НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

До захисту допущено
Завідувач кафедри

_____ **Дмитро ЛАНДЕ**
(підпис)

«20» червня 2025 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи, технології та математичні методи кібербезпеки»
спеціальності 125 «Кібербезпека»

на тему: Класифікація криптовалютних адрес за транзакційною поведінкою на основі методів машинного навчання як інструмент блокчейн форензики

Виконав (-ла): здобувач вищої освіти **IV** курсу, групи ФБ-12
(шифр групи)

Карабінський Василь Дмитрович
(прізвище, ім'я, по батькові) (підпис)

Керівник доктор філософії з кібербезпеки, асистент кафедри ІБ

Полуциганова Вікторія Ігорівна
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент доктор технічних наук, професор кафедри ММЗІ

Кудін Антон Михайлович
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Здобувач вищої освіти _____
(підпис)

Київ – 2025 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 125 «Кібербезпека»
Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Дмитро Ланде
(підпис)

«20» червня 2025 р.

ЗАВДАННЯ
на дипломну роботу здобувачу вищої освіти

Карабінський Василь Дмитрович
(прізвище, ім'я, по батькові)

1. Тема роботи Класифікація криптовалютних адрес за транзакційною поведінкою на основі методів машинного навчання як інструмент блокчейн форензики

науковий керівник роботи Полуциганова Вікторія Ігорівна, доктор філософії з кібербезпеки, асистент кафедри ІБ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «26» травня 2025 р. № 1761-с

2. Термін подання здобувачем дипломної роботи 13.06.2025 р.

3. Вихідні дані: Літературні джерела, які містять інформацію щодо блокчейн форензики, механізмів роботи криптовалют та застосування машинного навчання.

4. Зміст роботи: Аналіз наявних літературних джерел з блокчейн форензики, виділення особливостей та труднощів цієї дисципліни, розробка методики по класифікації криптовалютних адрес за

транзакційною поведінкою, збір реальних адрес відомих сутностей за п’ятьма категоріями(біржі, майнінгові пули, гемблінг платформи, даркнет маркети, адреси пов’язані з ransomware), збір транзакційної інформації для зібраних адрес, розробка вибірки ознак, класифікація методами машинного навчання, оцінювання моделей, інтерпретація отриманих результатів.

5. Орієнтовний перелік ілюстративного матеріалу: презентація

6. Дата видачі завдання 12 вересня 2024 року

Календарний план

з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Визначення теми дипломної роботи	12.09.2024 –27.09.2024	Виконано
2	Збір літературних джерел по темі	30.09.2024 – 10.10.2024	Виконано
3	Аналіз зібраної літератури, визначення мети роботи	15.10.2024 – 01.12.2024	Виконано
4	Визначення основних способів блокчейн форензики, визначення проблем. Робота над першим розділом	20.01.2025 – 14.03.2025	Виконано
5	Розробка методики по класифікації криптовалютних адрес. Робота над другим розділом	20.03.2025 – 10.04.2025	Виконано
6	Проходження переддипломної практики	14.04.2025 – 18.05.2025	Виконано
7	Програмна реалізація розробленої методики. Робота над 3 розділом	20.05.2025 – 10.06.2025	Виконано
8	Передзахист дипломної роботи	13.06.2025	Виконано
9	Захист дипломної роботи	20.06.2025	

Здобувач вищої освіти _____ Василь Карабінський
(підпис) (Власне ім’я, ПРІЗВИЩЕ)

Керівник роботи _____ Вікторія Полуциганова
(підпис) (Власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Дана робота містить 118 сторінок, 39 ілюстрацій, 7 таблиць, 47 джерел за переліком посилань.

Мета дослідження – підвищення ефективності виявлення та класифікації підозрілих криптовалютних адрес з метою покращення аналітики у блокчейн форензиці шляхом використання методів машинного навчання та інтерпретованих моделей аналізу транзакційної поведінки.

Об’єкт дослідження – криптовалютні адреси у публічних блокчейнах.

Предмет дослідження – атрибути криптовалютних адрес та їх транзакційна поведінка.

Методи дослідження: аналіз наукових джерел з блокчейн форензики, збір адрес, які належать відомим сутностям, побудова інформативних ознак на основі транзакційної поведінки відповідних адрес, класифікація методами машинного навчання, оцінка ефективності, інтерпретація роботи моделі.

У результаті дослідження була створена ефективна модель класифікації криптовалютних адрес, яка дозволяє інтерпретувати вплив окремих ознак на прийняття рішення. Також було проаналізовано особливості транзакційної поведінки різних категорій адрес

Ключові слова: блокчейн форензика, мультикласова класифікація, транзакції, машинне навчання, криптовалютні злочини.

ABSTRACT

This work comprises 118 pages, 39 illustrations, 7 tables, and 47 sources in the list of references.

The aim of the research is to improve the effectiveness of detecting and classifying suspicious cryptocurrency addresses to enhance analytics in blockchain forensics through the use of machine learning methods and interpretable models of transactional behavior analysis.

The object of the research is cryptocurrency addresses on public blockchains.

The subject of the research is the attributes of cryptocurrency addresses and their transactional behavior.

Research methods: analysis of scientific literature on blockchain forensics, collection of addresses belonging to known entities, construction of informative features based on the transactional behavior of the respective addresses, classification using machine learning methods, performance evaluation, and model interpretation.

As a result of the research, an effective classification model of cryptocurrency addresses was developed, enabling interpretation of the impact of individual features on decision-making. Additionally, the transactional behavior characteristics of different categories of addresses were analyzed.

Keywords: blockchain forensics, multiclass classification, transactions, machine learning, cryptocurrency crimes.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – прикладний програмний інтерфейс

UTXO – вихід невитраченої транзакції

ICO – Initial Coin Offering

DAO – Decentralized Autonomous Organization

BNB – скорочення від Binance Coin

BTC – скорочення від Bitcoin

P2PKH – Pay To Public Key Hash

P2SH – Pay To Script Hash

P2WPKH – Pay To Witness Public Key Hash

ML – машинне навчання

OSINT – розвідка на основі відкритих джерел

CSV – файл з значеннями розділеними комами

TP – істинно позитивні передбачення

FP – хибно позитивне передбачення

FN – хибно негативне передбачення

JSON – JavaScript Object Notation

ВСТУП

Актуальність роботи: Актуальність роботи обумовлюється популярністю використання криптовалют в різних цифрових злочинах, такия як: фінансування тероризму, оплати в даркнеті, взломи, шахрайства, обхід санкцій, ransomware кампанії. В цьому контексті виникає потреба в аналізі криптовалютних операцій та розробці ефективних алгоритмів для протидії цим злочинам. Обмежена кількість відкритих інструментів та рішень у сфері блокчейн форензики теж сприяє актуальності даного дослідження.

Мета дослідження: підвищення ефективності виявлення та класифікації підозрілих криптовалютних адрес з метою покращення аналітики у блокчейн форензиці шляхом використання методів машинного навчання для класифікації та інтерпретованих моделей для аналізу транзакційної поведінки.

Завдання дослідження:

1. Проаналізувати існуючі рішення та відкриті датасети у сфері блокчейн форензики

2. Сформувати власний датасет із ознаками, побудованими на основі транзакційної активності адрес

3. Реалізувати класифікацію адрес за допомогою методів машинного навчання

4. Проаналізувати результати класифікацій для формування визначних ознак для категорій криптовалютних адрес.

Об’єкт дослідження: відомі криптовалютні адреси у публічному блокчейні

Предмет дослідження: атрибути криптовалютних адрес та їх

транзакційна поведінка

Методи дослідження: аналіз наукових джерел з блокчейн форензики, збір адрес, які належать відомим сутностям, побудова інформативних ознак на основі транзакційної поведінки відповідних адрес, класифікація методами машинного навчання, оцінка ефективності, інтерпретація роботи моделі.

Наукова новизна одержаних результатів: сформовано вибірку з 45 ознак, отримані на основі поведінкових характеристик адрес, яка дозволяє отримати високу точність класифікації для деяких використаних моделей машинного навчання. Додатково використано сучасні методи інтерпретації моделей, зокрема SHAP, який дозволяє прояснити вплив окремих ознак на прийняття рішень та покращити прозорість класифікації.

Практичне значення одержаних результатів: Результати дослідження можуть бути використані для розробки інструментів автоматичної класифікації криптовалютних адрес за їх транзакційною поведінкою. Створена вибірка ознак та проінтерпретовані результати класифікації можуть стати основою більш комплексної системи, здатної працювати з великими обсягами блокчейн-даних. Запропонований підхід може бути адаптований для інших блокчейнів та використаний у практиці для подальшого дослідження.

Апробація результатів роботи: Результати дослідження були представлені на XXIII Всеукраїнській науково-практичній конференції студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики» (14 - 17.05.2025 р., м. Київ, Україна).

Публікації: Матеріали дослідження опубліковано у збірнику XXIII Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики», 2025 рік.

1 ТЕОРЕТИЧНІ ЗАСАДИ БЛОКЧЕЙН ФОРЕНЗИКИ

1.1 Поняття та значення блокчейн форензики

У сучасному цифровізованому світі досить широкого загалу набули криптовалюти, як засіб розрахунку, обміну, торгівлі або зберігання активів. Децентралізованість, прозорість, псевдонімність, конвертованість, глобальність – ці властивості привернули увагу не лише звичайних користувачів, бажаючих використати криптовалюти в легітимних цілях, але й злочинців, які використовують цю технологію для фінансових махінацій.

Постійний розвиток та зацікавленість користувачів у криптовалюті призвели до експоненційного зростання капіталізації ринку криптовалют. У грудні 2024 року, ринок криптовалют досяг відмітки в 3,73 трильйона доларів, подолавши попередній максимум 2021 року, можемо переглянути це на рисунку 1.1 нижче.

Зі зростанням ролі криптовалют в глобальних фінансах зросла загроза використання цифрових активів для відмивання грошей, плати за нелегальну продукцію, різних видів шахрайств, на рисунку 1.2 можемо переглянути статистику від Chainanalysis [1] по сукупній вартості криптовалютних активів, отриманих з протизаконних адрес. У цьому контексті зростає потреба в аналізі, відстеженні криптовалютних операцій та деанонімізації їх власників, що зумовило виникнення такої дисципліни як криптовалютна форензика.

Рисунок 1.1 – Графік зміни капіталізації ринку криптовалют

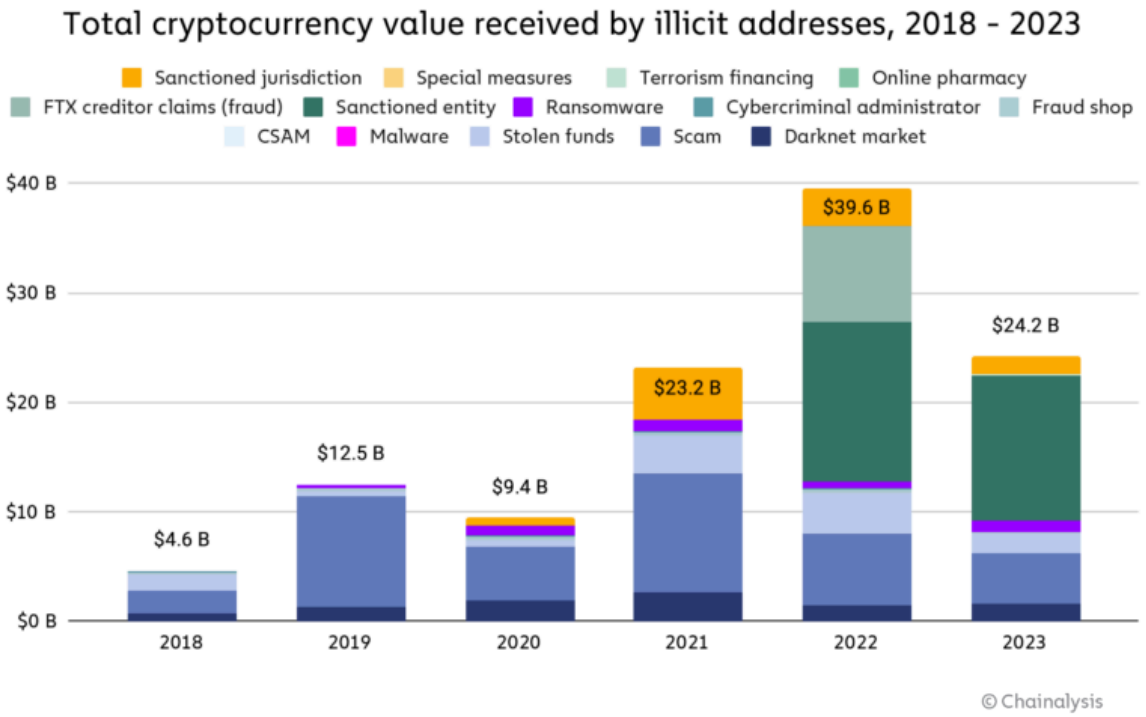


Рисунок 1.2 – Загальна вартість криптовалюти отриманої з злочинних адрес

Криптовалютна (блокчейн) форензика [2] – це вид цифрової форензики, направлений на розслідування та аналіз даних блокчейну для виявлення, ідентифікації, протидії правопорушень, в яких була використана криптовалюта.

Головною метою блокчейн-форензики є розкриття цифрових правопорушень, пов'язаних із використанням криптовалют, шляхом детального аналізу потенційно зловмисних транзакцій, встановлення зв'язку між цифровими активами та реальними особами, а також надання органам правопорядку достовірних цифрових доказів, які можуть бути використані в подальших слідчих або судових діях.

Задача блокчейн форензики є досить комплексною і аналіз транзакцій потребую приділення уваги до усіх аспектів. Що саме робить цю задачу такою важкою? [3]

1. Псевдонимність: на відміну від традиційної банківської системи, де кожна транзакція пов'язана з юридичною або фізичною особою, в блокчейні користувачі ідентифікуються криптографічними адресами. Ці адреси не містять жодної інформації про особу, яка ними володіє, що значно ускладнює процес встановлення власника.

2. Використання міксерів та chain-hop операцій: За допомогою сервісів міксерів транзакції користувачів змішуються та задача слідування оригінальному потоку коштів стає комплексною, chain-hop операції полягають в переміщенні активів між блокчейнами, що ускладнює побудову єдиного ланцюга транзакцій.

3. Використання анонімних криптовалют: Dash, Monero, Zcash розроблені спеціально, щоб покращити анонімність в порівнянні з Bitcoin, ообфускуючи транзакційні дані. Більш детально будуть розглянуті в пункті 1.3.

4. Транскордонні розслідування: складність кооперації з різними юрисдикціями та міжнародними обмінними платформами.

5. Великий об'єм дослідження: спеціалістам з блокчейн форензики приходиться проаналізувати сотні, а то й тисячі адрес та десятки й тисячі пов'язаних транзакцій для виявлення зв'язків між гаманцями, джерел походження коштів, ідентифікації шаблонів відмивання грошей або встановлення зв'язків з відомими незаконними сутностями, як наприклад даркнет ринками, підсанкційними особами чи кампаніями, шахраями.

Основні кроки розслідування [3][4]:

1. **Початковий збір даних:** збір даних з блокчейну, а саме адрес, потенційно залучених в злочині та історії їх транзакцій, використовуючи відкриті джерела та блокчейн експлорели.
2. **Аналіз зібраних даних:** Початковий аналіз зібраних даних, відстеження підозрілих транзакцій, знаходження патернів.
3. **Аналіз потоку коштів:** Аналіз транзакцій, ідентифікація входів та виходів, кластеризація адрес, які потенційно належать одній сутності.
4. **Виявлення використання міксерів та кросчейн операцій:** Зловмисники використовують міксери та пересилають активи між різними блокчейнами для ускладнення встановлення зв'язків між транзакціями.
5. **Знаходження вихідної точки:** знаходження вихідної транзакції в якій цифрові активи конвертуються у фіат, або пересилаються у централізовані біржі, де їх можна пов'язати з реальними людьми.
6. **Документування:** Документування отриманих результатів та оформлення доказів для потенційних судових процесів.

З розширенням впливу криптовалют на світову економіку та її інтеграцією в повсякденне життя звичайних громадян, список злочинів, в яких вона фігурує як об’єкт або інструмент злочинної діяльності стає дедалі більшим. Цифрові злочини не єдина сфера застосування, криптовалюти широко використовуються злочинцями в реальному світі для оплати нелегальних послуг і товарів та фінансування інших протиправних дій. Можемо виділити найбільш поширені правопорушення, які є об’єктом для дослідження в блокчейн форензиці [5]:

1. **Оплата програм вимагачів(ransomware):** ransomware атаки використовують шкідливе програмне забезпечення для шифрування усіх файлів в системі жертви та вимагають оплати ключа дешифрування в цифровій валюті.
2. **Оплати в даркнеті:** Криптовалюти є превалюючим методом оплати в даркнеті, за допомогою них злочинці можуть купувати наркотики, зброю, викраденні дані тощо.
3. **Зломи та експлойти:** Зломи гаманців користувачів, заволодіння приватними ключами та паролями для відновлення; Експлойти вразливостей в смартконтрактах (DAO, 2016; Binance, BNB 2022); Зломи централізованих бірж для обміну криптовалютою(Bybit 2025, Mt.Gox 2014, Bitstamp 2015).
4. **Шахрайство:** фішингові кампанії, направлені на заволодіння криптовалютою жертви; шахрайські інвестиційні схеми; Ponzi-схеми(Фінансові піраміди); IOC сками [6]; Rug Pull [7];.
5. **Відмивання коштів:** Кінцевою метою усіх криптовалютних злочинів є легалізація коштів для подальшого використання. Злочинці використовують різні методи для переказу коштів на різні адреси, для того, щоб приховати оригінальне походження. Після

розподілення, кошти консолідуються та надходять наче з легітимного джерела на централізовані обмінники.

1.2 Технологічні аспекти блокчейну та транзакцій в криптовалютах

1.2.1 Структура блокчейну

Блокчейн [8] - це розподілений реєстр, ланцюжок блоків, який зберігає в собі записи про транзакції та контракти. Дані записи поширюються між усіма вузлами мережі, забезпечуючи прозорість, цілісність та незмінність даних.

Після здійснення певної кількості транзакцій, вони видобуваються в блок і після цього процесу нічого в блоці не може бути змінено або додано.

Структура блоку на прикладі Біткоїн [9] :

- розмір блоку(в нашому випадку 4 байти)
- заголовок блоку (80 байті)
- кількість транзакцій (1 - 9 байтів)
- Самі транзакції

Розглянемо заголовок блоку. Заголовок складається з 6 частин [9]:

- Версія
- Хеш попереднього блоку
- Merkle root
- Timestamp
- Difficulty target

- Nonce

Кожна з частин заголовка блоку виконує важливу роль у забезпеченні надійності та цілісності блокчейн-системи. Версія вказує на те, за якими правилами створено блок, і дає змогу відстежувати зміни та оновлення в протоколі. Хеш попереднього блоку зв’язує поточний блок із попереднім, утворюючи єдиний ланцюг — якщо змінити хоча б один блок, це порушить весь ланцюг і зробить його недійсним. Merkle root (схема роботи на рисунку 1.3) є своєрідним "зведеним хешем" усіх транзакцій у блоці — завдяки цьому елементу можна швидко перевірити, чи не була змінена жодна з транзакцій, не переглядаючи кожну окремо. Timestamp фіксує момент створення блоку і допомагає впорядковувати блоки в хронологічному порядку. Difficulty target визначає складність майнінгу, тобто наскільки складно знайти правильне значення хешу для блоку — це потрібно, щоб мережа працювала стабільно й блоки не створювалися занадто швидко. Останнім елементом є Nonce, тобто число, яке майнери багаторазово змінюють у спробах знайти хеш, що відповідає заданому рівню складності. Усі ці елементи разом забезпечують прозорість, безпечність і стабільність роботи блокчейн-мережі.

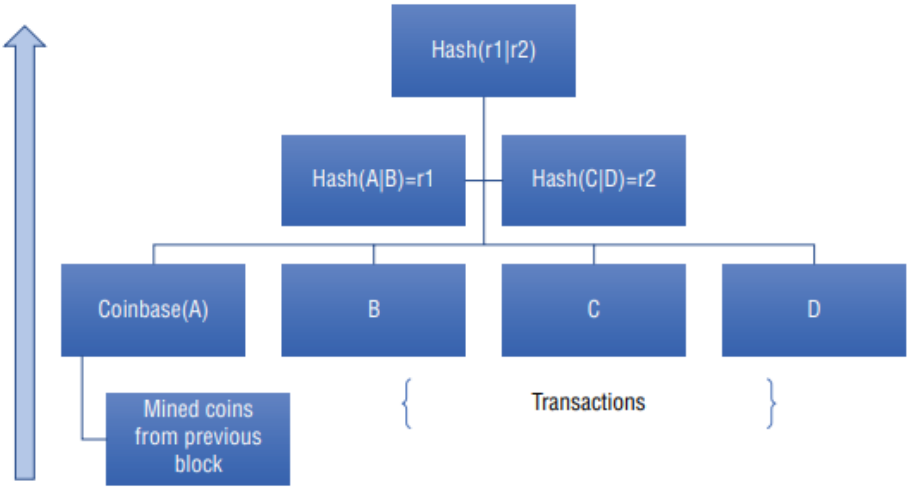


Рисунок 1.3 – Схема Merklee tree [9]

1.2.2 Транзакції

Для розуміння концепту транзакцій в блокчейн треба відсторонитися від вже відомих нам традиційних платежів, таких як готівкові та електронні, через банківську систему. Криптовалютні транзакції на відміну від звичайних грошових, не контролюються центральним органом виконавчої влади або адміністрацією банківських установ. Таким чином, зміни в даних транзакцій можуть бути досягнені лише за допомогою консенсусу серед учасників децентралізованої мережі.

Транзакція [10] зазвичай складається з адреси відправника, адреси реципієнта, кількості надісланої криптовалюти, комісії за пересилання, часової мітки та цифрового підпису. Адреси в блокчейні [11] – це унікальний криптографічний ідентифікатор, що використовується для отримання або надсилання цифрових активів, вони є похідними від публічних ключів, які в свою чергу, отримані з приватного ключа. Маючи контроль над приватним ключем, ви маєте контроль над усіма адресами похідними від нього, а отже і над криптоактивами, які на цих адресах зберігаються.

