VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

ANALYSIS, VISUALIZATION ON MOVIES & SO FORTH THEIR RECOMMENDATION

PROJECT RESEARCH ARTICLE

CSE3020

DATA VISUALIZATION

By

19BCE1535 – ARASADA EKAVEERA ANEEL KUMAR [1]

19BCE1585 – PALAMANGALAM VARSHITH [1]

Under the guidance of

Dr. K. P. VijayaKumar [2]

School of Computer Science and Engineering

Fall Semester 2021-2022

- 1 - Undergraduate, Vellore Institute of Technology, Chennai
- 2 - Assistant Professor Senior Grade 2, Vellore Institute of Technology, Chennai

ABSTRACT:

The presentation of data in a graphical representation is known as data visualization. It genuinely aids in demonstrating and comprehending the true meaning of information by illustrating and presenting large quantities of data in a simple and easy-to-understand style, as well as aiding in the clear and efficient communication of information. In this paper, we look at how to use Python to visualize data and comprehend it better. When applied to the Movies Dataset, data visualization aids in the comprehension of the data by presenting a variety of relevant insights.

The significance of movie visualization in relation to the movie's audience and review trends becomes more apparent. Moviemakers want to know not only how popular their film is in terms of how many people have watched it, but also how well it has been accepted by those who have seen it. They need to figure out how spectators and reviews are related if they want the picture to be a hit or can it generate more income.

KEYWORDS:

Graphical representation, Movies Visualization and Analysis, Recommender System, Content Based Filtering, Audience, IMDB Movies dataset, Web Scraping

PROBLEM STATEMENT:

Profitability is the primary goal of film production. Some films generate a lot of money, while others suffer losses. A movie analysis can assist you understand how aspects like runtime, average voting, and genres affect income. The problem may be divided into two sub-problems that target various aspects of the issue. The first sub-problem tries to examine and visualize the relation among various characteristics of a movie. Prior to the invention of the recommendation system, people would physically select movies to watch from movie libraries. They had to either read the user reviews and select a movie based on the review or choose a random movie. This approach is not viable since there are a large number of spectators who have a distinct taste in films. So, the movie portals are vying with each other to discover the best and most effective approach to apply this technology in order to boost customer satisfaction and experience. As a result, the second sub-problem focuses on recommending related movies to a user using a content-based filtering approach.

INTRODUCTION:

Audiences gain from a more accessible and efficient means of acquiring information in today's enhanced internet and mobile environments. Nonetheless, the overwhelming amount of data makes it more difficult to determine what is relevant to the viewers.

Similarly, because there are so many movies in the globe, audiences have a difficult time deciding which one is ideal for them. They frequently visit movie-portal websites to read other people's reviews or ratings, and online portals have become one of the most crucial determinants in movie producers' profit margins. Furthermore, due to the rapid growth of social networks, knowledge quickly spreads to others (i.e. social friends). People sometimes assume that if a film has a high number of positive reviews, it is an excellent film. However, while this is true in some cases, it is not always the case.

In our work, to address these aspects, we identify to make a visualization of movie ratings and ranking based on time series. Clear visualization on huge user's data is important since compact abstraction or a large amount of information helps people decide proper choices. With the help of web scraping, we have the extracted data from movie portal as it allows quick and efficient extraction. We scraped the online movie reviews data to recognize relationship between users and reviews. Even if we tackle two aspects (movie maker, audience), our main target is movie distributors (i.e movie makers). We explore the movie's audience patterns and influence the reviews on movie.

LITERATURE SURVEY:

Jaehoon Lee1 , Giseop Noh 2 , Chong-kwon Kim3

The following are the primary contributions of this paper: They proposed a visualization approach to find clearly hidden relations between movies and their evaluation. They Analyzed the patterns with reviews, found out the influence of word-of-mouth effects. The rest of the paper is presented as follows: they demonstrated a visualization approach and details the concept of user interface and visualization methods in section III. In section IV they have explained the implementation environments, also analyzed the findings.

SRS Reddy, Sravani Nalluri, Subramanyam Kunisetti, S. Ashok and B. Venkatesh

The main contributions in this paper are summarized as follows: The recommendation algorithm in this article is based on the genres that the user may choose to watch. If a person gives a high rating to a film of a specific genre, films of like genres will be recommended to him. Recommendation systems are commonly utilised in today's Web 2.0 age to find trustworthy and relevant content. The method used to do this is content-based filtering with genre correlation. The system makes use of the Movie Lens dataset.

Bruce W. Herr, Weimao Ke, Elisha Hardy & Katy Börner

The findings of an analysis and visualisation of 428,440 movies from the Internet Movie Database (IMDb) for the Graph Drawing 2005 contest are presented in this presentation. Simple data, as well as a tapestry of all movies with an overlay of the co-actor network's massive component, are shown. The winners of the Academy Awards are highlighted. Major insights are discussed.

MOTIVATION:

In today's world, the film industry has grown enormously, so we thought of making it possible to determine and visualize people's interest in various genres as well as lucrative measures for a production firm. We came upon the Netflix app while we were thinking about viewing a movie but weren't sure what we wanted to see. On the first page, it suggested a few movies to watch, so we picked the first one. We loved the movie and thought how NETFLIX can predict our genre choices. So, we choose this project to grasp the background functionality of how the movies are recommended on different OTT apps like Netflix, Amazon Prime Video, Disney + Hotstar, Zee5 etc.

PROPOSED SYSTEM:

❖ DATASET AND PRE-PROCESSING:

We have scraped the movie dataset from a famous internet movie review site (IMDb.com) using a web-scraper tool named Parsehub. Using Parsehub application we have extracted important data such as Movie title, Year of Release, IMDb rating, etc. We note that any other online movie site can be applied our approach. The dataset consists of 2066 movies. Again, to analyze the size of data is non-trivial. As the above data set contains only few movies, in order to make the system more appropriate and precise rather of relying on approximations, we extracted additional dataset including 4802 movies, which is beneficial and may be applied to even enormous datasets.

