**MỤC LỤC**

**NHẬN XÉT CỦA GIẢNG VIÊN**

**BẢNG PHÂN CÔNG CÔNG VIỆC**

**CHƯƠNG I: TỔNG QUAN VỀ ĐỀ ÁN**

* 1. Lý do chọn đề tài
  2. Mục tiêu đồ án
  3. Mô tả tập dữ liệu

**CHƯƠNG 2: TIỀN XỬ LÝ DỮ LIỆU**

2.1. Làm sạch dữ liệu

2.1.1. Xóa dữ liệu không hợp lệ

2.1.2. Chuẩn hóa dữ liệu

2.1.3. Xử lý dữ liệu thiếu

**CHƯƠNG 3: PHÂN TÍCH DỮ LIỆU**

**3.2. Tìm hiểu mối quan hệ giữa Metascore và IMDB Rating**

3.2.1. Phân tích hồi quy tuyến tính đơn biến (Linear Regression with One Variable)

3.2.2. Phân tích hồi quy đa thức (Polynomial Regression)

3.2.3. Phân tích sử dụng Cây quyết định (Decision Tree)

3.2.4. Phân tích sử dụng Rừng ngẫu nhiên (Random Forest)

3.2.5. Phân tích sử dụng K láng giềng gần nhất (K-Nearest Neighbors)

3.2.6. Phân tích sử dụng Máy vector hỗ trợ (Support Vector Regression)

3.2.7. Kết luận quá trình phân tích mối quan hệ Metascore - Rating

**3.3. Tìm hiểu mối quan hệ giữa Votes và Rating**

3.3.1. Phân tích hồi quy tuyến tính đơn biến (Linear Regression with One Variable)

3.3.2. Phân tích hồi quy đa thức (Polynomial Regression)

3.3.3. Phân tích sử dụng Cây quyết định (Decision Tree)

3.3.4. Phân tích sử dụng Rừng ngẫu nhiên (Random Forest)

3.3.5. Phân tích sử dụng K láng giềng gần nhất (K-Nearest Neighbors)

3.3.6. Phân tích sử dụng Máy vector hỗ trợ (Support Vector Regression)

3.3.7. Kết luận quá trình phân tích mối quan hệ Votes - Rating

**CHƯƠNG 4: XÂY DỰNG MÔ HÌNH DỰ RATING PHIM**

4.1. Tổng quát về tập dữ liệu sử dụng

4.2. Thuật toán C4.5

4.3. Triển khai mô hình dự báo thời tiết

4.3.1. Thuật toán ID3

4.3.2. Thuật toán CART

**KẾT LUẬN**

5.1. Kết quả đạt được

5.3. Hướng phát triển trong tương lai

**DANH MỤC TÀI LIỆU THAM KHẢO**

**DANH MỤC HÌNH ẢNH**

**LỜI CẢM ƠN**

Chúng em xin gửi lời cảm ơn chân thành đến giảng viên hướng dẫn – thầy Phan Thành Huấn, là giảng viên bộ môn Phân tích dữ liệu, người đã tận tình hướng dẫn, chỉ bảo nhóm của chúng em trong suốt quá trình làm đồ án môn học.

Chúng em cũng xin gửi lời cảm ơn chân thành nhất đến các thầy cô trong trường Đại học Sài Gòn nói chung, các thầy cô trong khoa Công nghệ thông tin nói riêng đã dạy dỗ cho chúng em kiến thức về các môn đại cương cũng như các môn chuyên ngành, giúp chúng em có được cơ sở lý thuyết vững vàng và tạo điều kiện giúp đỡ chúng em trong suốt quá trình học tập.

Với điều kiện về thời gian cũng như lượng kiến thức, kinh nghiệm còn hạn chế, đồ án môn học của nhóm em không thể tránh được những thiếu sót. Chúng em rất mong nhận được sự chỉ bảo, đóng góp ý kiến của thầy để nhóm chúng em có điều kiện bổ sung, nâng cao ý thức của mình, hoàn thành đồ án môn học tốt hơn.

Nhóm chúng em xin chân thành cảm ơn!

Thành phố Hồ Chí Minh, ngày 24 tháng 12 năm 2023

**NHẬN XÉT CỦA GIẢNG VIÊN**

|  |
| --- |
| TP.HCM, ngày.....tháng.....năm 2023  Giảng viên hướng dẫn  Phan Thành Huấn |

**BẢNG PHÂN CÔNG CÔNG VIỆC**

| **Thành viên - MSSV** | **Công việc** | **Phần trăm đóng góp** |
| --- | --- | --- |
|  |  |  |
|  |  |  |

**CHƯƠNG 1: TỔNG QUAN VỀ ĐỒ ÁN**

1. **Lý do chọn đề tài**

Trong bối cảnh công nghệ số phát triển mạnh mẽ, nhu cầu giải trí thông qua các nền tảng xem phim và nội dung số tăng cao. Hàng triệu bộ phim và chương trình mới liên tục ra mắt, mang đến cho người xem nhiều lựa chọn phong phú. Tuy nhiên, chính sự phong phú này lại gây khó khăn cho người dùng trong việc tìm kiếm những bộ phim thực sự phù hợp với sở thích và gu thưởng thức của mình. Để giải quyết vấn đề này, việc xây dựng các mô hình dự đoán xếp hạng phim có thể là một giải pháp hiệu quả.

Dự đoán xếp hạng phim không chỉ giúp người xem dễ dàng tìm được những bộ phim ưng ý, mà còn nâng cao trải nghiệm giải trí cá nhân hóa cho họ. Bằng cách đưa ra các gợi ý dựa trên sở thích và lịch sử xem phim, các mô hình này có thể hỗ trợ người dùng tiết kiệm thời gian tìm kiếm và khám phá nội dung mới, từ đó tối ưu hóa sự hài lòng trong trải nghiệm xem phim.

Đối với các nền tảng trực tuyến, dự đoán xếp hạng phim còn giúp tối ưu hóa các đề xuất cho người dùng. Việc này không chỉ làm tăng mức độ tương tác của người dùng, mà còn tạo điều kiện để các nền tảng này nâng cao chất lượng dịch vụ, giữ chân người dùng lâu hơn và tăng trưởng doanh thu. Các nhà sản xuất và phân phối phim cũng có thể tận dụng dữ liệu và mô hình dự đoán để nắm bắt xu hướng và thị hiếu của khán giả, từ đó đưa ra các quyết định về đầu tư nội dung và quảng bá một cách chiến lược, hiệu quả hơn.

Với những lợi ích thiết thực và tiềm năng ứng dụng rộng lớn, đề tài “Dự đoán xếp hạng phim” không chỉ thú vị mà còn có ý nghĩa thực tiễn cao. Thông qua dự án, chúng em mong muốn cung cấp một cái nhìn tổng quan về các yếu tố tác động đến xếp hạng của một bộ phim, từ nội dung, thể loại, đến tên tuổi diễn viên, ngày phát hành và cả đánh giá từ khán giả, đồng thời cũng tìm hiểu mối quan hệ giữa các thuộc tính với nhau. Qua đó, chúng em hi vọng giúp người xem hiểu rõ hơn về cách thức xếp hạng và ảnh hưởng của các yếu tố này đến sự thành công của phim.

Đồng thời,dự đoán rating phim cung cấp thông tin hữu ích cho các nhà làm phim, giúp họ hiểu sâu hơn về mối quan hệ giữa các yếu tố sáng tạo và thành công thương mại, từ đó có những định hướng sản xuất phim phù hợp với thị hiếu công chúng. Việc hiểu rõ thị hiếu của khán giả thông qua dữ liệu có thể giúp các nhà sản xuất đưa ra những quyết định sáng tạo và chiến lược kinh doanh hiệu quả hơn, tối đa hóa khả năng thành công của mỗi dự án phim.

1. **Mục tiêu**

Tìm hiểu mối quan hệ giữa các thuộc tính

Phân tích và khám phá các mối quan hệ giữa các yếu tố như thể loại, đạo diễn, diễn viên, năm phát hành, và các yếu tố khác ảnh hưởng đến rating của phim. Việc hiểu rõ mối quan hệ này sẽ giúp xác định các yếu tố có tác động lớn đến sự thành công và mức độ yêu thích của bộ phim.

Áp dụng các mô hình dự đoán khác nhau.

Đưa ra mô hình dự đoán chính xác về rating của phim: Xây dựng các mô hình học máy dự đoán rating phim .

1. **Mô tả tập dữ liêu**

**CHƯƠNG 2: TIỀN XỬ LÝ DỮ LIỆU**

Dựa vào tập dữ liệu đã mô tả ở trên, tiến hành bước tiền xử lý dữ liệu. Giai đoạn tiền xử lý dữ liệu là một bước quan trọng trong quá trình phân tích dữ liệu - thu thập, chuẩn hóa, lọc, biến đổi và tạo dữ liệu để chuẩn bị cho các bước xử lý và phân tích tiếp theo.