З одного приватного ключа можна зробити близько 2 мільярдів публічних ключів, завдяки цьому користувач для більшої анонімності може використовувати для кожної транзакції нову адресу, ускладнюючи встановлення зв’язку між транзакціями які належать одній людині. Підтверджуючи транзакцію, користувач підписує її своїм приватним ключем, таким чином підтверджуючи, що акаунт з якого надсилаються цифрові активи належить користувачеві й він погоджується на умови транзакції [12]. Місце призначення активів, публічна адреса, також міститься в транзакції, це дозволяє реципієнту отримати доступ до коштів, за допомогою асоційованого приватного ключа. Підписана транзакція

повинна бути провалідована усіма вузлами мережі, використовуючи публічний ключ для перевірки наявності коштів для переведення та для перевірки вірності підписання.

В цьому контексті, треба розглянути дві важливі моделі для управління транзакціями та збереження стану балансу криптовалютних активів:

- UTXO model (Unspent Transaction Output) [13]
- Account model [14]

UTXO model

Модель використовувана в таких проектах як Bitcoin, Litecoin, Cardano, Dogecoin. ЇЇ Основна концепція полягає в тому, що цифрові активи асоційовані не з конкретною адресою, а з унікальними UTXO, виходами транзакції, які можуть бути використані як входи для нової транзакції. Кожна транзакція створює один або декілька виходів. Одна адреса може потенційно бути асоційована з багатьма UTXO, сума яких і складає її загальний баланс. Важливо особливістю моделі UTXO є те, що неможливо витратити частину виходу.

Наприклад, якщо акаунт має одну UTXO в розмірі двох біткоїнів і бажає заплатити за товар коштом в 1 біткоїн, система змушена надіслати усі 2 біткоїни в транзакцію, в такому випадку створюється транзакція яка має два виходи:

- 1 Біткоїн надісланий одержувачу(spent output)
- 1 Біткоїн повертається на адресу відправника, як здача(unspent output)

Unspent output в розмірі одного Біткоїна відповідно може бути використаний як вхід для подальших транзакцій.

Account model

Модель була популяризована з появою платформи Ethereum. Концепція полягає в тому, що кожен користувач має акаунт, в якому зберігається загальний баланс активів, не застосовуючи систему входів та виходів. На відміну від моделі UTXO, тут реалізується більш спрощена система, достатньо всього відняти суму з акаунту відправника та додати до акаунта одержувача, аналогічно до банківського рахунку. Така реалізація значно спрощує роботу блокчейну зі смарт-контрактами, дозволяючи їм швидко керувати власними балансами та взаємодіяти з іншими контрактами.

В такій мережі як Ethereum існує два типи акаунтів [15]:

- Externally Owned Accounts (EOAs): акаунти контрольовані приватними ключами, які використані особами.
- Contract Accounts: акаунти контрольовані кодом смарт-контрактів, можуть виконувати код під час отримання транзакцій.

1.3 Види криптовалют і їх вплив на форензичний аналіз

1.3.1 Псевдоанонімні криптовалюти

На відміну від думки більшості пересічних користувачів, абсолютна більшість криптовалют не являються приватними, вони базуються на псевдоанонімній структурі.

Усі транзакції, адреси відправника та одержувача, часові мітки, кількість крипто активів записані в публічний блокчейн, який дає змогу

переглядати усі деталі. Хоч адреси, транзакції напряму і не пов’язані з реальними людьми, але при ретельному аналізі є вірогідність відстеження потоку коштів та ідентифікація осіб.

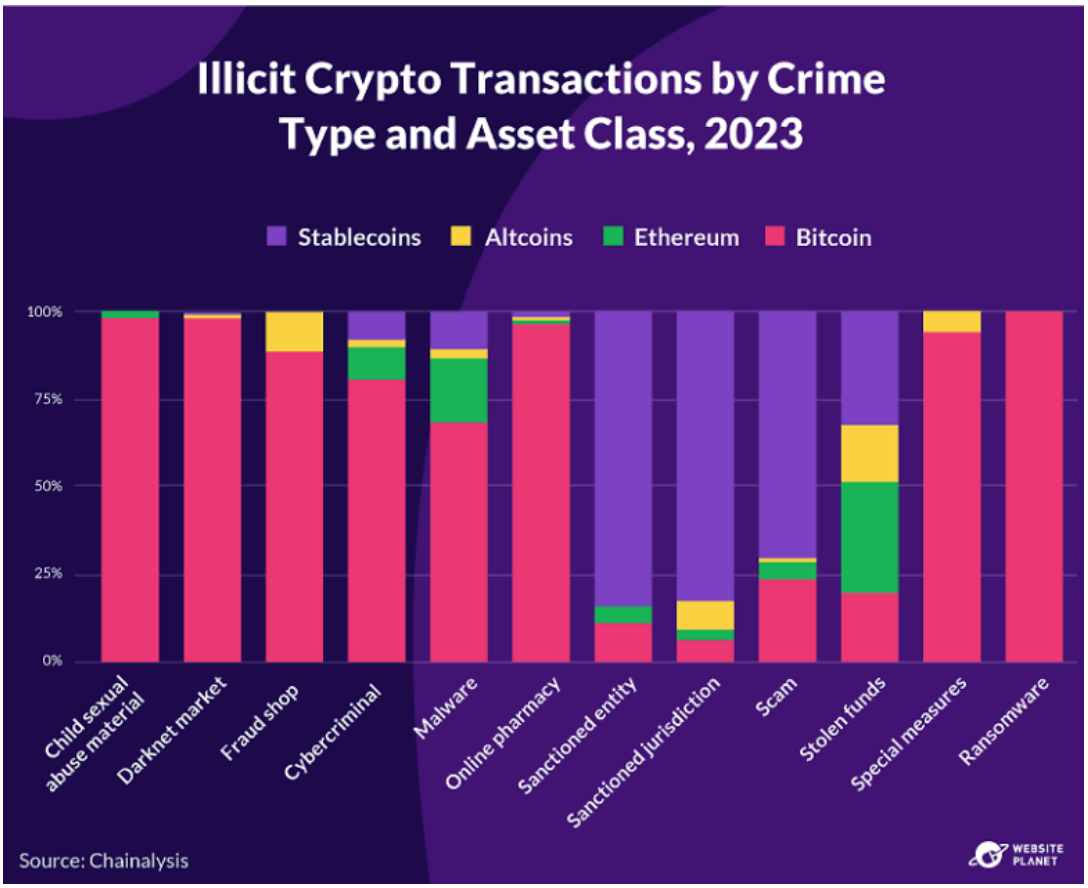


Рисунок 1.4 – Відсоток злочинних транзакцій по злочинах та криптовалютам

Згідно зі статистикою Chainanalysis за 2023 рік [16] (Рисунок 1.4), використання не анонімних криптовалют в кримінальних активностях, можемо коротко розглянути найбільш популярних представників:

Bitcoin

Перша криптовалюта, створена Сатоші Накамото у 2008 році [10], саме з Біткоїна почався шлях децентралізованих цифрових активів. Капіталізація складає понад 1,7 трильйона доларів США, що робить біткоїн найбільш ліквідною криптовалютою. Використовує протокол консенсусу PoW(proof-of work), згідно з яким майнери змагаються за право

додати новий блок до блокчейну, розв’язуючи складні обчислювальні задачі, які вимагають потужних комп’ютерних ресурсів, що забезпечує безпеку мережі. Майнер, який перший розв’язав задачу отримує винагороду в токенах BTC.

Через глобальну поширеність, найбільшу ліквідність, відносно невисоку волатильність, Біткоїн досі займає провідні позиції серед криптовалют, використаних в протизаконних діях. Попри те, що останнім часом стейблкоїни набувають більшого поширення і витискають Біткоїн залишається домінуючою криптовалютою в таких злочинах як:

- торгівля забороненим контентом
- програми вимагачі, шкідливе програмне забезпечення
- торгівля в даркнеті

Ethereum

Друга за капіталізацією криптовалюта, створена Віталіком Бутеріним у 2013 році. Реалізує більш складну архітектуру, дозволяє деплоїти смарт-контракти. Смарт-контракти можна представити як повнозв’язні комп’ютерні програми, написані мовою програмування solidity, та можуть бути активовані транзакціями, які передають вхідні дані до їх функцій [17].

Смарт-контракти дозволяють розроблення децентралізованих програм (dApps), які доволі часто використовуються зловмисниками для впровадження протизаконних дій. Згідно зі статистикою ця криптовалюта найбільш застосовується в викраденні коштів та шкідливому програмному забезпеченні. Аналіз коду смарт-контрактів, хоч і не розглянутий в даній роботі, але теж є важливою частиною блокчейн форензики, оскільки хакери все частіше використовують вразливості коду для заволодіння активами та реалізації інших злочинів. Навіть відомий Open Worldwide

Application Security Project створив свій список провідних вразливостей в смарт-контрактах OWASP Smart Contract Top 10 [18].

Stablecoins

Стейблкоїни це цифрові валюти, які прив’язані до реальних фіатних валют, як наприклад Американський долар (usd). Мають попит у злочинців через швидкість транзакцій, стабільність та низьку волатильність порівняно з іншими криптовалютами. Найбільшої популярності набули в секторі ухилення від санкцій, користуючись такими токенами як USDT (USD Tether) підсанкційні сутності можуть переміщувати великі суми грошей за досить швидкий час, без необхідності конвертації в більш волатильні монети, як наприклад Біткоїн.

1.3.2 Анонімні криптовалюти

Прагнення користувачів до підвищення своєї приватності в блокчейні, зокрема страхування від стеження, фінансового профілювання, зломів стало каталізатором появи анонімних криптовалют, таких як Dash, Monero, Zcash. Властивості підвищеної конфіденційності привернули увагу не лише легітимних користувачів, але й злочинців, які за допомогою цього механізму прагнуть обфускувати сліди протизаконної діяльності.

Monero

Форк Bitcoin, забезпечує повну приватність транзакцій за замовчуванням. Реалізує принцип “змішування”, використовує одноразові адреси в кожному платежі, кожна така адреса прив’язана до реальної

публічної адреси, яка і отримує кошти [19]. Гаманець набирає довільні виходи з блокчейну, ховає серед них, той який хоче витратити та скріплює це кільцевим підписом, для того, щоб валідатор транзакції знав, що відправник дійсно має один з виходів і він не є витраченим. Приховує кількість відправленої криптовалюти за допомогою механізму RingCT.

Характеристика приватності транзакції залежить від розміру кільцевого підпису, чим більше виходів використовується в замішування, тим складніше відстежити реального відправника, на сьогодні мінімальний розмір становить 16.

Dash

Форк Bitcoin, сам по собі не є приватним токеном за замовченням, але має функцію PrivateSend, яка працює за принципом CoinJoin, дозволяє користувачам приховати джерело своїх коштів [20]. Певна кількість користувачів одночасно ініціюють запит на відправлення токенів, Dash розбиває їхні суми на номінали виду 0.01, 0.1, 1 Dash тощо, транзакції об’єднуються в одну спільну, що представляє їхні перекази до різної кількості одержувачів, цей процес робить завдання відстеження автентичних входів та виходів досить важким. Чим більше раундів цих “мікшувань” активів, тим більше безпечним та анонімним є процес надсилання. Механізм анонімізації Dash є більш тривіальним, порівняно з тими ж Monero та Zcash.

Zcash

Форк Біткоїну, з’явився як реалізація протоколу ZeroCash. Був розроблений з метою мати можливість приватності в транзакціях [19]. Дана технологія дозволяє повність скривати адресу відправника та

отримувача разом з кількістю надісланої криптовалюти, єдине, що видно при передачі це часова мітка.

Приватність не є обов'язковою, користувачі можуть використовувати як прозорі, так і захищені транзакції. Юзери, які бажають підвищити свою конфіденційність можуть скористатися так званими shielded transactions. Існує два типи адрес це:

- **t-addr**

  Прозора адреса, подібна до тих, що у Біткоїн, відповідно всі транзакції публічні та видимі у блокчейні

- **z-addr**

  Захищена адреса, використовує революційну технологію zk-SNARKs, яка дозволяє приховати деталі транзакції за допомогою шифрування**.**

zk-SNARKs базується на принципі "Zero Knowledge Proofs" [21], тобто за допомогою цієї технології можна провалідувати певну транзакцію, а саме zk-SNARK створює доказ того, що:

- Ти володієш потрібними приватними ключами.
- Сума транзакції правильна.
- Баланс не порушується.
- Ніхто не дублює монети.

при цьому не розкриваючи її деталей.

1.4 Методи блокчейн форензики

Кластеризація адрес, Графовий аналіз, залучення машинного навчання є ключовими техніками для відстеження потоку коштів та виявлення адрес пов’язаних з злочинною діяльністю [22] [23]. Консолідація цих методів може допомогти для проведення більш точного розслідування, розглянемо кожен з них окремо.

1.4.1 Кластеризація адрес (Address Clustering)

Задача кластеризації полягає в тому, щоб розбити задану вибірку об’єктів на кластери, об’єкти в яких мають схожі риси. В контексті криптовалютної форензики задачею кластерного аналізу є об’єднання певної кількості блокчейн адрес в кластер, який ймовірно належить одній людині чи кампанії. Основною технікою кластеризації наразі є евристика спільних входів (common input ownership **heuristic**), що засновується на припущенні про те, що якщо декілька адрес використовуються разом як входи до одиничної транзакції, то вони належать одному власнику, оскільки транзакцію можна підписати лише після створення. Можемо вважати безпечним припущення, що той, хто створив транзакцію, також контролює всі приватні ключі, які були використані для її підписання.

Ще однією важливою евристикою є виявлення адрес решти (change adress detection heuristic). Даний аналіз заснований на особливостях моделі UTXO, як ми вже знаємо, що важко заплатити одержувачу точну суму в криптовалюті і відправник отримує решту коштів через адреси решти,

виявлення таких адрес є досить важливим, тому що вони належать одній і тій самій сутності.

Основні способи виявлення адрес решти:

- **Повторне використання адреси:** цей метод ґрунтується на перевірці адрес виходів транзакції. Якщо серед адрес виходів існує адреса яка раніше ніколи не зустрічалася і не використовувалась до даної транзакції, то з великою ймовірністю це адреса решти. Більшість сучасних гаманців використовують створення нових адрес для кожного залишкового виходу.

  Повинні виконуватись такі умови: Один з виходів відомий реципієнт(адреса на яку актив був відправлений)

  Інший вихід ніколи не зустрічався і не був використаний до цієї транзакції(адреса решти)
- **Тип використаного скрипту:** Дуже ймовірно, що якщо всі адреси входів використовують скрипт одного типу(наприклад, P2PKH, P2SH, P2WPKH тощо), то адреса решти буде використовувати той самий тип, на відміну від інших виходів.
- **Аналіз значення виходу:** Виходи решти зазвичай мають більш “безладне” значення кількості криптовалюти та мають більше знаків після коми. Це пов’язано з тим, що після відправлення основної суми, залишок (решта) розраховується автоматично з урахуванням комісії за транзакцію, яка також вираховується саме з цієї суми, тому ми можемо бачити значення виходів 0.00067 BTC тощо.

1.4.2 Графовий аналіз

Метод візуалізації дерева транзакцій або гаманців для знаходження ключових залежностей. В якості вузлів графу виступають адреси гаманців або транзакції, а ребрами є зв’язки які виникають при переказі активу. Візуалізація графу полегшує задачу слідування потоку коштів та ідентифікації поведінкових патернів та використання міксерів. Метою аналізу є знаходження так званої кінцевої точки, де криптовалюта переведена в фіат, або виведена на біржу обмінник

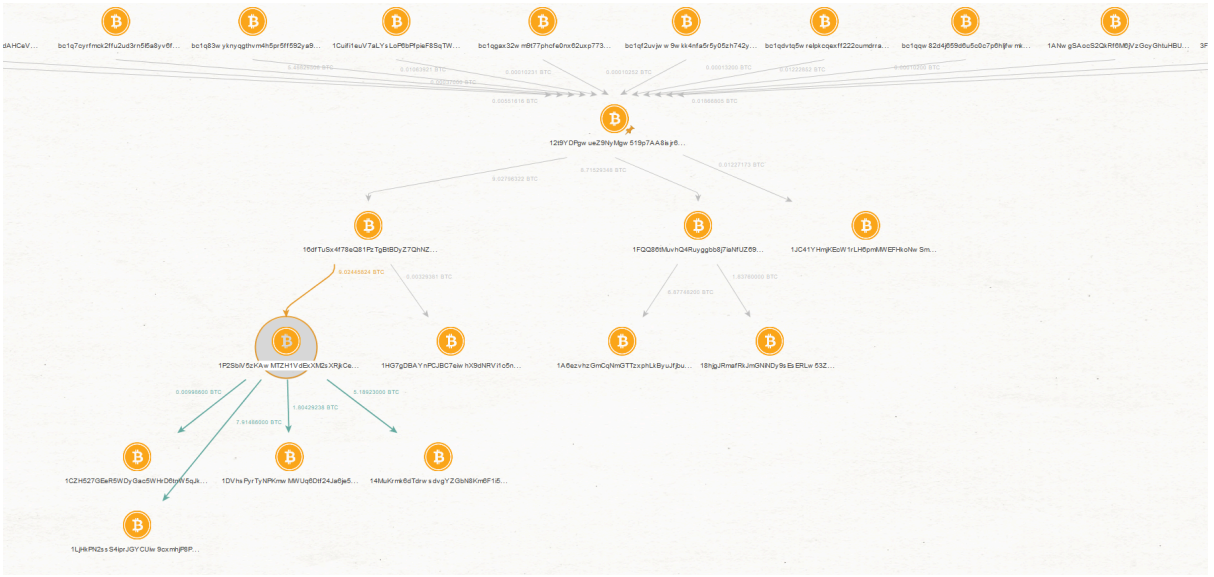


Рисунок 1.5 – Граф потоку коштів для злочинної адреси

Приклад використання графового аналізу до адреси програми вимагача WannaCry **12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw** за допомогою Maltego [24].

В верхній частині рисунка можемо бачити адреси всіх жертв, які надсилали кошти на гаманець зловмисника, ребра вказують на зв’язки між гаманцями та кількість надісланого активу. Адреса зловмисника має 3 вихідні транзакції, при чому у двох з ним консолідована найбільша

кількість активу **16dfTuSx4f78eQ81PzTgBtBDyZ7QhNZ8Vy** з 9 Біткоїнами та **1FQQ86tMuvhQ4Ruyggbb8j7iaNfUZ69gpY** з 8.7**.** Таким чином можна слідувати далі за вузлами графу для знаходження точки виходу

1.4.3 Застосування машинного навчання

Машинне навчання (ML) стає дедалі популярнішим інструментом та відіграє ключову роль у контексті блокчейн форензики, воно дозволяє автоматизовано виявляти закономірності які важко помітити за допомогою традиційних методів аналізу. Проаналізувавши [25], [26], [27] можемо виділити основні напрямки застосування:

- **Кластеризація** – користуючись методами машинного навчання можливо автоматично виявляти кластери криптовалютних адрес, які належать одній сутності, тим самим виявляючи підозрілу поведінку або знаходячи патерни координованих дій
- **Детектування аномальних транзакцій** – полягає в ідентифікації аномальних транзакційних патернів, які ймовірно мають відношення до криптовалютних злочинів.
- **Детектування злочинних адрес** – ще один важливий напрям, що передбачає класифікацію адрес за типами з метою виявлення тих, що можуть бути пов’язані з незаконною діяльністю. Для цього застосовуються алгоритми класифікації, які навчаються на основі вже позначених адрес і дозволяють передбачати належність нових адрес до певних категорій.

Хоч машинне навчання в цьому контексті має значні переваги, як наприклад здатність до узагальнення, масштабованість, знаходження не

тривіальних зв’язків у даних, але при цьому, воно вимагає наявності підготованих даних, якісно сформованих ознак та методів вирішення нерівномірного розподілення даних.

1.5 Моделі класифікації

В рамках даного підрозділу будуть розглянуть класичні моделі для мульти класової класифікації, які потенційно можуть бути застосовані в даному дослідженні.

1.5.1 Decision tree

Decision tree (Дерево рішень) [28] - це алгоритм машинного навчання, який використовує структуру у вигляді дерева для прийняття рішень, рекурсивно поділяючи датасет до тих пір поки не залишаються лише листові вузли (pure leaf nodes) з одним типом класу. На кожному етапі дерево обирає ту ознаку та порогове значення, які дозволяють найбільш ефективно поділити дані — тобто зменшити невизначеність або максимізувати «чистоту» підмножин, використовуючи такі критерії, як Gini impurity, information gain або entropy.

Дерево рішень досить просте в інтерпретації та візуалізації і не потребує особливої підготовки даних, в той самий час ця модель схильна до перенавчання, при побудові надто комплексних дерев.

1.5.2 Random Forest

Random Forest [29] – Ансамблевий метод машинного навчання, оснований на комбінації множини дерев прийняття рішень, що дозволяє зменшити ризик переобучення та підвищити точність в порівнянні з одиничним деревом. Алгоритм базується на принципі беггінгу, тобто кожне дерево навчається на випадковій вибірці з датасету, а під час прогнозування усі дерева голосують за той клас, який вони передбачають, і результатом стає клас із більшістю голосів, при умові, що одне дерево має один голос. Має високу точність, практично не чутливий до шуму в даних, але потребує більших обчислювальних ресурсів.