❖ MODELS USED IN THE SYSTEM:

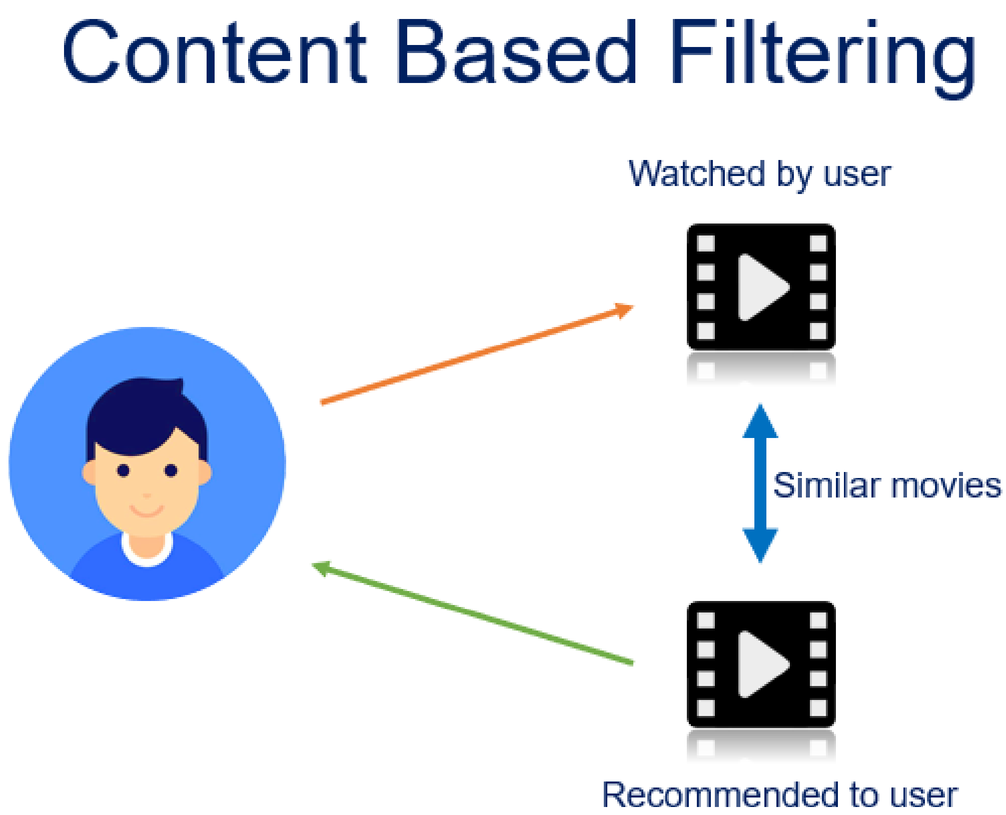
To depict the relationships, we applied several visualization plots such as bar graphs, scatter plots, line graph, and so on using different modules in python such as numpy, pandas, seaborn, etc. Apart from the visualization we have also employed Content Based filtering.

This concept is entirely based on comparing user interests to product attributes. The items with the greatest overlapping features with user interests are the ones that are recommended. Given the importance of product features in this system, it is critical to understand how the user's preferred features are determined.

Two approaches can be employed in this situation (possibly in combination). To begin, consumers may be presented with a selection of traits from which they could select the one that most closely resembles their own. Second, the algorithm may keep track of the goods that the user has already selected and include those attributes into the user's data.

Product features, on the other hand, can be recognized by the product's developers in our case the features are average rating, total vote count, genres and others. Furthermore, users might be asked which aspects they feel most closely relate to the items. The algorithm employed in this model is cosine similarity algorithm.

1. Select the required features from the dataset
2. Replace the null values in the dataset if they are present
3. If the selected features are text data then vectorize them, else proceed to the next step
4. Find the cosine similarity among the selected features/attributes
5. Now, based on the similarity index, retrieve the top movies recommended to the user based on user input.



ADVANTAGES OF THIS MODEL:

Because of the little amount of data, this approach is readily scalable. Furthermore, unlike previous models, this one does not need to compare data with other users, it may provide specialized findings tailored to the present user.

This methodology, however, necessitates a substantial level of domain expertise from those attributing attributes to items. As a result, its accuracy is heavily reliant on the correctness of that knowledge. Furthermore, content-based filtering is heavily reliant on previously established user interests.

IMPLEMENTATION:

CODING:

Importing the libraries required

```
import numpy as np # linear algebra

import pandas as pd # data processing

import seaborn as sns

import re

import matplotlib.pyplot as plt

%matplotlib inline


import chart_studio.plotly as py

from plotly.offline import init_notebook_mode, iplot

init_notebook_mode(connected=True)

import plotly.graph_objs as go


from wordcloud import WordCloud


import os
```

Reading the dataset

```
data = pd.read_csv("IMDb_movies_dataset.csv")
```

Analysis of the dataset

- Printing the attributres in our movie data set

```
data.columns
```

```
Index(['name', 'year', 'runtime', 'genres', 'IMDb_rating', 'IMDb_votes',
       'director', 'director_url', 'lead_actor', 'lead_actor_url',
       'certificate_category', 'IMDb_metascore', 'gross_collection'],
      dtype='object')
```

- Getting the information about the data types of attributes

data.info()

```
... <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 2066 entries, 0 to 2065
    Data columns (total 13 columns):
     #   Column                Non-Null Count  Dtype
    ---  ------                --------------  -----
     0   name                  2066 non-null   object
     1   year                  2066 non-null   object
     2   runtime               2066 non-null   object
     3   genres                2066 non-null   object
     4   IMDb_rating           2066 non-null   float64
     5   IMDb_votes            2066 non-null   object
     6   director              2066 non-null   object
     7   director_url          2066 non-null   object
     8   lead_actor            2066 non-null   object
     9   lead_actor_url        2066 non-null   object
     10  certificate_category  2063 non-null   object
     11  IMDb_metascore        1124 non-null   float64
     12  gross_collection      1971 non-null   object
    dtypes: float64(2), object(11)
    memory usage: 210.0+ KB
```

