка максимально розділяє різні класи даних є його головною перевагою. Це особливо корисно при роботі зі складними даними та дозволяє проводити дуже точну класифікацію. SVM може обробляти дані з великою кількістю шуму та аномалій, це робить його стійким до таких типів збою [34].

Для того щоб знайти межу рішень, яка максимально розділяє різні класи, SVM використовує опорні вектори, оскільки ці точки мають найбільший вплив на її

положення. Відстань від межі рішення до найближчих опорних векторів називається запасом. Алгоритм намагається максимізувати запас, для того щоб створити більш надійну та узагальнену модель [34].

Існує кілька типів алгоритму SVM, які використовуються для вирішення різних типів проблем і наборів даних [35].

1. Лінійний SVM — найпростіший тип алгоритму, використовується для бінарної класифікації, коли точки даних можна розділити прямою лінією [35].
2. Нелінійний SVM — цей тип використовується для класифікації, де точки даних не можна розділити прямою лінією. Для того щоб перетворити дані у простір з більшою вимірністю, щоб знайти лінійну межу, використовує нелінійну функцію ядра [35].
3. SVM з м’якими межами — у деяких випадках точки даних можуть бути невіддільними, і алгоритм може неправильно класифікувати. SVM з м’якими межами допускає деяку неправильну класифікацію та може бути надійнішим і більш узагальненим, ніж SVM з жорстокими межами [35].
4. SVM для регресії — SVM також використовується для завдань регресії, метою якої є прогнозування постійного значення. Алгоритм для регресії намагається якомога точніше підібрати точки даних, зберігаючи хороший запас [35].
5. Багатокласовий SVM — цей тип алгоритму виконує завдання класифікації з більш ніж двома класами, використовуючи комбінацію стратегій «один проти одного» та «один проти решти» щоб класифікувати дані за кількома класами [35].

SVM — потужний алгоритм машинного навчання, який підходить як для завдань класифікації, так і для регресії. Адаптивність алгоритму SVM через функції ядра дозволяє ефективно обробляти як лінійні, так і не лінійні дані. Однак при

використанні даного алгоритму слід врахувати такі проблеми, як налаштування параметрів і можливий повільний час навчання на великих наборах даних [35].

Рекурентна нейронна мережа (RNN) — це модель глибокого навчання, яка навчена обробляти та перетворювати послідовні вхідні дані в певні послідовні вихідні дані [36].

На рисунку 2.4 зображено схема RNN.

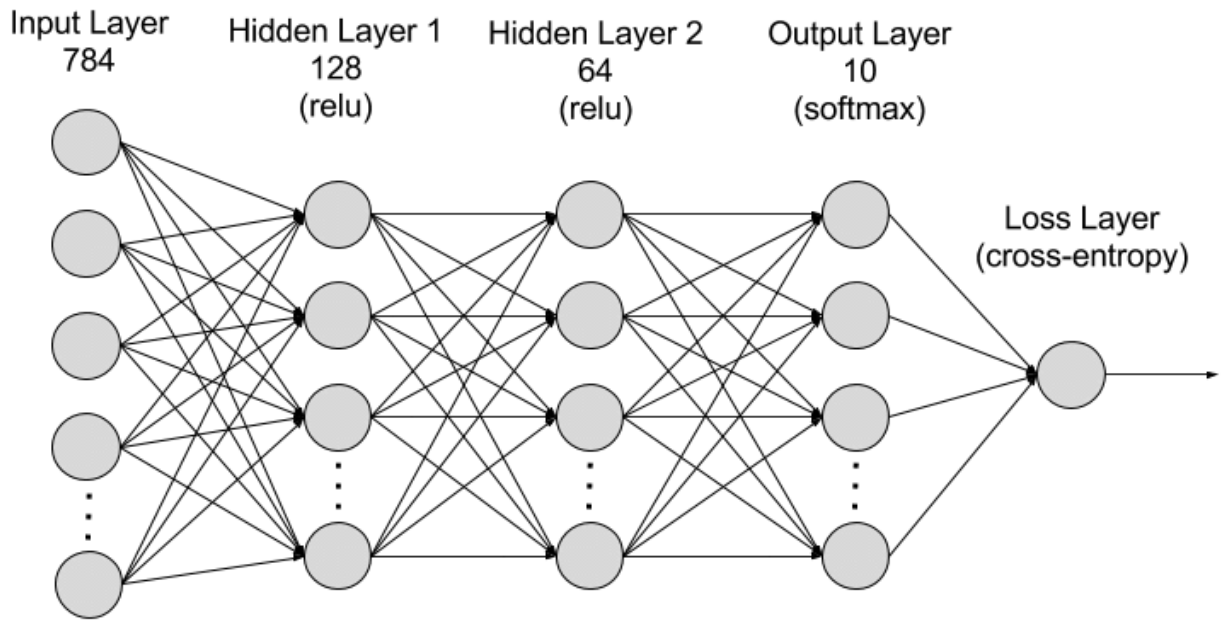



Рис. 2.4 Схема RNN [36]

Рекурентна нейронна мережа складається з нейронів: вузлів обробки даних, які працюють разом для виконання складних завдань. Вхідний рівень отримує інформацію для обробки, а вихідний рівень забезпечує результат. Обробка даних, аналізу і прогнозування відбувається в прихованому шарі [37].

RNN передає поетапно послідовні дані до прихованих шарів. Крім того, алгоритм має самоциклічний або повторюваний робочий процес: прихований рівень здатний запам’ятовувати та використовувати попередні вхідні дані для майбутніх прогнозів завдяки короткочасній пам’яті [37].

Функція активації — це математична функція, яка додається до виходу кожного шару нейронної мережі, щоб ввести не лінійність і дозволити мережі навчатися складнішим шаблонам у даних. Без функції активації, рекурентна мережа могла б лише виконувати лінійні перетворення вхідних даних, що унеможливлювало б її обробку нелінійних задач. Нелінійність є важливою для навчання та моделювання складних шаблонів, особливо в таких задачах, як аналіз часових рядів і послідовне прогнозування даних. Функція активації контролює величину виходу нейронів, зберігаючи значення в певному діапазоні, що допомагає запобігти занадто великому або малому зростанню значення під час переходу вперед і назад. Рекурентній нейронній мережі, функція активації застосовується на кожному кроці часу до прихованих станів, керуючи тим, як мережа оновлює свою внутрішню пам'ять на основі поточного введення та минулих прихованих станів [37].

Рекурентну нейрону мережу часто характеризують архітектурою «один до одного», тобто одна вхідна послідовність пов’язана з одним виходом. Однак існує ще три типи архітектури RNN: один до багатьох, багато до одного та багато до багатьох [37]:

один до багатьох — цей тип архітектури представляє сценарій, коли мережа отримує один вхід, але генерує послідовність виходів. Тобто RNN приймає одну інформацію як вхідну, обробляє вхідні дані та генерує послідовність виходів з часом. Довжина цієї послідовності залежить від конкретного завдання [37];

багато до одного — така архітектура відноситься до певного типу RNN, де мережа обробляє послідовність вхідних даних, але створює єдиний вихід. Тобто, RNN приймає послідовність точок даних протягом певного часу, після обробки всієї послідовності генерує єдине вихідне значення [37];

багато до багатьох — така архітектура описує сценарій, коли мережа обробляє послідовність вхідних даних і генерує відповідну послідовність вихідних даних. Тобто і вхід, і вихід мають кілька елементів, оброблених протягом певного часу [37].

Рекурентні нейроні мережі можуть запам’ятовувати попередні входи та виходи, оскільки вони здатні вчитись на минулому досвіді, вони можуть робити точні прогнози. Також мережі RNN розуміють часовий аспект даних, що робить їх ідеальними для обробки послідовних даних [37].

## 2.3 Порівняння алгоритмів машинного навчання для виявлення SQL-ін’єкцій

В останні роки виявленню SQL-ін’єкцій за допомогою машинного навчання приділяється все більше уваги. Методи машинного навчання пропонують нові можливості для аналізу великої кількості даних, виявлення аномальних шаблонів запитів, які можуть вказувати що це SQL-ін’єкція. Для визначення найефективнішого підходу, який забезпечить високу точність та швидкість виявлення загроз у реальному часі було обрано для порівняння та аналізу наступні алгоритми:

- логістична регресія;
- дерево рішень;
- KNN;
- Linear SVM;
- рекурентна нейрона мережа (RNN);
- Gaussian Naive Bayes.

Для порівняння ефективності алгоритмів машинного навчання у виявлені SQL-ін’єкцій було сформовано набір даних, який складається з 30 тисяч шаблонів запитів, шкідливих та не шкідливих. Такий обсяг дозволить більш точно навчити моделі та підвищити їх здатність розпізнавати небезпечні шаблони запитів. Дані були розділено на тренувальну та тестову вибірку, у співвідношенні 80:20. Тобто 80% даних використовуються для навчання моделі, а 20% — для тестування та оцінки точності моделі. Розділення даних є важливим кроком у процесі моделювання даних для навчання моделі. Це допоможе оцінити продуктивність і точність моделей на різних

наборах даних та уникнути перенавчання або недонавчання. Перенавчання відбувається, коли модель добре вивчає навчальні дані, але не може узагальнювати для нових або невідомих даних. Недонавчання, виникає, коли модель навпаки недостатньо вчиться на навчальних даних і не може вловити основні закономірності або зв'язки.

```
X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2, random_state=42)
```

Рис. 2.5 Розділення даних для навчання та тестування

Для обробки текстових даних, використовувався метод TfidfVectorizer. За допомогою цього методу текст запитів було перетворено на числові вектори з використанням методу TF-IDF, це дозволить виділити важливі слова у кожному запиті [38].

TfidfVectorizer працює наступним чином:

частота термінів (TF) — вимірює, як часто слова з'являються в документі, та припускає, що чим частіше термін з'являється в документі, тим він важливіший[38];

інверсна частота документів (IDF) — вимірює важливість термінів, враховуючи, як часто він з'являється в усіх документах у наборі даних. Чим більше документів містить термін, тим менш важливим він є. Значення IDF зменшується зі збільшенням кількості документів, що місять цей термін[38];

TF-IDF — становить добуток TF і IDF, оцінює наскільки важливий термін у конкретному документі, одночасно зменшуючи вагу типових термінів, які є менш інформативними [38].

Оцінка TF-IDF обчислюється за формулою (2.13):

$$tf - idf(t,d) = tf(t,d) * idf, \tag{2.13}$$

де — *tf(t,d)* це частота термінів *t* у документі *d*;

*idf(t)* є зворотною частотою документа терміну *t*, та розраховується за формулою (2.14) [38]:

$$idf(t) = \log(\frac{N}{1+df(t)}) , \quad (2.14)$$

де — *N* загальна кількість документів;

*df(t)* кількість документів, що містять термін *t* [38].

Перший алгоритм, що використовується для порівняння — це алгоритм K-найближчих сусідів. Він належить до простих і інтуїтивно зрозумілих алгоритмів класифікації та регресії.

Для класифікації, в першу чергу, потрібно визначити набір гіперпараметрів, що найкраще підходить для класифікатора. Підбір гіперпараметрів, відбувався за методом GridSearch. Цей метод використовується для автоматичного підбору найкращих гіперпараметрів для моделей машинного навчання (Рисунок 2.5).

```
knn_params = {
    'n_neighbors': [3, 5, 7],
    'weights': ['uniform', 'distance']
}
knn_gs = GridSearchCV(KNeighborsClassifier(), knn_params, scoring='accuracy', cv=5)
knn_gs.fit(X_train, y_train)
print("Best parameters for KNN:", knn_gs.best_params_)
evaluate_model(knn_gs.best_estimator_, X_test, y_test, "KNN (Best)")
```

```
Best parameters for KNN: {'n_neighbors': 3, 'weights': 'distance'}
```

Рис. 2.5 Оптимальний набір гіперпараметрів для класифікатора KNN

Для оцінки класифікатора було використано стандартні метрики класифікації, а саме:

точність (accuracy) — для загальної оцінки результатів;

повнота (Recall) та точність виявлення (Precision) — для оцінки здатності алгоритма виявляти вразливості без пропуску пеальних загроз;

f1-міра — для збалансування оцінки між точністю та повнотою.

