[hw2]

September 2, 2024

Download the dataset from: https://github.com/bellawillrise/Introduction-to-Numerical-Computing-in-Python/

Submit a pdf file, which is a rendered saved version of the jupyter notebook. Make sure to execute all the codes so the output can be viewed in the pdf.

Also include the link to the public github repository where the jupyter notebook for the assignment is uploaded.

Link to the github repository: https://github.com/dreeew05/CMSC-197/tree/main/Assignment%201

[49]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[50]:

```
# %matplotlib inline
```

[51]:

```
data = pd.read_csv("data/movie_metadata_cleaned.csv")
```

[52]:

```
data.head(2)
```

[52]:

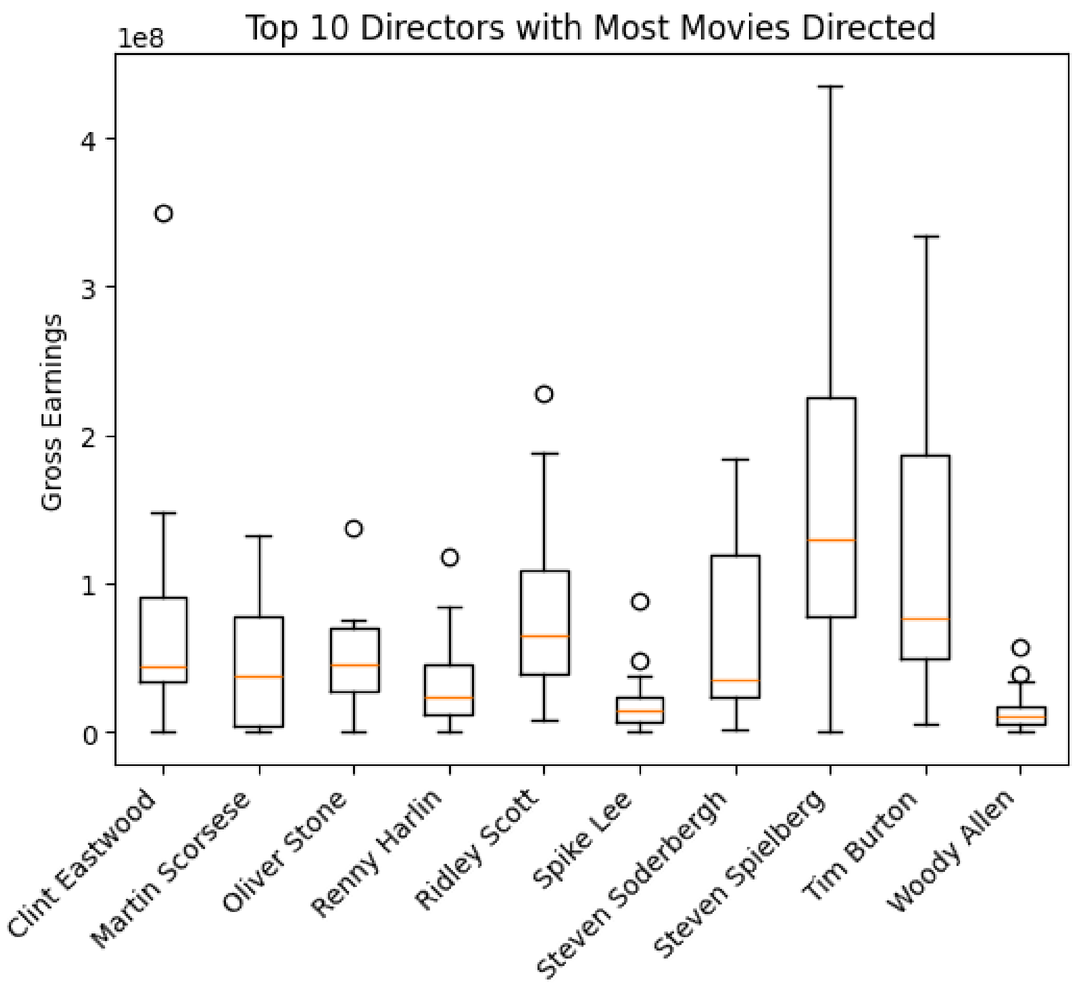
```
   Unnamed: 0                                  movie_title  color  \
0           0                                   b'Avatar'  Color
1           1  b"Pirates of the Caribbean: At World's End"  Color

    director_name  num_critic_for_reviews  duration  director_facebook_likes  \
0   James Cameron                   723.0     178.0                      0.0
1  Gore Verbinski                   302.0     169.0                    563.0

   actor_3_facebook_likes      actor_2_name  actor_1_facebook_likes  …  \
0                   855.0  Joel David Moore                  1000.0  …
1                  1000.0     Orlando Bloom                 40000.0  …

   num_user_for_reviews language country  content_rating       budget  \
0                3054.0  English     USA           PG-13  237000000.0
1                1238.0  English     USA           PG-13  300000000.0
```

```
  title_year  actor_2_facebook_likes imdb_score aspect_ratio  \
0     2009.0                   936.0        7.9         1.78
1     2007.0                  5000.0        7.1         2.35

   movie_facebook_likes
0               33000.0
1                   0.0

[2 rows x 29 columns]
```

