ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

MẠNG XÃ HỘI

Đề tài:

Transaction Graph Anomaly Detection

Giảng viên hướng dẫn:

Ths. Hà Lê Hoài Trung

Sinh viên thực hiện:

Nguyễn Hoàng Đăng Khoa – 21520999

Cù Ngọc Hoàng - 21522086

Nguyễn Trần Gia Kiệt – 21522258

Thành phố Hồ Chí Minh, tháng 11 năm 2024

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

MẠNG XÃ HỘI

Đề tài:

Transaction Graph Anomaly Detection

Giảng viên hướng dẫn:

Ths. Hà Lê Hoài Trung

Sinh viên thực hiện:

Nguyễn Hoàng Đăng Khoa – 21520999

Cù Ngọc Hoàng - 21522086

Nguyễn Trần Gia Kiệt – 21522258

Thành phố Hồ Chí Minh, tháng 11 năm 2024

LỜI CẢM ƠN

Trước hết, nhóm chúng em xin gửi lời cảm ơn sâu sắc đến tập thể quý thầy cô trường Đại học Công nghệ Thông tin - Đại học Quốc gia TP.HCM và quý thầy cô khoa Hệ thống thông tin đã tạo điều kiện, giúp chúng em học tập và có được những kiến thức cơ bản làm tiền đề giúp chúng em hoàn thành được dự án này.

Đặc biệt, nhóm chúng em xin gửi lời cảm ơn chân thành và sâu sắc tới thầy Hà Lê Hoài Trung (Giảng viên giảng dạy lý thuyết môn Mạng xã hội – IS353). Nhờ sự hướng dẫn tận tình và chu đáo của thầy, nhóm chúng em đã học hỏi được nhiều kinh nghiệm và hoàn thành thuận lợi, đúng tiến độ cho dự án của mình.

Ngoài ra, chúng em cũng gửi lời cảm ơn đến tập thể lớp IS353.P11 khoảng thời gian qua đã đồng hành cùng nhau. Cảm ơn sự đóng góp của tất cả các bạn cho những buổi học luôn sôi nổi, thú vị và dễ tiếp thu.

Trong quá trình thực hiện tiểu luận, nhóm chúng em luôn giữ một tinh thần cầu tiến, học hỏi và cải thiện từ những sai lầm, tham khảo từ nhiều nguồn tài liệu khác nhau và luôn mong đạt được kết quả nhất có thể. Tuy nhiên, do vốn kiến thức còn hạn chế trong quá trình trau dồi từng ngày, nhóm chúng em không thể tránh được những sai sót, vì vậy chúng em mong rằng quý thầy cô sẽ đưa ra nhận xét một cách chân thành để chúng em học hỏi thêm kinh nghiệm nhằm mục đích phục vụ tốt các dự án khác trong tương lai. Xin chân thành cảm ơn quý thầy cô!

Nhóm thực hiện

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

……, ngày…….tháng……năm 2024

Người nhận xét

(Ký tên và ghi rõ họ tên)

MỤC LỤC

DANH MỤC HÌNH ẢNH

CHƯƠNG 1: GIỚI THIỆU

1.1. Giới thiệu lý do vì sao chọn chủ đề

- Trong bối cảnh dữ liệu lớn ngày càng phức tạp và khối lượng giao dịch tài chính tăng trưởng mạnh mẽ, việc phát hiện các giao dịch bất thường đóng vai trò quan trọng trong việc ngăn ngừa các hành vi gian lận và bảo vệ hệ thống tài chính. Những giao dịch bất thường, chẳng hạn như rửa tiền, gian lận thẻ tín dụng, hoặc giao dịch giả mạo, thường có xu hướng lẩn tránh các hệ thống phát hiện thông thường bằng cách ngụy trang như các giao dịch hợp pháp. Do đó, yêu cầu cấp thiết đặt ra là cần có một phương pháp phát hiện bất thường tiên tiến, có khả năng khai thác sâu các mối quan hệ và tương tác phức tạp trong dữ liệu giao dịch.
- Trong lĩnh vực này, *Graph Neural Networks (GNN)*, đặc biệt là *Graph Attention Networks (GAT)*, đã nổi lên như một công cụ hiệu quả trong việc xử lý các dữ liệu có cấu trúc đồ thị. GAT cho phép hệ thống phát hiện tự động học và xác định các giao dịch bất thường dựa trên mối quan hệ của các giao dịch với nhau, từ đó tăng độ chính xác và hiệu quả phát hiện.
- Vì vậy, nhóm quyết định nghiên cứu và áp dụng GAT vào chủ đề "Transaction Graph Anomaly Detection" để phát hiện các bất thường trong giao dịch và đánh dấu lại. Nhóm tin rằng đây là một hướng đi đầy hứa hẹn và có ý nghĩa thực tiễn.

1.2. Giới thiệu các nội dung lý thuyết phục vụ cho việc thực hiện chủ đề.

- Để thực hiện tốt chủ đề này, nhóm đã tìm hiểu về các kiến thức lý thuyết liên quan:
    - Anomaly Detection: Giới thiệu về lý thuyết phát hiện bất thường, các loại bất thường khác nhau (như bất thường ở mức nút, mức cạnh, và mức đồ thị), và các phương pháp phổ biến hiện nay. Đặc biệt, cần làm rõ cách phát hiện bất thường dựa trên đồ thị (graph-based

anomaly detection) và tầm quan trọng của nó trong lĩnh vực tài chính.

- GAT: một biến thể của GNN, có khả năng tự động học và điều chỉnh trọng số của các nút lân cận dựa trên tầm quan trọng của chúng. Điều này giúp tập trung vào các nút và cạnh quan trọng trong quá trình học, tối ưu cho việc phát hiện các giao dịch có xu hướng bất thường.

1.3. Xác định mục tiêu thực hiện chủ đề

- Mục tiêu chính của chủ đề *Transaction Graph Anomaly Detection* là áp dụng *Graph Attention Network (GAT)* để phát hiện bất thường hiệu quả, giúp phát hiện các giao dịch khả nghi trong các mạng lưới giao dịch phức tạp. Cụ thể:
    - Phát triển mô hình GAT cho bài toán phát hiện bất thường: Sử dụng GAT để học các mối quan hệ giữa các tài khoản và giao dịch, từ đó xác định các giao dịch có dấu hiệu bất thường và đánh dấu lại tài khoản đó. Mô hình sẽ tận dụng cơ chế attention để tập trung vào các giao dịch quan trọng, hỗ trợ tối ưu hóa quá trình phát hiện.
    - Tạo cơ sở cho các ứng dụng thực tiễn: Thiết lập một mô hình có thể dễ dàng triển khai trong thực tế, đáp ứng được nhu cầu giám sát và cảnh báo bất thường cho các hệ thống giao dịch tài chính và các nền tảng thương mại điện tử.
    - Cung cấp giải pháp phân tích và giám sát giao dịch toàn diện: Ngoài phát hiện các giao dịch bất thường, mô hình còn giúp cung cấp cái nhìn sâu sắc về mối quan hệ giữa các thực thể trong mạng lưới giao dịch, hỗ trợ việc phân tích và đưa ra quyết định cho các hệ thống phòng chống gian lận.

CHƯƠNG 2: NỀN TẢNG LÝ THUYẾT

2.1. Giới thiệu các nội dung lý thuyết liên quan phục vụ thực hiện chủ đề

2.1.1. Anomaly Detection

2.1.1.1. Giới thiệu

- Trong thời đại phát triển của học máy và học sâu, những điểm bất thường trong dữ liệu đóng vai trò quan trọng hơn so với những điểm thông thường. Công việc này được gọi là phát hiện bất thường, tập trung vào việc phát hiện các tín hiệu hoặc mẫu "khác biệt so với bình thường" bằng cách phân tích một lượng lớn dữ liệu, qua đó xác định và ngăn chặn các lỗi nghiêm trọng.
- Công việc này đóng vai trò quan trọng trong nhiều lĩnh vực có ảnh hưởng lớn, bao gồm:
  + An ninh mạng: phát hiện xâm nhập mạng, lỗi mạng, hoặc chương trình độc hại.
  + Tài chính: phát hiện gian lận thẻ tín dụng, tài khoản độc hại, giao dịch tiền bất hợp pháp,gian lận cho vay.
  + Thương mại điện tử: phát hiện các đánh giá giả mạo.
  + Mạng xã hội: phát hiện người dùng bất thường, giao dịch tiền thật.
  + Giám sát công nghiệp: phát hiện lỗi hệ thống.

2.1.1.2. Các vấn đề liên quan

a. Dữ liệu

- Các vấn đề cụ thể liên quan đến dữ liệu đồ thị bao gồm: tính chất quan hệ, tính không đồng nhất của đồ thị, tính động của đồ thị, sự đa dạng trong định nghĩa, thiếu các thước đo khoảng cách/độ tương đồng nội tại, và kích thước không gian tìm kiếm.
  + Tính chất quan hệ: Tính chất quan hệ của dữ liệu khiến việc định lượng mức độ bất thường của các đối tượng trong đồ thị trở nên

thách thức.Ở phương pháp phát hiện ngoại lai truyền thống, các đối tượng hoặc dữ liệu được xem như độc lập và phân phối đồng nhất, thì các dữ liệu trong đồ thị lại có mối quan hệ tương quan theo từng cặp. Do đó, cần phải xem xét cẩn thận "sự lan truyền bất thường" hoặc "sự liên đới thông qua quan hệ." Ví dụ: các người dùng thực hiện gian lận không chỉ có các đặc điểm bất thường mà còn hành động bất thường trong mối quan hệ tương tác. Họ có thể thực hiện đồng thời nhiều giao dịch và tương tác chuyển tiền với các nhà cung cấp cụ thể, điều mà các phương pháp truyền thống dựa trên trích xuất đặc trưng khó có thể khai thác.

+ Tính không đồng nhất của đồ thị: Tương tự như vấn đề không đồng nhất trong dữ liệu chung, loại thực thể và loại quan hệ trong đồ thị thường rất đa dạng. Ví dụ: trong đồ thị hệ thống máy tính, có ba loại thực thể: tiến trình (P), tệp (F), và INETSocket (I), cùng nhiều loại quan hệ: tiến trình khởi tạo tiến trình khác (P→P), tiến trình truy cập tệp (P→F), tiến trình kết nối với socket mạng (P→I)... Do tính không đồng nhất của các thực thể (nút) và các quan hệ (cạnh) trong đồ thị không đồng nhất, sự đa dạng giữa các quan hệ khác nhau có thể khác biệt rất lớn, làm tăng đáng kể độ khó trong việc xử lý đồng thời các nút và cạnh này.
+ Tính động của đồ thị: Do dữ liệu được thu thập định kỳ hoặc liên tục, đồ thị được xây dựng cũng thể hiện tính chất động. Việc phát hiện bất thường trở nên khó khăn do bản chất động này. Một số hoạt động bất thường có thể thể hiện các mẫu rõ ràng nhưng lại cố gắng che giấu chúng trong một đồ thị lớn, trong khi những hoạt động khác lại có các mẫu ẩn.Ví dụ: một mẫu bất thường rõ ràng trong hệ thống gợi ý: người dùng bất thường thường sử dụng nhiều tài khoản để

tăng cường sản phẩm mục tiêu, tạo ra một liên kết dày đặc với các cạnh giữa các tài khoản và sản phẩm trong một khoảng thời gian ngắn. Ngoài ra, dù các tài khoản này thực hiện các hành vi bất thường trong một số thời điểm, chúng vẫn hoạt động bình thường trong phần lớn thời gian, khiến cho các hành vi bất thường dài hạn bị che giấu và làm tăng độ khó trong việc phát hiện.

+ Sự đa dạng trong định nghĩa: Định nghĩa về các bất thường trong đồ thị đa dạng hơn nhiều so với phát hiện ngoại lai truyền thống nhờ khả năng biểu diễn phong phú của đồ thị. Ví dụ: các loại bất thường mới liên quan đến cấu trúc con của đồ thị rất được quan tâm trong nhiều ứng dụng.
+ Thiếu các thước đo khoảng cách/độ tương đồng nội tại: Không có thước đo khoảng cách hoặc độ tương đồng nội tại rõ ràng. Ví dụ, trong các hệ thống máy tính thực tế, với hai chương trình có hàng nghìn sự kiện hệ thống liên quan, việc đo lường khoảng cách/độ tương đồng giữa chúng là một nhiệm vụ khó khăn.
+ Kích thước không gian tìm kiếm: đây là một vấn đề lớn liên quan đến các bất thường phức tạp. Ví dụ như cấu trúc con của đồ thị là không gian tìm kiếm rất lớn. Việc liệt kê tất cả các cấu trúc con mang tính tổ hợp, khiến việc tìm kiếm các bất thường trở thành một việc khó khăn hơn rất nhiều. Không gian tìm kiếm này càng mở rộng hơn khi đồ thị có các thuộc tính, vì các khả năng không chỉ nằm ở cấu trúc đồ thị mà còn ở không gian thuộc tính.

– Do đó, các thuật toán phát hiện bất thường dựa trên đồ thị cần được thiết kế không chỉ để đảm bảo tính hiệu quả mà còn phải đáp ứng khả năng hoạt động nhanh và khả năng mở rộng.

b. Các nhiệm vụ

- Do tính chất đặc thù của nhiệm vụ phát hiện bất thường, các vấn đề còn phát sinh từ các thách thức sau: số lượng và chất lượng nhãn, mất cân bằng lớp và lỗi bất đối xứng, cũng như các bất thường mới.
  + Số lượng và chất lượng nhãn: Vấn đề lớn nhất của việc phát hiện bất thường là dữ liệu thường không có hoặc có rất ít nhãn phân loại. Thông tin về dữ liệu nào là bất thường hoặc bình thường thường không được biết trước. Thông thường, việc thu thập nhãn chính xác từ chuyên gia trong lĩnh vực đòi hỏi nhiều chi phí và thời gian. Hơn nữa, do sự phức tạp của dữ liệu, các nhãn được cung cấp có thể bị nhiễu và thiên lệch. Điều này hạn chế hiệu suất của các thuật toán học máy có giám sát. Ngoài ra, việc thiếu nhãn cũng gây khó khăn trong việc đánh giá các kỹ thuật phát hiện bất thường.
  + Mất cân bằng lớp: Vì các bất thường rất hiếm và chỉ chiếm một phần nhỏ của dữ liệu, nên dữ liệu thường mất cân bằng nghiêm trọng. Hơn nữa, chi phí khi gắn nhãn sai giữa một dữ liệu bình thường và một dữ liệu bất thường có thể thay đổi tùy thuộc vào ứng dụng, và đôi khi rất khó ước lượng trước. Ví dụ, nếu dự đoán nhầm một kẻ gian lận chuyển tiền là người dùng bình thường, điều này có thể gây hại nghiêm trọng đến toàn bộ hệ thống tài chính. Ngược lại, việc dự đoán nhầm một người dùng bình thường là kẻ gian lận có thể làm mất lòng trung thành của khách hàng. Do đó, mất cân bằng lớp ảnh hưởng nghiêm trọng đến các phương pháp dựa trên học máy.
  + Bất thường mới: Trong một số lĩnh vực, chẳng hạn như phát hiện gian lận hoặc phần mềm độc hại, các bất thường do con người tạo ra. Chúng được ngụy trang để trông giống như dữ liệu bình thường nhằm qua mặt hệ thống. Do đó, không chỉ các thuật toán cần thích nghi với dữ liệu thay đổi và mở rộng theo thời gian, mà chúng còn cần phải phát hiện

được các bất thường mới.

c. Model

– Ngoài các vấn đề liên quan đến dữ liệu và nhiệm vụ, việc áp dụng mạng nơ-ron đồ thị (Graph Neural Network - GNN) trực tiếp vào nhiệm vụ phát hiện bất thường cũng gặp nhiều thách thức do những đặc tính đặc thù của mô hình, chẳng hạn như tập trung đồng nhất và dễ tổn thương.

  + Tập trung đồng nhất: Hầu hết các mô hình GNN được thiết kế cho đồ thị đồng nhất, chỉ xem xét một loại nút và cạnh. Trong nhiều ứng dụng thực tế, dữ liệu thường được biểu diễn dưới dạng đồ thị không đồng nhất. Tuy nhiên, các GNN truyền thống coi tất cả các đặc trưng như nhau, tất cả đều được ánh xạ và lan truyền cùng nhau để tạo ra biểu diễn của các nút. Vì vậy, các đặc trưng như tuổi tác, giới tính, giáo dục không liên quan đến vai trò của nút có thể làm giảm hiệu quả phân biệt khi tổng hợp từ hàng xóm. Điều này khiến các GNN truyền thống thất bại khi xử lý các nút có vai trò khác nhau.
  + Dễ tổn thương: Các nghiên cứu lý thuyết gần đây chỉ ra rằng GNN có nhiều hạn chế và dễ bị tấn công khi đồ thị có các nút và cạnh nhiễu. Một thay đổi nhỏ trong đặc trưng của nút có thể gây giảm hiệu suất đáng kể. Ngoài ra, GNN dễ bị ngụy trang bởi những kẻ gian lận có thể phá hoại hệ thống phát hiện gian lận dựa trên GNN.

d. Pipeline

Xây dựng và biến đổi đồ thị:

– Một hệ thống phát hiện bất thường trong thực tế gặp phải một số vấn đề đặc thù liên quan đến dữ liệu. Do đó, cần phải phân tích dữ liệu thô để giải quyết các vấn đề này. Sau đó, đồ thị có thể được xây dựng để nắm bắt các mối tương tác phức tạp và loại bỏ các dữ liệu dư thừa. Dựa trên loại của các thực thể dữ liệu và mối quan hệ, đồ thị có thể được xây dựng dưới dạng

đồ thị đồng nhất hoặc đồ thị không đồng nhất:

+ Đồ thị đồng nhất: Chỉ chứa một loại thực thể và một loại quan hệ duy nhất.
+ Đồ thị không đồng nhất: Chứa nhiều loại thực thể và nhiều loại quan hệ khác nhau.

– Dựa trên tính sẵn có của dấu thời gian, đồ thị có thể được xây dựng dưới dạng đồ thị tĩnh hoặc đồ thị động:
  + Đồ thị tĩnh: Có các nút và cạnh cố định.
  + Đồ thị động: Có các nút và/hoặc cạnh thay đổi theo thời gian.
– Dựa trên tính sẵn có của các thuộc tính của nút và/hoặc cạnh, đồ thị được xây dựng có thể là đồ thị thuần hoặc đồ thị có thuộc tính:
  + Đồ thị thuần: Chỉ chứa thông tin cấu trúc.
  + Đồ thị có thuộc tính: Có thêm thông tin thuộc tính trên các nút và/hoặc cạnh.
– Đối với đồ thị không đồng nhất, việc tổng hợp các hàng xóm một cách đơn giản sẽ không thể nắm bắt được các mối quan hệ ngữ nghĩa và cấu trúc giữa các loại thực thể khác nhau. Để giải quyết vấn đề về tính không đồng nhất của đồ thị, cần thực hiện chuyển đổi đồ thị để biến đồ thị không đồng nhất thành đồ thị đa kênh dựa trên các meta-paths.
  + Meta-path: Là một đường đi kết nối các loại thực thể thông qua một chuỗi các mối quan hệ trên một mạng không đồng nhất. Ví dụ, trong một hệ thống máy tính, meta-path có thể là các sự kiện trong hệ thống như P→P, P→F, và P→I, với mỗi sự kiện định nghĩa một mối quan hệ duy nhất giữa hai thực thể.
  + Đồ thị đa kênh là đồ thị với mỗi kênh được xây dựng thông qua một loại meta-path xác định. Giả sử đồ thị không đồng nhất $\mathcal{G}$ có một tập các meta-path $\mathcal{M} = \{M_1, \dots, M_{|\mathcal{M}|}\}$, đồ thị đa kênh đã được chuyển đổi

như sau:

$$\hat{\mathcal{G}} = \{\mathcal{G}_i | \mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i, A_i, i = 1,2,3, \dots |M|)\}$$

Trong đó:

- $\mathcal{G}_i$: Là một kênh trong đồ thị, được xây dựng từ một meta-path M_i.
- $\mathcal{V}_i$: Tập đỉnh trong kênh i.
- $\mathcal{E}_i$:Các liên kết đồng nhất giữa các đỉnh trong i, được kết nối qua meta-path M_i
- A_i: Ma trận kề tương ứng với kênh i.
- $|M|$**:** Số lượng meta-paths.

