NETFLIX MOVIES & TV SHOWS

Data source: https://www.kaggle.com/datasets/shivamb/netflix-shows

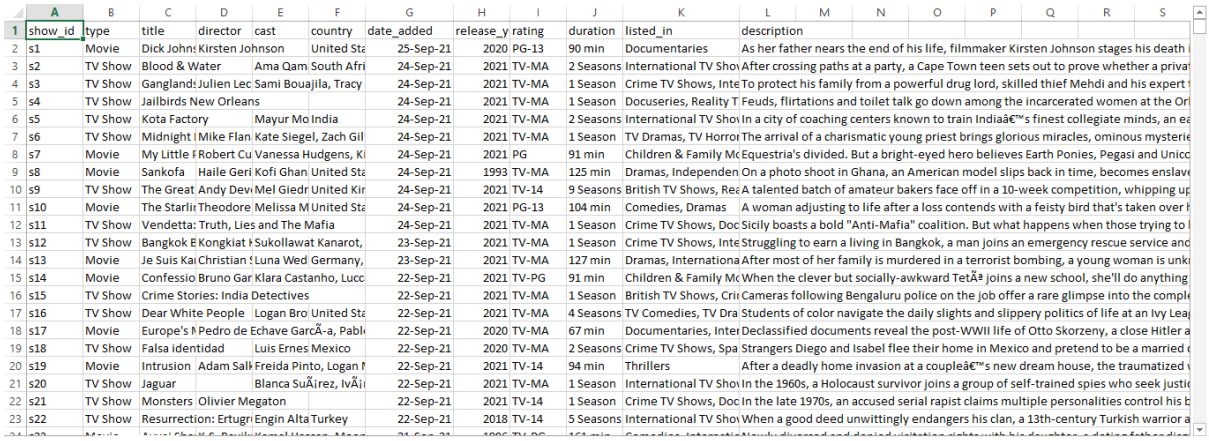
TỔNG QUAN

Chào mọi người, mình là Đỗ Hoàng Lâm với một project nhỏ dựa trên dataset Netflix Movies & TV Shows.

MÔ TẢ DATASET

Dataset này gồm 8807 hàng giá trị tương ứng 12 cột bao gồm:

1. show_id: id của các bộ phim
2. type: chủng loại (movie hoặc tv show) của bộ phim
3. title: tên của bộ phim
4. director: tên các đạo diễn
5. cast: tên các diễn viên chính
6. country: quốc gia xuất xứ của phim
7. date_added: ngày xuất hiện trên Netflix
8. release_year: năm phát hành
9. rating: MPAA rating của phim
10. duration: độ dài tính bằng phút với Movies và tính bằng season với TV Shows
11. listed_in: các category của phim
12. description: mô tả ngắn về phim



	A	B	C	D	E	F	G	H	I	J	K	L
1	show_id	type	title	director	cast	country	date_added	release_y	rating	duration	listed_in	description
2	s1	Movie	Dick Johns	Kirsten Johnson		United Sta	25-Sep-21	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmmaker Kirsten Johnson stages his death
3	s2	TV Show	Blood & Water		Ama Qam	South Afri	24-Sep-21	2021	TV-MA	2 Seasons	International TV Sho	After crossing paths at a party, a Cape Town teen sets out to prove whether a privat
4	s3	TV Show	Ganglands	Julien Lec	Sami Bouajila, Tracy		24-Sep-21	2021	TV-MA	1 Season	Crime TV Shows, Inte	To protect his family from a powerful drug lord, skilled thief Mehdi and his expert
5	s4	TV Show	Jailbirds New Orleans				24-Sep-21	2021	TV-MA	1 Season	Docuseries, Reality T	Feuds, flirtations and toilet talk go down among the incarcerated women at the Or
6	s5	TV Show	Kota Factory		Mayur Mo	India	24-Sep-21	2021	TV-MA	2 Seasons	International TV Sho	In a city of coaching centers known to train Indiaâ€™s finest collegiate minds, an ea
7	s6	TV Show	Midnight I	Mike Flan	Kate Siegel, Zach Gil		24-Sep-21	2021	TV-MA	1 Season	TV Dramas, TV Horror	The arrival of a charismatic young priest brings glorious miracles, ominous mysteri
8	s7	Movie	My Little F	Robert Cu	Vanessa Hudgens, Ki		24-Sep-21	2021	PG	91 min	Children & Family Mc	Equestria's divided. But a bright-eyed hero believes Earth Ponies, Pegasi and Unicc
9	s8	Movie	Sankofa	Haile Geri	Kofi Ghan	United Sta	24-Sep-21	1993	TV-MA	125 min	Dramas, Independen	On a photo shoot in Ghana, an American model slips back in time, becomes enslav
10	s9	TV Show	The Great	Andy Dev	Mel Giedr	United Kir	24-Sep-21	2021	TV-14	9 Seasons	British TV Shows, Rea	A talented batch of amateur bakers face off in a 10-week competition, whipping up
11	s10	Movie	The Starli	Theodore	Melissa M	United Sta	24-Sep-21	2021	PG-13	104 min	Comedies, Dramas	A woman adjusting to life after a loss contends with a feisty bird that's taken over
12	s11	TV Show	Vendetta: Truth, Lies and The Mafia				24-Sep-21	2021	TV-MA	1 Season	Crime TV Shows, Doc	Sicily boasts a bold "Anti-Mafia" coalition. But what happens when those trying to
13	s12	TV Show	Bangkok E	Kongkiat K	Sukollawat Kanarot,		23-Sep-21	2021	TV-MA	1 Season	Crime TV Shows, Inte	Struggling to earn a living in Bangkok, a man joins an emergency rescue service and
14	s13	Movie	Je Suis Kai	Christian S	Luna Wed	Germany,	23-Sep-21	2021	TV-MA	127 min	Dramas, Internationa	After most of her family is murdered in a terrorist bombing, a young woman is unki
15	s14	Movie	Confessio	Bruno Gar	Klara Castanho, Lucc		22-Sep-21	2021	TV-PG	91 min	Children & Family Mc	When the clever but socially-awkward TetÃ¤ joins a new school, she'll do anything
16	s15	TV Show	Crime Stories: India Detectives				22-Sep-21	2021	TV-MA	1 Season	British TV Shows, Cri	Cameras following Bengaluru police on the job offer a rare glimpse into the compl
17	s16	TV Show	Dear White People	Logan Bro		United Sta	22-Sep-21	2021	TV-MA	4 Seasons	TV Comedies, TV Dra	Students of color navigate the daily slights and slippery politics of life at an Ivy Lea
18	s17	Movie	Europe's M	Pedro de	Echave GarcÃ-a, Pabl		22-Sep-21	2020	TV-MA	67 min	Documentaries, Inter	Declassified documents reveal the post-WWII life of Otto Skorzeny, a close Hitler a
19	s18	TV Show	Falsa identidad		Luis Ernes	Mexico	22-Sep-21	2020	TV-MA	2 Seasons	Crime TV Shows, Spa	Strangers Diego and Isabel flee their home in Mexico and pretend to be a married
20	s19	Movie	Intrusion	Adam Salk	Freida Pinto, Logan M		22-Sep-21	2021	TV-14	94 min	Thrillers	After a deadly home invasion at a coupleâ€™s new dream house, the traumatized
21	s20	TV Show	Jaguar		Blanca SuÃ¡rez, IvÃ¡n		22-Sep-21	2021	TV-MA	1 Season	International TV Sho	In the 1960s, a Holocaust survivor joins a group of self-trained spies who seek justi
22	s21	TV Show	Monsters	Olivier Megaton			22-Sep-21	2021	TV-14	1 Season	Crime TV Shows, Doc	In the late 1970s, an accused serial rapist claims multiple personalities control his b
23	s22	TV Show	Resurrection: Ertugr		Engin Alta	Turkey	22-Sep-21	2018	TV-14	5 Seasons	International TV Sho	When a good deed unwittingly endangers his clan, a 13th-century Turkish warrior a

MỤC TIÊU

Với tập dữ liệu này, mình sẽ sử dụng Python để thực Predictive và Prescriptive Analysis, cụ thể về mỗi phần mình sẽ trình bày ở phần sau.

