Cracking the Cinematic Code: Data-Driven Insights for Successful Movies

Project 1
Team 3

Daniela Montiel
Esaú Cervantes
Jessica Montoya

Exploratory Data Analysis in Entertainment | Movies

Industry Importance

The entertainment industry is not an ethereal element that only enthusiasts or people in the field have a stake in. Almost everyone has a form of entertainment that they enjoy, from music, television, theater or the amazing field of movies.

According to the Motion Picture Association of America's (MPAA) Theatrical Market Statistics Report for 2021, the U.S. and Canadian box office came in at $4.5 billion in 2021, up 105% from 2020 due to theater re-openings following the COVID-19 pandemic lockdowns, but remained well below pre-pandemic levels (1).

Datasets used

Movies CSV | **Kaggle**

OMDb API | **OMDb**

Project Objective

This project aims to identify the key factors that contribute to a movie's success by analyzing a comprehensive database of films. It seeks to answer questions about the influence of factors such as genre, actors, director and budget on a movie's performance, and ultimately define the characteristics of a "perfect" movie

Pepperdine, Business. "A Changing Tide How the Entertainment Industry Is Making Waves in the Modern Era." A Changing Tide – How the Entertainment Industry is Making Waves in the Modern Era | Pepperdine Graziadio Business School, September 26, 2022. https://bschool.pepperdine.edu/blog/posts/entertainment-industry-trends-2022.htm#:~:text=The%20entertainment%20industry%20is%20also,action%20and%20driving%20positive%20change.

Overview

Initial data frame found contains Movie information from 1936 to 2019.

Data frame was connected with IMDB API to add the following movie characteristics: Director, Writer, Actors, Language, Country, Awards, Poster, Metascore, IMDB Rating, IMDB Votes, BoxOffice. The outcome of this merge provide a more robust data to analyze.

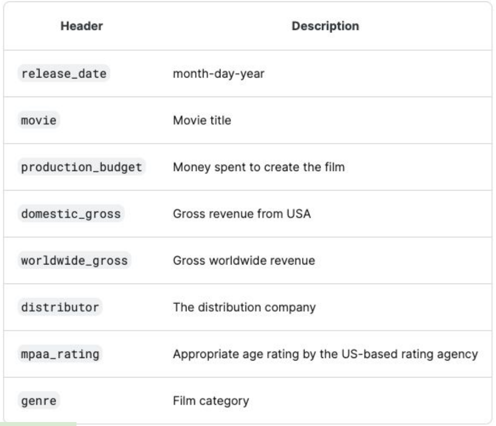
Total range of rows: 3400
Selected year range for analysis - **2000-2019**
Total range of rows per selected year range - **2467**

Questions to be answered:

- How is the behavior of the industry (revenue, distribution) over the past 10 years?
- What genre is the most profitable genre in the past 10 years?*
- Which actor generates the highest profit on average (per genre)?

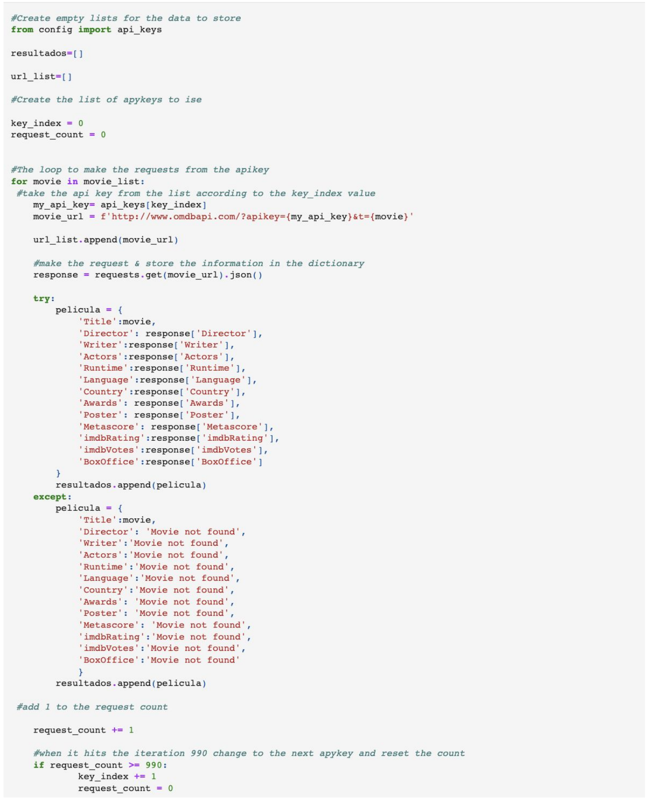
Connecting the .csv file with the OMDb API

Original .csv file contained a limited variety of columns, connecting this source with the OMDb API allowed the team to retrieve the following: **Director, Writer, Actors, Runtime, Language, Country, Awards, Poster, Metascore, imdb Rating, imdb Votes and Box Office**. As a result, it was possible to create a DataFrame with more robust information the team can explore and use in the analysis



Header	Description
`release_date`	month-day-year
`movie`	Movie title
`production_budget`	Money spent to create the film
`domestic_gross`	Gross revenue from USA
`worldwide_gross`	Gross worldwide revenue
`distributor`	The distribution company
`mpaa_rating`	Appropriate age rating by the US-based rating agency
`genre`	Film category

Original available data



```python
#Create empty lists for the data to store
from config import api_keys

resultados=[]

url_list=[]

#Create the list of apykeys to ise

key_index = 0
request_count = 0


#The loop to make the requests from the apikey
for movie in movie_list:
 #take the api key from the list according to the key_index value
    my_api_key= api_keys[key_index]
    movie_url = f'http://www.omdbapi.com/?apikey={my_api_key}&t={movie}'

    url_list.append(movie_url)

    #make the request & store the information in the dictionary
    response = requests.get(movie_url).json()

    try:
        pelicula = {
            'Title':movie,
            'Director': response['Director'],
            'Writer':response['Writer'],
            'Actors':response['Actors'],
            'Runtime':response['Runtime'],
            'Language':response['Language'],
            'Country':response['Country'],
            'Awards': response['Awards'],
            'Poster': response['Poster'],
            'Metascore': response['Metascore'],
            'imdbRating':response['imdbRating'],
            'imdbVotes':response['imdbVotes'],
            'BoxOffice':response['BoxOffice']
        }
        resultados.append(pelicula)
    except:
        pelicula = {
            'Title':movie,
            'Director': 'Movie not found',
            'Writer':'Movie not found',
            'Actors':'Movie not found',
            'Runtime':'Movie not found',
            'Language':'Movie not found',
            'Country':'Movie not found',
            'Awards': 'Movie not found',
            'Poster': 'Movie not found',
            'Metascore': 'Movie not found',
            'imdbRating':'Movie not found',
            'imdbVotes':'Movie not found',
            'BoxOffice':'Movie not found'
            }
        resultados.append(pelicula)

 #add 1 to the request count

    request_count += 1

    #when it hits the iteration 990 change to the next apykey and reset the count
    if request_count >= 990:
            key_index += 1
            request_count = 0
```

Data cleaning

For easy reading and smooth handling the data cleaning process drop determined columns such as ‘Unnamed: 0.1’ and ‘Unnamed: 0’.

```
movies_df_complete['Title'].value_counts()
movies_df_complete = movies_df_complete.drop_duplicates(subset='Title',keep='first')
movies_df_complete = movies_df_complete.drop(columns=['Unnamed: 0.1','Unnamed: 0'])
movies_df_complete['Title'].value_counts()
movies_df_complete.head(2)
```

To make sure the Research questions can be answered, 4 additional columns were created based on the information of the DataFrame: ‘Revenue’, ‘%_Revenue’, ‘release_date’ and ‘Year’

