BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM HÀ NỘI



**Báo Cáo**

**CĐTN Dự án Khoa học/Công nghệ**

**Và THDA**

**Đề tài 4: Dự đoán liên kết**

**Chủ đề: Dự đoán liên kết của dữ liệu các phim và chương trình trên Netflix và gợi ý phim**

Giảng viên hướng dẫn: TS. Lê Thị Tú Kiên

Thầy Lê Xuân Hiền

Sinh viên thực hiện: Lê Minh Tuấn - 705105134

Năm học: 2023-2024

Mục lục

[**CHƯƠNG 1: TỔNG QUAN** 4](#_Toc155452859)

[**1.** **Bài toán dự đoán liên kết** 4](#_Toc155452860)

[**2.** **Mục tiêu nghiên cứu của đề tài:** 4](#_Toc155452861)

[**3.** **Đối tượng và phạm vi thực nghiệm nghiên cứu:** 5](#_Toc155452862)

[**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ PHƯƠNG PHÁP ỨNG DỤNG** 6](#_Toc155452863)

[**I. Cơ sở lý thuyết** 6](#_Toc155452864)

[**1.** **Thuật toán Node2vec (Là thuật toán ánh xạ nút sang vector)** 6](#_Toc155452865)

[**2.** **Metapath2vec** 10](#_Toc155452866)

[2.1 Định nghĩa bài toán 10](#_Toc155452867)

[2.2 Metapath2vec 11](#_Toc155452868)

[**3.** **Thuật toán K – nearest neighbors** 14](#_Toc155452869)

[**4.** **Thuật toán Topological link prediction: Common Neighbors** 14](#_Toc155452870)

[**5.** **Logistic Regression** 15](#_Toc155452871)

[5.1 Input và Output 15](#_Toc155452872)

[5.2 Hàm sigmoid ( Là một dạng của hàm logistic) 15](#_Toc155452873)

[5.3 Hàm mất mát và tối ưu hàm mất 15](#_Toc155452874)

[**6.** **Tham số đánh giá mô hình và AUC PR** 16](#_Toc155452875)

[6.1 Precision và Recall 16](#_Toc155452876)

[6.2 AUC PR ( Area Under The CurvePrecision-Recall) 16](#_Toc155452877)

[**II. Phương pháp ứng dụng** 17](#_Toc155452878)

[**1. Dùng node2vec kết hợp với KNN để dự đoán liên kết** 17](#_Toc155452879)

[**2. Dùng Common Neighbors để dự đoán liên kết** 17](#_Toc155452880)

[**3. Dùng node2vec kết hợp với Logistic Regression để dự đoán liên kết** 18](#_Toc155452881)

[**4. Dùng metapath2vec kết hợp với Logistic Regression** 18](#_Toc155452882)

[**CHƯƠNG 3: THỰC NGHIỆM ỨNG DỤNG TRONG DỰ ĐOÁN LIÊN KẾT ĐỂ GỢI Ý SẢN PHẨM GIẢI TRÍ TRUYỀN HÌNH** 19](#_Toc155452883)

[**1. Chủ đề ứng dụng** 19](#_Toc155452884)

[**2. Mục tiêu ứng dụng:** 19](#_Toc155452885)

[**3. Đối tượng và phạm vi ứng dụng:** 19](#_Toc155452886)

[**II. Phân tích và mô tả cơ bản** 20](#_Toc155452887)

[**1.** **Thu thập và mô tả dữ liệu:** 20](#_Toc155452888)

[1.1 Tổng quan về tập dữ liệu: 20](#_Toc155452889)

[1.2 Mô tả các thuộc tính của tập dữ liệu 20](#_Toc155452890)

[**2. Phân tích vấn đề** 21](#_Toc155452891)

[**III. Cơ sở dữ liệu đồ thị và phân tích dữ liệu** 22](#_Toc155452892)

[**1. Đọc và phân tích/ xử lý dữ liệu** 22](#_Toc155452893)

[**2. Cơ Sở Dữ Liệu đồ thị** 24](#_Toc155452894)

[**IV. Dự đoán liên kết và gợi ý tv show và phim theo xu hướng sở thích** 32](#_Toc155452895)

[**1. Sử dụng Node2vec và KNN để dự đoán liên kết** 32](#_Toc155452896)

[1.1 Tạo các đồ thị con để chạy thuật toán 32](#_Toc155452897)

[1.2 Dự đoán liên kết Show-Genre-Show(SIMILAR\_GENRE)/ SAME-GENRE với đồ thị con undirectedGenre để gợi ý phim theo thể loại 33](#_Toc155452898)

[1.3 Dự đoán liên kết Show-Actor-Show(SIMILAR\_ACTOR)/ SAME\_ACTOR với đồ thị con undirectedActor để gợi ý phim theo diễn viên 34](#_Toc155452899)

[1.4 Dự đoán 2 Show có cả 2 mối quan hệ SAME\_GENRE và SAME\_ACTOR (Similar) trên đồ thị con undirectedActorAndGenre 36](#_Toc155452900)

[**2. Sử dụng Common Neighbors để dự đoán liên kết** 38](#_Toc155452901)

[2.1 Gợi ý các Show có chung thể loại với một Show bằng Common Neighbors ( Dự đoán liên kết SAME\_GENRE) 38](#_Toc155452902)

[2.2 Gợi ý các Show có chung diễn viên với một Show bằng Common Neighbors ( Dự đoán liên kết SAME\_ACTOR) 39](#_Toc155452903)

[**3. Sử dụng Node2vec và Logistic Regression để dự đoán liên kết SAME\_ACTOR** 40](#_Toc155452904)

[3.1 Tạo ra đồ thị con undirectedSameActor với nút nhãn Show và mối quan hệ SAME\_ACTOR 40](#_Toc155452905)

[3.2 Dự đoán liên kết SAME\_ACTOR trên đồ thị con undirectedSameActor 40](#_Toc155452906)

[Source code của báo cáo: https://github.com/OVmath/Link-prediction 43](#_Toc155452907)

[*Nguồn tham khảo:* 43](#_Toc155452908)

# **CHƯƠNG 1: TỔNG QUAN**

1. **Bài toán dự đoán liên kết**

Hiện nay, rất nhiều vấn đề thực tế được giải quyết bằng cách mô hình hóa dưới dạng đồ thị và được tính toán dựa trên CSDL đồ thị. Ví dụ: Mạng lưới đường giao thông vận tải và logistic; mạng xã hội với những liên kết bạn bè, hội nhóm; mạng lưới điện truyền tải;...

Bài toán dự đoán liên kết là một bài toán trong lĩnh vực khoa học máy tính và trí tuệ nhân tạo trên cơ sở dữ liệu đồ thị, liên quan đến việc dự đoán khả năng tồn tại của một liên kết giữa hai nút trong một đồ thị. Bài toán này có nhiều ứng dụng trong thực tế, chẳng hạn như:

* Phân tích mạng xã hội: Dự đoán các mối quan hệ bạn bè, theo dõi, hoặc tương tác giữa các cá nhân trong mạng xã hội.
* Phân tích dữ liệu thương mại: Dự đoán các giao dịch mua bán, hợp tác kinh doanh giữa các doanh nghiệp.
* Phân tích mạng tài chính: Dự đoán các mối quan hệ đầu tư, hợp tác giữa các tổ chức tài chính.
* Phân tích mạng y tế: Dự đoán các mối quan hệ giữa các bệnh.

Mô hình dự đoán liên kết là một mô hình học máy được sử dụng để giải quyết bài toán dự đoán liên kết. Mô hình này thường được xây dựng dựa trên dữ liệu lịch sử về các liên kết trong đồ thị. Dữ liệu này có thể bao gồm thông tin về các thuộc tính của các nút, các mối quan hệ giữa các nút, hoặc các sự kiện xảy ra giữa các nút.

Có thể hiểu đơn giản là dự đoán việc tồn tại cạnh kết nối giữa 2 nút. Nếu cặp nút tồn tại liên kết thì cặp nút đó gán nhãn 1, ngược lại gán nhãn 0.

1. **Mục tiêu nghiên cứu của đề tài:**

**-** Sử dụng các thuật toán về học biểu diễn nút như node2vec, metapath2vec và một số thuật toán khác để dự đoán liên kết trên cơ sở dữ liệu đồ thị.

- Ứng dụng thực nghiệm vào bài toán dự đoán liên kết thực tế cụ thể, từ đó có các đánh giá hiệu quả các thuật toán và phương pháp ứng dụng.

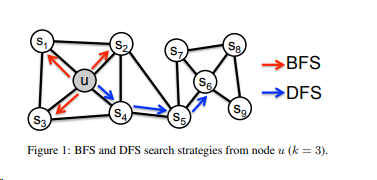
1. **Đối tượng và phạm vi thực nghiệm nghiên cứu:**

* Đối tượng: Sự liên kết giữa các nút trong CSDL đồ thị.
* Phạm vi thực nghiệm: Một CSDL nhỏ gồm hàng ngàn nút.

# **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ PHƯƠNG PHÁP ỨNG DỤNG**

# **I. Cơ sở lý thuyết**

1. **Thuật toán Node2vec (Là thuật toán ánh xạ nút sang vector)**

Hình 1

1.1 Learning Feature Framework

- Ta sẽ coi việc học feature( thuộc tính) trong mạng là bài toán tối ưu hóa maximum hàm likelihood. Nhìn chung là từ thuộc tính của dữ liệu ta sẽ tìm vector tương ứng để làm đầu vào cho mô hình learning.

Đăt G = (V,E) là một mạng/ đồ thị ( Có thể không hoặc có hướng, không hoặc có trọng số).

Đặt f : V -> Rd là một hàm ánh xạ mỗi nút trong tập các nút thành vector biểu diễn feature tương ứng. Tham số d xác định số chiều của vector biểu diễn feature. Như vậy f sẽ là một ma trận có kích thước |V|xd tham số.

Với mỗi nút u, ta định nghĩa NS(u) ⊂ V là một “ mạng láng giềng” của nút u tạo ra bởi chiến lược lấy mẫu( chiến lược đi trên đồ thị từ nút nguồn u) láng giềng S.

