TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO BÀI TẬP LỚN
MÔN HỌC MÁY

Đề tài: Dự đoán doanh thu phim

Giáo viên hướng dẫn: **TS. Nguyễn Nhật Quang**

Sinh viên thực hiện:

1. Ngô Văn Tuấn - **20200559**
2. Vũ Văn Lợi - **20204577**
3. Bùi Lâm Thanh - **20204606**
4. Hà Thị Thanh Huyền - **20200288**
5. Phạm Thị Hồng Hạnh - **20204546**

Mã lớp: **136805**

Hà Nội, 12/2022

Mục lục

Danh mục hình ảnh

1. Mô tả bài toán

Theo con số thống kê từ các hệ thống rạp lớn nhất trên cả nước, tổng doanh thu màn ảnh Việt Nam năm 2019 là trên 4.100 tỷ đồng, tăng 26% so với năm trước đó. Các phim điện ảnh Việt Nam có mức tăng trưởng hơn 40% so với năm 2018, năm 2019 trở thành năm có doanh thu phim Việt cao nhất trong lịch sử phòng vé. Những con số ấn tượng này chứng tỏ sự tăng trưởng không ngừng của ngành công nghệ phim ảnh tại Việt Nam. Đối với một nhà đầu tư, yếu tố tiên quyết để họ đầu tư vào một sản phẩm điện ảnh là sự chắc chắn vào thành công mà nó đem lại, hay nói dễ hiểu hơn, bộ phim phải đem lại lợi nhuận vượt xa kinh phí. Hiểu được tầm quan trọng của việc dự đoán doanh thu của một sản phẩm điện ảnh, nhóm quyết định thực hiện những nghiên cứu, thực hành để phát triển một hệ thống "Dự đoán doanh thu" đối với sản phẩm cụ thể là các sản phẩm điện ảnh.

Một hệ thống dự đoán doanh thu đáng tin cậy không chỉ có ý nghĩa lớn trong việc thu hút các nhà đầu tư, mà còn giúp đội ngũ sản xuất phim tối ưu chiến lược quảng cáo sản phẩm. Tuy nhiên, yêu cầu đặt ra của bài toán này không hề đơn giản. Trong quá khứ, vài nhà phát triển đã sử dụng đánh giá của người xem như một thước đo cho sự thành công của bộ phim. Vài nghiên cứu khác chỉ ra mối liên hệ giữa sự xuất hiện các yếu tố như lãng mạn hay bạo lực và doanh thu phòng vé. Thế nhưng, ngoài những yếu tố đó ra, nhiều nhân tố khác có thể ảnh hưởng đến kết quả của bài toàn này. Trong phạm vi bài tập lớn này, nhóm mong muốn khám phá tác động của các yếu tố khác nhau đến doanh thu bộ phim, đồng thời tạo ra một hệ thống dự đoán doanh thu đạt độ chính xác cao trong thời gian ngắn.

Để hiện thực hóa mục tiêu đó, nhóm sử dụng ba mô hình học máy, đó là: Hồi quy tuyến tính, Hồi quy Ridge và Random Forest.

2. Các phương pháp học máy sử dụng

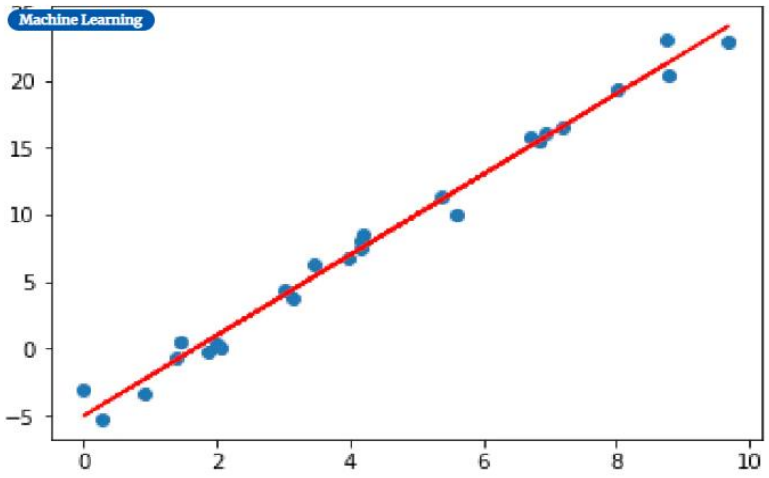
2.1. Hồi quy tuyến tính

2.1.1. Tổng quan

Phân tích hồi quy tuyến tính (Linear Regression) là một phương pháp phân tích quan hệ giữa biến phụ thuộc Y với một hoặc nhiều biến độc lập (biến dự báo) X. Mô hình hóa sử dụng hàm tuyến tính. Các tham số của mô hình được ước lượng từ dữ liệu. Hồi quy tuyến tính được sử dụng rộng rãi trong thực tế do tính chất đơn giản hóa của hồi quy.

2.1.2. Cách hoạt động

Về bản chất của hồi quy tuyến tính, kỹ thuật hồi quy tuyến tính đơn giản cố gắng vẽ một đồ thị đường giữa hai biến dữ liệu, x và y. Các biến độc lập còn được gọi là biến giải thích hoặc biến dự báo. Biến phụ thuộc Y, sẽ được vẽ trên trục tung (OY). Từ đó có thể tham chiếu các giá trị X hoặc Y lên đồ thị đó để dự đoán các giá trị chưa biết.



Hình 1. Hình minh họa cho kĩ thuật hồi quy tuyến tính

2.1.3. Hồi quy tuyến tính bội (multiple linear regression):

Có một số loại hồi quy tuyến tính phổ biến như hồi quy tuyến tính đơn giản, hồi quy tuyến tính bội, hồi quy logistic, trong đó, bài toán dự đoán doanh thu giá phim của chúng ta sẽ sử dụng mô hình hồi quy tuyến tính bội.

Trong phân tích hồi quy tuyến tính bội, tập data chứa một biến phụ thuộc và nhiều biến dự báo. Hàm hồi quy tuyến tính thay đổi để bao gồm nhiều yếu tố như sau:

$$Y= \beta 0*X0 + \beta 1X1 + \beta 2X2+..... \beta nXn+ \varepsilon$$

Khi số lượng biến dự báo tăng lên, các hằng số βn cũng tăng lên tương ứng. Đây cũng chính là loại hồi quy tuyến tính mà chúng ta sẽ sử dụng trong bài toán chúng ta đang giải quyết. Đây cũng là mô hình mà chúng ta sẽ sử dụng trong bài toán chúng ta sắp làm.

2.2. Hồi quy Ridge

2.2.1. Tổng quát

Một mục tiêu tiên quyết để có thể áp dụng được mô hình vào thực tiễn đó là chúng ta cần giảm thiểu hiện tượng *quá khớp*. Để thực hiện được mục tiêu đó, mô hình

được huấn luyện được kì vọng sẽ nắm bắt được **qui luật tổng quát** từ *tập huấn luyện* (*train dataset*) mà qui luật đó phải đúng trên những dữ liệu mới mà nó chưa được học. Thông thường tập dữ liệu mới đó được gọi là *tập kiểm tra* (*test dataset*). Đây là một tập dữ liệu độc lập được sử dụng để đánh giá mô hình.

2.2.2. Sự thay đổi của hàm mất mát trong hồi quy Ridge

Hàm mất mát trong hồi quy Ridge sẽ có sự thay đổi so với phần hồi quy tuyến tính đó là thành phần điều khiển được cộng thêm vào hàm mất mát như sau:

Bài toán tối ưu hàm mất mát của hồi qui *Ridge* về bản chất là tối ưu song song hai thành phần bao gồm tổng bình phương phần dư và *thành phần điều chuẩn*. Hệ số α có tác dụng điều chỉnh độ lớn của *thành phần điều chuẩn* tác động lên hàm mất mát.

- Trường hợp α=0, *thành phần điều chuẩn* bị tiêu giảm và chúng ta quay trở về bài toán hồi qui tuyến tính.
- Trường hợp α nhỏ thì vai trò của *thành phần điều chuẩn* trở nên ít quan trọng. Mức độ kiểm soát *quá khớp* của mô hình sẽ trở nên kém hơn.
- Trường hợp α lớn chúng ta muốn gia tăng mức độ kiểm soát lên độ lớn của các hệ số ước lượng và qua đó giảm bớt hiện tượng *quá khớp*.

Khi tăng dần hệ số α thì *hồi qui Ridge* sẽ có xu hướng thu hẹp hệ số ước lượng từ mô hình.

2.2.3. Nghiệm tối ưu của hồi quy Ridge

Giải bài toán tối ưu *hàm mục tiêu* của *hồi quy Ridge* theo đạo hàm bậc nhất của vector w:

$$\begin{aligned}\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} &= \frac{1}{N}\frac{\partial \|\bar{\mathbf{X}}\mathbf{w}-\mathbf{y}\|_2^2}{\partial \mathbf{w}} + \alpha\frac{\partial \|\mathbf{w}\|_2^2}{\partial \mathbf{w}} \\ &= \frac{2}{N}\bar{\mathbf{X}}^\intercal(\bar{\mathbf{X}}\mathbf{w}-\mathbf{y}) + 2\alpha\mathbf{w} \\ &= \frac{2}{N}[(\bar{\mathbf{X}}^\intercal\bar{\mathbf{X}} + N\alpha\mathbf{I})\mathbf{w} - \bar{\mathbf{X}}^\intercal\mathbf{y}] \\ &= 0\end{aligned}$$

Thật vậy, từ dòng 1 suy ra dòng 2 là vì theo công thức product-rule trong matrix caculus thì:

$$\nabla_{\mathbf{w}} f(\mathbf{w})^\intercal g(\mathbf{w}) = \nabla_{\mathbf{w}}(f)g + \nabla_{\mathbf{w}}(g)f$$

Khi f=g thì đạo hàm trở thành:

$$\nabla_{\mathbf{w}} f(\mathbf{w})^\intercal f(\mathbf{w}) = \nabla_{\mathbf{w}} \|f(\mathbf{w})\|_2^2 = 2\nabla_{\mathbf{w}}(f) f$$

Tương tự ta cũng có:

$$\frac{\partial \|\mathbf{w}\|_2^2}{\partial \mathbf{w}} = 2\mathbf{w}$$

Như vậy ta nhận thấy dòng 1 suy ra dòng 2 là hoàn toàn đúng. Ở dòng thứ 3 chúng ta áp dụng thêm một tính chất Iw=w trong đó I là ma trận đơn vị.