На рисунку 2.6 зображено оцінку класифікатора KNN.

```
KNN (Best) Results:
Accuracy: 0.9387455080039203
Precision: 0.9429161968736643
Recall: 0.9387455080039203
F1 Score: 0.9376088214755789
```

Рис. 2.6 Оцінка класифікатора KNN

Для вимірювання продуктивності моделі класифікації та для відображення кількості точних і не точних випадків на основі прогнозів моделі, використовують матрицю помилок. Матриця помилок для класифікатора KNN зображена на рисунку 2.7.

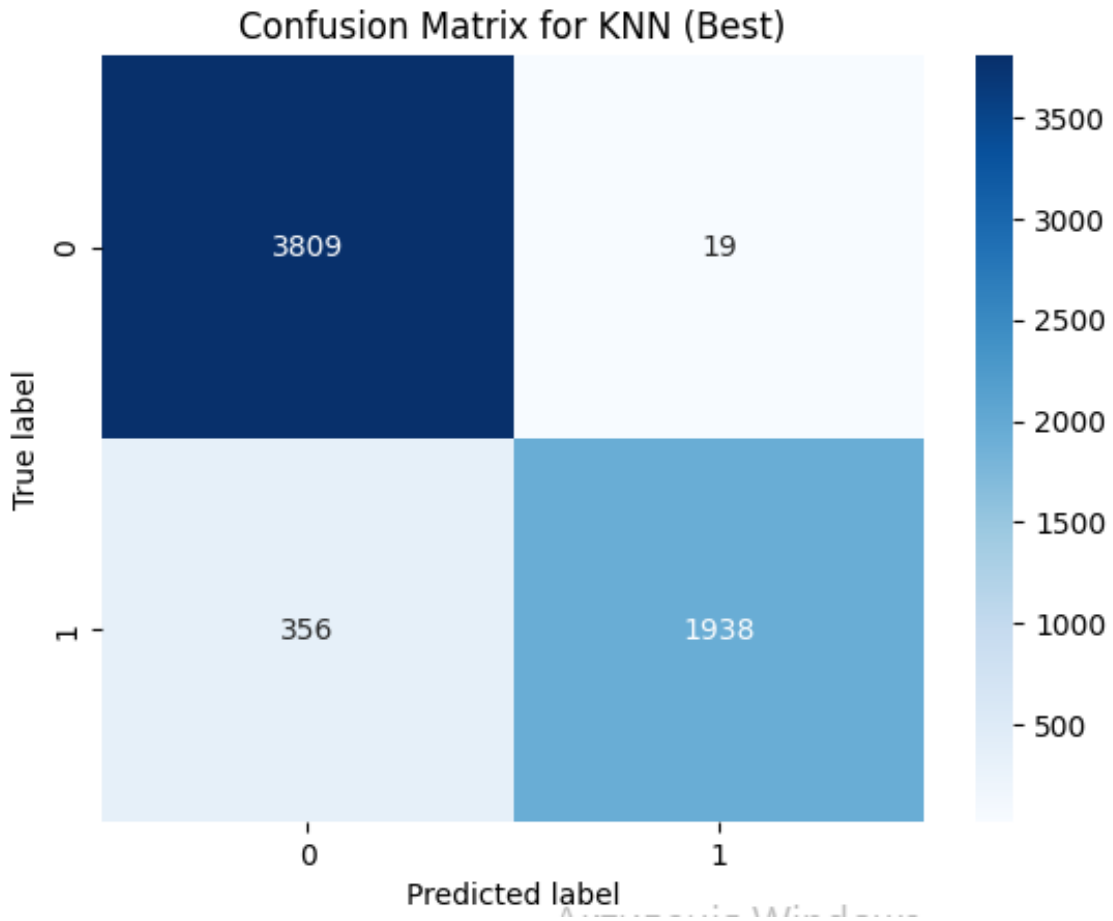



Рис. 2.7 Матриця помилок для класифікатора KNN

Матриця відображає кількість екземплярів, створених моделлю на тестових даних та має чотири основних терміни [39]:

true positive — кількість випадків, коли модель правильно класифікувала позитивний клас [39];

true negative — кількість випадків, коли модель правильно класифікувала негативний клас [39];

false positive — кількість випадків, коли модель помилково класифікувала негативний клас як позитивний [39];

false negative — кількість випадків, коли модель помилково класифікувала позитивний клас як негативний [39].

Матриця помилок показала, що модель правильно класифікує більшість зразків класу 1, що являються SQL-ін'єкціями та показала високу точність — 94% та точність виявлення — 99%.

Наступний алгоритм машинного навчання для аналізу — це Gaussian Naive Bayes. Цей алгоритм базується на теоремі Баєса і застосовується для задач класифікації. Його особливість, полягає в тому, що він передбачає нормальний розподіл значень кожної ознаки.

В першу чергу, визначаємо оптимальні гіперпараметри для оцінки класифікатора.

```
gnb_params = {
    'var_smoothing': [1e-9, 1e-8, 1e-7]
}
gnb_gs = GridSearchCV(GaussianNB(), gnb_params, scoring='accuracy', cv=5)
gnb_gs.fit(X_train, y_train)
print("Best parameters for GaussianNB:", gnb_gs.best_params_)
evaluate_model(gnb_gs.best_estimator_, X_test, y_test, "Gaussian Naive Bayes (Best)")
```

```
Best parameters for GaussianNB: {'var_smoothing': 1e-07}
```

Рис. 2.8 Оптимальні гіперпараметри для класифікатора Gaussian Naive Bayes

Оцінюємо класифікатор Gaussian Naive Bayes за визначеними метриками.

```
Gaussian Naive Bayes (Best) Results:
Accuracy: 0.7856909506697157
Precision: 0.8566243436130402
Recall: 0.7856909506697157
F1 Score: 0.7877474405917254
```

Рис. 2.9 Оцінка класифікатора Gaussian Naive Bayes

Виводимо матрицю помилок.

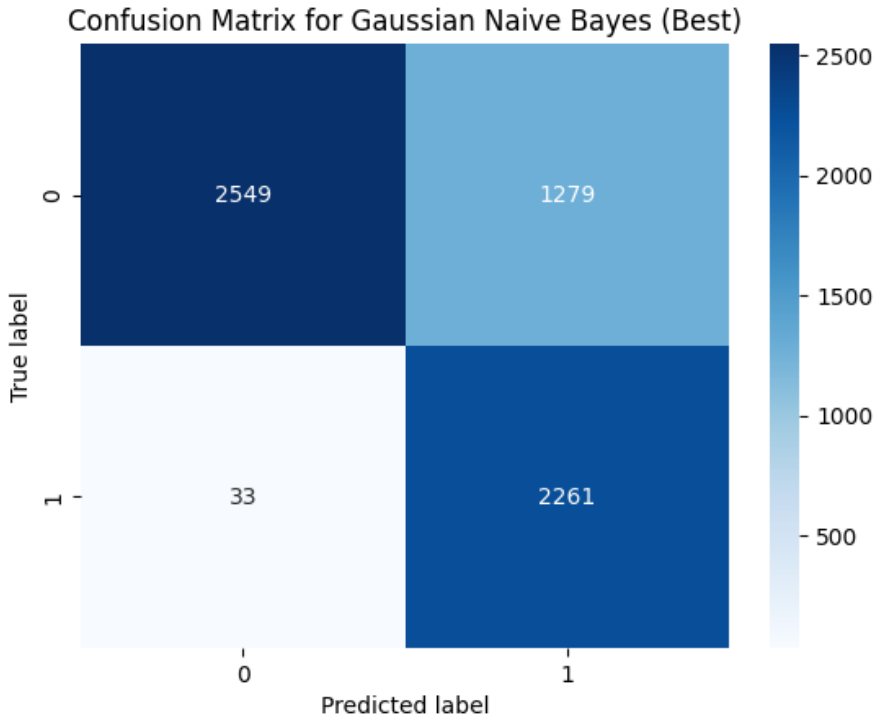



Рис. 2.10 Матриця помилок для класифікатора Gaussian Naive Bayes

Модель добре визначає SQL-ін’єкції, проте має певні проблеми з хибно позитивними прогнозами. Це означає, що модель іноді помилково позначає нормальні запити як атаки, що може призводити до зайвих тривог.

Дерево рішень — цей алгоритм використовується для класифікації та регресії. Він працює шляхом розбиття набору даних на менші підмножини на основі певних ознак або атрибутів, формуючи дерево рішень.

Визначаємо набір оптимальних гіперпараметрів для класифікатора дерево рішень та навчаємо модель.

```
dt_params = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10]
}
dt_gs = GridSearchCV(DecisionTreeClassifier(), dt_params, scoring='accuracy', cv=5)
dt_gs.fit(X_train, y_train)
print("Best parameters for Decision Tree:", dt_gs.best_params_)
evaluate_model(dt_gs.best_estimator_, X_test, y_test, "Decision Tree (Best)")
```

```
Best parameters for Decision Tree: {'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 10}
```

Рис. 2.11 Набір оптимальних гіперпараметрів для класифікатора дерево рішень

Виконуємо оцінку моделі.

```
Decision Tree (Best) Results:
Accuracy: 0.9934661875204182
Precision: 0.9934772908213565
Recall: 0.9934661875204182
F1 Score: 0.9934603960797044
```

Рис. 2.12 Оцінка класифікатора дерево рішень

Створюємо матрицю помилок для класифікатора дерево рішень, для оцінки якості роботи класифікатора.

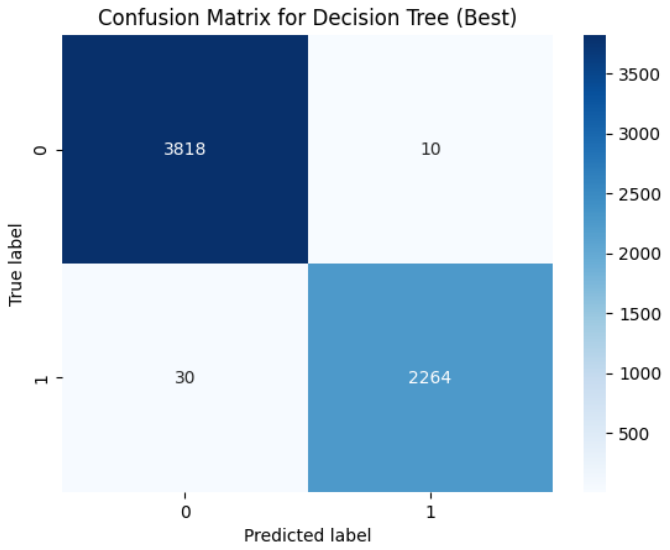



Рис. 2.13 Матриця помилок для класифікатора дерево рішень

Матриця помилок показує, що модель має високу точність, більшість випадків класифікуються правильно. Вона має низький рівень помилок, лише 40 зразків з усіх передбачених були класифіковані не правильно. Оцінка результатів за визначеними метриками, вказують на дуже ефективну роботу моделі дерева рішень, модель досягає високої точності і добре збалансована.

Логістична регресія — алгоритм машинного навчання, що використовується для задач класифікації, особливо бінарної класифікації. Вона є потужним інструментом для розв'язування задач класифікації, що дозволяє зрозуміти вплив різних факторів на ймовірність певного результату.

Визначаємо набір оптимальних гіперпараметрів для класифікатора логістичної регресії та навчаємо модель.

```
lr_params = {
    'C': [0.01, 0.1, 1, 10, 100],
    'solver': ['liblinear', 'lbfgs']
}
lr_gs = GridSearchCV(LogisticRegression(), lr_params, scoring='accuracy', cv=5)
lr_gs.fit(X_train, y_train)
print("Best parameters for Logistic Regression:", lr_gs.best_params_)
evaluate_model(lr_gs.best_estimator_, X_test, y_test, "Logistic Regression (Best)")
```

```
Best parameters for Logistic Regression: {'C': 100, 'solver': 'lbfgs'}
```

Рис. 2.14 Набір оптимальних параметрів для класифікатор логістична регресія

Проведемо оцінку класифікатора.

```
Logistic Regression (Best) Results:
Accuracy: 0.9772950016334532
Precision: 0.977796588439529
Recall: 0.9772950016334532
F1 Score: 0.9771711321964034
```

Рис. 2.15 Оцінка класифікатора

Створюємо матрицю помилок для логістичної регресії, для оцінки якості роботи класифікатора.