0.1 Get the top 10 directors with most movies directed and use a boxplot for their gross earnings

[53]:

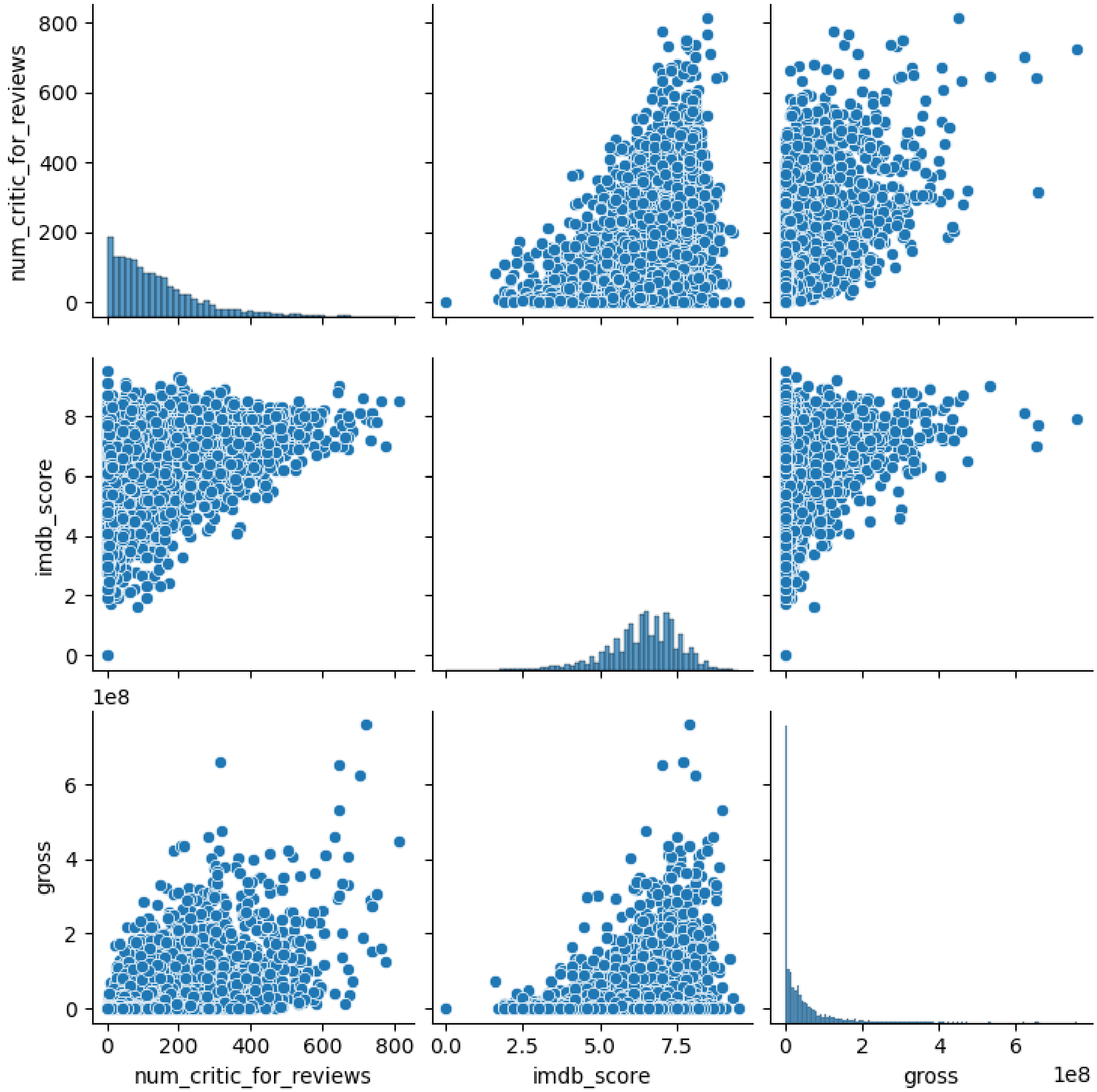
```
# Filter data [Remove directors that are named '0']
filtered_directors = data[data['director_name'] != '0']

# Group by director and get the top 10 directors with most movies directed
top_directors = filtered_directors.groupby('director_name').size().
  ↪sort_values(ascending=False).head(10)

# Filter the original dataframe to include only these top directors
top_directors_data = data[data['director_name'].isin(top_directors.index)]

# Group the data by director and get the gross earnings for each of their movie
gross_earnings_by_director = top_directors_data.
  ↪groupby('director_name')['gross'].apply(list)

# Create a boxplot for the top directors' gross earnings
plt.boxplot(gross_earnings_by_director, tick_labels=gross_earnings_by_director.
  ↪index)

plt.title('Top 10 Directors with Most Movies Directed')
plt.ylabel('Gross Earnings')
plt.xticks(rotation=45, ha="right")
plt.show()
```



