

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

**«Розробка вебсервісу аналітики транзакцій
цифрових активів на наявність операцій з
підвищеним фінансовим ризиком»**

Виконав студент групи ІПЗм-24-1
КОЧУРА Віталій Вікторович

Керівник роботи:
ЛОКТИКОВА Тамара Миколаївна

Рецензент:
ВАКАЛЮК Тетяна Анатоліївна

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«18» вересня 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КОЧУРА Віталій Вікторович**

Керівник роботи: **ЛОКТИКОВА Тамара Миколаївна**

Тема роботи: **«Розробка вебсервісу аналітики транзакцій цифрових активів на наявність операцій з підвищеним фінансовим ризиком»**,

затверджена Наказом закладу вищої освіти від «18» вересня 2025 р. № 491/с

Вихідні дані для роботи: **об'єктом дослідження є процеси аналітики блокчейн-даних цифрових активів, пов'язані з формуванням та оцінюванням кластерів адрес на основі транзакційних зв'язків, а предметом дослідження є методи та програмні засоби аналізу криптовалютних транзакцій, що використовуються для побудови вебсервісу аналітики операцій з цифровими активами та оцінювання ризиковості кластерів адрес.**

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Локтікова Т.М.	22.09.2025р.	22.09.2025р.
2	Локтікова Т.М.	22.09.2025р.	22.09.2025р.
3	Локтікова Т.М.	22.09.2025р.	22.09.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота магістра присвячена розробці вебсервісу аналітики транзакцій цифрових активів на наявність операцій з підвищеним фінансовим ризиком. Пояснювальна записка до випускної роботи складається з 65 сторінок друкованого тексту, 4 таблиць, 23 рисунків. Список використаних джерел нараховує 40 найменувань і займає 4 сторінки. В роботі є 2 додатки. Загальний обсяг роботи – 73 сторінки.

Метою роботи є проектування та створення вебсервісу для аналітики транзакцій цифрових активів, який забезпечує виявлення операцій із підвищеним фінансовим ризиком за допомогою кластеризації адрес та подальшої класифікації методами машинного навчання.

У роботі було визначено основні вимоги до розроблюваного вебсервісу, проведено аналіз існуючих рішень, обрані відповідні засоби, спроектовано основні модулі програмного продукту та реалізовано його. Особливу увагу приділено застосуванню методів штучного інтелекту для класифікації сформованих кластерів адрес, а також алгоритмам їх побудови на основі евристичних правил. Для створення застосовувалися наступні інструменти: мова програмування PHP, її фреймворк Symfony, API Platform, бази даних MySQL та PostgreSQL, мова програмування Python, Docker, вебсервер Nginx та JavaScript-бібліотека React.

КЛЮЧОВІ СЛОВА: ЦИФРОВІ АКТИВИ, БЛОКЧЕЙН, ТРАНЗАКЦІЯ, ДАТАСЕТ, МАШИННЕ НАВЧАННЯ, ЕВРИСТИКИ, КЛАСТЕРИЗАЦІЯ.

ABSTRACT

The master's thesis is devoted to the development of a web service for analyzing digital asset transactions in order to identify operations with increased financial risk. The explanatory note of the thesis consists of 65 pages of printed text, 4 tables, and 23 figures. The list of references includes 40 sources and spans 4 pages. The thesis has 2 appendices. The total volume of the thesis is 73 pages.

The purpose of the thesis is to design and develop a web service for digital asset transaction analytics that enables the detection of operations with increased financial risk through address clustering and subsequent classification using machine learning methods.

The thesis defines the main requirements for the developed web service, analyzes existing solutions, selects appropriate tools, designs the core modules of the software product, and implements them. Special attention is paid to the application of artificial intelligence methods for the classification of formed address clusters, as well as to algorithms for their construction based on heuristic rules. The following tools were used in the development process: the PHP programming language and its Symfony framework, API Platform, MySQL and PostgreSQL databases, the Python programming language, Docker, the Nginx web server, and the JavaScript library React.

KEYWORDS: DIGITAL ASSETS, BLOCKCHAIN, TRANSACTION, DATASET, MACHINE LEARNING, HEURISTICS, CLUSTERING.

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ	9
1.1 Теоретичні засади та вихідні положення	9
1.2 Постановка задачі	11
1.3 Аналіз існуючих програмних продуктів	12
1.4 Аналіз методів аналітики транзакцій цифрових активів.....	16
1.5 Обґрунтування вибору засобів реалізації програмного продукту	20
Висновки до першого розділу	22
РОЗДІЛ 2 РОЗРОБКА АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ ТА МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ	24
2.1 Аналіз використовуваного набору даних.....	24
2.2 Реалізація алгоритмів машинного навчання.....	28
2.3 Аналіз результатів роботи та ефективності моделей класифікації	36
2.4 Алгоритми кластеризації та прогнозування категорії кластерів	40
Висновки до другого розділу.....	47
РОЗДІЛ 3 ОПИС ПРОГРАМНОГО ПРОДУКТУ ТА ПРИНЦИПІВ ЙОГО ФУНКЦІОНУВАННЯ.....	48
3.1 Архітектура системи та взаємодія її компонентів	48
3.2 Структура інтерфейсу та порядок роботи з вебсервісом	49
Висновки до третього розділу	57
ВИСНОВКИ.....	59
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	61
ДОДАТКИ.....	65

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API (Application Programming Interface) – набір інструкцій, які дозволяють різним програмам спілкуватись між собою та обмінюватись даними.

HTTP (Hypertext Transfer Protocol) – протокол, в якому описано правила передачі даних. Він допомагає браузеру завантажувати вебсторінки, а серверу отримати інформацію, яку користувач ввів на сайті.

JSON (JavaScript Object Notation) – текстовий формат, призначений для зберігання структурованих даних.

ML (Machine Learning) – підгалузь штучного інтелекту, що базується на побудові моделей, здатних робити прогнози або приймати рішення на основі даних.

ORM (Object-Relational Mapping) – технологія, що дозволяє працювати з базою даних як з об'єктами програмного коду, не пишучи SQL-запити вручну.

REST (Representational State Transfer) – стиль побудови вебсервісів, що передбачає використання HTTP-запитів для доступу до ресурсів та роботи з ними.

UTXO (Unspent Transaction Output) – метод, який використовує протокол Bitcoin для відстеження балансів під час їхнього переміщення між криптогаманцями.

ШІ – штучний інтелект.

ВСТУП

Актуальність теми. У сучасних умовах стрімкого розвитку ринку цифрових активів та збільшення обсягу транзакцій з криптовалютами зростає потреба у надійних інструментах оцінювання ризиків і виявлення потенційно небезпечної активності. Псевдонімність, децентралізований характер блокчейн-мереж та відсутність централізованого контролю створюють сприятливе середовище для здійснення шахрайських операцій, відмивання коштів, фінансування незаконної діяльності та прихованого переміщення капіталу. У таких умовах виникає гостра потреба у системах, здатних автоматично аналізувати транзакційну поведінку та визначати рівень ризику взаємодії з певними адресами.

Вирішенням цієї проблеми є створення сервісу аналізу транзакцій, який здатний об'єднувати адреси у кластери, розраховувати їхні поведінкові характеристики та прогнозувати категорії ризику за допомогою моделей машинного навчання. Він дозволить не лише зменшити ймовірність взаємодії з підозрілими гаманцями, але й підвищить рівень прозорості та безпеки у сфері цифрових активів.

Метою кваліфікаційної роботи є проектування та розробка вебсервісу аналітики транзакцій цифрових активів на наявність операцій з підвищеним фінансовим ризиком, здатного формувати кластери адрес та виконувати їх подальшу класифікацію за допомогою методів машинного навчання.

Щоб досягти поставленої мети, необхідно виконати такі **завдання**:

1. Аналіз предметної області та порівняння існуючих аналогів.
2. Вибір технологій необхідних для реалізації вебсервісу.
3. Побудова, навчання та інтегрування моделей машинного навчання для прогнозування категорії кластерів.
4. Розробка алгоритмів кластеризації адрес на основі евристичних правил та формування набору ознак для подальшої класифікації.
5. Інтеграція всіх модулів системи.

6. Перевірка роботи кінцевого продукту, усунення виявлених недоліків та оптимізація.

Об'єктом дослідження є процеси аналітики блокчейн-даних цифрових активів, пов'язані з формуванням та оцінюванням кластерів адрес на основі транзакційних зв'язків.

Предметом дослідження є методи та програмні засоби аналізу криптовалютних транзакцій, що використовуються для побудови вебсервісу аналітики операцій з цифровими активами та оцінювання ризиковості кластерів адрес.

Наукова новизна полягає у поєднанні класичних евристик кластеризації блокчейн-адрес з методами машинного навчання для оцінювання ризиковості кластерів. Такий підхід дозволяє формувати узагальнені характеристики груп адрес і автоматично класифікувати їх за рівнем потенційної небезпеки.

Практичне значення такого вебсервісу полягає у можливості швидкої попередньої перевірки криптовалютних адрес, виявлення зв'язків між ними та оцінки ризиків взаємодії. Розроблені методи можуть використовуватися фінансовими установами, підприємствами та звичайними користувачами для підвищення безпеки операцій та запобігання участі у підозрілих транзакціях.

Апробація. За результатами роботи було:

1) опубліковано тези доповіді Кочура В.В., Локтікова Т.М., Кушнір Н.О. Аналітичний вебсервіс для виявлення ризикових транзакцій цифрових активів на основі AML та KYC. Тези доповідей XV Міжнародної науково-технічної конференції «Інформаційно-комп'ютерні технології», 28–29 березня 2025 року. Житомир: «Житомирська політехніка», 2025. С. 19–20 [1];

2) подано та прийнято до опублікування наукову статтю Kochura V., Loktikova T., Kushnir N. Analytical web service for identifying suspicious high-risk digital asset transactions. Computer systems and information technologies. 2025. № 4 [2].

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Теоретичні засади та вихідні положення

У сучасному віртуальному просторі значну роль відіграють цифрові активи – одиниці вартості, які існують виключно у електронній формі та можуть зберігатися, передаватися й обмінюватися без участі традиційних фінансових установ. Найвідомішим різновидом таких активів є криптовалюта. На відміну від звичайних грошей, криптовалюта не існує у матеріальній формі та не випускається централізованим органом. Її робота базується на застосуванні криптографії та спеціальних протоколів, які забезпечують захист даних і цілісність операцій [3].

Першою та найбільш поширеною криптовалютою стала монета Bitcoin [4]. Саме вона продемонструвала можливість створення цифрових грошей, які можуть передаватися між користувачами напрямку, без банків чи посередників. Обіг Bitcoin, як і більшості інших криптовалют, здійснюється на основі технології блокчейн – розподіленого реєстру, у якому зберігається вся історія транзакцій. Цей реєстр є відкритим, прозорим і незмінним, що робить інформацію про перекази доступною для аналізу та дослідження.

Блокчейн можна розглядати як послідовність блоків, у кожному з яких зберігається набір підтверджених транзакцій [5, 6]. Кожен блок пов'язаний з попереднім за допомогою криптографічних методів, що забезпечує цілісність і незмінність даних. Завдяки такій структурі мережа працює без центрального керуючого органу, а перевірка транзакцій здійснюється самими учасниками системи.

Користувачі взаємодіють із блокчейн-мережею через криптовалютні гаманці та адреси, які виконують роль ідентифікаторів для здійснення операцій. Транзакції між такими адресами формують складну мережу взаємозв'язків, що дозволяє досліджувати рух цифрових активів та виявляти характерні закономірності [7]. Хоча адреси є публічними, вони не пов'язані безпосередньо

з особистими даними власників, що створює псевдонімність і ускладнює ідентифікацію без застосування спеціалізованих методів аналізу.

У низці криптовалютних мереж транзакції здійснюються за моделлю UTXO, зокрема в мережі Bitcoin [8, 9]. У цій моделі не існує поняття балансу рахунку: замість цього мережа оперує набором «невитрачених виходів» попередніх транзакцій. Кожен UTXO може бути використаний як вхід у новій операції, а отримані в ній виходи створюють нові UTXO. Сума всіх доступних UTXO визначає баланс конкретної адреси. Такий підхід забезпечує прозорість руху коштів і дає змогу точно відстежувати їх походження, що є важливим у контексті подальшого аналізу транзакцій.

Окремі адреси можуть брати участь у спільних транзакціях, повторюваних операціях або формувати стійкі взаємозв'язки між собою. У результаті виникають групи адрес, що поводяться узгоджено та фактично належать одному користувачу або одному об'єкту у мережі. Такі групи зазвичай називають кластерами [10]. Аналіз їх структури та поведінки дає змогу точніше відстежувати взаємодії учасників і виявляти характерні моделі руху активів. Можливість об'єднання адрес у кластери відкриває шлях до глибшого дослідження транзакцій та створює підґрунтя для побудови систем, здатних оцінювати потенційні ризики взаємодії з різними учасниками блокчейн-мережі та протидіяти відмиванню коштів [11].

Походження таких кластерів може суттєво відрізнитися: одні формуються навколо звичайних користувачів або сервісів, тоді як інші можуть бути пов'язані з діяльністю платформ сумнівного характеру чи з операціями незрозумілого походження [12]. У таких випадках взаємодія з адресами подібних кластерів може становити потенційну загрозу, що підсилює необхідність їх детального дослідження та попереднього аналізу перед встановленням будь-яких транзакційних взаємозв'язків.

1.2 Постановка задачі

Проведений аналіз особливостей функціонування блокчейн-мереж та моделей руху цифрових активів показує, що прозорість транзакцій поєднується з високим рівнем псевдонімності їх учасників. Незважаючи на відкриту структуру даних, встановлення зв'язків між адресами та визначення їх належності до конкретних користувачів або сервісів потребує спеціалізованих методів обробки та інтерпретації. Це ускладнює процес оцінювання характеру транзакцій та виявлення потенційно небажаних взаємодій у мережі.

Однією з основних проблем сучасного аналізу цифрових активів є складність оцінювання ризиковості окремих адрес. Хоча блокчейн містить повну історію транзакцій, дані представлені у вигляді великого обсягу низькорівневої інформації, яка потребує попередньої обробки, агрегування та інтерпретації. Традиційні підходи до фінансового моніторингу, розроблені для централізованих систем, виявляються недостатньо ефективними у децентралізованому середовищі, де відсутня пряма прив'язка адрес до користувачів. Це підсилює актуальність створення інструментів, здатних на основі транзакційних даних формувати кластери, їх узагальнені характеристики та застосовувати методи машинного навчання для оцінювання.

Метою даного дослідження є розробка вебсервісу для аналітики транзакцій цифрових активів, який забезпечує автоматизоване формування транзакційних характеристик адрес і кластерів, визначення можливих взаємозв'язків між ними та оцінювання ризиковості взаємодії з такими об'єктами. Це дозволить проводити поглиблений аналіз руху цифрових активів, виявляти потенційно небажані взаємодії та підвищувати рівень безпеки користувачів у блокчейн-середовищі.

Під час проектування та розробки слід виконати ряд задач.

- 1) Здійснити огляд та аналіз існуючих підходів і програмних рішень для аналітики транзакцій цифрових активів.
- 2) Дослідити сучасні методи кластеризації адрес у блокчейн-мережах.

- 3) Визначити набір характеристик, що можуть бути використані для оцінювання поведінки кластерів.
- 4) Розробити архітектуру вебсервісу, яка забезпечує збір, обробку та аналіз транзакційних даних.
- 5) Реалізувати серверну частину, що забезпечує отримання, структурування та попередню обробку даних про кластери для подальшого формування їхніх характеристик;
- 6) Інтегрувати модуль машинного навчання для оцінювання ризику кластерів на основі сформованих ознак.
- 7) Реалізувати зручний та інтуїтивно-зрозумілий користувацький інтерфейс, що забезпечуватиме ефективну взаємодію користувачів із сервісом.
- 8) Провести тестування сервісу на реальних даних для оцінки якості формування ознак та точності роботи моделей ризиковості.

Результатом виконання роботи стане вебсервіс, здатний аналізувати транзакційні дані блокчейн-мереж, формувати узагальнені характеристики кластерів та оцінювати можливі ризики взаємодії з їх адресами. Запропонований підхід сприятиме підвищенню прозорості транзакцій, дасть змогу застосовувати сучасні методи аналізу до блокчейн-даних та робити висновки про поведінку кластерів.

1.3 Аналіз існуючих програмних продуктів

З огляду на складність транзакційних взаємозв'язків у блокчейн-мережах і потребу в інструментах, які спрощують їх дослідження, на ринку з'явився ряд спеціалізованих продуктів, що пропонують різні підходи до аналізу цифрових активів. Ці сервіси дають можливість досліджувати структуру транзакцій, візуалізувати рух коштів, виявляти взаємопов'язані групи адрес та оцінювати їх поведінкові особливості.

Для того, щоб сформувати остаточне бачення, варто проаналізувати схожі продукти, визначивши їх переваги та недоліки. Порівняємо декілька з них.

Elliptic (<https://www.elliptic.co/>) є однією з провідних платформ у сфері блокчейн-аналітики, яка пропонує широкий спектр інструментів для дослідження транзакцій цифрових активів [13]. На рис. 1.1 представлено інтерфейс сервісу Elliptic. Сервіс забезпечує побудову транзакційних графів, визначення взаємозв'язків між адресами та кластеризацію об'єктів на основі їхньої поведінки. Його головна перевага – потужний алгоритм машинного навчання, який дозволяє виявляти складні схеми відмивання грошей. Завдяки цьому Elliptic активно використовується для глибокого аналізу процесів у блокчейн-мережах та дослідження структури руху активів.

Попри широкі аналітичні можливості, платформа має певні обмеження. Інтерфейс Elliptic орієнтований на досвідчених аналітиків, що може ускладнювати використання сервісу для нових користувачів. Платформа є закритою, а процес отримання доступу до її інструментів може займати певний час. Окрім цього, сервіси надаються лише на платній основі та орієнтовані переважно на корпоративний сегмент.



Рисунок 1.1 – Інтерфейс сервісу Elliptic

Crystal Intelligence (<https://crystalintelligence.com/>) – це аналітична платформа, орієнтована на дослідження та візуалізацію транзакцій цифрових активів [14]. На рис. 1.2 зображено інтерфейс сервісу Crystal Intelligence. Сервіс пропонує інструменти для побудови детальних транзакційних графів, виявлення взаємопов'язаних адрес та аналізу структури руху активів. Однією з основних переваг Crystal Intelligence є наочна графічна інтерпретація даних, яка дозволяє легко простежувати ланцюги переказів та виявляти взаємодії між учасниками мережі. Завдяки цьому платформа активно використовується для глибокого аналізу транзакційних потоків і дослідження поведінкових особливостей адрес.

