`

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Інтелектуальне та програмне забезпечення системи інтелектуального аналізу графів Bitcoin»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Шелехов І.В.**

Студента групи ІН – 73 – 9 **Черненко Д.В.**

СУМИ 2021

`

ЗМІСТ

`

ВСТУП

Валюта є невід'ємною частиною повсякденного життя кожної людини. Перша валюта з'явилась близько 600 років до н.е. Її засновником вважають короля Лідії (теперішня Турція) Алітеса. Тогочасна валюта мала вигляд монети на якій було викарбовано голову лева. Минали століття і, згодом, форма валюти, яка раніше мала виглаяд монети, була замінена на банкноти. Ця подія мала місце приблизно у 1661 році н.е. З розвитком технологій, у 1946 році з'явилася перша кредитна картка. Розвиток у цій сфері не зупиняється і на сьогоднішній день: у світі існує тенденція з діджиталізації фінансів. Суть діджиталізації полягає у тому, що тепер гроші не зберігаються у фізичному вигляді, а є бітами та байтами інформації прив'язаної до рахунків клієнтів банка чи біржи. Таким чином маючи інформацію потрібну для аунтифікації, кожна людина має доступ до своїх фінансів з будь-якої точки планети.

Паралельно з поширенням тенденції використання електроних грошей збільшується кількість інтернет шахраїв, які полюють на кошти інших людей. Наприклад, у останні роки серед кібер-злочинців стають дедалі більш популярні програми-вимагачі (ransomeware). Їхня робота полягає у тому, щоб заблокувати доступ до комп'ютерної системи жертви або зашифрувати файли, які зберігаються на ній. У результаті, для продовження нормальної роботи чи розшифрування файлів подібні програми вимагають перевести певну кількість коштів на рахунок шахраїв. Зазвичай кіберзлочинці використовують криптогаманець та криптовалюту для одержання кошт, здобутих нелегальним шляхом. Існує припущення, що використання криптовалюти є абсолютно анонімним. Для його підтвердження чи спростування ціллю даної роботи було обрано інтелектуальне дослідження транзакцій криптовалюти, а саме "Bitcoin" транзакцій, за допомогою систем машинного навчання та математичних систем аналізу даних на шахрайську активність. Машинним навчанням є галузь інформаційних технологій, що застосовує алгоритми штучного інтелекту, що надають системам можливість автоматично "навчатись" та

`

самовдосконалюватись за допомогою раніше отриманих результатів без модулів які б були на це запрограмовані.

`

1 АНАЛІЗ ПРОБЛЕМИ. ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1Аналіз інструментів для вирішення поставленої задачі

1.1 Система біткоін

Біткоін являє собою кріптовалюту яка була створена у 2008 році невідомою групою людей під псевдонімом «Сатоші Накамото» і була введена в обіг у 2009, коли її імплементація була випущена як ПО з відкритим програмним кодом. Дана технологія - це децентралізована електронна валюта, яка не має центрального банку чи головної управляючої структури. Біткоін може бути надісланий через мережу біткоіна яка має топологію рівний-рівному без будь-яких проміжних точок. Транзакції верефікуються кінечними користувачами мережі, так званими «нодами», за допомогою криптографії та записуються до публічного сховища даних, яке несе назву «блокчейн». Біткоіни створюються як виногорода за процесс знаний як «майнінг». Вони можуть бути обміняні на інші валюти, продукти чи послуги. Блокчейн є публічним сховщем даних яке зберігає записи всіх транзакцій біткоіна, який імплементовано як ланцюг із блоків. Кожен блок зберігає хеш попереднього блоку, аж до так званого первинного блоку. Мережа біткоіну підтримується за допомогою нодів, на якіх запущено ПО біткоіну. Ноди мережі можуть верифікувати транзакції, добавляти їхні копії у локальне сховище даних, а потім транслювати цю інформацію до інших користувачів. Щоб досягти незалежної верифікації ланцюга, який зберігає інформацію про право власності, кожен користувач мережі зберігає власну копію блокчейну. Приблизно кожні 10 хвилин, нова група підтверджених транзакцій, яка називається блоком, приєднується до блокчейну та швидко публікується всім користувачам, без потреби центрального нагляду. Ця процедура визначає коли певний біткоін був витрачений, що дозволяє уникнут подвійного використання одної і тої ж грошової одиниці. Публічне сховище записів зберігає перекази актуальних

`

платежів або векселів, які існують окремо. Біткоїн є єдиним місцем де біткоїни можуть існувати у формі невитраченого результату транзакції.

1.1.2 Транзакції

Транзакції у системі біткоїн визачаються, використовуючи мову сценаріїв четвертого покоління. Транзакції складаються з одного чи декількох вхідних даних та одного чи декількох вихідних. Коли користувач надсилає біткоіни, він позначає кожну адресу для надсилання криптовалюти та її кількість у вихідних даних. Щоб уникнути “подвійного витрачання”, кожна одиниця вхідних даних повинна посилатися на попередню невитрачені вихідні данні у блокчейну. Використання множини вхідних даних відповідає використанню множині монету у грошовій транзакції. В тому випадку коли транзакції можуть мати множинні вихідні данні, тоді користувач може надсилати біткоїни зразу кільком адресатам. Так же як і в звичайних грошових транзакціях, у транзакціях біткоїну сума вхідних даних може перевищувати суму потрібну для здійснення платежів. У такому випадку, використовується допоміжний вихід який повертає кошти відправнику. Будь-які вхідні дані, що представляють собою “сатощі”, які не прописані у вихідних даних транзакції, стають комісією за транзакцію. Загальний приклад транзакції продемонстрований на рис. 1.1

`

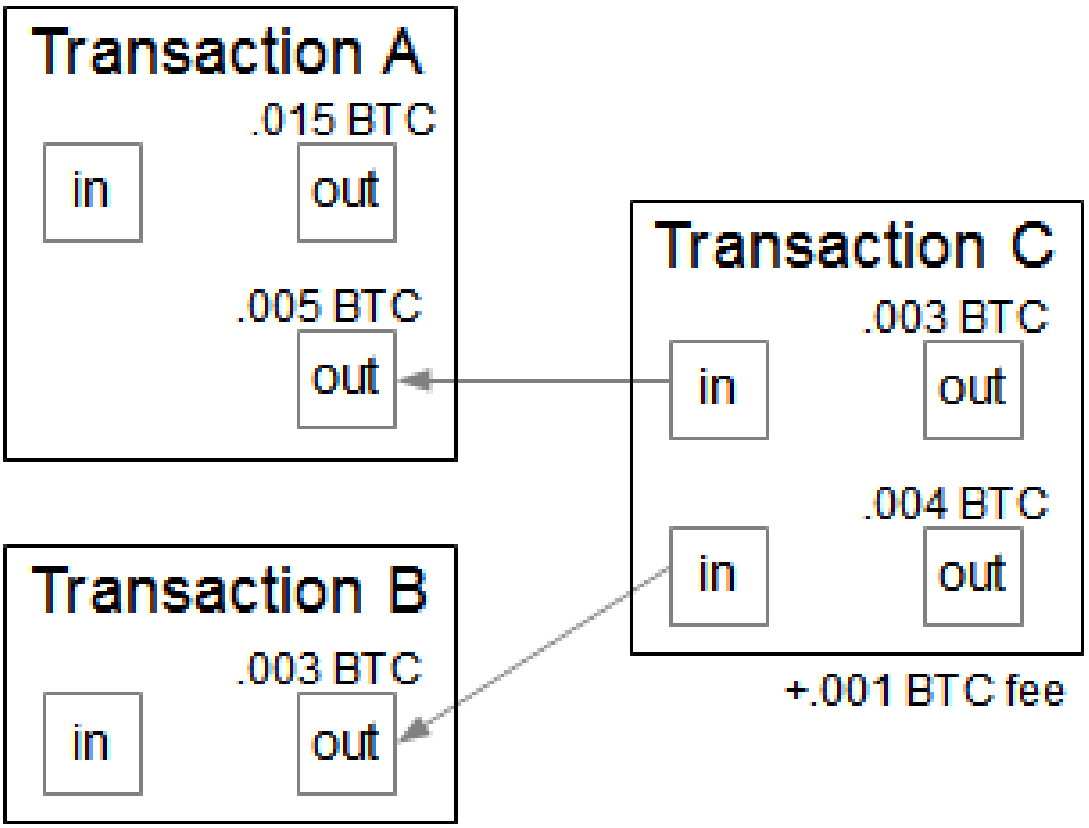


Рисунок 1.1 – Приклад транзакції

1.1.3 Комісії за транзакцію

Хоча комісія за транзакції є необов'язковою майнери можуть вирішувати які транзакції вони будуть обробляти і обирають ті, дохід від яких є максимальним. Також майнери можуть зважати на відношення комісії за транзакцію та на розмір сховища необхідний для її зберігання. Комісія за транзакцію вимірюється у одиниці виміру "sat/b", що означає кількість сатоши на один байт транзакції. Розмір транзакції залежить від кількості вхідних даних, що використовуються для її створення, та кількості вихідних даних.