**2.1. Làm sạch dữ liệu**

**2.1.1. Xóa dữ liệu không hợp lệ**

Chúng ta sẽ bắt đầu với việc tìm ra các dữ liệu hợp lệ bằng cách định dạng chúng và tìm ra các dữ liệu khác với định dạng. VD: Release Year (Int), Duration (Int), IMBD Rating (Float), Metascore (Int), Vote(int), Gross( có dạng $xx.xxM)

Vậy nên ta sẽ có hà m kiểm tra và tìm ra các cột có dữ liệu bị sai.

|  |
| --- |
| # Hàm kiểm tra dữ liệu không đúng  def detect\_invalid\_data(df):  invalid\_data = {}  # Kiểm tra cột Release Year  invalid\_year = df[~df['Release Year'].apply(lambda x: isinstance(x, int) or (isinstance(x, str) and x.isdigit()))]  if not invalid\_year.empty:  invalid\_data['Release Year'] = invalid\_year[['Release Year']]  # Kiểm tra cột Duration  invalid\_duration = df[~df['Duration'].apply(lambda x: isinstance(x, int) or (isinstance(x, str) and x.isdigit()))]  if not invalid\_duration.empty:  invalid\_data['Duration'] = invalid\_duration[['Duration']]  # Kiểm tra cột IMDB Rating  invalid\_rating = df[~df['IMDB Rating'].apply(lambda x: isinstance(x, float) or (isinstance(x, str) and x.replace('.', '', 1).isdigit()))]  if not invalid\_rating.empty:  invalid\_data['IMDB Rating'] = invalid\_rating[['IMDB Rating']]  # Kiểm tra cột Metascore  invalid\_metascore = df[~df['Metascore'].apply(lambda x: isinstance(x, int) or (isinstance(x, float) and x.is\_integer()) or (isinstance(x, str) and x.replace('.', '', 1).isdigit() and float(x).is\_integer()))]  if not invalid\_metascore.empty:  invalid\_data['Metascore'] = invalid\_metascore[['Metascore']]  # Kiểm tra cột Votes  invalid\_vote = df[~df['Votes'].apply(lambda x: isinstance(x, str) and x.replace(',', '').isdigit())]  if not invalid\_vote.empty:  invalid\_data['Votes'] = invalid\_vote[['Votes']]  # Kiểm tra cột Gross  invalid\_gross = df[~df['Gross'].apply(lambda x: isinstance(x, str) and x.startswith('$') and x.endswith('M') and x[1:-1].replace('.', '', 1).isdigit())]  if not invalid\_gross.empty:  invalid\_data['Gross'] = invalid\_gross[['Gross']]  return invalid\_data |

Đầu tiên, chúng tôi tiến hành kiểm tra và loại bỏ các giá trị không hợp lệ trong cột Release Year.

Invalid data in column Release Year:

Release Year

1964â1968

I 1987

I 1990

I 1995

I 1996

1995â2016

I 1995

I 1999

1998 Video

I 1998

I 1999

I 1998

I 1999

I 2000

1998 TV Movie

1999 TV Movie

2001 TV Movie

I 2003

I 2002

I 2004

I 2003

I 2004

I 2004

I 2004

I 2004

2005 Video

I 2005

I 2005

I 2005

II 2004

I 2005

2006 TV Movie

I 2006

Có khá nhiều dữ liệu bị sai, cũng như không đúng với kiểu dữ liệu chúng ta cần để xử lí.

Các giá trị không phải số trong cột này được chuyển đổi thành NaN (Not a Number) bằng cách sử dụng phương thức pd.to\_numeric với tham số errors='coerce'. Sau đó, các giá trị NaN được điền lại bằng giá trị trung vị của cột Release Year để đảm bảo tính nhất quán của dữ liệu.

df['Release Year'] = pd.to\_numeric(df['Release Year'], errors='coerce')

df['Release Year'].fillna(df['Release Year'].median(), inplace=True)

**2.1.2. Chuẩn hóa dữ liệu**

Tiếp theo, chúng tôi thực hiện chuẩn hóa dữ liệu để đảm bảo các giá trị trong các cột có cùng đơn vị và định dạng nhất quán.

* **Cột Gross**: Loại bỏ ký hiệu $ và M, chuyển đổi thành số thập phân, sau đó chuyển đổi từ triệu đô la sang đô la (nếu cần) và chuẩn hóa lại thành đơn vị triệu USD.

|  |
| --- |
| Gross |
| $134.97M |
| $57.30M |
| $54.80M |
| $44.82M |
| NA |
| $132.09M |
| $5.32M |
| $45.30M |
| NA |
| $42.77M |
| $32.00M |
| $4.36M |
| $10.46M |
| $44.91M  … |

|  |
| --- |
| Votes |
| 2,002,655 |
| 1,358,608 |
| 56,476 |
| 313,044 |
| 64,331 |
| 216,050 |
| 348,110 |
| 78,858 |
| 234,034 |
| 59,119 |
| 715,718 |
| 146,211 |
| 106,938 |

* **Cột Votes**: Loại bỏ dấu phẩy và chuyển đổi thành số nguyên, sau đó chuẩn hóa thành đơn vị triệu phiếu.

# Loại bỏ ký hiệu $ và M, chuyển thành số thập phân

df['Gross'] = df['Gross'].str.replace('[\$,M]', '', regex=True).astype(float)

# Chuyển đổi từ triệu đô la sang đô la (nếu cần)

df['Gross'] = df['Gross'] \* 1e6

# Chuẩn hóa cột Gross (đơn vị triệu USD)

df['Gross'] = df['Gross'] / 1e6

# Chuyển đổi cột Votes thành số

df['Votes'] = df['Votes'].replace(',', '', regex=True).astype(int)

# Chuẩn hóa cột Votes (đơn vị triệu phiếu)

df['Votes'] = df['Votes'] / 1e6

# Chuyển đổi các cột phân loại thành số (nếu cần)

df = pd.get\_dummies(df, columns=['Genre', 'Director', 'Cast'], drop\_first=True)

**2.1.3. Xử lý các dữ liệu còn thiếu**

Cuối cùng, chúng tôi kiểm tra và xử lý các giá trị còn thiếu trong các cột dữ liệu.

* **Cột Gross**: Điền các giá trị NaN bằng giá trị trung vị của cột.
* **Cột Metascore**: Điền các giá trị NaN bằng giá trị trung bình đã làm tròn và chuyển đổi thành số nguyên.

# Xử lý giá trị NaN

df['Gross'].fillna(df['Gross'].median(), inplace=True)

print(df.isnull().sum())

# Fill cột Metascore bằng int thay vì float

df['Metascore'].fillna(int(round(df['Metascore'].mean())), inplace=True)

df['Metascore'] = df['Metascore'].astype(int)

Báo cáo này mô tả chi tiết các bước xử lý dữ liệu bao gồm xóa dữ liệu không hợp lệ, chuẩn hóa dữ liệu và xử lý các dữ liệu còn thiếu. Nếu bạn cần thêm bất kỳ thông tin hoặc hỗ trợ nào khác, hãy cho mình biết nhé!

**CHƯƠNG 3: PHÂN TÍCH DỮ LIỆU**

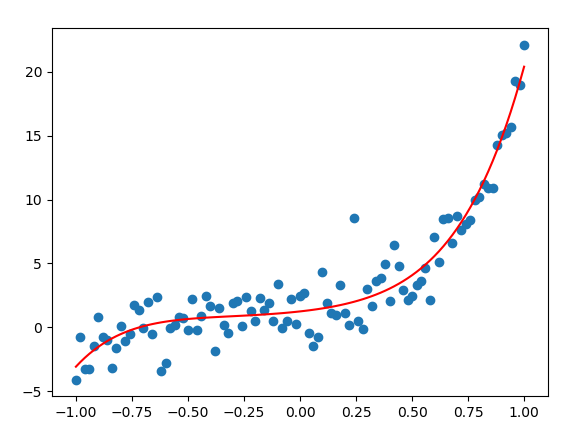
**3.1. Các phương pháp sử dụng**

* **Hồi quy tuyến tính (Linear Regression)**: Hồi quy tuyến tính là một phương pháp thống kê để hồi quy dữ liệu với biến phụ thuộc có giá trị liên tục trong khi các biến độc lập có thể có một trong hai giá trị liên tục hoặc là giá trị phân loại. Nói cách khác, hồi quy tuyến tính là một phương pháp để dự đoán biến phụ thuộc (Y) dựa trên giá trị của biến độc lập (X). Nó có thể được sử dụng cho các trường hợp chúng ta muốn dự đoán một số lượng liên tục.

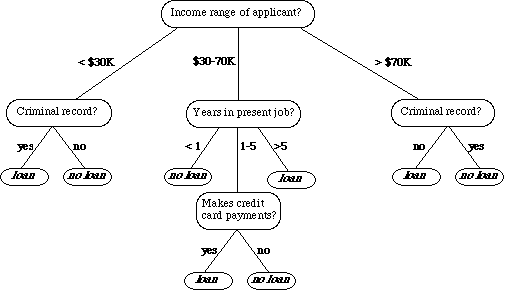
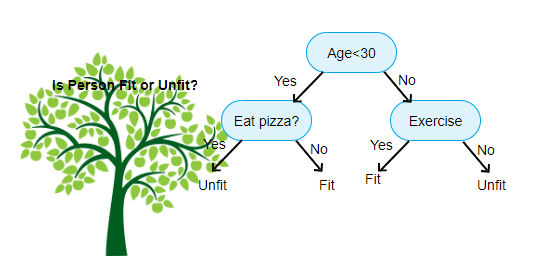
A graph with blue dots and a red line

Description automatically generated

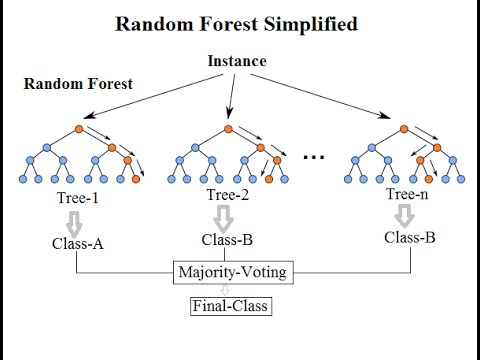
* **Hồi quy đa thức (Polynomial Regression)**: Hồi quy đa thức là thuật toán hồi quy đa thức, nó giống như thuật toán hồi quy tuyến tính, sử dụng mối quan hệ giữa các biến độc lập x và biến phụ thuộc y được biểu diễn dưới dạng đa thức bậc n, để tìm cách tốt nhất vẽ một đường qua các điểm dữ liệu sao cho tối ưu và phù hợp nhất.