Втім, Crystal Intelligence має певні обмеження. Повноцінна робота з сервісом передбачає наявність платного доступу, що може обмежувати можливості швидкого ознайомлення з інструментами сервісу.

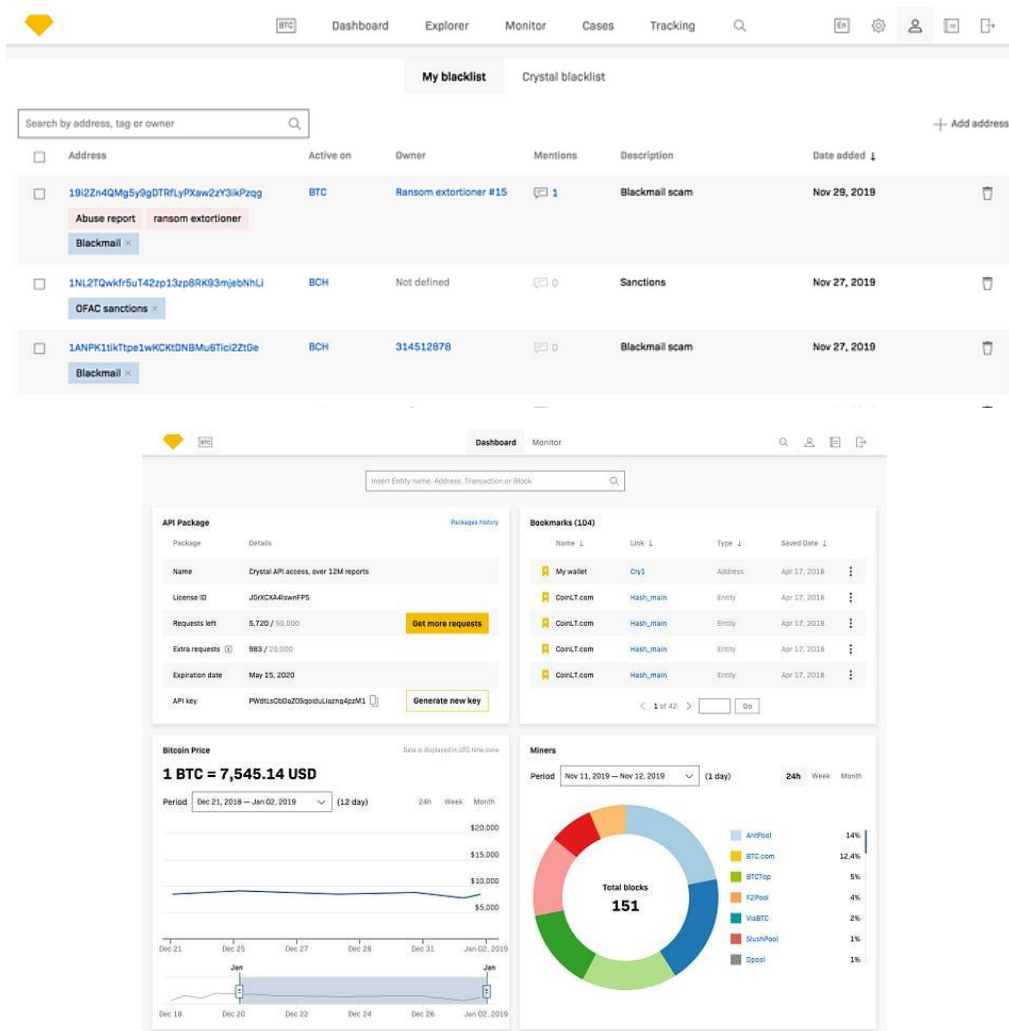


Рисунок 1.2 – Інтерфейс сервісу Crystal Intelligence

Check Crypto Address (<https://checkcryptoaddress.com/>) – це онлайн-інструмент, орієнтований на базову перевірку криптовалютних гаманців. Рис. 1.3 демонструє інтерфейс сервісу Check Crypto Address. Сервіс дозволяє визначати валідність адреси, переглядати її основні характеристики та отримувати короткі відомості щодо активності. Інструмент працює без реєстрації, має простий інтерфейс та підтримує низку різних адресних форматів. Завдяки цьому Check Crypto Address є зручним для швидкої первинної перевірки, коли користувачеві потрібно лише підтвердити коректність адреси або її наявність у загальнодоступних базах.

Однак сервіс має обмежений функціонал у контексті глибокої аналітики блокчейн-даних. Він не надає детальної інформації про історію транзакцій чи взаємозв'язки з іншими адресами. Доступна інформація здебільшого містить загальну оцінку ризику та застереження про можливі ознаки підозрілої активності, сформовані автоматично або на основі повідомлень спільноти. Це робить Check Crypto Address корисним лише для поверхневої перевірки, але недостатнім для комплексного аналізу цифрових активів.

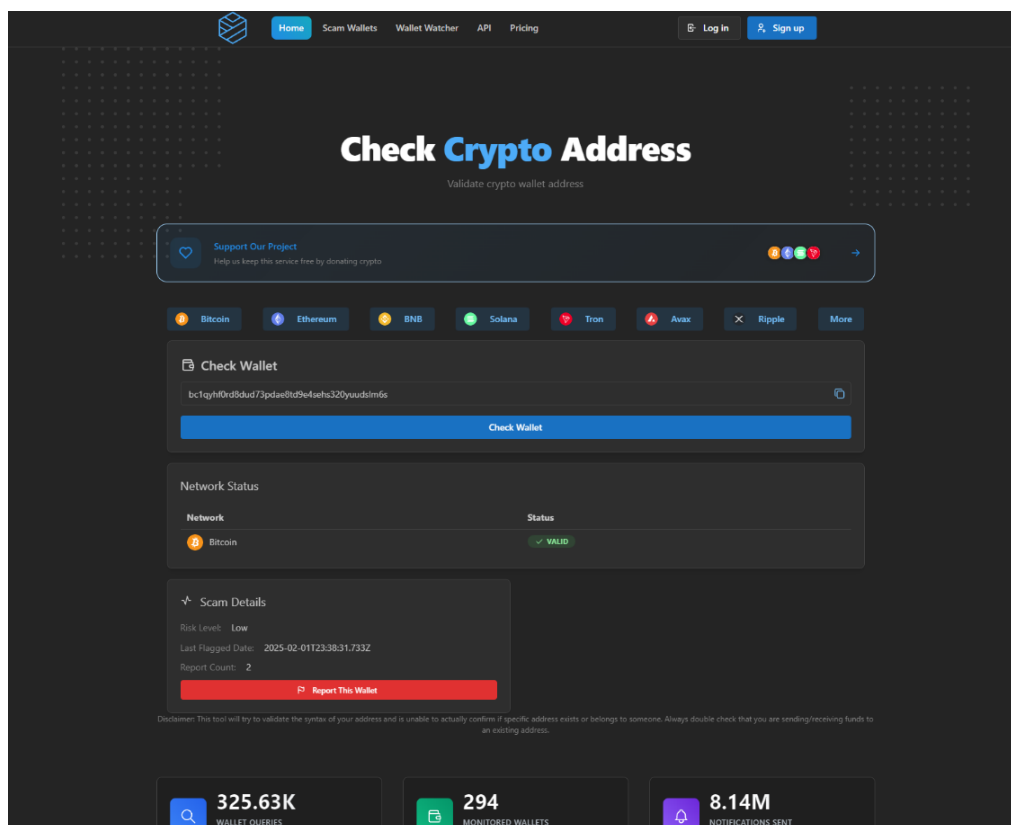


Рисунок 1.3 – Інтерфейс сервісу Check Crypto Address

Після проведення аналізу існуючих аналогів, було виділено основні критерії та створено зведену таблицю їх порівняння (табл. 1.1).

Таблиця 1.1

Порівняльна таблиця аналогів

Найменування	Elliptic	Crystal Intelligence	Check Crypto Address
Точність та якість аналізу	Висока	Висока	Низька
Кластеризація адрес	Присутня	Частково присутня	Відсутня
Оцінка ризиковості	Присутня	Присутня	Відсутня
Використання ІІІ	Присутнє	Присутнє	Відсутнє
Зручність та привабливість інтерфейсу	Середня	Висока	Висока
Візуалізація даних та аналізу	Середня	Висока	Низька
Доступність	Платний доступ	Платний доступ	Вільний доступ

Кожен з обраних ресурсів має свої переваги та недоліки, які обов'язково будуть враховані при розробці власного продукту. Деякі з ідей будуть використані або вдосконалені в розроблюваній системі.

Ознайомившись з аналогами було остаточно сформовано бачення розробки та підхід до неї. Також за допомогою аналізу визначено основні аспекти, завдяки яким в майбутньому розроблюваний продукт зможе конкурувати на ринку. Для цього необхідно забезпечити високу точність аналізу транзакційних даних, підтримку кластеризації адрес та формування характеристик їх поведінки, використання моделей машинного навчання для оцінювання ризиковості, а також створити зручний інтерфейс.

1.4 Аналіз методів аналітики транзакцій цифрових активів

Одним із найбільш важливих і відповідальних етапів аналізу транзакцій цифрових активів є кластеризація адрес – процес об'єднання кількох публічних

ключів у групи, які з високою ймовірністю належать одному користувачу або одному об'єкту у мережі. У контексті блокчейну публічний ключ, як правило, використовується у вигляді адреси – ідентифікатора, на який надходять кошти та з якого здійснюються транзакції. Оскільки сам реєстр не містить прямої інформації про власників таких адрес, для побудови кластерів застосовуються спеціальні евристики, що дозволяють зменшити рівень псевдонімності та отримати узагальнену картину руху активів [15].

Однією з найпоширеніших евристик у мережах, що використовують модель UTXO, є Multi-Input Heuristic (також відома як Common-Input-Ownership Heuristic або Multi-Input Address Clustering) [16, 17]. На рис 1.4 наведено схематичну діаграму цієї евристики. В її основі лежить припущення, що якщо транзакція має кілька вхідних адрес, то всі ці адреси контролює один користувач або організація. Це пояснюється тим, що для створення транзакції необхідно мати доступ до приватних ключів усіх вхідних адрес, а отже, вони мають належати одному власнику. Завдяки простоті й ефективності цей метод отримав широке застосування та став фактично стандартом у початкових етапах кластеризації адрес.

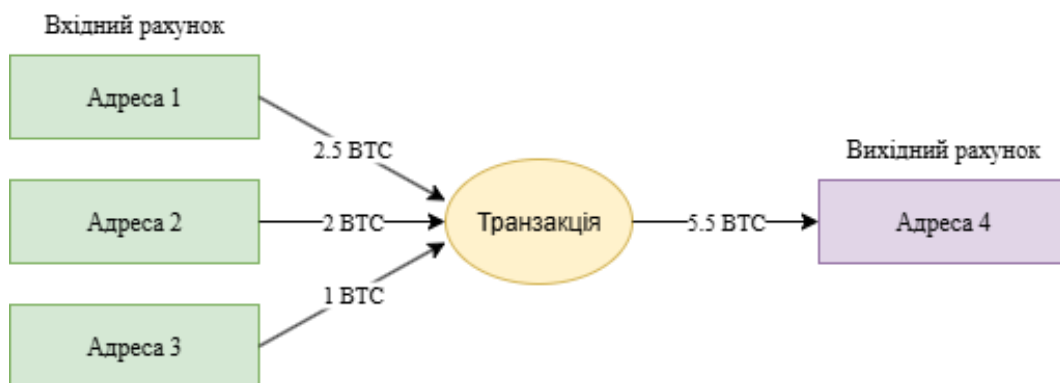


Рисунок 1.4 – Схематична діаграма евристики Multi-Input Heuristic

Ще одним важливим підходом є евристика Change Address Heuristic, спрямована на виявлення адреси «решти» [18 – 20]. На рис 1.5 представлено схематичну діаграму цієї евристики. Під час створення транзакції користувач витрачає один або кілька UTXO, а сума, що перевищує необхідний розмір переказу, повертається йому у вигляді решти. У більшості сучасних гаманців така решта надходить на нову адресу, а не на вже існуючу. Вона генерується

автоматично для підвищення приватності користувача. Якщо одна адреса виходу транзакції була використана раніше, а інша є новою, то останню з високою ймовірністю можна вважати адресою «решти». Таким чином, кластер може бути розширений за рахунок додаткових адрес, що контролюються тим самим користувачем.

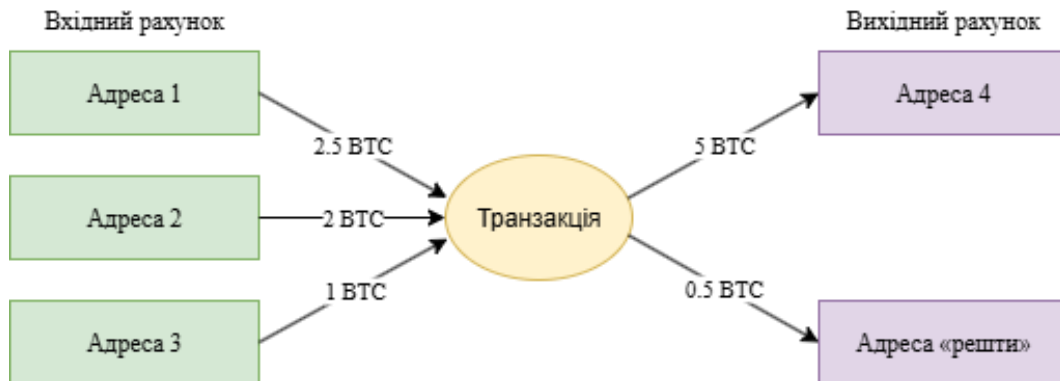


Рисунок 1.5 – Схематична діаграма евристики Change Address Heuristic

Крім того, для підвищення точності ідентифікації застосовується евристика Force Merge of Inputs Heuristic [21]. Вона базується на аналізі ситуацій, коли на одній адресі бракує коштів для здійснення платежу, і гаманець змушений об'єднати UTXO з кількох контрольованих адрес, створюючи транзакцію з кількома входами. У цьому контексті аналіз сум дозволяє точно визначити адресу «решти». Евристика спирається на математичне співвідношення: якщо різниця між загальною сумою вхідних коштів та мінімальною необхідною сумою для здійснення переказу є меншою за суму, що надходить отримувачу, то вихід із меншою сумою вважається адресою «решти».

Важливо зазначити, що подібні евристики мають ймовірнісний характер і не гарантують стовідсоткової точності. Похибки можуть виникати у випадках навмисного приховування зв'язків, роботи зі складними схемами розподілу коштів або використання сервісів з підвищеним рівнем приватності.

Після етапу кластеризації виникає потреба в інструментах, здатних проаналізувати узагальнені характеристики таких кластерів і класифікувати їх за певними поведінковими ознаками.

Для вирішення цього завдання доцільно застосовувати методи машинного навчання, які можуть працювати з параметрами кластерів, виявляти приховані залежності та визначати їх ймовірну категорію. Серед методів, що показують високу ефективність у задачах класифікації та аналізу структурованих даних, можна виділити моделі Random Forest, Extra Trees та XGBoost, які дозволяють формувати точні прогностичні оцінки на основі багатовимірних ознак.

Random Forest є одним із базових методів ансамблевого машинного навчання, заснованим на побудові великої кількості незалежних дерев рішень [22]. Кожне дерево формує власне припущення, а фінальний прогноз отримується шляхом агрегування їхніх результатів. Такий підхід дозволяє істотно зменшити ризик перенавчання, підвищує стійкість моделі до шуму та забезпечує високу якість класифікації на складних даних. Random Forest добре працює з нерівномірними транзакційними ознаками, що робить його придатним для задачі визначення категорій кластера за його поведінковими характеристиками.

Extra Trees (Extremely Randomized Trees) є варіацією методу Random Forest, але використовує підвищену випадковість під час побудови дерев [23]. На відміну від класичного підходу, де поділ вузлів обирається за найбільш інформативним критерієм, Extra Trees обирає точки розбиття випадково, що робить модель більш простою та швидкою у навчанні. Попри таку випадковість, метод часто демонструє високу точність завдяки великій кількості дерев. Extra Trees ефективно працює в задачах, де ознаки мають складні або нерівномірні розподіли, що характерно для транзакційних даних блокчейну.

XGBoost належить до методів градієнтного бустингу і побудований як послідовність дерев рішень, де кожне наступне дерево намагається компенсувати помилки попередніх [24]. Такий підхід дозволяє моделі захоплювати складні залежності між ознаками та досягати високої точності на структурованих даних при значній швидкості обробки. XGBoost використовує регуляризацію, що зменшує ризик перенавчання, а також підтримує роботу з

пропущеними значеннями та великою кількістю параметрів. Завдяки своїй гнучкості та ефективності XGBoost добре підходить для аналізу кластерів криптовалютних адрес та визначення їх можливої ризиковості.

Підсумовуючи, евристики дозволять сформувати кластери адрес, тоді як методи машинного навчання забезпечать подальший етап – класифікацію на основі їх ознак.

1.5 Обґрунтування вибору засобів реалізації програмного продукту

Вибір засобів реалізації вебсервісу є важливим етапом на початковій стадії розробки, оскільки від нього залежить не лише успішність проекту, але й ефективність його розробки, масштабованість та подальша підтримка.

Для розробки серверної частини було обрано широковідому мову програмування PHP, яка зарекомендувала себе як надійний інструмент з широкою спільнотою розробників, що говорить про її популярність та робить легкою у вивченні й використанні.

PHP має досить велику кількість фреймворків, що значно спрощують та прискорюють процес розробки. Одним із таких є Symfony, його модульна система компонентів є досить гнучкою, і дозволяє виконувати стандартні завдання без написання зайвого коду [25]. Крім того, він забезпечує високий рівень безпеки програмного забезпечення та його стабільність. При цьому, фреймворк пропонує широкий вибір готового функціоналу, компонентів та бібліотек, що може суттєво мінімізувати витрату часу при реалізації проекту. Symfony також надає зручну систему налаштувань і конфігурацій, що дозволить легко адаптувати фреймворк до специфічних вимог проекту.

Для оптимізації обробки запитів було обрано фреймворк API Platform, який забезпечує повноцінний набір інструментів для швидкої та зручної роботи з API відповідно до сучасних стандартів. Фреймворк легко інтегрується з Symfony, підтримує автоматичну генерацію CRUD-операцій та спрощує розгортання будь-яких сервісів, що значно пришвидшить розробку.

