

PREDICTION IMDB SCORES

project title: predicted for IMDb scores

phase 3: Development

part 1:

Data Preprocessing:

Data preprocessing is a crucial step within the statistics analysis and gadget gaining knowledge of pipeline.

It includes a sequence of strategies and operations finished on uncooked statistics to clean, organize, and transform it right into a layout that is suitable for analysis or device mastering version schooling.

Data preprocessing goals to enhance the first-class of the records, making it greater reliable and conducive to generating accurate consequences.

Here are some common tasks and techniques involved in data preprocessing:

Data Cleaning:

Handling missing values: Deciding how to deal with missing data, whether by imputing values or removing incomplete records.

Outlier detection and treatment: Identifying and handling data points that significantly deviate from the norm.

Noise reduction:

Smoothing noisy data through techniques like filtering.

Data Transformation:

Data normalization: Scaling numerical features to a standard range (e.g., between 0 and 1) to ensure that they have similar influence in the analysis.

Encoding categorical variables: Converting categorical data into numerical format, such as one-hot encoding or label encoding.

Feature engineering: Creating new features or modifying existing ones to capture more meaningful information from the data.

Dimensionality reduction: Reducing the number of features while retaining essential information, using methods like Principal Component Analysis (PCA).

Data Integration:

Merging or joining datasets: Combining data from multiple sources into a single dataset for analysis.

Aggregation: Summarizing data at a higher level of granularity, such as aggregating daily sales into monthly totals.

Data Reduction:

Sampling: Reducing the size of a large dataset by randomly selecting a representative subset.

Binning: Grouping continuous data into discrete bins to simplify analysis.

Filtering: Selecting a subset of data based on specific criteria.

Data Standardization:

Ensuring that data follows a consistent format and structure.

Date and time format conversion: Converting date and time data into a uniform format.

Currency conversion: Converting monetary values into a common currency.

Data Scaling:

Scaling numerical data to a common range to prevent some features from dominating the analysis.

Data preprocessing is an iterative process that may involve several of these steps in various orders, depending on the specific dataset and the analysis goals. Proper data preprocessing is essential for improving the accuracy and effectiveness of machine learning models, as well as for making data more accessible for traditional statistical analysis.

Here is the data preprocessing codes along with the output of the given dataset:

Importing the libraries:

Import three basic libraries which are very common in machine learning and will be used every time you train a model

NumPy: it is a library that allows us to work with arrays and as most machine learning models work on arrays NumPy makes it easier

matplotlib: this library helps in plotting graphs and charts, which are very useful while showing the result of your model

Pandas: pandas allows us to import our dataset and also creates a matrix of features containing the dependent and independent variable.

Code:

```
#Import libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plot
import seaborn as sns
from plotnine import *
```

1.1 Background

This dataset contains the information about the movies . For a movie to be commercial success , it depends on various factors like director, actors ,critic reviews and viewers reaction. Imdb score is one of the important factor to measure the movie's success.

1.2 Description of dataset attributes

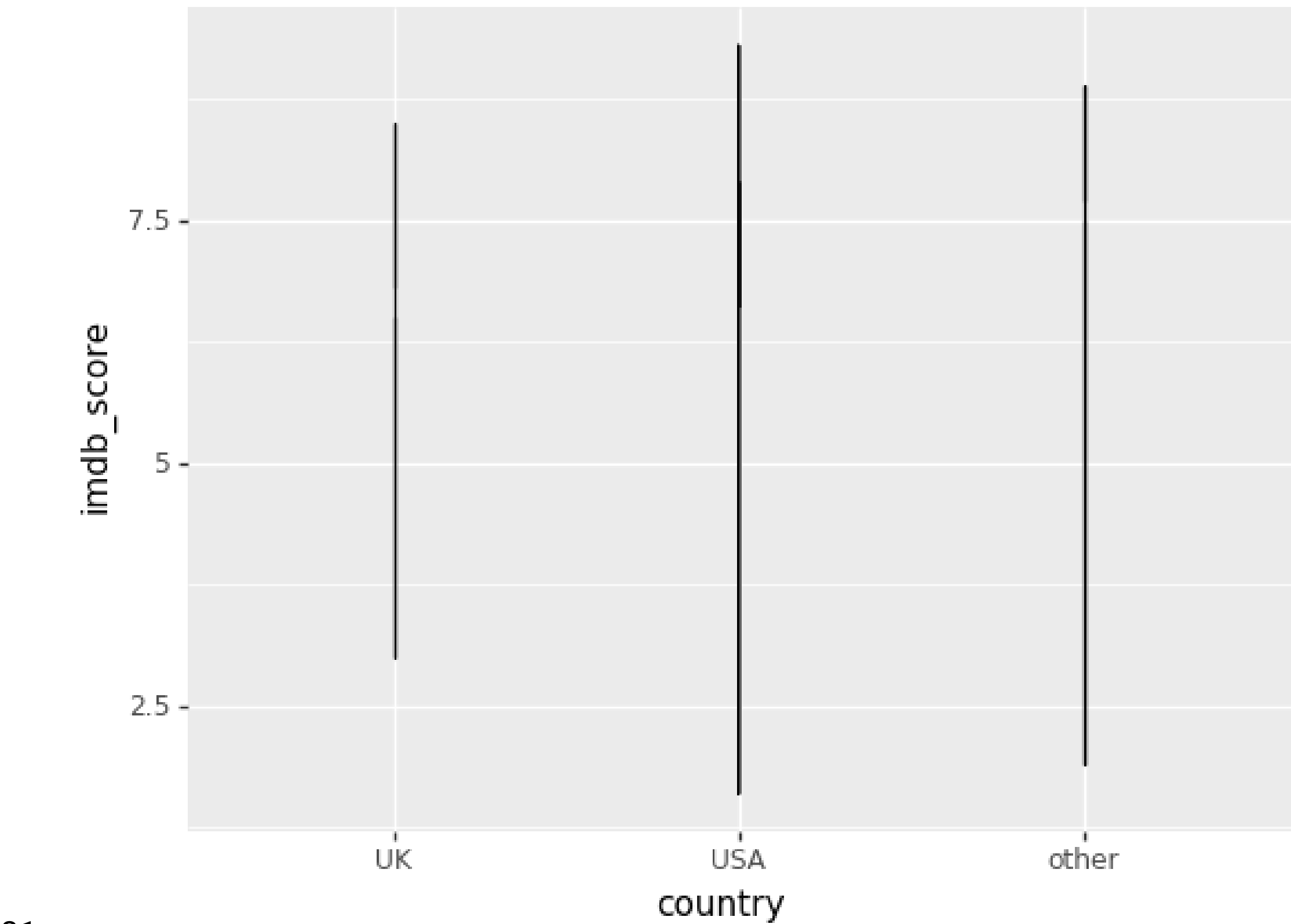
Please find the details for the dataset attributes:-

1. Color :- Movie is black or coloured
2. Director_name:- Name of the movie director
3. num_critic_for_reviews :- No of critics for the movie
4. duration:- movie duration in minutes
5. director_facebook_likes:-Number of likes for the Director on his Facebook Page
6. actor_3_facebook_likes:- No of likes for the actor 3 on his/her facebook Page
7. actor2_name:- name of the actor 2
8. actor_1_facebook_likes:- No of likes for the actor 1 on his/her facebook Page
9. gross:- Gross earnings of the movie in Dollars
10. genres:- Film categorization like 'Animation', 'Comedy', 'Romance', 'Horror', 'Sci-Fi', 'Action', 'Family'
11. actor_1_name:- Name of the actor 1
12. movie_title:-Title of the movie
13. num_voted_users:-No of people who voted for the movie
14. cast_total_facebook_likes:- Total facebook like for the movie
15. actor_3_name:- Name of the actor 3
16. facenumber_in_poster:- No of actors who featured in the movie poster
17. plot_keywords:-Keywords describing the movie plots
18. movie_imdb_link:-Link of the movie link
19. num_user_for_reviews:- Number of users who gave a review
20. language:- Language of the movie
21. country:- Country where movie is produced
22. content_rating:- Content rating of the movie
23. budget:- Budget of the movie in Dollars
24. title_year:- The year in which the movie is released
25. actor_2_facebook_likes:- facebook likes for the actor 2
26. imdb_score:- IMDB score of the movie

- 27. aspect_ratio :- Aspect ratio the movie was made in
- 28. movie_facebook_likes:- Total no of facebook likes for the movie

29. **1.3 Case Study**

30. The dataset here gives the massive information about the movies and their IMDB scores respectively. We are going to analyze each and every factors which can influence the imdb ratings so that we can predict better results. The movie with the higher imdb score is more successful as compared to the movies with low imdb score



2. Data Preprocessing

```
#Reading the Data
movie_df=pd.read_csv("/kaggle/input/imdb-5000-movie-dataset/movie_metadata.csv")

#Displaying the first 10 records
movie_df.head(10)
```

#Shape of the dataset (no of rows and no of columns)

```
movie_df.shape
```

Out[4]:

```
(5043, 28)
```

#Displaying the data type of the dataset attributes

```
movie_df.dtypes
```

Out[5]:

```
color                object
director_name        object
num_critic_for_reviews  float64
duration             float64
director_facebook_likes  float64
actor_3_facebook_likes  float64
actor_2_name         object
actor_1_facebook_likes  float64
gross                float64
genres               object
actor_1_name         object
movie_title          object
num_voted_users       int64
cast_total_facebook_likes  int64
actor_3_name         object
facenumber_in_poster  float64
plot_keywords         object
movie_imdb_link       object
num_user_for_reviews  float64
language             object
country              object
content_rating        object
budget              float64
title_year           float64
actor_2_facebook_likes  float64
imdb_score            float64
aspect_ratio          float64
movie_facebook_likes  int64
dtype: object
```

****We can say we have the dataset divided into categorical and numeric columns "**

Categorical Columns

Color,Director name, actor name,genres,movie_title,language,country,content_rating.

