**Федеральное государственное образовательное бюджетное учреждение высшего образования**

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ**

**ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**(Финансовый университет)**

**Факультет информационных технологий и анализа больших данных**

**Департамент анализа данных и машинного обучения**

**Пояснительная записка к курсовой работе**

по дисциплине «Технологии анализа данных и машинного обучения»

на тему:

«Классификация фишинговых криптовалютных аккаунтов»

Выполнил:

студент группы ПИ20-3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Выродов А. Г.

Научный руководитель:

к.ф.-м.н., доцент Попов В. Г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

МОСКВА – 2023

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc134795468)

[ОСНОВНАЯ ЧАСТЬ 4](#_Toc134795469)

[Введение в предметную область 4](#_Toc134795470)

[**Краткие сведения о технологии блокчейн, криптовалютах и платформе Ethereum** 4](#_Toc134795471)

[**Угроза фишинговых аккаунтов для сети** 4](#_Toc134795472)

[**Набор данных и формулировка задачи** 6](#_Toc134795473)

[Описательный анализ данных 6](#_Toc134795474)

[**Целевая переменная.** 8](#_Toc134795475)

[**Подграфы и избавление от чрезмерной несбалансированности** 13](#_Toc134795476)

[Архитектуры нейронных сетей для работы с графами 15](#_Toc134795477)

[**Выбор библиотеки для работы с GNN** 18](#_Toc134795478)

[Графовая свёрточная нейронная сеть 18](#_Toc134795479)

[**Создание** 18](#_Toc134795480)

[**Обучение** 20](#_Toc134795481)

[Графовая сеть внимания 23](#_Toc134795482)

[**Создание** 23](#_Toc134795483)

[**Обучение** 25](#_Toc134795484)

[ЗАКЛЮЧЕНИЕ 28](#_Toc134795485)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ИНТЕРНЕТ-РЕСУРСОВ 29](#_Toc134795486)

# **ВВЕДЕНИЕ**

В настоящее время технология блокчейн и её наиболее известное применение - криптовалюты – обретают большую значимость в нашей жизни, и не только в качестве средства инвестирования. Эти технологии позволяют безопасно и прозрачно проводить финансовые транзакции, хранить и передавать данные, а также устанавливать договорённости между пользователями без привлечения третьих лиц. Однако, «одноранговость» блокчейн проектов никак не защищает их от мошеннических действий. Наличие фишинговых аккаунтов на платформах, использующих криптовалюты, может представлять угрозу безопасности для пользователей, которые могут стать жертвами цифровых преступников, действующих в децентрализованном мире.

Проблема классификации фишинговых криптовалютных аккаунтов может быть решена с помощью применения нейронных сетей. Так, целью данной работы является создание моделей, способной из множества аккаунтов сети выделять мошеннические. Созданные модели будут обучена на графе транзакций сети Ethereum. По итогам обучения архитектуры разной сложности будут сравнены по своей производительности и способности корректно распознавать аккаунты различных типов.

Данная курсовая работа содержит описательный анализ графа транзакций платформы Ethereum на 2019 год, большое внимание, ввиду особенности формата данных уделяет формированию датасета, пригодного к обучению модели, а также архитектуре графовых нейронных сетей, подходящих для решения данного типа задач.

Исходный код проекта курсовой работы доступен на GitHub: <https://github.com/dizpatcher/eth_phishing_nodes_classification>

# **ОСНОВНАЯ ЧАСТЬ**

## **Введение в предметную область**

### **Краткие сведения о технологии блокчейн, криптовалютах и платформе Ethereum**

Блокчейн - технология, позволяющая хранить и передавать информацию без необходимости централизованного контроля. Она основана на создании цепочки блоков, которые содержат информацию о транзакциях или событиях, происходящих в системе. Каждый блок содержит уникальный код - хэш, который связывает его с предыдущим блоком и обеспечивает целостность системы.

Криптовалюта — это наиболее известное применение блокчейна, цифровая валюта, которая использует криптографические методы для обеспечения безопасности и анонимности транзакций. Она является одним из применений технологии блокчейн, которая позволяет создавать децентрализованные и автономные системы для обмена цифровыми активами.

