project

November 12, 2024

1 Project6-Movie Popularity and Rating Trends Analysis

1.1 Outline

1. Data Cleaning: Preprocess the data to handle missing values, outliers, and duplicates.
2. Movie Rating Analysis: Analyze the average ratings (vote_average) and identify the highest and lowest rated movies.
3. Popularity Analysis: Study the relationship between movie popularity (popularity) and ratings, as well as the number of votes (vote_count).
4. Genre Analysis: Use genre data (genre_ids) to analyze the ratings and popularity of different movie genres.
5. Temporal Analysis: Explore trends in movie ratings and popularity across different release years.
6. Data Visualization: Visualize the distribution, ratings, and popularity of movies using charts and graphs.
7. Recommendation System Basics: Attempt to recommend movies to users using a simple algorithm.

1.2 Requirements:

1. Use Python for data processing and analysis.
2. Employ the Pandas library for data manipulation.
3. Use Matplotlib, Seaborn, or Plotly for data visualization.
4. Document the analysis process and results in a Jupyter Notebook.
5. Submit a report that includes code, analysis results, and visualizations.

1.3 Task1:Data Cleaning

1.3.1 Code 01

[61]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 1.Load the data
df = pd.read_csv('top_rated_9000_movies_on_TMDB.csv')

# 2.Clean the data
# Check for missing values
```

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

# Delete missing values
df = df.dropna()

# Check duplicate value
df = df.drop_duplicates()
```

```
id                  0
title               0
original_language   0
release_date        0
vote_average        0
vote_count          0
popularity          0
overview            0
genre_ids           0
Genres              0
dtype: int64
```

1.3.2 Analysis 01

In this step, we read the data in the file and check the integrity of the file and the uniqueness of the data.

1.4 Task2:Movie Rating Analysis

1.4.1 Code 02

[62]:
```
# Analyze average ratings
highest_rated = df[df['vote_average'] == df['vote_average'].max()]
lowest_rated = df[df['vote_average'] == df['vote_average'].min()]
print(highest_rated)
print(lowest_rated)
```

```
    id                     title original_language release_date  vote_average  \
0  278  The Shawshank Redemption                en   1994-09-23         8.706

   vote_count  popularity                                           overview  \
0       26840     150.307  Imprisoned in the 1940s for the double murder …

  genre_ids              Genres
0  [18, 80]  ['Drama', 'Crime']
         id                         title original_language release_date  \
9629  40016  Birdemic: Shock and Terror                en   2010-02-27

      vote_average  vote_count  popularity  \
9629           2.2         331       6.548
```

```
                                                  overview        genre_ids  \
9629  A platoon of eagles and vultures attacks the r…  [10749, 27, 53]

                                  Genres
9629  ['Romance', 'Horror', 'Thriller']
```