I. PREDICTIVE (CLASSIFICATION)

Dataset cung cấp một trường thông tin có tên description, đây là một dòng mô tả ngắn (trung bình khoảng 150 ký tự) sơ lược về nội dung bộ phim. Nhận ra điều này, mình muốn tạo ra một mô hình để có thể xác định được thể loại của phim dựa trên dòng mô tả ngắn đó thông qua việc sử dụng Python.

Phương pháp thực hiện:

Sau quá trình preprocessing dữ liệu, tệp dữ liệu được tách ra thành training set và test set, input sẽ là description của bộ phim, output sẽ là một mảng bao gồm n phần tử (với n là tổng số lượng thể loại của tập dữ liệu), giá trị của các phần tử sẽ bằng 1 ứng với các category thực tế của bộ phim, và bằng 0 cho tất cả các giá trị còn lại. Hay nói cách khác, đây là dạng bài toán Multilabels Classification, tức là trong một tập hợp các features, predict để đưa ra một hoặc nhiều features thích hợp với input.

Để thực hiện bài toán này, mình thực hiện thuật toán Binary Relevance theo phương thức One-vs-Rest, đây là thuật toán tách bài toán lớn chọn m labels trong n features thành n bài toán nhỏ hơn, mỗi bài toán là 1 bài Binary Classification với 2 biến là feature đó và toàn bộ feature còn lại để từ đó tìm ra các feature đúng. Rồi từ đó có thể tính performance của từng bài toán nhỏ rồi gộp lại tính trung bình cho bài toán lớn. Sau đó sẽ dùng các thuật toán classification bao gồm Logistic Regression, Naive Bayes, Linear SVM từ đó tìm xem thuật toán cho performance tốt nhất.

Với dạng bài toán Multilabel Classification, việc đo lường performance sẽ hơi khác so với các bài toán classification dạng Binary (predict ra 0 hoặc 1) hay Multiclass (predict chỉ cho ra 1 giá trị của 1 feature trong nhiều features) bởi ở 2 dạng trên ở mỗi biến chỉ có 1 giá trị, do đó nó chỉ có đúng và sai, nhưng ở Multilabel chính vì việc có thể có nhiều giá trị nên xuất hiện thêm một trường hợp nữa là đúng một phần (tức đúng một vài giá trị chứ không đúng hết), do đó ngoài các metric thường thấy như precision, recall, F1 tương tự như các bài toán Binary hay Multiclass, việc đo lường performance sẽ tập trung hơn vào các chỉ số bao gồm Accuracy tính theo Hamming Score (độ tương thích trung bình giữa kết quả dự đoán với kết quả thực tế), AUC ROC score (khả năng phân biệt đúng sai của mô hình), Exact Match Ratio (đây là accuracy score đối với các bài toán thông thường, riêng đối với dạng Multilabel thì nó thể hiện tỉ lệ giữa số dự đoán chính xác hoàn toàn so với tổng số dự đoán) và Hamming Loss (số giá trị dự đoán sai chia cho tổng số giá trị dự đoán).

Vì biến đầu vào là description, là dữ liệu dạng chuỗi chữ viết, do đó đòi hỏi phải xử lý nó bằng các phương pháp xử lý ngôn ngữ tự nhiên (NLP) để đưa về dạng gọn nhất, giúp máy tính dễ dàng xử lý nhất, sau đó dùng kỹ thuật TF-IDF để vectorize chuỗi ký tự đó thành vector để có thể đem vào mô hình.

Về NLP, nó bao gồm nhiều task nhỏ với mục tiêu là biến câu từ bình thường thành dạng rút gọn nhất, sao cho vẫn giữ nguyên được ý nghĩa và sắc thái, nhằm mục đích giúp cho máy tính có khả năng "hiểu" nội dung của các tài liệu, bao gồm các sắc thái ngữ cảnh của ngôn ngữ bên trong chúng một cách dễ dàng hơn. Có rất nhiều task NLP có thể sử dụng, tuy nhiên trong bài này nhóm chỉ sử dụng 6 task thông qua 6 function (trong đó có 5 function tự định nghĩa) bao gồm:

- lower: biến toàn bộ thành dạng chữ thường.
- decontract: mở các dạng từ viết tắt thành dạng chuẩn, ví dụ như: won't thành will not; I'm thành I am; 've thành have;…
- clearPunc: xóa toàn bộ các dấu trong đoạn
- keepAlpha: xóa tất cả các ký tự không phải alphabet (chỉ giữ các ký tự từ a đến z)
- removeStopwords: xóa những stop words, đó là những từ thường xuyên xuất hiện trong câu nhưng không cung cấp nhiều ý nghĩa, ví dụ như: a, about, above, after, again,… (xem đầy đủ tại: https://www.ranks.nl/stopwords)
- stemming: đây là quá trình rút gọn từ thành gốc từ của nó gắn với hậu tố và tiền tố hoặc gốc của các từ được gọi là bổ đề. Nói một cách đơn giản, nó rút gọn một từ thành từ gốc của nó. Ví dụ: cared thành care, caring thành care, cares thành care,…

Nói thêm về TF-IDF, đây là một kĩ thuật sử dụng trong khai phá dữ liệu văn bản, nó chuyển các dữ liệu văn bản thô sang một ma trận tần số xuất hiện của các từ. Tần số này được sử dụng để đánh giá tầm quan trọng của một từ trong một văn bản. Giá trị cao thể hiện độ quan trọng cao và nó phụ thuộc vào số lần từ xuất hiện trong văn bản nhưng bù lại bởi tần suất của từ đó trong tập dữ liệu.

