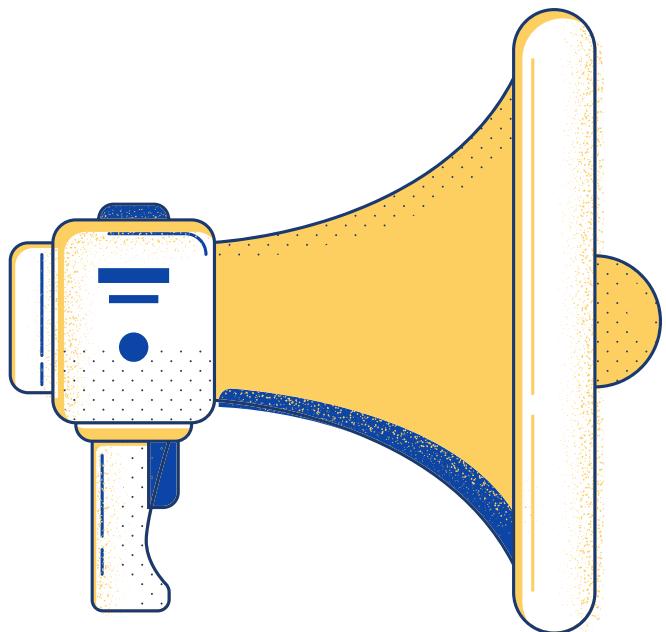


Nhóm 5 - 21_21

Nhập môn Khoa học Dữ liệu



Vấn đáp bảo vệ
đồ án cuối kỳ



Thành viên:

21120045 - Bùi Hồng Đăng

21120108 - Nguyễn Tiến Nhật

21120169 - Thái Chí Vỹ

21120201 - Bùi Đình Bảo

TABLE OF CONTENT

- 1 Giới thiệu đề tài
- 2 Thu thập dữ liệu
- 3 Tiền xử lý dữ liệu
- 4 Phân tích khám phá dữ liệu
- 5 Mô hình hóa dữ liệu
- 6 Kết luận và tự đánh giá



Giới thiệu đề tài và động lực thực hiện

1





Analyzing about movies on TMDB

<https://www.themoviedb.org/movie>



Scan me

The screenshot shows the TMDB website's movie search interface. On the left, there are filters for sorting, searching, and genres. The main area displays a grid of movie cards for "Popular MOVIES". Each card includes the movie title, its rating (in a green circle), and its release date. The cards are:

- REBEL MOON** - Rating: 65% (Dec 15, 2023)
- AQUAMAN** - Rating: 66% (Dec 20, 2023)
- ĐẦU TRƯỞNG SINH TỬ** - Rating: 73% (Nov 15, 2023)
- KẾ HOẠCH BẢO VỆ GIA ĐÌNH** - Rating: 74% (Dec 14, 2023)
- SILENT NIGHT** - Rating: 58% (Nov 30, 2023)
- Chương Trình Truyền Hình** (Thumbnail)
- Khởi Chiến** (Thumbnail)
- Đầu Trưởng Sinh Tử** (Thumbnail)
- Kế Hoạch Bảo Vệ Gia Đình** (Thumbnail)
- Silent Night** (Thumbnail)

ĐỘNG LỰC THỰC HIỆN

Hoàn thành đồ án cuối kỳ môn NMKHDL
Xử lý vấn đề bằng quy trình Khoa học Dữ liệu hoàn chỉnh
Đề tài về các bộ phim được chọn gần gũi, tạo hứng thú
Bộ dữ liệu trên website dễ thu thập

Thu thập dữ liệu

2



CRAWL DATA FLOW



Dùng thư viện BeautifulSoup.

Thu thập đủ 5000 dòng dữ liệu.

Các thuộc tính:

- Tên - Name
- Năm ra mắt - Released Year
- Thể loại - Genre
- Đạo diễn - Director
- Thời lượng - Runtime
- Điểm số - Score
- Trạng thái - Status
- Ngôn ngữ - Language
- Ngân sách - Budget
- Doanh thu - Revenue

Lưu dataframe vào một file .csv
cho các bước tiếp theo.

Raw Data

	Name	Released year	Genre	Director	Runtime	Score	Status	Language	Budget	Revenue
0	Fast X	2023	[Action, Crime, Thriller]	Dan Mazeau	2h 22m	72.0	Released	English	\$340,000,000.00	\$704,709,660.00
1	Trolls Band Together	2023	[Animation, Family, Music, Fantasy, Comedy]	Thomas Dam	1h 32m	72.0	Released	English	\$95,000,000.00	\$173,800,000.00
2	Robot Apocalypse	2021	[Science Fiction, Action]	Marcus Friedlander	1h 27m	21.0	Released	English	-	-
3	Five Nights at Freddy's	2023	[Horror, Mystery]	Emma Tammi	1h 50m	78.0	Released	English	\$20,000,000.00	\$286,700,000.00
4	Oppenheimer	2023	[Drama, History]	Christopher Nolan	3h 1m	81.0	Released	English	\$100,000,000.00	\$951,000,000.00
...
4995	Secret in Their Eyes	2015	[Thriller, Mystery, Drama, Crime]	Billy Ray	1h 51m	64.0	Released	English	\$19,500,000.00	\$34,854,990.00
4996	The Vatican Tapes	2015	[Thriller, Horror]	Mark Neveldine	1h 31m	53.0	Released	English	\$13,000,000.00	\$1,784,763.00
4997	Song to Song	2017	[Romance, Drama, Music]	Terrence Malick	2h 9m	55.0	Released	English	\$10,000,000.00	\$1,710,528.00
4998	Divine Intervention	2023	[Comedy]	Pedro Pablo Ibarra	1h 40m	82.0	Released	Spanish; Castilian	-	-
4999	Armour of God	1986	[Adventure, Action, Comedy]	Jackie Chan	1h 38m	70.0	Released	Cantonese	\$15,000,000.00	-

5000 rows × 10 columns

Tiền xử lý dữ liệu

3



LÀM SẠCH DỮ LIỆU

Data shape

```
n_rows=movie_df.shape[0]  
n_cols=movie_df.shape[1]  
n_rows, n_cols  
  
(5000, 10)
```

Duplicated rows

```
duplicated_rows = movie_df[movie_df.duplicated()]  
len(duplicated_rows)  
  
0
```



LÀM SẠCH DỮ LIỆU

Data type

```
movie_df.info()

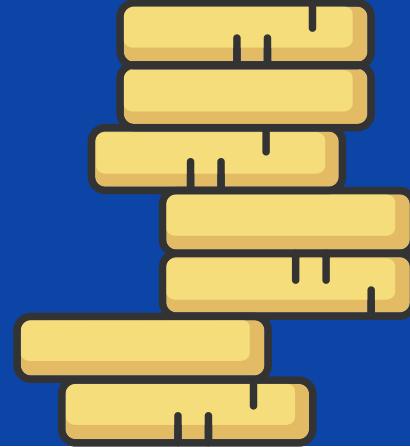
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Name         5000 non-null   object  
 1   Released year 5000 non-null  int64   
 2   Genre        5000 non-null   object  
 3   Director     5000 non-null   object  
 4   Runtime       5000 non-null   object  
 5   Score         5000 non-null   float64 
 6   Status        5000 non-null   object  
 7   Language      5000 non-null   object  
 8   Budget        5000 non-null   object  
 9   Revenue       5000 non-null   object  
dtypes: float64(1), int64(1), object(8)
memory usage: 390.8+ KB
```

Missing value

```
movie_df.replace(['[]', '-',' -'], np.nan, inplace=True)
missing_values = movie_df.isnull().sum()
# Calculate missing ratio for each column
missing_ratio = (missing_values / len(movie_df)) * 100
missing_ratio

Name           0.00
Released year  0.00
Genre          0.12
Director       0.10
Runtime        0.86
Score          0.00
Status          0.00
Language        0.00
Budget          34.50
Revenue         32.30
dtype: float64
```

KỸ THUẬT ĐẶC TRƯNG



- **Categorical features:** Vì một bộ phim có thể thuộc nhiều thể loại nên ta sẽ xử lý về định dạng phù hợp để thực hiện những công đoạn tiếp theo.
- **Numerical features:** Sử dụng thống kê mô tả để khảo sát các cột dữ liệu số.