Sau cùng nghiệm của đạo hàm bậc nhất trở thành:

$$\begin{aligned}
\frac{2}{N}[(\bar{\mathbf{X}}^\intercal \bar{\mathbf{X}} + N\alpha \mathbf{I})\mathbf{w} - \bar{\mathbf{X}}^\intercal \mathbf{y}] &= 0 \\
(\bar{\mathbf{X}}^\intercal \bar{\mathbf{X}} + N\alpha \mathbf{I})\mathbf{w} &= \bar{\mathbf{X}}^\intercal \mathbf{y} \\
\mathbf{w} &= (\bar{\mathbf{X}}^\intercal \bar{\mathbf{X}} + N\alpha \mathbf{I})^{-1} \bar{\mathbf{X}}^\intercal \mathbf{y}
\end{aligned}$$

2.2.4. Sự đảm bảo lời giải của hồi quy Ridge

Để chứng minh hồi qui Ridge luôn tồn tại nghiệm chúng ta dựa vào ba tính chất lý.

1. Ma trận $A=\bar{X}^\intercal \bar{X}$ là một ma trận thực đối xứng *bán xác định dương* (*positive semi-definite*). Như vậy các *trị riêng* (*eigenvalues*) của nó là $\mu_1,\ldots,\mu_N$ không âm.
2. Nếu μ là trị riêng của ma trận A vuông thì μ+β là trị riêng của ma trận A+βI.
3. Định thức của ma trận A bằng tích các trị riêng của A.

Giả định μ là véc tơ trị riêng của ma trận A. Như vậy từ tính chất 2 suy ra *trị riêng* của ma trận A+NαI là λ=μ+Nα.

Mặt khác theo tính chất 1 thì μ≥0 do A bán xác định dương. Từ đó suy ra λ≥Nα>0. Như vậy ma trận (A+NαI) có khác trị riêng khác 0. Theo tính chất 3 ta suy ra det(A)≠0 do các trị riêng đều khác 0. Như vậy (A+NαI) là một ma trận không suy biến và *hồi quy Ridge* đảm bảo tồn tại nghiệm.

2.2.5. Điều chuẩn Tikhokov

Khi xây dựng mô hình trên những bộ dữ liệu có số lượng lớn các biến đầu vào thì thường xuất hiện hiện tượng đa cộng tuyến khiến ước lượng từ mô hình bị chệch. Chúng ta có thể khắc phục hiện tượng này thông qua áp dụng thành phần điều chuẩn Tikhonov:

$$\lambda R(\mathbf{w}) = \|\Gamma \mathbf{w}\|_2^2$$

Trong đó Γ là một ma trận vuông, thông thường được lựa chọn là một ma trận đường chéo.

Nếu giải bài toán tối ưu theo đạo hàm bậc nhất thì ta thu được nghiệm khi sử dụng điều chuẩn Tikhokov:

$$\begin{aligned}\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} &= \frac{1}{N}\frac{\partial \|\bar{\mathbf{X}}\mathbf{w}-\mathbf{y}\|_2^2}{\partial \mathbf{w}} + \alpha\frac{\partial \|\Gamma\mathbf{w}\|_2^2}{\partial \mathbf{w}} \\ &= \frac{2}{N}\bar{\mathbf{X}}^\intercal(\bar{\mathbf{X}}\mathbf{w}-\mathbf{y}) + 2\alpha\Gamma^\intercal\Gamma\mathbf{w} \\ &= \frac{2}{N}[(\bar{\mathbf{X}}^\intercal\bar{\mathbf{X}} + N\alpha\Gamma^\intercal\Gamma)\mathbf{w} - \bar{\mathbf{X}}^\intercal\mathbf{y}] \\ &= 0\end{aligned}$$

Nghiệm tối ưu:

$$\mathbf{w} = (\bar{\mathbf{X}}^\intercal\bar{\mathbf{X}} + N\alpha\Gamma^\intercal\Gamma)^{-1}\bar{\mathbf{X}}^\intercal\mathbf{y}$$

2.3. Random Forest

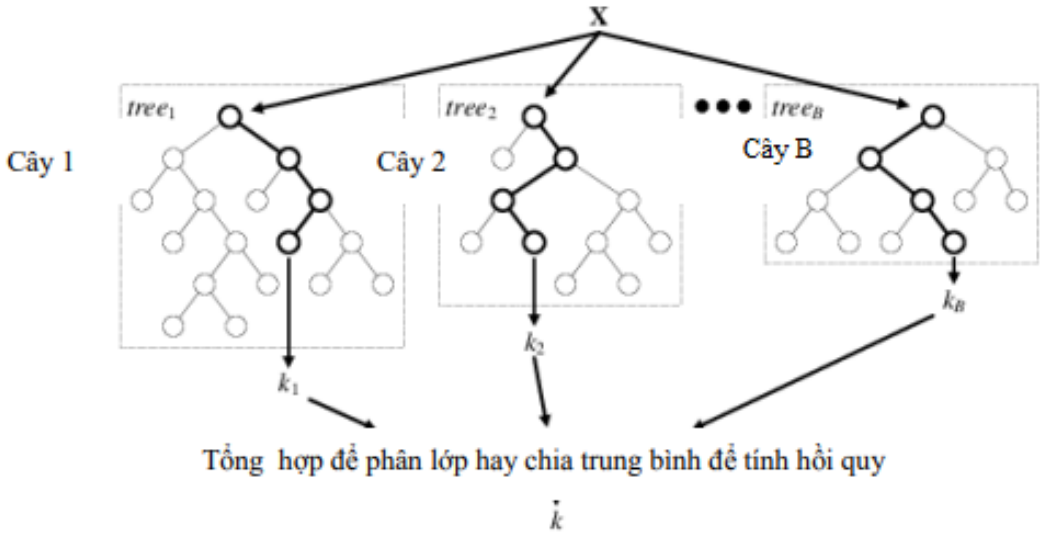
2.3.1. Tổng quan

Random forests là một phương pháp được phát triển bởi Leo Breiman năm 2001, có thể giải quyết được cả hai loại bài toán là phân loại và hồi quy, hiện nay Random forests cũng được dùng để giải quyết nhiều loại bài toán khác.

Các mô hình khác thông thường sẽ sử dụng một hàm để học và phán đoán thay vào đó. Random forests sẽ xây dựng nhiều cây quyết định (Decision tree) ngẫu nhiên khác nhau, thu thập tất cả các phán đoán sau đó lấy trung bình và đưa ra phán đoán tốt nhất. Random forests có ý tưởng đơn giản nhưng lại cho thấy tính hiệu quả vì phương pháp này có khả năng tìm ra thuộc tính quan trọng hơn so với những thuộc tính khác, thậm chí nó có thể chỉ ra rằng một số thuộc tính là không có tác dụng trong cây quyết định.

Các nguyên tắc chính của Random Forests:

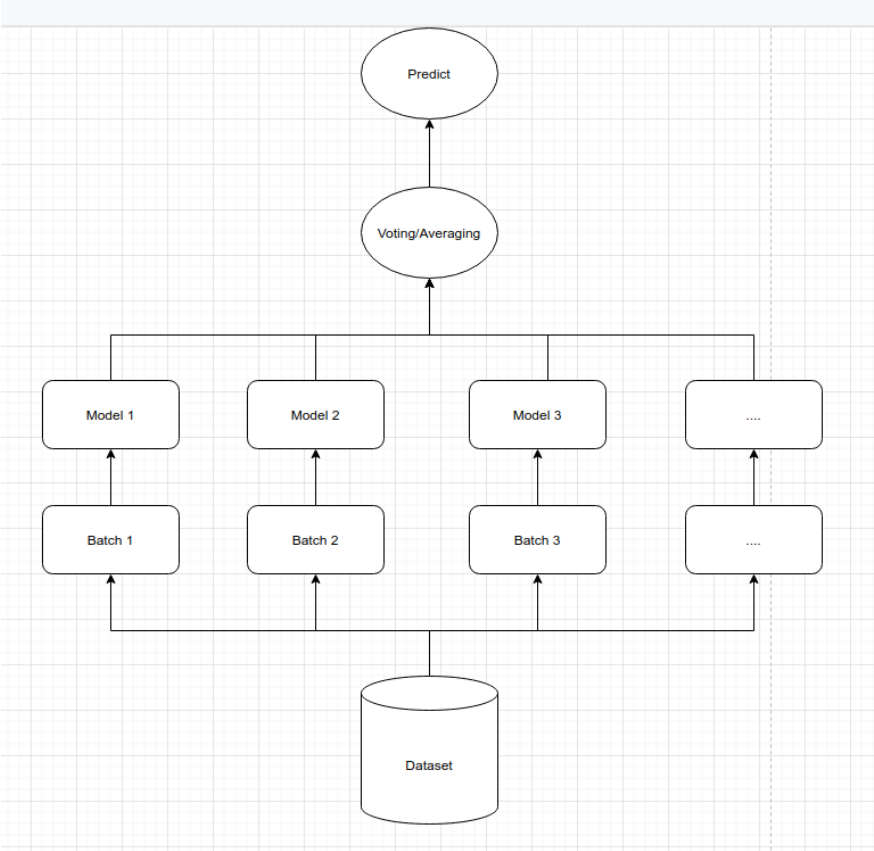
- Ngẫu nhiên hóa và không cắt tỉa: Mỗi cây quyết định trong Rừng ngẫu nhiên được phát triển một cách ngẫu nhiên: tại mỗi node trong cây, các thuộc tính được chọn một cách ngẫu nhiên. Sau đó phát triển các nhánh con từ các node và cây được phát triển hết cỡ và không cắt tỉa.
- Kết hợp (ensembling): Trong mỗi lần phán đoán, các cây quyết định sẽ được tổng hợp lại lấy trung bình để chọn ra quyết định cuối cùng.
- Lấy mẫu tái lặp (boostrapping): Đối với mỗi một cây sẽ được sinh ngẫu nhiên một tập huấn luyện bằng cách cách lấy mẫu có trùng lặp từ tập dữ liệu ban đầu.



Hình 2. Sơ đồ biểu diễn các cây quyết định trong phương pháp Random Forest

2.3.2. Lấy mẫu tái lập (boostrapping)

Giả định dữ liệu huấn luyện mô hình là một tập $D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_N, y_N)\}$ bao gồm N quan sát. Thuật toán rừng cây sẽ sử dụng phương pháp lấy mẫu tái lập để tạo thành B tập dữ liệu con. Quá trình lấy mẫu tái lập này còn gọi là bỏ túi (bagging). Tức là chúng ta sẽ thực hiện M lượt nhặt các mẫu từ tổng thể và bỏ vào túi để tạo thành tập $B_i = \{(x_1^{(i)}, y_1^{(i)}), (x_2^{(i)}, y_2^{(i)}), (x_3^{(i)}, y_3^{(i)}), \dots, (x_M^{(i)}, y_M^{(i)})\}$. Tập B_i cho phép các phần tử được lặp lại. Như vậy sẽ tồn tại những quan sát thuộc D nhưng không thuộc B_i. Đây là những quan sát chưa được bỏ vào túi và chúng ta gọi chúng là nằm ngoài túi (out of bag).



