movie_analysis

August 5, 2025

1 TMDB

2025 08 01

1.1

1.
2.
3. (ROI)
4.

1.2 1.

[1]:
```
#
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import ast #    python
from collections import Counter


#
sns.set_style("whitegrid")
plt.rcParams['font.sans-serif'] = ['SimHei']  #
plt.rcParams['axes.unicode_minus'] = False    #

#
movies_df = pd.read_csv("..\\data\\tmdb_5000_movies.csv")
credits_df = pd.read_csv("..\\data\\tmdb_5000_credits.csv")

print("    ",movies_df.shape)
print("     ",credits_df.shape)
```

```
    (4803, 20)
     (4803, 4)
```

[2]:
```
#     2
print(movies_df.head(2))
print("*"*50)
#
print(movies_df.info())
```

```
      budget                                             genres  \
0  237000000  [{"id": 28, "name": "Action"}, {"id": 12, "nam…
1  300000000  [{"id": 12, "name": "Adventure"}, {"id": 14, "…

                                      homepage     id  \
0                  http://www.avatarmovie.com/  19995
1  http://disney.go.com/disneypictures/pirates/    285

                                            keywords original_language  \
0  [{"id": 1463, "name": "culture clash"}, {"id":…                en
1  [{"id": 270, "name": "ocean"}, {"id": 726, "na…                en

                             original_title  \
0                                    Avatar
1  Pirates of the Caribbean: At World's End

                                            overview  popularity  \
0  In the 22nd century, a paraplegic Marine is di…  150.437577
1  Captain Barbossa, long believed to be dead, ha…  139.082615

                                production_companies  \
0  [{"name": "Ingenious Film Partners", "id": 289…
1  [{"name": "Walt Disney Pictures", "id": 2}, {"…

                                production_countries release_date     revenue  \
0  [{"iso_3166_1": "US", "name": "United States o…   2009-12-10  2787965087
1  [{"iso_3166_1": "US", "name": "United States o…   2007-05-19   961000000

   runtime                                   spoken_languages    status  \
0    162.0  [{"iso_639_1": "en", "name": "English"}, {"iso…  Released
1    169.0           [{"iso_639_1": "en", "name": "English"}]  Released

                                          tagline  \
0                     Enter the World of Pandora.
1  At the end of the world, the adventure begins.

                                      title  vote_average  vote_count
0                                    Avatar           7.2       11800
1  Pirates of the Caribbean: At World's End           6.9        4500
**************************************************
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype
```

```
---  ------                --------------  -----
 0   budget                4803 non-null   int64
 1   genres                4803 non-null   object
 2   homepage              1712 non-null   object
 3   id                    4803 non-null   int64
 4   keywords              4803 non-null   object
 5   original_language     4803 non-null   object
 6   original_title        4803 non-null   object
 7   overview              4800 non-null   object
 8   popularity            4803 non-null   float64
 9   production_companies  4803 non-null   object
 10  production_countries  4803 non-null   object
 11  release_date          4802 non-null   object
 12  revenue               4803 non-null   int64
 13  runtime               4801 non-null   float64
 14  spoken_languages      4803 non-null   object
 15  status                4803 non-null   object
 16  tagline               3959 non-null   object
 17  title                 4803 non-null   object
 18  vote_average          4803 non-null   float64
 19  vote_count            4803 non-null   int64
dtypes: float64(3), int64(4), object(13)
memory usage: 750.6+ KB
None
```

[3]:
```
#      2
print(credits_df.head(2))
print("*"*50)
#
print(credits_df.info())
```

```
   movie_id                                     title  \
0     19995                                    Avatar
1       285  Pirates of the Caribbean: At World's End

                                                cast  \
0  [{"cast_id": 242, "character": "Jake Sully", "…
1  [{"cast_id": 4, "character": "Captain Jack Spa…

                                                crew
0  [{"credit_id": "52fe48009251416c750aca23", "de…
1  [{"credit_id": "52fe4232c3a36847f800b579", "de…
**************************************************
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
```

```
 0   movie_id  4803 non-null   int64
 1   title     4803 non-null   object
 2   cast      4803 non-null   object
 3   crew      4803 non-null   object
dtypes: int64(1), object(3)
memory usage: 150.2+ KB
None
```

[4]:
```
#
movies_df[["budget","revenue","popularity","runtime","vote_average","vote_count"]].
  ↪describe()
```

[4]:
```
             budget       revenue   popularity      runtime  vote_average  \
count  4.803000e+03  4.803000e+03  4803.000000  4801.000000   4803.000000
mean   2.904504e+07  8.226064e+07    21.492301   106.875859      6.092172
std    4.072239e+07  1.628571e+08    31.816650    22.611935      1.194612
min    0.000000e+00  0.000000e+00     0.000000     0.000000      0.000000
25%    7.900000e+05  0.000000e+00     4.668070    94.000000      5.600000
50%    1.500000e+07  1.917000e+07    12.921594   103.000000      6.200000
75%    4.000000e+07  9.291719e+07    28.313505   118.000000      6.800000
max    3.800000e+08  2.787965e+09   875.581305   338.000000     10.000000

         vote_count
count   4803.000000
mean     690.217989
std     1234.585891
min        0.000000
25%       54.000000
50%      235.000000
75%      737.000000
max    13752.000000
```

: -   4803   -   (budget)   (revenue)   0   -   (genres) JSON   -   6.09 (10)

1.3 2.