Phân tích khám phá dữ liệu

4





Các công đoạn:

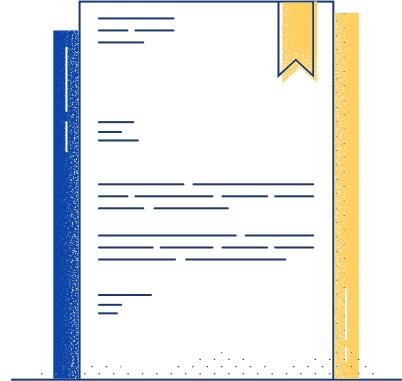
- **Purpose**
- **Analyzing**
- **Visualization**
- **Give observations**



CÂU HỎI 1

Lợi nhuận ròng và biên lợi nhuận ròng của ngành công nghiệp phim ảnh theo thời gian (1961-nay).

NOTE



Lợi nhuận ròng = Doanh thu - Kinh phí

Biên lợi nhuận ròng = Lợi nhuận ròng / Doanh thu

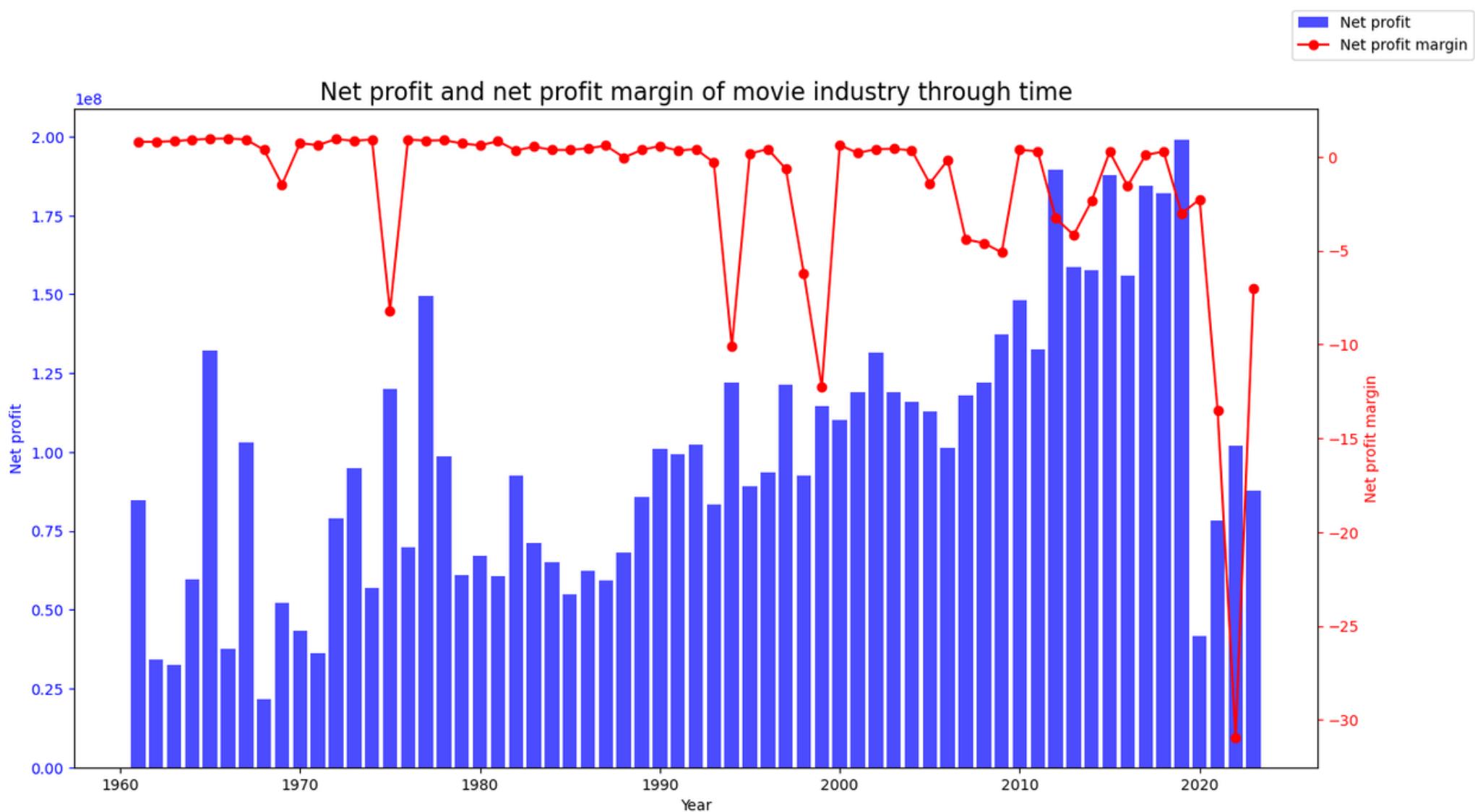
Để thuận tiện trong việc trực quan hóa dữ liệu,
ta chỉ chọn mốc thời gian từ năm 1961-nay.

Analyzing the question

```
cleaned_df = movie_df[['Released year', 'Budget ($)', 'Revenue ($)']].dropna()
cleaned_df['Net profit'] = cleaned_df['Revenue ($)'] - cleaned_df['Budget ($)']
cleaned_df['Net profit margin'] = (cleaned_df['Revenue ($)'] - cleaned_df['Budget ($)']) / cleaned_df['Revenue ($)']
cleaned_df = cleaned_df.groupby('Released year').mean()
cleaned_df = cleaned_df.iloc[25:]
cleaned_df
```

	Budget (\$)	Revenue (\$)	Net profit	Net profit margin
Released year				
1961	4.284250e+06	8.897500e+07	8.469075e+07	0.792492
1962	7.733333e+06	4.179871e+07	3.406538e+07	0.785605
1963	4.280000e+06	3.685647e+07	3.257647e+07	0.823427
1964	5.230000e+06	6.493319e+07	5.970319e+07	0.899623
1965	6.325000e+06	1.385671e+08	1.322421e+08	0.948515
...
2019	6.540769e+07	2.643783e+08	1.989706e+08	-3.027020
2020	5.444927e+07	9.609668e+07	4.164742e+07	-2.278529
2021	7.202993e+07	1.502569e+08	7.822694e+07	-13.504777
2022	5.873064e+07	1.607896e+08	1.020589e+08	-30.928192
2023	6.620874e+07	1.540363e+08	8.782757e+07	-6.993948

63 rows × 4 columns





CÂU HỎI 2

Xác định những đạo
diễn phim thành công
nhất mọi thời đại.