- Analyzing the head of dataset

data.head()

```
...
Output exceeds the size limit. Open the full output data in a text editor
                     name  year  runtime                genres  IMDb_rating  \
0                Jai Bhim  2021  164 min          Crime, Drama          9.5
1  The Shawshank Redemption  1994  142 min                 Drama          9.3
2           The Godfather  1972  175 min          Crime, Drama          9.2
3          Soorarai Pottru  2020  153 min                 Drama          9.1
4          The Dark Knight  2008  152 min  Action, Crime, Drama          9.0

  IMDb_votes              director  \
0    142,015          T.J. Gnanavel
1  2,495,893         Frank Darabont
2  1,721,252  Francis Ford Coppola
3    102,565         Sudha Kongara
4  2,446,825     Christopher Nolan

                                        director_url      lead_actor  \
0  https://www.imdb.com/name/nm4377096/?ref_=adv_...          Suriya
1  https://www.imdb.com/name/nm0001104/?ref_=adv_...     Tim Robbins
2  https://www.imdb.com/name/nm0000338/?ref_=adv_...   Marlon Brando
3  https://www.imdb.com/name/nm1464314/?ref_=adv_...          Suriya
4  https://www.imdb.com/name/nm0634240/?ref_=adv_...  Christian Bale

                                      lead_actor_url certificate_category  \
0  https://www.imdb.com/name/nm1421814/?ref_=adv_...                  NaN
1  https://www.imdb.com/name/nm0000209/?ref_=adv_...                    R
2  https://www.imdb.com/name/nm0000008/?ref_=adv_...                    R
```

```
...
   IMDb_metascore gross_collection
0             NaN              NaN
1            80.0          $28.34M
2           100.0         $134.97M
3             NaN              NaN
4            84.0         $534.86M
```

- Analyzing the trail content of dataset

data.tail()

```
...  Output exceeds the size limit. Open the full output data in a text editor
                        name  year  runtime                       genres  \
2061     Batman & Robin  1997  125 min              Action, Sci-Fi
2062           Catwoman  2004  104 min      Action, Crime, Fantasy
2063  Meet the Spartans  2008   87 min             Comedy, Fantasy
2064         Epic Movie  2007   86 min  Adventure, Comedy, Fantasy
2065              Radhe  2021  135 min     Action, Crime, Thriller

      IMDb_rating IMDb_votes         director  \
2061          3.8    242,563  Joel Schumacher
2062          3.4    115,424            Pitof
2063          2.8    106,411  Jason Friedberg
2064          2.4    104,122  Jason Friedberg
2065          1.8    173,525      Prabhu Deva

                                            director_url  \
2061  https://www.imdb.com/name/nm0001708/?ref_=adv_...
2062  https://www.imdb.com/name/nm0685759/?ref_=adv_...
2063  https://www.imdb.com/name/nm0294997/?ref_=adv_...
2064  https://www.imdb.com/name/nm0294997/?ref_=adv_...
2065  https://www.imdb.com/name/nm0222150/?ref_=adv_...
```

```
                  lead_actor  \
2061  Arnold Schwarzenegger
2062             Halle Berry
2063           Aaron Seltzer
...
      IMDb_metascore gross_collection
2061             NaN          $107.33M
2062             NaN           $40.20M
2063             NaN           $38.23M
2064             NaN           $39.74M
2065             NaN               NaN
```

data.index.name="index"

data.tail()

```
...  Output exceeds the size limit. Open the full output data in a text editor
                     name  year  runtime                        genres  \
index
2061      Batman & Robin  1997  125 min              Action, Sci-Fi
2062            Catwoman  2004  104 min       Action, Crime, Fantasy
2063   Meet the Spartans  2008   87 min             Comedy, Fantasy
2064          Epic Movie  2007   86 min  Adventure, Comedy, Fantasy
2065               Radhe  2021  135 min      Action, Crime, Thriller

       IMDb_rating IMDb_votes         director  \
index
2061           3.8    242,563  Joel Schumacher
2062           3.4    115,424            Pitof
2063           2.8    106,411  Jason Friedberg
2064           2.4    104,122  Jason Friedberg
2065           1.8    173,525      Prabhu Deva

                                            director_url  \
index
2061   https://www.imdb.com/name/nm0001708/?ref_=adv_...
2062   https://www.imdb.com/name/nm0685759/?ref_=adv_...
2063   https://www.imdb.com/name/nm0294997/?ref_=adv_...
2064   https://www.imdb.com/name/nm0294997/?ref_=adv_...
2065   https://www.imdb.com/name/nm0222150/?ref_=adv_...
```

```
                   lead_actor  \
...
index
2061               NaN         $107.33M
2062               NaN          $40.20M
2063               NaN          $38.23M
2064               NaN          $39.74M
2065               NaN              NaN
```

- As the data such as runtime, IMDb_votes & gross_collection are stored as string we need to convert to numericals to analyze them

```
def remove_min(minutes):
```

```python
    minutes = re.sub("[^0-9]", "", minutes)
    return minutes

data["runtime"] = data["runtime"].apply(remove_min)
data['runtime'].astype(str).astype(int)
data.loc[:8,["year","runtime","genres"]]
data['runtime'] = data['runtime'].astype(float, errors = 'raise')
data.info()
```

```
... <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 2066 entries, 0 to 2065
    Data columns (total 13 columns):
     #   Column                Non-Null Count  Dtype
    ---  ------                --------------  -----
     0   name                  2066 non-null   object
     1   year                  2066 non-null   object
     2   runtime               2066 non-null   float64
     3   genres                2066 non-null   object
     4   IMDb_rating           2066 non-null   float64
     5   IMDb_votes            2066 non-null   object
     6   director              2066 non-null   object
     7   director_url          2066 non-null   object
     8   lead_actor            2066 non-null   object
     9   lead_actor_url        2066 non-null   object
     10  certificate_category  2063 non-null   object
     11  IMDb_metascore        1124 non-null   float64
     12  gross_collection      1971 non-null   object
    dtypes: float64(3), object(10)
    memory usage: 210.0+ KB
```

```python
data['IMDb_votes'] = data["IMDb_votes"].replace(",", "", regex=True)
data['IMDb_votes'] = data['IMDb_votes'].astype(str).astype(float,
errors = 'raise')
data.info()
```

```
... <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 2066 entries, 0 to 2065
    Data columns (total 13 columns):
     #   Column               Non-Null Count  Dtype  
    ---  ------               --------------  -----  
     0   name                 2066 non-null   object 
     1   year                 2066 non-null   object 
     2   runtime              2066 non-null   float64
     3   genres               2066 non-null   object 
     4   IMDb_rating          2066 non-null   float64
     5   IMDb_votes           2066 non-null   float64
     6   director             2066 non-null   object 
     7   director_url         2066 non-null   object 
     8   lead_actor           2066 non-null   object 
     9   lead_actor_url       2066 non-null   object 
     10  certificate_category 2063 non-null   object 
     11  IMDb_metascore       1124 non-null   float64
     12  gross_collection     1971 non-null   object 
    dtypes: float64(4), object(9)
    memory usage: 210.0+ KB
```