Numerical Columns

num_critic_for_reviews,duration,director_facebook_likes ,actor_3_facebook_likes,actor_1_facebook_likes ,gross,num_voted_users,cast_total_facebook_likes,facenumber_in_poster,num_user_for_reviews ,budget,title_year, actor_2_facebook_likes ,imdb_score,aspect_ratio,movie_facebook_like

```
Index(['director_name', 'num_critic_for_reviews', 'duration',
      'director_facebook_likes', 'actor_3_facebook_likes', 'actor_2_name',
      'actor_1_facebook_likes', 'gross', 'genres', 'actor_1_name',
      'movie_title', 'num_voted_users', 'cast_total_facebook_likes',
      'actor_3_name', 'facenumber_in_poster', 'plot_keywords',
      'num_user_for_reviews', 'language', 'country', 'content_rating',
      'budget', 'title_year', 'actor_2_facebook_likes', 'imdb_score',
      'aspect_ratio', 'movie_facebook_likes'],
      dtype='object')
```

```
director_name      True
num_critic_for_reviews  True
duration           True
director_facebook_likes  True
actor_3_facebook_likes  True
actor_2_name       True
actor_1_facebook_likes  True
gross              True
genres             False
actor_1_name       True
movie_title        False
num_voted_users    False
cast_total_facebook_likes  False
actor_3_name       True
facenumber_in_poster  True
plot_keywords      True
num_user_for_reviews  True
language           True
country            True
content_rating     True
budget             True
title_year         True
actor_2_facebook_likes  True
imdb_score         False
aspect_ratio       True
movie_facebook_likes  False
dtype: bool
```

```
movie_df.isna().sum()
director_name      104
num_critic_for_reviews  50
duration           15
director_facebook_likes  104
actor_3_facebook_likes  23
actor_2_name       13
actor_1_facebook_likes  7
gross              884
genres             0
actor_1_name       7
movie_title        0
num_voted_users    0
cast_total_facebook_likes  0
actor_3_name       23
```



```
facenumber_in_poster    13
plot_keywords           153
num_user_for_reviews    21
language                12
country                 5
content_rating          303
budget                 492
title_year             108
actor_2_facebook_likes  13
imdb_score              0
aspect_ratio           329
movie_facebook_likes    0
dtype: int64
```

```
movie_df.dropna(axis=0,subset=['director_name', 'num_critic_for_reviews','duration','director_faceb
ook_likes','actor_3_facebook_likes','actor_2_name','actor_1_facebook_likes','actor_1_name','actor_3
_name','facenumber_in_poster','num_user_for_reviews','language','country','actor_2_facebook_likes',
plot_keywords'],inplace=True)
```

```
movie_df.shape
```

```
(4737, 26)
```

We lost only 6% of the data which is acceptable

#Replacing the content rating with Value R as it has highest frequency

```
movie_df["content_rating"].fillna("R", inplace = True)
```

In [15]:

#Replacing the aspect_ratio with the median of the value as the graph is right skewed

```
movie_df["aspect_ratio"].fillna(movie_df["aspect_ratio"].median(),inplace=True)
```

In [16]:

```
linkcode
```

#We need to replace the value in budget with the median of the value

```
movie_df["budget"].fillna(movie_df["budget"].median(),inplace=True)
```

We need to replace the value in gross with the median of the value

```
movie_df['gross'].fillna(movie_df['gross'].median(),inplace=True)
```

In [18]:

Recheck that all the null values are removed

```
movie_df.isna().sum()
```

Out[18]:

```
director_name      0
num_critic_for_reviews  0

director_facebook_likes  0
actor_3_facebook_likes  0
actor_2_name        0
actor_1_facebook_likes  0
gross              0
genres             0
```

```
actor_1_name      0
movie_title       0
num_voted_users   0
cast_total_facebook_likes  0
actor_3_name      0
facenumber_in_poster  0
plot_keywords     0
num_user_for_reviews  0
language          0
country           0
content_rating    0
budget            0
title_year        0
actor_2_facebook_likes  0
imdb_score        0
aspect_ratio      0
movie_facebook_likes  0
dtype: int64
```



#Removing the duplicate values in the dataset

```
movie_df.drop_duplicates(inplace=True)
movie_df.shape
```

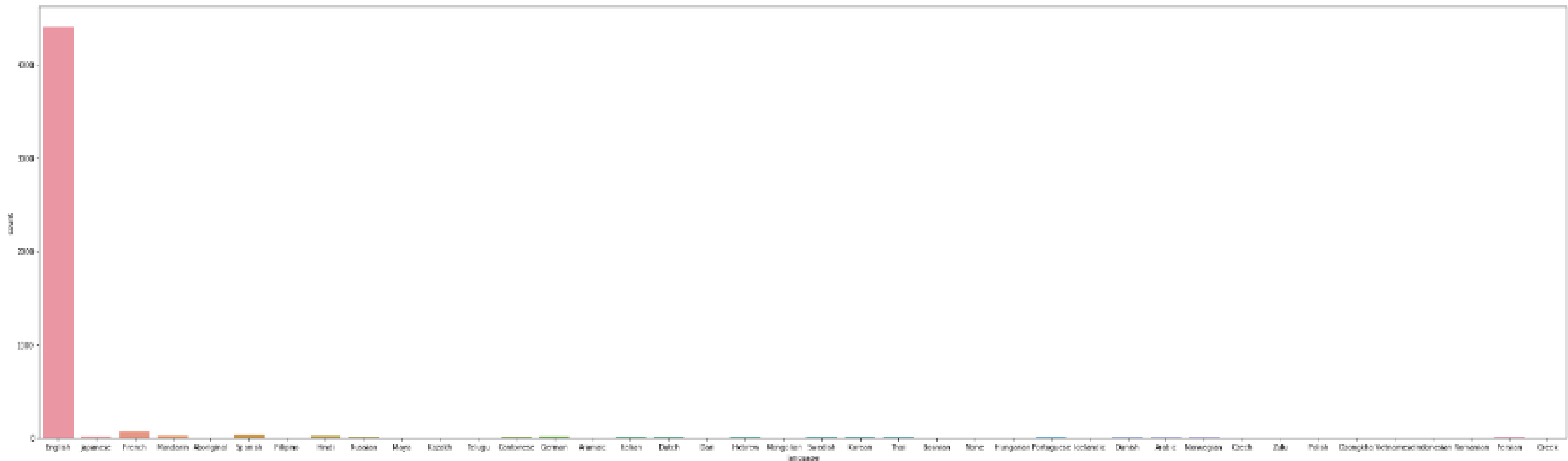
Count of the language values

```
movie_df["language"].value_counts()
English      4405
French       69
Spanish      35
Hindi        25
Mandarin     24
German       18
Japanese     16
Russian      11
Italian      10
Cantonese   10
Portuguese   8
Korean       8
Danish       5
Norwegian    4
Swedish      4
Hebrew       4
Dutch        4
Persian      4
Arabic       3
Thai         3
Indonesian   2
None         2
Aboriginal   2
Dari         2
Zulu         2
Hungarian    1
Mongolian    1
Greek        1
```



```
Romanian      1
Bosnian       1
Telugu        1
Maya          1
Polish        1
Filipino      1
Czech         1
Dzongkha     1
Kazakh        1
Vietnamese   1
Icelandic    1
Aramaic       1
Name: language, dtype: int64
```

```
# Graphical presentaion
plt.figure(figsize=(40,10))
sns.countplot(movie_df["language"])
plt.show()
```



#Most of the values for the languages is english we can drop the english column

```
movie_df.drop('language',axis=1,inplace=True)
```

In [23]:

```
linkcode
#Creating a new column to check the net profit made by the company (Gross-Budget)

movie_df["Profit"]=movie_df['budget'].sub(movie_df['gross'], axis = 0)
value_counts=movie_df["country"].value_counts()
print(value_counts)
```

```
USA      3568
UK        420
France   149
Canada   107
Germany   96
Australia 53
Spain     32
India     27
China     24
Japan     21
```

Italy	20
Hong Kong	16
New Zealand	14
South Korea	12
Ireland	11
Denmark	11
Russia	11
Mexico	11
South Africa	8
Brazil	8
Norway	7
Netherlands	5
Sweden	5
Thailand	4
Iran	4
Argentina	4
Czech Republic	3
Switzerland	3
Belgium	3
Israel	3
West Germany	3
Poland	2
Taiwan	2
Iceland	2
Romania	2
Hungary	2
Greece	2
Soviet Union	1
Slovakia	1
Finland	1
Official site	1
Turkey	1
Peru	1
Libya	1
Afghanistan	1
Cambodia	1
Indonesia	1
Nigeria	1
Kyrgyzstan	1
Colombia	1
New Line	1
Philippines	1
Bahamas	1
Bulgaria	1
Georgia	1
Aruba	1
Chile	1

