Movies and Ratings re(IMDB)

Introduction

Movies have been a part of our culture and entertainment for over a century. Understanding the trends and patterns in movie production and ratings can provide insights into the evolution of the film industry. This data analysis aims to explore the types of movies produced over time, their genres, and how they have been rated by audiences. By leveraging datasets from IMDB, we can uncover trends in movie genres, production rates, and ratings.

Data

The datasets include movies.csv and ratings.csv, which are part of the IMDB datasets found online. These datasets provide a comprehensive understanding of the types of movies produced over time and their ratings by different users. The analysis was conducted using Jupyter Notebook, with pandas for data organization and Matplotlib and Seaborn for graphing.

The movies.vsc file contains information about 10,329 movies from 1902 to 2015. It includes each movie's name, release year, IMDB movie ID, and genres.

```
[202]: movie_data
```

[202]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
10324	146684	Cosmic Scrat-tastrophe (2015)	Animation|Children|Comedy
10325	146878	Le Grand Restaurant (1966)	Comedy
10326	148238	A Very Murray Christmas (2015)	Comedy
10327	148626	The Big Short (2015)	Drama
10328	149532	Marco Polo: One Hundred Eyes (2015)	(no genres listed)

10329 rows × 3 columns

2.1 Sample of movie_data

The data type of movieID is the integer, the data type of title is the object, and the data type of genres also is the object. There are no missing data.

```
[5]: movie_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10329 entries, 0 to 10328
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   movieId  10329 non-null  int64 
 1   title    10329 non-null  object
 2   genres   10329 non-null  object
dtypes: int64(1), object(2)
memory usage: 242.2+ KB
```

2.2 info of movie_data

The ‘ratings_data’ contains the rating for these movies from different users, the record from the timestamp 828564954 (Wednesday, April 3, 1996 8:55:54 PM(GMT)) to 1452404919 (Sunday, January 10, 2016 5:48:39 AM (GMT)), total 105,339 comments.

```
[45]: rating_data
```

[45]:

	userId	movieId	rating	timestamp
0	1	16	4.0	1217897793
1	1	24	1.5	1217895807
2	1	32	4.0	1217896246
3	1	47	4.0	1217896556
4	1	50	4.0	1217896523
...	...	...	...	...
105334	668	142488	4.0	1451535844
105335	668	142507	3.5	1451535889
105336	668	143385	4.0	1446388585
105337	668	144976	2.5	1448656898
105338	668	148626	4.5	1451148148

105339 rows × 4 columns

2.3 Sample of ratings_data

The data types of userID, movieID, and timestamp are integer, and rating is the float, no missing data in this dataset.

```
[37]: rating_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 105339 entries, 0 to 105338
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype  
---  ------     --------------   -----  
 0   userId     105339 non-null  int64  
 1   movieId    105339 non-null  int64  
 2   rating     105339 non-null  float64
 3   timestamp  105339 non-null  int64  
dtypes: float64(1), int64(3)
memory usage: 3.2 MB
```

2.4 info of ratings_data

Since the title in movie_data includes the movie's name and release year, these were separated into two columns, name and year.

```
[7]: #separate the name and year
def separate_year(title):
    return title[-5:-1] if title.endswith(')') else None

def separate_name(title):
    return title[:-7] if title.endswith(')') else title

movie_data['year'] = movie_data['title'].apply(separate_year)
movie_data['name'] = movie_data['title'].apply(separate_name)
```

```
[8]: #drop the original title
movie_data.drop(columns=['title'], inplace=True)

movie_data= movie_data[['movieId','name','year','genres']]
movie_data
```

[8]:

	movieId	name	year	genres
0	1	Toy Story	1995	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji	1995	Adventure|Children|Fantasy
2	3	Grumpier Old Men	1995	Comedy|Romance
3	4	Waiting to Exhale	1995	Comedy|Drama|Romance
4	5	Father of the Bride Part II	1995	Comedy
...	...	...	...	...
10324	146684	Cosmic Scrat-tastrophe	2015	Animation|Children|Comedy
10325	146878	Le Grand Restaurant	1966	Comedy
10326	148238	A Very Murray Christmas	2015	Comedy
10327	148626	The Big Short	2015	Drama
10328	149532	Marco Polo: One Hundred Eyes	2015	(no genres listed)

10329 rows × 4 columns

2.5 Separate the data from the title

Methods

Count Movies by Genre: Separate the genres by '|', and count the number of movies for each genre. Since a movie can belong to multiple genres, this involves using the DataFrame explode method to handle multiple entries.