[5]:
```
#
credits_df = credits_df[["movie_id","cast","crew"]]
movies_df = movies_df.merge(credits_df,left_on="id",right_on="movie_id")
print(movies_df.columns)
```

```
Index(['budget', 'genres', 'homepage', 'id', 'keywords', 'original_language',
       'original_title', 'overview', 'popularity', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'tagline', 'title', 'vote_average',
       'vote_count', 'movie_id', 'cast', 'crew'],
      dtype='object')
```

[6]:
```
#
key_columns =␣
  ↪["id","title","genres","keywords","release_date","runtime","budget","revenue","original_lan
                  ␣
  ↪"vote_count","cast","crew","overview","production_companies","production_countries","popula
movies_df = movies_df[key_columns]
```

[7]:
```
#
print("    ")
print(movies_df.isnull().sum()) #           / /

#  runtime
movies_df["runtime"] = movies_df["runtime"].fillna(movies_df["runtime"].
  ↪median())
#  release_date
movies_df = movies_df.dropna(subset=["release_date"])
```

```
id                      0
title                   0
genres                  0
keywords                0
release_date            1
runtime                 2
budget                  0
revenue                 0
original_language       0
vote_average            0
vote_count              0
cast                    0
crew                    0
overview                3
production_companies    0
production_countries    0
popularity              0
dtype: int64
```

[8]:
```
#              0     1000
movies_df = movies_df[(movies_df["budget"]>1000) & (movies_df["revenue"]>1000)]
# movies_df = movies_df[movies_df["vote_count"] >= 0]
```

[9]:
```
#
movies_df["release_date"] = pd.to_datetime(movies_df["release_date"]) #␣
  ↪    datetime
movies_df["release_year"] = movies_df["release_date"].dt.year #
movies_df["release_month"] = movies_df["release_date"].dt.month #
```

```
#     profit     ROI
movies_df["profit"] = movies_df["revenue"] - movies_df["budget"]
movies_df["roi"] = (movies_df["profit"]/movies_df["budget"])*100 #

#  RIO
movies_df.loc[movies_df["budget"] <= 0,'roi'] = np.nan
movies_df["roi"] = movies_df["roi"].replace([np.inf,-np.inf] , np.nan)
```

[10]:
```
#  JSON  (genres, keywords, cast, crew, production_companies,␣
 ↪production_countries)
#        JSON    Python   /
def parse_json_column(column):
    try:
        return ast.literal_eval(column)
    except (ValueError,SyntaxError):
        return [] #

json_columns =␣
 ↪["genres","keywords","cast","crew","production_companies","production_countries"]
for col in json_columns:
    movies_df[col] = movies_df[col].apply(parse_json_column)
```

[11]:
```
# crew
def get_director(crew_list):
    for person in crew_list:
        if person["job"] == "Director":
            return person["name"]
    return np.nan

movies_df["director"] = movies_df["crew"].apply(get_director)

#  genres
def get_genres_list(genre_list):
    if isinstance(genre_list,list):
        return [genre["name"] for genre in genre_list]
    else:
        return []

movies_df["genres_list"] = movies_df["genres"].apply(get_genres_list)
```

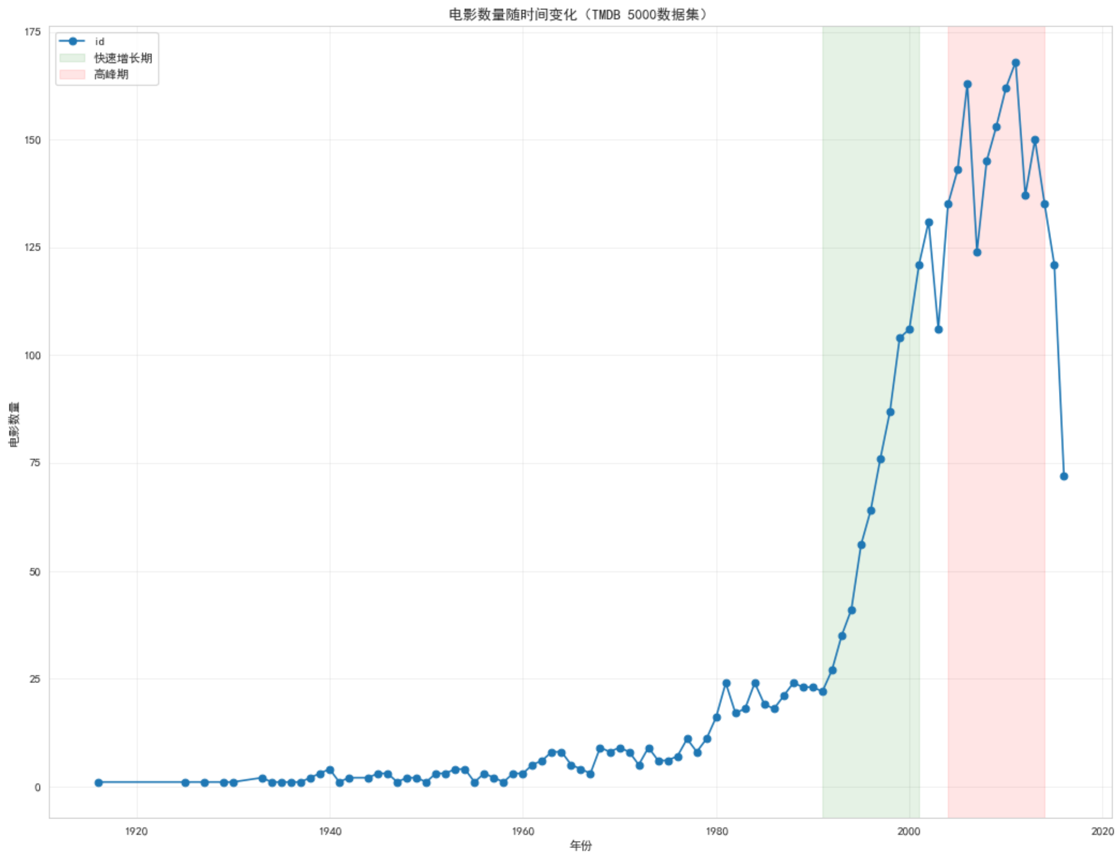
[12]:
```
#
print("     ",movies_df.shape)
movies_df[["title","release_year","budget","revenue","profit","roi","genres_list","director"]]
 ↪head(3)
```

```
(3211, 23)
```