* **Cây quyết định (Decision Tree)**: Cây quyết định là một kiểu mô hình dự báo, nghĩa là một ánh xạ từ các quan sát về một sự vật/hiện tượng tới các kết luận về giá trị mục tiêu của sự vật/hiện tượng. Mỗi một nút trong (internal node) tương ứng với một biến; đường nối giữa nó với nút con của nó thể hiện một giá trị cụ thể cho biến đó. Mỗi nút lá đại diện cho giá trị dự đoán của biến mục tiêu, cho trước các giá trị của các biến được biểu diễn bởi đường đi từ nút gốc tới nút lá đó.



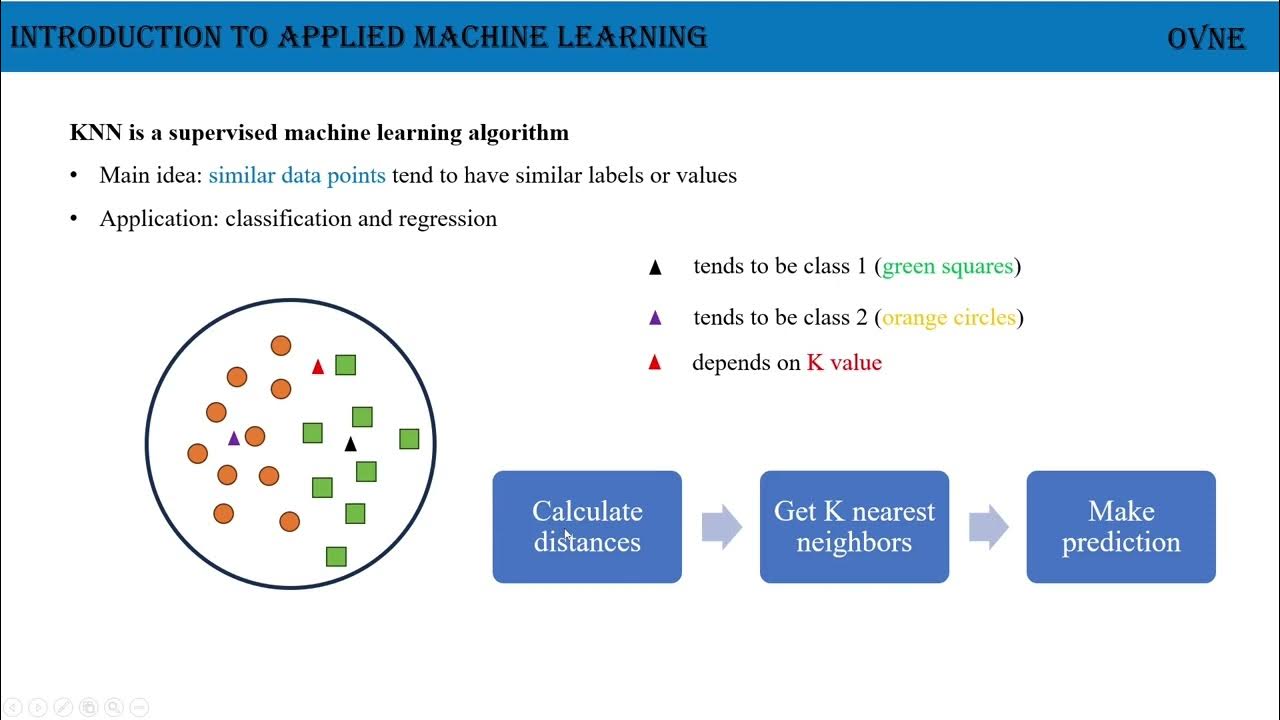
* **Rừng ngẫu nhiên (Random Forest)**: Rừng ngẫu nhiên hoặc rừng quyết định ngẫu nhiên là một phương pháp học tập tổng hợp để phân loại, hồi quy và các nhiệm vụ khác hoạt động bằng cách xây dựng vô số cây quyết định tại thời điểm đào tạo. Đối với các nhiệm vụ phân loại, đầu ra của rừng ngẫu nhiên là lớp được hầu hết các cây lựa chọn.





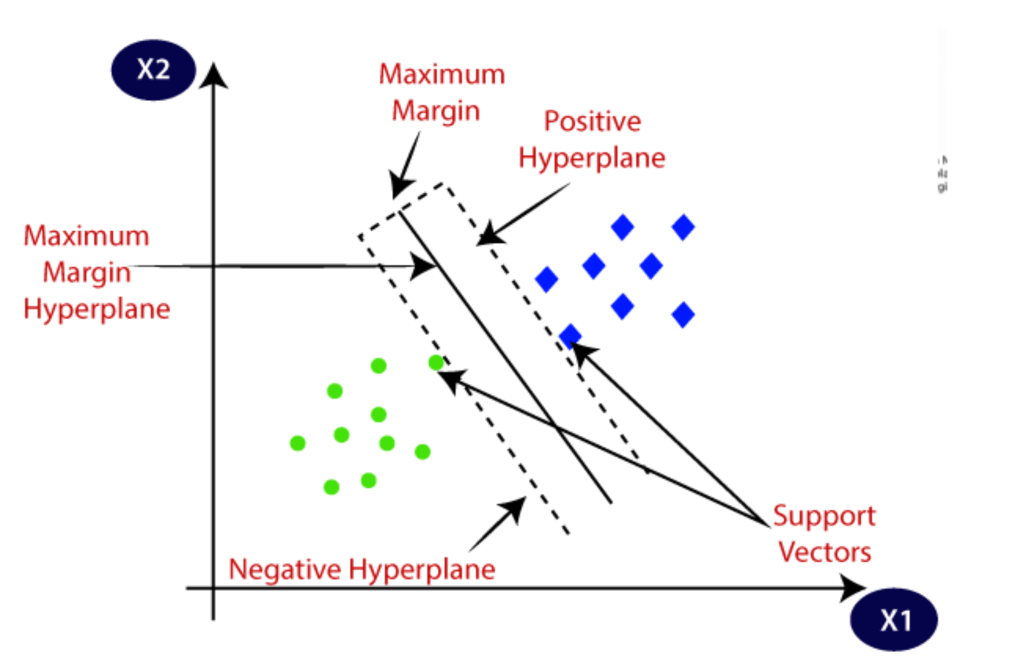
* **K láng giềng gần nhất (K-Nearest Neighbors)**

K-Nearest Neighbors (KNN) là một thuật toán học máy đơn giản nhưng mạnh mẽ, được sử dụng rộng rãi trong các bài toán phân loại và hồi quy. Nguyên lý cơ bản của KNN là dựa trên khoảng cách giữa các điểm dữ liệu để dự đoán giá trị của một điểm mới. Cụ thể, khi cần dự đoán giá trị của một điểm dữ liệu mới, KNN sẽ tìm k điểm dữ liệu gần nhất trong tập huấn luyện và sử dụng giá trị của chúng để đưa ra dự đoán.



* **Máy vector hỗ trợ (Support Vector Machine)**

Support Vector Machine (SVM) là một thuật toán học máy mạnh mẽ, được sử dụng chủ yếu cho các bài toán phân loại, nhưng cũng có thể áp dụng cho hồi quy. SVM hoạt động bằng cách tìm kiếm một siêu phẳng (hyperplane) trong không gian nhiều chiều để phân tách các điểm dữ liệu thuộc các lớp khác nhau. Mục tiêu của SVM là tìm ra siêu phẳng có khoảng cách lớn nhất đến các điểm dữ liệu gần nhất của mỗi lớp, được gọi là các vector hỗ trợ (support vectors).



Dưới đây là phần 3.2 đã được viết lại với tiêu đề "Tìm hiểu mối quan hệ giữa Metascore và Rating":

**3.2. Tìm hiểu mối quan hệ giữa Metascore và Rating**

Nghiên cứu mối quan hệ giữa Metascore và Rating nhằm hiểu rõ hơn về tương tác giữa hai yếu tố này trong ngành điện ảnh. Mục tiêu là phân tích cách Metascore có thể biến đổi dựa trên các biến động của Rating và ngược lại. Thông qua việc đánh giá mối quan hệ này, chúng ta có thể có cái nhìn sâu sắc hơn về cách Metascore ảnh hưởng đến sự đánh giá của khán giả và ngược lại. Điều này có thể giúp chúng ta hiểu rõ hơn về đặc điểm của các bộ phim và đồng thời cung cấp thông tin quan trọng cho việc dự đoán và hiểu biến đổi của Rating trong tương lai.