Name: country, dtype: int64

We can see most of the movies are from USA ,UK and the rest of the countries

In [26]:

```
##get top 2 values of index
```

```
##get top 2 values of index
```

```
vals = value_counts[:2].index
```

```
print (vals)
```

```
movie_df['country'] = movie_df.country.where(movie_df.country.isin(vals), 'other')
```

```
Index(['USA', 'UK'], dtype='object')
```

In [27]:

```
#Successfully divided the country into three catogories
movie_df["country"].value_counts()
```

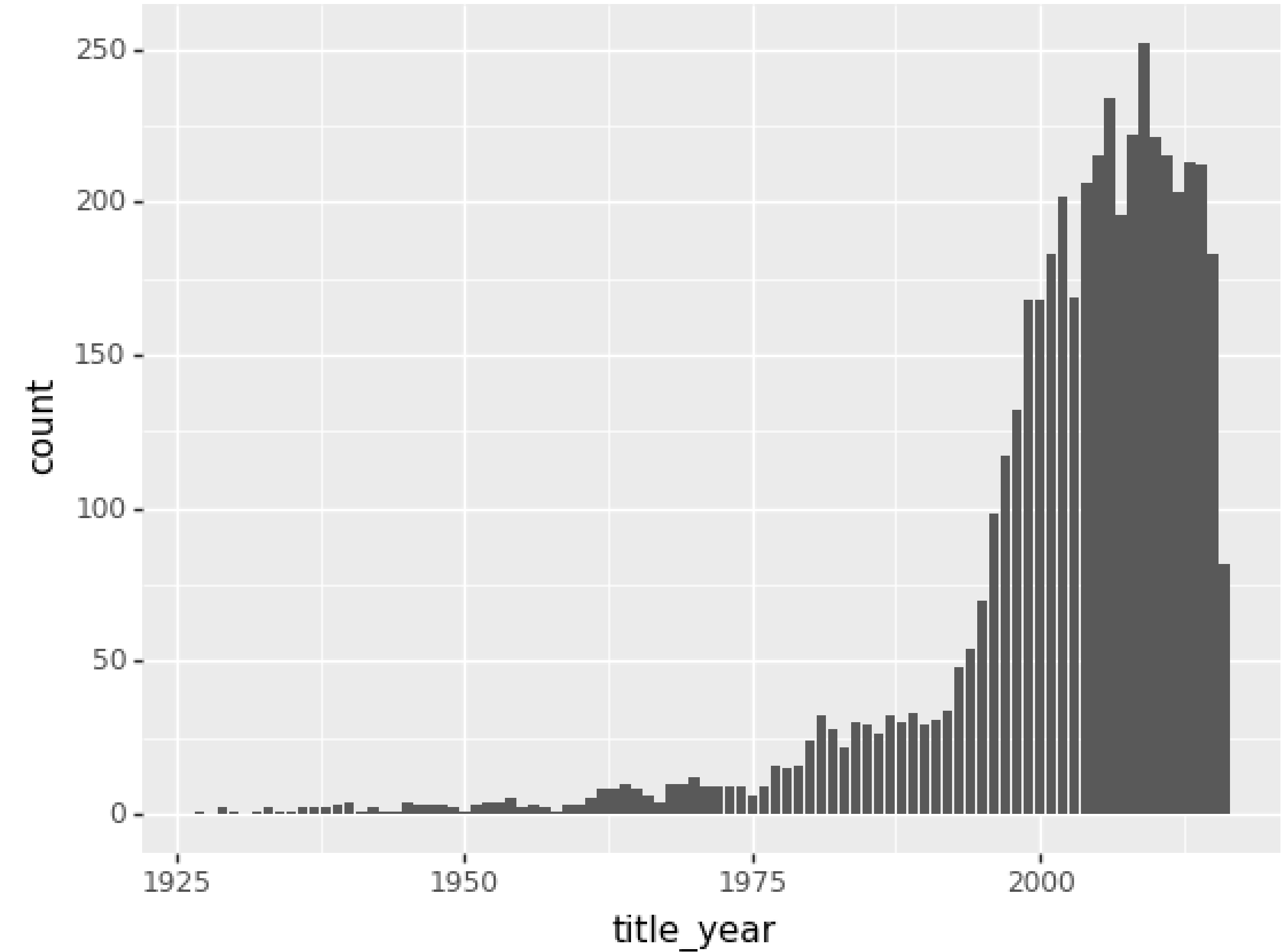
Out[27]:

```
USA    3568
other   707
UK      420
Name: country, dtype: int64
```

In [28]:

```
linkcode
movie_df.head(10)
```

```
/opt/conda/lib/python3.6/site-packages/plotnine/coords/coord_cartesian.py:31: MatplotlibDepre
cationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use t
ypes.SimpleNamespace instead.
  self.limits = Bunch(xlim=xlim, ylim=ylim)
/opt/conda/lib/python3.6/copy.py:274: MatplotlibDeprecationWarning: The Bunch class was depr
ecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.
  y = func(*args)
/opt/conda/lib/python3.6/site-packages/plotnine/facets/facet.py:151: MatplotlibDeprecationWar
ning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.Simp
leNamespace instead.
  scales = Bunch()
/opt/conda/lib/python3.6/site-packages/plotnine/facets/layout.py:147: MatplotlibDeprecationWar
ning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.Simp
leNamespace instead.
  return Bunch(x=xsc, y=ysc)
```



#Relationship between the imdb score and the profit made by the movie

```

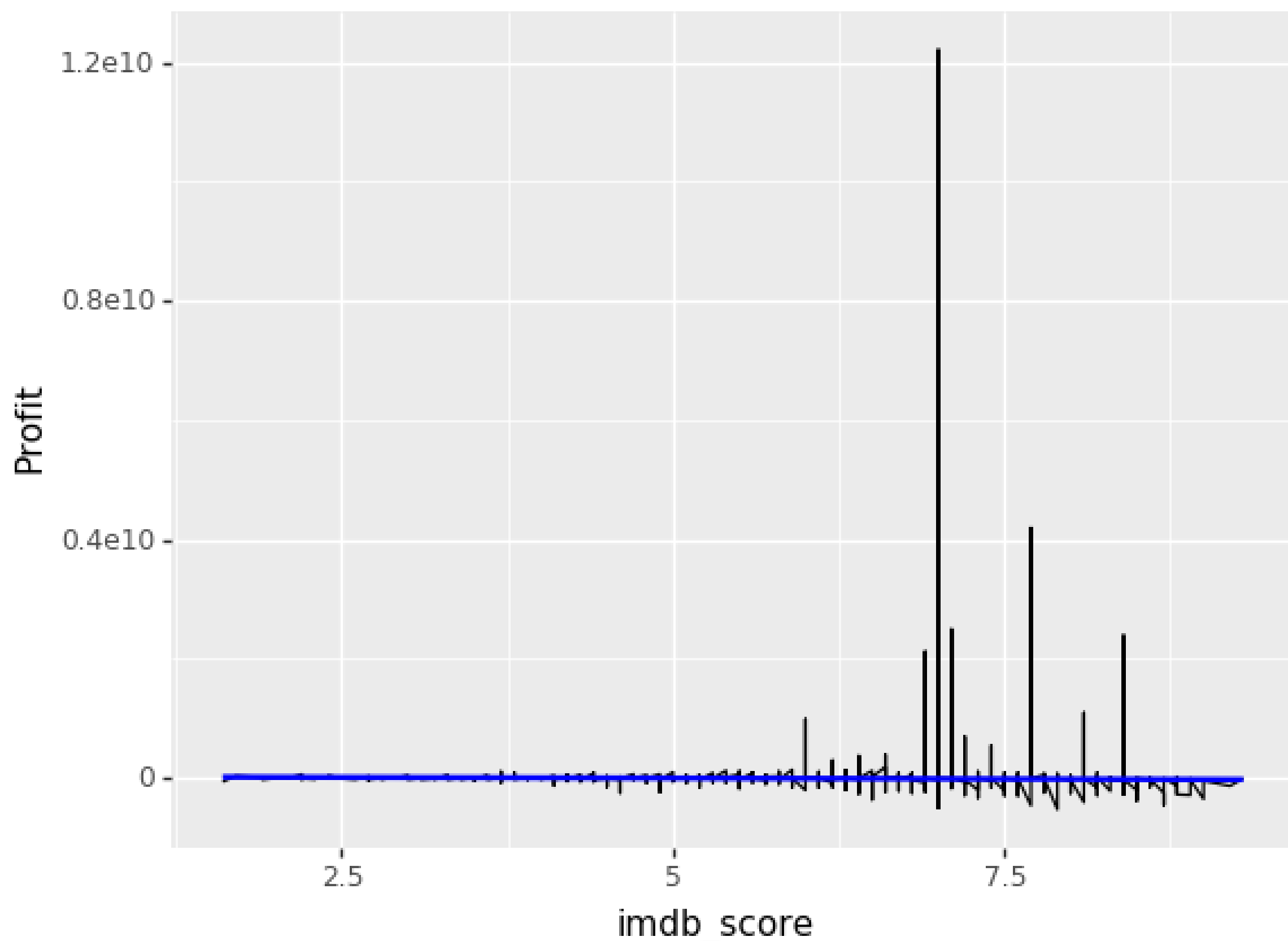
ggplot(aes(x='imdb_score', y='Profit'), data=movie_df) +\
  geom_line() +\
  stat_smooth(colour='blue', span=1)
/opt/conda/lib/python3.6/site-packages/plotnine/coords/coord_cartesian.py:31: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.
  self.limits = Bunch(xlim=xlim, ylim=ylim)
/opt/conda/lib/python3.6/copy.py:274: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.
  y = func(*args)
/opt/conda/lib/python3.6/site-packages/plotnine/facets/facet.py:151: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.
  scales = Bunch()
/opt/conda/lib/python3.6/site-packages/plotnine/facets/layout.py:147: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.
  return Bunch(x=xsc, y=ysc)
/opt/conda/lib/python3.6/site-packages/numpy/core/fromnumeric.py:2389: FutureWarning: Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.