[12]:
```
                                      title  release_year     budget  \
0                                    Avatar          2009  237000000
1  Pirates of the Caribbean: At World's End          2007  300000000
2                                   Spectre          2015  245000000

      revenue      profit          roi  \
0  2787965087  2550965087  1076.356577
1   961000000   661000000   220.333333
2   880674609   635674609   259.459024

                                    genres_list         director
0  [Action, Adventure, Fantasy, Science Fiction]    James Cameron
1                  [Adventure, Fantasy, Action]  Gore Verbinski
2                    [Action, Adventure, Crime]       Sam Mendes
```

1.4 3. EDA

[13]:
```python
#
plt.figure(figsize=(16,12),dpi=80)
movies_per_year = movies_df.groupby('release_year')['id'].count()
movies_per_year.plot(kind='line',marker='o')
plt.title("     TMDB 5000   ")
plt.xlabel(" ")
plt.ylabel("  ")
plt.grid(True,alpha=0.3)
plt.axvspan(1991,2001,color='green',alpha=0.1,label='   ')
plt.axvspan(2004,2014,color='red',alpha=0.1,label='  ')
plt.legend()
plt.show()
```



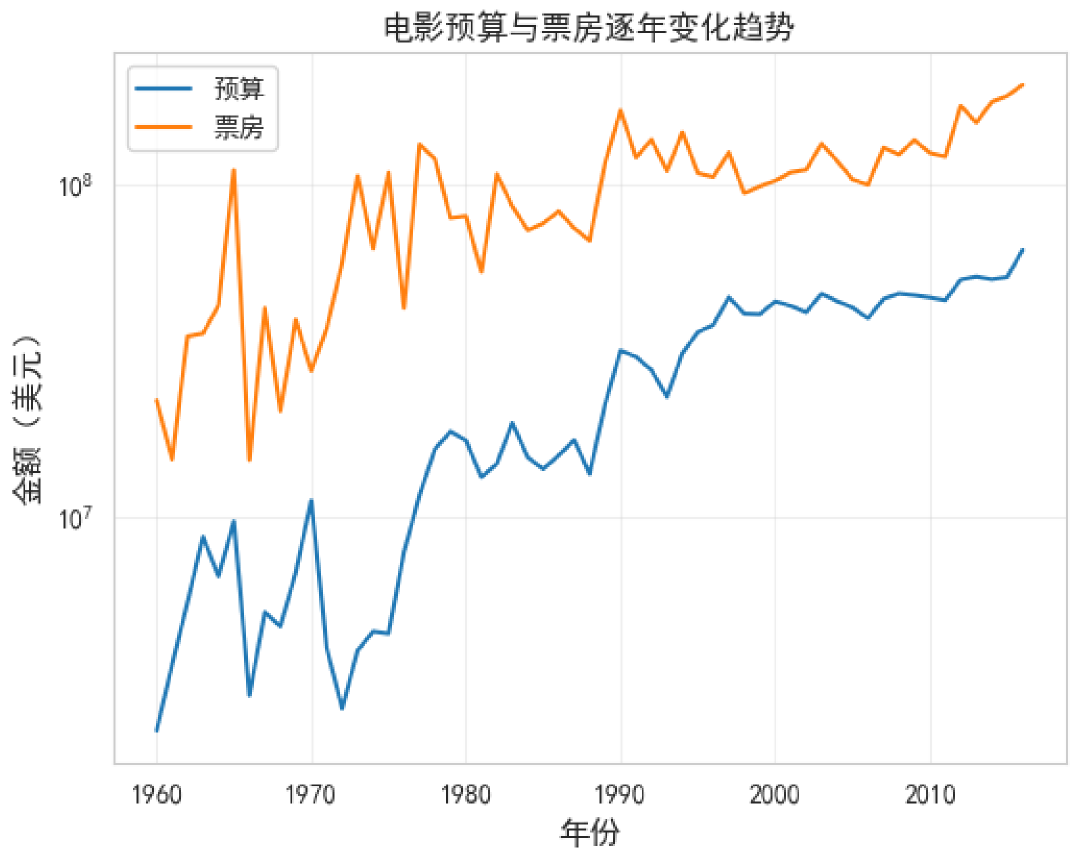
: - 1960 1960

[14]:
```
#
movies_df = movies_df[movies_df['release_year'] >= 1960]
#
print("1960      ",movies_df.shape)
```

