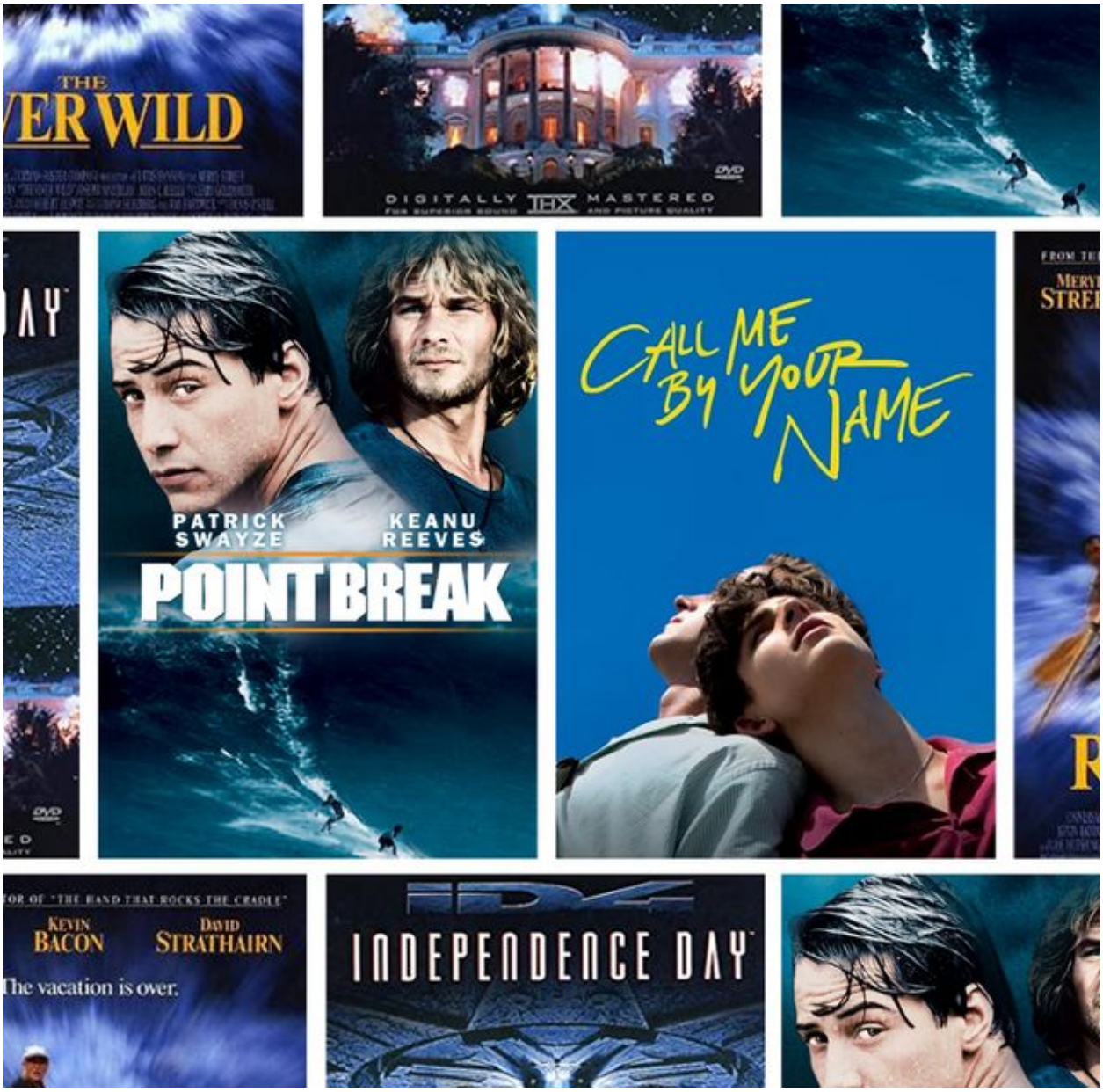
Project Report
on
Data Analysis and Modeling of IMDB Movie Rating Dataset



By:
Rajat Dua

1 Introduction

1.1 Background

A commercial success movie not only entertains the audience, but also enables film companies to gain tremendous profit. A lot of factors such as good directors and experienced actors are considered for creating good movies. However, famous directors and actors can always bring an expected box-office income but cannot guarantee a highly rated IMDb score.

1.2 Objective

The objective of this project is as follows:

- Statistically analyze and interpret movie_metadata.csv
- Perform Exploratory Data Analysis on IMDb movies data to understand the factors that make a movie more successful than others and also to analyze what kind of movies get a higher IMDB score and therefore are more successful
- Apply different Machine Learning Algorithms in order to determine which algorithm best able to capture variance in the data and also serve as a useful tool to predict future outcome based on current data

In this project, we take imdb_score as the response variable and focus on operating predictions by analyzing the rest of variables in the IMDb movie data set. The results can help film companies to understand the secret of generating a commercial and/or critically successful movie.

1.3 Data Description

The data set contains 28 columns for 5043 movies, spanning across 100 years in 66 countries. There are 2399 unique director names, and thousands of actors/actresses.

```
df.shape
```

```
(5043, 28)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 28 columns):
color                      5024 non-null object
director_name              4939 non-null object
num_critic_for_reviews     4993 non-null float64
duration                   5028 non-null float64
director_facebook_likes    4939 non-null float64
actor_3_facebook_likes     5020 non-null float64
actor_2_name               5030 non-null object
actor_1_facebook_likes     5036 non-null float64
gross                      4159 non-null float64
genres                     5043 non-null object
actor_1_name               5036 non-null object
movie_title                5043 non-null object
num_voted_users            5043 non-null int64
cast_total_facebook_likes  5043 non-null int64
actor_3_name               5020 non-null object
facenumber_in_poster       5030 non-null float64
plot_keywords              4890 non-null object
movie_imdb_link            5043 non-null object
num_user_for_reviews       5022 non-null float64
language                   5031 non-null object
country                    5038 non-null object
content_rating             4740 non-null object
budget                     4551 non-null float64
title_year                 4935 non-null float64
actor_2_facebook_likes     5030 non-null float64
imdb_score                 5043 non-null float64
aspect_ratio               4714 non-null float64
movie_facebook_likes       5043 non-null int64
dtypes: float64(13), int64(3), object(12)
```

```
df.describe()
```

	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_1_facebook_likes	gross	num_voted_users	cast_total_f
count	4993.000000	5028.000000	4939.000000	5020.000000	5036.000000	4.159000e+03	5.043000e+03	
mean	140.194272	107.201074	686.509212	645.009761	6560.047061	4.846841e+07	8.366816e+04	
std	121.601675	25.197441	2813.328607	1665.041728	15020.759120	6.845299e+07	1.384853e+05	
min	1.000000	7.000000	0.000000	0.000000	0.000000	1.620000e+02	5.000000e+00	
25%	50.000000	93.000000	7.000000	133.000000	614.000000	5.340988e+06	8.593500e+03	
50%	110.000000	103.000000	49.000000	371.500000	988.000000	2.551750e+07	3.435900e+04	
75%	195.000000	118.000000	194.500000	636.000000	11000.000000	6.230944e+07	9.630900e+04	
max	813.000000	511.000000	23000.000000	23000.000000	640000.000000	7.605058e+08	1.689764e+06	

We are interested in analyzing the effect of variables in predicting the IMDb score of the movies. Therefore, the column “imdb_score” is the response variable while the other 27 variables are possible predictors. The response variable “imdb_score” is numerical, and the predictors are mixed with numerical and categorical variables. Our main goal is to estimate the predicting power of certain variables on the IMDb ratings of the movies.

2 Data Cleaning

2.1 Missing Values

The data set movie_metadeta.csv contains a lot of missing values in multiple category for multiple movies. Let us look at the columns where there are null values.

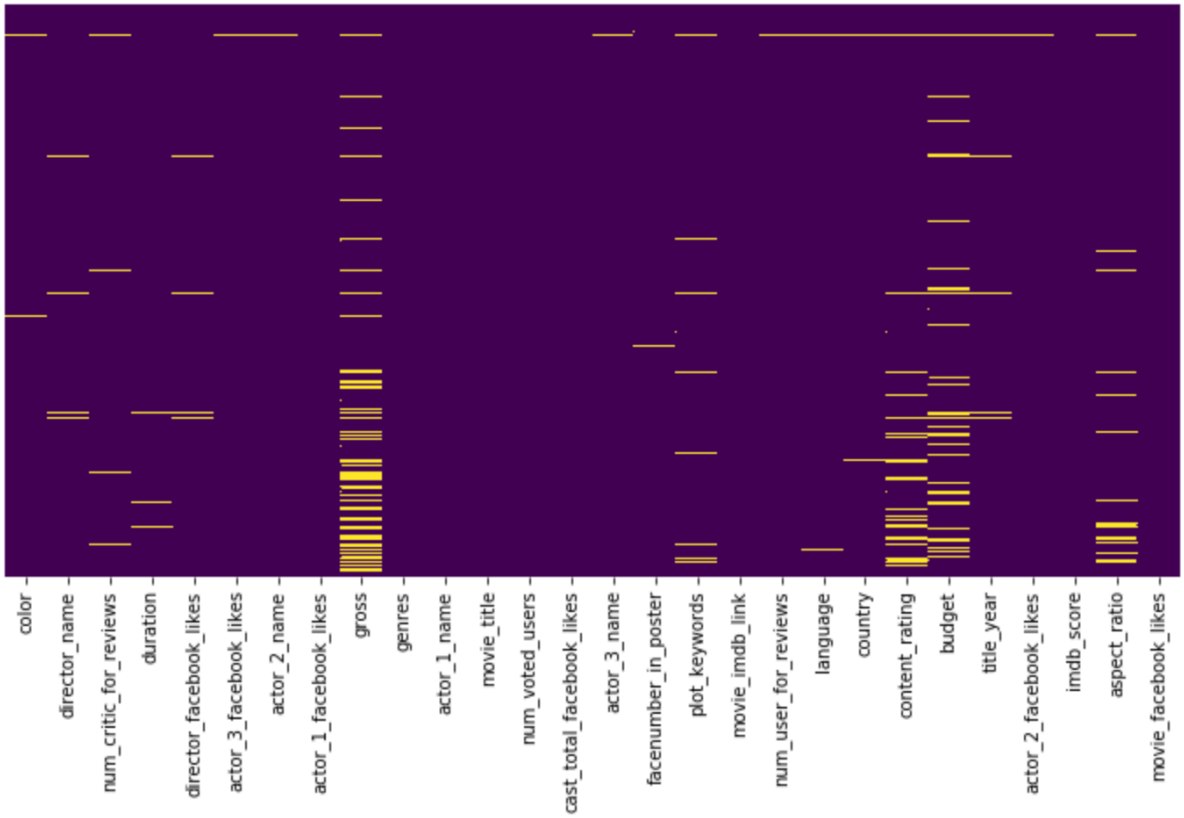
```
df.isnull().sum()
```

```
color                       19
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
movie_imdb_link              0
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
```

This table shows the number of missing values present in each column of the dataset. Let’s plot a heatmap to visualize missing values.

```
plt.figure(figsize = (12,6))
sns.heatmap(df.isnull(), yticklabels = False, cbar = False, cmap = 'viridis')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc5e1f9d2e8>
```



This heatmap shows the location of the missing values present in multiple columns of the data set. The yellow lines represent the cells containing missing values in the data. Our first job is to handle these missing values without sacrificing much of the rows in the data and therefore increasing the predictive power of the machine learning models on the data set.

2.1.1 Handling Duplicate Data

In the IMDb data set, there are multiple rows with duplicate data. We want to remove these row and keep the unique ones.

```
dups=df.duplicated()
df[dups].head()
```

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross
137	Color	David Yates	248.0	110.0	282.0	103.0	Alexander Skarsgård	11000.0	124051759.0
187	Color	Bill Condon	322.0	115.0	386.0	12000.0	Kristen Stewart	21000.0	292298923.0
204	Color	Hideaki Anno	1.0	120.0	28.0	12.0	Shin'ya Tsukamoto	544.0	NaN
303	Color	Joe Wright	256.0	111.0	456.0	394.0	Cara Delevingne	20000.0	34964818.0
389	Color	Josh Trank	369.0	100.0	128.0	78.0	Reg E. Cathey	596.0	56114221.0

5 rows × 28 columns

```
df[dups].shape[0]
```

```
45
```

```
df.drop_duplicates(inplace = True)
```

After removing the duplicates, we have 4998 observations left.

2.1.2 Delete some Rows

Since gross and budget have too many missing values, and we want to keep these two variables for the following analysis, we can only delete rows with null values for gross and budget because imputation will not do a good job here.

```
df = df.dropna(subset = ['gross', 'budget'])
```

```
df.shape
```

```
(3857, 28)
```

So, we have omitted about 20% of our observations and now the data has 3857 observations.

2.1.3 Analyze Aspect Ratio

Aspect ratio column has a lot of missing values. Before trying to impute the missing values, we want to check how important is this variable.

```
df['aspect_ratio'].value_counts(dropna = False)
```

```
2.35     1995
1.85     1600
NaN        74
1.37       50
1.78       41
1.66       40
1.33       19
2.39       11
2.20       10
2.40        3
2.76        3
2.00        3
1.75        2
1.18        1
2.24        1
1.77        1
16.00       1
2.55        1
1.50        1
Name: aspect_ratio, dtype: int64
```

The most common aspect ratios are 2.35 and 1.85. For analyzing purposes, we group other ratios together. In order to compute the mean of IMDb score for different aspect_ratio, we need to replace NA with 0 first.

```
df.fillna({'aspect_ratio': 0.00},inplace = True)
```

```
df['imdb_score'][df['aspect_ratio'] == 2.35].mean()
```

```
6.508471177944862
```

```
df['imdb_score'][df['aspect_ratio'] == 1.85].mean()
```

```
6.373937500000018
```

```
df['imdb_score'][(df['aspect_ratio'] != 1.85) & (df['aspect_ratio'] != 2.35)].mean()
```

```
6.672519083969459
```

Average IMDB scores varies from 6.3 to 6.7. Therefore, aspect ratio does not have much effect on IMDb score. So, we can remove this column.

```
df = df.drop('aspect_ratio', axis = 1)
```

2.1.4 Deal with NaN values in numerical variables

There are some NaN values in the numerical variable columns which we will replace by respective column mean.

```
df.fillna({'num_critic_for_reviews': df['num_critic_for_reviews'].mean(),
           'duration': df['duration'].mean(),
           'director_facebook_likes': df['director_facebook_likes'].mean(),
           'actor_3_facebook_likes' : df['actor_3_facebook_likes'].mean(),
           'actor_1_facebook_likes': df['actor_1_facebook_likes'].mean(),
           'actor_2_facebook_likes': df['actor_2_facebook_likes'].mean(),
           'facenumber_in_poster': df['facenumber_in_poster'].mean()
          },inplace = True)
```

Now we have finished imputing the numeric missing values. There are still some categorical missing values, which we will take a look.