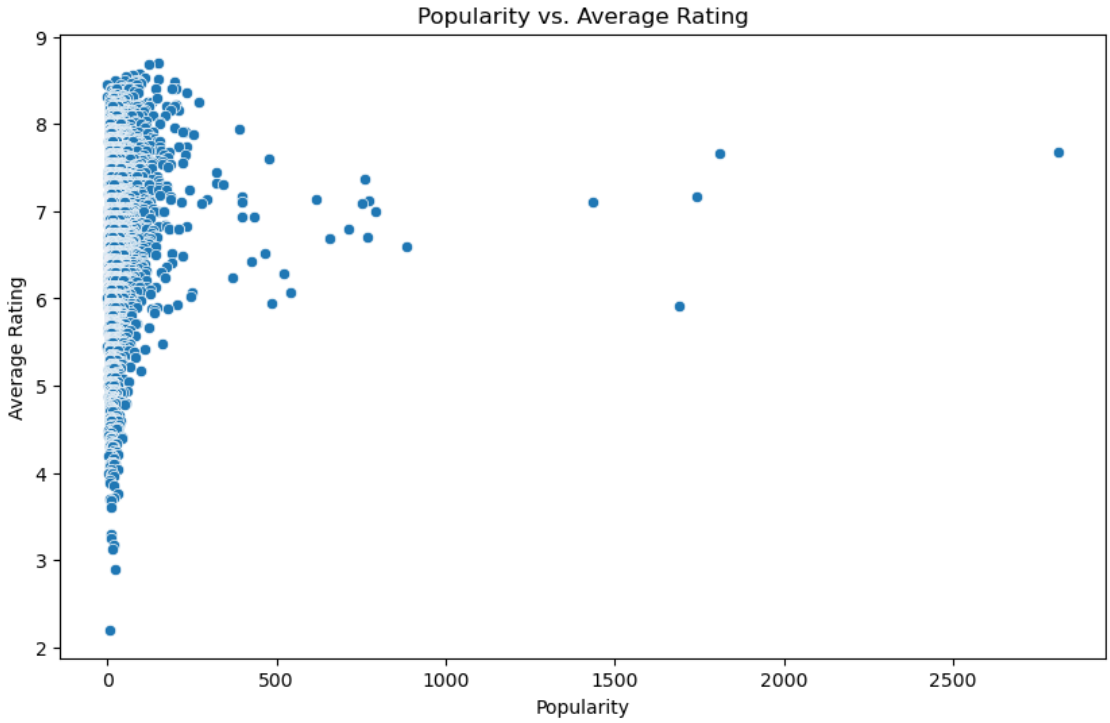
1.4.2 Analysis 02

In this step, we found the two movies in the data with the highest and lowest median values in the "vote_average" category.

1.5 Task3:Popularity Analysis

1.5.1 Code 03

[63]:
```
# Relationship between popularity and ratings
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='popularity', y='vote_average')
plt.title('Popularity vs. Average Rating')
plt.xlabel('Popularity')
plt.ylabel('Average Rating')
plt.show()
```



1.5.2 Analysis 03

In this step, we use both “popularity” and “average vote” data to create a chart that more clearly shows the conclusion that “the higher the” average vote “of the film, the higher the general” popularity ” .

1.6 Task4:Genre Analysis

1.6.1 Code 04

[64]:
```
# Analyze ratings and popularity by genre
genre_ratings = df.explode('genre_ids').groupby('genre_ids')['vote_average'].
 ↪mean()
genre_popularity = df.explode('genre_ids').groupby('genre_ids')['popularity'].
 ↪mean()
genre_merge_df = pd.merge(genre_ratings, genre_popularity, on='genre_ids',␣
 ↪how='left')

# Print the results
print(genre_ratings)
print(genre_popularity)
print(genre_merge_df)
```

```
genre_ids
[10402, 10749, 18]                    6.583333
[10402, 10749, 35]                    7.000000
[10402, 10751, 18]                    5.504000
[10402, 14, 35, 878, 10751, 10770]    7.400000
[10402, 16, 10751, 14]                7.104000
                                          …
[9648, 80, 18]                        6.343000
[9648, 80, 53]                        6.849444
[9648, 878, 53]                       5.803800
[9648, 878]                           6.970000
[9648]                                6.837000
Name: vote_average, Length: 2045, dtype: float64
genre_ids
[10402, 10749, 18]                     9.739333
[10402, 10749, 35]                     8.852000
[10402, 10751, 18]                     8.256000
[10402, 14, 35, 878, 10751, 10770]    24.786000
[10402, 16, 10751, 14]                17.846000
                                          …
[9648, 80, 18]                        12.094000
[9648, 80, 53]                        14.863778
[9648, 878, 53]                       15.947800
[9648, 878]                           17.019500
[9648]                                13.301500
Name: popularity, Length: 2045, dtype: float64
```