Quá trình thực hiện:

Đầu tiên cần kiểm tra số dòng không null của dữ liệu bằng method df.info:

```
df.info()
```

Kết quả:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64 
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
```

Như đã nói ở trên, phần bài làm của mình sẽ tập trung vào các cột title, listed_in, description, cả 3 cột nói trên đều không chứa giá trị null. Do đó ta đi vào phần tiếp theo là Data Preprocessing.

Data Preprocessing:

Với mong muốn tạo ra được 1 dataframe với 2 cột title, description và được nối theo sau bởi n cột (với n là tổng số lượng category mà dataset có), giá trị của mỗi cột đó sẽ bằng 1 ứng với category thực tế tương ứng với title và description của bộ phim đó, và bằng 0 ứng với các category khác, mình thực hiện như sau:

```
df2 = pd.DataFrame(index = df.index)
for idx, cats in enumerate(df["listed_in"]):
    for cat in cats.split(", "):
        df2.loc[idx,cat] = 1
df2.fillna(0,inplace = True)
df2
```

Kết quả:

	Documentaries	International TV Shows	TV Dramas	TV Mysteries	Crime TV Shows	TV Action & Adventure	Docuseries	Reality TV	Romantic TV Shows	TV Comedies	...	Science & Nature TV	Teen TV Shows	Cult Movies	TV Shows	Faith & Spirituality	LGBTQ Movies	Stand-Up Comedy	Movies
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8802	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
8803	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8804	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8805	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8806	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

8807 rows × 42 columns

Sau đó nối df2 vừa tạo rồi với cột title và description của bảng ban đầu, ta thu được bảng như mong muốn:

```
description_category = pd.concat([df[['title','description']], df2], axis=1)
description_category.head()
```

Kết quả:

	title	description	Documentaries	International TV Shows	TV Dramas	TV Mysteries	Crime TV Shows	TV Action & Adventure	Docuseries	Reality TV	...	Science & Nature TV	Teen TV Shows	Cult Movies	TV Shows	Faith & Spirituality	LGBTQ Movies	Stand-Up Comedy	Movies
0	Dick Johnson Is Dead	As her father nears the end of his life, filmm...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	Blood & Water	After crossing paths at a party, a Cape Town t...	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	Ganglands	To protect his family from a powerful drug lor...	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	Jailbirds New Orleans	Feuds, flirtations and toilet talk go down amo...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	Kota Factory	In a city of coaching centers known to train I...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

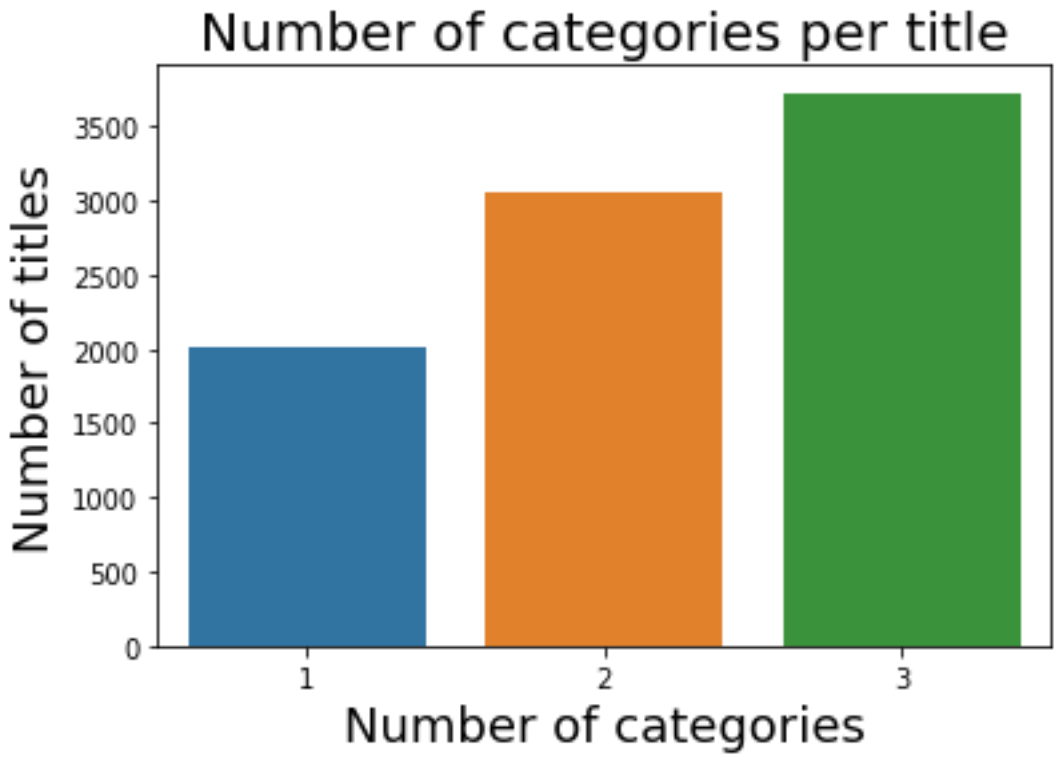
5 rows × 44 columns

Một số insight rút ra được từ bảng này:

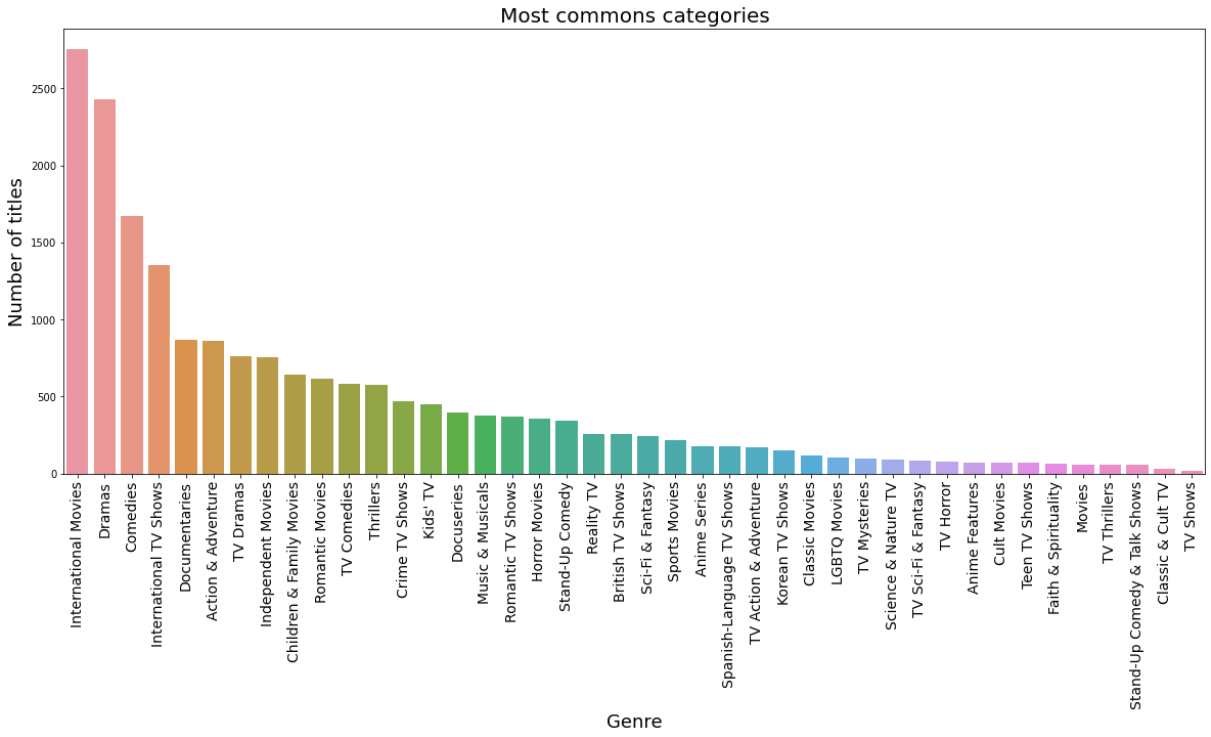
- Số lượng phim theo số lượng category:

```
1.0    2020
2.0    3058
3.0    3729
dtype: int64
```

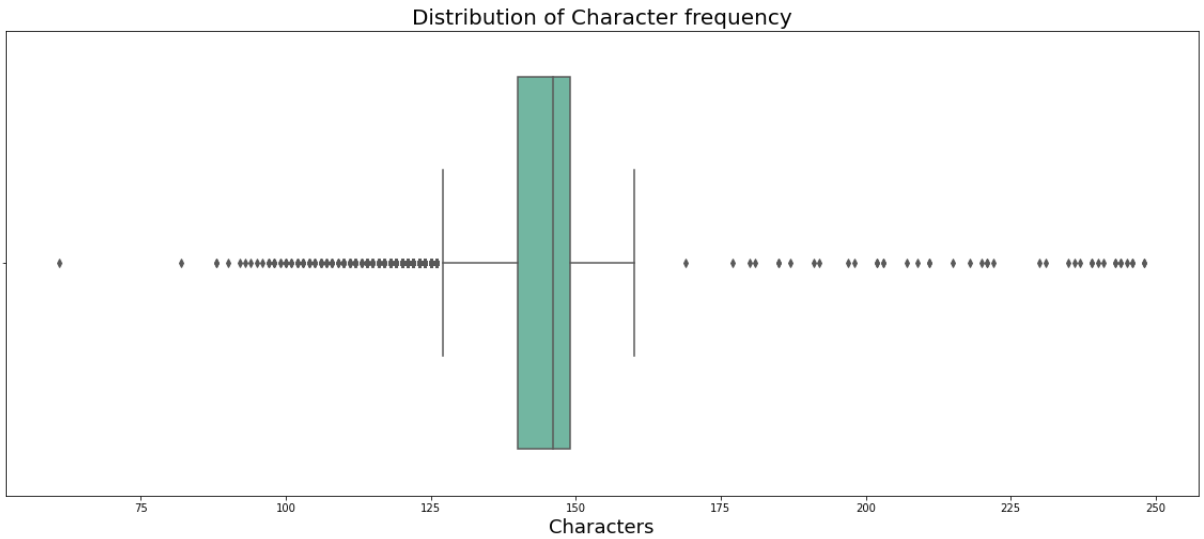
Trực quan hoá kết quả:



- Số lượng bộ phim ứng với mỗi category:



- Phân bổ tần số số lượng ký tự của phần description:



Qua đó có thể thấy được số ký tự trung bình rơi vào khoảng 140 - 150 ký tự, các description có số ký tự quá ít (khoảng dưới 125 ký tự) và quá cao (khoảng trên 160 ký tự) được xem là outliers của tập hợp.

Xử lý ngôn ngữ tự nhiên (NLP):

Như đã nói ở phần đầu, quá trình này sẽ thông qua 6 hàm (trong đó có 5 hàm tự định nghĩa) để thực hiện các task NLP nhằm rút gọn phần description. Các hàm được định nghĩa như sau:

- Hàm decontract:

```
import re
def decontract(sentence):
    sentence = re.sub("won't", "will not", sentence)
    sentence = re.sub("can\'t", "can not", sentence)
    sentence = re.sub("n\'t", " not", sentence)
    sentence = re.sub("\'re", " are", sentence)
    sentence = re.sub("\'s", " is", sentence)
    sentence = re.sub("\'d", " would", sentence)
    sentence = re.sub("\'ll", " will", sentence)
    sentence = re.sub("\'t", " not", sentence)
    sentence = re.sub("\'ve", " have", sentence)
    sentence = re.sub("\'m", " am", sentence)
    return sentence
```

- Hàm cleanPunc:

```
def cleanPunc(sentence): #clean punctuation
    cleaned = re.sub('[?|!|\'|"|#|.|,|;|)|(|\|/]','',sentence)
    cleaned = cleaned.strip()
    cleaned = cleaned.replace("\n"," ")
```

```
    return cleaned
```

- Hàm keepAlpha:

```
def keepAlpha(sentence): #only keep alphabet letters
    alpha_sent = ""
    for word in sentence.split():
        alpha_word = re.sub('[^a-z A-Z]+', '', word)
        alpha_sent += alpha_word
        alpha_sent += " "
    alpha_sent = alpha_sent.strip()
    return alpha_sent
```

- Hàm removeStopwords:

```
import nltk
from nltk.corpus import stopwords
from nltk.stem.snowball import SnowballStemmer
nltk.download('stopwords')

stopwords = set(stopwords.words('english'))

def removeStopWords(sentence): # remove stop words (common words which doesn't
add much information)
    filtered_sentence = ""
    for word in sentence.split():
        if word not in stopwords:
            filtered_sentence += word
            filtered_sentence += " "
    filtered_sentence = filtered_sentence.strip()
    return filtered_sentence
```

- Hàm stemming:

```
stemmer = SnowballStemmer("english")
def stemming(sentence): #reducing the word to its word stem
    stemSentence = ""
    for word in sentence.split():
        stem = stemmer.stem(word)
        stemSentence += stem
        stemSentence += " "
    stemSentence = stemSentence.strip()
    return stemSentence
```

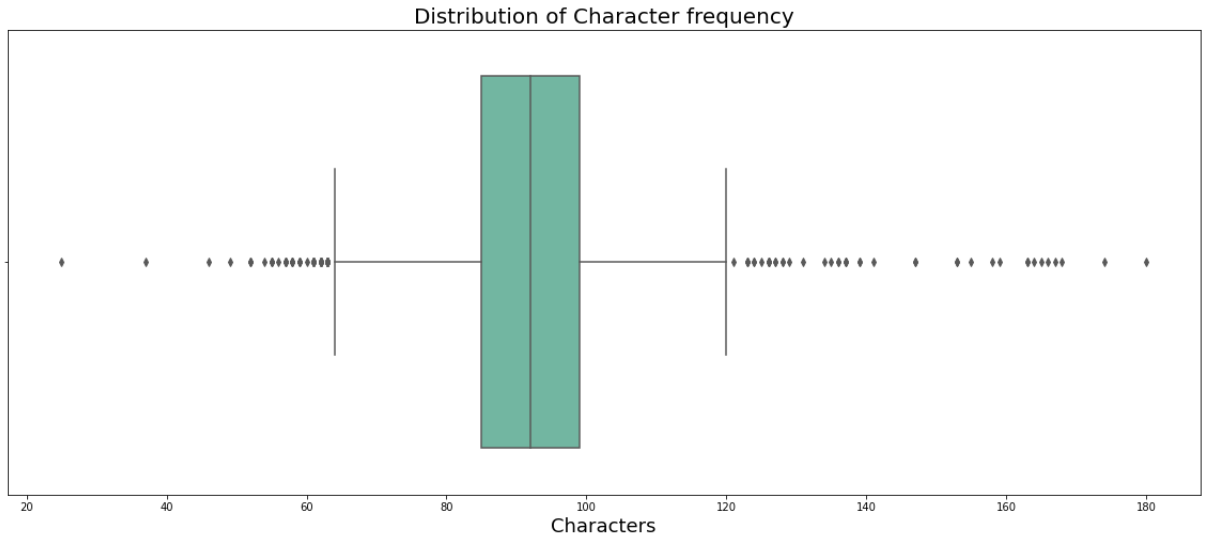
Sau khi định nghĩa xong, ta apply những hàm đó vào cột description trong bảng description_category và thu được kết quả:

	title	description	Documentaries	International TV Shows	Crime TV Shows	Docuseries	TV Dramas	Children & Family Movies	Dramas	British TV Shows	...	Romantic Movies	LGBTQ Movies	TV Sci-Fi & Fantasy	Sports Movies	Korean TV Shows	Science & Nature TV	Faith & Spirituality	Teen TV Shows
0	Dick Johnson Is Dead	father near end life filmmak kirsten johnson s...	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
1	Blood & Water	cross path parti cape town teen set prove whet...	0	1	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0
2	Ganglands	protect famili power drug lord skill thief meh...	0	1	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0
3	Jailbirds New Orleans	feud flirtat toilet talk go among incarcer wom...	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0
4	Kota Factory	citi coach center known train india finest col...	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

5 rows × 44 columns

Dữ liệu đã được xử lý xong để có thể đưa vào mô hình.

Còn dưới đây là boxplot thể hiện sự phân bổ của số lượng ký tự của các description trong dataset sau khi được NLP. Có thể thấy số lượng ký tự trung bình đã giảm đi đáng kể sau khi NLP, từ khoảng 140 - 150 ký tự xuống còn khoảng 85 – 100 với mean là 92, các description có số ký tự quá ít (khoảng dưới 62 ký tự) và quá cao (khoảng trên 120 ký tự) được xem là outliers của tập hợp.



Classification:

- Chia dataframe thành training set và test set: ở đây, ta thực hiện chia test size = 20% dataset, từ đó ta thu được X_train và y_train có 7045 dòng, X_test và y_test có 1762 dòng.

```
from sklearn.model_selection import train_test_split
```

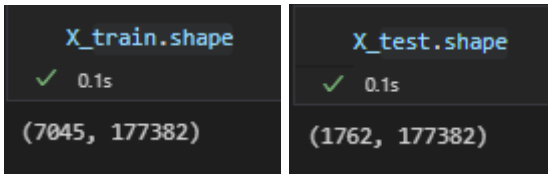
```
X = description_category['description']
y = description_category[description_category.columns[2:]]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 8, shuffle = True)
```

- Vectorize X_train và x_test bằng kỹ thuật TF-IDF:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(strip_accents='unicode', analyzer='word',
ngram_range=(1,3), norm='l2')
vectorizer.fit(X_train)
X_train = vectorizer.transform(X_train)
X_test = vectorizer.transform(X_test)
```

Kết quả:



Đến lúc này thì dữ liệu đã sẵn sàng, bây giờ nhóm sẽ thực hiện predict theo 3 thuật toán Logistic Regression, Naive Bayes, Linear SVM bằng phương pháp Binary Relevance (tạo vòng lặp để tách bài toán lớn thành nhiều bài toán dạng Binary nhỏ hơn). Mỗi thuật toán đều thực hiện thông qua các bước:

1. Tạo classifier.
2. Tạo một vòng lặp chạy qua từng category, tức description_category.columns[2:], hay nó sẽ thực hiện 42 lần.
3. Ở mỗi lần chạy trong vòng lặp sẽ thực hiện việc fit X_train với cột category chạy qua đó của y_train rồi thực hiện predict. Đồng thời sẽ tính các chỉ số accuracy và AUC ROC score của cột đó cộng gộp lại qua tất cả các cột, và lưu kết quả predict được của cột vào một dataframe.
4. Tính trung bình của các chỉ số vừa rồi bằng cách lấy tổng gộp có được sau vòng lặp chia cho 42, đồng thời tính các chỉ số Exact Match Ratio, Hamming Loss, precision, recall, f1 theo dataframe kết quả của predict và viết tất cả vào bảng result_df.

Kết quả dự đoán:

- Logistic Regression:

```
0.0    1487
1.0     174
2.0      99
3.0       2
dtype: int64
```

Trong 1762 kết quả có được, Logistic Regression cho 1487 kết quả có 0 category, 174 kết quả có 1 category, 99 kết quả có 2 category, 2 kết quả có 3 category.

- Naïve Bayes:

```
0.0    1757
1.0       2
2.0       3
dtype: int64
```

Trong 1762 kết quả có được, Naïve Bayes cho 1757 kết quả có 0 category, 2 kết quả có 1 category, 3 kết quả có 2 category, 0 kết quả cho 3 category.

- Linear SVM:

```
0.0    699
1.0    622
2.0    335
3.0     88
4.0     17
6.0      1
dtype: int64
```

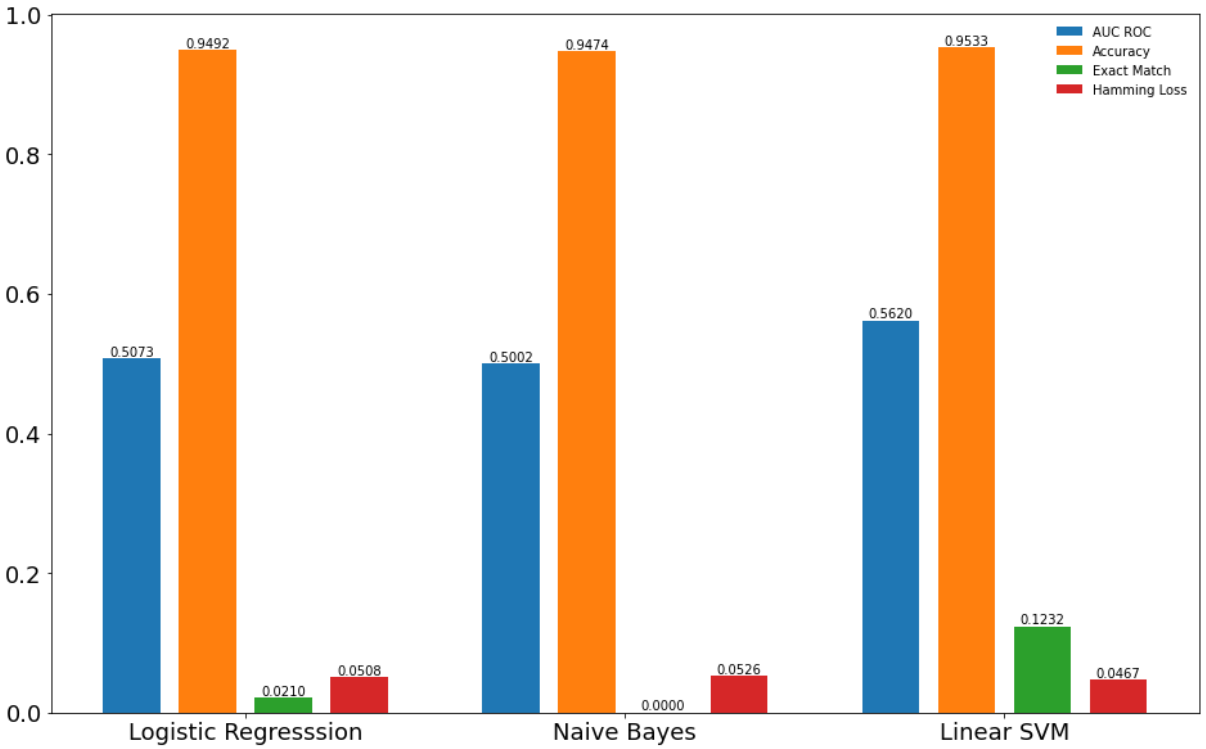
Trong 1762 kết quả có được, Linear SVM cho 699 kết quả có 0 category, 622 kết quả có 1 category, 335 kết quả có 2 category, 88 kết quả có 3 category, 17 kết quả có 4 category, 1 kết quả có 6 category.