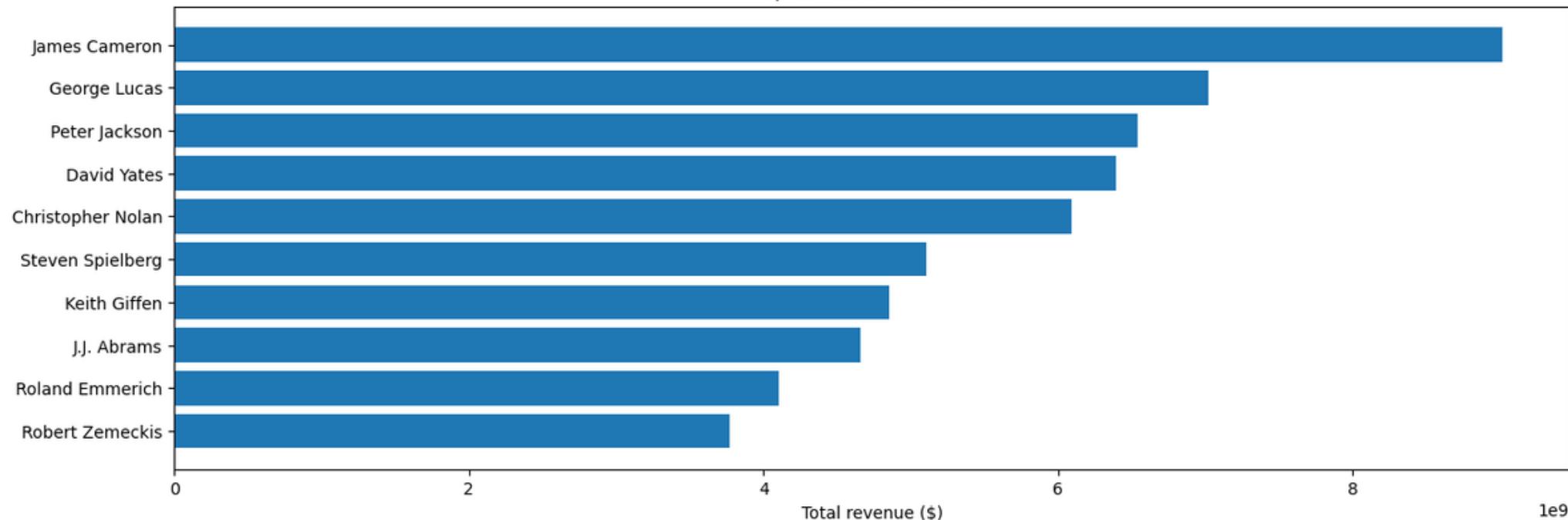
Analyzing the question

```
cleaned_df = movie_df.groupby('Director')['Revenue ($)'].sum().sort_values(ascending=False).head(10)[::-1]  
cleaned_df
```

```
Director  
Robert Zemeckis      3.772318e+09  
Roland Emmerich     4.106669e+09  
J.J. Abrams         4.653993e+09  
Keith Giffen        4.852415e+09  
Steven Spielberg    5.101908e+09  
Christopher Nolan   6.093476e+09  
David Yates         6.395881e+09  
Peter Jackson       6.537412e+09  
George Lucas        7.022914e+09  
James Cameron        9.019808e+09  
Name: Revenue ($), dtype: float64
```

Để đơn giản hóa, ta quy ước những đạo diễn thành công nhất là top 10 những đạo diễn có tổng doanh thu thu được từ phim cao nhất.

Top directors of all time

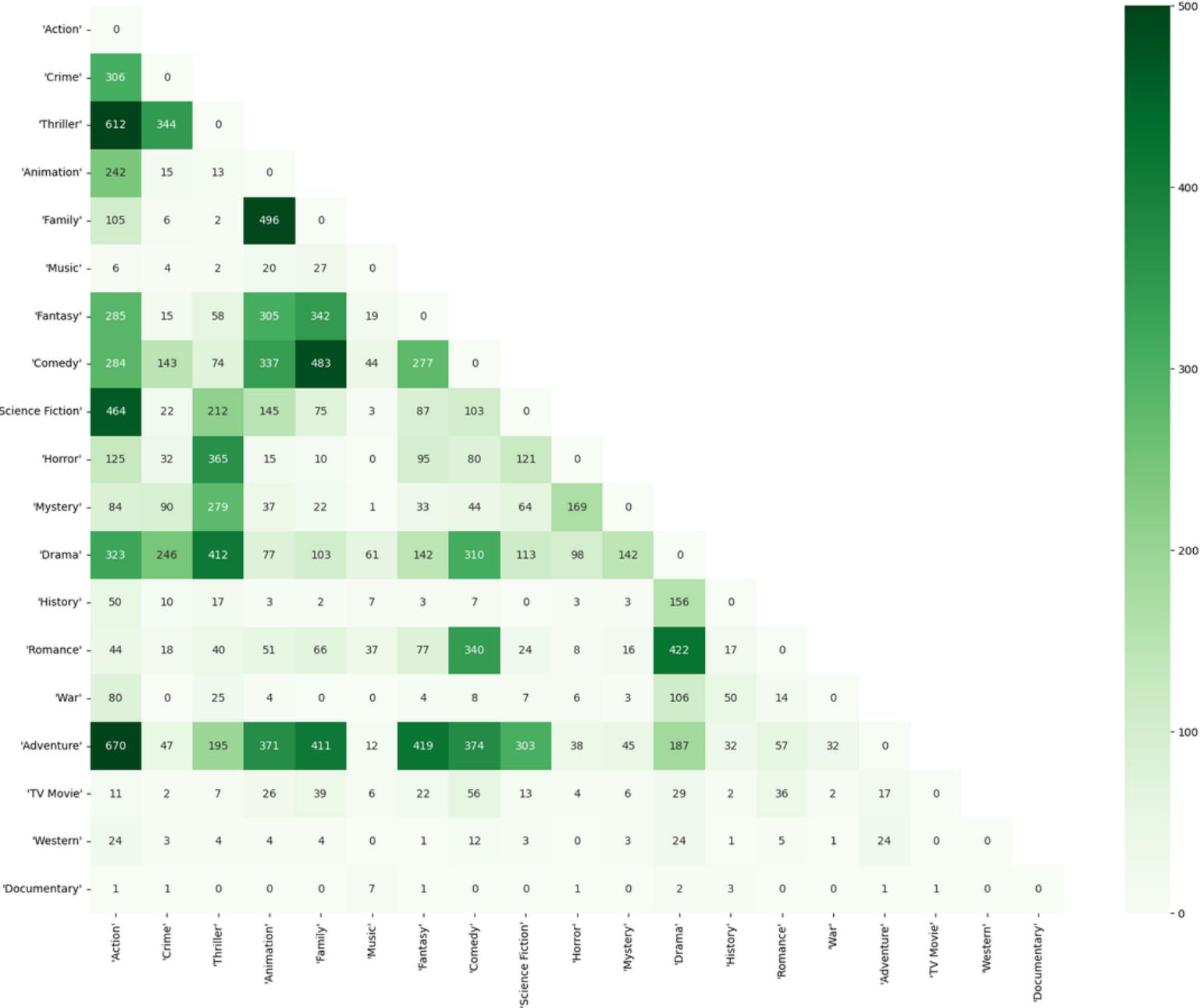


CÂU HỎI 3

Những thể loại phim
nào có thể dễ dàng kết
hợp được với nhau?

Analyzing the question

	'Action'	'Crime'	'Thriller'	'Animation'	'Family'	'Music'	'Fantasy'	'Comedy'	'Science Fiction'	'Horror'	'Mystery'	'Drama'
'Action'	0	0	0	0	0	0	0	0	0	0	0	0
'Crime'	306	0	0	0	0	0	0	0	0	0	0	0
'Thriller'	612	344	0	0	0	0	0	0	0	0	0	0
'Animation'	242	15	13	0	0	0	0	0	0	0	0	0
'Family'	105	6	2	496	0	0	0	0	0	0	0	0
'Music'	6	4	2	20	27	0	0	0	0	0	0	0
'Fantasy'	285	15	58	305	342	19	0	0	0	0	0	0
'Comedy'	284	143	74	337	483	44	277	0	0	0	0	0
'Science Fiction'	464	22	212	145	75	3	87	103	0	0	0	0
'Horror'	125	32	365	15	10	0	95	80	121	0	0	0
'Mystery'	84	90	279	37	22	1	33	44	64	169	0	0
'Drama'	323	246	412	77	103	61	142	310	113	98	142	0
'History'	50	10	17	3	2	7	3	7	0	3	3	150
'Romance'	44	18	40	51	66	37	77	340	24	8	16	42
'War'	90	9	25	4	9	0	4	9	7	6	3	19





CÂU HỎI 4

Liệu một bộ phim có điểm số cao thì xu hướng doanh thu cũng sẽ cao hay không?

