Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Кластеризація sybil-адрес на блокчейні методами машинного навчання
(тема)

Виконав:
студент 2 курсу, групи ДСм-22-2
Аріткулова Ю.Р.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Науки про дані
(повна назва спеціалізації)
Керівник доц. Вітько О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Науки про дані
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Аріткуловій Юлії Равілівні
(прізвище, ім'я, по батькові)

1. Тема роботи Кластеризація sybil-адрес на блокчейні методами машинного навчання

затверджена наказом університету від 17 листопада 20 23 р. № 1365Ст
2. Термін подання студентом роботи до екзаменаційної комісії 18 січня 20 24 р.
3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел щодо тематики кваліфікаційної роботи; Python-документація

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі
2) Дослідження методів вирішення задачі
3) Процес кластерізації

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	17.11.2023	виконано
2	Аналіз предметної галузі та постановка задачі	24.11.2023	виконано
3	Аналіз методів вирішення поставленої задач	27.11.2023	виконано
4	Вибір методу вирішення задачі	30.11.2023	виконано
5	Розробка базового прототипу	15.12.2023	виконано
6	Розробка програми	31.12.2023	виконано
7	Написання пояснювальної записки	07.01.2024	виконано
8	Попередній захист	16.01.2024	виконано
9	Захист перед ЕК	11.06.2024	

Дата видачі завдання 17 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Вітько О.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 53 с., 40 рис., 2 дод., 20 джерел.

БЛОКЧЕЙН, КЛАСТЕР, МАШИННЕ НАВЧАННЯ, МНОЖИННІ ІДЕНТИЧНОСТІ, СИБІЛ, ТРАНЗАКЦІЯ.

Об’єкт дослідження – методи машинного навчання для розпізнавання кластерів.

Предмет дослідження – sybil-clusters (кластери сибілів), тобто множина адрес на блокчейні, створена однією чи групою осіб з цілями отримати переваги від цих множинних ідентичностей.

Мета роботи – дослідити явище sybils та sybil-attacks (сибіл-атак), особливо зосереджуючись на появі та динаміці так званих кластерів адрес у мережах блокчейну.

Методи дослідження – теоретичний, емпіричний.

В результаті були проведені дослідження для вирішення задачи розпізнавання кластерів сибілів за допомогою методів машинного навчання.

В якості мови програмування було обрано Python. Дане дослідження може бути корисне в подальшому для розробки систем, що будуть стійкі до атак чи зловживання з боку множинних ідентичностей.

ABSTRACT

Master’s thesis contains: 53 p., 40 fig., 2 ann., 20 references.

BLOCKCHAIN, CLUSTER, MACHINE LEARNING, MULTIPLE IDENTITIES, SYBIL, TRANSACTION.

The object of research is machine learning methods for cluster recognition.

The subject of the research sybil-clusters, which are sets of addresses on the blockchain created by one or a group of individuals with the aim of gaining advantages from these multiple identities.

The purpose of the study is to investigate the phenomenon of sybils and sybil-attacks, with a particular focus on the emergence and dynamics of so-called address clusters in blockchain networks.

Research methods – theoretical, empirical.

As a result, study was conducted to address the problem of sybil cluster recognition using machine learning 4methods. Python was chosen as the programming language. This research may be useful in the future for developing systems that are resistant to attacks or abuse from multiple identities.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Адреса (Address) – унікальний ідентифікатор, який використовується для надсилання та отримання криптовалюти на блокчейні. Вона складається з букв і цифр і може виглядати як набір символів.

Блокчейн (Blockchain) – розподілена база даних або реєстр, що складається з ланцюжка блоків, кожен з яких містить записи транзакцій. Блокчейн є основою більшості криптовалют і забезпечує прозорість, безпеку та незмінність даних.

Ейдроп (Airdrop) – процес роздачі криптовалютних токенів безкоштовно певній групі користувачів. Часто використовується для залучення нових користувачів або як маркетингова стратегія.

Номер блоку (Block Number) – порядковий номер блоку в блокчейні, який показує його позицію у ланцюжку. У кожного блоку є своя відмітка часу – timestamp.

Нонс (Nonce) – число, яке використовується один раз у кожному гаманці і позначає номер транзакції по порядку для цього гаманця.

Токен (Token) – цифровий актив, створений на основі блокчейн-технології, який може представляти різні цінності, включаючи валюти, права на активи або доступ до певних послуг.

Транзакція (Transaction) – дія, яка записується в блокчейн, включає передачу криптовалюти або інших активів між адресами. Транзакція містить інформацію про відправника, отримувача та суму переказу.

ВСТУП

У світі технології блокчейн, що постійно розвивається, проблема атак Sybil стала критичною проблемою, створюючи значні загрози цілісності та безпеці децентралізованих систем. Sybil-атаки влаштовують зловмисники, що створюють кілька облікових записів, чи ідентичностей, для маніпулювання мережею, що призводить до компрометації механізмів консенсусу та довіри, на якій покладається блокчейн.

Ця робота має на меті дослідити явище сибілів, тобто множинних ідентичностей, створених однією людиною чи групою людей з метою отримати переваги в порівнянні з одиничним обліковим записом, особливо зосереджуючись на появі та динаміці так званих кластерів адрес у мережах блокчейну. Sybil-кластери, які характеризуються скоординованими групами зловмисних вузлів, що працюють у тандемі, представляють деякий виклик для надійності децентралізованих систем. Це дослідження має на меті пролити світло на те, як можна виявити такі кластери за допомогою методів машинного навчання, що, зрештою, сприятиме вдосконаленню протоколів безпеки в децентралізованих середовищах. Оскільки блокчейн продовжує революціонізувати різні галузі, захист від кластерів Sybil стає вкрай необхідною умовою для забезпечення стійкості та надійності цих трансформаційних технологій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд предметної галузі

1.1.1 Що таке блокчейн

Блокчейн – це децентралізована та розподілена технологія реєстру, яка лежить в основі криптовалют, найвідомішою з яких є біткоін, але її застосування виходить далеко за межі цифрових валют. За своєю суттю блокчейн – це ланцюжок блоків, де кожен блок містить список транзакцій. Можна провести аналогію з базою даних. Ключовими аспектами є децентралізація, незмінність та механізми консенсусу.

Децентралізація – на відміну від традиційних централізованих систем, де контроль контролює центральна влада, блокчейн працює в одноранговій мережі, де вузли рівні між собою. Ця децентралізована структура підвищує прозорість і знижує ризик єдиної точки відмови.

Незмінність – коли блок додано до ланцюжка, його надзвичайно важко змінити. Це досягається за допомогою криптографічних хешів і механізмів консенсусу, що робить дані в блокчейні захищеними від втручання.

Механізми консенсусу – мережі блокчейнів покладаються на алгоритми консенсусу для перевірки та узгодження стану реєстру. До популярних механізмів належать Proof of Work (використовується Bitcoin) і Proof of Stake (наприклад, Ethereum) [1].

1.1.2 Способи використання блокчейну

Криптовалюти: найвідомішим застосуванням блокчейну є створення та передача цифрових валют. Біткойн, Ефір та інші криптовалюти використовують блокчейн для безпечних транзакцій без посередників [2].

Смарт-контракти: Ethereum запропонував концепцію смарт-контрактів, тобто програм з різними умовами, наприклад, для надсилання коштів, записаними безпосередньо в код. Ці контракти автоматизують і забезпечують виконання прописаних умов, і прибирають потребу довіряти іншій стороні для виконання угоди, використовувати посередника, тощо, адже код завжди виконується однаково та детерміновано.

Керування ідентифікацією: блокчейн пропонує безпечний і децентралізований спосіб керування цифровими ідентичностями. Це може бути особливо корисним у ситуаціях, коли перевірка особи є надзвичайно важливою, наприклад, у системах голосування чи доступі до фінансових послуг.

Охорона здоров’я: блокчейн може підвищити безпеку та сумісність записів охорони здоров’я, так як він, по суті, є великим реєстром. Пацієнти, лікарі та інші уповноважені сторони можуть отримати доступ до історії пацієнта.

Фінанси та банківська справа: оптимізація та покращення різноманітних фінансових процесів, у тому числі платежів за кордон, фінансування торгівлі та розрахунків за цінними паперами, доступність фінансових інструментів для звичайного користувача.

Токенізація активів: дозволяє токенізувати активи реального світу, перетворюючи фізичні чи неліквідні активи в цифрові токени, якими можна торгувати на блокчейні.

1.1.3 Airdrops, retrodrops

Airdrop (ейрдроп, дослівно – роздача з повітря) – це маркетингова стратегія, яка використовується блокчейн-проектами для розповсюдження безкоштовних токенів [3]. Мета полягає в тому, щоб підвищити обізнаність про новий проект, винагородити лояльних користувачів і заохотити участь у спільноті. Зазвичай умови роздачі відомі заздалегідь і не потребують

витрат, часто потрібно виконати певні соціальні завдання, що виконують роль реклами – репост, лайк, зняти відео про проект, тощо. Таким чином, проект винагороджує своїми токенами активних користувачів за рекламу та розголос.

Ретроспективний дроп (retrodrop) – грошова винагорода за використання продукту на ранній стадії [4]. На відміну від airdrop, ретродроп має на увазі активності з грошовими витратами при роботі з платформою. Також, умови роздачі токенів майже ніколи не описані заздалегідь, можна лише здогадуватися, аналізуючі схожі проекти. Не можна знати навіть, чи буде ця роздача взагалі, чи гроші на транзакції були витрачені впусту.

Це може бути корисно проєкту для створення спільноти, децентралізації, маркетингу.

Створення спільноти, так як ретродропи – це спосіб залучити спільноту до нового проєкту. Розповсюджуючи токени серед широкої аудиторії, проекти сподіваються залучити користувачів, які зацікавлені в цілях і функціях проєкту.