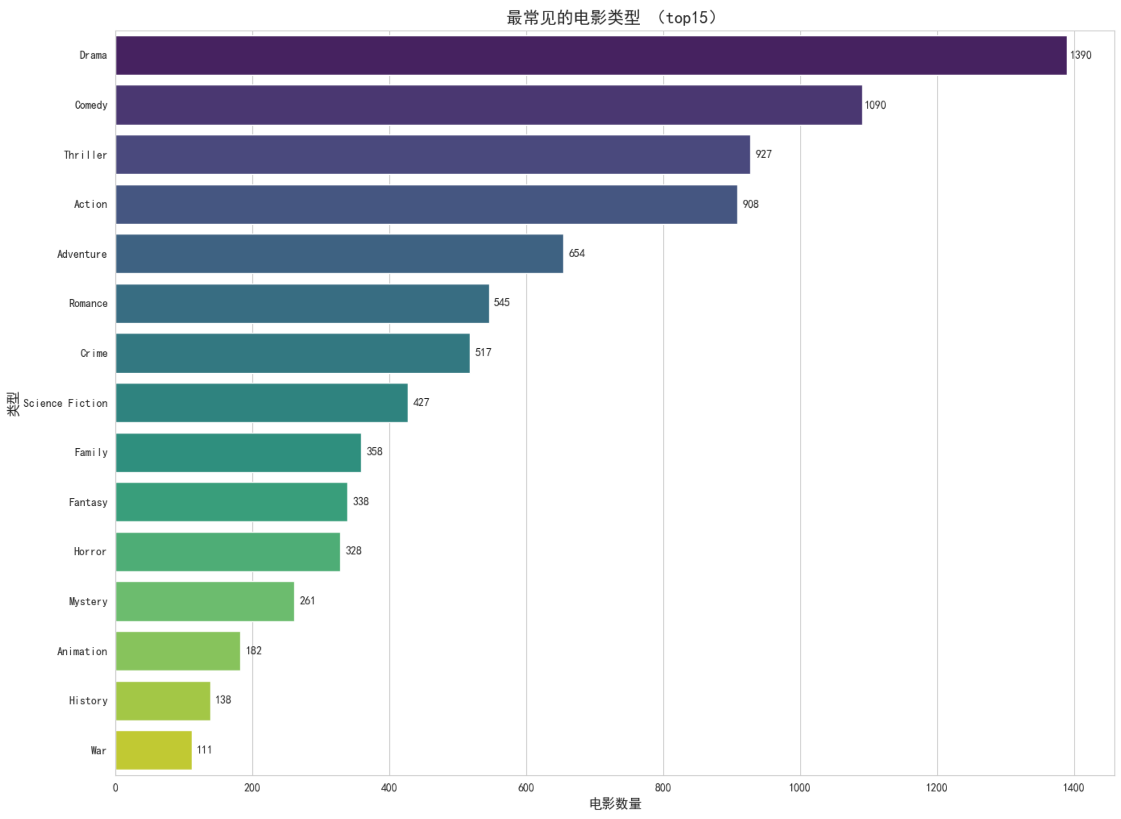
```
1960       (3150, 23)
```

[15]:
```
#
plt.figure(figsize=(16,12),dpi=80)
movies_df.groupby("release_year")[["budget","revenue"]].mean().plot(kind='line')
plt.title("        ")
plt.xlabel("  ",fontsize=12)
plt.ylabel("    ",fontsize=12)
plt.legend(["  ","  "])
plt.yscale('log') #
plt.grid(True,alpha=0.3)
plt.show()
```

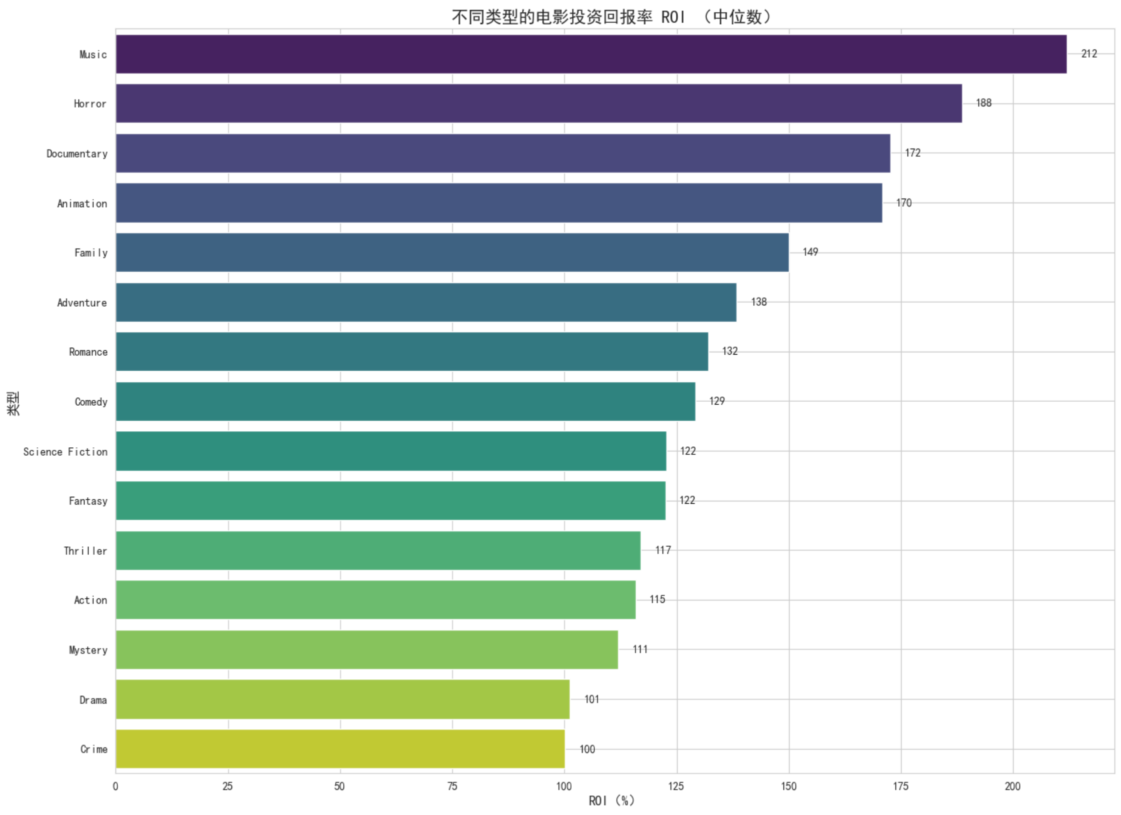
```
<Figure size 1280x960 with 0 Axes>
```