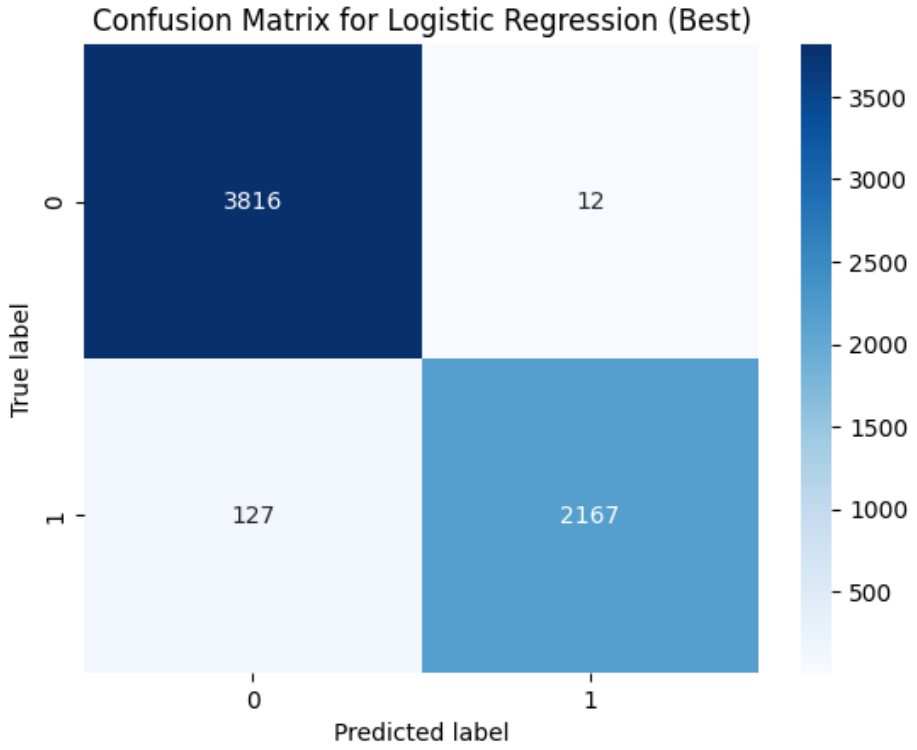



Рис. 2.16 Матриця помилок для класифікатора логістична регресія

Модель логістичної регресії показала високу ефективність у виявленні SQL-ін’єкцій. Загальна продуктивність моделі дуже висока, незважаючи на невелику кількість хибно позитивних і хибно негативних результатів, це робить її придатною для використання у веб-безпеці.

LinearSVC — алгоритм, що використовується для задач класифікації, належить до класу методів опорних векторів. Даний алгоритм швидко навчається та ефективний для великих наборів даних.

На рисунку 2.17 зображено набір оптимальних гіперпараметрів для класифікатора LinearSVC та навчання моделі.

```
linear_svm_params = {
    'C': [0.01, 0.1, 1, 10],
    'max_iter': [1000, 2000, 3000]
}

linear_svm_gs = GridSearchCV(LinearSVC(), linear_svm_params, scoring='accuracy', cv=5)
linear_svm_gs.fit(X_train, y_train)
print("Best parameters for Linear SVM:", linear_svm_gs.best_params_)
evaluate_model(linear_svm_gs.best_estimator_, X_test, y_test, "Linear SVM (Best)")
```

```
Best parameters for Linear SVM: {'C': 1, 'max_iter': 1000}
```

Рис. 2.17 Набір оптимальних гіперпараметрів для класифікатора LinearSVC

Проведемо оцінку ефективності класифікатора.

```
Linear SVM (Best) Results:
Accuracy: 0.9676576282260699
Precision: 0.9684550770165369
Recall: 0.9676576282260699
F1 Score: 0.9674218034849389
```

Рис. 2.18 Оцінка класифікатора LinearSVC

Для оцінки якості класифікації моделі створюємо матрицю помилок.

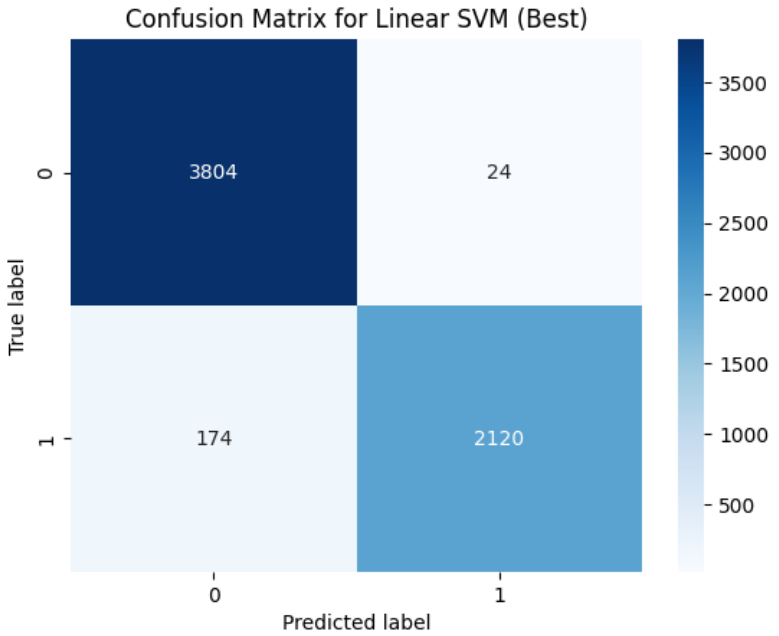



Рис. 2.19 Матриця помилок класифікатора LinearSVC

Модель LinearSVC показує високу точність, прецизійність, повноту та F1, це вказує на добру збалансованість класифікації. Модель успішно виявляє SQL-ін’єкції і майже не допускає хибних спрацювань. Проте, є незначна кількість хибних відхилень, що призводить до пропуску частини випадків класу 1.

RNN — тип нейронної мережі, спеціально розроблений для роботи з послідовними даними, такими як текст, часоряди або аудіо. Він використовує приховані стані для запам’ятовування попередніх вхідних даних, що дозволяє враховувати контекст.

Визначаємо оптимальний набір гіперпараметрів для рекурентної нейронної мережі.

```
early_stopping = EarlyStopping(monitor='val_loss',patience=3)
def build_rnn_model(hp):
    model = Sequential()
    model.add(Embedding(input_dim=5000, output_dim=hp.Choice('embedding_dim', values=[64, 128, 256]), input_length=100))
    model.add(LSTM(hp.Choice('lstm_units', values=[32, 64, 128]), return_sequences=True))
    model.add(LSTM(hp.Choice('lstm_units', values=[32, 64, 128])))
    model.add(Dense(hp.Choice('dense_units', values=[10, 20]), activation='relu'))
    model.add(Dropout(hp.Choice('dropout_rate', values=[0.2, 0.3, 0.5])))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(optimizer=Adam(learning_rate=hp.Choice('learning_rate', values=[1e-3, 1e-4])),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model
```

Рис. 2.20 Оптимальний набір гіперпараметрів для RNN

Виконуємо оцінку моделі.

```
Accuracy: 0.9964
Precision: 0.9974
Recall: 0.9930
F1 Score: 0.9952
```

Рис. 2.21 Оцінка моделі RNN

Для вимірювання продуктивності моделі класифікації та для відображення кількості точних і не точних випадків на основі прогнозів моделі створюємо матрицю помилок.

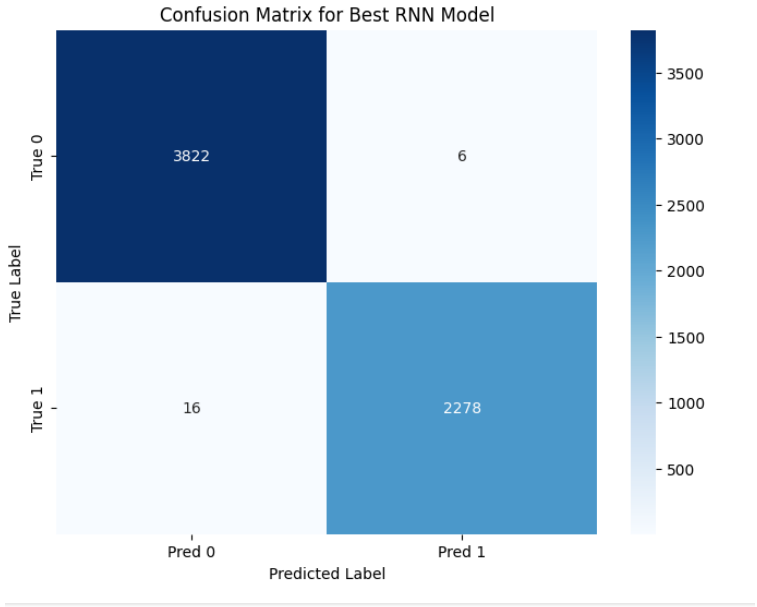



Рис. 2.23 Матриця помилок для класифікатора RNN

Модель RNN має високий рівень точності та добре збалансована у класифікації обох класів. Високі показники прецизійності і повноти роблять її кращим вибором, особливо якщо критично важливо мінімізувати  хибні спрацювання.

Для дослідження було порівняно декілька алгоритмів машинного навчання з метою визначення найефективнішого алгоритму для виявлення SQL-ін’єкцій. Для оцінки ефективності кожного алгоритму використовувались такі показники, як частота помилково позитивних результатів (FPR) та частота істинних позитивних результатів (TPR). FPR показує, скільки випадків було неправильно ідентифіковані, це важливо для мінімізації хибних тривог. TPR у свою чергу, є мірою того, скільки позитивних випадків у наборі даних правильно класифіковано. Порівняння цих показників дозволить визначити, який із алгоритмів забезпечує найкращий баланс між точністю виявлення та мінімізації помилок [40].

Результати порівняння алгоритмів наведено у таблиці 2.2.

Таблиця 2.2

Порівняння алгоритмів машинного навчання

| Назва алгоритму | KNN | Gaussian Naive Bayes | Decision Tree | Logistic Regression | Linear SVM | RNN |
|---|---|---|---|---|---|---|
| Accuracy | 0,939 | 0,785 | 0,993 | 0,977 | 0,967 | 0,996 |
| Precision | 0,942 | 0,856 | 0,993 | 0,977 | 0,968 | 0,997 |
| Recall | 0,939 | 0,785 | 0,993 | 0,977 | 0,967 | 0,993 |
| F1 | 0,937 | 0,787 | 0,993 | 0,977 | 0,967 | 0,995 |
| TP | 1938 | 2261 | 2264 | 2167 | 2120 | 2278 |
| FN | 356 | 33 | 30 | 127 | 174 | 16 |
| FP | 19 | 1279 | 10 | 12 | 24 | 6 |
| TN | 3809 | 2549 | 3818 | 3816 | 3804 | 3822 |
| FPR | 0,004% | 0,33% | 0,002% | 0,003% | 0,006% | 0,001% |
| TPR | 0,84% | 0,98% | 0,98% | 0,94% | 0,92% | 0,99% |

Проаналізувавши таблицю, можемо зробити висновок, що для виявлення SQL-ін’єкцій, найкращою моделлю є RNN. Вона показує кращі результати за точністю і найнижчий рівень помилок, тому вона є оптимальним вибором для вирішення поставлених задач.