Якщо токен використовується для керування протоколом, це забезпечує децентралізацію, на відміну від випадків, коли значна кількість влади зосереджена в одних руках, а отже, ця людина може маніпулювати рішеннями на свою користь.

Маркетинг і просування: ретродропи є формою маркетингу, створюючи розголос і привертаючи увагу до проєкту. Роздаючи токени, компанії прагнуть викликати інтерес і спонукати користувачів досліджувати проект і потенційно інвестувати в нього.

Важливо зазначити, що специфіка роздачі може сильно відрізнятися від проєкту до проєкту. Крім того, криптовалютний простір є динамічним, і з часом можуть з’являтися нові терміни чи поняття, зокрема – варіації механізмів розподілу.

1.1.4 Полювання за ретродропами

Так як ретродропи можуть принести відносно великі кошти за відносно малу кількість зусиль, люди стали полювати на них. З’явився так званий ретрохантинг (від retro + hunting, полювання на ретро), що включає в себе аналіз потенційних проектів, що можуть надати токени раннім користувачам. Але більше того, в пошуках примноження вигоди люди стали створювати так звані «ферми», тобто сукупності акаунтів/гаманців (тобто sybils – одна з перших згадок цього поняття описана в джерелі [5]), що дозволяє примножити можливу вигоду, виконуючі однакові дії на потенційно перспективних проектах чи блокчейнах, у яких ще немає власного токену.

Водночас, так як мета кожного проєкту залучити якомога більше індивідуальних особистостей і нагородити активних користувачів, вони намагаються відстежити такі ферми (по суті, кластери, що мають певні спільні ознаки – IP-адресу, повторювані дії на гаманцях, тощо) і відсіяти їх. Іде протиборство, хто кого перехитрить. З цієї боротьби і з’явилась ідея випробувати сили машинного навчання для пошуку таких кластерів сибілів.

1.2 Аналіз джерел

1.2.1 Sybil Attack

Атака Sybil (атака сибілів) – це форма порушення онлайн-безпеки, коли суб’єкт має численні підроблені ідентифікатори в блокчейні зі зловмисних причин [5]. Для одного погляду ці різні особи виглядають як звичайні користувачі, але за лаштунками одна сутність називається невідомим зловмисником, який контролює всі ці підроблені сутності одночасно [6]. Походження терміну «Сибіла» можна простежити до роману Флори Шрайбер 1973 року. У ньому є персонаж на ім'я Сибіла, яка має

медичну проблему, пов'язану з ідентичністю. Вона завжди намагалася оперувати кількома ідентичністями. Цей термін увійшов у технологію після того, як Джон Р. Дусер опублікував коротку дослідницьку статтю кілька десятиліть тому.

1.2.2 Які загрози можуть спричинити Sybil Attack

Sybil Attack можуть бути катастрофічними для цілісності блокчейну. Дві основні загрози, які вони можуть спричинити, це порушення конфіденційності, захоплення чи утримання блокчейну від роботи.

Технологія блокчейн приділяє особливу увагу безпеці та конфіденційності, але Sybil Attack атакує аспект конфіденційності. Зловмисник може запустити шкідливий вузол, щоб збирати конфіденційні дані, які передають чесні вузли. Однією з деталей може бути незаконне отримання IP-адрес тих, хто стоїть за чесними вузлами.

Захоплення блокчейну – у контексті блокчейну однією з вразливостей, яку може спричинити атака Sybil, є атака 51%. Вона відбувається, коли одна особа або група осіб отримує повний контроль над блокчейном, і, таким чином, неавтентичні або шкідливі вузли перевищують кількість чесних вузлів. Це повноваження дає зловмиснику перевагу в процесах прийняття рішень протоколу. Це саме те, що сталося під час атаки Litecoin Cash у 2019 році.

Атака блокування – у цьому типі атаки значна кількість валідаторів, які діють проти інтересів блокчейну, відмовляються додавати нові блоки до ланцюжка. Це може призвести до того, що він з часом перейде в заморожений стан і стане непридатним для використання, і необхідне буде втручання для відновлення його функціональності.

Є спроби ідентифікувати атаку сибілів в соціальних мережах за допомогою машинного навчання, наприклад [7].

1.2.3 Сибіли в контексті ретродропів

В контексті ретродропів шкода сибілів не є такою однозначною. Звісно, з точки зору децентралізації вони негативно впливають на рівномірний розподіл токенів, що суперечіть меті роздачі токенів таким чином. Але, з іншого боку, вони часть перевіряють систему на стійкість, збільшуючи навантаження на неї. Також, це перетворюється у своєрідне змагання, хто кого перехитрить: чи проект з новими способами аналізу, що дозволяє відстежити сибілів, чи користувачі з новими способами обійти систему.

Наприклад, цитуючи Arbitrum Foundation [8] (переклад з англійської): «Цей проект має на меті видалити адреси Sybil з аірдропу Arbitrum, гарантуючи, що лише законні користувачі отримають токени аірдропу [8].»

Далі вони трохи описують методологію: «Ми використовуємо дані з блокчейну, щоб ідентифікувати пов’язані адреси, що належать одному користувачеві, і видаляючи адреси не-користувачів, наприклад мости, біржі та смарт-контракти, використовуючи дані Nansen [9], Hop [10], [11] і OffChain Labs [12] (компанії, що спеціалізуються на ончейн-аналітиці). Деякі адреси також видаляються під час перевірки вручну [8].»

Також згадується про кластери: «Ідентифікація Sybil-кластерів. Кластери генеруються шляхом розбиття наведених нижче графів (рисунки 1.1, 1.2) на сильнозв’язні та слабозв’язні підграфи. Великі підграфи розбиваються за допомогою Louvain Community Detection Algorithm, що забезпечує точніші результати та точніше усуває адреси Sybil. Ми ідентифікуємо кластери Сибіл на основі відомих шаблонів, наприклад:

– адреси переказу коштів у кластері з понад 20 адрес;
– адреси, які фінансуються з одного джерела;
– адреси зі схожою діяльністю [8].»

Алгоритм Лувена – це алгоритм виявлення спільнот, який використовується в аналізі мережі для ідентифікації груп або спільнот у

складній мережі [13]. Два кроки – оптимізація модульності та агрегація спільноти – виконуються, доки в мережі не буде більше змін і не буде досягнуто максимальної модульності. Модульність – це показник, який використовується для оцінки якості поділу мережі на спільноти, вимірюється як щільність зв’язків усередині спільнот порівняно зі зв’язками між спільнотами.

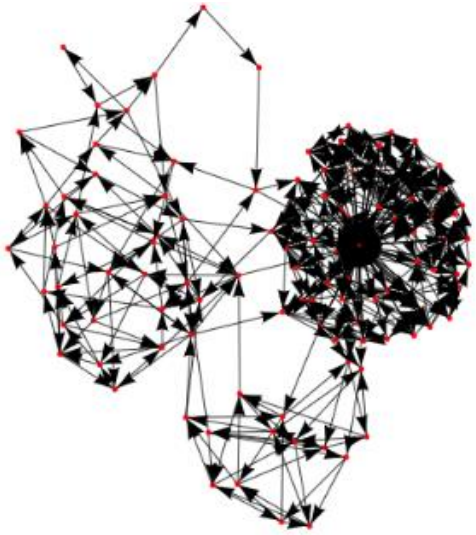


Рисунок 1.1 – Кластер 319 з 110 адрес, що задовольняють умови [8]

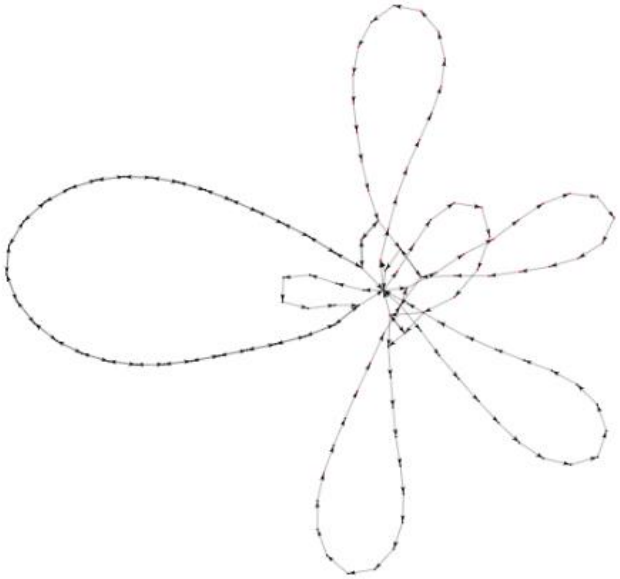


Рисунок 1.2 – Кластер 1544 з 56 адрес, що задовольняють умови [8]

1.3 Постановка задачі

В кваліфікаційні роботі ставиться задача дослідити алгоритми кластерізації, що дозволять об’єднувати адреси на блокчейні за певними метриками, приклад взяти з джерела [8]. Об’єктом дослідження обрати один блокчейн, наприклад zkSync, як один із нещодавно запущених, що предоставляє цікавість для власників множинних ідентичностей, так як не має свого токену, є досить відомим та отримав інвестиції від відомих фондів.

Дослідження буде надавати можливі способи пошуку таких кластерів, включаючи метрики, що варто врахувати, способи збирання даних, приклади алгоритмів, що зможуть вирішити дану задачу з достатньою точністю.

2 ДОСЛІДЖЕННЯ МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ

2.1 Кластерізація

Кластерізація – це метод машинного навчання, який використовується для групування схожих даних [14]. Цей метод не використовує заздалегідь позначені категорії, а натомість шукає спільні особливості або характеристики в даних та об'єднує їх на основі цих рис. Цей процес допомагає виявити приховані закономірності, визначити схожість і зрозуміти структуру даних.