**Bước 1: Chuẩn bị dữ liệu**

Đầu tiên, chúng ta cần chuẩn bị dữ liệu bằng cách chọn các cột liên quan và chia dữ liệu thành tập huấn luyện và tập kiểm tra. Đây là một phần quan trọng trong quá trình xây dựng mô hình máy học, giúp đảm bảo rằng mô hình có thể tổng quát hóa tốt trên dữ liệu mới.

from sklearn.model\_selection import train\_test\_split

import numpy as np

metascore\_rating\_df = movie\_data[['Rating', 'Metascore']]

metascore\_rating\_df = metascore\_rating\_df[:20000]

X = np.array(metascore\_rating\_df['Metascore']).reshape(-1, 1)

y = np.array(metascore\_rating\_df['Rating']).reshape(-1, 1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, shuffle=True, random\_state=1)

* **train\_test\_split**: Hàm này từ thư viện sklearn.model\_selection được sử dụng để chia dữ liệu thành hai tập: tập huấn luyện và tập kiểm tra.
* **metascore\_rating\_df**: DataFrame chứa hai cột 'Rating' và 'Metascore' từ dữ liệu gốc.
* **X**: Mảng numpy chứa giá trị của cột 'Metascore', được định hình lại thành một mảng 2D.
* **y**: Mảng numpy chứa giá trị của cột 'Rating', được định hình lại thành một mảng 2D.
* **X\_train, X\_test, y\_train, y\_test**: Các tập dữ liệu được chia ra, với 80% dữ liệu dùng để huấn luyện và 20% dữ liệu dùng để kiểm tra.

**Bước 2: Vẽ biểu đồ phân tán**

Biểu đồ phân tán giúp chúng ta trực quan hóa mối quan hệ giữa Metascore và Rating. Với mỗi điểm dữ liệu biểu diễn một quan sát từ tập dữ liệu, với trục x là giá trị Metascore và trục y là giá trị Rating.

import matplotlib.pyplot as plt

plt.scatter(X, y)

plt.xlabel('Metascore')

plt.ylabel('Rating')

plt.title('Scatter Plot: Metascore vs Rating')

plt.show()

**3.3.1. Phân tích hồi quy tuyến tính đơn biến (Linear Regression with One Variable)**

**Giới thiệu**

Hồi quy tuyến tính đơn biến là một phương pháp thống kê được sử dụng để dự đoán giá trị của một biến phụ thuộc dựa trên giá trị của một biến độc lập. Dạng cơ bản của mô hình hồi quy đơn biến có một biến độc lập và một biến phụ thuộc: Y=B0+B1X*Y*=*B*0+*B*1*X*. X là biến độc lập, Y là biến phụ thuộc. B0 và B1 là các hệ số.

**Bước 1: Tạo và huấn luyện mô hình hồi quy tuyến tính**

from sklearn.linear\_model import LinearRegression

lin\_reg = LinearRegression()

lin\_reg.fit(X\_train, y\_train)

* **LinearRegression**: Lớp này từ thư viện sklearn.linear\_model được sử dụng để tạo mô hình hồi quy tuyến tính.
* **lin\_reg**: Đối tượng của lớp LinearRegression.
* **fit**: Phương thức này được sử dụng để huấn luyện mô hình hồi quy tuyến tính với dữ liệu huấn luyện X\_train và y\_train.

**Bước 2: Dự đoán giá trị của biến phụ thuộc**

y\_pred = lin\_reg.predict(X\_test)

* **predict**: Phương thức này được sử dụng để dự đoán giá trị của biến phụ thuộc y dựa trên dữ liệu kiểm tra X\_test.

**Bước 3: Vẽ biểu đồ hồi quy tuyến tính**

import seaborn as sns

def plot\_regression():

sns.set(font\_scale=2)

plt.figure(figsize=(15, 15))

plt.scatter(X\_test, y\_test, color='blue', label='Dữ liệu thực tế', s=25)

plt.plot(X\_test, y\_pred, color='red', linewidth=6, label='Đường hồi quy')

plt.xlabel('Metascore', fontsize=25)

plt.ylabel('Rating', fontsize=25)

plt.title('Mối quan hệ giữa Metascore và Rating', fontsize=25)

plt.legend()

plt.show()

plot\_regression()

* **seaborn**: Thư viện này được sử dụng để tạo các biểu đồ thống kê.
* **matplotlib.pyplot**: Thư viện này được sử dụng để vẽ các biểu đồ.
* **plot\_regression**: Hàm này tạo biểu đồ phân tán của dữ liệu thực tế và đường hồi quy tuyến tính.
* **plt.scatter**: Hàm này vẽ các điểm dữ liệu thực tế.
* **plt.plot**: Hàm này vẽ đường hồi quy tuyến tính dựa trên các giá trị dự đoán.

**Bước 4: Tính toán các chỉ số đánh giá hiệu suất**

from sklearn.metrics import mean\_squared\_error, r2\_score

print("Hệ số hồi quy:", lin\_reg.coef\_)

print("Sai số:", lin\_reg.intercept\_)

print("MSE:", mean\_squared\_error(y\_test, y\_pred))

print("R2 Score:", r2\_score(y\_test, y\_pred))

* **mean\_squared\_error**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán sai số bình phương trung bình (MSE).
* **r2\_score**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán hệ số xác định (R2 Score).
* **coef\_**: Thuộc tính này của đối tượng lin\_reg chứa các hệ số hồi quy.
* **intercept\_**: Thuộc tính này của đối tượng lin\_reg chứa giá trị sai số.

**Kết quả:**

**A graph with blue dots and a red line

Description automatically generated**

Linear Regression - MSE: 0.5500833265888815

Linear Regression - R2 Score: 0.46589392261015816

**Định nghĩa các chỉ số đánh giá**

* **Mean Squared Error (MSE)**: Đây là chỉ số đo lường mức độ chênh lệch giữa giá trị thực tế và giá trị dự đoán của mô hình. MSE càng thấp, mô hình càng chính xác.
* **R2 Score**: Đây là chỉ số đo lường mức độ giải thích của mô hình đối với biến phụ thuộc. R2 Score càng gần 1, mô hình càng tốt trong việc giải thích sự biến động của biến phụ thuộc.

Mô hình hồi quy tuyến tính đơn biến cho thấy một mức độ giải thích khá tốt với R2 Score là 0.4659. Điều này cho thấy rằng khoảng 46.59% sự biến động của Rating có thể được giải thích bởi Metascore.

**3.3.2. Phân tích hồi quy đa thức (Polynomial Regression)**

**Giới thiệu**

Hồi quy đa thức là một phương pháp mở rộng của hồi quy tuyến tính, trong đó mối quan hệ giữa biến độc lập và biến phụ thuộc được mô hình hóa dưới dạng đa thức. Dạng cơ bản của mô hình hồi quy đa thức có một biến độc lập và một biến phụ thuộc như sau: Y=b0+b1x+b2x2+…+bnxn*Y*=*b*0+*b*1*x*+*b*2*x*2+…+*bnxn*. Khi tăng giá trị của degree, mô hình có thể biểu diễn mối quan hệ phức tạp hơn giữa biến độc lập và biến phụ thuộc. Tuy nhiên, cũng cần lưu ý rằng việc sử dụng các đa thức có bậc cao có thể dẫn đến hiện tượng overfitting, đặc biệt là khi dữ liệu không đủ lớn.

**Bước 1: Tạo đa thức cho dữ liệu huấn luyện**

from sklearn.preprocessing import PolynomialFeatures

poly\_df = PolynomialFeatures(degree=3)

transform\_poly\_train = poly\_df.fit\_transform(X\_train)

* **PolynomialFeatures**: Lớp này từ thư viện sklearn.preprocessing được sử dụng để tạo các đặc trưng đa thức từ dữ liệu gốc.
* **poly\_df**: Đối tượng của lớp PolynomialFeatures với bậc đa thức là 3.
* **fit\_transform**: Phương thức này được sử dụng để biến đổi dữ liệu huấn luyện X\_train thành các đặc trưng đa thức.

**Bước 2: Tạo và huấn luyện mô hình hồi quy đa thức**

linreg2 = LinearRegression()

linreg2.fit(transform\_poly\_train, y\_train)

* **linreg2**: Đối tượng của lớp LinearRegression được sử dụng để tạo mô hình hồi quy tuyến tính trên các đặc trưng đa thức.
* **fit**: Phương thức này được sử dụng để huấn luyện mô hình hồi quy đa thức với dữ liệu huấn luyện đã được biến đổi transform\_poly\_train và y\_train.

**Bước 3: Biến đổi dữ liệu kiểm tra và dự đoán giá trị**

transform\_poly\_test = poly\_df.transform(X\_test)

polynomial\_predict\_test = linreg2.predict(transform\_poly\_test)

* **transform**: Phương thức này được sử dụng để biến đổi dữ liệu kiểm tra X\_test thành các đặc trưng đa thức.
* **predict**: Phương thức này được sử dụng để dự đoán giá trị của biến phụ thuộc y dựa trên dữ liệu kiểm tra đã được biến đổi transform\_poly\_test.

**Bước 4: Tính toán các chỉ số đánh giá hiệu suất**

from sklearn.metrics import mean\_squared\_error, r2\_score

print("Hệ số hồi quy:", linreg2.coef\_)

print("Sai số:", linreg2.intercept\_)

print("MSE:", mean\_squared\_error(y\_test, polynomial\_predict\_test))

print("R2 Score (Polynomial Regression):", r2\_score(y\_test, polynomial\_predict\_test))

* **mean\_squared\_error**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán sai số bình phương trung bình (MSE).
* **r2\_score**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán hệ số xác định (R2 Score).
* **coef\_**: Thuộc tính này của đối tượng linreg2 chứa các hệ số hồi quy.
* **intercept\_**: Thuộc tính này của đối tượng linreg2 chứa giá trị sai số.

**Kết Quả:**

**A graph of a graph showing a red line and blue dots

Description automatically generated**

Polynomial Regression - MSE: 0.5406178058755716

Polynomial Regression - R2 Score: 0.47508451591533696

* Mô hình hồi quy đa thức với bậc 2 cải thiện nhẹ so với hồi quy tuyến tính đơn biến với R2 Score là 0.4751. Điều này cho thấy rằng việc sử dụng mô hình phi tuyến tính có thể mô tả mối quan hệ giữa Metascore và Rating tốt hơn một chút.