```

```

return ptp(axis=axis, out=out, **kwargs)

```



#Finding the corelation between imdb_rating with respect to no of facebook likes

```

(ggplot(movie_df)
+ aes(x='imdb_score', y='movie_facebook_likes')
+ geom_line()
+ labs(title='IMDB_Score vs. Facebook like for Movies', x='IMDB scores', y='Facebook Likes for movies')
)

```

/opt/conda/lib/python3.6/site-packages/plotnine/coords/coord_cartesian.py:31: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

```
self.limits = Bunch(xlim=xlim, ylim=ylim)
```

/opt/conda/lib/python3.6/copy.py:274: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

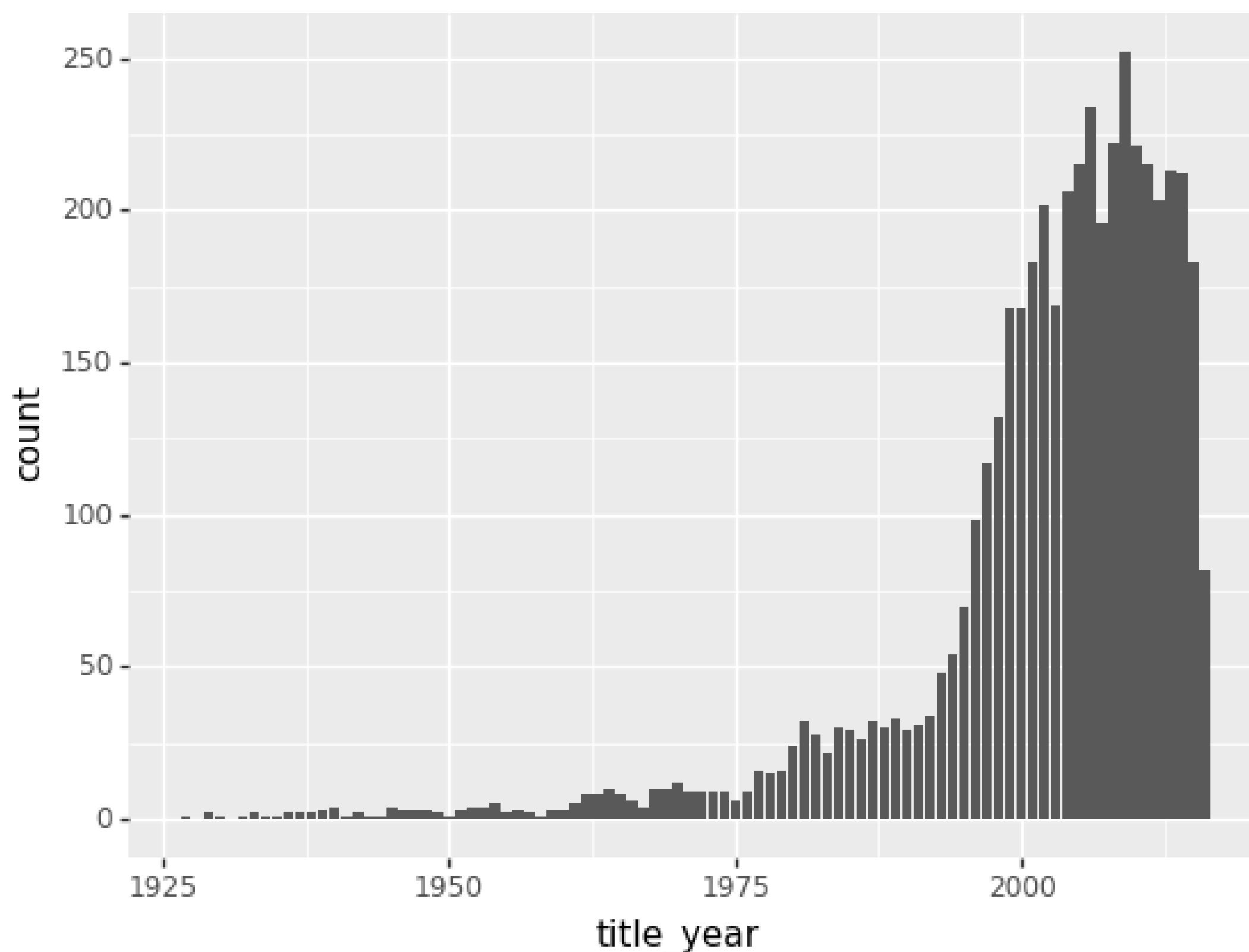
```
y = func(*args)
```

/opt/conda/lib/python3.6/site-packages/plotnine/facets/facet.py:151: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

```
scales = Bunch()
```

/opt/conda/lib/python3.6/site-packages/plotnine/facets/layout.py:147: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.

```
return Bunch(x=xsc, y=ysc)
```



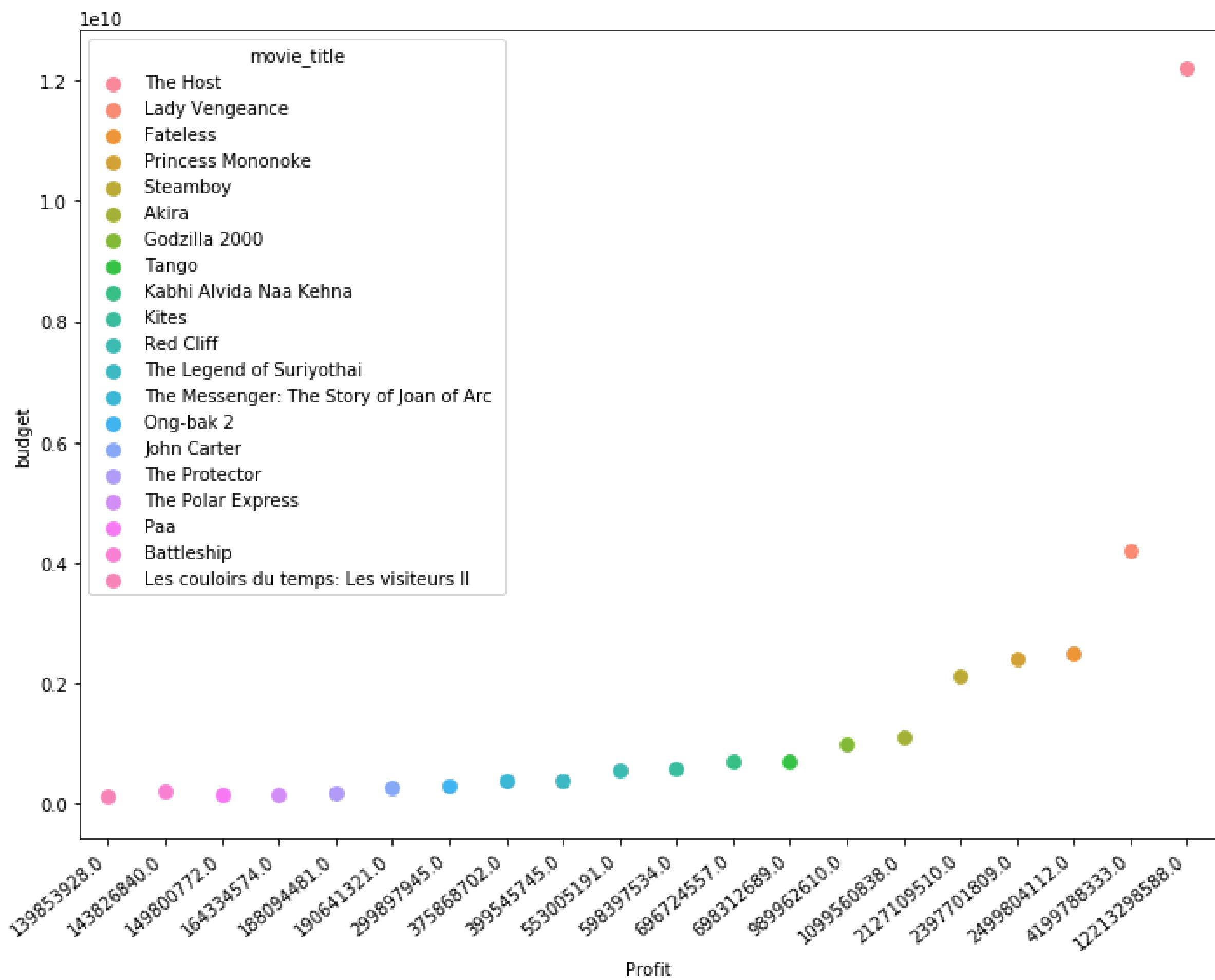
#Top 20 movies based on the profit they made

```
plt.figure(figsize=(10,8))
movie_df= movie_df.sort_values(by ='Profit' , ascending=False)
movie_df_new=movie_df.head(20)
ax=sns.pointplot(movie_df_new['Profit'], movie_df_new['budget'], hue=movie_df_new['movie_title'])
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")
plt.tight_layout()
plt.show()
```

