Федеральное государственное бюджетное образовательное учреждение высшего образования

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**Департамент анализа данных и машинного обучения**

Пояснительная записка к курсовой работе

по дисциплине «Машинное обучение в семантическом и сетевом анализе»

на тему:

«Предсказание центральности узлов по PageRank с помощью графовых нейронных сетей»

Выполнил:

студент группы ПМ21-2 факультета информационных технологий и анализа больших данных

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Астахов В.И.

Научный руководитель:

д.ф.н., доцент, профессор, Золотова Т.В.

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

2024 г

Оглавление

[**Задача** 3](#_Toc175762755)

[**Цель** 3](#_Toc175762756)

[**Желаемый результат** 3](#_Toc175762757)

[**Актуальность** 3](#_Toc175762758)

[**Этапы** 4](#_Toc175762759)

[***Что такое PageRank, центральность, и зачем они нужны*** 4](#_Toc175762760)

[***Алгоритм генерации, описание и обработка данных*** 6](#_Toc175762761)

[***Создание моделей нейронных сетей*** 9](#_Toc175762762)

[***Описание используемых слоев*** 9](#_Toc175762763)

[***Описание работы оптимизаторов*** 12](#_Toc175762764)

[***Обучение и тестирование моделей на небольшом графе*** 19](#_Toc175762765)

[***Сравнительный анализ:*** 21](#_Toc175762766)

[***Обучение и тестирование моделей на графах большего размера*** 23](#_Toc175762767)

[**Вывод** 26](#_Toc175762768)

[**Список литературы** 27](#_Toc175762769)

# **Задача**

Провести исследование, сравнив несколько моделей графовых нейронных сетей с разными видами слоев и оптимизаторов, чтобы выявить лучшую для решения задачи регрессии по предсказанию центральности узлов в графе с помощью алгоритма PageRank. Попытаться создать модель, которую в дальнейшем будет возможно использовать в простой рекомендательной системе на динамических графах или в другой похожей прикладной задаче.

# **Цель**

Продемонстрировать владение навыками работы с графами и нейронными сетями, в частности с графовыми нейронными сетям (GNN), владение основными инструментальными средствами библиотек networkx и tourch языка программирования Python, средствами визуализации графовых данных, использования и усовершенствования обучаемой модели. Продемонстрировать знание теории графов и понимание предметной области.

# **Желаемый результат**

Рабочая модель графовой нейронной сети, которую будет возможно применять в прикладных задачах.

# **Актуальность**

Предсказание центральности узлов в сетях является крайне актуальной задачей в современном обществе, начиная сферой рекламы и заканчивая биоинформатикой. Например, с алгоритмом PageRank мы так или иначе сталкиваемся каждый день, когда ищем что-то в интернете, ведь он применяется для ранжирования результатов поиска, чтобы пользователь в первую очередь видел наиболее подходящие и релевантные страницы, подходящие под его запрос. Благодаря нему мы зачастую находим нужную нам информацию уже на первой странице поиска. Также этот алгоритм может применяться для показа наиболее релевантной рекламы, генной инженерии, чтобы понять, какие из генов являются более важными и как их мутации могут влиять на другие гены, составление транспортных маршрутов для грузовых перевозок и т.д.. Существует бесчисленное множество самых неожиданных областей, где концепция центральности узлов и алгоритм PageRank могут быть использованы.

# **Этапы**

1. Описание теоретической базы, необходимой для реализации проекта и понимая его работы. Сформулировать актуальность и возможность практического применения проекта.
2. Генерация, описание и обработка данных.
3. Создание моделей графовых нейронных сетей с разными оптимизаторами, типами слоев и данных.
4. Описание работы выбранных оптимизаторов.
5. Обучение моделей и их тестирование на графе небольшого размера.
6. Проведение сравнительного анализа для выявления наилучших сочетаний модель + оптимизатор.
7. Отсеивание наихудших кандидатов и тестирование моделей на графах большего размера.

## ***Что такое PageRank, центральность, и зачем они нужны***

Для начала стоит дать небольшие определения основным терминам и сделать введение в теорию графов.

*Центральность* — это мера, используемая в теории графов для оценки важности узлов в сети.

*PageRank* — это алгоритм анализа ссылок, разработанный и используемый Google для оценки важности веб-страниц в интернете.

Если рассматривать веб страницы как узлы графа, то ссылки между страницами рассматриваются как рёбра этого графа. PageRank узла — это численное значение, которое обозначает важность узла в графе. Она вычисляется исходя их числа входящих ссылок на узел, существующих в графе. Страница считается важной, если на неё ссылаются другие важные страницы.