Học biểu diễn đồ thị

Sau khi đồ thị được xây dựng và biến đổi, việc học biểu diễn đồ thị được tiến hành để thu được biểu diễn mới thích hợp cho đồ thị. Thông thường, các mạng nơ-ron đồ thị (GNNs - Graph Neural Networks) được xây dựng bằng cách xếp chồng bảy loại phép toán cơ bản, bao gồm: hàm tổng hợp nơ-ron AGG(), hàm ánh xạ tuyến tính $MAP_{linear}()$, hàm ánh xạ phi tuyến tính $MAP_{nonlinear}()$, hàm perceptron nhiều lớp MLP(), phép nối đặc trưng: CONCAT(), phép hợp nhất đặc trưng theo cơ chế chú ý $COMB_{att}()$, hàm đọc kết quả Readout().

- Hàm ánh xạ tuyến tính $MAP_{linear}()$:

$$MAP_{linear}(x) = W_x$$

+ x: vector đặc trưng đầu vào
+ W: Là ma trận trọng số có thể huấn luyện được.
+ Mục đích: Thực hiện phép biến đổi tuyến tính trên vector đầu vào. Ma trận trọng số W sẽ được điều chỉnh trong quá trình huấn luyện để tối ưu hóa mô hình.
+ Ví dụ: Biến đổi vector đặc trưng từ không gian đầu vào sang một không gian biểu diễn khác với kích thước mong muốn.

- Hàm ánh xạ tuyến tính $MAP_{nonlinear}()$:

$$MAP_{linear}(x) = \sigma W_x$$

  + x: vector đặc trưng đầu vào
  + W: Là ma trận trọng số có thể huấn luyện được.
  + Mục đích: Giới thiệu tính phi tuyến vào mô hình, cho phép mạng nơ-ron học được các quan hệ phức tạp và biểu diễn dữ liệu tốt hơn. Nếu không có hàm kích hoạt phi tuyến, mạng sẽ chỉ là một tổ hợp tuyến tính đơn giản, làm giảm khả năng học của mô hình.
  + Hàm kích hoạt:
    - ReLU: $\sigma(x) = \max(0, x)$.
    - Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$
- Hàm perceptron nhiều lớp MLP():

$$MLP(x) = \sigma(W^K \dots \sigma(W^1 x))$$

  + x: vector đặc trưng đầu vào
  + W^1, W^2, W^k: Là các ma trận trọng số của từng lớp trong mạng perceptron nhiều lớp.
  + Mục đích:
    - MLP là một mạng nơ-ron nhiều lớp bao gồm nhiều lớp biến đổi tuyến tính (Wx) kết hợp với hàm kích hoạt phi tuyến giữa các lớp.
    - Cho phép mạng học được các đặc trưng phức tạp thông qua việc áp dụng nhiều lần các phép biến đổi và phi tuyến hóa.
- Phép nối đặc trưng:

$$CONCAT(x_1, \dots x_n) = [x_1, \dots x_n]$$

  + $x_1, \dots x_n$: vector đặc trưng đầu vào.
  + Mục đích: nối các vector đặc trưng đầu vào thành 1 vector duy nhất.
- Phép hợp nhất đặc trưng theo cơ chế chú ý $COMB_{att}()$:

$$COMB_{att}(x_1, \dots x_n) = \sum_{i=1}^{n} softmax(x_i)x_i$$

$$softmax(x_i) = \frac{\exp\left(MAP(x_i)\right)}{\sum_{j=1}^{n} \exp\left(MAP(x_j)\right)}$$

+ MAP(): hàm ánh xạ, có thể là tuyến tính hoặc phi tuyến tính.
+ exp(): hàm mũ.
+ Mục đích: kết hợp các đặc trưng $x_1, \dots x_n$ bằng cách gán trọng số (được tính bằng softmax) cho từng đặc trưng. Softmax đảm bảo rằng tổng các trọng số bằng 1, giúp mô hình tập trung vào những đặc trưng quan trọng hơn.

– Hàm tổng hợp nơ ron AGG(): dựa trên cấp độ của đối tượng để tổng hợp, có thể chia thành 3 loại:

+ Tổng hợp nút (AGG_{node}):

$$h_v{}^{(i)(k)} = AGG_{node}(h_v{}^{(i)(k-1)}, \{h_u{}^{(i)(k-1)}\}_{u \in N_v{}^i})$$

  - i: Chỉ số của meta-path (quan hệ).
  - k: Chỉ số lớp.
  - $h_v{}^{(i)(k)}$: Vector đặc trưng của nút v cho quan hệ M_i tại lớp k.
  - $N_v{}^i$: Tập hợp các nút lân cận của v dưới quan hệ M_i.
  - M_i: meta-path thứ i.
  - Mục đích: Tổng hợp thông tin từ các nút lân cận của một nút cụ thể.

+ Tổng hợp lớp (AGG_{layer}):

$$l_v{}^{(i)(k)} = AGG_{layer}(l_v{}^{(i)(k-1)}, h_v{}^{(i)(k)})$$

  - $l_v{}^{(i)(k-1)}$: Biểu diễn tổng hợp từ các lân cận (k−1) bước cho nút v.

- $l_v^{(i)(k)}$: Biểu diễn đặc trưng của nút v sau khi tổng hợp thông tin từ k-hop lân cận dưới mối quan hệ M_i.
- $h_v^{(i)(k)}$: Đặc trưng của nút v tại lớp k.
- K-hops: Khi k tăng lên, GNN thu thập thông tin toàn cục nhiều hơn.
- Mục đích**:** Tổng hợp thông tin từ các lớp khác nhau (các "hops" khác nhau trong đồ thị).

+ Tổng hợp lớp (AGG_{path}):

$$p_v^{(i)} = l_v^{(i)(k)}$$

$$p_v = AGG_{path}(p_v^{(1)}, \dots p_v^{(|M|)})$$

- $p_v^{(i)}$: Biểu diễn cuối cùng của nút v cho quan hệ M_i.
- AGG_{path}: Tổng hợp thông tin từ nhiều meta-paths để tạo thành biểu diễn cuối cùng của nút.
- Mục đích**:** Tổng hợp thông tin từ các mối quan hệ khác nhau (meta-paths).

– Biểu diễn nút cuối cùng được tổng hợp từ các meta-path.

$$h_v^{(final)} = p_v$$

+ $h_v^{(final)}$: Biểu diễn cuối cùng của nút v.
+ p_v: Biểu diễn tổng hợp từ các meta-path khác nhau liên quan đến nút v.

– Biểu diễn đồ thị thông qua Readout():

$$G = Readout(h_{v_1}^{(final)}, \dots h_{v_v}^{(final)})$$

+ g: Biểu diễn tổng hợp cho toàn bộ đồ thị.
+ $h_{v_1}^{(final)}, \dots h_{v_v}^{(final)}$: Biểu diễn cuối cùng của tất cả các nút trong đồ thị.

+ Readout(): Hàm tổng hợp tất cả các biểu diễn nút thành một biểu diễn chung cho đồ thị. Có thể là hàm Sum, Mean, Max.

– **Các cấp độ biểu diễn đồ thị:**

+ Cấp độ nút: Biểu diễn đặc trưng cho từng nút riêng lẻ. Đây là cấp độ cơ bản nhất, thường được sử dụng trong các bài toán như phân loại nút hoặc dự đoán thuộc tính nút.
+ Cấp độ cạnh: Biểu diễn đặc trưng cho các mối quan hệ giữa các nút (các cạnh). Thường dùng trong các bài toán như dự đoán liên kết.
+ Cấp độ đồ thị: Biểu diễn toàn bộ đồ thị, thường được sử dụng trong các bài toán như phân loại đồ thị hoặc so sánh đồ thị. Biểu diễn này được tạo bằng cách tổng hợp tất cả các nút hoặc cạnh trong đồ thị thông qua hàm Readout().

e. Dự đoán

– Sau khi biểu diễn đồ thị được học, chúng sẽ được đưa vào giai đoạn dự đoán. Tùy thuộc vào nhiệm vụ và nhãn mục tiêu, có hai loại dự đoán: phân loại và so khớp

+ Dự đoán dựa trên phân loại: Sử dụng khi có đủ dữ liệu bất thường đã gắn nhãn. Bộ phân loại được huấn luyện để xác định liệu một đồ thị có bất thường hay không.
+ Dự đoán dựa trên so khớp: Sử dụng khi dữ liệu bất thường rất ít hoặc không có. So sánh biểu diễn của mẫu mới với mẫu bất thường hoặc bình thường đã học:

    - Nếu giống với mẫu bất thường thì hệ thống sẽ cảnh báo.
    - Nếu không giống bất kỳ mẫu bình thường nào thì hệ thống sẽ gửi đi cảnh báo.

2.1.2. Graph Attention Network (GAT)

- Graph Attention Network (GAT) là một kiến trúc mạng nơ-ron mới, hoạt động trên dữ liệu có cấu trúc dạng đồ thị.
- Trong **Graph Convolutional Networks (GCNs)**, đối với một nút đích i, tầm quan trọng của nút láng giềng j được xác định bởi trọng số của cạnh Aij giữa chúng (được chuẩn hóa theo bậc của các nút). Tuy nhiên, trong thực tế, đồ thị đầu vào có thể chứa nhiều nhiễu, và các trọng số cạnh này không thể phản ánh đúng mức độ quan trọng thực sự giữa hai nút. Do đó, một cách tiếp cận có cơ sở hơn là tự động học được tầm quan trọng của từng nút láng giềng.
- GAT ra đời dựa trên ý tưởng này và cố gắng học được tầm quan trọng của mỗi nút láng giềng thông qua cơ chế Attention. Cơ chế này đã được sử dụng rộng rãi trong nhiều tác vụ như hiểu ngôn ngữ tự nhiên (ví dụ, dịch máy, trả lời câu hỏi) và thị giác máy tính (ví dụ, trả lời câu hỏi dựa trên hình ảnh và chú thích hình ảnh). Phần sau đây sẽ giới thiệu cách áp dụng Attention trong mạng nơ-ron đồ thị.

2.1.2.1. Graph Attention Layer:

- GAT được xây dựng từ các lớp attention đơn giản mà có thể xếp chồng lên nhau để tạo ra mạng attention đồ thị.
- Giả sử đầu vào của lớp là một tập hợp các đặc trưng của các nút, h = $\{\overrightarrow{h_1}, \overrightarrow{h_2}, \dots, \overrightarrow{h_N}\}$ trong đó $\overrightarrow{h_\iota}\ \varepsilon\ R^F$ và N là số nút trong đồ thị. F là số lượng đặc trưng của mỗi nút. Đầu ra của lớp này là một tập hợp đặc trưng mới cho các nút, có thể có kích thước khác $h' = \{\overrightarrow{h'_1}, \overrightarrow{h'_2}, \dots, \overrightarrow{h'_N}\}$, trong đó $\overrightarrow{h'_\iota}\ \varepsilon\ R^{F'}$.
- Để lớp này đủ khả năng chuyển đổi đặc trưng đầu vào thành đặc trưng cấp cao hơn, cần có ít nhất một phép biến đổi tuyến tính có thể học được. Do đó, bước đầu tiên là áp dụng một phép biến đổi tuyến tính, được tham số hóa bởi ma trận trọng số $W \in R^{F' \times F}$, lên mỗi nút. Sau đó, self-attention (cơ chế tự chú ý) được áp dụng trên các nút – một cơ chế attention chia sẻ a: $R^{F'} \times R^{F'} \to R$ sẽ

tính các hệ số attention giữa các nút:

$$e_{ij} = a\left(W\overrightarrow{h_i}, W\overrightarrow{h_j}\right)$$

- Hệ số này biểu thị tầm quan trọng của đặc trưng của nút j đối với nút i. Trong dạng tổng quát nhất, mô hình cho phép mỗi nút chú ý đến tất cả các nút khác, bỏ qua thông tin cấu trúc của đồ thị. Tuy nhiên, để đưa cấu trúc đồ thị vào, GAT chỉ tính e_{ij} cho các nút j $\in N_i$, trong đó N_i là tập hợp các nút lân cận của i (bao gồm cả i nếu cần thiết). Để các hệ số này có thể so sánh được giữa các nút, chúng ta chuẩn hóa chúng qua hàm softmax:

$$a_{ij} = softmax_j(e_{ij}) = \frac{\exp\left(e_{ij}\right)}{\sum_{k\in\mathbb{N}(i)} \exp\left(e_{ik}\right)}$$

- Trong các thử nghiệm, cơ chế attention a được triển khai bằng một mạng nơ-ron feedforward một lớp, tham số hóa bởi một vector trọng số $\vec{a} \in R^{2F'}$, và sử dụng hàm kích hoạt LeakyReLU (với hệ số góc âm là 0.2) . Công thức chi tiết cho hệ số attention a_{ij} là:

$$a_{ij} = \frac{\exp(LeakyReLU(\vec{a}^T[W\overrightarrow{h_i}\|W\overrightarrow{h_j}]))}{\sum_{k\in\mathbb{N}(i)} \exp\left(LeakyReLU(\vec{a}^T[W\overrightarrow{h_i}\|W\overrightarrow{h_k}])\right)}$$

  - $.^T$: đại diện cho chuyển vị
  - || : là phép nối 2 vector
  - a_{ij} : hệ số attention. Nó thể hiện mức độ "quan trọng" của đỉnh j đối với đỉnh i khi lan truyền thông tin qua đồ thị.
  - $\overrightarrow{h_i}$: Là vector đặc trưng của đỉnh i.
  - W**:** Là ma trận trọng số học được, dùng để chuyển đổi vector đặc trưng của các đỉnh sang không gian mới.
  - $\vec{a}^T$: Là vector trọng số cho cơ chế attention dùng để tính toán độ quan trọng
  - $LeakyReLU$: Là hàm kích hoạt phi tuyến được áp dụng lên kết quả của

phép nhân $\vec{a}^T[W\overrightarrow{h_i}||W\overrightarrow{h_j}]$. Hàm này giúp mô hình học được sự phức tạp trong mối quan hệ giữa các đỉnh.

- exp: Hàm mũ được dùng để đảm bảo hệ số attention luôn là số dương và giúp chuẩn hóa.
- N(i): Tập lân cận của đỉnh i, bao gồm các đỉnh kết nối với i.
- $\sum_{k\in\mathbb{N}(i)}$ ... : Tổng các hệ số chú ý trên tất cả các đỉnh lân cận kkk của đỉnh i. Việc này giúp chuẩn hóa giá trị a_{ij} thành phân phối xác suất.

- Khi đã có các hệ số attention được chuẩn hóa, chúng ta tính đặc trưng đầu ra của nút i bằng cách lấy tổ hợp tuyến tính của các đặc trưng của các nút lân cận, có trọng số theo hệ số attention (và có thể áp dụng thêm một hàm kích hoạt phi tuyến σ)

$$\vec{h}_i' = \sigma\left(\sum_{j\in N_i} a_{ij} w\vec{h}_j\right)$$

- N(i): Tập lân cận của đỉnh i, bao gồm các đỉnh kết nối với i
- $\overrightarrow{h'_j}$: Là vector đặc trưng mới của đỉnh i sau khi áp dụng lớp attenttion trong GAT.
- a_{ij} : hệ số attention
- $W\overrightarrow{h_j}$: Là vector đặc trưng của đỉnh j sau khi được biến đổi qua ma trận trọng số W
- σ: hàm kích hoạt phi tuyến, ví dụ ReLu hoặc Sigmoid. Hàm này giúp mạng học được các biểu diễn phức tạp hơn.

2.1.2.2. Multi-head Attention:

- GAT có một bí mật: nó không chỉ nhìn các nút hàng xóm bằng một cách duy nhất, mà nhìn nhiều cách khác nhau cùng lúc. Đây gọi là multi-head attention. Việc này nhằm mục đích để quá trình học của GAT trở nên ổn định. Cụ thể, có K cơ chế attention độc lập hay nói cách khác như là "cách nhìn". Mỗi "cách

nhìn" sẽ tính toán hệ số attention a_{ij} riêng, rồi tạo ra một phần thông tin mới. Sau đó, các phần này được ghép lại với nhau, giống như bạn tham khảo nhiều ý kiến để đưa ra quyết định cuối cùng:

$$\vec{h}_i' = ||_{k=1}^{K} \sigma \left(\sum_{j \in N_i} {a_{ij}}^k W^k \vec{h}_j \right)$$

- || : là phép nối 2 vector, biểu thị việc ghép các vector đầu ra của nhiều đầu attention (K là số lượng đầu). Mỗi đầu attention học cách tập trung vào các neighbor j ε N_i khác nhau của node i.
- ${a_{ij}}^k$: hệ số attention được tính toán bởi cơ chế attention thứ k
- W^k: là ma trận trọng số của phép biến đổi tuyến tính của đầu vào tương ứng.
- $\overrightarrow{{h'}_i}$: Là vector đặc trưng mới của đỉnh i sau khi áp dụng lớp attenttion trong GAT.

- Đặc biệt, nếu chúng ta thực hiện multi-head attention trên lớp (dự đoán) cuối cùng của mạng, việc ghép nối khong còn hợp lý nữa, thay vào đó, chúng ta lấy trung bình các biểu diễn:

$$\vec{h}_i' = \sigma \left(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in N_i} {a_{ij}}^k W^k \vec{h}_j \right)$$

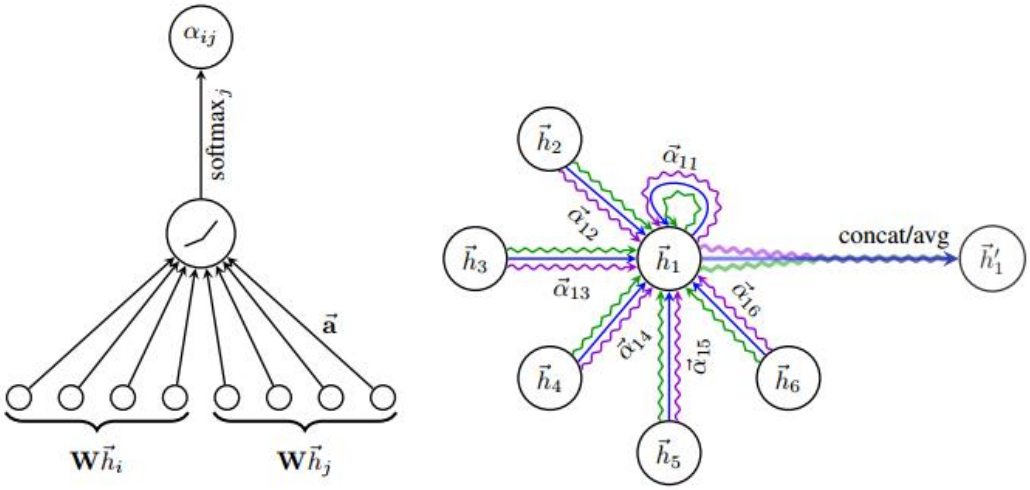


Figure 1 Graph Attention Network

- Hình bên trái mô tả cơ chế attention:
  - Các nút tròn: Đại diện cho các node (đỉnh) trong đồ thị. Mỗi node chứa thông tin đặc trưng của một đối tượng.
  - Các cạnh: Biểu diễn các mối quan hệ giữa các node.
  - $W\overrightarrow{h_j}$: Là vector đặc trưng của đỉnh j sau khi được biến đổi qua ma trận trọng số W
  - Hàm softmax: Chuẩn hóa thành trọng số attention a_{ij} đảm bảo tổng các trọng số attention từ node i đến tất cả các neighbor bằng 1

  ➔ Đây là cách mà GAT tính trọng số a_{ij} giữa các node i và j.
- Hình bên phải mô tả quá trình cập nhập đặc trưng của một node trong GAT:
  - Phép toán concat/avg: Sau khi tính toán được các trọng số attention, ta sẽ nhân các trọng số này với đặc trưng của các node láng giềng tương ứng, sau đó thực hiện phép nối (concatenate) các kết quả lại và tính trung bình. Điều này giúp ta thu được một vector đặc trưng mới cho node hiện tại.