Hình 3. Lấy mẫu tái lập (boostrapping)

Với mỗi tập dữ liệu B_i chúng ta xây dựng một mô hình cây quyết định và trả về kết quả dự báo là $\hat{y}_j^{(i)} = f_i(x_j)$. Trong đó $\hat{y}_j^{(i)}$là dự báo của quan sát thứ từ mô hình thứ , là giá trị véc tơ đầu vào, là hàm dự báo của mô hình thứ . Mô hình dự báo từ cây quyết định là giá trị trung bình hoặc bầu cử của B cây quyết định.

– Đối với mô hình dự báo: Chúng ta tính giá trị trung bình của các dự báo từ mô hình con.

$$\hat{y}_j = \frac{1}{B}\sum_{i=1}^{B} \hat{y}_j^{(i)}$$

– Đối với mô hình phân loại: Chúng ta thực hiện bầu cử từ các mô hình con để chọn ra nhãn dự báo có tần suất lớn nhất.

$$\hat{y}_j = \arg\max_c \sum_{i=1}^{B} p(\hat{y}_j^{(i)} = c)$$

Như vậy phương sai của mô hình trong trường hợp đối với bài toán dự báo:

$$\sigma_{\hat{y}}^2 = Var\left(\frac{1}{B}\sum_{i=1}^{B}\hat{y}^{(i)}\right) = \frac{1}{B^2}\left[\sum_{i=1}^{B}\hat{y}^{(i)} + 2\sum_{1\le m<n\le B} cov\left(y^{(m)}, y^{(n)}\right)\right]$$

Do kết quả của mô hình con A không chịu ảnh hưởng hoặc phụ thuộc vào mô hình con B nên ta có thể giả định kết quả dự báo từ các mô hình là hoàn toàn độc lập nhau. Tức là ta có $cov\left(y^{(m)}, y^{(n)}\right),\ \forall\ 1 \le m < n \le B$. Đồng thời giả định chất lượng các mô hình là đồng đều, được thể hiện qua phương sai dự báo là đồng nhất $Var\left(\hat{y}^{(i)}\right) = , \forall i = \overline{1, B}$. Từ đó suy ra:

$$\sigma_{\hat{y}}^2 = \frac{1}{B^2}\left[\sum_{i=1}^{B} Var\left(\hat{y}^{(i)}\right)\right] = \frac{1}{B^2}\ B\ \sigma^2 = \frac{1}{B}\ \sigma^2$$

Như vậy nếu sử dụng dự báo là trung bình kết hợp từ nhiều mô hình cây quyết định thì phương sai có thể giảm B lần so với chỉ sử dụng một mô hình duy nhất. Trong một mô hình rừng cây, số lượng các cây quyết định là rất lớn. Do đó phương sai dự báo từ mô hình có thể giảm gấp nhiều lần và tạo ra một dự báo ổn định hơn.

3. Dữ liệu

3.1. Giới thiệu về tập dữ liệu

Dữ liệu được nhóm em tìm kiếm và thu thập trên trang web Kaggle. Dữ liệu là file "**movies_metadata.csv"** chứa 24 trường thuộc tính khác nhau như: ngân sách, doanh thu, thời lượng phim, số lượng người đánh giá,v.v… của 45466 bộ phim khác nhau.

	adult	belongs_t	budget	genres	homepag	id	imdb_id	original_l	original_ti	overview	popularity	poster_pa	productio	productio	release_d	revenue	runtime	spoken_la	status	tagline	title	video	vote_aver	vote_coun
2	FALSE	{'id': 1019	3E+07	[{'id': 16,	'http://toy	862	tt011470	en	Toy Story	Led by Wc	21.9469	/rhlRbceo	[{'name':	[{'iso_316	########	3.7E+08	81	[{'iso_639	Released		Toy Story	FALSE	7.7	5415
3	FALSE		6.5E+07	[{'id': 12, 'name': 'Ac		8844	tt011349	en	Jumanji	When sibl	17.0155	/vzmL6fP	[{'name':	[{'iso_316	########	2.6E+08	104	[{'iso_639	Released	Roll the di	Jumanji	FALSE	6.9	2413
4	FALSE	{'id': 1190	0	[{'id': 10749, 'name'		15602	tt011322	en	Grumpier	A family v	11.7129	/6ksm1sjI	[{'name':	[{'iso_316	########	0	101	[{'iso_639	Released	Still Yellin	Grumpier	FALSE	6.5	92
5	FALSE		1.6E+07	[{'id': 35, 'name': 'Cc		31357	tt011488	en	Waiting t	Cheated c	3.8595	/16XOMp	[{'name':	[{'iso_316	########	8.1E+07	127	[{'iso_639	Released	Friends ar	Waiting t	FALSE	6.1	34

Hình 4. Dữ liệu trong file gốc "movies_metadata.csv"

3.2. Phân tích dữ liệu và các vấn đề của dữ liệu

Dữ liệu em thu thập được ban đầu chứa rất nhiều trường thuộc tính, mục đích của bài tập lớn là dự đoán doanh thu, tuy nhiên không phải trường thuộc tính nào cũng ảnh hưởng đến việc dự đoán doanh thu và bản thân tập dữ liệu cũng chứa rất nhiều vấn đề.

3.2.1. Xóa các cột dữ liệu không sử dụng đến

Ở bước này nhóm em đã loại bỏ các cột dữ liệu mà nhóm em cho rằng sẽ không ảnh hưởng đến kết quả doanh thu như:

+ id (id của bộ phim)
+ imdb_id (id của bộ phim trên imdb)
+ tagline (khẩu hiệu của bộ phim)
+ original_tital (tiêu đề gốc của phim)

+ overview (tóm tắt nội dung phim)
+ title (tiêu đề tiếng anh của bộ phim)
+ poster_path (đường link poster)
+ video (cho biết bộ phim có video ngắn giới thiệu không)
+ belongs_to_collection (thông tin bộ phim thuộc serie nào)

Thông tin bộ phim thuộc serie nào thực tế sẽ có ảnh hưởng đến doanh thu của bộ phim. Vì có rất nhiều serie nổi tiếng chỉ cần nghe qua tên cũng đã thu hút khán giả đến xem rồi. Tuy nhiên nhóm em đã xóa cột này đi vì việc biết được serie nào có độ nổi tiếng hơn hay serie nào không nổi tiếng là rất khó

3.2.2. Phân tích các trường thuộc tính còn lại

3.2.2.1. Cột revenue: doanh thu bộ phim

Cột revenue chứa rất nhiều giá trị 0, 38052/45466 hàng, chiếm khoảng 83%. Việc để nguyên giá trị 0 hay thay toàn bộ giá trị 0 bằng các giá trị khác như giá trị trung bình, min, max sẽ ảnh hưởng đến kết quả dự đoán do số lượng hàng mang giá trị 0 là rất lớn.

Dữ liệu luôn được ví như vàng bạc, tuy nhiên do chưa tìm được phương pháp thích hợp để xử lý nên nhóm em sẽ xóa các hàng này đi. Sau khi xóa sẽ còn khoảng 7000 hàng, một con số không quá nhỏ

3.2.2.2. Cột budget: ngân sách bộ phim

Cột budget cũng xảy ra tình trạng tương tự với revenue. Và nhóm em cũng sẽ xóa các hàng mang giá trị 0 đi

Kết quả cuối cùng sau khi xóa các hàng mang giá trị 0 của revenue và budget sẽ còn 5381 hàng

```
In [17]: df['revenue'] = df['revenue'].replace(0, np.nan) # thay thế giá trị 0 bằng giá trị nan

In [20]: df['budget'] = df['budget'].replace(0, np.nan)

In [21]: df = df.dropna(subset=['budget', 'revenue'])
```

3.2.2.3. Cột adult: giới hạn người trên 18 tuổi xem

Sau khi kiểm tra nhóm em thấy toàn bộ các hàng đều mang giá trị False

```
In [23]: df['adult'].value_counts()

Out[23]: False    5381
         Name: adult, dtype: int64
```

Do đó sẽ xóa trường thuộc tính này

3.2.2.4. Cột production_countries: các quốc gia nơi bộ phim được quay

Giá trị của trường thuộc tính này là danh sách các quốc gia, và ngoài ra còn có các hàng bị thiếu dữ liệu. Nhóm em sẽ điền vào các hàng trống này 1 danh sách rỗng.

Vì nếu để ở dạng danh sách các chuỗi sẽ không thể huấn luyện mô hình. Do đó nhóm em sẽ chuyển đổi danh sách các quốc gia về số lượng các quốc gia

```
In [27]: df['production_countries'] = df['production_countries'].apply(lambda x: len(x))
```

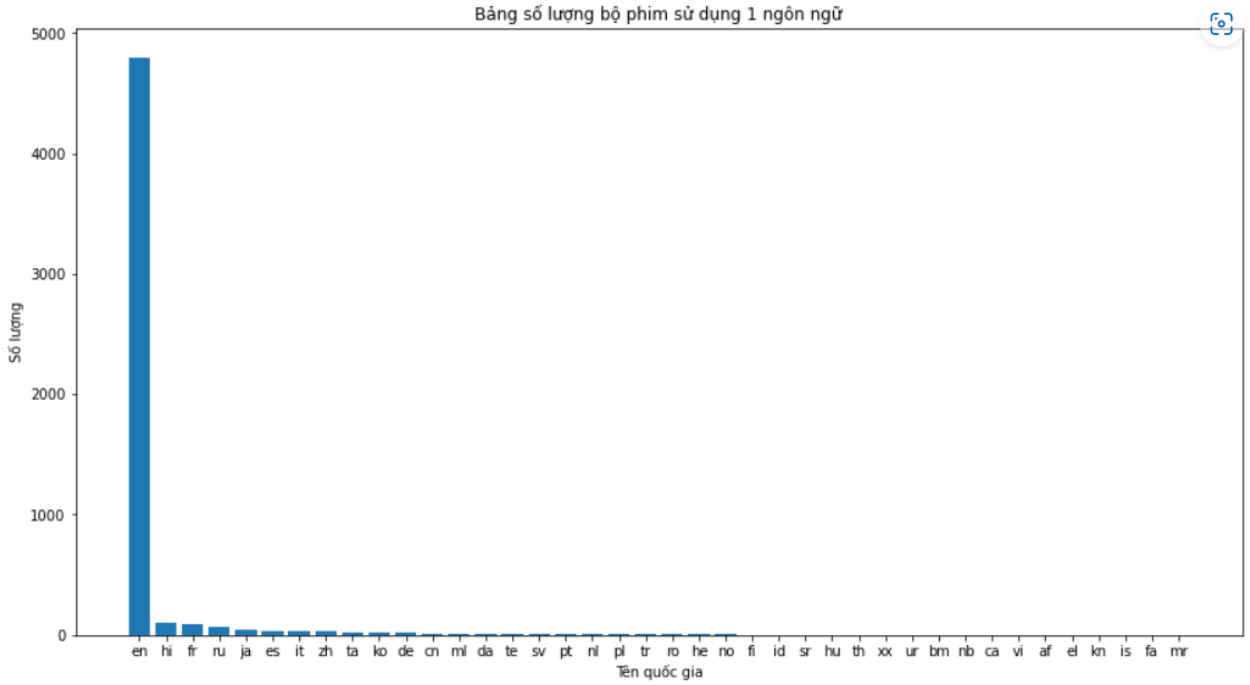
3.2.2.5. Cột production_companies: danh sách các công ty thực hiện bộ phim