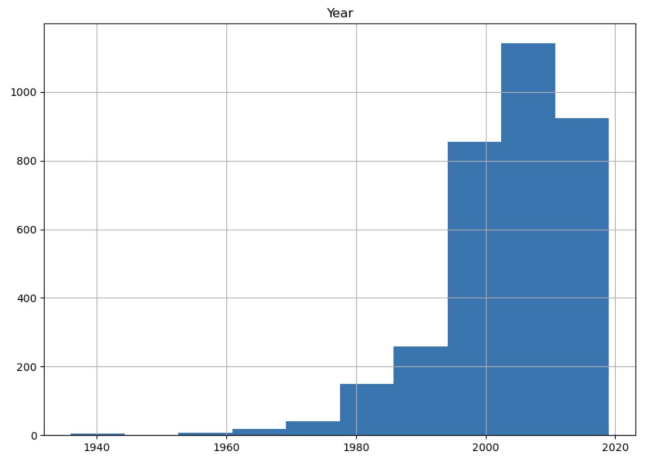
```
movies_df_complete['Revenue'] = movies_df_complete['worldwide_gross']-movies_df_complete['production_budget']

movies_df_complete['%_Revenue'] = (movies_df_complete['Revenue']/movies_df_complete['production_budget'])*100

movies_df_complete['release_date'] = pd.to_datetime(movies_df_complete['release_date'])

movies_df_complete['Year'] = movies_df_complete['release_date'].dt.year

movies_df_complete.head(2)
```

Movie industry over the years | Count of movies

Analysis per year was conducted to identify any trends and to pinpoint which timeframe will be analyzed.

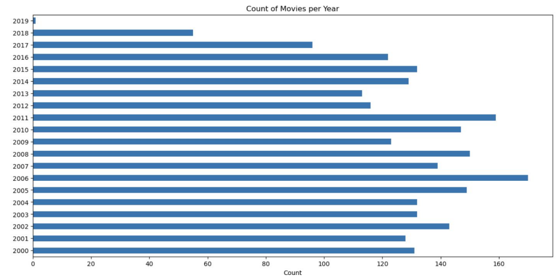
The amount of movies available for analysis are low from 1940 until the end of the century; starting 2000 the amount of movies went up; nevertheless, in order to perform a valuable analysis and with the ultimate purpose of providing relevant insights we choose to consider 2000 to 2019 for our analysis

```python
# Histogram of the all the movies in the data frame by year

movies_df_complete.hist(column='Year',figsize=(10,7))
plt.savefig('./figures/hist_all_movies_by_year.png')
plt.show()
```



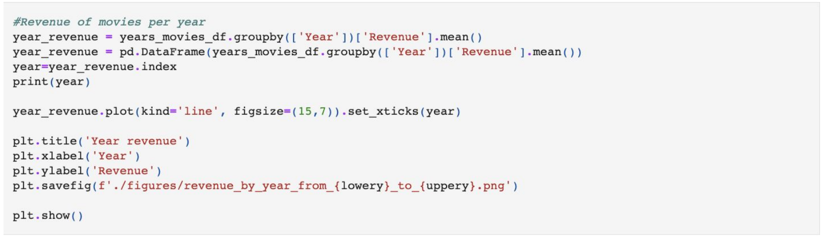
```python
#Define the time frame for the analysis
lowery=2000
uppery=2019

#Filter the Data Frame with the selected years
years_movies_df = movies_df_complete[movies_df_complete['Year'].between(lowery,uppery)]
```

```python
# Generate plot with the count of movies per year for the selected time frame

year_count_filtered = years_movies_df[['Year']].value_counts().reset_index().sort_values(by='Year')

year_count_filtered.plot(kind='barh', x='Year', figsize=(15,7), legend=False)
plt.xlabel('Count')
plt.title("Count of Movies per Year")
plt.savefig(f'./figures/count_movies_per_year_from_{lowery}_to_{uppery}.png')

plt.show()
```

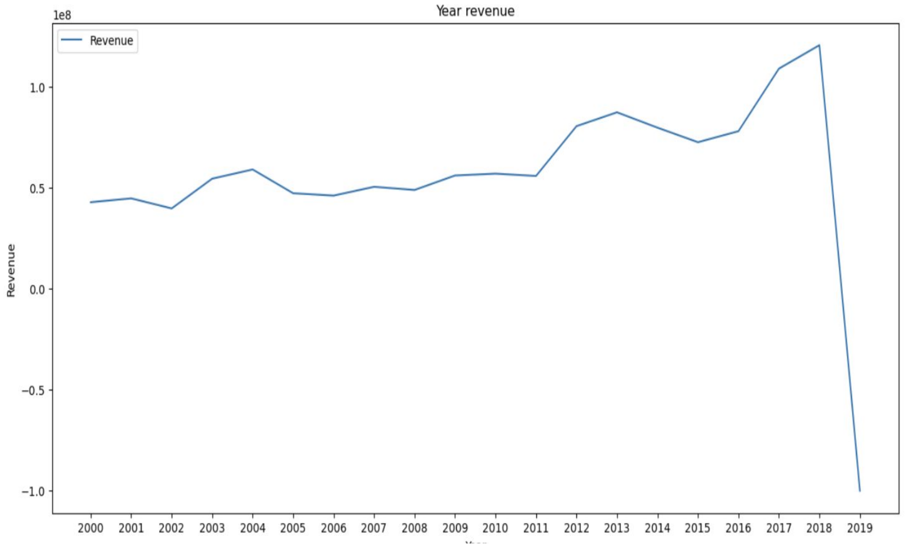


Sum of revenue per year

General data showed revenue double over the timeframe selected.



```
#Revenue of movies per year
year_revenue = years_movies_df.groupby(['Year'])['Revenue'].mean()
year_revenue = pd.DataFrame(years_movies_df.groupby(['Year'])['Revenue'].mean())
year=year_revenue.index
print(year)

year_revenue.plot(kind='line', figsize=(15,7)).set_xticks(year)

plt.title('Year revenue')
plt.xlabel('Year')
plt.ylabel('Revenue')
plt.savefig(f'./figures/revenue_by_year_from_{lowery}_to_{uppery}.png')

plt.show()
```



Revenue per distributor | Top 5 distributors

To ensure we can provide a list of reliable Distributors based on the financial success of they productions we create the Top 5 list based in Revenue

		Revenue
distributor	**Year**	
20th Century Fox	**2000**	842424835
	2001	292116692
	2002	765970108
	2003	565714264
	2004	1023380586
...	**...**	...
XLrator Media	**2014**	-4072926
Yari Film Group Rel...	**2006**	67292062
	2007	-9739445
Yash Raj Films	**2018**	-1401194
Zeitgeist	**2003**	-326515

	Revenue
distributor	
20th Century Fox	25710068126
Universal	23341222260
Warner Bros.	22630862445
Sony Pictures	17597827520
Paramount Pictures	16645716358

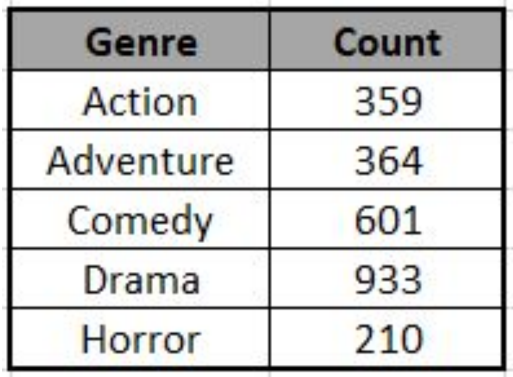
Analysis by Genre

We ventured into a thorough examination of the multifaceted world of movie genres and their influence on a film's success. This journey revealed intriguing patterns in both production and revenue across various genres.