- Converting gross collection data to numericals

```
data['gross_collection'].astype(str)

data['gross_collection'] =
data["gross_collection"].str.replace('$','',regex=True)

data['gross_collection'] =
data["gross_collection"].str.replace('M','',regex=True)

data.loc[:8,["certificate_category","IMDb_metascore","gross_collection
"]]

data['gross_collection'] = data['gross_collection'].astype(float,
errors = 'raise')
```

```
data.info()
```

```
... <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 2066 entries, 0 to 2065
    Data columns (total 13 columns):
     #   Column               Non-Null Count  Dtype  
    ---  ------               --------------  -----  
     0   name                 2066 non-null   object 
     1   year                 2066 non-null   object 
     2   runtime              2066 non-null   float64
     3   genres               2066 non-null   object 
     4   IMDb_rating          2066 non-null   float64
     5   IMDb_votes           2066 non-null   float64
     6   director             2066 non-null   object 
     7   director_url         2066 non-null   object 
     8   lead_actor           2066 non-null   object 
     9   lead_actor_url       2066 non-null   object 
     10  certificate_category 2063 non-null   object 
     11  IMDb_metascore       1124 non-null   float64
     12  gross_collection     1971 non-null   float64
    dtypes: float64(5), object(8)
    memory usage: 210.0+ KB
```

data.describe()

```
...             runtime  IMDb_rating    IMDb_votes  IMDb_metascore  \
    count  2066.000000  2066.000000  2.066000e+03     1124.000000
    mean    115.684898     7.065924  2.795328e+05       75.386121
    std      21.533238     0.824921  2.486194e+05        9.942686
    min      64.000000     1.800000  1.001520e+05       61.000000
    25%     100.000000     6.500000  1.329210e+05       67.000000
    50%     113.000000     7.100000  1.932100e+05       74.000000
    75%     127.000000     7.700000  3.180790e+05       83.000000
    max     242.000000     9.500000  2.495893e+06      100.000000

           gross_collection
    count       1971.000000
    mean          93.068752
    std           97.988945
    min            0.000000
    25%           31.010000
    50%           63.220000
    75%          125.325000
    max          936.660000
```

```
data1 = data.head(7)
melted = pd.melt(frame=data1,id_vars = "name",value_vars=["genres"])
melted
```

```
...                        name variable                  value
    0                  Jai Bhim   genres          Crime, Drama
    1  The Shawshank Redemption   genres                 Drama
    2             The Godfather   genres          Crime, Drama
    3           Soorarai Pottru   genres                 Drama
    4           The Dark Knight   genres  Action, Crime, Drama
    5    The Godfather: Part II   genres          Crime, Drama
    6              12 Angry Men   genres          Crime, Drama
```

```
mean_of_IMDb_user_rating = data["IMDb_rating"].mean()
mean_of_IMDb_user_rating
```

```
data["vote_level"] = [ "high_level" if each>mean_of_IMDb_user_rating else "down_level"  for each in data.IMDb_rating]
```

```
data.loc[:2066,["name","vote_level","IMDb_rating"]]
```

```
...                           name  vote_level  IMDb_rating
    index
    0                     Jai Bhim  high_level          9.5
    1     The Shawshank Redemption  high_level          9.3
    2                The Godfather  high_level          9.2
    3              Soorarai Pottru  high_level          9.1
    4              The Dark Knight  high_level          9.0
    ...                        ...         ...          ...
    2061            Batman & Robin  down_level          3.8
    2062                  Catwoman  down_level          3.4
    2063         Meet the Spartans  down_level          2.8
    2064                Epic Movie  down_level          2.4
    2065                     Radhe  down_level          1.8

    [2066 rows x 3 columns]
```

```
data2 = data[data.IMDb_rating > 9]
```

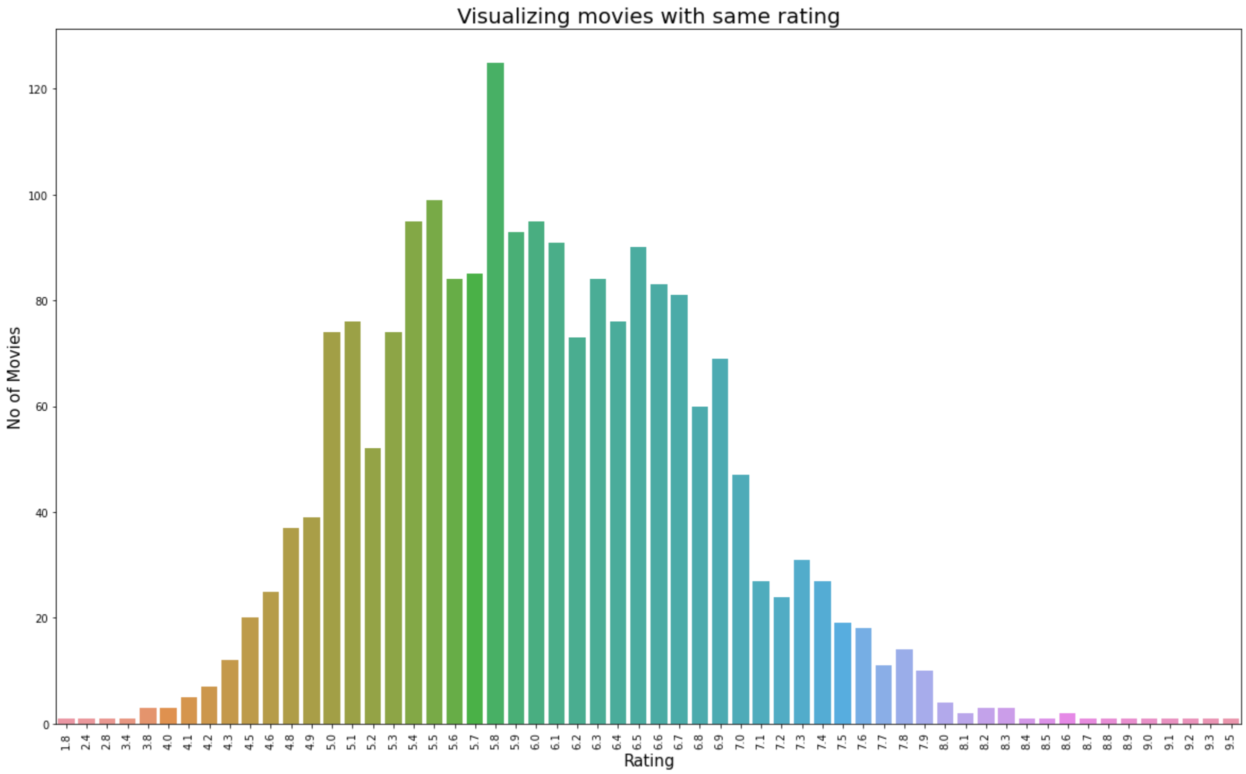
```
data2.loc[:,["name","IMDb_rating"]]
```

```
...                          name  IMDb_rating
    index
    0                    Jai Bhim          9.5
    1    The Shawshank Redemption          9.3
    2               The Godfather          9.2
    3             Soorarai Pottru          9.1
```