2.1.5 Analyzing Content Rating

There are still some missing values in the content rating column. We drop the rows containing these missing data since these cannot be replaced with reasonable data.

```
df.dropna(subset = ['content_rating'], inplace=True)
```

```
df['content_rating'].value_counts(dropna = False)
```

```
R             1723
PG-13         1314
PG             573
G               91
Not Rated       42
Unrated         24
Approved        17
X               10
NC-17            6
Passed           3
M                2
GP               1
Name: content_rating, dtype: int64
```

According to the naming convention, we find that M = GP = PG, and X = NC-17. Therefore, we want to replace M and GP with PG, replace X with NC-17, because these two are what we use nowadays. Also, we want to replace “Approved”, “Not Rated”, “Passed”, “Unrated” with the most common rating “R”.

```
df.replace({'content_rating': {'M':'PG', 'GP': 'PG', 'Approved': 'R',
                               'Passed': 'R', 'Unrated': 'R', 'Not Rated': 'R',
                               'X': 'NC-17'}}, inplace=True)
```

```
df['content_rating'].value_counts(dropna = False)
```

```
R         1809
PG-13     1314
PG         576
G           91
NC-17       16
Name: content_rating, dtype: int64
```

Now we have only 5 different content ratings.

2.2 Add Columns

We have gross and budget information. So, let’s add two columns, that are, profit and percentage return on investment for further analysis.

```
df['profit'] = df['gross'] - df['budget']
df['return_on_investment'] = (df['profit']/df['budget'])*100
```

Profit column is defined as the difference between gross and budget column values and the Return on Investment column is defines as the ratio of profit and budget in percentage.

2.3 Remove Columns

Now we will determine which variables are significant in predicting the IMDb score of a movie.

2.3.1 Is the color definition of the movie predictive?

To answer this question, we first determine the count of each values in the color column, as follows:

```
df['color'].value_counts(dropna = False)
```

```
Color               3680
 Black and White     124
NaN                    2
Name: color, dtype: int64
```

Almost 96% of the movies are colored, so, color has no predictive power for IMDb score. Therefore, we can remove color column.

2.3.2 Is language an important predictor of IMDb score?

First, we determine the value counts of language column, as follows:

```
df['language'].value_counts(dropna = False)
```

```
English       3644
French          34
Spanish         24
Mandarin        14
German          11
Japanese        10
Cantonese        7
Italian          7
Korean           5
Portuguese       5
Hindi            5
Norwegian        4
Thai             3
Danish           3
Dutch            3
Persian          3
Aboriginal       2
NaN              2
Hebrew           2
Indonesian       2
Dari             2
Russian          1
Zulu             1
Mongolian        1
Kazakh           1
Czech            1
Arabic           1
Romanian         1
None             1
Aramaic          1
Maya             1
Vietnamese       1
Filipino         1
Bosnian          1
Hungarian        1
Name: language, dtype: int64
```

More than 90% of the movies are in English language, therefore, language has no predictive power for IMDb score. We can remove the language column.

2.3.3 Is country an important predictor of IMDb score?

We first determine the value counts of country column, as follows:

```
df['country'].value_counts(dropna = False)
```

```
USA               3025
UK                 316
France             103
Germany             79
Canada              63
Australia           40
Spain               22
Japan               15
China               13
Hong Kong           13
New Zealand         11
Italy               11
Mexico              10
Denmark              9
South Korea          8
Ireland              7
India                5
Brazil               5
Iran                 4
Thailand             4
Norway               4
Czech Republic       3
Russia               3
Argentina            3
Netherlands          3
South Africa         3
Taiwan               2
Romania              2
Israel               2
Hungary              2
Iceland              1
Official site        1
Indonesia            1
Afghanistan          1
Poland               1
Greece               1
Belgium              1
Chile                1
```

Around 79% of the movies are from the USA, 8% from the UK and the rest 13% are from other countries. So, we group other countries together to make this categorical variable with less value counts, as follows:

```
df.loc[(df['country'] != 'USA') & (df['country'] != 'UK'), 'country'] = 'Others'
```

```
df['country'].value_counts()
```

```
USA       3025
Others     465
UK         316
Name: country, dtype: int64
```

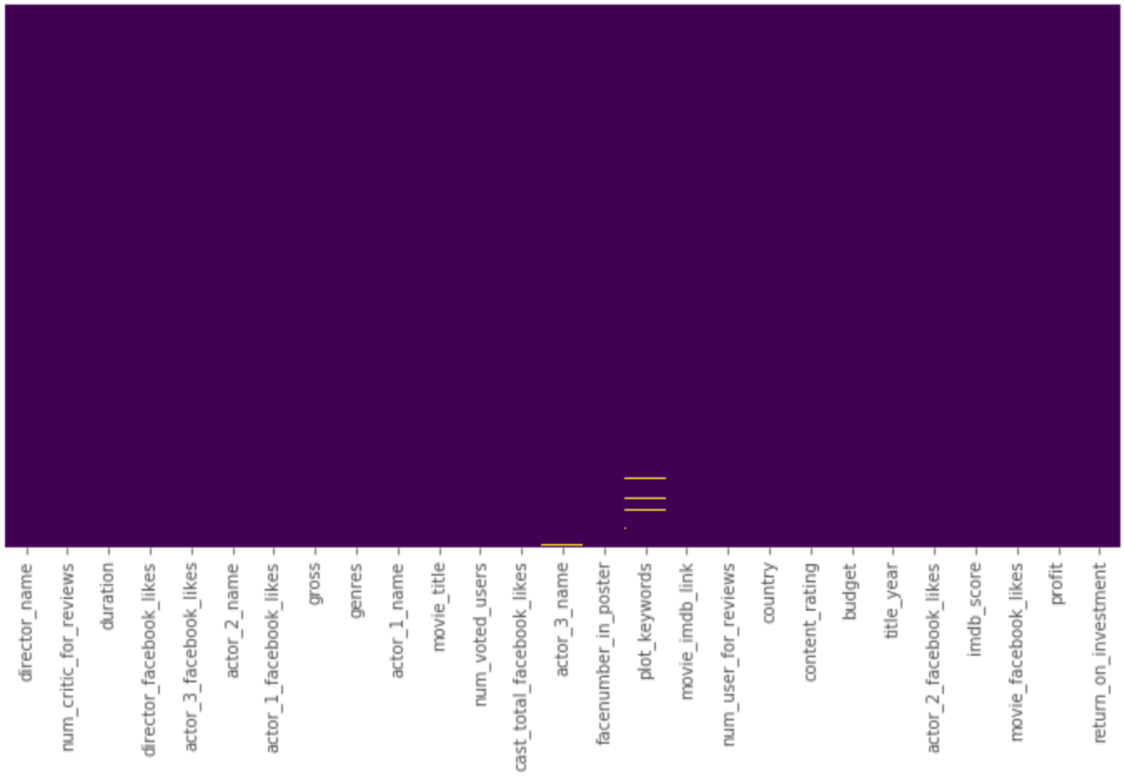
Therefore, the data is now almost cleaned off any missing values.

2.4 Checking for Missing Values in cleaned data

Since we have almost cleaned the data, we will now check for any missing values in the dataset using the heatmap as follows:

```
plt.figure(figsize = (12,6))
sns.heatmap(df.isnull(), yticklabels = False, cbar = False, cmap = 'viridis')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc5cca61e48>
```




```
df.shape
```

```
(3806, 27)
```

Now, it can be shown that our data has only a few remaining missing data in plot_keywords and actor_3_name column. These variables are categorical variables which does not seem to have predicting power over IMDb score. We will drop these columns later on after performing Exploratory Data Analysis (EDA). Now our data set have 3806 observations remaining.

2.5 Analyzing Genre Column

Each record of genres is combined with a few types, which will cause the difficulty of analyzing, as follows:

```
df['genres'].head()
```

```
0    Action|Adventure|Fantasy|Sci-Fi
1           Action|Adventure|Fantasy
2          Action|Adventure|Thriller
3                   Action|Thriller
5            Action|Adventure|Sci-Fi
Name: genres, dtype: object
```

We want to know if the variable genre is related to IMDb score. We will achieve this by first dividing the string into several substrings by the separator '|', and then saving each substring along with its corresponding IMDb score in the other data frame genres.df, as follows:

```
genre_df = df[['genres', 'imdb_score']]
```

```
genre_df['Action'] = genre_df['genres'].apply(lambda x: 1 if 'Action' in x.split('|') else 0)
genre_df['Adventure'] = genre_df['genres'].apply(lambda x: 1 if 'Adventure' in x.split('|') else 0)
genre_df['Animation'] = genre_df['genres'].apply(lambda x: 1 if 'Animation' in x.split('|') else 0)
genre_df['Biography'] = genre_df['genres'].apply(lambda x: 1 if 'Biography' in x.split('|') else 0)
genre_df['Comedy'] = genre_df['genres'].apply(lambda x: 1 if 'Comedy' in x.split('|') else 0)
genre_df['Crime'] = genre_df['genres'].apply(lambda x: 1 if 'Crime' in x.split('|') else 0)
genre_df['Documentary'] = genre_df['genres'].apply(lambda x: 1 if 'Documentary' in x.split('|') else 0)
genre_df['Drama'] = genre_df['genres'].apply(lambda x: 1 if 'Drama' in x.split('|') else 0)
genre_df['Family'] = genre_df['genres'].apply(lambda x: 1 if 'Family' in x.split('|') else 0)
genre_df['Fantasy'] = genre_df['genres'].apply(lambda x: 1 if 'Fantasy' in x.split('|') else 0)
genre_df['History'] = genre_df['genres'].apply(lambda x: 1 if 'History' in x.split('|') else 0)
genre_df['Horror'] = genre_df['genres'].apply(lambda x: 1 if 'Horror' in x.split('|') else 0)
genre_df['Musical'] = genre_df['genres'].apply(lambda x: 1 if 'Musical' in x.split('|') else 0)
genre_df['Mystery'] = genre_df['genres'].apply(lambda x: 1 if 'Mystery' in x.split('|') else 0)
genre_df['Romance'] = genre_df['genres'].apply(lambda x: 1 if 'Romance' in x.split('|') else 0)
genre_df['Sci-Fi'] = genre_df['genres'].apply(lambda x: 1 if 'Sci-Fi' in x.split('|') else 0)
genre_df['Sport'] = genre_df['genres'].apply(lambda x: 1 if 'Sport' in x.split('|') else 0)
genre_df['Thriller'] = genre_df['genres'].apply(lambda x: 1 if 'Thriller' in x.split('|') else 0)
genre_df['War'] = genre_df['genres'].apply(lambda x: 1 if 'War' in x.split('|') else 0)
genre_df['Western'] = genre_df['genres'].apply(lambda x: 1 if 'Western' in x.split('|') else 0)
genre_df['Film-Noir'] = genre_df['genres'].apply(lambda x: 1 if 'Film-Noir' in x.split('|') else 0)
```

```
genre_df.head()
```

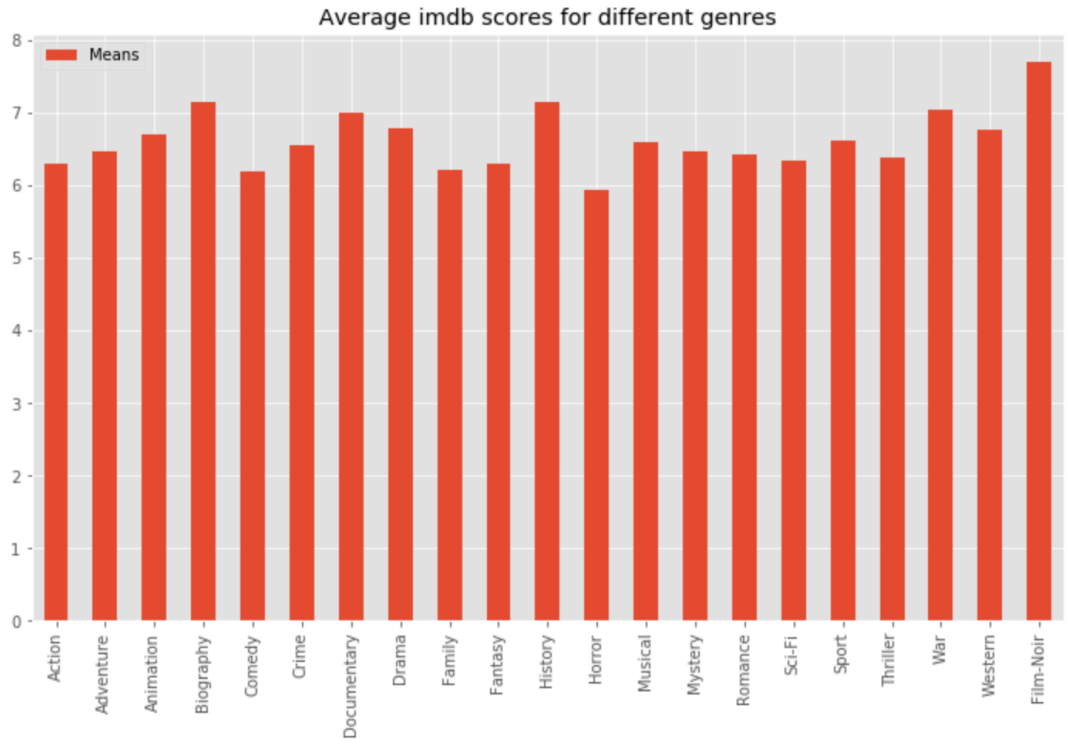
	genres	imdb_score	Action	Adventure	Animation	Biography	Comedy	Crime	Documentary	Drama	...	Horror	Musical	Mystery	I
0	Action|Adventure|Fantasy|Sci-Fi	7.9	1	1	0	0	0	0	0	0	...	0	0	0	
1	Action|Adventure|Fantasy	7.1	1	1	0	0	0	0	0	0	...	0	0	0	
2	Action|Adventure|Thriller	6.8	1	1	0	0	0	0	0	0	...	0	0	0	
3	Action|Thriller	8.5	1	0	0	0	0	0	0	0	...	0	0	0	
5	Action|Adventure|Sci-Fi	6.6	1	1	0	0	0	0	0	0	...	0	0	0	

5 rows × 23 columns

Now, we will plot a histogram for the score and genres to see if they are relative or not, as follows:

```
genre.plot(kind = 'bar', figsize = (12,7), title = 'Average imdb scores for different genres')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc5cf6485f8>
```



There isn't much difference in the averages of IMDb score across different genres, and almost all the different genres have the IMDb score averages are in the range of 6 to 8. So, the predictor "genres" can be removed because it's not really related to the IMDb score.