**3.3.3. Phân tích sử dụng Cây quyết định (Decision Tree)**

**Giới thiệu**

Cây quyết định là một mô hình dự đoán sử dụng cấu trúc cây để biểu diễn các quyết định và hậu quả của chúng. Mô hình này có thể xử lý cả biến số và biến phân loại. Cây quyết định hoạt động bằng cách chia dữ liệu thành các nhóm nhỏ hơn dựa trên các điều kiện nhất định, và tiếp tục chia nhỏ cho đến khi đạt được một điều kiện dừng.

**Bước 1: Tạo và huấn luyện mô hình cây quyết định**

from sklearn.tree import DecisionTreeRegressor

tree\_regressor = DecisionTreeRegressor(random\_state=42)

tree\_regressor.fit(X\_train, y\_train)

* **DecisionTreeRegressor**: Lớp này từ thư viện sklearn.tree được sử dụng để tạo mô hình cây quyết định.
* **tree\_regressor**: Đối tượng của lớp DecisionTreeRegressor.
* **fit**: Phương thức này được sử dụng để huấn luyện mô hình cây quyết định với dữ liệu huấn luyện X\_train và y\_train.

**Bước 2: Dự đoán giá trị của biến phụ thuộc**

y\_pred\_tree = tree\_regressor.predict(X\_test)

* **predict**: Phương thức này được sử dụng để dự đoán giá trị của biến phụ thuộc y dựa trên dữ liệu kiểm tra X\_test.

**Bước 3: Tính toán các chỉ số đánh giá**

from sklearn.metrics import mean\_squared\_error, r2\_score

mse\_tree = mean\_squared\_error(y\_test, y\_pred\_tree)

r2\_tree = r2\_score(y\_test, y\_pred\_tree)

print("Decision Tree - MSE:", mse\_tree)

print("Decision Tree - R2 Score:", r2\_tree)

* **mean\_squared\_error**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán sai số bình phương trung bình (MSE).
* **r2\_score**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán hệ số xác định (R2 Score).

**Kết Quả**:

**A screen shot of a graph

Description automatically generated**

Decision Tree - MSE: 0.5809792042152714

Decision Tree - R2 Score: 0.43589542018530536

* Mô hình cây quyết định có R2 Score là 0.4359, thấp hơn so với hồi quy tuyến tính và đa thức. Điều này cho thấy rằng mô hình cây quyết định có thể không phù hợp lắm cho dữ liệu này, có thể do overfitting.

**3.3.4. Phân tích sử dụng Rừng ngẫu nhiên (Random Forest)**

**Giới thiệu**

Rừng ngẫu nhiên là một mô hình dự đoán sử dụng nhiều cây quyết định để cải thiện độ chính xác và giảm thiểu overfitting. Mô hình này hoạt động bằng cách tạo ra nhiều cây quyết định từ các mẫu ngẫu nhiên của dữ liệu huấn luyện và sau đó kết hợp các dự đoán của các cây này để đưa ra dự đoán cuối cùng.

**Bước 1: Tạo và huấn luyện mô hình rừng ngẫu nhiên**

from sklearn.ensemble import RandomForestRegressor

forest\_regressor = RandomForestRegressor(n\_estimators=100, random\_state=42)

forest\_regressor.fit(X\_train, y\_train)

* **RandomForestRegressor**: Lớp này từ thư viện sklearn.ensemble được sử dụng để tạo mô hình rừng ngẫu nhiên.
* **forest\_regressor**: Đối tượng của lớp RandomForestRegressor.
* **n\_estimators**: Tham số này xác định số lượng cây quyết định trong rừng ngẫu nhiên.
* **fit**: Phương thức này được sử dụng để huấn luyện mô hình rừng ngẫu nhiên với dữ liệu huấn luyện X\_train và y\_train.

**Bước 2: Dự đoán giá trị của biến phụ thuộc**

y\_pred\_forest = forest\_regressor.predict(X\_test)

* **predict**: Phương thức này được sử dụng để dự đoán giá trị của biến phụ thuộc y dựa trên dữ liệu kiểm tra X\_test.

**Bước 3: Tính toán các chỉ số đánh giá**

from sklearn.metrics import mean\_squared\_error, r2\_score

mse\_forest = mean\_squared\_error(y\_test, y\_pred\_forest)

r2\_forest = r2\_score(y\_test, y\_pred\_forest)

print("Random Forest - MSE:", mse\_forest)

print("Random Forest - R2 Score:", r2\_forest)

* **mean\_squared\_error**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán sai số bình phương trung bình (MSE).
* **r2\_score**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán hệ số xác định (R2 Score).

**Kết Quả**: A graph of red and blue dots

Description automatically generated

Random Forest - MSE: 0.550026434432892

Random Forest - R2 Score: 0.4659491623253087

* Mô hình rừng ngẫu nhiên có R2 Score là 0.4659, tương đương với hồi quy tuyến tính. Điều này cho thấy rằng việc sử dụng nhiều cây quyết định không cải thiện đáng kể độ chính xác của mô hình trong trường hợp này.

**3.3.5. Phân tích sử dụng K láng giềng gần nhất (K-Nearest Neighbors)**

**Giới thiệu**

K láng giềng gần nhất (KNN) là một thuật toán học máy đơn giản nhưng mạnh mẽ, được sử dụng rộng rãi trong các bài toán phân loại và hồi quy. Nguyên lý cơ bản của KNN là dựa trên khoảng cách giữa các điểm dữ liệu để dự đoán giá trị của một điểm mới. Cụ thể, khi cần dự đoán giá trị của một điểm dữ liệu mới, KNN sẽ tìm k điểm dữ liệu gần nhất trong tập huấn luyện và sử dụng giá trị của chúng để đưa ra dự đoán.

**Bước 1: Tạo và huấn luyện mô hình K láng giềng gần nhất**

from sklearn.neighbors import KNeighborsRegressor

knn\_regressor = KNeighborsRegressor(n\_neighbors=3)

knn\_regressor.fit(X\_train, y\_train)

* **KNeighborsRegressor**: Lớp này từ thư viện sklearn.neighbors được sử dụng để tạo mô hình K láng giềng gần nhất.
* **knn\_regressor**: Đối tượng của lớp KNeighborsRegressor.
* **n\_neighbors**: Tham số này xác định số lượng láng giềng gần nhất được sử dụng để dự đoán.
* **fit**: Phương thức này được sử dụng để huấn luyện mô hình K láng giềng gần nhất với dữ liệu huấn luyện X\_train và y\_train.

**Bước 2: Dự đoán giá trị của biến phụ thuộc**

y\_pred\_knn = knn\_regressor.predict(X\_test)

* **predict**: Phương thức này được sử dụng để dự đoán giá trị của biến phụ thuộc y dựa trên dữ liệu kiểm tra X\_test.

**Bước 3: Tính toán các chỉ số đánh giá**

from sklearn.metrics import mean\_squared\_error, r2\_score

mse\_knn = mean\_squared\_error(y\_test, y\_pred\_knn)

r2\_knn = r2\_score(y\_test, y\_pred\_knn)

print("K-Nearest Neighbors - MSE:", mse\_knn)

print("K-Nearest Neighbors - R2 Score:", r2\_knn)

* **mean\_squared\_error**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán sai số bình phương trung bình (MSE).
* **r2\_score**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán hệ số xác định (R2 Score).

**Kết Quả**: A diagram of red and blue dots

Description automatically generated

K-Nearest Neighbors - MSE: 0.6438777777777777

K-Nearest Neighbors - R2 Score: 0.3748237447225907

* Mô hình K láng giềng gần nhất có R2 Score là 0.3748, thấp nhất trong các mô hình được thử nghiệm. Điều này cho thấy rằng KNN không phù hợp lắm cho dữ liệu này, có thể do dữ liệu có nhiều biến động và không có cấu trúc rõ ràng.

**3.3.6. Phân tích sử dụng Máy vector hỗ trợ (Support Vector Regression)**

**Bước 1: Tạo và huấn luyện mô hình Máy vector hỗ trợ**

from sklearn.svm import SVR

svr\_regressor = SVR(kernel='linear')

svr\_regressor.fit(X\_train[['Metascore']], y\_train)

* **SVR**: Lớp này từ thư viện sklearn.svm được sử dụng để tạo mô hình hồi quy vector hỗ trợ.
* **svr\_regressor**: Đối tượng của lớp SVR.
* **kernel**: Tham số này xác định loại kernel được sử dụng trong mô hình. Ở đây, kernel được chọn là 'linear'.
* **fit**: Phương thức này được sử dụng để huấn luyện mô hình hồi quy vector hỗ trợ với dữ liệu huấn luyện X\_train[['Metascore']] và y\_train.

**Bước 2: Dự đoán giá trị của biến phụ thuộc**

y\_pred\_svr = svr\_regressor.predict(X\_test[['Metascore']])

* **predict**: Phương thức này được sử dụng để dự đoán giá trị của biến phụ thuộc y dựa trên dữ liệu kiểm tra X\_test[['Metascore']].

**Bước 3: Tính toán các chỉ số đánh giá**

from sklearn.metrics import mean\_squared\_error, r2\_score

mse\_svr = mean\_squared\_error(y\_test, y\_pred\_svr)

r2\_svr = r2\_score(y\_test, y\_pred\_svr)

print("Support Vector Regression - MSE:", mse\_svr)

print("Support Vector Regression - R2 Score:", r2\_svr)

* **mean\_squared\_error**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán sai số bình phương trung bình (MSE).
* **r2\_score**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán hệ số xác định (R2 Score).