Одна из самых распространённых реализаций блокчейна, обладающих собственной криптовалютой - проект Ethereum — это платформа для создания децентрализованных приложений. Она позволяет создавать смарт-контракты - автономные программы, выполняющие действия, определенные участниками сети. Кроме того, Ethereum использует криптовалюту под названием Ether (ETH) – «Эфир» - для оплаты транзакционных сборов и вознаграждения валидационных узлов, которые обрабатывают транзакции и поддерживают работу сети.

### **Угроза фишинговых аккаунтов для сети**

В Ethereum аккаунты могут быть двух видов - внешние аккаунты (аккаунты пользователей) и аккаунты смарт-контрактов. Каждый аккаунт в Ethereum имеет адрес, который представляет собой уникальную комбинацию 40 шестнадцатеричных символов. Этот адрес используется для отправки и получения токенов между аккаунтами. В рамках данной работы понятие аккаунт, адрес и кошелёк - взаимозаменяемы. В рассматриваемом датасете именно первый тип (внешние аккаунты) является точкой данных.

Фишинговые аккаунты в блокчейн сетях, таких как Ethereum, представляют серьезную угрозу для пользователей. Эти аккаунты могут использоваться злоумышленниками для создания поддельных транзакций и обмана пользователей, что может привести к потере доступа к криптовалютным счетам и утечке финансовых средств. Фишинговые адреса также могут использоваться для проведения атак на смарт-контракты, которые работают на блокчейне, с целью изменения логики контракта и внесения изменений в блокчейн, вследствие чего платформа может потерпеть огромные экономические потери.

Фишинговые адреса могут действовать путем создания поддельных транзакций, включая переводы средств, и использования спам-рассылок для привлечения новых пользователей. Они могут также использоваться для проведения атак на смарт-контракты, которые работают на блокчейне, с целью изменения логики контракта и внесения изменений в блокчейн. Фишинговые адреса могут также использоваться для создания мошеннических ICO (Initial Coin Offering), которые обманывают инвесторов и могут привести к значительным финансовым потерям.

Поэтому важно уметь определять фишинговые адреса в блокчейн сетях. Для этого могут использоваться различные методы, включая анализ транзакций и паттернов поведения адресов, идентификацию схожих адресов, а также анализ контента и контекста сообщений, отправляемых фишинговыми адресами.

Таким образом, классификация фишинговых криптовалютных адресов является важной задачей, которая может помочь защитить пользователей блокчейн сетей от мошеннических действий и потерь финансовых средств. Использование нейронных сетей для классификации фишинговых адресов может быть особенно эффективным, так как это позволит автоматизировать процесс определения подозрительных адресов и повысить точность классификации.

### **Набор данных и формулировка задачи**

В [датасете](https://www.kaggle.com/datasets/xblock/ethereum-phishing-transaction-network), на примере которого решается задача классификации, собраны фишинговые узлы из Ethereum, которые были промаркированы на обозревателе сети Etherscan. Начиная с этих фишинговых узлов, создатели датасета «просканировали» огромную транзакционную сеть Ethereum, используя поиск в ширину. И хотя данные относятся к 2019 году, а сама платформа, особенно за последний год, претерпела колоссальные изменения, фишинг до сих пор распространён в сети [3].

Этот набор данных сохранён в формате «pickle» и является объектом библиотеки NetworkX - графом. В приведённом наборе данных аккаунты и транзакции в Ethereum рассматриваются как узлы и ребра, таким образом, обнаружение фишинговых учетных записей может быть смоделировано как проблема бинарной классификации узлов. Каждый узел представляет собой адрес с атрибутом «isp», указывающим, является ли он фишинговым узлом. Каждое ребро имеет два атрибута - сумму транзакции и время её исполнения. По описанию на Kaggle набор данных содержит 2973489 узлов, 13551303 ребра и 1165 узлов, помеченных как фишинговые.

## **Описательный анализ данных**

Одним из наиболее популярных и широко используемых инструментов для анализа графовых структур является библиотека NetworkX. Она предоставляет множество функций и методов для создания, манипулирования, визуализации и анализа графов. Благодаря ей о графе можно извлечь такие свойства как связность и центральность узлов, извлечь информацию об атрибутах узлов и рёбер, провести кластеризацию, выполнить поиск кратчайших путей и т. д. NetworkX также предоставляет возможность импорта и экспорта графов в различных форматах, что позволяет использовать эту библиотеку в сочетании с другими инструментами и программами (например, в данной работе – с Gephi) [4].