Visualizing the dataset

Bar Graphs

```
from collections import Counter

df=data.copy()
unique = list(df.IMDb_rating.unique())

list_ratio = df.pivot_table(columns=['IMDb_rating'], aggfunc='size')
# print(list_ratio)

df2 = pd.DataFrame({"Rating":unique,"No_of_movies":list_ratio})
new_index = (df2.No_of_movies.sort_values(ascending =
False)).index.values
sorted_data= df2.reindex(new_index)



# #Visualization
plt.figure(figsize = (20,12))
sns.barplot(x= sorted_data["Rating"],y  = sorted_data["No_of_movies"])
plt.xticks(rotation=90)
plt.xlabel("Rating",fontsize=15)
plt.ylabel("No of Movies",fontsize= 15)
plt.title("Visualizing movies with same rating",fontsize= 20)
```



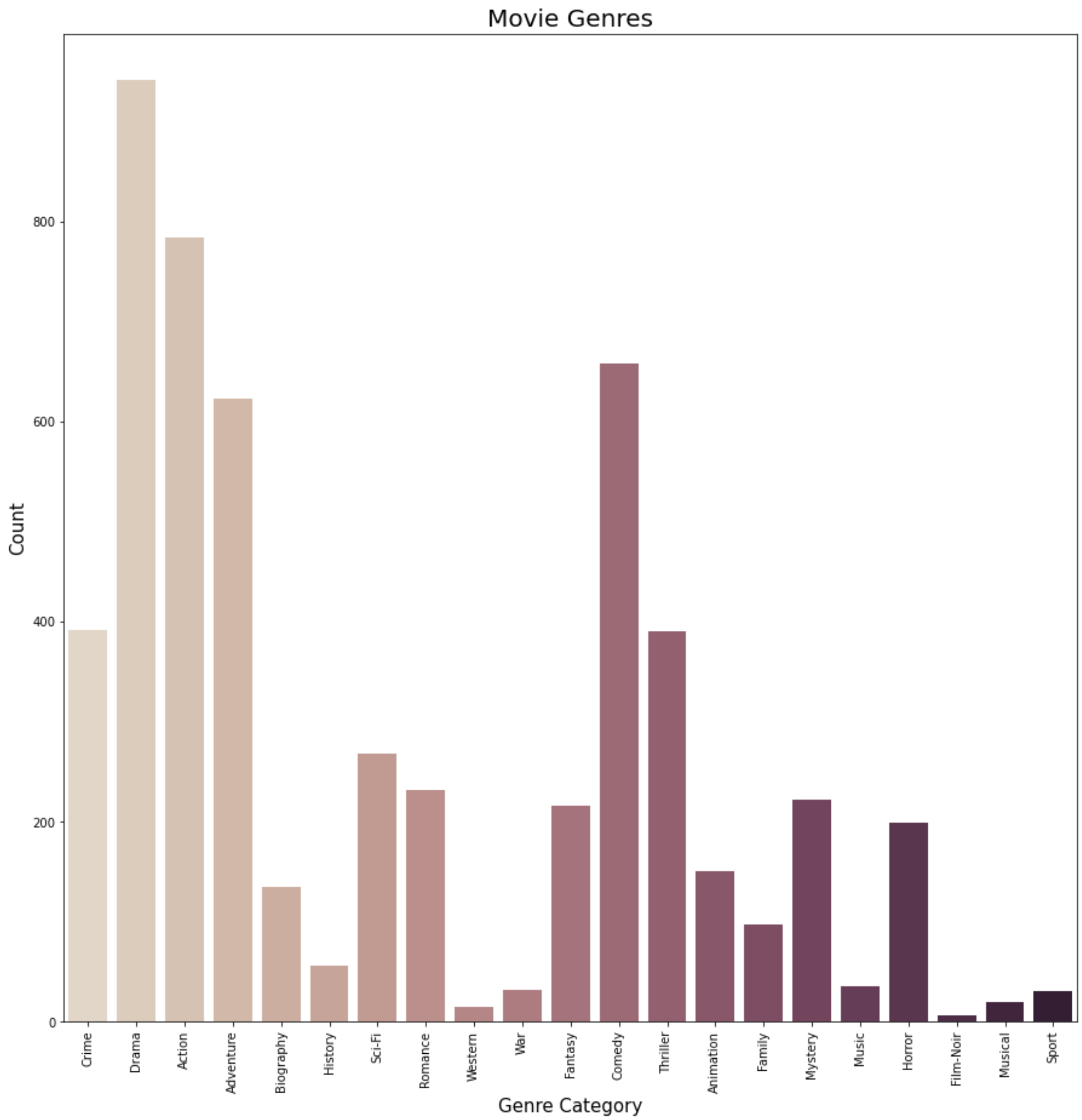

```
from collections import Counter

df = data.genres.copy()

list_kind = df.str.split(", ")
a = []
for each in list_kind:
    for i in each:
        a.append(i)

c=[]
for each in a:
    if each != "":
         c.append(each)
```

```
f= dict(Counter(c))



df3 = pd.DataFrame(list(f.items()),columns = ["kind","ratio"])
new_index =( df3.ratio).index.values
new = df3.reindex(new_index)
order_c = df3.ratio.sort_values(ascending=True)

new
```