CHƯƠNG 3: CÁC NGHIÊN CỨU LIÊN QUAN

3.1. Giới thiệu các kỹ thuật giải quyết chủ đề của các tác giả liên quan

Vessel Anomaly Detection Using Graph Attention Networks: Bài báo tập trung vào việc phát hiện hành vi bất thường của tàu, một yếu tố quan trọng để đảm bảo an toàn hàng hải và nâng cao hiệu quả giám sát vùng biển dựa trên mạng đồ thị chú ý (GAT).

Các kỹ thuật liên quan:

- Cửa sổ trượt: Kỹ thuật này được sử dụng để chia dữ liệu quỹ đạo tàu thành các đoạn nhỏ hơn có độ dài cố định, giúp cho việc phân tích và xử lý dữ liệu hiệu quả hơn. Cửa sổ trượt cho phép xử lý dữ liệu theo từng đoạn, tạo điều kiện cho việc phát hiện bất thường trong thời gian thực.
- Trích xuất đặc trưng: Các đặc trưng của quỹ đạo tàu, bao gồm vị trí (kinh độ, vĩ độ), tốc độ và hướng đi, được trích xuất từ dữ liệu AIS.
- Mạng đồ thị chú ý (GAT): được sử dụng để phân tích sự phụ thuộc giữa các điểm quỹ đạo tàu. Khi sự phụ thuộc giữa một điểm quỹ đạo nhất định thay đổi (vượt quá ngưỡng đã đặt), quỹ đạo tàu được đánh giá là một đoạn quỹ đạo bất thường, từ đó xác định hành vi bất thường của tàu.
- Đơn vị GRU (Gate Recurrent Unit): được sử dụng để nắm bắt sự phụ thuộc dài hạn giữa các đặc trưng, từ đó cải thiện hiệu quả và hiệu suất thời gian thực của việc phát hiện hành vi bất thường của tàu.
- Mô hình dự đoán: được sử dụng để dự đoán giá trị của các đặc trưng quỹ đạo tàu tại dấu thời gian tiếp theo. Bất kỳ sai lệch đáng kể nào so với giá trị dự đoán đều có thể được gắn cờ là bất thường.
- Mô hình tái tạo : được sử dụng để học phân phối tiềm ẩn của dữ liệu. Nó cố gắng tái tạo dữ liệu đầu vào từ một biểu diễn nén. Các điểm dữ liệu khó tái tạo được coi là bất thường
- Phương pháp POT (Peak Over Threshold): được sử dụng để xác định

ngưỡng động, tự động cập nhật ngưỡng dựa trên dữ liệu bình thường, giúp cải thiện hiệu quả của việc phát hiện bất thường quỹ đạo tàu.

3.2. Tổ chức theo thời gian hoặc theo nhóm kỹ thuật

- Kỹ thuật xử lý dữ liệu đầu vào:
  + Cửa sổ trượt.
  + Trích xuất đặc trưng.
- Kỹ thuật mô hình hóa và phân tích:
  + Mạng đồ thị chú ý (GAT).
  + Đơn vị GRU (Gate Recurrent Unit).
  + Mô hình dự đoán.
  + Mô hình tái tạo.
- Kỹ thuật xác định Ngưỡng:
  + Phương pháp POT (Peak Over Threshold).

3.3. Đánh giá các kỹ thuật đã giải quyết chủ đề này: điểm yếu, điểm mạnh

- Cửa sổ trượt:
  + Điểm mạnh: Việc sử dụng cửa sổ trượt giúp mô hình nắm bắt được sự thay đổi của hành vi tàu theo thời gian. Điều này đặc biệt hữu ích trong việc phát hiện các bất thường có tính chất ngắn hạn.
  + Điểm yếu: Việc xác định kích thước cửa sổ trượt tối ưu là một vấn đề quan trọng. Nếu cửa sổ quá nhỏ, mô hình có thể bỏ sót các mẫu bất thường kéo dài. Ngược lại, cửa sổ quá lớn có thể làm giảm độ nhạy của mô hình với các thay đổi đột ngột.
- Trích xuất đặc trưng: Các đặc trưng của quỹ đạo tàu, bao gồm vị trí (kinh độ, vĩ độ), tốc độ và hướng đi, được trích xuất từ dữ liệu AIS.
  + Điểm mạnh: Việc sử dụng các đặc trưng vị trí, tốc độ và hướng đi giúp mô hình nắm bắt được các khía cạnh quan trọng của hành vi tàu
  + Điểm yếu: Bài báo sử dụng một số đặc trưng cơ bản. Việc bổ sung thêm

các đặc trưng khác, chẳng hạn như loại tàu, tải trọng, điều kiện thời tiết, có thể giúp nâng cao hiệu quả phát hiện. Tuy nhiên, có thể làm tăng độ phức tạp của mô hình.

- Mạng đồ thị chú ý (GAT): được sử dụng để phân tích sự phụ thuộc giữa các điểm quỹ đạo tàu. Khi sự phụ thuộc giữa một điểm quỹ đạo nhất định thay đổi (vượt quá ngưỡng đã đặt), quỹ đạo tàu được đánh giá là một đoạn quỹ đạo bất thường, từ đó xác định hành vi bất thường của tàu.
  + Điểm mạnh: GAT cho phép mô hình học hỏi sự phụ thuộc giữa các điểm dữ liệu trong chuỗi thời gian một cách hiệu quả. Điều này giúp GAT vượt trội hơn so với các phương pháp truyền thống, vốn thường bỏ qua sự phụ thuộc này.
  + Điểm yếu: Độ phức tạp tính toán của GAT có thể là một vấn đề lớn khi xử lý các đồ thị có kích thước lớn.
- Đơn vị GRU (Gate Recurrent Unit): được sử dụng để nắm bắt sự phụ thuộc dài hạn giữa các đặc trưng, từ đó cải thiện hiệu quả và hiệu suất thời gian thực của việc phát hiện hành vi bất thường của tàu.
  + Điểm mạnh: có khả năng ghi nhớ thông tin từ các bước thời gian trước đó, giúp mô hình nắm bắt được các mẫu bất thường phức tạp.
  + Điểm yếu: có thể yêu cầu nhiều dữ liệu huấn luyện và thời gian huấn luyện dài.
- Mô hình dự đoán: được sử dụng để dự đoán giá trị của các đặc trưng quỹ đạo tàu tại dấu thời gian tiếp theo. Bất kỳ sai lệch đáng kể nào so với giá trị dự đoán đều có thể được gắn cờ là bất thường.
  + Điểm mạnh: dự đoán giá trị dựa trên dữ liệu bình thường, hiệu quả với chuỗi thời gian có tính chu kỳ.
  + Điểm yếu: dễ bị ảnh hưởng bởi nhiễu, hạn chế trong dự đoán dài hạn.
- Mô hình tái tạo : được sử dụng để học phân phối tiềm ẩn của dữ liệu. Mô

hình cố gắng tái tạo dữ liệu đầu vào từ một biểu diễn nén. Các điểm dữ liệu khó tái tạo được coi là bất thường

+ Điểm mạnh: Nắm bắt phân phối dữ liệu tổng thể, mạnh mẽ với dữ liệu nhiễu.
+ Điểm yếu: Có thể bỏ qua các bất thường đột ngột.

– Phương pháp POT (Peak Over Threshold): được sử dụng để xác định ngưỡng động, tự động cập nhật ngưỡng dựa trên dữ liệu bình thường, giúp cải thiện hiệu quả của việc phát hiện bất thường quỹ đạo tàu.
  + Điểm mạnh: cho phép cập nhật ngưỡng một cách linh hoạt, dựa trên sự thay đổi của dữ liệu. Điều này giúp mô hình thích nghi với các điều kiện môi trường khác nhau.
  + Điểm yếu: Việc lựa chọn ngưỡng ban đầu cho POT có thể ảnh hưởng đến hiệu quả phát hiện. Tác giả sử dụng ngưỡng dựa trên kinh nghiệm, tuy nhiên, việc xác định ngưỡng này có thể phụ thuộc vào từng khu vực biển cụ thể.

CHƯƠNG 4: THỰC NGHIỆM

4.1. Giới thiệu dataset thực nghiệm

Dataset:https://www.kaggle.com/datasets/vipin20/transaction-data?resource=download

4.1.1. Thông tin dataset thực nghiệm

Dataset được tạo ra để mô phỏng lại các giao dịch của một trang web thương mại e-commerce, bao gồm có 1083818 dòng 8 cột:

STT	Features	Type	Description
1	UserId	Float64	Mã người dùng của giao dịch
2	TransactionId	Int64	Mã giao dịch
3	TransactionTime	Datetime	Thời gian giao dịch
4	ItemCode	Float64	Mã sản phẩm giao dịch
5	ItemDescription	String	Mô tả sản phẩm
6	NumberOfItemPurchased	Int64	Số sản phẩm trong giao dịch
7	CostPerItem	Float64	Giá trị của một sản phẩm
8	Country	String	Địa điểm nơi diễn ra giao dịch

```
df = pd.read_csv('transaction_data.csv',na_values=(" ","?","_","-1"), encoding="ISO-8859-1")

df
```

	UserId	TransactionId	TransactionTime	ItemCode	ItemDescription	NumberOfItemsPurchased	CostPerItem	Country
0	278166.0	6355745	Sat Feb 02 12:50:00 IST 2019	465549.0	FAMILY ALBUM WHITE PICTURE FRAME	6	11.73	United Kingdom
1	337701.0	6283376	Wed Dec 26 09:06:00 IST 2018	482370.0	LONDON BUS COFFEE MUG	3	3.52	United Kingdom
2	267099.0	6385599	Fri Feb 15 09:45:00 IST 2019	490728.0	SET 12 COLOUR PENCILS DOLLY GIRL	72	0.90	France
3	380478.0	6044973	Fri Jun 22 07:14:00 IST 2018	459186.0	UNION JACK FLAG LUGGAGE TAG	3	1.73	United Kingdom
4	NaN	6143225	Mon Sep 10 11:58:00 IST 2018	1733592.0	WASHROOM METAL SIGN	3	3.40	United Kingdom
...	...	...	...	...	...	...	...	...
1083813	313131.0	6040298	Mon Jun 18 09:18:00 IST 2018	437976.0	DENIM PATCH PURSE PINK BUTTERFLY	30	2.28	EIRE
1083814	295743.0	6387117	Sat Feb 16 09:14:00 IST 2019	484113.0	RECYCLED ACAPULCO MAT PINK	6	11.39	United Kingdom
1083815	NaN	6361817	Tue Feb 05 05:26:00 IST 2019	497595.0	DOILY THANK YOU CARD	15	1.15	United Kingdom
1083816	324765.0	5945500	Fri Mar 23 06:26:00 IST 2018	470883.0	REGENCY CAKESTAND 3 TIER	48	15.12	United Kingdom
1083817	351645.0	6118145	Tue Aug 21 08:21:00 IST 2018	471849.0	HEART OF WICKER SMALL	9	2.28	United Kingdom

1083818 rows × 8 columns

Figure 2. Dataset

Mục tiêu của việc sử dụng dataset này nhằm tìm ra được mối quan hệ giữa người mua (user) và sản phẩm (Item) bằng cách sử Graph Attention Network để có thể tìm thấy được các điểm dị dường thông qua việc cho model học các hành vi bình thường khi một người dùng (user) mua một sản phẩm (item), từ đó phát hiện ra và đánh dấu các giao dịch (transaction) khác thường nhằm để phân tích sâu hơn.

4.1.2. Biểu diễn dưới dạng Graph

Dưới dạng Graph 2D:

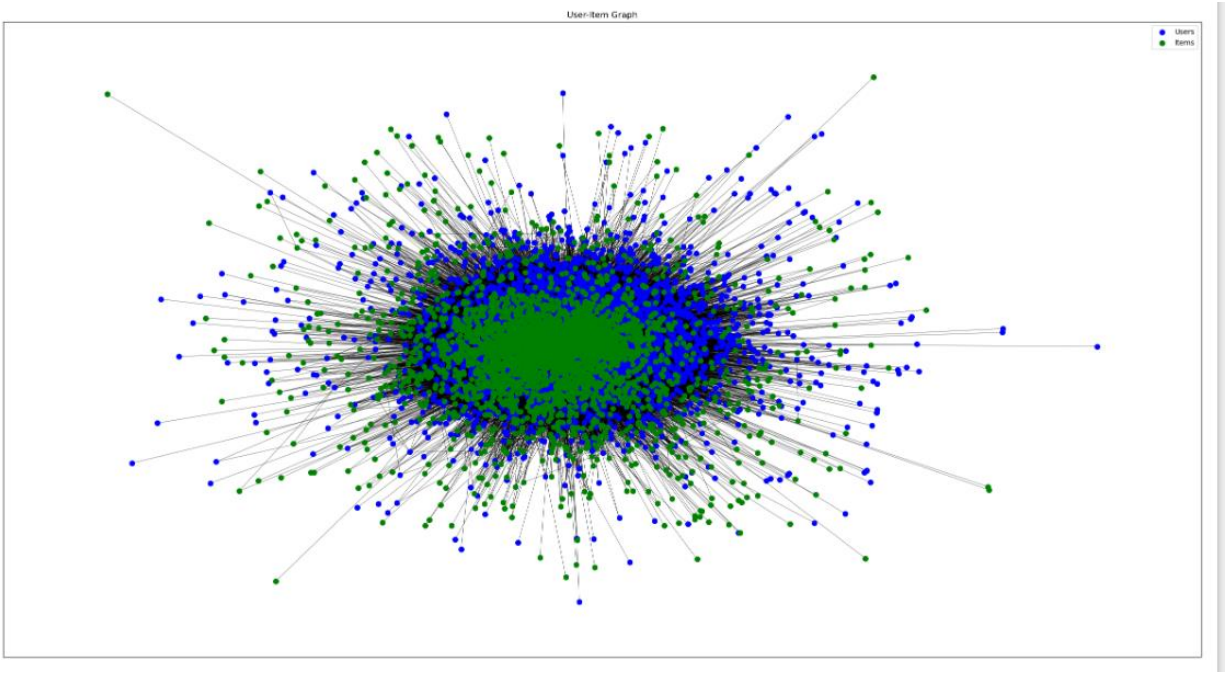


Figure 3. Biểu Graph User-Item 2D

Dưới dạng Graph 3D:

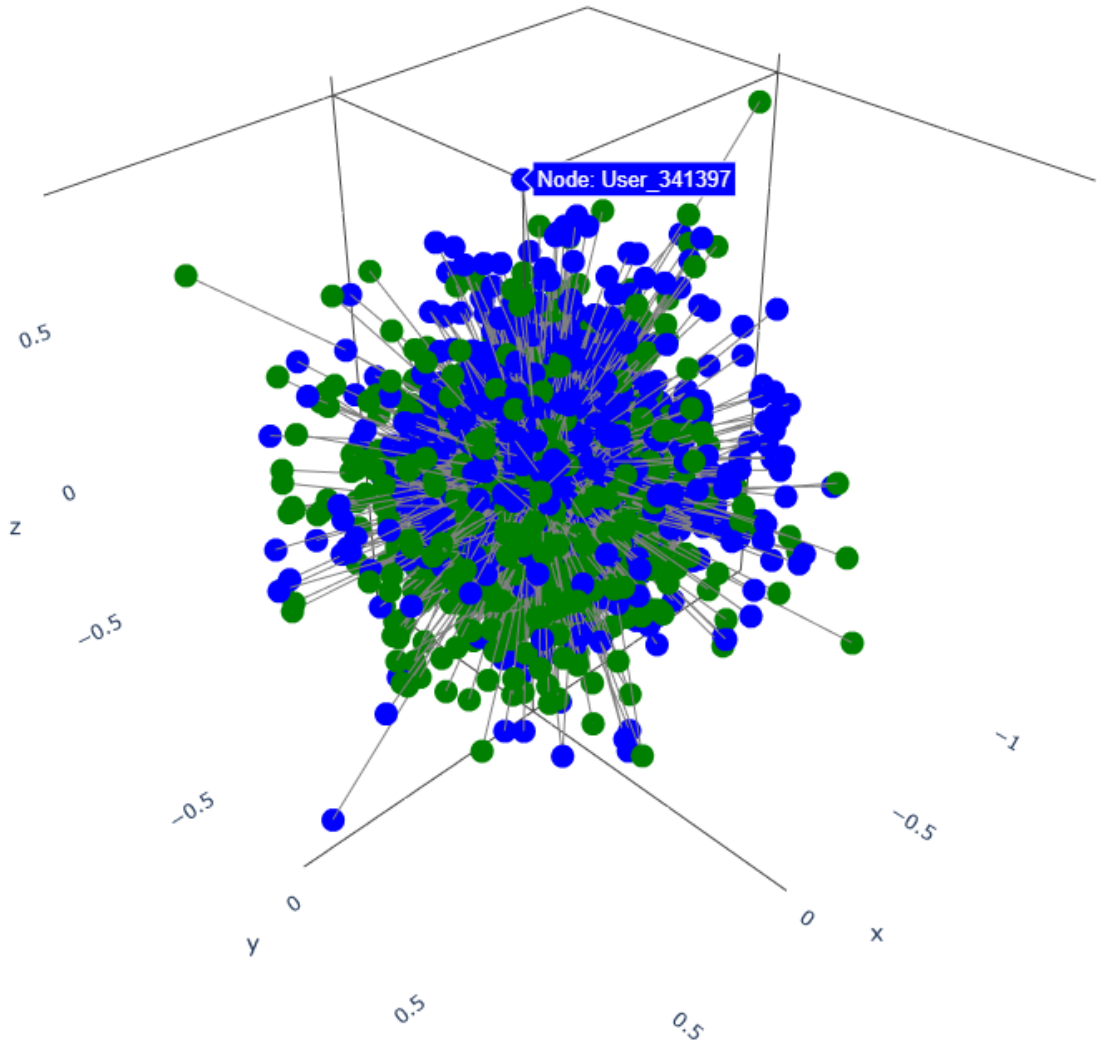


Figure 4. Biểu Graph User-Item 3D

Trong đó:

- User node: Đại diện bởi màu xanh dương, bao gồm có 4362 user trong dataset, node chỉ chứa đặc trưng để phân biệt là user và item là type.
- Item node: Đại diện bởi màu xanh lá, bao gồm có 3206 item trong dataset, node item chứa đặc trưng để phân biệt giữa user và item là type, ngoài ra cũng chữa Description để mô tả sản phẩm đó là gì.