2.6 Analyzing the Cleaned Data

For analyzing the remaining dataset, we will divide the data into categorical and numerical databases to check to see the number of variables in both the databases, as follows:

```
numeric_data = df.select_dtypes(include = [np.number])

categorical_data = df.select_dtypes(exclude = [np.number])

print('There are {0} numerical and {1} categorical features in the data'.\
     format(numeric_data.shape[1], categorical_data.shape[1]))
```

```
There are 17 numerical and 9 categorical features in the data
```

```
categorical_data.head()
```

	director_name	actor_2_name	actor_1_name	movie_title	actor_3_name	plot_keywords	movie_imdb_link	country
0	James Cameron	Joel David Moore	CCH Pounder	Avatar	Wes Studi	avatar|future|marine|native|paraplegic	http://www.imdb.com/title/tt0499549/?ref_=fn_t...	USA
1	Gore Verbinski	Orlando Bloom	Johnny Depp	Pirates of the Caribbean: At World's End	Jack Davenport	goddess|marriage ceremony|marriage proposal|pi...	http://www.imdb.com/title/tt0449088/?ref_=fn_t...	USA
2	Sam Mendes	Rory Kinnear	Christoph Waltz	Spectre	Stephanie Sigman	bomb|espionage|sequel|spy|terrorist	http://www.imdb.com/title/tt2379713/?ref_=fn_t...	UK
3	Christopher Nolan	Christian Bale	Tom Hardy	The Dark Knight Rises	Joseph Gordon-Levitt	deception|imprisonment|lawlessness|police offi...	http://www.imdb.com/title/tt1345836/?ref_=fn_t...	USA
5	Andrew Stanton	Samantha Morton	Daryl Sabara	John Carter	Polly Walker	alien|american civil war|male nipple|mars|prin...	http://www.imdb.com/title/tt0401729/?ref_=fn_t...	USA

```
numeric_data.head()
```

	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_1_facebook_likes	gross	num_voted_users	cast_total_facebook_
0	723.0	178.0	0.0	855.0	1000.0	760505847.0	886204	
1	302.0	169.0	563.0	1000.0	40000.0	309404152.0	471220	
2	602.0	148.0	0.0	161.0	11000.0	200074175.0	275868	
3	813.0	164.0	22000.0	23000.0	27000.0	448130642.0	1144337	

Therefore, there are 17 numerical and 9 categorical features remaining in the dataset which we will use to perform Exploratory Data Analysis (EDA).

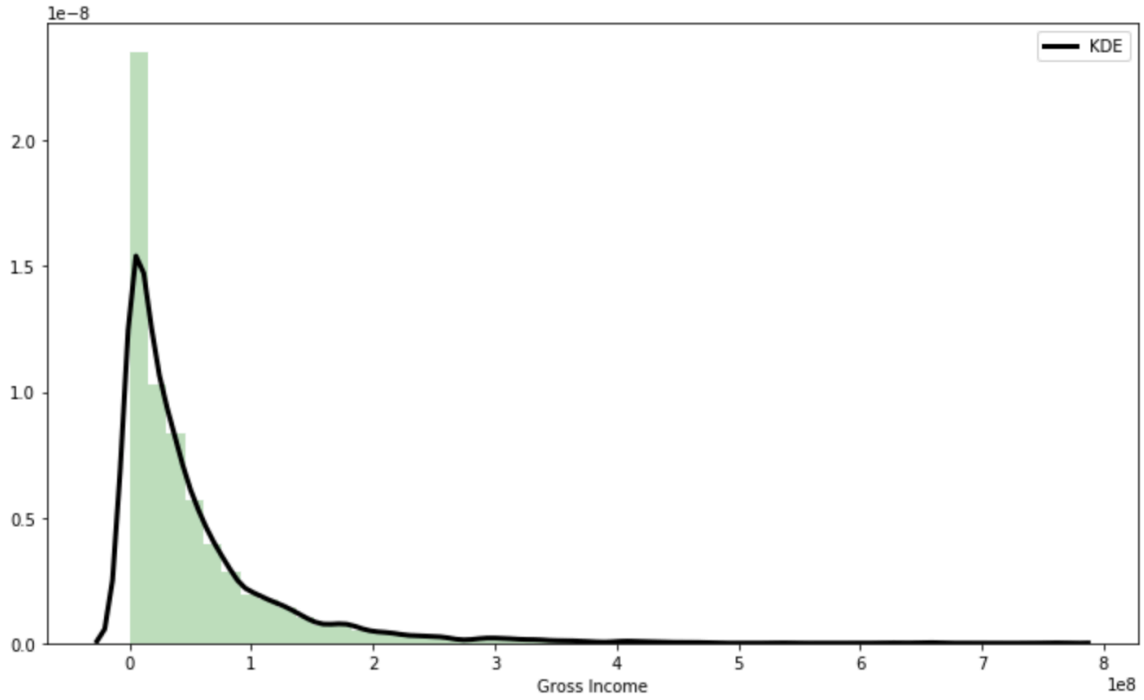
3 Exploratory Data Analysis

3.1 Distance Plot of Gross Revenue and log of Gross Revenue

Statistical analysis on a particular variable works best if the distribution of the variable is normally distributed. However, in many real-life situations, distributions are not normally distributed, and they need to be transformed using a function such as log transformation in order to exhibit normal distribution.

```
plt.figure(figsize = (12,7))
sns.distplot(df['gross'], bins = 50, kde_kws={"color": "k", "lw": 3, "label": "KDE"},
             hist_kws={"linewidth": 3, "alpha": 0.25, "color": "g"})
plt.xlabel('Gross Income')
```

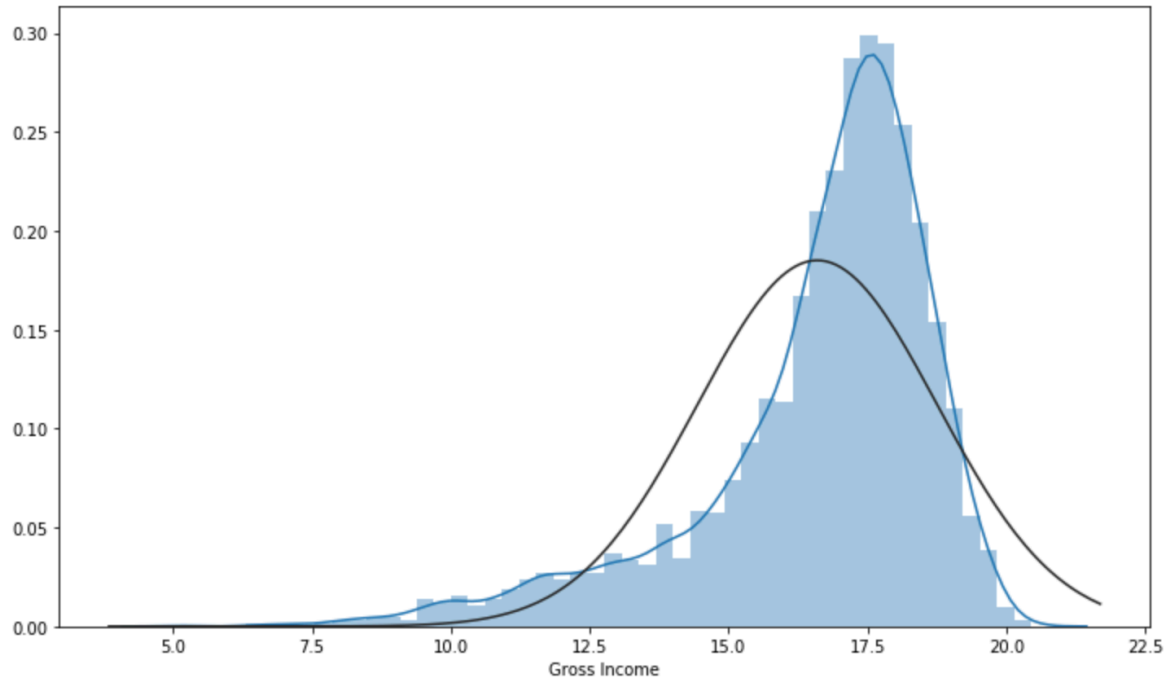
```
Text(0.5, 0, 'Gross Income')
```



This distance plot clearly depicts that Gross variable in not normally distributed and rather heavily skewed towards the right. This suggests that majority of movies have the gross income closer to the median income. There are a number of movies which have lower than the median gross revenue and only a few which have earned way more than the median gross revenue. This distribution is a perfect example of non-normal distribution. Therefore we will take log normal distribution of the gross revenue variable, as follows:

```
from scipy.stats import norm
plt.figure(figsize = (12,7))
sns.distplot(np.log(df['gross']), fit = norm)
plt.xlabel('Gross Income')
```

```
Text(0.5, 0, 'Gross Income')
```



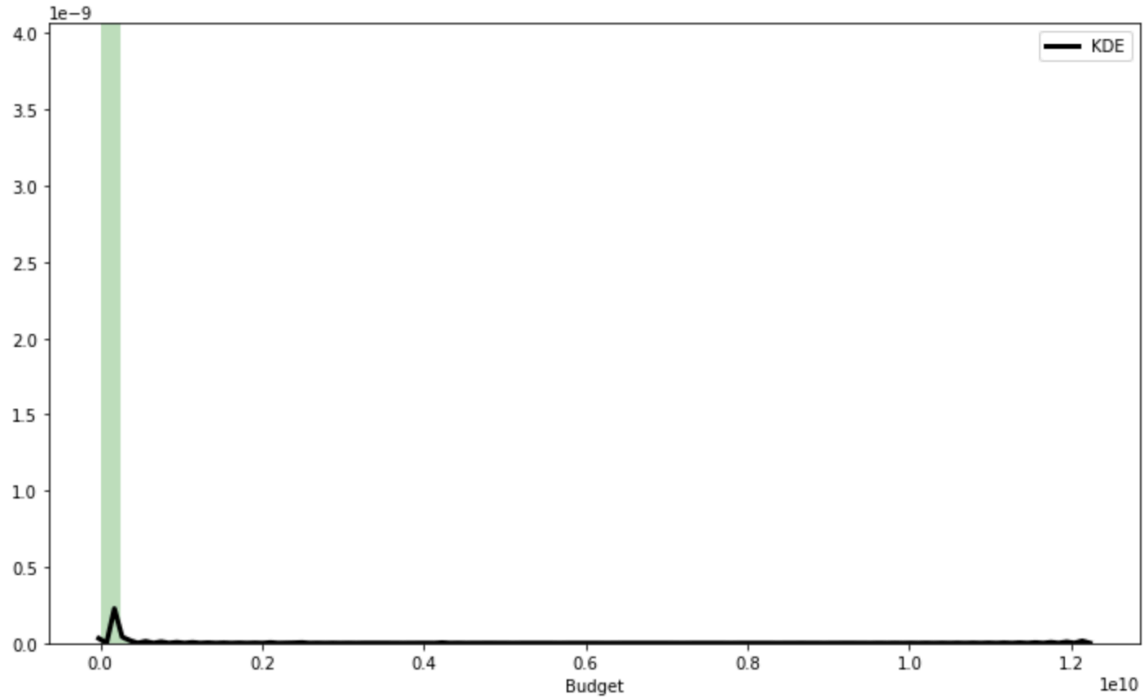
Analyzing distance plot using the log transformed price data shows that the transformed data is more likely to display normal distribution as you can see from the figure above.

3.2 Distance Plot of Budget and log of Budget

Not let us look for the distribution of the Budget column.

```
plt.figure(figsize = (12,7))
sns.distplot(df['budget'], bins = 50, kde_kws={"color": "k", "lw": 3, "label": "KDE"},
             hist_kws={"linewidth": 3, "alpha": 0.25, "color": "g"})
plt.xlabel('Budget')
```

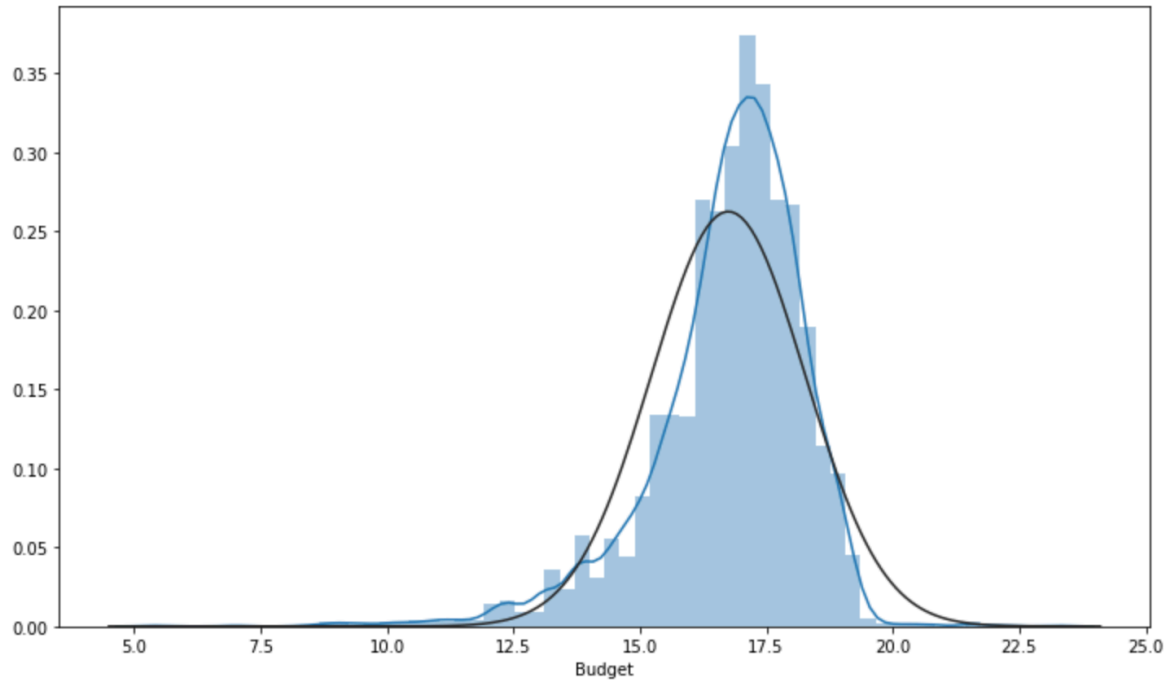
```
Text(0.5, 0, 'Budget')
```



Similar to Gross revenue column, Budget column also seem to be not normally distributed and skewed heavily towards the rights. It means that high concentration of movies has budget less than or similar to the median value and very few movies have budget way more than the median value. Let's plot distance plot for log of budget column to check if it is normally distributed, as follows:

```
from scipy.stats import norm
plt.figure(figsize = (12,7))
sns.distplot(np.log(df['budget']), bins = 60, fit = norm)
plt.xlabel('Budget')
```