Movies Produced Each Year: Sort the movies by 'year' and count the number of movies produced each year. This provides insights into the trends in movie production over time. The results are visualized using a line chart, with different colors representing different genres.

Genre Combinations: Analyze how movie genres are combined using a heatmap. This reveals popular genre combinations and helps identify market trends.

Average Movie Ratings: Combine the ratings data to find the average rating for each movie. Identify the most popular movies by sorting them according to the number of ratings they received. This typically indicates a movie's popularity.

Popular Movie Genres: Determine the genres of the top 100 most popular movies and analyze their rating distributions using visualizations like heatmaps and scatter plots.

Analysis

Genres Count

To find out how many movies belong to each genre, split the genres by '|', use the DataFrame 'explode' method to convert each genre into a row, and then use 'value_counts' to count the data.

```
[9]: movie_data['genres'] = movie_data['genres'].str.split('|')
```

```
[10]: exploded_movie_data = movie_data.explode('genres')
```

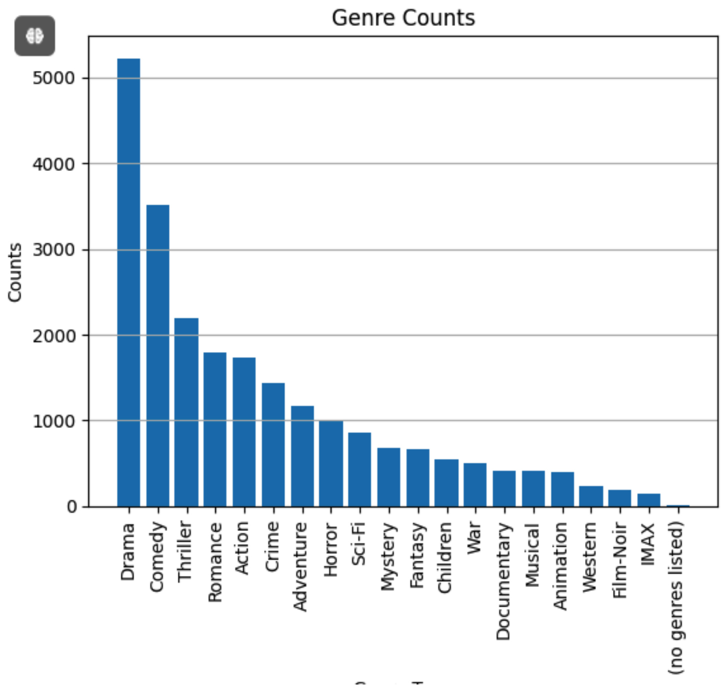
```
[37]: movie_genres_count = exploded_movie_data['genres'].value_counts()
      movie_genres_count
```

```
[37]: genres
      Drama                 5220
      Comedy                3515
      Thriller              2187
      Romance               1788
      Action                1737
      Crime                 1440
      Adventure             1164
      Horror                1001
      Sci-Fi                 860
      Mystery                675
      Fantasy                670
      Children               540
      War                    503
      Documentary            415
      Musical                409
      Animation              401
      Western                235
      Film-Noir              195
      IMAX                   152
      (no genres listed)       7
      Name: count, dtype: int64
```

4.1 table of genres count

Converting the count data into a graph reveals that drama and comedy are the most popular genres, indicating that many directors prefer to make drama or comedy movies.

```
#movie_genres_count.plot(kind = 'bar')
#make the movie type to graph
#movie_genres_count.plot(kind='bar')
plt.bar(movie_genres_count.index,movie_genres_count.values)
plt.title('Genre Counts')
plt.xlabel('Genre Type')
plt.ylabel('Counts')
plt.xticks(rotation=90)
plt.grid(axis='y')
plt.show()
```