Метою кластеризації є розділення набору даних на групи або кластери, щоб точки даних в одному кластері були більш схожими одна на одну порівняно з точками в різних кластерах.

Для кластеризації використовуються різні алгоритми та методи, кожен із яких має свої сильні та слабкі сторони. Деякі поширені методи кластеризації включають кластеризацію K-середніх, ієрархічну кластеризацію, DBSCAN (просторова кластеризація додатків на основі щільності з шумом) і моделі суміші Гауса (GMM).

Застосування кластерного аналізу у загальному вигляді зводиться до наступних етапів:

а) вибір вибірки об'єктів для кластеризації;

б) визначення множини змінних, по яким будуть оцінюватись об'єкти у вибірці. за потреби потрібно нормалізацувати значення змінних;

в) обчислення значень міри схожості між об'єктами;

г) застосування методу кластерного аналізу до створення груп подібних об'єктів (кластерів);

д) подання результатів аналізу.

2.2 Міри відстаней

Міри відстані відіграють вирішальну роль в алгоритмах кластеризації, оскільки вони визначають, як кількісно визначається подібність або відмінність між точками даних. Ці заходи допомагають алгоритмам кластеризації групувати точки даних, які знаходяться близько одна до одної в просторі ознак [15]. Ось огляд основних вимірів, які зазвичай використовуються в кластеризації:

- Евклідова відстань;
- Манхеттенська відстань;
- відстань Мінковського;
- косинус подібності;
- подібність Жаккара;
- відстань Хеммінга.

2.3 Алгоритми кластеризації

Існує багато алгоритмів кластеризації, кожен з яких має свої переваги та недоліки. Розглянемо нижче деякі з них [16].

2.3.1 K-середніх

Переваги: простий, швидкий, ефективний для великих наборів даних.

Недоліки: чутливий до викидів, залежить від вибору початкових центрів кластерів, не завжди знаходить оптимальне рішення.

2.3.2 Ієрархічна кластеризація

Переваги: може знаходити кластери складної форми, не залежить від вибору початкових центрів кластерів.

Недоліки: може бути повільною для великих наборів даних, складно інтерпретувати результати.

2.3.3 DBSCAN

Переваги: ефективний для кластерів з нечіткими межами, стійкий до шуму.

Недоліки: може бути чутливим до вибору параметрів, не завжди знаходить кластери правильної форми.

2.3.4 Метод зсуву середнього

Переваги: простий, швидкий, ефективний для кластерів сферичної форми.

Недоліки: чутливий до викидів, не може знаходити кластери складної форми.

2.3.5 Spectral clustering

Переваги: може знаходити кластери складної форми, може використовуватись для нелінійно розділених даних.

Недоліки: може бути повільною для великих наборів даних, потребує знання спектральної теорії.

2.3.6 Fuzzy clustering

Переваги: може знаходити кластери з перекриттям, може використовуватися для нечітких даних.

Недоліки: може бути складною для інтерпретації, може бути повільним для великих наборів даних.

2.3.7 OPTICS

Переваги: може знаходити кластери різної щільності, стійкий до шуму.

Недоліки: може бути чутливим до вибору параметрів, може бути повільним для великих наборів даних.

2.3.8 Gaussian Mixture Models (GMM)

Переваги: гнучкість у роботі з кластерами різних форм і розмірів, забезпечує імовірнісні призначення кластерів, добре працює для даних із базовим розподілом Гауса.

Недоліки: вимагає припущення розподілу Гауса, чутливий до параметрів ініціалізації, може сходитися до локальних екстремумів.

2.3.9 Метод поширення спорідненості (Affinity Propagation)

Affinity Propagation – це алгоритм кластеризації, який працює шляхом надсилання повідомлень між парами точок даних, доки не з’явиться набір екземплярів. Для цього не потрібно заздалегідь вказувати кількість кластерів, що робить його доцільним вибором, коли кількість кластерів невідома або її важко визначити.

Переваги: простота, гнучкість, стійкість до шуму, можливість знаходити кластери складної форми – алгоритм може знаходити кластери складної форми, на відміну від алгоритмів, таких як k-середніх.

Недоліки: чутливість до параметрів, обчислювальна складність, складність інтерпретації результатів.

2.4 Вибір алгоритму

Для поставленої задачі було обрано алгоритм AffinityPropagation. На відміну від багатьох алгоритмів кластеризації, які вимагають заздалегідь вказати кількість кластерів (наприклад, K-means), він автоматично визначає кількість кластерів на основі вхідних даних. Це особливо корисно, коли оптимальна кількість кластерів невідома заздалегідь.

Affinity Propagation може ефективно обробляти кластери різної форми, включно з неопуклими формами. Це перевага перед такими алгоритмами, як K-means, які мають тенденцію знаходити сферичні кластери.

Алгоритм не вимагає початкового припущення щодо положення центрів кластерів, що допомагає зменшити залежність від початкових умов і робить результати більш стабільними та відтворюваними [17].

Так як Affinity Propagation може не включати кожну точку даних у кластер і дозволяє деяким точкам залишатися викидами, він може ефективніше обробляти шумні набори даних.

Алгоритм можна застосовувати до даних різного масштабу без значної втрати продуктивності, що робить його універсальним для різних типів наборів даних [18]. Affinity Propagation був успішно застосований у різних областях, таких як обробка зображень, біоінформатика та аналіз соціальних мереж, демонструючи його універсальність і ефективність для різних типів даних.

2.5 Інструментальні засоби для реалізації досліджень

Було обрано бібліотеку scikit-learn для мови програмування Python. Вона представляє широкий вибір алгоритмів кластеризації з документацією до них (рисунок 2.1).

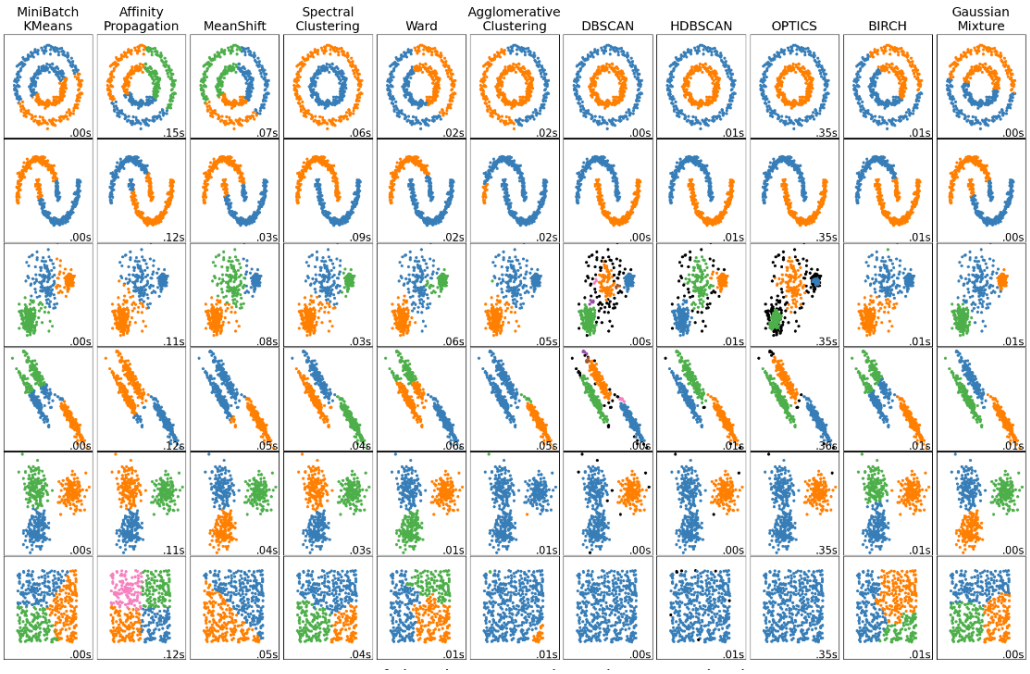


Рисунок 2.1 – Спектр алгоритмів кластеризації в scikit-learn [16]

Також на сайті бібліотеки наведені приклади демо, що дозволяють швидше зрозуміти принципи роботи кожного алгоритму [19], а також наведені приклади візуалізації результатів роботи алгоритму.

3 ПРОЦЕС КЛАСТЕРІЗАЦІЇ

3.1 Відбір даних

Для знаходження сирих даних був використаний аналітичний сервіс Dune [20], що дозволяє створювати запити по ончейн-даним на мові, подібної до SQL (DuneSQL [21]).

Запити задавались по таблиці zksync.transactions, що зберігає в собі транзакції з блокчейну ZkSync, а саме: час транзакції, nonce, чи успішною була транзакція, хеш, хто був ініціатором транзакції та хто був отримувачем. Дані постійно оновлюються та є актуальними.

Було створено два запити:

1) перші транзакції (рисунок 3.1), виконані з адреси (тобто nonce = 0). Nonce – порядковий номер для кожної транзакції з конкретного гаманця;

2) депозит-транзакції, тобто ті транзакції, які перші вносили кошти на адресу, і відбулися до того, як адреса зробила свою першу транзакцію, тобто funding transaction (рисунок 3.2).

```
1    SELECT block_number,value,nonce,"from",to
2    FROM zksync.transactions
3    WHERE
4        block_time > CURRENT_TIMESTAMP - INTERVAL '1' day
5        AND
6        success = true
7        and
8        nonce = 0
```

Рисунок 3.1 – Запит для знаходження першої транзакції на різних гаманцях

```
SELECT
  zksync.transactions.block_number,
  value,
  nonce,
  zksync.transactions."from",
  zksync.transactions.to
from
  (
    select "from", block_number
    FROM
      zksync.transactions
    WHERE
      block_time > CURRENT_TIMESTAMP - INTERVAL '12' hour
      AND success = true
      AND nonce = 0
  ) as first_time_senders,
  zksync.transactions
where
  first_time_senders."from" = zksync.transactions.to
  and zksync.transactions.block_time > CURRENT_TIMESTAMP - INTERVAL '7' day
  and zksync.transactions.success = true
  and zksync.transactions.block_number < first_time_senders.block_number
```

Рисунок 3.2 – Запит для знаходження депозит-транзакцій

Було вирішено обмежити часові рамки, оскільки обчислювальні потужності ноутбука обмежені і неможливо обробити величезні масиви даних. Тому у першому запиті було обрано проміжок у 1 день, і кількість рядків датасету виявилась рівною 15 479.