- Các chỉ số đo lường performance:

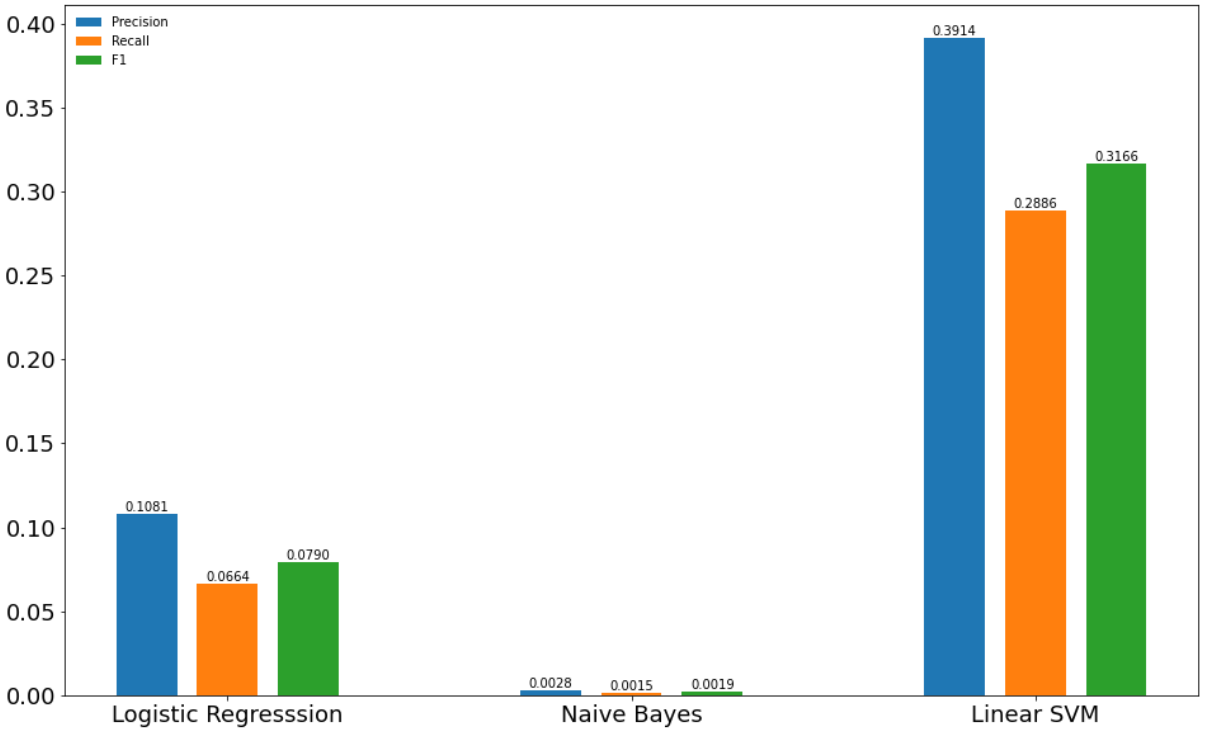
	Logistic Regresssion	Naive Bayes	Linear SVM
Avg AUC ROC Scores	0.507304	0.500176	0.561959
Avg Accuracy Scores	0.949246	0.947408	0.953273
Avg Exact Match Ratio	0.020999	0.000000	0.123156
Avg Hamming Loss	0.050754	0.052592	0.046727
Avg Precision Score	0.108116	0.002838	0.391411
Avg Recall Score	0.066402	0.001513	0.288593
Avg F1 Score	0.078982	0.001930	0.316641

Trực quan hoá kết quả:

- Các giá trị AUC ROC score, Accuracy, Exact Match Ratio, Hamming Loss:



- Các giá trị Precision, Recall, F1:



Nhận xét:

- Accuracy: như đã nói ở trên, đây là accuracy được tính theo Hamming Score, nó được tính bằng công thức: số features dự đoán đúng chia cho tổng số features dự đoán ở từng feature, rồi tính trung bình với tất cả features, tức là mức độ tương

đồng giữa kết quả nhận được từ model và kết quả thực sự tính theo phần trăm, tính trung bình qua toàn bộ kết quả. Ở trường hợp bài này, accuracy không thể hiện quá nhiều ý nghĩa, bởi ta có thể hiểu khi ta cho input là 1 description, thì output ta thu được sẽ là 1 mảng có 42 giá trị, trong đó phần lớn là số 0, bởi kết quả mong muốn là một chuỗi có 1-3 số 1 nên 39 số 0 của chuỗi nhận được gần như luôn đúng, khác biệt chỉ là nằm ở chỗ model có thể đưa ra được các số 1 được hay không (đây là tình trạng dữ liệu bị mất cân bằng – imbalanced data). Thực nghiệm đã chỉ ra rằng phần lớn kết quả thu được là 1 mảng có 42 số 0, khi ta đem nó so với kết quả thực tế là từ 1 đến 3 số 1 thì nó sẽ cho rằng kết quả không khác biệt quá nhiều. Chính bởi vậy accuracy tính theo Hamming Score không thể hiện được nhiều điều ở trường hợp bài này, có thể thấy điều này thông qua kết quả gần như tương đồng ở cả 3 model.

- AUC ROC score: hay Area Under the Curve là phần diện tích dưới đường ROC (Receiver Operator Characteristic). Hiểu một cách đơn giản, nếu AUC ROC score càng gần 1, tức khả năng để model dự đoán positive và negative 1 cách chính xác càng gần 100%, ngược lại, nếu AUC ROC score càng gần 0, tức khả năng để model dự đoán positive thành negative và negative thành positive càng cao. Điều đó có nghĩa khi AUC ROC score = 0.5, thì không thể phân biệt được các dự đoán của model sẽ là đúng hay sai, tức nó cho thấy khả năng dự đoán của mô hình rất tệ. Nhìn chung, một model cho kết quả AUC ROC score ở mức dưới 0.7 thì được xem là có khả năng phân biệt tệ, ở mức 0.7 - 0.8 là mức chấp nhận và có thể sử dụng được, và ở mức trên 0.8 là một model tốt, đương nhiên là càng gần 1 càng tốt. Ở bài này, có thể thấy AUC ROC score của model Logistic Regression và Naïve Bayes đều xấp xỉ 0.5, điều đó cho thấy khả năng dự đoán của 2 mô hình này rất tệ. Ở Linear SVM thì khá hơn một chút, tuy nhiên mức 0.56 vẫn còn là quá thấp.
- Exact Match Ratio: đây là con số thể hiện tỉ lệ giữa số dự đoán đúng hoàn toàn chia cho tổng số dự đoán, có thể thấy ở Linear SVM cho kết quả tốt nhất với 12.32% tỉ lệ cho kết quả đúng hoàn toàn, đây là một kết quả không quá tốt, tuy nhiên khi xét đến 2 model còn lại là Logistic Regression với 2.1% và Naïve Bayes với 0% thì vẫn là tốt hơn khá nhiều.