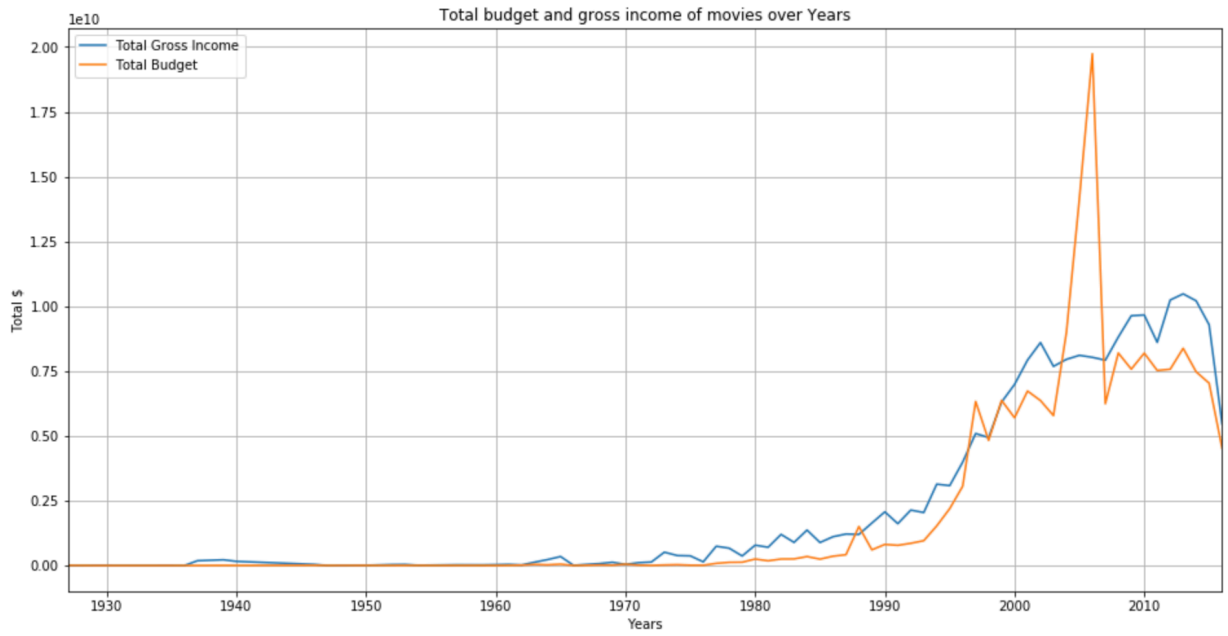
```
Text(0.5, 0, 'Budget')
```



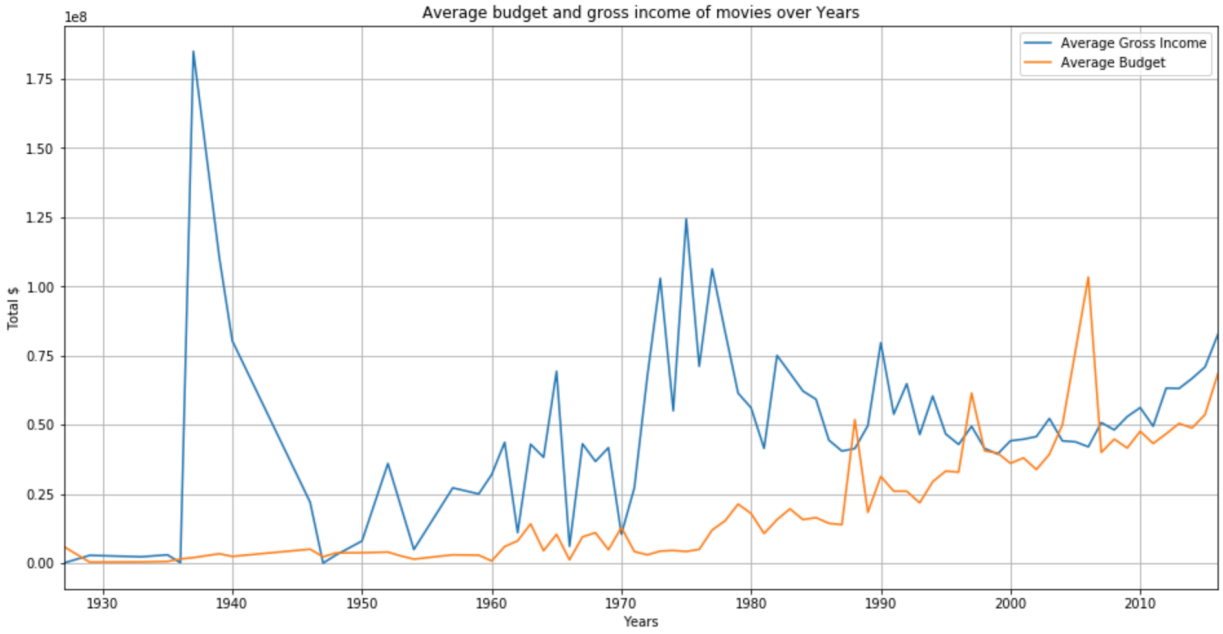
Therefore, this plot exhibits approximate normally distribution.

3.3 Variation of Sum and Average of Total Budget and Income with Time in Years

Let's plot a line graph to compare the variation of total sum of budget and gross income of all movies released over years.



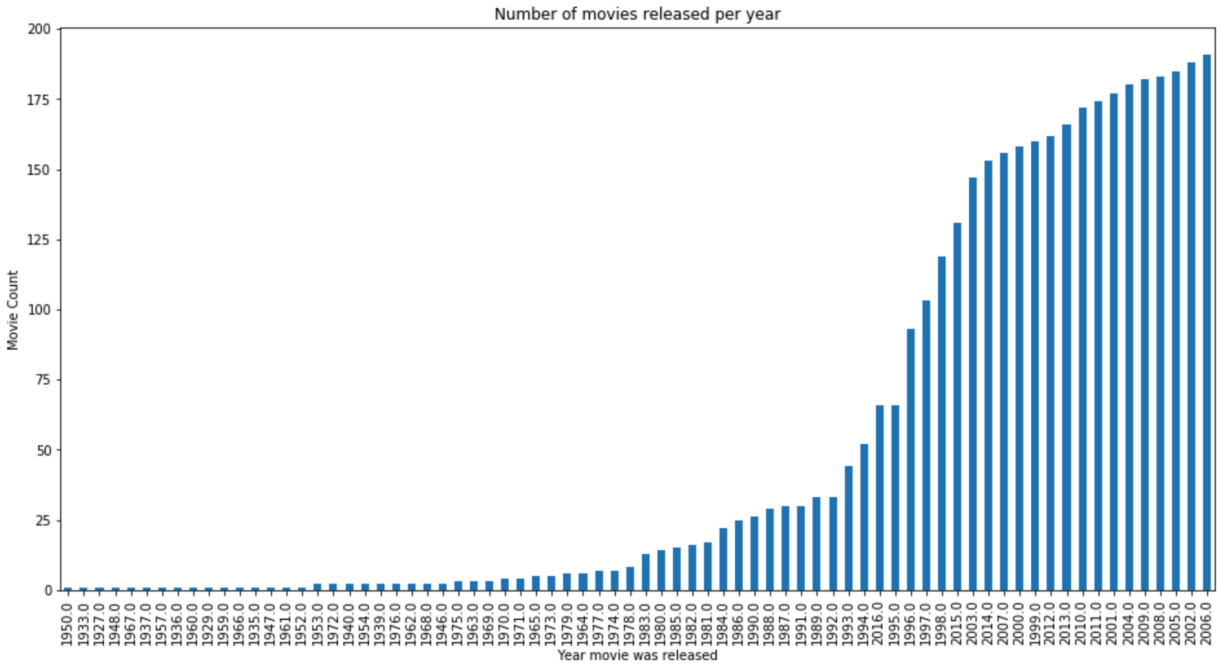
This plot clearly show that total gross income generated from movies generally exceeds the total budget of those movies over years. Therefore, movie industry generally is a profitable industry. However, during certain time periods such as around 1988, 1997 and 2005 to 2007, total budget exceeds total gross income showing the periods of loss for the industry. Let's see how the average budget and gross income varied over the years for the movie industry.



This plot also confirms that even though there are certain periods wherein the average budget exceeds average gross income, the movie industry in general is a profitable business.

3.4 Histogram of Movie Released

Movie production just exploded after year 1990. It could be due to advancement in technology and commercialization of internet. Let's plot a Histogram to check the number of movies released per year.

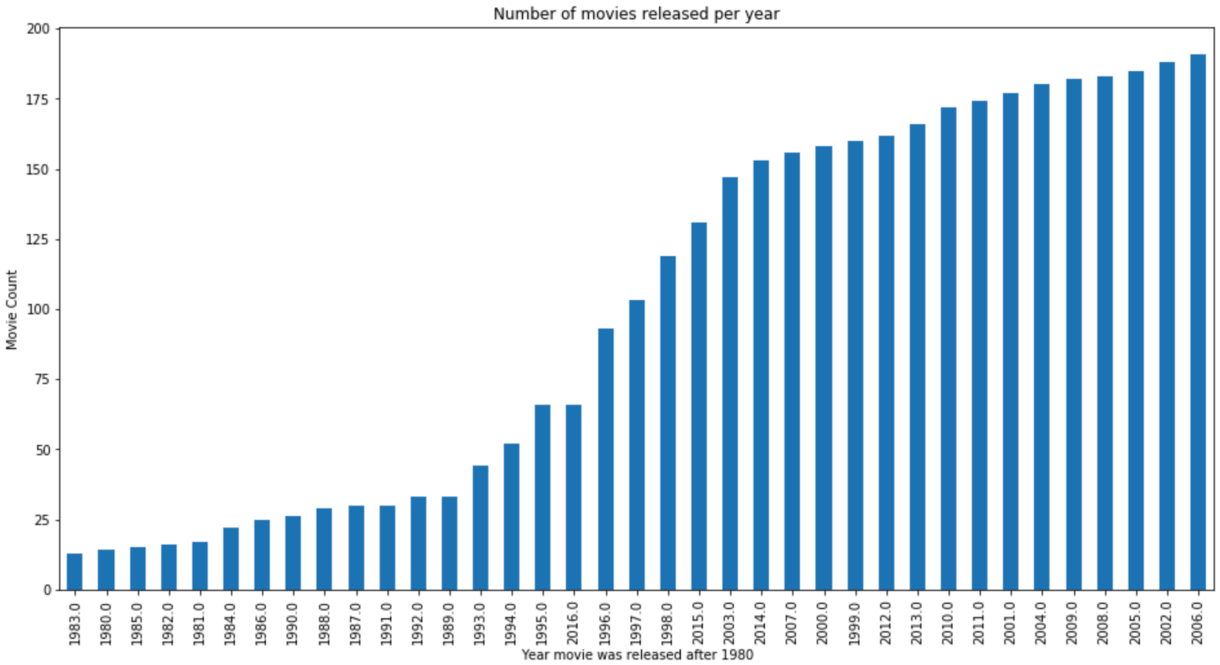


From this plot, we see that there are not many records of movies released before 1980. It's better to remove those records because they might not be representative.

```
df = df.loc[df['title_year'] >= 1980]
```

```
df['title_year'].value_counts().sort_values().plot(kind = 'bar', figsize = (16,8))
plt.title('Number of movies released per year')
plt.xlabel('Year movie was released after 1980')
plt.ylabel('Movie Count')
```

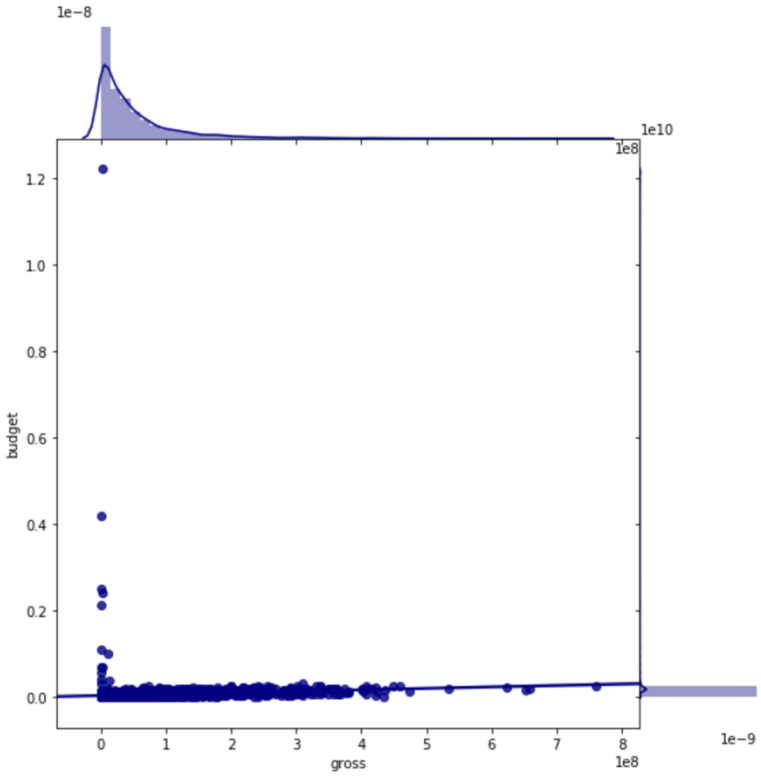
```
Text(0, 0.5, 'Movie Count')
```



Therefore, the number of movies released have increased over the years especially post 1990.

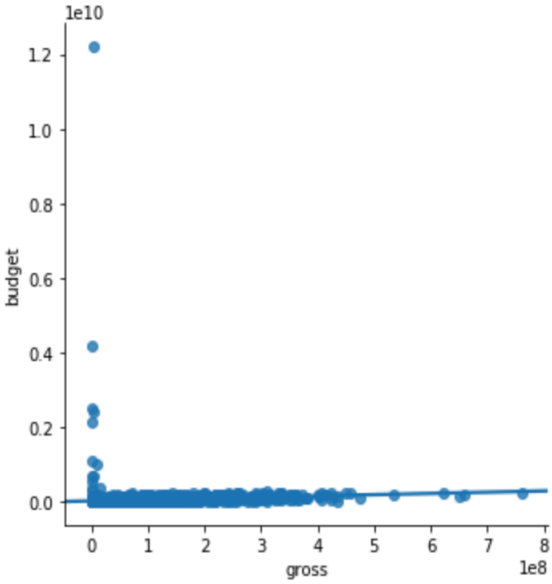
3.5 Plotting Budget vs Gross Income

In this section, we want to compare and interpret the relationship between budget and gross columns in order to determine whether higher budget movies yield higher gross income or not. Let's plot a joint plot and lmplot for budget vs gross.