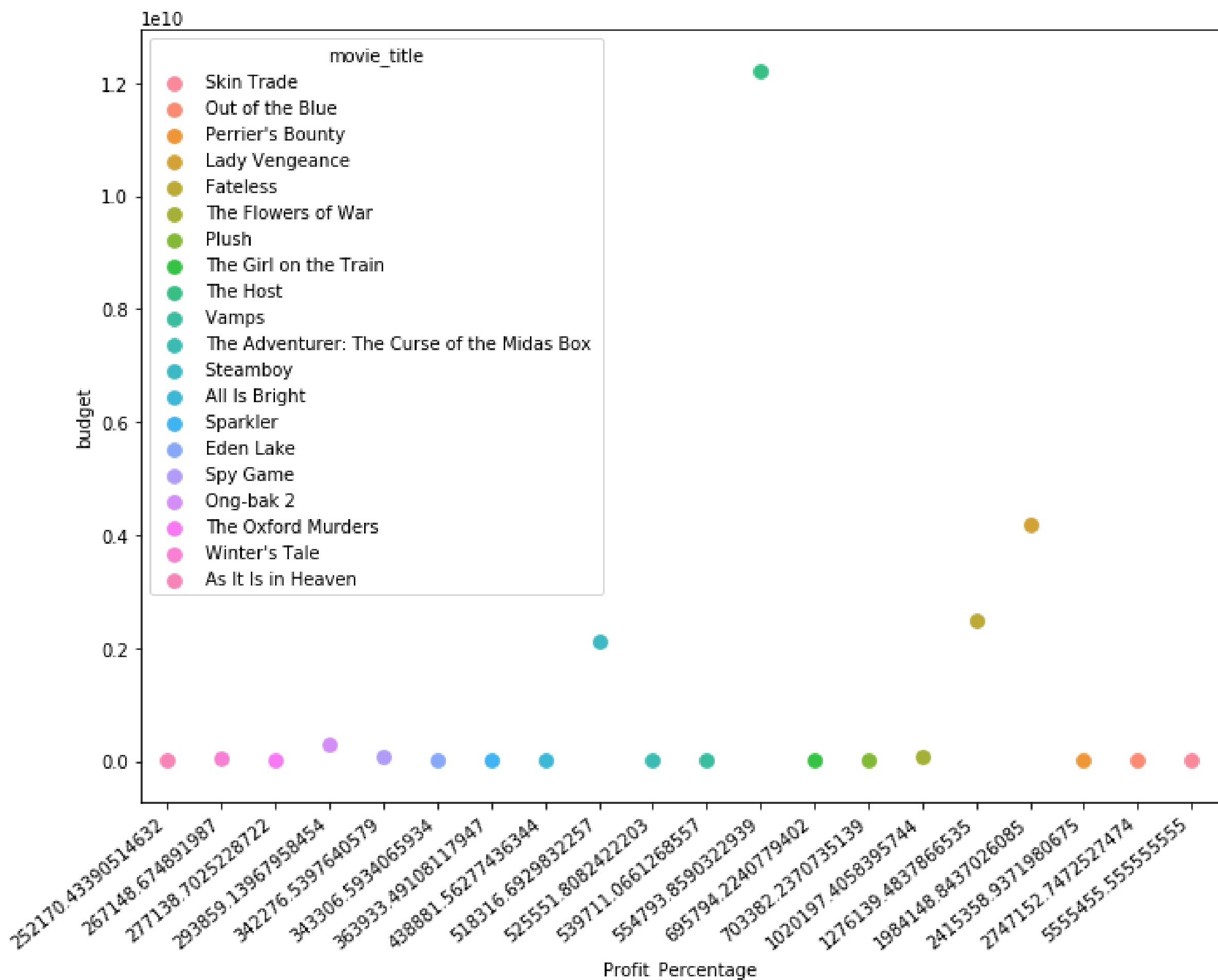
#Top 20 movies based on the profit they made

```
plt.figure(figsize=(10,8))
movie_df= movie_df.sort_values(by ='Profit' , ascending=False)
movie_df_new=movie_df.head(20)
ax=sns.pointplot(movie_df_new['Profit'], movie_df_new['budget'], hue=movie_df_new['movie_title'])
```

```
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")
plt.tight_layout()
```



```
# Top 20 movies based on the profit percentage
plt.figure(figsize=(10,8))
movie_df= movie_df.sort_values(by ='Profit_Percentage' , ascending=False)
movie_df_new=movie_df.head(20)
ax=sns.pointplot(movie_df_new['Profit_Percentage'], movie_df_new['budget'], hue=movie_df_new['movie_title'])
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")
plt.tight_layout()
plt.show()
```

#Top 20 directors based on the IMDB ratings

```
plt.figure(figsize=(10,8))
```

```
movie_df= movie_df.sort_values(by ='imdb_score' , ascending=False)
```

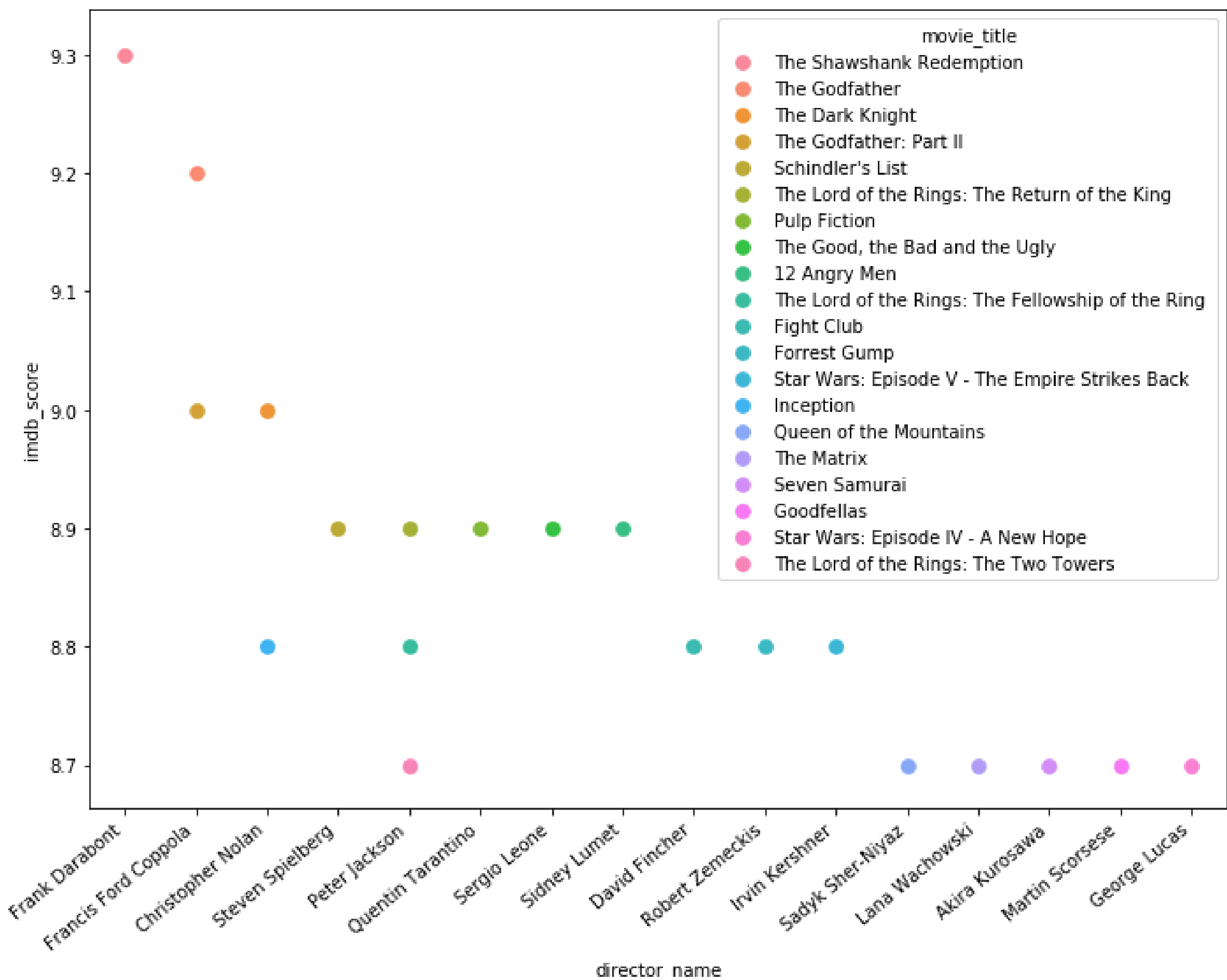
```
movie_df_new=movie_df.head(20)
```

```
ax=sns.pointplot(movie_df_new['director_name'], movie_df_new['imdb_score'], hue=movie_df_new['movie_title'])
```

```
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")
```

```
plt.tight_layout()
```

```
plt.show()
```