У другому запиті було обрано часовий проміжок 12 годин, протягом яких відстежувались перші транзакції (навіть при такому скромному часовому проміжку кількість підходящих рядків сягнула 11 тисіч), і протягом 7 днів відстежувались депозит-транзакції, що передували цим першим транзакціям, що результувало у кількості 6 958 рядків.

По тим же причинам обмеженої обчислювальної потужності було вирішено обмежитись двома запитами.

3.2 Завантаження даних

Для завантаження транзакцій було використано мову програмування Python разом з Dune API, що дозволяє отримувати результати конкретного запиту (рисунок 3.3).

```
base_url = f"https://api.dune.com/api/v1/query/{query_id}/results/csv"
result_response = requests.request("GET", base_url, headers=headers, params={})
```

Рисунок 3.3 – Завантаження результату запиту

3.3 Передпроцесинг, нормалізація

Розгляднемо процеси предпроцесингу та нормалізації на прикладі другого запиту. Як можна побачити на рисунку 3.4, завантажені дані не є зручними для обробки. У назвах атрибутів деінде присутні лапки, числа зображені у науковій нотації, а усі поля, навіть числові, подані у вигляді об’єктів (рисунок 3.5).

```
1 tx.head()
```

	block_number"	"value"	"nonce"	"from"	"to
0	2.9297726e+07"	"10045080010000000"	"0"	"0x40a49b8623f1dbe9892afcb0282682313172ec2a"	"0x9f755cc3537812cf2048ba76192dd8383e673fe1
1	2.9297724e+07"	"11000000000000000"	"186254"	"0xf89d7b9c864f589bbf53a82105107622b35eaa40"	"0xc835b4a3c410621640b63d70d5b5a1e2a0e46fe5
2	2.9298201e+07"	"3160290000000000"	"672297"	"0x7aed074ca56f5050d5a2e512ecc5bf7103937d76"	"0xe92280a05d1769041fd13865c46fdb047fc03e53
3	2.9298069e+07"	"7000000000000000"	"216"	"0x9aaf68ee6a063979c1257b0fec1875e16d40c11d"	"0x02dd71f4f0ad04b37e8a46cdacddfd7d33f5aa8c
4	2.9298065e+07"	"12537410000000000"	"1.889169e+06"	"0x05411bcc63167a82057be42bc1b7022c12c99aa4"	"0x85f0bb974ef168da411a8dd6a6e8e27048dbee09

Рисунок 3.4 – Завантажений датасет

```
1  tx.dtypes
```

```
block_number    object
value           object
nonce           object
from            object
to              object
dtype: object
```

Рисунок 3.5 – Типи даних

Першим кроком було прибрано лапки, як показано на рисунку 3.6, та приведено усі поля до числового чи строкового типу змінної (рисунок 3.7):

```
1  tx = tx.rename({'block_number"': 'block_number', '"value"': 'value',
2              '"nonce"': 'nonce', '"from"': 'from', '"to': 'to',}, axis='columns')
3  tx = tx.replace({'"': ''}, regex=True)
```

Рисунок 3.6 – Видалення лапок у назвах атрибутів та у даних

```
1  tx["from"] = tx["from"].astype(str)
2  tx["to"] = tx["to"].astype(str)
3  tx["block_number"] = tx["block_number"].astype(float).astype(int)
4  tx["value"] = tx["value"].astype(float)
5  tx["nonce"] = tx["nonce"].astype(float).astype(int)
```

Рисунок 3.7 – Приведення полів до відповідних типів

Оскільки адреси to та from не несуть корисної інформації самі по собі, а цікаві лише тим, що можуть повторюватись, була створена функція мапінгу, що ставить у відповідність кожній адресі унікальний номер (рисунок 3.8).

```
def mapAddressToNumber(series, feature_map, start_count):
  for i in series.unique():
    if i not in feature_map:
      feature_map[i]=start_count
      start_count=start_count+1
  return feature_map, start_count
```

Рисунок 3.8 – Функція мапінгу

Після виклику функції, було виявлено, що атрибут from має 4838 унікальних значень з 6958 можливих, атрибут to – 3401 з 6958 можливих, а якщо поєднати, то ці два атрибути містять всього 7621 унікальні адреси з 13916 можливих (рисунок 3.9).

```
map_only_from, amount_of_from_addresses = mapAddressToNumber(tx.loc[:,"from"], dict(), 0)
amount_of_from_addresses
```

```
4838
```

```
map_only_to, amount_of_to_addresses = mapAddressToNumber(tx.loc[:,"to"], dict(),0)
amount_of_to_addresses
```

```
3401
```

```
map_both, amount_of_both_addresses = mapAddressToNumber( tx.loc[:,"from"], map_only_to, amount_of_to_addresses)
amount_of_both_addresses
```

```
7621
```

Рисунок 3.9 – Знаходження однакових адрес

Далі була застосувана операція відображення на атрибутах to та from (рисунок 3.10).

```
tx.loc[:,"from"] = tx.loc[:,"from"].map(map_both)
tx.loc[:,"to"] = tx.loc[:,"to"].map(map_both)
```

Рисунок 3.10 – Операція мапування

Після описаних дій було отримано датасет, що містить в собі тільки числові характеристики. На рисунку 3.11 зображені основні метрики кожного атрибуту, а на рисунку 3.12 – візуалізовано кожен з атрибутів.

```
1 tx.describe()
```

	block_number	value	nonce	from	to
count	6.958000e+03	6.958000e+03	6.958000e+03	6958.000000	6958.000000
mean	2.922471e+07	3.085716e+18	2.703725e+05	4487.047284	956.643863
std	7.169036e+04	4.525059e+19	5.883833e+05	1759.644025	1055.622445
min	2.872310e+07	0.000000e+00	0.000000e+00	19.000000	0.000000
25%	2.919847e+07	1.000000e+16	7.000000e+00	3404.000000	113.000000
50%	2.922471e+07	1.500000e+16	1.250000e+02	4269.500000	245.000000
75%	2.926915e+07	4.499725e+16	1.449250e+03	5926.750000	1791.750000
max	2.929995e+07	8.269500e+20	1.889176e+06	7620.000000	3400.000000

Рисунок 3.11 – Описові метрики атрибутів

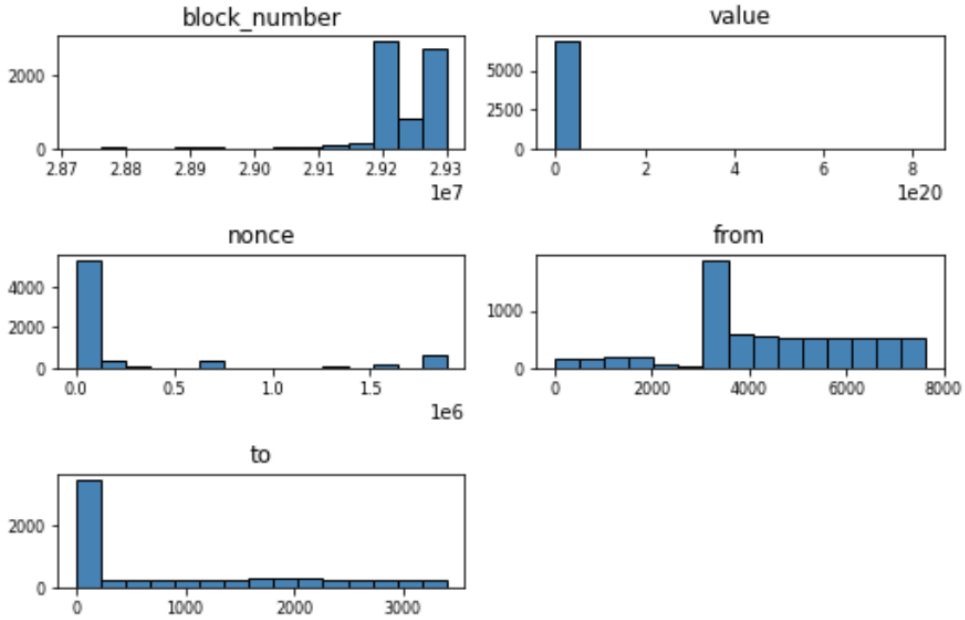


Рисунок 3.12 – Візуалізація значень атрибутів

Можна зробити висновки, що діапазон значень у атрибутів досить різний, що може призвести до викривлення результату, а отже, потрібно нормалізувати дані.

Для коректної нормалізації треба визначити, що означає кожна з характеристик.

Номер блоку являється свого роду відміткою часу, тому транзакції, що відбувались у одному або суміжних блоках, можуть бути пов’язані. Nonce позначає номер по порядку транзакції для гаманця, що її відправив. «Значення», тобто value, може позначати кілька речей:

– пересилання нативної валюти, тобто ефіру, з одного гаманця на інший. В такому разі value вказує на кількість цієї валюти;

– виконання певних дій на смарт-контракті, і тоді value – кількість валюти, потрібної для виконання цих дій;

– пересилання токенів, відмінних від нативної валюти – в такому разі, value дорівнює нулю.