0.2 Plot the following variables in one graph:

- num_critic_for_reviews
- IMDB score
- gross

[54]:
```
sns.pairplot(data[['num_critic_for_reviews', 'imdb_score', 'gross']])

plt.show()
```



0.3 Compute Sales (Gross - Budget), add it as another column

[55]:
```
data['computed_sales'] = data['gross'] - data['budget']
data.head()
```

[55]:
```
   Unnamed: 0                                        movie_title  color  \
0           0                                          b'Avatar'  Color
1           1          b"Pirates of the Caribbean: At World's End"  Color
2           2                                         b'Spectre'  Color
3           3                           b'The Dark Knight Rises'  Color
4           4  b'Star Wars: Episode VII - The Force Awakens   …      0
```

```
       director_name  num_critic_for_reviews  duration  \
0      James Cameron                   723.0     178.0
1     Gore Verbinski                   302.0     169.0
2         Sam Mendes                   602.0     148.0
3  Christopher Nolan                   813.0     164.0
4        Doug Walker                     0.0       0.0

   director_facebook_likes  actor_3_facebook_likes      actor_2_name  \
0                      0.0                   855.0  Joel David Moore
1                    563.0                  1000.0     Orlando Bloom
2                      0.0                   161.0      Rory Kinnear
3                  22000.0                 23000.0    Christian Bale
4                    131.0                     0.0        Rob Walker

   actor_1_facebook_likes  …  language country content_rating       budget  \
0                  1000.0  …   English     USA          PG-13  237000000.0
1                 40000.0  …   English     USA          PG-13  300000000.0
2                 11000.0  …   English      UK          PG-13  245000000.0
3                 27000.0  …   English     USA          PG-13  250000000.0
4                   131.0  …         0       0              0          0.0

   title_year actor_2_facebook_likes  imdb_score aspect_ratio  \
0      2009.0                  936.0         7.9         1.78
1      2007.0                 5000.0         7.1         2.35
2      2015.0                  393.0         6.8         2.35
3      2012.0                23000.0         8.5         2.35
4         0.0                   12.0         7.1         0.00

  movie_facebook_likes  computed_sales
0              33000.0     523505847.0
1                  0.0       9404152.0
2              85000.0     -44925825.0
3             164000.0     198130642.0
4                  0.0             0.0

[5 rows x 30 columns]
```

0.4 Which directors garnered the most total sales?

[56]:

```
# filtered_directors is a data frame used above
# Get the total sales using aggregation
director_agg = filtered_directors.groupby('director_name').agg(
    total_sales = ('gross', 'sum')
)
# Sort and print the directors with most total sales
director_agg.sort_values(by='total_sales', ascending=False).head()
```