Giá trị của trường thuộc tính này là danh sách các công ty, về cơ bản thì khá giống với cột production_countries và nhóm em cũng xử lý dữ liệu tương tự như bên trên

```
In [29]: df['production_companies'] = df['production_companies'].apply(lambda x: len(x))
```

3.2.2.6. Cột original_language: ngôn ngữ gốc mà bộ phim được quay

Giá trị của trường thuộc tính này được để ở dạng object như: ‘en’, ‘hi’,...
Nhóm em vẽ đồ thị để xem số lượng các giá trị trong trường thuộc tính này



Hình 5. Đồ thị hiển thị độ phổ biến của ngôn ngữ trong các bộ phim

Nhìn vào đồ thị có thể thấy được phim sử dụng tiếng anh chiếm số lượng áp đảo. Do đó nhóm em sẽ chuyển những hàng mang giá trị ‘en’ về 1, các hàng còn lại mang giá trị 0.

```
In [32]: df['lang_english'] = df['original_language'].apply(lambda x: 1 if x=='en' else 0)
```

3.2.2.7. Cột homepage: trang chủ của bộ phim

Giá trị của cột này chứa đường link trang chủ của bộ phim. Cách xử lý của nhóm em là nhưng hàng rỗng sẽ mang giá trị 0, còn lại mang giá trị 1.

Kết quả sau khi convert như sau:

```
In [37]: df['homepage'].value_counts()

Out[37]: 0    3466
         1    1915
         Name: homepage, dtype: int64
```

3.2.2.8. Cột status: bộ phim đã phát hành hay chưa

Sau khi kiểm tra nhóm em thấy rằng số lượng bộ phim phát hành chiếm số lượng áp đảo, lên đến 99,9%, được xem gần như là toàn bộ các hàng đều mang giá trị "Released" (đã phát hành)

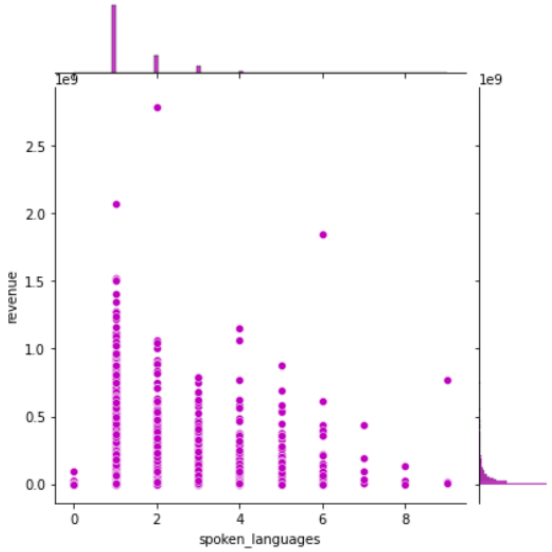
```
In [38]: df['status'].value_counts()

Out[38]: Released           5377
         Rumored               2
         Post Production       2
         Name: status, dtype: int64
```

Do đó nhóm em quyết định không sử dụng cột này vào mục đích huấn luyện và xóa cột này đi

3.2.2.9. Cột spoken_language: danh sách các ngôn ngữ trong bộ phim

Giá trị của trường dữ liệu này là danh sách tên các ngôn ngữ. Do đó nhóm em đã chuyển về thành số lượng các ngôn ngữ trong bộ phim. Các chuyển tương tự các trường thuộc tính bên trên



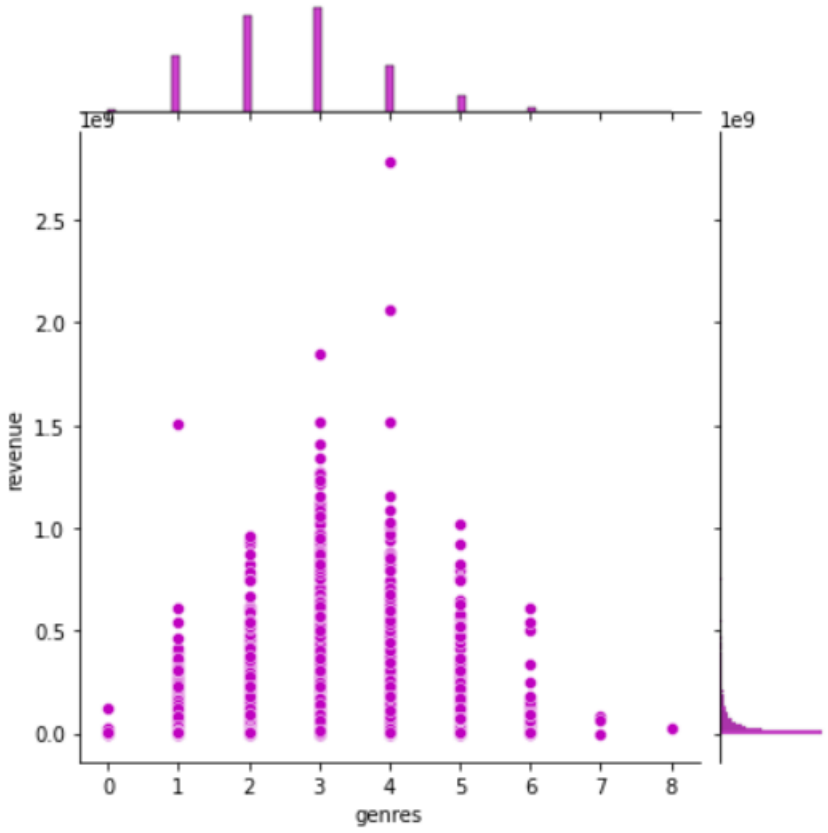
Hình 6. Đồ thị hiển thị sự tương quan giữa revenue và spoken_languages

Nhóm em đã biểu diễn sự tương quan giữa số lượng ngôn ngữ và doanh thu, có thể thấy số lượng ngôn ngữ càng nhiều thì doanh thu càng giảm.

Ngoài ra còn có thể nhận ra 1 số giá trị khác biệt so với phần còn lại. Phần này sẽ được trình bày trong mục xử lý nhiễu sau

3.2.2.10. Cột genres: thể loại

Giá trị của trường dữ liệu được để ở dạng danh sách và cũng có những hàng bị thiếu giá trị. Để phục vụ việc huấn luyện, nhóm em sẽ điền vào những hàng thiếu 1 danh sách rỗng, và chuyển về số thể loại trong 1 bộ phim. Các làm tương tự như các trường thuộc tính bên trên



Hình 7. Đồ thị tương quan giữa revenue với genres

Nhìn vào đồ thị bên trên có thể thấy doanh thu cao tập trung vào các bộ phim có số thể loại từ 1 đến 6. Và có thể thấy một số dữ liệu cao khác biệt so với phần còn lại

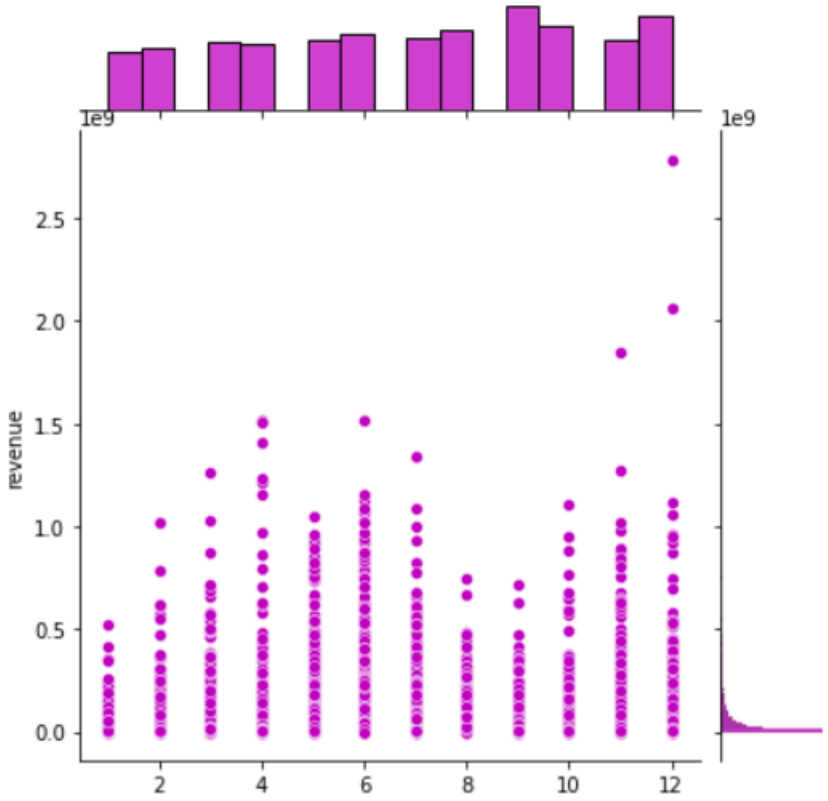
3.2.2.11. Cột release_date: thời gian phát hành

Giá trị của trường thuộc tính này để ở dạng "năm-tháng-ngày". Sau khi thảo luận nhóm em cho rằng năm phát hành và tháng phát hành có thể ảnh hưởng đến doanh thu

Ví dụ trong thực tế: những bộ phim phát hành sớm vào trước năm 2000 có doanh thu thấp hơn do vấn đề lạm phát và độ phổ biến phim ngày trước không cao. Và những bộ phim phát hành vào dịp kì hè, nghỉ lễ có xu hướng thu hút khách hơn vào các dịp khác

Nhóm em tạo 1 cột mới đặt tên là "year" dùng để lưu năm

```
In [125]: # Tách năm
df['year'] = pd.to_datetime(df['release_date'], errors='coerce').apply(lambda x: str(x).split('-')[0] if x != np.nan else np.nan)
```



Hình 8. Đồ thị hiển thị sự tương quan giữa month và revenue

Nhìn vào đồ thị bên trên có thể thấy doanh thu cao tập trung vào các tháng 4 5 6 (các tháng dịp nghỉ hè) và 10 11 12 (các tháng vào dịp cuối năm). Do đó nhóm em đã tạo ra 1 cột mới là "**holiday**" với các tháng 4 5 6 10 11 12 mang giá trị 1, các tháng còn lại mang giá trị 0.