**Обзор Алгоритма:**

**Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание**

## ***Алгоритм генерации, описание и обработка данных***

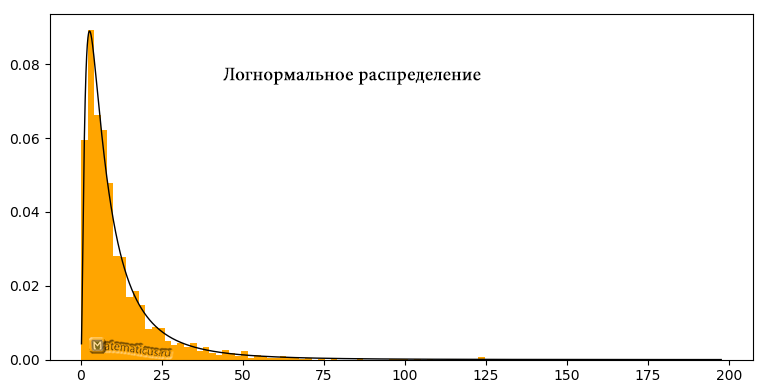
С помощью функций из networkx я генерирую несколько не связанных между собой графов двух видов (метод Эрдёша и безмасштабный метод), вычисляю значения PageRank для каждого узла с помощью встроенной функции networkx, после чего объединяю их в один объект и объединяю массивы их значений PageRank, тем самым имитируя несколько условных сайтов в интернете. Далее я преобразую граф, чтобы узлы были пронумерованы целыми числами и граф был в формате DGL. Конвертирую значения PageRank в формат PyTorch tensor и задаю признаки узлов (либо torch eye, либо случайные значения в диапазоне [0;1). Устанавливаю значения PageRank метками узлов. Формирую тренировочную(70%), валидационную(15%) и тестовую(15%) выборки.

**Графы методом Эрдёша:**

Граф Эрдёша-Реньи - модель генерации случайных графов, в которой все графы с фиксированным набором вершин и фиксированным набором рёбер одинаково вероятны. В нашей работе используется биноминальная модель данного алгоритма G(n, p), в которой граф строится путём случайного добавления рёбер. Каждое ребро включается в граф с вероятностью p независимо от остальных рёбер. Эквивалентно, все графы с n узлами и M рёбрами имеют одинаковую вероятность.

**Графы безмасштабным методом:**

Безмасштабный граф — это модель графа, в которой сущетсвуют узлы, притягивающие непропорционально большое число новых узлов (концентраторы). Число связей у отдельно взятого узла распределяется по логарифмическому закону, значит большинство узлов имеют ограниченное число связей, а концентраторы напротив, имеют аномально большое их число.



В данном случае мы можем представить, что по оси Y у нас откладывается число узлов с определенным количеством связей, а по оси X количество этих самых связей.

Безмасштабные графы могут использоваться в качестве модуляции социальных связей, транспортных сетей, социальных сетей, галактик и т.д..

Безмасштабные сети крайне неустойчивы к удалению узлов, ведь при удалении даже 5% узлов граф рассыпется на мелкие кусочки и гигантские кластеры перестанут существовать. Также, рост сети не приводит к увеличению ее устойчивости к случайным воздействиям.

Безмасштабная сеть может быть построена путём постепенного добавления узлов к существующей сети и создания связей с существующими узлами с предпочтительным присоединением таким образом, что вероятность создания связи с данным узлом i пропорциональна количеству существующих связей k\_i, которые имеет этот узел:

**Изображение выглядит как Шрифт, текст, Графика, белый

Автоматически созданное описание**

Пример из кода:

Изображение выглядит как текст, снимок экрана, часы, дисплей

Автоматически созданное описание

В данном случае программа создаст один направленный граф по модели Эрдёша, состоящий из 15 узлов, в котором вероятность появления связи между двумя узлами равна 20%, и один направленный граф по модели безмасштабного графа, тоже с 15 узлами с параметрами по умолчанию из алгоритму из networkx, а именно:

1. alpha = 0.41: Вероятность добавления нового узла, соединенного с существующим узлом, выбранным случайным образом в соответствии с распределением по степени.
2. beta = 0.54: Вероятность добавления ребра между двумя существующими узлами. Один существующий узел выбирается случайным образом в соответствии с распределением входящих степеней, а другой выбирается случайным образом в соответствии с распределением исходящих степеней.
3. gamma = 0.05: Вероятность добавления нового узла, соединенного с существующим узлом, выбранным случайным образом в соответствии с распределением исходящих степеней.