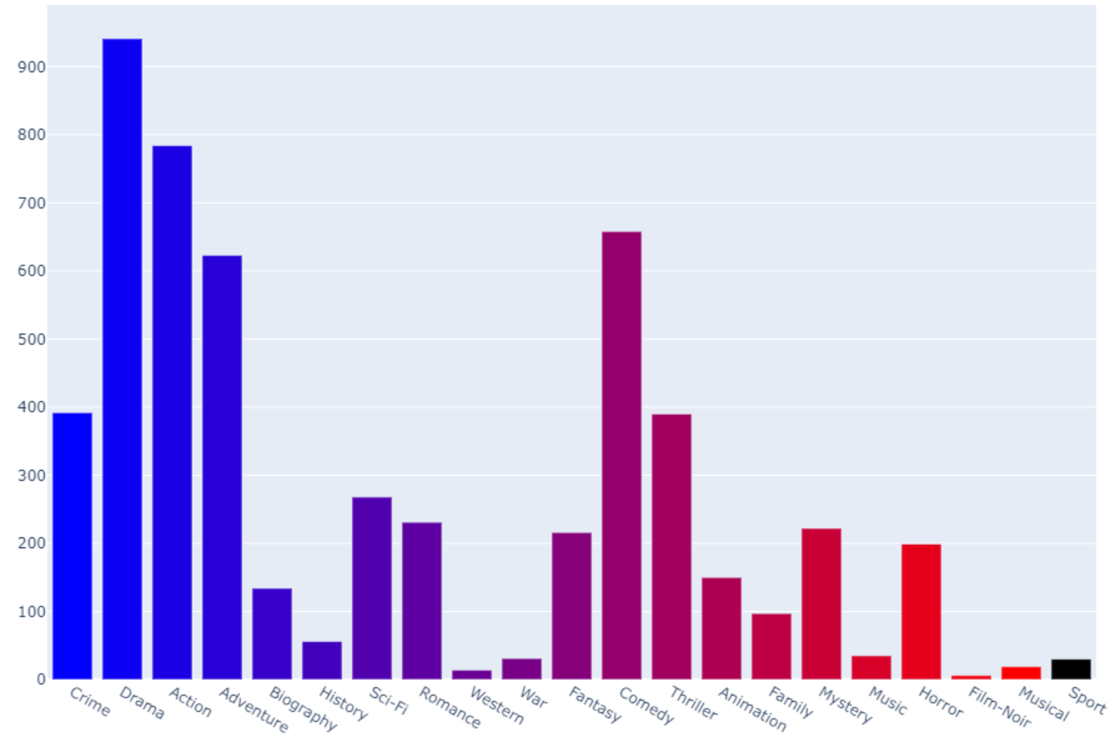
...	kind	ratio
0	Crime	392
1	Drama	941
2	Action	784
3	Adventure	623
4	Biography	134
5	History	56
6	Sci-Fi	268
7	Romance	231
8	Western	14
9	War	31
10	Fantasy	216
11	Comedy	658
12	Thriller	390
13	Animation	150
14	Family	97
15	Mystery	222
16	Music	35
17	Horror	199
18	Film-Noir	6
19	Musical	19
20	Sport	30

```
plt.figure( figsize = (15,15))
sns.barplot(x="kind",y="ratio",data=new, palette="ch:.25")
plt.xticks(rotation = 90)
plt.xlabel("Genre Category",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.title("Movie Genres",fontsize = 20)
```

```
Text(0.5, 1.0, 'Movie Genres')
```



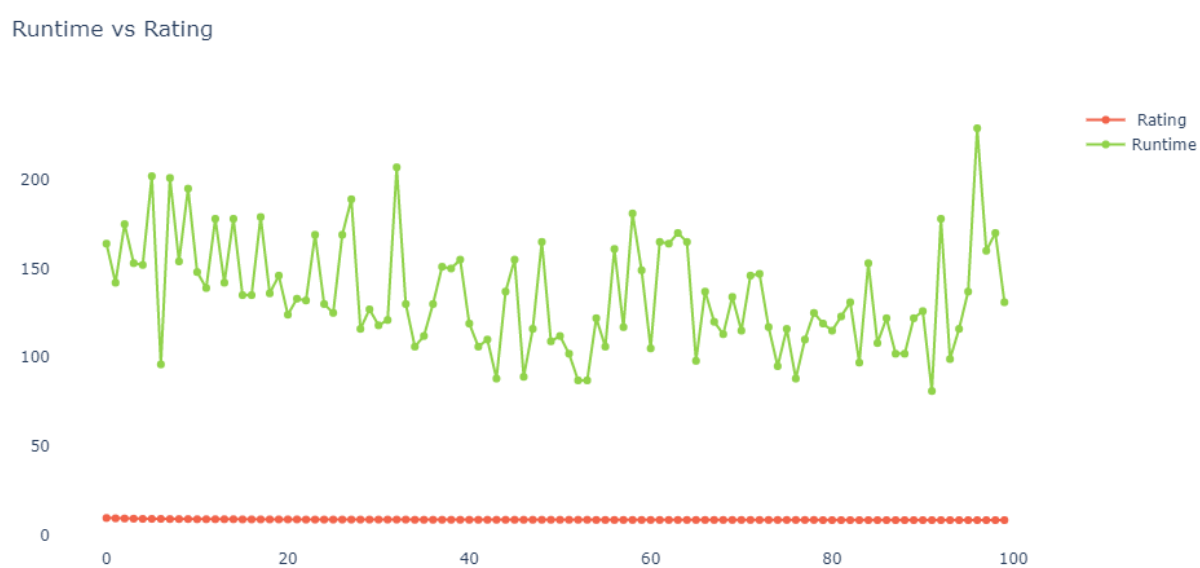
```
trace1 = go.Bar(
    x  = df3.kind,
    y = df3.ratio,
    name = "Ratio",
    marker = dict(
        color = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19],
        colorscale = "Bluered")
)

data1= [trace1]
layout = dict(
    autosize = False,
    width = 1000,
    height = 720,
    barmode = "group",)
fig  = dict (data = data1, layout = layout)
iplot ( fig)
```