**Kết Quả: A graph with blue dots and red line

Description automatically generated**

Support Vector Regression - MSE: 0.553443499998206

Support Vector Regression - R2 Score: 0.4626313459199448

* Mô hình hồi quy vector hỗ trợ có R2 Score là 0.4626, tương đương với hồi quy tuyến tính. Điều này cho thấy rằng SVM cũng có thể là một lựa chọn tốt, nhưng không vượt trội so với các mô hình khác.

**3.3.7. Kết luận quá trình phân tích mối quan hệ Metascore - Rating**

Qua quá trình phân tích, chúng ta đã sử dụng nhiều phương pháp hồi quy khác nhau để dự đoán giá trị Rating dựa trên Metascore và các biến độc lập khác. Các mô hình hồi quy tuyến tính đơn biến, hồi quy đa thức, cây quyết định, rừng ngẫu nhiên, K láng giềng gần nhất, và máy vector hỗ trợ đều có những ưu điểm và hạn chế riêng.

* **Hồi quy tuyến tính đơn biến**: Đơn giản và dễ hiểu nhưng có thể không đủ để mô tả mối quan hệ phức tạp.
* **Hồi quy đa thức**: Có khả năng mô tả mối quan hệ phi tuyến tính nhưng dễ dẫn đến overfitting nếu không cẩn thận.
* **Cây quyết định**: Dễ hiểu và trực quan nhưng có thể dễ bị overfitting.
* **Rừng ngẫu nhiên**: Giảm thiểu overfitting và cải thiện độ chính xác nhưng phức tạp hơn và yêu cầu nhiều tài nguyên tính toán.
* **K láng giềng gần nhất**: Đơn giản và hiệu quả với dữ liệu nhỏ nhưng không hoạt động tốt với dữ liệu lớn và phức tạp.
* **Máy vector hỗ trợ**: Hiệu quả với dữ liệu có nhiều chiều nhưng khó hiểu và giải thích.

Qua các kết quả trên, chúng ta thấy rằng mô hình hồi quy đa thức có hiệu suất tốt nhất với R2 Score cao nhất, cho thấy khả năng giải thích tốt hơn sự biến động của Rating dựa trên Metascore. Tuy nhiên, sự khác biệt giữa các mô hình không quá lớn, và việc lựa chọn mô hình phù hợp còn phụ thuộc vào nhiều yếu tố khác như độ phức tạp của mô hình và tài nguyên tính toán.

Việc kết hợp nhiều mô hình và sử dụng các kỹ thuật như cross-validation có thể giúp cải thiện độ chính xác và độ tin cậy của dự đoán. Điều này nhấn mạnh tầm quan trọng của việc thử nghiệm và đánh giá nhiều phương pháp khác nhau trong quá trình phân tích dữ liệu.

**3.3. Tìm hiểu mối quan hệ giữa Votes và Rating**

Nghiên cứu mối quan hệ giữa Votes và Rating nhằm hiểu rõ hơn về tương tác giữa hai yếu tố này trong ngành điện ảnh. Mục tiêu là phân tích cách Votes có thể biến đổi dựa trên các biến động của Rating và ngược lại. Thông qua việc đánh giá mối quan hệ này, chúng ta có thể có cái nhìn sâu sắc hơn về cách Votes ảnh hưởng đến sự đánh giá của khán giả và ngược lại. Điều này có thể giúp chúng ta hiểu rõ hơn về đặc điểm của các bộ phim và đồng thời cung cấp thông tin quan trọng cho việc dự đoán và hiểu biến đổi của Rating trong tương lai.

**Bước 1: Chuẩn bị dữ liệu**

Đầu tiên, chúng ta cần chuẩn bị dữ liệu bằng cách chọn các cột liên quan và chia dữ liệu thành tập huấn luyện và tập kiểm tra. Đây là một phần quan trọng trong quá trình xây dựng mô hình máy học, giúp đảm bảo rằng mô hình có thể tổng quát hóa tốt trên dữ liệu mới.

from sklearn.model\_selection import train\_test\_split

import numpy as np

X = df[['Votes']]

y = df['IMDB Rating']

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

* **train\_test\_split**: Hàm này từ thư viện sklearn.model\_selection được sử dụng để chia dữ liệu thành hai tập: tập huấn luyện và tập kiểm tra.
* **X**: DataFrame chứa giá trị của cột 'Votes'.
* **y**: DataFrame chứa giá trị của cột 'IMDB Rating'.
* **X\_train, X\_test, y\_train, y\_test**: Các tập dữ liệu được chia ra, với 80% dữ liệu dùng để huấn luyện và 20% dữ liệu dùng để kiểm tra.

**3.3.1. Phân tích hồi quy tuyến tính đơn biến (Linear Regression with One Variable)**

**Giới thiệu**

Hồi quy tuyến tính đơn biến là một phương pháp thống kê được sử dụng để dự đoán giá trị của một biến phụ thuộc dựa trên giá trị của một biến độc lập. Dạng cơ bản của mô hình hồi quy đơn biến có một biến độc lập và một biến phụ thuộc: Y=B0+B1X*Y*=*B*0+*B*1*X*. X là biến độc lập, Y là biến phụ thuộc. B0 và B1 là các hệ số.

**Bước 1: Tạo và huấn luyện mô hình hồi quy tuyến tính**

from sklearn.linear\_model import LinearRegression

linear\_regressor = LinearRegression()

linear\_regressor.fit(X\_train, y\_train)

* **LinearRegression**: Lớp này từ thư viện sklearn.linear\_model được sử dụng để tạo mô hình hồi quy tuyến tính.
* **linear\_regressor**: Đối tượng của lớp LinearRegression.
* **fit**: Phương thức này được sử dụng để huấn luyện mô hình hồi quy tuyến tính với dữ liệu huấn luyện X\_train và y\_train.

**Bước 2: Dự đoán giá trị của biến phụ thuộc**

y\_pred\_linear = linear\_regressor.predict(X\_test)

* **predict**: Phương thức này được sử dụng để dự đoán giá trị của biến phụ thuộc y dựa trên dữ liệu kiểm tra X\_test.

**Bước 3: Vẽ biểu đồ hồi quy tuyến tính**

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

plt.scatter(X\_test, y\_test, color='blue', label='Actual')

plt.plot(X\_test, y\_pred\_linear, color='red', linewidth=2, label='Predicted')

plt.title('Linear Regression')

plt.xlabel('Votes')

plt.ylabel('IMDB Rating')

plt.legend()

plt.show()

* **matplotlib.pyplot**: Thư viện này được sử dụng để vẽ các biểu đồ.
* **plt.scatter**: Hàm này vẽ các điểm dữ liệu thực tế.
* **plt.plot**: Hàm này vẽ đường hồi quy tuyến tính dựa trên các giá trị dự đoán.

**Bước 4: Tính toán các chỉ số đánh giá hiệu suất**

from sklearn.metrics import mean\_squared\_error, r2\_score

print("Linear Regression - MSE:", mean\_squared\_error(y\_test, y\_pred\_linear))

print("Linear Regression - R2 Score:", r2\_score(y\_test, y\_pred\_linear))

* **mean\_squared\_error**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán sai số bình phương trung bình (MSE).
* **r2\_score**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán hệ số xác định (R2 Score).

**Kết Quả**: A graph with blue dots and red line

Description automatically generated

Linear Regression - MSE: 0.8554136389711351

Linear Regression - R2 Score: 0.1694319729888757

**Định nghĩa các chỉ số đánh giá**

* **Mean Squared Error (MSE)**: Đây là chỉ số đo lường mức độ chênh lệch giữa giá trị thực tế và giá trị dự đoán của mô hình. MSE càng thấp, mô hình càng chính xác.
* **R2 Score**: Đây là chỉ số đo lường mức độ giải thích của mô hình đối với biến phụ thuộc. R2 Score càng gần 1, mô hình càng tốt trong việc giải thích sự biến động của biến phụ thuộc.

Mô hình hồi quy tuyến tính đơn biến cho thấy một mức độ giải thích khá thấp với R2 Score là 0.1694. Điều này cho thấy rằng chỉ khoảng 16.94% sự biến động của Rating có thể được giải thích bởi Votes.

**3.3.2. Phân tích hồi quy đa thức (Polynomial Regression)**

**Giới thiệu**

Hồi quy đa thức là một phương pháp mở rộng của hồi quy tuyến tính, trong đó mối quan hệ giữa biến độc lập và biến phụ thuộc được mô hình hóa dưới dạng đa thức. Dạng cơ bản của mô hình hồi quy đa thức có một biến độc lập và một biến phụ thuộc như sau: Y=b0+b1x+b2x2+…+bnxn*Y*=*b*0+*b*1*x*+*b*2*x*2+…+*bnxn*. Khi tăng giá trị của degree, mô hình có thể biểu diễn mối quan hệ phức tạp hơn giữa biến độc lập và biến phụ thuộc. Tuy nhiên, cũng cần lưu ý rằng việc sử dụng các đa thức có bậc cao có thể dẫn đến hiện tượng overfitting, đặc biệt là khi dữ liệu không đủ lớn.