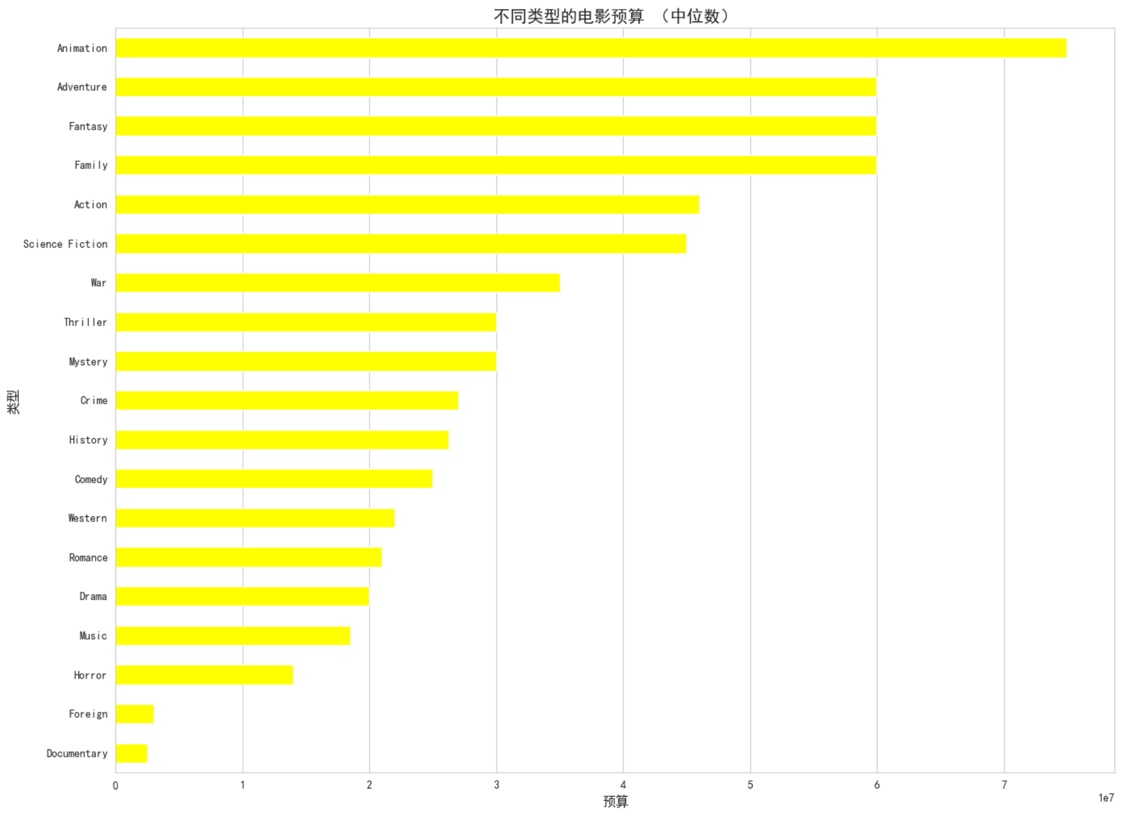
[16]:
```
#
#
all_genres = []
for genres in movies_df["genres_list"]:
    all_genres.extend(genres)

#
plt.figure(figsize=(16,12))
top_genres = pd.Series(all_genres).value_counts().head(15)
genres_bar = sns.barplot(x=top_genres.values , y=top_genres.index␣
  ↪,hue=top_genres.index , palette='viridis',legend=False)
plt.title("      top15 ",fontsize=15)
plt.xlabel("   ",fontsize=12)
plt.ylabel(" ",fontsize=12)
#
for i in genres_bar.patches:
    #
    width = i.get_width()
    #
    plt.text(width + 20, #
             i.get_y() + i.get_height()/2, #
             int(width), #
             ha='center', va='center')
plt.show()
```



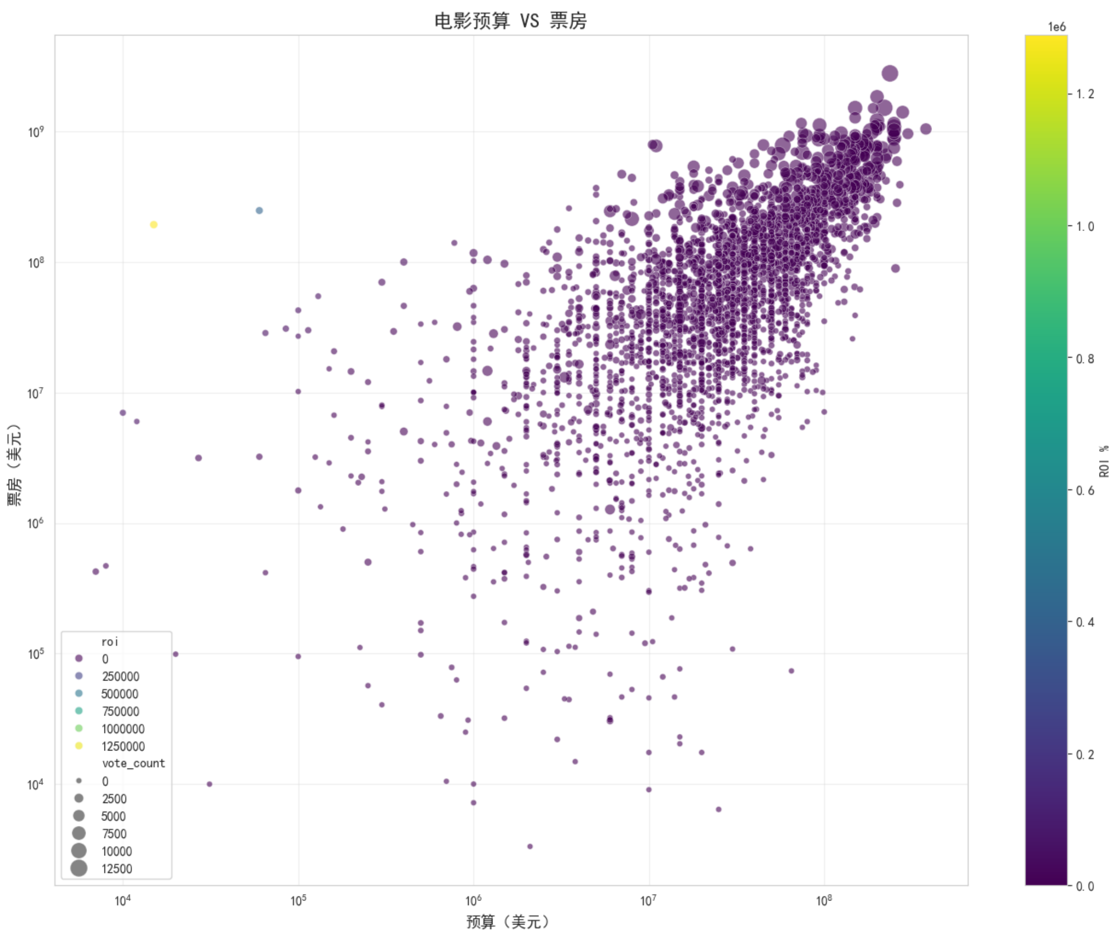
[17]:
```
#     roi           explode
genre_roi = movies_df.explode('genres_list').groupby('genres_list')['roi'].
  ↪median().sort_values(ascending=False).dropna().head(15)

plt.figure(figsize=(16,12))
genres_roi_bar = sns.barplot(x=genre_roi.values , y=genre_roi.index␣
  ↪,hue=genre_roi.index , palette='viridis',legend=False)
plt.title("       ROI    ",fontsize=15)
plt.xlabel("ROI % ",fontsize=12)
plt.ylabel("  ",fontsize=12)
plt.grid(axis='y')
#
for i in genres_roi_bar.patches:
    #
    width = i.get_width()
    #
    plt.text(width + 5, #
             i.get_y() + i.get_height()/2, #
             int(width), #
             ha='center', va='center')
plt.show()
```