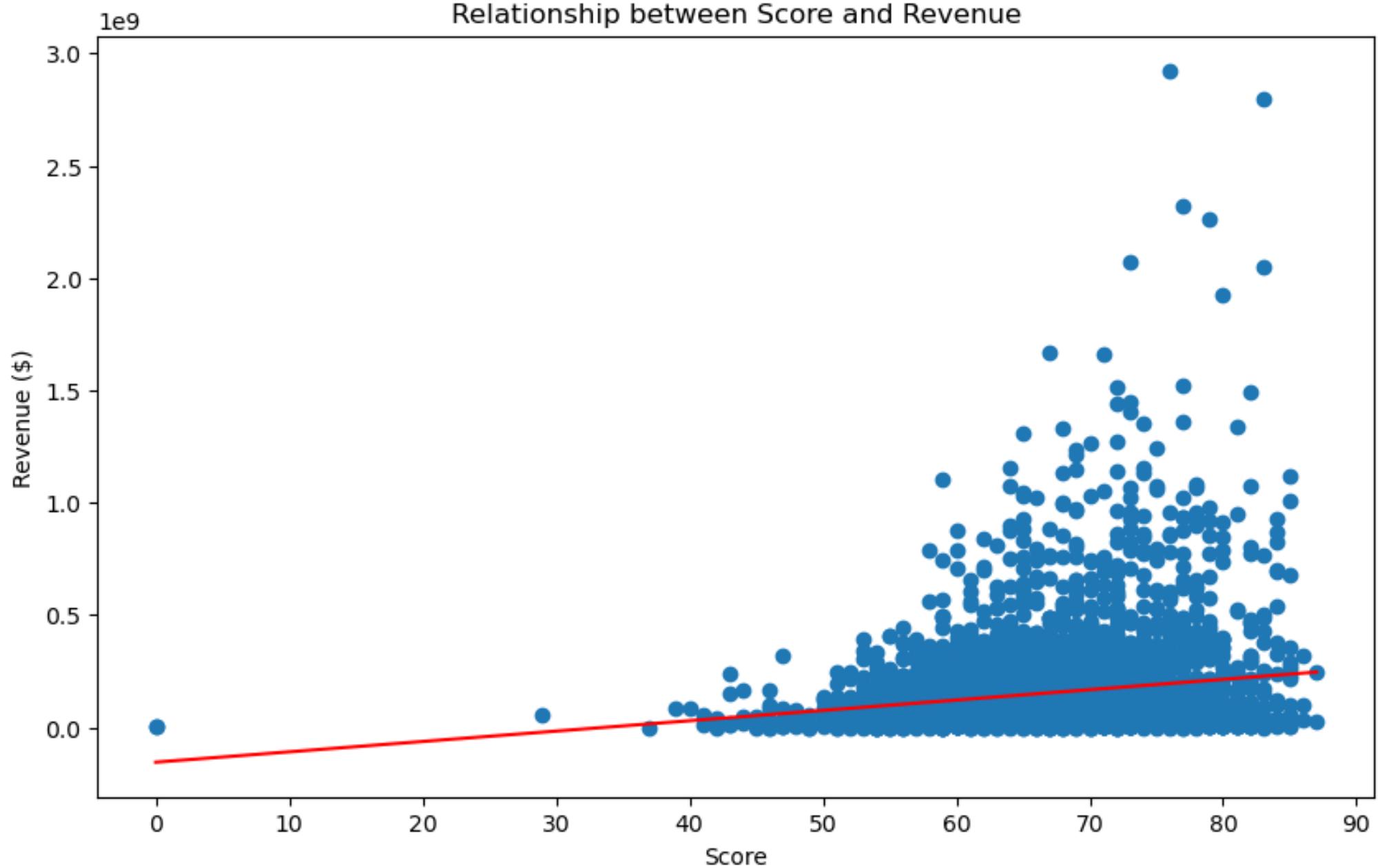
Analyzing the question

```
#Filter out columns 'Name', 'Score' and 'Revenue ($)'
df = movie_df[['Name', 'Score', 'Revenue ($)']]
df = df[df['Revenue ($)'].notna()]
df
```

	Name	Score	Revenue (\$)
0	Fast X	72.0	704709660.0
1	Trolls Band Together	72.0	173800000.0
3	Five Nights at Freddy's	78.0	286700000.0
4	Oppenheimer	81.0	951000000.0
7	Freelance	65.0	8000000.0
...
4992	Tucker: The Man and His Dream	67.0	19652638.0
4994	Who Am I	76.0	7700259.0
4995	Secret in Their Eyes	64.0	34854990.0
4996	The Vatican Tapes	53.0	1784763.0
4997	Song to Song	55.0	1710528.0

3385 rows × 3 columns

Relationship between Score and Revenue

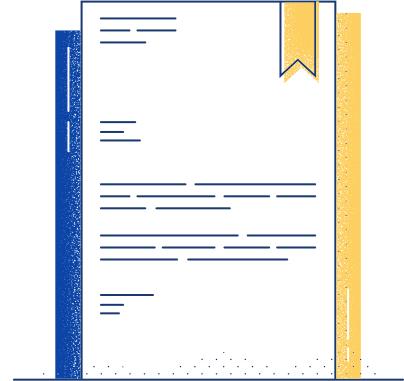




CÂU HỎI 5

Xác định thời
lượng lý tưởng cho
một bộ phim.

NOTE



Ta chỉ khảo sát thời lượng của những bộ phim đáng xem nhất, được quy ước như sau:

- Doanh thu của bộ phim đó lớn hơn tứ phân vị thứ 3 (quantile 75) của mẫu số liệu tương ứng.
- Điểm số của bộ phim đó lớn hơn tứ phân vị thứ 3 (quantile 75) của mẫu số liệu tương ứng.

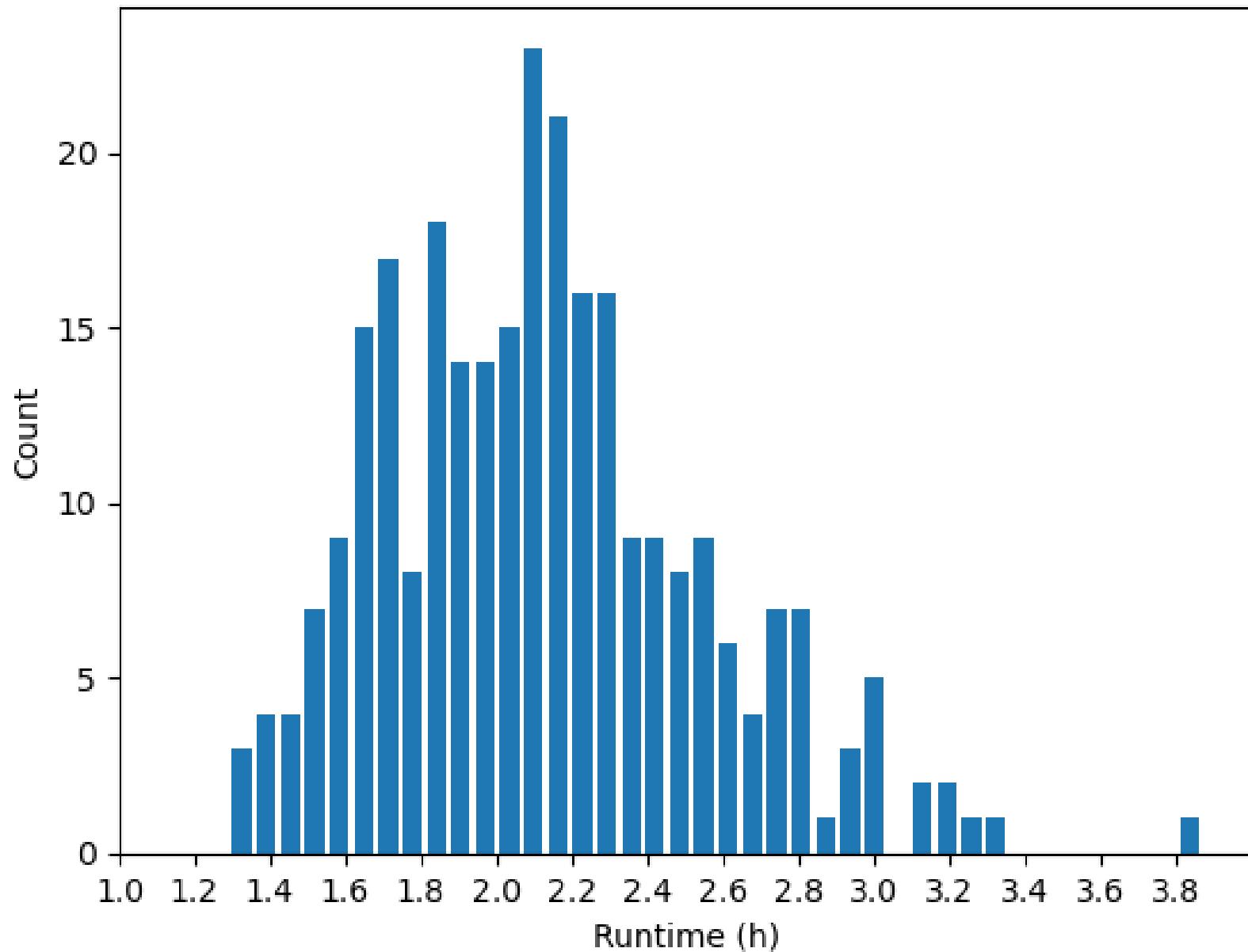
Analyzing the question

```
#Filter out columns 'Name', 'Runtime (h)', 'Score' and 'Revenue ($)'  
df = movie_df[['Name', 'Runtime (h)', 'Score', 'Revenue ($)']]  
df = df[(df['Revenue ($)'].notna()) & (df['Runtime (h)').notna())]  
  
#Calculate the 75th percentile of score and revenue  
score_75 = df['Score'].quantile(0.75)  
revenue_75 = df['Revenue ($)'].quantile(0.75)  
  
#Filter out good quality movies with their scores and revenues above the 75th percentile  
df = df[(df['Score'] >= score_75) & (df['Revenue ($)'] >= revenue_75)]  
df
```