Так, в результате первой итерации анализа графа транзакций Ethereum были получены наиболее общие сведения о нём.

|  |  |
| --- | --- |
| Тип графа | Направленный |
| Количество вершин (аккаунты) | 2973489 |
| Количество рёбер (транзакции) | 13551303 |
| Средняя степень вершины (транзакции на аккаунт) | 4.56 |
| Атрибуты вершин | Да, «isp» (булевый) – является ли аккаунт фишинговым |
| Атрибуты рёбер | Да,   1. «Amount» (десятичный) – сумма транзакции в Эфирах 2. «Timestamp» (целочисленный) – время транзакции в секундах с 1 января 1970 |

Таблица 1. Общие сведение о графе транзакций Ethereum

В случае ненаправленного графа невозможно было бы обнаруживать перемещение криптовалюты с адреса на адрес и выявлять мошенничество.

Направление транзакций может указывает на то, кто является источником средств, и кто является их получателем. Очевидно, что это основополагающий фактор при анализе и обнаружении аномалий (фишинга) в переводах Эфира.

Количество узлов и рёбер говорит о том, что граф очень большой – его загрузка в среднем занимает 30 секунд - и обучение на всём датасете может быть очень долгим или невозможным в зависимости от имеющейся вычислительной мощности. Ниже приведены шаги, которые могут уменьшить размера датасета, при этом не удаляя из него значимых для обучения деталей:

- Стоит проверить наличие изолированных вершин (вершин, которые не связаны с другими вершинами ребром). Если такие вершины есть, то их можно удалить из графа, так как они не будут участвовать в обучении модели.

- Стоит проанализировать распределение степеней вершин и определить наиболее важные вершины с точки зрения обучения модели.

- Проверить наличие циклов в графе и определить их роль в обучении модели.

- Проверить наличие мультирёбер (несколько ребер между одними и теми же вершинами) и петель (ребро, соединяющее вершину саму с собой) и определить их роль в обучении модели.

- Проверить наличие подграфов (сообществ) в графе и определить их роль в обучении модели.

Приведённые действия будут рассмотрены далее, после исследования целевой переменной.

### **Целевая переменная.**

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 1. Вычисление количества фишинговых аккаунтов

Из почти 3 млн аккаунтов, только 1165 являются фишинговыми, что составляет лишь 0.039%. Таким образом, датасет является чрезвычайно несбалансированным. Необходимо, принимая шаги, описанные выше, или увеличить число фишинговых аккаунтов, или сократить количество «обычных» адресов.

Для того, чтобы определить шаги, необходимые для достижения баланса в классах выборки, проанализируем, какие характеристики узлов могут быть полезными признаками при выделении одного из типов.

Одним из ключевых аспектов при анализе графа является рассмотрение степеней узлов. Степень узла определяется как количество связей, которые имеет данный узел – в контексте решаемой задачи – исходящие или входящие транзакции аккаунта. Изучение степеней узлов позволяет лучше понимать характеристики графа, его структуру и поведение.

Рассмотрим распределение транзакций по аккаунтам. Красная линия в районе нуля – медианное значение, говорящее о том, что большинство пользователей минимально взаимодействовали с блокчейном.

Изображение выглядит как чек, текст, линия, снимок экрана

Автоматически созданное описание

Рисунок 2. Распределение количества транзакций по аккаунтам

При этом присутствуют аккаунты, которые можно назвать локальными центрами в их поле действия – их степень достигает 102000. Такие узлы могут быть важными точками данных для классификации их соседей.

Стоит отметить, что ни один из топ-10 аккаунтов по проведённым транзакциям не является фишинговым:

|  |  |
| --- | --- |
| Адрес | Проведённые транзакции |
| 0xfbb1b73c4f0bda4f67dca266ce6ef42f520fbb98 | 102480 |
| 0xea674fdde714fd979de3edf0f56aa9716b898ec8 | 67183 |
| 0x2b5634c42055806a59e9107ed44d43c426e58258 | 67031 |
| 0x3f5ce5fbfe3e9af3971dd833d26ba9b5c936f0be | 56124 |
| 0x5e575279bf9f4acf0a130c186861454247394c06 | 48570 |
| 0x52bc44d5378309ee2abf1539bf71de1b7d7be3b5 | 46971 |
| 0x2984581ece53a4390d1f568673cf693139c97049 | 46602 |
| 0x4b01721f0244e7c5b5f63c20942850e447f5a5ee | 38732 |
| 0x59a5208b32e627891c389ebafc644145224006e8 | 32052 |
| 0xe03c23519e18d64f144d2800e30e81b0065c48b5 | 24620 |