Враховуючи вищеперераховане, було вирішено призначити атрибуту value значення 0, якщо value дорівнює нулю, і значення 1, якщо value не дорівнює нулю. Для інших же атрибутів застосувати нормалізацію у двох варіантах для порівняння, за формулами 3.1 та 3.2:

$$X = X - X_{min}, \tag{3.1}$$

$$X = \frac{X - X_{min}}{X_{max} - X_{min}}, \tag{3.2}$$

де X_{min} – мінімальне значення даного атрибуту;

X_{max} – максимальне значення даного атрибуту.

Код для мапування атрибуту value зображений на рисунку 3.13, для нормалізації за формулою 3.1 – на рисунку 3.14, а для нормалізації за формулою 3.2 – на рисунку 3.15.

```
updated_value=list()
for v in tx['value']:
   if(v==0):
      updated_value.append(0)
   else:
      updated_value.append(1)

tx['value'] = updated_value
```

Рисунок 3.13 – Операція мапування атрибуту value

```
def normalize(series):
   return series - min(series)

for col in tx.columns:
   if(col != "value"):
      tx.loc[:, col] = normalize(tx.loc[:,col])
```

Рисунок 3.14 – Операція нормалізації за формулою 3.1

```
def normalize2(series):
   return (series - min(series))/(max(series) - min(series))

tx_norm = tx.copy(deep=True)

for col in tx.columns:
   if(col != "value"):
      tx_norm.loc[:, col] = normalize2(tx.loc[:,col])
```

Рисунок 3.15 – Операція нормалізації за формулою 3.2

3.4 Кластерізація

3.4.1 Підготовка

Спочатку датасет був розбитий на датасети меншної розмірності з екземплярами, взятими випадковим чином (рисунок 3.16), оскільки кластерізація займала досить багато часу.

```
X25 = np.array(tx.sample(frac=0.25, random_state=0))
X5 = np.array(tx.sample(frac=0.5, random_state=1))
X75 = np.array(tx.sample(frac=0.75, random_state=1))
X = np.array(tx.sample(frac=1, random_state=1))

X25_norm = np.array(tx_norm.sample(frac=0.25, random_state=1))
X5_norm= np.array(tx_norm.sample(frac=0.5, random_state=1))
X75_norm = np.array(tx_norm.sample(frac=0.75, random_state=1))
X_norm = np.array(tx_norm.sample(frac=1, random_state=1))

print(X.shape, X25.shape, X5.shape, X75.shape)
```

```
(6958, 5) (1740, 5) (3479, 5) (5218, 5)
```

Рисунок 3.16 – Розбиття датасету на менші датасети

Далі було створено функцію для більш зручного застосування алгоритму кластерізації (рисунок 3.17).

```
def aPropagation(preference, random_state, damping, X_param):
    af = AffinityPropagation(preference=preference, random_state=random_state, damping = damping).fit(X_param)
    cluster_centers_indices = af.cluster_centers_indices_
    labels = af.labels_

    n_clusters_ = len(cluster_centers_indices)

    print("Estimated number of clusters: %d" % n_clusters_)

    print(
        "Silhouette Coefficient: %0.3f"
        % metrics.silhouette_score(X_param, labels, metric="sqeuclidean")
    )
    drawPlot(X_param, cluster_centers_indices, labels, n_clusters_)

    return (af, cluster_centers_indices, labels, n_clusters_)
```

Рисунок 3.17 – Функція кластерізації

Зображена функція кластеризує датасет, переданий в якості параметра, та повертає кількість знайдених кластерів, центри кластерів, коефіцієнт силуету (англ. Silhouette coefficient) та зображує знайдені кластери різними кольорами за допомогою функції drawPlot (рисунок 3.18).

Коефіцієнт силуету, близький до 1, вказує на те, що елементи вибірки знаходяться далеко від сусідніх кластерів [22]. Значення 0 вказує на те, що елементи вибірки знаходяться на межі між двома сусідніми кластерами або близько до неї. Від'ємні значення вказують на те, що ці елементи вибірки могли бути призначені неправильному кластеру.

Таким чином чим більший коефіцієнт силуєту, тим більш точно елементи визначених кластерів належать саме їм.

```
def drawPlot(dataset, cluster_centers_indices, labels, n_clusters_):
    plt.figure()
    plt.clf()
    i = 3
    j = 4
    colors = cm.rainbow(np.linspace(0, 1, n_clusters_))
    biggest_cluster = 1

    for k, col in zip(range(n_clusters_), colors):
        class_members = labels == k

        if(len(dataset[class_members]) >biggest_cluster):
            biggest_cluster = len(dataset[class_members])

        cluster_center = dataset[cluster_centers_indices[k]]
        plt.scatter(
            dataset[class_members, i], dataset[class_members, j], color=col, marker="."
        )
        plt.scatter(
            cluster_center[i], cluster_center[j], s=14, color=col, marker="o"
        )
        for x in dataset[class_members]:
            plt.plot(
                [cluster_center[i], x[i]], [cluster_center[j], x[j]], color=col
            )

    print("Size of a biggest cluster: ", biggest_cluster)
    plt.title("Estimated number of clusters: %d" % n_clusters_)
    plt.show()
```

Рисунок 3.18 – Функція зображення графіку

Для розширення можливостей візуалізації, були написані такі функції [23]:

– draw3DCluster – зображує найбільший кластер серед знайдених у трьохвимірній площині (рисунок 3.19);

– drawBiggestCluster – зображує найбільший кластер серед знайдених (додаток Б, лістинг 1);

– drawOneCluster5D – зображує найбільший кластер серед знайдених, беручи попарно кожні два атрибути (додаток Б, лістинг 2).

```
def draw3DCluster(n_clusters_, labels, cluster_centers_indices, dataset):
    fig = plt.figure(figsize=(8, 6))
    ax = fig.add_subplot(111, projection='3d')
    num_of_elements = 1
    selected_class_members = 0

    for k in range(n_clusters_):
        class_members = labels == k

        if(len(dataset[class_members]) > num_of_elements):
            num_of_elements = len(dataset[class_members])
            selected_class_members = class_members

    xs =  dataset[selected_class_members, 3]
    ys =  dataset[selected_class_members, 4]
    zs =  dataset[selected_class_members, 0]
    ax.scatter(xs, ys, zs, s=60, alpha=0.6, edgecolors='w')

    ax.set_xlabel('from')
    ax.set_ylabel('to')
    ax.set_zlabel('block')
```

Рисунок 3.19 – Функція зображення у трьох площинах

3.4.2 Тестування різних параметрів

Affinity Propagation працює шляхом передачі повідомлень між точками даних, доки не з’явиться хороший набір екземплярів (центрів кластерів) і відповідних кластерів.

Ці повідомлення оновлюються ітеративно на основі поточного стану алгоритму. Однак, якщо ці повідомлення будуть оновлені занадто раптово, алгоритм може постраждати від числових коливань, де значення шалено коливаються, не сходячись. Щоб цього не сталось, використовується демпфування (англ. dampling). Воно допомагає стабілізувати оновлення повідомлень, контролюючи, яка частина нового обчисленого значення включена в поточне значення. Коефіцієнт демпфування – це значення від 0,5 до 1, не включаючи одиницю (тобто [0,5, 1,0)).

Параметр переваги (англ. preference) визначає, наскільки ймовірно, що кожна точка даних буде обрана як екземпляр кластеру. Тобто це значення, присвоєне кожній точці даних, що вказує на її придатність служити частиною датасету.

На рисунок 3.20 зображено частина програми, що запускає алгоритм кластеризації для різних значеннях параметрів.

```
preference = [None,-50]
random_state=1
dampings =[0.99, 0.92, 0.75, 0.6]
X_param=X25
for p in preference:
  for d in dampings:
    print("preference: ", p, "\ndampling: ", d)
    aPropagation(preference=p, random_state=random_state, damping =d, X_param=X_param)
```

Рисунок 3.20 – Тестування кластерізації з різними параметрами

Були обрані різні значення параметрів серед допустимих значеннях, також були відкинуті ті, при яких кількість знайдених кластерів була 1 чи 0 – тоді алгоритм надавав помилку «Affinity propagation не збігається, ця модель може повертати вироджені центри кластерів і мітки» (рисунок 3.21).

Програма запускалась на нормалізованих та «менш нормалізованих» даних.

```
preference:  None
dampling:  0.99
C:\Users\Yuliia\AppData\Roaming\Python\Python38\site-packages\sklearn\cluster\_affinity_propagation.py:164: ConvergenceWarning: Affinity propagation did not converg
e and this model will not have any cluster centers.
  warnings.warn(
Estimated number of clusters: 0
```

Рисунок 3.21 – Приклад помилки

3.5 Результати кластерізації

Розглянемо найбільш наочні та відмінні одне від одного результати.

3.5.1 «Менш нормалізовані» дані

Як можна побачити на рисунках 3.22–3.25, кількість кластерів сильно відрізняється в залежності від параметрів. І хоча найбільший коефіцієнт силуету у випадку кластеризації на меншу кількість кластерів, таке розбиття в рамках задачі даної роботи викликає сумніви, адже навряд чи одна людина «захопила» половину транзакцій мережі.