#Commercial success vs critial acclaim

```
movie_df= movie_df.sort_values(by ='Profit_Percentage' , ascending=False)
```

```
movie_df_new=movie_df.head(20)
```

```
(ggplot(movie_df_new)
```

```
+ aes(x='imdb_score', y='gross',color = "content_rating")
```

```
+ geom_point()
```

```
+ geom_hline(aes(yintercept = 600)) +
```

```
geom_vline(aes(xintercept = 10)) +
```

```
xlab("Imdb score") +
```

```
ylab("Gross money earned in million dollars") +
```

```
ggtitle("Commercial success Vs Critical acclaim") +
```

```
annotate("text", x = 8.5, y = 700, label = "High ratings \n & High gross"))
```

```
/opt/conda/lib/python3.6/site-packages/plotnine/coords/coord_cartesian.py:31: Mat
```

```
/opt/conda/lib/python3.6/site-packages/plotnine/coords/coord_cartesian.py:31: MatplotlibDepre
cationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use t
ypes.SimpleNamespace instead.
```

```
self.limits = Bunch(xlim=xlim, ylim=ylim)
```

```
/opt/conda/lib/python3.6/copy.py:274: MatplotlibDeprecationWarning: The Bunch class was depr
ecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.
```

```
y = func(*args)
```

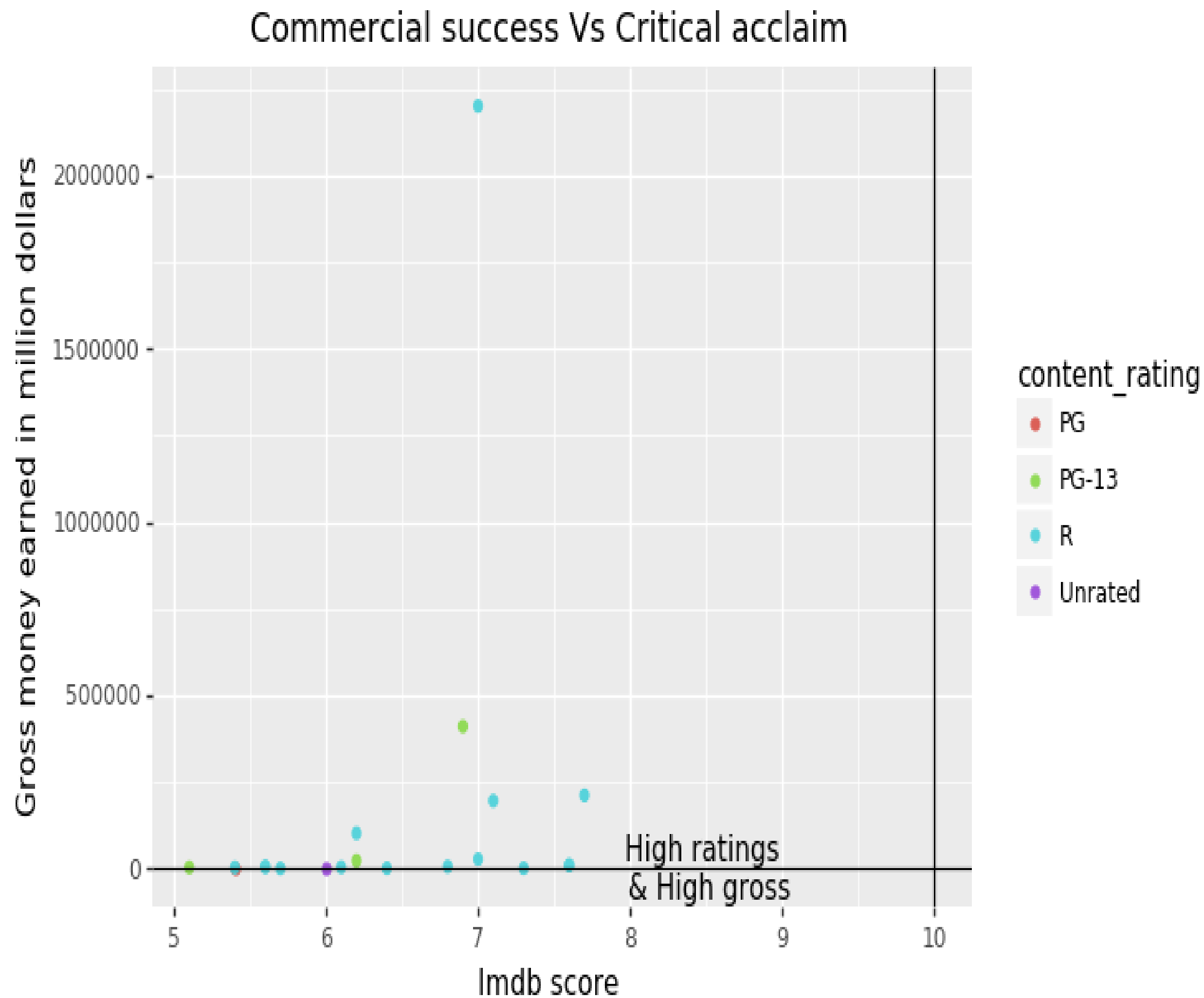
```
/opt/conda/lib/python3.6/site-packages/plotnine/layer.py:520: MatplotlibDeprecationWarning: isi
nstance(..., numbers.Number)
```

```
return not cbook.iterable(value) and (cbook.is_numlike(value) or
```

```
/opt/conda/lib/python3.6/site-packages/plotnine/facets/facet.py:151: MatplotlibDeprecationWar
ning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.Simp
leNamespace instead.
```

```
scales = Bunch()
```

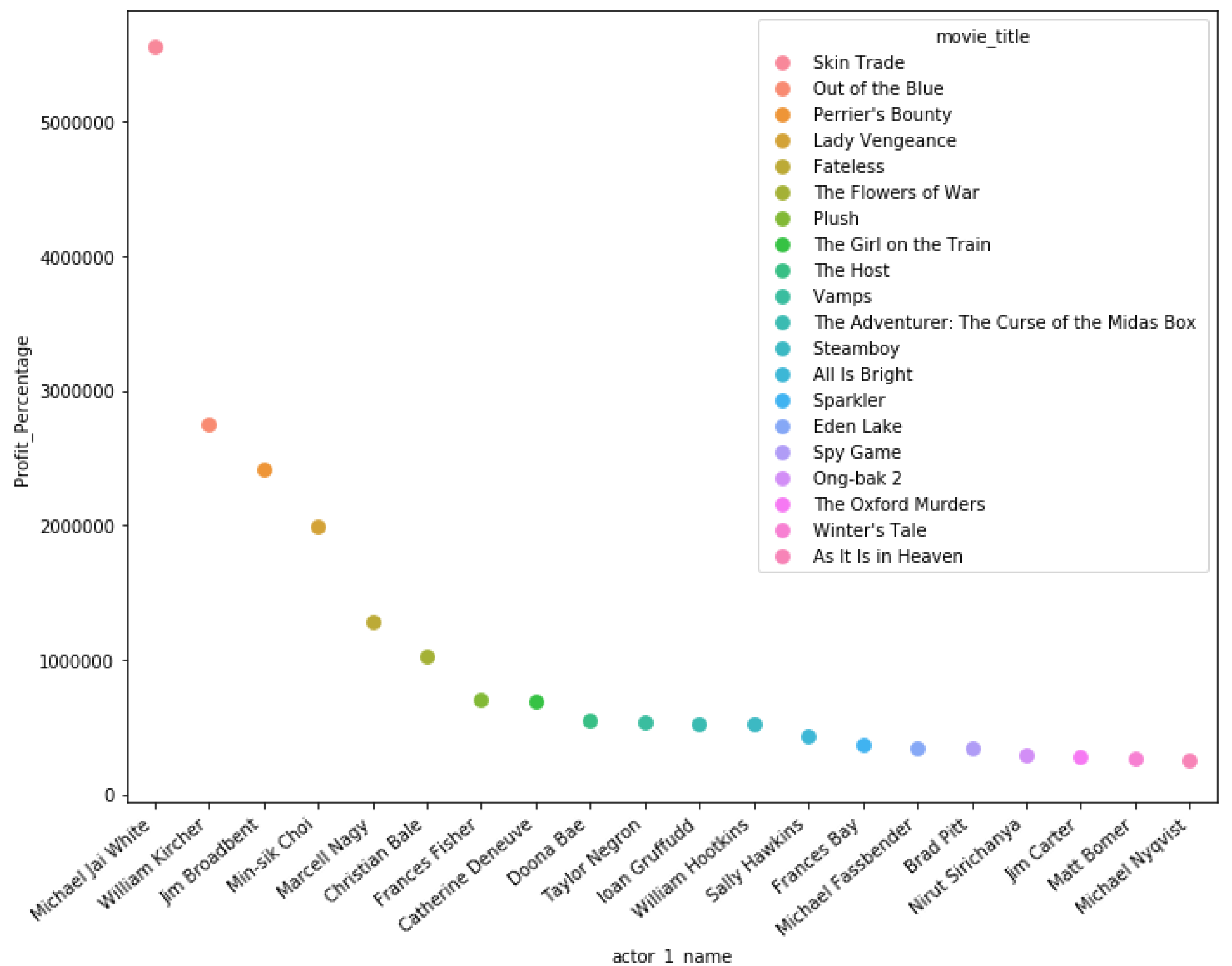
```
/opt/conda/lib/python3.6/site-packages/plotnine/facets/layout.py:147: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.
    return Bunch(x=xsc, y=ysc)
/opt/conda/lib/python3.6/site-packages/plotnine/coords/coord.py:144: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.
    y=panel_params['y_range'])
/opt/conda/lib/python3.6/site-packages/plotnine/guides/guide_legend.py:179: MatplotlibDeprecationWarning: The Bunch class was deprecated in Matplotlib 3.0 and will be removed in 3.2. Use types.SimpleNamespace instead.
    self.glayers.append(Bunch(geom=geom, data=data, layer=l))
```