Line Graphs

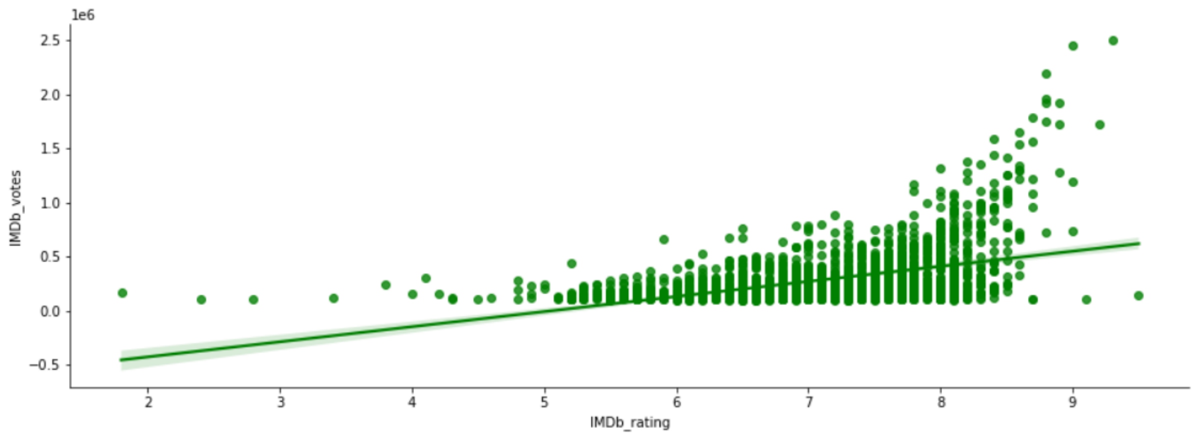
```
import plotly.graph_objs as go

df = data.head(100).copy()

trace1 =go.Scatter(
    x =df.index,
    y = df.IMDb_rating,
    mode ="lines + markers",
    name = " Rating",
    marker = dict(color = "rgb(242, 99, 74,0.7)"),
```

```
    text = df.name,
)
trace2 = go.Scatter(
    x = df.index,
    y = df.runtime,
    mode = "lines + markers",
    name = "Runtime",
    marker = dict( color = "rgb(144, 211, 74,0.5)"),
    text = df.name
)
data1=[trace1,trace2]
layout = dict(title = "Runtime vs Rating", hovermode = "x",xaxis =
{'showgrid' : False},yaxis = {'showgrid' : False},

paper_bgcolor='rgba(0,0,0,0)',plot_bgcolor='rgba(0,0,0,0)')
fig = dict ( data = data1 , layout = layout)
iplot(fig)
```



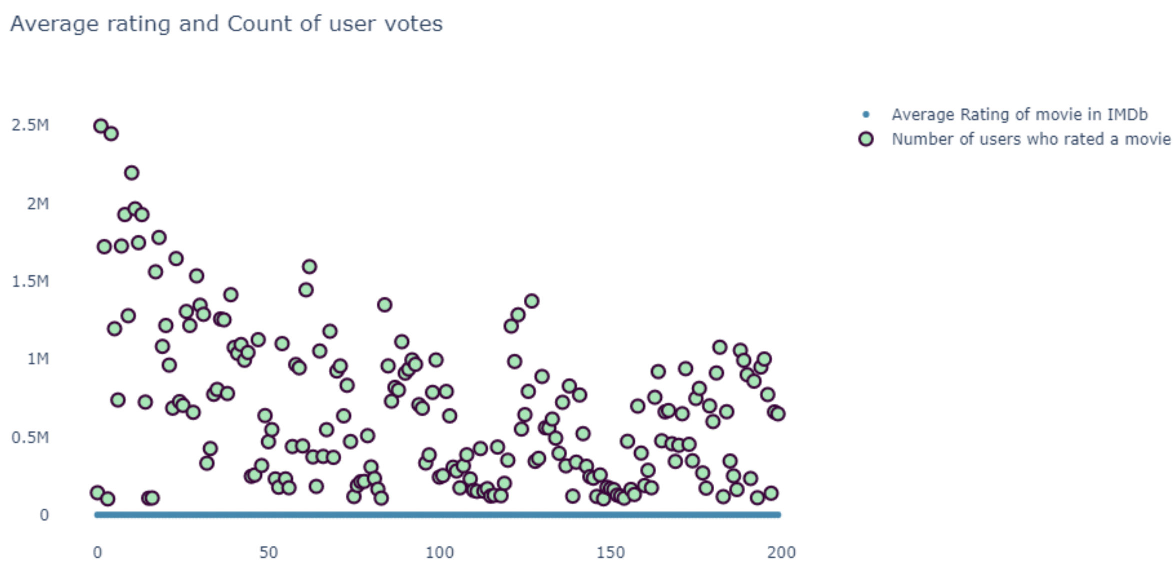
Scatter plots

```
plt.figure( figsize = (15,5))
sns.regplot( x = data.IMDb_rating, y = data.IMDb_votes, color = "g" ,
data=data)
sns.despine()
plt.show()
```




```
df = data.head(200).copy()

trace1 = go.Scatter(
    x = df.index,
    y = df.IMDb_rating,
    mode = "markers",
    name = "Average Rating of movie in IMDb",
    marker =dict( color = "rgb(70,136,173)",size=5,
        ),
    text = df.name
)
```