Ta sẽ quan tâm đến việc tối ưu hàm mục tiêu sau, là tối đa hóa xác suất log trong mạng láng giềng NS(u) của nút u dựa trên biểu diễn feature của nó, cho bởi f:

(1)

Pr ở đây là xác suất có điều kiện

Để giúp cho bài toán tối ưu đơn giản hơn, ta đặt ra 2 giả thiết chuẩn hóa:

+ Tính toán với giả sử các xác suất xuất hiện của các nút trọng mạng láng giềng NS của u với điều kiện u xuất hiện là độc lập, như vậy ta có:

+ Tính đối xứng trong không gian feature. Nút nguồn và nút láng giềng có ảnh hưởng đối xứng nhau trong không gian feature. Ta mô hình hóa điều kiện xảy ra của mỗi cặp nút nguồn-láng giềng dưới dạng softmax được tham số hóa bằng tích vô hướng các feature của chúng:

Với điều giả sử trên, và hàm likelihood ở PT (1), ta đơn giản hóa thành:

(2)

Hàm . Việc tính toán hàm này rất phức tạp nếu như áp dụng với mạng lớn. Ta sẽ tối ưu PT (2) bằng việc sử dụng stochastic gradient ascent trên các mô hình xác định tham số feature f.d

1.2 Node 2 vec

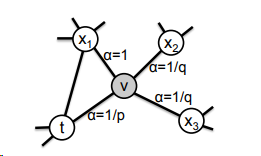
1.2.1 Random walk

Cho nút nguồn u, giả sử random walk với độ dài cố định l. Giả sử, ci là nút thứ i trên đường đi, bắt đầu từ c0=u. Nút ci được xuất hiện theo phân phối xác suất sau:

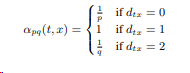
Z là hằng số chuẩn hóa

1.2.2 Search bias α

- Cách đơn giản nhất để xác định xu hướng dịch chuyển của random walk là lấy nút tiếp theo dựa trên trọng số của cạnh nối với nó: wvx tức là: wvx , trong TH không có trọng số thì chọn wvx = 1. Tuy nhiên cách này không tính đến cấu trúc của mạng và hướng tìm kiếm để khám phá ra các loại mạng láng giềng khác nhau.



- Ta định nghĩa 2nd random walk với 2 tham số p và q với hướng đi là vừa di chuyển trên cạnh (t,v) và hiện tại đang ở nút v. Bước đi tiếp theo được xác định dựa trên xác suất dịch chuyển trên cạnh (v, x) bắt đầu từ nút v. Ta định nghĩa xác suất dịch chuyển chưa chuẩn hóa là trong đó:



Trong đó, dtx là khoảng cách giữa t và x.

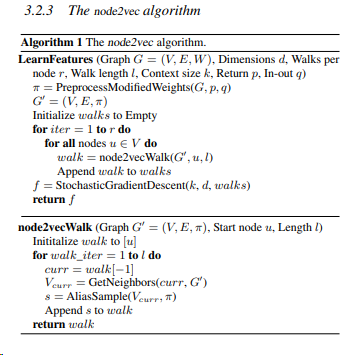
Tham số return p kiểm soát khả năng quay trở lại một nút trong quá trình walk.

Tham số int-out q cho phép tìm kiếm phân biệt giữa các node “inward” và “outward”. Hiểu đơn giản là tìm kiếm rộng hơn.

- Ví dụ: Trong Hình 1, giả sử ta có 1 random walk bắt đầu từ u là {u, s4, s5, s6, s8, s9} có độ dài là l=6, với context size k=3 ta có kết quả là: NS(u) = {s4, s5, s6}, NS(s4) = {s5, s6, s8} and NS(s5) = {s6, s8, s9}.

- Có thể thấy, với random walk độ dài l>k, ta sẽ lấy được mẫu kích thước k với l-k nút.

1.2.3 Thuật toán node2vec



Input của thuật toán là: Đồ thị G = (V,E,W); số chiều d, số lần walk mỗi node là r, độ dài của walk : l, kích thước ngữ cảnh k, Tham số return p, tham số int-out q

Output: là ma trận biểu diễn vector feature f cho mỗi nút thuộc V

B1: Đánh trọng số là xác suất chưa chuẩn hóa π cho đồ thị. Ta có đồ thị mới G’=(V,E,π)

B2: Khởi tạo list rỗng walks

B3: Chạy vòng lặp for như sau, vòng for thứ nhất chạy 1 -> r tức mỗi nút walk r lần, và vòng lặp thứ 2 là duyệt tất cả các nút thuộc V. Trong vòng lặp for thì:

walk = node2vecWalk(G’ , u, l) là câu lệnh lấy ra walk bắt đầu từ nút u chạy trong l lần với đồ thị G’ và theo chiến lược random walk với xác suất là dựa trên trọng số dịch chuyển π . Cuối cùng là thêm walk vào list walks.

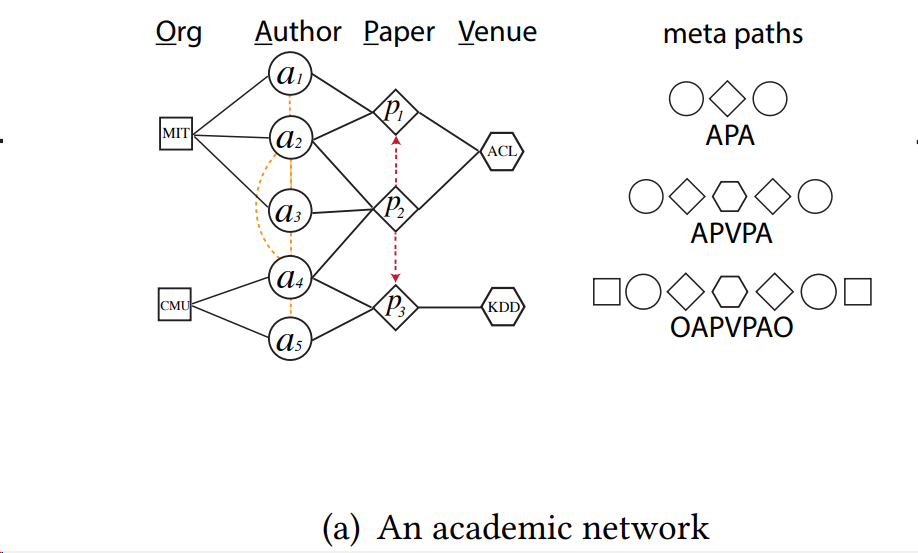
B4: Tham số f sẽ được cập nhật bằng StochasticGradient(k, d, walks) theo hàm cần tối ưu. Tức cập nhật gradient descent theo từng context kích thước k của mỗi walk trong danh sách walks với vector biều diễn mỗi nút có số chiều là d và f là ma trận |V|xd.

1. **Metapath2vec**

2.1 Định nghĩa bài toán

2.1.1 Heterogeneous Network (Mạng không đồng nhất)

- Định nghĩa Mạng không đồng nhất: Mạng không đồng nhất được định nghĩa là một đồ thị G = (V, E, T) với mỗi nút v và mỗi cạnh e theo thứ tự ta có hàm ánh xạ tương ứng ϕ(v) : V → TV và φ(e) : E → TE. TV và TE biểu diễn cho tập các loại đối tượng và loại mối quan hệ, với | TV | + | TE |>2.



- VD: Ta có thể biểu diễn một mạng về học thuật như hình (a), với Authors(A), paper(P), venues(V), organization(O) là các loại của nút. Trong đó các cạnh thể hiện mối quan hệ, đồng tác giả (A–A), xuất bản (A–P, P–V), liên kết(O-A).

2.1.2 Bài toán Representation Learning trong mạng không đồng nhất

- Cho một mạng không đồng nhất G, mục tiêu là học(tìm) một latent representations ( có thể hiểu là ma trận biểu diễn vector) d-chiều XR|V|xd, d << |V| sao cho khái phá được mối quan hệ cấu trúc và ngữ nghĩa trong mạng.

- Input của bài toán chính là mạng không đồng nhất là đồ thị G = (V,E,T). Output của bài toán là ma trận X, với hàng thứ v là vector d chiều Xv tương ứng biểu diễn cho nút v ( Giống node2vec ở điểm là ánh xạ mỗi nút sang vector).

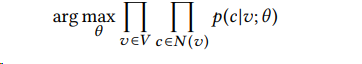
- Việc học biểu diễn các nút trong mạng không đồng nhất này rất hữu ích trong phần loại nút, phân cụm nút, tìm kiếm tương tự,…

2.2 Metapath2vec

2.2.1 Homogeneous Network Embedding ( Nhúng mạng đồng nhất)

- Mạng đồng nhất (Homogeneous Network) có thể hiểu chính là mạng hay đồ thị đầu vào G = (V,E) được đề cập trong phần 1. Thuật toán node2vec Chương 2 ở trên.

**-** Cho một tập văn bản, mô hình word2vec được đề xuất để tìm biểu diễn về phân bố các từ trong tập. Mô hình word2vec được ứng dụng trong mạng đồng nhất. Dựa ý tưởng word2vec thì node2vec nhằm mục đích áp dụng khái niệm words-context(ngữ cảnh từ) trong tập văn bản sang network( mạng ). Node2vec đã sử dụng random walk để làm điều này và dùng mô hình skip-gram để học biểu diễn(vector) của node, điều này giúp dự đoán được cấu trúc ngữ cảnh và các láng giềng gần trong mạng đồng nhất. Như thường, cho một mạng G = (V,E), mục tiêu là cần tối ưu( tối đa hóa) xác suất mạng trong mỗi cấu trúc lân cận, đó là bài toán tối ưu:

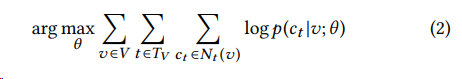


Trong đó N(v) là tập các nút láng giềng của nút v trong mạng G, có thể được định nghĩa(hiểu) theo nhiều cách khác nhau, chẳng hạn như những nút cách nút v 1 cạnh. p(c|v; θ ) được hiểu là xác suất có điều kiện của việc có nút ngữ cảnh c khi cho trước nút v, θlà tham số cần tối ưu.