	Name	Runtime (h)	Score	Revenue (\$)
3	Five Nights at Freddy's	1.83	78.0	2.867000e+08
4	Oppenheimer	3.02	81.0	9.510000e+08
8	The Hunger Games: The Ballad of Songbirds & Sn...	2.62	73.0	2.584000e+08
9	Mission: Impossible - Dead Reckoning Part One	2.73	76.0	5.671490e+08
20	Avatar: The Way of Water	3.20	77.0	2.320250e+09
...
3452	Minority Report	2.42	73.0	3.583729e+08
3488	The Stranger	1.58	73.0	3.220000e+08
3522	Paddington 2	1.73	75.0	2.273000e+08
4305	As Good as It Gets	2.32	74.0	3.141780e+08
4494	JFK	3.15	76.0	2.054055e+08

279 rows × 4 columns

Runtime of good quality movies





CÂU HỎI 6

Thể loại phim phổ biến theo thời gian.

Analyzing the question

	'Action'	'Adventure'	'Animation'	'Comedy'	'Crime'	'Documentary'	'Drama'	'Family'	'Fantasy'	'History'
0	1	0	0	0	1	0	0	0	0	0
1	0	0	1	1	0	0	0	1	1	0
2	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	0	1
...
4995	0	0	0	0	1	0	1	0	0	0
4996	0	0	0	0	0	0	0	0	0	0
4997	0	0	0	0	0	0	1	0	0	0
4998	0	0	0	1	0	0	0	0	0	0
4999	1	1	0	1	0	0	0	0	0	0

4982 rows × 19 columns

Analyzing the question

```
scent_df = one_hot_df.join(released_movies['Year bin'])

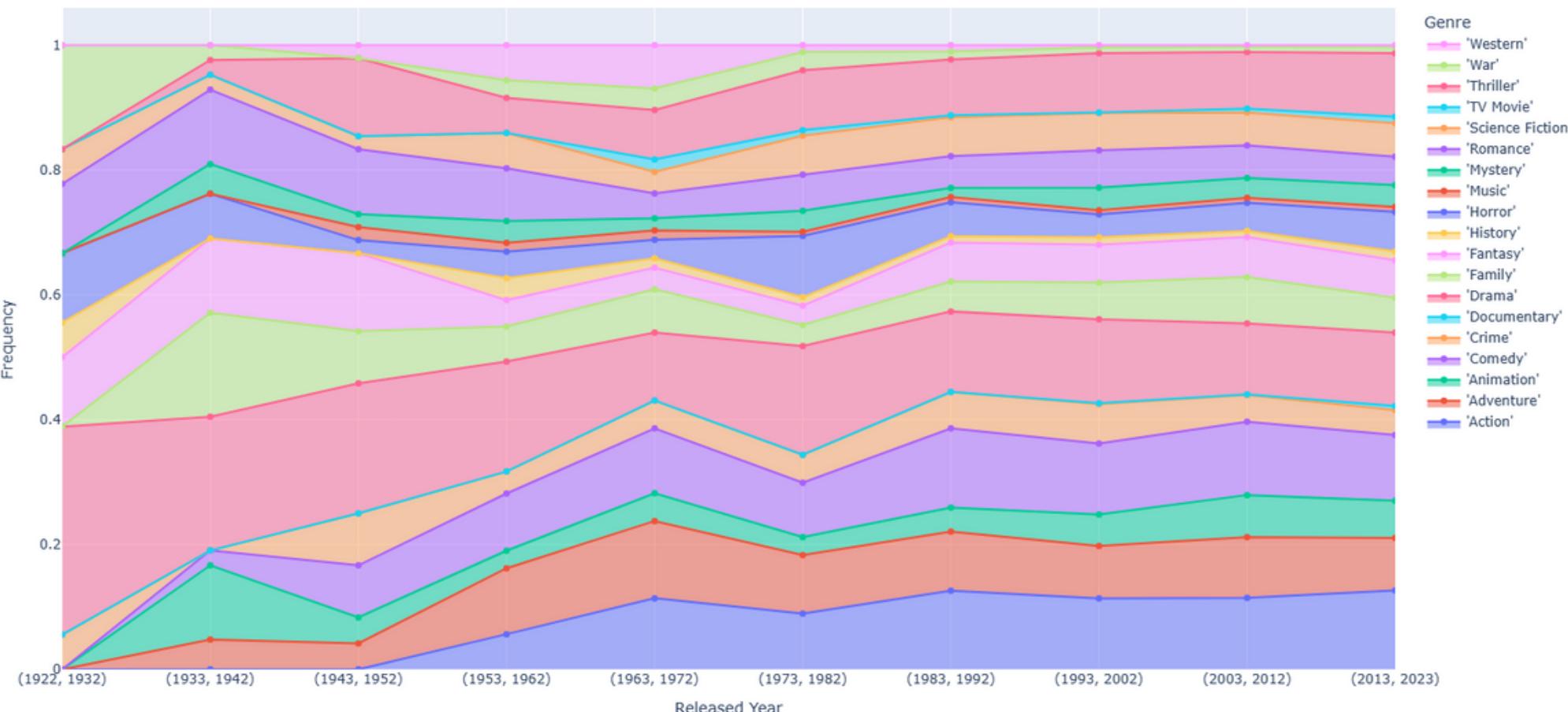
distribution = pd.DataFrame()
mask = scent_df.drop(columns = ['Year bin']).eq(1)
result = mask.groupby(scent_df['Year bin']).sum()
distribution = pd.concat([distribution, result], axis = 1)

# convert to frequency
for i, row in distribution.iterrows():
    sum_row = row.sum()
    for col in distribution.columns:
        distribution.at[i, col] = row[col] / sum_row
distribution
```

Year bin	Action	Adventure	Animation	Comedy	Crime	Documentary	Drama	Family	Fantasy	History	Horror	Music	Mystery	Romance	Science Fiction
0	0.000000	0.000000	0.000000	0.000000	0.055556	0.000000	0.333333	0.000000	0.111111	0.055556	0.111111	0.000000	0.000000	0.111111	0.055556
1	0.000000	0.047619	0.119048	0.023810	0.000000	0.000000	0.214286	0.166667	0.119048	0.000000	0.071429	0.000000	0.047619	0.119048	0.023810
2	0.000000	0.041667	0.041667	0.083333	0.083333	0.000000	0.208333	0.083333	0.125000	0.000000	0.020833	0.020833	0.020833	0.104167	0.020833
3	0.056338	0.105634	0.028169	0.091549	0.035211	0.000000	0.176056	0.056338	0.042254	0.035211	0.042254	0.014085	0.035211	0.084507	0.056338
4	0.113861	0.123762	0.044554	0.103960	0.044554	0.000000	0.108911	0.069307	0.034653	0.014851	0.029703	0.014851	0.019802	0.039604	0.034654
5	0.089286	0.093750	0.029018	0.087054	0.044643	0.000000	0.174107	0.033482	0.031250	0.013393	0.098214	0.006696	0.033482	0.058036	0.062500
6	0.125911	0.094693	0.038502	0.126951	0.058273	0.000000	0.129032	0.047867	0.062435	0.010406	0.054110	0.008325	0.014568	0.050989	0.062435
7	0.113530	0.083981	0.050804	0.113530	0.063245	0.001037	0.134785	0.058580	0.060653	0.011923	0.036807	0.006739	0.036288	0.059616	0.059611
8	0.114715	0.097022	0.067532	0.117370	0.042760	0.001180	0.113536	0.074314	0.064288	0.009437	0.045119	0.007962	0.032144	0.052197	0.052495
9	0.126390	0.084057	0.059845	0.105375	0.039744	0.006700	0.117405	0.055885	0.059997	0.014009	0.063195	0.008375	0.034567	0.045683	0.053600