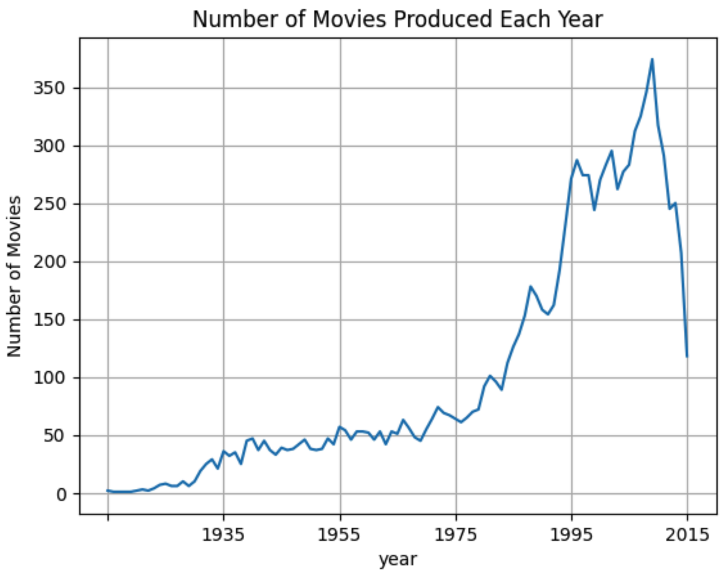
4.2 graph of genre count

Movies Released Each Year

By counting the value for the year, we can determine how many movies they release each year. This data is presented in a line graph.

```python
sorted_year_movie = movie_data['year'].value_counts().sort_index()
```

```python
#I want to know the trend of how many movies they make each year
sorted_year_movie.plot(kind='line')
#sns.lineplot(x=sorted_year_movie.index, y=sorted_year_movie.values)
plt.title('Number of Movies Produced Each Year')
plt.ylabel('Number of Movies')
plt.grid(True)
```



4.3 how many movies have been released each year

We noticed the line went down at 2015, I think that is because the dataset didn't include all the data for 2015.

Movies Released Each Year by Genre

Group the data by 'year' and 'genres' to understand the number of movies released in each genre annually. This is visualized using a line graph with different colors representing each genre.

```python
year_genre_change = exploded_movie_data.groupby(['year','genres']).size().unstack(fill_value=0)
year_genre_change
```

genres	(no genres listed)	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	IMAX	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
year																				
	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1902	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
1915	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0
1916	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1919	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2011	0	65	33	14	13	82	35	26	151	19	2	28	20	4	21	30	33	86	8	2
2012	1	56	24	17	7	72	33	20	114	18	1	27	23	10	7	20	26	64	7	2
2013	0	59	32	15	8	67	39	14	121	23	1	27	28	1	16	34	30	70	4	1
2014	1	55	26	13	11	64	32	10	102	11	0	12	13	3	12	16	25	46	11	6
2015	1	38	26	10	6	33	17	7	45	6	0	7	0	0	5	6	20	30	1	3