Таблица 2. 10 адресов с наибольшим количеством транзакций

При этом, при поиске изолированных вершин (неактивных аккаунтов) не будет найдено ни одного примера, в то время как поиск аккаунтов без входящих или без исходящих вершин вернёт большое число результатов:

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рисунок 3. Поиск аккаунтов без исходящих и без входящих транзакций

В коде выше осуществляется итерирование по преемникам («successors») это все вершины, в которые есть направленное ребро, исходящее из данной вершины, и по предшественникам («predecessors») — это все вершины, из которых есть направленное ребро, входящее в данную вершину.

Оценив наличие в каждой из категорий аккаунтов без входящих или без исходящих фишинговых аккаунтов, можно сделать о том, что активность аккаунта в сети не является характерным признаком для причисления адреса к числу мошеннических.

Изображение выглядит как снимок экрана, диаграмма, апельсин, круг

Автоматически созданное описание

Рисунок 4. Доля фишинговых аккаунтов в числе не примимавших и не отправлявших Эфир

Даже фишинговые аккаунты могут эксплуатироваться, проводя транзакции только "в одну сторону". Поэтому гипотеза о том, что мы можем уменьшить размер датасета за счет изолированных вершин, или что направленность транзакций может являться хорошим признаком для идентификации фишинговых узлов не подтвердилась.

Однако, рассмотрев активность фишинговых аккаунтов, то есть степени вершин с «isp» = 1, можно заметить, что она значительно превышает среднюю по сети: 25.9 против 4.6. Хотя это и не удивительно, ведь акт мошенничества невозможно совершить без определённом подготовительной работы в сети, этот признак может являться достаточно весомым для обучения модели.

Изображение выглядит как текст, снимок экрана, Прямоугольник, линия

Автоматически созданное описание

Рисунок 5. Сравнение среднего количества транзакций между фишинговыми и "обычными" адресами

Дальнейшее «ручное» извлечение признаков и стремление найти характеристики, способные отличать одни аккаунты от других касается непосредственно взаимосвязей между узлами, а именно определения наличия циклов, мультирёбер и петель, сообществ (подграфов) [5].

Предоставленные библиотекой NetworkX функции для анализа графа позволили за минимальное для такого большого графа время определить:

1. В графе присутствуют циклы. Циркуляция Эфира, ни в одной из которых не участвовали фишинговые аккаунты.
2. 9483148 рёбер в графе являются мультирёбрами. Это значит, что 70% транзакций в датасете не были уникальными.
3. В изучаемой выборке 1321 аккаунт, из которых 22 – фишинговые, проводил транзакции самому себе. Всего же таких транзакций было очень мало, относительно общего количества – всего 5926.
4. 63% фишинговых аккаунтов совершали повторяющиеся транзакции. То есть большая часть мошенников взаимодействовала с другими аккаунтами не единожды.

Таким образом, фишинговые аккаунты вписываются в паттерн поведения большинства и не выделяются по критерию переводов между собой, однако более половина из них совершали неуникальные транзакции.

Проделанный анализ позволяет заключить, что на вход будущей нейронной сети может быть достаточно передавать информацию о транзакциях с некоторым «акцентов» на связности тех или иных узлов (степеней).

### **Подграфы и избавление от чрезмерной несбалансированности**

Выделение сообществ (подграфов) может быть полезным при работе с графовыми нейронными сетями, которые используют графы для представления данных. Этот процесс может помочь улучшить представление данных в графовых нейронных сетях, позволяя модели сосредоточиться на более узких и информативных подграфах исходного графа, что может привести к лучшей производительности и более точным предсказаниям. Более того, в случае нахождения сообществ, которые будут абстрагированы от миноритарного класса – удаление таковых будет являться неискажающим выборку шагом для достижения сбалансированности между размерностями представленных классов.