2.2.2 Heterogeneous Network Embedding( Nhúng mạng không đồng nhất ): metapath2vec

- Để mô hình hóa láng giềng không đồng nhất( tức nút khác loại xung quanh và các mối quan hệ khác nhau) của một nút. Để đưa một cấu trúc của một mạng không đồng nhất vào mô hình skip-gram, ta sẽ sử dụng meta-path-based random walkstrong mạng không đồng nhất.

**- Heterogeneous Skip-Gram.** Trong metapath2vec, ta sẽ dùng skipgram để tìm cách biểu diễn nút trong mạng không đồng nhất G = (V,T,E), |TV|>1 bằng cách tối ưu( tối đa hóa ) xác suất có được tập nút context không đồng nhất Nt(v) t ∈ TV với điều kiện cho trước là nút v:



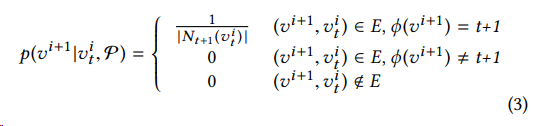
Trong đó, Nt(v) là tập láng giềng của nút v và là các nút thuộc loại t và p(ct |v; θ ) thường là một hàm softmax, công thức là :

Trong đó, Xv là hàng v của ma trận X tức vector biểu diễn cho nút v.

- Có thể thấy trong hình (a), láng giềng của nút tác giả a4 có thể gần cấu trúc với các nút author(ví dụ: a2, a3 và a5), venues VD: ACL và KDD, orginizations ( vd: CMU và MIT, papers( vd: p2 và p3).

- Cho một mẫu âm kích thước M, công thức (2): log p(ct|v, θ) sẽ được cập nhật thành: , trong đó P(u) được xác định là phân phối xác suất mà nút âm um được lấy ra trong M lần. metapath2vec xây dựng phân phối tần suất nút bằng cách coi các loại nút khác nhau đồng nhất và lấy mẫu các nút(âm) bất kể loại của chúng.

- Meta-Path-Based Random Walks. Như random walk trong node2vec, ta cũng có xác suất dịch chuyển p(vi+1|vi). Ta có, sơ đồ meta-path ***P*** được định nghĩa là path được biểu diễn dưới dạng của V1---- R1 → V2------ R2 → · · ·Vt------ Rt → Vt+1 · · · ------- Rl−1 → Vl. Trong đó, R = R1 ◦ R2 ◦ · · · ◦ Rl−1 được định nghĩa là mối quan hệ liên hợp giữa nút V1 và Vl. VD: Trong hình (a), metapath “APA” biểu diễn mối quan hệ đồng tác giả trên paper(A) giữa 2 author (A), và “APVPA” biểu diễn 2 tác giả (A) công bố papers(P) trong cùng một địa điểm venues(V). Như vậy, cho một mạng không đồng nhất G = (V,T,E) và meta-path ***P***: V1---- R1 → V2------ R2 → · · ·Vt------ Rt → Vt+1 · · · ------- Rl−1 →Vl. Xác suất dịch chuyển tại bước thứ i được định nghĩa là:



Trong đó, vti∈ V và Nt+1(vti) là tập láng giềng loại Vt+1 của nút vti, nói cách khác, vi+1 ∈ Vt+1, đó là luồng của walker dựa theo metapath ***P***. Hơn nữa, meta-paths thường theo đường đối xứng, tức là loại của nút V1 sẽ giống với loại của nút cuối là Vl . Như vậy, ta cũng sẽ có:



Chiến lược meta-path dựa trên chiến lược random walk đảm bảo rằng mối quan hệ ngữ nghĩa giữa các loại của nút có thể tương thích với skip – gram. VD: trong hình (a) ở trên, với cách random walk thông thường, thì các nút kế tiếp của nút a4 ( vừa được di chuyển từ nút CMU), có thể là tất cả các loại nút xung quanh nó: —a2, a3, a5, p2, p3 và CMU. Tuy nhiên, với sơ đồ meta-path ‘OAPVPAO’, như vd trên thì walker sẽ có xu hướng đi về nút loại paper (P) nếu bước trước của nó ở nút loại orgination(O) là CMU.

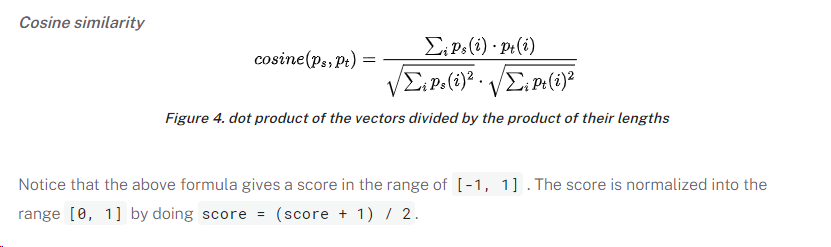
- Cách hoạt động của metapath2vec cũng tương tự như node2vec, với đầu ra output là một ma trận biểu diễn cho từng nút X ∈ R|V|xd, đầu vào input có một chút khác biệt gồm các tham số: đồ thị ( mạng không đồng nhất) G= (V,E,T), sơ đồ meta-path ***P***, số lần walk ở mỗi nút w, walk length l, số chiều d, kích thước láng giềng( tương tự context size). Khác biệt trong cách hoạt động là ở chiến thuật random walk, meta2path sẽ không sử dụng 2 tham số return p và in-out q mà thay vào đó sẽ dựa trên sơ đồ meta-path ***P***. Quy tắc cập nhật cũng dùng SGD như node2vec nhưng khác biệt là ở hàm cần tối ưu.

1. **Thuật toán K – nearest neighbors**

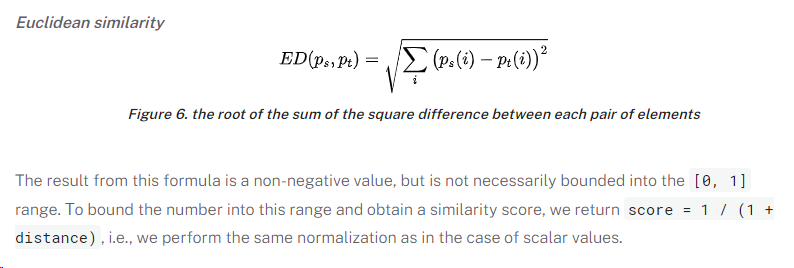
**-** Đây là một thuật toán đơn giản, với độ phức tạp O(N). Hiểu đơn giản là có n phần tử, để tìm k phần tử gần với 1 phần tử cho trước ta so sánh độ đo của phần tử đó với n phần tử và lấy ra k phần tử có độ đo thể hiện gần phần tử đó nhất.

- Hai độ đo phổ biến nhất là ***Cosine similarity***  và ***Euclidean similarity*** có giá trị trong khoảng [0,1]. Càng gần 0 thì 2 phần tử càng khác nhau và càng gần một thì 2 phần tử càng giống nhau.

- Độ đo ***Cosine similarity***  :



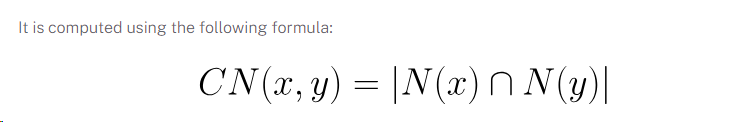
- Độ đo ***Euclidean similarity:***



- Ở đây: ps(i) là giá trị của thuộc tính thứ i của nút s, tương tự với pt(i)

1. **Thuật toán Topological link prediction: Common Neighbors**

- Hiểu đơn giản là giữa 2 nút, ta xác định số nút chung kết nối trực tiếp với 2 nút đó. Từ độ đo này của mỗi cặp nút, ta sẽ dùng để dự đoán liên kết.



- Trong đó, N(x) là tập hợp nút cạnh nối với nút x, tương tự với N(y).

- CN(x,y) là giá trị **Common Neighbors** giữa nút x và nút y

1. **Logistic Regression**

5.1 Input và Output

- Input: Training set X=[x1,x2,…,xN]∈Rd×N  là tập xữ liệu có d chiều với N điểm dữ liệu và  y=[y1,y2,…,yN] là các nhãn tương ứng với tập huấn luyện.

- Output: f(x)=θ(wTx) là dự đoán nhãn qua f(x) và θ được gọi là hàm logistic

5.2 Hàm sigmoid ( Là một dạng của hàm logistic)

- Công thức là: f(s)=1/(1+e-s) ký hiệu là: σ(s)

- σ(s) bị chặn trong khoảng: (0,1)

- Đạo hàm σ(s) là: σ-1(s) = σ(s)(1−σ(s))

5.3 Hàm mất mát và tối ưu hàm mất

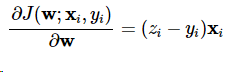
- Ta sẽ sử dụng phương pháp [Stochastic Gradient Descent](https://machinelearningcoban.com/2017/01/16/gradientdescent2/#-stochastic-gradient-descent) (SGD) để tối ưu hàm mất mát.

-  Hàm mất mát với chỉ một điểm dữ liệu (xi,yi) là:



Trong đó: zi=f(wTxi) là giá trị dự đoán và chọn f là hàm sigmoid

- Hàm mất mát có đạo hàm đạo hàm:



- Và công thức cập nhật (theo thuật toán [SGD](https://machinelearningcoban.com/2017/01/16/gradientdescent2/#-stochastic-gradient-descent)) cho logistic regression là:

w=w+η(yi−zi)xi

Trong đó, η là hằng số learning rate.