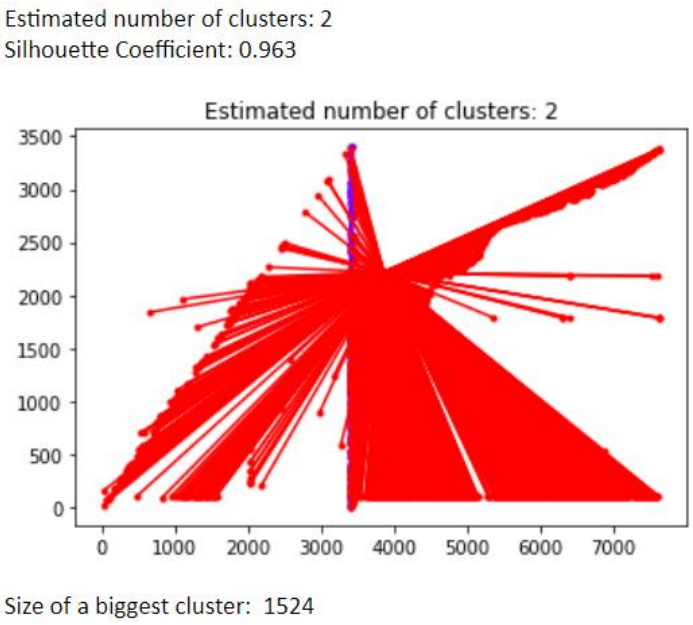
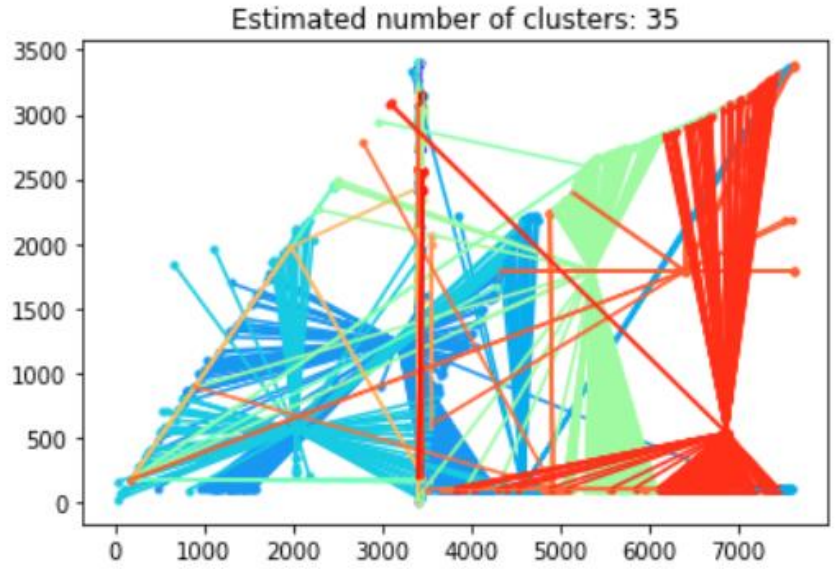


Рисунок 3.22 – Кластеризація з damping=0.99, preference=None

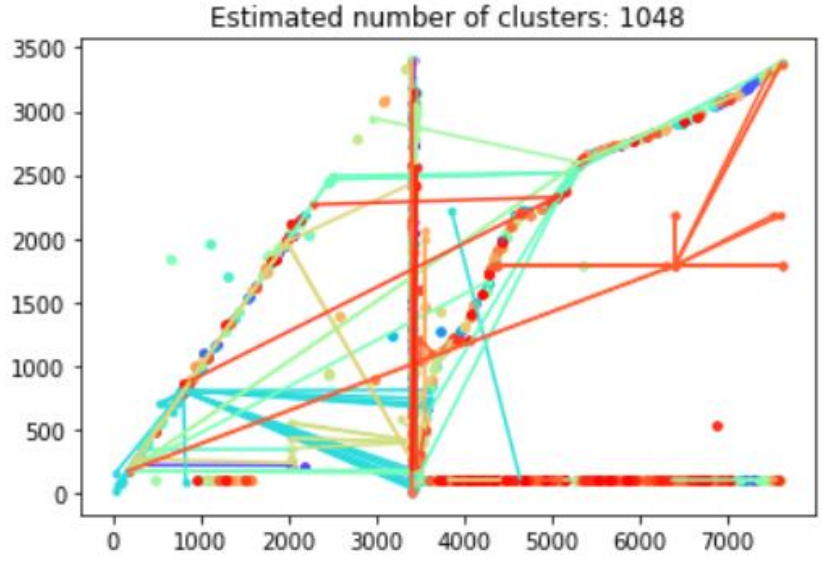
Estimated number of clusters: 35
Silhouette Coefficient: 0.722



Size of a biggest cluster: 327

Рисунок 3.23 – Кластеризація з damping=0.75, preference=None

Estimated number of clusters: 1048
Silhouette Coefficient: 0.259



Size of a biggest cluster: 65

Рисунок 3.24 – Кластеризація з damping=0.6, preference= None

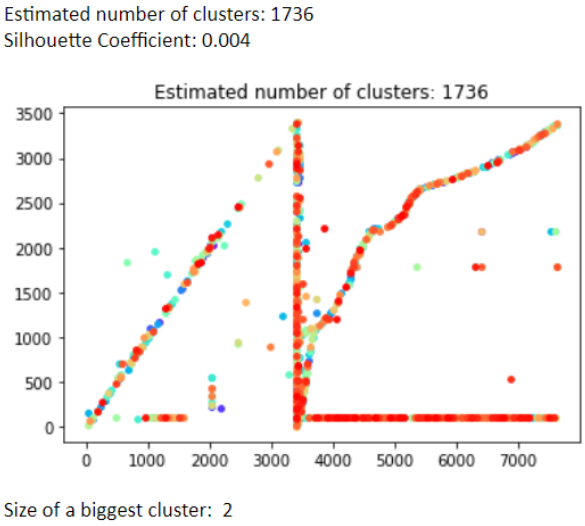


Рисунок 3.25 – Кластеризація з damping= 0.6, preference= -50

3.5.2 Нормалізовані дані

При запуску алгоритму з параметрами, описанимим у пункті 3.5.1, програма повертала помилку з написом, що знайдено лише один кластер. Тому програма була запущена з іншими значеннями параметрів (рисунки 3.26–3.29).

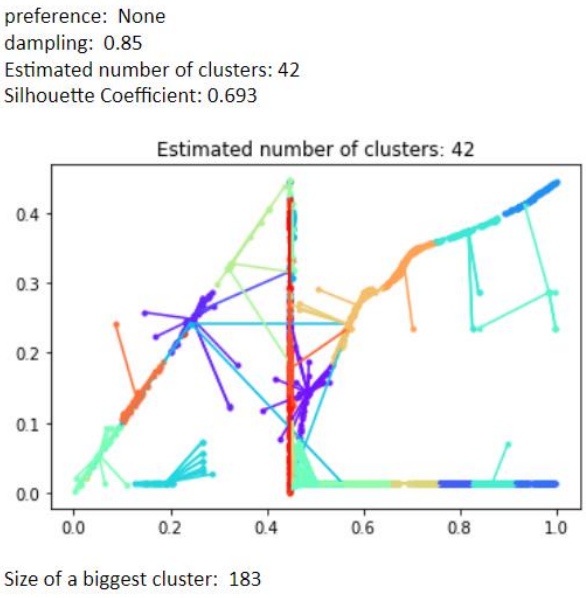


Рисунок 3.26 – Кластеризація з damping= 0.85, preference= None

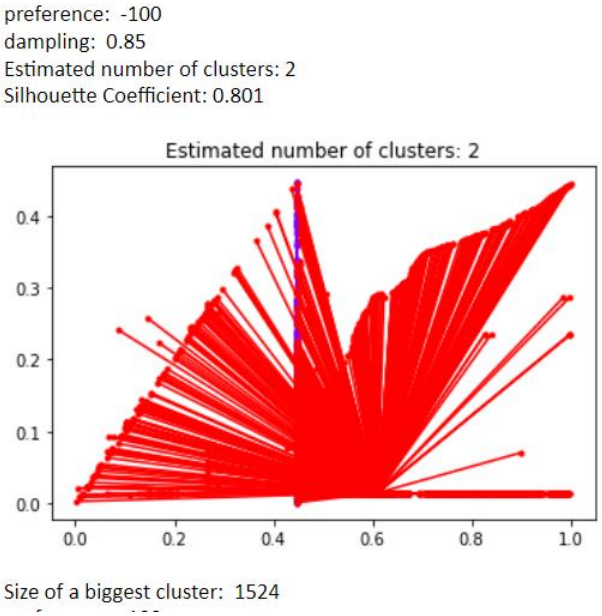


Рисунок 3.27 – Кластеризація з damping= 0.85, preference= -100

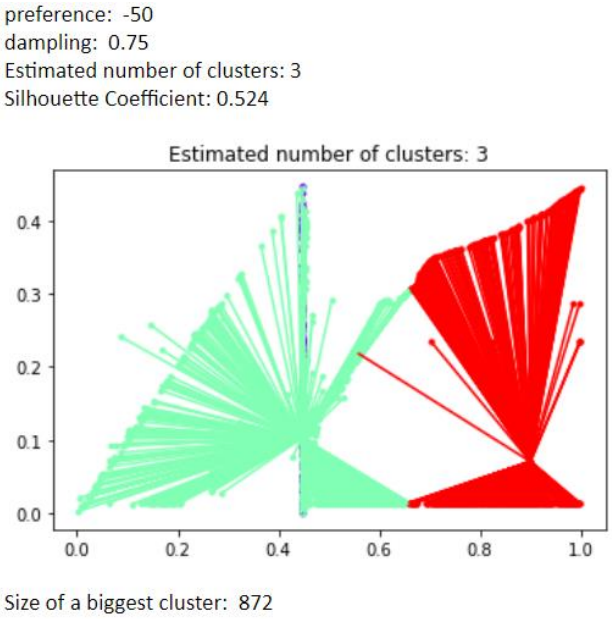


Рисунок 3.28 – Кластеризація з damping= 0.75, preference= -50

Як можна побачити, кількість кластерів теж сильно відрізняється в залежності від параметрів. Рисунки виглядають більш наочно, але і зрозуміти їх значення складніше.