В библиотеке NetworkX реализовано несколько алгоритмов поиска сообществ в графах, наиболее универсальные из которых:

1. Алгоритм Лувена (Louvain algorithm) - использует метод модулярности, который позволяет определить, какие узлы в графе находятся ближе друг к другу, чем к другим узлам, и какие узлы составляют сообщество. Алгоритм Лувена является одним из наиболее распространенных алгоритмов поиска сообществ, так как он обладает высокой скоростью работы и хорошей точностью. Он был разработан в 2008 году.
2. Алгоритм Гирвана-Ньюмана (Girvan-Newman algorithm) - использует иерархическое разбиение графа на сообщества, основанное на удалении ребер с наибольшими межкластерными весами. Этот алгоритм относительно медленный, но он может работать с различными типами графов, включая графы с большим количеством узлов и ребер.

Для выделения сообществ в графе транзакций Ethereum был выбран первый вариант.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 6. Выделение сообществ в графе с помощью алгоритма Louvain

Алгоритм Louvain отработал в два этапа за 7 минут 10 секунд и обнаружил 15719 сообществ. На первом этапе был применён жадный алгоритм, который максимизировал модулярность в каждом сообществе путем слияния близлежащих узлов. На втором этапе он объединил сообщества, полученные на первом этапе, в более крупные сообщества, чтобы максимизировать общую модулярность [6].

Сравнительно большое число найденных сообществ дало простор для исследования большого числа подграфов, чтобы удалить те из них, которые не связаны с фишинговыми узлами. Выполнение последовательности действий

1. Обход сообществ;
2. Определение подграфов без фишинговых узлов;
3. Удаление из графов узлов, которые есть в таких сообществах.

позволило избавиться от 82% вершин и 85% рёбер без влияния на ближайшие связи фишинговых аккаунтов. При этом в графе не возникло изолированных вершин.

Граф значительно-уменьшенного размера стало возможным визуализировать на имеющихся мощностях. Однако, стоит отметить, что наиболее наглядная картина формируется при отображении лишь фишинговых аккаунтов. Несмотря на их общую «направленность деятельности» связи между ними не прослеживаются, связана лишь пара вершин.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 7. Визуализация фишинговых вершин графа в редакторе Gephi

Проделанная работу по анализу и понимаю датасета, его упрощению, позволила сформировать представление о типах архитектур нейронных сетей, которые могут помочь решить проблему классификации.

## **Архитектуры нейронных сетей для работы с графами**

Для решения задачи классификации графовых узлов распространены следующие классы моделей [1]:

1. Графовые нейронные сети (Graph Neural Networks, GNNs) — это класс моделей, разработанных специально для работы с графами. Они обрабатывают графы, используя свёрточные слои, агрегирующие слои и пространственную инвариантность.

2. Модели на основе узловых эмбеддингов — это модели, которые сначала преобразуют каждый узел в вектор, называемый узловым эмбеддингом, а затем используют эти эмбеддинги для предсказания атрибутов узлов. Примерами таких моделей являются node2vec и DeepWalk.

3. Модели на основе метрик — это модели, которые используют метрики для измерения сходства между узлами в графе. Затем они используют эти метрики для предсказания атрибутов узлов. Примером такой модели является модель на основе косинусного расстояния.

В данной работе созданы различные варианты первого типа - Графовых нейросетей. Это обусловлено прежде всего их специализацией на работе с графами, сохранением связей между рёбрами внутри. Кроме того, они достаточно легко расширяемы.

Для того, чтобы обучить модель на графовых данных, необходимо выполнить преобразование элементов графа в матричное представление. Особенности, которые необходимо учесть:

1. Выделение меток («labels» - целевой переменной, которую нужно предсказывать) в отдельный массив – атрибут «isp» узлов;

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 8. Формирование из атрибута узлов тензора целевой переменной

1. Рёбра имеют два атрибута разных форматов, которые тоже необходимо передавать в модель для обучения;

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 9. Формирование тензора признаков для каждого ребра

1. Рёбра должны быть преобразованы в матрицу смежности или инцидентности, а названия узлов переведены из строкового типа в числовой;

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 10. Присвоение каждому адресу в блокчейне индекса - перевод строки в число

1. Компенсирование отсутствия атрибутов узлов. Среди вариантов – one-hot вектора узлов, степени узлов, векторы соседей.