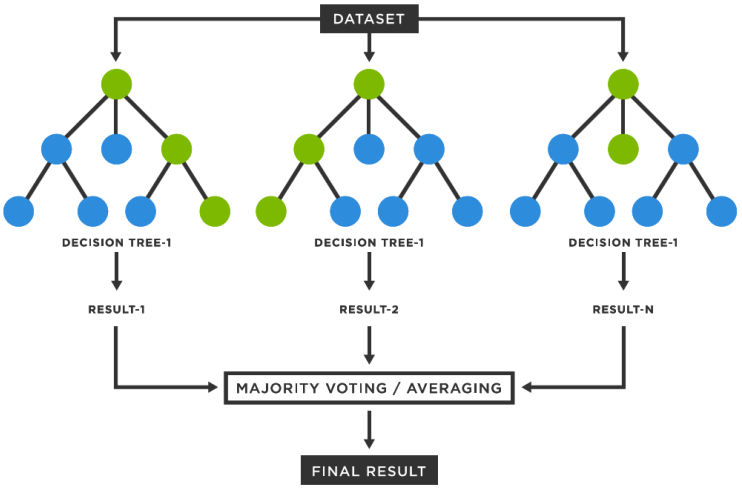


Рисунок 1.6 – Візуалізація алгоритму Random Forest

1.5.3 Boosting

Також ансамблевий метод машинного навчання, який полягає в послідовному навчанні слабких моделей, як наприклад дерев рішень, де кожна наступна модель навчається на помилках попередніх. В результаті з слабких класифікаторів отримується сильний класифікатор, який складається з поетапно покращуваних прогнозів [30]. В порівнянні з Random Forest, де дерева будуються паралельно, в boosting-моделях процес ітеративний і кожне дерево намагається виправити помилки попереднього. Використовуються ваги, де більша вага надається об’єктам, які були класифіковані не правильно, таким чином поступово зменшуючи загальну похибку. Без налаштування гіперпараметрів модель схильна до перенавчання.

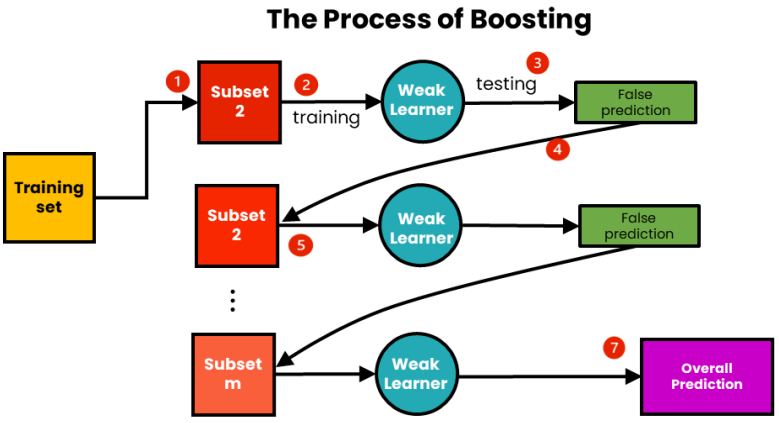


Рисунок 1.7 – Візуалізація Boosting алгоритму

1.5.4 K-Nearest Neighbours

KNN [31] – один з найбільш простих та популярних методів машинного навчання, використовуваний для регресії та класифікації.

Незважаючи на простоту, алгоритм є досить ефективним і базується на припущенні, що схожі об’єкти даних розміщуються поруч один з одним й можуть бути згруповані на основі їхньої просторової близькості. Щоб визначити, до якого класу належить нова точка, обчислюється відстань до всіх точок навчальної вибірки, обираються k найближчих, і визначається клас, який серед них зустрічається найчастіше. Його ефективність залежить від вибору правильного значення параметра k та метрики відстані. KNN чутливий до масштабування, вимагає стандартизації та також має повільну швидкість роботи на великих об’ємах даних

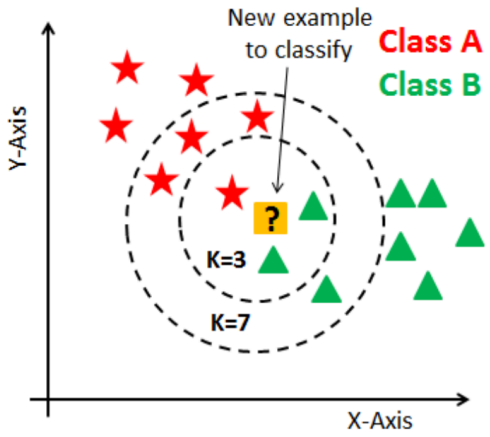


Рисунок 1.8 – Зображення алгоритму KNN

1.5.5 Logistic Regression

Логістична регресія – це також базовий метод машинного навчання, використовуваний, в більшості випадків, для класифікації. У випадку бінарної класифікації, логістична регресія обчислює ймовірність належності об’єкта до класу 1 за допомогою сигмоїдної функції, яка стискає вихідне значення у діапазон від 0 до 1. Якщо ймовірність

перевищує певний поріг, модель відносить об’єкт до позитивного класу. У випадку нашої задачі, може бути використане розширення Softmax-регресії, яка дозволяє моделі повертати ймовірності для декількох класів одночасно [32]. Даний алгоритм чутливий до не масштабованих даних та може погано працювати, коли класи не є лінійно роздільними, оскільки модель базується на лінійних гіперплощинах для поділу класів.

1.5.6 SVM

Support Vector Machine – це метод машинного навчання, переважно використовуваний для класифікації. Основна ідея алгоритму полягає у пошуку оптимальної гіперплощини, яка якнайкраще відділяє об’єкти різних класів у наборі даних. SVM шукає таку роздільну площину, яка максимізує відстань між найближчими точками кожного класу та самою гіперплощиною. Модель може гарно працювати з лінійно не роздільними даними, за допомогою використання ядрових функцій, як наприклад: поліноміальне ядро, радіальне базисне ядро, сигмоїдне ядро. За допомогою цього дані перетворюються у вищій вимір, де стає можливим знайти лінійну межу між класами. Можна адаптувати до багатокласової класифікації використовуючи підходи One-vs-Rest (OvR) та One-vs-One (OvO) [33].

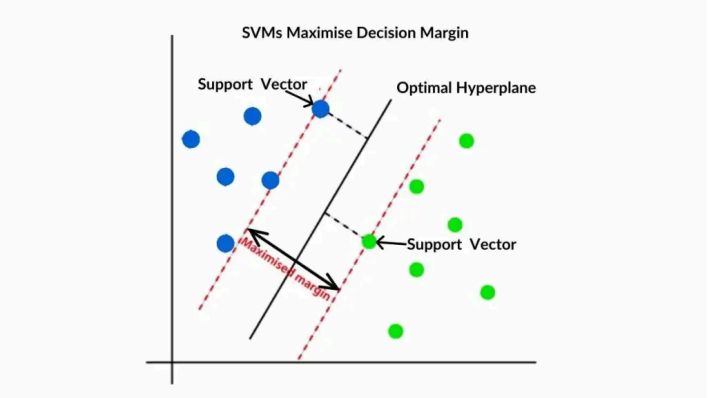


Рисунок 1.9 – Зображення схеми роботи SVM

1.5.7 MLP

Multilayer Perceptron [34] – це один з базових типів нейронних мереж, може бути застосований до задачі класифікації. Як можемо бачити на рисунку 1.12, MLP складається з вхідного шару, одного або декількох прихованих шарів та вихідного шару. Кожен шар складається з нейронів, які з'єднані з нейронами попереднього і наступного шарів.У нашому випадку, а саме багатокласовій класифікації, вихідний шар буде містити стільки нейронів, скільки ми маємо класів. Модель здатна до розпізнавання складних, нелінійних залежностей в даних, через використання нелінійних функцій активації. Для генерації ймовірностей приналежності до кожного класу використовується функція Softmax, яка перетворює вихід моделі у ймовірнісний розподіл. MLP є досить чутливим до якості вхідних даних, зокрема масштабованості ознак та налаштуванні параметрів(вибір кількості шарів, функції активації, нейронів), але при цьому модель є доволі гнучкою і може показувати високу точність класифікації при достатній наявності навчальних даних.

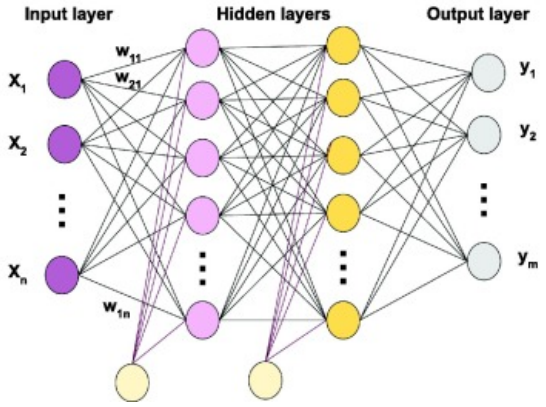


Рисунок 1.10 - Зображення взаємодії шарів в MLP

Висновки до розділу 1

У першому розділі були розглянуті теоретичні основи блокчейн форензики, а саме: мета, труднощі, основні кроки розслідування та злочини, які є об’єктами дослідження в блокчейн форензиці. Зроблено висновок, щодо комплексності даної дисципліни та необхідності розвитку її методів.

Також було досліджено основні технологічні аспекти блокчейну, транзакцій та криптовалютних адрес. Окремо виділені та описані анонімні та псевдоанонімні криптовалюти і їх вплив на аналіз. Визначено, що криптовалюта Біткоїн займає левову частину потоку нелегальних цифрових активів, але поступово витісняється стейблкоїнами в деяких категоріях.

Було здійснено огляд та характеристику основних методів блокчейн форезики, включаючи: графовий аналіз, кластеризацію та застосування машинного навчання. Зосереджена увага на машинному навчанні, розкрита його важливість в контексті задачі відстеження злочинів, при цьому зроблено висновок щодо необхідності якісно підготовлених даних.

Описано основні алгоритми машинного навчання для багатокласової класифікації та вивчено основні теоретичні засади, на яких ґрунтуються ці моделі.

У цьому розділі було розглянуто теоретичні засади, які є основою для подальшого формулювання практичної методики побудови моделі класифікації криптовалютних адрес на основі транзакційних даних.

2 РОЗРОБКА МЕТОДИКИ КЛАСИФІКАЦІЇ КРИПТОВАЛЮТНИХ АДРЕС

2.1 Загальна стратегія створення датасету

Однією з основних частин будь-якої класифікаційної моделі є наявність структурованого датасету з достовірними мітками класів та відповідними ознаками. Проте у відкритому доступі бракує помічених (labelled) датасетів щодо приналежності певної адреси або транзакції до злочинних чи комерційних легітимних сутностей. З наявних публічно можна виділити Eliptic dataset [35], Bitcoin Heist Ransomware Address Dataset [36], Bitcoin Labeled Addresses And Transactions [37], але кожен з них має певні недоліки:

- **Eliptic dataset** – розроблений однойменною компанією блокчейн аналітики Eliptic. Містить понад 200 тис. транзакцій у мережі Bitcoin, кожна з яких класифікована як легітимна, нелегітимна або невідома. Основним недоліком цього датасету є анонімізація точного опису розроблених ознак та відсутність інформації про конкретні джерела міток.
- **Bitcoin Heist Ransomware Address Dataset** – фокусується на адресах, пов’язаних з програмами вимагачами. Його обмеження полягає в вузькій спрямованості на один тип злочинної активності та браку точної інформації щодо приналежності білих (white labelled) адрес до доброчесних користувачів.

- **Bitcoin Labeled Addresses And Transactions** – містить мітки адрес, пов’язані з різними сервісами, такими як: майнери, гемблінг платформи, обмінники. Не містить адрес використаних в злочинах.

Спираючись на все вище перераховане, було прийнято рішення щодо збору потрібних даних та розробки власного датасету для подальшого використання в задачах класифікації. У межах цієї роботи планується реалізувати таку послідовну стратегію створення датасету:

1. **Збір адрес**

На першому етапі планується збір адрес криптовалютних гаманців, які можуть бути віднесеними до певних категорій. Серед категорій будуть зібрані адреси як легітимних сервісів, так і залучених в певних видах злочинів. В якості джерел можуть виступати:

- публічні списки санкційних або злочинних адрес
- ресурси, які містять адреси легітимних сервісів(біржі, майнінг пули тощо)
- OSINT-ресурси

2. **Отримання транзакційної інформації**

Для кожної зібраної адреси витягується статистична інформація про адресу та історія транзакцій, за допомогою API блокчейн експлорерів. Зокрема отримується інформація про: поточний баланс адреси, кількість транзакцій, дату та час транзакції тощо.

3. **Обробка отриманої інформації та формування ознак**

На основі зібраних на попередньому етапі даних формується вибірка з числових ознак, які можуть мати цінність при класифікації. Наприклад:

- середня комісія переводу
- середня кількість активу в транзакціях
- максимальна кількість отриманої криптовалюти

4. **Збереження та структурування даних**

Всі отриманні та оброблені на попередніх кроках дані зберігаються в єдиному файлі у вигляді структурованої таблиці у форматі CSV, де кожен рядок відповідає окремій адресі, а стовпці – відповідним ознакам та мітці класу.

5. **Попередня обробка даних**

Фінальним етапом формування датасету є попередній аналіз та за потреби очищення, зокрема застосовуючи такі методи:

- перевірка та обробка пропущених значень;
- перевірка на наявність дублікатів та усунення;
- візуалізація;
- стандартизація
- виведення статистики по стовпцях(min, max, mean, std)

Реалізація вищеописаної стратегії дозволяє отримати власний набір даних, який може бути використаний у класифікації методами машинного навчання.

2.2 Використання публічних джерел для отримання "позначених" адрес та транзакційної інформації

Початковим етапом при створенні власного датасету є збір адрес, для яких відома їхня належність до певної категорії. Для цього в роботі використовуються публічні джерела, де публікується інформація про адреси, що пов’язані з різними видами діяльності.

У межах дослідження були обрані наступні категорії адрес, що відносяться як до легітимних, так і до злочинних сутностей:

- **Гемблінг платформи (Gambling services)** – сервіси, пов’язані з різними видами азартних ігор, оплата в яких, приймається в криптовалюті;
- **Біржі (Exchanges)** – централізовані платформи для обміну та торгівлі криптовалютою;
- **Майнінгові пули (Mining Pools)** – компанії з великих об'єднань майнерів, які комбінують обчислювальні можливості для збільшення шансів отримання винагороди за видобуток блоку;
- **Даркнет-маркети (Darknet markets)** – онлайн-майданчики, які працюють у даркнеті, створені для торгівлі незаконними товарами та послугами;
- **Програми-вимагачі (Ransomware)** – адреси, які приймали викуп від жертв програм-вимагачів.

Для отримання адрес з достовірними мітками були відібрані такі ресурси:

- **WalletExplorer** [38] – Біткоїн блок експлорер з групуванням адрес та маркуванням гаманців. Має зручний інтерфейс для перегляду та документований API, який допомагає автоматизувати процес збору адрес. Даний сервіс ідеально підходить для збору 3 з 5 категорій адрес, а саме: Гемблінг платформ, майнінгових пулів та бірж/

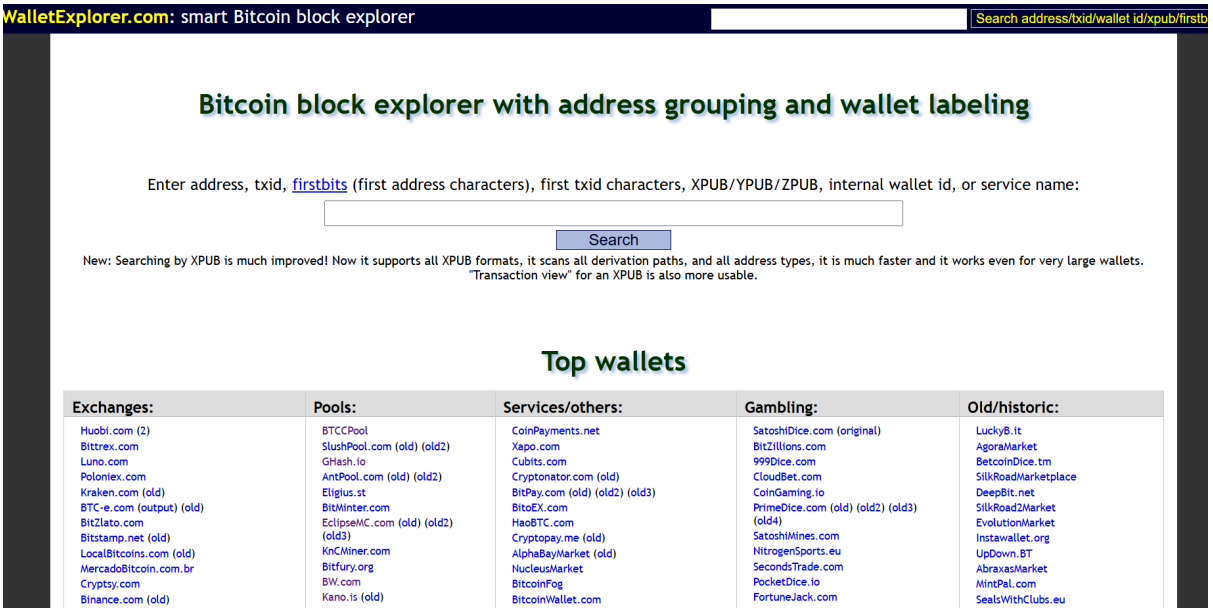


Рисунок 2.1 – Інтерфейс WalletExplorer

- **Arkham Intelligence** [39] – криптовалютна аналітична платформа, яка відкрито публікує інформацію щодо приналежності адрес в блокчейн мережі до реальних сутностей. На рисунку 1.7 продемонстрований профіль одного з найбільших російських даркнет маркетів Hydra. Даний ресурс буде використаний, в межах роботи, для збору адрес, що пов’язані з даркнет-маркетами. Arkham Intelligence має комерційний API, тому завдання полягатиме в копіюванні останніх транзакцій з вкладки ”Transfers” , парсингу даних і витягненні з них потрібних адрес.

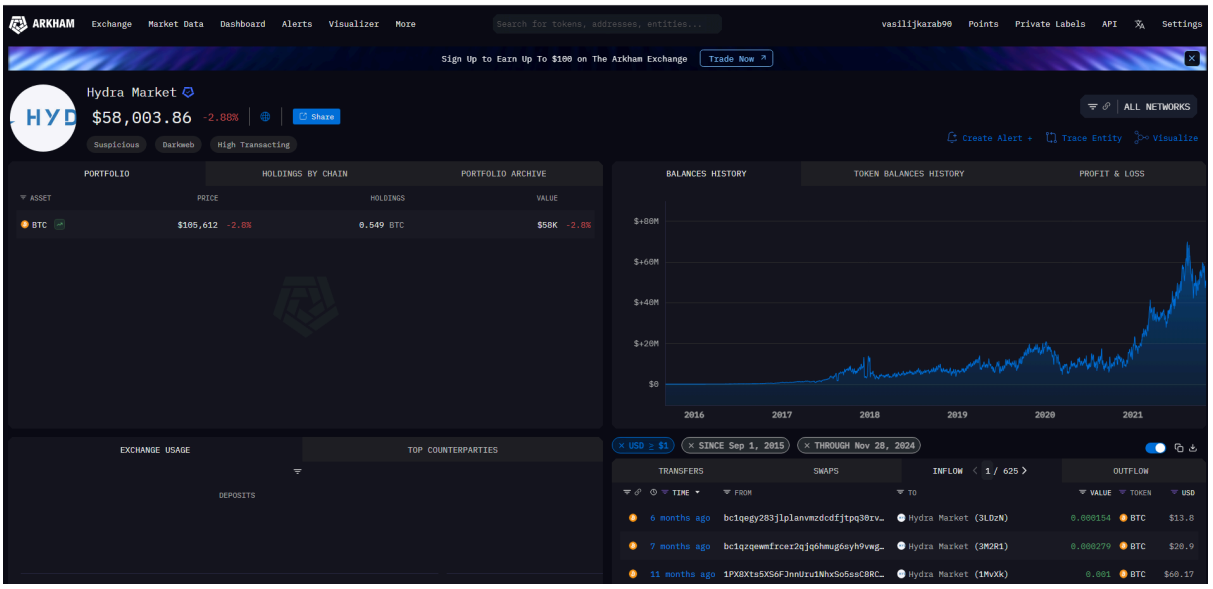


Рисунок 2.2 – Профіль Hydra Market в Arkham Intelligence

- **Ransomwhe.re** [40] – відкрита краудсорсингова платформа для відстеження платежів програмам вимагачам. Вона збирає адреси, на які надходили викупи, та надає можливість завантажити ці дані у зручному форматі. Цей сайт буде використано в роботі для збору адрес, пов’язаних із ransomware-кампаніями

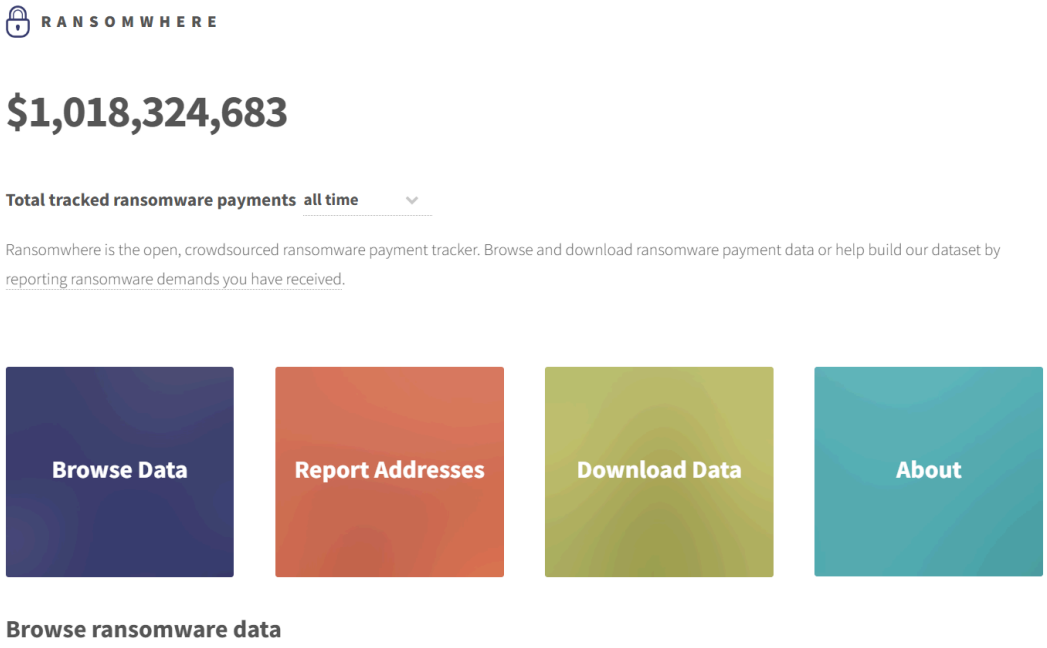


Рисунок 2.3 – Інтерфейс Ransomwhe.re

Після збору списку адрес наступним етапом є отримання для них повної транзакційної історії. Для цього використовується API блокчейн-експлорерів. Через запити до API отримуються дані про вхідні та вихідні транзакції, час кожної операції, суми переказів, кількість входів/виходів тощо. Ця інформація лягає в основу подальшої обробки та побудови ознак для кожної адреси.