## Висновки

У другому розділі було проведено аналіз методів машинного навчання, що включає контрольоване та неконтрольоване навчання. Оскільки виявлення SQL-ін’єкцій є задачею класифікації, найкраще підходять методи контрольованого навчання. Модель контрольованого навчання навчається на основі даних з мітками. Це дозволить алгоритму виявляти патерни та особливості характерні для SQL-ін’єкцій. Методи контрольованого навчання дозволять забезпечити високу точність і надійність у виявлені ін’єкцій.

Було порівняно шість класифікаторів для виявлення SQL-ін’єкцій, а саме: KNN, Gaussian Naive Bayes, Decision Tree, Logistic Regression, Linear SVM та RNN. Для оцінки ефективності кожного алгоритму використовувались такі ключові метрики, як точність, точність позитивного класу, повнота, F1 міра, показник хибнопозитивної частоти (FPR) та частота позитивних результатів (TPR). Було створено таблицю, в які відображається продуктивність кожного алгоритму на основі отриманих результатів.

Результати, свідчать про те, що серед проаналізованих алгоритмів найбільш ефективним для виявлення SQL-ін’єкцій є алгоритм RNN. Він показав найвищі результати точності, найнижчий показник FPR — 0,001% та найвищий показник виявлення атак TPR — 0,99%. Це робить його оптимальним вибором для завдання виявлення SQL-ін’єкцій.

# 3 РОЗРОБКА МЕТОДУ ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ WEB-ДОДАТКІВ ДО SQL-ІН'ЄКЦІЙ З ВИКОРИСТАННЯМ МЕТОДІВ МАШИННОГО НАВЧАННЯ

## 3.1 Опис розробки методу

SQL-ін'єкції виникають через вразливість у самому способі взаємодії додатка з базою даних. Найбільш поширеним прикладом є введення зловмисником додаткового SQL-коду через вхідні поля форми, такі як логін або пошук, де додаток очікує отримати текст, але не перевіряє його на шкідливий вміст. При цьому, якщо запит не захищений, шкідливий код стає частиною SQL-запиту, що виконується на сервері бази даних.

Для мінімізації наслідків атак SQL-ін'єкціями, необхідно створити надійний механізм, що здатний виявляти потенційно небезпечні SQL-запити та блокувати їх ще до виконання. Поєднання машинного навчання і динамічного аналізу коду дозволить забезпечити швидкість та точність виявлення атак, та знизити ймовірність витоків даних і дозволить запобігати атакам ще на етапі передачі запитів.

Комбінований метод буде автоматично розпізнавати підозрілі SQL-запити на основі навченої моделі та перевіряти їх за допомогою інструмента динамічного аналізу, для того щоб підвищити точність виявлення і знизить кількість хибних спрацювань.

Гібридний підхід забезпечить:

високу точність виявлення SQL-ін'єкцій шляхом здійснення первинної класифікації за допомогою машинного навчання та підтвердження за допомогою інструмента динамічного аналізу;

зниження кількості хибних спрацювань шляхом фільтрації запитів, які модель помилково позначила як потенційно небезпечні.

Розробка методу виявлення вразливостей веб-додатків до SQL-ін’єкцій, включає кілька етапів: збір даних, навчання моделі, оцінка моделі, інтеграція з інструментом динамічного аналізу.

На першому етапі — збір даних, було підготовлено великий набір даних, що включає безпечні SQL-запити та шкідливі. Цей етап є ключовим, оскільки різноманітність даних впливає на здатність моделі правильно розпізнавати небезпечні запити в реальних умовах. Було здійснено попередню обробку, щоб модель могла аналізувати текст запитів. Попередня обробка включає:

розбиття SQL-запитів на окремі токени;

очищення від символів, які не впливають на аналіз;

перетворення запитів для зменшення різноманітності, це допоможе моделі зосередитися на структурі запитів.

Моделі машинного навчання працюють лише з числовими даними, тому SQL-запити необхідно перетворити на числові вектори. Для цього був використаний метод Label Encoding, який перетворює категорійні стовпці у числові. На завершальному етапі збору даних потрібно розділити їх на тестовий та тренувальний набір. Дані були розділенні у відношенні 80:20. Де 80% — це тренувальний набір, а 20% — тестовий.

На другому етапі — навчання моделі, для розробки методу була обрана рекурентна нейрона мережа, оскільки вона добре підходить для аналізу послідовних даних та може зберігати інформацію про контекст. Для того щоб краще зберігати контекст на тривалих відстанях, була використана вдосконалена архітектура LSTM.

Навчання моделі здійснювалося на зібраному наборі даних, щоб модель могла розрізняти безпечні та потенційно небезпечні SQL-запити. Для цього була створена архітектура RNN, що включає шари для обробки текстових послідовностей та виявлення шаблонів, характерних для SQL-ін’єкцій.

Після навчання моделі було визначено оптимальний поріг для класифікації SQL-запитів. Для цього було обчислено метрики якості моделі такі, як accuracy, precision, recall, F1 score. За допомогою цих метрик можна зрозуміти як модель поводиться при строгіших або більш гнучких умовах класифікації. F1 використовувався як основний критерій, оскільки він збалансовує precision та recall. В результаті було визначено оптимальний поріг, при якому модель демонструє найкращий баланс. Що дозволить підвищити ефективність виявлення загроз.

На Рисунку 3.1 зображено результат порівняння.

| | Threshold | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| 0 | 0.1 | 0.990199 | 0.990275 | 0.990199 | 0.990212 |
| 1 | 0.5 | 0.994446 | 0.994451 | 0.994446 | 0.994443 |
| 2 | 0.7 | 0.994610 | 0.994629 | 0.994610 | 0.994604 |

Рис. 3.1 Результат аналізу метрик при різних порогах

За результати аналізу визначено оптимальний поріг — 0,7.

На рисунку 3.2 зображено архітектуру моделі.

```
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
X_train_pad, X_test_pad, y_train, y_test = train_test_split(X_pad, y, test_size=0.2, random_state=42)


rnn_model = Sequential()
rnn_model.add(Embedding(input_dim=5000, output_dim=128, input_length=100))
rnn_model.add(LSTM(64, return_sequences=True))
rnn_model.add(LSTM(32))
rnn_model.add(Dense(10, activation='relu'))
rnn_model.add(Dense(1, activation='sigmoid'))

rnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])


early_stopping = EarlyStopping(monitor='val_loss',patience=3)


history = rnn_model.fit(X_train_pad, y_train, epochs=10, batch_size=32,
                        validation_data=(X_test_pad, y_test), callbacks=[early_stopping])
```

Рис. 3.2 Архітектура моделі

Спочатку текстові запити були перетворенні у вектори з фіксованою довжиною за допомогою шару Embedding, це дозволить моделі краще зрозуміти семантичні зв’язки між словами. Два шара LSTM обробляють послідовності, враховуючи залежності між словами, для виявлення структури, що притаманні небезпечним запитам. Шар Dense з функцією активації sigmoid, генерує ймовірність належності запиту до класу небезпечних або безпечних. Така архітектура дозволяє моделі зберігати та аналізувати послідовності символів і слів у SQL-запитах.

На третьому етапі була здійснена оцінка моделі за такими метриками: Accuracy, Precision, Recall, F1 Score.

На Рисунку 3.3 зображено оцінка моделі.

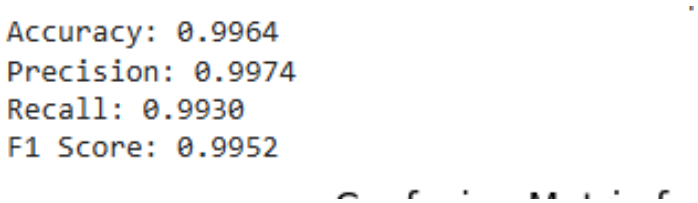



Рис. 3.3 Оцінка моделі

На четвертому етапі здійснюється інтеграція з інструментом динамічного аналізу, таким як OWASP ZAP, для створення двоетапної системи. Спершу модель аналізує SQL-запит та визначає його безпечність. Якщо запит позначений як підозрілий, він передається інструменту динамічного аналізу для детальної перевірки.

На Рисунку 3.4 зображено алгоритм роботи методу. Спочатку веб-додаток надсилає SQL-запит на обробку. Прийнятий запит проходить етап токенізації та перетворення в векторне представлення для обробки моделлю. Модель виконує класифікацію запиту, який класифікується як «безпечний» або «підозрілий». Якщо запит позначено як «безпечний», повертається результат «безпечний» і відбувається завершення процесу. Якщо запит позначено як «підозрілий», відбувається перехід до етапу динамічного аналізу. Запускається інструмент OWASP ZAP для додаткової перевірки. В результаті перевірки запит позначається як «небезпечний» або

«безпечний». Якщо підтвердилась небезпека запиту, повертається «SQL-ін’єкція виявлена», у разі якщо небезпека не підтверджується, повертається «Запит безпечний».

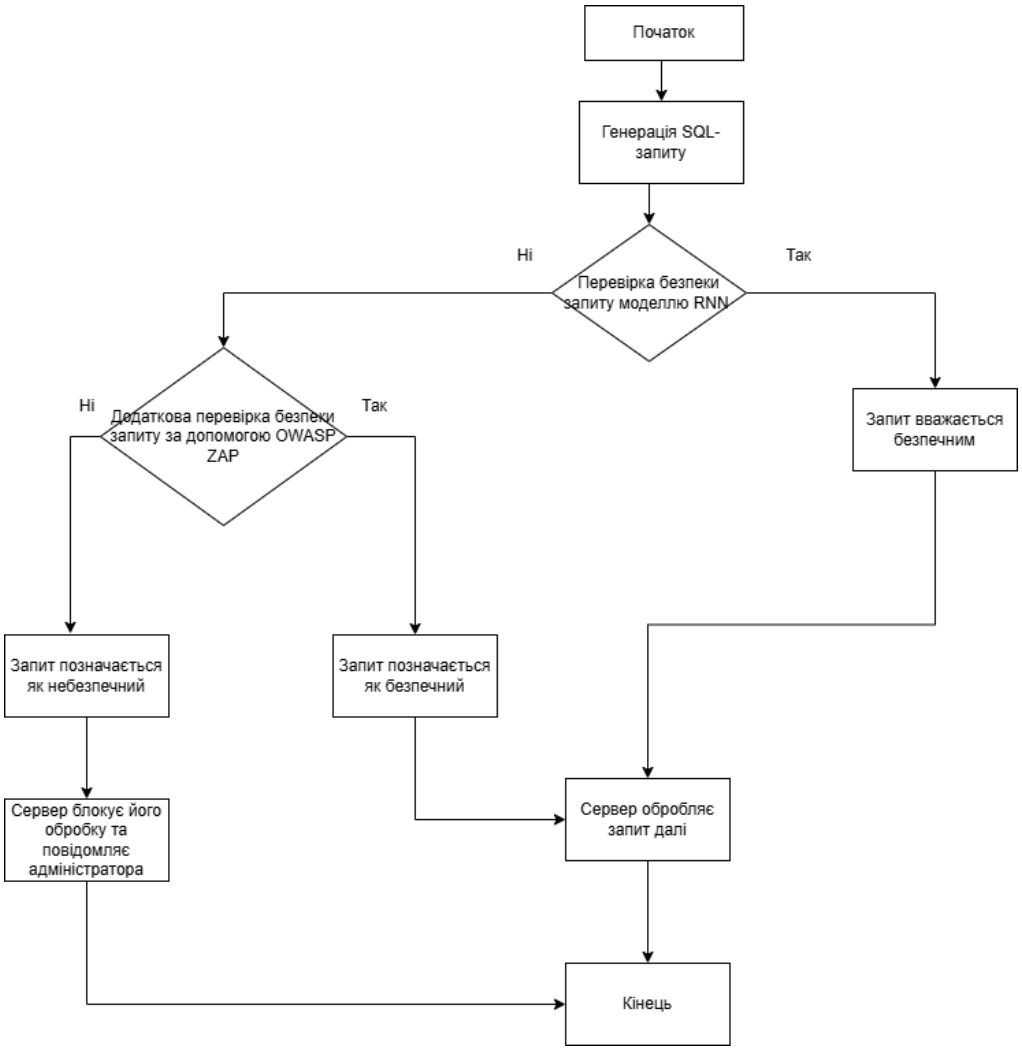



Рис. 3.4 Алгоритм роботи методу

## 3.2 Опис використаних програмних засобів

Для розробки було обрано мову програмування Python. Це універсальна мова, яку можна використовувати для широкого спектру додатків, від веб-розробки до машинного навчання. Вона має багато переваг, що роблять її популярною мовою:

завдяки своєму простому синтаксису, Python вважається легкою мовою для вивчення, його легко читати та розуміти [41];

одна з головних переваг Python, є його гнучкість завдяки великій бібліотеки модулів і пакетів. Python можна використовувати для веб-розробки, аналізу даних, машинного навчання та наукових обчислень [41];

Python можна інтегрувати в інші мови програмування та програми [41];

простота та легкість використання Python робить його ідеальною мовою для створення прототипів, його синтаксис стислий і зрозумілий, що дозволяє легко писати код і експериментувати з різними ідеями [41].