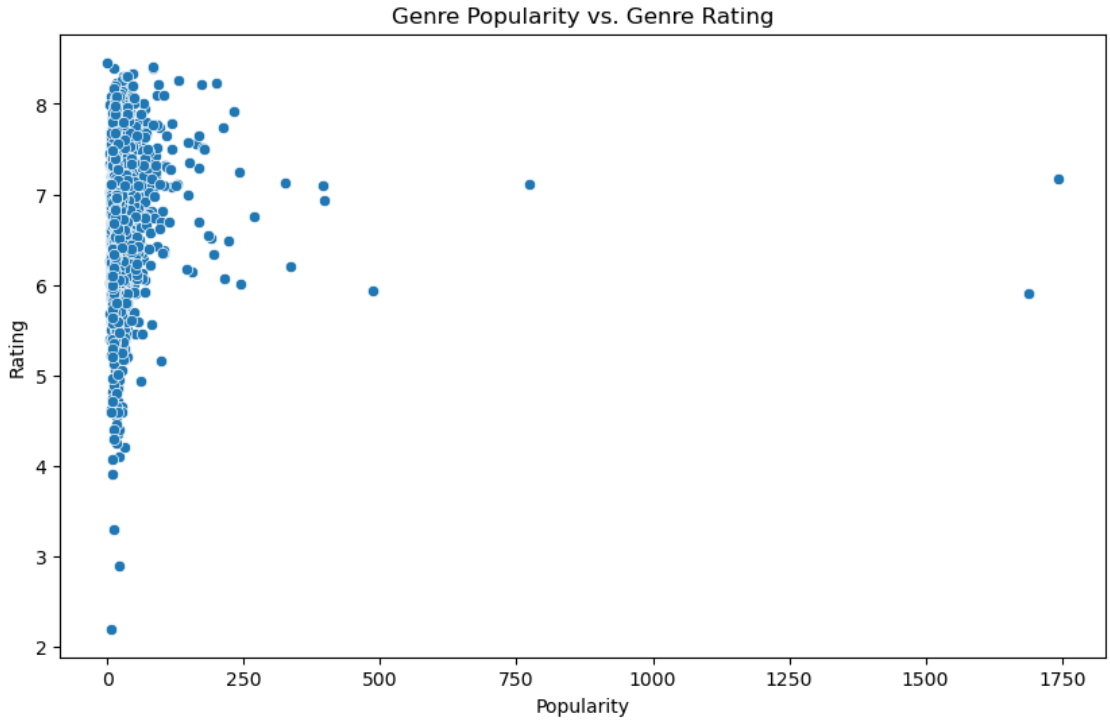
```
                                    vote_average  popularity
genre_ids
[10402, 10749, 18]                      6.583333    9.739333
[10402, 10749, 35]                      7.000000    8.852000
[10402, 10751, 18]                      5.504000    8.256000
[10402, 14, 35, 878, 10751, 10770]      7.400000   24.786000
[10402, 16, 10751, 14]                  7.104000   17.846000
…                                          …          …
[9648, 80, 18]                          6.343000   12.094000
[9648, 80, 53]                          6.849444   14.863778
[9648, 878, 53]                         5.803800   15.947800
[9648, 878]                             6.970000   17.019500
[9648]                                  6.837000   13.301500

[2045 rows x 2 columns]
```

1.6.2 Analysis 04

In this step, we merged the same movies by genre and calculated their average rating and popularity. We then merge the new dataframe based on "genre_ids".

1.6.3 Code 05

[65]:
```
# Data visualization
plt.figure(figsize=(10, 6))
sns.scatterplot(data=genre_merge_df, x='popularity', y='vote_average')
plt.title('Genre Popularity vs. Genre Rating')
plt.xlabel('Popularity')
plt.ylabel('Rating')
plt.show()
```



1.6.4 Analysis 05

In this step, we visualized the data based on the newly synthesized dataframe. They concluded that "films with higher average ratings are generally more popular".

1.7 Task5:Temporal Analysis

1.7.1 Code 06

[66]:
```
# Trends in ratings and popularity over years
df['release_year'] = pd.to_datetime(df['release_date']).dt.year
yearly_ratings = df.groupby('release_year')['vote_average'].mean()
yearly_popularity = df.groupby('release_year')['popularity'].mean()

# Print results
print(yearly_ratings)
print(yearly_popularity)
```

```
release_year
1902    7.919000
1903    7.000000
1915    6.030000
1916    7.081000
1918    7.300000
          ...
```