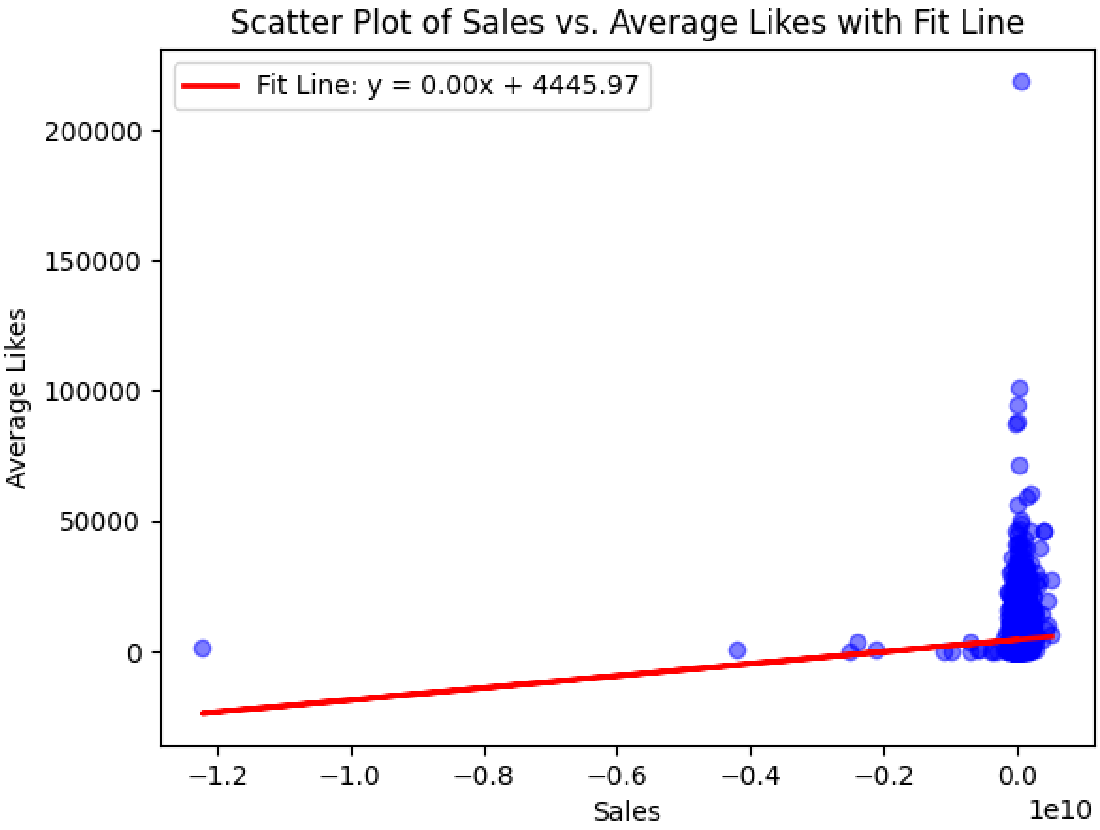
[56]:

```
                   total_sales
director_name
Steven Spielberg  4.114233e+09
Peter Jackson     2.592969e+09
Michael Bay       2.231243e+09
Tim Burton        2.071275e+09
Sam Raimi         2.049549e+09
```

0.5 Plot sales and average likes as a scatterplot. Fit it with a line.

[57]:

```
# Add average_likes as a new column
total_likes = [
    'movie_facebook_likes',
    'actor_1_facebook_likes',
    'actor_2_facebook_likes',
    'actor_3_facebook_likes',
    'director_facebook_likes',
    'cast_total_facebook_likes'
]
data['average_likes'] = data[total_likes].mean(axis=1)

plt.scatter(data['computed_sales'], data['average_likes'], color='blue',␣
 ↪alpha=0.5)

#Linear Fit
slope, intercept = np.polyfit(data['computed_sales'], data['average_likes'], 1)
fit_line = slope * data['computed_sales'] + intercept

# Plot the fit line
plt.plot(data['computed_sales'], fit_line, color='red', linewidth=2,␣
 ↪label=f'Fit Line: y = {slope:.2f}x + {intercept:.2f}')

plt.title('Scatter Plot of Sales vs. Average Likes with Fit Line')
plt.xlabel('Sales')
plt.ylabel('Average Likes')
plt.legend()
plt.show()
```