1.1.4 Право власності

В системі блокчейн, біткоїни прив'язані до певних адрес. Створення біткоїн адреси процес вибору випадкового приватного ключа та визначення адреси. Ця процедура може бути виконана за долю секунди, але зворотня дія

`

(отримання приватного ключа з біткоїн адреси) вважається практично нездійсненим. Завдяки цій властивості користувачі можуть повідомляти іншим свою публічну адресу, не хвилюючись, що їхній приватний ключ може бути вкрадено. Більш того множина дійсних ключей настільки велика, що майже неможливо що хтось знайде пару ключем які використовуються і привласне кошти собі. Для того щоб мати можливість витратити біткоїни користувач має знати відповідний приватний ключ та за допомогою нього верифікувати транзакцію цифровим підписом. Мережа перевіряє підпис за допомогою публічного ключа. Якщо приватний ключ було втрачено, біткоїн-мережа не зможе надати доступ до коштів користувача іншим чином. Одже можна вважати що доступ до коштів заблокований назавжди.

1.1.5 Видобуток або майнінг

Майнінг це процес зберігання інформації за допомогою обчислювальної потужності. Майнери або добувачі допомагають зберігати послідовності, завершеності та незмінюваності системи біткоїн шляхом групування нещодавно виконаних транзакцій у блок, який транслюється всією мережою та підтверджується кінечними вузлами. Кожен блок містить хеш SHA-256 попереднього блоку, таким чином утворюючи ланцюг із блоків. Щоб бути прийнятим всією мережою біткоїн має містити “proof-of-work” (PoW). Система PoW вимагає від майнерів знаходження цифри яка носить назву “nonce”, за допомогою якої можна було б хешувати блок контенту, в наслідок чого результат є меншим ніж складність мережної цілі. Такий результат є легким у верифікації, але дуже складний у генеруванні. Кожні 2016 блоків (приблизно 14 днів, 10 хвилин на кожен блок), складність цілі змінюється базуючись на нещодавній статистиці мережі. Метою цього процесу є підтримка часу створення блоків у проміжку 10 хвилин. У даному випадку система автоматично адаптується до загальної кількості обчислювальної сили у мережі. Також система “proof-of-work”, зв’язуючи блоки один з одним, робить модифікацію блокчейну дуже складною. Як щоб атакуючий хотів підтвердити

`

модифікацію одного блока, йому б довелося змінити всі попередні блоки. З появою нових блоків у блокчейні, складність модифікації зростає за часом та кількістю попередніх блоків.

1.2 Постановка задачі

Метою даної практичної роботи є інтелектуальний аналіз шахрайських транзакцій системи біткоїн за допомогою інструментів штучного інтелекту.

`

2 ВИБІР МЕТОДУ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Машинне навчання

Машинне навчання це сфера яка вивчає здобуття корисної інформації з даних. Машинне навчання являє собою перетин таких сучасних наук як статистика, штучний інтелект, інформатика, прогностична аналітика, статистичне навчання. Впровадження машинного навчання у повсякденне життя стало розповсюдженою справою у наш час. Результатами робіт даної сфери користуються у багатьох напрямках, починаючи з рекомендацій стосовно вибору фільмів, доставки їжі та закінчуючи персоналізацією музичних вподобань та розпізнанням знайомих людей на фотографіях. Коли людина використовує сайти зі скалдною веб-архітектурою наприклад такі як Netflix, YouTube, Facebook скоріш за все кожен розділ цього сайту містить декілька моделей машинного навчання. Що стосується сфер які не є комерційно-прибутковими, то машинне навчання використовується у астрономії для знаходження віддалених планет, у біології та медицині для аналізу послідовностей ДНК та адаптації лікування злоякісних пухлин під особливості людини. На початку розвитку “розумних” додатків багато систем використовували “hardcoded” правила які являли собою конструкції коду наприклад “if”, “else”. Для кращого розуміння можна привести приклад: спам-фільтр задача якого полягає в тому, щоби відправляти відповідні листи, що надходять до користувача, до спам папки. В цьому випадку користувач може створити список якій містить словосполучення які допомагають спам-фільтру визначити чи є лист спамом. Такий підхід буде прикладом дизайну системи за допомого експертного рішення, для проектування “розумного” застосунку. Правила відбору створені вручну, в більшості ситуацій є прийнятними, зокрема у тих випадках коли людина розуміє процес моделювання. У іншому разі підхід з використання статично закодованих правил має два головних недоліки:

`

- Логіка застосунку звужує вибір варіанту рішення до певної задачі чи області. Навіть незначна зміна поставленої задачі, може спричинити повну реконструктурізацію системи.
- Проектування системи правил вимагає глибокого розуміння того, як рішення повинно бути прийнято людиною експертом.

Наприклад, приведений вище підхід не може бути використаний у розпізнанні облич людей. Але це не заважає будьякому новітньому смартфону розпізнавати обличчя користувачів. Не зважаючи на швидкий розвиток технологій, дана задача залишалася не вирішеною аж до 2001 року. Головна проблема полягала у тому що компьютер сприймає набір пік селів, з яких складається зображення, відмінно від людини. Ця відмінність у сприйняті робить майже неможливим для людини розроблення набору правил який би унікальноо ідентифікував обличчя на електронному зображенні. Використання ж машинного навчання, просто представляє собою програму з великою кількістю зображень облич людей, якої достатньо для алгоритму щоб визначити які характеристики потрібні для розпізнання обличчя людини.

Найбільш успішними типами алгоритмів машинного навчання є ті які автоматизують процес прийняття рішень шляхом узагальнення з доступних прикладів. В розглянутому вище прикладі, який несе назву “навчання під наглядом”, користувач надає алгоритму пари вхідних та необхідних вихідних даних. Таким чином алгоритм знаходить шлях для створення необхідного результату. У даному випадку алгоритм має змогу створити результат за введеними даними без допомоги людини. Повертаючись до нашого приклада з класифікацією спаму, використовуючи машинне навчання, користувач забезпечує алгоритм великою кількістю електронних листів, разом з інформацією яка повідомляє чи є листи спамом. Згодом, забезпечивши алгоритм новим листом, алгоритм вирахує припущення, чи є даний електронний лист спамом.

`

Алгоритми машинного навчання котрі навчаються з пар даних входу/виходу називаються керованими алгоритмами навчання, тому що “наглядач” надає правильні результати що до прикладів на яких навчається алгоритм. В той же час як створення набору даних для входу/виходу э трудоміською роботою, навчання під наглядом є більш зрозумілим та є легким у вимірюванні ефективності. Якщо ваш додаток може бути преведений до форми яка потребує навчання під наглядом і розробник може створити набір даних якій включає в себе бажані результати тоді, імовірніше, що машинне навчання є вирішенням поставленої задачі.

Приклади завдань які вирішуються за допомогою машинного навчання під наглядом:

- Індентифікація поштового коду, написаного вручну, на поштовому конверті. В даній задачі вхідними даними є зображення написаного поштового коду, а бажаним результатом є реальна цифра поштового коду.
- Визначення типу пухлини на основі медичного знімка. В даному випадку на вхід поступає зображення, а на вихід відповідь, чи є пухлина доброякісною або злоякісною. Також розробнику системи знадобиться експертне рішення що до всіх зображень пухлин.
- Визначення шахрайськіх фінансових транзакцій. У цьому випадку на вхід подаються дані транзакцій, а на виході отримаємо відповідь чи була транзакція здійснена шахраями чи ні. Дану задачу можно розглядати припустивши що власником системи з приняття рішень є організація, яка займається фінансовими питаннями, зберігає всі транзакції та фіксує звернення користувачів з приводу шахрайських дій.