- Edge (cạnh): Tồn tại giữa User và Item và có tổng cộng 260655 cạnh, với mỗi đại diện cho transaction chứa thông tin: quantity (số lượng sản phẩm), unit_price (giá của 1 sản phẩm), transaction_status (tình trạng của giao dịch), invoice_date (thời gian diễn ra giao dịch), country (nơi diễn ra giao dịch).

[33]:
```
node_id = "User_287112"
if node_id in G:
    node_attributes = G.nodes[node_id]
    print(f"Attributes of node {node_id}:")
    for attr, value in node_attributes.items():
        print(f" - {attr}: {value}")
else:
    print(f"Node {node_id} does not exist in the graph.")
```

```
Attributes of node User_287112:
 - type: user
```

Figure 5. Kiểm tra một node

[34]:
```
node1 = "User_362103"
node2 = "Item_1528779.0"

if G.has_edge(node1, node2):
    edge_attributes = G[node1][node2]
    print(f"Attributes of edge between {node1} and {node2}:")
    for attr, value in edge_attributes.items():
        print(f" - {attr}: {value}")
else:
    print(f"No edge exists between {node1} and {node2}.")
```

```
Attributes of edge between User_362103 and Item_1528779.0:
 - quantity: 12
 - unit_price: 4.08
 - transaction_status: 0
 - invoice_date: 2018-04-06 06:15:00
 - country: United Kingdom
```

Figure 6. Kiểm tra một cạnh giữa User-Item

4.2. Vẽ sơ đồ thiết kế thực nghiệm (kiến trúc hệ thống, hoặc lưu đồ giải thuật thực hiện)

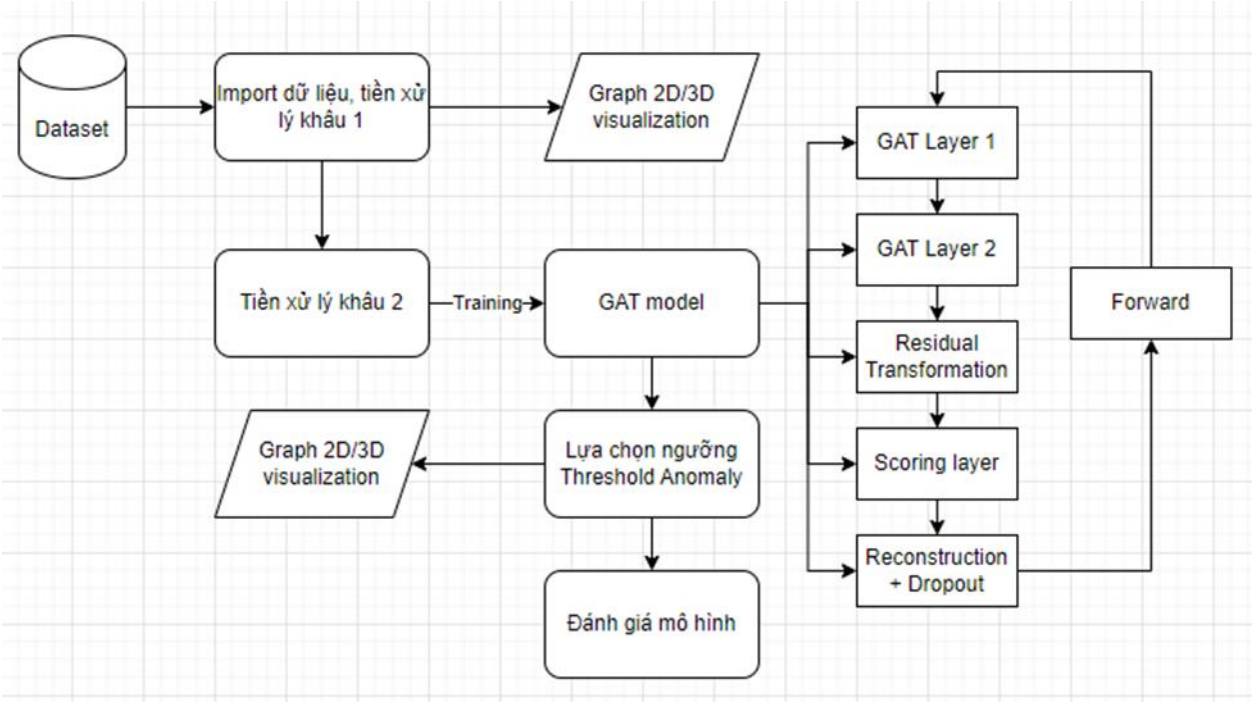


Figure 7. Sơ đồ thiết kế thực nghiệm

4.2.1. Nhập dữ liệu và tiền xử lý khâu 1

- Nhập dữ liệu dataset:

```
[2]: df = pd.read_csv('transaction_data.csv',na_values=(" ","?","_","-1"), encoding="ISO-8859-1")

     df
```

[2]:

	UserId	TransactionId	TransactionTime	ItemCode	ItemDescription	NumberOfItemsPurchased	CostPerItem	Country
0	278166.0	6355745	Sat Feb 02 12:50:00 IST 2019	465549.0	FAMILY ALBUM WHITE PICTURE FRAME	6	11.73	United Kingdom
1	337701.0	6283376	Wed Dec 26 09:06:00 IST 2018	482370.0	LONDON BUS COFFEE MUG	3	3.52	United Kingdom
2	267099.0	6385599	Fri Feb 15 09:45:00 IST 2019	490728.0	SET 12 COLOUR PENCILS DOLLY GIRL	72	0.90	France
3	380478.0	6044973	Fri Jun 22 07:14:00 IST 2018	459186.0	UNION JACK FLAG LUGGAGE TAG	3	1.73	United Kingdom
4	NaN	6143225	Mon Sep 10 11:58:00 IST 2018	1733592.0	WASHROOM METAL SIGN	3	3.40	United Kingdom
...	...	...	...	...	...	...	...	...
1083813	313131.0	6040298	Mon Jun 18 09:18:00 IST 2018	437976.0	DENIM PATCH PURSE PINK BUTTERFLY	30	2.28	EIRE
1083814	295743.0	6387117	Sat Feb 16 09:14:00 IST 2019	484113.0	RECYCLED ACAPULCO MAT PINK	6	11.39	United Kingdom
1083815	NaN	6361817	Tue Feb 05 05:26:00 IST 2019	497595.0	DOILY THANK YOU CARD	15	1.15	United Kingdom
1083816	324765.0	5945500	Fri Mar 23 06:26:00 IST 2018	470883.0	REGENCY CAKESTAND 3 TIER	48	15.12	United Kingdom
1083817	351645.0	6118145	Tue Aug 21 08:21:00 IST 2018	471849.0	HEART OF WICKER SMALL	9	2.28	United Kingdom

Figure 8. Nhập dữ liệu

Tiền xử lý khâu 1:

- Đổi tên các thuộc tính (features) cho dễ nhận dạng: Đổi tên sẽ giúp cho việc dễ nhận dạng hơn và sử dụng cho các tác vụ sau trong giải thuật.

```
# Map tên cho phù hợp
column_mappings = {
    'UserId': 'CustomerID',
    'TransactionId': 'InvoiceNo',
    'TransactionTime': 'InvoiceDate',
    'ItemCode': 'StockCode',
    'ItemDescription': 'Description',
    'NumberOfItemsPurchased': 'Quantity',
    'CostPerItem': 'UnitPrice',
    'Country': 'Country'
}

# Đổi tên
df = df.rename(columns=column_mappings)
# Sort theo ngày
df = df.sort_values(by='InvoiceDate')

df
```

	CustomerID	InvoiceNo	InvoiceDate	StockCode	Description	Quantity	UnitPrice	Country
432402	362103.0	5960185	Fri Apr 06 06:15:00 IST 2018	1528779.0	PINK PILLAR CANDLE SILVER FLOCK	3	4.08	United Kingdom
446762	362103.0	5960185	Fri Apr 06 06:15:00 IST 2018	478086.0	PHOTO FRAME CORNICE	12	4.08	United Kingdom
464683	362103.0	5960185	Fri Apr 06 06:15:00 IST 2018	1528779.0	IVORY PILLAR CANDLE GOLD FLOCK	3	4.08	United Kingdom
400290	362103.0	5960185	Fri Apr 06 06:15:00 IST 2018	1732164.0	WOOD BLACK BOARD ANT WHITE FINISH	6	8.91	United Kingdom
837706	362103.0	5960185	Fri Apr 06 06:15:00 IST 2018	451185.0	RETROSPOT HEART HOT WATER BOTTLE	30	6.84	United Kingdom
...	...	...	...	...	...	...	...	...
264426	372750.0	6162508	Wed Sep 26 13:29:00 IST 2018	482916.0	TRAVEL CARD WALLET VINTAGE TICKET	72	0.58	United Kingdom
980985	372750.0	6162508	Wed Sep 26 13:29:00 IST 2018	358764.0	LOVE POTION MASALA INCENSE	75	0.29	United Kingdom
513245	372750.0	6162508	Wed Sep 26 13:29:00 IST 2018	357252.0	ORIGAMI JASMINE INCENSE/CANDLE SET	30	1.18	United Kingdom
124068	372750.0	6162508	Wed Sep 26 13:29:00 IST 2018	464415.0	METAL SIGN EMPIRE TEA	36	1.10	United Kingdom
717499	372750.0	6162508	Wed Sep 26 13:29:00 IST 2018	464436.0	METAL SIGN HIS DINNER IS SERVED	36	1.10	United Kingdom

1083818 rows × 8 columns

Figure 9. Đổi tên dữ liệu

- Xử lý các transaction bị thiếu dữ liệu (NaN): Đối với các transaction bị thiếu dữ liệu, nguyên nhân có thể do lỗi hệ thống lúc ghi nhận transaction nên để vào có thể gây khó khăn cho model GAT học được các đặc trưng.

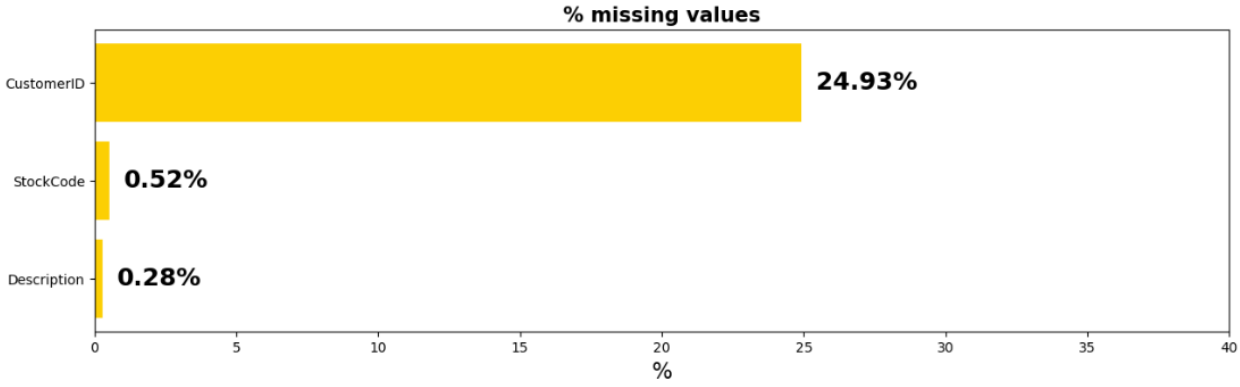


Figure 10. Phần trăm dữ liệu bị thiếu của các cột

- Ba thuộc tính CustomerID, StockCode và Description là 3 thuộc tính bị thiếu dữ liệu nhiều nhất. Do tính chất của các thuộc tính này, phương án tốt nhất là loại bỏ chúng do việc điền vào không khả thi.

```
df[df['CustomerID'].isnull() | df['Description'].isnull()].head()
```

	CustomerID	InvoiceNo	InvoiceDate	StockCode	Description	Quantity	UnitPrice	Country
887319	NaN	6121489	Fri Aug 24 12:39:00 IST 2018	471576.0	NATURAL SLATE CHALKBOARD LARGE	3	6.84	United Kingdom
534750	NaN	6121489	Fri Aug 24 12:39:00 IST 2018	478842.0	FAUX FUR CHOCOLATE THROW	6	27.54	United Kingdom
577586	NaN	6121489	Fri Aug 24 12:39:00 IST 2018	471576.0	NATURAL SLATE CHALKBOARD LARGE	3	6.84	United Kingdom
407380	NaN	6121489	Fri Aug 24 12:39:00 IST 2018	478842.0	FAUX FUR CHOCOLATE THROW	6	27.54	United Kingdom
614635	NaN	6261607	Fri Dec 14 11:50:00 IST 2018	494319.0	WALL ART LOVES' SECRET	6	8.22	United Kingdom

```
# Loại bỏ những bị thiếu một trong hai giá trị CustomerID và Description
df = df.dropna(subset=['CustomerID'])
```

Figure 11. Tập hợp và xử lý hàng có dữ liệu bị thiếu

- Xử lý các transaction bị trùng lập: Tương tự như các transaction bị thiếu dữ liệu, trùng lặp có thể xảy ra do lỗi hệ thống và việc để lại các transaction này có thể nhiễu trong việc GAT model học.

```
# Tìm tất cả các hàng bị lặp
duplicate_rows = df[df.duplicated(keep=False)]

duplicate_rows_sorted = duplicate_rows.sort_values(by=['InvoiceNo', 'StockCode', 'Description', 'CustomerID', 'Quantity'])

duplicate_rows_sorted.head(10)
```

	CustomerID	InvoiceNo	InvoiceDate	StockCode	Description	Quantity	UnitPrice	Country
1024276	374850.0	5900015	Mon Feb 12 04:26:00 IST 2018	456330.0	GLASS STAR FROSTED T-LIGHT HOLDER	18	5.87	United Kingdom
920591	374850.0	5900015	Mon Feb 12 04:26:00 IST 2018	456330.0	GLASS STAR FROSTED T-LIGHT HOLDER	18	5.87	United Kingdom
55169	374850.0	5900015	Mon Feb 12 04:26:00 IST 2018	477792.0	SET 7 BABUSHKA NESTING BOXES	6	10.56	United Kingdom
508218	374850.0	5900015	Mon Feb 12 04:26:00 IST 2018	477792.0	SET 7 BABUSHKA NESTING BOXES	6	10.56	United Kingdom
323263	374850.0	5900015	Mon Feb 12 04:26:00 IST 2018	1492113.0	WHITE METAL LANTERN	18	4.68	United Kingdom
635173	374850.0	5900015	Mon Feb 12 04:26:00 IST 2018	1492113.0	WHITE METAL LANTERN	18	4.68	United Kingdom
528742	374850.0	5900015	Mon Feb 12 04:26:00 IST 2018	1764609.0	KNITTED UNION FLAG HOT WATER BOTTLE	18	4.68	United Kingdom
538323	374850.0	5900015	Mon Feb 12 04:26:00 IST 2018	1764609.0	KNITTED UNION FLAG HOT WATER BOTTLE	18	4.68	United Kingdom
918916	374850.0	5900015	Mon Feb 12 04:26:00 IST 2018	1764609.0	RED WOOLLY HOTTIE WHITE HEART.	18	4.68	United Kingdom
412255	374850.0	5900015	Mon Feb 12 04:26:00 IST 2018	1764609.0	RED WOOLLY HOTTIE WHITE HEART.	18	4.68	United Kingdom

```
# Loại bỏ các transaction bị trùng lập
df.drop_duplicates(inplace=True)

df.shape[0]
```

```
401569
```

Figure 12. Xử lý dữ liệu bị trùng lặp

- Xử lý các transaction bị hủy (cancelled): Khác với các bị thiếu dữ liệu và trùng lặp, transaction khi bị hủy sẽ có thuộc tính Quantity là âm. Điều này không ảnh hưởng tới việc GAT model học mà có thể hỗ trợ trong

việc phát hiện ra dị thường, vì vậy ta có thể giữ lại transaction cancelled và thêm một Transaction_Status để model có thể nhận ra transaction này bị hủy hay không trong quá trình học:

```
# Lọc ra các transaction có số lượng là âm (transaction bị hủy), Tạo cột mới 'Transaction_Status' để thể hiện tình trạng cancelled
df['Transaction_Status'] = df['Quantity'] < 0
df['Transaction_Status'] = df['Transaction_Status'].astype(int)

cancelled_transactions = df[df['Transaction_Status'] == 1]
cancelled_transactions.describe().drop('CustomerID', axis=1)
```

	InvoiceNo	StockCode	Quantity	UnitPrice	Transaction_Status
count	8.871000e+03	8.506000e+03	8871.000000	8871.000000	8871.0
mean	6.149158e+06	6.531598e+05	-92.328373	26.089900	1.0
std	1.426151e+05	4.381926e+05	3516.947922	614.397875	0.0
min	5.900169e+06	4.200000e+01	-242985.000000	0.020000	1.0
25%	6.022484e+06	4.655332e+05	-18.000000	2.010000	1.0
50%	6.156370e+06	4.771200e+05	-6.000000	4.080000	1.0
75%	6.269945e+06	4.873890e+05	-3.000000	6.840000	1.0
max	6.397259e+06	1.894431e+06	-3.000000	53778.600000	1.0

Figure 13. Xử lý transaction bị hủy (cancelled)

- Làm sạch StockCode: StockCode đại diện cho 1 item do chứa mã sản phẩm. Ở đây qua sàn lọc sơ bộ cho thấy có tổng cộng 3209 StockCode riêng biệt và chủ yếu rơi vào từ 7 đến 8 ký tự số nhưng có lẽ ngoài rơi vào 3 hoặc 0 ký tự số.

```
# Tìm ra có bao nhiêu chữ số trong các stock code
unique_stock_codes = df['StockCode'].unique()
numeric_char_counts_in_unique_codes = pd.Series(unique_stock_codes).apply(lambda x: sum(c.isdigit() for c in str(x))).value_counts()

print(numeric_char_counts_in_unique_codes)
```

```
7    2530
8     678
0       1
3       1
Name: count, dtype: int64
```

```
#T Tìm stock code có số chữ số là 0 và 3
anomalous_stock_codes = [code for code in unique_stock_codes if sum(c.isdigit() for c in str(code)) in (0, 3)]

for code in anomalous_stock_codes:
    print(code)
```

```
nan
42.0
```

```
# Loại bỏ các dòng chưa stock code là 0 và 3
df = df[~df['StockCode'].isin(anomalous_stock_codes)]

df.shape[0]
```

```
399654
```

Figure 14. Làm sạch StockCode

- Làm sạch và chuẩn hóa Description: Quan sát sơ bộ cho thấy Description phần đông là uppercase, do đó đối với các trường hợp lowercase ta sẽ chuẩn hóa toàn bộ thành uppercase. Ngoài ra trong các Description ta cũng nhìn thấy một số thuộc loại hình dịch vụ thay vì là sản phẩm, có thể giữ lại nhưng do đồ án chủ yếu tập trung và sản phẩm nên sẽ loại các transaction có sản phẩm là loại dịch vụ.