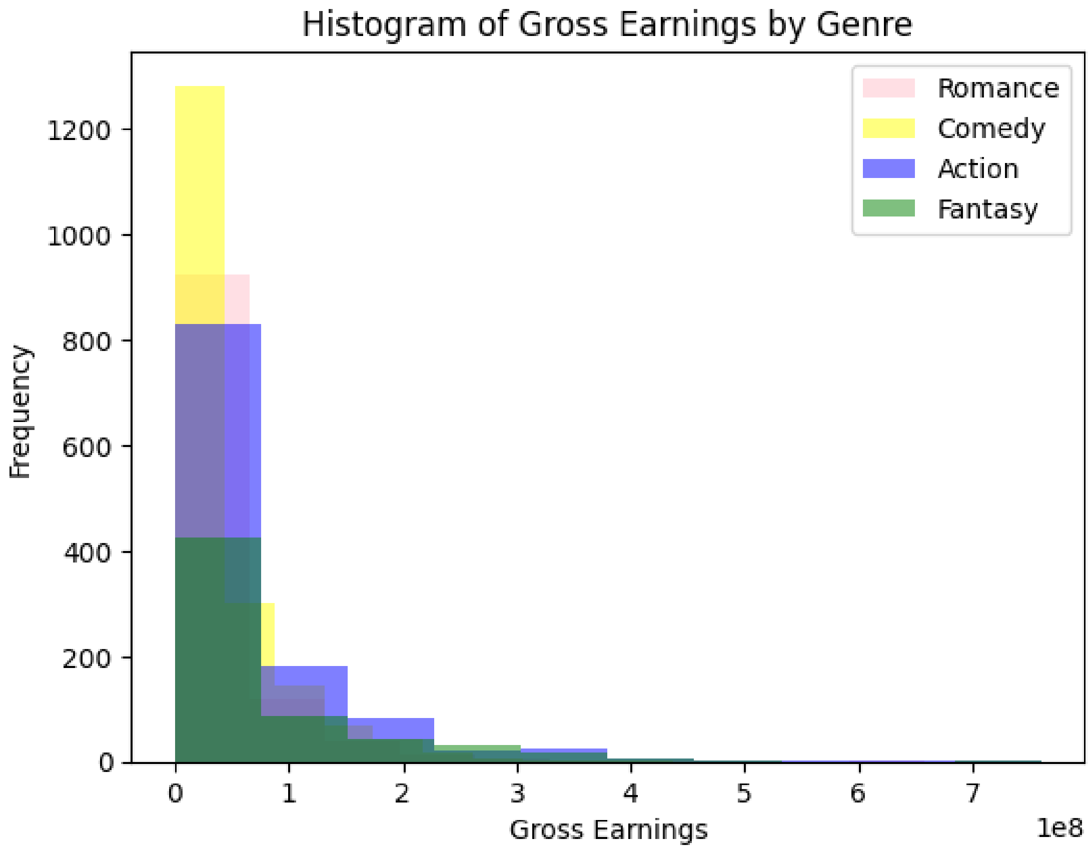
0.6 Which of these genres are the most profitable? Plot their sales using different histograms, superimposed in the same axis.

- Romance
- Comedy
- Action
- Fantasy

[58]:

```
# Filter data by the specific genres
romance = data[data['genres'].str.contains('Romance', na=False)]
comedy = data[data['genres'].str.contains('Comedy', na=False)]
action = data[data['genres'].str.contains('Action', na=False)]
fantasy = data[data['genres'].str.contains('Fantasy', na=False)]

# Romance
plt.hist(romance['gross'].dropna(), alpha=0.5, label='Romance', color='pink')

# Comedy
plt.hist(comedy['gross'].dropna(), alpha=0.5, label='Comedy', color='yellow')
```

```
# Action
plt.hist(action['gross'].dropna(), alpha=0.5, label='Action', color='blue')

# Fantasy
plt.hist(fantasy['gross'].dropna(), alpha=0.5, label='Fantasy', color='green')

plt.xlabel('Gross Earnings')
plt.ylabel('Frequency')
plt.title('Histogram of Gross Earnings by Genre')
plt.legend()
plt.show()
```



0.7 For each of movie, compute average likes of the three actors and store it as a new variable

Read up on the mean function.

Store it as a new column, average_actor_likes.