Модулі машинного навчання будуть реалізовані на мові Python, яка є стандартом у сфері аналізу даних та штучного інтелекту [26]. Її популярність зумовлена широкою екосистемою бібліотек для роботи з моделями, зокрема NumPy та Pandas для обробки даних, а також scikit-learn, що використовується для навчання моделей Random Forest та Extra Trees. Для моделі XGBoost застосовується окрема бібліотека xgboost. Завдяки використанню фреймворку FastAPI Python-модуль може працювати як окремий сервіс, обробляючи запити на класифікацію та повертаючи результати прогнозування у режимі реального часу.

У проєкті буде використовуватися система управління базами даних MySQL, яка вирізняється зручністю у використанні, простотою в освоєнні та широкою популярністю, завдяки чому легко можна знайти потрібну документацію. MySQL визнана однією з найбезпечніших та найнадійніших систем керування базами даних, яка використовується в різних популярних вебдодатках.

В якості вебсервера обрано Nginx. Він відзначається високою продуктивністю, можливістю ефективно обробляти велику кількість одночасних підключень та надійністю. Крім того, Nginx забезпечує гнучке керування навантаженням та підтримку кешування, що покращить швидкість і стабільність роботи вебзастосунку.

Щоб реалізувати клієнтську частину було вибрано найпопулярнішу JavaScript-бібліотеку – React. Він використовується для швидкого та ефективного створення інтерактивних користувацьких інтерфейсів і вебдодатків із застосуванням значно меншої кількості коду, ніж під час використання звичайного JavaScript. React пропонує розробникам розділити призначені для користувача інтерфейси на окремі компоненти, які можна використовувати повторно.

Для розгортання системи використовуватиметься Docker, котрий дозволяє представляти компоненти системи у вигляді контейнерів, легко управляти ними та масштабувати [27]. Однією з основних переваг є ізоляція

цих компонентів та їх залежностей, що гарантує стабільну роботу системи. Його використання значно підвищить ефективність та швидкість розробки продукту.

Даний набір технологій забезпечить зручну та ефективну розробку й при його правильному використанні буде отримано довершену систему, котра задовольнить поставлені вимоги та завдання.

Висновки до першого розділу

У першому розділі було проведено аналіз теоретичних засад, необхідних для побудови системи аналітики транзакцій цифрових активів. Розглянуто особливості функціонування блокчейн-мереж, модель UTXO, механізми формування транзакцій та фактори, що ускладнюють ідентифікацію взаємозв'язків між адресами. Проаналізовано природу появи кластерів адрес, їх роль у дослідженні руху активів та причини, що зумовлюють необхідність їхнього детального вивчення в контексті оцінювання ризиковості.

Проведено аналіз існуючих програмних продуктів у сфері блокчейн-аналітики, серед яких Elliptic, Crystal Intelligence та Check Crypto Address. На основі їхнього порівняння визначено переваги, обмеження та функціональні особливості кожного рішення. Після чого сформувалось чітке бачення майбутнього вебсервісу, що дозволить створити якісний продукт із урахуванням сучасних вимог ринку.

Окрему увагу приділено сучасним підходам до аналітики блокчейн-даних, зокрема методам кластеризації адрес. Досліджено ключові евристики – Multi-Input Heuristic, Change Address Heuristic та Force Merge of Inputs Heuristic, які дозволяють відновлювати ймовірні зв'язки між адресами та формувати кластери на основі їх транзакційної поведінки. Також проаналізовано можливості застосування моделей машинного навчання, таких як Random Forest, Extra Trees та XGBoost, що забезпечують ефективну класифікацію кластерів і визначення потенційного рівня ризику взаємодії з ними.

Зрештою обґрунтовано вибір інструментів та технологій, необхідних для реалізації вебсервісу. Визначено доцільність використання PHP та фреймворку Symfony для серверної частини, API Platform для побудови REST-інтерфейсів, Python для модулів машинного навчання, MySQL та PostgreSQL для збереження даних, Nginx як продуктивного вебсервера, а також React для створення сучасного клієнтського інтерфейсу.

Таким чином, у першому розділі було сформовано теоретичне підґрунтя для подальшої реалізації системи. Проведений аналіз дав змогу визначити проблематику, актуальність, основні задачі, методи та засоби розв'язання, що формує необхідні умови для проектування та розробки вебсервісу аналітики транзакцій цифрових активів.

РОЗДІЛ 2 РОЗРОБКА АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ ТА МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

2.1 Аналіз використовуваного набору даних

Для реалізації задачі класифікації кластерів адрес у даній роботі використано відкритий науковий датасет, побудований на основі транзакційної активності в мережі Bitcoin, опублікований дослідницькою групою у виданні Scientific Data (Nature Portfolio) [28]. Набір даних містить агреговану інформацію про кластери Bitcoin-адрес, їхню структурну та транзакційну активність, а також категорії ризиковості. Використання цього датасету гарантує роботу з перевіреними даними, що сприяє підвищенню достовірності результатів і точності подальшого аналізу.

За будовою датасет складається з двох основних компонентів: множини вузлів, що описують окремі кластери Bitcoin-адрес, та множини ребер, які відображають транзакційні взаємодії між такими кластерами. Кожен вузол містить детальний набір характеристик, які сформовані на основі первинних даних з блокчейну. Для потреб даного дослідження використовуються саме вузли, тобто кластери, оскільки вони містять повний набір інформаційних ознак, необхідних для класифікації. Ребра не залучаються в процесі роботи з моделями машинного навчання. Такий підхід дозволяє зосередитися на поведінкових характеристиках окремих кластерів без аналізу глобальної топології всієї мережі.

Датасет надається у форматі дампу PostgreSQL, що містить повну структуру бази даних та наповнення відповідних таблиць. Такий підхід дозволяє відтворити вихідне середовище зберігання даних, уникнути втрати інформації під час імпорту та забезпечує зручність подальшої обробки великих обсягів записів.

Основним джерелом даних для побудови моделей класифікації є таблиця «node_features». Саме в ній міститься повний набір кількісних та структурних характеристик, що описують поведінку кластера в транзакційній мережі. Поля «node_features» формують багатовимірний профіль кожного вузла та

використовуються як ознаки для навчання моделей машинного навчання. Нижче подано опис атрибутів цієї таблиці, що були використані під час роботи (табл. 2.1).

Таблиця 2.1

Основні атрибути таблиці «node_features»

Назва поля	Опис
Alias	Ідентифікатор вузла (кластера)
Label	Категорія, що описує тип сутності, яку представляє кластер
total_transaction_in	Загальна кількість вхідних переказів, отриманих кластером
total_transaction_out	Загальна кількість вихідних переказів, відправлених кластером
first_transaction_in	Номер блоку, в якому відбулося перше надходження коштів на адреси кластера
last_transaction_in	Номер блоку, в якому відбулося останнє надходження коштів на адреси кластера
first_transaction_out	Номер блоку, в якому відбулася перша відправка коштів з адрес кластера
last_transaction_out	Номер блоку, в якому відбулася остання відправка коштів з адрес кластера
min_sent	Мінімальна сума, відправлена кластером за одну транзакцію
max_sent	Максимальна сума, відправлена кластером за одну транзакцію
total_sent	Загальна сума, відправлена кластером за весь час активності
min_received	Мінімальна сума, отримана кластером за одну транзакцію
max_received	Максимальна сума, отримана кластером за одну транзакцію
total_received	Загальна сума, отримана кластером за весь час активності
cluster_size	Кількість адрес, що входять до складу кластера
cluster_num_edges	Кількість транзакційних зв'язків між адресами всередині кластера
cluster_num_cc	Кількість компонент зв'язності у внутрішньому транзакційному графі кластера
cluster_num_nodes_in_cc	Кількість адрес у кластері, які мають хоча б одну внутрішню транзакцію (входять до складу компонент зв'язності)

Серед усіх доступних вузлів у наборі даних, загальна кількість яких перевищує 252 мільйони, лише певна частина містить визначені категорії, які потрібні для задачі класифікації. Загальна кількість таких маркованих вузлів

становить приблизно 34 тисячі, і саме вони використовуються як навчальна вибірка для побудови моделей машинного навчання.

Далі наведено перелік категорій наявних у датасеті та їх опис, що дозволяє зрозуміти змістове призначення кожної з них у контексті аналізу транзакційної поведінки кластерів.

Individual являє собою гаманець, що належить окремій особі та не пов'язаний з діяльністю сервісів, платформ чи організацій.

Mining являє собою користувача або організацію, що здійснює підтвердження транзакцій у мережі Bitcoin і отримує винагороду за створення блоків.

Exchange являє собою онлайн-платформу, що забезпечує обмін, купівлю та продаж криптовалют і фіатних валют.

Marketplace являє собою онлайн-платформу, де користувачі можуть купувати та продавати товари чи послуги з використанням Bitcoin як засобу оплати.

Gambling являє собою онлайн-платформу, що надає можливість робити ставки, брати участь у лотереях або грати в азартні ігри за допомогою Bitcoin.

Bet являє собою спеціально створену адресу, яку гемблінг-сервіс використовує для отримання коштів за конкретну ставку.

Faucet являє собою сервіс, що роздає невелику кількість монет Bitcoin як винагороду за виконання дій, перегляд реклами або інші прості завдання.

Mixer являє собою сервіс, що підвищує приватність транзакцій, ускладнюючи можливість їх відстеження шляхом змішування коштів різних користувачів.

Ropzi являє собою фінансову аферу, де виплати старим інвесторам здійснюються за рахунок залучення нових, без реальної інвестиційної діяльності [29].

Ransomware являє собою зловмисне ПЗ, яке блокує доступ до файлів або системи та вимагає викуп за їх розблокування [30].

Bridge являє собою протокол, що забезпечує передачу активів між Bitcoin та іншими блокчейн-мережами (наприклад, Ethereum).

Аналіз маркованої частини датасету показав, що розподіл категорій є нерівномірним: деякі типи кластерів представлені значно частіше, тоді як інші зустрічаються у поодиноких випадках (рис. 2.1). Така диспропорція впливає на процес навчання моделей машинного навчання, оскільки моделі можуть схилитися до переваги більш чисельних класів. Тому особливу увагу приділено оцінюванню моделей за допомогою метрик, стійких до дисбалансу, зокрема F1-score.

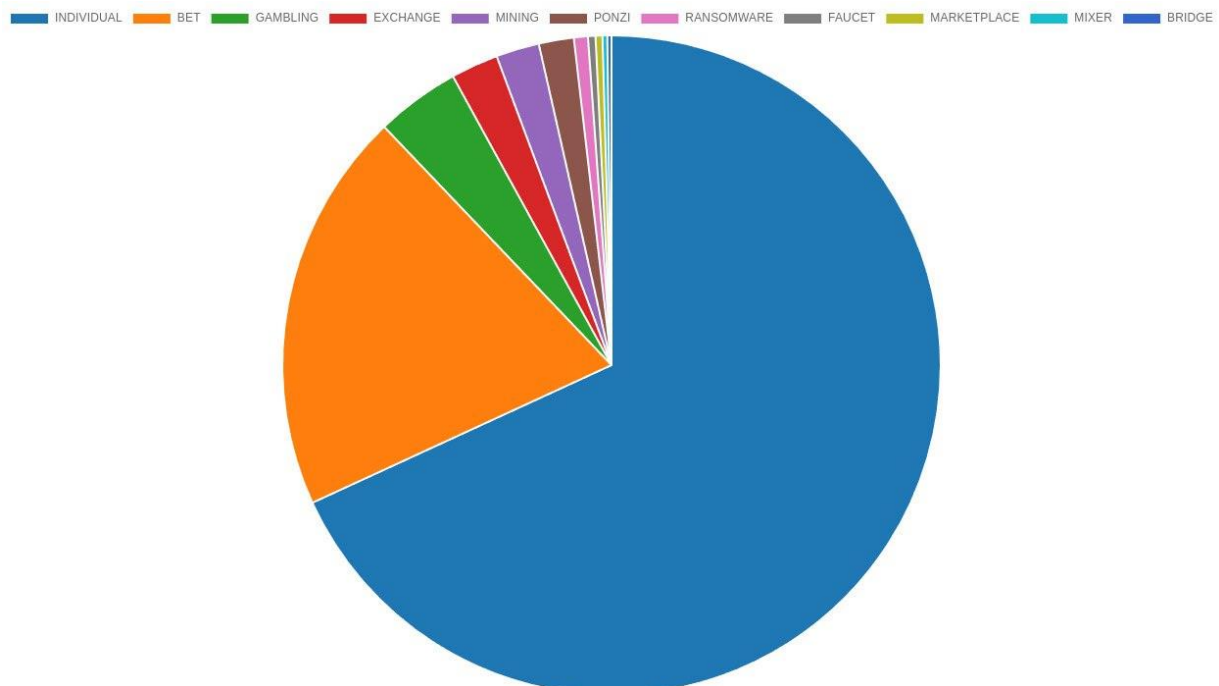


Рисунок 2.1 – Розподіл категорій серед маркованих вузлів

Таким чином, проаналізований набір даних забезпечує усі необхідні умови для подальшого формування вибірок, обчислення ознак та побудови моделей класифікації. Детальна структура таблиці «node_features» та наявність маркованої підмножини кластерів дозволяють здійснювати контрольоване машинне навчання (supervised learning) і досліджувати поведінкові характеристики різних типів кластерів. Отримані результати аналізу даних стали основою для наступного етапу – опрацювання ознак кластерів та побудови моделей для їх класифікації.

2.2 Реалізація алгоритмів машинного навчання

Після проведеного аналізу структури та характеристик використовуваного набору даних, було здійснено практичну реалізацію алгоритмів машинного навчання мовою програмування Python. На цьому етапі виконано формування ознак кластерів, підготовку навчальних вибірок, побудову моделей та налаштування параметрів їх навчання. Особливу увагу приділено коректній обробці маркованих даних, вибору інформативних характеристик та реалізації алгоритмів Random Forest, Extra Trees та XGBoost.

Перш за все, варто було завантажити марковані дані з таблиці «node_features», котра описана вище. Для роботи з базою даних було використано допоміжний клас PostgresqlDatabase, який реалізує логіку підключення до PostgreSQL за допомогою бібліотеки psycopg2, виконання SQL-запитів та повернення результатів у зручному форматі. Він містить універсальний метод select, який дозволяє здійснювати вибірку даних з таблиць, у тому числі з використанням фільтрації за певними значеннями.

Для формування навчальної вибірки необхідно було отримати лише ті вузли, які мають визначену категорію. Перелік категорій попередньо задається в константі CATEGORIES. Таким чином, відбувається фільтрація лише маркованих кластерів, що надалі виступають прикладами для тренування моделей.

```
db = PostgresqlDatabase(**settings.postgresql_database)

labelled_nodes = db.select(
    table="node_features",
    columns=["alias", "label"],
    where_column="label",
    where_in=CATEGORIES
)

labelled_nodes = { int(row["alias"]): row["label"] for row in labelled_nodes }
```

У результаті формується словник, де ключем виступає унікальний ідентифікатор кластера, а значенням – його категорія.

Наступним кроком є розподіл на тренувальну та тестову вибірки. Це необхідно для об'єктивного оцінювання якості моделей машинного навчання,

оскільки дозволяє перевірити їхню роботу на даних, які не використовувалися під час навчання.

Для цього застосовано стандартний метод `train_test_split` із бібліотеки `scikit-learn`, який забезпечує випадкове та репрезентативне розділення на дві множини: тренувальну вибірку (70% даних), тестову вибірку (30% даних).

```
train_nodes, test_nodes = train_test_split(node_ids, test_size=0.3, random_state=42)
```

Таким чином, усі кластери поділяються на дві множини за єдиним принципом. У результаті тренування моделі відбувається на більшій частині маркованих вузлів, а їхня узагальнювальна здатність перевіряється на тестових даних, яких модель раніше «не бачила».

Після формування вибірок наступним етапом є отримання повних наборів ознак для кожного кластера та їх попередня обробка. Ці операції реалізовані за допомогою двох компонентів: `NodeFeaturesExtractor` та `Preprocessor`.

```
extractor = NodeFeaturesExtractor(db=db, max_nodes=10000)
```

```
X_train, y_train = extractor.extract(train_nodes, label_values=CATEGORIES)
```

```
X_test, y_test = extractor.extract(test_nodes, label_values=CATEGORIES)
```

```
y_train = pd.Series(y_train)
```

```
y_test = pd.Series(y_test)
```

```
...
```

```
preprocessor = Preprocessor()
```

```
preprocessor.fit(X_train)
```

```
X_train = preprocessor.transform(X_train)
```

```
X_test = preprocessor.transform(X_test)
```

Клас `NodeFeaturesExtractor` відповідає за отримання з бази даних повної інформації про кластери, ідентифікатори яких входять до тренувальної або тестової вибірки. Оскільки кількість вузлів може бути значною, цей компонент підтримує механізм обробки даних частинами, що запобігає перевантаженню.

```
chunks = [nodes[i:i + self.max_nodes] for i in range(0, len(nodes), self.max_nodes or len(nodes))]
```

Далі, для кожної частини виконується SQL-вибірка.

```
features = []
```

```
for chunk in tqdm(chunks, desc="Завантаження", leave=False):
```

```
    rows = self.db.select(
        table="node_features",
        where_column="alias",
        where_in=chunk
```

```
)
features.extend(rows)
```

Отримані записи перетворюються у DataFrame, після чого формуються матриця ознак X та масив відповідних міток y , які представлені у вигляді числових індексів.

```
X = df.drop(columns=["label", "alias", "degree", "degree_in", "degree_out"],
errors="ignore")
X.index = df["alias"]

if label_values:
    label_map = { label: i for i, label in enumerate(label_values) }
    y = { alias: label_map.get(label) for alias, label in zip(df["alias"], df["label"]) }
else:
    y = { alias: label for alias, label in zip(df["alias"], df["label"]) }
```

Таким чином, NodeFeaturesExtractor забезпечує ізольований і керований процес завантаження характеристик кластерів.

Для моделей машинного навчання недостатньо даних з таблиці «node_features». Необхідно сформувані додаткові, більш інформативні характеристики – похідні ознаки, що відображають поведінкові та часові патерни кластера.

Це реалізовано в класі Preprocessor (Додаток А), який виконує створення нових ознак на основі співвідношень, середніх значень та діапазонів; нормалізацію значень; обробку пропусків; додавання економічного контексту на основі ринкових даних (BTC/USD); збереження однакової структури ознак для тренувальної та тестової вибірок.

Нижче подано перелік основних похідних ознак, сформованих у процесі попередньої обробки (табл. 2.2).