2.3 Розробка набору ознак

Розробка ознак (feature engeneering) [41] – важливий етап в побудові моделі, адже якість навчання моделі напряму залежить від того, наскільки інформативними та відповідними є використані дані. Процес полягає в перетворенні необроблених (raw data) даних та створенні ознак (features), які будуть використані при прогнозуванні моделлю машинного навчання.

Розробка ознак містити такі техніки:

- Feature transforming – процес трансформації ознак, для більшої інформативності, як наприклад, перетворення категоріальних ознак в числові та навпаки, залежно від вимог моделі.
- Feature scaling – метод стандартизації, який застосовується для приведення ознак до певного числового масштабу, для того, щоб зменшити вплив великих значень на модель.
- Feature extraction/creation – створення ознак на основі вже наявних в датасеті, наприклад шляхом обчислення статистик, або розрахунок нових атрибутів, методом обробки та обчислення сирих даних.

Отримавши помічені адреси та відповідну їм транзакційну інформацію, на попередніх кроках, можливо перейти до етапу створення вибірки ознак, яка буде використовуватися для навчання класифікаційної моделі. Ознаки повинні описувати активність адреси у мережі, її взаємодію з іншими адресами, часові характеристики та інші особливості, які можуть допомогти відрізнити адресу, яка належить до одного класу від іншого.

В контексті криптовалюти Біткоїн та на основі аналізу [42] [43] [44] можна виділити такі основні типи ознак та відповідні ним приклади:

1. **Часові ознаки**

   Дані ознаки описують поведінку адреси у часовому вимірі, як наприклад:

   - Час першої та останньої транзакції;
   - Проміжки між транзакціями;
   - Частота активності (наприклад, кількість транзакцій на день/тиждень);
   - Проміжок життя адреси.
   - Кількість днів з останньої транзакції

2. **Статистичні ознаки**

   Ці характеристики описують загальні числові властивості адреси, включають, але не обмежуються:

   - кількість вхідних/вихідних транзакцій;
   - сума транзакцій ;
   - кількість комісії;

3. **Поведінкові ознаки**

   Допомагають зрозуміти типову модель використання адреси, наприклад:

- Відношення кількості вхідних до вихідних транзакцій;
- Кількість унікальних входів/виходів;
- Частка транзакцій, у яких адреса є єдиним входом/виходом;
- Чи переважають великі одноразові перекази, чи часті мікроплатежі.

Кожен з цих типів ознак відіграє свою роль при класифікації, оскільки дозволяє моделі виявляти як загальні закономірності, так і специфічні патерни транзакційної поведінки, притаманні різним категоріям адрес — від стабільної активності бірж до нерегулярної та маскуючої поведінки даркнет-маркетів чи програм-вимагачів.

2.4 Практичне застосування моделей класифікації

У межах цього дослідження задача визначається як багатокласова класифікація з фіксованою множиною класів, до яких може належати адреса. Визначені класи:

- біржі (exchange);
- майнінгові пули (mining pool);
- даркнет маркети (darknet market);
- сервіси онлайн-гемблінгу (gambling);
- адреси, пов’язані з програмами-вимагачами (ransomware).

Для розв’язання задачі використовуються різні методи машинного навчання, призначені для багатокласової класифікації. Після навчання, вони порівнюються один з одним, за основними метриками, для вибору найбільш ефективної моделі для нашого дослідження.

2.4.1 Логістична регресія

У даному дослідженні логістична регресія використовується як базовий підхід до багатокласової класифікації криптовалютних адрес з використанням Softmax-регресії. Серед переваг моделі — простота реалізації, швидкість навчання та інтерпретованість вагових коефіцієнтів, що допомагає у початковому аналізі ознак.

Однак модель ефективна лише за лінійної роздільності класів і потребує масштабування даних. В умовах складної структури транзакційної поведінки вона часто поступається більш потужним алгоритмам. У нашому випадку логістична регресія виконує роль початкової моделі для порівняння результатів з іншими методами. Дана модель, як і інші в цьому дослідженні, буде реалізована засобами бібліотеки scikit-learn [45], а саме модулем sklearn.linear_model. Для покращення якості класифікації планується провести тестування та підбір таких гіперпараметрів, як метод регуляризації (penalty, наприклад, L1 або L2), коефіцієнт регуляризації (C), а також вибір алгоритму оптимізації (solver).

2.4.2 KNN

Метод KNN використовується у дослідженні як проста та інтуїтивна модель для класифікації криптовалютних адрес на основі подібності їх транзакційної поведінки. Основна ідея полягає в тому, що адреси з подібними характеристиками (наприклад, обсягами переказів, кількістю вхідних/вихідних транзакцій тощо) найімовірніше належать до одного

типу. Клас нової адреси визначається за більшістю серед k найближчих сусідів у навчальній вибірці. Використовується клас KNeighborsClassifier модуля sklearn.neighbors. Ключовими параметрами для оптимізації є кількість сусідів *(k)* та вибір правильної метрики розрахунку відстаней для нашої задачі. Алгоритм чутливий до масштабу ознак, тому потребує попередньої стандартизації. Також він погано масштабований на великі обсяги даних через потребу обчислювати відстані до всіх зразків. Проте завдяки своїй простоті, KNN доцільно використовувати на етапі первинного аналізу, а також як базову модель для порівняння з більш складними підходами.

2.4.3 Decision tree

Для класифікації криптовалютних адрес на п’ять класів — дерева рішень є ефективним інструментом, що дозволяє послідовно розділяти дані на основі транзакційних ознак.

На кожному вузлі дерева обирається ознака, що максимізує чистоту підмножин згідно з критеріями Gini impurity або information gain. Це дозволяє виділити характерні патерни поведінки кожного класу. Наприклад, адреси бірж зазвичай мають високу кількість вхідних транзакцій від великої кількості різних адрес.

Завдяки деревам рішень можна не лише класифікувати адреси, а й інтерпретувати логіку прийняття рішень, вбудованими методами, що є особливо важливим у блокчейн форензиці.

Для реалізації використовується клас DecisionTreeClassifier з модуля sklearn.tree, важливим аспектом при налаштуванні є вибір правильних гіперпараметрів для уникнення можливості перенавчання.

2.4.4 SVM

Метод опорних векторів використовується для класифікації криптовалютних адрес на п’ять класів (біржі, гемблінг, майнінг, даркнет, програми-вимагачі) шляхом побудови гіперплощин, що розділяють класи. У разі нелінійної роздільності ознак застосовуються ядрові функції (радіальне базисне або поліноміальне ядро), які перетворюють дані у вищий простір ознак.

Для багатокласової класифікації використовуються стратегії One-vs-Rest або One-vs-One. Реалізація здійснюється за допомогою класу SVC з модуля sklearn.svm.

Ефективність SVM залежить від вибору ядра, параметрів регуляризації та потребує стандартизації даних. Незважаючи на високу обчислювальну вартість для великих вибірок, метод демонструє високу точність у задачах зі складною структурою даних.

2.4.5 Random Forest

У рамках задачі класифікації криптовалютних адрес за транзакційною поведінкою метод Random Forest виявився доцільним завдяки своїй здатності працювати зі складними, нерівномірно розподіленими наборами ознак, притаманними таким класам, як даркнет-маркети або програми-вимагачі. Завдяки вбудованій структурі випадковості на етапах вибору ознак та підвибірки даних, модель демонструє високу стійкість до перенавчання навіть при значному класовому дисбалансі.

Особливо цінною є властивість Random Forest оцінювати важливість ознак, що дозволяє інтерпретувати вплив таких характеристик, як середній обсяг транзакцій, частота вхідних переказів або середня комісія, на приналежність адреси до певної категорії. Це робить модель не лише ефективним класифікатором, а й інструментом для глибшого аналізу поведінкових шаблонів у блокчейні.

Оскільки передбачення ґрунтуються на колективному рішенні ансамблю дерев, результат є стабільним навіть за наявності шумів або неповних даних, що часто трапляється у відкритих блокчейн-даних.

Random Forest реалізується засобами sklearn.ensemble, а для підвищення якості класифікації здійснюється налаштування таких гіперпараметрів, як кількість дерев, максимальна глибина, мінімальна кількість зразків для поділу та параметри, що враховують дисбаланс між класами.

2.4.6 XGboost

У задачі класифікації криптовалютних адрес XGBoost виступає як один з найбільш ефективних методів для роботи з високовимірними і неоднорідними даними, характерними для транзакційної поведінки гаманців різного типу. Його здатність послідовно фокусуватися на прикладах, які важко класифікувати — наприклад, адреси, що мають риси як майнінгових пулів, так і бірж — дозволяє покращити розрізнення складних або гібридних шаблонів.

На відміну від Random Forest, де дерева працюють незалежно, у XGBoost наступні ітерації моделі адаптуються до помилок попередніх, що дає змогу краще відстежувати приховані закономірності у транзакційних

ознаках, особливо коли мова йде про рідкісні або аномальні класи, як от адреси програм-вимагачів. Завдяки регуляризації, вбудованій у XGBoost, модель краще контролює складність дерев і може ефективно уникати перенавчання при правильному налаштуванні гіперпараметрів. Реалізація в цьому дослідженні здійснюється за допомогою бібліотеки xgboost, з налаштуванням ключових параметрів, зокрема глибини дерев, темпу навчання, кількості ітерацій та ступеня регуляризації, що суттєво впливає на якість класифікації.

2.4.7 MLP

Multilayer Perceptron (MLP) застосовується в рамках задачі класифікації як потужний інструмент для розпізнавання складних, нелінійних залежностей між транзакційними ознаками. У багатокласовій постановці, де потрібно віднести адресу до одного з п’яти класів, вихідний шар мережі містить відповідну кількість нейронів, а результати обчислюються за допомогою функції активації Softmax.

MLP здатна ефективно моделювати взаємозв’язки між характеристиками адрес, як-от кількість вхідних та вихідних транзакцій, середні суми, розподіли комісій тощо. У процесі моделювання використовуються різні конфігурації глибин і ширини мережі, що дозволяє оптимізувати точність і стійкість класифікації. Однак через високу кількість параметрів MLP потребує ретельного налаштування, масштабування вхідних ознак та достатнього об’єму даних для уникнення перенавчання.

Незважаючи на це, при правильному підході модель демонструє конкурентну продуктивність і здатна виявляти як лінійні, так і складні нелінійні шаблони у поведінці криптовалютних адрес.

2.5 Оцінки якості та методи інтерпретації моделі

В результаті кожної, використаної нами моделі класифікації, ми повинні оцінити ефективність класифікації, використовуючи відомі метрики, для того, щоб порівняти між собою моделі та керуючись отриманими оцінками, зробити вибір в бік найбільш відповідної моделі для нашої конкретної задачі. Обравши найкращу модель, на меті стоїть інтерпретація, тобто розуміння чому та як модель приймає рішення.

2.5.1 Метрики для оцінки ефективності класифікації

Перед описом метрик, використаних в даній роботі, треба розглянути таке поняття, як матриця помилок. Матриця помилок – це інструмент для візуалізації результатів класифікації, дозволяє оцінити як модель класифікує кожен клас та які помилки допускає. В контексті багатокласової класифікації матриця помилок має розмірність $n \times n$, де n – це кількість класів. Кожен рядок відповідає реальному класу, а кожен стовпчик передбаченому. Діагональні елементи відображають кількість правильних передбачень, всі інші – помилки класифікації між різними класами. Розглянемо простий приклад в таблиці 2.1, припустимо, що ми класифікуємо адреси в три категорії: клас 0 – адреси даркнету, клас 1 – адреси майнінг пулів, клас 2 – адреси бірж.

Таблиця 2.1 – Матриця помилок

	Передбачення:0	Передбачення:1	Передбачення:2
Факт:0	113	7	3
Факт:1	8	100	6
Факт:2	4	7	108

Розглянемо головну діагональ:

- 113 об’єктів з класу 0 були правильно класифіковані як 0
- 100 об’єктів з класу 1 були правильно класифіковані як 1
- 108 об’єктів з класу 1 були правильно класифіковані як 1

В той самий час:

- 7 об’єктів з класу 0 були помилково класифіковані як клас 1
- 8 об’єктів з класу 1 були помилково класифіковані як клас 0
- 7 об’єктів з класу 2 були помилково класифіковані як клас 1 тощо

В нашій методиці будуть використовуватися стандартні метрики, а саме:
Accuracy, Precision, Recall, F1-Score, але з урахування багатокласової задачі [46].
Розглянемо кожну з цих оцінок більш детально:

1. Accuracy – це загальна частка правильно класифікованих об’єктів серед усіх. Сума елементів на головній діагоналі ділиться на суму усіх елементів матриці.

$$Accuracy = \frac{\text{Кількість правильних передбачень}}{\text{Загальна кількість передбачень}}$$

Precision, Recall, F1-score розраховуються для кожного класу окремо, а потім усереднюються. В даній роботі використано macro averaging – усереднення метрик для кожного класу без урахування розміру класу.

2. Precision – відношення правильно передбачених елементів певного класу (true positives) до загальної кількості об’єктів, передбачених, як цей клас (false positive + true positive).

$$Precision = \frac{TP}{TP+FP}$$

3. Recall – показник здатності моделі виявити всі об’єкти певного класу. Обчислюється як відношення правильно передбачених об’єктів певного класу до загальної кількості об’єктів цього класу в даних

$$Recall = \frac{TP}{TP+FN}$$

4. F1-score – зважене середнє між precision та recall, яке дозволяє збалансовано оцінити якість класифікації, беручи до уваги як точність, так і повноту моделі

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

2.5.2 Методи інтерпретації ознак

Інтерпретація моделей машинного навчання є досить важливим етапом, який полягає в пояснені того, як алгоритм приймає рішення на основі вхідних даних. У контексті класифікації криптовалютних адрес за транзакційною поведінкою це має особливу цінність: ми можемо виявити, які характеристики активності гаманця найбільш інформативні для визначення його типу.

Можемо виділити дві основні категорії для пояснення моделей [47]:

- Модель-залежні (model-specific) методи
- Модель-незалежні (model-agnostic) методи

Модель-залежні – це ті методи, які розраховують важливість ознак, використовуючи внутрішню структуру моделі. Для моделей дерева рішень (Decision Tree, Random Forest, Gradient Boosting), які використовувалися в нашому дослідженні, ми можемо безпосередньо отримати вбудовану оцінку важливості ознак (feature_importances_). Чим більше загальне зменшення невпорядкованості приносить певна ознака – тим вищою є її важливість. Таку оцінку важливості ознак легко візуалізувати стовпчастою діаграмою, що дозволяє наочно порівняти вплив кожної ознаки на класифікацію та зробити висновок щодо найбільш й найменш інформативних характеристик транзакційної поведінки.

Модель-незалежні – методи, які можуть бути застосовані до будь-якої моделі машинного навчання в не залежності від типу та характеристик. Вони зосереджуються на аналізі зміни вихідного прогнозу при зміні вхідних ознак.

- **SHAP (SHapley Additive exPlanations) –** оцінює внесок кожної ознаки в кожне конкретне передбачення. Може надавати як локальну

(для конкретного прикладу) так і глобальну (для моделі в цілому) інтерпретацію. Для глобальної інтерпретації будуть застосовані summary plot та dependence plot. Summary plot надає узагальнене уявлення про важливість ознак і характер їхнього впливу на передбачення. Dependence plot дозволяє проаналізувати взаємозв'язок між значенням окремої ознаки та її впливом на результат моделі, з урахуванням можливої взаємодії з іншими ознаками. Водночас для локальної інтерпретації буде використано force plot та decision plot. Force plot відображає внесок кожної ознаки в конкретне передбачення адреси, показуючи, які транзакційні характеристики вплинули на рішення моделі. Decision plot ілюструє послідовний вплив ознак на формування прогнозу, дозволяючи простежити логіку моделі для окремої адреси.

Висновки до розділу 2

У даному розділі була описана загальна методика класифікації криптовалютних адрес за їх характеристиками. Проаналізувавши відкриті джерела, зроблено висновок, щодо нестачі публічних якісних датасетів для нашого дослідження, тому було сформовано загальну стратегію побудови власного датасету.

З використанням відкритих джерел було ідентифіковано п'ять ключових класів адрес, а саме: біржі, гемблінг-платформи, даркнет-маркети, майнінгові пули та ransomware-гаманці. Надалі, саме ці категорії будуть використані для класифікації.

Окрема увага була приділена розробці ознак, описано часові, статистичні та поведінкові характеристики адреси та зроблено висновок щодо важливості цього етапу, оскільки правильно підібрані ознаки можуть дозволяти моделі знаходити приховані закономірності в даних.

Розглянуті різні моделі машинного навчання, доступні для багатокласової класифікації, виділені особливості кожного алгоритму, їх переваги та недоліки в контексті нашого дослідження.

Було описано методи оцінки якості класифікації, з урахуванням багатокласового випадку. Окрім ефективності, висвітлені методи інтерпретації класифікаційної моделі, включаючи як модель-залежні, так і модель-незалежні техніки.

Таким чином, в розділі охарактеризовані основні кроки для практичної реалізації описаної методики.

3 РЕАЛІЗАЦІЯ ЗАПРОПОНОВАНОЇ МЕТОДИКИ

В даному розділі практично реалізується запропонована методика, яка містити 5 основних етапів, а саме: збір відомих адрес, збір відповідної транзакційної інформації, розробка набору ознак, навчання моделей класифікації, інтерпретація обраної моделі. Схема методики на рисунку 3.1.

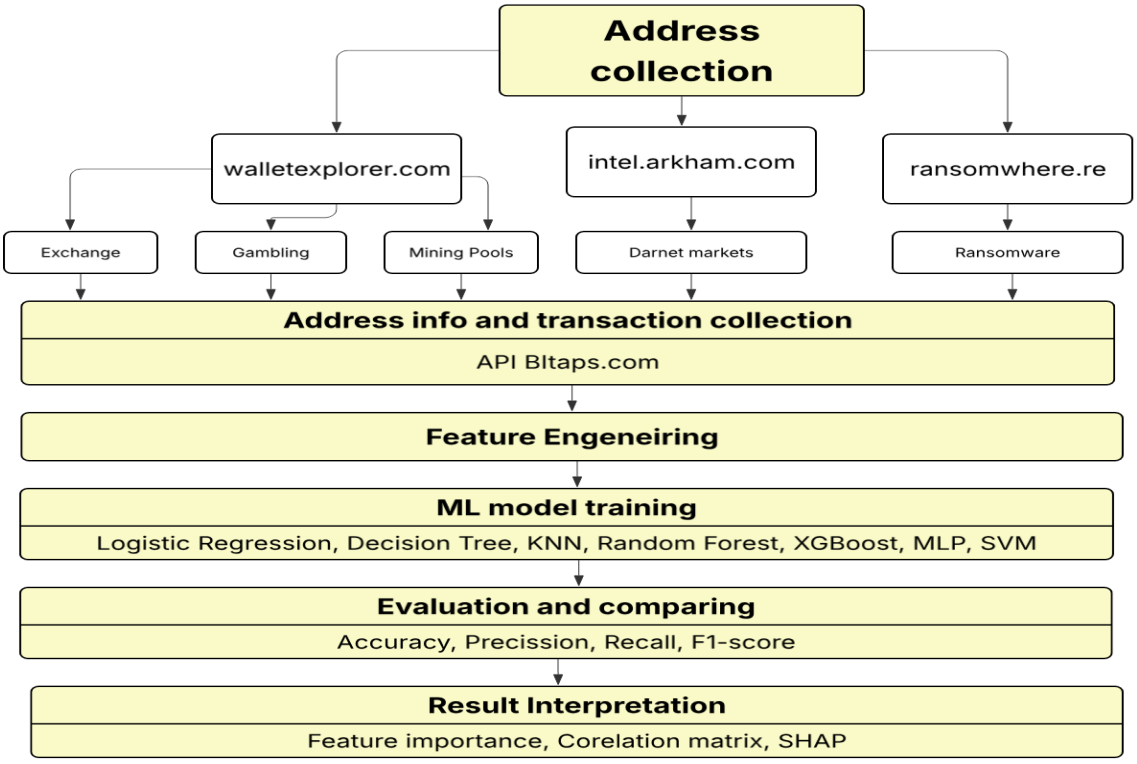


Рисунок 3.1 – Схема розробленої методики

3.1 Збір даних

У цій секції розглядається два початкових етапи нашої методики, а саме: збір адрес з відомою категорією та отримання транзакційної інформації до цих адрес відповідно

3.1.1 Збір адрес

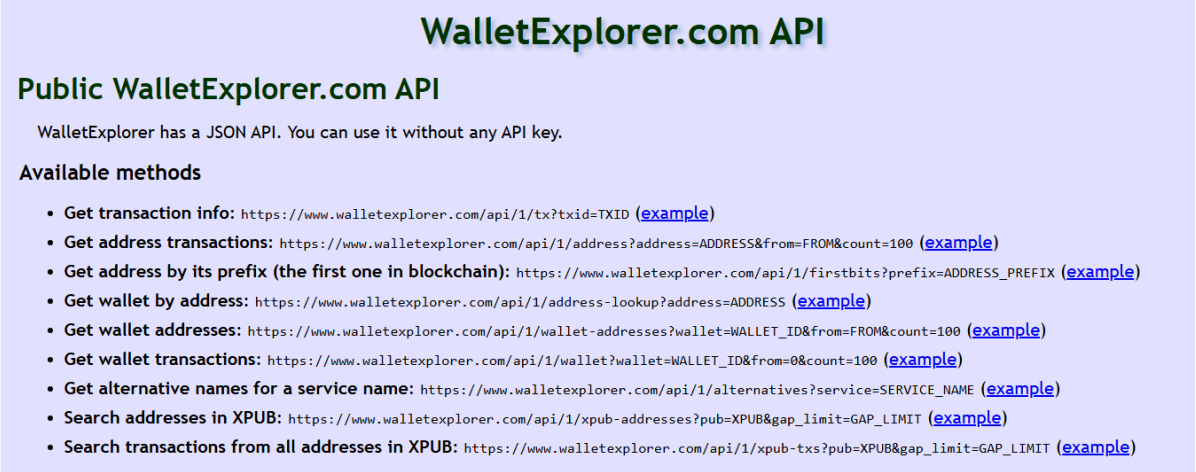
Першим кроком є збір адрес для 5 визначених нами категорій: біржі (exchanges), майнінгові пули (mining pools), гемблінг платформи (gambling), Даркнет маркети (Darknet_Markets), адреси програм вимагачів (Ransomware). Користуючись 3 публічними платформами:

- ransomwhere.re для Ransomware адрес;
- Arkham Intel для Darknet_Markets адрес;
- WalletExplorer для exchanges, mining pools, gambling адрес

було зібрано 1000 адрес кожної категорії, причому для різноманітності джерел адреси були вибрані з різних компаній або сервісів, що належать до відповідної категорії. Для кожної публічної платформи були написані методи мовою python, реалізовані в known_wallets.py, для отримання адрес та парсингу. Розглянемо підхід до кожної:

- WalletExplorer

Має доступний api, що досить полегшує задачу отримання адрес



WalletExplorer.com API

Public WalletExplorer.com API

WalletExplorer has a JSON API. You can use it without any API key.