## ***Создание моделей нейронных сетей***

После создания и обработки датасета, мы приступаем к созданию нескольких моделей нейронных сетей с разными типами слоев, оптимизаторов и данных в качестве признаков. Я использую слои GCN. SAGE и GAT, а в качестве оптимизаторов использую Adam, AdamW, Adamax, Adadelta и SGD. Таким образом, на первом тапе я получаю 30 моделей с разными сочетаниями типа слоев, типа оптимизатора и типа данных, используемых в качестве признаков узлов. Объединяет их только то, что во всех из них используется 16 скрытых слоев.

Пример из кода:

Изображение выглядит как текст, снимок экрана, дисплей, Шрифт

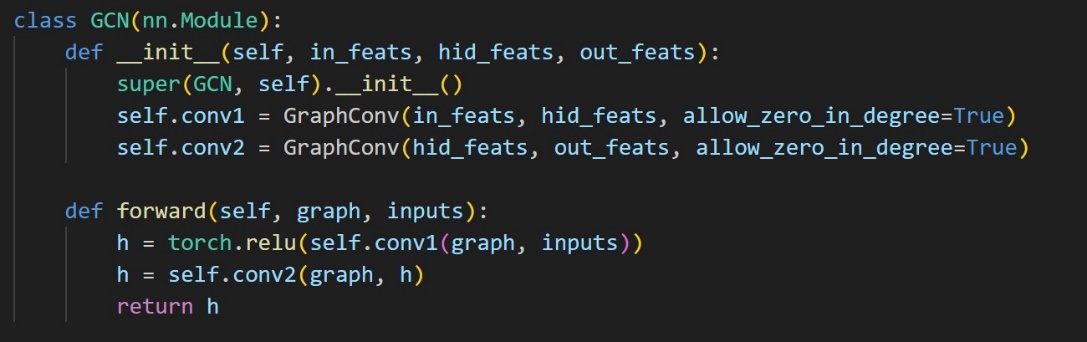
Автоматически созданное описание

## ***Описание используемых слоев***

В нашем сравнении используются три вида слоев: GCN, SAGE и GAT. Стоит отдельно остановиться на каждом из них, чтобы понимать их отличия.

**GCN (Graph Convolutional Networks):**

Операция свертки в GCN агрегирует информацию из соседних узлов с учетом их признаков, нормализует эту информацию с использованием степеней узлов, применяет линейное преобразование с помощью матрицы весов и применяет нелинейную функцию активации для получения окончательного представления узлов на следующем слое.

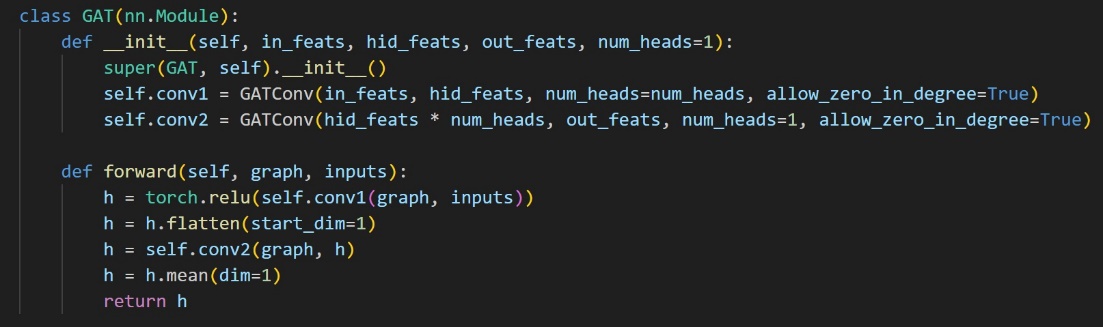


**GAT (Graph Attention Network):**

Модель графовых нейронных сетей, которая использует механизм внимания для эффективного агрегирования информации из соседних узлов в графе. Данные слои способны учитывать разный вклад соседей, что делает их особенно полезными для сложных графов с разнообразными типами связей.

В отличие от классического GCN, который вычисляет взвешенную сумму признаков соседних узлов с постоянными весами, GAT использует механизм внимания для динамического определения весов на основе содержимого узлов. Это позволяет модели уделять больше внимания более важным узлам в графе. GAT позволяет каждому узлу иметь разные веса в зависимости от контекста, что делает модель более гибкой и мощной для обработки различных графов.