Рисунок 11. Использование входящих и исходящих степеней вершин графа в качестве признаков (атрибутов) узлов

### **Выбор библиотеки для работы с GNN**

Наиболее популярные библиотеки, которые позволяют реализовать GNN на Python - PyTorch Geometric (PyG) и Deep Graph Library (DGL). Обе они основаны на Torch, однако PyG предоставляет более понятный интерфейс (высокоуровневый API) и в последние годы ввиду своего удобства обрёл большую популярность в сообществе. При этом же DGL является более гибкой библиотекой и не ограничена сетями передачи сообщений (классические графовые свёрточные сети) [3].

В данной работе из-за более понятного интерфейса будет использоваться PyTorch Geometric, поэтому необходимо формализованную в тензоры информацию о графе упаковать в объект Data библиотеки PyG, который в дальнейшем и будет передаваться в нейронную сеть [2].

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рисунок 12. Инициализация объекта Data и указание необходимых для работы моделей свойств (признаки узлов, матрица смежности, тензор признаков рёбер, целевая переменная, количество классов для предсказания, количество атрибутов узлов)

## **Графовая свёрточная нейронная сеть**

### **Создание**

В качестве первого класса нейронной сети для решения задачи выбрана графовая свёрточная нейронная сеть (Graph Convolutional Network - GCN). Спроектированная архитектура упрощённо показана на схеме:

Изображение выглядит как диаграмма, линия, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 13. Используемая архитектура графовой свёрточной нейронной сети

Созданная архитектура представляет собой двухслойную модель с двумя слоями GCN-сверток и одним полносвязным слоем. Входной слой имеет размерность количества узлов в графе (500000), скрытый слой -16, а выходной слой – 2 (так как классификация бинарная). Внутри каждого GCN-слоя используется GCNConv, который выполняет свертку входных признаков в узлах графа с соседними узлами на основе их реберной структуры. После свертки применяется функция активации ReLU, которая нелинейно преобразует выходные данные. Кроме того, входные ребра также имеют атрибуты, которые учитываются во время свертки. В конце модели применяется полносвязный слой, который интерпретирует выходные данные как один из двух классов аккаунта.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 14. Исходный код спроектированной GCN

Ключевой слой модели – свёрточный. GCNConv - это один из наиболее распространенных слоёв в графовых нейронных сетях, который был представлен в статье «Semi-Supervised Classification with Graph Convolutional Networks» в 2017 году. Он реализует свертку на графах, используя матрицу смежности и атрибуты узлов.

GCNConv использует методы линейной алгебры, чтобы создать матрицу признаков для каждого узла, объединяя атрибуты узла и информацию о его соседях. Представление каждого узла получается путем умножения матрицы смежности на матрицу признаков и взвешивания полученных сумм с помощью весов, которые обучаются во время процесса обучения.

GCNConv является одним из наиболее простых и эффективных способов представления графов в виде векторов и используется в различных задачах классификации и кластеризации графов.

Данная архитектура по сравнению с другими GNN, например, GAT (рассмотрена далее) или GraphSAGE, не имеет механизмов внимания и агрегации, содержит лишь 16 нейронов в скрытом слое, что делает ее менее сложной и понятной в использовании.

### **Обучение**

В качестве функции потерь выбрана отрицательная логарифмическая функция правдоподобия (negative log-likelihood loss - NLL loss). Она является стандартной функцией потерь для задачи классификации в машинном обучении. Функция NLL loss вычисляет вероятность принадлежности данного объекта к положительному классу и минимизирует отрицательный логарифм этой вероятности, при этом максимизируя вероятность отнесения объекта к правильному классу. Эта функция позволяет эффективно оптимизировать параметры модели и достигать высоких показателей точности классификации.

В качестве оптимизатора для обновления параметров модели в процессе обучения используется torch.optim.Adam(). Он позволяет автоматически адаптировать скорость обучения в зависимости от градиентов параметров модели, что улучшает процесс оптимизации.

Для выбора определенных элементов из датасета, например, узлов или ребер графа, которые будут использоваться для обучения, валидации или тестирования модели в PyTorch Geometric используется маска.

Она позволяет определить, какие узлы или ребра графа должны быть выбраны для каждой из трех выборок.

Для разбиения графа на тренировочную и тестовую выборки, можно создать булева маску, где значение «True» соответствует узлам или ребрам, которые должны быть использованы для тренировки модели, а значение «False» - для тестирования модели.