101 rows × 20 columns

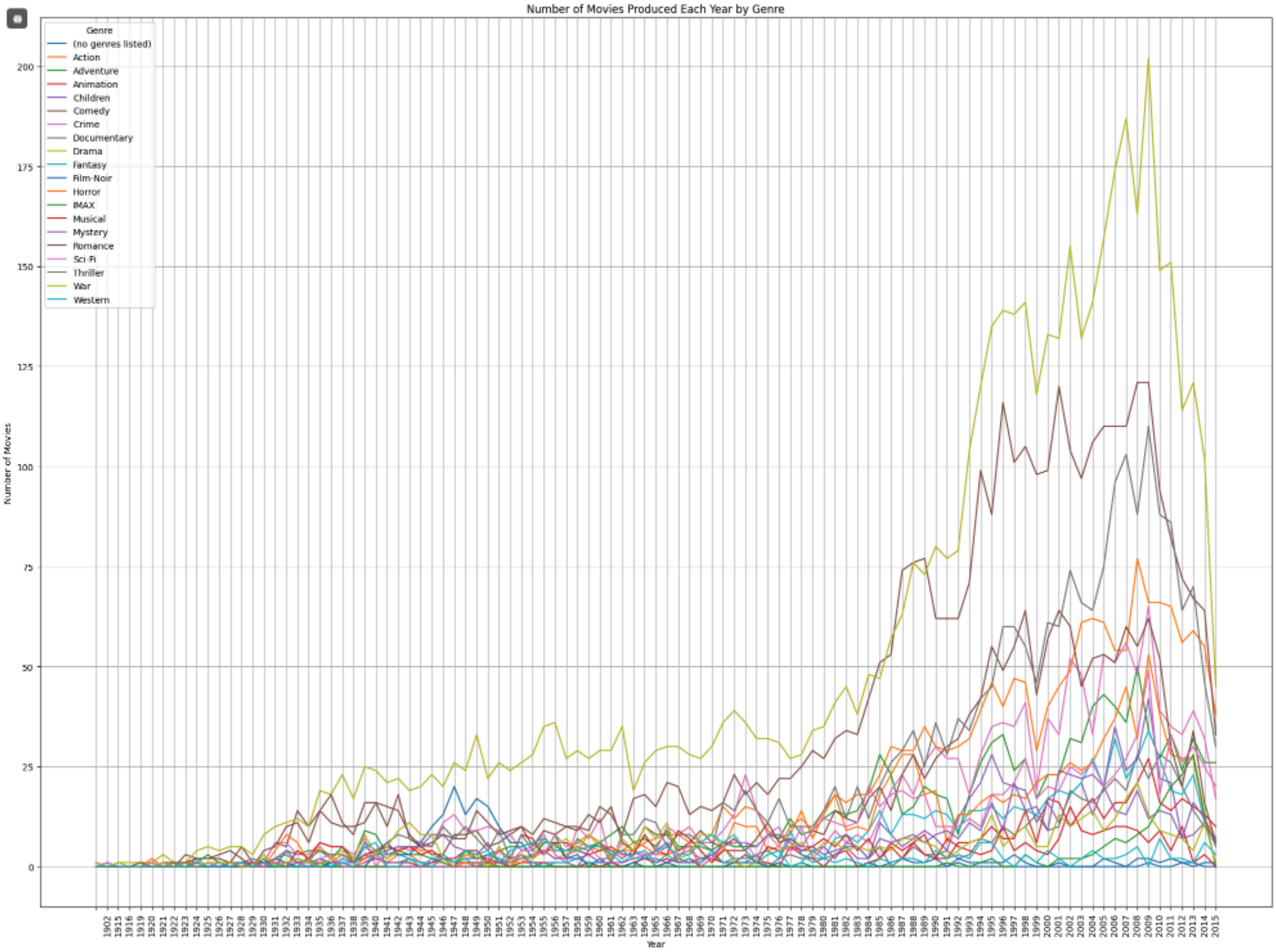
4.4 how many movies are released each year in each genre

```
plt.figure(figsize=(20, 15))

#genre_year_counts.index 是年份。
#genre_year_counts[genre] 是该类型在每年的电影数量。
#label=genre 为每条线指定了标签
#获取所有的类型列
for genre in year_genre_change.columns:
    #对于每一个类型绘制一条折线。
    plt.plot(year_genre_change.index, year_genre_change[genre], label=genre)


plt.title('Number of Movies Produced Each Year by Genre')
plt.xlabel('Year')