В нашей задаче данный тип слоев должен подойти лучше всего.

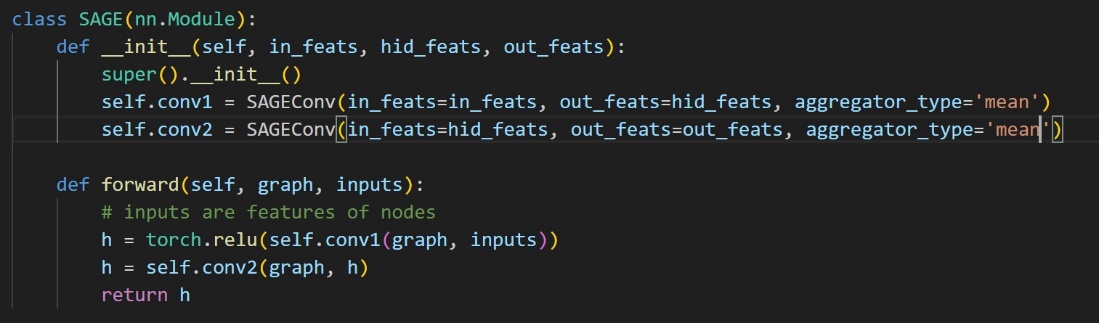


**SAGE:**

Предназначен для индуктивного обучения на графах. часто применяется для задач классификации узлов, предсказания связей и классификации графов.

Работает, собирая информацию от соседних узлов для обновления представления текущего узла. Это делает его особенно эффективным для захвата локальной структурной информации.

SAGE поддерживает несколько функций агрегации, но в нашей работе используется только mean. После агрегации признаки узла можно комбинировать с агрегированной информацией с помощью линейных преобразований и нелинейностей, таких как ReLU, что мы и делаем.



## ***Описание работы оптимизаторов***

Дадим краткое математическое описание работы выбранных оптимизаторов, а именно Adam, AdamW, Adamax, Adadelta и SGD.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Описание алгоритма в документации PyTorch:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Описание алгоритма в документации PyTorch:

Изображение выглядит как текст, снимок экрана, Шрифт, рукописный текст

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Описание алгоритма в документации PyTorch:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

*Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание*

*Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание*

Описание алгоритма в документации PyTroch:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Описание алгоритма в документации PyTroch:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

## ***Обучение и тестирование моделей на небольшом графе***

Теперь, мы переходим к основному этапу работы, а именно обучению и тестированию моделей. В качестве показателя качества модели мы используем метрику loss, а чтобы не допускать переобучения моделей и ускорить их обучение мы предусмотрели алгоритм преждевременной остановки, который отслеживает изменения validation loss, и если нет улучшений в течение 10 эпох, то обучение прекращается и программа переходит к тестированию модели на тестовой выборке. Чтобы этот процесс был максимально информативным, мы реализовали отображение графика изменения Loss, отображение графа с реальными значениями PageRank, которые были вычислены при помощи встроенного алгоритма networkx, а также отображение графа с предсказанными значениями центральности. В конце программа выводит график, где синими точками обозначены реальные значения, а оранжевым предсказанные. Таким образом, можно максимально наглядно оценить то, насколько хорошо наша модель справилась с поставленной задачей.

Как было сказано выше, мы используем метрику Loss в качестве оценки качества обучения модели. Это показатель, который указывает, насколько хорошо модель выполняет свою задачу. Loss измеряет расхождение между предсказанными моделью значениями и реальными значениями в обучающем наборе данных. Чем меньше значение функции потерь, тем лучше. В нашей работе в качестве функции потерь используется средняя квадратичная ошибка (MSE), так как она является оптимальным вариантом для решения задач регрессии. Суть метода заключается в том, чтобы минимизировать сумму квадратов отклонений фактических значений от расчётных. Формула MSE имеет следующий вид:

**Изображение выглядит как Шрифт, текст, рукописный текст, Графика

Автоматически созданное описание**

где y\_i – истинное значение, y\_i^ - предсказанное значение, n – количество наблюдений.