Цікаво зазначити, що навіть якщо в ціх трьох прикладах вхідні/вихідні данні здаються дуже схожими, то методи збору інформації в значній мірі відрізняються між собою. Якщо зчитування поштового коду є кропіткою роботою, то принанні воно просте та дешеве. Що стосується отримання медичних знімків та діагностики, то вони вимагають не тільки спеціального

`

обладнання а й дорогої експертної оцінки. Також у роботі з медичними знімками може підніматися питання етики та приватної інформації. У випадку виявлення шахрайських фінансових транзакцій, набір даних, що використовується є набагато простішим. Це обумовлено тим, що користувачі фінансової установи самостійно нададуть провайдеру послуг необхідні вихідні данні, повідомивши про крадіжку. Все що повинен зробити провайдер це структурувати отриману інформацію у вигляді пар вхідних/вихідних даних підтверджених шахрайських транзакцій та помилкових.

Самостійні алгоритми або алгоритми без нагляду є типом алгоритмів при використанні яких відомими є тільки вхідні данні. Не зважаючи на те, що є багато успішних імплементацій машинного навчання котрі включають в себе дані алгоритми, вони є більш складними у сприйнятті та у аналізі їхньої ефективності.

Приклади реалізації алгоритмів які використовують метод навчання без нагляду:

- *Визначення тем у блоку постів*

Якщо людина має велику кількість текстових даних, в такому випадку, може з’явитись необхідність узагальнити інформацію наведену в ній та знайти відповідні теми. Людина може не знати заздалегідь які теми висвітлюються у тексті чи скільки таких тем може бути. В даному випадку користувач немає вихідної інформації.

- *Сегментування клієнтів на групи за схожими інтересами*

Часто впродовж процесу аналізу даних клієнтів виникає необхідність з’ясувати схожість між ними та між їхніми вподобаннями. Наприклад для веб-сайту що займається продажем клієнти можуть поділятися на наступні підгрупи: “батьки”, “читачі”, “геймери”. У такому разі власники веб-сайту не можуть знати в повній мірі скількі чи які групи можуть бути, тому система немає жодних вихідних даних.

`

- *Виявлення підозрілої активності на сайті*

Для знаходження багів та вразливостей веб-сайтів, є дуже зручним знайти схеми поведінки які відрізняються від норми. Кожен випадок підозрілої активності може бути унікальним і також розробники сайту можуть взагалі не мати зафіксованих випадків підозрілої поведінки. Оскільки що в цій задачі розробник може тільки спостерігати за мережевим трафіком та не знати як виглядає нормальна та підозріла поведінка рішенням є алгоритм машинного навчання без нагляду.

Для обох алгоритмів є важливим представлення дати у тій формі яка буде легкою для розуміння комп'ютера. У сфері машиного навчання поширеною практикою є представлення даних у вигляді таблиці. Кожна одиниця даних представляє собою рядок, а кожна властивість яка її описує це стовпчик. У випадку сегментування клієнтів розробник може описувати конжну людину за віком, статтю, датою створення акаунту, та частотою купівль. Що стосується зображень пухлин, то іхній опис може здійснюватись за допомогою відносної кількості сірого кольору у кожному пікселі, а тип може бути визначений за допомогою аналізу їхнього розміру, форми та кольору.

В машинному навчанні кожен запис або рядок називається зразком, а стовпці, характеристики які описують ці записи – властивостсями.

`

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

Існує багато інструментів для задач які вимагають аналізу значної кількості інформації. Наприклад, Matlab, середовище для розрахунків та аналізу, яке підтримує різні парадигми обчислювання. Дана система дозволяє маніпулювати матрицями, графічно демонструвати функції та дані, проводити імплементацію алгоритмів, розроблювати графічні інтерфейси для користувачів, взаємодіяти з іншими програмними засобами. Подібними інструментами є Maple, IDL, Wolfram Mathematica, та інші. Всі ці програмні застосунки є вкрай необхідними у таких сферах як машинне навчання, глибоке навчання, наука о даних, але в нашому випадку кількість функціоналу, який надають приведені вище програмні засоби, є надлишковим. Також важливими факторами є пропрієтарна архітектура застосунків та їхня ціна. Ми не можемо змінювати програмні засоби згідно потреб нашої задачі. Зважаючи на ці особливості, інструментом для вирішення поставленої задачі було обрано середовище Anaconda. Anaconda являє собою безкоштовний дистрибутив з відкритим програмним кодом для мов програмування Python та R, що має на меті спростити управління та розгортання пакетів для математичних та аналітичних розрахунків. Цей дистрибутив включає в себе аналітичні пакети сумісні з такими системами як Windows, Linux та MacOS.

3.2 Опис вхідних даних

Наша математична модель потребує великої кількості вхідних даних для навчання. Тому створення тестових даних вручну є дуже трудомісткою та тривалою роботою. Взявши це до уваги було вирішено використати сервіс “archive.ics.uci.edu”. Цей сервіс являє собою колекцію баз даних, теорії доменів та генераторів даних, що використовуються членами сфери машинного навчання для емперичного аналізу. Обрані нами дані являють собою матрицю, рядки та стовпці якої є зразками та характеристиками відповідно. Один зразок

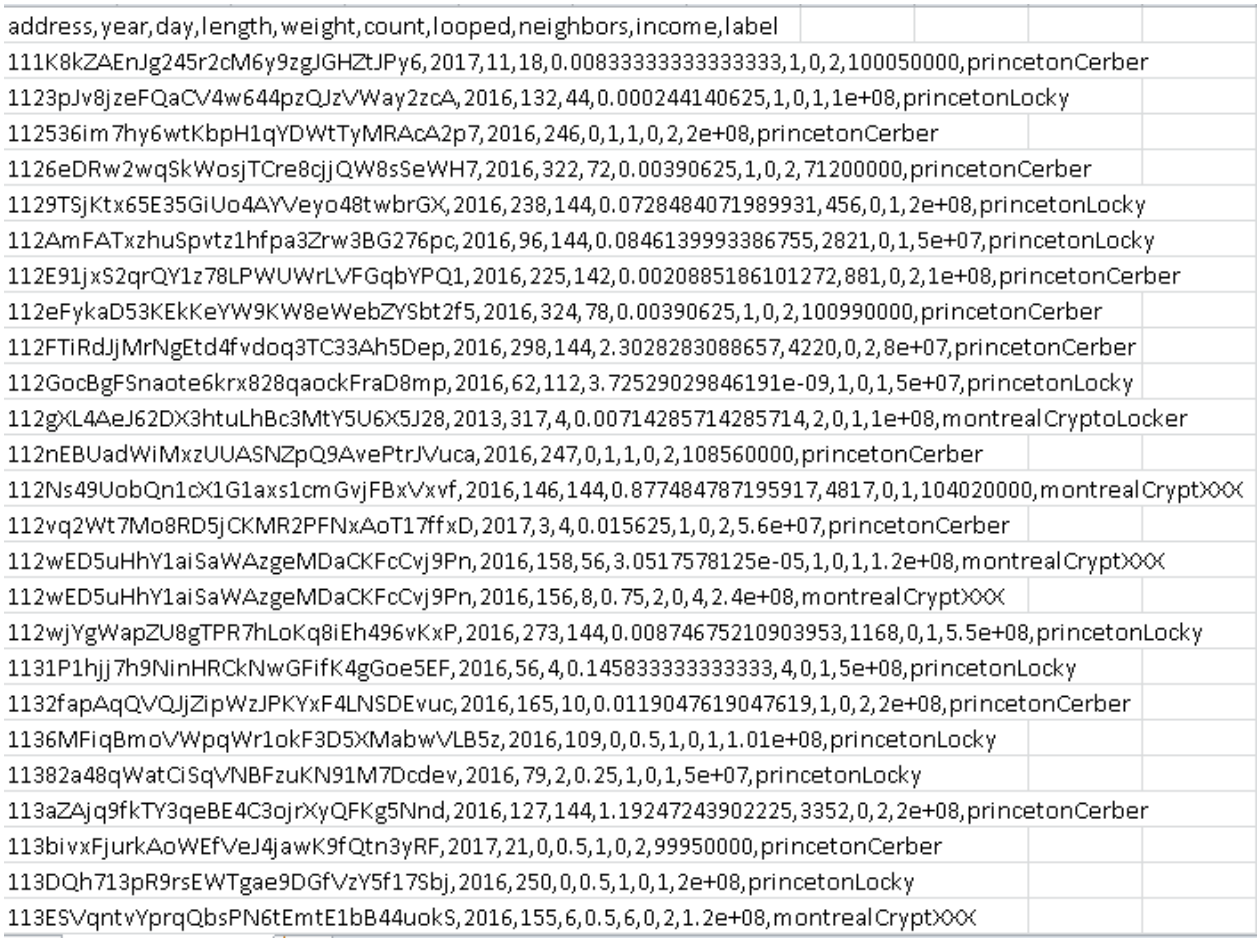
`

описує одну транзакцію біткоїна. Що стосується харатеристик то вони описани наступними атрибутами:

- Адреса: тип даних рядок
- Рік: цілочисельний тип даних
- День: цілочисельний тип даних
- Довжина: цілочисельний тип даних
- Ширина: числовий тип даних з плаваючою комою.
- Лічильник: цілочисельний тип даних.
- Цикл: цілочисельний тип даних.
- Сусідні: тип даних рядок.
- Прибуток: цілочисельний тип даних.
- Назва категорії: тип даних рядок.