0.8 Copying the whole dataframe

[59]:
```
df = data.copy()
df.head()
```

[59]:
```
   Unnamed: 0                                         movie_title  color  \
0           0                                           b'Avatar'  Color
1           1          b"Pirates of the Caribbean: At World's End"  Color
2           2                                          b'Spectre'  Color
3           3                            b'The Dark Knight Rises'  Color
4           4  b'Star Wars: Episode VII - The Force Awakens  …        0

      director_name  num_critic_for_reviews  duration  \
0      James Cameron                   723.0     178.0
1     Gore Verbinski                   302.0     169.0
2         Sam Mendes                   602.0     148.0
3  Christopher Nolan                   813.0     164.0
4        Doug Walker                     0.0       0.0

   director_facebook_likes  actor_3_facebook_likes      actor_2_name  \
0                      0.0                   855.0  Joel David Moore
1                    563.0                  1000.0     Orlando Bloom
2                      0.0                   161.0      Rory Kinnear
3                  22000.0                 23000.0    Christian Bale
4                    131.0                     0.0        Rob Walker

   actor_1_facebook_likes  …  country content_rating       budget  \
0                  1000.0  …      USA          PG-13  237000000.0
1                 40000.0  …      USA          PG-13  300000000.0
2                 11000.0  …       UK          PG-13  245000000.0
3                 27000.0  …      USA          PG-13  250000000.0
4                   131.0  …        0              0          0.0

   title_year  actor_2_facebook_likes imdb_score  aspect_ratio  \
0      2009.0                   936.0        7.9          1.78
1      2007.0                  5000.0        7.1          2.35
2      2015.0                   393.0        6.8          2.35
3      2012.0                 23000.0        8.5          2.35
4         0.0                    12.0        7.1          0.00

  movie_facebook_likes computed_sales  average_likes
0              33000.0    523505847.0    6770.833333
1                  0.0      9404152.0   15818.833333
2              85000.0    -44925825.0   18042.333333
3             164000.0    198130642.0   60959.833333
4                  0.0            0.0      69.500000
```

```
[5 rows x 31 columns]
```

0.9 Min-Max Normalization

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges.

The min-max approach (often called normalization) rescales the feature to a hard and fast range of [0,1] by subtracting the minimum value of the feature then dividing by the range. We can apply the min-max scaling in Pandas using the .min() and .max() methods.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

0.9.1 Normalize each numeric column (those that have types integer or float) of the copied dataframe (df)

[60]:
```python
numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns
df[numeric_columns] = df[numeric_columns].apply(lambda x: (x - x.min()) / (x.max() - x.min()))
df.head()
```

[60]:
```
   Unnamed: 0                                         movie_title  color  \
0    0.000000                                           b'Avatar'  Color
1    0.000198        b"Pirates of the Caribbean: At World's End"  Color
2    0.000397                                          b'Spectre'  Color
3    0.000595                             b'The Dark Knight Rises'  Color
4    0.000793  b'Star Wars: Episode VII - The Force Awakens   …      0

      director_name  num_critic_for_reviews  duration  \
0     James Cameron                0.889299  0.941799
1    Gore Verbinski                0.371464  0.894180
2        Sam Mendes                0.740467  0.783069
3 Christopher Nolan                1.000000  0.867725
4       Doug Walker                0.000000  0.000000

   director_facebook_likes  actor_3_facebook_likes      actor_2_name  \
0                 0.000000                0.037174  Joel David Moore
1                 0.024478                0.043478     Orlando Bloom
2                 0.000000                0.007000      Rory Kinnear
3                 0.956522                1.000000    Christian Bale
4                 0.005696                0.000000        Rob Walker

   actor_1_facebook_likes  …  country content_rating    budget  title_year  \
0                0.001563  …      USA          PG-13  0.019402    0.996528
1                0.062500  …      USA          PG-13  0.024559    0.995536
```

```
2                0.017188  …       UK          PG-13  0.020056    0.999504
3                0.042188  …      USA          PG-13  0.020466    0.998016
4                0.000205  …        0              0  0.000000    0.000000

   actor_2_facebook_likes imdb_score  aspect_ratio movie_facebook_likes  \
0                0.006832   0.831579      0.111250             0.094556
1                0.036496   0.747368      0.146875             0.000000
2                0.002869   0.715789      0.146875             0.243553
3                0.167883   0.894737      0.146875             0.469914
4                0.000088   0.747368      0.000000             0.000000

  computed_sales  average_likes
0       1.000000       0.030964
1       0.959637       0.072341
2       0.955371       0.082510
3       0.974454       0.278777
4       0.958898       0.000318

[5 rows x 31 columns]
```