Available methods

- **Get transaction info:** https://www.walletexplorer.com/api/1/tx?txid=TXID (example)
- **Get address transactions:** https://www.walletexplorer.com/api/1/address?address=ADDRESS&from=FROM&count=100 (example)
- **Get address by its prefix (the first one in blockchain):** https://www.walletexplorer.com/api/1/firstbits?prefix=ADDRESS_PREFIX (example)
- **Get wallet by address:** https://www.walletexplorer.com/api/1/address-lookup?address=ADDRESS (example)
- **Get wallet addresses:** https://www.walletexplorer.com/api/1/wallet-addresses?wallet=WALLET_ID&from=FROM&count=100 (example)
- **Get wallet transactions:** https://www.walletexplorer.com/api/1/wallet?wallet=WALLET_ID&from=0&count=100 (example)
- **Get alternative names for a service name:** https://www.walletexplorer.com/api/1/alternatives?service=SERVICE_NAME (example)
- **Search addresses in XPUB:** https://www.walletexplorer.com/api/1/xpub-addresses?pub=XPUB&gap_limit=GAP_LIMIT (example)
- **Search transactions from all addresses in XPUB:** https://www.walletexplorer.com/api/1/xpub-txs?pub=XPUB&gap_limit=GAP_LIMIT (example)

Рисунок 3.2 – API ресурсу WalletExplorer

До отриманих даних з WalletExplorer були додані ще 2 колонки, які описують тип адреси та до якої організації вона належить. В результаті для кожної організації отримуємо csv файли з адресами.

Приклад для категорії гемблінг, файл **BitZillions.com.csv:**

Unnamed: 0	address	balance	incoming_txs	last_used_in_blc	label	name
0	1changemCPo7	0	79345	673722	Gambling	BitZillions.com
1	1bones2wX8sqC	0	40823	468538	Gambling	BitZillions.com
2	1bonesEeTcABF	0	40220	468538	Gambling	BitZillions.com
3	1bones5gF1HJe	0	37967	380150	Gambling	BitZillions.com
4	1bonesUhqtbLA	0	37110	380150	Gambling	BitZillions.com
5	1bonesBjs3DQL	0	35860	673722	Gambling	BitZillions.com
6	1bonesL3Cvs29	0	34046	496722	Gambling	BitZillions.com
7	1bones8VWK2L	0	28872	360547	Gambling	BitZillions.com
8	1bonesBos9NC2	0	25607	380150	Gambling	BitZillions.com
9	1bones2foMuSF	0	24144	360543	Gambling	BitZillions.com

Рисунок 3.3 – Файл BitZillions.com.csv

- **Arkham Intel**

Досить потужна платформа для блокчейн аналітики, має API, але доступ надається тільки комерційно. Ресурс дозволяє копіювати інформацію щодо останніх транзакцій (рисунок 3.4), тому був розроблений метод парсингу текстового файлу, для витягнення адрес.

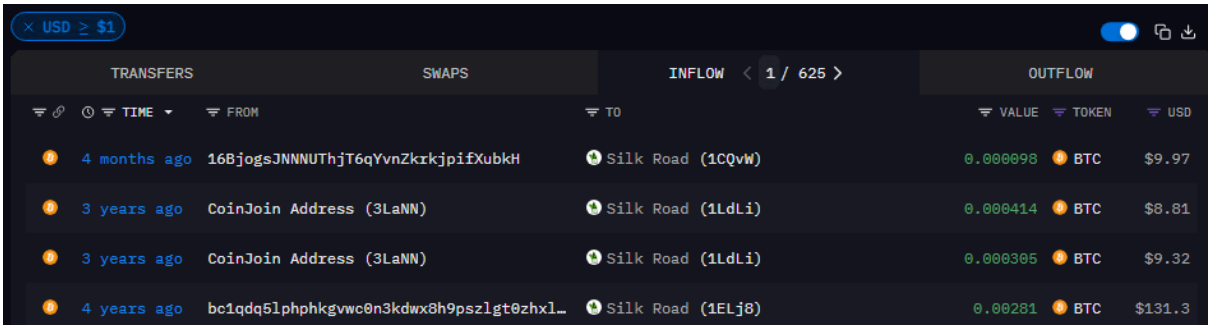


Рисунок 3.4 – Останні транзакції для даркнет маркету Silk Road

Отримуємо файл, який містить адресу, категорію та назву. Результат для файлу Hydra_Market.csv:

	address	label	name
2	12HdKbtSX9J6F	Darknet_Market	Hydra Market
3	12yMstb3Dw2qL	Darknet_Market	Hydra Market
4	13LmQMpaTPM	Darknet_Market	Hydra Market
5	13bTWpNCVKZ	Darknet_Market	Hydra Market
6	14XNWXeqtvN1	Darknet_Market	Hydra Market
7	153jsMxJcUgyiC	Darknet_Market	Hydra Market
8	15YiVsMGNPvh	Darknet_Market	Hydra Market
9	15aTA4Vy8Y63y	Darknet_Market	Hydra Market

Рисунок 3.5 – Файл Hydra_Market.csv

- ransomwhere.re

Платформа для збору адрес, пов’язаних з програмами вимагачами, надає можливість завантажити актуальний список таких адрес у форматі JSON. Структура запису для однієї адреси на рисунку 3.6.

Рисунок 3.6 – Структура даних для адрес ransomware

Після обробки даного JSON файлу, отримуємо результат, файл Ransomware_data_1000.csv:

	address	label	name
2	17TMc2UkVRSc	Ransomware	Netwalker (Mailto)
3	1DTE5x3Rjn2q7	Ransomware	Netwalker (Mailto)
4	3KmK5z4CAvn3	Ransomware	Qlocker
5	1C7FeXMf18mC	Ransomware	Netwalker (Mailto)
6	3PDfzkTnD1E7c	Ransomware	Qlocker
7	3LLzycFNFh7mI	Ransomware	Qlocker
8	37m57HiP5rPce	Ransomware	Qlocker
9	18x7PieotZcmM	Ransomware	Netwalker (Mailto)
10	3Gwz3yVmrGr5A	Ransomware	Qlocker

Рисунок 3.7 – Файл Ransomware_data_1000.csv

3.1.2 Збір транзакційної інформації

Сформувавши вибірки з відомими адресами, маємо можливість переходити до етапу збору транзакційної інформації. Для цієї задачі, в рамках цієї роботи, використовується блокчейн експлорер, які надають доступ до історичних даних про транзакції, пов’язані з конкретними адресами. Для вибору найбільш ефективного інструменту був зроблений порівняльний аналіз найбільш популярних блокчейн експлорерів, продемонстрований в Таблиці 3.1.

Таблиця 3.1 – Порівняння блокчейн експлорерів

Експлорер	**Підтримка API**	**Формат відповіді**	**Ліміт запитів**	**Переваги**	**Недоліки**
BlockCypher	Так	JSON	Безплатна версія: 100 за годину, 1000 за день,	Проста та зрозуміла документація, Достатня швидкість	безплатна версія дуже обмежена в функціоналі
Blockchain.com	Так	JSON	1 запит раз в 10 секунд	Безплатний ресурс, висока надійність, зрозуміла структура відповіді	Занадто строгий ліміт на запити в секунду
SoChain	Так	JSON	Не більше 300 запитів за хвилину	Дуже висока швидкість відповіді, зрозуміла документація, інтуїтивно зрозуміла структура відповідей	Ресурс з появою 3 версії став повністю платним
BitQuery	Так	JSON	10 запитів в хвилину	Явних переваг в безплатній версії не виявлено	Безплатна версія обмежена в функціоналі

Кінець таблиці 3.1

Blockchair	Так	JSON	5 запитів в секунду	Можливість виконання складних фільтрацій	потрібна комерційна версія для великої кількості запитів
Bitaps	Так	JSON	15 запитів раз в 5 секунд	Експлорер конкретно для Bitcoin, безплатний, має чітко структуровану відповідь	Швидкість відповіді інколи може знижуватися

Порівнявши всі описані фактори, та випробувавши кожен з безплатних експлорерів, було прийнято рішення використати Bitaps. Цей ресурс є безплатний, має лояльні обмеження на кількість запитів, зрозумілу документацію та API відповідь, при цьому заточений чітко на мережу Bitcoin, що дає деталізацію для транзакційної інформації.

Рисунок 3.8 – Веб-інтерфейс BitApps

Було реалізовано метод для взаємодії з платформою Bitaps та запису транзакційної інформації у JSON файл. Для кожної адреси кожного типу ми витягнули 50 останніх транзакцій та отримали атрибути. При цьому реалізована логіка безпечного API запиту, без можливості блокування. На рисунку 3.9 можемо переглянути, на прикладі однієї адреси, в якому форматі надається інформація


```
[
  {
    "address": "114e3redgvwP71LxVSqjfVpQ8jwtJt5SPu",
    "label": "Darknet_Market",
    "name": "Buybest Market",
    "balance": 2950088,
    "total_received": 10081013,
    "received_tx": 42,
    "total_sent": 7130925,
    "sent_tx": 18,
    "largest_received_Tx_amount": 731817,
    "largest_spent_Tx_amount": 164862,
    "received_Outs_Count": 42,
    "spent_Outs_Count": 20,
    "type": "PUBKEY+P2PKH",
    "transactions": [
      {
        "page": 1,
        "limit": 50,
        "pages": 2,
        "list": [
          {
            "segwit": false,
            "rbf": false,
            "txId": "7fa040f5b01b47acdc7f0392c94a3c1d099997d957effc66d31042101e20f9c4",
            "version": 1,
            "size": 192,
            "vSize": 192,
            "bSize": 192,
            "lockTime": 0,
            "confirmations": 275826,
            "blockTime": 1584595298,
            "blockIndex": 209,
            "coinbase": false,
            "fee": 18573,
            "data": null,
            "amount": 731817,
            "weight": 768,
            "blockHeight": 622150,
            "timestamp": 1584595298,
            "inputsAmount": 750390,
            "inputAddressCount": 1,
            "outAddressCount": 1,
            "inputsCount": 1,
            "outsCount": 1,
            "outputsAmount": 731817,
            "addressReceived": 731817,
            "addressOuts": 1,
            "addressSent": 0,
            "addressInputs": 0
          },
```


Рисунок 3.9 – Структура відповіді Bitaps

Поля address, label, name додаються нами окремо при кожному запиті, для всіх інших полів значення повертає BitApps. Деякі поля та їх

значення, можуть бути використані нами, як ознаки, одразу, без необхідності обробки, як наприклад: received_tx, sent_tx, received_Outs_Count, spent_Outs_Count. Деякі поля, потребують переведення з одиниць сатоші (0.00000001 BTC) до біткоїна, як наприклад: balance, total_received, total_sent. Транзакційна ж інформація в свою чергу, потребує агрегування та чіткої розробки інформативних ознак.

3.2 Побудова набору ознак

Отримавши транзакційну інформацію для кожної адреси на минулому етапі, ми розробили вибірку з 45 числових ознак, які відображають основні патерни поведінки в блокчейні. Ці ознаки призначені для подальшого використання в задачі класифікації та є результатом як прямого вилучення з сирих даних, так і статистичної агрегації. Мотивація розроблення вибірки базується на припущенні, що адреси одного типу мають однакові характеристики. Для прикладу, адреси залучені до програм вимагачів часто мають нульовий баланс, на момент отримання інформації, оскільки кошти швидко виводяться й адреси перестають функціонувати. Вони також нерідко платять більші комісії, для швидшого переводу активів. Натомість такі сервіси як біржі, мають великі значення надісланої\отриманої криптовалюти та високу транзакційну активність.

В таблиці 3 наведений перелік розроблених нами ознак та короткий опис до них. Всього було отримано 13 ознак та для більшості з них обчислено статистичні характеристики, а саме: максимум, мінімум, сума, середня, стандартне відхилення.

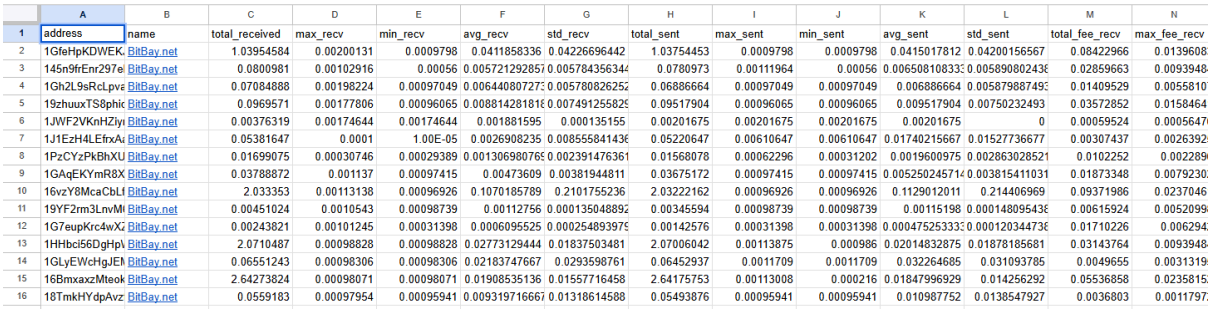
Таблиця 3.2 – Перелік ознак

adress	адреса
name	назва сутності
received BTC (total\max\min\avg\std)	Кількість отриманих Біткоїнів адресою (total\max\min\avg\std)
sent BTC (total\max\min\avg\std)	Кількість надісланих біткоїнів адресою (total\max\min\avg\std)
received fee (total\max\min\avg\std)	Кількість комісії для вхідних транзакцій (total\max\min\avg\std)
sent fee (total\max\min\avg\std)	Кількість комісії для вихідних транзакцій (total\max\min\avg\std)
received transaction size (total\max\min\avg\std)	Розмір вхідних транзакцій (total\max\min\avg\std)
sent transaction size (total\max\min\avg\std)	Розмір вихідних транзакцій (total\max\min\avg\std)
transaction inputs (total\max\min\avg\std)	Кількість входів в транзакції (total\max\min\avg\std)
transaction outputs (total\max\min\avg\std)	Кількість виходів в транзакції (total\max\min\avg\std)

Кінець таблиці 3.2

rbf ratio	Частка Replace-By-Fee транзакцій серед звичайних
balance	Баланс адреси на момент отримання інформації
n_tx	Загальна кількість транзакцій адреси
n_tx_received	Кількість вхідних транзакцій
n_tx_sent	Кількість вихідних транзакцій

На рисунках 3.10 та 3.11 можемо переглянути перші записи отриманого датасету.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	address	name	total_received	max_recv	min_recv	avg_recv	std_recv	total_sent	max_sent	min_sent	avg_sent	std_sent	total_fee_recv	max_fee_recv
2	1GfeHpKDWEK.	BitBay.net	1.03954584	0.00200131	0.0009798	0.0411858336	0.04226696442	1.03754453	0.0009798	0.0009798	0.0415017812	0.04200156567	0.08422966	0.0139608
3	145n9frEnr297e	BitBay.net	0.0800981	0.00102916	0.00056	0.0057212928	0.00578435634	0.0780973	0.00111964	0.00056	0.0065081083	0.00589080243	0.02859663	0.0093948
4	1Gh2L9sRcLpva	BitBay.net	0.07084888	0.00198224	0.00097049	0.0064408072	0.00578082625	0.06886664	0.00097049	0.00097049	0.006886664	0.00587988749	0.01409529	0.0055810
5	19zhuuxTS8phic	BitBay.net	0.0969571	0.00177806	0.00096065	0.0088142818	0.00749125582	0.09517904	0.00096065	0.00096065	0.009517904	0.00750232493	0.03572852	0.0158464
6	1JWF2VKnHZiy	BitBay.net	0.00376319	0.00174644	0.00174644	0.001881595	0.000135155	0.00201675	0.00201675	0.00201675	0.00201675	0	0.00059524	0.0005647
7	1J1EzH4LEfrxA	BitBay.net	0.05381647	0.0001	1.00E-05	0.0026908235	0.00855584143	0.05220647	0.00610647	0.00610647	0.01740215667	0.01527736677	0.00307437	0.0026392
8	1PzCYzPkBhXU	BitBay.net	0.01699075	0.00030746	0.00029389	0.0013069807	0.00239147636	0.01568078	0.00062296	0.00031202	0.0019600975	0.00286302852	0.0102252	0.002289
9	1GAqEKYmR8X	BitBay.net	0.03788872	0.001137	0.00097415	0.00473609	0.00381944811	0.03675172	0.00097415	0.00097415	0.005250245714	0.00381541103	0.01873348	0.0079230
10	16vzY8McaCbLf	BitBay.net	2.033353	0.00113138	0.00096926	0.1070185789	0.2101755236	2.03222162	0.00096926	0.00096926	0.1129012011	0.214406969	0.09371986	0.0237046
11	19YF2rm3LnvM	BitBay.net	0.00451024	0.0010543	0.00098739	0.00112756	0.00013504889	0.00345594	0.00098739	0.00098739	0.00115198	0.00014809543	0.00615924	0.0052099
12	1G7eupKrc4wX	BitBay.net	0.00243821	0.00101245	0.00031398	0.0006095525	0.00025489397	0.00142576	0.00031398	0.00031398	0.000475253333	0.00012034473	0.01710226	0.006294
13	1HHbci56DgHpV	BitBay.net	2.0710487	0.00098828	0.00098828	0.02773129444	0.01837503481	2.07006042	0.00113875	0.000986	0.02014832875	0.01878185681	0.03143764	0.0093948
14	1GLyEWcHgJEh	BitBay.net	0.06551243	0.00098306	0.00098306	0.02183747667	0.0293598761	0.06452937	0.0011709	0.0011709	0.032264685	0.031093785	0.0049655	0.0031319
15	16BmxaxzMteok	BitBay.net	2.64273824	0.00098071	0.00098071	0.01908535136	0.01557716458	2.64175753	0.00113008	0.000216	0.01847996929	0.014256292	0.05536858	0.0235815
16	18TmkHYdpAvz	BitBay.net	0.0559183	0.00097954	0.00095941	0.0093197166	0.01318614588	0.05493876	0.00095941	0.00095941	0.010987752	0.0138547927	0.0036803	0.0011797

Рисунок 3.10 – Перша частина датасету

_	avg_inputs_per_	std_inputs_per_t	total_outputs	max_outputs_pe	min_outputs_pe	avg_outputs_pe	std_outputs_per	balance	n_tx	n_tx_received	n_tx_sent	label
	159.2	179.2044642	1923	201	1	38.46	64.38515667	0.00200131	51	26	25	Exchanges
	152.4230769	170.8258612	1371	201	1	52.73076923	74.56208445	0.0020008	26	14	12	Exchanges
	125.4285714	124.8433576	1522	201	1	72.47619048	84.60474998	0.00198224	21	11	10	Exchanges
	148.2857143	159.5165741	1329	201	1	63.28571429	80.33213367	0.00177806	21	11	10	Exchanges
	144	202.2325394	26	23	1	8.666666667	10.14341604	0.00174644	3	2	1	Exchanges
	2.391304348	2.85503568	334	290	2	14.52173913	58.73216256	0.00161	23	20	3	Exchanges
	111.7619048	151.0635012	1567	250	1	74.61904762	111.3174252	0.00130997	21	13	8	Exchanges
	122.2	137.8720179	1060	204	1	70.66666667	82.83692145	0.001137	15	8	7	Exchanges
	141.2162162	164.6168582	1227	202	1	33.16216216	61.49995101	0.00113138	37	19	18	Exchanges
	90.28571429	89.07094169	809	202	1	115.5714286	98.93411537	0.0010543	7	4	3	Exchanges
	145.2857143	185.3249766	897	250	1	128.1428571	111.6697025	0.00101245	7	4	3	Exchanges
	187.5	157.168222	1369	201	1	27.38	51.72886622	0.00098828	178	100	78	Exchanges
	95.6	108.3597711	410	201	1	82	89.63481466	0.00098306	5	3	2	Exchanges
	177.34	166.7211576	1776	201	1	35.52	55.56554328	0.00098071	115	60	55	Exchanges
	165.3636364	215.2939946	653	201	1	59.36363636	84.3017014	0.00097954	11	6	5	Exchanges

Рисунок 3.11 – Друга частина датасету

Розроблений набір із 45 ознак можна умовно поділити на кілька логічних категорій, кожна з яких відображає різні аспекти транзакційної активності адреси. Така класифікація допомагає краще зрозуміти мотивацію створення ознак та їхню роль у майбутній моделі класифікації:

Таблиця 3.3 – Опис ознак

Назва категорії	Перелік ознак	Пояснення
Об’єми адреси	total_received, total_sent, balance	Характеризують загальну кількість криптовалюти, що пройшла через адресу. Адреси пулів і бірж часто мають великі значення total_received, total_sent, при нульовому балансі.
Статистики по сумам транзакцій	avg_sent, max_received, min_sent тощо	Ці ознаки корисні для виявлення типової та не типової поведінки кожної категорії. Наприклад майнінгові пули часто мають високі значення total_sent, бо вони регулярно розподіляють винагороди.