Завантажений набір даних має наступний вигляд рис. 1.

`



```
address,year,day,length,weight,count,looped,neighbors,income,label
111K8kZAEnJg245r2cM6y9zgJGHZtJPy6,2017,11,18,0.00833333333333333,1,0,2,100050000,princetonCerber
1123pJv8jzeFQaCV4w644pzQJzVWay2zcA,2016,132,44,0.000244140625,1,0,1,1e+08,princetonLocky
112536im7hy6wtKbpH1qYDWtTyMRAcA2p7,2016,246,0,1,1,0,2,2e+08,princetonCerber
1126eDRw2wqSkWosjTCre8cjjQW8sSeWH7,2016,322,72,0.00390625,1,0,2,71200000,princetonCerber
1129TSjKtx65E35GiUo4AYVeyo48twbrGX,2016,238,144,0.0728484071989931,456,0,1,2e+08,princetonLocky
112AmFATxzhuSpvtz1hfpa3Zrw3BG276pc,2016,96,144,0.0846139993386755,2821,0,1,5e+07,princetonLocky
112E91jxS2qrQY1z78LPWUWrLVFGqbYPQ1,2016,225,142,0.0020885186101272,881,0,2,1e+08,princetonCerber
112eFykaD53KEkKeYW9KW8eWebZYSbt2f5,2016,324,78,0.00390625,1,0,2,100990000,princetonCerber
112FTiRdJjMrNgEtd4fvdoq3TC33Ah5Dep,2016,298,144,2.3028283088657,4220,0,2,8e+07,princetonCerber
112GocBgFSnaote6krx828qaockFraD8mp,2016,62,112,3.72529029846191e-09,1,0,1,5e+07,princetonLocky
112gXL4AeJ62DX3htuLhBc3MtY5U6X5J28,2013,317,4,0.00714285714285714,2,0,1,1e+08,montrealCryptoLocker
112nEBUadWiMxzUUASNZpQ9AvePtrJVuca,2016,247,0,1,1,0,2,108560000,princetonCerber
112Ns49UobQn1cX1G1axs1cmGvjFBxVxvf,2016,146,144,0.877484787195917,4817,0,1,104020000,montrealCryptXXX
112vq2Wt7Mo8RD5jCKMR2PFNxAoT17ffxD,2017,3,4,0.015625,1,0,2,5.6e+07,princetonCerber
112wED5uHhY1aiSaWAzgeMDaCKFcCvj9Pn,2016,158,56,3.0517578125e-05,1,0,1,1.2e+08,montrealCryptXXX
112wED5uHhY1aiSaWAzgeMDaCKFcCvj9Pn,2016,156,8,0.75,2,0,4,2.4e+08,montrealCryptXXX
112wjYgWapZU8gTPR7hLoKq8iEh496vKxP,2016,273,144,0.00874675210903953,1168,0,1,5.5e+08,princetonLocky
1131P1hjj7h9NinHRCkNwGFifK4gGoe5EF,2016,56,4,0.145833333333333,4,0,1,5e+08,princetonLocky
1132fapAqQVQJjZipWzJPKYxF4LNSDEvuc,2016,165,10,0.0119047619047619,1,0,2,2e+08,princetonCerber
1136MFiqBmoVWpqWr1okF3D5XMabwVLB5z,2016,109,0,0.5,1,0,1,1.01e+08,princetonLocky
11382a48qWatCiSqVNBFzuKN91M7Dcdev,2016,79,2,0.25,1,0,1,5e+07,princetonLocky
113aZAjq9fkTY3qeBE4C3ojrXyQFKg5Nnd,2016,127,144,1.19247243902225,3352,0,2,2e+08,princetonCerber
113bivxFjurkAoWEfVeJ4jawK9fQtn3yRF,2017,21,0,0.5,1,0,2,99950000,princetonCerber
113DQh713pR9rsEWTgae9DGfVzY5f17Sbj,2016,250,0,0.5,1,0,1,2e+08,princetonLocky
113ESVqntvYprqQbsPN6tEmtE1bB44uokS,2016,155,6,0.5,6,0,2,1.2e+08,montrealCryptXXX
```

Рисунок 1 – Вигляд набору даних

3.3 Аналіз та розробка

3.3.1 Передумови

Нещодавно результатом деяких досліджень є те що технологія біткоїн має певні обмеження приватності. Прикладами таких досліджень є аналіз стронніх ресурсів потенційних злодіїв і комбінування даної інформації з такими техніками як знаходження контексту і аналіз потоку. З іншого боку аналіз статичних характеристик транзакційнокго графу відповідає на питання поведінки середньостатистичного користувача, поведінки витрачання/придбання і потоком біткоїнів між різними акаунтами одного користувача. Розуміючи потрібність більш чіткої приватності у графі біткоїну було розроблено розширення до технології біткоїн яке імплементує протокол який дозволяє здійснювати повністю анонімні транзакції.

Модель загрози

`

4.1 Ціль атакуйочого: прив`язати реальні імена до транзакцій

Дійсним або підходящим іменем може бути реальне ім`я людини або її фамілія з публічних форумів або з будья-якого іншого публічного ресурсу. Ціллю є зв`язок множинних криптографічних ідентифікаторів з реальним користувачем.

4.2 Можливості атакуйочого

Поперше, нападних має доступ до всіх ресурсів публічної інформації, які включають в себе форуми, сайти пожертвувань, і публічні мережі через які будьхто може зібрати інформацію про біткоїн адресу яка була навмисно чи ненавмисно розголошена. В такому випадку атакуючий може зібрати дані о реальному імені, публічному ключі з сайтів.

По-друге, атакуючий може “підслухати” інформацію щодо даних транзакції у певних користувачів. Наприклад, атакуючий може підслухати повідомлення “Олександр це Олексій, я надіслав 5000 грн біткоїнів вчора ввечері”. Таким чином атакуючий може почути реальне ім`я та деяку інформацію що

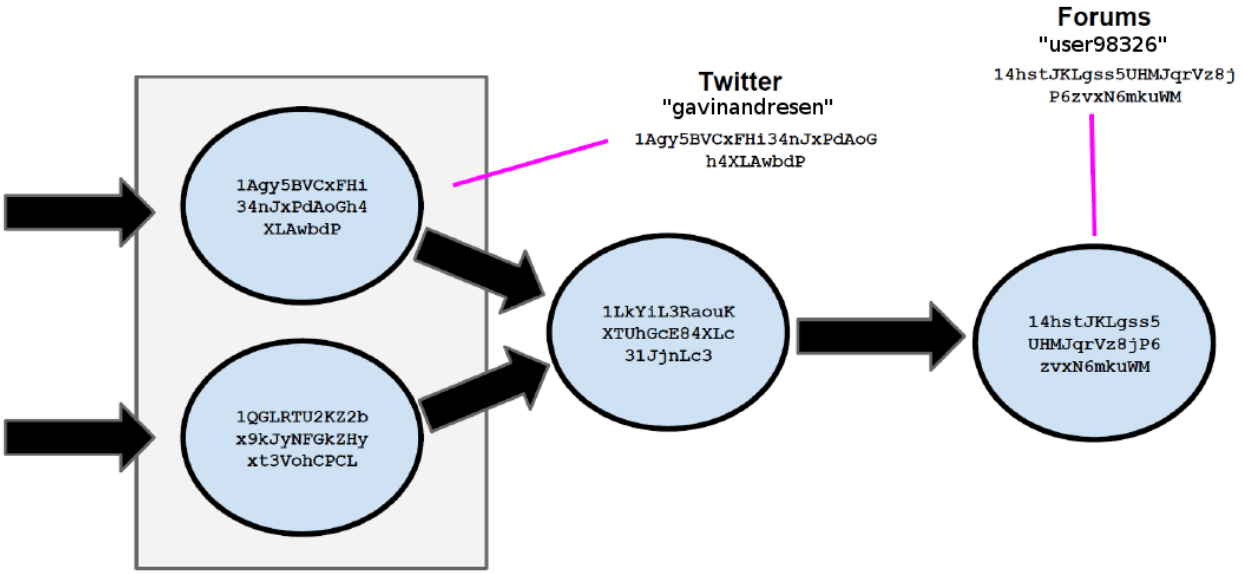


Рисунок 1.1 – Навмисне розкриття даних

`

Рисунок 1.2 – Ненавмисне розкриття даних

Як ми можемо бачити на рис. 1.1 показаний графік коментованих транзакцій, в тому випадку коли на рис 1.2 показано ненавмисне опублікування інформації що стосується біткоїн адреси власником онлайн магазину “Шовковий шлях”.