Таблиця 2.2

Похідні ознаки, сформовані в межах класу Preprocessor

Назва	Опис
1	2
avg_sent	Середній обсяг коштів (у криптовалюти), що відправляється кластером за одну транзакцію
avg_received	Середній обсяг коштів, що надходить на адреси кластера за одну транзакцію

1	2
sent_received_ratio	Співвідношення між сумою відправлених та отриманих коштів
sent_range	Різниця між максимальною та мінімальною сумою відправлення
received_range	Різниця між максимальною та мінімальною сумою отримання
Age	Вік кластера, визначений як різниця між блоком останньої та першої зафіксованої активності
total_transactions_in_age_ratio	Кількість вхідних транзакцій, приведена до віку кластера
total_transactions_out_age_ratio	Кількість вихідних транзакцій, приведена до віку кластера
cluster_size_age_ratio	Відношення розміру кластера (кількості адрес) до його віку
cluster_num_edges_age_ratio	Кількість внутрішніх транзакцій у кластері, нормалізована на час його існування
time_before_first_transaction	Різниця між часом першої вхідної та першої вихідної транзакції
transaction_per_day	Інтенсивність активності: кількість транзакцій за день існування
min_sent_usd, max_sent_usd, total_sent_usd	Еквіваленти показників у доларах з урахуванням історичного курсу BTC
min_received_usd, max_received_usd, total_received_usd	Аналогічні USD-оцінки отриманих сум
avg_sent_usd, avg_received_usd	Середні значення обсягів транзакцій у доларах
avg_tx_value_usd	Середня вартість транзакції (вхідні + вихідні) у доларах

Окрему увагу під час формування ознак приділено визначенню еквівалентів певних метрик у доларах США. Для цього в межах Preprocessor реалізовано механізм обчислення медіанного курсу Bitcoin у період активності

кластера. Функція `calculate_median_price` визначає часовий діапазон між першою та останньою транзакцією кластера, після чого отримує історичні ринкові значення BTC/USD у відповідному інтервалі та обчислює їх медіану. Саме цей показник використовується для конвертації криптовалютних величин у доларовий еквівалент. Такий підхід дозволяє коректно інтерпретувати обсяг транзакцій у загальноприйнятій валюті, незалежно від волатильності курсу Bitcoin у різні періоди часу.

Узагальнюючи, застосування `Preprocessor` забезпечує перехід від базових даних до інформативного та збалансованого набору характеристик, який значно підвищує здатність моделей машинного навчання робити коректні висновки. Створені ознаки дозволяють урахувати поведінкові патерни, часову динаміку та економічний контекст, що в комплексі формує якісну основу для подальшої побудови та оцінювання моделей класифікації кластерів.

Після завершення етапів завантаження даних та їх попередньої обробки було здійснено побудову та навчання моделей машинного навчання, спрямованих на класифікацію кластерів Bitcoin-адрес за їх характеристиками. У межах дослідження розглядалися три моделі: `Random Forest`, `Extra Trees` та `XGBoost`, які є ефективними для класифікації об'єктів на основі числових та поведінкових характеристик.

Кожна з моделей реалізується як окремий модуль та навчається незалежно від інших. Такий підхід забезпечує ізоляцію параметрів, можливість варіювати методики налаштування та дає змогу коректно порівняти результати їхньої роботи.

Також для кожної моделі передбачено можливість автоматичного пошуку оптимальних гіперпараметрів за допомогою методу `GridSearchCV` [31, 32]. Метод перебирає комбінації параметрів, тестуючи їх на різних підмножинах даних та оцінює ефективність за метрикою `f1_macro`, яка є найбільш корисною в контексті нерівномірно збалансованих вибірок.

```
param_grid = {  
    "n_estimators": [100, 200, 300],  
    ...  
}
```

```

grid_search = GridSearchCV(
    estimator=...Classifier(...),
    param_grid=param_grid,
    cv=2,
    scoring="f1_macro",
    verbose=1,
    n_jobs=-1
)

grid_search.fit(X_train, y_train)

```

Підібрані раніше оптимальні параметри для Random Forest, Extra Trees та XGBoost були зафіксовані й використані для навчання у фінальній реалізації.

Модель Random Forest навчається на підготовлених ознаках `X_train` та відповідних мітках `y_train`. Комбінація гіперпараметрів підібрана попередньо вищеповисаним методом.

```

model = RandomForestClassifier(
    random_state=42,
    n_estimators=300,
    max_depth=20,
    min_samples_split=5,
    max_features="log2",
    bootstrap=False,
    class_weight="balanced",
    n_jobs=-1
)

```

Використання `class_weight="balanced"` є важливим для компенсації дисбалансу між класами, що суттєво впливає на якість класифікації рідкісних категорій.

Модель Extra Trees є варіантом ансамблевого методу на основі дерев рішень, але з використанням максимальної випадковості для розбиття вузлів. Це забезпечує більшу швидкість обробки та інколи вищу стійкість на нерівномірних даних.

```

model = ExtraTreesClassifier(
    random_state=42,
    n_estimators=100,
    max_depth=None,
    min_samples_split=5,
    max_features="sqrt",
    class_weight="balanced",
    n_jobs=-1
)

```

Для моделі XGBoost додатково враховано дисбаланс класів шляхом застосування `sample_weight`, що дозволяє приділити більше ваги рідкісним категоріям.

```

sample_weight = y_train.map(lambda c: total / label_counts[c]).astype("float32")

```

Це особливо важливо для задачі класифікації кластерів, де деякі категорії зустрічаються значно рідше.

```
model = XGBClassifier(  
    random_state=42,  
    n_estimators=300,  
    max_depth=6,  
    objective="multi:softprob",  
    eval_metric="mlogloss",  
    tree_method="hist",  
    colsample_bytree=1.0,  
    learning_rate=0.1,  
    min_child_weight=3,  
    subsample=1.0,  
    reg_lambda=1.0,  
    reg_alpha=0.0,  
    n_jobs=-1  
)
```

У результаті здійснено навчання трьох моделей на наборі попередньо оброблених ознак.

Після завершення процесу навчання кожна модель проходить етап оцінювання, спрямований на аналіз якості класифікації та оцінку її роботи на тестовій вибірці, що не брала участі у процесі навчання. На цьому етапі здійснюється прогнозування категорій для набору X_{test} та порівняння їх зі справжніми мітками y_{test} .

Для первинного аналізу використовується вбудована функція `classification_report`, що відображає такі метрики, як precision, recall та F1-score для кожного класу окремо.

```
print(classification_report(y_test, predictions))
```

Також здійснюється підрахунок загальних метрик якості, включно з accuracy, макро- та weighted-варіантами precision, recall і F1-score. Макро-метрики використовуються для рівномірної оцінки ефективності по всіх класах, тоді як зважені метрики забезпечують врахування дисбалансу в розподілі даних [33].

```
accuracy = accuracy_score(y_test, predictions)  
precision_macro = precision_score(y_test, predictions, average="macro")  
recall_macro = recall_score(y_test, predictions, average="macro")  
f1_macro = f1_score(y_test, predictions, average="macro")  
  
precision_weighted = precision_score(y_test, predictions, average="weighted")  
recall_weighted = recall_score(y_test, predictions, average="weighted")  
f1_weighted = f1_score(y_test, predictions, average="weighted")
```

Для детальнішого аналізу результатів будується матриця невідповідностей (confusion matrix), яка демонструє, які саме класи модель частіше плутає між собою. Матриця нормалізується за рядками, що дозволяє оцінити розподіл помилок незалежно від кількості прикладів у кожній категорії. Результат візуалізується у вигляді теплової карти Heatmap та зберігається у відповідному каталозі моделі.

```
cm = confusion_matrix(y_test, predictions)
cm_normalized = cm.astype("float") / cm.sum(axis=1, keepdims=True)

plt.figure(figsize=(10, 7))
sns.heatmap(
    cm_normalized,
    annot=True,
    fmt=".2f",
    cmap="Oranges",
    xticklabels=CATEGORIES,
    yticklabels=CATEGORIES,
    vmin=0,
    vmax=1
)

plt.title("Матриця невідповідностей")
plt.xlabel("Предбачена мітка")
plt.ylabel("Справжня мітка")
plt.tight_layout()

plt.savefig("app/models/.../confusion_matrix.png", dpi=300)
```

Окремо аналізується важливість ознак (feature importance) [34]. Це дозволяє визначити, які характеристики кластерів відіграють ключову роль у процесі класифікації. З метою зручності візуалізації виводиться топ-10 найважливіших ознак, що формують найбільший внесок у рішення моделі.

```
feature_importance = model.feature_importances_
feature_names = X_train.columns

indices = np.argsort(feature_importance)[::-1]

TOP_N = 10
top_indices = indices[:TOP_N]

top_features = feature_names[top_indices]
top_values = feature_importance[top_indices]

plt.figure(figsize=(12, 8))
sns.barplot(
    x=top_values,
    y=top_features,
    orient="h",
    color=sns.color_palette("Oranges_r")[2]
)

plt.title(f"Найважливіші ознаки моделі (Топ-{TOP_N})")
```

```
plt.xlabel("Важливість ознаки")
plt.ylabel("Назва ознаки")
plt.tight_layout()

plt.savefig("app/models/.../feature_importance.png", dpi=300)
```

Після завершення оцінювання модель зберігається у файл для подальшого використання у вебсервісі. Для Random Forest та Extra Trees це здійснюється за допомогою бібліотеки `joblib`.

```
os.makedirs(SAVE_DIR, exist_ok=True)

MODEL_PATH = os.path.join(SAVE_DIR, "...joblib")

joblib.dump(model, MODEL_PATH)
```

У випадку XGBoost використовується власний механізм збереження у форматі `.json`.

```
os.makedirs(SAVE_DIR, exist_ok=True)

MODEL_PATH = os.path.join(SAVE_DIR, "xgboost_model.json")

model.save_model(MODEL_PATH)
```

Це забезпечує можливість використання моделі у подальшому без потреби повторного навчання.

2.3 Аналіз результатів роботи та ефективності моделей класифікації

Після навчання моделей було проведено аналіз їхньої ефективності з метою визначення, наскільки коректно вони здатні класифікувати кластери Bitcoin-адрес. Оцінювання здійснювалося за низкою метрик, однак основним критерієм якості виступає Macro F1-score, оскільки саме вона найбільш повно відображає загальну ефективність моделі в умовах наявного дисбалансу між категоріями [35, 36].

Зведені результати роботи моделей наведено у табл. 2.3. Усі три алгоритми демонструють високий рівень загальної точності, однак значення Ассурасу не є достатньо інформативним у контексті дисбалансу даних, оскільки моделі можуть добре класифікувати масові категорії, одночасно помиляючись на рідкісних. Натомість порівняння за Macro F1-score дозволяє побачити реальну здатність моделей працювати з усіма класами рівномірно.

Порівняння метрик ефективності моделей машинного навчання

Модель	Accuracy	Macro Precision	Macro Recall	Macro F1-score
Random Forest	0.9222	0.7342	0.5538	0.6159
Extra Trees	0.9199	0.7541	0.5864	0.6484
XGBoost	0.9205	0.7053	0.6402	0.6635

Найнижче значення Macro F1-score спостерігається у моделі Random Forest, що свідчить про недостатню ефективність у класифікації рідкісних категорій. Модель Extra Trees показує вищий результат, демонструючи краще узагальнення на нерівномірних даних. Найкращий показник серед усіх моделей отримано у XGBoost – 0.6635, що підтверджує її здатність ефективно балансувати між точністю та повнотою та забезпечувати найвищу якість класифікації в умовах складного, дисбалансного набору даних.

Подальший аналіз якості класифікації здійснено за допомогою матриць невідповідностей, які дозволяють детальніше оцінити характер помилок моделей та визначити категорії, що найчастіше плутаються між собою. Такий підхід є важливим доповненням до узагальнених метрик, оскільки дозволяє встановити структурні закономірності в поведінці моделей.

Модель добре класифікує масові категорії, такі як Individual, Mining та Bridge, що підтверджується високими значеннями на головній діагоналі (рис. 2.2). Водночас спостерігається значна кількість помилок при класифікації рідкісних категорій, зокрема Mixer, Faucet, Ponzi та інші, які часто помилково відносяться до класів із подібними транзакційними патернами. Окремо варто зазначити, що категорія Bet класифікується майже ідеально, оскільки має дуже вузьку та специфічну поведінкову структуру.

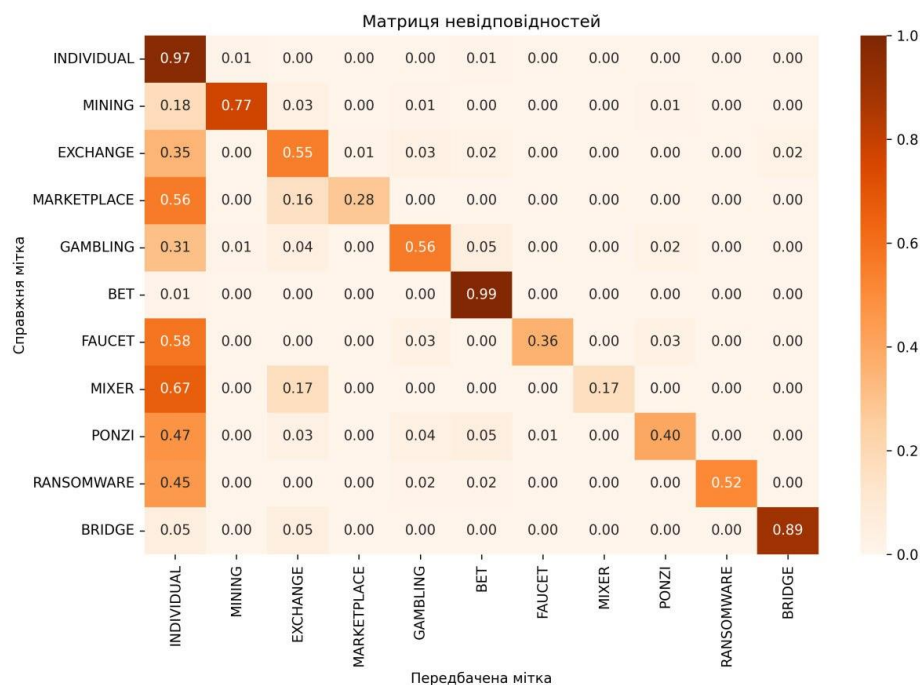


Рисунок 2.2 – Матриця невідповідностей моделі Random Forest

У порівнянні з Random Forest, модель Extra Trees демонструє дещо кращу розмежованість між рідкісними класами (рис. 2.3). Покращується класифікація категорій Faucet, Marketplace, частково – Mixer. Водночас складності залишаються для Ransomware та Ponzi, які мають схильність до помилкової ідентифікації як більш чисельні категорії.

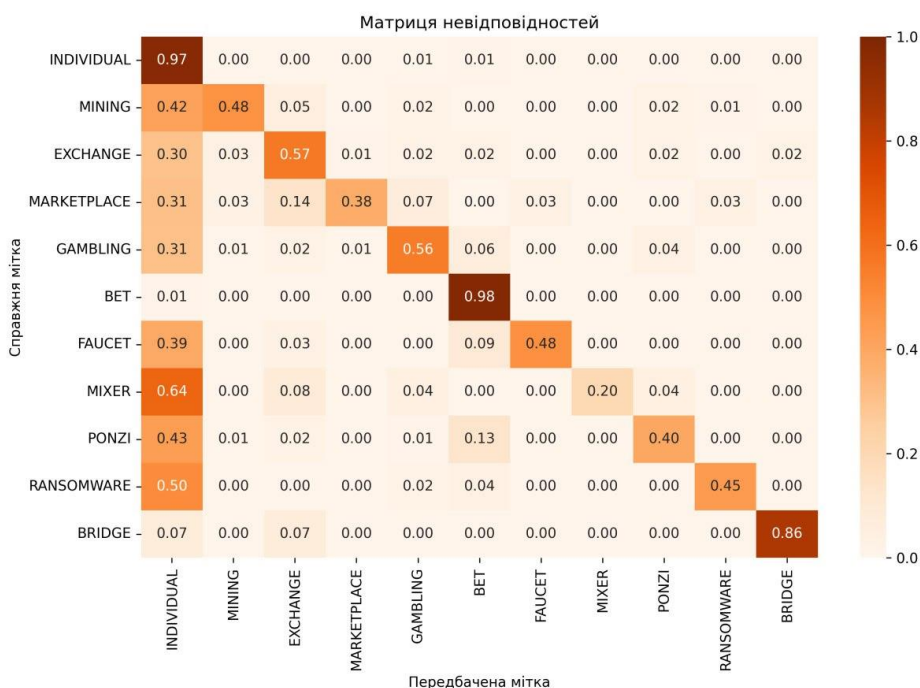


Рисунок 2.3 – Матриця невідповідностей моделі Extra Trees

Модель XGBoost демонструє найкраще розділення класів. Значення на діагоналі для більшості категорій є найвищими серед усіх моделей, що підтверджує здатність алгоритму ефективно працювати навіть із рідкісними класами (рис. 2.4). Особливо помітне покращення у категоріях Gambling, Ponzi, та Ransomware. Це свідчить про те, що XGBoost краще враховує взаємозв'язки між ознаками та є більш стійким до шуму і дисбалансу. Помилки все ще присутні, але їх кількість та інтенсивність суттєво нижчі, ніж у двох попередніх моделей.