Продовження таблиці 3.3

Розміри транзакцій	avg_tx_size_received, max_tx_size_sent тощо	Скільки байт займають транзакції. Великі значення означають багато входів/виходів, які притаманні даркнет маркетам, які часто користуються міксерами та агрегують, потім розподіляючи кошти. Водночас адреси програм вимагачів часто не мають складних транзакцій
Кількість входів/виходів	total_outputs, std_inputs_per_tx тощо	Визначає структуру транзакцій. Багато входів — агрегація коштів, багато виходів — розподіл. Біржі часто мають, як велику кількість входів, так і виходів, даркнет маркети велику кількість входів, майнінгові пули – велику кількість виходів.
Комісії	avg_fee_sent, max_fee_sent тощо	Витрати на комісію, великі комісії можуть пришвидшити процес переказу коштів. Адреси програм-вимагачів часто використовують великі комісії для швидкого переказу

Кінець таблиці 3.3

Загальні характеристики	n_tx, n_tx_sent, n_tx_received, rbf_ratio	Характеризують активність адреси та використання механізму rbf, який дозволяє замінити не підтверджену транзакцію, на нову версію з більшою комісією, для швидшого підтвердження. Майнінговим пулам та Гемблінг платформам притаманна велика кількість транзакцій, при цьому низький відсоток вихідних. Водночас адреси програм-вимагачів часто використовуються в ролі транзиту та мають 1 або декілька транзакцій

3.3 Порівняння та вибір класифікаційної моделі

В межах даного дослідження, ми класифікуємо криптовалютні адреси за транзакційною інформацією, користуючись сімома моделями, з метою знайти найбільш ефективну для нашої задачі. Моделі включають:

- Logistic Regression
- SVM (Support Vector Machine)
- KNN (k-Nearest Neighbours)
- Decision tree
- Random Forest
- XGBoost

- MLP (Multi Layer Perceptron)

Для ефективного навчання кожна з цих моделей потребує правильного підбору гіперпараметрів, для цього використовується техніка Grid search [30].

Для правильної роботи перерахованих моделей цільова змінна повинна бути числовою, тому ми застосували кодування міток, можемо переглянути тип адреси та відповідну їй мітку в таблиці 4.

Таблиця 3.4 – Мітки класів

Тип адреси	Мітка
Exchanges	4
Gambling	3
Pools	2
Darknet_Market	1
Ransomware	0

3.3.1 Підбір параметрів GridSearch

Для досягнення оптимальної якості класифікації було використано метод GridSearchCV, що є одним із найпоширеніших підходів до налаштування гіперпараметрів машинного навчання. Цей метод реалізує вичерпний перебір усіх можливих комбінацій параметрів, заданих у сітці (grid), та обирає найкращу конфігурацію на основі обраної метрики оцінки якості моделі. В якості метрики була вибрана точність, в результаті отриманий такий набір найкращих параметрів для кожної моделі:

- Logistic Regression

```
Tested model: Logistic Regression
Best parameters: {'model__C': 10, 'model__multi_class': 'multinomial', 'model__penalty': 'l2', 'model__solver': 'lbfgs'}
Accuracy on test: 0.7531
```

Рисунок 3.12 – Параметри для Logistic Regression

- Support Vector Machine

```
Tested model: SVM
Best parameters: {'model__C': 10, 'model__decision_function_shape': 'ovo', 'model__gamma': 'scale', 'model__kernel': 'rbf'}
Accuracy on test: 0.7895
```

Рисунок 3.13 – Параметри для SVM

- XGBoost

```
Tested model: XGBoost
Best parameters: {'model__gamma': 0, 'model__learning_rate': 0.1, 'model__max_depth': 5, 'model__n_estimators': 200, 'model__subsample': 0.8}
Accuracy on test: 0.9464
```

Рисунок 3.14 – Параметри для XGBoost

- RandomForest

```
Tested model: RandomForest
Best parameters: {'model__max_depth': None, 'model__max_features': 'sqrt', 'model__min_samples_leaf': 1, 'model__min_samples_split': 2, 'model__n_estimators': 200}
Accuracy on test: 0.9305
```

Рисунок 3.15 – Параметри для Random Forest

- KNN

```
Tested model: KNN
Best parameters: {'model__metric': 'manhattan', 'model__n_neighbors': 3, 'model__weights': 'distance'}
Accuracy on test: 0.8776
```

Рисунок 3.16 – Параметри для KNN

- Decision Tree

```
Tested model: Decision Tree
Best parameters: {'model__criterion': 'entropy', 'model__max_depth': None, 'model__min_samples_split': 5}
Accuracy on test: 0.8637
```

Рисунок 3.17 – Параметри для Decision Tree

- MLP

```
Tested model: MLP
Best parameters: {'model__activation': 'relu', 'model__alpha': 0.0001, 'model__hidden_layer_sizes': (100, 50), 'model__learning_rate': 'adaptive', 'model__solver': 'adam'}
Accuracy on test: 0.8709
```

Рисунок 3.18 – Параметри для MLP

3.3.2 Навчання моделей. Оцінка ефективності

Визначивши гіперпараметри для кожної з моделей, ми навчили 7 класифікаторів на навчальних даних, причому розмір навчальної вибірки склав 70% від всієї, а тестової відповідно 30%. Для покращення результативності деяких алгоритмів була застосована стандартизація. Тестова вибірка використовувалася для оцінки ефективності кожної моделі, за визначеними нами метриками: accuracy, precision, recall, f1-score. Обраховані загальні показники, користуючись macro avg.

В таблиці 5 можемо переглянути результати класифікації. Можемо зробити висновок, що ансамблеві моделі деревоподібної структури, XGBoost та RandomForest, демонструють найвищу ефективність на тестових даних за всіма метриками, перевищуючи позначку в 94 та 93 відсотки відповідно. Це свідчить про те, що для отриманих ознак та для даного розміру вибірки ці алгоритми краще ніж інші, знаходять закономірності у транзакційній інформації.

Моделі логістичної регресії та SVM показали слабші результати, оскільки вони все ж таки більш заточені на бінарні випадки і не можуть бути гнучкими й ефективно узагальнювати дані для багатокласової класифікації. MLP продемонстрував конкурентоздатну ефективність, для більшого набору даних, ніж той, що розглядається в даній роботі, ця

модель може показувати кращі результати ніж інші, хоча й потребує достатніх обчислювальних ресурсів та часу навчання.

Таблиця 3.5 – Результати класифікації

	accuracy	macro avg precision	macro avg recall	macro avg f1-score
Decision Tree	0.88	0.88	0.88	0.88
Random Forest	**0.931**	**0.931**	**0.931**	**0.931**
KNN	0.865	0.866	0.866	0.865
SVM	0.778	0.788	0.781	0.779
XGBoost	**0.944**	**0.944**	**0.944**	**0.944**
Logistic Regression	0.748	0.768	0.752	0.748
MLP	0.858	0.859	0.86	0.859

Таким чином, результати підтверджують доцільність розробленої вибірки ознак, яка дозволяє отримати точність більше 93% для двох моделей. Фінальною моделлю для подальшої інтерпретації була вибрана XGBoost, як найбільш ефективна за усіма показниками.

3.4 Інтерпретація обраної моделі

Після тестування різних моделей машинного навчання наступним кроком є інтерпретація фінальної моделі з метою кращого розуміння поведінки, обґрунтованість рішень та впливу конкретних ознак на

класифікацію кожного типу адрес. В таблиці 6 наведені результати класифікації по кожній категорії окремо, як можемо бачити, загалом усі оцінки f1-score знаходяться в межах 0.94 - 0.95, це свідчить про збалансованість моделі в класифікації різних категорій адрес. Клас Exchanges продемонстрував найвищу повноту (**recall = 0.97**), що означає, що модель майже не пропускає адреси цього типу.

Таблиця 3.6 – Результат класифікації по класам для XGBoost

	precision	recall	f1-score
Ransomware	0.95	0.94	0.95
Darknet Markets	0.95	0.92	0.94
Pools	0.93	0.95	0.94
Gambling	0.96	0.94	0.95
Exchanges	0.93	0.97	0.95

На рисунку 3.19 продемонстрована матриця помилок для моделі XGBoost, головна діагональ представляє правильні передбачення та підтверджує ефективність обраного методу. Як видно з графіка найбільша плутанина припадає на пари гемблінг сервісів та бірж, зокрема у 17 випадках адреси категорії Gambling були класифіковані як Exchanges. Це може бути пов’язано з подібною транзакційною поведінкою таких типів адрес, оскільки обидві ці категорії взаємодіють з великою кількістю сторонніх адрес, мають багато входів та виходів. Цей факт може бути темою для майбутніх робіт і розробки більш комплексних ознак, які б чітко розділяли ці класи.

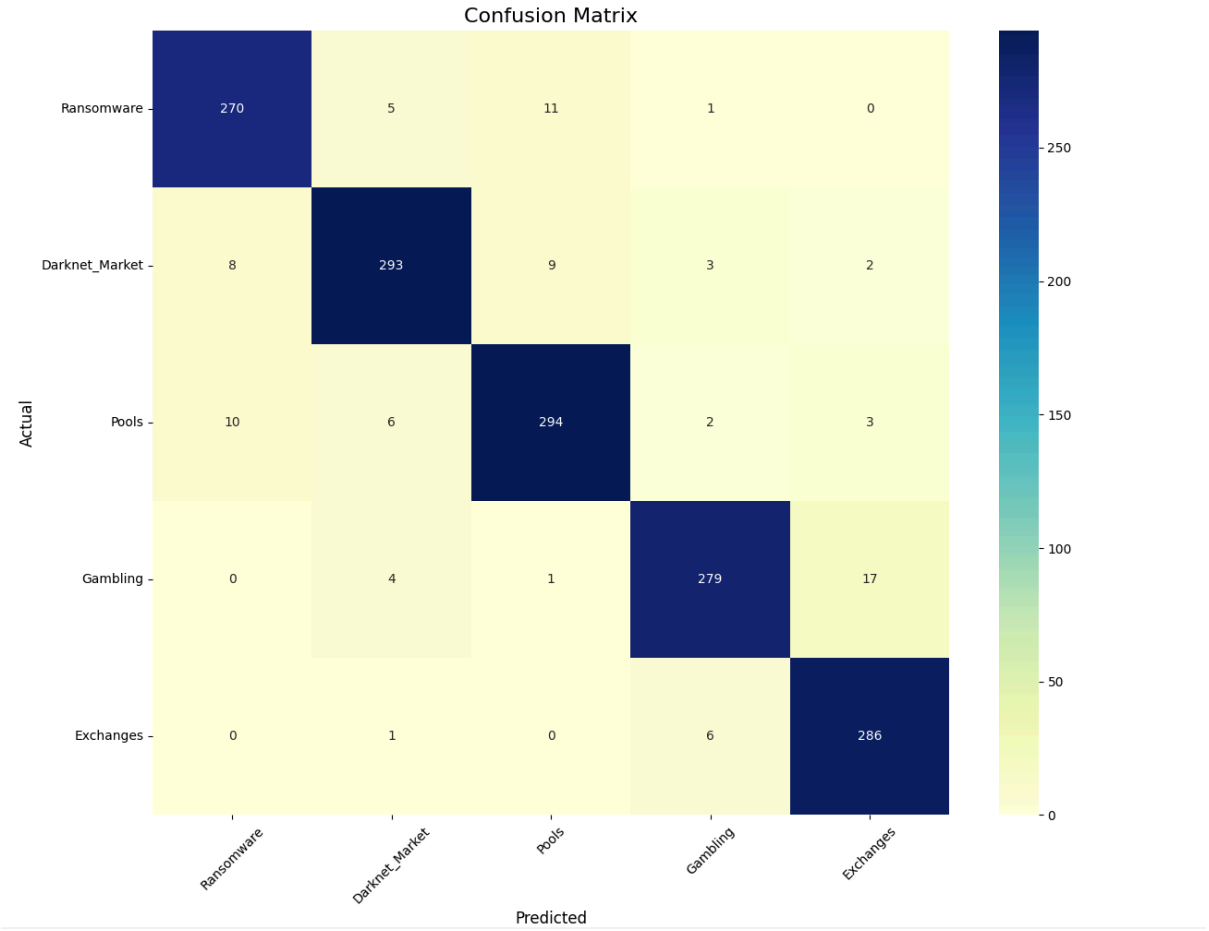


Рисунок 3.19 – Confusion Matrix

На рисунку 3.20 наведено графік важливості ознак, найбільший вплив на результат класифікації виявили ознаки min_sent, max_outputs_per_tx, total_tx_size_recv.

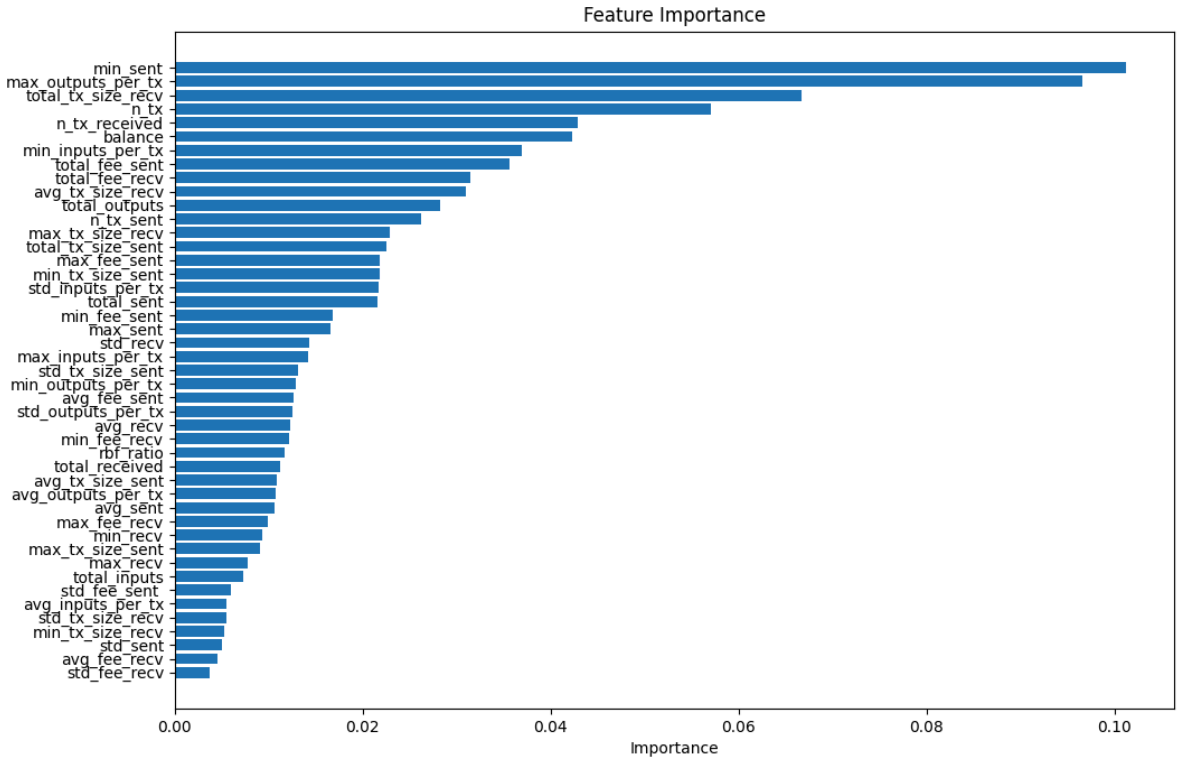


Рисунок 3.20 – Feature Importance

Переглянемо агреговані характеристики кожної з ознак, для розуміння їх важливості:

- Аналіз розподілу ознаки **min_sent** для різних категорій адрес (рисунок 3.21) демонструє суттєві відмінності у мінімальних сумах переказів між класами. Категорія Ransomware має досить високе середнє і медіанне значення цієї ознаки, це свідчить про те, що ці адреси часто використовуються для одного або декількох невеликих платежів. Pools мають найбільше середнє, але дуже низьку медіану і надзвичайно велике стандартне відхилення, що означає на різко змінну поведінку – деякі транзакції мають дуже великі мінімальні суми, але більшість — малі. Це типовий патерн для адрес, що пов’язані з майнінгом. Darknet_Markets та Gambling мають малі значення **min_sent,** оскільки Даркнет маркети части використовують мікроплатежі на вивід, для заплутування потоку коштів, а гемблінг

платформи, в свою чергу, роблять багато дрібних виплат для гравців. Exchanges мають найнижчу медіану і водночас значне стандартне відхилення, що говорить про широкий діапазон мінімальних переказів, це логічно, оскільки різні користувачі мають різний об'єм транзакцій. Таким чином **min_sent** дійсно є досить інформативною ознакою, бо вона не просто вимірює найменші перекази, а фактично відображає фінансову поведінку певних типів адрес.

label	mean	median	std	sum
Darknet_Market	0.231145	0.024354	0.627946	237.848224
Exchanges	0.162230	0.000331	3.265901	160.607799
Gambling	0.078030	0.002232	0.742440	77.015479
Pools	9.782228	0.010402	213.665066	9567.018518
Ransomware	4.176275	1.500000	25.419508	4385.088610

Рисунок 3.21 – Агреговані характеристики для min_sent

- Для ознаки **max_outputs_per_tx** можемо виділити, що біржі та майнінгові пули мають дуже високі показники виходів, що підтверджує характер їхньої діяльності. Біржі оброблюють транзакції клієнтів часто використовуючи багато виході, майнінгові пули розподіляють нагороди численним майнерам. Ransomware мають найнижчі значення, оскільки часто адреси такого типу мають декілька транзакцій та використовуються в якості транзитних.

label	mean	median	std	sum
Darknet_Market	19.827017	4.0	48.616093	20402
Exchanges	2095.881818	501.0	2951.869934	2074923
Gambling	190.498480	9.0	522.838229	188022
Pools	1118.570552	13.0	2725.663267	1093962
Ransomware	10.432381	2.0	85.279296	10954

Рисунок 3.22 – Агреговані характеристики для max_outputs_per_tx

- Для **total_tx_size_recv** можна зазначити, що Exchanges мають надзвичайно високі значення середнього та стандартного відхилення, що означає велику кількість комплексних транзакцій, з багатьма входами та виходами. Pools, в свою чергу, має велике середнє значення, але низьку медіану, що свідчить про те, що більшість транзакцій є малими, але є декілька виключно великих. Злочинні категорії, як Ransomware та Darknet_Market показують, що для цих класів часто використовуються не комплексні перекази, які не містять одночасно багато входів та виходів.

label	mean	median	std	sum
Darknet_Market	6.957597e+03	1782.0	3.055849e+04	7159367
Exchanges	1.179653e+06	163170.0	1.769018e+06	1167856200
Gambling	4.260327e+04	10691.0	1.637064e+05	42049425
Pools	5.732475e+05	6815.0	1.636418e+06	560636073
Ransomware	3.088960e+03	373.0	1.563625e+04	3243408

Рисунок 3.23 – Агреговані характеристики для total_tx_size_recv

- Аналіз ознаки **n_tx** демонструє, що гемблінг платформи мають найвищу кількість транзакцій, що обумовлюється постійним прийняттям ставок та надсиланням винагород. Біржі теж мають досить високе середнє значення транзакцій, оскільки ця категорія здійснює численні транзакції для обслуговування клієнтів. Ransomware мають найменші показники транзакцій, що і властиво даній категорії, зазвичай адреси програм вимагачів використовуються як одноразові для отримання одиничного платежу.

label	mean	median	std	sum
Darknet_Market	43.482021	6.0	190.479502	44743
Exchanges	1408.993939	497.0	20324.050396	1394904
Gambling	2806.897670	251.0	14575.490139	2770408
Pools	293.838446	4.0	900.152501	287374
Ransomware	4.907619	2.0	27.465830	5153

Рисунок 3.24 – Агреговані характеристики для n_tx

Після інтерпретації рішень моделі за допомогою вбудованого feature importane виникла потреба у глибшому аналізі впливу окремих ознак на результати класифікації. Для цього було застосовано метод SHAP, а саме summary_plot, який дозволяє нам визначити, які ознаки позитивно впливають, а які негативно. Цей графік відображає розподіл значень SHAP для кожної ознаки, що дає змогу побачити не лише ступінь важливості ознаки, але й напрямок її впливу (підвищення або зниження ймовірності належності до певного класу).

Кожна крапка на графіку – один приклад із тестової вибірки. По вісі X значення SHAP, наскільки ознака зсунула передбачення моделі в напрямку класу (ліворуч – негативний вплив, праворуч – позитивний).

Приклади з найвищими значеннями мають червоний колір, з низькими – синій. По вісі Y розташовані назви ознак.

Можемо переглянути графіки для категорій Ransomware, Darknet_Markets, Pools, Gambling на рисунку 3.25, коротко розглянемо кожну:

1. **Ransomware** – n_tx_sent(кількість відправлених транзакцій), (низькі значення) — зсувають модель у бік передбачення "Ransomware". (високі значення) — знижують цю ймовірність. Це може свідчити про факт, що адреси цього типу часто є транзитними і не мають великої кількості надісланих транзакцій; min_sent (високі значення) — зсувають модель у бік передбачення "Ransomware". (низькі значення) — знижують цю ймовірність. rbf_ratio Високе значення (рожеві точки праворуч) зміщує прогноз у бік ransomware. Можливо, Ransomware часто використовує транзакції з Replace-by-Fee (RBF) — це може бути індикатором спроби швидкого виводу, що є типовим для таких злочинів.
2. **Darknet_Markets** – великі значення стандартного відхилення входів для транзакції (std_inputs_per_tx) позитивно впливає на класифікацію, в той же час велика кількість отриманих транзакцій знижує ймовірність класифікації адреси як даркнет маркет, хоча це й не зовсім типово для такого типу діяльності. Водночас ознаки пов’язані з комісією для відправлення (total_fee_sent, max_fee_sent), великі значення впливають позитивно, що характерно для даркнет маркетів, оскільки вони часто намагаються максимально пришвидшити вивід коштів.
3. **Pools** – низькі значення балансу зміщують передбачення в сторону класу pools, це пов’язано з тим, що майнінгові пули швидко розподіляють нагороди, не тримаючи залишків. avg recv, max recv теж мають позитивний вплив на класифікацію, оскільки майнінгові пули отримують велику кількість винагород. Майнінгові пули також мають високу частку найменших входів в транзакціях що пов’язано з об’єднанням внесків багатьох майнерів у пулі.
4. **Gambling** – великі значення кількості транзакцій (n_tx_received, n_tx) зсувають модель у бік передбачення цього класу, це й не дивно оскільки сервіс такого типу працює з великою кількістю різних користувачів та їх адрес. Низький баланс частіше пов’язаний зі зменшенням ймовірності Gambling.