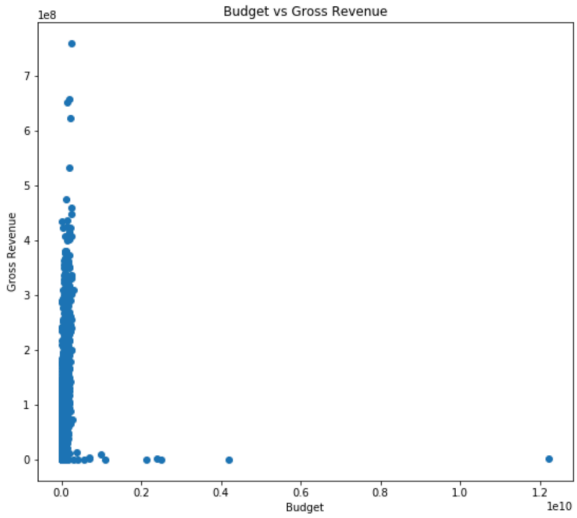


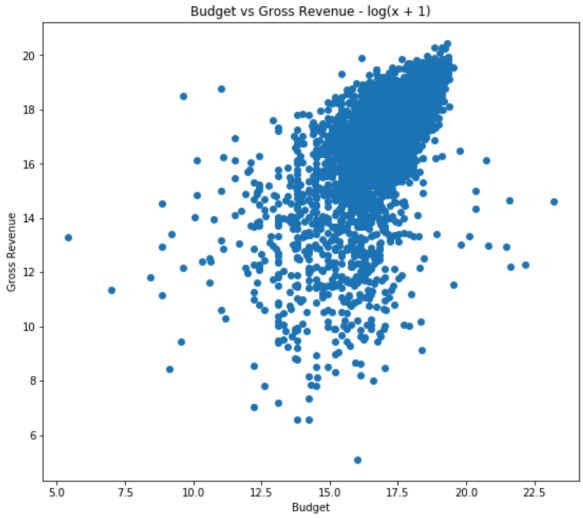

```
sns.lmplot('gross', 'budget', data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fc5e27b5a58>
```



From these plots we can see that gross income is positively related to budget, however, the positive correlation between the variables is weak. This proves that it is not necessary that high budget movie will yield higher gross income. Let's plot a scatter plot between these two variables to examine the relationship further, as follows:

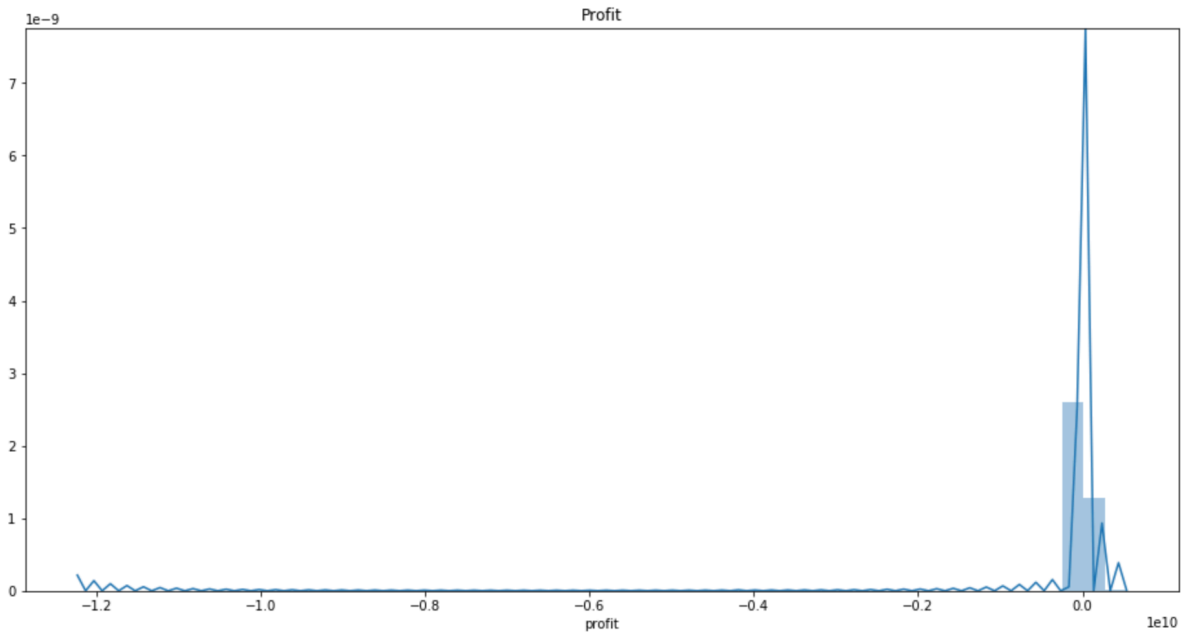




These plots confirm the weak positive relationship between budget and gross revenue variables.

3.6 Distance Plot of Profit Column

Let's plot the distance plot of the derived variable profit column, as follows:



The distance plot is skewed towards the left which means that majority of movies have profit in the range of median profit and even exceeding it. However, as expected there are some movies which faced huge losses as well. Let's check the top 10 and bottom 10 movies based on profit.

Top 10 Movies based on Profit

```
top_movies = df.sort_values(by='profit', ascending=False)
top_movies.head(10)[['movie_title', 'profit', 'budget', 'gross']]
```

	movie_title	profit	budget	gross
0	Avatar	523505847.0	237000000.0	760505847.0
29	Jurassic World	502177271.0	150000000.0	652177271.0
26	Titanic	458672302.0	200000000.0	658672302.0
3080	E.T. the Extra-Terrestrial	424449459.0	10500000.0	434949459.0
17	The Avengers	403279547.0	220000000.0	623279547.0
509	The Lion King	377783777.0	45000000.0	422783777.0
240	Star Wars: Episode I - The Phantom Menace	359544677.0	115000000.0	474544677.0
66	The Dark Knight	348316061.0	185000000.0	533316061.0
439	The Hunger Games	329999255.0	78000000.0	407999255.0
812	Deadpool	305024263.0	58000000.0	363024263.0

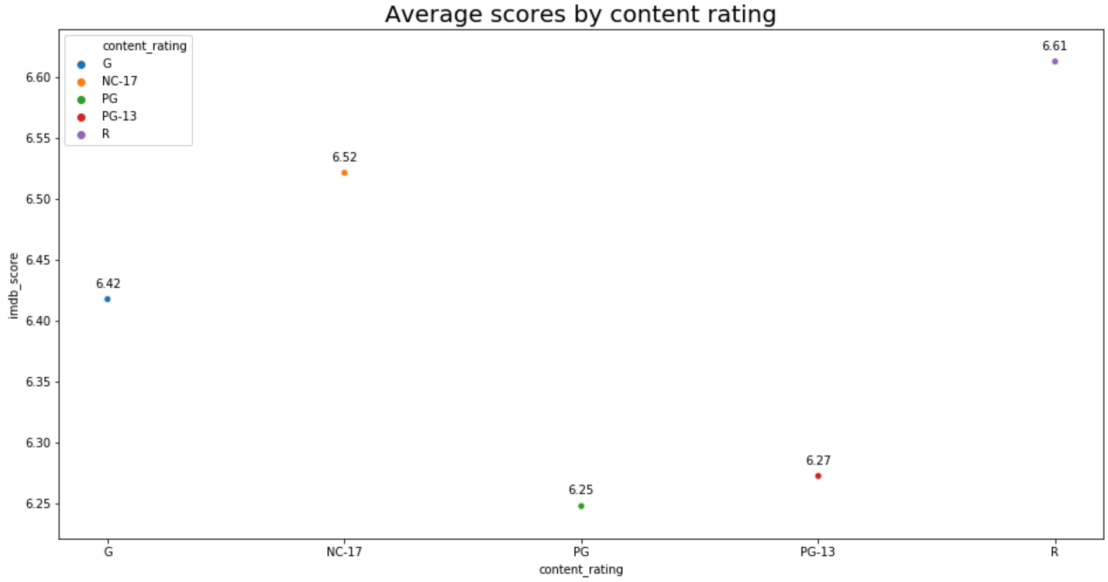
Bottom 10 Movies based on Profit

```
worst_movies = df.sort_values(by='profit', ascending=True)
worst_movies.head(10)[['movie_title', 'profit', 'budget', 'gross']]
```

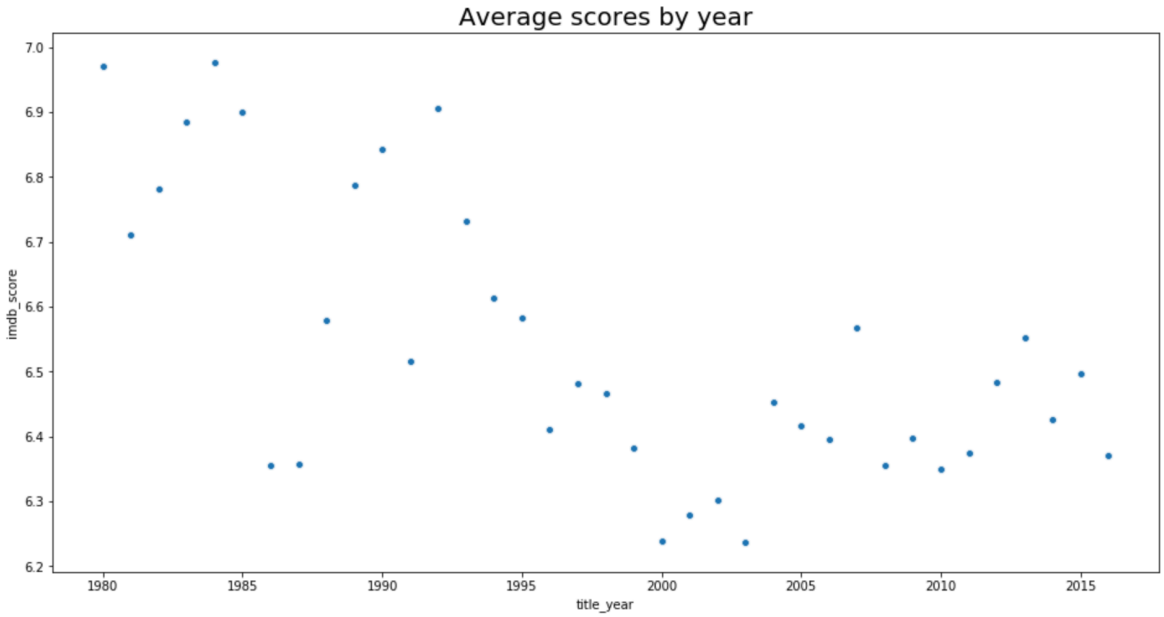
	movie_title	profit	budget	gross
2988	The Host	-1.221330e+10	1.221550e+10	2201412.0
3859	Lady Vengeance	-4.199788e+09	4.200000e+09	211667.0
3005	Fateless	-2.499804e+09	2.500000e+09	195888.0
2323	Princess Mononoke	-2.397702e+09	2.400000e+09	2298191.0
2334	Steamboy	-2.127110e+09	2.127520e+09	410388.0
3423	Akira	-1.099561e+09	1.100000e+09	439162.0
4542	Godzilla 2000	-9.899626e+08	1.000000e+09	10037390.0
3851	Tango	-6.983127e+08	7.000000e+08	1687311.0
3075	Kabhi Alvida Naa Kehna	-6.967246e+08	7.000000e+08	3275443.0
1338	Red Cliff	-5.530052e+08	5.536320e+08	626809.0

3.7 Average IMDb score by content rating and by year after 1980

In this section, we will see the average value of IMDb score per content rating and by year, as follows:



We can see that the average IMDb score vary in the range of 6.25 to 6.61, which is not much variation.



When we plot average IMDb score against years we see that even though the variation is only from 6.2 to 7.0, there is much variation in the score per year. This shows that there are some years where the quality of movies released is high and there are some years when the quality is low.

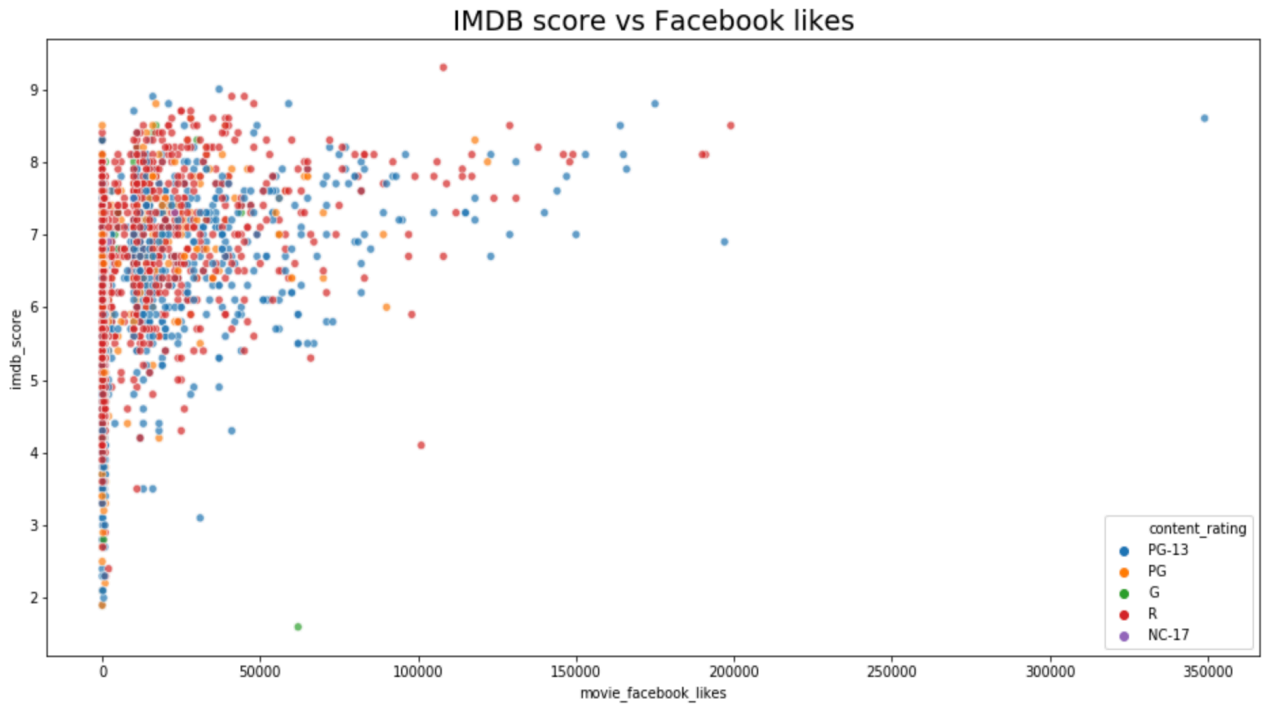
3.8 Top 20 directors with highest average IMDB score