Незважаючи на численні переваги, Python має свої недоліки:

Python повільніший, ніж скомпільовані мови, такі як C++ або Java. Тому, що Python є інтерпретованою мовою, це означає, що кожен рядок коду виконується інтерпретатором по одному [41];

споживання пам’яті та збирання сміття є двома додатковими потенційними недоліками використання Python для реальних завдань. Його динамічний характер та інтерпретоване виконання можуть призвести до більшого використання пам’яті порівняно зі скомпільованими мовами [41];

Python дозволяє змінювати тип даних змінної під час виконання без необхідності явного оголошення типу, хоч це і робить код більш гнучким і легшим для написання, але також може призвести до помилок і неочікуваної поведінки [41].

Python є найпопулярнішою мовою програмування для машинного навчання, завдяки широким бібліотекам і фреймворкам. Завдяки таким бібліотекам, як NumPy, pandas і TensorFlow, Python дає змогу виконувати складні операції з масивними наборами даних. Масштабованість Python має першочергове значення для вирішення складних завдань ШІ, машинного навчання та глибокого навчання. Завдяки різноманітним бібліотекам та фреймворкам паралельної обробки Python може ефективно розподіляти обчислювальні завдання між кількома ядрами чи машинами, максимізуючи продуктивність. Проекти машинного навчання вимагають обробки та

візуалізації величезних обсягів складних даних. Бібліотеки Python, такі як Matplotlib, Plotly, Seaborn, ggplot і Altair, пропонують інтерактивні інструменти візуалізації даних, що дозволяють створювати зрозумілі діаграми та графіки [41].

TensorFlow — це бібліотека машинного навчання з відкритим кодом, розроблений Google. Вона використовується для створення та навчання моделей глибокого навчання, оскільки полегшує створення обчислювальних графіків і ефективне виконання на різних апаратних платформах [42].

Є три окремі частини, що визначають робочий процес бібліотеки: обробка даних, побудова моделі та навчання моделі робити прогнози. Фреймворк вводить дані як багатовимірний масив, що називається тензорами, і виконується двома способами. Перший спосіб є побудова обчислювального графіка, який визначає потік даних для навчання моделі. Другий спосіб полягає в використанні активного виконання, яке слідує імперативним принципам програмування та негайно оцінює операції [42].

TensorFlow використовують для різних завдань, що включають обробку природної мови, розпізнавання зображень, рукописного тексту та різні обчислювальні симуляції. Головні переваги бібліотеки полягають у її здатність виконувати низькорівневі операції на багатьох платформах прискорення, автоматичному обчисленні градієнтів, масштабованості на виробничому рівні та сумісному експорті графів [42].

Flask — це мікрофреймворк, призначений для швидкого й простого створення та масштабування веб-додатків. Він вважається найкращим фреймворком для обслуговування легких веб-додатків [43].

Переваги Flask наступні:

фреймворк прости для розуміння, має простий та інтуїтивно зрозумілий дизайн, який полегшує вивчення та використання [43];

Flask має високий рівень гнучкості та дозволяє розробникам обирати бібліотеки та інструменти, які відповідають їхнім потребам [43];

фреймворк забезпечує можливість швидкої розробки та підходить для створення невеликих проектів і прототипів [43].

Недоліки Flask:

має менше вбудованих функцій і може потребувати додаткових бібліотек розширень порівняно з іншими фреймворками [43];

відсутність стандартизації [43].

OWASP ZAP — інструмент тестування на проникнення, який допомагає виявляти та знаходити вразливості у веб-додатках. Він працює як проксі-сервер, що збирає передані дані та визначає, як програма реагує на ймовірно зловмисні запити. ZAP пропонує два типи сканування — активне та пасивне. Пасивне сканування перевіряє HTTP-запити та відповіді додатків на відомі індикатори вразливості безпеки та не може вносити зміни в запити. Активне сканування навпаки може змінювати та створювати запити, надсилаючи тестові запити, які виявляють вразливості, що не можна виявити пасивним скануванням. ZAP можна розгорнути як настільну програму або автоматично через API [44].

OWASP ZAP виконує кілька функцій безпеки:

пасивне сканування веб-запитів [44];

використовує списки словників для пошуку файлів і папок на веб-серверах [44];

використовує сканери для визначення структури сайту та отримання всіх посилань і URL-адрес [44];

перехоплює, відображає, зміни та пересилання веб-запитів між браузерами та веб-додатками [36].

### 3.3 Тестування методу

Для реалізації методу виявлення вразливості веб-додатків до SQL-ін’єкцій, було розроблено модель машинного навчання, яка виконує первину перевірку SQL-запиту.

Після перевірки вразі визначення запиту як «небезпечний», використовується OWASP ZAP для запуску активного сканування запиту або URL-адреси пов’язаної із ним.

Для тестування розробленого методу, було створено веб-додаток, який містить форму реєстрації.

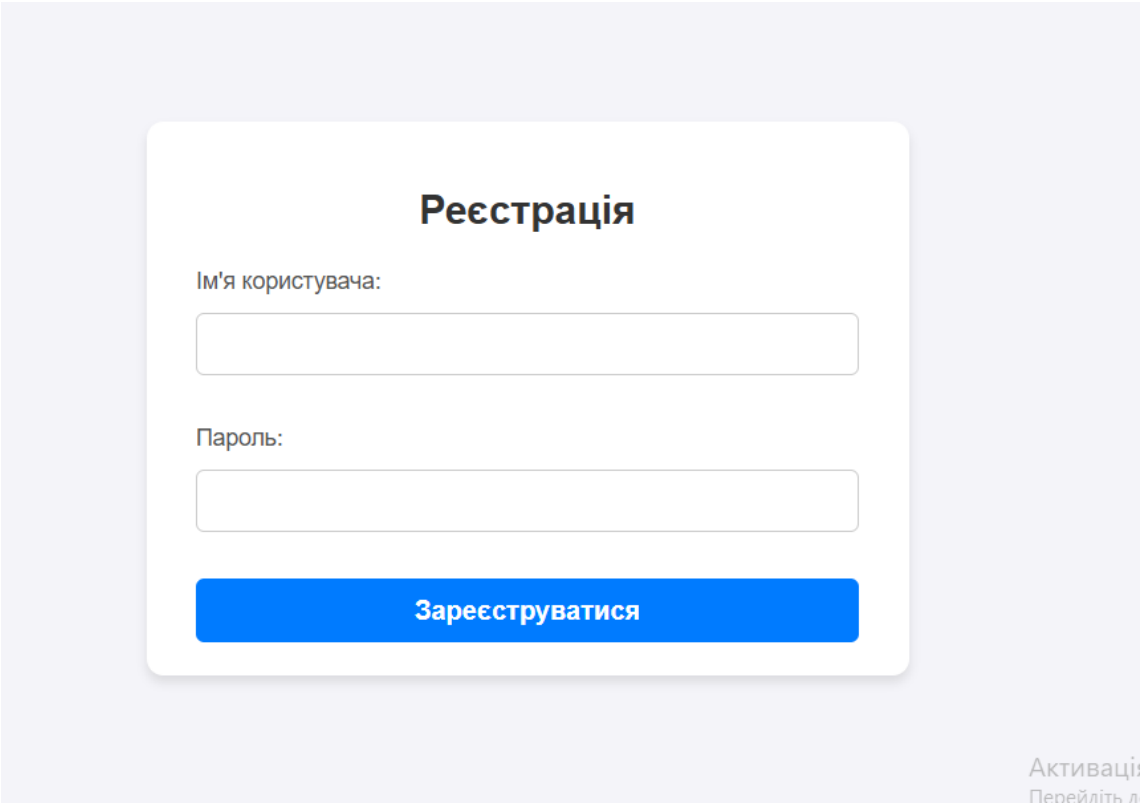



Рис. 3.5 Веб-додаток для тестування

Користувач вводить дані у форму реєстрації, генерується SQL-запит. Запит передається у функцію, яка токенізує текст та перетворює у числову послідовність, після чого передається моделі RNN. Вона оцінює ймовірність підозрілості запиту. Якщо запит підозрілий запускається функція сканування OWASP ZAP.

Тестування було проведено за двома сценаріями: введення нормальних даних користувача та введення SQL-ін’єкцій.

На рисунку 3.6 зображено введення нормальних даних користувача.

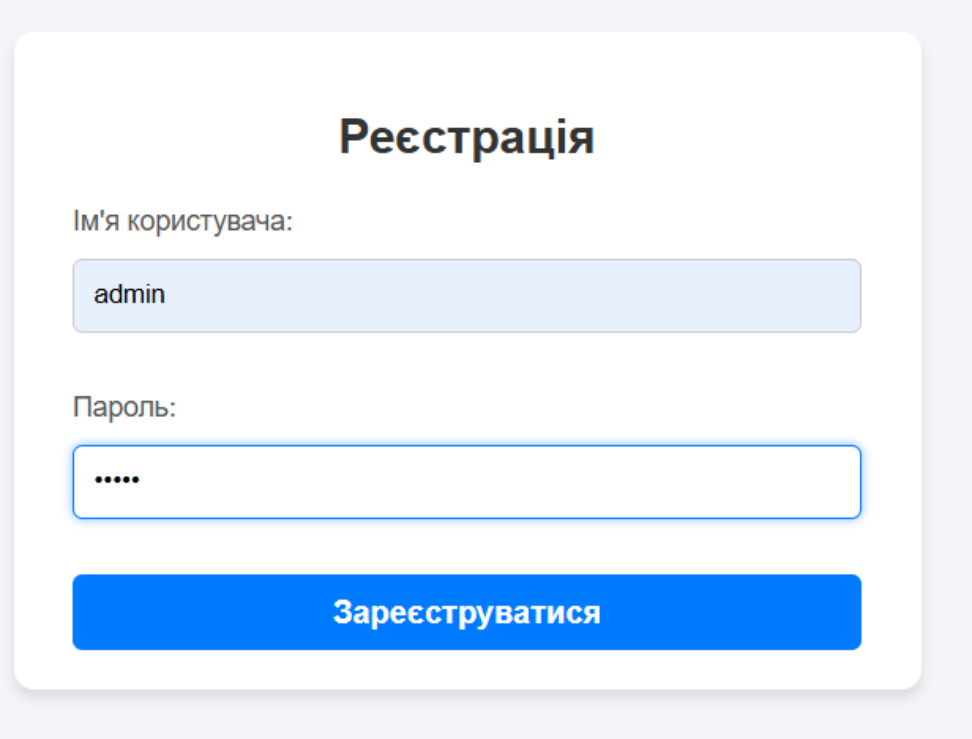



Рис. 3.6 Введення нормальних даних

При введені нормальних даних, додаток повідомляє про успішну реєстрацію, а в консолі виводить інформація про те що запит безпечний.