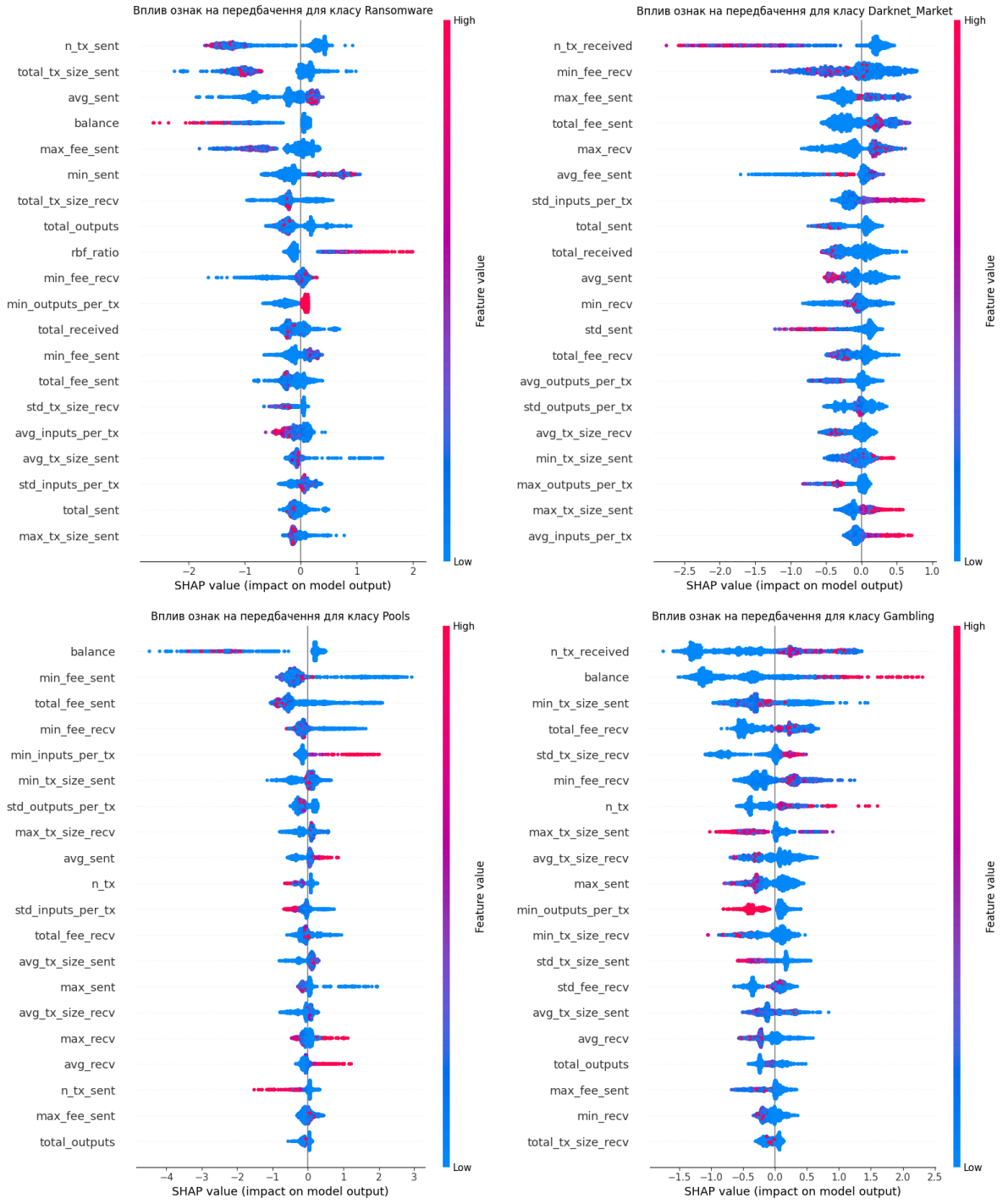


Рисунок 3.25 – Вплив ознак на передбачення для 4 класів

На рисунку 3.26 розглянемо графік для категорії Exchanges:

- Exchanges – Висока кількість транзакцій, особливо n_tx_received збільшує ймовірність адреси бути класом біржі. max_inputs_per_tx, max_outputs_per_tx, високе значення позитивно впливає на класифікацію. Біржі часто мають дуже велику кількість транзакції та багато входів та виходів до них. Водночас ознака balance не є такою однозначною, низькі значення балансу можуть як позитивно, так і негативно впливати на віднесення прикладу до цієї категорії

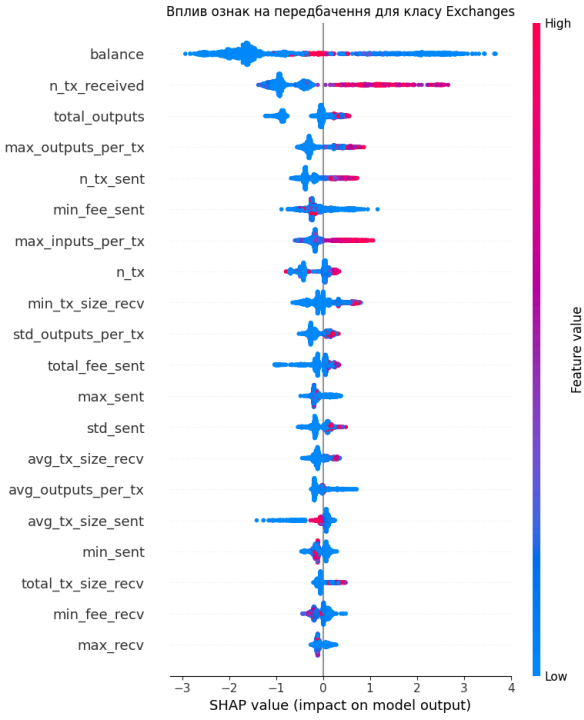


Рисунок 3.26 – Вплив ознак на передбачення для класу Exchanges

Висновки до розділу 3

У даному розділі було практично імплементовано методику по класифікації криптовалютних адрес за транзакційною поведінкою. Зібрано

реальні адреси відомих сутностей, які відносяться до 5 категорій, а саме: біржі, майнінгові пули, гемблінг платформи, адреси даркнет маркетів, адреси пов’язані з програмами вимагачами. Отримано відповідно транзакційну інформацію для кожної адреси, користуючись відкритим API блокчейн експлореру. На основі зібраної інформації сформовано вибірку ознак, що описують поведінку адрес. Розробка ознак дозволила виділити відмінні патерни для кожної категорії адрес, що є досить важливим для подальшої класифікації. До сформованої вибірки було застосовано класифікацію з використанням семи різних моделей машинного навчання, порівнявши результати роботи кожного з цих алгоритмів за чотирма метриками, зроблено висновок щодо найбільшої ефективності XGBoost та Random Forest для даного датасету. Здійснено інтерпретацію класифікації, обраної моделі XGBoost, візуалізація впливу ознак дозволила встановити ключові характеристики, що найбільше впливають на прийняття рішень моделлю. Інтерпретація результатів є важливою складовою дослідження, оскільки підвищує довіру до методики та відкриває можливості для її подальшого вдосконалення.

ВИСНОВКИ

У межах даного дослідження було розроблено методику автоматичної класифікації криптовалютних адрес на основі їх транзакційної поведінки з використанням методів машинного навчання. З метою побудови ефективної моделі класифікації був створений власний датасет, який містить адреси, що належать до п’яти різних категорій: біржі, майнінгові пули, гемблінг-платформи, даркнет-маркети та адреси, пов’язані з програмами-вимагачами. Для кожної з адрес було зібрано статистичні та поведінкові характеристики транзакцій за допомогою API блокчейн-експлорерів. На основі цих даних сформовано вибірку з 45 ознак, що описують транзакційну активність.

Було протестовано низку алгоритмів машинного навчання для мультикласової класифікації. Сформований датасет дозволив досягти високої якості класифікації. Усі ознаки були ретельно підібрані з урахуванням реальної транзакційної активності, що забезпечило інформативність вибірки та її релевантність для задачі мультикласової класифікації. За результатами порівняльного аналізу, на основі 4 метрик, було встановлено, що модель XGBoost демонструє найкращі показники — понад 94% по всіх ключових метриках. Така стабільна продуктивність свідчить про високу інформативність та структурованість розробленої вибірки. Отриманий результат підтверджує доцільність обраного підходу до побудови ознак, а також вказує на відповідність вибірки реальним відмінностям між категоріями адрес.

Окрім оцінки якості класифікації, було проведено інтерпретацію моделей з використанням методу SHAP для виявлення найбільш впливових ознак у прийнятті рішень. Аналіз показав, що для кожної з

категорій адрес (біржі, майнінг-пули, даркнет, гемблінг, ransomware) існують характерні шаблони транзакційної поведінки, які найсильніше впливають на класифікацію. Це дозволяє не лише підвищити ефективність моделі, а й поглибити розуміння функціонування різних типів криптовалютних сутностей.

Подальший розвиток запропонованої методики вбачається в розширенні набору ознак, шляхом врахування більш складних характеристик транзакційної поведінки, зокрема часових послідовностей, графових структур, обчисленні більш комплексних залежностей. Додаткове залучення інших методів блокчейн форензики може створити передумови для розробки комплексної системи моніторингу криптовалютної активності в реальному часі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Chainalysis. Official Website. – URL: https://www.chainalysis.com/

2. Curtis, Dehv. Intro to Blockchain Forensics. – Medium. – URL: https://medium.com/@dehvcurtis/intro-to-blockchain-forensics-cdac3c782426

3. Garima Singh. A Comprehensive Guide to Crypto Forensic. – LinkedIn. – URL: https://www.linkedin.com/pulse/comprehensive-guide-crypto-forensic-garima-singh-vkqlf/

4. Sourabh Kumar Das. Unveiling the Secrets of the Chain: Blockchain Forensics in Digital Investigations. – Medium. – URL: https://medium.com/@sourabhkumardas/unveiling-the-secrets-of-the-chain-blockchain-forensics-in-digital-investigations-c4792b4f1167

5. SSRN. Blockchain Forensics: An Overview of Challenges and Tools. – URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4358561

6. Kohn, Kohn & Colapinto. Reporting Initial Coin Offering Scams & Fraud – FAQ. URL: https://kkc.com/frequently-asked-questions/reporting-initial-coin-offering-scams-fraud/

7. DataVisor. Rug Pull Scams. — URL: https://www.datavisor.com/wiki/rug-pull-scams/

8. Investopedia. Blockchain. – URL: https://www.investopedia.com/terms/b/blockchain.asp

9. Фурно Н. Investigating Cryptocurrencies: Understanding, Extracting, and Analyzing Blockchain Evidence / N. Furneaux. – Wiley, 2018. – 320 с.

10. Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. – URL: https://bitcoin.org/bitcoin.pdf

11. Coinbase. What Is a Wallet Address? — URL: https://www.coinbase.com/ru/learn/wallet/what-is-a-wallet-address

12. Kraken. How Do Cryptocurrency Transactions Work? — URL: https://www.kraken.com/ru/learn/how-do-cryptocurrency-transactions-work

13. Binance Academy. Unspent Transaction Output (UTXO). — URL: https://academy.binance.com/ru/glossary/unspent-transaction-output-utxo

14. Kaleido. UTXO vs Account Model. — URL: https://www.kaleido.io/blockchain-blog/utxo-vs-account-model

15. Preethi Kasireddy. How Does Ethereum Work Anyway? – Medium. – URL: https://preethikasireddy.medium.com/how-does-ethereum-work-anyway-22d1df506369

16. Chainalysis. Crypto Crime Midyear 2023 Update: Ransomware & Scams. – URL: https://www.chainalysis.com/blog/crypto-crime-midyear-2023-update-ransomware-scams/

17. Ethereum Foundation. Ethereum Whitepaper. — URL: https://whitepaper.io/document/5/ethereum-whitepaper

18. OWASP. Smart Contract Top 10 Project. – URL: https://owasp.org/www-project-smart-contract-top-10/

19. ECOS. Top Privacy Coins of 2025: The Best Anonymous Cryptocurrencies and How They Work. – URL: https://ecos.am/en/blog/top-privacy-coins-of-2025-the-best-anonymous-cryptocurrencies-and-how-they-work/

20. CoinMarketCap Academy. What Are Privacy Coins? – URL: https://coinmarketcap.com/academy/article/what-are-privacy-coins

21. Coin Bureau. What Is Zcash? – URL: https://coinbureau.com/education/what-is-zcash/

22. Blockchain Forensics: A Systematic Literature Review of Techniques, Applications, Challenges, and Future Directions – URL: https://wrap.warwick.ac.uk/id/eprint/188402/1/electronics-13-03568-v2.pdf

23. Blockchain Forensics: A Modern Approach to Investigating Cybercrime in the Age of Decentralisation – URL: https://www.researchgate.net/publication/368910848_Blockchain_Forensics_A_Modern_Approach_to_Investigating_Cybercrime_in_the_Age_of_Decentralisation

24. Maltego. – URL: https://www.maltego.com/

25. RocketMeUpCybersecurity. Blockchain Forensics: Tracking Cybercrime on the Blockchain. – Medium. – URL: https://medium.com/@RocketMeUpCybersecurity/blockchain-forensics-tracking-cybercrime-on-the-blockchain-c9443f835a31

26. Bitfury. Clustering Whitepaper. – URL: https://bitfury.com/content/downloads/clustering_whitepaper.pdf

27. Detecting Anomalous Cryptocurrency Transactions: an AML/CFT Application of Machine Learning-based Forensics – URL: https://arxiv.org/abs/2206.04803

28. Scikit-learn. Decision Trees Module. – URL: https://scikit-learn.org/stable/modules/tree.html

29. Scikit-learn. RandomForestClassifier. – URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

30. Fraidoon Omarzai. XGBoost Classification In-depth. – Medium. – URL: https://medium.com/@fraidoonomarzai99/xgboost-classification-in-depth-979f11ef4bf9

31. Robu Rishabh. KNN – K-Nearest Neighbour. – Medium. – URL: https://medium.com/@RobuRishabh/knn-k-nearest-neighbour-5ae18ae8e274

32. Jason Brownlee. Multinomial Logistic Regression with Python. – URL: https://machinelearningmastery.com/multinomial-logistic-regression-with-python/

33. GeeksforGeeks. Multi-class Classification Using Support Vector Machines (SVM). – URL: https://www.geeksforgeeks.org/multi-class-classification-using-support-vector-machines-svm/

34. Scikit-learn. Neural Networks Supervised Module. – URL: https://scikit-learn.org/stable/modules/neural_networks_supervised.html

35. Kaggle. Elliptic Dataset. – URL: https://www.kaggle.com/datasets/ellipticco/elliptic-data-set

36. Kaggle. Ransomware Heist Dataset. – URL: https://www.kaggle.com/datasets/gopalmahadevan/bitcoin-heist-ransomware-address-dataset

37. Kaggle. Labeled Bitcoin Addresses and Transactions. – URL: https://www.kaggle.com/datasets/leonidgarin/labeled-bitcoin-addresses-and-trfansactions

38. WalletExplorer. Blockchain Analysis Tool. – URL: https://www.walletexplorer.com/

39. Arkham Intel.– URL: https://intel.arkm.com/

40. Ransomwhere. – URL: https://ransomwhe.re/#about

41. IBM. What is feature-engineering?. – URL: https://www.ibm.com/think/topics/feature-engineering

42.Identification of High Yielding Investment Programs in Bitcoin via Transactions Pattern Analysis — URL: https://www.researchgate.net/publication/322513048_Identification_of_High_Yielding_Investment_Programs_in_Bitcoin_via_Transactions_Pattern_Analysis

43. Feature Engineering for Anomaly Detection and Classification of Blockchain Transactions — URL: https://www.researchgate.net/publication/369552917_Feature_Engineering_for_Anomaly_Detection_and_Classification_of_Blockchain_Transactions

44. BlockSci: Design and applications of a blockchain analysis platform. – URL: https://arxiv.org/abs/1709.02489.

45. Scikit-learn. Official Documentation. – URL: https://scikit-learn.org/stable/

46. Metrics for Multi-Class Classification: an Overview. – URL: https://arxiv.org/abs/2008.05756

47. DataCamp. Explainable AI: Understanding and Trusting ML Models. – URL: https://www.datacamp.com/tutorial/explainable-ai-understanding-and-trusting-machine-learning-models

ДОДАТОК А

```
import pandas as pd
from urllib.request import urlretrieve as retrieve
import time
#url =
"https://www.walletexplorer.com/api/1/wallet-addresses?wallet=WALLET_ID&
from=FROM&count=100"
import csv
import os
import json
import re
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)

def get_wallet_addresses(WALLET_ID, type, page = 1):
    url_csv =
f"https://www.walletexplorer.com/wallet/{WALLET_ID}/addresses?page={pag
e}&format=csv"
    save = f"./addresses/{type}/{WALLET_ID}.csv"
    retrieve(url_csv, save)

def parse_address_csv(type, WALLET_ID):
    df = pd.read_csv(f'./addresses/{type}/{WALLET_ID}.csv',
    skiprows=2)
    df.columns = ["address", "balance", "incoming_txs", "last_used_in_block"]
    df['label'] = type
    df['name'] = WALLET_ID
    df.to_csv(f'./addresses/{type}/{WALLET_ID}.csv')
    return df.head()

def parse_darknet_adress(input_file, name):
    entries = set()

    with open(input_file, 'r', encoding='utf-8') as f:
        lines = f.readlines()
```

```
    input_file = input_file.rpartition('.')[0]
    output_file = f'./addresses/Darknet_Markets/{input_file}.csv'

    for line in lines[1:]:
        parts = line.strip().split('\t')
        if len(parts) < 6:
            continue
        to_address = parts[4]
        to_label = parts[5]
        if to_label.startswith(name):
            cleaned_label = re.sub(r'\s*\(.*?\)', '', to_label)
            entries.add((to_address, cleaned_label))

    with open(output_file, 'w', newline='', encoding='utf-8') as f:
        writer = csv.writer(f)
        writer.writerow(['address', 'label', 'name'])
        for address, label in sorted(entries):
            writer.writerow([address, 'Darknet_Market', label])
    print("CSV-файл створено:", output_file)

def concat_csv(category_folder):
    cocncat_csv = pd.DataFrame()
    folder_path = f'./addresses/{category_folder}'
    existing_merged = os.path.join(folder_path, f'{category_folder}_merged.csv')
    if os.path.exists(existing_merged):
        os.remove(existing_merged)
    else:
        print(f'Файл {existing_merged} не існує.')
    for file in os.listdir(folder_path):
        if file.endswith(".csv") and not file.endswith("old.csv"):
            file_path = os.path.join(folder_path, file)
            df = pd.read_csv(file_path)
            if 'Unnamed: 0' in df.columns:
                df = df.drop('Unnamed: 0', axis = 1)
            cocncat_csv = pd.concat([cocncat_csv, df], ignore_index=True)
    #cocncat_csv = cocncat_csv.drop(["Unnamed: 0"])
```

```
cocncat_csv.to_csv(f'./addresses/{category_folder}/{category_folder}_merged.c
sv', index = False)
    return cocncat_csv
def extract_ransomware_address(file):
    f = open(file)
    data = json.load(f)
    output_file = './addresses/Ransomware/Ransomware_data.csv'
    with open(output_file,'w', newline='', encoding='utf-8') as f:
        writer = csv.writer(f)
        writer.writerow(['address', 'lable', 'name'])
        type = 'Ransomware'
        for entry in data:
            address = entry.get('address', '')
            family = entry.get('family', '')
            writer.writerow([address, type, family])
    print("CSV файл створено:", output_file)
    return data
if __name__ == "__main__":
    wallets = {
        "Exchanges": ["Huobi.com", "Bittrex.com", "Cex.io", "OKCoin.com",
"BitBay.net", "MaiCoin.com", "Hashnest.com", "BtcTrade.com", "YoBit.net",
"BitZlato.com"],
        "Gambling": ["SatoshiDice.com", "BitZillions.com", "999Dice.com",
"CloudBet.com", "CoinGaming.io", "PocketDice.io", "FortuneJack.com",
"Rollin.io", "BitZino.com", "SatoshiBet.com"],
        "Pools": ["BTCCPool", "GHash.io", "BW.com", "BitMinter.com",
"KnCMiner.com", "Eligius.st", "Telco214", "Bitfury.org", "SlushPool.com",
"AntPool.com"]
    }
    categories = ["Exchanges", "Gambling", "Pools", "Darknet_Markets"]
    darknet_markets = ["Hydra Market", "Nucleus Marketplace", "Kraken
Darknet", "Buybest Market", "Agora Market", "Alphabay Market", "Evolution
Darknet", "Silk Road"]
    darknet_filenames = ["Hydra_Market.txt", "Nucleus_Market.txt",
"Kraken_Market.txt", "Buybest_Market.txt", "Agora_Market.txt",
"Alphabay_Market.txt", "Evolution_Market.txt", "Silk_Road_Market.txt"]
```

```
ransomware = "./addresses/ransomware_data.json"


"""
for wallet_type, wallet_list in wallets.items():
    for WALLET_ID in wallet_list:
        try:
            if os.path.exists(f"./addresses/{wallet_type}/{WALLET_ID}.csv"):
                continue
            print(f"Processing {WALLET_ID} from type {wallet_type}")
            get_wallet_addresses(WALLET_ID, wallet_type)
            time.sleep(3)
            df = parse_address_csv(wallet_type, WALLET_ID)
             print(f" {WALLET_ID}: {len(df)} addresses loaded.")
        except Exception as e:
             print(f"Failed to process {WALLET_ID}: {e}")
"""
for category in categories:
    concat_csv(category)
"""
ransom = extract_ransomware_address(ransomware)
print(ransom[0])
"""
for i in range(len(darknet_markets)):
    parse_darknet_adress(darknet_filenames[i], darknet_markets[i])
```