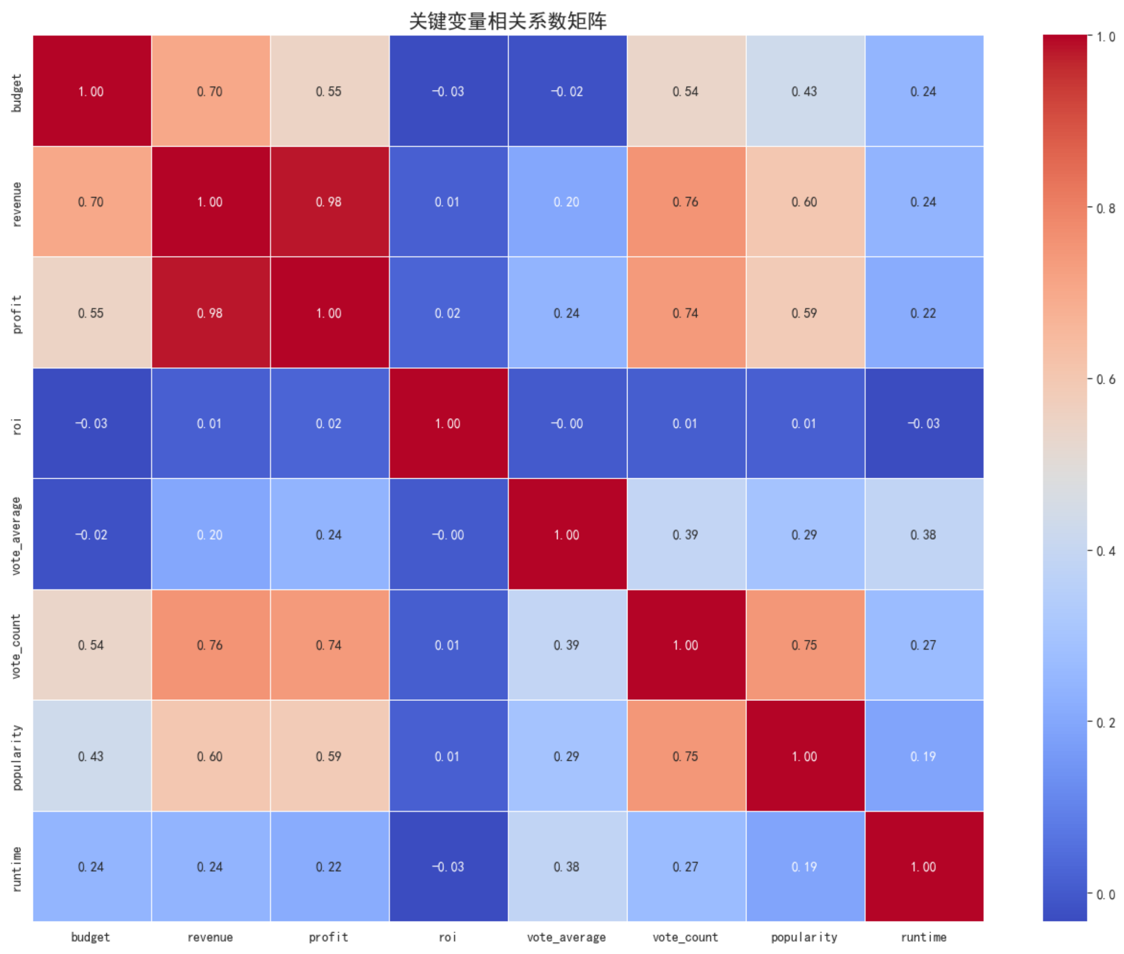
```
[18]: #     budget
genre_budget = movies_df.explode('genres_list').
  ↪groupby('genres_list')['budget'].median().sort_values().dropna()

plt.figure(figsize=(16,12))
genre_budget.plot(kind='barh',color='yellow')
plt.title("         ",fontsize=15)
plt.xlabel(" ",fontsize=12)
plt.ylabel(" ",fontsize=12)
plt.grid(axis='y')
plt.show()
```



```
[19]: #
plt.figure(figsize=(16,12))
ax = sns.scatterplot(x="budget",y="revenue",data=movies_df,alpha=0.
  ↪6,hue='roi',palette='viridis',size='vote_count',sizes=(20,200))
ax.set_title("   VS   ",fontsize=15)
ax.set_xlabel("    ",fontsize=12)
ax.set_ylabel("    ",fontsize=12)
ax.set_xscale('log')
ax.set_yscale('log')
ax.grid(True,alpha=0.3)
#
norm = plt.Normalize(movies_df['roi'].min(), movies_df['roi'].max())
sm = plt.cm.ScalarMappable(cmap="viridis", norm=norm)
sm.set_array([])
plt.colorbar(sm,label='ROI %',ax=ax) #  hue=roi
plt.show()
```