Kết quả sau khi thực hiện chuyển đổi như sau:

```
In [132]: df = df.drop('month', axis=1)
          df['holiday'].value_counts()

Out[132]: 1    2743
          0    2638
          Name: holiday, dtype: int64
```

3.2.2.12. Cột runtime: thời lượng bộ phim

Sau khi kiểm tra giá trị của cột runtime nhóm em thấy rằng có 1 hàng không chứa giá trị. Do đó nhóm đã tiến hành điền vào ô đó giá trị trung bình của cột runtime

```
In [133]: df.runtime.isnull().sum()

Out[133]: 1

In [134]: df['runtime'] = df['runtime'].fillna(df['runtime'].mean())
```

3.2.2.13. Cột popularity: điểm phổ biến của phim được đánh giá bởi TMDB

Do giá trị của trường dữ liệu này lúc đầu ở dạng object. Do đó nhóm tiến hành convert về dạng float64

```
In [135]: df['popularity'] = df['popularity'].apply(convert_to_float).astype('float')
```

3.2.2.14. Cột vote_average: điểm xếp hạng trung bình của bộ phim

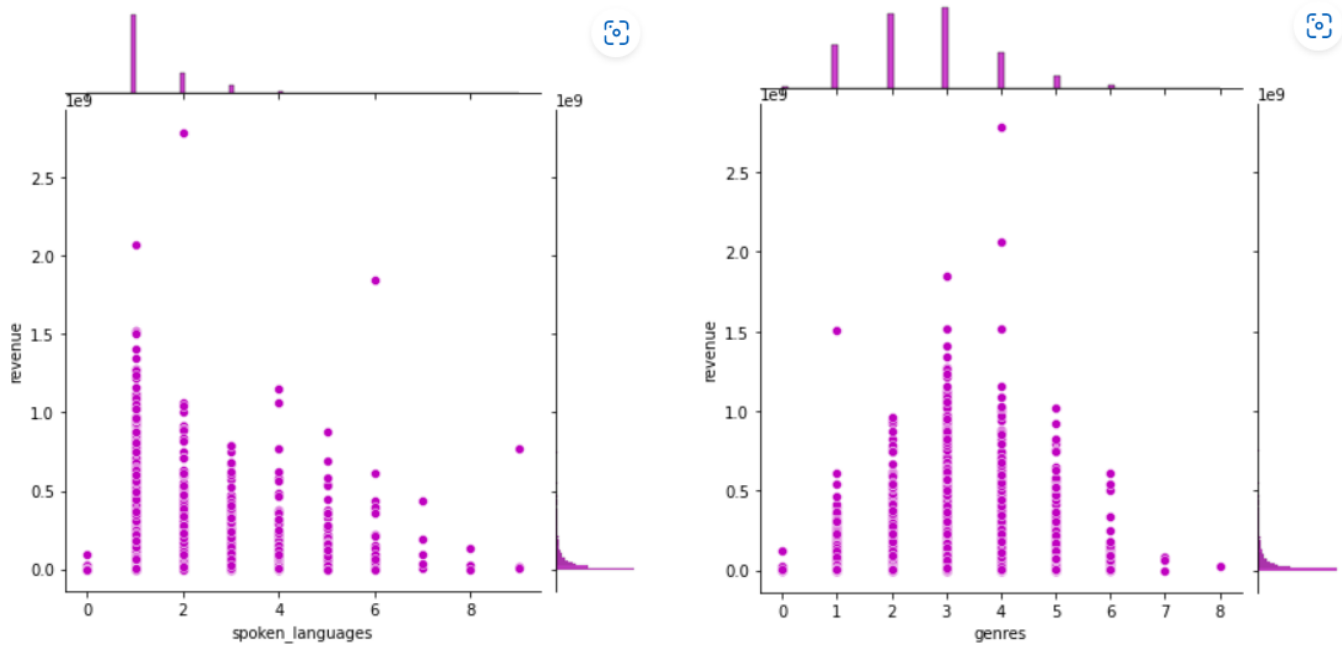
Dữ liệu của trường thuộc tính này khá ổn khi không có hàng nào thiếu giá trị, kiểu dữ liệu ở dạng float64 nên không cần điều chỉnh

3.2.2.15. Cột vote_count: số lượng người vote của bộ phim

Cột vote_count cũng tương tự như cột vote_average nên không cần điều chỉnh gì nhiều

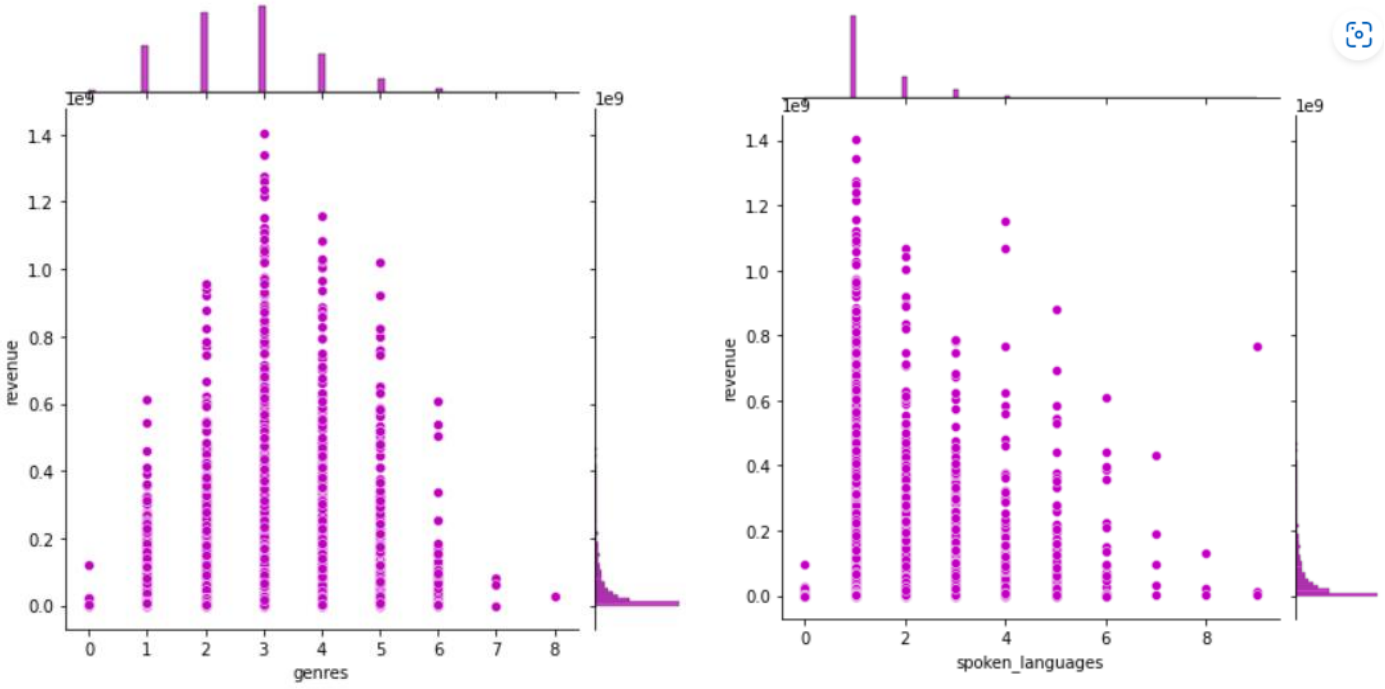
3.3. Xử lý nhiễu

Như đã trình bày bên trên, sau khi biểu sự tương quan giữa genres và revenue, spoken_language với revenue, nhóm đã phát hiện ra một vài dữ liệu khác biệt so với phần còn lại



Hình 9. Đồ thị hiển thị dữ liệu nhiễu

Nhìn vào đồ thị có thể thấy dữ liệu nhiễu có giá trị revenue lớn hơn 1.5 * 10^9. Nên nhóm đã xóa các giá trị đó đi. Sau khi xóa thu được kết quả như sau:



Hình 20. Đồ thị hiển thị sau khi đã xóa nhiễu

Sau khi xóa thì thấy dữ liệu đã ổn hơn trước.

Khi nhìn vào biểu đồ thể hiện sự tương quan giữa spoken_languages với revenue, ở spoken_ languages = 9 thấy có một điểm cao khác biệt so với phần còn lại. Nhóm tiến hành tìm và xóa thủ công.

Table "default" - Rows: 5375 Spec - Columns: 14 Properties Flow Variables

Row ID	budget	genres	homep...	popularity	product...	product...	runtime	spoken...	vote_a...	vote_c...	lang_e...	year	holiday	reve...
Row4882	1000000	3	1	6.404	2	1	107	9	7.1	285	0	2015	1	1777043
Row3807	4317946	1	1	12.453	6	3	106	9	7.3	76	0	2011	0	14624826
Row3469	200000000	3	1	16.699	4	2	158	9	5.6	4994	1	2009	1	769653595
Row3689	18000000	4	0	4.558	4	2	338	8	6.7	50	1	2010	1	871279

Đến đây nhóm đã có tập dữ liệu tương đối ổn định. Nhóm sẽ sử dụng 2 tập dữ liệu chứa nhiễu và không chứa nhiễu để train và so sánh kết quả.

4. Xây dựng mô hình và kết quả

4.1. Hồi quy tuyến tính

Bước 1: load dataset từ file dữ liệu data_unnoise.csv đã được tiền xử lí dữ liệu trước đó.

```
df = pd.read_csv('data_noise.csv')
len(df)
```

```
5381
```

Bước 2: Tiến hành chia tập dữ liệu ra thành 2 tập là train và test: Chia tập dữ liệu thành 2 tập con là X và y. Với X là các biến dự báo (input) để dự đoán doanh thu phim và y (output) là doanh thu thực tế của bộ phim đó

```
X, y = df.drop('revenue', axis=1), df['revenue']
```

Bước 3: Chúng ta chia tập dữ liệu X, y thành 2 tập trainset X_train, y_train với kích thước chiếm 80% tập dữ liệu và tập testset X_test, y_test có kích thước 20% tập dữ liệu (sử dụng thư viện train_test_split)

```
[15] from sklearn.model_selection import train_test_split

[17] X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, test_size=0.2)

print(f"Độ dài của tập train là {len(X_train)}")

print(f"Độ dài của tập test là {len(X_test)}")
```

```
Độ dài của tập train là 4304
Độ dài của tập test là 1077
```

Bước 4: Train mô hình (cách thức hoạt động đã được giải thích rõ ở mục 2.1)

```
[35] lr = LinearRegression() #gọi tới hàm LR()

lr_model.fit(X_train, y_train)
```

```
LinearRegression()
```

Bước 5: Thực hiện dự đoán:

```
In [29]: test_score_lr = lr_model.score(X_test, y_test)
         print(f'Test score of linear regression: {test_score_lr}')
```

```
Test score of linear regression: 0.745778275666596
```

Trong đó, predict_test_lr là kết quả output thu được khi thử dự đoán trên lr_model với input là X_test sau khi đã train cho chính model này bằng (X_train, y_train).