1. **Tham số đánh giá mô hình và AUC PR**

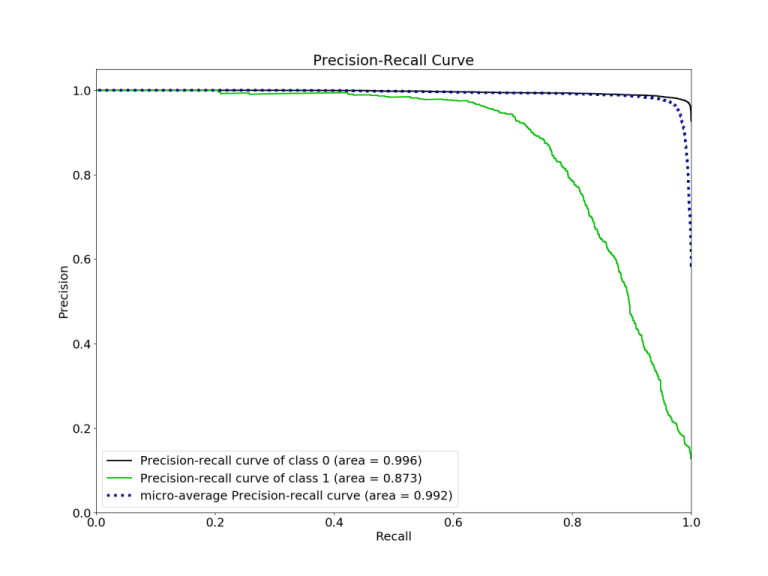
### 6.1 Precision và Recall

- Đối với bài toán link prediction, có thể coi cặp 2 nút có liên kết là lớp positive và ngược lại là cặp được gán nhãn negative.

- Precision **=** TP/(TP+FP). Precision được định nghĩa là tỉ lệ số điểm true positive trong số những điểm được phân loại là positive. Ở công thức này, TP à true positive và FP là false positive.

- Recall = TP/(TP + FN). Recall được định nghĩa là tỉ lệ số điểm true positive trong số những điểm thực sự là *positive* (TP + FN).FN là false negative, tức dự đoán là negative nhưng sai vì nhãn thật là positive.

### 6.2 AUC PR ( Area Under The CurvePrecision-Recall)



- Đường cong Precision – Recall là được thể hiện bởi tọa độ dạng Oxy với các trục giới hạn trong khoảng (0,1) với mỗi điểm trong tọa độ tương ứng với cặp hai giá trị (Recall, Precision).

- Giá trị AUC PR chính là phần diện tích bị giới hạn dưới đường cong Precision – Recall có giá trị trong khoảng (0,1) thể hiện cho hiệu suất của mô hình, giá trị càng gần 1 thì khả năng dự đoán của mô hình càng tốt.

- Với Precision-Recall Curve, AUC còn có một tên khác là **Average precision (AP).**

**-** Để dễ hiểu hơn, tagiả sử có N ngưỡng để tính precision và recall, với mỗi ngưỡng cho một cặp giá trị precision, recall là Pn,Rn, n=1,2,…,N. Precision-Recall curve được vẽ bằng cách vẽ từng điểm có toạ độ (Rn,Pn) trên trục toạ độ và nối chúng với nhau. AP được xác định bằng:

ở đó (Rn−Rn−1)\*Pn chính là diện tích hình chữ nhật có chiều rộng (Rn−Rn−1) và chiều cao Pn, đây cũng gần với cách tính tích phân dựa trên cách tính diện tích của từng hình chữ nhật nhỏ.

## **II. Phương pháp ứng dụng**

## **1. Dùng node2vec kết hợp với KNN để dự đoán liên kết**

- Đầu tiên, dùng node2vec để học biểu diễn các nút trong đồ thị qua cấu trúc và các mối quan hệ của các nút trong đồ thị.

- Sau khi có được vector biểu diễn của từng nút, dùng KNN để đoán ra K nút có vector biểu diễn gần nút cho trước nhất là có mối liên kết tương đồng giữa chúng. VD: Mối quan hệ cùng thể loại giữa 2 bộ phim Movie-inGenre->Genr<-inGenre-Movie thì 2 nút Movie mà chung thể loại sẽ gần nhau về vector biểu diễn hơn.

## **2. Dùng Common Neighbors để dự đoán liên kết**

- Tình toán số nút láng giềng chung giữa 2 nút, nếu 2 nút các có nhiều láng giềng chung tức liêu kết càng chặt chẽ. VD: 2 nút loại Phim(Movie) và có chung thể loại thì sẽ có chung một đến nhiều nút láng giềng loại Genre(Thể loại)

## **3. Dùng node2vec kết hợp với Logistic Regression để dự đoán liên kết**

- Đầu tiên dùng node2vec để học biểu diễn nút qua đồ thị đầu vào.

- Sau đó dùng hồi quy Logistic để dự đoán xem 2 cặp nút có mối quan liên kết hay không, dự đoán 1 nếu có, 0 nếu không tồn tại. Đầu vào của hồi quy Logistic sẽ là các thuộc tính lấy từ độ đo giữa 2 vector biểu diễn của 2 nút như L2, Cosine,...

## **4. Dùng metapath2vec kết hợp với Logistic Regression**

**-** Cũng học biểu diễn nút như node2vec nhưng linh hoạt hơn nhiểu, có thể dùng để dự đoán các loại mối liên kết đa dạng khác nhau. Chẳng hạn như dự đoán một cặp diễn viên (A) có thể hợp tác diễn chung mội phim (M) với thể loại (G) trong tương lai hay không dựa trên bằng sơ đồ meta-path: AMGMA.

- Cách dự đoán bằng hồi quy Logistic sau khi học biểu diễn nút cũng tương tự như phương pháp 3 ở trên.

# **CHƯƠNG 3: THỰC NGHIỆM ỨNG DỤNG TRONG DỰ ĐOÁN LIÊN KẾT ĐỂ GỢI Ý SẢN PHẨM GIẢI TRÍ TRUYỀN HÌNH**

**I. Tổng quan về chủ đề**

**1. Chủ đề ứng dụng**

* Trong đời sống hiện nay, cùng với sự phát triển của internet thì nhu cầu giải trí xem truyền hình qua mỗi chiếc TV, smart phone hay laptop/ máy tính.
* Đặc biệt là giới trẻ hiện nay và bản thân em, nhu cầu này là phát triển rất nhiều với sự phổ cập thiết bị giải trí và dịch vụ mạng xã hội.
* Việc dự đoán sở thích để gợi ý sản phẩm truyền hình theo đúng thị hiếu người dùng đã trở nên vô cùng hữu ích khi giúp mọi người tìm được thứ mình thích để xem, thỏa mãn được nhu cầu giải trí. Đồng thời giúp tăng doanh thu cho các trang Web dịch vụ giải trí truyền hình.
* Trong các sản phẩm truyền hình giải trí thì TV show, đặc biệt là phim được mọi người quan tâm hơn cả. Trong vô vàn rất nhiều bộ phim và TV Show, người dùng cần được gợi ý những sản phẩm hợp với sở thích của bản thân nhất.

**2. Mục tiêu ứng dụng:**

* Tìm kiếm mối liên hệ giữa các phim và TV show với nhau. rồi từ đó, đưa ra gợi ý phù hợp nhất theo xu hướng sở thích người dùng dựa trên thể loại và diễn viên yêu thích.
* Đề xuất phương pháp hiệu quả để thực hiện điều này.

**3. Đối tượng và phạm vi ứng dụng:**

* Đối tượng: Sự liên kết về mối quan hệ giữa các bộ phim và TV Show.
* Phạm vi: Dữ liệu về các bộ phim và TV Show trên nền tảng truyền hình giải trí phổ biến nhất thế giới Netflix.

**II. Phân tích và mô tả cơ bản**

1. **Thu thập và mô tả dữ liệu:**

- Sử dụng bộ cơ sở dữ liệu “Netflix popular movies dataset” :

<https://www.kaggle.com/datasets/shivamb/netflix-shows>

* 1. Tổng quan về tập dữ liệu:
* Dữ liệu dạng csv. Gồm hơn 6000 bộ phim và TV show.
* Gồm các thuộc tính: show\_id, type, title, director, cast, country, release\_year, rating, duration, list\_in, duration, description

### 1.2 Mô tả các thuộc tính của tập dữ liệu

* show\_id: id riêng biệt và duy nhất cho mỗi TV Show và Movie.
* type: Là loại chương trình, có 2 giá trị TV Show và Movie(Phim). Mỗi phần tử có giá trị type là 1 trong 2 giá trị trên.
* title: Tiêu đề của Movie/TV Show.
* director: Thông tin về tên của đạo diễn.
* country: Quốc gia sản xuất.
* cast: Những diễn viên trong Movie/TV Show
* date\_add: Ngày đăng bộ phim trên Netflix
* release\_year: Năm xuất bản
* rating: Xếp hạng truyền hình của movie/show
* list\_in: Các thể loại của movie/show
* description: Thông tin mô tả tổng quan về movie/show dạng text

## **2. Phân tích vấn đề**

* Một người khi xem phim hay chương trình truyền hình họ sẽ thường có xu hướng chủ yếu thích về thể loại và diễn viên đóng. Nguyên nhân là do mỗi người có tính cách riêng phù hợp với những thể loại nhất định VD: Người thích xem phim tài liệu vì thích sự tìm hiểu khám phá, người lại thích xem phim tình cảm vì có nội tâm ưa thích sự lãng mạn và mỗi người sẽ có những diễn viên mà họ thường hâm mộ.
* Mỗi khi xem xong và giải trí thỏa mãn một bộ phim hay TV show, chúng ta thường không có nhu cầu xem đi xem lại nhiều lần những thứ ta đã xem. Trái lại, ta lại có xu hướng tìm những bộ phim và chương trình TV mới phù hợp với sở thích để giải trí sau mỗi lúc lao động và vào khi thời gian rảnh.
* Trong cơ sở dữ liệu các trang web về truyền hình để người xem giải trí, thường có hàng chục nghìn đến hàng triệu bộ phim/ TV show. Ta cần chọn lọc ra một vài sản phẩm phù hợp nhất với thị hiếu người dùng( chủ yếu là về thể loại và diễn viên) để gợi ý.
* Để làm được điều này ta cần dự đoán được mối liên hệ giữa các phim/show dựa trên các thuộc tính của từng phần tử trên bộ dữ liệu để tìm sản phẩm gần với sở thích nhất.
* Để dự đoán những mối liên hệ này, ta cần cấu hình tập dữ liệu dưới dạng CSDL đồ thị gồm các nút và các đường liên kết giữa các nút. Như vậy, sẽ sử dụng python chạy trên nền tảng Jupyer Notebook và phần mềm quản trị CSDL Neo4j Desktop để tạo CSDL đồ thị và thao tác.