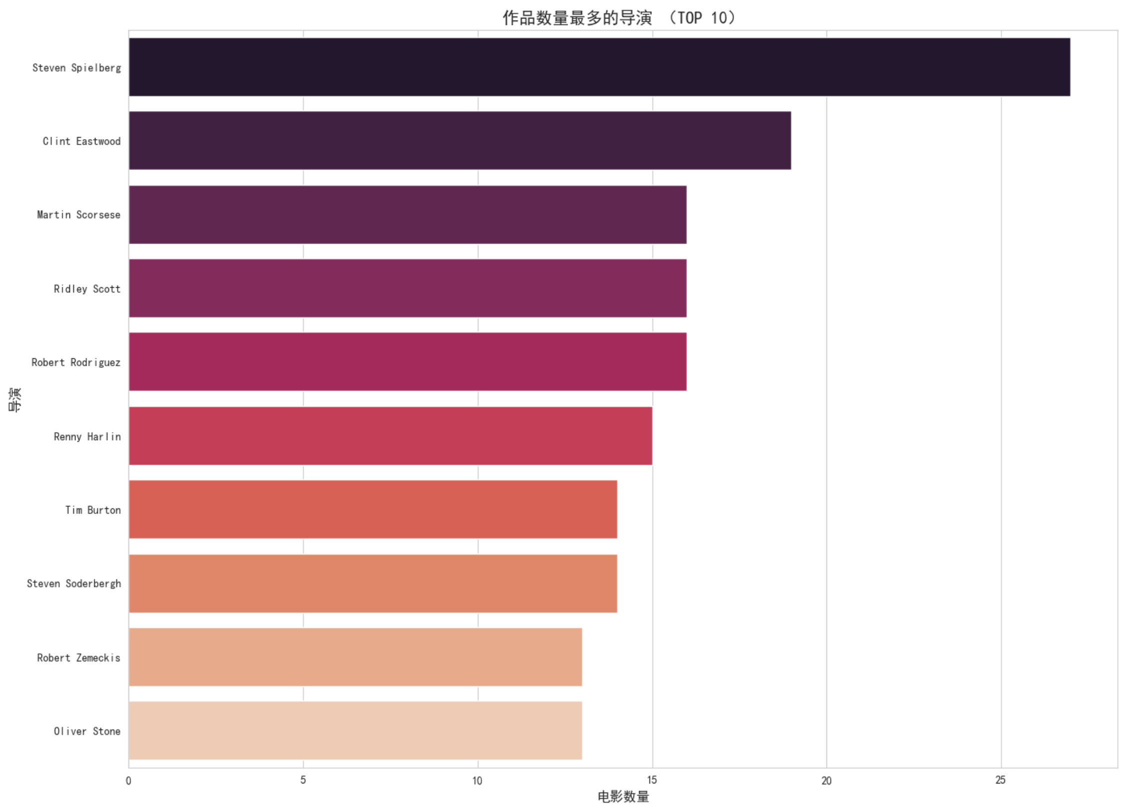


```
[21]: #
      corr_matrix =␣
        ↪movies_df[["budget","revenue","profit","roi","vote_average","vote_count","popularity","runt
        ↪corr()
      plt.figure(figsize=(16,12))
      sns.heatmap(corr_matrix,annot=True,cmap='coolwarm',fmt='.2f',linewidths=0.5)
      plt.title("      ",fontsize=15)
      plt.show()
```