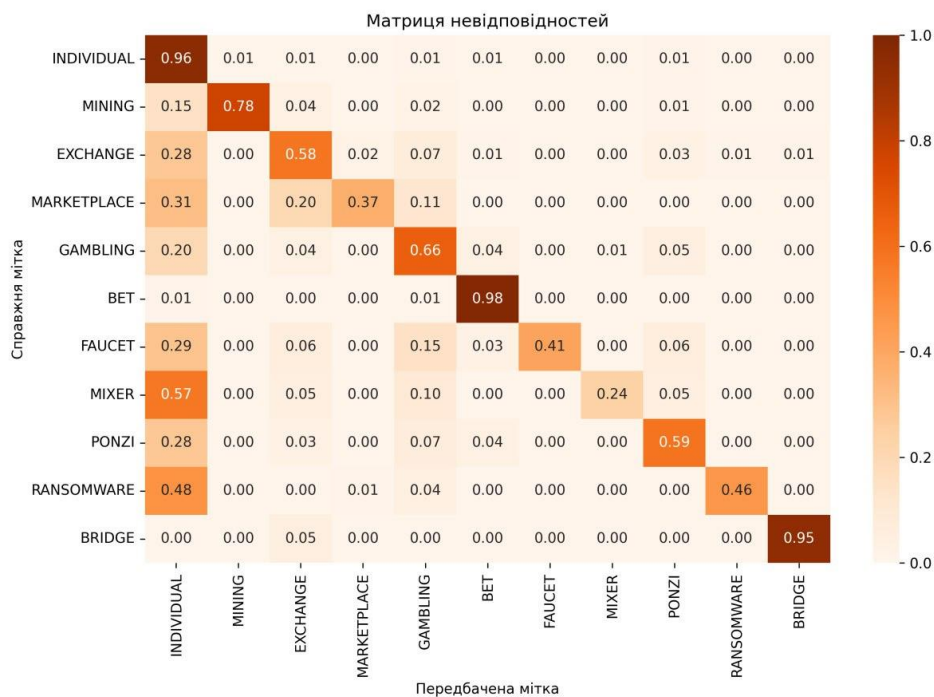


Рисунок 2.4 – Матриця невідповідностей моделі XGBoost

На заключному етапі оцінювання було проаналізовано важливість ознак моделі XGBoost, яка продемонструвала найвищу ефективність серед розглянутих алгоритмів. Такий аналіз дозволяє визначити, які характеристики кластерів найбільше впливають на прийняття рішень під час класифікації.

На діаграмі наведено десять найважливіших ознак, що формують основу роботи моделі (рис. 2.5). Найбільший внесок мають показники, пов'язані з фінансовою активністю, зокрема `avg_received` та `max_received_usd`, що підкреслює значущість обсягів отриманих транзакцій для віднесення кластера до певної категорії. Часові характеристики `first_transaction_in`,

first_transaction_out також є визначальними, відображаючи вплив часової динаміки на поведінку різних типів кластерів.

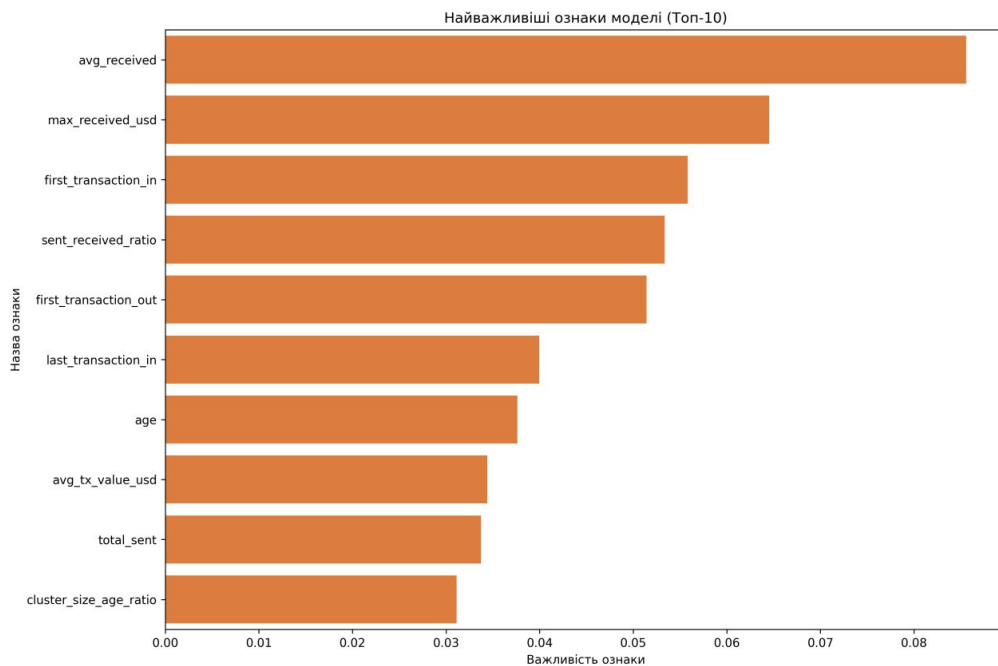


Рисунок 2.5 – Топ-10 найважливіших ознак моделі XGBoost

У сукупності ці ознаки формують інформативну основу, завдяки якій XGBoost забезпечує найкращу якість класифікації серед досліджених моделей.

2.4 Алгоритми кластеризації та прогнозування категорії кластерів

Після побудови та навчання моделей класифікації постає завдання їхньої інтеграції в роботу вебсервісу, забезпечивши можливість автоматичного аналізу Bitcoin-адрес. У рамках цього етапу реалізована кластеризація на бекенд-частині, тобто Symfony, логіка формування ознак для подальшої передачі до моделей машинного навчання, а також взаємодія між модулями сервісу в процесі створення звіту про адресу.

Одним із основних етапів є формування кластера адрес та обчислення його характеристик. Це здійснюється шляхом виклику методу createClusterFeatures сервісу ClusterService (Додаток Б).

Перш за все, необхідно отримати вихідні дані про початкову адресу. У реальних умовах подібна інформація традиційно береться з API блокчейн-

експлорерів (наприклад, Blockchair, Blockchain.com і т.д.), проте ці сервіси працюють на платній основі та мають суттєві обмеження [37].

З огляду на це, в межах кваліфікаційної роботи було прийнято рішення сформувати синтетичний набір даних, який моделює властивості та поведінку реальних Bitcoin-адрес. Основою для його побудови слугував формат відповіді, що використовується реальними блокчейн-експлорерами (зокрема Blockchain.com), що дозволило відтворити структуру транзакцій, зв'язків та атрибутів, характерних для справжніх API-запитів [38].

Синтетичні дані охоплюють різні категорії кластерів та містять повну інформацію про адреси та їх транзакційну активність, необхідну для демонстрації роботи алгоритму кластеризації та подальшого визначення ознак.

Збереження реалізовано у вигляді JSON-файлів у спеціальній директорії, а їхнє завантаження виконується за допомогою сервісу BlockchainDataService. Його функція `fetchSingleAddress` забезпечує отримання первинної інформації про адресу та слугує точкою входу у процес кластеризації. Саме вона викликається на початку роботи методу `createClusterFeatures`.

Після отримання даних про адресу наступним кроком є формування самого кластера – тобто групи адрес, які з високою ймовірністю належать одному власнику. За це відповідає метод `buildCluster`, який ініціює процес обходу та розширення кластера відповідно до заданих евристичних правил. Він реалізує послідовне опрацювання адрес, використовуючи структуру черги. Кожна адреса обробляється один раз, після чого до черги додаються нові, знайдені за допомогою евристик.

Евристики застосовуються в межах методу `applyHeuristics` сервісу `HeuristicsService`. У ньому відбувається поетапний аналіз кожної транзакції, пов'язаної з адресою.

Передусім виконується перевірка чи адреса справді є одним із відправників у транзакції. Якщо ж вона зустрічається лише у вихідних даних, така транзакція не несе інформаційної цінності для кластеризації в контексті досліджених евристик і пропускається. Для всіх релевантних транзакцій

застосовуються правила, спрямовані на виявлення потенційно пов'язаних адрес.

Першою з евристик є Multi-Input Heuristic, яка ґрунтується на припущенні, що якщо кілька різних адрес одночасно використовуються як вхідні в одній транзакції, то вони, з великою ймовірністю, контролюються одним власником. У такому випадку всі ці адреси додаються до набору кандидатів для включення в кластер.

```
// H1: Multi-Input Heuristic (Common-Input-Ownership Heuristic/Multi-Input Address Clustering)
if (count($transaction['inputs']) > 1) {
    foreach ($transaction['inputs'] as $input) {
        $inputAddress = $input['prev_out']['addr'] ?? null;

        if ($inputAddress !== null && $inputAddress !== $address['address']) {
            $addresses[] = $inputAddress;
        }
    }
}
```

Наступною застосовується евристика Change Address Heuristic, що базується на типовій поведінці гаманців Bitcoin. У більшості звичайних транзакцій один із виходів повертається відправнику як решта. Якщо транзакція містить два виходи, і один з них не був використаний раніше, то з високою ймовірністю це й є адреса «решти». Така адреса додається до списку пов'язаних.

```
// H2: Change Address Heuristic (Address Reuse)
if (count($transaction['out']) === 2) {
    $newOutputs = array_values(array_filter($outputs, function ($output) use ($reusedOutputs) {
        return !in_array($output, $reusedOutputs);
    }));

    if (count($newOutputs) === 1) {
        $addresses[] = $newOutputs[0];

        $reusedOutputs = array_unique(array_merge($reusedOutputs, $outputs));

        continue;
    }
}
```

Третя реалізована евристика Force Merge of Inputs Heuristic – варіація підходу для випадків, коли транзакція містить багато входів та два виходи. Вона враховує співвідношення сум UTXO, визначає, який із виходів є

платіжним, а який – поверненням решти, й додає відповідну адресу до можливих елементів кластера.

```
// H3: Force Merge of Inputs Heuristic
if (count($transaction['inputs']) > 1 && count($transaction['out']) === 2) {
    $inputs = [];

    foreach ($transaction['inputs'] as $input) {
        $inputValue = $input['prev_out']['value'] ?? null;

        if ($inputValue !== null) {
            $inputs[] = $inputValue;
        }
    }

    $total = array_sum($inputs);
    $min = min($inputs);

    $firstOutput = $transaction['out'][0];
    $secondOutput = $transaction['out'][1];

    $payment = ($firstOutput['value'] >= $secondOutput['value']) ? $firstOutput :
$secondOutput;
    $change = ($firstOutput['value'] < $secondOutput['value']) ? $firstOutput :
$secondOutput;

    if (
        $change['addr'] !== null
        && $change['addr'] !== $address['address']
        && !in_array($change['addr'], $reusedOutputs)
        && (($total - $min) < $payment['value'])
    ) {
        $addresses[] = $change['addr'];
    }
}
```

У підсумку метод `applyHeuristics` повертає набір адрес, для яких існують аргументовані підстави вважати їх пов'язаними з поточною адресою. Після завершення роботи `buildCluster` формується повний набір адрес, що належать одному кластеру, і саме він передається на наступний ключовий етап – обчислення характеристик кластера, який реалізовано в методі `calculateClusterFeatures`.

Саме цей метод завершує логіку `createClusterFeatures`, перетворюючи множину адрес та їхні транзакційні дані на структурований набір ознак, що потім надсилається до моделей машинного навчання. Якщо попередні етапи відповідали за визначення складу кластера, то `calculateClusterFeatures` відповідає за побудову його «портрета» – опису активності, руху коштів, структури внутрішнього графа та часових характеристик. Саме на основі цих

показників моделі Random Forest, Extra Trees та XGBoost здійснюють класифікацію.

Робота методу починається з ініціалізації масиву features, де задаються всі необхідні показники: кількість вхідних та вихідних транзакцій, сумарні обсяги отриманих та відправлених коштів, мінімальні та максимальні значення, часові межі активності, а також характеристики внутрішньої структури. На цьому ж етапі формується окремий перелік адрес, що входять до кластера окрім кореневої, щоб відобразити їх у звіті.

Далі відбувається покроковий аналіз транзакцій кожної адреси. Визначається, чи є вона вхідною або вихідною відносно кластера. Це здійснюється шляхом перевірки входів та виходів: якщо серед входів присутні адреси, що не належать кластеру, транзакція вважається вхідною; якщо ж серед виходів є адреси поза кластером, транзакція класифікується як вихідна. На основі цього поступово оновлюються всі агреговані характеристики: кількість транзакцій відповідного типу, мінімальні й максимальні обсяги, загальні суми, а також часові межі активності, що беруться з номерів блоків і часових міток транзакцій.

Разом з обчисленням цих характеристик визначаються й внутрішні взаємозв'язки адрес. Якщо в одній транзакції одночасно зустрічаються вхідні та вихідні адреси, що належать кластеру, це вказує на внутрішній зв'язок між ними, і відповідний лічильник збільшується. Окрім цього формується допоміжний граф, у якому кожна адреса є вузлом, а зв'язки між адресами додаються як ребра графа. Такий граф відображає структуру внутрішніх транзакційних зв'язків кластера й використовується на завершальному етапі для визначення компонентів зв'язності. Метод `findConnectedComponents` виконує пошук груп адрес, які пов'язані між собою через транзакції, формуючи окремі компоненти всередині кластера [39]. У кожній знайденій компоненті підраховується її розмір, а також кількість таких компонент.

Після формування кластера та обчислення його ознак заключним етапом є інтеграція отриманих характеристик із модулем машинного навчання. Це

здійснюється в процесі створення звіту, де кожен кластер оцінюється за трьома, попередньо навченими моделями. На цьому етапі формується повноцінний об'єкт Report, до якого додається створений кластер та всі адреси, що до нього входять. Після цього відбувається передавання набору ознак до модуля MLService, який відповідає за звернення до Python-сервісу, реалізованого на основі FastAPI, та отримання прогнозів.

Параметри передаються через виклик методу predict. Він реалізує HTTP-запит до контейнера python, що працює за адресою `http://python:8000/python/predict`. Дані передаються у форматі JSON, після чого Python-сервіс виконує обробку: трансформує отримані ознаки за допомогою збереженого Preprocessor, підвантажує відповідні моделі (Random Forest, Extra Trees та XGBoost) та обчислює результати для кожної з них. Відповідь повертається у вигляді масиву з прогнозованою категорією, ймовірністю та оцінкою ризику.

```
public function predict(array $features): ?array
{
    $features['alias'] = 0;

    unset($features['first_transaction_in_time'],
    $features['last_transaction_in_time'], $features['first_transaction_out_time'],
    $features['last_transaction_out_time'], $features['addresses']);

    try {
        $response = $this->httpClient->request('POST',
        'http://python:8000/python/predict', [
            'json' => $features
        ]);

        return $response->toArray();
    } catch (Throwable $exception) {
        throw new RuntimeException(
            'Під час передачі параметрів для моделей виникла помилка: ' . $exception-
            >getMessage(),
            Response::HTTP_BAD_REQUEST
        );
    }
}
```

Отримані передбачення записуються у звіт через ReportPredictionService, який створює окремі записи для кожної моделі, включно з категорією, ймовірністю та рівнем ризику. Таким чином формується комплексний аналітичний опис кластера, який пізніше буде показано користувачеві у вигляді сторінки звіту.

Оскільки процес створення звіту охоплює кілька самостійних сутностей, важливо забезпечити їхнє цілісне та структуроване зберігання. Для цього в межах проекту реалізовано модель даних із використанням MySQL, що складається з чотирьох взаємопов'язаних таблиць (рис. 2.6). Вони фіксують усі результати обробки: сам звіт, сформований кластер, адреси, які до нього входять, та оцінки моделей. Така структура дозволяє не лише відтворювати звіт у будь-який момент, але й забезпечує зручність подальшого дослідження, агрегування статистики та можливого розширення функціоналу сервісу.

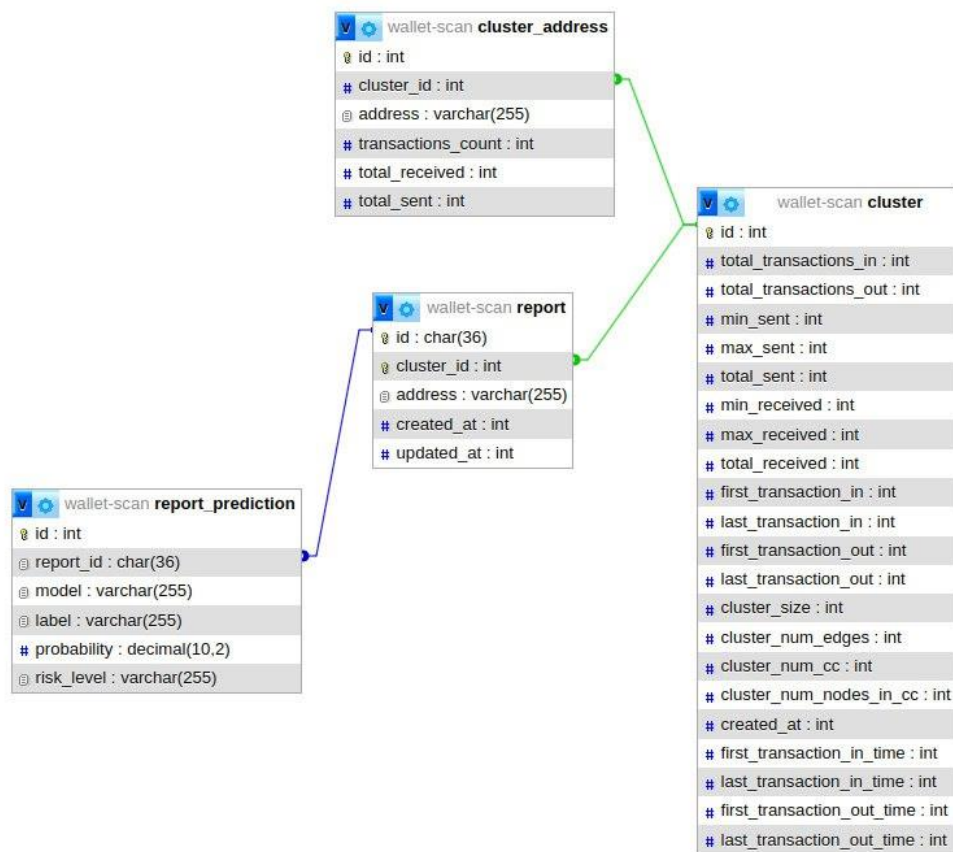


Рисунок 2.6 – Структура бази даних MySQL

Наведені алгоритми та структура зберігання даних формують цілісний процес аналітики: від побудови кластера і розрахунку його характеристик до отримання прогнозу моделі та збереження результатів у зручному для подальшого використання вигляді.

Висновки до другого розділу

У другому розділі було проведено комплексний аналіз набору даних, що використовується для класифікації кластерів Bitcoin-адрес, а також здійснено практичну реалізацію алгоритмів машинного навчання та логіки кластеризації у вебсервісі. Розгляд структури наукового датасету та його маркованої підмножини дав змогу визначити необхідний набір вхідних ознак, оцінити їх властивості та забезпечити коректне формування навчальних вибірок для побудови моделей. Особливу увагу приділено питанню дисбалансу даних, який істотно впливає на якість прогнозування та зумовлює необхідність використання метрик, стійких до нерівномірного розподілу категорій.

Практичний етап реалізації алгоритмів машинного навчання охопив завантаження та попередню обробку даних, формування додаткових похідних характеристик, налаштування гіперпараметрів та навчання трьох моделей класифікації: Random Forest, Extra Trees та XGBoost. Аналіз ефективності показав, що всі моделі демонструють високий рівень загальної точності, однак найкращий результат за основною метрикою Macro F1-score забезпечує модель XGBoost. Це свідчить про її здатність краще працювати з рідкісними категоріями та враховувати складні взаємозв'язки між ознаками кластерів.