There are some popular directors in the industry which are known to direct quality movies, and are therefore have a higher average IMDb score for their movies. Top 20 directors with highest average IMDB score are as follows:

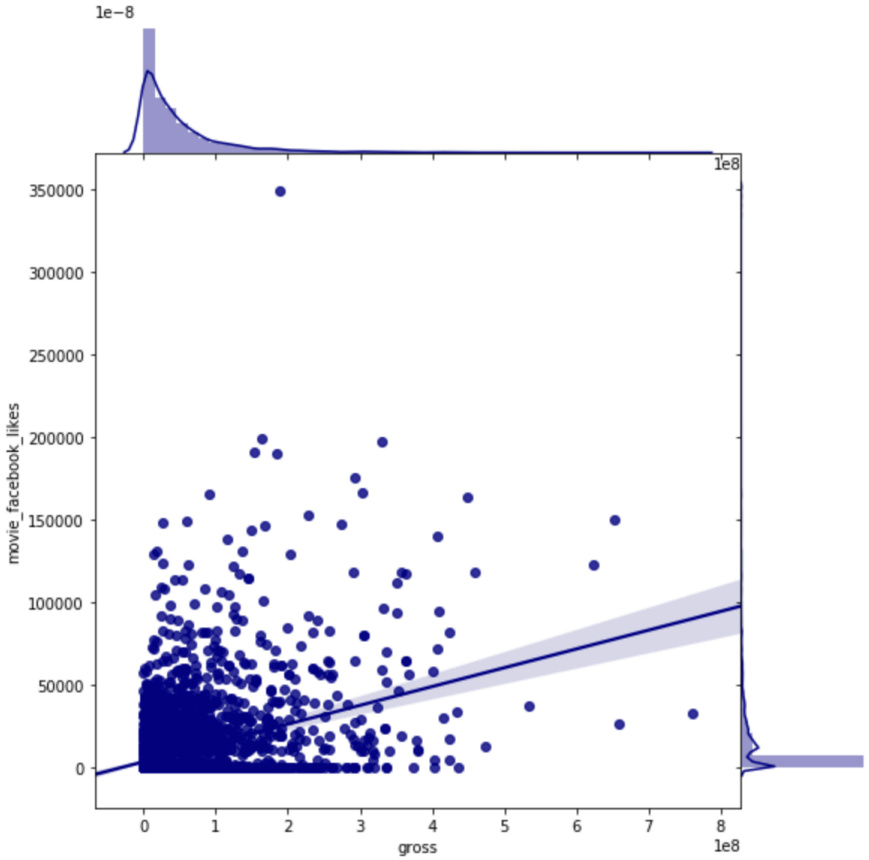
	director_name	imdb_score
0	Tony Kaye	8.600000
1	Majid Majidi	8.500000
2	Damien Chazelle	8.500000
3	Ron Fricke	8.500000
4	Christopher Nolan	8.425000
5	Marius A. Markevicius	8.400000
6	Sergio Leone	8.400000
7	Asghar Farhadi	8.400000
8	Richard Marquand	8.400000
9	Lenny Abrahamson	8.300000
10	Lee Unkrich	8.300000
11	Pete Docter	8.233333
12	Hayao Miyazaki	8.225000
13	Quentin Tarantino	8.200000
14	Joshua Oppenheimer	8.200000
15	Juan José Campanella	8.200000
16	David Sington	8.100000
17	Terry George	8.100000
18	Je-kyu Kang	8.100000
19	Tim Miller	8.100000

3.9 Relationship between number of Facebook likes and IMDb score/gross revenue

Generally, a good movie will have a higher number of Facebook likes because it indicates popularity. Higher number of Facebook likes means that the movie is popular either because the movie is commercially successful or critically successful or both. Let us examine the relationship between the number of Facebook likes variable with first IMDb score variable and then gross revenue variable.



This scatter plot is divided by content-rating. It can easily be interpreted that movie with extremely high Facebook likes tend to have higher IMDb score. But the score for movie with low Facebook likes vary in a very wide range.



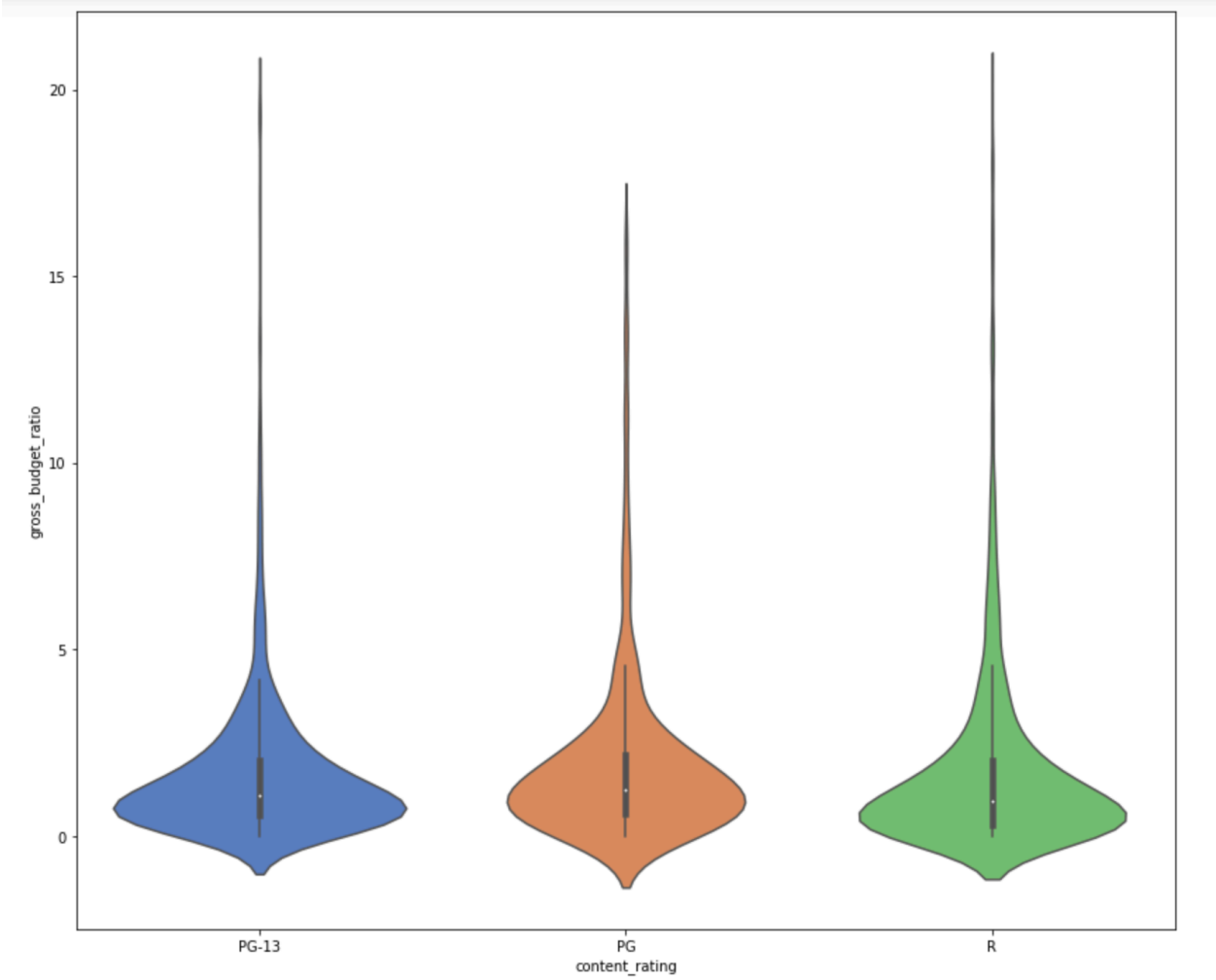
Gross income also shows positive relationship with movie facebook likes variable which means that higher number of Facebook like is strongly correlated with gross income, and commercially successful movies will generally have higher Facebook likes.

3.10 Is the type of film content rating important for its profitability?

To know that, we need to create a new column, revenue_budget_ratio.

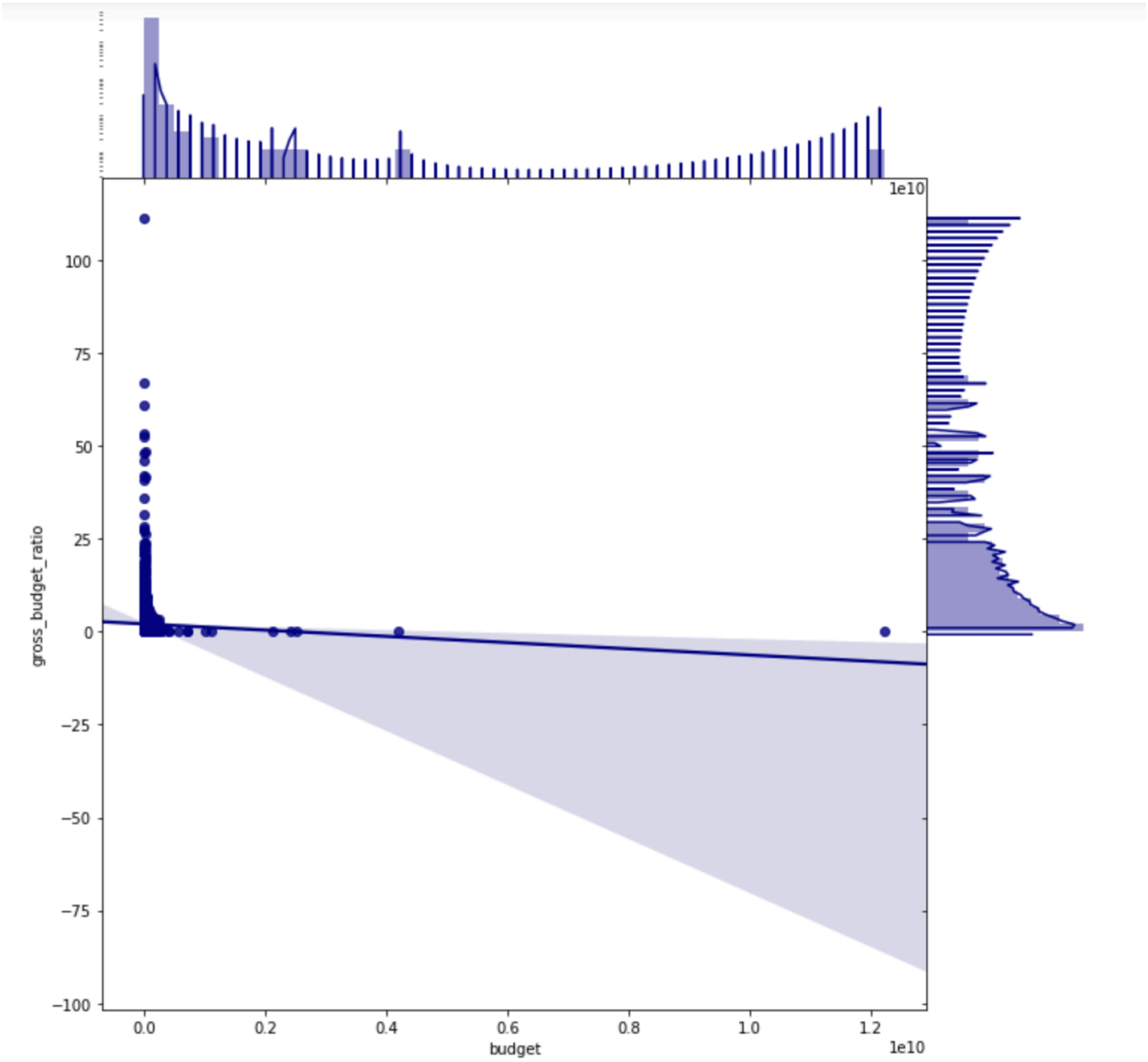
```
#We'll filter by a budget of $100k to avoid independent films
minbudget = 100000
data['gross_budget_ratio'] = data['gross'][data['budget']>minbudget]/data['budget'][data['budget']>minbudget]

#We will filter to show only ratings with more than 100 films in the DB
ratinglist = df.content_rating.value_counts()[df.content_rating.value_counts() >= 100]

fig, ax = plt.subplots(figsize=(14,12));
ax = sns.violinplot(x='content_rating', y="gross_budget_ratio"
                    ,data=data[data.content_rating.isin(ratinglist.index.values[:])][data.gross_budget_ratio<20]
                    , palette="muted", split=True)
ax.set_ylim = (100)

print(data[['gross_budget_ratio','content_rating']][data.content_rating.isin(ratinglist.index.values[:])]
      .groupby('content_rating').mean())
```

```
                gross_budget_ratio
content_rating
PG                        2.453530
PG-13                     1.728231
R                         2.131983
```