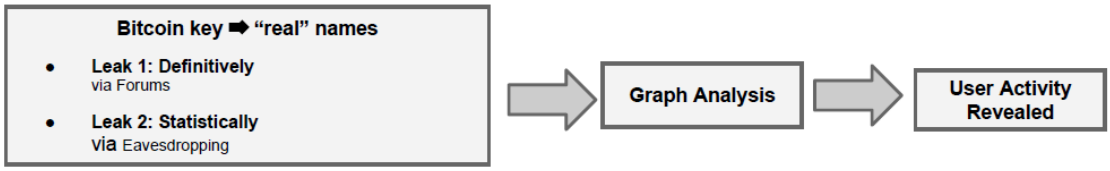


Рис 1.3 – Модель атакуйочого

5. Імплементація

В цьому пункті ми описуємо кілька кроків які зв’язані з встановленням інформаціх щодо біткоїн активності користувача шляхом використання публічно доступної інформації щодо транзакцій. Як описано в попередньому параграфі, нам важливі обидва статичний і дефенетивний підхіди.

5.1 Передоброблення інформації

Як попередник обох попередніх підходів, сирі транзакції мають бути витягнути із повного блокчейну. На поточний момент приблизно 275,000 блоків добуті в технології блокчейн. Кожен блок складається з приблизно сотін транзакцій. У наступному пункті буде розглянуто добування інформації з блокчейну.

5.1.1

`

В тому випадку коли стандартний клієнт біткоїну, а саме “biycoin-0.8.5” автоматично завантажує повний блокчейн у манері P2P, нами було зауважено, що для зниження високої частоти завантаження мережі можна використовувати технологію “torrent”. Осстанні блоки які залишились були завантажені автоматично шляхом оновлення даних за допомогою біткоїн клієнту, після індексації. В той час як у попередніх пунктах було використано розгалужену версію технології bitcointools, новіщі клієнти біткоїну проіндексували повний блокчейн використовуючи технологію LevelDB, замість того щоб робити, публічно доступну програму bitcointools застарілою. Замість цього ми використали програму “Armory”, щоб отримати інформацію з блокчейну, і написати клас обгортку, який отримує необхідну інформацію для побудови транзакційного графа.

5.1.2 Добуванян інформаціх з веб-ресурсів

Багато користувачів, а саме початківці, зацікавлені в тому щоб впровадити використання біткоїну в більш публічне товариство/більшість. Один із методів це зробити це спробувати заохочувати транзакції. Загальноприйнятою практикою у цьому напрямі є додавання біткоїн адреси у вигляді підпису до електроних листів або постів на форумах. В часності у постах на форумах, користувачі підтримують ком’юніті, наприклад з новим програмним обладнанням для майнінгу криптовалюти або у гайді щодо до налаштування комп’ютера для отримання біткоїнів. Вони залишають їхню біткоїн адресу у блоці підпису. Вони очікують у тому випадку коли читаці будуть задоволені інформацією яку отримали, вони отримають подяку у вигляді біткоїну. Така поведінка створює новий вектор атаки на анонімність системи блокчейн. Ми можемо з легкістю прив’язати інформацію щодо юзера до транзакцій у блокчейні. Ми використали програмний пакет мови програмування Python під назвою Scrapy, для того щоб отримати дані з сторінок у форумі. Також ми написали програму яка сканує bitcointalk.org в широту, на наявність блоків з підписами які можуть містити біткоїн адреси. Після отримання цього рядка і

`

проходження перевірки на коректність публічної адреси біткоїну (адреса біткоїна включає вбудований результат хеш-функції) для того щоб попередити велику кількість нодів які не можуть імовірнісно з'явитися у системі блокчейн.

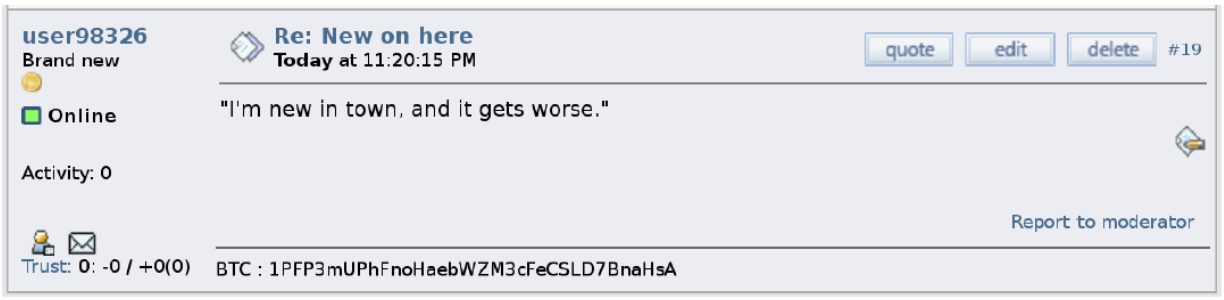


Рисунок 1.3 – Загальний приклад підпису на форумі якій включає публічну адресу біткоїна для пожертвувань

Нами було знайдено багату кількість користувачів форуму які можуть бути напряму зв`язані з їхніми публічними ключами в транзикційному графі. Ми запустили нашу програму на 30 годин. На протязі цього часу ми слідували 4 посиланням в глибину починаючи з головної сторінки. Таким чином наша програма покрила 44,086 сторінок та 89,088 постів які включають коректну біткоїн адресу. Поміж цього ми знайшли 2,322 унікальних користувача з 2,404 адресами які пройшли нашу перевірку.

5.2 Визначення транзакцій

У цьому пункті ми аналізуємо складність аналізу інформації стосовно часу транзакції та об'єму та співставлення їх з відповідними транзакціями у блокчейні. Наприклад якщо б ми підслухали як Олександр каже Олексію, я відправив тобі 10000 грн вчора у веччері, ми виявляємо складність знайти відповідну транзакцію у блокчейні. Представимо що припустили що біткоїн коливався у рамках 100 грн вчора і те що часова відмітка у транзакції актуальна у рамках 5 хвилин. Таким чином ми маємо декілька кандидатів-транзакцій для оцінювання. Продовжуючи цей приклад ми конвертуємо з грн у біткоїн використовуючи поточні показники на біржових ринках з BlockChain. Потім ми аналізуємо всі транзакції які опинилися у проміжку від 9999 грн. до 10100 грн та часом від 18:00 до 23:59.

`

Щоб узагальнити цей приклад, ми аналізуємо кожну транзакцію в цьому блокчейні, і створюємо час та грошову суму шляхом змінення кількості для того щоб побачити скільки іще траназакцій можуть відповідати цим критеріям. Рисунок внизу показує що що для обраної суми доларів і проміжку часу середня кількість транзакцій що буде відповідати будьякій певній критерії. З плином часу біткоїни стали більш популярними і частіше обмінюваними, таким чином ми будемо отримувати більше кандидатів з відповідної сумою доларів та критерієм часу за останні місяці.