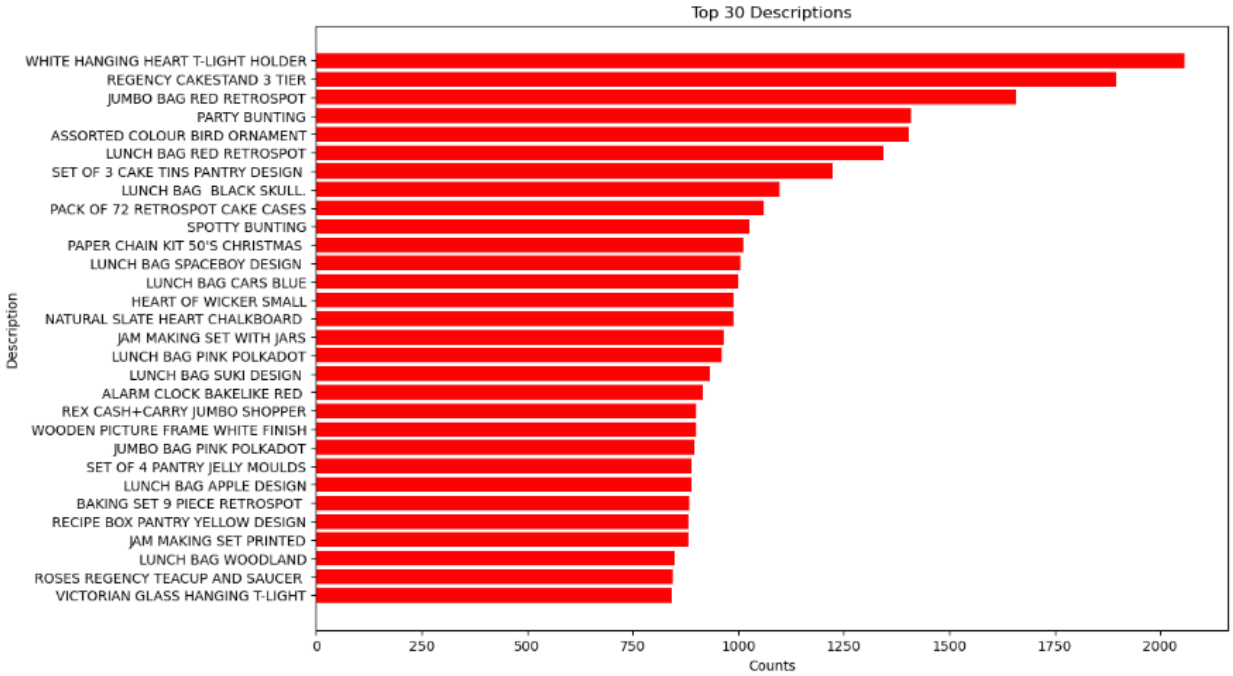


Figure 15. Số lần xuất hiện của các Description trong transaction

```
# Xử lý các description có lower case
lowercase_descriptions = df['Description'].unique()
lowercase_descriptions = [desc for desc in lowercase_descriptions if any(char.islower() for char in desc)]

for desc in lowercase_descriptions:
    print(desc)
```

```
BAG 250g SWIRLY MARBLES
BAG 125g SWIRLY MARBLES
POLYESTER FILLER PAD 45x45cm
ESSENTIAL BALM 3.5g TIN IN ENVELOPE
3 TRADITIONAl BISCUIT CUTTERS  SET
FLOWERS HANDBAG blue and orange
POLYESTER FILLER PAD 30CMx30CM
POLYESTER FILLER PAD 45x30cm
POLYESTER FILLER PAD 40x40cm
FOLK ART GREETING CARD,pack/12
THE KING GIFT BAG 25x24x12cm
FRENCH BLUE METAL DOOR SIGN No
Next Day Carriage
BAG 500g SWIRLY MARBLES
NUMBER TILE COTTAGE GARDEN No
NUMBER TILE VINTAGE FONT No
POLYESTER FILLER PAD 65CMx65CM
High Resolution Image
POLYESTER FILLER PAD 60x40cm
```

```
service_related_descriptions = ["Next Day Carriage", "High Resolution Image"]

# Loại các transaction thuộc dạng service
df = df[~df['Description'].isin(service_related_descriptions)]

# Chuẩn hóa Description sang uppercase
df['Description'] = df['Description'].str.upper()

df.shape[0]
```

```
399571
```

Figure 16. Làm sạch Description

- Lọc các UnitPrice = 0: Các UnitPrice có giá trị bằng 0 có thể là do đây là transaction đại diện cho quà tặng hoặc là sản phẩm bị lỗi. Do ta cần đang tập trung vào transaction giao dịch mua bán nên cần phải loại bỏ chúng đi.

Xử lý Unit Price = 0

[23]:
```
df[df['UnitPrice']==0].describe()[['Quantity']]
```

[23]:

	Quantity
count	33.000000
mean	1261.545455
std	6530.140825
min	3.000000
25%	6.000000
50%	33.000000
75%	108.000000
max	37620.000000

[24]:
```
df = df[df['UnitPrice'] > 0]
```

Figure 17. Xử lý UnitPrice = 0

Thông qua tiền xử lý lần 1, ta còn lại 399538 dòng transactions.

```
df.shape[0]

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 399538 entries, 0 to 399537
Data columns (total 9 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   CustomerID          399538 non-null  float64
 1   InvoiceNo           399538 non-null  object
 2   InvoiceDate         399538 non-null  object
 3   StockCode           399538 non-null  object
 4   Description         399538 non-null  object
 5   Quantity            399538 non-null  int64
 6   UnitPrice           399538 non-null  float64
 7   Country             399538 non-null  object
 8   Transaction_Status  399538 non-null  int32
dtypes: float64(2), int32(1), int64(1), object(5)
memory usage: 25.9+ MB
```

Figure 18. Thông tin dataset sau khai xử lý khâu 1

4.2.2. Mô phỏng Graph dưới dạng 2D và 3D

- **Biễu diễn dưới dạng 2D:**

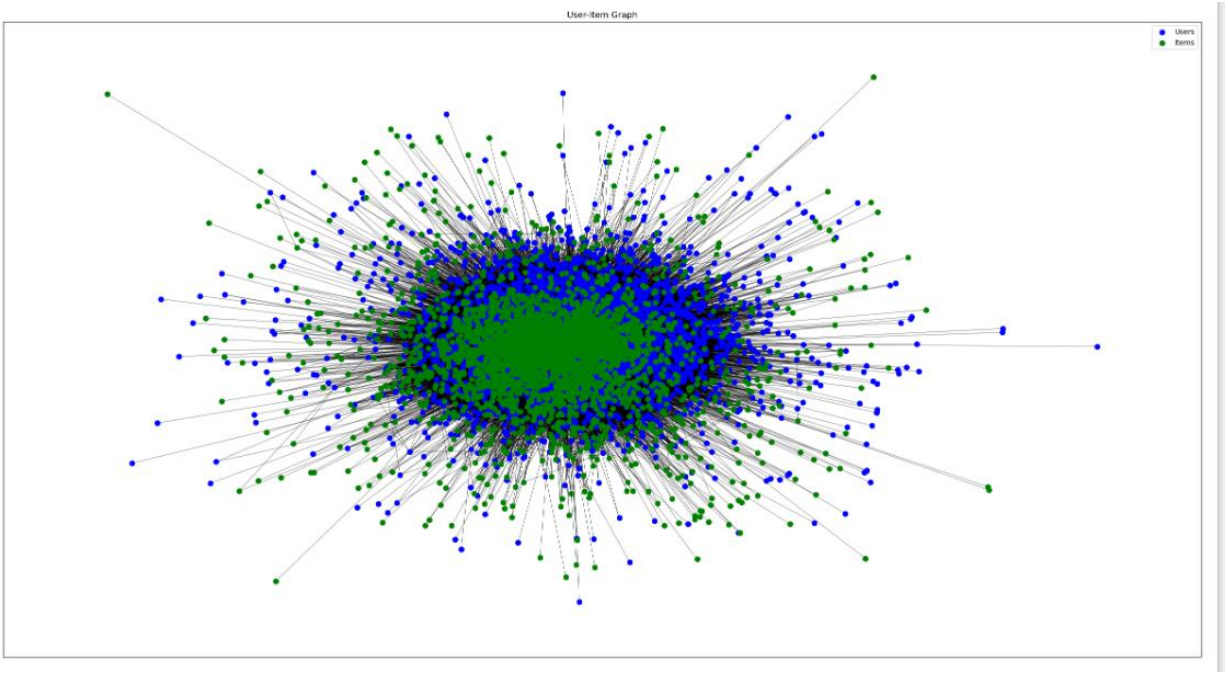


Figure 19. Graph Item-User 2D

- **Biểu diễn dưới dạng 3D:**

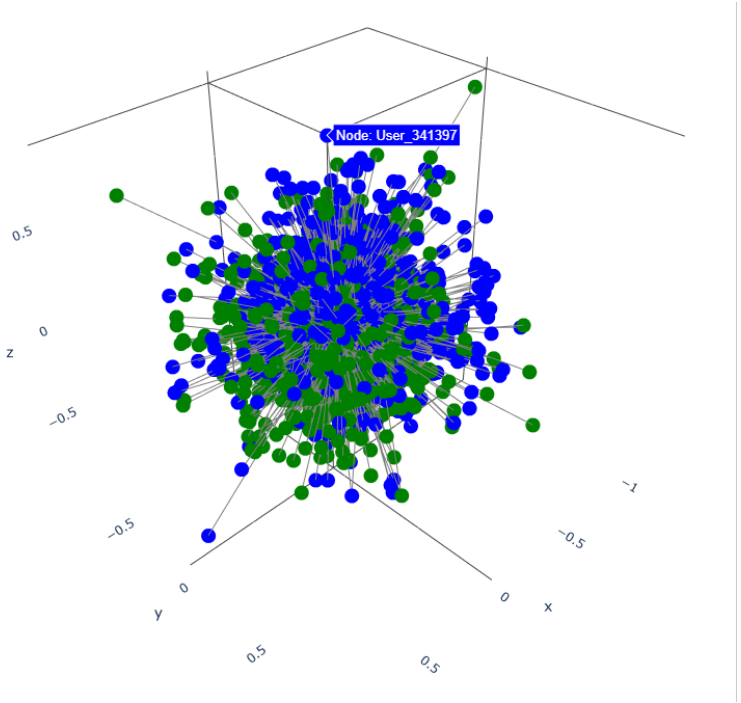


Figure 20. Graph Item-User 3D

4.2.3. Tiền xử lý khâu 2 và chuẩn bị GAT model

Khác với tiền xử lý khâu 1, mục tiêu chính của khâu 2 nhằm chuẩn bị dữ liệu đã có sau khâu 1 thành dữ liệu có thể học được bởi model GAT do GAT model không thể học dữ liệu dạng thô ở hiện tại.

- Tổng hợp các feature cho user và item: Mục tiêu của bước này nhằm giúp cho model GAT có thể nhận diện ra đặc trưng giữa user và item bằng cách sử dụng MinMaxScaler để chuẩn hóa dữ liệu.

```
df['Total_Spent'] = df['Quantity'] * df['UnitPrice']
```

```
# Tổng hợp các features cho users
user_features = df.groupby('CustomerID').agg(
    total_transactions=('InvoiceNo', 'count'),
    successful_transactions=('Transaction_Status', lambda x: (x == 0).sum()),
    canceled_transactions=('Transaction_Status', lambda x: (x == 1).sum()),
    total_spent=('Total_Spent', 'sum'),
    avg_spent_per_transaction=('Total_Spent', 'mean'),
    unique_items_bought=('StockCode', 'nunique')
).reset_index()

print("User Features:")
print(user_features)
```

```
User Features:
      CustomerID  total_transactions  successful_transactions  \
0       259266.0                   2                        1
1       259287.0                 182                      182
2       259308.0                  27                       27
3       259329.0                  72                       72
4       259350.0                  16                       16
...          ...                 ...                      ...
4357    383880.0                  10                       10
4358    383901.0                   7                        7
4359    383922.0                  13                       12
4360    383943.0                 719                      719
4361    384027.0                  70                       70

      canceled_transactions  total_spent  avg_spent_per_transaction  \
0                         1         0.00                   0.000000
1                         0     17879.70                  98.240110
2                         0      5991.12                 221.893333
3                         0      6043.92                  83.943333
4                         0      1221.60                  76.350000
...                     ...          ...                        ...
4357                      0       748.65                  74.865000
4358                      0       334.80                  47.828571
4359                      1       733.05                  56.388462
4360                      0      8465.61                  11.774145
4361                      0      7629.12                 108.987429

      unique_items_bought
0                       1
1                      99
2                      21
3                      71
4                      16
...                   ...
4357                   10
4358                    7
4359                   12
4360                  258
4361                   57

[4362 rows x 7 columns]
```

Figure 21. Tổng hợp feature cho user

```
# Tổng hợp các feature cho items
item_features = df.groupby('StockCode').agg(
    total_purchases=('Quantity', 'sum'),
    unique_buyers=('CustomerID', 'nunique'),
    canceled_ratio=('Transaction_Status', lambda x: (x == 1).sum() / len(x)),
    avg_quantity=('Quantity', 'mean'),
    avg_unit_price=('UnitPrice', 'mean')
).reset_index()

print("Item Features:")
print(item_features)
```

```
Item Features:
      StockCode  total_purchases  unique_buyers  canceled_ratio  avg_quantity  \
0      210042.0             2469             40        0.000000     50.387755
1      211680.0              873             19        0.000000     41.571429
2      212520.0              576             25        0.000000     19.862069
3      212583.0               15              3        0.000000      5.000000
4      212604.0               99              7        0.000000     11.000000
...         ...              ...            ...             ...           ...
3201  1894389.0             1230             10        0.066667     41.000000
3202  1894410.0              792             11        0.000000     41.684211
3203  1894431.0               21              4        0.166667      3.500000
3204  1894452.0               27              2        0.000000      9.000000
3205  1894494.0             2058             15        0.000000     17.589744

      avg_unit_price
0           1.180000
1           0.570476
2           0.290000
3           0.900000
4           0.580000
...              ...
3201        2.804000
3202        2.237895
3203        8.220000
3204        5.180000
3205        1.211026

[3206 rows x 6 columns]
```

Figure 22. Tổng hợp feature cho Item

```
# Chuẩn hóa toàn bộ feature
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
user_features.iloc[:, 1:] = scaler.fit_transform(user_features.iloc[:, 1:])
item_features.iloc[:, 1:] = scaler.fit_transform(item_features.iloc[:, 1:])
```

```
print(item_features)
```

```
      StockCode  total_purchases  unique_buyers  canceled_ratio  avg_quantity
0      210042.0         0.030458       0.044018        0.000000      0.030813
1      211680.0         0.023407       0.020316        0.000000      0.028593
2      212520.0         0.022094       0.027088        0.000000      0.023127
3      212583.0         0.019616       0.002257        0.000000      0.019386
4      212604.0         0.019987       0.006772        0.000000      0.020896
...         ...              ...            ...             ...           ...
3201  1894389.0         0.024984       0.010158        0.066667      0.028449
3202  1894410.0         0.023049       0.011287        0.000000      0.028621
3203  1894431.0         0.019642       0.003386        0.166667      0.019008
3204  1894452.0         0.019669       0.001129        0.000000      0.020393
3205  1894494.0         0.028642       0.015801        0.000000      0.022555

      avg_unit_price
0           0.000094
1           0.000043
2           0.000019
3           0.000070
4           0.000043
...              ...
3201        0.000231
3202        0.000183
3203        0.000687
3204        0.000431
3205        0.000097

[3206 rows x 6 columns]
```

Figure 23. Chuẩn hóa feature cho Item

```
print(user_features)
```

```
      CustomerID  total_transactions  successful_transactions  \
0       259266.0            0.000128                 0.000130
1       259287.0            0.023214                 0.023744
2       259308.0            0.003335                 0.003523
3       259329.0            0.009106                 0.009393
4       259350.0            0.001924                 0.002087
...          ...                 ...                      ...
4357    383880.0            0.001154                 0.001305
4358    383901.0            0.000770                 0.000913
4359    383922.0            0.001539                 0.001566
4360    383943.0            0.092087                 0.093803
4361    384027.0            0.008850                 0.009132

      canceled_transactions  total_spent  avg_spent_per_transaction  \
0                  0.004505     0.000121                   0.000024
1                  0.000000     0.000560                   0.000027
2                  0.000000     0.000268                   0.000030
3                  0.000000     0.000270                   0.000026
4                  0.000000     0.000151                   0.000026
...                     ...          ...                        ...
4357               0.000000     0.000140                   0.000026
4358               0.000000     0.000130                   0.000025
4359               0.004505     0.000139                   0.000026
4360               0.000000     0.000329                   0.000025
4361               0.000000     0.000309                   0.000027

      unique_items_bought
0                0.000000
1                0.057715
2                0.011779
3                0.041225
4                0.008834
...                   ...
4357             0.005300
4358             0.003534
4359             0.006478
4360             0.151355
4361             0.032980

[4362 rows x 7 columns]
```

Figure 24. Chuẩn hóa feature cho User

- Kết hợp dữ liệu đã được xử lý thành dataframe: Chuẩn bị cho bước sau để có thể biến thành Pytorch Geometric tensor để model có thể học.

```
# Đảm bảo user và item theo đúng thứ tự
user_features.set_index('CustomerID', inplace=True)
item_features.set_index('StockCode', inplace=True)

user_features.index = user_features.index.map(lambda x: f"User_{x}")
item_features.index = item_features.index.map(lambda x: f"Item_{x}")

# Kết hợp user và item ra thành dataframe
node_features_df = pd.concat([user_features, item_features], axis=0)
```

```
node_features_df.index = node_features_df.index.map(
    lambda x: x.replace('.0', '') if x.startswith('User_') else x
)
```

```
node_features_df
```

	total_transactions	successful_transactions	canceled_transactions	total_spent	avg_spent_per_transaction	unique_items_bought	total_purchases	unique_b
User_259266	0.000128	0.000130	0.004505	0.000121	0.000024	0.000000	NaN	
User_259287	0.023214	0.023744	0.000000	0.000560	0.000027	0.057715	NaN	
User_259308	0.003335	0.003523	0.000000	0.000268	0.000030	0.011779	NaN	
User_259329	0.009106	0.009393	0.000000	0.000270	0.000026	0.041225	NaN	
User_259350	0.001924	0.002087	0.000000	0.000151	0.000026	0.008834	NaN	
...	...	...	...	...	...	...	...	
Item_1894389.0	NaN	NaN	NaN	NaN	NaN	NaN	0.024984	0.0
Item_1894410.0	NaN	NaN	NaN	NaN	NaN	NaN	0.023049	0.0
Item_1894431.0	NaN	NaN	NaN	NaN	NaN	NaN	0.019642	0.0
Item_1894452.0	NaN	NaN	NaN	NaN	NaN	NaN	0.019669	0.0
Item_1894494.0	NaN	NaN	NaN	NaN	NaN	NaN	0.028642	0.0

7568 rows × 11 columns

Figure 25. Kết hợp lại thành Dataframe

- Thay đổi thứ tự, điền vào dữ liệu trống và thêm vào node_type: Việc đổi lại index thứ tự nhằm giúp cho dữ liệu có thứ tự tương tự với Graph model. Do là có hai loại node riêng biệt nên node user sẽ không có dữ liệu của noded item, và ngược tại, từ đó ta sẽ điền 0 và các dữ liệu trống này. Ngoài ra ta thêm vào node_type để model phân biệt giữa item và user node (1 là item, 0 là user)

```
node_features_df = node_features_df.reindex(node_order)

node_features_df = node_features_df.fillna(0)

node_features_df['node_type'] = node_features_df.index.str.startswith('Item').astype(float)

node_features_df
```

	total_transactions	successful_transactions	canceled_transactions	total_spent	avg_spent_per_transaction	unique_items_bought	total_purchases	unique_b
User_362103	0.050276	0.050359	0.031532	0.000985	0.000026	0.097762	0.000000	0.0
Item_1528779.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.027422	0.0
Item_478086.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.061101	0.3
Item_1732164.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.097602	0.3
Item_451185.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.063367	0.4
...	...	...	...	...	...	...	...	
User_265608	0.002437	0.002609	0.000000	0.000152	0.000026	0.010012	0.000000	0.0
User_334320	0.007311	0.007567	0.000000	0.000138	0.000025	0.030035	0.000000	0.0
User_376908	0.000000	0.000130	0.000000	0.000158	0.000061	0.000000	0.000000	0.0
User_315084	0.019495	0.019961	0.000000	0.000246	0.000025	0.088339	0.000000	0.0
User_309162	0.003463	0.003653	0.000000	0.000140	0.000025	0.015901	0.000000	0.0