Qua đó, ta thấy đấy được độ hiệu quả của mô hình này với tập dữ liệu testset là 74,57%

Bước 6: Tính sai số tuyệt đối trung bình

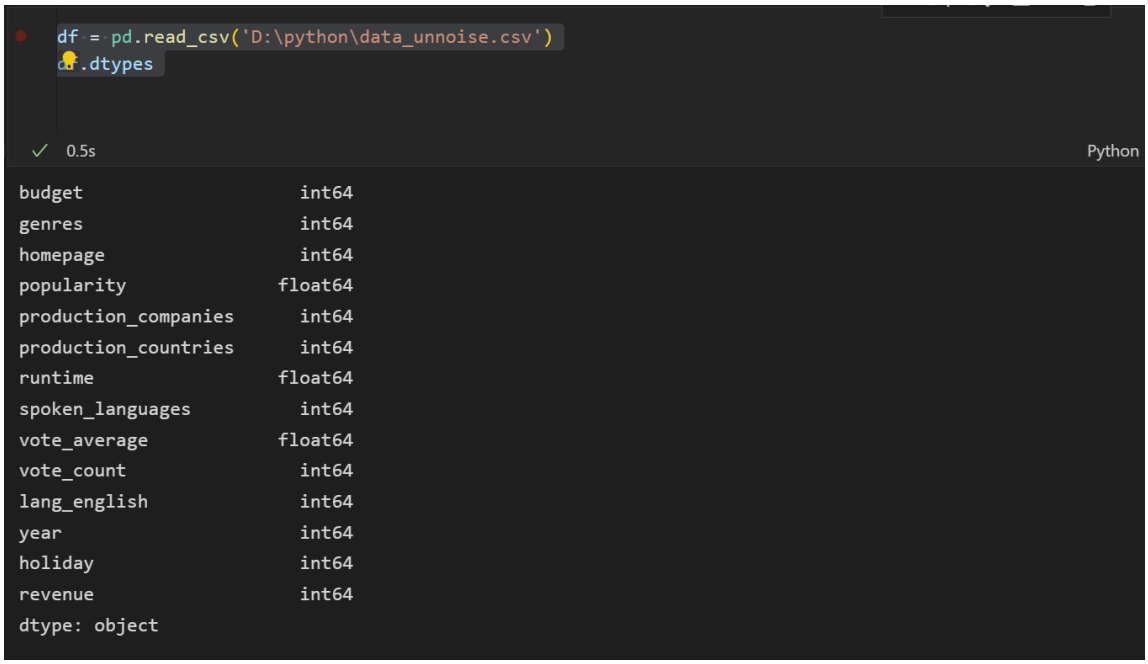
Kết quả sai số là khoảng 44,16 triệu đô

```
MAEofTest = getMAE(y_values, predict_test_lr)
print(f"MAE of test set is {MAEofTest}")
```

```
MAE of test set is 44163629.242623836
```

4.2. Hồi quy Rigde

Bước 1: Đầu tiên, như mô hình hồi quy tuyến tính, nhóm load dataset từ file dữ liệu data_unnoise.csv đã được tiền xử lý dữ liệu trước đó.



Bước 2**:** tiến hành chia giá trị của 2 thuộc tính budget và revenue cho 10^6 để giảm giá trị của vector trọng số w.

```
df['budget'] = df['budget'] / 1e+6
df['revenue'] = df['revenue'] / 1e+6
df.describe()
```

✓ 0.4s

	budget	genres	homepage	popularity	production_companies	production_countries	runtime	spoken_languages	vote_average	vote_count	lang_english	year	holiday	revenue
count	5374.000000	5374.000000	5374.000000	5374.000000	5374.000000	5374.000000	5374.000000	5374.000000	5374.000000	5374.000000	5374.000000	5374.000000	5374.000000	5374.000000
mean	30.866970	2.602531	0.355043	9.710683	2.938035	1.360811	109.774624	1.479903	6.271902	722.476926	0.891701	1999.743952	0.509118	88.200895
std	39.674773	1.126224	0.478571	13.563602	2.188934	0.791189	21.990667	0.920569	0.927915	1218.011351	0.310787	15.923695	0.499963	154.152374
min	0.000001	0.000000	0.000000	0.000001	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1915.000000	0.000000	0.000001
25%	5.000000	2.000000	0.000000	5.432929	1.000000	1.000000	95.000000	1.000000	5.700000	77.000000	1.000000	1994.000000	0.000000	7.000565
50%	17.000000	3.000000	0.000000	8.380698	2.000000	1.000000	106.000000	1.000000	6.300000	276.000000	1.000000	2004.000000	1.000000	29.863699
75%	40.000000	3.000000	1.000000	11.696451	4.000000	2.000000	120.000000	2.000000	6.900000	795.000000	1.000000	2011.000000	1.000000	99.025405
max	380.000000	8.000000	1.000000	547.488298	26.000000	12.000000	338.000000	9.000000	9.100000	14075.000000	1.000000	2017.000000	1.000000	1405.403694

Bước 3: Tiến hành chia tập dữ liệu thành 2 tập trainset và testset: ta tiến hành chia tập dữ liệu thành 2 tập con X và y. Với X là các giá trị đầu vào để dự đoán doanh thu phim và y là doanh thu thực tế của bộ phim đó. Chúng ta chia tập dữ liệu X, y thành 2 tập trainset X_train, y_train có kích thước 80% tập dữ liệu và tập testset X_test, y_ test có kích thước 20% tập dữ liệu ban đầu bằng cách dùng thư viện train_test_split.

tách dataset thành train và test

```
X, y = df.drop('revenue', axis=1), df['revenue']
idx = np.arange(X.shape[0])
X_train, X_test, y_train, y_test, idx_train, idx_test = train_test_split(X, y, idx, test_size=0.2, random_state=20)
```

[5] ✓ 0.5s

Bước 4: Tìm hệ số alpha tối ưu cho mô hình hồi quy Ridge

Để tìm ra hệ số phù hợp nhất ứng với thành phần điều chuẩn thì chúng ta sẽ thực hiện grid search trên không gian tham số . Tiêu chuẩn lựa chọn mô hình sẽ là metric của sai số được đo lường trên tập kiểm tra là nhỏ nhất, thông thường metric này được lựa chọn là MSE(Mean_squared_error).

Chúng ta sẽ sử dụng 1000 giá trị alpha từ 1 đến 1000 để tính giá trị của MSE trong mô hình hồi quy Rigde, từ đó ta tìm được giá trị alpha tương ứng với MSE lớn nhất. Với mô hình này, chúng ta giá trị alpha tối ưu = 14 với Bestscore = sqrt(MSE) = 79.97

```python
# Khởi tạo phân chia tập train/test cho mô hình. Đánh dấu các giá trị thuộc tập train là -1 và tập test là 0
split_index = [-1 if i in idx_train else 0 for i in idx]
ps = PredefinedSplit(test_fold=split_index)

# Khởi tạo pipeline gồm 2 bước, 'scaler' để chuẩn hoá đầu vào và 'model' là bước huấn luyện
pipeline = Pipeline([
                    ('scaler', StandardScaler()),
                    ('model', Ridge())
])

# GridSearch mô hình trên không gian tham số alpha
search = GridSearchCV(pipeline,
                      {'model__alpha':np.arange(1, 1000,1)}, # Tham số alpha từ 1->10 huấn luyện mô hình
                      cv = ps, # validation trên tập kiểm tra
                      scoring="neg_mean_squared_error", # trung bình tổng bình phương phần dư
                      verbose=3
                      )


search.fit(X, y)
print(search.best_estimator_)
print('Best core: ', np.sqrt(np.abs(search.best_score_)))
```