Також у межах вебсервісу було реалізовано повний цикл опрацювання Bitcoin-адреси – від завантаження даних і побудови кластера до обчислення його характеристик та отримання прогнозів моделей машинного навчання. Розроблений алгоритм кластеризації, що базується на евристичних правилах, дозволяє коректно формувати групу пов'язаних адрес і обчислювати необхідні транзакційні, структурні та часові ознаки.

Отримані ознаки інтегруються з модулем машинного навчання, який взаємодіє з Python-сервісом для обчислення прогнозів. Результати прогнозування зберігаються у базі даних MySQL у вигляді структурованого звіту, що містить характеристики кластера, перелік адрес та оцінки трьох моделей. Такий підхід забезпечує відтворюваність результатів, можливість подальшої аналітики та розширення функціоналу системи.

РОЗДІЛ 3 ОПИС ПРОГРАМНОГО ПРОДУКТУ ТА ПРИНЦИПІВ ЙОГО ФУНКЦІОНУВАННЯ

3.1 Архітектура системи та взаємодія її компонентів

Для наочного відображення структури розробленого вебсервісу та принципів взаємодії його складових було побудовано діаграму розгортання (рис. 3.1). Вона демонструє, на яких вузлах розміщується система, які компоненти виконуються на кожному з них, а також які протоколи та механізми використовуються для обміну даними між окремими частинами програмного комплексу [40].

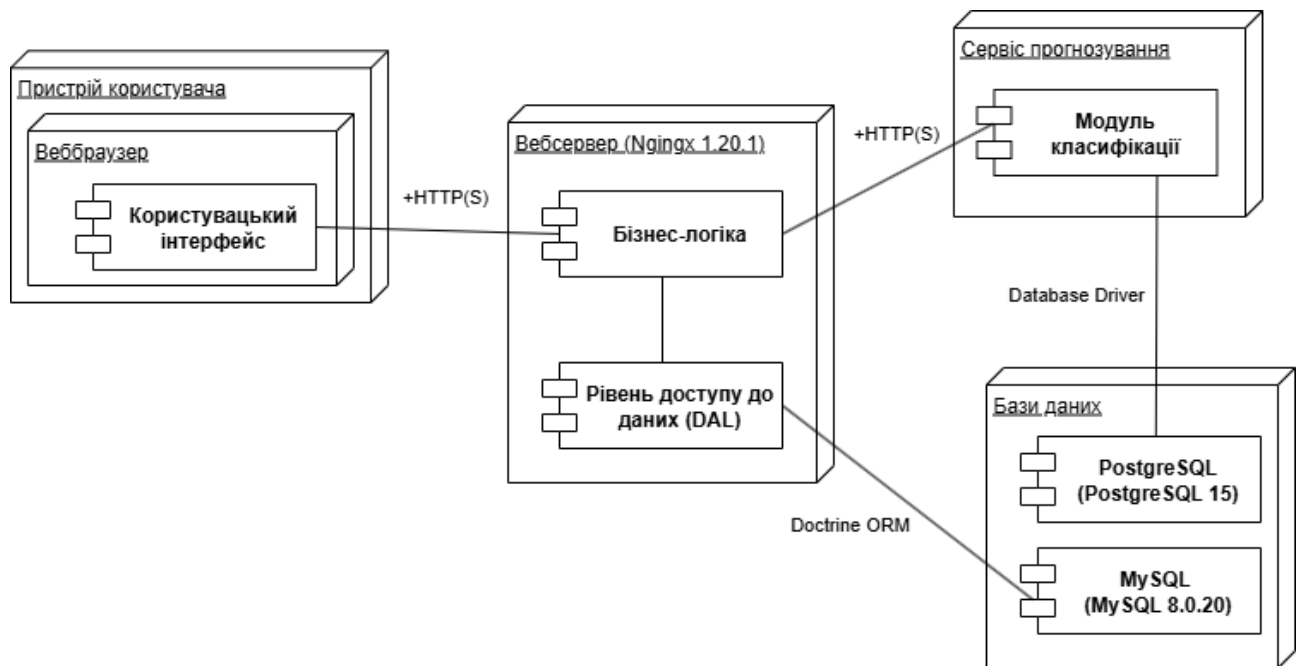


Рисунок 3.1 – Діаграма розгортання

Вузол «Пристрій користувача», включає в себе вузол «Веббраузер», у якому виконується клієнтська частина системи. Через HTTP(S)-з'єднання браузер взаємодіє з вебсервером, отримує згенерований звіт, відображає результати класифікації та формує запити до backend-частини.

Центральним елементом системи є вузол вебсервера, де розгорнута бізнес-логіка. Саме тут виконуються операції з кластеризації адрес, обчислення характеристик, формування структур даних та підготовка інформації для відправлення до модуля класифікації. На цьому ж вузлі розміщується рівень доступу до даних, який забезпечує роботу з однією з баз даних.

Окремим вузлом представлено сервіс прогнозування, що відповідає за застосування моделей машинного навчання. Він отримує сформований набір ознак через HTTP(S)-запит та повертає прогнозовану категорію, ймовірність та ризикову оцінку. Усередині цього сервісу функціонує модуль класифікації, який завантажує попередньо натреновані моделі та застосовує їх до переданих даних. Така ізоляція в окремому сервісі спрощує масштабування, оновлення та тестування алгоритмів машинного навчання.

Третім важливим елементом є вузол баз даних. Перша база використовується для зберігання звітів, кластерів та адрес – взаємодія відбувається через ORM-рівень. Друга база застосовувалася на етапі побудови моделей машинного навчання. Вона використовувалась для формування вибірок, тренування моделей та оцінювання їх ефективності; її дані не беруть участі у роботі користувацької частини вебсервісу.

Окрім описаної структури, важливо відзначити, що система розгортається у контейнеризованому середовищі. Завдяки використанню Docker та Docker Compose процес інсталяції є повністю автоматизованим і не потребує ручного налаштування. Для розгортання вебсервісу достатньо виконати кілька базових кроків: клонувати репозиторій з програмним кодом та запустити інфраструктуру командою `./recreate_docker.sh`, також при роботі на бекенді, варто час від часу виконувати команду `./clear_cache.sh`, що дозволяє очищати кеш.

3.2 Структура інтерфейсу та порядок роботи з вебсервісом

Даний підрозділ демонструватиме користувацький інтерфейс вебсервісу та логіку взаємодії з ним. Наведені скриншоти ілюструють основні етапи роботи – від доступу до головної сторінки до отримання сформованого звіту. Такий опис дозволяє продемонструвати практичне застосування реалізованого функціоналу та показати, як користувач взаємодіє з основними компонентами системи.

Розпочавши роботу з вебсайтом, користувача зустрічає головна сторінка на якій розміщено центральний елемент інтерфейсу – поле для введення адреси криптогаманця та кнопка запуску аналізу (рис. 3.2). Тут же подано короткий опис сервісу та призначення інструмента. Нижче розташовані інформаційні блоки, які презентують основні переваги вебсервісу: автоматичну кластеризацію адрес, розрахунок транзакційних та поведінкових ознак, використання моделей машинного навчання для оцінки ризику, а також загальну простоту роботи з інструментом. Таким чином, вже на стартовому екрані користувач отримує загальне уявлення про те, що це за сервіс і які можливості він має.

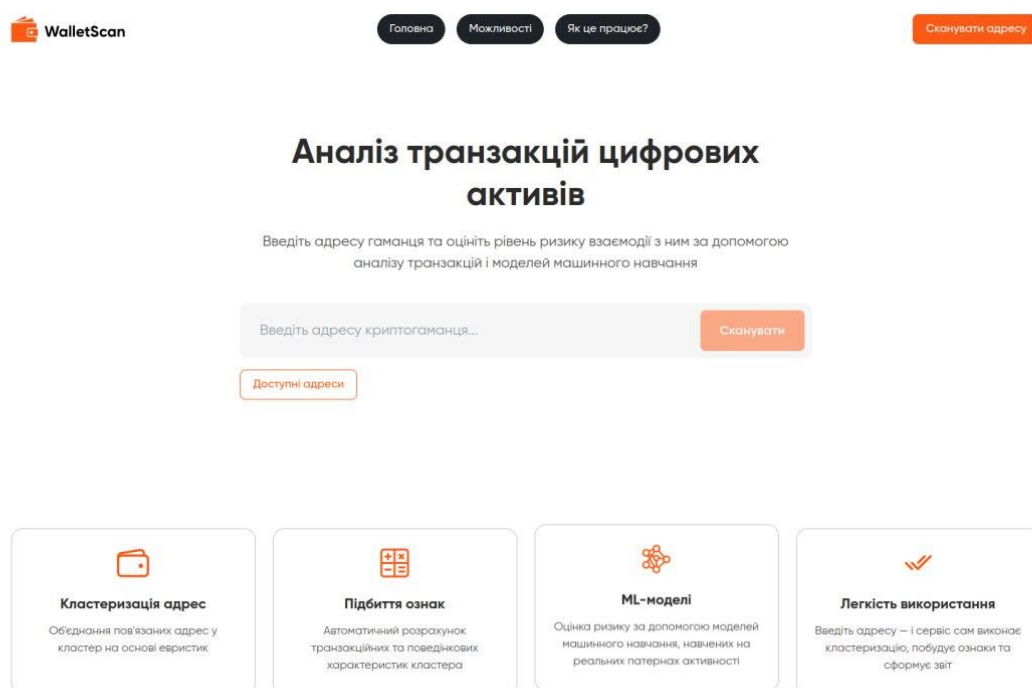


Рисунок 3.2 – Головна сторінка

Нижче, користувач бачить блок «Як це працює?», у якому покроково пояснюється логіка роботи сервісу (рис. 3.3). На відміну від попередніх інформаційних блоків, цей розділ демонструє послідовність дій, які слід виконати щоб отримати звіт. Такий покроковий опис допомагає користувачу швидко зрозуміти принцип роботи вебсервісу та очікуваний результат взаємодії.

Як це працює?



Рисунок 3.3 – Покрокове пояснення роботи сервісу

Останнім елементом головної сторінки є блок визначених категорій ризику, який демонструє класифікацію, що використовується сервісом для оцінювання поведінки кластера (рис. 3.4). У цьому блоці категорії згруповано за рівнем потенційної небезпеки: від низького ризику взаємодії, характерного для звичайної користувацької або біржової активності, до високого ризику, що включає такі типи поведінки, як міксери, фінансові шахрайські схеми або діяльність, пов'язана зі шкідливим програмним забезпеченням. Кожна категорія супроводжується коротким поясненням, що дозволяє користувачу швидко зрозуміти природу відповідної групи.

Які категорії ризику ми визначаємо



Рисунок 3.4 – Категорії ризику

Оскільки вебсервіс працює на основі синтезованих даних, створених спеціально для демонстрації алгоритмів кластеризації та класифікації, на головній сторінці також передбачено можливість швидко обрати одну з

доступних тестових адрес (рис. 3.5). Після натискання кнопки «Доступні адреси» відкривається модальне вікно, у якому зібрано приклади гаманців, що належать до різних категорій ризику. Кожна група містить декілька синтетичних адрес, які можна скопіювати та використати для запуску аналізу.

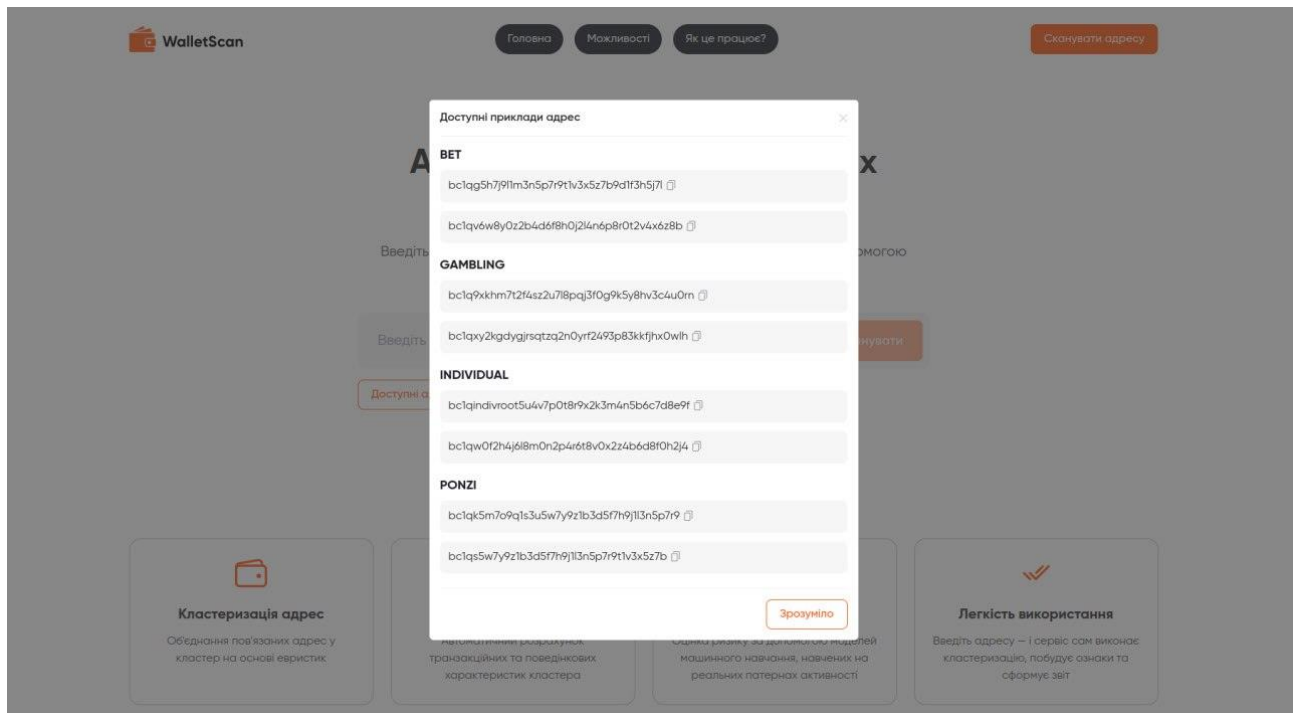


Рисунок 3.5 – Модальне вікно з доступними адресами

Після вибору однієї з доступних тестових адрес у модальному вікні, користувач сканує її. У відповідь вебсервіс формує звіт, у якому подано результати кластеризації та обчислені транзакційні характеристики (рис. 3.6). А також у межах демонстраційної версії одночасно відображаються результати трьох моделей класифікації, щоб показати різницю в їхніх прогнозах. Як було встановлено раніше, найкращу точність продемонструвала модель XGBoost, тому саме її прогноз підсвічується як основний.

У верхній частині сторінки відображається введена адреса, загальний рівень ризику та результати роботи моделей. Для кожної з них показано передбачену категорію та ймовірність, із якою вона була визначена. Нижче наведені основні показники кластера: сумарна кількість отриманих та відправлених коштів, кількість адрес у кластері, кількість транзакцій, а також часові межі активності.

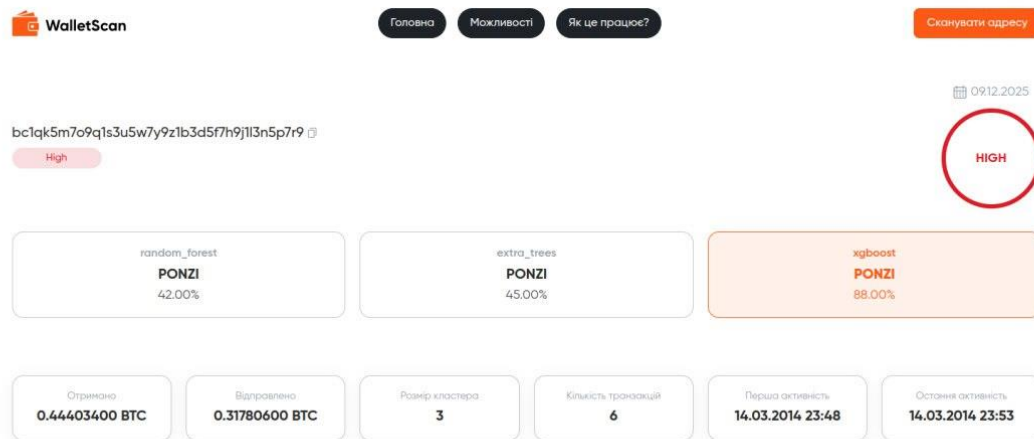


Рисунок 3.6 – Сторінка звіту про адресу

Нижче на сторінці звіту подано розділ «Детальна інформація про кластер», у якому зібрані всі обчислені характеристики, отримані під час аналізу вибраної адреси (рис. 3.7). Цей блок розділений на секції, що дозволяє зручно переглядати основні параметри кластера.

У першій частині наведено загальні характеристики, серед яких: розмір кластера (кількість пов'язаних адрес), загальна кількість транзакцій, а також кількість вхідних та вихідних операцій. Додатково відображено показники внутрішньої структури – число транзакцій між адресами кластера, кількість компонент зв'язності та кількість вузлів у найбільшій компоненті.

Наступні секції містять детальні дані про вхідні та вихідні транзакції кластера. Для кожного напрямку наведено мінімальні, максимальні та сумарні обсяги коштів, а також часові межі активності – моменти першої та останньої зафіксованої операції. Ці показники дозволяють користувачу побачити фінансовий профіль кластера, інтенсивність взаємодій і характер руху коштів.