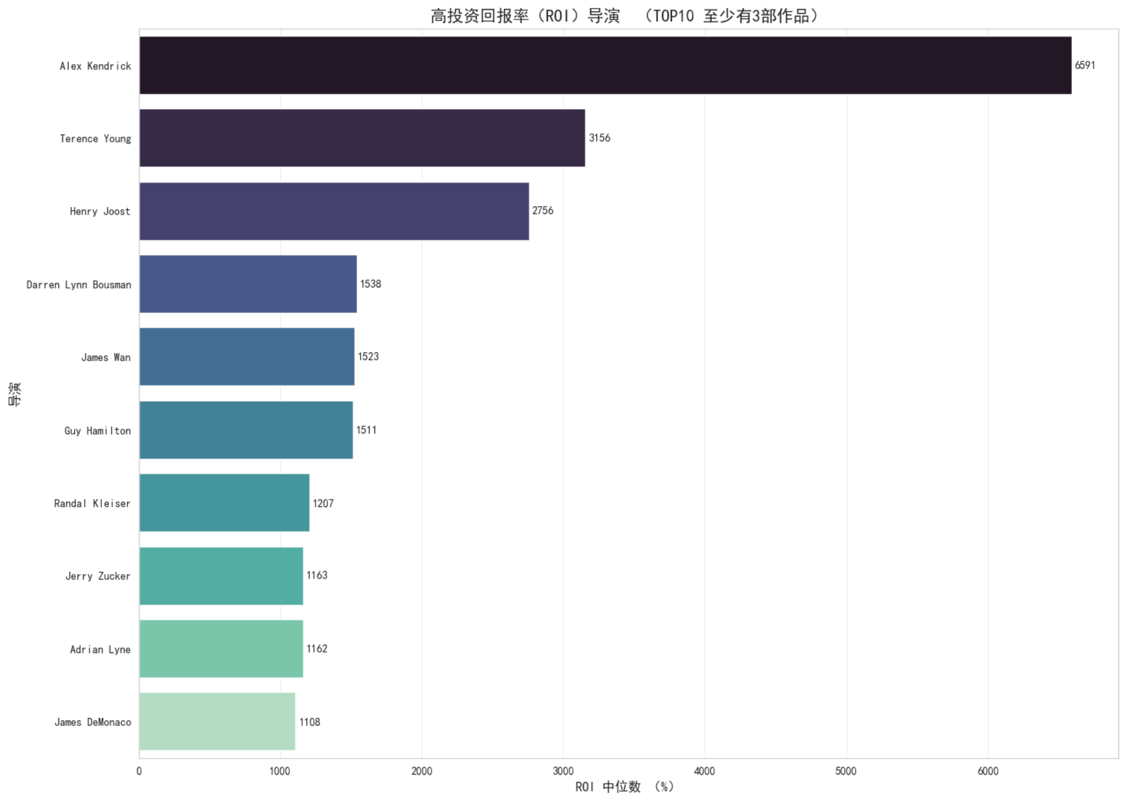


	budget	revenue	profit	roi	vote_average	vote_count	popularity	runtime
budget	1.00	0.70	0.55	-0.03	-0.02	0.54	0.43	0.24
revenue	0.70	1.00	0.98	0.01	0.20	0.76	0.60	0.24
profit	0.55	0.98	1.00	0.02	0.24	0.74	0.59	0.22
roi	-0.03	0.01	0.02	1.00	-0.00	0.01	0.01	-0.03
vote_average	-0.02	0.20	0.24	-0.00	1.00	0.39	0.29	0.38
vote_count	0.54	0.76	0.74	0.01	0.39	1.00	0.75	0.27
popularity	0.43	0.60	0.59	0.01	0.29	0.75	1.00	0.19
runtime	0.24	0.24	0.22	-0.03	0.38	0.27	0.19	1.00

[22]:
```
#    -
top_directors = movies_df["director"].value_counts().head(10)
plt.figure(figsize=(16,12))
sns.barplot(x=top_directors.values,y=top_directors.index,hue=top_directors.
↪index,palette='rocket')
plt.title("       TOP 10 ",fontsize=15)
plt.xlabel("   ",fontsize=12)
plt.ylabel(" ",fontsize=12)
plt.show()
```



[23]:
```
#     - ROI      3
directors_stats = movies_df.groupby("director").agg(
    movie_count=('id','count'),
    median_roi=('roi','median'),
    median_revenue=('revenue','median'),
    median_budget=('budget','median')
).reset_index()
#    3
directors_stats = directors_stats[directors_stats['movie_count']  >= 3]
#
top_directors_roi = directors_stats.sort_values('median_roi',ascending=False).
  ↪head(10)
plt.figure(figsize=(16,12))
director_roi_bar = sns.
  ↪barplot(data=top_directors_roi,x='median_roi',y='director',hue='director',palette='mako')
plt.title("    ROI     TOP10   3   ",fontsize=15)
plt.xlabel("ROI     %",fontsize=12)
plt.ylabel("  ",fontsize=12)
plt.grid(axis='x',alpha=0.3)
#
for i in director_roi_bar.patches:
    #
    width = i.get_width()
    #
    plt.text(width + 100, #
             i.get_y() + i.get_height()/2, #
             int(width), #
             ha='center', va='center')
plt.show()
print(top_directors_roi[['director','movie_count']])
```



```
                 director  movie_count
28          Alex Kendrick            3
1286        Terence Young            4
483           Henry Joost            3
263   Darren Lynn Bousman            3
533             James Wan            6
466          Guy Hamilton            5
1050       Randal Kleiser            3
573          Jerry Zucker            4
6             Adrian Lyne            4
520        James DeMonaco            3
```

1.5 4.

1.5.1

1.
    - 1990 2004-2014
    - (9.9%) (12.1%)
2.
    - **:** (44.2%) (34.7%) (29.4%) 28.8%
    - **ROI :**
        - (ROI 212%)
        - (ROI 187%)
        - (ROI 172%)
        - (ROI 170%)
    - **:**
        -
3.
    - (r=0.7)
    - ROI (r=-0.02)
    - 18.2% 5000 ROI
    - ” ” 4.2% <2000 >1
4.
    - **:** · (28) · (19)
    - **ROI :**
        - >=3
            * · (ROI 6591%) 3
            * · (ROI 3156%) 4
            * · (ROI 2756%) 3
        - >=10
            * · (ROI 480%) 12
            * · (ROI 325%) 27
            * · (ROI 321%) 12
5.
    - (r=0.2)
    - ( ) (r=0.76)

1.5.2

1. **:**
    - **:** / / / (ROI)
    - **:** / / (ROI)
    - **:** / (ROI)
2. **:**
    - : >1 → >4
    - : 3000 -1 → ( )
    - : <3000 → /
3. **:**

ROI ·
( ) ·

4. :
   - AI
   -
   - 30-50%
5. :
   - /
   - / /
   - / ( )

1.5.3

1. “ ” / ROI
2. “ ” ROI 100%+
3. “ ” (3000-8000) ” ”

” - - ” / ROI

1.6 5.

: - Python : Pandas, NumPy - : Matplotlib, Seaborn - : JSON - : ROI

: 1. (/ 1000) 2. JSON genres director 3. 4. / ROI 5.

: - TMDB ( ) - -

1.7

[24]:
```python
#  JSON
def parse_json_column(column):
    try:
        return ast.literal_eval(column)
    except (ValueError,SyntaxError):
        return [] #
```

[25]:
```python
#
def get_genres_list(genre_list):
    if isinstance(genre_list,list):
        return [genre["name"] for genre in genre_list]
    else:
        return []
```

[26]:
```
#  ROI
#    profit     ROI
movies_df["profit"] = movies_df["revenue"] - movies_df["budget"]
movies_df["roi"] = (movies_df["profit"]/movies_df["budget"])*100 #

#  RIO
movies_df.loc[movies_df["budget"] <= 0,'roi'] = np.nan
movies_df["roi"] = movies_df["roi"].replace([np.inf,-np.inf] , np.nan)
```