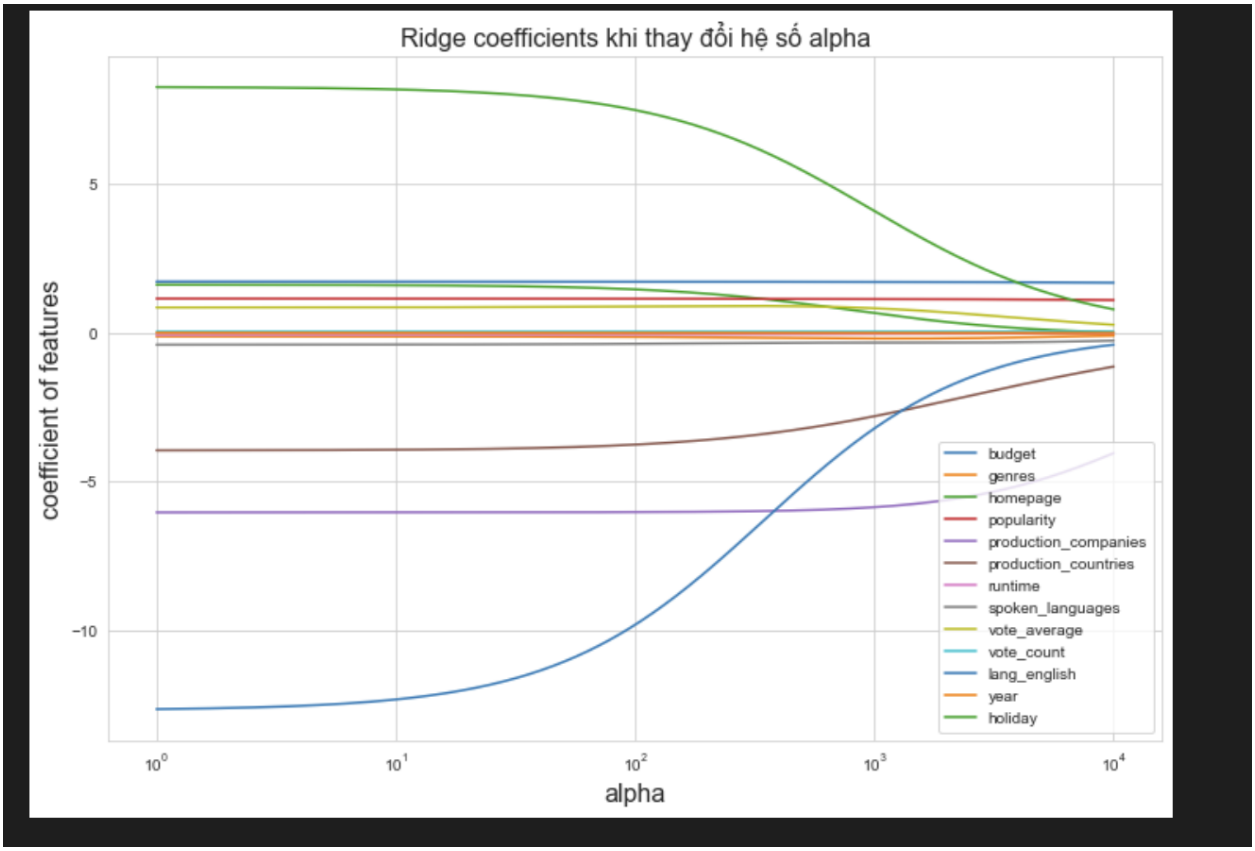
```
...  Output exceeds the size limit. Open the full output data in a text editor
Fitting 1 folds for each of 999 candidates, totalling 999 fits
[CV 1/1] END .................model__alpha=1;, score=-6396.677 total time=   0.0s
[CV 1/1] END .................model__alpha=2;, score=-6396.664 total time=   0.0s
[CV 1/1] END .................model__alpha=3;, score=-6396.652 total time=   0.0s
[CV 1/1] END .................model__alpha=4;, score=-6396.641 total time=   0.0s
[CV 1/1] END .................model__alpha=5;, score=-6396.631 total time=   0.0s
[CV 1/1] END .................model__alpha=6;, score=-6396.623 total time=   0.0s
[CV 1/1] END .................model__alpha=7;, score=-6396.615 total time=   0.0s
[CV 1/1] END .................model__alpha=8;, score=-6396.608 total time=   0.0s
[CV 1/1] END .................model__alpha=9;, score=-6396.602 total time=   0.0s
[CV 1/1] END ................model__alpha=10;, score=-6396.598 total time=   0.0s
[CV 1/1] END ................model__alpha=11;, score=-6396.594 total time=   0.0s
[CV 1/1] END ................model__alpha=12;, score=-6396.591 total time=   0.0s
[CV 1/1] END ................model__alpha=13;, score=-6396.590 total time=   0.0s
[CV 1/1] END ................model__alpha=14;, score=-6396.589 total time=   0.0s
[CV 1/1] END ................model__alpha=15;, score=-6396.589 total time=   0.0s
[CV 1/1] END ................model__alpha=16;, score=-6396.591 total time=   0.0s
[CV 1/1] END ................model__alpha=17;, score=-6396.593 total time=   0.0s
[CV 1/1] END ................model__alpha=18;, score=-6396.596 total time=   0.0s
[CV 1/1] END ................model__alpha=19;, score=-6396.601 total time=   0.0s
[CV 1/1] END ................model__alpha=20;, score=-6396.606 total time=   0.0s
[CV 1/1] END ................model__alpha=21;, score=-6396.612 total time=   0.0s
[CV 1/1] END ................model__alpha=22;, score=-6396.620 total time=   0.0s
[CV 1/1] END ................model__alpha=23;, score=-6396.628 total time=   0.0s
[CV 1/1] END ................model__alpha=24;, score=-6396.637 total time=   0.0s
...
[CV 1/1] END ...............model__alpha=998;, score=-6673.078 total time=   0.0s
[CV 1/1] END ...............model__alpha=999;, score=-6673.514 total time=   0.0s
Pipeline(steps=[('scaler', StandardScaler()), ('model', Ridge(alpha=14))])
Best core:  79.97867873756577
```

Sau đây là biểu đồ thể hiện sự ảnh hưởng của giá trị alpha đến các thành phần của vector trọng số w khi giá trị alpha thay đổi từ 1 đến 1000

```python
# Thay đổi alphas từ 1 --> 1000
n_alphas = 1000
alphas = np.logspace(0, 4, n_alphas)
coefs = []

# Huấn luyện model khi alpha thay đổi.
for a in alphas:
    ridge = Ridge(alpha=a, fit_intercept=False)
    ridge.fit(X_train, y_train)
    coefs.append(ridge.coef_)

# Hiển thị kết quả mô hình cho các hệ số alpha
plt.figure(figsize= (12, 8))
ax = plt.gca()
ax.plot(alphas, coefs)
ax.set_xscale('log')
ax.set_xlim(ax.get_xlim())
plt.xlabel('alpha', fontsize=16)
plt.ylabel('coefficient of features', fontsize=16)
plt.legend(X.columns)
plt.title('Ridge coefficients khi thay đổi hệ số alpha', fontsize=16)
plt.axis('tight')
plt.show()
```



Bước 5: Xây dựng mô hình hồi quy Rigde với giá trị alpha được xác định ở bước trên.

Trong mô hình này ta có alpha là hằng số phạt, các tác dụng làm hạn chế độ lớn của vector trọng số w.

+ rr.coef_ : lưu trữ hệ số tương ứng đối với mỗi đặc điểm trong vectơ trọng số w.

+ r2_score: đánh giá độ chính xác của mô hình hồi quy Ridge.

In [21]:
```
rr = Ridge(alpha= 1)
rr.fit(X_train, y_train)
print(rr.coef_)

pred_test_rr= rr.predict(X_test)

print("\n")
print(f'Sai số MAE: {mean_absolute_error(y_test, pred_test_rr) } triệu đô')

test_score_ridge = rr.score(X_test, y_test)
print("The test score for ridge model is {}".format(test_score_ridge))
```

```
[  1.76364072  -0.9514754    5.09008252   1.14762802  -5.13062954
  -3.80129417  -0.07435737  -0.4476153   -0.55069086   0.06142763
 -18.1587289   -0.42284177   8.59901918]


Sai số MAE: 44.060430998809196 triệu đô
The test score for ridge model is 0.7122112329871508
```

Kết quả thu được sau khi chạy mô hình 71.22% với sai số dự đoán 44.06 triệu đô với tập test.

4.3. Random Forest

Bước 1: Load dữ liệu và chia tập dữ liệu thành X_train, X_test, y_train, y_test với tập train chiếm 80% tập dữ liệu ban đầu, tập test chiếm 20%

Bước 2: Train mô hình: vì các giá trị mặc định hầu như đều đã tối ưu rồi, nên nhóm quyết định sử dụng giá trị mặc định để train.

Tuy nhiên, vì nhóm sử dụng **sai số tuyệt đối trung bình MAE** để đánh giá kết quả mô hình, nên tham số criterion (dùng để đánh giá tiêu chí) sẽ mang giá trị là "absolute_error" thay vì giá trị mặc định là "squared_error"

Cần phải xác định criterion vì một mô hình được train theo tiêu chí phù hợp sẽ chính xác hơn so với một mô hình train theo tiêu chí ngẫu nhiên

Train

In [88]:
```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators=100, criterion='absolute_error', random_state=0)
rf_model = regressor.fit(X_train, y_train)
```

Bước 3: Dự đoán và thu được kết quả với sai số khoảng 38,28 triệu đô và hệ số xác định R^2 là 80%

```
In [90]: regressor.score(X_test, y_test)
Out[90]: 0.8032538026525833

In [91]: y_real = y_test.values
         print(f'sai so mae: {sai_so_MAE(y_real, y_predict)} $')
         sai so mae: 38281655.07042792 $
```

4.4. Hàm đánh giá độ chính xác mô hình

Ở bên trên nhóm em có sử dụng hàm để đánh giá độ chính xác của mô hình, trong phần này nhóm sẽ giải thích về nó

Hệ số xác định R^2 hay độ chính xác được tính như sau:

$$(R)^2 = 1 - \frac{u}{v}$$

với $u = \sum_{k=0}^{n}(y_{true} - y_{pred})^2$

$v = \sum_{k=0}^{n}(y_{true} - y_{true\_trung\_bình})^2$

Giá trị R^2 nằm trong khoảng từ 0 đến 1. Giá trị tốt nhất có thể có là 1, lúc này các kết quả dự đoán trùng khớp với kết quả thực tế

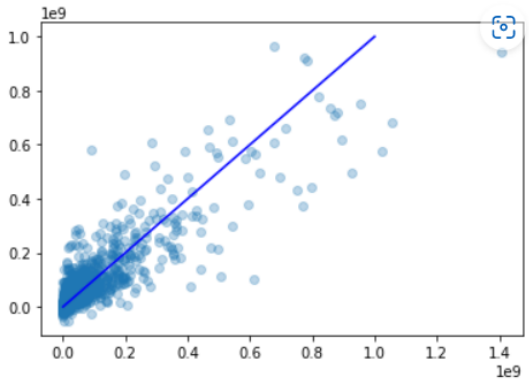
4.5. Kết quả của ba mô hình

4.5.1. Đồ thị trực quan hóa kết quả

4.5.1.1. Hồi quy tuyến tính

Sử dụng đồ thị scatter để có thấy được mối quan hệ giữa kết quả thực và kết quả dự đoán.

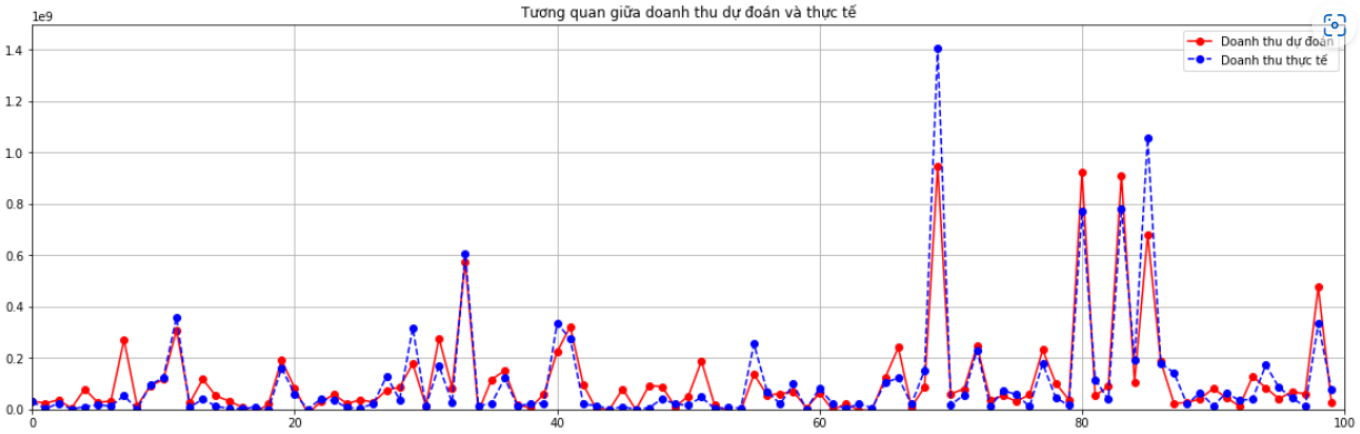
```
In [30]: plt.scatter(y_test, predict_test_lr, alpha=0.3)
         plt.plot([1, 1000000000], [1, 1000000000], color='blue')
Out[30]: [<matplotlib.lines.Line2D at 0x2cf93482220>]
```



Hình 11. Đồ thị scatter về mối quan hệ giữa kết quả thực tế và dự đoán (Hồi quy tuyến tính)

Như vậy thì các phim có dự đoán tốt nhất sẽ là các điểm gần nhất với đường thẳng y = x (y_real = y_predict).