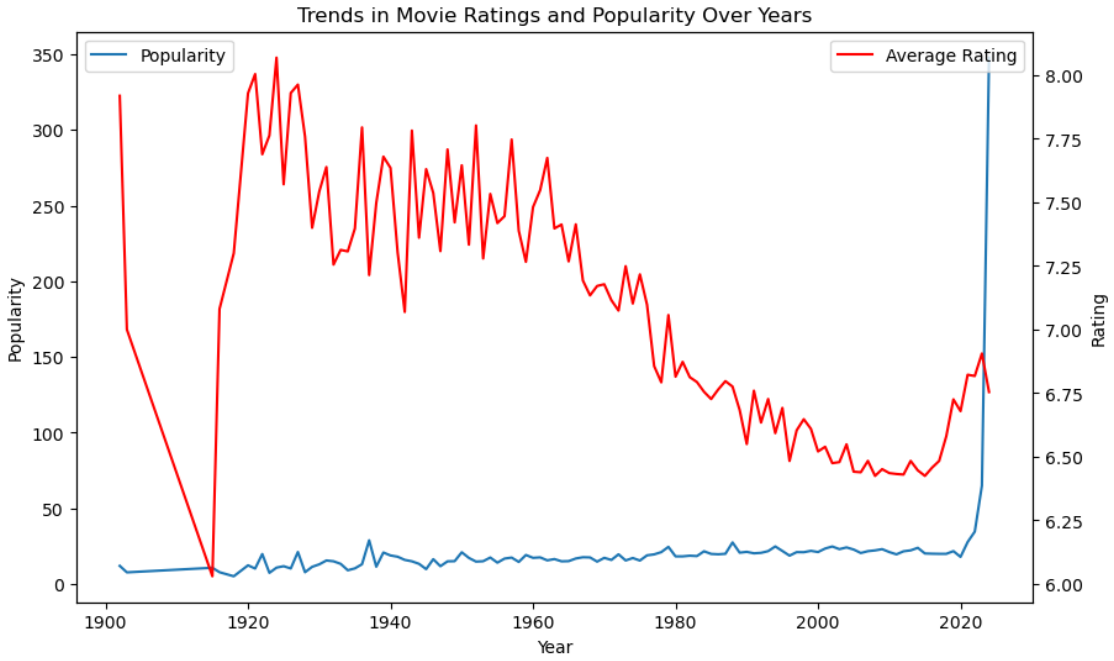
```
2020    6.679186
2021    6.822403
2022    6.817432
2023    6.905922
2024    6.754358
Name: vote_average, Length: 110, dtype: float64
release_year
1902     12.065000
1903      7.767000
1915     10.808000
1916      7.869000
1918      5.188000
            …
2020     17.975848
2021     27.813003
2022     34.685953
2023     65.091060
2024    347.780235
Name: popularity, Length: 110, dtype: float64
```

1.7.2 Analysis 06

In this step, we calculated the average rating and popularity of each released film for each year and made dataframes.

1.7.3 Code 07

[67]:
```
# Data visualization
fig, ax1 = plt.subplots(figsize=(10, 6))
sns.lineplot(data=yearly_popularity, label='Popularity')
plt.title('Trends in Movie Ratings and Popularity Over Years')
plt.xlabel('Year')
plt.ylabel('Popularity')
ax2 = ax1.twinx()
sns.lineplot(data=yearly_ratings, label='Average Rating', color='red')
ax2.set_ylabel('Rating')
ax2.tick_params(axis='y')
ax2.set_ylim(ax2.get_ylim()[0], ax2.get_ylim()[1])
plt.legend()
plt.show()
```



1.7.4 Analysis 07

In this step, we visualize the two dataframes we obtained in the previous step. Two conclusions were drawn.

1. In general, the more recent the films, the more popular they are. And there will be an explosive growth in movies after 2020.It shows that people pay more attention to movies released in recent years.