Distribution of Movie Genres over the Years

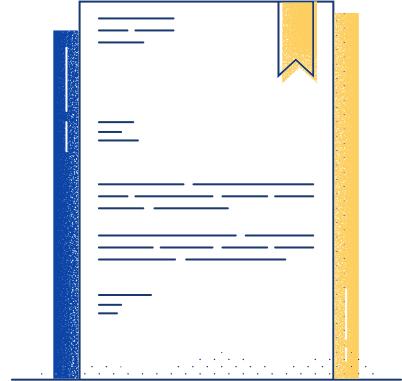




CÂU HỎI 7

Xác định những bộ
phim đáng xem nhất
theo từng thể loại.

NOTE



Quy ước cách tìm những bộ phim đáng xem nhất của từng thể loại:

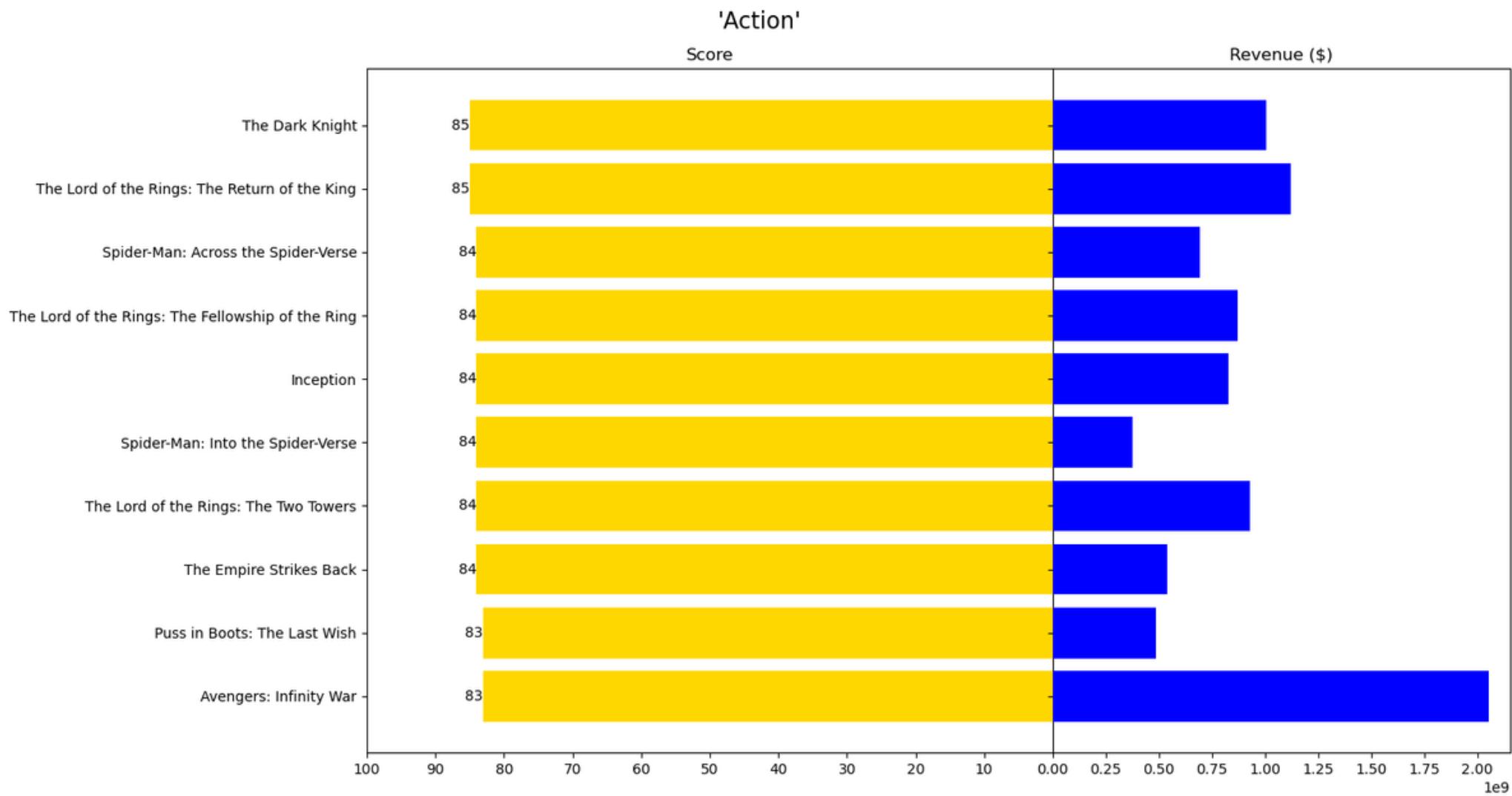
- Chỉ chọn những bộ phim có doanh thu lớn hơn tứ phân vị thứ 3 (quantile 75) của mẫu số liệu tương ứng.
- Gom nhóm bộ dữ liệu theo thể loại, chọn ra những bộ phim được đánh giá với điểm số cao nhất chính là những bộ phim đáng xem cần tìm.

Analyzing the question

```
cleaned_df = movie_df[movie_df['Revenue ($)'] > movie_df['Revenue ($)'].quantile(0.75)][['Name', 'Genre',  
                                         'Score', 'Revenue ($)']].reset_index(drop=True)  
cleaned_df['Genre'] = cleaned_df['Genre'].str.split(',')  
cleaned_df = cleaned_df.explode('Genre').sort_values(['Genre', 'Score'], ascending=[True, False]).groupby('Genre').head(10)  
cleaned_df
```

		Name	Genre	Score	Revenue (\$)
88		The Dark Knight	'Action'	85.0	1.004558e+09
102		The Lord of the Rings: The Return of the King	'Action'	85.0	1.118889e+09
8		Spider-Man: Across the Spider-Verse	'Action'	84.0	6.905000e+08
71		The Lord of the Rings: The Fellowship of the Ring	'Action'	84.0	8.713684e+08
73		Inception	'Action'	84.0	8.255328e+08
...	
342		The Revenant	'Western'	75.0	5.329505e+08
721		True Grit	'Western'	73.0	2.523000e+08
150		Rango	'Western'	68.0	2.457246e+08
675		The Lone Ranger	'Western'	61.0	2.605021e+08
663		Wild Wild West	'Western'	53.0	2.221047e+08

167 rows × 4 columns



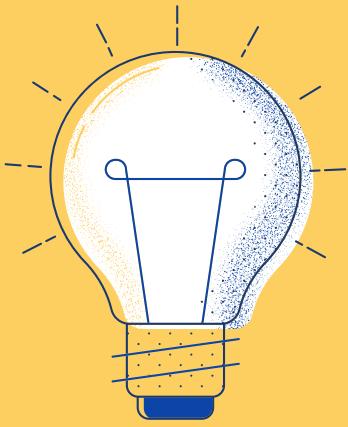
Mô hình hóa dữ liệu

5



Dự đoán điểm số của bộ phim





MÔ HÌNH SỬ DỤNG

- 1 LASSO
- 2 SVM
- 3 Random Forest Regressor
- 4 Decision Tree Regressor

Preprocessing

Genre_Adventure	Genre_Animation	Genre_Comedy	Genre_Crime	Genre_Drama	Genre_Family	Genre_Fantasy	Genre_History	Genre_Mystery	Genre_Romantic	Genre_SciFi	Genre_Suspense	Genre_Thriller	Genre_War	Genre_Western
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	1	1	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	1	0	0	0	1	0	0	0	0	0	0
0	0	1	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	0	1	0	0	0	0	0	0	0	0	0	0

Mã hóa biến phân loại **Genre**
thành dạng số bằng
One-hot Encoding

Language_encoded
5
5
5
5
5
5
5
5
5
5

Mã hóa biến phân loại
Language thành dạng
số bằng Label Encoder

Training model & Prediction

1.2 Data Preparation

```
features = df.drop('Score', axis = 1).columns.values.tolist()
target = 'Score'

data = df[features + [target]]  
  
X_train, X_test, y_train, y_test = train_test_split(data[features], data[target], test_size = 0.2, random_state = 0)
```

1.3 Training model & Prediction

```
models = [Lasso(), SVR(), RandomForestRegressor(), DecisionTreeRegressor()]
model_names = ['Lasso', 'SVM', 'Random Forest Regressor', 'Decision Tree Regressor']
y_preds = {}

for model, name in zip(models, model_names):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_preds[name] = y_pred
```

Evaluation

Mean Squared Error (MSE): ước tính giá trị trung bình của bình phương các sai số.