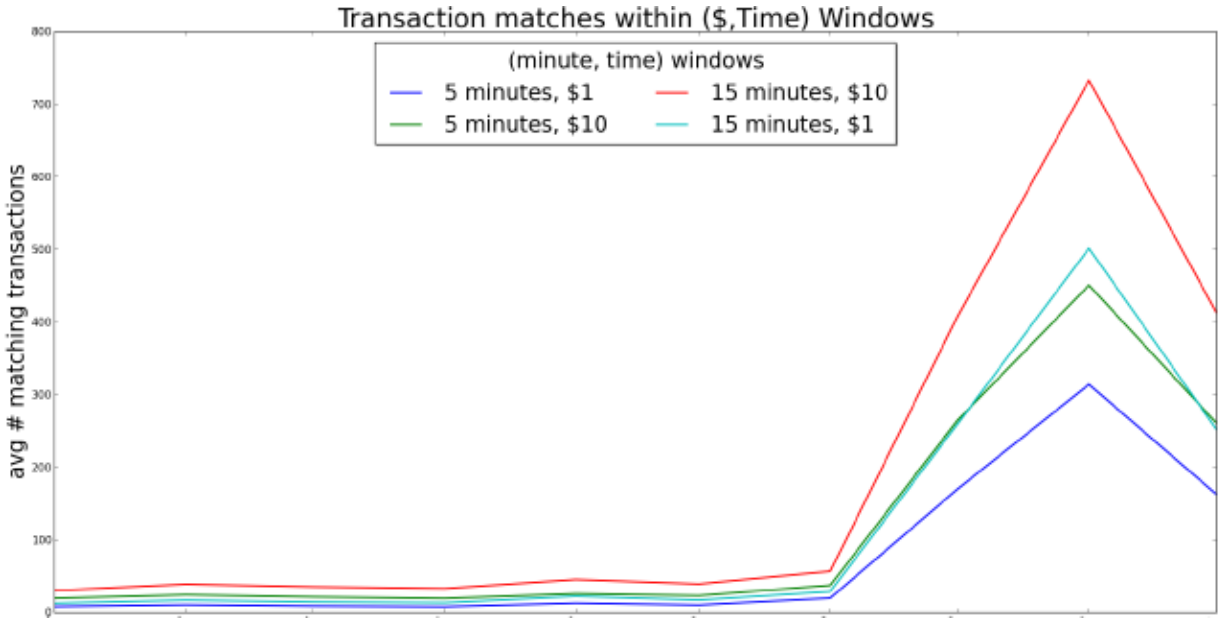


Рисунок 1.4 - Неоднозначність операцій, обумовлена неточним часом та приблизною вартістю доларів США

5.3 Аналіз графа

Ми розробили фреймворк для аналізу графа для того щоб деанонімізувати користвачів через інформацію яка була зібрана публічно, така наприкалад як з форумів, з даних транзакцій. Рисунок 1.5 відображає розроблений нами фреймворк

`

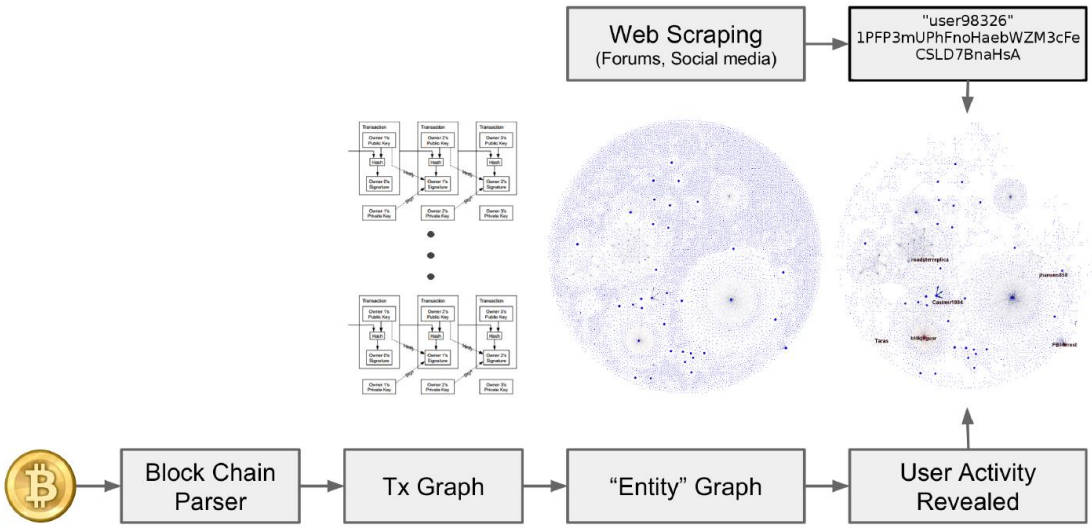


Рисунок 1.5 - Конвеєр аналізу графів, який був використаний для виявлення ідентичності користувача, будує графік мережі користувача як показано, і коментує користувачів на графіку результатами, скрапленими в Інтернеті.

5.3.1 Транзикційний граф

Тоді коли записи транзакції добуті з блокчейну, ми будуємо транзикційний граф який надає нам приблизний вигляд переміщення біткоїнів впродовж певного часу. Більш точніше, транзикційний граф це – напрправлений граф де ноди розшифровують публічну адресу анонімних користувачів, або сутностей, і направлений вектор представляє собою конкретну транзакцію від адреси джерела до адреси цілі. З тих пір як обидва і джерело і ресурс можуть у дозвільному порядку генерувати нові пари ключів які складаються з публічних та приватних ключів, для кожної наступної транзакції, багато публічних адрес-ключів можуть з`являтися один або декілька раз у транзикційному графі. Додатково, типічні транзакції в поточному блокчейні є транзакціями з множиними входами виходами. Для нашого дослідження ми побудували граф для 24 періодів на данний момент. Даний граф складається з 89,806 транзакцій, з 80,030 унікальними вершинами (або публічними ключами-адрессами). Ми також спроектували транзикційний граф проміжком 7 місяців, який складається з 1,669,728 транзакцій.

5.3.2 Граф користувача

`

В цьому підпункті ми консентруємость на першому дні коли граф був спроектований і описуємо наші знахідки на поточній активності, яка відображається вдразу після того як ми застосували алгоритм графа. Використовуючи граф, ми будуємо проксі направлений граф, який називається користувацьким графом, схожий до того який описаний у попередньому пункті

`

3.4 Початок розроблення програмної реалізації

Першим кроком э завантаження даних до середовища JupyterNotebook. Після цього за допомогою функції load_csv() приводимо їх до форми, яка потрібна для обробки. Далі ми розділяємо дані на дві групи: перша – для тренування математичної моделі, друга – для її перевірки. Після цього починаємо процес тренування за допомогою методу fit(). Наведений лістинг програми можна побачити далі:

```
import pandas as pd
bitcoin_dataset = pd.read_csv(“BitcoinDataHeist.csv”)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    bitcoin_dataset['data'], bitcoin_dataset['target'],
random_state=0)
print("X_train shape: {}".format(X_train.shape))
print("y_train shape: {}".format(y_train.shape))
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
X_new = pd.read_csv(“newbitcoin_transaction”)
print("X_new.shape: {}".format(X_new.shape))
prediction = knn.predict(X_new)

Predicted target name: [‘pricetonLocky’]
```

Повний ліснінг програми наведений у додатках А-І

ВИСНОВКИ

В ході дипломної роботи було проведено дослідження транзакцій криптовалюти під назвою “Біткоїн”. В проведеному дослідженні

`

використовувався метод машинного навчання. Було детально проаналізовані різні методи впровадження штучного інтелекту, технічна складова, математичне підґрунтя.

На закінчення ми показали, що, використовуючи кілька джерел загальнодоступної інформації через веб-скрап форумах та книзі транзакцій Біткойн, мережа транзакцій біткойн, як показано, не є повністю анонімною мус. Крім того, ми змогли зв’язати користувачів форуму з біткойнами лише з оригінальними вузлами Шовкового шляху єдиний посередник. Ми також змогли успішно знайти транзакції, які безпосередньо пов’язували зішкріб користувачі біткойн-форуму з такими відомими організаціями, як SatoshiDICE та Wikileaks, маючи на увазі, що вони могли мати справу з такими суб’єктами, які підтримуються або взаємодіють з ними.