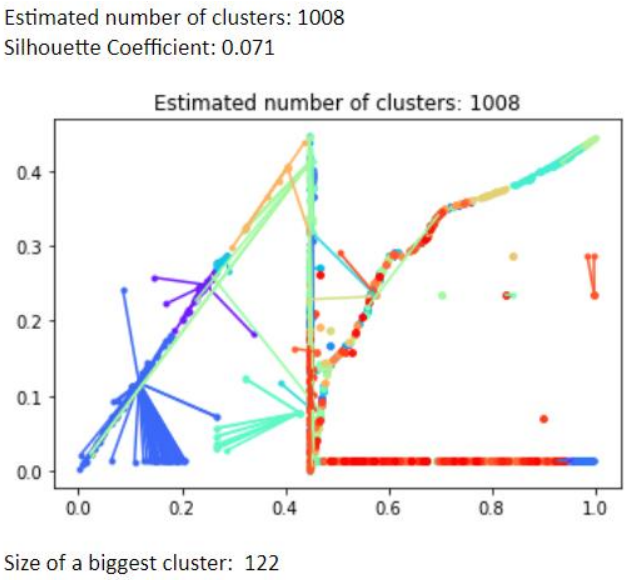


Рисунок 3.29 – Кластеризація з damping= 0.6, preference= -100

3.5.3 Поодинокі кластера

Розглянемо деякі з знайдених кластерів, щоб зрозуміти, що саме було знайдено і чи надає це корисну інформацію. Наприклад, на рисунок 3.30 зображено найбільший кластер, знайдений з параметрами damping=0.95, preference= -50. Він містить в собі 3 елементи, і показує, що з одного гаманця було відправлено транзакцію на 3 інших, при чому порядковий номер цих транзакцій знаходиться близько одне до одного (від 640821 до 640828). Також блоки,в яких були зроблені ці транзакції, знаходяться неподалік одне від одного – від 536993 до 537009. Виглядає цікаво з точки зору вирішення поставленої задачі.

```
[[537009      1 640826   3454    780]
 [537009      1 640828   3454    781]
 [536993      1 640821   3454    778]]
```

Рисунок 3.30 – Список елементів у знайденому кластері

Розглянемо даний кластер за допомогою написаних методів візуалізації. Спочатку подивимось, як виглядають попарно усі його атрибути (рисунок 3.31):

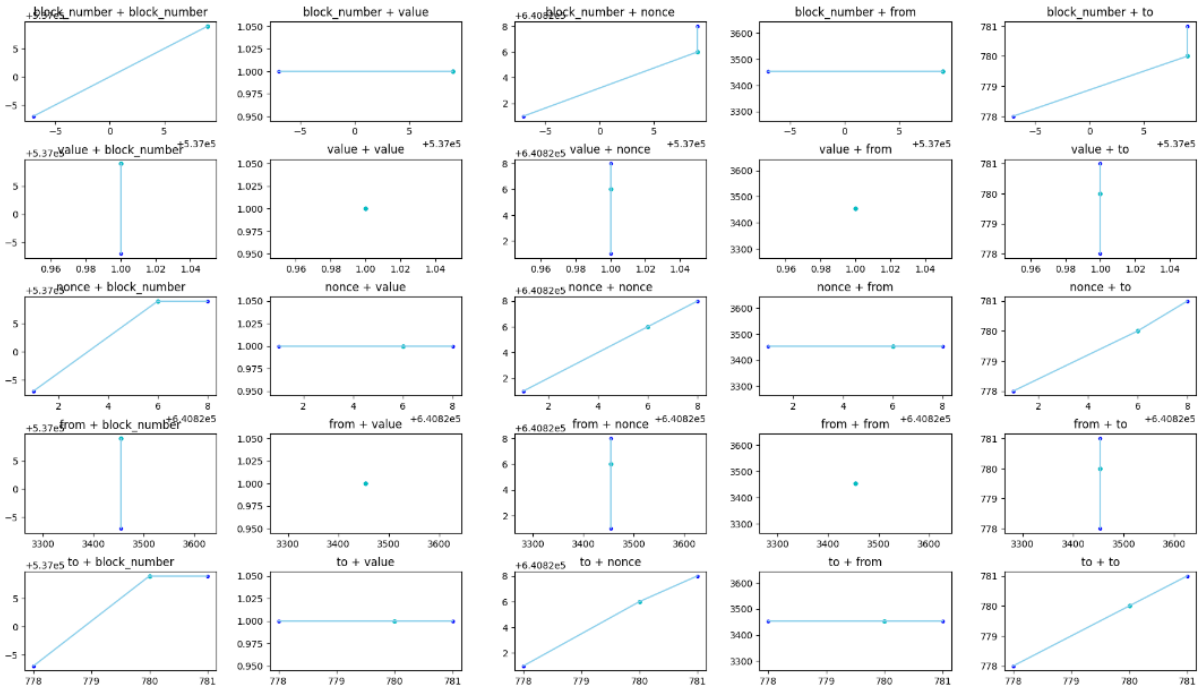


Рисунок 3.31 – Усі атрибути, зображені попарно

Більш-менш зрозумілу інформацію надають ті графіки, де присутній хоча б один атрибут з адресою.

Розглянемо графік, де не одній осі буде значення to, а на іншій from – рисунок 3.32. Він зображує те, що з однієї адреси було відправлено транзакції на три інші, що, відповідає дійсності. Форма у вигляді прямо дозволяє швидко помітити це.

Також можна подивитись на зображення у трьохвимірному просторі. Наприклад, атрибути from, to та block_number (рисунок 3.33) або атрибути from, to та nonce (рисунок 3.34) виглядають досить схоже.

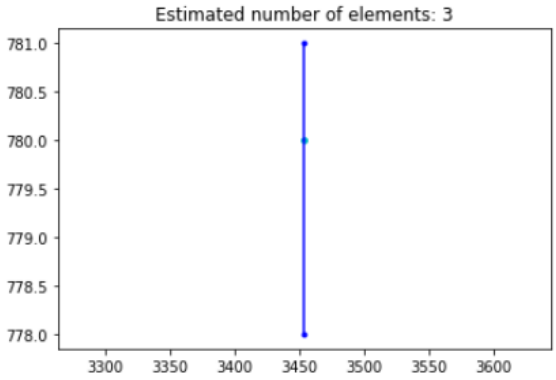


Рисунок 3.32 – Один кластер

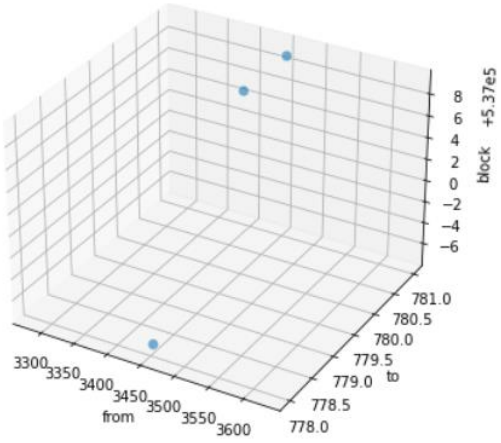


Рисунок 3.33 – Кластер у трьохвимірному просторі, block_number

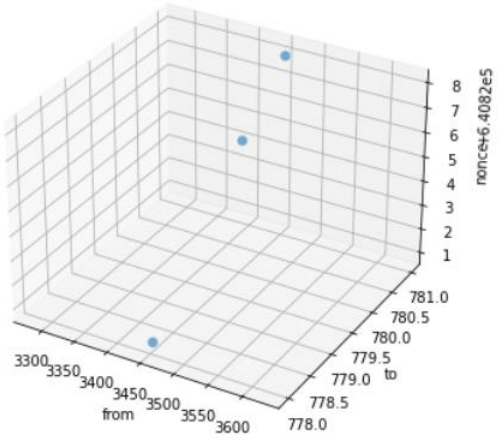


Рисунок 3.34 – Кластер у трьохвимірному просторі, nonce

Розглянемо кластер на 284 елементи, що було знайдено параметрами damping= 0.65, preference= None. На рисунку 3.35 можна побачити перелік елементів у ньому, і звязок між ними не сильно зрозумілий, окрім величини block_number.

```
1  cluster[0:20, 0:4]
```

```
array([[476587,    1,   128,  4027],
       [473610,    1,   184,  5933],
       [471213,    1,   320,  5564],
       [471814,    1,  1322,  5667],
       [475651,    1,   612,  6336],
       [471645,    1,     0,  5649],
       [473530,    1,   302,  5919],
       [475252,    1,     0,  6253],
       [474503,    1,   120,  6107],
       [473702,    1,   289,  5962],
       [473378,    1,    56,  5895],
       [475398,    1,   353,  6272],
       [470068,    1,   118,  5382],
       [472694,    1,    58,  5817],
       [471989,    1,    77,  5722],
       [472093,    1,    49,  5737],
       [471410,    1,    56,  5605],
       [471844,    1,   109,  5694],
       [470784,    1,     0,  5485],
       [475396,    1,   209,  6270]], dtype=int64)
```