Рис. 3.7 Повідомлення про успішну реєстрацію

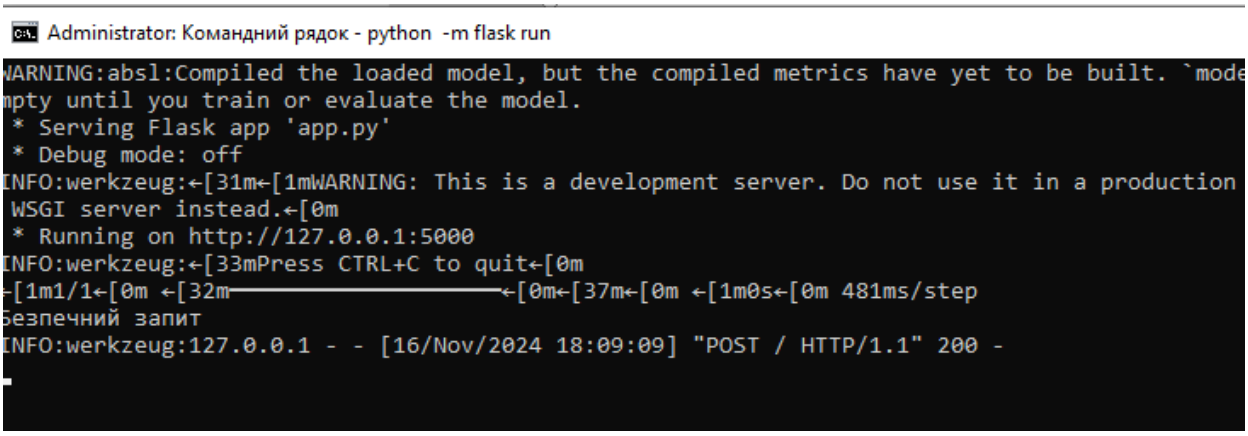



Рис. 3.8 Повідомлення в консолі про безпеку запиту

На рисунку 3.9 зображено спробу ввести SQL-ін’єкцію.

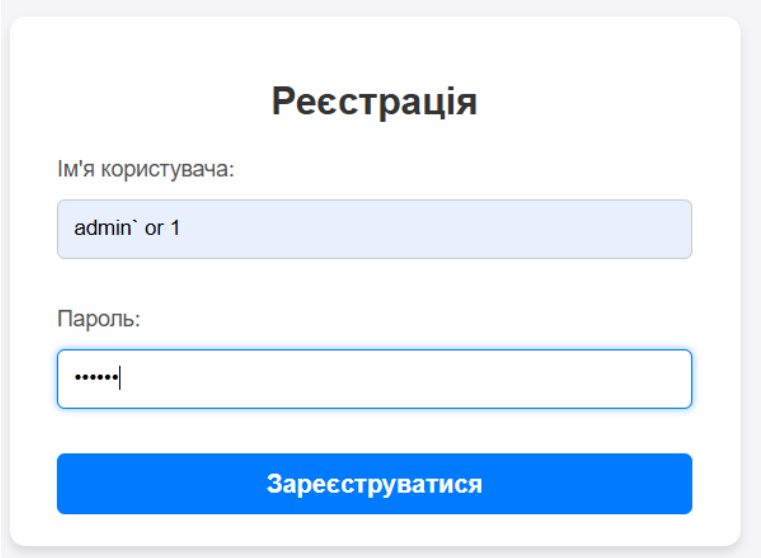



Рис. 3.9 Введення SQL-ін’єкції

При введені шкідливого запиту, на екранні з’являється повідомлення про те, що запит шкідливий. В консолі виводиться результати сканування.

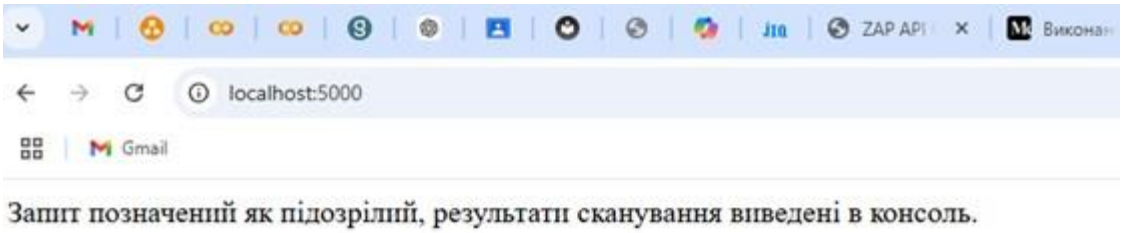



Рис. 3.10 Повідомлення про шкідливий запит

На рисунку 3.11 зображено результати аналізу в консолі.


Рис. 3.11 Результати аналізу запиту

**Висновки**

У третьому розділі було визначено, що для мінімізації наслідків атак SQL-ін’єкціями, необхідно створити надійний механізм, який здатний виявляти потенційно небезпечні запити. Для забезпечення швидкості та точності виявлення атак, потрібно поєднати методи машинного навчання і метод динамічного аналізу коду.

Для виявлення SQL-ін’єкцій, була обрана модель RNN, щоб інтегрувати інструмент динамічного аналізу, використовувався  OWASP ZAP API. Було обрано такі програмні засоби для реалізації методу: Python, бібліотека TensorFlow, Flask та OWASP ZAP API. Розроблений метод працює наступним чином: користувач вводить дані, формується SQL-запит, що перевіряється навченою моделлю на ймовірність підозрілості, якщо запит підозрілий запускається аналіз  через OWASP ZAP API. В результаті виводиться звіт в консолі. Для тестування методу було розроблено веб-додаток з формую реєстрації. Метод протестовано за двома сценаріями: введення безпечних даних та введення ін’єкцій.

# ВИСНОВКИ

1. Проаналізовано види SQL-ін’єкцій та визначено, що атаки спрямовані на отримання доступу адміністраторських привілеїв у базі даних є найнебезпечнішим видом атаки. Оскільки вони дозволяють здійснювати критичні зміни в системі. Основні методи виявлення ін’єкцій, це статичний аналіз коду, динамічний аналіз та виявлення на основі сигнатур. Комбінування декількох методів виявлення дозволить ефективно реагувати на різні типи атак.
2. Було розглянуто методи машинного навчання. Визначено переваги та недоліки контрольованого та не контрольованого навчання. Для вирішення поставлених завдань було обрано контрольоване машинне навчання. Оскільки, для чіткої класифікації запитів на безпечні та небезпечні, потрібно дані з мітками.
3. Порівняно алгоритми машинного навчання для виявлення SQL-ін’єкцій. Для порівняння було обрано такі алгоритми: KNN, Gaussian Naive Bayes, Decision Tree, Logistic Regression, Linear SVM та RNN. Для оцінки ефективності кожного алгоритму використовувались такі ключові метрики, як точність, точність позитивного класу, повнота, F1 міра, показник хибнопозитивної частоти (FPR) та частота позитивних результатів (TPR). Результати, показали, що серед проаналізованих алгоритмів найбільш ефективним для виявлення SQL-ін’єкцій є алгоритм RNN. Він показав найвищі результати точності, найнижчий показник FPR — 0,001% та найвищий показник виявлення атак TPR — 0,99%.
4. Розроблено метод виявлення вразливостей веб-додатків до SQL-ін’єкцій з використанням методів машинного навчання. Метод поєднує машинне навчання на основі моделі RNN із динамічними інструментами аналізу, таким як OWASP ZAP API. Завдяки комбінуванню цих методів, вдалося створити ефективну гібридну систему. Автоматизація процесу дозволить зменшити потребу в ручній перевірці та покращити захист веб-додатків.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Wohlwend B. SQL injection – A serious web security threat. *Medium*. URL: https://medium.com/@brandon93.w/sql-injection-a-serious-web-security-threat-5cf7e1e14816 (date of access: 18.11.2024).

2. Identifying data leaksvia sql injection / K. Chintanica et al. *Jetir*. 2024. Vol. 11, no. 5.

3. Types of SQL injection?. *Acunetix*. URL: https://www.acunetix.com/websitesecurity/sql-injection2/ (date of access: 18.11.2024).

4. In-Band SQL injection. *Invicti*. URL: https://www.invicti.com/learn/in-band-sql-injection/ (date of access: 18.11.2024).

5. Blind SQL injection. *Invicti*. URL: https://www.invicti.com/learn/blind-sql-injection/ (date of access: 18.11.2024).

6. Out-of-Band SQL injection. *Invicti*. URL: https://www.invicti.com/learn/out-of-band-sql-injection-oob-sqli/ (date of access: 18.11.2024).

7. Singh J. P. Analysis of SQL injection detection techniques. *Theoretical and applied informatics*. 2017. Vol. 28, no. 1&2. P. 37–55. URL: https://doi.org/10.20904/281-2037 (date of access: 18.11.2024).

8. Paige. SQL injections -manual / auto. *Cybercortex*. URL: https://randomhathacking.hashnode.dev/sql-injections-manual-auto (date of access: 18.11.2024).

9. Що таке sql-ін'єкція - терміни та визначення в кібербезпеці. *VPN Unlimited - Fast &Secure VPN service*. URL: https://www.vpnunlimited.com/ua/help/cybersecurity/sql-injection (дата звернення: 18.11.2024).

10. Static code analysis | OWASP foundation. *OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation*. URL: https://owasp.org/www-community/controls/Static_Code_Analysis (date of access: 18.11.2024).

11. What is dynamic code analysis?. *checkpoint*. URL: https://www.checkpoint.com/cyber-hub/cloud-security/what-is-dynamic-code-analysis/ (date of access: 18.11.2024).

12. What is signature-based detection? | corelight. *Corelight: Evidence-Based NDR and Threat Hunting Platform*. URL: https://corelight.com/resources/glossary/signature-based-detection (date of access: 18.11.2024).

13. Gowthami S., Prasanna Kumar K. Detecting SQL injection attacks in web application using REGEX and query result size. *International journal of innovative research in computer science and engineering*. 2016. Vol. 2, no. 5.

14. Paige. SQL injections -manual / auto. *Cybercortex*. URL: https://randomhathacking.hashnode.dev/sql-injections-manual-auto (date of access: 18.11.2024).

15. Best SQL injection (sqli) detection tools for 2024: strengthen your web application security. *MonoVM.com*. URL: https://monovm.com/blog/best-sqli-detection-tools/ (date of access: 18.11.2024).

16. Levy E. Preventing SQL injection: is WAF enough?. *Security Engineering Notebook*. URL: https://www.securityengineering.dev/waf-sql-injection/ (date of access: 18.11.2024).

17. Machine learning (ML): all there is to know. *ISO*. URL: https://www.iso.org/artificial-intelligence/machine-learning (date of access: 18.11.2024).

18. Харченко В. О. Основи машинного навчання : навч. посіб. Суми : Сум. держ. університет, 2023. 264 с.

19. IBM. What is supervised learning? | IBM. *IBM - United States*. URL: https://www.ibm.com/topics/supervised-learning (date of access: 18.11.2024).

20. GeeksforGeeks. Regression in machine learning - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/regression-in-machine-learning/ (date of access: 18.11.2024).

21. GeeksforGeeks. Getting started with Classification - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/getting-started-with-classification/ (date of access: 18.11.2024).

22. GeeksforGeeks. Unsupervised Learning - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/ml-types-learning-part-2/ (date of access: 18.11.2024).

23. Clustering in machine learning - geeksforgeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/clustering-in-machine-learning/ (date of access: 18.11.2024).

24. DeepAI. Association learning. *DeepAI*. URL: https://deepai.org/machine-learning-glossary-and-terms/association-learning (date of access: 18.11.2024).

25. GeeksforGeeks. Introduction to dimensionality reduction - geeksforgeeks. *GeeksforGeeks*.

26. GeeksforGeeks. Unsupervised Learning - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/ml-types-learning-part-2/ (date of access: 18.11.2024).

27. 10 machine learning algorithms to know in 2024. *Coursera*. URL: https://www.coursera.org/articles/machine-learning-algorithms (date of access: 18.11.2024).

28. GeeksforGeeks. Linear Regression in Machine learning - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/ml-linear-regression/ (date of access: 18.11.2024).

29. Kanade V. Everything you need to know about logistic regression - spiceworks. *Spiceworks Inc*. URL: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/ (date of access: 18.11.2024).