`

СПИСОК ЛІТЕРАТУРИ

1. Fergal Reid and Martin Harrigan. An Analysis of Anonymity in the Bitcoin System. arXiv, 2011.
2. Dorit Ron and Adi Shamir. How did dread pirate roberts acquire and protect his bitcoin wealth?
3. [4] Dorit Ron and Adi Shamir. Quantitative analysis of the full bitcoin transaction graph. Cryptology ePrint
4. Archive, Report 2012/584, 2012. http://eprint.iacr.org/.
5. Рихтер Джеффри. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C#. 4-е изд. / Джеффри Рихтер. - Питер, 2013. - 896 с.
6. Стиллмен Э. Изучаем C#. Включая C#.NET 4.0 и Visual Studio 2010. 2-е издание / Э. Стиллмен., Дж. Грин - O'Reilly, 2012. - 689 с.
7. Троелсен Эндрю. Язык программирования C# 2010 и платформа.NET 4 / Эндрю Троелсен. - Вильямс, 2010. - 1392 с.
8. Фленов М. Библия C#, 2-е издание / М. Фленов. - БХВ-Петербург, 2011. - 560 с.
9. Шилдт Герберт. C# 4.0: Полное руководство / Герберт Шилдт. - Вильямс, 2011. - 1056 с.
10. Агуров П. C#. Сборник рецептов / П. Агуров. - БХВ-Петербург, 2007. - 432 с.
11. Павловская Т.А. C#. Программирование на языке высокого уровня: Учебник для вузов / Т.А. Павловская. - Питер, 2009. - 432 с.
12. Бьюли А. Изучаем SQL / Алан Бьюли. - Символ-Плюс, 2007. - 312 с.
13. Молинаро Энтони. SQL. Сборник рецептов / Энтони Молинаро. - O'Reilly, 2009. - 672 с.
14. Бураков П.В. Введение в системы баз данных. Учебное пособие / П.В. Бураков., В.Ю. Петров. - СПбГУ ИТМО, 2010. 129 с.

`

15. Опис офіційного API сервісу TheOldReader // Git Hub. [Електронний ресурс] - Режим доступу: https://github.com/theoldreader/api.
16. Офіційний сайт TheOldReader // TheOldReader. [Електронний ресурс]- Режим доступу: https://theoldreader.com/pages/apps.
17. Сайт Microsoft developer network // Microsoft developer network. [Електронний ресурс]- Режим доступу: https://msdn.microsoft.com/uk-ua.
18. Сайт Microsoft Virtual Academy // MVA [Електронний ресурс]- Режим доступу: https://mva.microsoft.com/.
19. Стандарт ECMA-404 JSON [Електронний ресурс] - Режим доступу: http://www.json.org/.
20. Сайт SQLite // SQLite Home Page [Електронний ресурс]- Режим доступу: http://sqlite.org/.
21. Сайт NewtonsoftJson // Json.NET [Електронний ресурс]- Режим доступу: http://www.newtonsoft.com/json.
22. Галерея додатків VisualStudio // NuGet [Електронний ресурс]- Режим доступу: https://www.nuget.org/packages/.
23. Microsoft Visual Studio // Материал з Вікіпедії - вільної енциклопедії. [Електронний ресурс] - Режим доступу: http://ru.wikipedia.org/wiki/ Microsoft_Visual_Studio.
24. Microsoft.NET. // Материал з Вікіпедії - вільної енциклопедії. [Електронний ресурс]- Режим доступу: http://uk.wikipedia.org/wiki/.
25. Microsoft_.NET.
26. NET Framework Developer Center // Головна сторінка. [Електронний ресурс]- Режим доступу: http://msdn.microsoft.com/ruru/netframework/ default.aspx.
27. Стандарти ECMA C# та CLI [Електронний ресурс]- Режим доступу: http://www.webcitation.org/6HZF1RbUz.
28. Стандарт ECMA-334 C# [Електронний ресурс]- Режим доступу: http://www.ecmainternational.org/publications/standards/Ecma-334.htm.

`

29. Документація компілятора C# Roslyn // Github [Електронний ресурс]- Режим доступу: https://github.com/dotnet/roslyn.
30. Мова програмування C# [Електронний ресурс]- Режим доступу: https://uk.wikipedia.org/wiki/C_Sharp.
31. XAML // Матеріал з Вікіпедії - вільної енциклопедії. [Електронний ресурс]- Режим доступу: https://ru.wikipedia.org/wiki/XAML
32. Детальний опис синтаксису XAML // Microsoft Developers Network. [Електронний ресурс]- Режим доступу: https://msdn.microsoft.com/ru-ru/library/ms788723(v=vs.110).aspx

`

Додаток А

```
# Import all packages
from tkinter import *
from tkinter.filedialog import askopenfilename
from tkinter import filedialog
import tkinter as tk

import pandas as pd
import csv

import matplotlib
matplotlib.use("TkAgg")
import matplotlib.pyplot as plt
from matplotlib import style
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure

# Setup font variables for buttons/labels
titleFont = ("Verdana", 36)
buttonFont = ("Verdana", 15)
helpFont = ("Verdana", 15)

style.use('ggplot')
```

`

Додаток В

```
# Main class and init function
class Main(tk.Tk):

        def __init__(self, *args, **kwargs):

                tk.Tk.__init__(self, *args, **kwargs)
                container = tk.Frame(self)
                container.pack(side="top", fill="both", expand = True)
                container.grid_rowconfigure(0, weight = 1)
                container.grid_columnconfigure(0, weight = 1)

                self.frames = {}
                for i  in (mainMenu, graphPage, helpPage):
                        frame = i(container, self)
                        self.frames[i] = frame
                        frame.grid(row= 0, column = 0, sticky = "nsew")

                self.display(mainMenu)
```

`

Додаток Г

```
class mainMenu(tk.Frame):                                                                    # Setup mainMenu

        def __init__(self, parent, controller):
                tk.Frame.__init__(self,parent)                                          # Init tkinter and get parent class

                label = tk.Label(self, text="Forensics Bitcoin Software", font=titleFont)
                label.configure(foreground="#000000", background="#ffffff")
                label.pack(pady=10,padx=10)

                # Graph button
                graph_btn = tk.Button(self, text="Display Graph", command=lambda: controller.display(graphPage), font=buttonFont)
                graph_btn.config(height = 2, width = 20, background = "#5F9ED4")
                graph_btn.pack(pady=(80,0))

                # Help button for the helpPage
                help_btn = tk.Button(self, text="Help", command=lambda: controller.display(helpPage), font=buttonFont)
                help_btn.config(height = 2, width = 20, background = "#5F9ED4")
                help_btn.pack(pady=(20,20))

                # Exit applicaton button
                exit_btn = tk.Button(self, text="Exit", command=self.exit, font=buttonFont)
                exit_btn.config(height = 2, width = 20,  background = "#5F9ED4")

                exit_btn.pack()
```

`

Додаток Д

```
class graphPage(tk.Frame):                                                  # Grap

        def __init__(self, parent, controller):
                tk.Frame.__init__(self, parent)
                label = tk.Label(self, text="Bitcoin Graph", font=titleFont)
                label.configure(foreground="#000000", background="#ffffff")
                label.pack(pady=(30,0),padx=0)

                # Ask for .csv file
                f = askopenfilename(title="Choose .csv file")

                # Return to main menu button
                back_btn = tk.Button(self, text="Back", command=lambda: controller.display(mainMenu))
                back_btn.config(height = 2, width = 20,  background = "#5F9ED4")
                back_btn.pack(pady=(0,0), padx=(0,0))

                # Configure graph on page
                pd.options.mode.chained_assignment = None
                new_f = pd.read_csv(f) #Load a csv file
                keep_col = ['value', 'timestamp'] #Get the arguments of the csv file needed
                bitcoins = new_f[keep_col]
                bitcoins.to_csv("newFile.csv", index=False) #new csv file
                bitcoins.head()

                # Convert date and time
                bitcoins['timestamp'] = pd.to_datetime(bitcoins.timestamp)
                bitcoins.index = bitcoins['timestamp']
```

`

Додаток Е

```
#!/usr/bin/env python3
import cx_Freeze
from cx_Freeze import *
import sys
import matplotlib

base = None

if sys.platform == 'win32':
        base = "Win32GUI"

executables = [cx_Freeze.Executable("forensics-bitcoin-software.py", base=base)]
cx_Freeze.setup (
        name = "ForensicsBItcoinSoftware",
        options = {"build_exe": {"packages": ["tkinter", "matplotlib", "pandas", "csv", "numpy"], "include_files":["electrum-history.csv"]}},
        versionn = "1.00",
        description = "Forensics Bitcoin applicaton",
        executables = executables
        )
```

`

Додаток Э

```
        def __init__(self, parent, controller):
                tk.Frame.__init__(self, parent)
                label = tk.Label(self, text="Help", font=titleFont)
                label.configure(foreground="#000000", background="#ffffff")
                label.pack(pady=(30,0),padx=10)

                back_btn = tk.Button(self, text="Back", command=lambda: controller.display(mainMenu))
                back_btn.config(height = 2, width = 20,  background = "#5F9ED4")
                back_btn.pack(pady=(0,0), padx=(0,0))

                label = tk.Label(self, text="Welcome to the Forensics Bitcoin software!\n\nThe main me
                        "The first button will display the graph. The second button will display the h
                        "To display new Bitcoin data to analyse, a csv file format is required. Add th
                label.configure(foreground="#000000", background="#ffffff")
                label.pack(pady=(30,0),padx=10)

# Call main app
app = Main()
app.mainloop()
```