Пример из кода:

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

**Изображение выглядит как снимок экрана, График, линия, текст

Автоматически созданное описание**

**Изображение выглядит как диаграмма, линия, снимок экрана, карта

Автоматически созданное описание**

**Изображение выглядит как диаграмма, линия, снимок экрана, карта

Автоматически созданное описание**

**Изображение выглядит как программное обеспечение, текст, Мультимедийное программное обеспечение, Значок на компьютере

Автоматически созданное описание**

## ***Сравнительный анализ:***

Далее нам нужно сравнить результаты каждой модели, чтобы выбрать те, с которыми мы продолжим работать. Для этого мы составили следующие таблицы:

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание

Как можем заметить, хуже всего себя показали модели со слоями типа SAGE вне зависимости от оптимизатора и типа признаков, поэтому мы исключим их из дальнейшего сравнения. Лучше всех себя показали модели со слоями типа GAT, так как они даже смогли дать достаточно точные результаты, когда в качестве признаков использовался шум, чего не смогли продемонстрировать модели с слоями типа GCN, хотя они успешно справляются в случаях, когда в качестве признаков выступают torcheye.

## ***Обучение и тестирование моделей на графах большего размера***

После предыдущего этапа из нашего сравнения выбыли модели со слоями типа SAGE. Продолжаем тестировать модели на графе больших размеров и получаем следующий результат:

Изображение выглядит как текст, снимок экрана, Шрифт, часы

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

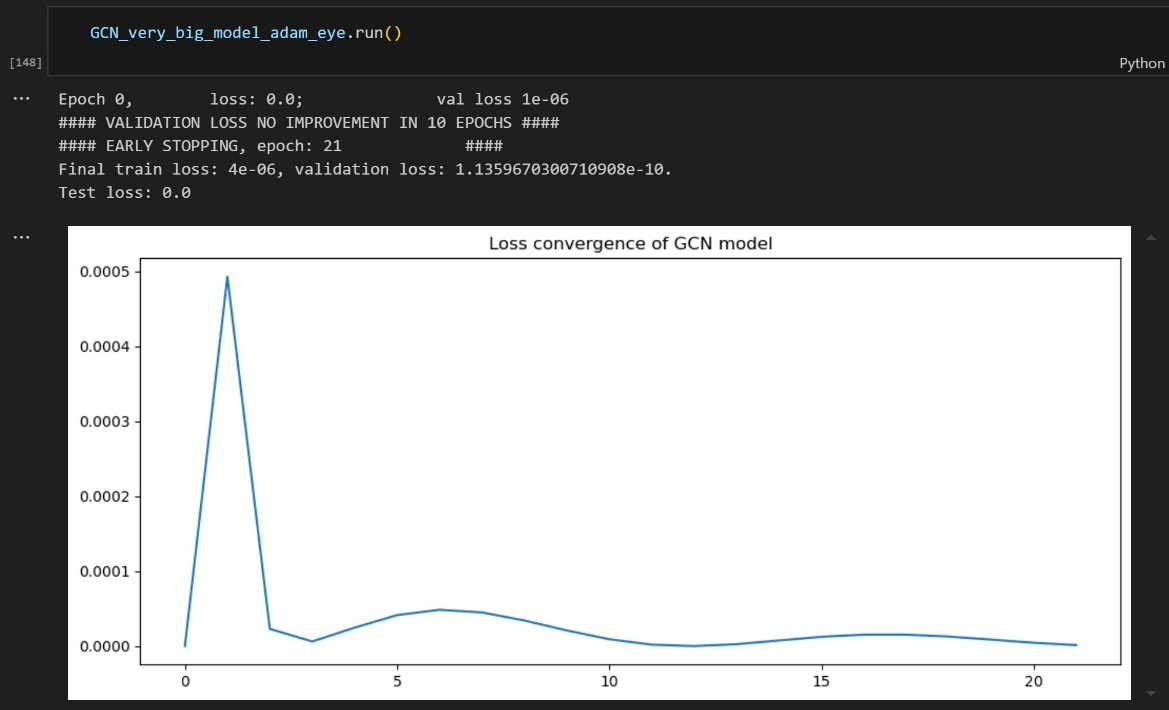
Как мы можем видеть, на графах такого размера уже ни одна вариация модели не может справиться в случаях, когда в качестве признаков узлов выступает шум. Но в случаях, когда признаками выступают torch eye оба типа слоев хорошо справляются на трех оптимизаторах: Adam, AdamW, Adamax. Проведем последние тестирование на графе огромных размеров, который будет единственным графом во всем датасете, и он будет создан по методу Эрдёша.