Рисунок 15. Инициализация масок для разбиения датасета PyTorch Geometric на обучающую и тестовую выборки

Процесс обучения модели представляется из себя вызов функций «train» и «test» каждую эпоху.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 16. Функции процесса обучения модели

Функция «train» выполняет обучение GCN на выбранном пакете данных.

1. Вначале устанавливается режим обучения.

2. Все градиенты обнуляются, чтобы избежать накопления градиентов между разными итерациями обучения.

3. Модель получает выходные данные (предсказания) - вычисляется значение функции потерь.

4. Обратное распространение ошибки и обновление весов с помощью оптимизатора.

Функция «test» выполняет оценку производительности модели на тестовом наборе данных.

1. Устанавливается режим оценки.

2. Модель получает выходные данные для всего набора данных, вычисляются предсказания модели.

3. Далее, вычисляется точность («accuracy») модели на двух наборах данных: обучающем и тестовом. Это позволяет оценить способность модели к обобщению на новые данные.

4. Функция возвращает точность для обоих наборов данных.

Обучение модели проводилось 100 эпох, и заняло около 5 минут, проходило достаточно равномерно с точки зрения пропорционального уменьшения ошибок. Из графика ниже видно, что модель на обучающей выборке достигает более высокой точности («accuracy») на тренировочной выборке, чем на тестовой, что может говорить о переобучении. В районе 85 эпохи модель обрела достаточно сильную обобщаю способность что привело к резкому подъёму значения accuracy и выходу на плато при значении 0.95.

Изображение выглядит как линия, текст, снимок экрана, диаграмма

Автоматически созданное описание

Рисунок 17. Сравнение метрики «accuracy» в ходе обучения GCN на обучающей и тестовой выборке

Метрики «precision», «recall», «f1-мера» также достигли чрезвычайно высоких значений (1.0, 0.98, 0.99 соответственно), что может говорить о влиянии неравномерности классов на процесс обучения. Необходимо совершенствовать набор данных путём увеличения числа фишинговых аккаунтов или добавлять регуляризацию или механизмы внимания для улучшения распознавания фишинговых аккаунтов.

## **Графовая сеть внимания**

### **Создание**

В качестве второго класса модели выбрана графовая сеть внимания (Graph Attention Network). Это тип графовых нейронных сетей, который позволяет учитывать важность соседних узлов при агрегации информации, что достигается путем вычисления весовых коэффициентов для каждого соседнего узла, на основе которых взвешенно суммируется информация с соседних узлов. GAT может учитывать контекст, обрабатывать разреженные данные, обнаруживать скрытые зависимости и демонстрировать высокую производительность.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 18. Используемая архитектура графовой нейронной сети внимания

Приведённая архитектура содержит следующие слои:

1. Dropout: Применение операции исключения элементов для входных данных с вероятностью 60%.
2. GATConv1: Операция графового внимания на входных данных с использованием весов голов heads\_layer1 = 2 – именно благодаря ним регулируется механизм внимания. Входные данные связаны с ребрами графа (edge\_index) и атрибутами ребер (edge\_attr).
3. ReLU: Функции активации ReLU для активации выходов слоя.
4. Dropout: Операция исключения элементов для выходных данных после активации с вероятностью 60%.
5. GATConv2: Аналогично GATConv1, операция графового внимания на выходных данных после Dropout, с использованием весов голов heads\_layer2 = 2.
6. Log Softmax: Операция логарифмической softmax для получения вероятностных предсказаний.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 19. Исходный код спроектированной GAT

Стоит уточнить, что «внимание» регулируется параметров «голов» в слое GATConv. Каждая голова генерирует собственные весовые коэффициенты, а результаты всех голов объединяются или не объединяются, в зависимости от параметра «concat».

Число голов влияет на сложность модели и её способность захватывать и моделировать сложные зависимости между узлами графа. Чем больше голов, тем больше вариаций внимания и взаимодействий между узлами модель сможет учесть. Однако, более высокое количество голов также приводит к увеличению вычислительной сложности и требует больше параметров для обучения.

### **Обучение**

Данная модель GAT обучалась с такими же сопутствующими настройками, как и GCN:

1. Оптимизатор – Adam;
2. Функция потерь - логарифмическая функция правдоподобия (NLL loss);
3. Разделение на обучающую и тестовую выборки – 65% и 35%;
4. Длительность обучения - 100 эпох.

Обучение сети внимания длилось на минуту дольше, чем обучение свёрточной сети.