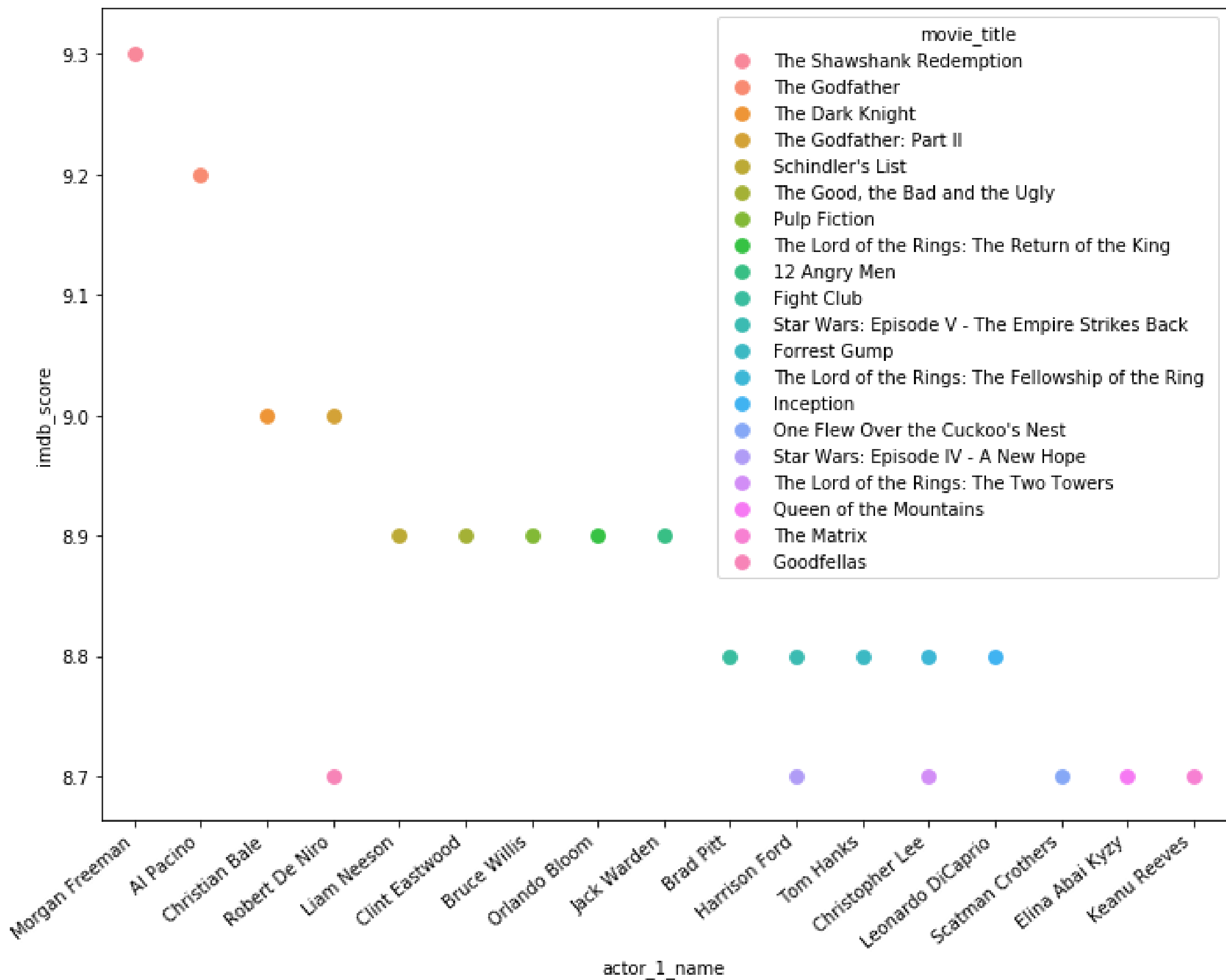


Top 20 actors of movies based on the commerical success

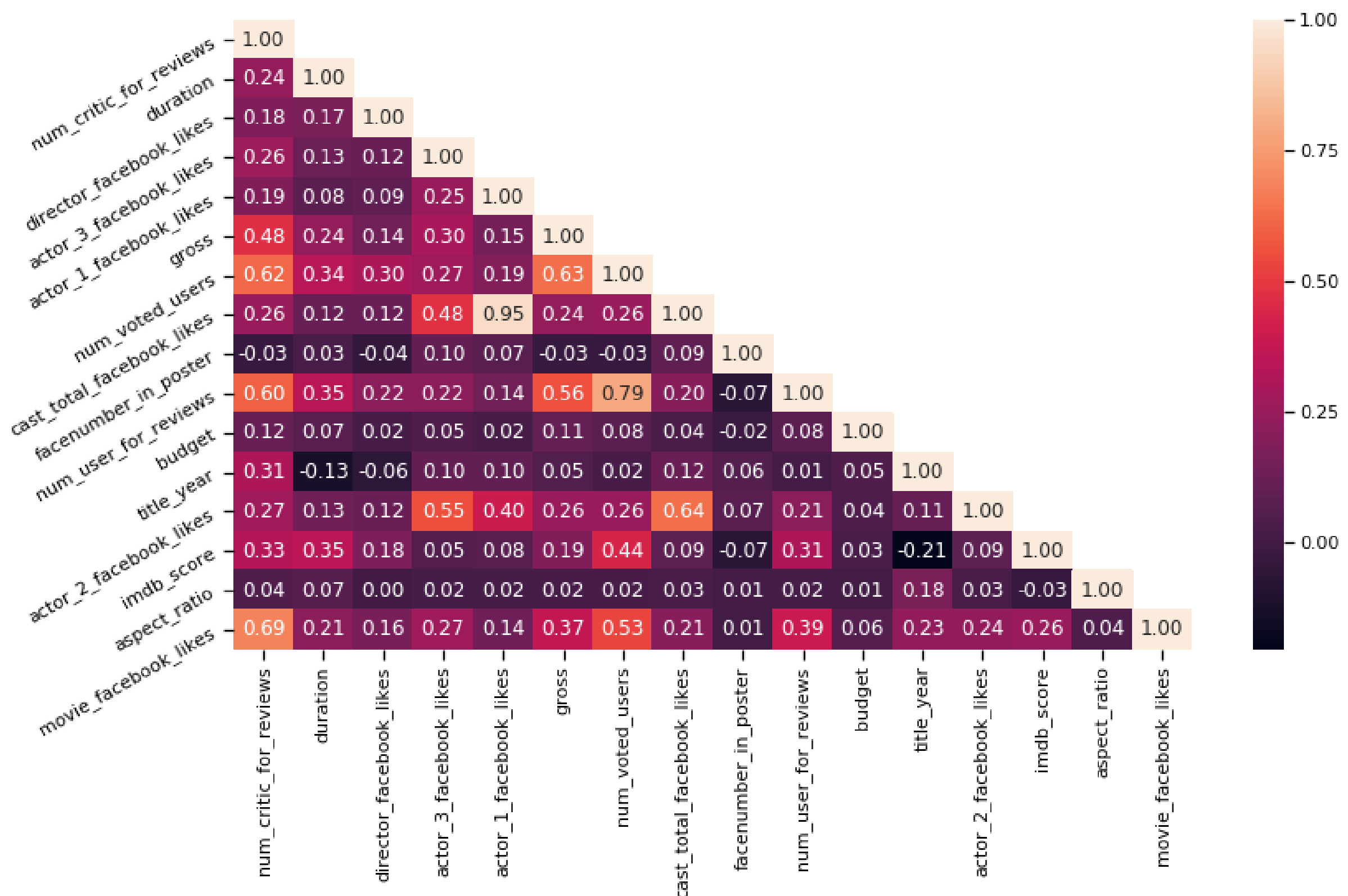
```
plt.figure(figsize=(10,8))

movie_df= movie_df.sort_values(by ='Profit_Percentage' , ascending=False)
movie_df_new=movie_df.head(20)
ax=sns.pointplot(movie_df_new['actor_1_name'], movie_df_new['Profit_Percentage'], hue=movie_df_new['movie_title'])
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")
plt.tight_layout()
plt.show()
```





```
# Correlation with heat map
import matplotlib.pyplot as plot
import seaborn as sns
corr = movie_df.corr()
sns.set_context("notebook", font_scale=1.0, rc={"lines.linewidth": 2.5})
plt.figure(figsize=(13,7))
# create a mask so we only see the correlation values once
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask, 1)] = True
a = sns.heatmap(corr, mask=mask, annot=True, fmt='.2f')
rotx = a.set_xticklabels(a.get_xticklabels(), rotation=90)
roty = a.set_yticklabels(a.get_yticklabels(), rotation=30)
```