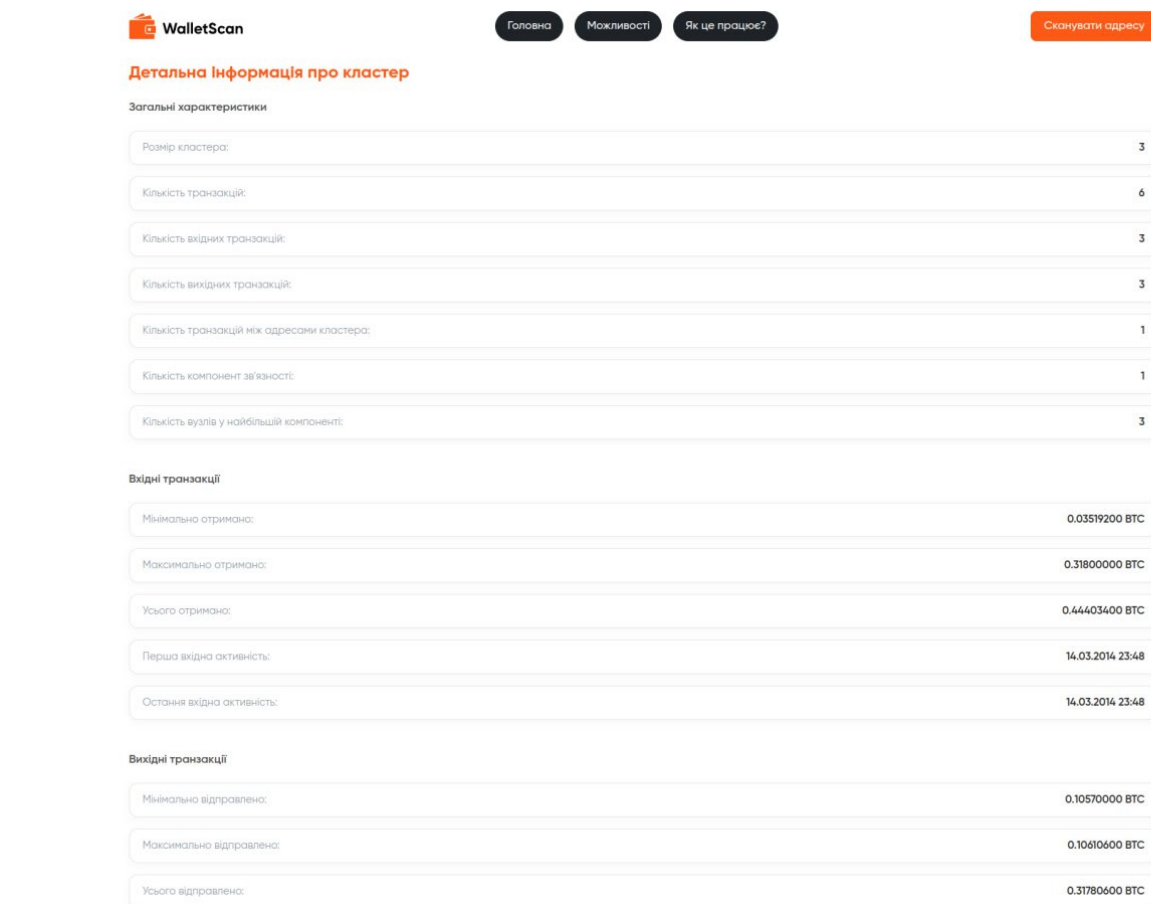


Рисунок 3.7 – Детальна інформація про кластер

Останнім елементом сторінки звіту є узагальнювальна таблиця, що містить інформацію про всі адреси, пов'язані з відповідним кластером (рис. 3.8). Для кожної адреси наведено кількість транзакцій, сумарний обсяг отриманих та відправлених коштів, що дозволяє швидко оцінити роль окремих учасників у структурі кластера.

Інформація про пов'язані адреси кластера

Адреса	Кількість транзакцій	Усього отримано	Усього відправлено
bc1qe8g0l2k4m6o8q0s2u4w6y8z0a2c4e6g8l0k2	1	0.09084200 BTC	0.00000000 BTC
bc1qt0u2w4y6z8a0c2e4g6i8k0m2o4q6s8u0w2y4	1	0.03519200 BTC	0.00000000 BTC

Завантажити PDF

© 2025 WalletScan
Державний університет «Житомирська політехніка»

Рисунок 3.8 – Таблиця пов'язаних адрес кластера та кнопка завантаження звіту

Під таблицею розміщено кнопку завантаження PDF-версії звіту, що забезпечує можливість зберегти всю згенеровану інформацію у зручному

форматі для подальшого використання, друку або додавання до документації (рис. 3.9). Генерація PDF відбувається автоматично, включаючи всі показники, результати класифікації та детальні дані кластера.

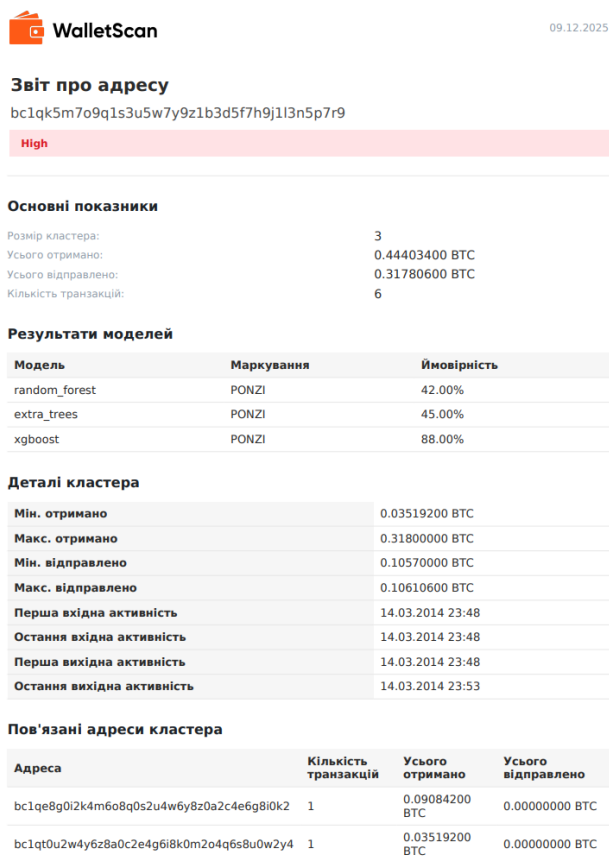


Рисунок 3.9 – Звіт у PDF-форматі

Слід зазначити, що вебсервіс є повністю адаптивним, кожна сторінка з легкістю може бути переглянута на мобільних пристроях. Це забезпечує користувачам зручний доступ до всіх функцій додатку, незалежно від типу або розміру їхнього екрану, що значно покращує загальний користувацький досвід.

Адаптивне відображення головної сторінки вебсервісу та модального вікна вибору пропонованих адрес підтверджує коректне масштабування інтерфейсу та збереження зручного розташування елементів введення, кнопок і списків. Це дозволяє користувачу виконувати основні дії без втрати функціональності під час зміни параметрів екрану (рис. 3.10).

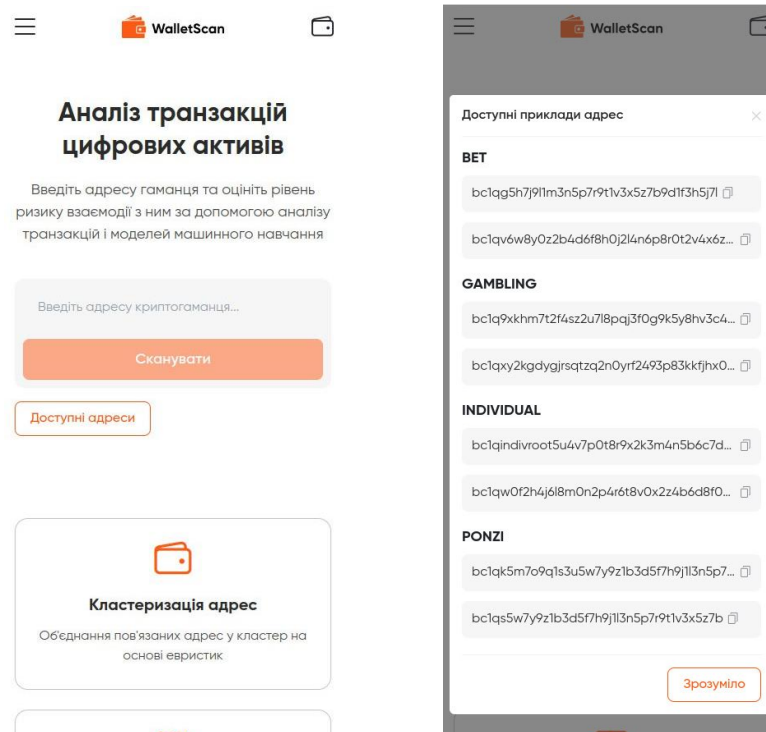


Рисунок 3.10 – Адаптивність вебсервісу

Правильне відображення адаптивного інтерфейсу також простежується на прикладі блоку з категоріями ризику та навігаційного меню, де списки категорій автоматично перебудовуються під доступну ширину екрана. Така організація інтерфейсу забезпечує читабельність назв і збереження логічної ієрархії інформації без горизонтального прокручування (рис. 3.11).

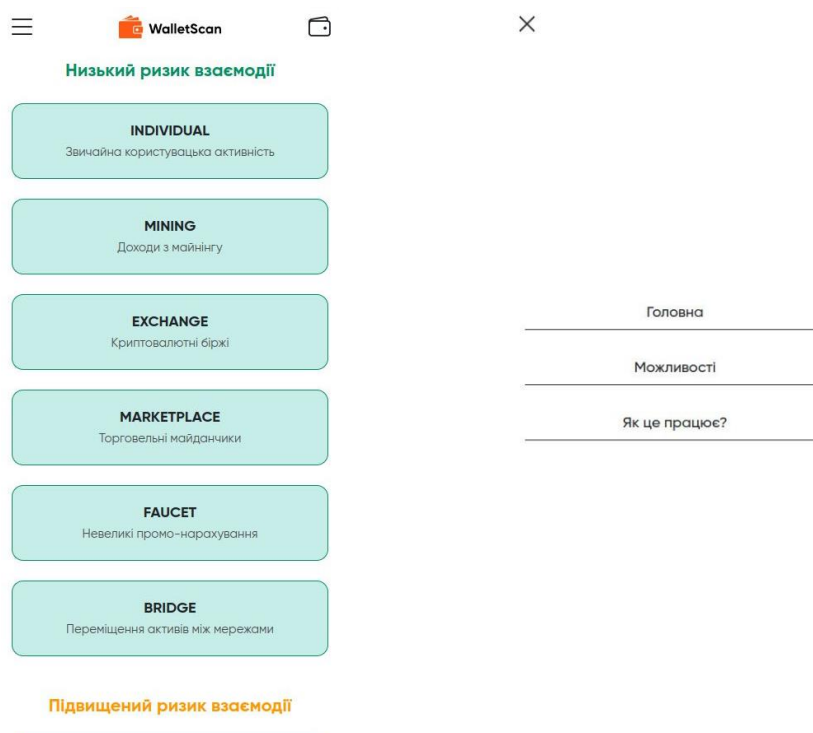


Рисунок 3.11 – Адаптивність вебсервісу

Адаптивне відображення сторінки звіту з результатами аналізу демонструє коректну трансформацію блоків з рівнем ризику, прогнозами моделей та інформацією про пов'язані адреси у вертикальне компонування. Це дозволяє зручно переглядати аналітичні дані на обмеженій площі екрану (рис. 3.12).

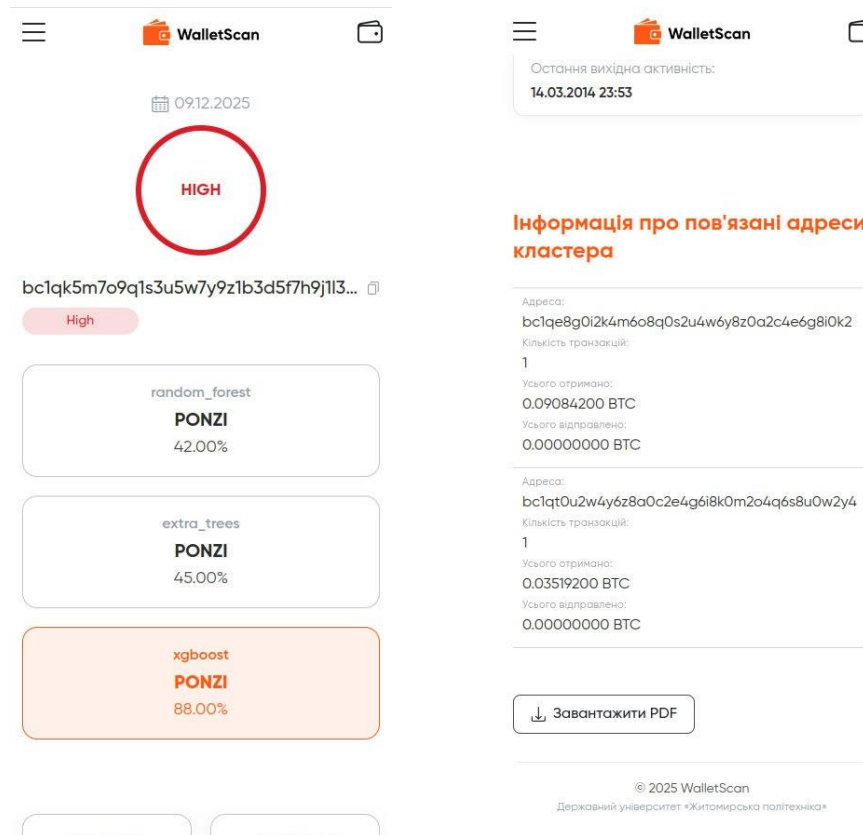


Рисунок 3.12 – Адаптивність вебсервісу

Отже, інтерфейс вебсервісу забезпечує послідовний та зрозумілий порядок взаємодії користувача з системою на всіх етапах роботи. Поєднання продуманої структури сторінок і адаптивного відображення елементів дозволяє ефективно використовувати сервіс на різних типах пристроїв.

Висновки до третього розділу

У третьому розділі було розглянуто програмну реалізацію розробленого вебсервісу. Побудована діаграма розгортання продемонструвала взаємодію основних функціональних компонентів: клієнтського інтерфейсу, серверної частини, сервісу класифікації та баз даних. Така структура забезпечує логічне

розмежування відповідальностей між модулями, спрощує масштабування та підвищує гнучкість системи.

Також було охарактеризовано структуру інтерфейсу й порядок взаємодії користувача з вебсервісом. Покроково описано основні екрани – від головної сторінки до сторінки згенерованого звіту, що містить результати кластеризації, детальні транзакційні характеристики та прогноз категорії ризику.

Таким чином, описано як технічну організацію роботи сервісу, так і практичний досвід користувача під час взаємодії з ним.

ВИСНОВКИ

На сьогоднішній день, коли технології цифрових активів швидко розвиваються, а обсяги транзакцій зростають, все більшої актуальності набувають інструменти для автоматизованого аналізу та оцінювання ризику транзакційної активності.

У ході написання кваліфікаційної роботи було, перш за все, проведено аналіз предметної області та окреслено актуальні проблеми у сфері аналітики транзакцій цифрових активів, описано мету створення розробленого вебсервісу та визначено перелік задач, що необхідно було виконати під час його проектування та реалізації. Детально розглянуто існуючі програмні продукти, визначено їхні сильні та слабкі сторони, що дозволило сформулювати чіткі вимоги до майбутнього рішення та бачення його функціональних можливостей.

Одним із головних етапів дослідження стала розробка алгоритмів кластеризації адрес та побудова моделей машинного навчання для прогнозування категорії кластерів. На основі наукового датасету було виконано формування вибірок, побудовано та налаштовано три моделі – Random Forest, Extra Trees та XGBoost. У ході оцінювання їхньої ефективності встановлено, що найвищу якість класифікації відповідно до метрики Macro F1-score демонструє модель XGBoost, що робить її найбільш придатною для застосування у вебсервісі. Ретельно опрацьовано й механізми формування ознак, включно з похідними характеристиками, які відображають поведінкові та часові особливості кластерів.

Подальша робота була зосереджена на тому, щоб інтегрувати отримані аналітичні рішення у вебсервіс та забезпечити їхнє практичне застосування. Реалізовано повний цикл опрацювання адреси – від формування кластера із використанням евристичних правил до обчислення всіх необхідних характеристик та передачі їх для класифікації. Алгоритми кластеризації та прогнозування були узгоджені між собою таким чином, щоб результати машинного навчання могли безпосередньо використовуватися під час формування структурованого звіту, який надається користувачеві. Такий підхід

забезпечив ефективне впровадження розроблених методів і дозволив створити завершений інструмент для аналізу транзакційної активності.

Окрему увагу приділено розробці інтерфейсу та демонстрації роботи вебсервісу. У завершальному розділі наведено послідовний огляд основних сторінок додатку, включаючи процес введення адреси, перегляд результатів класифікації та структури кластера, а також формування PDF-звіту. Це дозволило показати не лише внутрішню логіку системи, а й її прикладну цінність, зручність використання та коректність реалізованих алгоритмів.

Таким чином, після обробки великої кількості інформації та виконання поставлених завдань, мета кваліфікаційної роботи була досягнута. У результаті було створено повноцінний програмний продукт, що поєднує методи аналітики транзакцій цифрових активів, кластеризацію на основі евристик та сучасні моделі машинного навчання. Система успішно виконує поставлені завдання: аналізує транзакційні дані, формує кластери адрес, прогнозує категорії ризику та надає користувачеві інформативні й структуровані звіти. Впровадження такого інструменту надасть можливість оперативно оцінювати ризики взаємодії з окремими адресами та підсилює практичні механізми захисту користувачів під час роботи з цифровими активами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кочура В.В., Локтікова Т.М., Кушнір Н.О. Аналітичний вебсервіс для виявлення ризикових транзакцій цифрових активів на основі AML та KYC. Тези доповідей XV Міжнародної науково-технічної конференції «Інформаційно-комп'ютерні технології», 28–29 березня 2025 року. Житомир: «Житомирська політехніка», 2025. С. 19–20. URL: <https://conf.ztu.edu.ua/wp-content/uploads/2025/04/povnyj-tekst.pdf>.
2. Kochura V., Loktikova T., Kushnir N. Analytical web service for identifying suspicious high-risk digital asset transactions. Computer systems and information technologies. 2025. № 4 (Подано та прийнято до опублікування).
3. Що таке криптовалюта і як вона працює?. WhiteBIT Blog. URL: <https://blog.whitebit.com/uk/what-is-cryptocurrency/>.
4. Хто створив Біткоїн: перевірені факти про його походження. Crypto Crew. URL: <https://crew-c.com/blog/hto-stvoriv-bitkoyin/>.
5. Що таке блокчейн і як він працює?. Binance Academy. URL: <https://www.binance.com/uk-UA/academy/articles/what-is-blockchain-and-how-does-it-work>.
6. What is a blockchain?. Coinbase. URL: <https://www.coinbase.com/learn/crypto-basics/plp-what-is-a-blockchain>.
7. Liu X. F., Jiang X.-J., Liu S.-H., Tse C. K. Knowledge Discovery in Cryptocurrency Transactions: A Survey. IEEE Access. 2021. Т. 9. С. 37229–37254. URL: <https://doi.org/10.1109/ACCESS.2021.3062652>.
8. Що таке невитрачений вихід транзакції (UTXO) Bitcoin?. Kraken. URL: <https://www.kraken.com/uk/learn/what-is-bitcoin-unspent-transaction-output-utxo>.
9. Glossary. River Financial. URL: <https://river.com/learn/terms/>.
10. What Is Transaction Clustering in Crypto? Address Analysis. Nansen AI. URL: <https://www.nansen.ai/post/what-is-transaction-clustering-in-crypto-address-analysis>.
11. Wang M., Ichijo H., Xiao B. Cryptocurrency Address Clustering and Labeling. arXiv. 2020. URL: <https://doi.org/10.48550/arXiv.2003.13399>.