Изображение выглядит как текст, линия, диаграмма, снимок экрана

Автоматически созданное описание

Рисунок 20. Сравнение метрики accuracy у GCN и GAT на двух выборках по ходу обучения

Из графика сравнения точностей двух моделей на разных выборках видно, что более сложная архитектура графовой нейронной сети внимания раньше свёрточной начала находить закономерности в данных и уже на 20 эпохе имела accuracy более 0.8. Однако, достигнув 0.92, точность на обоих выборках у GAT почти не изменялась. Кроме того, по превышению тестового accuracy над тренировочным, можно сделать вывод, что GAT недообучилась.

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание

Таблица 3. Сравнение метрик классификации «precision», «recall», «f1-мера» для испытанных моделей

При сравнении других метрик, оцениваемых при решении задачи классификации напрашивается вывод, что архитектурно более простая графовая свёрточная нейронная сеть показывает результаты лучше, чем графовая сеть внимания.

Более низкий «recall» по сравнению с «accuracy» и «precision» может говорить о том, что модели более склонна к ошибке «false negatives», то есть классифицирует аккаунты неверно как "обычные", когда на самом деле они являются фишинговыми. Это связано с тем, что модели используют недостаточно сложную и несбалансированную выборку для обучения.

# **ЗАКЛЮЧЕНИЕ**

В результате работы по созданию модели глубокого обучения для классификации фишинговых криптовалютных аккаунтов можно сделать следующие выводы. Данные, которые представлены в виде графа, удобно обрабатывать и исследовать с помощью библиотеки NetworkX. Однако транзакционный граф Ethereum, из-за своего масштаба, представляет собой крайне несбалансированный набор данных. Эта особенность требует выравнивания соотношения классов путем удаления большого числа экземпляров данных. И хотя данная работа была проделана, выборка всё ещё не сбалансирована, о чём говорят результаты модели при определении фишинговых аккаунтов.

В качестве класса модели машинного обучения на графах наиболее оптимальными являются графовые нейронные сети. Для начала обучения сетей такого класса необходимо преобразовать граф в набор матрицы смежности или инцидентности и тензоров атрибутов аккаунтов и транзакций, что в рамках данной работы было достигнуто с помощью библиотеки PyTorch Geometric.

Графовая сверточная нейронная сеть показала лучшие, чем более сложная графовая сеть внимания, результаты, однако обучение проходило на крайне малом числе примеров фишинговых узлов, по сравнению с обычными адресами Ethereum. Так, результаты могут быть улучшены дополнительным исследованием данных, использованием более сложных алгоритмов, добавлением регуляризаций различного уровня.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ИНТЕРНЕТ-РЕСУРСОВ**

* 1. Jie Zhou, Ganqu Cui Graph Neural Networks: A Review of Methods and Applications. // [Электронный ресурс] URL: <https://arxiv.org/abs/1812.08434> (дата обращения: 11.05.2023)
  2. Matthias Fey, Jan Eric Lenssen Fast Graph Representation Learning with PyTorch Geometric // [Электронный ресурс] URL: <https://arxiv.org/abs/1903.02428> (дата обращения: 11.05.2023)
  3. Xuanchen Zhou, Wenzhong Yang Detecting Phishing Accounts on Ethereum Based on Transaction Records and EGAT // [Электронный ресурс] URL: <https://www.mdpi.com/2079-9292/12/4/993> (дата обращения: 11.05.2023)
  4. Графы в Python: введение и знакомство с лучшими библиотеками [Электронный ресурс] // URL: <https://habr.com/ru/companies/ruvds/articles/705368/> (дата обращения: 11.05.2023)
  5. Коломейченко М. И., Чеповский А. М. Визуализация и анализ графов больших размеров // Бизнес-информатика. 2014. №4 (30). URL: <https://cyberleninka.ru/article/n/vizualizatsiya-i-analiz-grafov-bolshih-razmerov> (дата обращения: 12.05.2023).
  6. Лобанова С.Ю., Чеповский А.А. Комбинированный алгоритм выделения сообществ в графах взаимодействующих объектов // Бизнес-информатика. 2017. №4 (42). URL: <https://cyberleninka.ru/article/n/kombinirovannyy-algoritm-vydeleniya-soobschestv-v-grafah-vzaimodeystvuyuschih-obektov> (дата обращения: 12.05.2023).