R-squared: tỷ lệ thay đổi của biến phụ thuộc có thể dự đoán được từ biến độc lập.

	Mean Squared Error	R-squared
Lasso	53.9839	0.0841
SVM	55.1330	0.0646
Random Forest Regressor	40.3382	0.3156
Decision Tree Regressor	87.6801	-0.4876

Fine-tuning

Dùng **Grid Search Cross-Validation** để tìm ra tổ hợp siêu tham số tốt nhất với từng mô hình.

Tập các tổ hợp
siêu tham số của
từng mô hình

```
# for lasso
param_grid_lasso = {'alpha': [.01, .1, 1, 10, 100]}
# for support vector machine
param_grid_svm = {'C': [.01, .1, 1, 10, 100],
                  'gamma': ['scale', 'auto']}
# for random forest regressor
param_grid_rfr = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
# decision tree regressor
param_grid_dtr = {
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

Fine-tuning

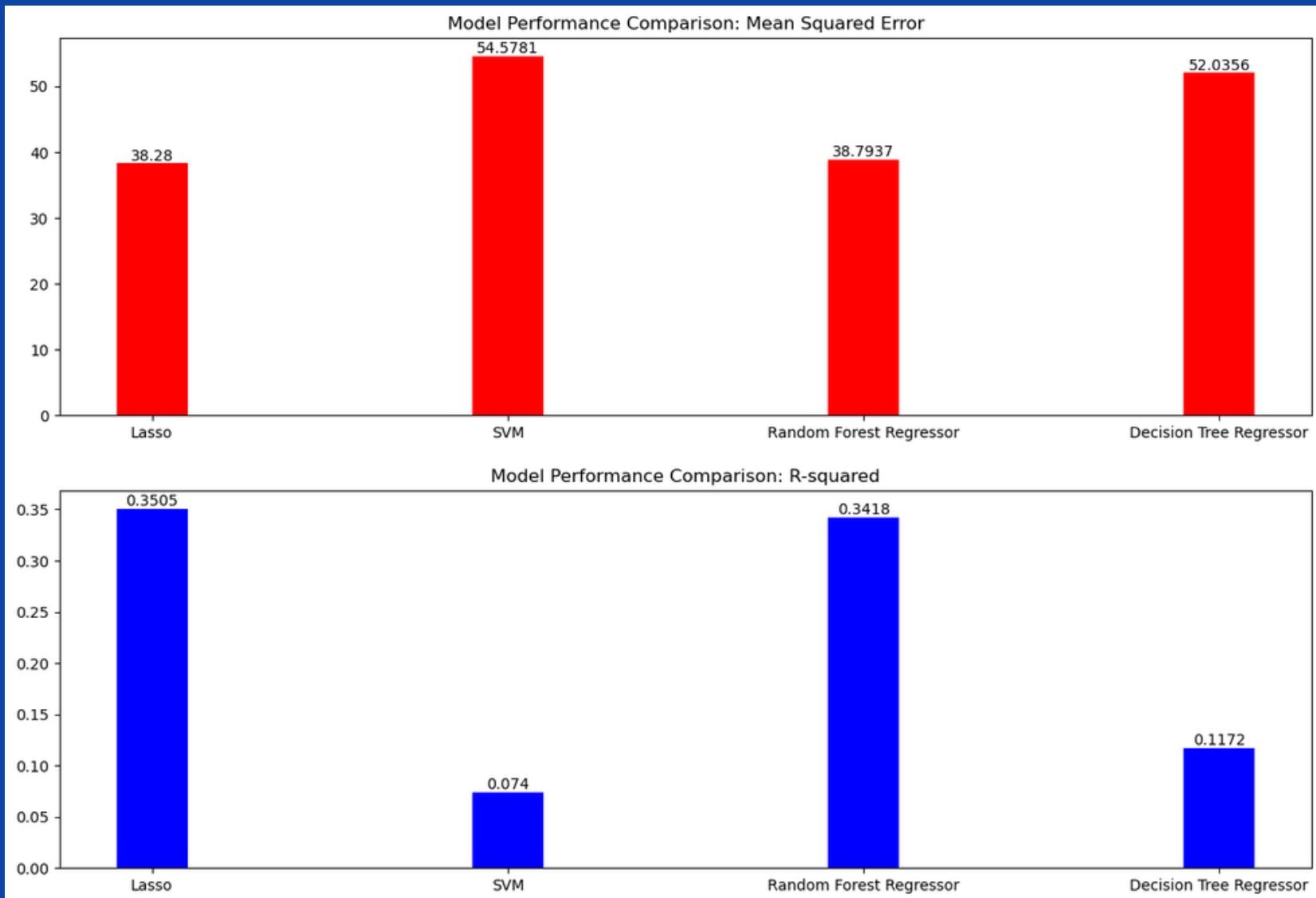
Các tổ hợp siêu tham số tốt nhất với từng mô hình:

```
Best parameters for Lasso: {'alpha': 0.01}
Best parameters for SVM: {'C': 10, 'gamma': 'scale'}
Best parameters for Random Forest Regressor: {'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 200}
Best parameters for Decision Tree Regressor: {'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 10}
```

Kết quả với
các siêu tham
số tốt nhất

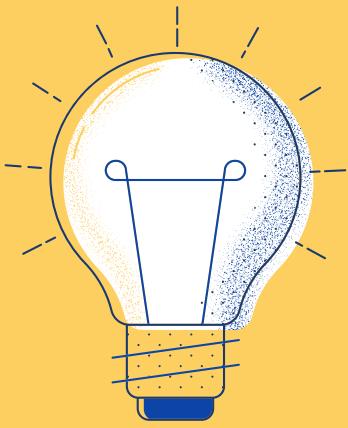
	Mean Squared Error	R-squared
Lasso	38.2800	0.3505
SVM	54.5781	0.0740
Random Forest Regressor	38.5388	0.3461
Decision Tree Regressor	51.9597	0.1184

Performance Comparison



Phân loại một bộ phim thành công hay không





MÔ HÌNH SỬ DỤNG

- 1 Random Forest
- 2 SVM
- 3 KNN
- 4 Multinomial Naive Bayes

Preprocessing

Quá trình Preprocessing tương tự như phần dự đoán điểm số nhưng có tạo thêm 1 cột Success để phân loại phim thành công với 2 yếu tố:

- Lợi nhuận lớn hơn 150% so với ngân sách.
- Score lớn hơn trung vị của cột điểm số trong bộ dữ liệu.

Success
Successful
Successful
Successful
Successful
Unsuccessful
...
Unsuccessful

Training model & Prediction

2.2 Data preparation

```
X = df.drop('Success', axis=1) # Features
y = df['Success'] # Target

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

2.3 Training model & Prediction

```
models = [RandomForestClassifier(), SVC(), KNeighborsClassifier(), MultinomialNB()]
model_names = ['Random Forest', 'SVM', 'KNN', 'Multinomial Naive Bayes']
y_preds = {}

for model, name in zip(models, model_names):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_preds[name] = y_pred
```

Evaluation

Accuracy: tỷ lệ phần trăm các mẫu được dự đoán chính xác trên tổng số mẫu.

Precision: tỷ lệ dự đoán dương thực sự trên tổng số dự đoán dương.

Recall: tỷ lệ dự đoán dương thực sự trên tổng số mẫu dương thực tế.