Key variables analyzed

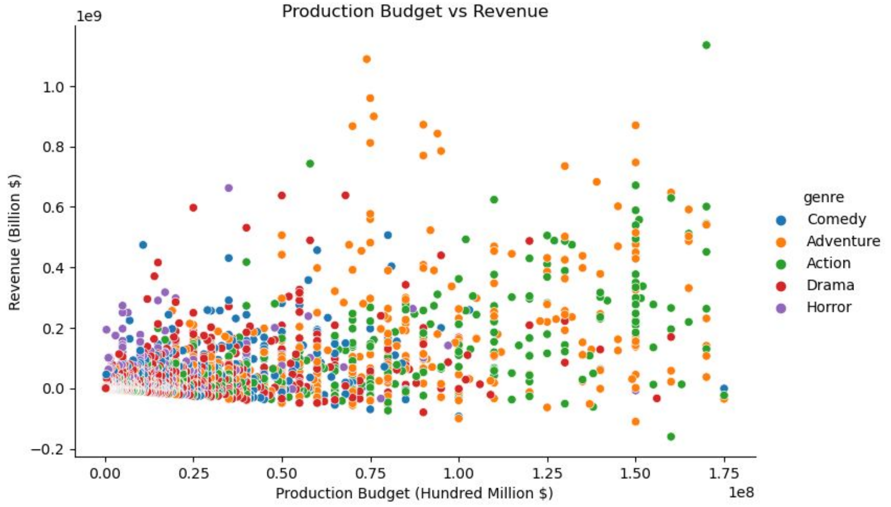
- Production Budget
- Revenue
- % Revenue = Revenue/Budget

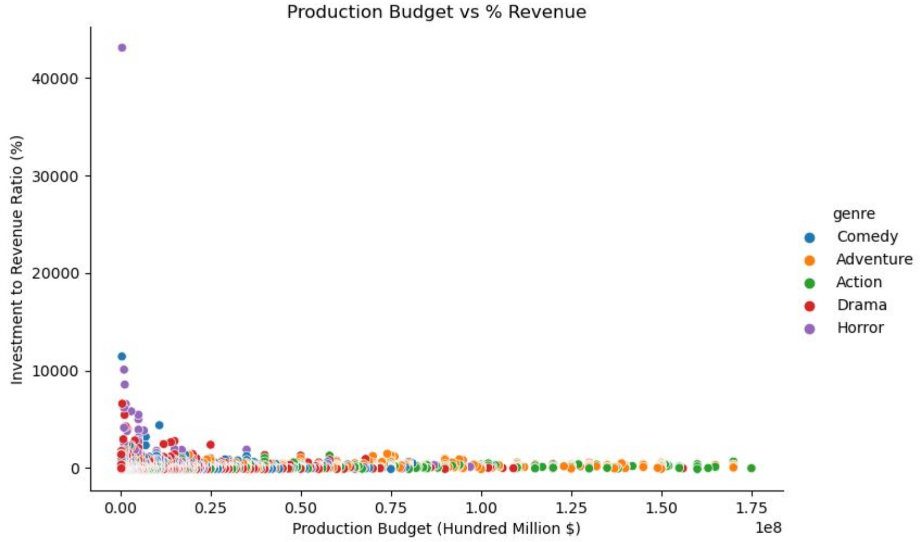
Grouped by Genre



Genre	Count
Action	359
Adventure	364
Comedy	601
Drama	933
Horror	210

Production budget vs Revenue | Correlation?



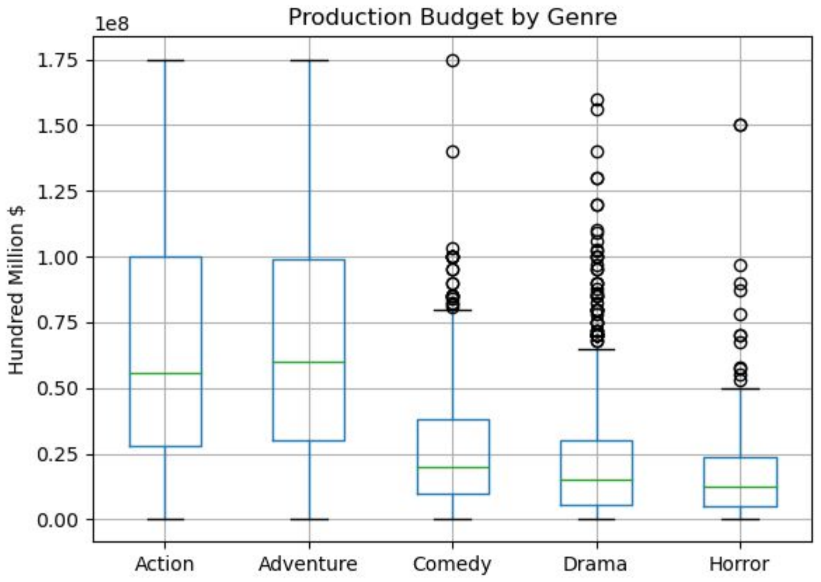


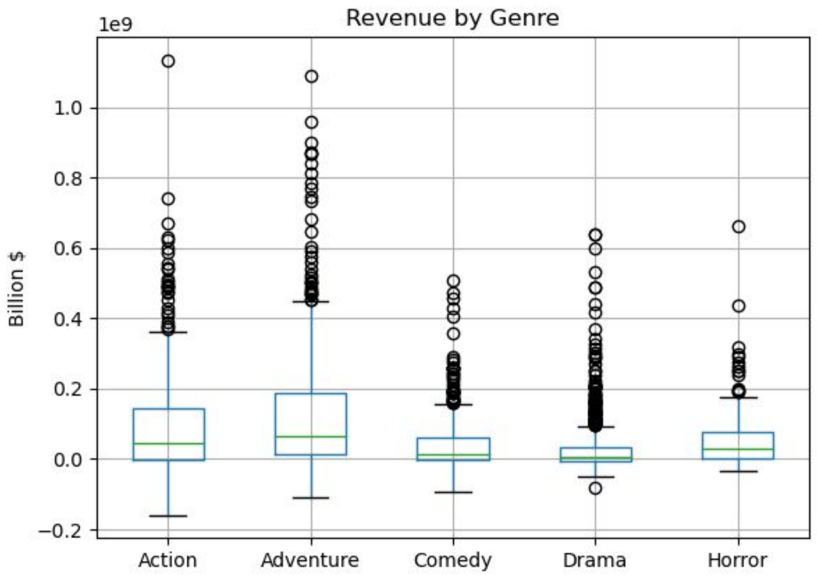
There is no substantial correlation between the budget allocated to a movie and the revenue it generates. This particular finding emphasizes a crucial point—the mere presence of a substantial financial investment does not guarantee a commensurate level of financial success.

Insights

- Although there is no correlation between the production budget and the revenue, by analyzing the behavior of the key variables by genre, we can glimpse different risk scenarios for the Return of Investment in each genre and have an estimation on the potential revenue.

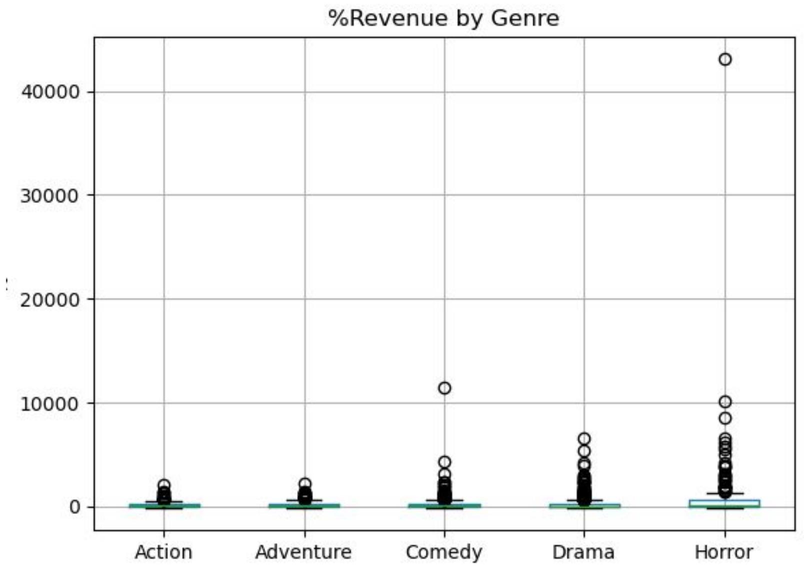
Production budget and Revenue | GroupBy Boxplots