# **III. Cơ sở dữ liệu đồ thị và phân tích dữ liệu**

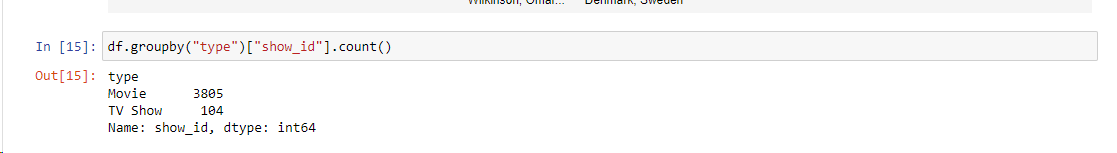
## **1. Đọc và phân tích/ xử lý dữ liệu**

- Đầu tiên ta sẽ khai báo 5 thư viện: numpy để xử lý các số liệu, pandas để xử lý dữ liệu đầu vào mà ta thu thập được, py2neo để tương tác với CSDL đồ thị neo4j bằng python, matplotlib để trực quan hóa dữ liệu, thư viện time để tính thời gian chạy của code.

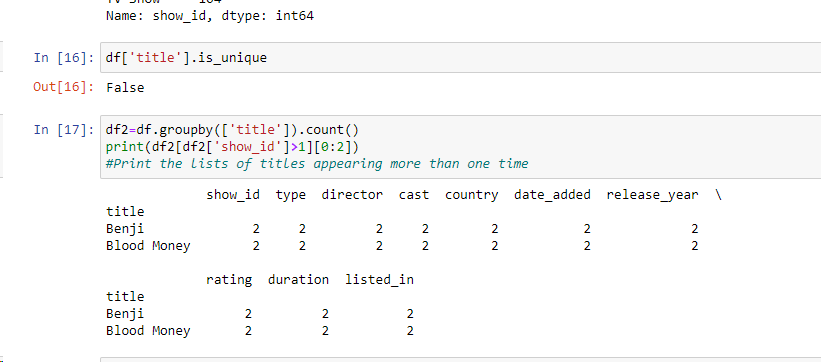
- Đọc dữ liệu sử dụng pandas trỏ đến file csv, sử dụng phương thức read\_csv, bỏ thuộc tính description và lọc bỏ dữ liệu bị thiếu thông tin cần thiết :

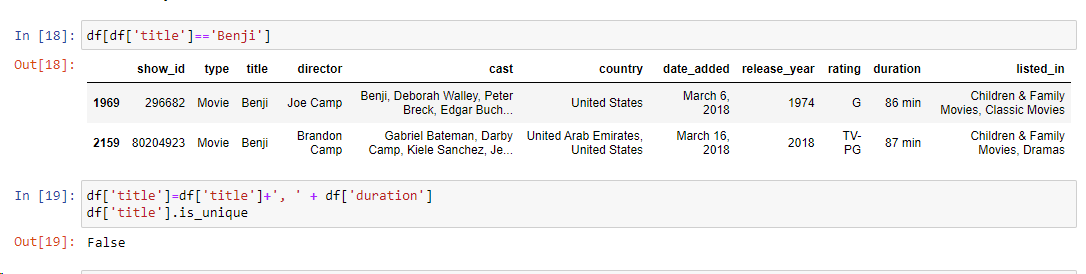


**-** Sau bước xử lý trên, dữ liệu hiện tại có 3805 Movie và 104 TV Show:

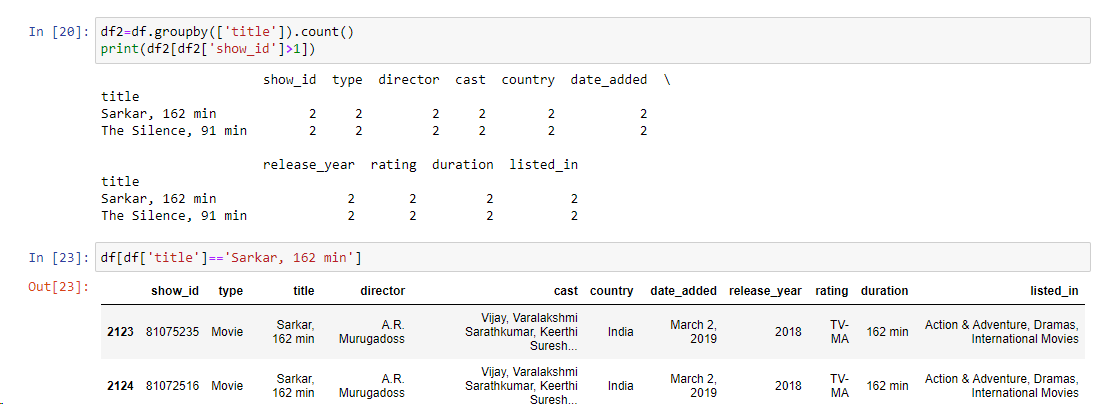


**-** Thuộc tính title tức tiêu đề của Movie/TV Show vẫn có sự trùng lặp, ta cần xử lý để khi gợi ý tiêu đề tránh sự nhầm lẫn:

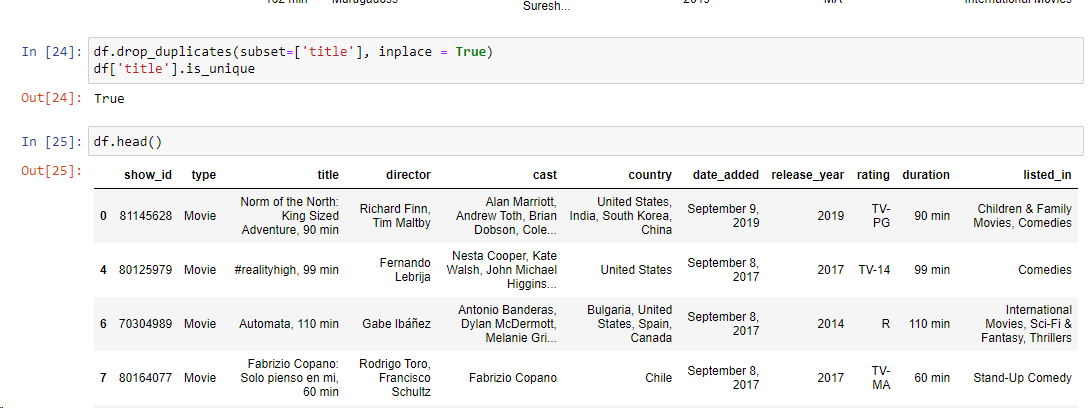
- Để xử lý sự trùng lặp title, ta sẽ ghép thuộc tính title với thuộc tính duration:



- Tuy nhiên, dữ liệu vẫn có sự trùng lặp title:



- Vì vậy, ta sẽ xóa bỏ phần dữ liệu title bị trùng lặp:



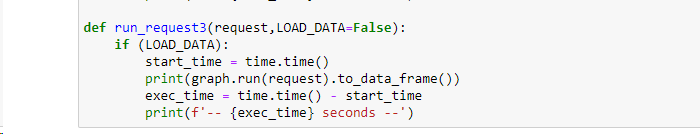
**-** Bây giờ, ta sẽ kết nối CSDL neo4j local và chuyển dữ liệu về dạng dữ liệu đồ thị.

## **2. Cơ Sở Dữ Liệu đồ thị**

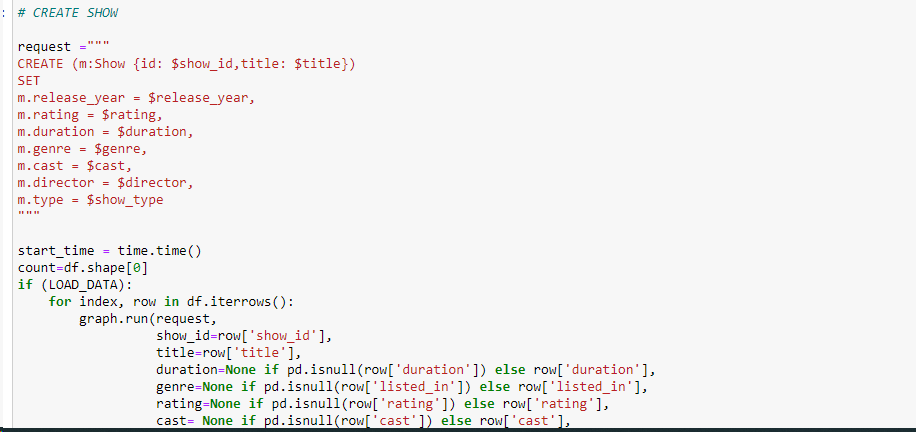
- Sử dụng phần mềm Neo4j Desktop tạo CSDL với tài khoản là mặc định: neo4j và mật khẩu tự thiết lập. Và chọn Start để chạy CSDL.