30. Naive bayes classifiers - geeksforgeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/naive-bayes-classifiers/ (date of access: 18.11.2024).

31. IBM. What is a decision tree? | IBM. *IBM - United States*. URL: https://www.ibm.com/topics/decision-trees (date of access: 18.11.2024).

32. K-Nearest neighbor(knn) algorithm - geeksforgeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/k-nearest-neighbours/ (date of access: 18.11.2024).

33. IBM. What is the k-nearest neighbors algorithm? | IBM. *IBM - United States*. URL: https://www.ibm.com/topics/knn (date of access: 18.11.2024).

34. IBM. What is support vector machine? | IBM. *IBM - United States*. URL: https://www.ibm.com/topics/support-vector-machine (date of access: 18.11.2024).

35. Fagbuyiro D. Understanding the support vector machine (SVM) model. *Medium*. URL: https://medium.com/@davidfagb/understanding-the-support-vector-machine-svm-model-c8eb9bd54a97 (date of access: 18.11.2024).

36. What is RNN? - recurrent neural networks explained - AWS. *Amazon Web Services, Inc*. URL: https://aws.amazon.com/what-is/recurrent-neural-network/ (date of access: 18.11.2024).

37. Sachinsoni. Recurrent neural networks (RNN) from basic to advanced. *Medium*. URL: https://medium.com/@sachinsoni600517/recurrent-neural-networks-rnn-from-basic-to-advanced-1da22aafa009 (date of access: 18.11.2024).

38. GeeksforGeeks. How to store a TfidfVectorizer for future use in scikit-learn? - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/how-to-store-a-tfidfvectorizer-for-future-use-in-scikit-learn/ (date of access: 18.11.2024).

39. GeeksforGeeks. Understanding the confusion matrix in machine learning - geeksforgeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/ (date of access: 18.11.2024).

40. What is false positive rate | deepchecks. *Deepchecks*. URL: https://www.deepchecks.com/glossary/false-positive-rate/ (date of access: 18.11.2024).

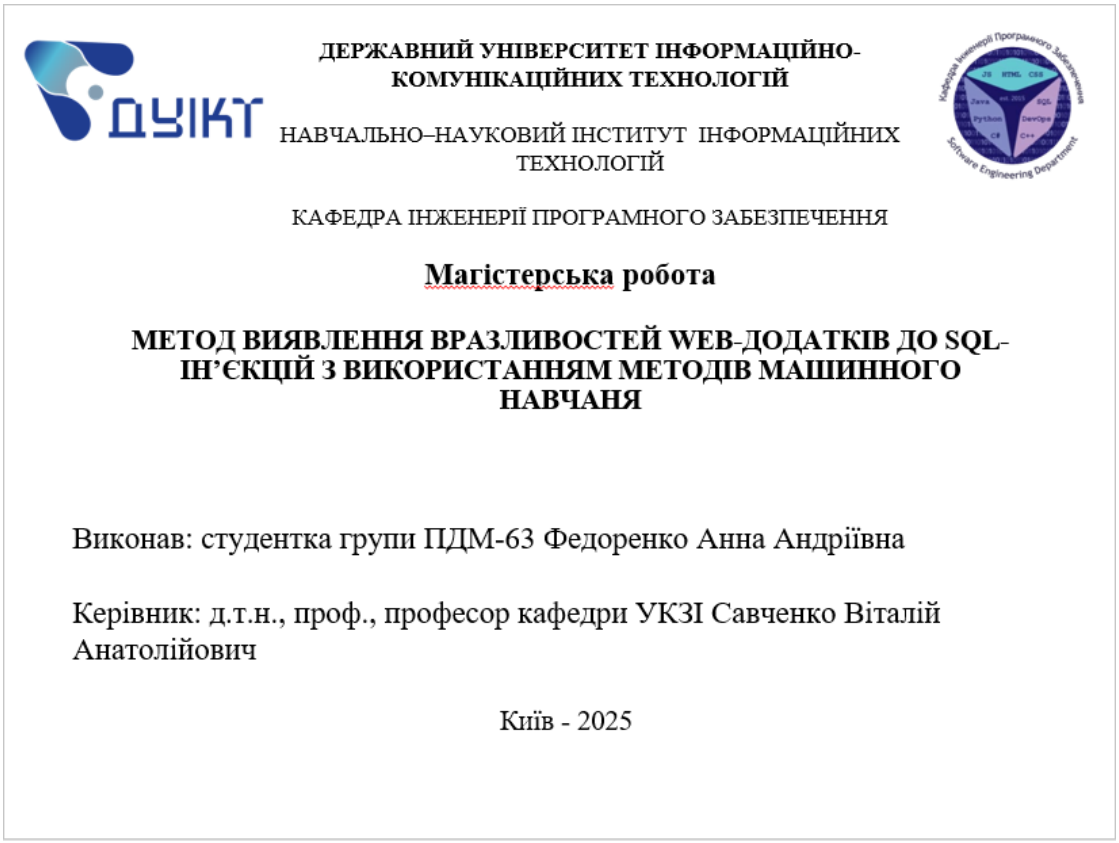
41. Karl T. 6 reasons why is python used for machine learning. *New Horizons*. URL: https://www.newhorizons.com/resources/blog/why-is-python-used-for-machine-learning (date of access: 19.11.2024).

42. What is signature-based detection? | corelight. *Corelight: Evidence-Based NDR and Threat Hunting Platform*. URL: https://corelight.com/resources/glossary/signature-based-detection (date of access: 18.11.2024).

43. L G. D. Comparison of flask, django, and fastapi: advantages, disadvantages, and use cases. *Medium*. URL: https://medium.com/@tubelwj/comparison-of-flask-django-and-fastapi-advantages-disadvantages-and-use-cases-63e7c692382a (date of access: 19.11.2024).

44. OWASP ZAP: 6 key capabilities and a quick tutorial | hackerone. *HackerOne | #1 Trusted Security Platform and Hacker Program*. URL: https://www.hackerone.com/knowledge-center/owasp-zap-6-key-capabilities-and-quick-tutorial (date of access: 18.11.2024).

# ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**Магістерська робота**

**МЕТОД ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ WEB-ДОДАТКІВ ДО SQL-ІН'ЄКЦІЙ З ВИКОРИСТАННЯМ МЕТОДІВ МАШИННОГО НАВЧАНЯ**

Виконав: студентка групи ПДМ-63 Федоренко Анна Андріївна

Керівник: д.т.н., проф., професор кафедри УКЗІ Савченко Віталій Анатолійович

Київ - 2025

## МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

**Мета роботи**: удосконалення процесу виявлення SQL-ін'єкцій з використанням методів машинного навчання.
**Об'єкт дослідження:** методи та засоби захисту веб-додатків від SQL-ін'єкцій.
**Предмет дослідження**: методи машинного навчання для виявлення SQL-ін'єкцій у web-додатках.

## МЕТОДИ ВИЯВЛЕННЯ SQL-ІН'ЄКЦІЙ

| Критерії | Статичний аналіз | Динамічний аналіз | Методи машинного навчання |
|---|---|---|---|
| Точність | Висока (залежить від якості статистичних моделей та правил) | Висока (оскільки аналізує реальну поведінку коду) | Дуже висока (здатна адаптуватись до нових атак) |
| Ймовірність хибнопозитивних результатів | Висока (правила можуть бути занадто загальними). | Низька (перевіряється реальна поведінка запиту). | Середня (залежить від якості навчання моделі). |
| Ймовірність хибнонегативних результатів | Висока (може пропустити нестандартні атаки). | Середня (залежить від якості ізольованого середовища). | Низька |
| Адаптивність | Середня (моделі потребують оновлення для нових вид атак) | Низька (моделі потребують оновлення для нових вид атак) | Висока (легко адаптується до нових даних та змін) |

3

## ПОРІВНЯННЯ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ SQL-ІН'ЄКЦІЙ

| Назва алгоритму | KNN | Gaussian Naive Bayes | Decision Tree | Logistic Regression | Linear SVM | RNN |
|---|---|---|---|---|---|---|
| Accuracy | 0,939 | 0,785 | 0,993 | 0,977 | 0,967 | 0,996 |
| Precision | 0,942 | 0,856 | 0,993 | 0,977 | 0,968 | 0,997 |
| Recall | 0,939 | 0,785 | 0,993 | 0,977 | 0,967 | 0,993 |
| F1 | 0,937 | 0,787 | 0,993 | 0,977 | 0,967 | 0,995 |
| TP | 1938 | 2261 | 2264 | 2167 | 2120 | 2278 |
| FN | 356 | 33 | 30 | 127 | 174 | 16 |
| FP | 19 | 1279 | 10 | 12 | 24 | 6 |
| TN | 3809 | 2549 | 3818 | 3816 | 3804 | 3822 |
| FPR | 0,004% | 0,33% | 0,002% | 0,003% | 0,006% | 0,001% |
| TPR | 0,84% | 0,98% | 0,98% | 0,94% | 0,92% | 0,99% |

4

# МАТЕМАТИЧНА МОДЕЛЬ

**Математична модель RNN**

У кожному кроці часу *t,* стан $h_t$ визначається за рекурентним правилом (1):

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b_h), \quad (1)$$

де, $x_t$ - вектор вхідних даних у момент часу $t$, $h_{t-1}$ - попередній стан, $W_h$ і $W_x$ - вагові матриці, $b_h$ - зсув, $\sigma$ – нелінійна активаційна функція.

**Класифікаційний результат визначається як (2):**

$$y = softmax(W_y h_T + b_y), \quad (2)$$

де, $h_T$ - останній прихований стан, $W_y$ і $b_y$ - ваги і зсув вихідного шару.

**Токенізація тексту**

SQL-запит спочатку перетворюється на послідовність токенів (3):

$$T = [t_1, t_2, \dots, t_n], \quad (3)$$

де, $t_i$ - окреме слово або символ.

**Використання вбудовувань (Embeddings)**

Для того щоб відобразити дискретні токени у вигляді безперервного векторного простору, використовується вбудовування (4):

$$Emb(x_i) = v_i \in R^d, \quad (4)$$

де, $v_i$ — це векторна розмірність *d,* що представляє токен $x_i$.

5

# МАТЕМАТИЧНА МОДЕЛЬ

Функція втрат під час тренування моделі визначається як (5):

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i), \quad (5)$$

де, $y_i$ — істинна мітка (0 – безпечний запит, 1 – SQL-ін'єкція), $\hat{y}_i$ - прогнозована ймовірність моделі.

**Класифікація запитів**

Модель навчається розрізняти два класи, безпечні запити *(y = 0)* та запити з ін'єкціями *(y = 1).*

Ймовірність визначається за допомогою активаційної функції sigmoid (6):

$$P(y = 1|X) = \frac{1}{1+e^{-z}}, \quad (6)$$

де, $z$ – вагова сума вихідного шару.