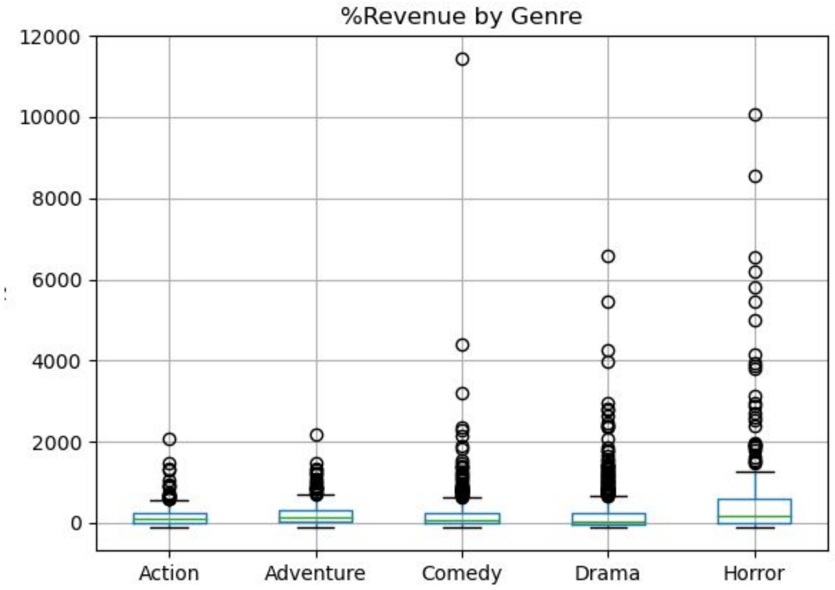




Significant disparities in the financial dynamics of different genres were observed. Adventure films, for instance, emerged as a favorable one in terms of revenue, boasting the highest mean returns.

Revenue by Genre | GroupBy Boxplots





Specific movie genres exert a notable impact on a film's financial success. However, it's crucial to acknowledge that the choice of genre is a multifaceted decision that encompasses not only the potential revenue but also the associated production costs, the inherent variability of outcomes, and the financial risks tied to each genre.

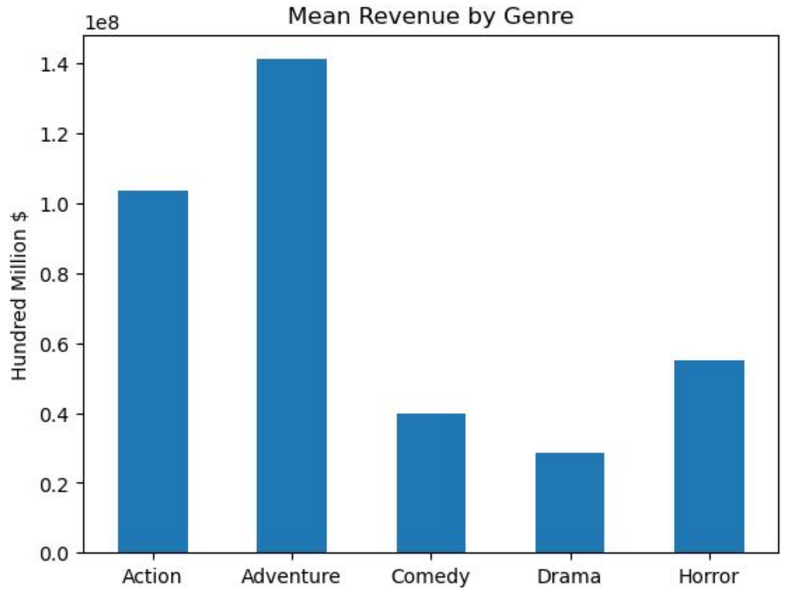
Findings

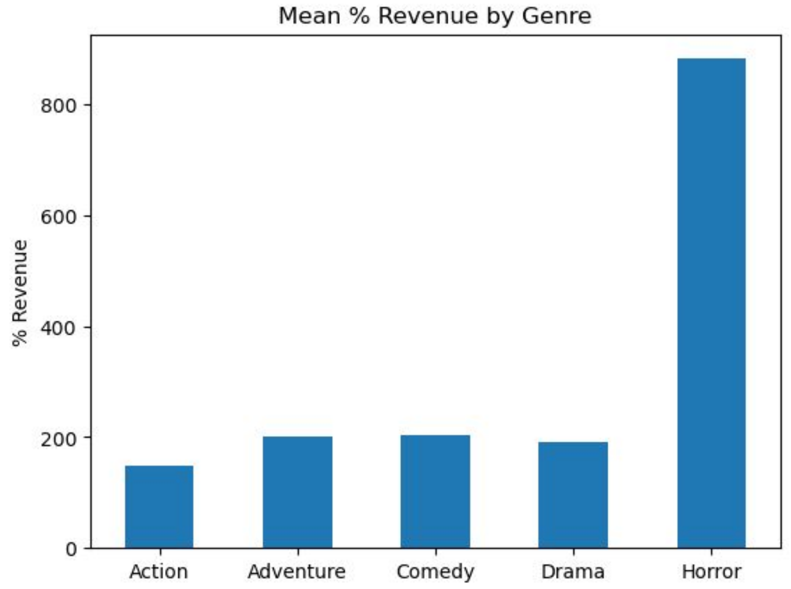
- The behavior of the key variables **does not follow a normal distribution.**

- There is independence between the genres which means each genre “has it own rules”.

- The industry has a considerable amount of outliers (blockbusters), this element hampers to predict with certainty the success of a film.

Mean Revenue vs % Revenue

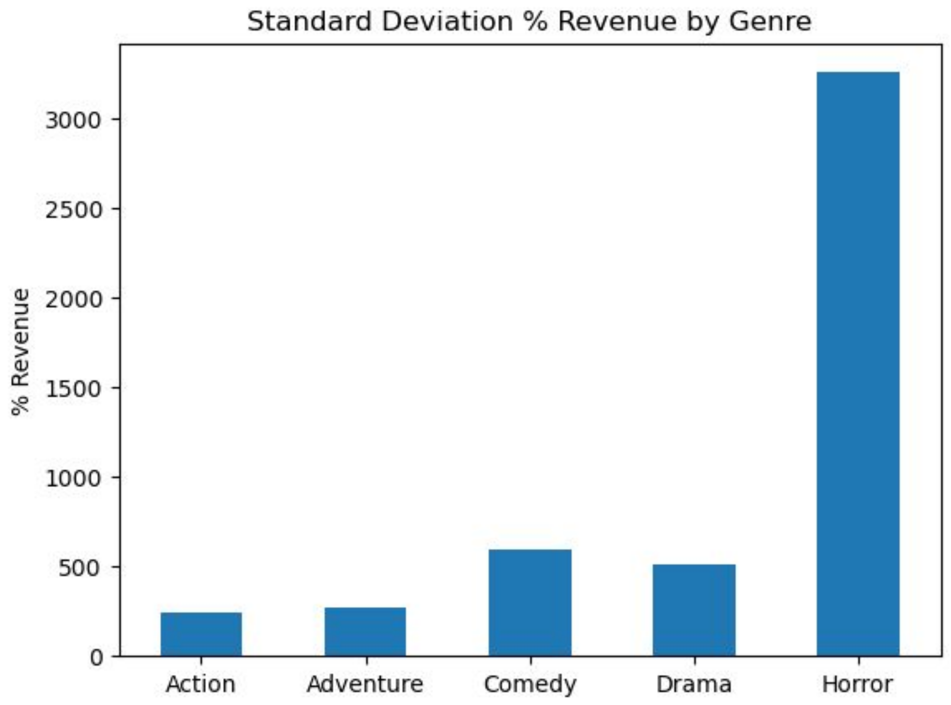




By scrutinizing the percentage revenue—calculated as the total revenue divided by the production budget—we uncovered an interesting revelation. The horror genre appeared as the leader in terms of profitability.

Variability | Std

A high standard deviation in Horror, means that the data points are widely scattered from the average. Within the Horror genre, we noticed the presence of numerous outliers that significantly elevated the standard deviation.