Рисунок 3.35 – Перелік елементів у кластері

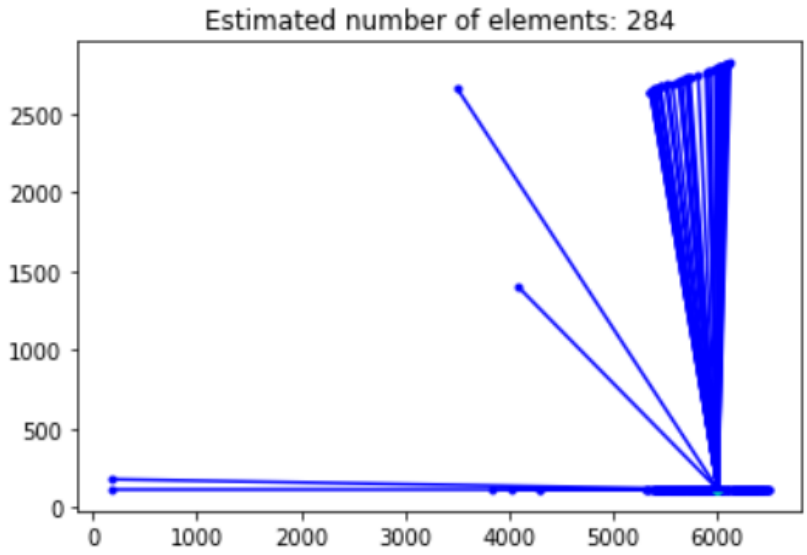


Рисунок 3.36 – Зображення елементів кластеру

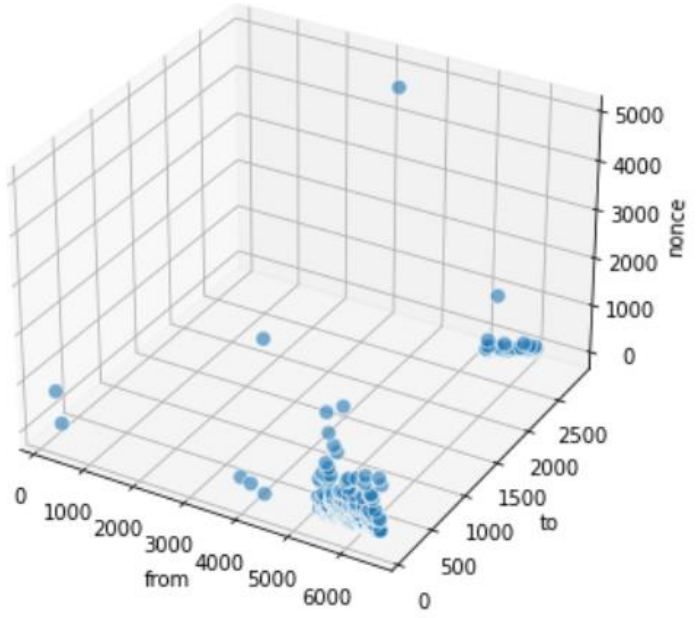


Рисунок 3.37 – Кластер у трьохвимірному просторі, nonce

На рисунку 3.36 можна побачити елементи кластеру у двовимірному просторі, наочно побачити звєязки важко. На рисунку 3.37 зображені елементи кластеру у трьохвимірному просторі, тут можна помітити певне скупчення, але винести з нього практичну цінність важко, потрібне більш детальне дослідження.

ВИСНОВКИ

У даній роботі було досліджено процес появи явища сибілів у мережах блокчейну та виявлення кластерів sybil-адрес на блокчейні за допомогою методів машинного навчання та аналізу даних, що має стати у нагоді проектам, що планують роздавати свої токени методом роздачі, та усім, кому цікава дана тема, оскільки питання децентралізації та децентралізованого керування протоколами є актуальним в світі технології блокчейн.

У результаті роботи у розділі 3.1 було показано, як можна отримати вихідні дані за допомогою безкоштовного сервісу Dune, а у зв’язці з мовою програмування Python зручно і швидко завантажити знайдені дані до середовища програмування (див. 3.2). Далі у розділі 3.3 було наведено приклад, як можна зробити попередню обробку та нормалізаію даних (формули 3.1, 3.2). Було проаналізовано, які метрики варто брати у розрахунок, а саме: номер блоку, value, nonce та адреси відправника та отримувача, де адреси являються основними критеріями. Продемонстровано роботу алгоритму Affinity Propagation з різними значеннями параметрів (див. 3.4.2), такими як preference та dampling. Для візуалізації було створено кілька функцій (рисунки 3.18, 3.19 та додаток Б, лістинг 1, лістинг 2), що дозволяють зобразити отримані кластери різним чином.

Так як отриманий датасет мав розмірність 5, візуалізація не була тривіальною задачею. Внаслідок досліджень усіх попарних комбінацій метрик (рисунок 3.31), було виявлено, що найбільш результативними атрибутами є адреси відправника та отримувача, а отже, саме зв’язки між ними було здебільшого зображено на графіках. Також для порівняння було створено графік з трьома вимірами (наприклад, рисунок 3.33).

На основі отриманих результатів, зображених у пунктах 3.5.1 – 3.5.3, можна зробити висновок, що можна знайти як дуже великі кластери, які

наврядчи будуть точними та виявлятимуть діяльність сибілів, так і невеличкі кластери, що дійсно показиватимуть підозрілу активність. Але, такі кластера потребують додаткової перевірки, такої як аналіз більшої кількості транзакцій за більший проміжок часу, з метою виявити нетипову поведінку або подальші закономірності в транзакціях між виявленимим адресами.

Перспективи розвитку кваліфікаційної роботи: спробувати різні алгоритми кластеризації для порівняння, дослідити більші об’єми даних та за більший проміжок часу для виявлення більшого числа зв’язків між різними адресами, дослідити адреси з виявлених кластерів в минулому чи майбутньому, з метою більш детально побачити історію взаємодій між знайденими адресами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1 Manoylov M. What is the difference between proof of work and proof of stake?. *The Block*. URL: https://www.theblock.co/learn/271536/what-difference-proof-of-work-pow-proof-of-stake-pos (дата звернення: 17.11.2023).

2 Why blockchains are trustless: replacing relationships with code - MINDFUL TECHNICS. *MINDFUL TECHNICS*. URL: https://mindfultechnics.com/why-blockchains-are-trustless-replacing-relationships-with-code/ (дата звернення: 18.11.2023).

3 What Is Crypto Airdrop And How Does It Work | WhiteBIT Blog. *WhiteBIT Blog*. URL: https://blog.whitebit.com/en/what-is-airdrop-crypto-meaning/ (дата звернення: 18.11.2023).

4 Kairon Labs | Airdrops and Retrodrops Decoded: A Comprehensive Guide. *Kairon Labs | Crypto Market Makers and Liquidity Providers*. URL: https://kaironlabs.com/blog/airdrops-and-retrodrops-decoded-a-comprehensive-guide (дата звернення: 18.11.2023).

5 Fáwọlé J., Ciattaglia L. Sybil Attack in Blockchain: Examples & Prevention. hacken.io. URL: https://hacken.io/insights/sybil-attacks/ (дата звернення: 15.11.2023).

6 Neeraj K., Shubhani A. Attacks on blockchain. scienceDirect. URL: https://www.sciencedirect.com/science/article/abs/pii/S0065245820300759 дата звернення: 16.11.2023).

7 Al-Qurish M., Alrubaian M. A prediction system of Sybil attack in social network using deep-regression model. ScienceDirect. URL: https://www.sciencedirect.com/science/article/abs/pii/S0167739X17300821 (дата звернення: 16.11.2023).

8 GitHub - ArbitrumFoundation/sybil-detection. GitHub. URL: https://github.com/ArbitrumFoundation/sybil-detection (дата звернення: 17.11.2023).

9 Nansen | Onchain Analytics for Crypto Investors & Teams. *Nansen | Onchain Analytics for Crypto Investors & Teams*. URL: https://www.nansen.ai (дата звернення: 25.12.2023).

10 hop-airdrop/src/data/blacklists/blacklist.ts at master · hop-protocol/hop-airdrop. *GitHub*. URL: https://github.com/hop-protocol/hop-airdrop/blob/master/src/data/blacklists/blacklist.ts (дата звернення: 25.12.2023).

11 hop-airdrop/ blob/master/src/data/eliminatedSybilAttackers.csv at master · hop-protocol/hop-airdrop. *GitHub*. URL: https://github.com/hop-protocol/hop-airdrop/blob/master/src/data/eliminatedSybilAttackers.cs дата звернення: 25.12.2023).

12 Offchain Labs. *Offchain Labs*. URL: https://www.offchainlabs.com (дата звернення: 25.12.2023).

13 Rita L. Louvain Algorithm. Medium. URL: https://towardsdatascience.com/louvain-algorithm-93fde589f58c (дата звернення: 16.11.2023).

14 URL: https://csc.knu.ua/media/study/asp/mod_probl_inf_tech_sys_analysis_ivohin/lecture/lec2.pdf (дата звернення: 01.01.2023).

15 *Видання ЧДУ імені Петра Могили*. URL: https://lib.chmnu.edu.ua/pdf/posibnuku/210/78.pdf (дата звернення: 26.12.2023).

16 scikit-learn: machine learning in Python – scikit-learn 1.5.0 documentation. *scikit-learn: machine learning in Python – scikit-learn 0.16.1 documentation*. URL: https://scikit-learn.org/stable/index.html (дата звернення: 02.12.2023).

17 Frey B. J., Dueck D. Clustering by passing messages between data points. 315th ed. Science, 2007.

18 " Dueck D., Frey B. J. Non-metric affinity propagation for unsupervised image categorization. *IEEE Xplore*. URL: https://ieeexplore.ieee.org/document/4408853 (date of access: 02.12.2023).

19 Demo of affinity propagation clustering algorithm. *scikit-learn*. URL: https://scikit-learn.org/stable/auto_examples/cluster/plot_affinity_propagation.html (дата звернення: 03.12.2023).

20 Welcome - Dune Docs. *Welcome - Dune Docs*. URL: https://docs.dune.com/home (дата звернення: 03.12.2023).

21 DuneSQL Overview - Dune Docs. *Welcome - Dune Docs*. URL: https://docs.dune.com/query-engine/index#sql-statement-syntax (дата звернення: 03.12.2023).

22 silhouette_score. *scikit-learn*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html (дата звернення: 04.12.2023).

23 Sarkar D. The Art of Effective Visualization of Multi-dimensional Data. *towardsdatascience*. URL: https://towardsdatascience.com/the-art-of-effective-visualization-of-multi-dimensional-data-6c7202990c57 (дата звернення: 09.12.2023).