plt.ylabel('Number of Movies')
plt.legend(title='Genre')
plt.grid(True)
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

4.5 code for making a line graph



4.6 graph for how many movies are released each year in each genre

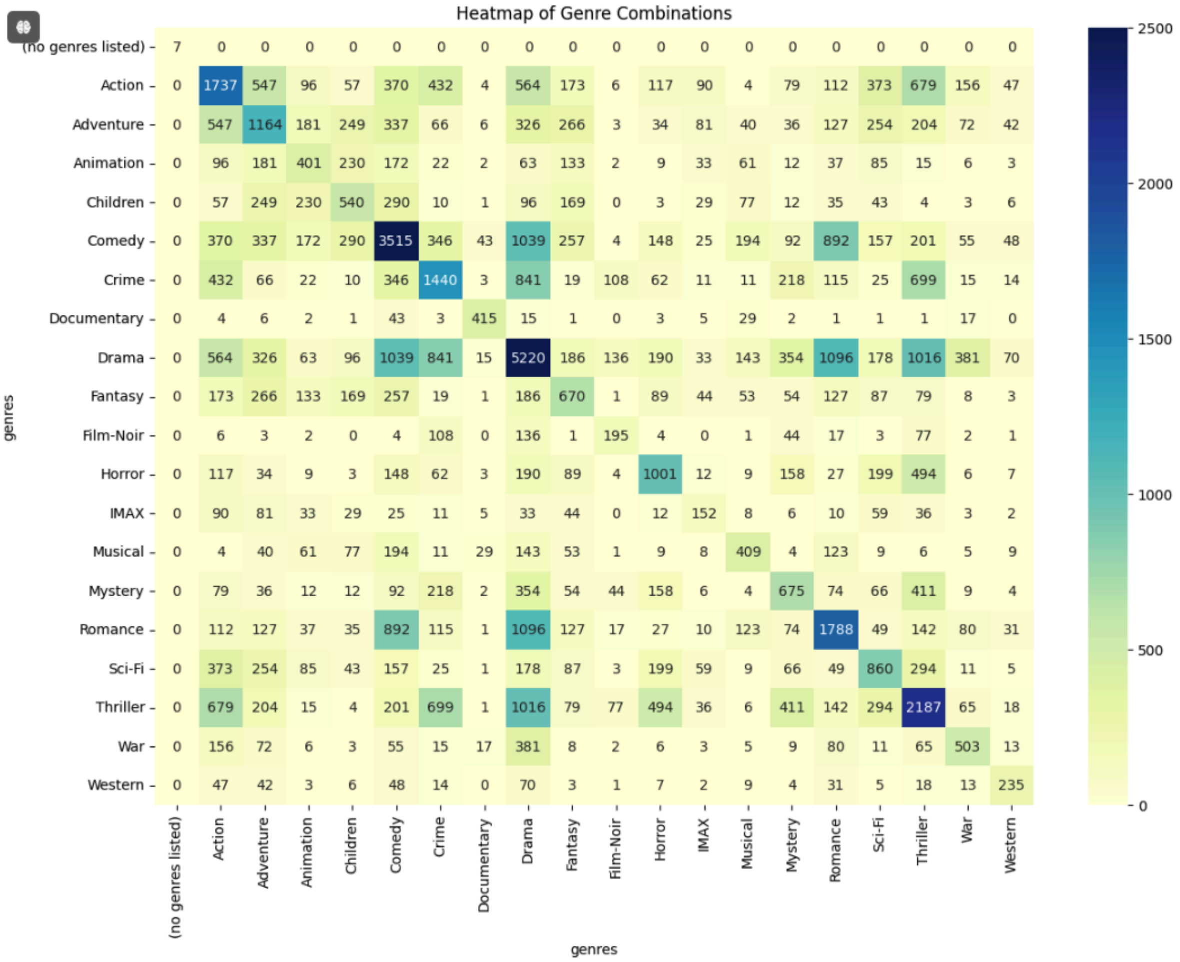
Heatmap of Genre Combinations

Create a heatmap to see the combinations of movie genres. Due to the high values for drama and comedy, the maximum value is set to 2500 to highlight other data.