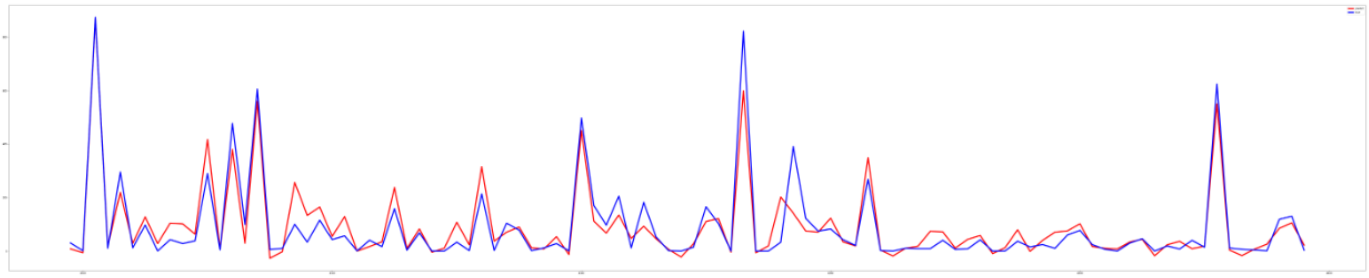
Tiếp theo nhóm sẽ trực quan hóa 100 bộ phim đầu tiên để biết được độ chênh lệch giữa kết quả dự đoán so với thực tế



Hình 12. Đồ thị thể hiện sự chênh lệch giữa kết quả dự đoán và thực tế (Hồi quy tuyến tính)

4.5.1.2. Hồi quy Ridge

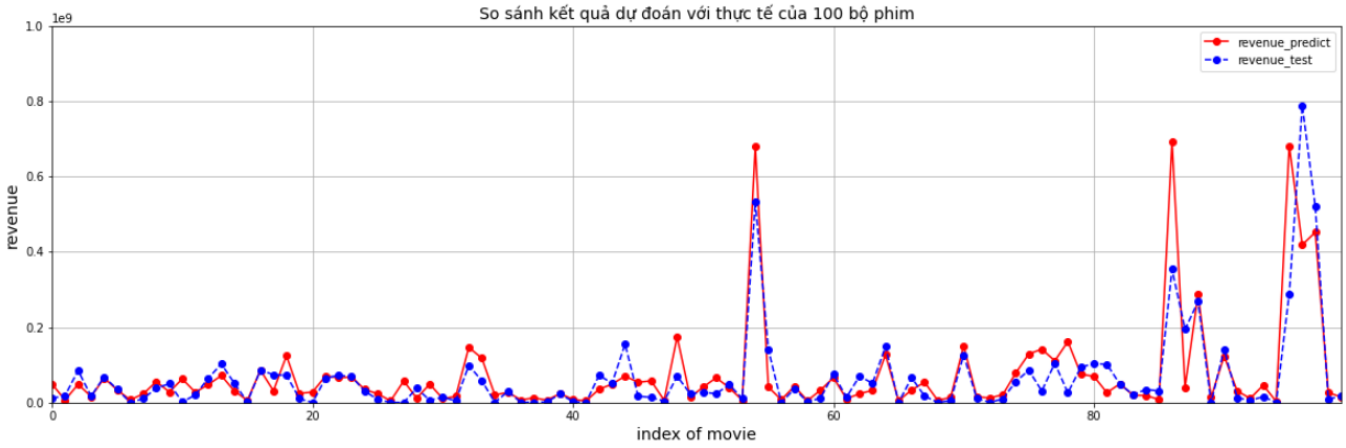
Dưới đây là biểu đồ so sánh giữa doanh thu thực tế và doanh thu đã được dự đoán ở 100 bộ phim đầu tiên của y_tets bởi mô hình



Hình 13. Đồ thị thể hiện sự chênh lệch giữa kết quả dự đoán và thực tế (Hồi quy Ridge)

4.5.1.3. Random Forest

Dưới đây là biểu đồ so sánh giữa doanh thu thực tế và doanh thu dự đoán của 100 bộ phim đầu tiên trong tập y_test



Hình 14. Đồ thị thể hiện sự chênh lệch giữa kết quả dự đoán và thực tế (Random Forest)

4.5.2. Sai số tuyệt đối trung bình MAE

Tiêu chí đánh giá	Hồi quy tuyến tính	Hồi quy Ridge	Random Forest
MAE	44,16 triệu đô	44.06 triệu đô	38.28 triệu đô
Độ chính xác	74,57%	71,22%	80.32%

5. Các vấn đề khó khăn gặp phải trong quá trình thực hiện

Các thành viên lần đầu tiếp xúc với Machine Learning nên không có nhiều kinh nghiệm. Hướng giải quyết: Chúng em đã tìm tòi nghiên cứu các tài liệu giấy cũng như video bài giảng liên quan đến môn học

Tập dữ liệu được thu thập từ internet còn chứa nhiều nhiễu lỗi, các thuộc tính bị thiếu nhiều bản ghi. Hướng giải quyết: xử lý nhiễu theo nhiều cách khác nhau

Một số thuộc tính trong dataset có chi phí lớn nên khó thu thập

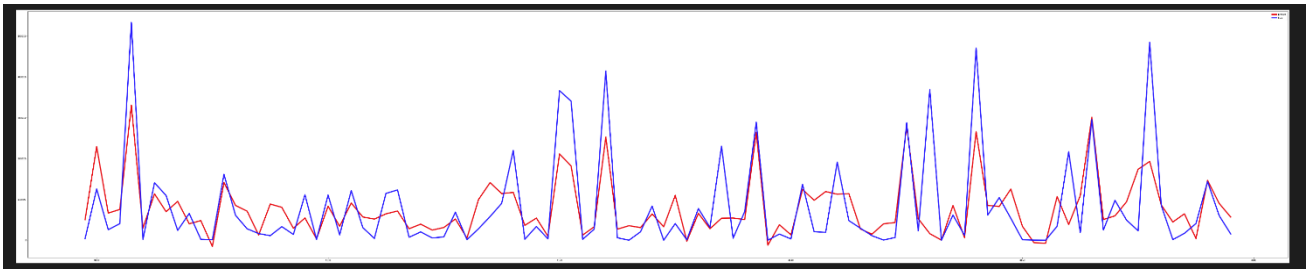
Sự biến động của thị trường, khủng hoảng tài chính, lạm phát kiến đồng tiền mất giá chưa được thể hiện trong dataset nên các thuộc tính chưa thể hiện được hết các giá trị của bộ phim.

6. Các tranh luận, khám phá, để cử cho việc phát triển trong tương lai

6.1. Các tranh luận

Nhóm đã sử dụng 3 mô hình học máy với các tập dữ liệu khác nhau. Nhóm có tranh luận với nhau về việc chỉ sử dụng **vài thuộc tính** được cho là quan trọng, với việc sử dụng **nhiều thuộc tính hơn** nhưng phải xử lý dữ liệu nhiều và kĩ càng hơn. Vì các thành viên trong nhóm đều mới bắt đầu tìm hiểu về học máy nên chưa có nhiều kinh nghiệm trong việc xử lý dữ liệu.

Ban đầu nhóm quyết định chỉ sử dụng các trường thuộc tính: budget, runtime, month, year, popularity để train. Tuy nhiên kết quả được biểu diễn trên đồ thị thì thấy lệch rất nhiều.



Nhóm đưa ra kết luận rằng do chưa xử lý các dữ liệu nhiễu. Nhưng sau khi xử lý bằng các phương pháp IQR và quy tắc 3 độ lệch chuẩn để xử lý nhưng kết quả vẫn không khả quan hơn.

Do đó nhóm đã quyết định xem xét lại toàn bộ tập dữ liệu. lựa chọn và xử lý kĩ càng hơn. Và cuối cùng cho ra được file "data_unnoise.csv" được dùng để train bên trên. Kết quả đã được cải thiện hơn rất nhiều

6.2. Hướng phát triển

Như đã trình bày trong báo cáo bên trên, có thể thấy kết quả của mô hình bị ảnh hưởng rất nhiều bởi tập dữ liệu, do đó trong tương lai có thể có các cách xử lý dữ liệu nhiễu tốt hơn hiện tại giúp cải thiện mô hình

TÀI LIỆU THAM KHẢO

1. Web tham khảo cách xử lý dữ liệu: https://www.kaggle.com/code/rounakbanik/the-story-of-film/notebook
2. Blog của Pham Dinh Khanh: https://phamdinhkhanh.github.io/deepai-book/ch_ml/RandomForest.html
3. Web VTI TechBlog: https://vtitech.vn/xgboost-bai-1-toan-canh-ve-ensemble-learning-phan-1/
4. Web Machine Learning Mastery: https://machinelearningmastery.com/ridge-regression-with-python/
5. Tài liệu pdf: http://www.cs.cmu.edu/~ggordon/10725-F12/slides/08-general-gd.pdf
6. Web datacamp: Regularization Tutorial: Ridge, Lasso & Elastic Net Regression | DataCamp
7. Kaggle: https://www.kaggle.com/questions-and-answers/196752
8. Trên towardsdatascience.com: https://towardsdatascience.com/random-forest-regression-5f605132d19d
9. Web scikit về Random Forest: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
10. Web aws.amazon.com: https://aws.amazon.com/vi/what-is/linear-regression/
11. Web scikit-learn.org về r2_score: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html
12. Tài liệu pdf về hồi quy Ridge: https://arxiv.org/pdf/1509.09169.pdf
13. Web tham khảo random forest: https://www.keboola.com/blog/random-forest-regression
14. Web tham khảo GridSearchCV: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html