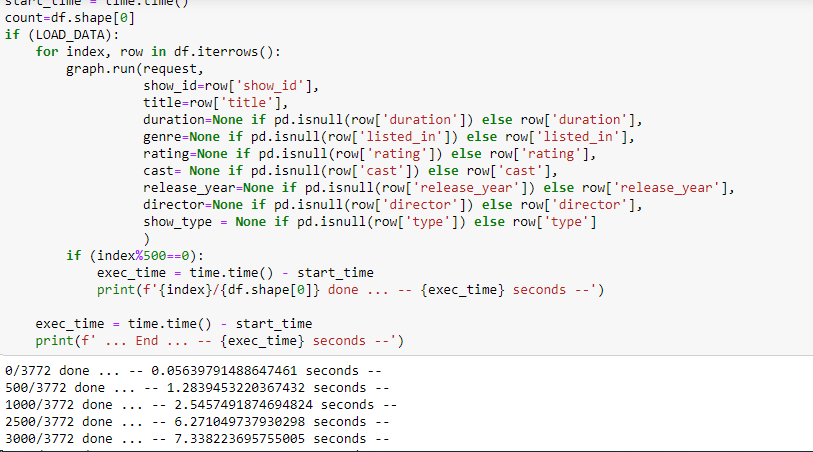
- Sau đó dùng python kết nối đến CSDL neo4j cần sử dụng và định nghĩa các hàm run\_request, run\_request2, run\_request3 để chạy câu lệnh truy vấn CQL và tính thời gian chạy thuật toán:



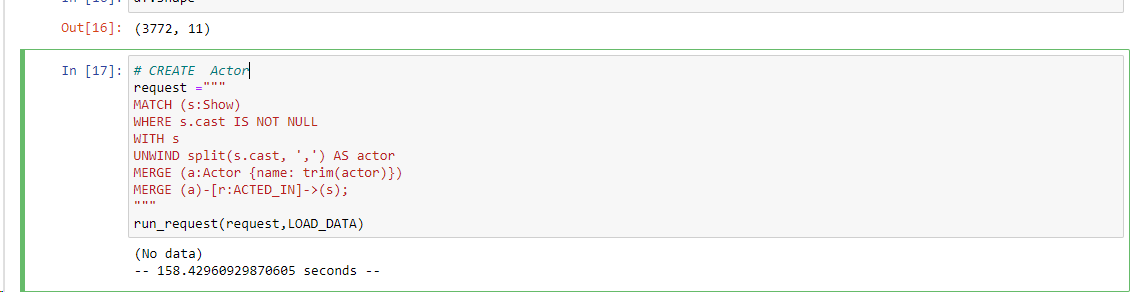


- Tạo các nút Show thể hiện cho các Movie và TV Show cùng với các thuộc tính release\_year, show\_id,title, rating, duration, gere, cast, director, type được truyền vào qua các biến:

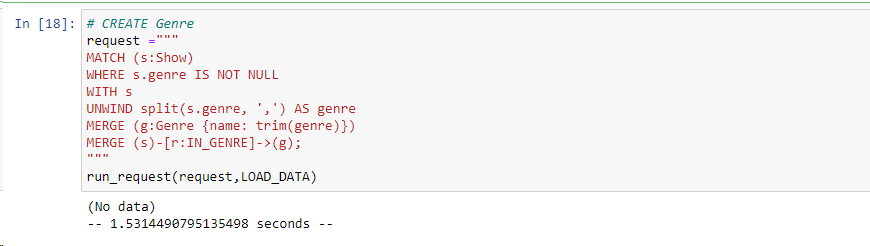




- Từ thuộc tính cast của mỗi nút Show ,tạo các nút Actor đại điện cho mỗi diễn viên đóng trong Movie/TV Show và cạnh ACTED\_IN kết nối với nút Show tương ứng:



**-** Tương tự, từ mỗi nút Show, tạo nút Genre từ thuộc tính genre của nút Show với cạnh Genre với cạnh IN\_GENRE nối từ nút Show tương ứng với thể loại Genre:



**-** Sau đó xóa thuộc tính cast và genre của các nút nhãn Show được xóa vì 2 thuộc tính này đã được biểu diễn qua nút nhãn Genre và Actor:



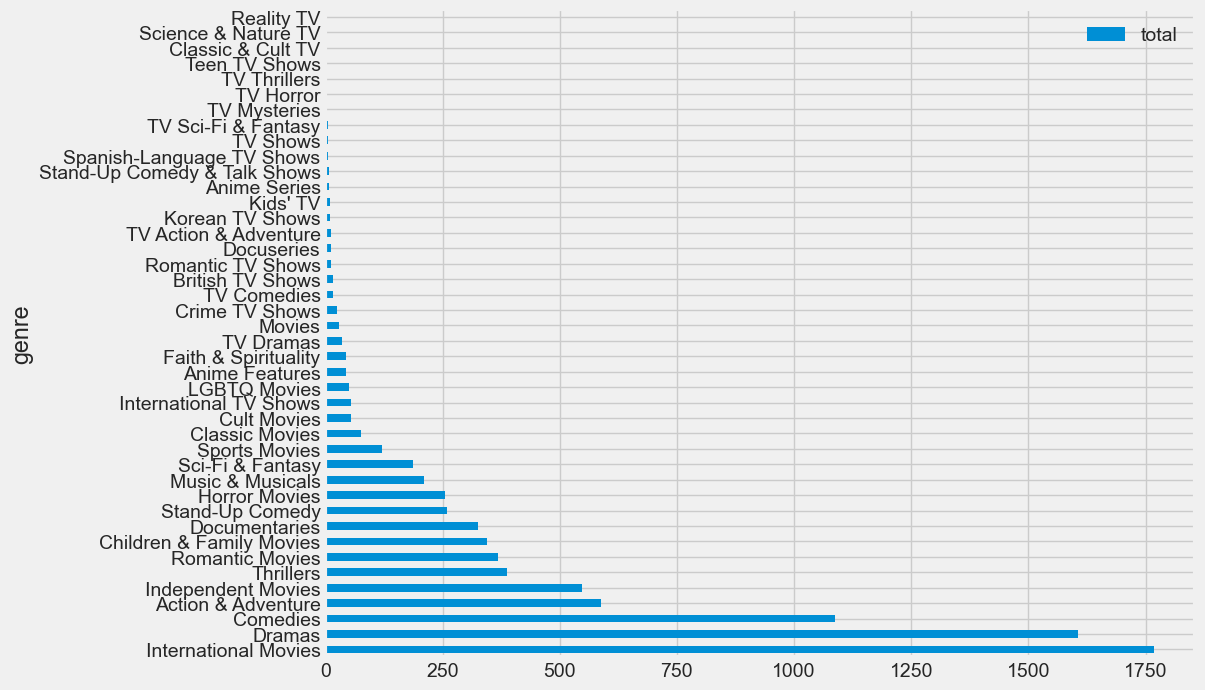
- Với mỗi loại liên kết Show – IN\_GENRE – Genre – IN\_GENRE – Show ta gộp thành liên kết Show- SAME\_GENRE – Show và Show – ACTED\_IN– Actor – ACTED\_IN – Show gộp thành Show- SAME\_ACTOR – Show:



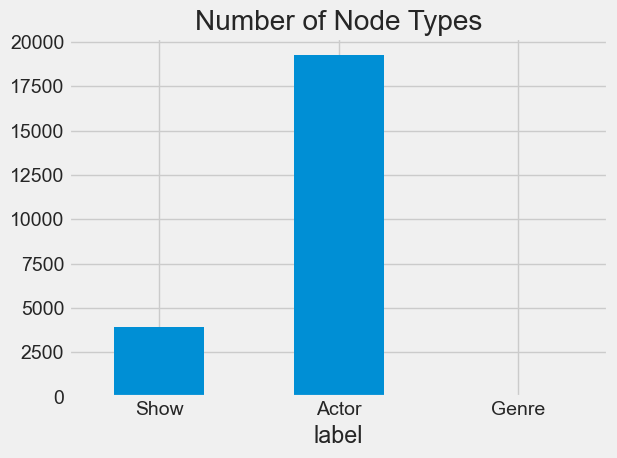
**-** Ta có số lượng relationship là như sau:



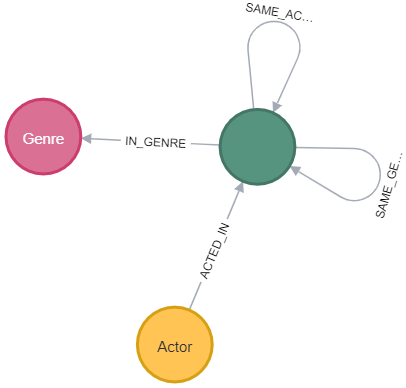
**-** Số lượng Movie/ TV Show với thể loại tương ứng:

****

- Số lượng các nút với nhãn tương ứng:

****

- Sau khi đưa dữ liệu vào CSDL neo4j, ta có được sơ đồ tổng quát các loại nút và các mối quan hệ bằng lệnh CALL db.schema.visualization như sau:

****

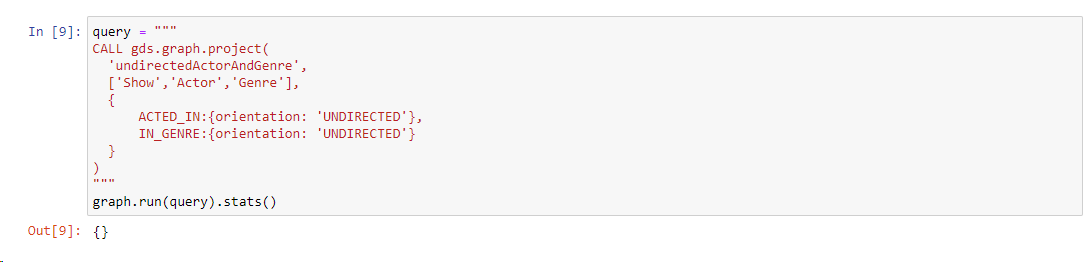
# **IV. Dự đoán liên kết và gợi ý tv show và phim theo xu hướng sở thích**

## **1. Sử dụng Node2vec và KNN để dự đoán liên kết**

- Ta sẽ áp dụng các thuật toán này qua thư viện Graph Data Science dùng trong câu lệnh truy vấn CQL.

### 1.1 Tạo các đồ thị con để chạy thuật toán





Ta có 3 đồ thị con là: undirectedGenre và undirectedActor, undirectedActorAndGenre.

### 1.2 Dự đoán liên kết Show-Genre-Show(SIMILAR\_GENRE)/ SAME-GENRE với đồ thị con undirectedGenre để gợi ý phim theo thể loại

- Dùng node2vec để học feature: thời gian chạy là: 3.47s.



- Dùng KNN để dự đoán liên kết SAME-GENRE bằng cách chọn 8 Show có đặc điểm gần nhất theo vector bởi node2vec: thời gian chạy là 0.78s.