```python
# Create a cross-tabulation of genres combinations
#pd.crosstab 函数用于创建一个交叉表，将ID和类型作为行和列，表格中的值表示某个电影属于某个类型的次数（在这个例子中应该是1或者0，因为每个电影属于的类型已经通过explode函数分解开了）
genre_combinations = pd.crosstab(exploded_movie_data['movieId'], exploded_movie_data['genres'])
#通过矩阵乘法计算类型组合的频率矩阵。
#genre_combinations.T：对交叉表进行转置，交换行和列。
#.dot(genre_combinations)：通过矩阵乘法，计算每种类型与其他类型组合出现的次数。
genre_combinations = genre_combinations.T.dot(genre_combinations)

# Plot the heatmap
plt.figure(figsize=(14, 10))
#annot=True：在热图的每个单元格上显示数值
#cmap='YlGnBu'：设置热图的颜色映射（黄-绿-蓝）。
sns.heatmap(genre_combinations, annot=True, fmt='d', cmap='YlGnBu', vmin=0, vmax=2500)
plt.title('Heatmap of Genre Combinations')
plt.show()
```

4.7 code to make a heat map



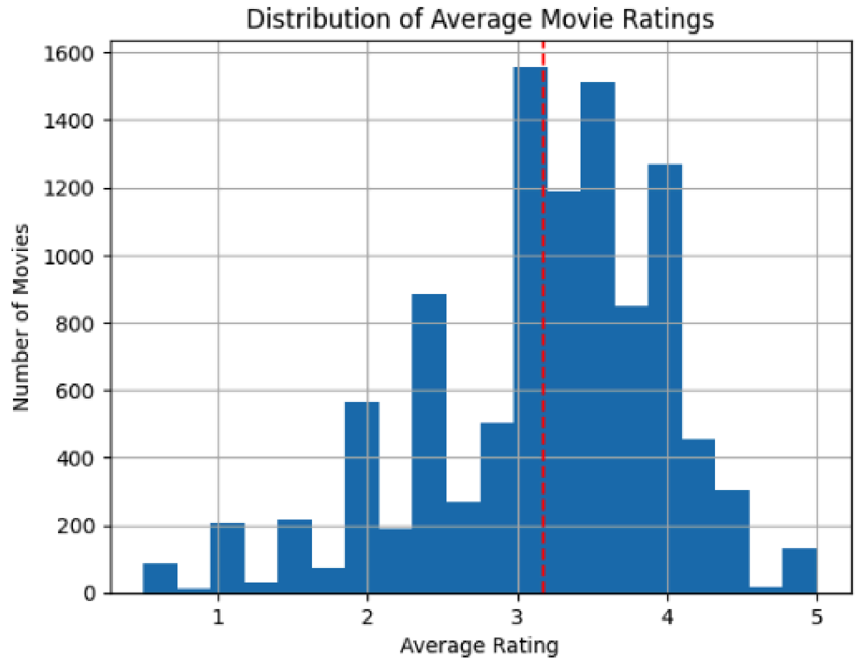
4.8 heat map for genre combination

Distribution of Average Movie Ratings

The average rating for all movies is around 3.1.

```
rat_ave = rating_data.groupby('movieId')['rating'].mean()
#rat_ave.rename({'rating': 'average_rating'}, inplace=True)
movie_ave_rating = pd.merge(movie_data, rat_ave, on='movieId', how='left')
```

```
movie_ave_rating['rating'].round(1).hist(bins=20)
average_rat = movie_ave_rating['rating'].mean()
plt.axvline(average_rat, color='red', linestyle='--', label=f'Average Rating: {average_rating:.2f}')
plt.title('Distribution of Average Movie Ratings')
plt.xlabel('Average Rating')
plt.ylabel('Number of Movies')
plt.show()
```



4.9 Average movie rating

Top 100 Popular Movies

Sort the movies by the number of ratings and select the top 100 most popular movies. Visualize the genres of these popular movies using a heatmap.

```
# Calculate the number of ratings for each movie
rating_counts = rating_data.groupby('movieId')['rating'].count().reset_index()
rating_counts.rename(columns={'rating': 'rating_count'}, inplace=True)