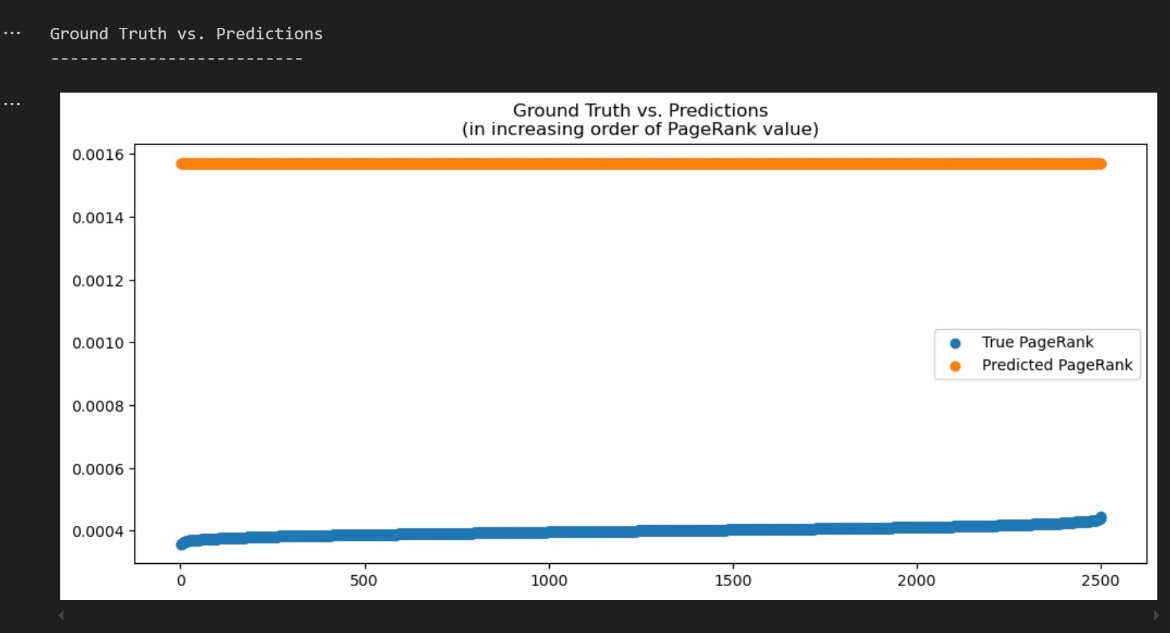
Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Также мы не будем пытаться тестировать модели на датасете с шумами, а будем использовать только torch eye в качестве признаков узлов.

Итоговые результаты для всех моделей имеют схожий вид:





В конечном итоге ни одна модель не смогла справиться с такой задачей, что может быть вызвано тем, что значения PageRank на таких огромных графах становятся слишком маленькими и не могут быть адекватно вычислены.

# **Вывод**

Мы пришли к тому, что лучше всего с задачей предсказания центральностей узлов по PageRank справились модели нейронной сети на слоях типа GAT с оптимизаторами Adam, AdamW и Adamax. В потенциале, такие модели даже могли бы найти применение в коммерческих прикладных задачах по типу рекомендательных систем, несмотря на свою несложную реализацию. Но с графами слишком больших размеров ни одна модель справиться не смогла, что ставит под сомнение возможность их использования в больших проектах.

# **Список литературы**

1. PAGERANK: алгоритм ранжирования WEB-страниц // Habr  
   URL: <https://habr.com/ru/articles/533096/> (дата обращения: 10.05.2024)
2. Прошлое, настоящее и будущее Google PageRank.   
   URL: <https://www.link-assistant.com/ru/news/google-pagerank-algorithm.html> (дата обращения: 10.05.2024)
3. Циликов Н.С., Федосин С.А. Графовые нейронные сети.  
   URL: <https://cyberleninka.ru/article/n/grafovye-neyronnye-seti> (дата обращения: 10.05.2024)
4. Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, Maosong Sun. Graph neural networks: A review of methods and applications // KeAi  
   URL: <https://arxiv.org/pdf/1812.08434> (дата обращения: 10.05.2024)
5. Graph Neural Networks: просто на математическом // Habr  
   URL: <https://habr.com/ru/articles/697704/> (дата обращения: 10.05.2024)
6. Jiaxing Zhang, Zhuomin Chen, Hao Mei, Dongsheng Luo, Hua Wei. RegExplainer: Generating Explanations for Graph Neural Networks in Regression Task   
   URL: <https://arxiv.org/pdf/2307.07840> (дата обращения: 10.05.2024)
7. URL: <https://docs.dgl.ai/en/0.8.x/api/python/nn-pytorch.html> (дата обращения: 10.05.2024)