7568 rows × 12 columns

Figure 26. Điều chỉnh lại thứ tự cho giống Graph

- Gán dữ liệu vào G Graph ban đầu, biến đổi thành PyTorch Geometric tensor: Trước khi đẩy dữ liệu vào G, ta sẽ copy dữ liệu ban đầu của G vào G_raw nhằm giúp cho việc truy xuất dữ liệu lúc sau.

```
# Giữ lại dữ liệu raw
G_raw = G.copy()

for node in G.nodes:
    G.nodes[node].clear()

node_features_dict = node_features_df.apply(lambda row: row.tolist(), axis=1).to_dict()
nx.set_node_attributes(G,
    {node: {'features': features} for node, features in node_features_dict.items()}
)
```

```
sample_node = list(G.nodes)[7567]
print(f"Node: {sample_node}, Features: {G.nodes[sample_node]['features']}")

print(all('features' in G.nodes[node] for node in G.nodes))
```

```
Node: User_309162, Features: [0.0034628703347441324, 0.0036529680365296807, 0.0, 0.0001396757900089561, 2.4917303117262466e-05, 0.015901060070671377,
0.0, 0.0, 0.0, 0.0, 0.0, [1, 0]]
True
```

Figure 27. Gán dữ liệu lại vào Graph

```
node_features = torch.tensor(
    [G.nodes[node]['features'] for node in G.nodes],
    dtype=torch.float
)
```

```
# Chuyển Graph sang Pytorch Geometrics
data = from_networkx(G)

# Gán nốt features
data.x = node_features
```

```
edge_index = data.edge_index
```

```
print(data)  #Kiểm tra Pytorch Geometric Data
print(data.edge_index)
print(f"Number of nodes: {data.num_nodes}")
print(f"Number of edges: {data.num_edges}")
print(f"Node feature shape: {data.x.shape}")
```

```
Data(edge_index=[2, 521310], features=[7568, 13], quantity=[521310], unit_price=[521310], transaction_status=[521310], invoice_date=[521310], country=[5
21310], num_nodes=7568, x=[7568, 13])
tensor([[   0,    0,    0,  ..., 7567, 7567, 7567],
        [   1,    2,    3,  ...,  600, 2358, 6345]])
Number of nodes: 7568
Number of edges: 521310
Node feature shape: torch.Size([7568, 13])
```

Figure 28. Chuyển Graph thành dạng Pytorch Geometric

- Chuẩn bị GAT model: Thư viện ta sẽ xài ở đây là GATConv (Graph Attention Network)

```
class GATAnomalyModel(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim, heads=8):
        super(GATAnomalyModel, self).__init__()

        self.conv1 = GATConv(input_dim, hidden_dim, heads=heads, concat=True, dropout=0.4)
        self.conv2 = GATConv(hidden_dim * heads, output_dim, heads=4, concat=False, dropout=0.4)

        self.residual_transform = torch.nn.Linear(input_dim, hidden_dim * heads)
        self.scorer = torch.nn.Sequential(
            torch.nn.Linear(output_dim, hidden_dim),
            torch.nn.ReLU(),
            torch.nn.Linear(hidden_dim, 1),
            torch.nn.ReLU()
        )
        self.reconstruction_layer = torch.nn.Linear(output_dim, input_dim)
        self.dropout = torch.nn.Dropout(0.3)

    def forward(self, x, edge_index, return_attention_weights=False):
        x_res = self.residual_transform(x)  # Transform x_res to match dimensions of x

        x, attn_weights_1 = self.conv1(x, edge_index, return_attention_weights=return_attention_weights)
        x = F.elu(x)
        x = x + x_res  # Add residual connection
        x = self.dropout(x)

        x, attn_weights_2 = self.conv2(x, edge_index, return_attention_weights=return_attention_weights)
        x = F.normalize(x, p=2, dim=-1)

        anomaly_scores = self.scorer(x).squeeze()
        reconstructed_x = self.reconstruction_layer(x)

        if return_attention_weights:
            return x, anomaly_scores, reconstructed_x, attn_weights_1[1], attn_weights_2[1]
        return x, anomaly_scores, reconstructed_x
```

Figure 29. GAT model

Hàm khởi tạo (__init__ method)

Các lớp GAT

- **self.conv1**: Một lớp **Graph Attention Convolution (GATConv)**.
    - Kích thước đầu vào: input_dim
    - Kích thước đầu ra: hidden_dim, được nhân với số lượng **đầu chú ý (heads)**.
    - Sử dụng **multi-head attention** (heads=heads) và dropout (dropout=0.4).
    - Đầu ra là sự kết hợp của các đầu chú ý.
- **self.conv2**: Một lớp GATConv khác.
    - Kích thước đầu vào: hidden_dim * heads (đầu ra của conv1).
    - Kích thước đầu ra: output_dim, sử dụng 4 đầu chú ý (heads=4) và không kết hợp các đầu lại.

Kết nối dư

- **self.residual_transform**: Một lớp tuyến tính để biến đổi đặc trưng ban đầu của các nút (input_dim) thành cùng không gian chiều với đầu ra của conv1 (hidden_dim * heads).
    - Dùng để thêm **kết nối dư** vào lớp GAT đầu tiên.

Bộ chấm điểm

- **self.scorer**: Một mạng nơ-ron truyền thẳng (feedforward) để tính điểm bất thường (anomaly score).
    - Lấy các embedding đầu ra từ conv2, sau đó đưa qua:
        - Linear → ReLU → Linear → ReLU.

- Đầu ra là một điểm duy nhất cho mỗi nút, được dùng để phát hiện bất thường.

Lớp tái tạo

- **self.reconstruction_layer**: Một lớp tuyến tính cố gắng tái tạo lại đặc trưng đầu vào ban đầu (input_dim) từ embedding của các nút (output_dim).
  - Cho phép mô hình học cách phát hiện bất thường dựa trên tái tạo (reconstruction-based anomaly detection).

Dropout

- **self.dropout**: Regularization bằng dropout (tỷ lệ dropout = 0.3) để tránh overfitting.

Quá trình lan truyền về phía trước (forward method)

Đầu vào

- x: Ma trận đặc trưng của nút (kích thước: [num_nodes, input_dim]).
- edge_index: Danh sách cạnh (kích thước: [2, num_edges]), biểu diễn kết nối giữa các nút.
- return_attention_weights: Cờ boolean để trả về trọng số chú ý.

Các bước thực hiện

- **Chuẩn bị kết nối dư**:
  - Biến đổi đặc trưng đầu vào x qua self.residual_transform để tạo ra x_res.
- **Lớp GAT đầu tiên**:
  - Đưa x và edge_index qua self.conv1.

- Áp dụng **hàm kích hoạt ELU** để tạo tính phi tuyến.
- Thêm **kết nối dư** (x = x + x_res).
- Áp dụng dropout lên kết quả.

- **Lớp GAT thứ hai**:
  - Đưa đầu ra của conv1 vào self.conv2.
  - Chuẩn hóa embedding (F.normalize) để ổn định.
- **Tính điểm bất thường**:
  - Sử dụng self.scorer để tính điểm bất thường cho từng nút.
- **Tái tạo**:
  - Sử dụng self.reconstruction_layer để tái tạo lại đặc trưng đầu vào từ embedding.
- **Đầu ra**:
  - Nếu return_attention_weights = True, trả về embedding, điểm bất thường, đặc trưng tái tạo, và trọng số chú ý từ cả hai lớp GAT.
  - Nếu không, chỉ trả về embedding, điểm bất thường, và đặc trưng tái tạo.

Chuẩn bị optimizer, hàm loss:

```python
# Optimizer
model = GATAnomalyModel(input_dim=13, hidden_dim=64, output_dim=13, heads=8)
optimizer = torch.optim.Adam(model.parameters(), lr=0.01, weight_decay=5e-4)

# Loss weights
alpha = 0.5  # Weight for reconstruction loss
beta = 0.5   # Weight for feature consistency loss

# Loss function
def combined_loss(edge_index, reconstructed_edge_logits, x, reconstructed_x):
    # 1. Reconstruction Loss (Edge-Level)
    # Extract edge logits for the edges in edge_index
    predicted_edge_logits = reconstructed_edge_logits[edge_index[0], edge_index[1]]  # Shape: [num_edges]
    true_labels = torch.ones_like(predicted_edge_logits)  # Ground truth for existing edges
    recon_loss = F.binary_cross_entropy_with_logits(predicted_edge_logits, true_labels)

    # 2. Feature Reconstruction Loss (Node-Level)
    feature_recon_loss = F.mse_loss(reconstructed_x, x)

    # Combined Loss
    return recon_loss + feature_recon_loss
```

Figure 30. Optimizer và hàm Loss

Bộ tưu hóa (Optimizer)

1. **`torch.optim.Adam`**:
    - Đây là thuật toán **Adam Optimizer**, một phương pháp tối ưu hóa dựa trên **Gradient Descent**.
    - Kết hợp giữa:
        - **Momentum**: Sử dụng trung bình động của gradient để tăng tốc hội tụ.
        - **RMSProp**: Chia gradient cho trung bình bình phương của gradient để điều chỉnh tốc độ học.
    - Thích hợp cho mạng nơ-ron phức tạp như GAT.
2. **`model.parameters()`**:
    - Lấy tất cả các tham số của mô hình cần tối ưu hóa (bao gồm trọng số và độ lệch).
3. **`lr=0.01`**:

- Tốc độ học (**learning rate**). Xác định kích thước bước đi trong không gian tham số ở mỗi lần cập nhật.
- Giá trị nhỏ (0.01) giúp tối ưu hóa ổn định hơn.

4. **`weight_decay=5e-4`**:
   - **Weight Decay (L2 Regularization)**: Một cơ chế để tránh **overfitting** bằng cách phạt các giá trị trọng số lớn trong mạng.
   - Bổ sung một thành phần λ||W||^2vào hàm mất mát, trong đó λ là hệ số `weight_decay`.

Trọng số của hàm mất mát

• **`alpha`**:

- Trọng số dành cho **Reconstruction Loss (mất mát tái tạo)**, biểu thị mức độ quan trọng của việc tái tạo đồ thị.
- Giá trị 0.5 có nghĩa là trọng số này được cân bằng.

• **`beta`**:

- Trọng số dành cho **Feature Consistency Loss (mất mát tái tạo đặc trưng)**.
- Cũng được cân bằng ở mức 0,5.

Hàm mất mát (Loss)

Hàm `combined_loss` kết hợp hai thành phần chính:

- **Reconstruction Loss**: Xác định mất mát ở cấp độ cạnh.
- **Feature Reconstruction Loss**: Xác định mất mát ở cấp độ nút.

Reconstruction Loss (Edge-Level)

- **`reconstructed_edge_logits[edge_index[0], edge_index[1]]`**:
  - Trích xuất xác suất của các cạnh dự đoán từ các **logits** tái tạo dựa trên `edge_index`.
  - **Logits** là đầu ra trước khi đưa qua hàm kích hoạt (softmax hoặc sigmoid).
- **`torch.ones_like(predicted_edge_logits)`**:
  - Sinh ra nhãn thực tế (**ground truth**) cho các cạnh đã tồn tại. Ở đây, tất cả các cạnh thực tế đều có nhãn là 1.
- **`F.binary_cross_entropy_with_logits`**:
  - Tính toán **Binary Cross Entropy Loss** giữa các logits dự đoán và nhãn thực tế:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_o)]$$

  - Trong đó:
    - y_i: Nhãn thực tế (ở đây là 1).
    - p_i: Xác suất dự đoán (sau sigmoid).
- **`recon_loss`**:
  - Kết quả là mất mát của việc tái tạo đồ thị (giá trị nhỏ hơn đồng nghĩa với việc tái tạo chính xác hơn).

Feature Reconstruction Loss (Node-Level)

1. **`F.mse_loss`**:

- Tính **Mean Squared Error Loss** giữa các đặc trưng tái tạo và đặc trưng gốc

$$L = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2$$

- Trong đó:
  - x_i: Đặc trưng gốc của nút i.
  - $\hat{x}_i$: Đặc trưng tái tạo của nút i.

2. **`feature_recon_loss`**:
   - Kết quả là mất mát của việc tái tạo đặc trưng nút (giá trị nhỏ hơn đồng nghĩa với tái tạo tốt hơn).

Tổng hợp các mất mát

• Hàm mất mát cuối cùng là tổng của:

- **`recon_loss`**: Mất mát tái tạo cạnh.
- **`feature_recon_loss`**: Mất mát tái tạo đặc trưng nút.

Hai mất mát này đảm bảo rằng mô hình học được cả cấu trúc đồ thị và đặc trưng nút.

4.2.4. Huấn luyện model

Ở đây sử dụng móc 200 epochs do bị giới hạn thiết bị để huấn luận cho model.

```python
num_epochs = 200
model.train()
edge_losses = []
feature_losses = []
total_losses = []
anomaly_means = []
anomaly_variances = []
silhouette_scores = []
dbi_scores = []

for epoch in range(num_epochs):
    optimizer.zero_grad()

    # Forward pass
    x_hat, anomaly_scores, node_features_reconstructed = model(data.x, data.edge_index)

    # Reconstruct adjacency matrix
    edge_logits = torch.matmul(x_hat, x_hat.T)

    # Compute combined loss
    recon_loss, feature_recon_loss, loss = combined_loss(
        data.edge_index,
        edge_logits,
        data.x,
        node_features_reconstructed
    )

    # Backward pass and optimization
    loss.backward()
    optimizer.step()

    edge_losses.append(recon_loss.item())
    feature_losses.append(feature_recon_loss.item())
    total_losses.append((recon_loss + feature_recon_loss).item())

    with torch.no_grad():
        anomaly_mean = anomaly_scores.mean().item()
        anomaly_variance = anomaly_scores.var().item()
        anomaly_means.append(anomaly_mean)
        anomaly_variances.append(anomaly_variance)

    embeddings = x_hat.cpu().detach().numpy()

    kmeans = KMeans(n_clusters=3, random_state=42).fit(embeddings)
    cluster_labels = kmeans.labels_

    # Compute Silhouette and Davies-Bouldin scores
    silhouette = silhouette_score(embeddings, cluster_labels)
    dbi = davies_bouldin_score(embeddings, cluster_labels)

    silhouette_scores.append(silhouette)
    dbi_scores.append(dbi)

    # Optional: print stats every few epochs
    if (epoch + 1) % 10 == 0:
        print(f"Epoch {epoch + 1}/{num_epochs}, Loss: {loss.item():.4f}, "
              f"Mean Anomaly Score: {anomaly_mean:.4f}, Silhouette Score: {silhouette:.4f}, DBI: {dbi:.4f}")
```

Figure 31. Huấn luyện model

4.3. Các phương pháp đánh giá thực nghiệm

4.3.1. Mất mái tái thiết lập (Reconstruction Loss)

Mất mát tái thiết lập đo lường khả năng của mô hình trong việc tái tạo lại dữ liệu đầu vào từ các biểu diễn mà mô hình đã học được. Trong mạng đồ thị (Graph Neural Networks) như GAT, nó thường được dùng để đánh giá khả năng tái tạo:

1. **Cấu trúc đồ thị (các cạnh)**: Kiểm tra xem mô hình dự đoán sự tồn tại của các cạnh đúng đến mức nào.
2. **Đặc trưng của nút**: Kiểm tra xem các đặc trưng nút có được tái thiết chính xác hay không.

Công thức (Mất mát tái thiết cạnh - Edge-Level Reconstruction Loss):

Binary Cross Entropy (BCE):

$$L_{recon} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_o)]$$

Đối với tái thiết đặc trưng nút, thường sử dụng **Mean Squared Error (MSE)**:

$$L = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2$$

- **Mất mát tái thiết thấp**:

  - Mô hình học và tái tạo tốt cấu trúc đồ thị và/hoặc đặc trưng nút.
  - Chỉ ra rằng dữ liệu tuân theo các mẫu bình thường hoặc dự đoán đúng.

- **Mất mát tái thiết cao**:

- Chỉ ra sự lệch lạc trong cấu trúc đồ thị hoặc đặc trưng nút, có thể là dấu hiệu của **bất thường (anomaly)**.
- Với các nút hoặc cạnh bất thường, mô hình không thể tái thiết chính xác.

```
reconstruction_loss = F.mse_loss( node_features_reconstructed, data.x)
print(f"Reconstruction Loss: {reconstruction_loss.item():.4f}")
```

```
Reconstruction Loss: 0.0084
```

Figure 32. Mất mát tái thiết lập của model

• **Cấu trúc đồ thị (cạnh)**: Mất mát tái thiết cho thấy mức độ mà GAT nắm bắt được mối quan hệ giữa các nút. Mất mát tái thiết cao ở một số cạnh có thể cho thấy **mối liên kết bất thường** trong đồ thị.

• **Đặc trưng nút**: Nếu đặc trưng của nút không được tái thiết tốt, các nút đó có thể bị gắn cờ là **bất thường** (Anomaly) hoặc khác biệt.

4.3.2. Độ bất định trung bình của trọng số Entropy (Average Entropy of Attention Weights)

Cơ chế attention trong GAT gán trọng số cho các cạnh, xác định mức độ quan trọng mà một nút đặt vào các nút lân cận khi tổng hợp thông tin. **Độ bất định trung bình (average entropy)** của các trọng số attention phản ánh:

- **Độ bất định thấp**: Mô hình tập trung vào một vài nút lân cận quan trọng (attention chọn lọc).
- **Độ bất định cao**: Mô hình phân tán sự chú ý giữa nhiều nút lân cận, đối xử chúng gần như bằng nhau.

Công thức cho độ bất định của trọng số attention:

Với trọng số attention a_{ij} của nút i lên các nút lân cận j, độ bất định là:

$$H_i = -\sum_{j \in N(i)} a_{ij} \cdot \log(a_{ij})$$

- N(i): Các nút lân cận của nút i.
- a_{ij}: Trọng số attention giữa nút i và nút lân cận j.

Độ bất định trung bình trên tất cả các nút là:

$$H_{avg} = \frac{1}{N}\sum_{i=1}^{N} H_i$$

• **Độ bất định trung bình thấp**:

- Cho thấy các nút tập trung sự chú ý vào một vài nút lân cận quan trọng.
- Gợi ý rằng có các mẫu rõ ràng hoặc mối quan hệ mạnh trong cấu trúc đồ thị.

• **Độ bất định trung bình cao**:

- Sự chú ý được phân tán rộng rãi giữa các nút lân cận, có thể do dữ liệu nhiễu hoặc kém cấu trúc.
- Có thể chỉ ra các mẫu yếu hoặc không đồng nhất trong đồ thị.