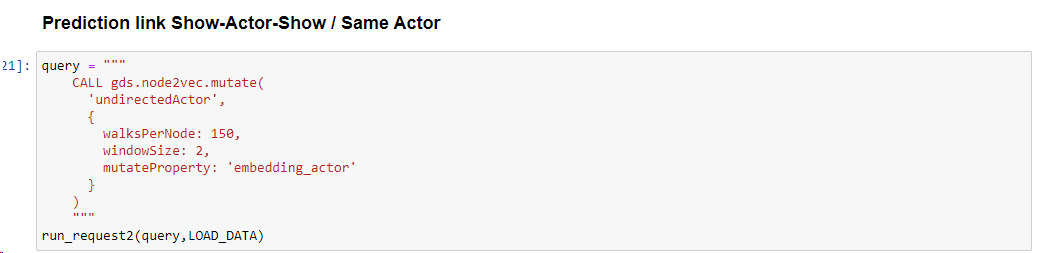
- Độ chính xác đạt: 99.9 %



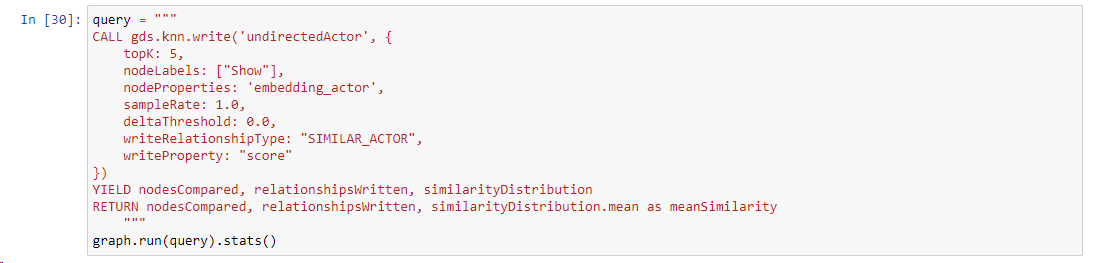
- Nhận xét: Do có rất nhiều liên kết dạng Show-Genre-Show vì số quan hệ IN\_GENRE tới hàng triệu nên node2vec học được nhiều hơn và có được vector feature tốt hơn cho mỗi nút Show, từ đó dự đoán liên kết qua KNN chính xác hơn.

### 1.3 Dự đoán liên kết Show-Actor-Show(SIMILAR\_ACTOR)/ SAME\_ACTOR với đồ thị con undirectedActor để gợi ý phim theo diễn viên

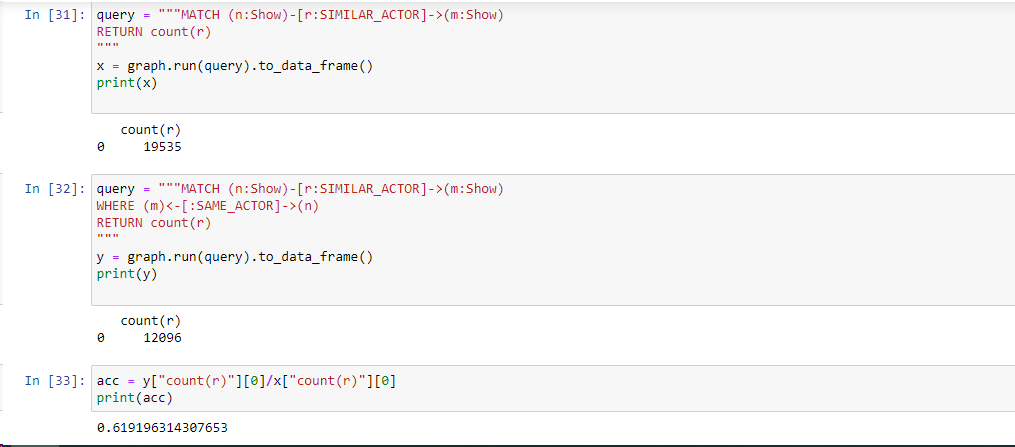
- Dùng node2vec để học feature: thời gian huấn luyện là 169.9s



- Dùng KNN để chọn 5 Show theo các diễn viên theo embedding vector tạo bởi node2vec trên đồ thị con undirectedActor:



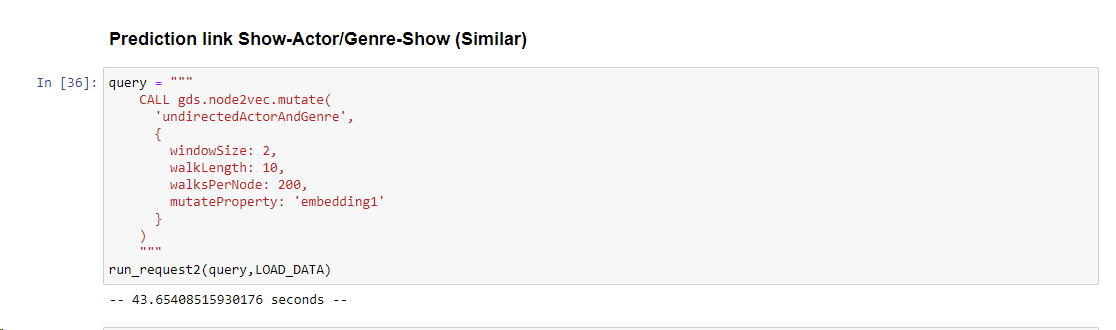
- Độ chính xác là: 61.9%



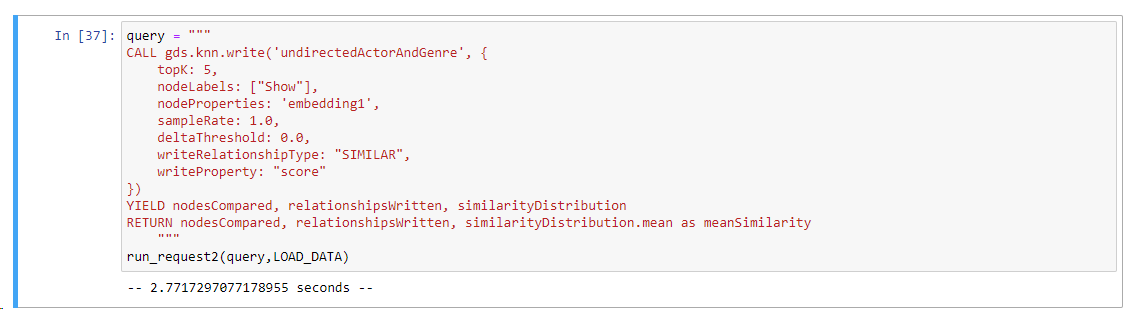
- Nhận xét: Vì chỉ có hơn 20k quan hệ dạng ACTED\_IN nên khả năng học feature cho Show cũng chưa được tốt so với việc dự đoán cùng chủ đề dù đã tăng số bước đi mỗi nút trong node2vec là 150 và thu hẹp lại hệ số k trong KNN xuống là 5. Thời gian chạy cũng tăng lên rõ khi tăng số bước, và để độ dài của mỗi walk mặc định là 80. Do đó, không hiệu quả khi gợi ý Show qua Actor khi sử dụng node2vec và KNN.

### 1.4 Dự đoán 2 Show có cả 2 mối quan hệ SAME\_GENRE và SAME\_ACTOR (Similar) trên đồ thị con undirectedActorAndGenre

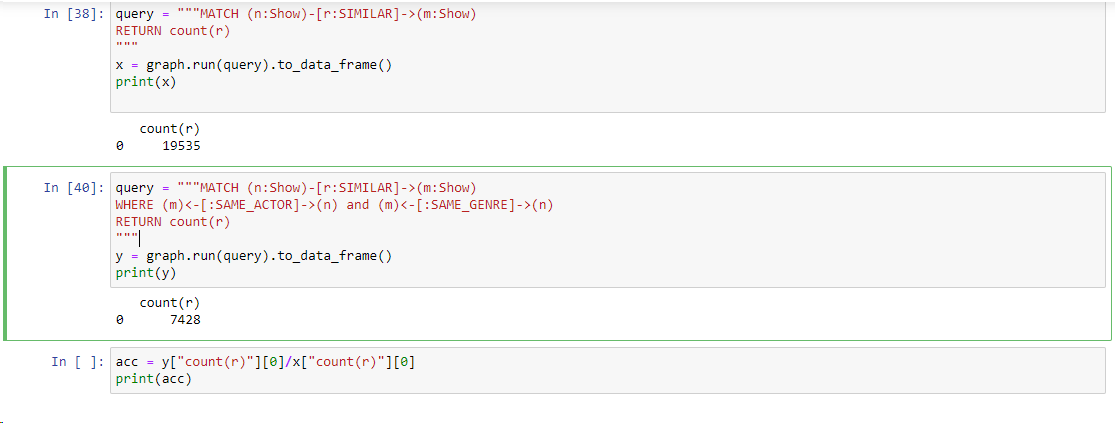
- Dùng node2vec để học feature: thời gian huyến luyện là 43.65s



- Dùng KNN để dự đoán các Show giống nhau cả thể loại lẫn diễn viên: thời gian chạy 2.77s



- Độ chính xác: 7428/19535 < 50%



- Như vậy sử dụng node2vec và KNN để dự đoán liên kết không hiệu quả trong trường hợp này.