Insights

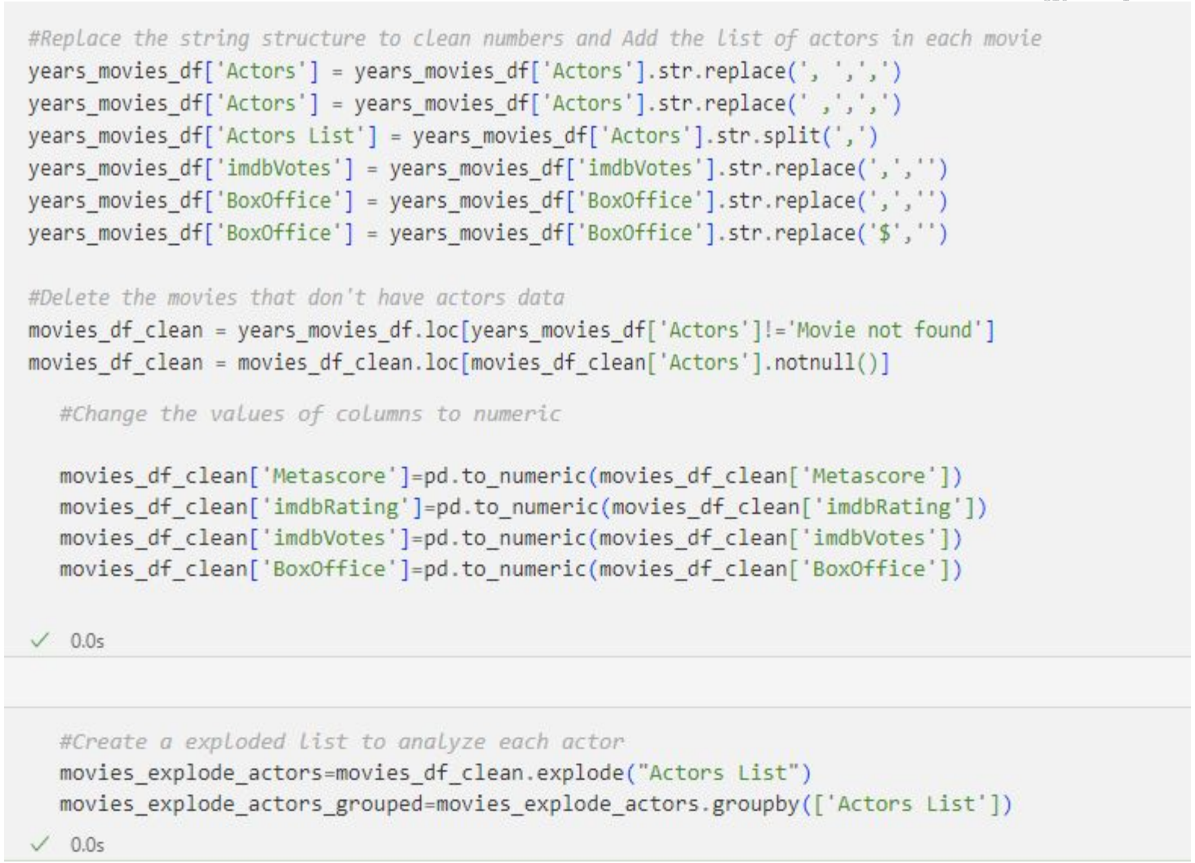
- The Adventure genre generates the biggest revenues but also has the biggest production cost.
- The Horror genre has the best investment to revenue ratio but it also has the biggest variability, making it a risky investment.
- There are other variables that should be considered for a professional proposal such as:
  - general economics
  - politics
  - theme relevance
  - director
  - writer
  - actors...

The actors | Data preparation

The influence of actor selection and their fame influence on a movie's success is a multifaceted topic that requires a comprehensive analysis of various metrics.

While it is possible to pinpoint top stars who tend to contribute to success in the film industry, it is imperative to consider a range of factors.

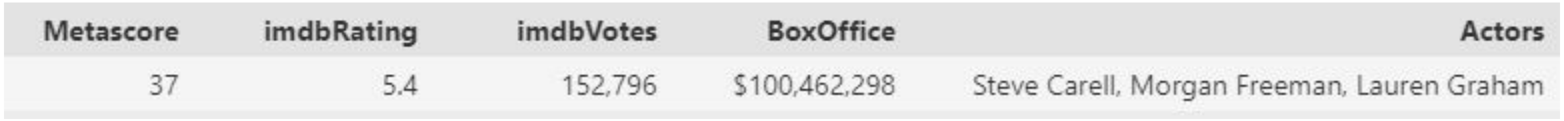
Let's start with data preparation...



```
#Replace the string structure to clean numbers and Add the list of actors in each movie
years_movies_df['Actors'] = years_movies_df['Actors'].str.replace(', ',',')
years_movies_df['Actors'] = years_movies_df['Actors'].str.replace(' ,',',')
years_movies_df['Actors List'] = years_movies_df['Actors'].str.split(',')
years_movies_df['imdbVotes'] = years_movies_df['imdbVotes'].str.replace(',','')
years_movies_df['BoxOffice'] = years_movies_df['BoxOffice'].str.replace(',','')
years_movies_df['BoxOffice'] = years_movies_df['BoxOffice'].str.replace('$','')

#Delete the movies that don't have actors data
movies_df_clean = years_movies_df.loc[years_movies_df['Actors']!='Movie not found']
movies_df_clean = movies_df_clean.loc[movies_df_clean['Actors'].notnull()]
  #Change the values of columns to numeric

  movies_df_clean['Metascore']=pd.to_numeric(movies_df_clean['Metascore'])
  movies_df_clean['imdbRating']=pd.to_numeric(movies_df_clean['imdbRating'])
  movies_df_clean['imdbVotes']=pd.to_numeric(movies_df_clean['imdbVotes'])
  movies_df_clean['BoxOffice']=pd.to_numeric(movies_df_clean['BoxOffice'])

✓ 0.0s
```

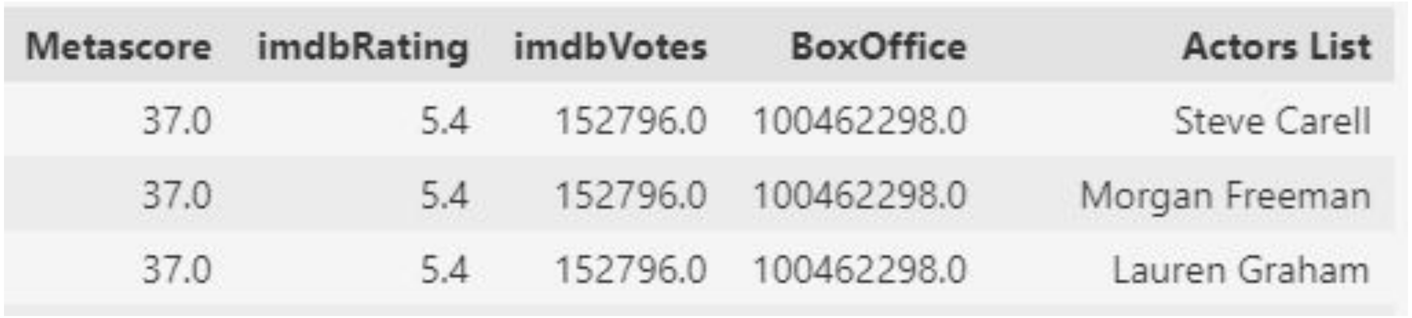
```
  #Create a exploded list to analyze each actor
  movies_explode_actors=movies_df_clean.explode("Actors List")
  movies_explode_actors_grouped=movies_explode_actors.groupby(['Actors List'])
✓ 0.0s
```

Initial Data Display:



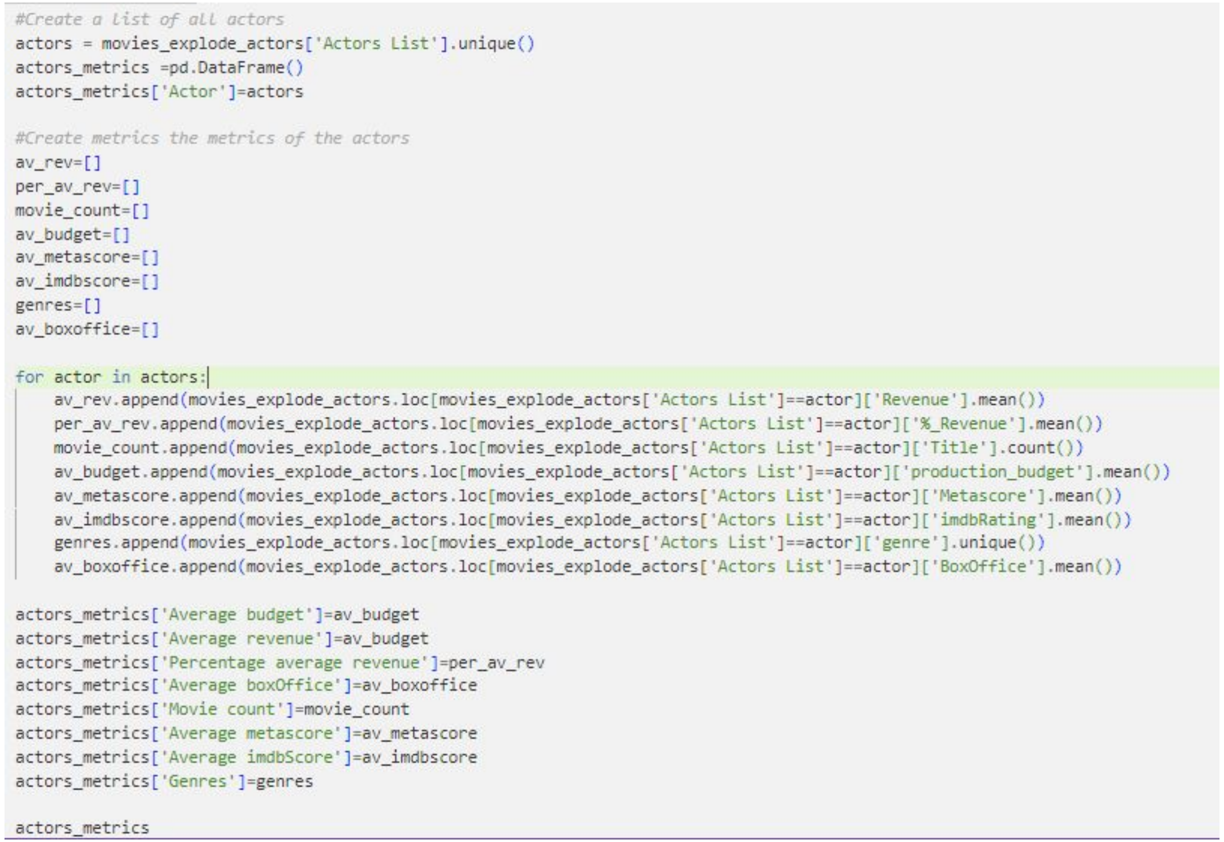
Metascore	imdbRating	imdbVotes	BoxOffice	Actors
37	5.4	152,796	$100,462,298	Steve Carell, Morgan Freeman, Lauren Graham

Resulting Data Display:



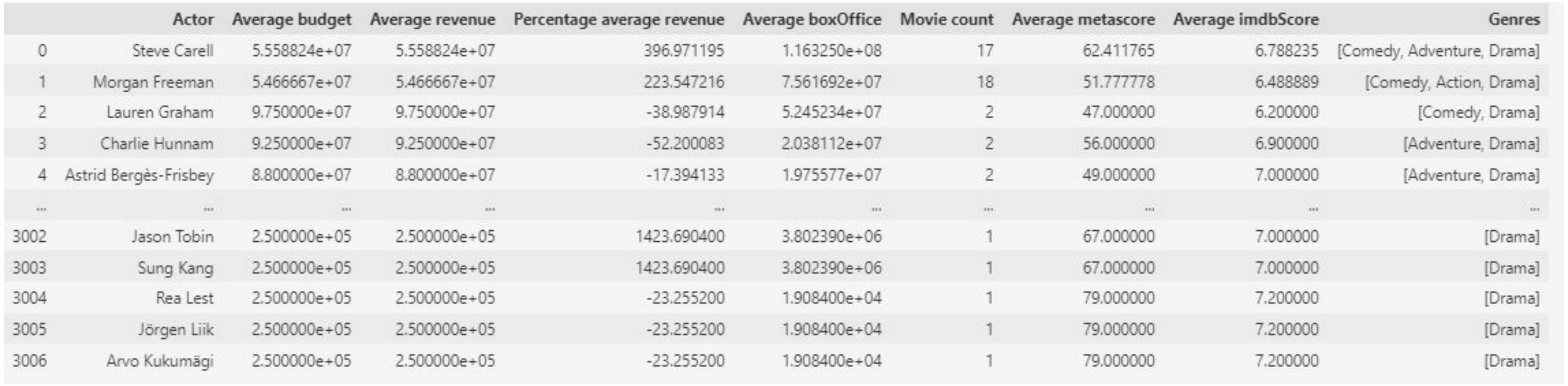
Metascore	imdbRating	imdbVotes	BoxOffice	Actors List
37.0	5.4	152796.0	100462298.0	Steve Carell
37.0	5.4	152796.0	100462298.0	Morgan Freeman
37.0	5.4	152796.0	100462298.0	Lauren Graham

The actors | Metric DataFrame Code



```python
#Create a list of all actors
actors = movies_explode_actors['Actors List'].unique()
actors_metrics =pd.DataFrame()
actors_metrics['Actor']=actors

#Create metrics the metrics of the actors
av_rev=[]
per_av_rev=[]
movie_count=[]
av_budget=[]
av_metascore=[]
av_imdbscore=[]
genres=[]
av_boxoffice=[]

for actor in actors:
    av_rev.append(movies_explode_actors.loc[movies_explode_actors['Actors List']==actor]['Revenue'].mean())
    per_av_rev.append(movies_explode_actors.loc[movies_explode_actors['Actors List']==actor]['%_Revenue'].mean())
    movie_count.append(movies_explode_actors.loc[movies_explode_actors['Actors List']==actor]['Title'].count())
    av_budget.append(movies_explode_actors.loc[movies_explode_actors['Actors List']==actor]['production_budget'].mean())
    av_metascore.append(movies_explode_actors.loc[movies_explode_actors['Actors List']==actor]['Metascore'].mean())
    av_imdbscore.append(movies_explode_actors.loc[movies_explode_actors['Actors List']==actor]['imdbRating'].mean())
    genres.append(movies_explode_actors.loc[movies_explode_actors['Actors List']==actor]['genre'].unique())
    av_boxoffice.append(movies_explode_actors.loc[movies_explode_actors['Actors List']==actor]['BoxOffice'].mean())

actors_metrics['Average budget']=av_budget
actors_metrics['Average revenue']=av_budget
actors_metrics['Percentage average revenue']=per_av_rev
actors_metrics['Average boxOffice']=av_boxoffice
actors_metrics['Movie count']=movie_count
actors_metrics['Average metascore']=av_metascore
actors_metrics['Average imdbScore']=av_imdbscore
actors_metrics['Genres']=genres

actors_metrics
```