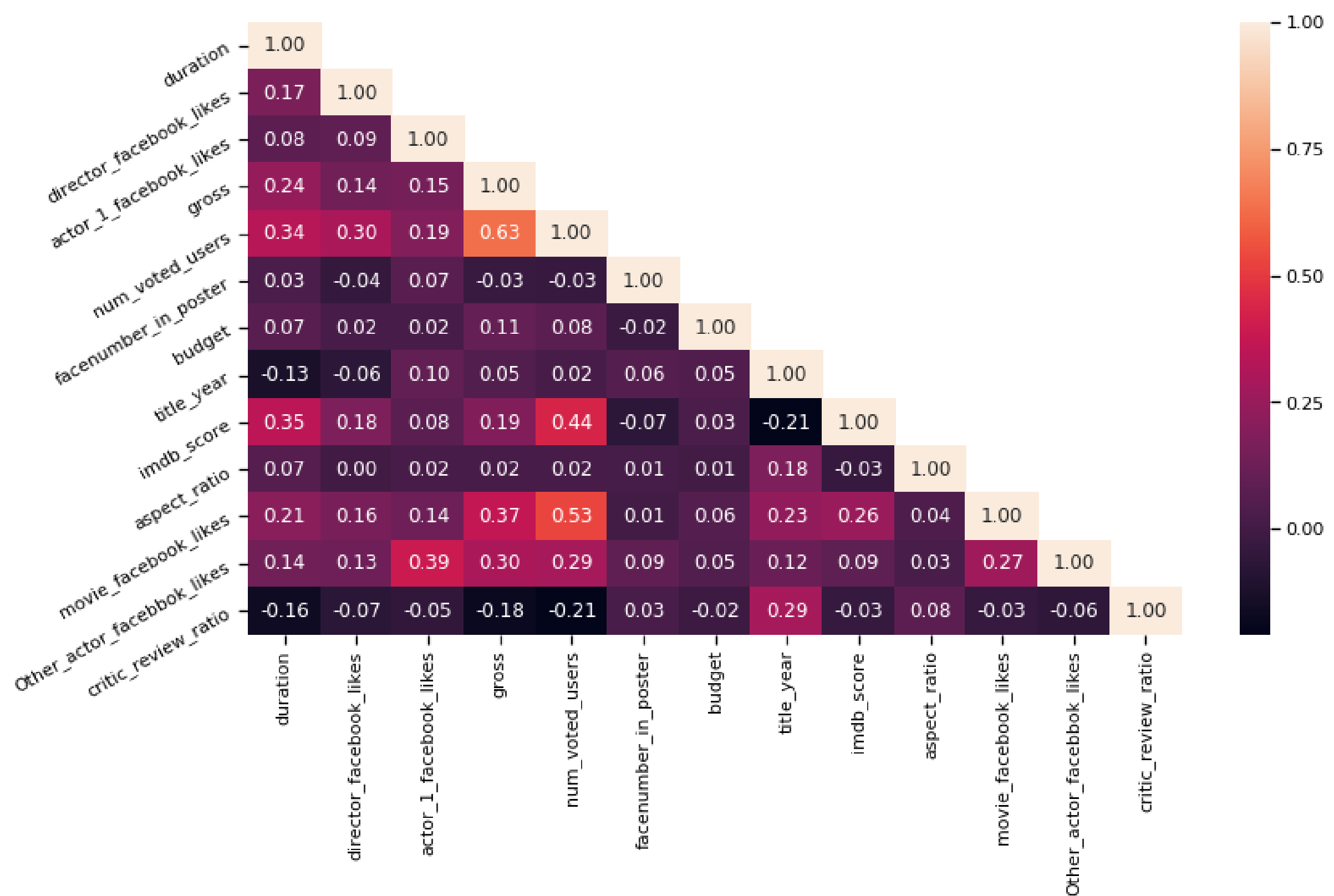


We can see that the cast_total_facebook_likes and actor_1_facebook_like are highly correlated to each other. Both actor2 and actor3 are also somehow correlated to the total. So we want to modify them into two variables: actor_1_facebook_likes and other_actors_facebook_likes.

There are high correlations among num_voted_users, num_user_for_reviews and num_critic_for_reviews. We want to keep num_voted_users and take the ratio of num_user_for_reviews and num_critic_for_reviews.

New Correlation matrix shown in the figure

```
import matplotlib.pyplot as plot
import seaborn as sns
corr = movie_df.corr()
sns.set_context("notebook", font_scale=1.0, rc={"lines.linewidth": 2.5})
plt.figure(figsize=(13,7))
# create a mask so we only see the correlation values once
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask, 1)] = True
a = sns.heatmap(corr,mask=mask, annot=True, fts='.2f')
rotx = a.set_xticklabels(a.get_xticklabels(), rotation=90)
roty = a.set_yticklabels(a.get_yticklabels(), rotation=30)
```

```

from sklearn.metrics import classification_report

print('Logistic Reports\n',classification_report(y_test, y_pred))
print('KNN Reports\n',classification_report(y_test, knnpred))
print('SVC Reports\n',classification_report(y_test, svcpred))
print('Naive BayesReports\n',classification_report(y_test, gaussiannbpred))
print('Decision Tree Reports\n',classification_report(y_test, dtreepred))
print('Ada Boosting\n',classification_report(y_test, abcl_pred))
print('Random Forests Reports\n',classification_report(y_test, rfcpred))
print('Bagging Clasifier',bgcl.oob_score_)
print('Gradient Boosting',classification_report(y_test, test_pred))
print('XGBoosting\n',classification_report(y_test, xgbprd))
logistic

```

Reports

	precision	recall	f1-score	support
1	0.00	0.00	0.00	46
2	0.50	0.25	0.33	378
3	0.72	0.92	0.81	924
4	0.84	0.52	0.65	61
accuracy			0.69	1409
macro avg	0.52	0.42	0.45	1409
weighted avg	0.64	0.69	0.65	1409

KNN Reports

	precision	recall	f1-score	support
1	0.00	0.00	0.00	46

2	0.46	0.41	0.44	378
3	0.73	0.83	0.78	924
4	1.00	0.20	0.33	61

accuracy			0.67	1409
macro avg	0.55	0.36	0.39	1409
weighted avg	0.64	0.67	0.64	1409

SVC Reports

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

1	0.14	0.02	0.04	46
2	0.42	0.42	0.42	378
3	0.74	0.79	0.76	924
4	0.57	0.33	0.42	61

accuracy			0.64	1409
macro avg	0.47	0.39	0.41	1409
weighted avg	0.62	0.64	0.63	1409

Naive BayesReports

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

1	0.05	0.91	0.09	46
2	0.50	0.00	0.01	378
3	0.71	0.01	0.01	924
4	0.11	0.85	0.19	61

accuracy			0.07	1409
macro avg	0.34	0.44	0.07	1409
weighted avg	0.61	0.07	0.02	1409

Decision Tree Reports

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

1	0.11	0.11	0.11	46
2	0.47	0.51	0.49	378
3	0.78	0.76	0.77	924
4	0.77	0.59	0.67	61

accuracy			0.67	1409
macro avg	0.53	0.49	0.51	1409
weighted avg	0.67	0.67	0.67	1409

Ada Boosting

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

1	0.17	0.15	0.16	46
2	0.46	0.51	0.48	378
3	0.77	0.75	0.76	924
4	0.65	0.51	0.57	61

accuracy			0.65	1409
macro avg	0.51	0.48	0.49	1409
weighted avg	0.66	0.65	0.66	1409

Random Forests Reports

	precision	recall	f1-score	support
1	1.00	0.04	0.08	46
2	0.62	0.47	0.53	378
3	0.77	0.92	0.84	924
4	0.96	0.44	0.61	61
accuracy			0.75	1409
macro avg	0.84	0.47	0.52	1409
weighted avg	0.75	0.75	0.72	1409

Bagging Clasifier 0.7429179978700745

Gradient Boosting

	precision	recall	f1-score	support
1	0.25	0.02	0.04	46
2	0.60	0.56	0.58	378
3	0.80	0.88	0.84	924
4	0.86	0.49	0.62	61
accuracy			0.75	1409
macro avg	0.63	0.49	0.52	1409
weighted avg	0.73	0.75	0.74	1409

XGBoosting

	precision	recall	f1-score	support
1	0.25	0.02	0.04	46
2	0.59	0.52	0.55	378
3	0.79	0.89	0.84	924
4	0.89	0.54	0.67	61
accuracy			0.75	1409
macro avg	0.63	0.49	0.53	1409
weighted avg	0.72	0.75	0.73	1409

opt/conda/lib/python3.6/site-packages/sklearn/metrics/classification.py:1437: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.

'precision', 'predicted', average, warn_for

10.Conclusion

The conclusion is that Random Forest Algorithm along with the gradient boosting have the accuracy of 74.5 and 75.5 respectively

Done by:

NAME: Shaik. Asif

NM ID: AU720921244047

JCT COLLEGE OF ENGINEERING