12. Yin H. H. S., Langenheldt K., Harlev M., Mukkamala R. R., Vatrpu R. Regulating Cryptocurrencies: A Supervised Machine Learning Approach to De-Anonymizing the Bitcoin Blockchain. *Journal of Management Information Systems*. 2019. T. 36, № 1. C. 37–73. URL: <https://doi.org/10.1080/07421222.2018.1550550>.
13. Elliptic Reviews & Ratings 2025. TrustRadius. URL: <https://www.trustradius.com/products/elliptic/reviews>.
14. Crystal Blockchain Reviews. Slashdot. URL: <https://slashdot.org/software/p/Crystal-Blockchain/>.
15. Decoding the Chain: How Data Science-Based Heuristics Reveal Blockchain Networks. Elementus. URL: <https://www.elementus.io/blog-post/decoding-the-chain-how-data-science-based-heuristics-reveal-blockchain-networks>.
16. Zhang Y., Wang J., Luo J. Heuristic-based address clustering in Bitcoin. *IEEE Access*. 2020. T. 8. C. 210581–210589. URL: <https://doi.org/10.1109/ACCESS.2020.3039570>.
17. He X., He K., Lin S., Yang J., Mao H. Bitcoin address clustering method based on multiple heuristic conditions. *IET Blockchain*. 2022. T. 2, № 2. C. 44–56. URL: <https://doi.org/10.1049/blc2.12014>.
18. Caringella M., Violante F., De Lucci F., Galantucci S., Costantini M. BACH: A tool for analyzing blockchain transactions using address clustering heuristics. *Information*. 2024. T. 15, № 10. C. 589. URL: <https://doi.org/10.3390/info15100589>.
19. Introduction to Bitcoin Heuristics. Medium. URL: <https://medium.com/cryptoquant/introduction-to-bitcoin-heuristics-487c298fb95b>.
20. Detecting Change Addresses. BCAM. URL: <https://blockchain-academy.hs-mittweida.de/courses/blockchain-introduction-technical-beginner-to-intermediate/lessons/lesson-4-privacy-in-blockchain-2/topic/detecting-change-addresses/>.

21. Schnoering H., Porthaux P., Vazirgiannis M. Assessing the efficacy of heuristic-based address clustering for Bitcoin. 2024. URL: <https://doi.org/10.48550/arXiv.2403.00523>.
22. Random Forests. IT Wiki. URL: <https://itwiki.dev/data-science/ml-reference/ml-glossary/random-forests>.
23. Extra Trees, Explained: A Visual Guide with Code Examples. Medium. URL: <https://medium.com/@samybaladram/extra-trees-explained-a-visual-guide-with-code-examples-4c2967cedc75>.
24. What is XGBoost?. IBM. URL: <https://www.ibm.com/think/topics/xgboost>.
25. Symfony Tutorial. Tutorials Point. URL: <https://www.tutorialspoint.com/symfony/index.htm>.
26. Мова програмування Python для машинного навчання та штучного інтелекту. GoIT. URL: <https://goit.global/ua/articles/mova-prohramuvannia-python-dlia-mashynnoho-navchannia-ta-shtuchoho-intelektu/>.
27. How to Get Started with Docker Compose and Symfony. Twilio. URL: <https://www.twilio.com/en-us/blog/developers/community/get-started-docker-symfony>.
28. Schnoering H., Vazirgiannis M. Bitcoin research with a transaction graph dataset. Scientific Data. 2025. T. 12, Ст. 404. URL: <https://doi.org/10.1038/s41597-025-04684-8>.
29. What is Ponzi Scheme. Tangem. URL: <https://tangem.com/en/glossary/ponzi-scheme/>.
30. What is Crypto Ransomware?. Check Point. URL: <https://www.checkpoint.com/cyber-hub/ransomware/what-is-crypto-ransomware/>.
31. Random Forest with Grid Search. Medium. URL: <https://medium.com/cloudvillains/random-forest-with-grid-search-b739fb0da311>.
32. Grid Search XGBoost Hyperparameters. XGBoosting. URL: <https://xgboosting.com/grid-search-xgboost-hyperparameters/>.

33. Micro, Macro & Weighted Averages of F1 Score, Clearly Explained. Towards Data Science. URL: <https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f/>.

34. Mastering Feature Importance in Machine Learning with Python. Train in Data. URL: <https://www.blog.trainindata.com/feature-importance-with-python/>.

35. F1 Score in Machine Learning. GeeksforGeeks. URL: <https://www.geeksforgeeks.org/machine-learning/f1-score-in-machine-learning/>.

36. F1 Score in Machine Learning: A Balanced Metric for Precision and Recall. DataCamp. URL: <https://www.datacamp.com/tutorial/f1-score>.

37. Оглядачі блокчейнів — що це, навіщо вони потрібні та як працюють. Trustee Plus. URL: <https://trusteeglobal.com/academy/oglyadachi-blokchejniv-shho-cze-navishho-vony-potribni-ta-yak-praczyuyut/>.

38. Blockchain Data API. Blockchain.com. URL: https://www.blockchain.com/explorer/api/blockchain_api/.

39. Connected Components in a Graph. Baeldung. URL: <https://www.baeldung.com/cs/graph-connected-components>.

40. Deployment Diagram Tutorial. Lucidchart. URL: <https://www.lucidchart.com/pages/uml-deployment-diagram>.

ДОДАТКИ

```

class Preprocessor:

    def __init__(self, fill_value: float = 0.0):
        self.block_dates = pd.read_csv("block_dates.csv", dtype={ "block": int,
"timestamp": float })
        self.block_dates = self.block_dates.set_index("block")["timestamp"]

        self.market_price = pd.read_csv("market_price.csv", dtype={ "timestamp": float,
"price": float })
        self.market_price = self.market_price.set_index("timestamp")["price"]

        self.fill_value = fill_value

        self.columns = None

    def fit(self, X: pd.DataFrame):
        X_ext = self.add_features(X.copy())
        self.columns = list(X_ext.columns)

    def transform(self, X: pd.DataFrame):
        X_ext = self.add_features(X.copy())
        X_ext = X_ext.fillna(self.fill_value)

        return X_ext[self.columns]

    def add_features(self, X: pd.DataFrame):
        X["avg_sent"] = X["total_sent"] / X["total_transactions_out"].replace(0,
np.nan)
        X["avg_received"] = X["total_received"] / X["total_transactions_in"].replace(0,
np.nan)
        X["sent_received_ratio"] = X["total_sent"] / X["total_received"].replace(0,
np.nan)

        X["sent_range"] = X["max_sent"] - X["min_sent"]
        X["received_range"] = X["max_received"] - X["min_received"]

        X["age"] = self.calculate_age(X)

        for col in ["total_transactions_in", "total_transactions_out", "cluster_size",
"cluster_num_edges"]:
            X[f"{col}_age_ratio"] = X[col] / np.clip(X["age"].fillna(1), 1, np.inf)

        X["time_before_first_transaction"] = X["first_transaction_out"] -
X["first_transaction_in"]

        X["transaction_per_day"] = (X["total_transactions_in"] +
X["total_transactions_out"]) / \
            np.clip(X["age"], 1, np.inf)

        median_prices = self.calculate_median_price(X)

        for col in ["min_sent", "max_sent", "total_sent", "min_received",
"max_received", "total_received"]:
            X[f"{col}_usd"] = X[col] * median_prices / 1e8

        X["avg_sent_usd"] = X["total_sent"] / X["total_transactions_out"].replace(0,
np.nan) * median_prices / 1e8
        X["avg_received_usd"] = X["total_received"] /
X["total_transactions_in"].replace(0, np.nan) * median_prices / 1e8

```

```

X["avg_tx_value_usd"] = (X["total_sent_usd"] + X["total_received_usd"]) / \
    (X["total_transactions_in"] +
X["total_transactions_out"]).replace(0, np.nan)

return X

def calculate_age(self, X: pd.DataFrame):
    first_block = X[["first_transaction_in", "first_transaction_out"]].min(axis=1)
    last_block = X[["last_transaction_in", "last_transaction_out"]].max(axis=1)

    return (last_block - first_block).rename("age")

def calculate_median_price(self, X: pd.DataFrame):
    def calculate_price(row):
        alias, first_transaction_in, last_transaction_in, last_transaction_out =
row

        if np.isnan(first_transaction_in) or np.isnan(last_transaction_in) or
np.isnan(last_transaction_out):
            return alias, np.nan

        first_block = first_transaction_in
        last_block = np.nanmax([last_transaction_in, last_transaction_out])

        if np.isnan(first_block) or np.isnan(last_block):
            return alias, np.nan

        first_block_key = int(1000 * (first_block // 1000))
        last_block_key = int(1000 * (last_block // 1000 + 1))

        first_block_index = self.block_dates.index.get_indexer([first_block_key],
method="nearest")[0]
        last_block_index = self.block_dates.index.get_indexer([last_block_key],
method="nearest")[0]

        first_transaction = self.block_dates.iloc[first_block_index]
        last_transaction = self.block_dates.iloc[last_block_index]

        if not first_transaction or not last_transaction or first_transaction <
self.market_price.index.min():
            return alias, np.nan

        prices = self.market_price.loc[(self.market_price.index >=
first_transaction)
                                         & (self.market_price.index <=
last_transaction)]

        if prices.empty:
            return alias, np.nan

        return alias, prices.median()

    rows = X.reset_index()[[
        "alias",
        "first_transaction_in",
        "last_transaction_in",
        "last_transaction_out"
    ]].values

    median_prices = Parallel(n_jobs=-1)(delayed(calculate_price)(row) for row in
rows)

```

```
return pd.Series(  
    [price[1] for price in median_prices],  
    index=[price[0] for price in median_prices],  
    name="median_price"  
)
```

```

class ClusterService
{
    public function __construct(
        private readonly EntityManagerInterface $entityManager,
        private readonly BlockchainDataService $blockchainDataService,
        private readonly HeuristicsService $heuristicsService
    ) {}

    /**
     * @param Report $report
     * @return array
     */
    public function createClusterFeatures(Report $report): array
    {
        $rootAddress = $this->blockchainDataService->fetchSingleAddress($report->
>getAddress());

        if (!$rootAddress) {
            throw new RuntimeException('Адресу не знайдено. Для демонстрації роботи
кластеризації необхідно використовувати адреси, що містяться у синтетичних JSON-
файлах.');
```

```

        }

        $clusterAddresses = $this->buildCluster($rootAddress);

        return $this->calculateClusterFeatures($clusterAddresses, $report->
>getAddress());
    }

    /**
     * @param array $rootAddress
     * @return array
     */
    private function buildCluster(array $rootAddress): array
    {
        $cluster = $reusedOutputs = [];

        $queue = [$rootAddress];

        while ($queue) {
            $address = array_shift($queue);

            if (isset($cluster[$address['address']])) {
                continue;
            }

            $cluster[$address['address']] = $address;

            $addresses = $this->heuristicsService->applyHeuristics($address,
$reusedOutputs);

            $queue = array_merge($queue, $this->blockchainDataService->
>fetchMultipleAddresses($addresses));
        }

        return $cluster;
    }
}

```

```

* @param array $clusterAddresses
* @param string $rootAddress
* @return array
*/
private function calculateClusterFeatures(array $clusterAddresses, string
$rootAddress): array
{
    $features = [
        'total_transactions_in'    => 0,
        'total_transactions_out'   => 0,
        'min_sent'                 => 0,
        'max_sent'                 => 0,
        'total_sent'               => 0,
        'min_received'             => 0,
        'max_received'             => 0,
        'total_received'          => 0,
        'first_transaction_in'     => null,
        'last_transaction_in'      => null,
        'first_transaction_out'    => null,
        'last_transaction_out'     => null,
        'first_transaction_in_time' => null,
        'last_transaction_in_time' => null,
        'first_transaction_out_time' => null,
        'last_transaction_out_time' => null,
        'cluster_size'            => count($clusterAddresses),
        'cluster_num_edges'       => 0,
        'cluster_num_cc'          => 0,
        'cluster_num_nodes_in_cc' => 0,
        'addresses'               => array_reduce(
            $clusterAddresses,
            function ($result, $address) use ($rootAddress) {
                if ($address['address'] === $rootAddress) {
                    return $result;
                }

                $result[] = [
                    'address'           => $address['address'],
                    'transactionsCount' => $address['n_tx'] ?? 0,
                    'totalReceived'     => $address['total_received'] ?? 0,
                    'totalSent'         => $address['total_sent'] ?? 0,
                ];
            }, []
        ),
    ];

    $addresses = array_keys($clusterAddresses);

    $graph = [];

    foreach ($addresses as $address) {
        $graph[$address] = [];
    }

    foreach ($clusterAddresses as $clusterAddress) {
        foreach ($clusterAddress['txs'] as $transaction) {
            $block = $transaction['block_height'];
            $time = $transaction['time'];

            $hasNotClusterInput = $hasNotClusterOutput = false;

```

```

$clusterInputs = $clusterOutputs = [];

$clusterReceived = $clusterSent = 0;

foreach ($transaction['inputs'] as $input) {
    if (in_array($input['prev_out']['addr'], $addresses)) {
        $clusterInputs[] = $input['prev_out']['addr'];
    } else {
        $hasNotClusterInput = true;
    }
}

foreach ($transaction['out'] as $output) {
    if (in_array($output['addr'], $addresses)) {
        $clusterOutputs[] = $output['addr'];

        if ($hasNotClusterInput) {
            $clusterReceived += $output['value'];
        }
    } else {
        $hasNotClusterOutput = true;

        if (count($clusterInputs) > 0) {
            $clusterSent += $output['value'];
        }
    }
}

if ($hasNotClusterInput && count($clusterOutputs) > 0) {
    $features['total_transactions_in']++;

    if ($features['min_received'] === 0 || $clusterReceived <
$features['min_received']) {
        $features['min_received'] = $clusterReceived;
    }

    if ($clusterReceived > $features['max_received']) {
        $features['max_received'] = $clusterReceived;
    }

    $features['total_received'] += $clusterReceived;

    if ($features['first_transaction_in'] === null || $block <
$features['first_transaction_in']) {
        $features['first_transaction_in'] = $block;
        $features['first_transaction_in_time'] = $time;
    }

    if ($features['last_transaction_in'] === null || $block >
$features['last_transaction_in']) {
        $features['last_transaction_in'] = $block;
        $features['last_transaction_in_time'] = $time;
    }
}

if ($hasNotClusterOutput && count($clusterInputs) > 0) {
    $features['total_transactions_out']++;

    if ($features['min_sent'] === 0 || $clusterSent <
$features['min_sent']) {
        $features['min_sent'] = $clusterSent;
    }
}

```

```

        if ($clusterSent > $features['max_sent']) {
            $features['max_sent'] = $clusterSent;
        }

        $features['total_sent'] += $clusterSent;

        if ($features['first_transaction_out'] === null || $block <
$features['first_transaction_out']) {
            $features['first_transaction_out'] = $block;
            $features['first_transaction_out_time'] = $time;
        }

        if ($features['last_transaction_out'] === null || $block >
$features['last_transaction_out']) {
            $features['last_transaction_out'] = $block;
            $features['last_transaction_out_time'] = $time;
        }
    }

    if (!empty($clusterInputs) && !empty($clusterOutputs)) {
        $features['cluster_num_edges']++;
    }

    foreach ($clusterInputs as $clusterInput) {
        foreach ($clusterOutputs as $clusterOutput) {
            if ($clusterInput !== $clusterOutput) {
                $graph[$clusterInput][] = $clusterOutput;
                $graph[$clusterOutput][] = $clusterInput;
            }
        }
    }
}

}

}

$this->findConnectedComponents($graph, $features);

return $features;
}

/**
 * @param array $graph
 * @param $features
 * @return void
 */
private function findConnectedComponents(array $graph, &$features): void
{
    $visited = [];

    foreach ($graph as $node => $neighbors) {
        if (isset($visited[$node])) {
            continue;
        }

        $componentSize = 0;

        $query = [$node];

        while ($query) {
            $current = array_pop($query);

            if (isset($visited[$current])) {

```

```

        continue;
    }

    $visited[$current] = true;
    $componentSize++;

    foreach ($graph[$current] as $neighbor) {
        if (!isset($visited[$neighbor])) {
            $query[] = $neighbor;
        }
    }
}

if ($componentSize < 2) {
    continue;
}

$features['cluster_num_cc']++;
$features['cluster_num_nodes_in_cc'] += $componentSize;
}
}

/**
 * @param array $features
 * @return Cluster
 */
public function createClusterObject(array $features): Cluster
{
    $cluster = new Cluster();

    $cluster
        ->setTotalTransactionsIn($features['total_transactions_in'])
        ->setTotalTransactionsOut($features['total_transactions_out'])
        ->setMinSent($features['min_sent'])
        ->setMaxSent($features['max_sent'])
        ->setTotalSent($features['total_sent'])
        ->setMinReceived($features['min_received'])
        ->setMaxReceived($features['max_received'])
        ->setTotalReceived($features['total_received'])
        ->setFirstTransactionIn($features['first_transaction_in'])
        ->setLastTransactionIn($features['last_transaction_in'])
        ->setFirstTransactionOut($features['first_transaction_out'])
        ->setLastTransactionOut($features['last_transaction_out'])
        ->setFirstTransactionInTime($features['first_transaction_in_time'])
        ->setLastTransactionInTime($features['last_transaction_in_time'])
        ->setFirstTransactionOutTime($features['first_transaction_out_time'])
        ->setLastTransactionOutTime($features['last_transaction_out_time'])
        ->setClusterSize($features['cluster_size'])
        ->setClusterNumEdges($features['cluster_num_edges'])
        ->setClusterNumCc($features['cluster_num_cc'])
        ->setClusterNumNodesInCc($features['cluster_num_nodes_in_cc']);

    $this->entityManager->persist($cluster);

    return $cluster;
}
}

```