movie_rating_count = pd.merge(movie_data,rating_counts,on = 'movieId',how='left')
```

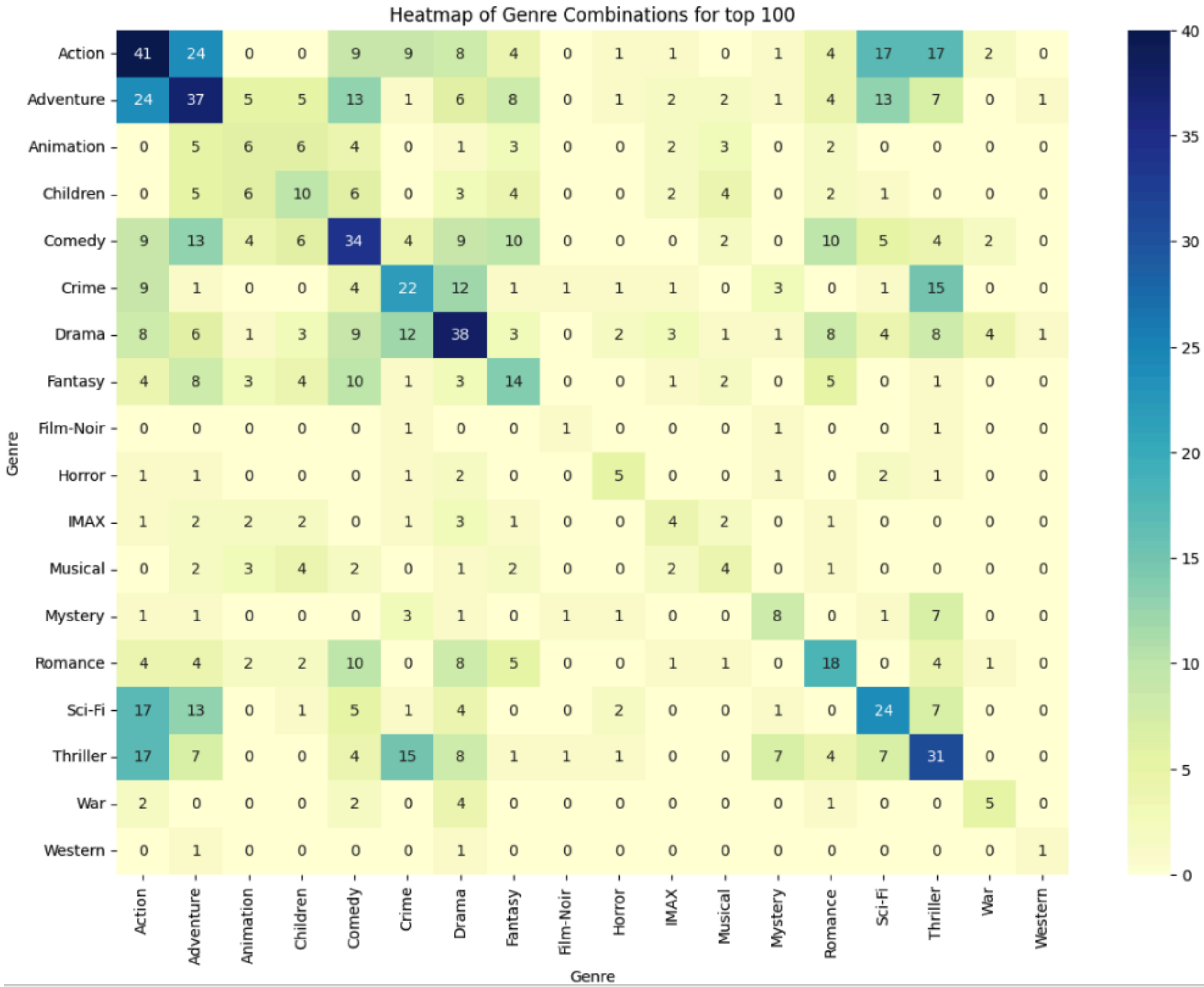
```
#top 100 popular movie
top100=movie_rating_count.sort_values(by='rating_count',ascending=False).head(100)
```

4.10 code to sort the top 100 popular movie

```
#To know in this 100 movies, genres distribution
exploded_top100 = top100.explode('genres')
combinations_top100 = pd.crosstab(exploded_top100['name'], exploded_top100['genres'])
combinations_top100 = combinations_top100.T.dot(combinations_top100)
```

```
plt.figure(figsize=(14, 10))
sns.heatmap(combinations_top100, annot=True, fmt='d', cmap='YlGnBu', vmin=0, vmax=40)
plt.title('Heatmap of Genre Combinations for top 100')
plt.xlabel('Genre')
plt.ylabel('Genre')
plt.show()
```

4.11 code to generate a heat map



4.12 heat map for the genre of the top 100 movies

Combined Movies and Ratings Data

Merge all the data into one DataFrame to analyze the average ratings for the top 100 movies. This is visualized using a plot graph.

```python
# 只保留 rating_data 中 movieId 在 top100 中的记录
top100_fix = movie_ave_rating[movie_ave_rating['movieId'].isin(top100['movieId'])]
```