`

Додаток Е

```
import lodash from "lodash/fp.js"
import axios from "axios"

const { get, map, flow } = lodash

export class CryptoCompareAPI {
  // API Urls
  static baseURL() {
    return "https://min-api.cryptocompare.com/data/"
  }

  static apiKey() {
    return process.env.CRYPTOCOMPARE_API_KEY
      ? `&api_key=${process.env.CRYPTOCOMPARE_API_KEY}`
      : ""
  }

  static History(time, coin, currency, past) {
    return (
      `${CryptoCompareAPI.baseURL()}${time}?fsym=${coin}` +
      `&tsym=${currency}&limit=${past + 50}&e=CCCAGG` +
      CryptoCompareAPI.apiKey()
    )
  }

  static Current(coin, currency) {
    return (
      `${CryptoCompareAPI.baseURL()}` +
      `price?fsym=${coin}&tsyms=${currency}` +
```

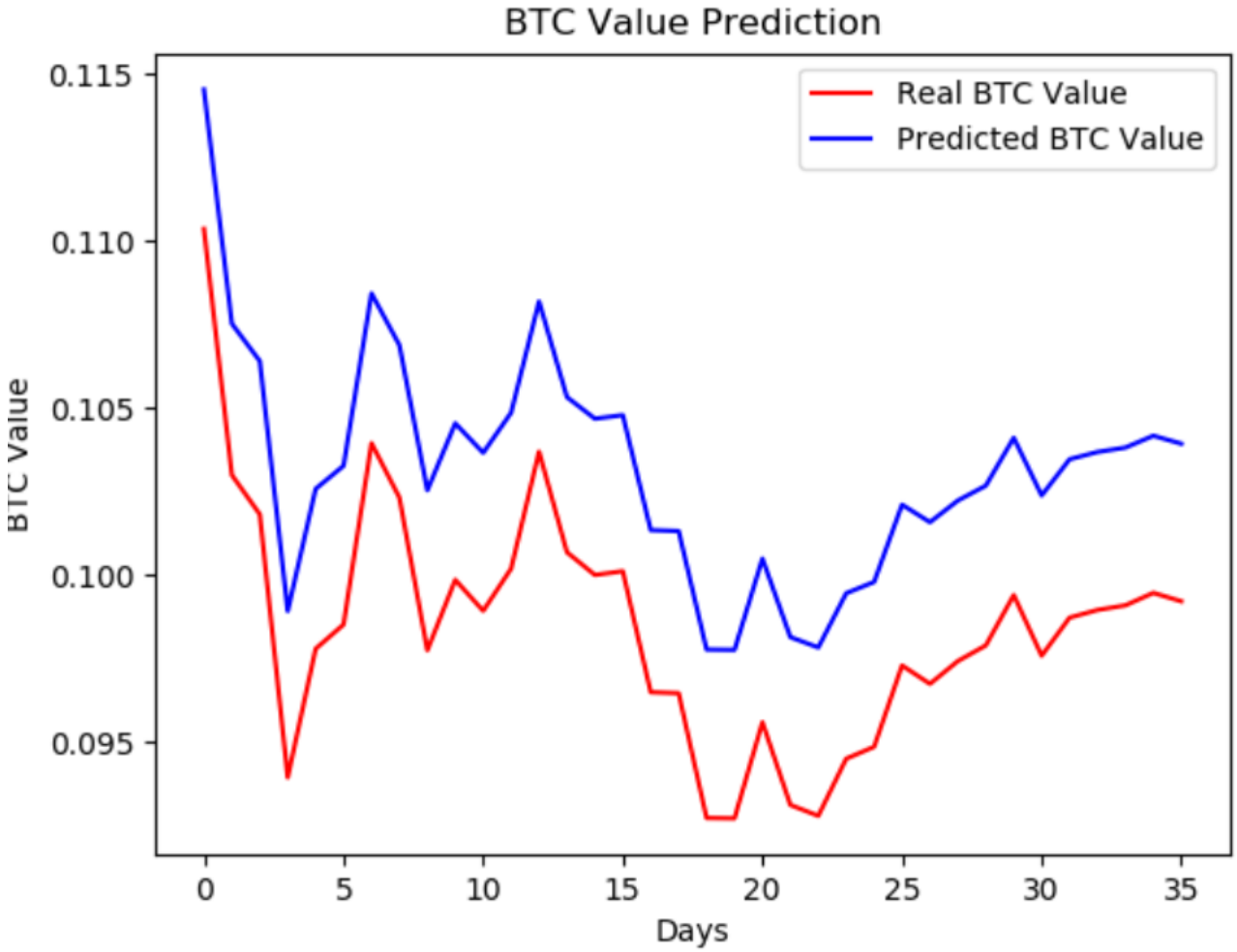
`

Додаток Ж

```
    return (
      `${CryptoCompareAPI.baseURL()}` +
      "all/coinlist" +
      CryptoCompareAPI.apiKey()
    )
  }

  static async fetchCoinList() {
    return flow(
      get("data.Data"),
      map("FullName")
    )(await axios.get(CryptoCompareAPI.CoinList()))
  }

  static async fetchCoinHistory(time, coin, currency, past) {
    const { data } = await axios.get(
      CryptoCompareAPI.History(time, coin, currency, past)
    )
    if (data.Response == "Error") {
      console.log(data.Message)
      process.exit(1)
    }
    return data.Data.map((d) => d.close)
  }

  static async fetchCoinPrice(coin, currency) {
    return (await axios.get(CryptoCompareAPI.Current(coin, currency))).data
  }
}
```

`

Додаток 3


BTC Value Prediction
Real BTC Value
Predicted BTC Value
BTC Value
0.115
0.110
0.105
0.100
0.095
0
5
10
15
20
25
30
35
Days

`

Додаток 3

```
"""
Definition of views.
"""

from django.shortcuts import render
from django.http import HttpRequest
from django.template import RequestContext

from app.BitcoinAPI.AddressUtils import *
from blockchain.blockexplorer import *
from datetime import datetime
from networkx.readwrite.json_graph import *
import json

def home(request, bitcoin_address=0, max_tx=50, depth_limit=5, branch_limit=5):
    """Renders the home page."""
    assert isinstance(request, HttpRequest)

    max_tx = int(max_tx)
    depth_limit = int(depth_limit)
    branch_limit = int(branch_limit)

    root_address = {}
    ingraph = None
    outgraph = None
    graph = None
    graph_data = None
```

`

Додаток И

```
if graph:
    graph_data = node_link_data(graph)
    graph_data['links'] = [
    {
        'source': graph_data['nodes'][link['source']]['id'],
        'target': graph_data['nodes'][link['target']]['id'],
        'value': link['value'],
        'color': link['color']
    }
    for link in graph_data['links']]
    graph_json = json.dumps(graph_data)

return render(
    request,
    'index.html',
    {
        'title':'Home Page',
        'year':datetime.now().year,
        'bitcoin_address': bitcoin_address,
        'graph_json': graph_json,
        'max_tx': max_tx,
        'depth_limit': depth_limit,
        'branch_limit': branch_limit
    }
)
```

`

Додаток І

```
(function ($) {
var $jQval = $.validator,
adapters,
data_validation = "unobtrusiveValidation";
function setValidationValues(options, ruleName, value) {
options.rules[ruleName] = value;
if (options.message) {
options.messages[ruleName] = options.message;
}
}
function splitAndTrim(value) {
return value.replace(/^\s+|\s+$/g, "").split(/\s*,\s*/g);
}
function escapeAttributeValue(value) {
// As mentioned on http://api.jquery.com/category/selectors/
return value.replace(/([!"#$%&'()*+,./:;<=>?@\[\\\]^`{|}~])/g, "\\$1");
}
function getModelPrefix(fieldName) {
return fieldName.substr(0, fieldName.lastIndexOf(".") + 1);
}
function appendModelPrefix(value, prefix) {
if (value.indexOf("*.") === 0) {
value = value.replace("*.", prefix);
}
return value;
}
function onError(error, inputElement) {  // 'this' is the form element
```