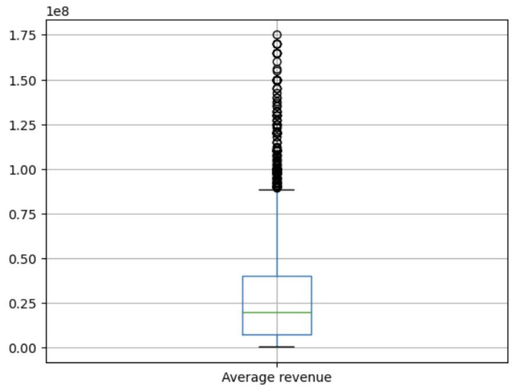
The actors | Metric DataFrame Result

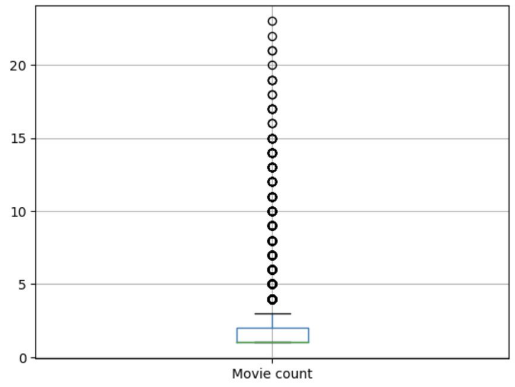


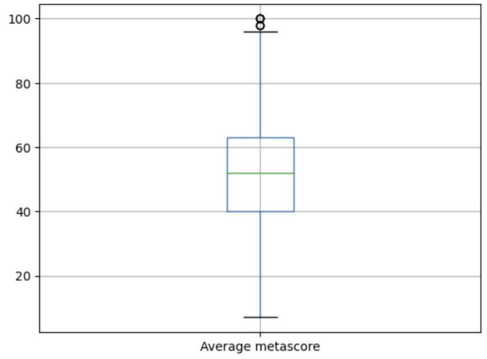
	Actor	Average budget	Average revenue	Percentage average revenue	Average boxOffice	Movie count	Average metascore	Average imdbScore	Genres
0	Steve Carell	5.558824e+07	5.558824e+07	396.971195	1.163250e+08	17	62.411765	6.788235	[Comedy, Adventure, Drama]
1	Morgan Freeman	5.466667e+07	5.466667e+07	223.547216	7.561692e+07	18	51.777778	6.488889	[Comedy, Action, Drama]
2	Lauren Graham	9.750000e+07	9.750000e+07	-38.987914	5.245234e+07	2	47.000000	6.200000	[Comedy, Drama]
3	Charlie Hunnam	9.250000e+07	9.250000e+07	-52.200083	2.038112e+07	2	56.000000	6.900000	[Adventure, Drama]
4	Astrid Bergès-Frisbey	8.800000e+07	8.800000e+07	-17.394133	1.975577e+07	2	49.000000	7.000000	[Adventure, Drama]
...	...	...	...	...	...	...	...	...	...
3002	Jason Tobin	2.500000e+05	2.500000e+05	1423.690400	3.802390e+06	1	67.000000	7.000000	[Drama]
3003	Sung Kang	2.500000e+05	2.500000e+05	1423.690400	3.802390e+06	1	67.000000	7.000000	[Drama]
3004	Rea Lest	2.500000e+05	2.500000e+05	-23.255200	1.908400e+04	1	79.000000	7.200000	[Drama]
3005	Jörgen Liik	2.500000e+05	2.500000e+05	-23.255200	1.908400e+04	1	79.000000	7.200000	[Drama]
3006	Arvo Kukumägi	2.500000e+05	2.500000e+05	-23.255200	1.908400e+04	1	79.000000	7.200000	[Drama]

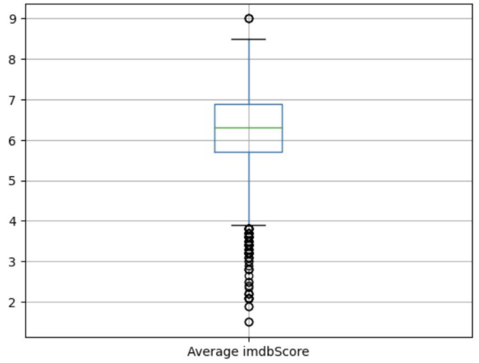
Actors | The Star Power (BoxPlots)

The influence of actor selection and Star Power on a movie's success is a complex interplay of various factors, and a holistic approach, encompassing experience, audience appeal, and expert assessment, is crucial in making informed decisions. While there are no guarantees in the ever-evolving entertainment industry, having a firm grasp of these essential metrics can substantially enhance the prospects of a successful cinematic endeavor.









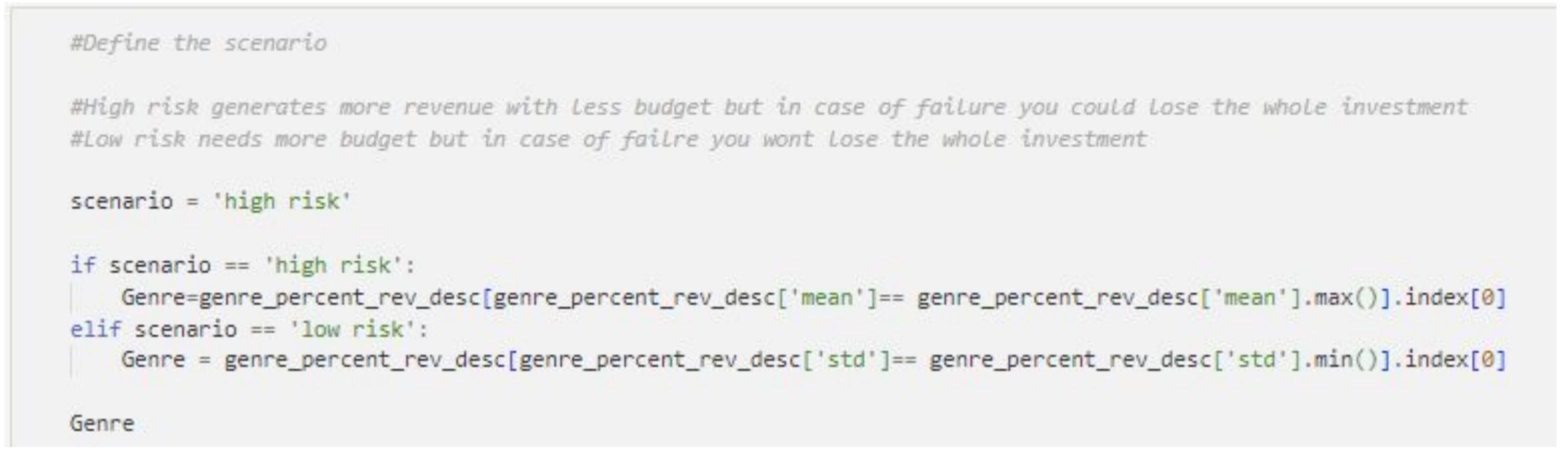
Predictive model

Through the meticulous collection and analysis of historical data, we have developed a sophisticated tool. This tool allows users to specify their preference for a low or high-risk scenario, guiding them in the decision-making process for a movie project. It assists in determining the appropriate genre for the movie, selecting a cast of actors who are likely to enhance its prospects, and providing estimates for both budget and projected revenue.

The formula for the perfect movie today

Genre attribute

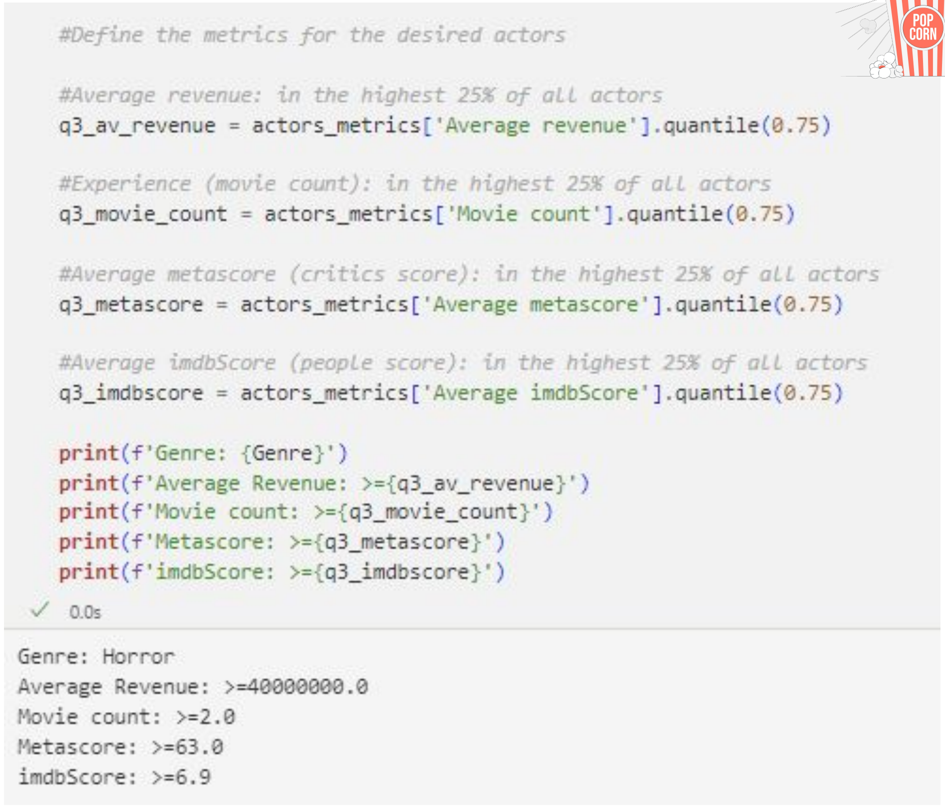
High Risk/High Reward vs Low Risk/Moderate reward



```python
#Define the scenario

#High risk generates more revenue with less budget but in case of failure you could lose the whole investment
#Low risk needs more budget but in case of failre you wont lose the whole investment

scenario = 'high risk'

if scenario == 'high risk':
    Genre=genre_percent_rev_desc[genre_percent_rev_desc['mean']== genre_percent_rev_desc['mean'].max()].index[0]
elif scenario == 'low risk':
    Genre = genre_percent_rev_desc[genre_percent_rev_desc['std']== genre_percent_rev_desc['std'].min()].index[0]

Genre
```