F1 Score: giá trị trung bình hài hòa của Precision và Recall.

	Random Forest	SVM	KNN	Multinomial Naive Bayes
Accuracy	0.9697	0.6751	0.6734	0.7121
Precision	0.9416	0.7055	0.6447	0.6679
Recall	0.9923	0.4423	0.5654	0.6808
F1 score	0.9663	0.5437	0.6025	0.6743

*Chọn Successful là nhãn “dương”

Fine-tuning

Đánh giá sự mất cân
bằng của dữ liệu tại
cột phân loại

```
Unsuccessful      1760
Successful        1209
Name: Success, dtype: int64
Imbalance ratio: 0.6869318181818181
```



Dùng **Synthetic Minority Over-sampling
Technique (SMOTE)** để cân bằng.

Fine-tuning

Sau khi xử lý Imbalance Data

	Random Forest	SVM	KNN	Multinomial Naive Bayes
Accuracy	0.9659	0.6818	0.7131	0.7031
Precision	0.9344	0.6569	0.6823	0.6997
Recall	1.0000	0.7222	0.7661	0.6813
F1 score	0.9661	0.6880	0.7218	0.6904

Fine-tuning

Dùng **Grid Search Cross-Validation** để tìm ra tổ hợp siêu tham số tốt nhất với từng mô hình.

Tập các tổ hợp siêu tham số của từng mô hình

```
#Create a param grid from each model's hyperparameter set
param_grid_svm = {'C': [0.01, 0.1, 1, 10, 100],
                  'gamma': ['scale', 'auto']}
param_grid_knn = {'n_neighbors': [1, 3, 5, 7, 9, 11, 13, 15],
                  'weights': ['uniform', 'distance'],
                  'metric': ['euclidean', 'manhattan']}
param_grid_mnb = {'alpha': [0.1, 0.5, 1, 2],
                  'fit_prior': [True, False],
                  'class_prior': [None, [0.3, 0.7], [0.5, 0.5]]},}
```

Fine-tuning

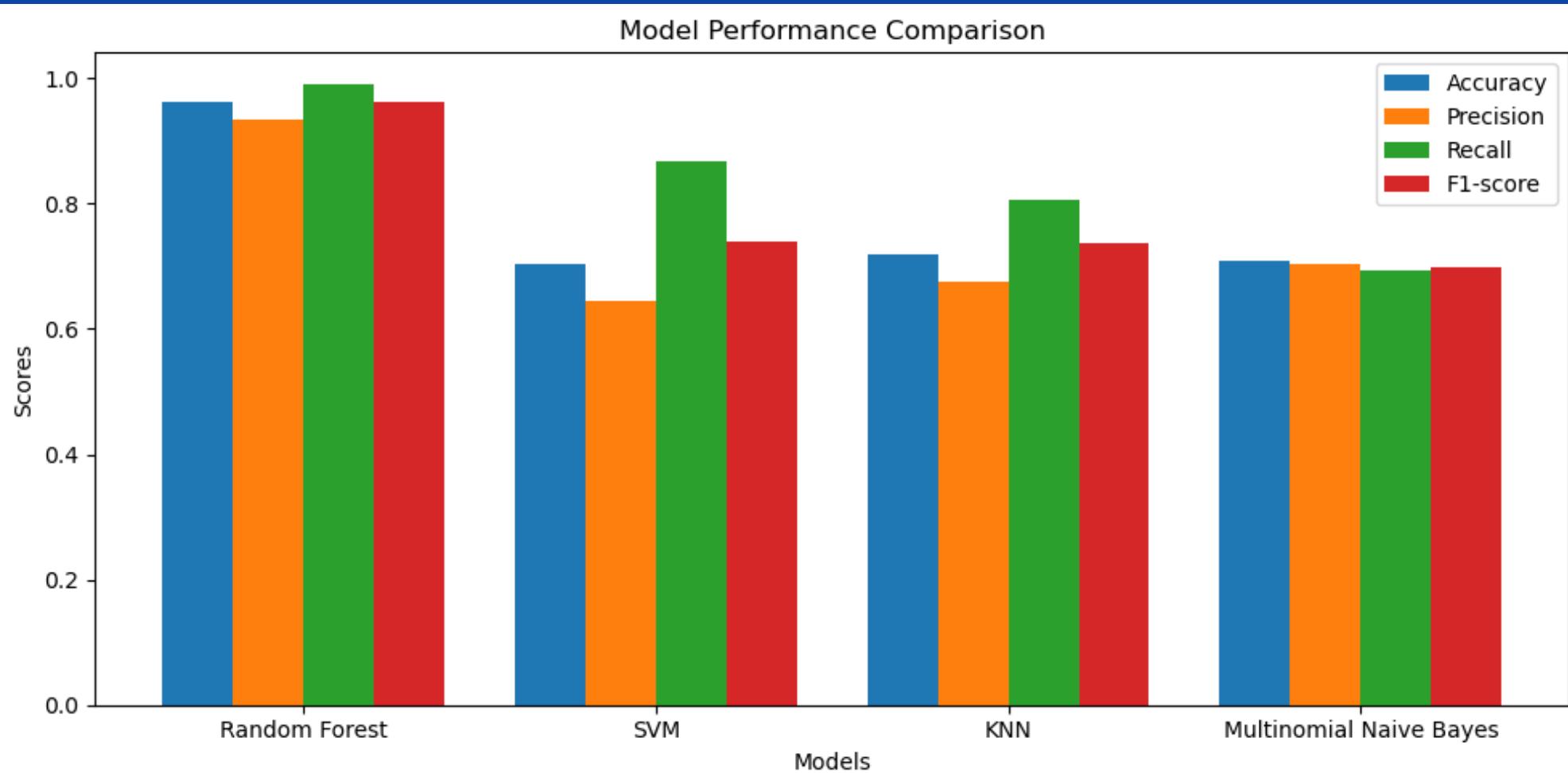
Các tổ hợp siêu tham số tốt nhất với từng mô hình:

```
Best parameters for SVM: {'C': 100, 'gamma': 'scale'}  
Best parameters for KNN: {'metric': 'euclidean', 'n_neighbors': 11, 'weights': 'distance'}  
Best parameters for Multinomial Naive Bayes: {'alpha': 0.1, 'class_prior': None, 'fit_prior': True}
```

Kết quả với
các siêu tham
số tốt nhất

	Random Forest	SVM	KNN	Multinomial Naive Bayes
Accuracy	0.9659	0.6989	0.7173	0.7031
Precision	0.9344	0.6413	0.6757	0.6997
Recall	1.0000	0.8626	0.8041	0.6813
F1 score	0.9661	0.7357	0.7343	0.6904

Performance Comparison



Kết luận và tự đánh giá

6



REFLECTION

Kiến thức học được:

- Quy trình chung của một dự án Khoa học Dữ liệu.
- Kỹ năng làm việc nhóm.
- Cách sử dụng các công cụ hỗ trợ làm việc hiệu quả hơn như GitHub, Trello.
- Kiến thức chuyên môn về mô hình học máy.
- Biết thêm một mảng kiến thức domain.
- Học hỏi lẫn nhau giữa các thành viên trong nhóm.



REFLECTION

Khó khăn gặp phải:

- Bộ dữ liệu chưa mang nhiều ý nghĩa, không có quá nhiều đặc trưng.
- Thiếu domain knowledge (kiến thức về kinh tế, điện ảnh).
- Chưa thể nghĩ ra được câu hỏi mang tính chuyên môn cao để phân tích (eda).
- Chưa hiểu rõ bản chất tường tận của thuật toán trong mô hình học máy, chỉ import module và sử dụng hàm có sẵn để thực hiện.
- Thiếu kinh nghiệm xử lý dữ liệu trong thực tế.



REFLECTION

Nếu còn thời gian:

- Nghiên cứu thêm về domain knowledge.
- Áp dụng sửa đổi theo data pipeline trong việc mô hình hóa dữ liệu.
- Nghiên cứu sâu hơn về thuật toán trong mô hình học máy để tinh chỉnh phù hợp hơn (fine-tuning).
- Làm thêm phase 3, triển khai mô hình dữ liệu trong thực tế.



THANK YOU!

Nhóm 5
Lớp 21_21
NMKHDL
Final Project