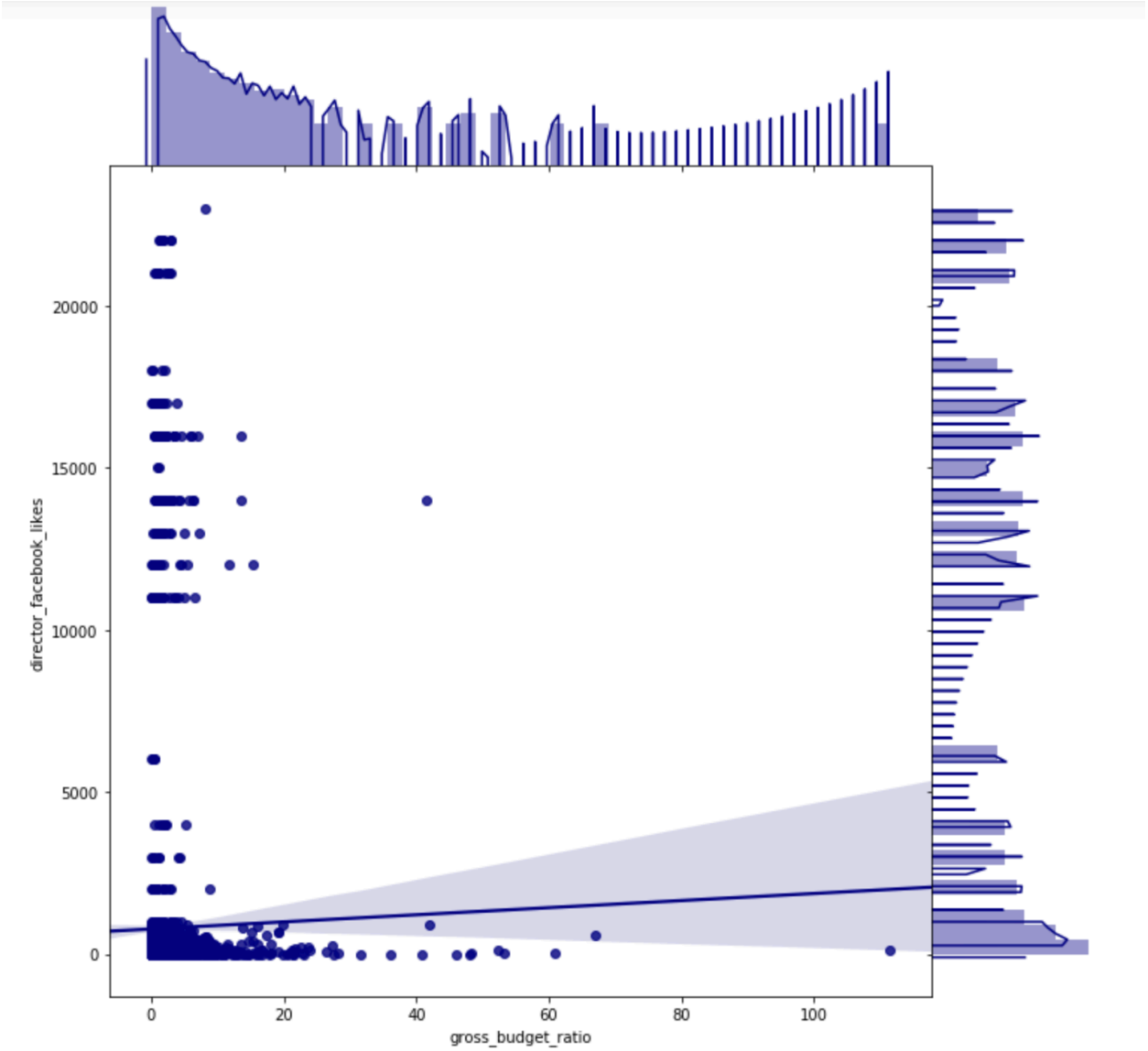
We can see how the content rating is in fact affecting the ratio distribution shape, with the PG rated films being the ones with a higher profitability ratio and the PG-13 have a lower one.

3.11 Is budget a factor in profitability of films?



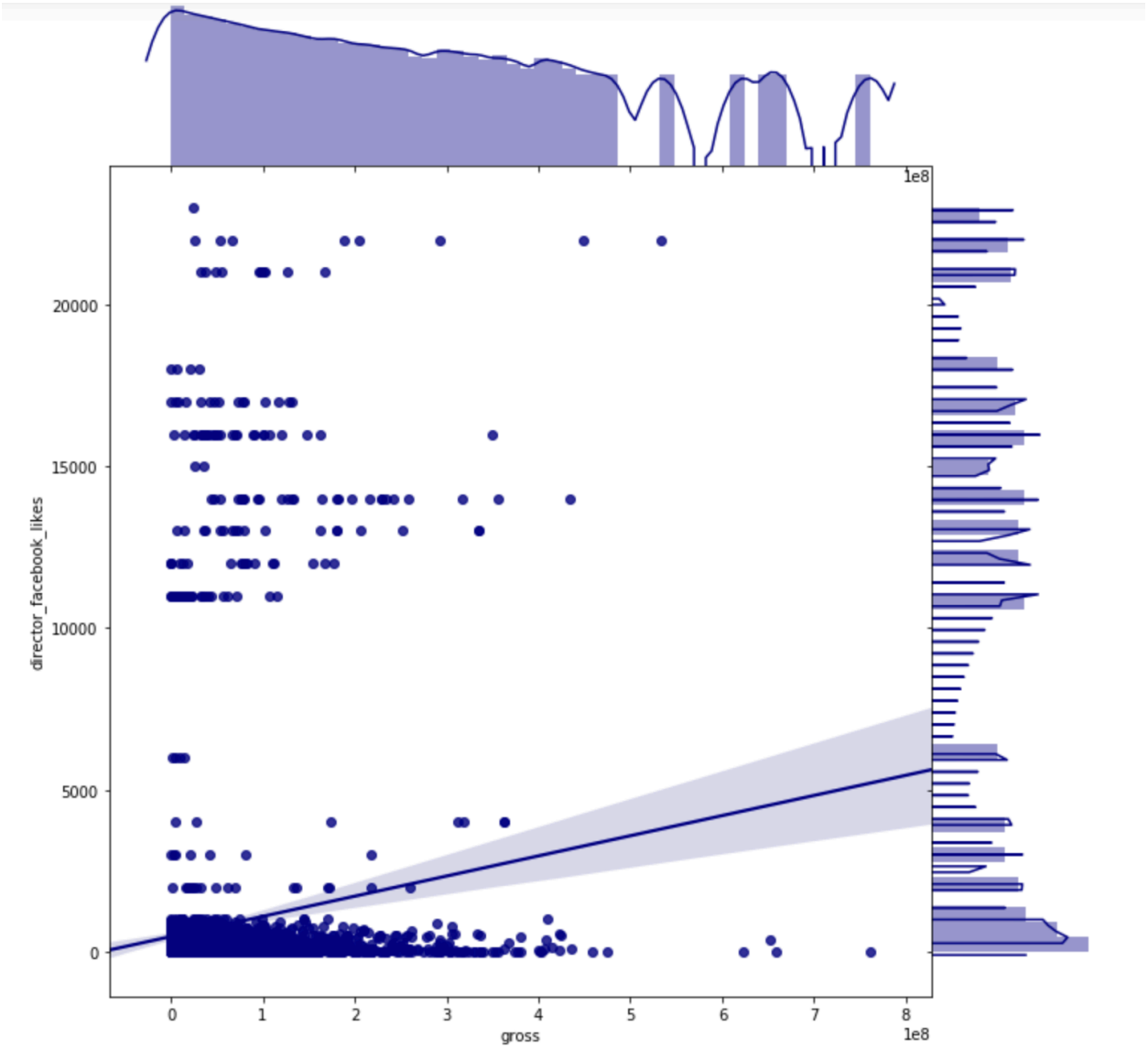
We can see how, although budget strongly correlates with revenue, it does negatively affect the films potential for profitability.

3.12 Does having more FB likes relates to a higher profitability for directors?



The number of FB likes of the director and film profitability (revenue/budget ratio) doesn't seem to be important.

3.13 What if we chose to compare against the gross revenue of their films?



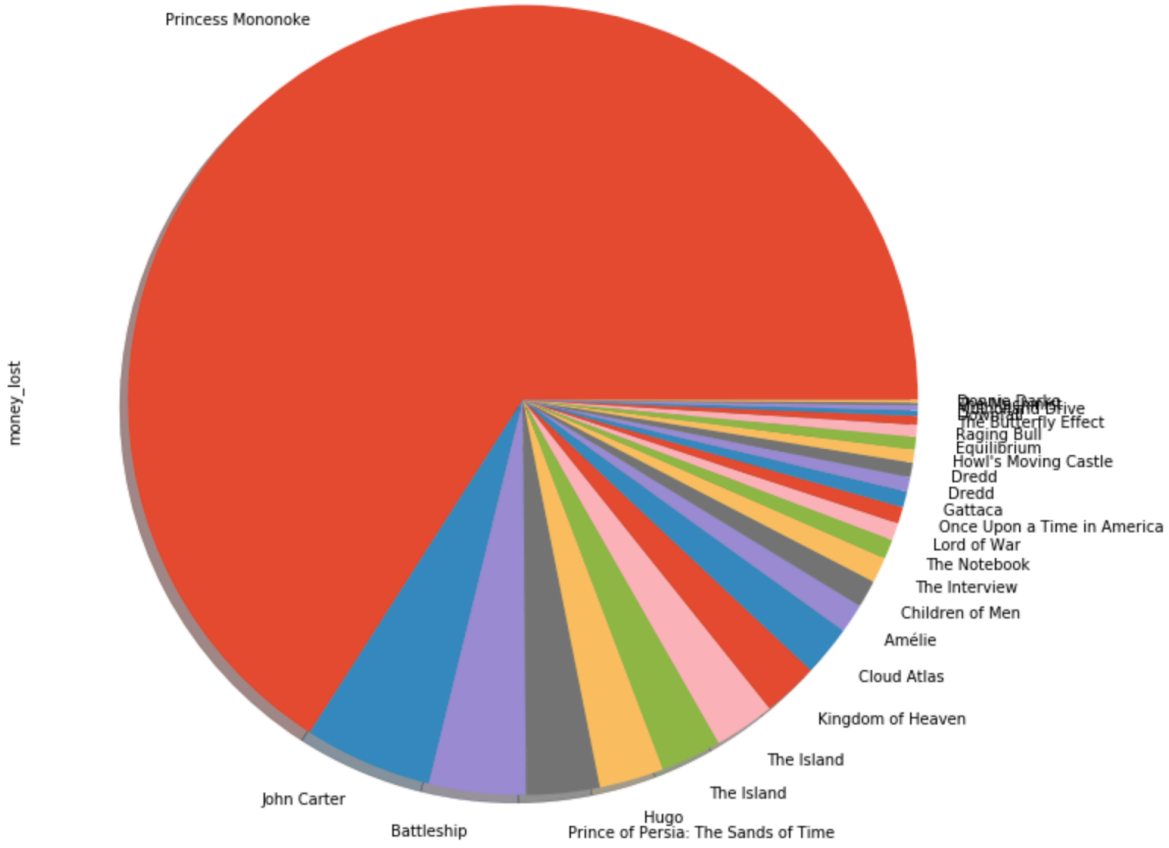
We can see how the directors of films with great gross revenues tend to have more FB likes, although that doesn't seem to imply that the profitability of their films is higher.

3.14 What films in our database are the biggest flops?

We're going to review now what films in our database are the biggest flops. To do so, we will:

- Filter first by films with budget lower than gross revenue.
- Filter by those films with cast total likes is under 100000 and number of voted users at least 200000.
- Finally, sort those films by the amount of money they lost.

```
df_flops = data[data.gross_budget_ratio<0.5][data.gross_budget_ratio>0]
df_flops['money_lost'] = (1-df_flops['gross_budget_ratio'])*df_flops['budget']
df_flops = df_flops[df_flops.cast_total_facebook_likes<100000][df_flops.num_voted_users>200000]
df_flops.sort_values(by='money_lost',ascending=False,inplace = True)

#print(df_flops[['original_title','money_lost','vote_average']])

fig, ax = plt.subplots(figsize=(14,12));
plt.style.use('ggplot')
explode=[0.1,0.1,0.1,0.1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
df_flops['money_lost'].plot(kind='pie', labels=df_flops.movie_title, fontsize=10,shadow=True)
plt.show()

df_flops.set_index('movie_title',inplace=True)

print(df_flops[['money_lost','gross_budget_ratio']])
```



movie_title	money_lost	gross_budget_ratio
Princess Mononoke	2.397702e+09	0.000958
John Carter	1.906413e+08	0.277052
Battleship	1.438268e+08	0.311833
Prince of Persia: The Sands of Time	1.092444e+08	0.453778
Hugo	9.617991e+07	0.434236
The Island	9.020097e+07	0.284119
The Island	9.020097e+07	0.284119
Kingdom of Heaven	8.260330e+07	0.364590
Cloud Atlas	7.490142e+07	0.265672
Amélie	4.379834e+07	0.431190
Children of Men	4.071357e+07	0.464295
The Interview	3.789482e+07	0.138754
The Notebook	2.893571e+07	0.002217
Lord of War	2.587210e+07	0.482558
Once Upon a Time in America	2.470000e+07	0.176667
Gattaca	2.366037e+07	0.342768
Dredd	2.159832e+07	0.382905
Dredd	2.159832e+07	0.382905
Howl's Moving Castle	1.928954e+07	0.196269
Equilibrium	1.880998e+07	0.059501
Raging Bull	1.795475e+07	0.002514
The Butterfly Effect	1.297605e+07	0.001842
Downfall	7.998060e+06	0.407551
Mulholland Drive	7.780422e+06	0.481305
The Machinist	3.917956e+06	0.216409
Donnie Darko	3.772117e+06	0.161752

4 Data Pre-processing

4.1 Remove Names

The name columns in this data set are “director_name”, “actor_1_name”, "actor_2_name", and "actor_3_name". Let us analyze the name columns to check to see if they hold any significance in predicting the IMDb score.

```
df['director_name'].nunique()
```

```
1660
```

```
df[["actor_1_name", "actor_2_name", "actor_3_name"]].nunique()
```

```
actor_1_name    1423
actor_2_name    2168
actor_3_name    2576
dtype: int64
```

Most of the values in the name columns are so different for the whole dataset, and the combinations of the name columns are all different for each movie. Therefore, there is no point to use name columns to predict IMDb score.