- Hamming Loss: đây là con số cho thấy tỉ lệ dự đoán sai, nó đúng bằng 1 trừ cho Accuracy, do đó thì nhận xét cũng tương tự như Accuracy.
- Precision, Recall, F1: nhìn vào biểu đồ ta có thể thấy sự khác biệt rất lớn giữa 3 model, điều này càng làm rõ thêm nhận định xuyên suốt rằng cả 3 model đều không đủ tốt, tuy nhiên nếu trong trường hợp bắt buộc thì vẫn có thể dùng được Linear SVC, còn Logistic Regression và đặc biệt là Naïve Bayes hoàn toàn không phù hợp để phục vụ cho bài toán dạng này.

Tổng kết:

Có thể thấy chỉ số chính để thể hiện performance của bài này là AUC ROC score và Exact Match Ratio (bởi accuracy theo Hamming score không thể hiện nhiều ý nghĩa trong trường hợp có 42 features mà chỉ chọn 1-3 features như bài này), và ở đó thì Linear SVM thể hiện tốt nhất, tuy nhiên vẫn là một model chưa tốt so với mặt bằng chung. Điều này diễn ra là bởi dữ liệu dùng để training chỉ có khoảng 7000 dòng, với 1 description sau khi thực hiện NLP có mean = 92 ký tự thì từng đó dữ liệu để train là chưa đủ nhiều để cover hết trường hợp, từ đó dẫn đến tình trạng nhiều kết quả có toàn bộ là giá trị 0, khiến việc dự đoán không có ý nghĩa và làm cho AUC ROC score thấp.

Để cải thiện điều này thì có 2 cách chính, thứ nhất là bổ sung thêm nhiều dữ liệu hơn nữa để có nhiều dữ liệu hơn để train, thứ hai là dùng những thuật toán deep learning mạnh hơn như Convolutional Neural Networks (CNNs), Long Short Term Memory Networks (LSTMs), Recurrent Neural Networks (RNNs). Theo mô hình do Rodolfo Saldanha[1] thực hiện, AUC ROC score thu được của các thuật toán deep learning nêu trên đều rơi vào khoảng 0.89, điều đó chứng tỏ performance của những model sử dụng deep learning là rất tốt.

[1] Rodolfo Saldanha (3/2020), Multilabel category prediction. Trích xuất từ: https://www.kaggle.com/code/rodsaldanha/multilabel-category-prediction#Preprocessing

II. PRESCRIPTIVE (RECOMMENDATION)

Mục tiêu của mình ở đây là có thể xây dựng 1 mô hình để recommend 10 bộ phim tương đồng nhất với 1 bộ phim đầu vào.

Phương pháp:

Với dữ liệu hiện có, mình sử dụng phương pháp Content-based Filtering. Về bản chất thì nó tương đối giống với Collaborative Filtering, chỉ khác là Collaborative Filtering sử dụng dữ liệu hành vi của các khách hàng để nhóm các khách hàng có hành vi tương tự lại với nhau, từ đó recommend những sản phẩm mà nhóm đó hay sử dụng, còn với Content-based Filtering sẽ sử dụng dữ liệu về thông tin của sản phẩm, từ đó nhóm các sản phẩm tương tự lại với nhau, và thực hiện recommend cho khách hàng các sản phẩm cùng nhóm với sản phẩm họ đã sử dụng. Chính bởi việc dataset chỉ cung cấp thông tin về các bộ phim nên chỉ có thể sử dụng Content-based Filtering.

Thuật toán:

Ở đây mình sử dụng Cosine Similarity để xác định sự tương quan giữa 2 vector, với mỗi vector là 1 bộ phim, sau đó đưa ra 10 vector có sự tương quan cao nhất với vector đầu vào, đó chính là 10 bộ phim có sự tương quan cao nhất đối với bộ phim đầu vào.

Mỗi vector được tạo thành từ các 5 thông tin title, director, cast, listed_in, description nối với nhau, ở giữa 2 thông tin liên tiếp ngăn cách bằng khoảng trắng, sau đó thực hiện xử lý ngôn ngữ tự nhiên (NLP) qua 6 bước tương tự như phần Predictive bên trên, sau đó sử dụng kỹ thuật TF-IDF (Term Frequency-Inverse Document Frequency) để biến chuỗi đó thành vector sử dụng cho Cosine Similarity (nhìn chung cách xử lý trước khi dùng Cosine Similarity để tạo matrix giống hệt như ở phần Predictive).

Quá trình thực hiện:

- Tạo ra bảng df_for_soup gồm 5 cột title, director, cast, listed_in, description của bảng gốc, rồi thực hiện fill các giá trị null bằng giá trị rỗng:

```
df_for_soup = df[['title', 'director', 'cast', 'listed_in', 'description']]
df_for_soup = df_for_soup.fillna("")
df_for_soup.head()
```

Kết quả:

	title	director	cast	listed_in	description
0	Dick Johnson Is Dead	Kirsten Johnson		Documentaries	As her father nears the end of his life, filmm...
1	Blood & Water		Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
3	Jailbirds New Orleans			Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...
4	Kota Factory		Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	International TV Shows, Romantic TV Shows, TV ...	In a city of coaching centers known to train I...

- Thực hiện tạo một cột mới (gọi là soup) là một chuỗi nối 5 giá trị từ các cột title, director, cast, listed_in, description, ở giữa 2 giá trị liên tiếp luôn cách nhau bằng 1 khoảng trắng, kể cả khi có giá trị cột nào đó là rỗng:

```
def create_soup(x):
    return x['title']+ ' '+ x['director']+ ' '+ x['cast']+ ' ' + x['listed_in']+
' ' + x['description']


df_for_soup['soup'] = df_for_soup.apply(create_soup, axis=1)
for i in range(4,1,-1):
    df_for_soup["soup"].replace(i*" ", " ", inplace=True, regex = True)
```

- Chỉ giữ lại cột title và cột soup, thực hiện NLP trên cột soup bằng 6 function gồm lower (chuyển về dạng chữ thường), decontract (mở các dạng từ viết tắt thành dạng chuẩn), clearpunc (xóa tất cả các thể loại dấu), keepalpha (chỉ giữ lại các ký tự alphabet), removeStopwords (xóa các stopword), stemming (chuyển các từ về dạng stem của nó) tương tự như phần Predictive bên trên.