• **Sự rõ ràng của cấu trúc đồ thị**: Độ bất định thấp cho thấy cơ chế attention xác định hiệu quả các nút lân cận quan trọng nhất, giúp mô hình học tốt cấu trúc đồ thị.
• Phát hiện bất thường:

- Độ bất định cao cho một số nút có thể chỉ ra rằng chúng không có mối liên kết rõ ràng, mạnh mẽ với các nút khác (**có khả năng bất thường**).
- Các nút với attention tập trung quá mức vào một vài nút cụ thể có thể đại diện cho các điểm bất thường.

```
def calculate_entropy(weights):
    normalized_weights = weights / weights.sum(dim=-1, keepdim=True)
    entropy = -(normalized_weights * torch.log(normalized_weights + 1e-9)).sum(dim=-1)
    return entropy

edge_attention_entropy_1 = calculate_entropy(attention_weights_1).cpu().detach().numpy()

print(f"Average Entropy of Attention Weights (Layer 1): {np.mean(edge_attention_entropy_1):.4f}")

edge_attention_entropy_2 = calculate_entropy(attention_weights_2).cpu().detach().numpy()

print(f"Average Entropy of Attention Weights (Layer 2): {np.mean(edge_attention_entropy_2):.4f}")
```

```
Average Entropy of Attention Weights (Layer 1): 2.0794
Average Entropy of Attention Weights (Layer 2): 1.3862
```

Figure 33. Độ bất định trung bình của trọng số Entropy hai lớp

4.4. Mô tả thực nghiệm thông qua các bảng, biểu đồ đánh giá, so sánh

4.4.1. Visualization các Anomaly trên Graph 2D và 3D (Threshold 95%):

Figure 34. Anomaly Graph User-Item 2D

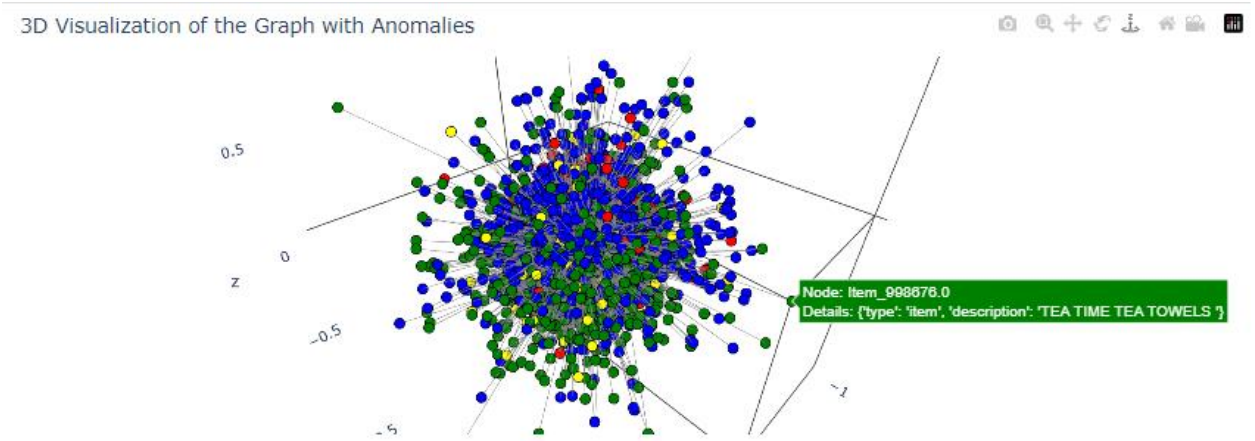


Figure 35. Anomaly Graph User-Item 3D

Trong đó:

- User bình thường: Màu xanh dương.
- Item bình thường: Màu xanh lá.
- Anomaly User: Màu đỏ.
- Anomaly Item: Màu vàng.

4.4.2. Phân bố phổ điểm Anomaly Score trên các nút:

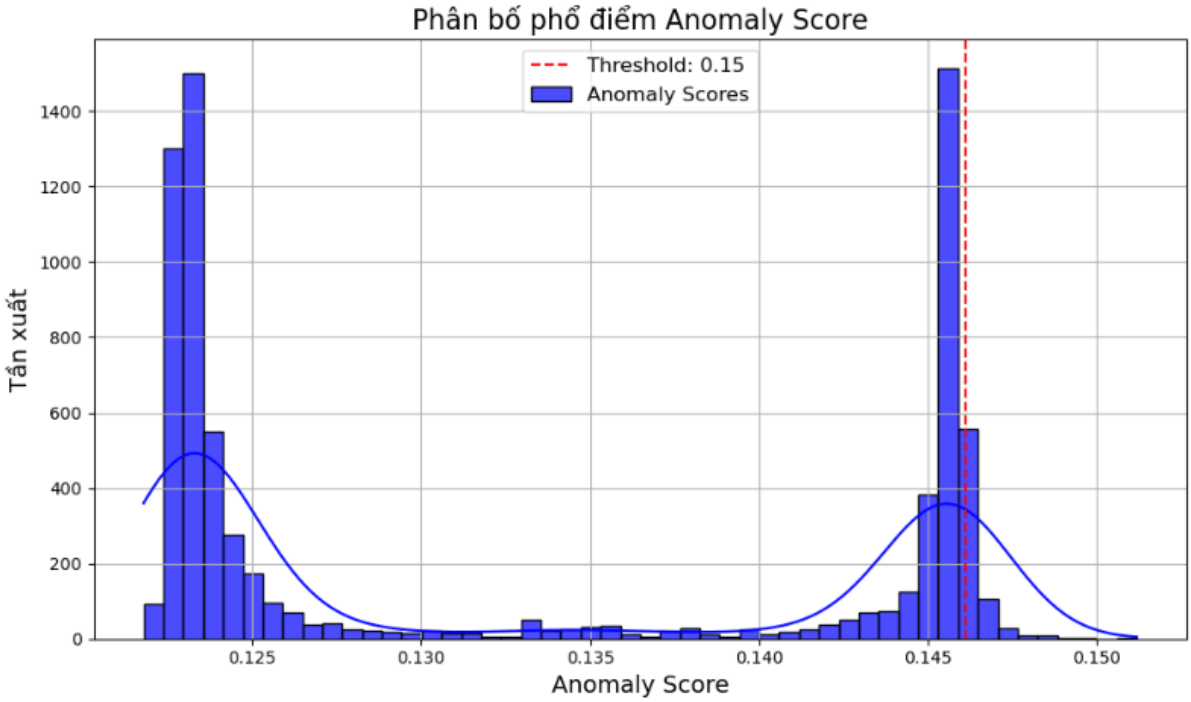


Figure 36. Phân bố phổ điểm Anomaly

- Đại đa số các node rơi vào khoảng dưới (<0.125 và 0.125-0.130) cho thấy nhều node vào trong khoảng bình thường.
- Có một lượng lớn các node trước threshold 95% (5% node anomaly) có số điểm anomaly cao gần kế với các node được tính là anomaly.

⇨ **Model có một sự phân tách rõ ràng giữa hai thái cực của dataset, có thể là do có sự phân chia giữa node user và item, in ra top 10 của hai loại node item và user cho thấy GAT dành attention nhiều cho node Item cho phổ điểm cao hơn.**

```
Top User Anomalies:
Node: User_334740, Score: 0.1366, Details: {'type': 'user'}
Node: User_281211, Score: 0.1363, Details: {'type': 'user'}
Node: User_383733, Score: 0.1361, Details: {'type': 'user'}
Node: User_337281, Score: 0.1361, Details: {'type': 'user'}
Node: User_341397, Score: 0.1361, Details: {'type': 'user'}
Node: User_317100, Score: 0.1361, Details: {'type': 'user'}
Node: User_308259, Score: 0.1360, Details: {'type': 'user'}
Node: User_365022, Score: 0.1360, Details: {'type': 'user'}
Node: User_282492, Score: 0.1360, Details: {'type': 'user'}
Node: User_275079, Score: 0.1359, Details: {'type': 'user'}

Top Item Anomalies:
Node: Item_1892163.0, Score: 0.1512, Details: {'type': 'item', 'description': 'PURPLE FRANGIPANI NECKLACE'}
Node: Item_433902.0, Score: 0.1497, Details: {'type': 'item', 'description': 'FIRST CLASS HOLIDAY PURSE '}
Node: Item_1892121.0, Score: 0.1495, Details: {'type': 'item', 'description': 'WHITE FRANGIPANI NECKLACE'}
Node: Item_433923.0, Score: 0.1494, Details: {'type': 'item', 'description': 'QUEEN OF THE SKIES HOLIDAY PURSE '}
Node: Item_432999.0, Score: 0.1493, Details: {'type': 'item', 'description': 'TROPICAL PASSPORT COVER '}
Node: Item_752115.0, Score: 0.1488, Details: {'type': 'item', 'description': 'ACRYLIC JEWEL SNOWFLAKE, PINK'}
Node: Item_1894368.0, Score: 0.1487, Details: {'type': 'item', 'description': 'PAIR OF PINK FLOWER CLUSTER SLIDE'}
Node: Item_1891239.0, Score: 0.1486, Details: {'type': 'item', 'description': 'DIAMANTE HAIR GRIP PACK/2 RUBY'}
Node: Item_464877.0, Score: 0.1484, Details: {'type': 'item', 'description': 'BATHROOM SET LOVE HEART DESIGN'}
Node: Item_454188.0, Score: 0.1484, Details: {'type': 'item', 'description': 'TRIANGULAR POUFFE VINTAGE '}
```

Figure 37. Top 10 User và Item có điểm Anomaly cao

4.4.3. Quan hệ giữa Anomaly Score và Degree:

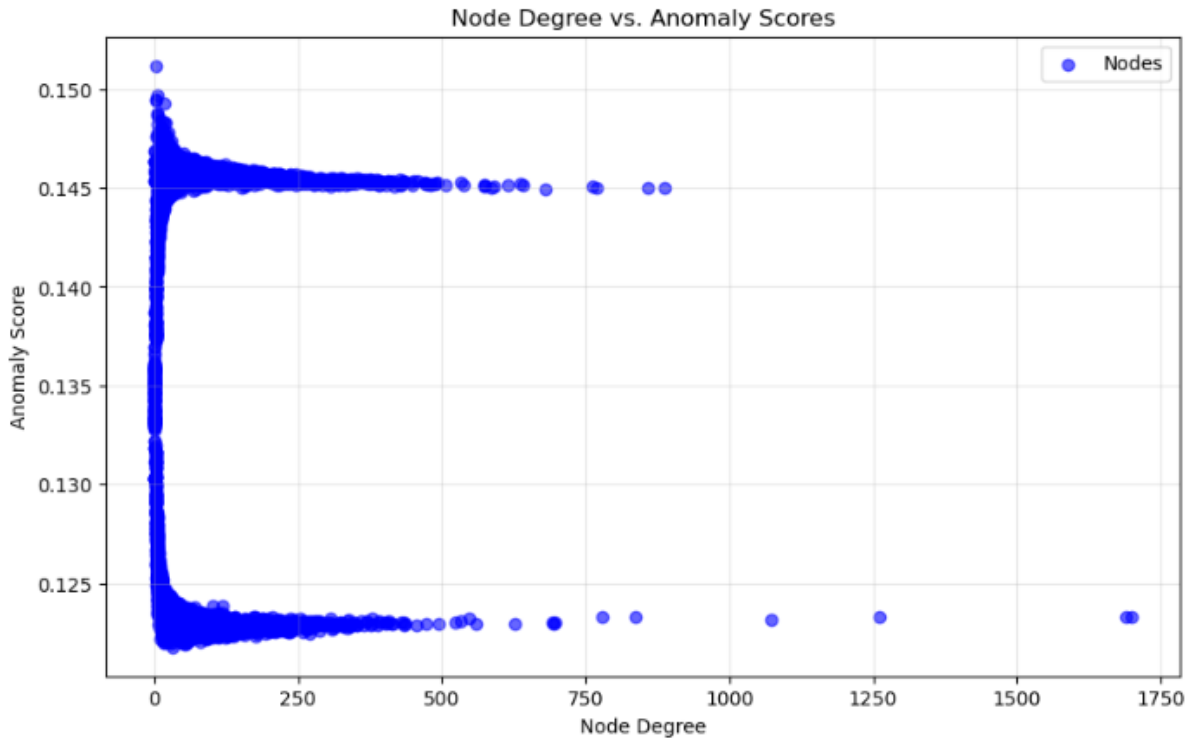


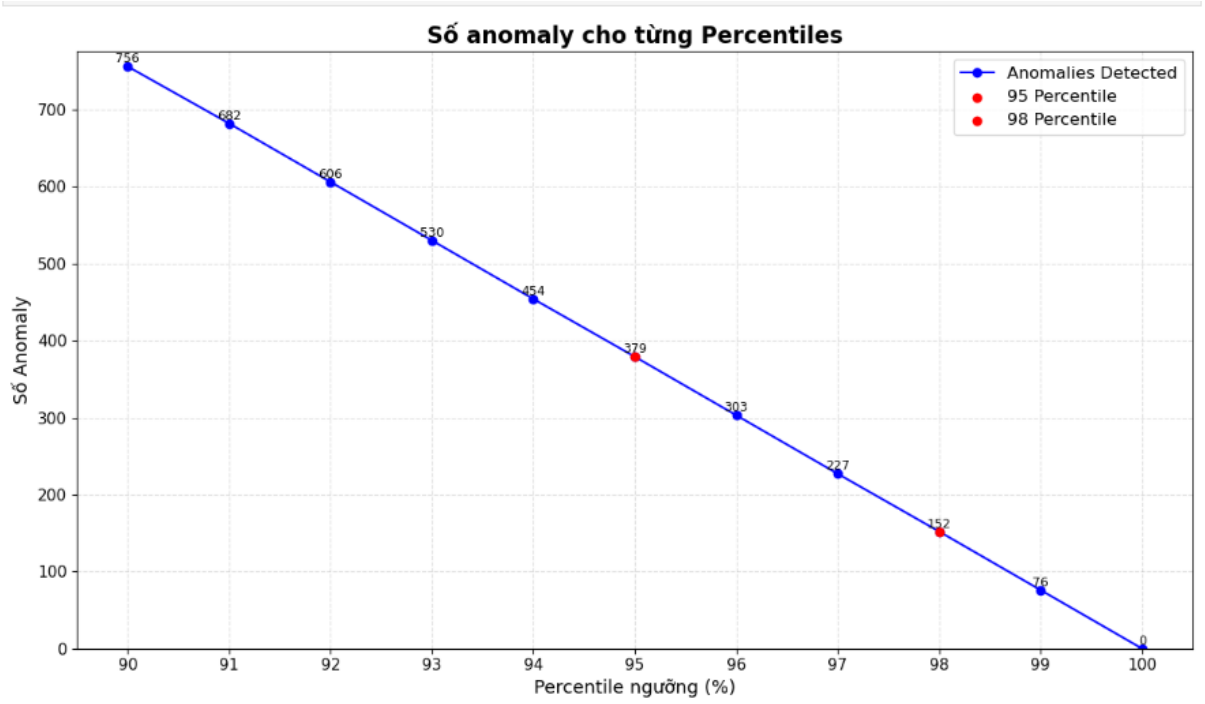
Figure 38. So sánh giữa Node Degree và diểm Anomaly

- Các node gần có degrees thấp (gần bằng 0) có phân bố đều trên toàn bộ phổ điểm Anomaly, khó có thể phân biệt và nhận ra các outlier.

- Một số node có số degree cao (> 500) lại tập trung giữa hai mốc 0.145 và dưới 0.125.

⇨ **Khó có thể thấy được mối liên hệ giữa degrees và điểm Anomaly do phân bố đều giữa các node có degree thấp trên mọi miền điểm Anomaly. Tuy nhiên một số node degree cao (> 500 & >1000) có thể quan sát thấy nằm ngoài outlier bình thường, có thể xem sét được.**

4.4.4. Phân bố số node được tính là anomaly qua các threshold:




```
Percentile: 90%, Anomalies: 756
Percentile: 91%, Anomalies: 682
Percentile: 92%, Anomalies: 606
Percentile: 93%, Anomalies: 530
Percentile: 94%, Anomalies: 454
Percentile: 95%, Anomalies: 379
Percentile: 96%, Anomalies: 303
Percentile: 97%, Anomalies: 227
Percentile: 98%, Anomalies: 152
Percentile: 99%, Anomalies: 76
Percentile: 100%, Anomalies: 0
```

Figure 39. Số node là Anomaly cho từng ngưỡng

- Có sự tụt giảm giữa các mức ngưỡng Threshold khác nhau nhưng chúng đều có chung đặc điểm là giữa các ngưỡng cách nhau 1%, số lượng node bị tính là Anomaly sẽ chênh lệch là 76 nodes.

⇨ **Model có thể chưa hiểu được chuyên sau các mối quan hệ giữa các node, dẫn đến việc nhiều node sẽ trùng số điểm Anomaly gây ra việc chênh lệch đồng đều giữa các mức ngưỡng. Hoặc có thể do dataset có tính đồng nhất cao, không có quá nhiều sự riêng biệt khiến cho model khó có thể nhận dạng các Anomaly.**

4.4.5. Heatmap của attention weight qua các lớp:

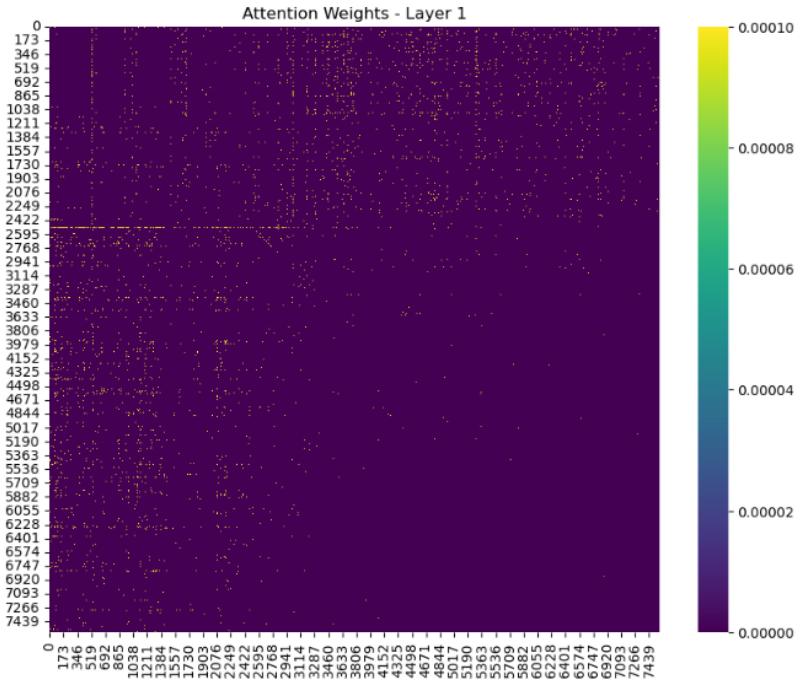


Figure 40. Phân bố mật độ điểm Attention (Layer 1)

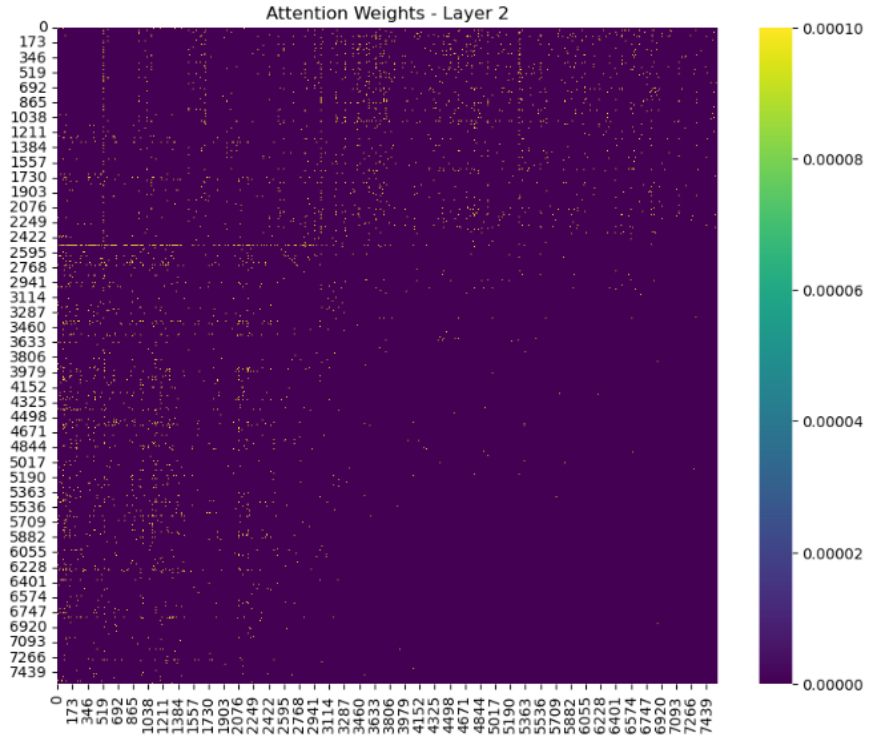


Figure 41. Phân bố mật độ điểm Attention (Layer 2)

- Heatmap giữa cả hai lớp GAT trong model cho thấy đại đa số gần bằng 0 với lượng ít tập trung nhiều vào một bộ phận nhỏ (subset) giữa các node.
- Không có sự khác biệt quá nhiều giữa layer 1 và layer 2.