Best Actors (the top 25%)

- Average revenue
- Experience
- Average metaScore
- Average imdbScore

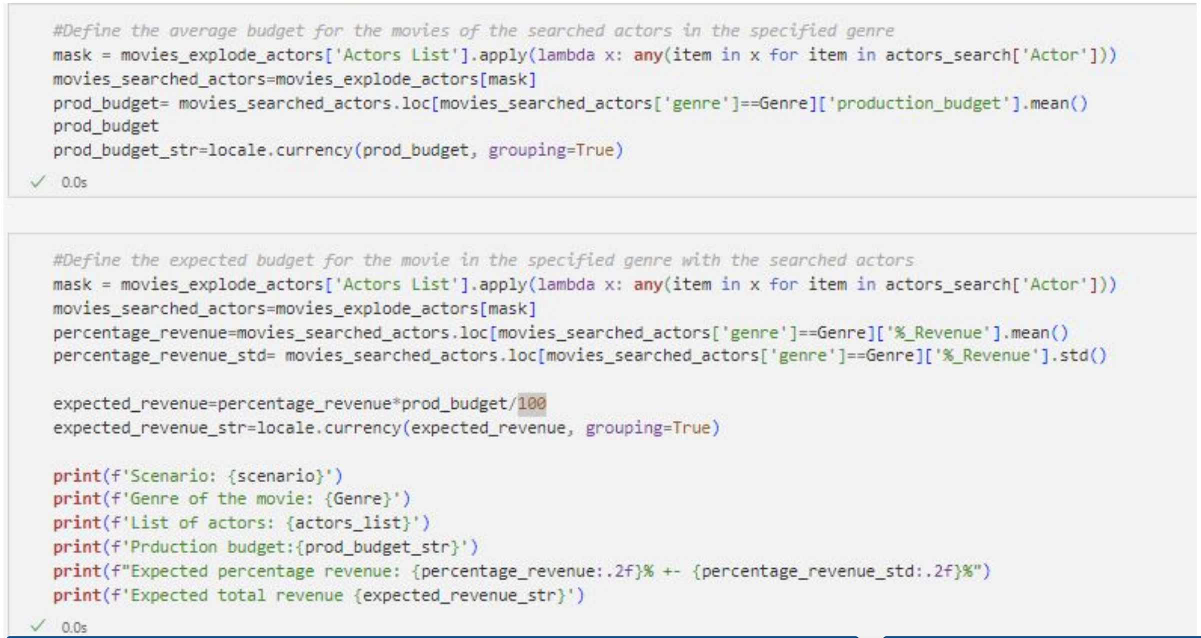


```
#Define the metrics for the desired actors

#Average revenue: in the highest 25% of all actors
q3_av_revenue = actors_metrics['Average revenue'].quantile(0.75)

#Experience (movie count): in the highest 25% of all actors
q3_movie_count = actors_metrics['Movie count'].quantile(0.75)

#Average metascore (critics score): in the highest 25% of all actors
q3_metascore = actors_metrics['Average metascore'].quantile(0.75)

#Average imdbScore (people score): in the highest 25% of all actors
q3_imdbscore = actors_metrics['Average imdbScore'].quantile(0.75)

print(f'Genre: {Genre}')
print(f'Average Revenue: >={q3_av_revenue}')
print(f'Movie count: >={q3_movie_count}')
print(f'Metascore: >={q3_metascore}')
print(f'imdbScore: >={q3_imdbscore}')
```

✓ 0.0s

```
Genre: Horror
Average Revenue: >=40000000.0
Movie count: >=2.0
Metascore: >=63.0
imdbScore: >=6.9
```

The perfect movie (High Risk/Low Risk):

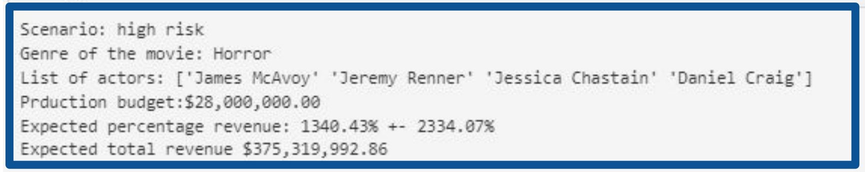


```python
#Define the average budget for the movies of the searched actors in the specified genre
mask = movies_explode_actors['Actors List'].apply(lambda x: any(item in x for item in actors_search['Actor']))
movies_searched_actors=movies_explode_actors[mask]
prod_budget= movies_searched_actors.loc[movies_searched_actors['genre']==Genre]['production_budget'].mean()
prod_budget
prod_budget_str=locale.currency(prod_budget, grouping=True)
```

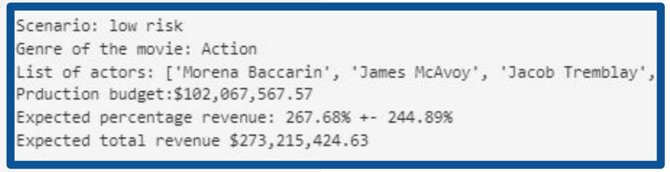
✓ 0.0s

```python
#Define the expected budget for the movie in the specified genre with the searched actors
mask = movies_explode_actors['Actors List'].apply(lambda x: any(item in x for item in actors_search['Actor']))
movies_searched_actors=movies_explode_actors[mask]
percentage_revenue=movies_searched_actors.loc[movies_searched_actors['genre']==Genre]['%_Revenue'].mean()
percentage_revenue_std= movies_searched_actors.loc[movies_searched_actors['genre']==Genre]['%_Revenue'].std()

expected_revenue=percentage_revenue*prod_budget/100
expected_revenue_str=locale.currency(expected_revenue, grouping=True)

print(f'Scenario: {scenario}')
print(f'Genre of the movie: {Genre}')
print(f'List of actors: {actors_list}')
print(f'Prduction budget:{prod_budget_str}')
print(f"Expected percentage revenue: {percentage_revenue:.2f}% +- {percentage_revenue_std:.2f}%")
print(f'Expected total revenue {expected_revenue_str}')
```

✓ 0.0s



```
Scenario: high risk
Genre of the movie: Horror
List of actors: ['James McAvoy' 'Jeremy Renner' 'Jessica Chastain' 'Daniel Craig']
Prduction budget:$28,000,000.00
Expected percentage revenue: 1340.43% +- 2334.07%
Expected total revenue $375,319,992.86
```



```
Scenario: low risk
Genre of the movie: Action
List of actors: ['Morena Baccarin', 'James McAvoy', 'Jacob Tremblay',
Prduction budget:$102,067,567.57
Expected percentage revenue: 267.68% +- 244.89%
Expected total revenue $273,215,424.63
```