ДОДАТОК Б

```
import requests
import time
import pandas as pd
import json
import os
def make_safe_request(url):
    while True:
        try:
            start_time = time.time()
            response = requests.get(url, timeout=30)
            duration = time.time() - start_time
            print(f" GET {url} took {duration:.2f} seconds")
            if response.status_code == 429:
                reset_time = int(response.headers.get("Ratelimit-Reset", "200"))
                print(f"Rate limit exceeded. Waiting for {reset_time} seconds")
                time.sleep(reset_time)
                continue
            elif response.status_code == 200:
                return response
            else:
                print(f"Error {response.status_code} when requesting {url}")
                return None
        except Exception as e:
            print(f"Exception during request to {url}: {e}")
```

```
            return None

def get_address_info(input_file, label,  delay=1.5):
    df = pd.read_csv(input_file)
    base_url = "https://api.bitaps.com/btc/v1/blockchain/address"
    output_list = []
    output_filename = f"./addresses/{label}/{label}_data.json"
    processed_address = []
    existing_address = set()
    output_dir = f"./addresses/{label}"
    #os.makedirs(output_dir, exist_ok=True)
    if os.path.exists(output_filename):
        with open(output_filename, 'r') as f:
            processed_address = json.load(f)
            existing_address = {entry["address"] for entry in processed_address}
    print(f"[*] Already processed {len(existing_address)} addresses.")
    total = len(df)
    print(f"[*] Processing {total} addresses in category '{label}'")
    for idx, row in df.iterrows():
        address = row["address"]
        if address in existing_address:
            continue
        address_label = row["label"]
        name = row['name']
        print(f"[{int(idx)+ 1}/{total}] Processing address: {address} ({name})")
        state_url = f"{base_url}/state/{address}"
        tx_url = f"{base_url}/transactions/{address}"
        try:
```

```
        state_response = make_safe_request(state_url)
        if state_response.status_code != 200:
            print(f"[{address}] Error getting state: {state_response.status_code}")
            return None
        state_data = state_response.json().get("data", {})
        time.sleep(delay)
        tx_response = make_safe_request(tx_url)
        if tx_response.status_code != 200:
                                print(f"[{address}] Error getting transactions:
{tx_response.status_code}")
            return None
        tx_data = tx_response.json().get("data", {})
        time.sleep(delay)
        output_dict =  {
            "address": address,
            "label": address_label,
            "name": name,
            "balance": state_data.get("balance"),
            "total_received": state_data.get("receivedAmount"),
            "received_tx": state_data.get("receivedTxCount"),
            "total_sent": state_data.get("sentAmount"),
            "sent_tx": state_data.get("sentTxCount"),
                                                    "largest_received_Tx_amount":
state_data.get("largestReceivedTxAmount"),
            "largest_spent_Tx_amount": state_data.get("largestSpentTxAmount"),
            "received_Outs_Count": state_data.get("receivedOutsCount"),
            "spent_Outs_Count": state_data.get("spentOutsCount"),
            "type": state_data.get("type"),
```

```
            "transactions": [tx_data]
        }
        processed_address.append(output_dict)
    except Exception as e:
        print(f"Error processing{address}: {e}")
  with open(output_filename, "w") as f:
    json.dump(processed_address, f, indent=2)
        print(f"Successfully updated\written transaction data for {label} in {output_filename}")
  return processed_address
#categories = ["Exchanges","Gambling", "Pools", "Darknet_Markets", "Ransomware"]
#categories = ["Gambling", "Pools", "Darknet_Markets"]
categories = ["Ransomware"]
def main():
  for cat in categories:
    #input_file = f"./addresses/{cat}/{cat}_merged.csv"
    input_file = f"./addresses/Ransomware/Ransomware_data_1000.csv"
    print(f"Extracting info for {cat} category")
    if not os.path.exists(input_file):
      print(f"cant find file: {input_file}")
      break
    get_address_info(input_file, cat)

if __name__ == "__main__":
  main()
```

ДОДАТОК В

```
import pandas as pd
from urllib.request import urlretrieve as retrieve
import time
import csv
import os
import json
import re
import numpy as np
SATOSHIS_PER_BTC = 1e8

def calc_stats(data):
    if not data:
        return (0, 0, 0, 0, 0)
    return (
        sum(data),
        min(data),
        max(data),
        sum(data) / len(data),
        np.std(data)
    )

def extract_features(add_data):
    feature_list = []
    received_tx_amounts = []
    sent_tx_amounts = []
```

```
received_tx_fees = []
sent_tx_fees = []
received_tx_sizes = []
sent_tx_sizes = []
inputs_counts = []
outs_counts = []
rbf_count = 0
txs = add_data.get("transactions", [])
final_tx = txs[0].get("list")
#print(txs)
#print(txs[0])
for tx in final_tx:
    amount = tx.get("amount", 0) / 1e8
    fee = tx.get("fee", 0) / 1e8
    size = tx.get("size", 0)
    input_n = tx.get("inputsCount", 0)
    output_n = tx.get("outsCount", 0)
    if amount > 0:
        received_tx_amounts.append(amount)
        received_tx_fees.append(fee)
        received_tx_sizes.append(size)
    elif amount < 0:
        sent_tx_amounts.append(-amount)
        sent_tx_fees.append(fee)
        sent_tx_sizes.append(size)
    inputs_counts.append(input_n)
    outs_counts.append(output_n)
    if tx.get("rbf"):
```

```
            rbf_count += 1
    rbf_ratio = rbf_count / len(final_tx) if len(final_tx) > 0 else 0
                        _, min_recv, max_recv, avg_recv, std_recv =
calc_stats(received_tx_amounts)
    _, min_sent, max_sent, avg_sent, std_sent = calc_stats(sent_tx_amounts)
     total_fee_recv, min_fee_recv, max_fee_recv, avg_fee_recv, std_fee_recv =
calc_stats(received_tx_fees)
      total_fee_sent, min_fee_sent, max_fee_sent, avg_fee_sent, std_fee_sent =
calc_stats(sent_tx_fees)
     total_tx_size_recv, min_tx_size_recv, max_tx_size_recv, avg_tx_size_recv,
std_tx_size_recv = calc_stats(received_tx_sizes)
      total_tx_size_sent, min_tx_size_sent, max_tx_size_sent, avg_tx_size_sent,
std_tx_size_sent = calc_stats(sent_tx_sizes)
       total_inputs, min_inputs_per_tx, max_inputs_per_tx, avg_inputs_per_tx,
std_inputs_per_tx = calc_stats(inputs_counts)
                    total_outputs, min_outputs_per_tx, max_outputs_per_tx,
avg_outputs_per_tx, std_outputs_per_tx = calc_stats(outs_counts)
    features = {
    "address": add_data.get("address"),
    "name": add_data.get("name"),
    "total_received": add_data.get("total_received", 0) / 1e8,
    "max_recv": add_data.get("largest_received_Tx_amount", 0) / 1e8,
    "min_recv": min_recv,
    "avg_recv": avg_recv,
    "std_recv": std_recv,
    "total_sent" : add_data.get("total_sent", 0) / 1e8,
    "max_sent": add_data.get("largest_spent_Tx_amount") / 1e8,
    "min_sent": min_sent,
```

"avg_sent": avg_sent,

"std_sent": std_sent,

"total_fee_recv": total_fee_recv,

"max_fee_recv": max_fee_recv,

"min_fee_recv": min_fee_recv,

"avg_fee_recv": avg_fee_recv,

"std_fee_recv": std_fee_recv,

"total_fee_sent": total_fee_sent,

"max_fee_sent": max_fee_sent,

"min_fee_sent": min_fee_sent,

"avg_fee_sent": avg_fee_sent,

"std_fee_sent ": std_fee_sent,

"total_tx_size_recv": total_tx_size_recv,

"max_tx_size_recv": max_tx_size_recv,

"min_tx_size_recv": min_tx_size_recv,

"avg_tx_size_recv": avg_tx_size_recv,

"std_tx_size_recv": std_tx_size_recv,

"total_tx_size_sent": total_tx_size_sent,

"max_tx_size_sent": max_tx_size_sent,

"min_tx_size_sent": min_tx_size_sent,

"avg_tx_size_sent": avg_tx_size_sent,

"std_tx_size_sent": std_tx_size_sent,

"total_inputs": total_inputs,

"max_inputs_per_tx": max_inputs_per_tx,

"min_inputs_per_tx": min_inputs_per_tx,

"avg_inputs_per_tx": avg_inputs_per_tx,

"std_inputs_per_tx": std_inputs_per_tx,

"total_outputs": total_outputs,

```
    "max_outputs_per_tx": max_outputs_per_tx,
    "min_outputs_per_tx": min_outputs_per_tx,
    "avg_outputs_per_tx": avg_outputs_per_tx,
    "std_outputs_per_tx": std_outputs_per_tx,
    "rbf_ratio": rbf_ratio,
    "balance" : add_data.get("balance", 0) / 1e8,
    "n_tx" : add_data.get("received_tx", 0) + add_data.get("sent_tx", 0),
    "n_tx_received" : add_data.get("received_tx", 0),
    "n_tx_sent" : add_data.get("sent_tx", 0),
    "label": add_data.get("label"),
    }
    column_names = list(features.keys())
    return features, column_names
categories = ["Exchanges","Gambling", "Pools", "Darknet_Markets", "Ransomware"]
#categories = ["Ransomware"]
all_feature_list = []
col_names = None
def main():
    global col_names
                                                    output_file = "./addresses/all_addresses_features_new_more_features2_same.csv"
    #output_file = "./addresses/ransomware_features.csv"
    for cat in categories:
        input_file = f"./addresses/{cat}/{cat}_data.json"
        print(f"Extracting features for {cat} category")
        if not os.path.exists(input_file):
          print(f"cant find file: {input_file}")
```

```
            break
        with open(input_file, 'r', encoding='utf-8') as f:
            add_data = json.load(f)
        for add in add_data:
            address_features, column_names = extract_features(add)
            if address_features:
                all_feature_list.append(address_features)
                if col_names is None:
                    col_names = column_names
    with open(output_file, mode='w', newline='') as file:
        writer = csv.DictWriter(file, fieldnames=col_names)
        writer.writeheader()  # Write header row
        writer.writerows(all_feature_list)  # Write data rows

if __name__ == "__main__":
    main()
```

ДОДАТОК Г

```
import pandas as pd
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split, RandomizedSearchCV
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
import shap
import numpy as np
from lime.lime_tabular import LimeTabularExplainer
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)

def data_preprocessing(df):
```

```
    print("-----------data information----------------")
    print("\n")
    print("+++++++++Columns data+++++++++")
    print(df.info())
    print("\n")
    print("+++++++++Checking for null values+++++++++")
    print(df.isnull().sum())
    print("\n")
    print("\n")
    print("----------------Statistical Analysis----------------")
    print(df.describe())
    print("\n")
    print("----------------Duplicates count----------------")
    duplicates = df.duplicated()
    print("Кількість Дублікатів в датасеті:", duplicates.sum())
    print("\n")
    print(duplicates)


def data_visualization(df):

    X = df.iloc[:, 2:-1]
    plt.figure(dpi=130)
    sns.heatmap(X.corr(), annot=True, fmt='.2f')
    plt.title("Feature correlation")

    num_features = X.shape[1]
    n_cols = 4
```

```
    n_rows = (num_features + n_cols - 1) // n_cols
    plt.figure(figsize=(5 * n_cols, 4 * n_rows))
    for i, column in enumerate(X.columns):
        plt.subplot(n_rows, n_cols, i + 1)
        sns.kdeplot(X[column], fill=True)
        plt.title(column)
        plt.tight_layout()
    plt.show()


def grid_search(X, y, model_name="Logistic Regression", n_iter=20):
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.3, random_state=42
    )

    models = {
        "Logistic Regression": {
            "model": LogisticRegression(max_iter=10000),
            "params": {
                'model__C': [0.01, 0.1, 1, 10],
                'model__penalty': ['l2', 'l1'],
                'model__solver': ['lbfgs', 'saga'],
                'model__multi_class': ['multinomial']
            }
        },
        "KNN": {
            "model": KNeighborsClassifier(),
            "params": {
```

```
        'model__n_neighbors': [3, 5, 7, 10],
        'model__weights': ['uniform', 'distance'],
        'model__metric': ['euclidean', 'manhattan']
    }
},
"Decision Tree": {
    "model": DecisionTreeClassifier(random_state=13),
    "params": {
        'model__max_depth': [7, 10, None],
        'model__min_samples_split': [2, 5, 10],
        'model__criterion': ['gini', 'entropy']
    }
},
"SVM": {
    "model": SVC(random_state=13),
    "params": {
        'model__C': [0.1, 1, 10],
        'model__kernel': ['poly', 'rbf', 'sigmoid'],
        'model__gamma': ['scale', 'auto'],
        'model__decision_function_shape': ['ovo', 'ovr']
    }
},
"MLP": {
    "model": MLPClassifier(max_iter=2000, random_state=13),
    "params": {
        'model__hidden_layer_sizes': [(50,), (100, 50)],
        'model__activation': ['relu'],
        'model__solver': ['adam'],
```

```
            'model__alpha': [0.0001, 0.05],
            'model__learning_rate': ['adaptive']
        }
    },
    "XGBoost": {
        "model": XGBClassifier(eval_metric='mlogloss', random_state=13),
        "params": {
            'model__n_estimators': [ 100, 150, 200],
            'model__max_depth': [3, 5],
            'model__learning_rate': [0.05, 0.1],
            'model__subsample': [0.8, 1.0],
            'model__gamma': [0, 1],
        }
    },
    "RandomForest": {
        "model": RandomForestClassifier(random_state=13),
        "params": {
            'model__n_estimators': [100, 200],
            'model__max_depth': [None, 10, 20],
            'model__min_samples_split': [2, 5],
            'model__min_samples_leaf': [1, 2],
            'model__max_features': ['sqrt', 'log2']

        }
    }
}

if model_name not in models:
```

```
            raise ValueError(f"Model '{model_name}' not found. Available: {list(models.keys())}")



    config = models[model_name]

    pipe = Pipeline([
        ('scaler', StandardScaler()),
        ('model', config['model'])
    ])

        grid = GridSearchCV(pipe, config['params'], cv=3, scoring='accuracy', n_jobs=-1)
    grid.fit(X_train, y_train)

    best_model = grid.best_estimator_
    y_pred = best_model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print("Tested model:", model_name)
    print("Best parameters:", grid.best_params_)
    print(f"Accuracy on test: {acc:.4f}")

    return best_model

def classification(X, y):
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=24)
```

```
sc = StandardScaler()
X_train_std = sc.fit_transform(X_train)
X_test_std = sc.transform(X_test)

tree = DecisionTreeClassifier(random_state=24, min_samples_split=5, criterion='entropy')
clf = RandomForestClassifier(max_depth=None, random_state=0, max_features='sqrt', min_samples_leaf=1, min_samples_split=2, n_estimators=200)
knn = KNeighborsClassifier(n_neighbors=3, weights='distance', metric='manhattan')
svm = SVC(C=10, kernel='rbf', gamma='scale', decision_function_shape='ovo')
xgBoost = XGBClassifier(gamma =0, objective='multi:softmax', num_class=5, learning_rate=0.1, n_estimators=200, subsample=0.8, max_depth=5)
logisticReg = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=10000, penalty='l2', C=10)
mlp = MLPClassifier(hidden_layer_sizes=(100, 50), alpha=0.0001, learning_rate='adaptive', random_state=1, solver='adam')

tree.fit(X_train_std, y_train)
clf.fit(X_train_std, y_train)
knn.fit(X_train_std, y_train)
svm.fit(X_train_std, y_train)
xgBoost.fit(X_train_std, y_train)
logisticReg.fit(X_train_std, y_train)
```

```
mlp.fit(X_train_std, y_train)

y_pred_tree = tree.predict(X_test_std)
y_pred_clf = clf.predict(X_test_std)
y_pred_knn = knn.predict(X_test_std)
y_pred_svm_ovo = svm.predict(X_test_std)
y_pred_xgboost = xgBoost.predict(X_test_std)
y_pred_LR = logisticReg.predict(X_test_std)
y_pred_mlp =mlp.predict(X_test_std)

  print("репорт класифікації decision_tree:\n", classification_report(y_test, y_pred_tree))
 print("репорт класифікації рандом форест:\n", classification_report(y_test, y_pred_clf))
      print("репорт класифікації knn:\n", classification_report(y_test, y_pred_knn))
    print("репорт класифікації svm ovo:\n", classification_report(y_test, y_pred_svm_ovo))
     print("репорт класифікації xgBoost:\n", classification_report(y_test, y_pred_xgboost))
              print("репорт класифікації Logistic regression:\n", classification_report(y_test, y_pred_LR))
       print("репорт класифікації MLP:\n", classification_report(y_test, y_pred_mlp))

reports = {
              "Decision Tree": classification_report(y_test, y_pred_tree, output_dict=True),
```

```
                "Random Forest": classification_report(y_test, y_pred_clf, output_dict=True),
    "KNN": classification_report(y_test, y_pred_knn, output_dict=True),
    "SVM": classification_report(y_test, y_pred_svm_ovo, output_dict=True),
                "XGBoost": classification_report(y_test, y_pred_xgboost, output_dict=True),
                "Logistic Regression": classification_report(y_test, y_pred_LR, output_dict=True),
    "MLP": classification_report(y_test, y_pred_mlp, output_dict=True),
  }

  summary_df = pd.DataFrame({
    model: {
      "accuracy": round(rep["accuracy"], 3),
      "macro avg precision": round(rep["macro avg"]["precision"], 3),
      "macro avg recall": round(rep["macro avg"]["recall"], 3),
      "macro avg f1-score": round(rep["macro avg"]["f1-score"], 3),
    }
    for model, rep in reports.items()
  }).T
  each_class_summary = pd.DataFrame({
    model: {
      "Exchanges": round(rep["4"], 3),
      "Gambling": round(rep["macro avg"]["precision"], 3),
      "Pools": round(rep["macro avg"]["recall"], 3),
      "Darknet_Market": round(rep["macro avg"]["f1-score"], 3),
      "Ransomware": round(rep["macro avg"]["f1-score"], 3),
    }
```

```
    for model, rep in reports.items()
}).T
#summary_df.to_csv('./addresses/metrics_report_newparams.csv')
#print(summary_df)


plt.show()


def explain_model_XGBoost(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=24)

    sc = StandardScaler()
    #X_train_std = sc.fit_transform(X_train)
    #X_test_std = sc.transform(X_test)
        xgBoost = XGBClassifier(objective='multi:softprob', num_class=5, learning_rate=0.1, n_estimators=200, subsample=0.8, max_depth=5)
    xgBoost.fit(X_train, y_train)
    y_pred_xgboost = xgBoost.predict(X_test)

    importances = xgBoost.feature_importances_

    feature_importance_df = pd.DataFrame({
        'feature': X_train.columns,
        'importance': importances
    }).sort_values(by='importance', ascending=False)
```

```
    labels = ['Ransomware', 'Darknet_Market', 'Pools', 'Gambling', 'Exchanges']

    cm_xgboost = confusion_matrix(y_test, y_pred_xgboost)
    print(cm_xgboost)
            sns.heatmap(cm_xgboost, annot=True, fmt='d', cmap='YlGnBu',
xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted', fontsize=12)
    plt.ylabel('Actual', fontsize=12)
    plt.title('Confusion Matrix', fontsize=16)
    plt.xticks(rotation=45)
    plt.yticks(rotation=0)
    plt.show()

    print(feature_importance_df)
    plt.figure(figsize=(10, 6))
                                        plt.barh(feature_importance_df['feature'],
feature_importance_df['importance'])
    plt.xlabel("Importance")
    plt.title("Feature Importance")
    plt.gca().invert_yaxis()
    plt.tight_layout()
    plt.show()

    print("\nCalculating SHAP values...")
    explainer = shap.TreeExplainer(xgBoost)
    shap_values = explainer.shap_values(X_test)
```

```
print("X_test shape:", X_test.shape)
print("shap_values shape:", shap_values.shape)

e = 0
max_plots = 5
    labels_list =["Ransomware", "Darknet_Market", "Pools", "Gambling",
"Exchanges"]

if shap_values.shape[2] == len(np.unique(y)):
    shap_values = shap_values.transpose((2, 0, 1))
    for i, label in enumerate(labels_list):
        shap.summary_plot(shap_values[i], X_test, show=False)
        plt.title(f"Вплив ознак на передбачення для класу {label}")
        plt.tight_layout()
        plt.show()
else:
    raise ValueError("Unexpected shape for SHAP values")


instance_idx = 400
class_idx = 4

shap.plots.waterfall(
    shap.Explanation(
        values=shap_values[class_idx][instance_idx],
        base_values=explainer.expected_value[class_idx],
        data=X_test.iloc[instance_idx],
        feature_names=X_test.columns
```

```
        )
    )
    print("\nCalculating LIME explanation...")




def main():
    # filename = './addresses/all_addresses_features.csv'
    filename = './addresses/all_addresses_features_new_more_features2.csv'
    #filename = './selected_features_dataset.csv'
    filenam_normalized = ""

    df = pd.read_csv(filename)
    #data_preprocessing(df)
        df = df.replace({'label': {'Exchanges': 4, "Gambling": 3, "Pools": 2,
"Darknet_Market": 1, "Ransomware": 0}})
    X = df.iloc[:, 2:-1]
    y = df['label']
    explain_model_XGBoost(X, y)
    #grid_search(X, y, model_name = 'MLP')
    #classification(X, y)
    #grid_search(X, y)

main()
```