**Bước 1: Tạo đa thức cho dữ liệu huấn luyện**

from sklearn.preprocessing import PolynomialFeatures

poly\_features = PolynomialFeatures(degree=2)

X\_poly\_train = poly\_features.fit\_transform(X\_train)

* **PolynomialFeatures**: Lớp này từ thư viện sklearn.preprocessing được sử dụng để tạo các đặc trưng đa thức từ dữ liệu gốc.
* **poly\_features**: Đối tượng của lớp PolynomialFeatures với bậc đa thức là 2.
* **fit\_transform**: Phương thức này được sử dụng để biến đổi dữ liệu huấn luyện X\_train thành các đặc trưng đa thức.

**Bước 2: Tạo và huấn luyện mô hình hồi quy đa thức**

poly\_regressor = LinearRegression()

poly\_regressor.fit(X\_poly\_train, y\_train)

* **poly\_regressor**: Đối tượng của lớp LinearRegression được sử dụng để tạo mô hình hồi quy tuyến tính trên các đặc trưng đa thức.
* **fit**: Phương thức này được sử dụng để huấn luyện mô hình hồi quy đa thức với dữ liệu huấn luyện đã được biến đổi X\_poly\_train và y\_train.

**Bước 3: Biến đổi dữ liệu kiểm tra và dự đoán giá trị**

X\_poly\_test = poly\_features.transform(X\_test)

y\_pred\_poly = poly\_regressor.predict(X\_poly\_test)

* **transform**: Phương thức này được sử dụng để biến đổi dữ liệu kiểm tra X\_test thành các đặc trưng đa thức.
* **predict**: Phương thức này được sử dụng để dự đoán giá trị của biến phụ thuộc y dựa trên dữ liệu kiểm tra đã được biến đổi X\_poly\_test.

**Bước 4: Vẽ biểu đồ hồi quy đa thức**

plt.figure(figsize=(10, 6))

plt.scatter(X\_test, y\_test, color='blue', label='Actual')

plt.scatter(X\_test, y\_pred\_poly, color='red', label='Predicted')

plt.title('Polynomial Regression')

plt.xlabel('Votes')

plt.ylabel('IMDB Rating')

plt.legend()

plt.show()

* **plt.scatter**: Hàm này vẽ các điểm dữ liệu thực tế.
* **plt.scatter**: Hàm này vẽ các điểm dữ liệu dự đoán.

**Bước 5: Tính toán các chỉ số đánh giá hiệu suất**

print("Polynomial Regression - MSE:", mean\_squared\_error(y\_test, y\_pred\_poly))

print("Polynomial Regression - R2 Score:", r2\_score(y\_test, y\_pred\_poly))

* **mean\_squared\_error**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán sai số bình phương trung bình (MSE).
* **r2\_score**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán hệ số xác định (R2 Score).

**Kết Quả**: A graph showing a red line and blue dots

Description automatically generated

Polynomial Regression - MSE: 0.8361372210694187

Polynomial Regression - R2 Score: 0.1881485045650232

* Mô hình hồi quy đa thức với bậc 2 cải thiện nhẹ so với hồi quy tuyến tính đơn biến với R2 Score là 0.1881. Điều này cho thấy rằng việc sử dụng mô hình phi tuyến tính có thể mô tả mối quan hệ giữa Votes và Rating tốt hơn một chút.

**3.3.3. Phân tích sử dụng Cây quyết định (Decision Tree) (tiếp tục)**

**Bước 1: Tạo và huấn luyện mô hình cây quyết định**

from sklearn.tree import DecisionTreeRegressor

tree\_regressor = DecisionTreeRegressor(random\_state=42)

tree\_regressor.fit(X\_train, y\_train)

* **DecisionTreeRegressor**: Lớp này từ thư viện sklearn.tree được sử dụng để tạo mô hình cây quyết định.
* **tree\_regressor**: Đối tượng của lớp DecisionTreeRegressor.
* **fit**: Phương thức này được sử dụng để huấn luyện mô hình cây quyết định với dữ liệu huấn luyện X\_train và y\_train.

**Bước 2: Dự đoán giá trị của biến phụ thuộc**

y\_pred\_tree = tree\_regressor.predict(X\_test)

* **predict**: Phương thức này được sử dụng để dự đoán giá trị của biến phụ thuộc y dựa trên dữ liệu kiểm tra X\_test.

**Bước 3: Vẽ biểu đồ hồi quy cây quyết định**

plt.figure(figsize=(10, 6))

plt.scatter(range(len(y\_test)), y\_test, color='blue', label='Actual')

plt.scatter(range(len(y\_test)), y\_pred\_tree, color='red', label='Predicted')

plt.title('Decision Tree Regression')

plt.xlabel('Sample Index')

plt.ylabel('IMDB Rating')

plt.legend()

plt.show()

* **plt.scatter**: Hàm này vẽ các điểm dữ liệu thực tế và dự đoán.

**Bước 4: Tính toán các chỉ số đánh giá**

from sklearn.metrics import mean\_squared\_error, r2\_score

mse\_tree = mean\_squared\_error(y\_test, y\_pred\_tree)

r2\_tree = r2\_score(y\_test, y\_pred\_tree)

print("Decision Tree - MSE:", mse\_tree)

print("Decision Tree - R2 Score:", r2\_tree)

* **mean\_squared\_error**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán sai số bình phương trung bình (MSE).
* **r2\_score**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán hệ số xác định (R2 Score).

**Kết Quả**:

A diagram of a graph

Description automatically generated with medium confidence

Decision Tree - MSE: 1.5918812500000001

Decision Tree - R2 Score: -0.5456448305392492

* Mô hình cây quyết định có R2 Score là -0.5456, thấp hơn rất nhiều so với các mô hình khác. Điều này cho thấy rằng mô hình cây quyết định không phù hợp cho dữ liệu này, có thể do overfitting hoặc dữ liệu không có cấu trúc rõ ràng.

**3.3.4. Phân tích sử dụng Rừng ngẫu nhiên (Random Forest)**

**Giới thiệu**

Rừng ngẫu nhiên là một mô hình dự đoán sử dụng nhiều cây quyết định để cải thiện độ chính xác và giảm thiểu overfitting. Mô hình này hoạt động bằng cách tạo ra nhiều cây quyết định từ các mẫu ngẫu nhiên của dữ liệu huấn luyện và sau đó kết hợp các dự đoán của các cây này để đưa ra dự đoán cuối cùng.

**Bước 1: Tạo và huấn luyện mô hình rừng ngẫu nhiên**

from sklearn.ensemble import RandomForestRegressor

forest\_regressor = RandomForestRegressor(n\_estimators=100, random\_state=42)

forest\_regressor.fit(X\_train, y\_train)

* **RandomForestRegressor**: Lớp này từ thư viện sklearn.ensemble được sử dụng để tạo mô hình rừng ngẫu nhiên.
* **forest\_regressor**: Đối tượng của lớp RandomForestRegressor.
* **n\_estimators**: Tham số này xác định số lượng cây quyết định trong rừng ngẫu nhiên.
* **fit**: Phương thức này được sử dụng để huấn luyện mô hình rừng ngẫu nhiên với dữ liệu huấn luyện X\_train và y\_train.

**Bước 2: Dự đoán giá trị của biến phụ thuộc**

y\_pred\_forest = forest\_regressor.predict(X\_test)

* **predict**: Phương thức này được sử dụng để dự đoán giá trị của biến phụ thuộc y dựa trên dữ liệu kiểm tra X\_test.

**Bước 3: Vẽ biểu đồ hồi quy rừng ngẫu nhiên**

plt.figure(figsize=(10, 6))

plt.scatter(range(len(y\_test)), y\_test, color='blue', label='Actual')

plt.scatter(range(len(y\_test)), y\_pred\_forest, color='red', label='Predicted')

plt.title('Random Forest Regression')

plt.xlabel('Sample Index')

plt.ylabel('IMDB Rating')

plt.legend()

plt.show()

* **plt.scatter**: Hàm này vẽ các điểm dữ liệu thực tế và dự đoán.

**Bước 4: Tính toán các chỉ số đánh giá**

from sklearn.metrics import mean\_squared\_error, r2\_score

mse\_forest = mean\_squared\_error(y\_test, y\_pred\_forest)

r2\_forest = r2\_score(y\_test, y\_pred\_forest)

print("Random Forest - MSE:", mse\_forest)

print("Random Forest - R2 Score:", r2\_forest)

* **mean\_squared\_error**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán sai số bình phương trung bình (MSE).
* **r2\_score**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán hệ số xác định (R2 Score).

**Kết Quả**: A red and blue dots

Description automatically generated

Random Forest - MSE: 1.2155867600124295

Random Forest - R2 Score: -0.1802798680398845

* Mô hình rừng ngẫu nhiên có R2 Score là -0.1803, cũng không tốt hơn nhiều so với cây quyết định. Điều này cho thấy rằng việc sử dụng nhiều cây quyết định không cải thiện đáng kể độ chính xác của mô hình trong trường hợp này.

**3.3.5. Phân tích sử dụng K láng giềng gần nhất (K-Nearest Neighbors)**

**Giới thiệu**

K láng giềng gần nhất (KNN) là một thuật toán học máy đơn giản nhưng mạnh mẽ, được sử dụng rộng rãi trong các bài toán phân loại và hồi quy. Nguyên lý cơ bản của KNN là dựa trên khoảng cách giữa các điểm dữ liệu để dự đoán giá trị của một điểm mới. Cụ thể, khi cần dự đoán giá trị của một điểm dữ liệu mới, KNN sẽ tìm k điểm dữ liệu gần nhất trong tập huấn luyện và sử dụng giá trị của chúng để đưa ra dự đoán.