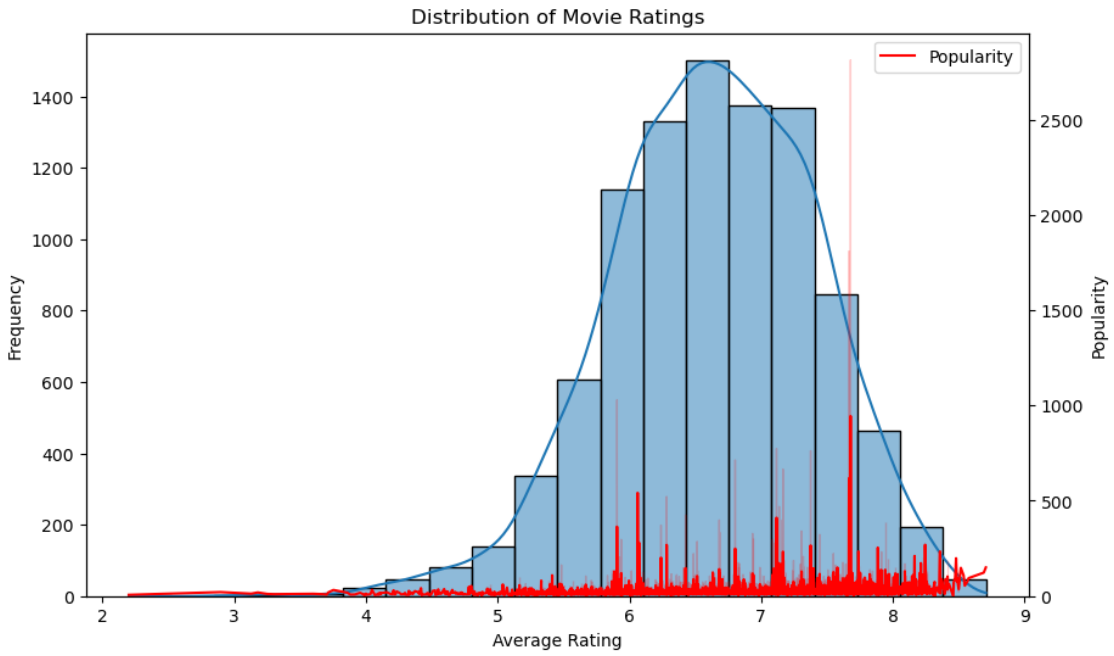
2. In general, older films have higher average ratings. Despite the downward trend in the overall rating, the average rating has rebounded in the past five years. It reflects people's praise for the past classics and their dissatisfaction with the quality of recent films.

1.8 Task6:Data Visualization

1.8.1 Code 08

[68]:
```
# Distribution of ratings
fig, ax1 = plt.subplots(figsize=(10, 6))
sns.histplot(df['vote_average'], bins=20, kde=True)
plt.title('Distribution of Movie Ratings')
plt.xlabel('Average Rating')
plt.ylabel('Frequency')
ax2 = ax1.twinx()
```

```
# Plot the popularity curve
sns.lineplot(data=df, x=df['vote_average'], y=df['popularity'], ax=ax2,␣
 ↪color='red', label='Popularity')
ax2.set_ylabel('Popularity')
ax2.tick_params(axis='y')
ax2.set_ylim(0, ax2.get_ylim()[1])

plt.show()
```



1.8.2 Analysis 08

In this step, we realized the data visualization of movie rating distribution and popularity. Two conclusions are drawn.

1. The movie rating distribution presents an elliptical distribution. Most of them are between 5.5 and 7.5. This means that the average level of film production is around 6.5, with not many films either very good or very bad.

2. The popularity of a film increases with its rating. It also shows that rating is powerful in judging whether a film is an excellent film recognized by the public.

1.9 Task7:Recommendation System Basics

1.9.1 Code 09

[69]:
```
# Simple recommendation based on highest ratings
def recommend_movies(n=10):
    df['recommendation_index'] = df['vote_average'] * 5 + df['popularity'] * 0.
 ↪01
    return df.nlargest(n, 'recommendation_index')[['title',␣
 ↪'recommendation_index','vote_average','popularity']]

# Example usage
print(recommend_movies(10))
```

```
                                          title  recommendation_index  \
818                          Deadpool & Wolverine              66.51272
848                                  Inside Out 2              56.42615
2557                              Despicable Me 4              53.26127
2766                       Beetlejuice Beetlejuice              49.84179
8028                                  Borderlands              46.42252
0                           The Shawshank Redemption              45.03307
1                                    The Godfather              44.67973
1723                                   Beetlejuice              44.49713
12    The Lord of the Rings: The Return of the King              44.38343
49              Spider-Man: Across the Spider-Verse              44.12974

      vote_average  popularity
818          7.679    2811.772
848          7.667    1809.115
2557         7.166    1743.127
2766         7.100    1434.179
8028         5.906    1689.252
0            8.706     150.307
1            8.690     122.973
1723         7.376     761.713
12           8.480     198.343
49           8.358     233.974
```

1.9.2 Analysis 09

In this step, we set up a function that calculates the recommendation index for each movie by rating and popularity in a ratio of 500:1 and returns the top ten movies in the recommendation index. Based on the comprehensive rating and popularity, 10 movies are recommended to users.