```python
top100_with_rating = pd.merge(top100_fix[['movieId', 'rating']],top100[['movieId', 'name', 'year', 'genres', 'rating_count']],on='movieId',how='inner')
top100_with_rating
```

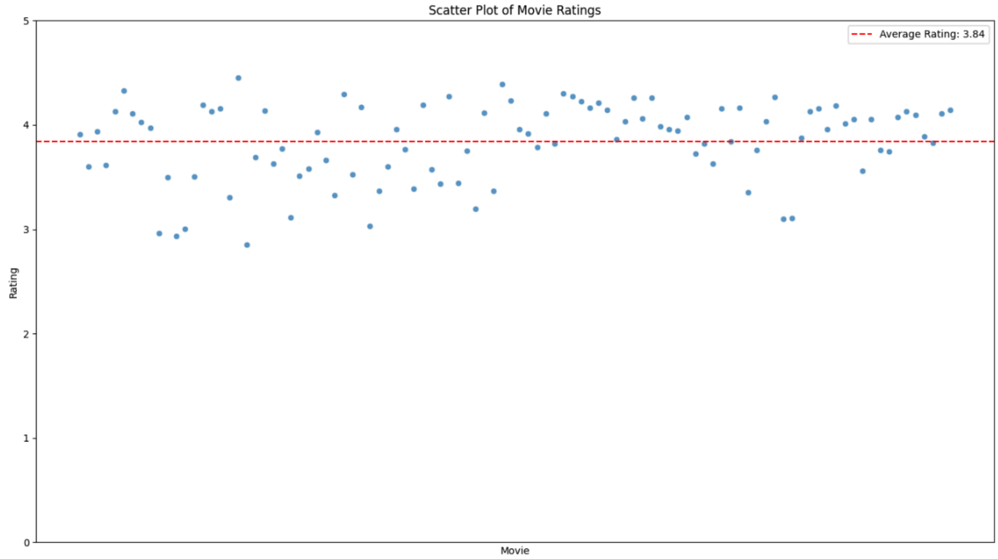
	movieId	rating	name	year	genres	rating_count
0	1	3.907328	Toy Story	1995	[Adventure, Animation, Children, Comedy, Fantasy]	232.0
1	10	3.600000	GoldenEye	1995	[Action, Adventure, Thriller]	135.0
2	32	3.939614	Twelve Monkeys (a.k.a. 12 Monkeys)	1995	[Mystery, Sci-Fi, Thriller]	207.0
3	34	3.616279	Babe	1995	[Children, Drama]	129.0
4	47	4.130102	Seven (a.k.a. Se7en)	1995	[Mystery, Thriller]	196.0
...	...	...	...	...	...	...
95	5952	4.095808	Lord of the Rings: The Two Towers, The	2002	[Adventure, Fantasy]	167.0
96	6377	3.888889	Finding Nemo	2003	[Adventure, Animation, Children, Comedy]	117.0
97	6539	3.829787	Pirates of the Caribbean: The Curse of the Bla...	2003	[Action, Adventure, Comedy, Fantasy]	141.0
98	7153	4.108434	Lord of the Rings: The Return of the King, The	2003	[Action, Adventure, Drama, Fantasy]	166.0
99	58559	4.141732	Dark Knight, The	2008	[Action, Crime, Drama, IMAX]	127.0

100 rows × 6 columns

4.13 code and result for combined movies and ratings

```python
plt.figure(figsize=(14, 8))
average_rating = top100_with_rating['rating'].mean()
plt.axhline(average_rating, color='red', linestyle='--', label=f'Average Rating: {average_rating:.2f}')
sns.scatterplot(data=top100_with_rating, x='name', y='rating', alpha=0.7)
plt.title('Scatter Plot of Movie Ratings')
plt.xlabel('Movie')
plt.ylabel('Rating')
plt.ylim(0, 5)
plt.xticks([])
plt.tight_layout()
plt.show()
```

4.14 code to generate plot graph



4.15 graph for top 100 movies rating

Results

Dominant Genres: Drama is the most produced genre, making up about half of all movies. Comedy is the second most popular genre, accounting for approximately 30% of movies.

Production Trends: The number of films produced each year has been increasing, with significant growth around 1990 and 2010.

Genre Evolution: From the 1930s to the 1960s, most movies were dramas. Since the 1960s, comedy has become more prevalent. Both drama and comedy saw a surge in production from the 1980s, with other genres increasing since the 1990s.

Popular Genre Combinations: The most common multi-genre movies are Comedy-Drama, Comedy-Romance, Drama-Crime, Drama-Romance, and Drama-Thriller.

Movie Ratings: The average rating for all movies is around 3.1, with most movies receiving ratings between 3 and 3.5. However, the top 100 most popular movies have a higher average rating of 3.8, indicating that highly rated movies tend to attract more people.

Conclusion

This analysis shows the film industry and audience preferences. While certain genres like drama and comedy have remained popular over the decades, but recent years other genres of movies also evolved. Understanding these trends can help producers make decisions to create movies that resonate with audiences.