  Kết quả:

	title	soup
0	Dick Johnson Is Dead	dick johnson dead kirsten johnson documentari ...
1	Blood & Water	blood water ama qamata khosi ngema gail mabala...
2	Ganglands	gangland julien leclercq sami bouajila traci g...
3	Jailbirds New Orleans	jailbird new orlean docuseri realiti tv feud f...
4	Kota Factory	kota factori mayur jitendra kumar ranjan raj a...

- Sau khi đã xử lý dữ liệu xong, bây giờ là lúc để vectorize dữ liệu bằng phương pháp TF-IDF như đã đề cập bên trên, sau đó fit và transform dữ liệu, rồi dùng nó để tạo một ma trận 8807x8807, với mỗi hàng và cột là một bộ phim, và giá trị tại vị trí giao nhau giữa hàng và cột chính là sự tương quan tính theo Cosine Similarity giữa 2 bộ phim đó, đương nhiên do vậy nên đường chéo của ma trận này toàn bộ sẽ nhận giá trị 1:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
vectorizer    =    TfidfVectorizer(strip_accents='unicode',    analyzer='word',
ngram_range=(1,3), norm='l2')
count_matrix = vectorizer.fit_transform(filledna['soup'])
```

```
cosine_sim = cosine_similarity(count_matrix, count_matrix)
```

Kết quả:

```
df_cosinesim = pd.DataFrame(cosine_sim)
df_cosinesim
```

✓ 0.1s

	0	1	2	3	4	5	6	7	8	9	...	8797	8798	8799	8800	8801	8802	8803	8804	8805	8806
0	1.000000	0.000000	0.004130	0.000000	0.001911	0.000000	0.000000	0.000000	0.003983	0.005315	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.001867	0.007342
1	0.000000	1.000000	0.009982	0.001825	0.010777	0.015614	0.002869	0.000681	0.005296	0.000376	...	0.005084	0.000651	0.006976	0.013292	0.000752	0.000366	0.014618	0.002641	0.000000	0.000707
2	0.004130	0.009982	1.000000	0.002266	0.016521	0.004221	0.005815	0.000367	0.009174	0.000000	...	0.001836	0.000351	0.000387	0.015974	0.000405	0.001525	0.016237	0.002543	0.000998	0.001425
3	0.000000	0.001825	0.002266	1.000000	0.008751	0.001940	0.003684	0.000000	0.012421	0.000000	...	0.000843	0.000000	0.000000	0.002365	0.000000	0.000000	0.003608	0.000000	0.004042	0.000000
4	0.001911	0.010777	0.016521	0.008751	1.000000	0.004557	0.000000	0.000396	0.009904	0.002016	...	0.001982	0.003846	0.000418	0.032760	0.005763	0.000000	0.024801	0.003604	0.005758	0.003530
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8802	0.000000	0.000366	0.001525	0.000000	0.000000	0.000389	0.002368	0.001145	0.000000	0.000470	...	0.000000	0.003894	0.004301	0.000474	0.004323	1.000000	0.000000	0.005207	0.004952	0.000845
8803	0.000000	0.014618	0.016237	0.003608	0.024801	0.012097	0.002093	0.000000	0.010469	0.000899	...	0.008815	0.000881	0.000000	0.016949	0.000000	0.000000	1.000000	0.020550	0.003211	0.000000
8804	0.000000	0.002641	0.002543	0.000000	0.003604	0.001750	0.000342	0.000756	0.000000	0.005829	...	0.000000	0.000997	0.000399	0.003666	0.000417	0.005207	0.020550	1.000000	0.009478	0.000393
8805	0.001867	0.000000	0.000998	0.004042	0.005758	0.003708	0.009939	0.000693	0.000000	0.004454	...	0.001640	0.000915	0.000366	0.000000	0.000383	0.004952	0.003211	0.009478	1.000000	0.001461
8806	0.007342	0.000707	0.001425	0.000000	0.003530	0.000426	0.001332	0.005260	0.003123	0.003244	...	0.000000	0.006978	0.004468	0.000916	0.004670	0.000845	0.000000	0.000393	0.001461	1.000000

8807 rows × 8807 columns

- Tạo 1 series có tên indices với index là cột title, value là index của bảng df_for_soup, bằng cách này ta sẽ dễ dàng lấy ra được index khi nhập vào 1 title bất kỳ:

```
indices = pd.Series(df_for_soup.index, index=df_for_soup['title'])
indices
```

Kết quả:

```
title
Dick Johnson Is Dead        0
Blood & Water               1
Ganglands                   2
Jailbirds New Orleans       3
Kota Factory                4
                         ...
Zodiac                   8802
Zombie Dumb              8803
Zombieland               8804
Zoom                     8805
Zubaan                   8806
Length: 8807, dtype: int64
```

- Tạo hàm get_recommendation với input là title của 1 bộ phim bất kỳ, hàm sẽ trả ra 1 series chứa 10 title của các bộ phim tương đồng với title đầu vào nhất. Dùng indices[title] để có được index của bộ phim đó, sau đó ta sort descending tại bảng df_cosinesim ở cột index vừa có, lấy ra các giá trị cao từ thứ 2 đến thứ 11 (vì thứ nhất đương nhiên chính là bộ phim đó với cosine similarity = 1). Đó chính là top 10 bộ phim giống title đầu vào nhất mà ta mong muốn:

```
def get_recommendation(title):
    idx = indices[title]
    sim_scores =
df_cosinesim[idx].sort_values(ascending=False).reset_index()[1:11]["index"]
    return df['title'].iloc[sim_scores]
```

- Thực hiện một số ví dụ, ta có kết quả như sau:

```
get_recommendation('Breaking Bad')
```

✓ 0.9s

```
2931                        Better Call Saul
3428       El Camino: A Breaking Bad Movie
5606                        Girlfriend's Day
8505                                The Show
1477                                Dare Me
6817                        Furthest Witness
3684                               Kakegurui
1955                The School Nurse Files
3744                                Unit 42
678     The Assassination of Gianni Versace
Name: title, dtype: object
```

```
get_recommendation("Shutter Island")
```

✓ 0.1s

```
8802              Zodiac
8272         The Departed
6826    Gangs of New York
6272     Before the Flood
8312          The Founder
3966       The Highwaymen
6886              Gothika
3391          The Command
8053         Solomon Kane
2860                Hugo
Name: title, dtype: object
```

```
get_recommendation("Zodiac")
```

✓ 0.2s

```
1358              Shutter Island
6886                      Gothika
8312                 The Founder
3966               The Highwaymen
6878    Good Night, and Good Luck
6200       Avengers: Infinity War
8003              Sherlock Holmes
8511          The Social Network
1612                        Chef
6147              American Psycho
Name: title, dtype: object
```

TÀI LIỆU THAM KHẢO

Rodolfo Saldanha (3/2020), Multilabel category prediction. Trích xuất từ: https://www.kaggle.com/code/rodsaldanha/multilabel-category-prediction#Preprocessing

Karan Shingde (10/2021), Netflix Recommendation and EDA. Trích xuất từ: https://www.kaggle.com/code/karan842/netflix-recommendation-and-eda?fbclid=IwAR1-B6MPoukwZ6vyjjr5-xXJqd85HVZa_cpOHa5qCvLUArRgRJvTWMadt9g#Content-based-filtering-on-multiple-metrics