⇨ **Model tập trung vào một số subset node nhất định, ảnh hưởng có thể do các đặc trưng tạo nên sự phân biệt (bias) nhất định và sự gần như đồng nhất giữa hai layer cho thấy không có quá nhiều trong hiểu biết của model.**

4.4.6. T-SNE (t-Distributed Stochastic Neighbor Embedding) và UMAP (Uniform Manifold Approximation and Projection) minh họa cho các node embedding:

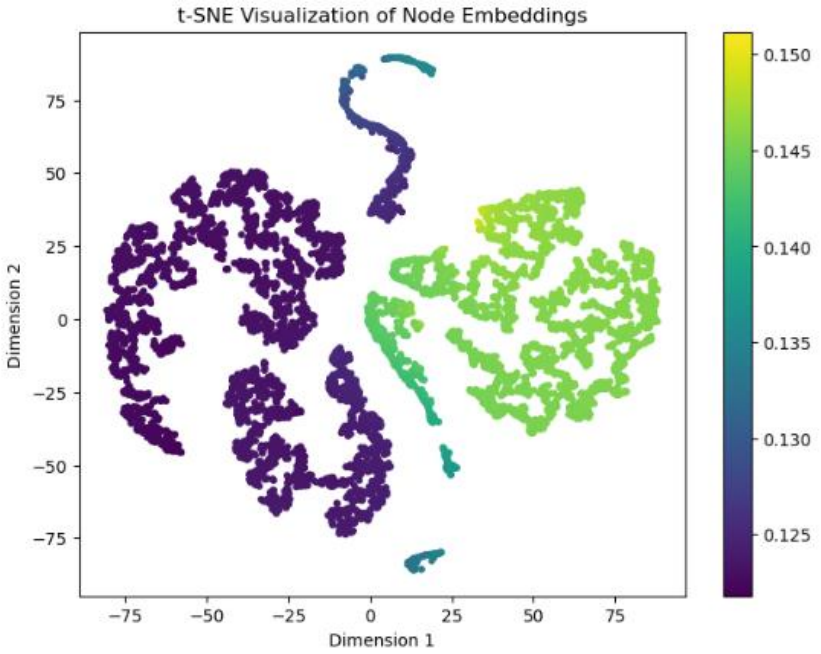


Figure 42. Clustering dựa trên node embedding theo t-SNE

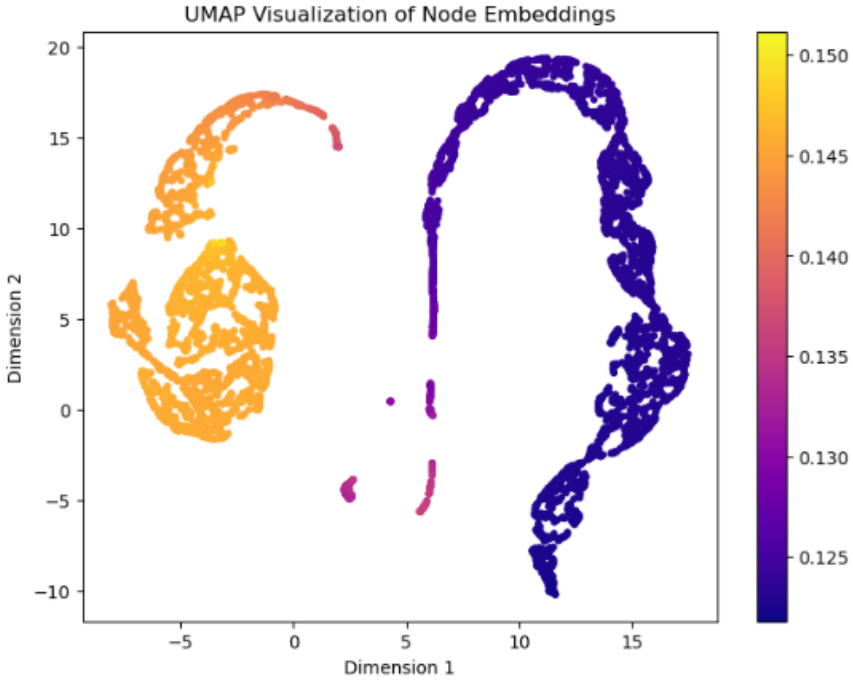


Figure 43. Clustering dựa trên node embedding theo UMAP

- Phân bố điểm giữa các node embedding trong quá trình học của GAT model cho thấy model có thể nhận dạng được các cluster rõ ràng, tương tự như phân bố phổ điểm giữa các node, có sự phân cực giữa các node.
- Đối với t-SNE, ta thấy phương pháp trả về số lượng phân bố giữa các node tạo ra nhiều cluster do mục tiêu chính là bảo tồn cấu trúc cục bộ (local structure) của dữ liệu. Nghĩa là t-SNE cố gắng đảm bảo rằng các điểm dữ liệu gần nhau trong không gian gốc sẽ gần nhau trong không gian giảm chiều. Nguyên lý hoạt động là phân phối xác suất để biểu diễn sự tương tự giữa các điểm dữ liệu. Trong không gian cao, nó dùng Gaussian, và trong không gian thấp, nó dùng t-distribution để giảm thiểu hiệu ứng crowding.
- Đối với UMAP, ta thấy tương tự như t-SNE, cũng trả về phân bố giữa các node với điểm Anomaly, tuy nhiên số lượng cluster giảm xuống gần như chỉ còn 2 clusters. Điều này là do nó bảo tồn cả cấu trúc cục bộ (local structure) lẫn cấu trúc tổng thể (global structure) của dữ liệu bằng cách xây dựng một biểu đồ lân cận (nearest neighbor graph) từ dữ liệu gốc, sau đó tối ưu hóa để biểu đồ này được bảo tồn trong không gian giảm chiều.

⇨ **Model phân biệt rõ giữa các node có anomaly score cao và anomaly score thấp. Một lần nữa cũng cố luận điểm có sự chênh lệch trong attention weight giữa các node Item và User, có thể là do số feature hoặc số lượng. -> Có thể phải điều chỉnh feature đầu vào để cân bằng attention giữa hai loại node.**

4.5. Visualize kết quả training và kết quả quan sát chính phân cụm anomaly:

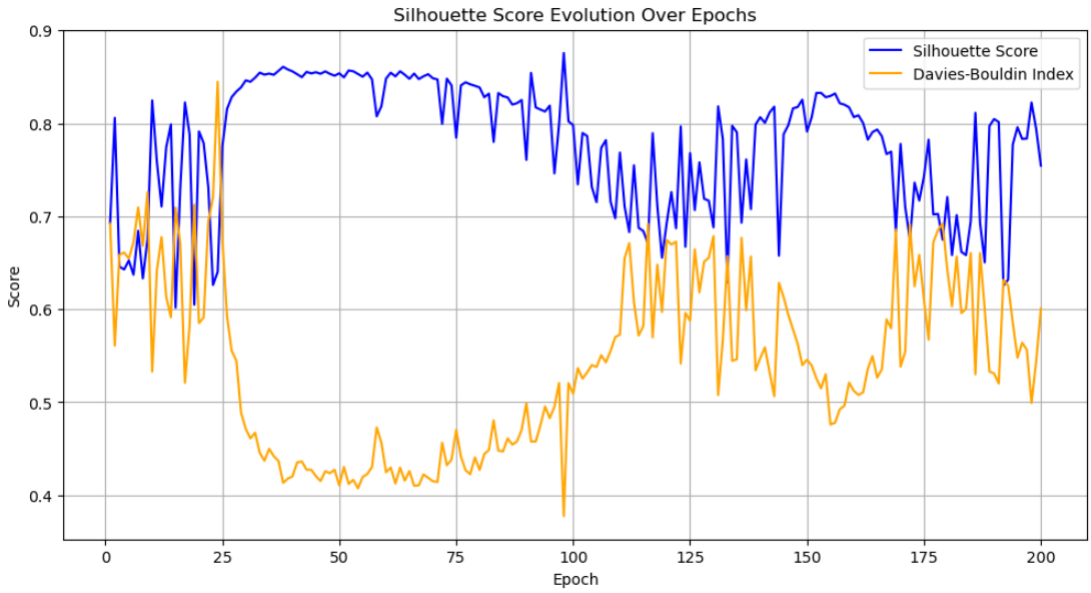


Figure 44. Chỉ số Sillhouette Scoure và DBI qua các Epochs

Dựa trên biểu đồ, nơi Silhouette Score và Davies-Bouldin Index (DBI) được vẽ qua các epoch, ta thấy:

- Silhouette Score (Đường màu xanh):
    - Ban đầu, Silhouette Score dao động nhưng dần ổn định sau khoảng ~30 epoch, cho thấy các embedding bắt đầu hình thành các cụm rõ ràng và tách biệt hơn.
    - Điểm Silhouette cao (gần 1) ở giữa các epoch cho thấy các cụm được hình thành tốt trong giai đoạn này.
    - Về cuối, Silhouette Score có một số dao động nhưng vẫn duy trì ở mức cao, chỉ ra chất lượng phân cụm khá tốt.

⇨ Mô hình đã cải thiện khả năng phân tách cụm của embedding trong quá trình huấn luyện.

⇨ Tuy nhiên, các dao động cuối cùng có thể cho thấy sự bất ổn nhẹ hoặc độ nhạy trong quá trình phân cụm, có thể do vấn đề với learning rate hoặc overfitting (quá khớp).

- Davies-Bouldin Index (Đường màu cam):
    - DBI cho thấy sự bất ổn cao trong ~20 epoch đầu tiên, phản ánh chất lượng phân cụm kém trong các giai đoạn huấn luyện ban đầu (điều này là bình thường).
    - Sau epoch ~30, DBI dần cải thiện (giá trị thấp hơn), cho thấy cụm trở nên chặt chẽ hơn và tách biệt rõ ràng hơn.
    - Ở giữa và cuối các epoch, DBI ổn định với một vài đỉnh nhọn, có thể do gián đoạn tạm thời trong việc hình thành cụm.

⇨ Giá trị DBI thấp hơn sau khi ổn định xác nhận rằng các cụm ngày càng chặt chẽ và rõ ràng hơn.

⇨ Các đỉnh nhọn có thể liên quan đến embedding nhiễu hoặc overfitting khi mô hình cố gắng tinh chỉnh thêm các cụm.

Quan sát chính:

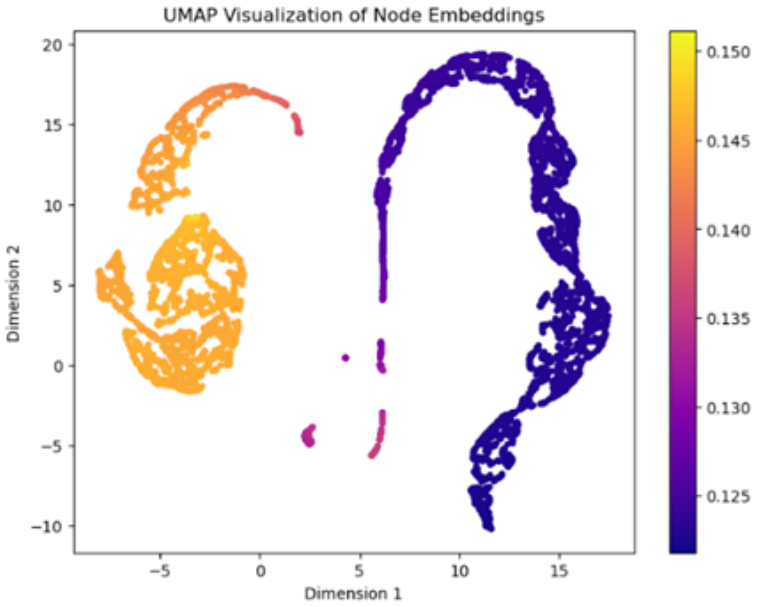


Figure 45. So sánh kết hợp clustering UMAP

- **Kết hợp với UMAP node embedding:**
    - Hai cụm riêng biệt được nhìn thấy trên UMAP phù hợp với Silhouette Score cao trong quá trình huấn luyện. Sự tách biệt rõ ràng này cho thấy không gian embedding mà mô hình học được đã nhóm các node thành các cụm có ý nghĩa.
    - Các phần chồng chéo nhỏ hoặc điểm ngoại lai trong biểu đồ UMAP có thể giải thích cho sự dao động nhỏ của Silhouette Score.
    - Sự **chặt chẽ** của các node trong mỗi cụm trên biểu đồ UMAP phù hợp với các giá trị DBI thấp hơn được quan sát sau khoảng 30 epoch. Các đỉnh tăng đột biến trong DBI có thể liên quan đến các node ngoại lai hoặc các điểm chuyển tiếp khi mô hình tinh chỉnh ranh giới cụm.
    - Nếu DBI tăng lên ở cuối quá trình huấn luyện, điều này có thể chỉ ra việc overfitting hoặc sự giảm độ chặt chẽ của cụm do nhiễu.

- ⇨ Kết hợp Silhouette, DBI và UMAP chỉ ra rằng mô hình đã học được embedding với khả năng tách cụm tốt. Các node có điểm dị thường cao hơn (màu vàng/cam) được tự nhiên nhóm lại với nhau, có thể đại diện cho hành vi bất thường hoặc điểm ngoại lai. Cụm các điểm có điểm dị thường thấp hơn (màu tím) phản ánh các node có hành vi bình thường, hỗ trợ khả năng phân biệt dị thường của mô hình.
- ⇨ Tuy nhiên, một số điểm ngoại lai nhỏ trong biểu đồ UMAP (nằm ở khoảng cách giữa các cụm) có thể đại diện cho các node bị phân loại sai hoặc chuyển tiếp, cần được phân tích thêm.

CHƯƠNG 5: KẾT LUẬN

5.1. Nhắc lại chủ đề

- Đề tài "Transaction Graph Anomaly Detection" tập trung vào việc phát hiện các giao dịch bất thường trong mạng lưới giao dịch tài chính, một vấn đề quan trọng trong việc ngăn chặn các hành vi gian lận, rửa tiền, và bảo vệ hệ thống tài chính. Bằng cách sử dụng Graph Attention Network (GAT), nhóm đã khai thác mối quan hệ phức tạp giữa các thực thể giao dịch để tăng độ chính xác trong việc phát hiện bất thường. Phương pháp này không chỉ hỗ trợ việc giám sát hiệu quả mà còn cung cấp cái nhìn sâu sắc về các giao dịch khả nghi trong các mạng lưới giao dịch phức tạp.

5.2. Tổng hợp các công việc đã làm

- Trong quá trình thực hiện dự án, nhóm đã hoàn thành các công việc sau:
    - Tìm hiểu lý thuyết nền tảng: Nghiên cứu các khái niệm liên quan đến phát hiện bất thường (Anomaly Detection), mạng nơ-ron đồ thị (Graph Neural Networks), và cơ chế attention trong GAT.
    - Hiện thực chương trình thành công

5.3. Đưa ra các ưu điểm và yếu điểm của bài báo cáo

- Ưu điểm:
    - Tính ứng dụng thực tiễn cao: Chủ đề tập trung giải quyết các vấn đề như gian lận tài chính và bảo mật hệ thống, mang tính ứng dụng rõ rệt.
    - Cách tiếp cận hiện đại: Áp dụng GAT, một phương pháp tiên tiến, để khai thác mối quan hệ phức tạp trong mạng lưới giao dịch.
    - Kết quả rõ ràng: Sử dụng các phương pháp trực quan để minh họa sự hiệu quả của mô hình, giúp dễ dàng hiểu được cách hoạt động và kết quả của mô hình.
- Yếu điểm:

- Hạn chế về dữ liệu: Dataset được sử dụng có thể chưa đủ đa dạng để phản ánh các tình huống thực tế phức tạp hơn.
- Phân tích chuyên sâu: Chưa đưa ra được mối liên hệ cụ thể giữa các kết quả thực nghiệm và các đặc điểm bất thường thực tế trong dữ liệu.
- Tối ưu mô hình: Chưa đề cập đến việc thử nghiệm các biến thể khác của GAT hoặc tối ưu hóa mô hình để cải thiện hiệu năng

5.4. Hướng phát triển trong tương lai

- Mở rộng dataset: Thu thập và sử dụng dataset lớn hơn, với các giao dịch thực tế từ nhiều nguồn khác nhau để tăng tính tổng quát.
- Tối ưu hóa mô hình: Khám phá các kỹ thuật tối ưu hóa mô hình như giảm chiều dữ liệu đầu vào, điều chỉnh hyperparameter, hoặc thử nghiệm các biến thể của GAT.
- Tích hợp thêm tính năng: Phát triển mô hình với các đặc trưng khác như thông tin thời gian thực hoặc các mối liên kết động để cải thiện khả năng phát hiện bất thường.
- Ứng dụng thực tiễn: Xây dựng hệ thống giám sát và cảnh báo tự động dựa trên mô hình đã triển khai để thử nghiệm trong các môi trường thực tế như ngân hàng, thương mại điện tử.

BẢNG PHÂN CÔNG CÔNG VIỆC

Nội dung công việc	Nguyễn Trần Gia Kiệt	Cù Ngọc Hoàng	Nguyễn Hoàng Đăng Khoa
Chương 1	x		
Chương 2	x	x	
Chương 3		x	
Chương 4			x
Chương 5	x	x	x

TÀI LIỆU THAM KHẢO

[1] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio, "Graph Attention Networks," 30 10 2017. [Trực tuyến]. Available: https://arxiv.org/abs/1710.10903.

[2] Lingfei Wu, Peng Cui, Jian Pei, Liang Zhao, Graph Neurral Networks.

[3] Hongdan Liu, Yan Liu, Zhicheng Jia, Peng Li, "Vessel Abnormal Behavior Detection and Monitoring Using Graph Attention Network Approaches," 7 10 2022. [Trực tuyến]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4240733.