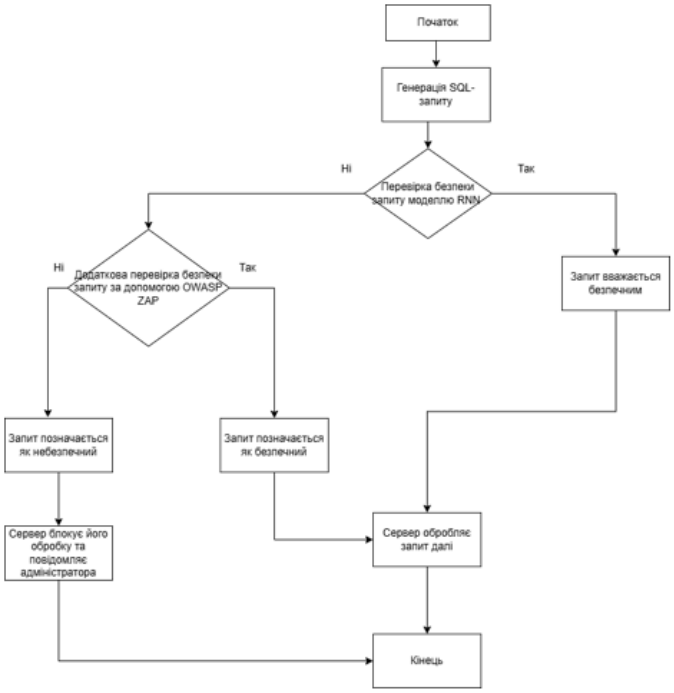
**Поріг для визначення підозрілих запитів**

На основі виходу моделі визначаємо, чи є запит підозрілим (7):

$$is\_suspicious = \begin{cases} 1, \text{якщо } P(y = 1|X) > 0{,}7 \\ 0, \text{інакше} \end{cases} \quad (7)$$

6

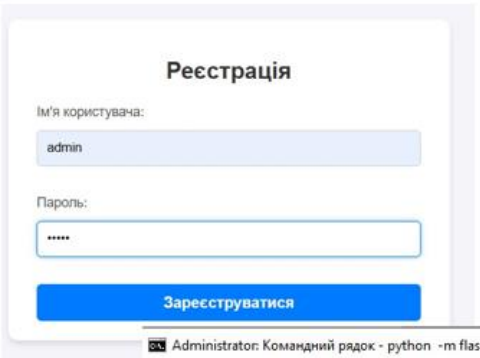
# АЛГОРИТМ РОБОТИ МЕТОДУ

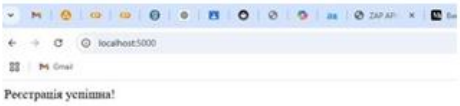


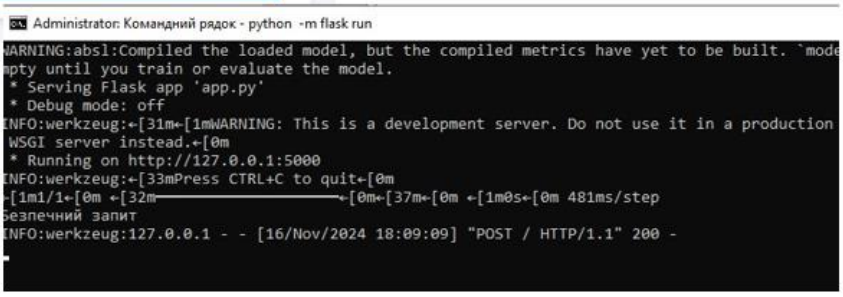

7

# ПРАКТИЧНИЙ РЕЗУЛЬТАТ

Безпечний запит



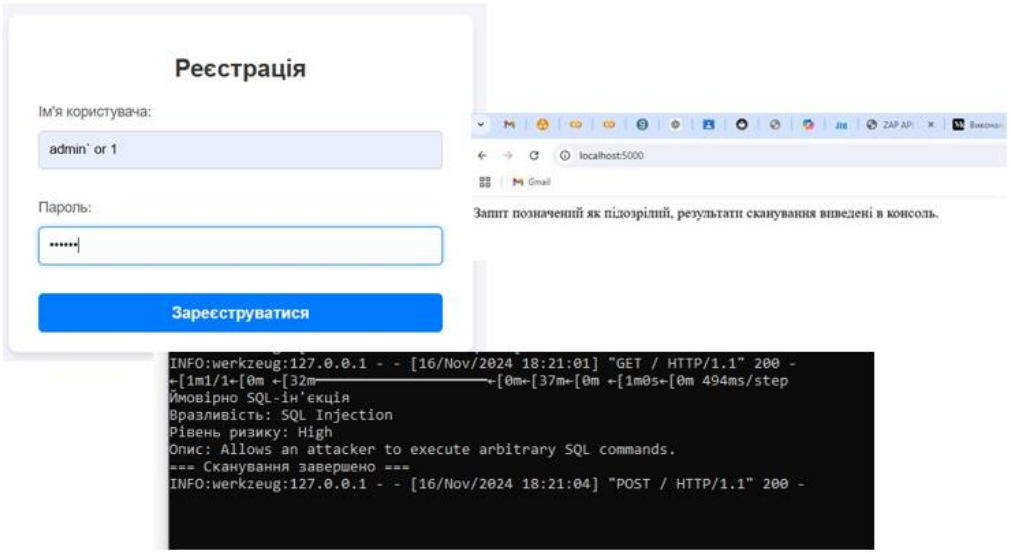




8

## ПРАКТИЧНИЙ РЕЗУЛЬТАТ

SQL-ін'єкція




9

## ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ ВИЯВЛЕННЯ

| Показник | Інструменти динамічного аналізу | Інструменти статичного аналізу | Розроблений метод |
|---|---|---|---|
| Precision | 0,65 | 0,85 | 0,87 |
| Recall | 0,32 | 0,66 | 0,67 |
| F1 | 0,43 | 0,75 | 0,78 |

10

## ВИСНОВКИ

1. Проаналізовано види SQL-ін'єкцій та визначено, що атаки спрямовані на отримання доступу адміністраторських привілеїв у базі даних є найнебезпечнішим видом атаки. Оскільки вони дозволяють здійснювати критичні зміни в системі.
2. Порівняно алгоритми машинного навчання для виявлення SQL-ін'єкцій. Для порівняння було обрано такі алгоритми: KNN, Gaussian Naive Bayes, Decision Tree, Logistic Regression, Linear SVM та RNN. Результати, показали, що серед проаналізованих алгоритмів найбільш ефективним для виявлення SQL-ін'єкцій є алгоритм RNN. Він показав найвищі результати точності, найнижчий показник FPR — 0,001% та найвищий показник виявлення атак TPR — 0,99%.
3. Розроблено метод виявлення вразливостей веб-додатків до SQL-ін'єкцій з використанням методів машинного навчання. Метод поєднує машинне навчання на основі моделі RNN із динамічними інструментами аналізу, таким як OWASP ZAP API. Завдяки комбінуванню цих методів, вдалося створити більш точну гібридну систему.



## ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

**Статті:**

1. Федоренко А. А., Осадчий Б. І., Коржик В. В. Аналіз методів виявлення вразливостей Web-ресурсів до SQL-ін'єкцій// Сучасний захист інформації. №3 (55), 2023. С. 57-61.

**Тези доповідей:**

1. Федоренко А.А., Жебка В.В. Аналіз засобів виявлення вразливостей веб-додатків до SQL-ін'єкцій. // Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ»– Київ: ДУІКТ, 2024. – С.243-245.

# ДОДАТОК Б. ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ

rnn_model.py

```
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(X)
X_seq = tokenizer.texts_to_sequences(X)
X_pad = pad_sequences(X_seq, maxlen=100)

label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
X_train_pad, X_test_pad, y_train, y_test = train_test_split(X_pad, y, test_size=0.2, random_state=42)

rnn_model = Sequential()
rnn_model.add(Embedding(input_dim=5000, output_dim=128, input_length=100))
rnn_model.add(LSTM(64, return_sequences=True))
rnn_model.add(LSTM(32))
rnn_model.add(Dense(10, activation='relu'))
rnn_model.add(Dense(1, activation='sigmoid'))

rnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

early_stopping = EarlyStopping(monitor='val_loss',patience=3)

history = rnn_model.fit(X_train_pad, y_train, epochs=10, batch_size=32,
                        validation_data=(X_test_pad, y_test), callbacks=[early_stopping])

rnn_loss, rnn_acc = rnn_model.evaluate(X_test_pad, y_test)
print("RNN Accuracy: ", rnn_acc)

rnn_pred = (rnn_model.predict(X_test_pad) > 0.5).astype("int32")

accuracy = accuracy_score(y_test, rnn_pred)
precision = precision_score(y_test, rnn_pred, average='weighted')
recall = recall_score(y_test, rnn_pred, average='weighted')
f1 = f1_score(y_test, rnn_pred, average='weighted')

print(f"RNN Results:")
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
conf_matrix = confusion_matrix(y_test, rnn_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title(f"Confusion Matrix for RNN")
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_curve, auc
import matplotlib.pyplot as plt

thresholds = [0.1, 0.5, 0.7]

results = []

for threshold in thresholds:

    pred_labels = (rnn_model.predict(X_test_pad) > threshold).astype("int32")
```

```
    accuracy = accuracy_score(y_test, pred_labels)
    precision = precision_score(y_test, pred_labels, average='weighted', zero_division=0)
    recall = recall_score(y_test, pred_labels, average='weighted', zero_division=0)
    f1 = f1_score(y_test, pred_labels, average='weighted', zero_division=0)

    results.append([threshold, accuracy, precision, recall, f1])

results_df = pd.DataFrame(results, columns=['Threshold', 'Accuracy', 'Precision', 'Recall', 'F1
Score'])

print("Results for different thresholds:")
print(results_df)

plt.figure(figsize=(10, 6))
plt.plot(results_df['Threshold'], results_df['Accuracy'], label='Accuracy', marker='o')
plt.plot(results_df['Threshold'], results_df['Precision'], label='Precision', marker='o')
plt.plot(results_df['Threshold'], results_df['Recall'], label='Recall', marker='o')
plt.plot(results_df['Threshold'], results_df['F1 Score'], label='F1 Score', marker='o')

plt.title('Metrics at Different Thresholds')
plt.xlabel('Threshold')
plt.ylabel('Metric Value')
plt.legend()
plt.grid()
plt.show()

fpr, tpr, _ = roc_curve(y_test, rnn_model.predict(X_test_pad))
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.title('ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.grid()
plt.show()

best_threshold = results_df.loc[results_df['F1 Score'].idxmax()]
print(f"Найкращий поріг: {best_threshold['Threshold']:.2f} з F1 Score: {best_threshold['F1
Score']:.4f}")
```

app.py

```
from flask import Flask, request, render_template
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from zapv2 import ZAPv2
import numpy as np
import pickle
import time

app = Flask(__name__)

model = load_model('model.h5')

with open('tokenizer.pkl', 'rb') as handle:
    tokenizer = pickle.load(handle)

zap = ZAPv2(apikey='', proxies={'http': 'http://localhost:8080'})

def preprocess_input(data):
```

```
    # Перетворення тексту у числову послідовність
    sequences = tokenizer.texts_to_sequences([data])
    # Паддінг послідовності до потрібної довжини
    padded_sequence = pad_sequences(sequences, maxlen=100, padding='post')
    return np.array(padded_sequence)

# Функція перевірки на SQL-ін'єкції за допомогою RNN
def is_suspicious(query):
    processed_query = preprocess_input(query)
    prediction = model.predict(processed_query)
    return prediction[0] > 0.7  # Поріг ймовірності, що визначає "підозрілий" запит

# Функція для сканування підозрілих запитів через OWASP ZAP
def scan_with_zap(target_url):
    print(f'Scanning target: {target_url}')
    zap.urlopen(target_url)
    time.sleep(2)

    scan_id = zap.ascan.scan(target_url)
    print(f'Active Scan started (ID: {scan_id})')

    # Очікування завершення сканування
    while int(zap.ascan.status(scan_id)) < 100:
        print(f'Scan progress: {zap.ascan.status(scan_id)}%')
        time.sleep(5)

    print('Scan completed!')
    vulnerabilities = zap.core.alerts(baseurl=target_url)
    for alert in vulnerabilities:
        print(f"Вразливість: {alert['name']}")
        print(f"Рівень ризику: {alert['risk']}")
        print(f"Опис: {alert['desc']}")
        print(f"URL: {alert['url']}\n")

# Головний маршрут для реєстрації
@app.route("/", methods=["GET", "POST"])
def register():
    if request.method == "POST":
        username = request.form.get("username")
        password = request.form.get("password")

        query = f"INSERT INTO users (username, password) VALUES ('{username}', '{password}');"

        # Перевірка на загрозу через RNN
        if is_suspicious(query):
            print("Підозрілий запит виявлено, запускаємо OWASP ZAP для перевірки")
            scan_with_zap("http://localhost:5000")

            return "Запит позначений як підозрілий та перевірений OWASP ZAP."
        else:
            print("Запит безпечний, зберігаємо дані (умовно)")
            return "Реєстрація успішна!"

    return render_template("register.html")

if __name__ == "__main__":
    app.run(debug=True)
```