## **2. Sử dụng Common Neighbors để dự đoán liên kết**

### 2.1 Gợi ý các Show có chung thể loại với một Show bằng Common Neighbors ( Dự đoán liên kết SAME\_GENRE)

- Gợi ý ra 5 sản phẩm chung nhiều thể loại nhất với sản phẩm cho trước bằng các tạo ra hàm rcm\_by\_cn\_genre(show) và tên sản phẩm truyền qua biến show:

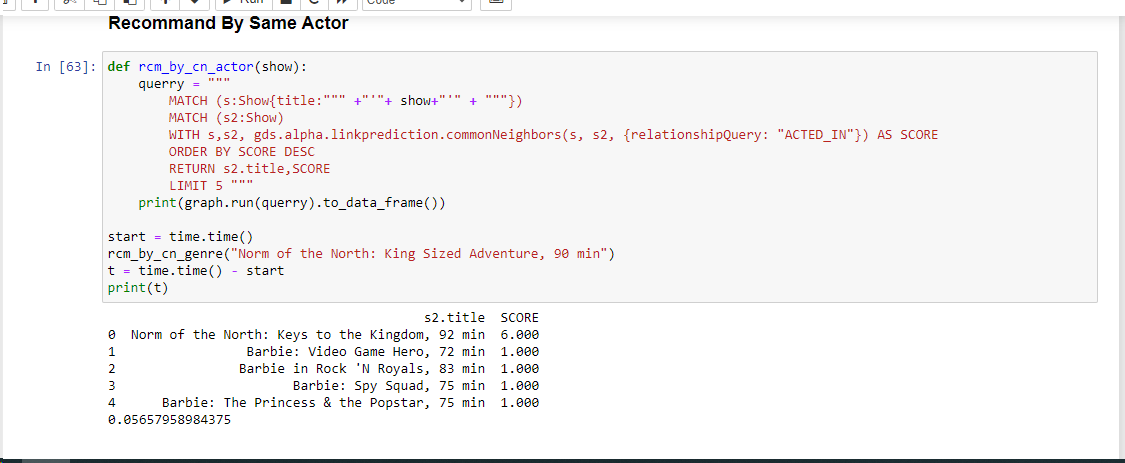


- Thời gian chạy là: 0.66s

- Nhận xét: Dự đoán đạt độ chính xác là 100% vì dễ thấy đây là liên kết Show-Genre-Show. Như vây, 2 Show có chung thể loại tức là sẽ có chung kết nối với nút Genre qua mối quan hệ IN\_GENRE.

### 2.2 Gợi ý các Show có chung diễn viên với một Show bằng Common Neighbors ( Dự đoán liên kết SAME\_ACTOR)

- Gợi ý ra 5 sản phẩm chung nhiều diễn viên nhất với sản phẩm cho trước bằng các tạo ra hàm rcm\_by\_cn\_actor(show) với tên tiêu đề sản phẩm cho trước được truyền qua biến Show:

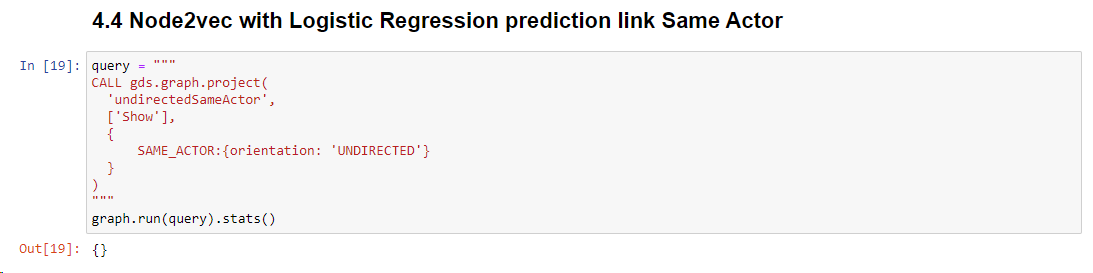


- Thời gian chạy là: 0.0566s

- Nhận xét: Dự đoán đạt độ chính xác là 100% vì dễ thấy đây là liên kết Show-Actor-Show. Như vậy, 2 Show có chung diễn tức là sẽ có chung kết nối với nút Actor qua mối quan hệ ACTED\_IN.

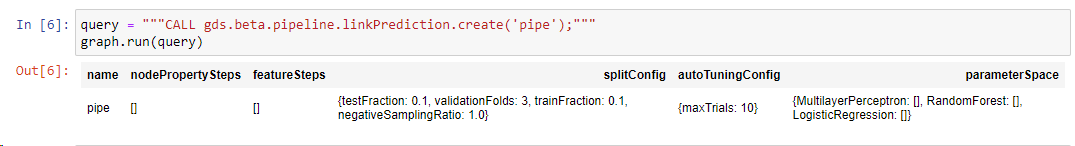
## **3. Sử dụng Node2vec và Logistic Regression để dự đoán liên kết SAME\_ACTOR**

### 3.1 Tạo ra đồ thị con undirectedSameActor với nút nhãn Show và mối quan hệ SAME\_ACTOR

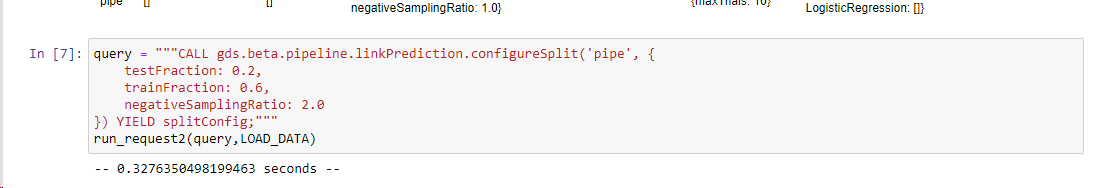


### 3.2 Dự đoán liên kết SAME\_ACTOR trên đồ thị con undirectedSameActor

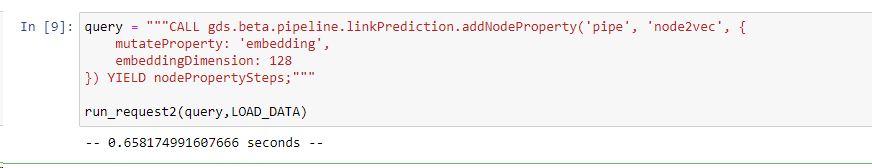
- Khởi tạo mô hình link prediction:



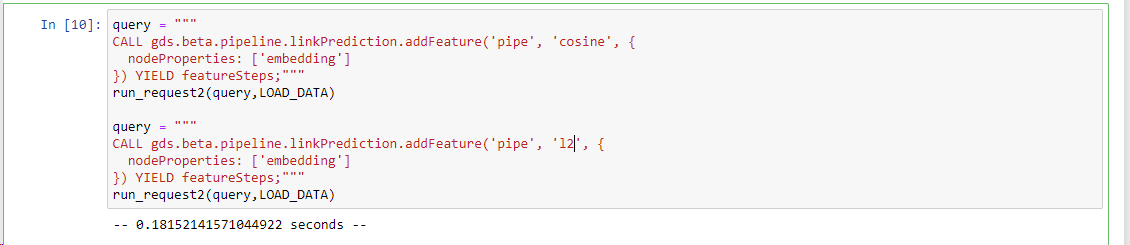
- Phân chia dữ liệu với 60% lấy làm tập huấn luyện, 20% làm tập kiểm tra với tỉ lệ lấy mẫu âm là 2, tức là với một cặp nút tồn lại liên kết thì ta lấy 2 cặp không tồn tại liên kết:



- Sử dụng node2vec để ánh xạ các nút Show sang vector biểu diễn qua thuộc tính nút là embedding: thời gian chạy 0.658s



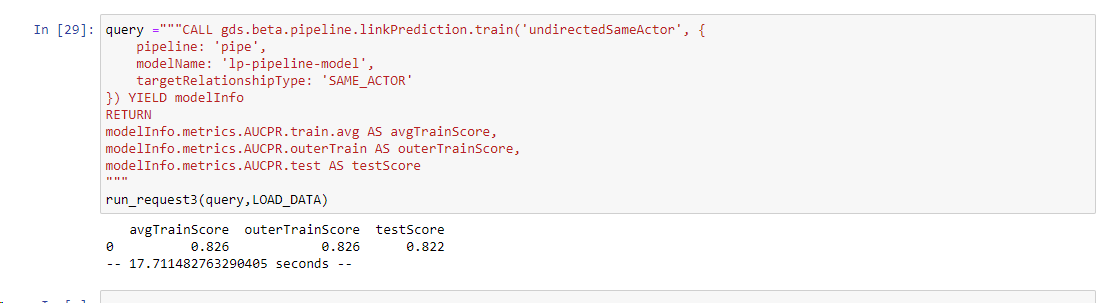
- Thêm độ đo L2 và Cosine giữa các cặp nút qua các thuộc tính các nút Show làm thuộc tính của mỗi cặp nút để sử dụng cho dự đoán liên kết giữa các cặp nút: thời gian chạy là 0.1815s



- Thêm mô hình Logistic Regression vào mô hình dự đoán liên kết: Với penalty là 0.01 tức khi giá trị hàm loss đạt 0.01 thì dừng thuật toán và patience: 2 là khi 2 lần cập nhật liên tiếp không có sự thay đổi đáng kể thì bỏ qua.



- Đào tạo mô hình và đánh giá mô hình: thời gian huấn luyện là 17.7s và độ đo AUCPR đạt 0.826 với tập training, 0.822 với tập test và 0.826 với tập dữ liệu khác với tập training.



- Nhận xét: Mô hình có hiệu suất dự đoán liên kết SAME\_ACTOR khá tốt với điển AUCPR đạt trên 0.82.

Source code của báo cáo: <https://github.com/OVmath/Link-prediction>

# *Nguồn tham khảo:*

[1] Tài liệu về node2vec: <https://arxiv.org/pdf/1607.00653.pdf>

[2] <https://neo4j.com/docs/graph-data-science/current/>

[3] <https://machinelearningcoban.com/2017/01/27/logisticregression/>

[4] <https://www.noron.vn/post/tim-hieu-chi-tiet-ve-auc---roc-trong-machine-learning-1fz9nhqo5ut>

[5] <https://machinelearningcoban.com/2017/08/31/evaluation/>

[6] <https://graphacademy.neo4j.com/courses/graph-data-science-fundamentals/2-machine-learning-procedures/4-link-prediction/>

[7] <https://ericdongyx.github.io/papers/KDD17-dong-chawla-swami-metapath2vec.pdf>

[8] <https://www.kaggle.com/code/alexanderborrayo/relationship-prediction-in-neo4j-graph-analysis>

[9] <https://www.kaggle.com/code/himj26/recommedations-using-node2vec>