We can also remove other columns which are too diverse to be used for prediction. These are "movie_title", "plot_keywords", and "movie_imdb_link". All of these columns are redundant variables which we will drop.

```
df = df.drop(["director_name", "actor_1_name", "actor_2_name", "actor_3_name",
              "movie_title", "plot_keywords","movie_imdb_link"], axis = 1)
```

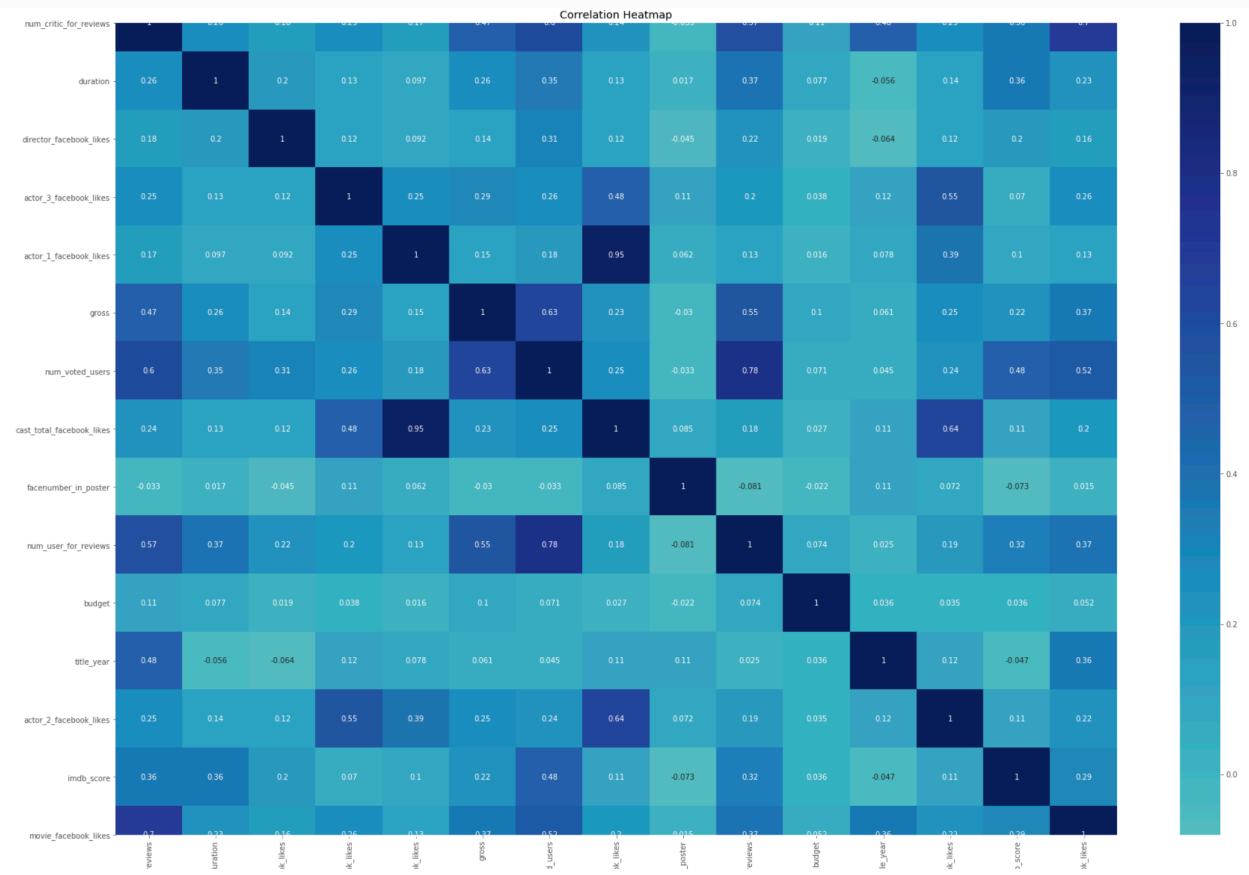
4.2 Remove Linear Dependent Variables

For the purpose of data exploration, we added two variables based on existing variables, which are, profit and return_on_investment. In order to avoid multicollinearity, here we remove these two added variables.

```
df.drop(['profit', 'return_on_investment'], axis = 1, inplace=True)
```

4.3 Remove Highly Correlated Variables

In order to make sure that our prediction is more accurate, we need to remove variable which are highly correlated with each other. This is done to avoid any error in prediction arising due to multicollinearity. First, we need to examine with variables are highly correlated with other. For that we will plot the correlation heatmap for our data, as follows:

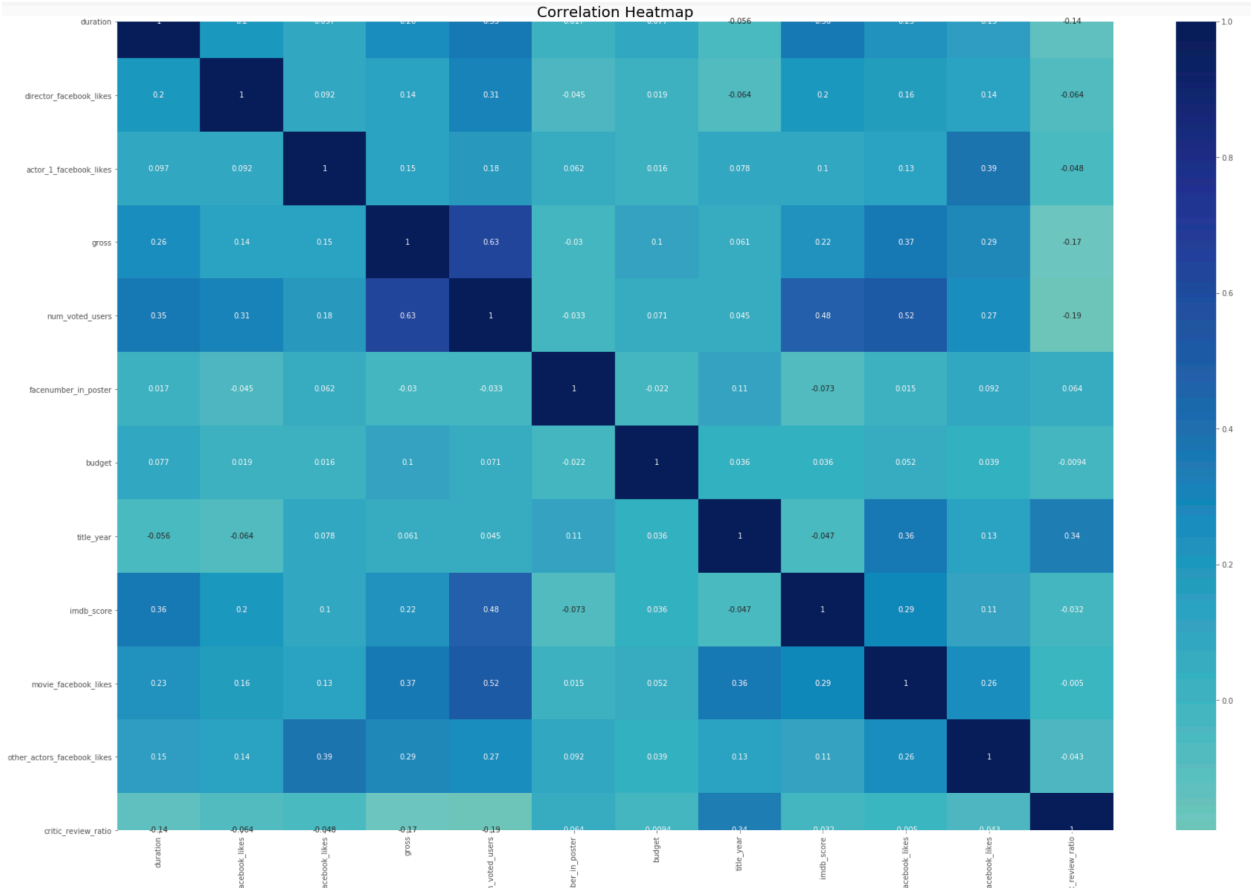


Based on the heatmap, we can see some high correlations (greater than 0.7) between predictors. According to the highest correlation value 0.95, we find actor_1_facebook_likes is highly correlated with the cast_total_facebook_likes, and both actor2 and actor3 are also somehow correlated to the total. 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.

```
df['other_actors_facebook_likes'] = df['actor_2_facebook_likes'] + df['actor_3_facebook_likes']

df['critic_review_ratio'] = df['num_critic_for_reviews'] / df['num_user_for_reviews']

df.drop(['cast_total_facebook_likes', 'actor_2_facebook_likes', 'actor_3_facebook_likes',
         'num_critic_for_reviews', 'num_user_for_reviews'], axis=1, inplace=True)
```

Now the correlation heatmap becomes like the figure below:



Now, we don't see any strong correlation (absolute value greater than 0.7) anymore.

4.4 Handling Categorical Variables using Label Encoder

We will look for the remaining data set as follows:

```
df.head()
```

	duration	director_facebook_likes	actor_1_facebook_likes	gross	num_voted_users	facenumber_in_poster	country	content_rating	budget	title_ye
0	178.0	0.0	1000.0	760505847.0	886204	0.0	USA	PG-13	237000000.0	2009
1	169.0	563.0	40000.0	309404152.0	471220	0.0	USA	PG-13	300000000.0	2007
2	148.0	0.0	11000.0	200074175.0	275868	1.0	UK	PG-13	245000000.0	2015
3	164.0	22000.0	27000.0	448130642.0	1144337	0.0	USA	PG-13	250000000.0	2012
5	132.0	475.0	640.0	73058679.0	212204	1.0	USA	PG-13	263700000.0	2012

Now, we have only country and content_rating columns as the categorical variables in our dataset. We will use LabelEncoder from sklearn.preprocessing library in python in order to convert these two remaining categorical variables in the dataset to numerical variables so that they can be easily processed through various machine learning algorithms for modeling.

```
from sklearn.preprocessing import LabelEncoder

df[['country', 'content_rating']] = df[['country', 'content_rating']].apply(LabelEncoder().fit_transform)
```

4.5 Splitting the Data into Testing and Training Dataset

For machine learning application, we will divide our dataset into testing and training data. For this dataset, we have used 70% of the data as training dataset and the remaining 30% of the data as testing dataset. We have used imdb_score column as our y variable and the remaining columns as our X variables, as follows:

```
from sklearn.model_selection import train_test_split

X = df.drop('imdb_score', axis = 1)
y = df['imdb_score']

X_train, X_test, y_train, y_test = train_test_split( X, y, test_size = 0.30, random_state = 7 )
```

5 Implement Algorithm for Data Modeling

We have used the following Machine Learning Algorithms to determine which algorithm best able to capture variance in the data and also serve as a useful tool to predict IMDb score future outcome based on current data.

1. Linear Regression
2. Lasso Regression
3. Ridge Regression
4. Elastic Net Regression
5. Decision Tree Regressor
6. Random Forest Regressor
7. Gradient Boosting Regressor
8. XG Boost Regressor
9. Light GBM Regressor

We have defined the output generated from these ML algorithms to be RMSE and R2_score. RMSE (Root Mean Square Error) is used to extract error in prediction from these ML algorithms

and R2_score is used to measure the total variance explained by these ML models. R2_score parameter generally vary from 0 to 1.

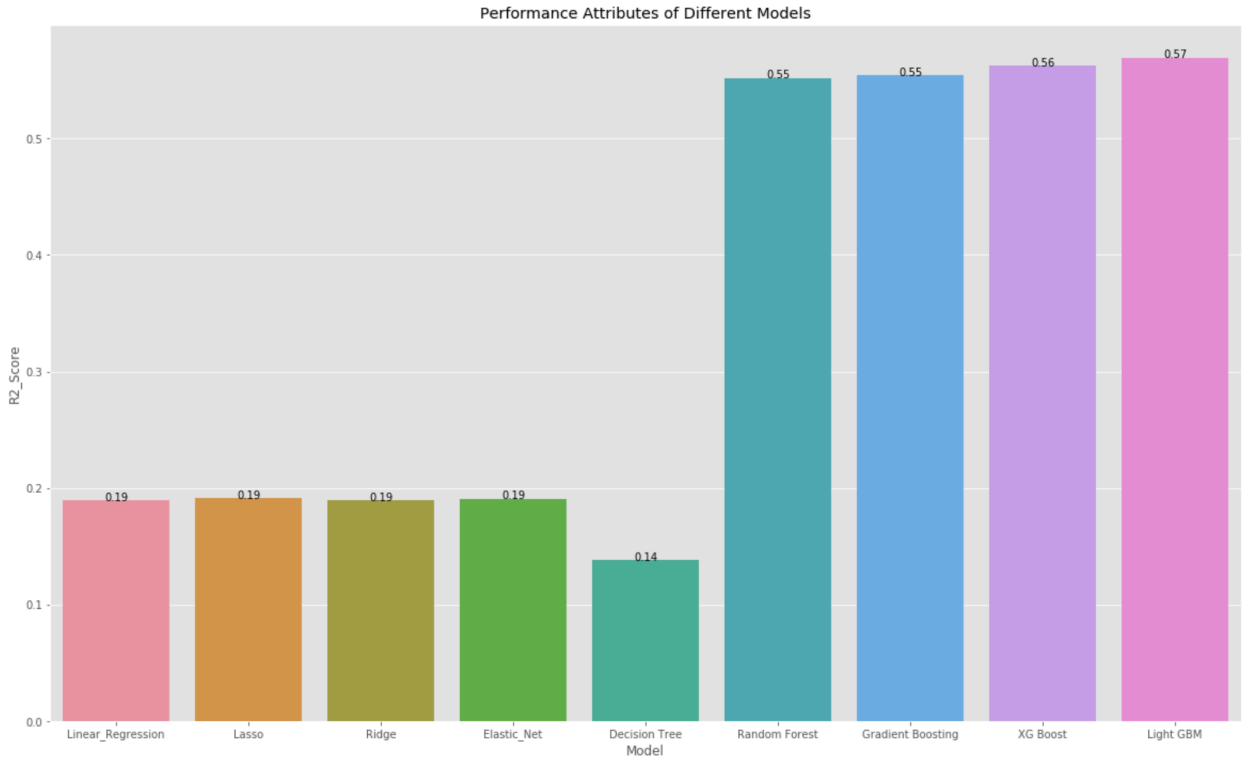
As the general criteria to test the effectiveness of our ML models, we ideally want the ML model to produce maximum R2_score with minimum RMSE or we want the optimum tradeoff between R2_score and RMSE to find the best ML algorithm.

After applying each ML algorithms on our training dataset and comparing the resultant output data against the testing dataset, we have tabulated the resultant output parameters (RMSE and R2_score) from each of the ML algorithms, as follows:

	Model	RMSE	R2_Score
0	Linear_Regression	0.965540	0.189746
1	Lasso	0.964501	0.191488
2	Ridge	0.965539	0.189746
3	Elastic_Net	0.964887	0.190841
4	Decision Tree	0.995898	0.137993
5	Random Forest	0.717907	0.552063
6	Gradient Boosting	0.716268	0.554106
7	XG Boost	0.709559	0.562420
8	Light GBM	0.703839	0.569446

Based on the overall performance, we find the best model is the Light GBM model, which gives least Root Mean Square Error (RMSE) of about 0.703 and a achieves the highest R2_score of around 0.57. Therefore, Light GBM model captures maximum variance in the dataset and also has the highest predictability power for the dataset to predict IMDb Score.

We have plotted the performance of each of the models using the bar chart, as follows:



6 Conclusion

The IMDb score for movies is dependent of several factors, many of which were mentioned in the movie metadata dataset. There were many features which were highly correlated and many which were irrelevant for capturing variance and also predicting IMDb score. We were successfully able to capture the factors responsible for variance of IMDb score column by plotting it against various different features in the dataset, through explanatory data analysis. We deduced the top features which best predicts IMDb score of a movie by computing and analyzing correlation matrix. Finally, we were able to perform data modeling through various machine learning modeling techniques in order to determine which model is best suitable to capture the variance in the dataset and also serve as a useful tool to predict the movements in IMDb score variable based on the dataset.