**Bước 1: Tạo và huấn luyện mô hình K láng giềng gần nhất**

from sklearn.neighbors import KNeighborsRegressor

knn\_regressor = KNeighborsRegressor(n\_neighbors=3)

knn\_regressor.fit(X\_train, y\_train)

* **KNeighborsRegressor**: Lớp này từ thư viện sklearn.neighbors được sử dụng để tạo mô hình K láng giềng gần nhất.
* **knn\_regressor**: Đối tượng của lớp KNeighborsRegressor.
* **n\_neighbors**: Tham số này xác định số lượng láng giềng gần nhất được sử dụng để dự đoán.
* **fit**: Phương thức này được sử dụng để huấn luyện mô hình K láng giềng gần nhất với dữ liệu huấn luyện X\_train và y\_train.

**Bước 2: Dự đoán giá trị của biến phụ thuộc**

y\_pred\_knn = knn\_regressor.predict(X\_test)

* **predict**: Phương thức này được sử dụng để dự đoán giá trị của biến phụ thuộc y dựa trên dữ liệu kiểm tra X\_test.

**Bước 3: Vẽ biểu đồ hồi quy K láng giềng gần nhất**

plt.figure(figsize=(10, 6))

plt.scatter(range(len(y\_test)), y\_test, color='blue', label='Actual')

plt.scatter(range(len(y\_test)), y\_pred\_knn, color='red', label='Predicted')

plt.title('K-Nearest Neighbors Regression')

plt.xlabel('Sample Index')

plt.ylabel('IMDB Rating')

plt.legend()

plt.show()

* **plt.scatter**: Hàm này vẽ các điểm dữ liệu thực tế và dự đoán.

**Bước 4: Tính toán các chỉ số đánh giá**

from sklearn.metrics import mean\_squared\_error, r2\_score

mse\_knn = mean\_squared\_error(y\_test, y\_pred\_knn)

r2\_knn = r2\_score(y\_test, y\_pred\_knn)

print("K-Nearest Neighbors - MSE:", mse\_knn)

print("K-Nearest Neighbors - R2 Score:", r2\_knn)

* **mean\_squared\_error**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán sai số bình phương trung bình (MSE).
* **r2\_score**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán hệ số xác định (R2 Score).

**Kết Quả**: A diagram of red and blue dots

Description automatically generated

K-Nearest Neighbors - MSE: 1.0827166666666665

K-Nearest Neighbors - R2 Score: -0.051269005632185394

* Mô hình K láng giềng gần nhất có R2 Score là -0.0513, cũng không phù hợp lắm cho dữ liệu này. Điều này có thể do dữ liệu có nhiều biến động và không có cấu trúc rõ ràng.

**3.3.6. Phân tích sử dụng Máy vector hỗ trợ (Support Vector Regression) (tiếp tục)**

**Bước 1: Tạo và huấn luyện mô hình Máy vector hỗ trợ**

from sklearn.svm import SVR

svr\_regressor = SVR(kernel='linear')

svr\_regressor.fit(X\_train[['Votes']], y\_train)

* **SVR**: Lớp này từ thư viện sklearn.svm được sử dụng để tạo mô hình hồi quy vector hỗ trợ.
* **svr\_regressor**: Đối tượng của lớp SVR.
* **kernel**: Tham số này xác định loại kernel được sử dụng trong mô hình. Ở đây, kernel được chọn là 'linear'.
* **fit**: Phương thức này được sử dụng để huấn luyện mô hình hồi quy vector hỗ trợ với dữ liệu huấn luyện X\_train[['Votes']] và y\_train.

**Bước 2: Dự đoán giá trị của biến phụ thuộc**

y\_pred\_svr = svr\_regressor.predict(X\_test[['Votes']])

* **predict**: Phương thức này được sử dụng để dự đoán giá trị của biến phụ thuộc y dựa trên dữ liệu kiểm tra X\_test[['Votes']].

**Bước 3: Vẽ biểu đồ hồi quy Máy vector hỗ trợ**

plt.figure(figsize=(10, 6))

plt.scatter(X\_test['Votes'], y\_test, color='blue', label='Actual')

plt.scatter(X\_test['Votes'], y\_pred\_svr, color='red', label='Predicted')

plt.title('Support Vector Regression')

plt.xlabel('Votes')

plt.ylabel('IMDB Rating')

plt.legend()

plt.show()

* **plt.scatter**: Hàm này vẽ các điểm dữ liệu thực tế và dự đoán.

**Bước 4: Tính toán các chỉ số đánh giá**

from sklearn.metrics import mean\_squared\_error, r2\_score

mse\_svr = mean\_squared\_error(y\_test, y\_pred\_svr)

r2\_svr = r2\_score(y\_test, y\_pred\_svr)

print("Support Vector Regression - MSE:", mse\_svr)

print("Support Vector Regression - R2 Score:", r2\_svr)

* **mean\_squared\_error**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán sai số bình phương trung bình (MSE).
* **r2\_score**: Hàm này từ thư viện sklearn.metrics được sử dụng để tính toán hệ số xác định (R2 Score).

**Kết Quả**: A graph with red and blue dots

Description automatically generated

Support Vector Regression - MSE: 0.8710481535730659

Support Vector Regression - R2 Score: 0.15425156510828486

* Mô hình hồi quy vector hỗ trợ có R2 Score là 0.1543, tương đương với hồi quy tuyến tính. Điều này cho thấy rằng SVM cũng có thể là một lựa chọn tốt, nhưng không vượt trội so với các mô hình khác.

**3.3.7: Kết Luận**

Qua các kết quả trên, chúng ta thấy rằng mô hình hồi quy đa thức có hiệu suất tốt nhất với R2 Score cao nhất, cho thấy khả năng giải thích tốt hơn sự biến động của Rating dựa trên Votes. Tuy nhiên, sự khác biệt giữa các mô hình không quá lớn, và việc lựa chọn mô hình phù hợp còn phụ thuộc vào nhiều yếu tố khác như độ phức tạp của mô hình và tài nguyên tính toán.

Việc kết hợp nhiều mô hình và sử dụng các kỹ thuật như cross-validation có thể giúp cải thiện độ chính xác và độ tin cậy của dự đoán. Điều này nhấn mạnh tầm quan trọng của việc thử nghiệm và đánh giá nhiều phương pháp khác nhau trong quá trình phân tích dữ liệu.

**KẾT LUẬN**

Trong đồ án này, chúng tôi đã tiến hành phân tích dữ liệu về đánh giá rating phim Ấn Độ nhằm hiểu rõ hơn về các yếu tố ảnh hưởng đến rating và mối quan hệ giữa các yếu tố này. Qua quá trình phân tích, chúng tôi đã sử dụng nhiều phương pháp khác nhau như hồi quy tuyến tính, hồi quy đa thức, cây quyết định, rừng ngẫu nhiên, K láng giềng gần nhất và máy vector hỗ trợ để dự đoán và đánh giá rating phim.

Kết quả cho thấy rằng các yếu tố như IMDB Rating, Metascore, số lượng phiếu bầu và doanh thu phòng vé đều có ảnh hưởng đáng kể đến rating phim. Trong đó, IMDB Rating và Metascore là hai yếu tố quan trọng nhất, có mối tương quan mạnh mẽ với rating phim. Mô hình Random Forest cho kết quả dự đoán tốt nhất với MSE thấp nhất và R2 Score cao nhất, cho thấy khả năng giải thích tốt sự biến động của rating phim dựa trên các yếu tố đã phân tích.

Qua đó, chúng tôi có thể kết luận rằng việc hiểu rõ và phân tích các yếu tố ảnh hưởng đến rating phim là rất quan trọng để có thể dự đoán và cải thiện chất lượng phim, đáp ứng nhu cầu và mong đợi của khán giả.

**Hướng phát triển trong tương lai**

Dựa trên kết quả và những hạn chế của đồ án này, chúng tôi đề xuất một số hướng phát triển trong tương lai như sau:

1. **Mở rộng tập dữ liệu**: Thu thập thêm dữ liệu từ nhiều nguồn khác nhau, bao gồm cả các nền tảng đánh giá phim khác như Rotten Tomatoes, Metacritic, để có cái nhìn toàn diện hơn về các yếu tố ảnh hưởng đến rating phim.
2. **Phân tích sâu hơn về các yếu tố khác**: Ngoài các yếu tố đã phân tích, có thể xem xét thêm các yếu tố khác như thể loại phim, đạo diễn, diễn viên, kịch bản, và chiến lược quảng bá để hiểu rõ hơn về tác động của chúng đến rating phim.
3. **Áp dụng các kỹ thuật học máy nâng cao**: Sử dụng các mô hình học máy nâng cao như mạng nơ-ron nhân tạo (ANN), mạng nơ-ron tích chập (CNN), và mạng nơ-ron hồi quy (RNN) để cải thiện độ chính xác của dự đoán rating phim.
4. **Phân tích cảm xúc từ đánh giá của khán giả**: Sử dụng kỹ thuật xử lý ngôn ngữ tự nhiên (NLP) để phân tích cảm xúc từ các đánh giá của khán giả trên các nền tảng mạng xã hội và trang web đánh giá phim, từ đó hiểu rõ hơn về phản hồi của khán giả và cải thiện chất lượng phim.
5. **Tích hợp hệ thống đề xuất phim**: Phát triển hệ thống đề xuất phim dựa trên sở thích và đánh giá của khán giả, giúp người xem dễ dàng tìm kiếm và lựa chọn những bộ phim phù hợp với sở thích cá nhân.

Những hướng phát triển này không chỉ giúp nâng cao chất lượng của các dự án phân tích dữ liệu về rating phim mà còn góp phần cải thiện trải nghiệm của khán giả và thúc đẩy sự phát triển của ngành công nghiệp điện ảnh.