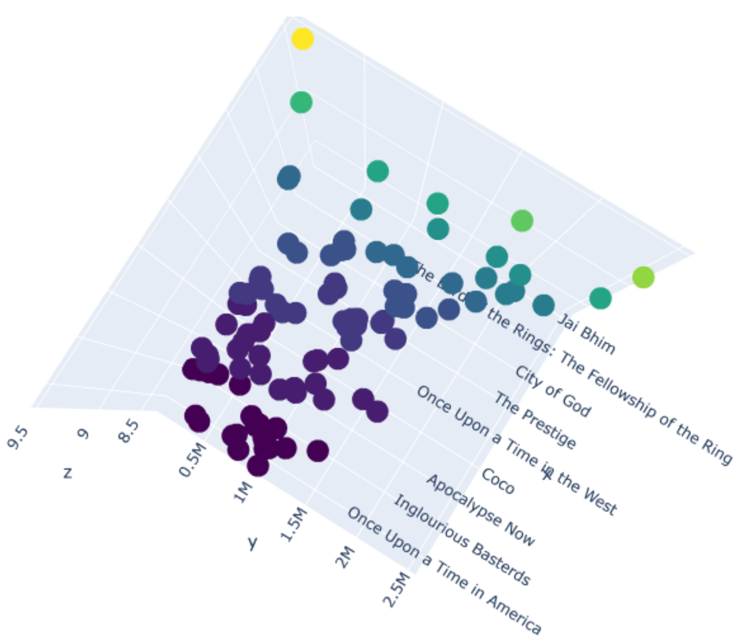
```
trace2 = go.Scatter(
    x = df.index,
    y = df.IMDb_votes,
    mode ="markers",
    name = "Number of users who rated a movie",
    marker =dict (
        color = "rgb(168, 229, 183)",
        size = 10,
        line = dict(
            color = "rgb(57, 4, 57)",
            width = 2
        )
    ),
    text = df.name
)
data1 = [trace1,trace2]
layout = dict( title = " Average rating and Count of user
votes",hovermode = "x",
              xaxis = {'showgrid' : False},yaxis = {'showgrid' :
False},

paper_bgcolor='rgba(0,0,0,0)',plot_bgcolor='rgba(0,0,0,0)')
fig = dict ( data = data1 , layout = layout)
iplot( fig)
```



3D Scatter plot

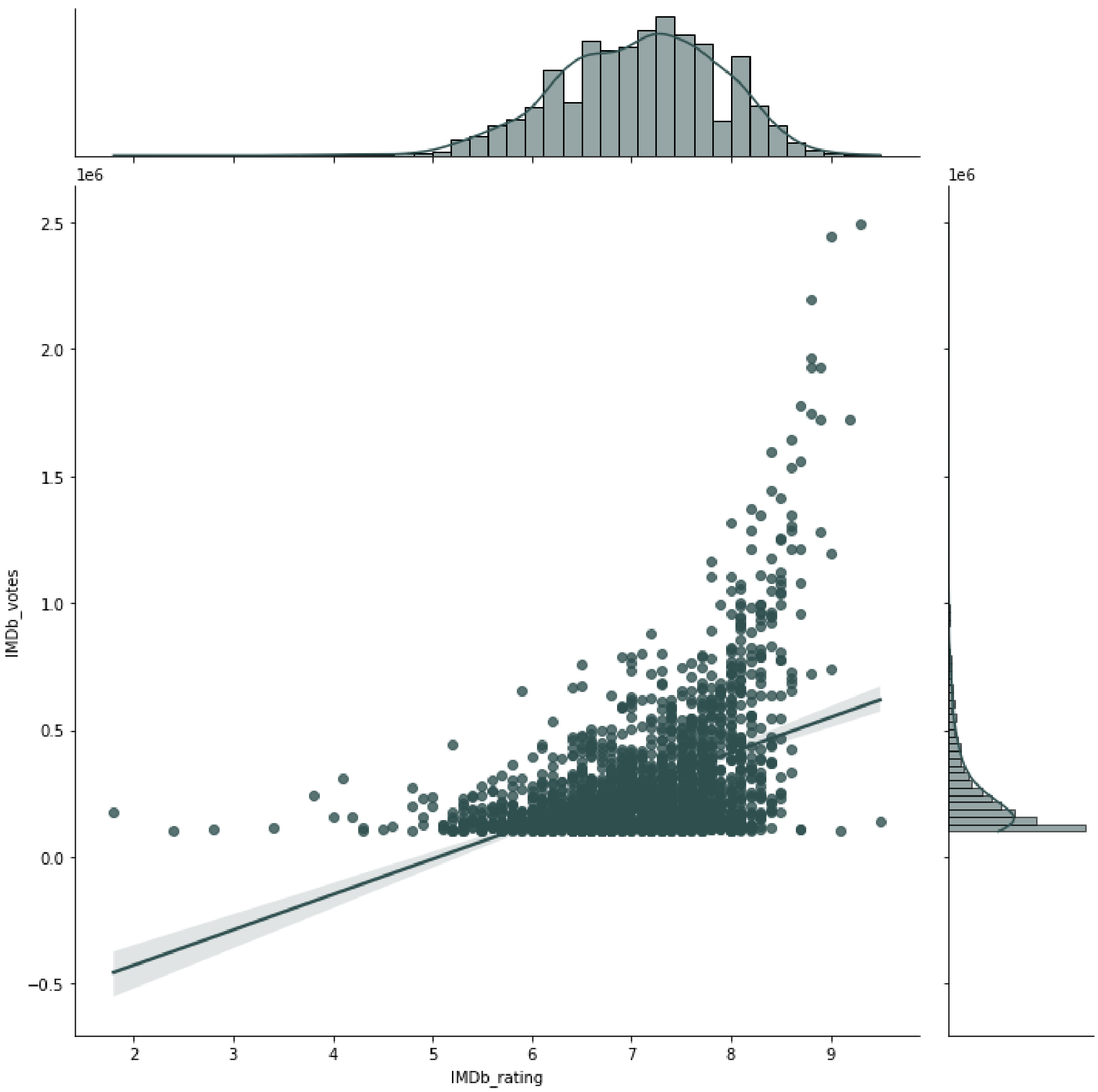
```
trace1=go.Scatter3d(
    x =data.name.head(100),
    y = data.IMDb_votes.head(100),
    z= data.IMDb_rating.head(100),
    mode = "markers",
    marker= dict(
        color= data.IMDb_rating.head(100),
        colorscale = "Viridis",
        size = 10
    )
)
data5 = [trace1]
layout = go.Layout(
```

```
    margin = dict (
        l=0,
        r=0,
        b=0,
        t=0
    )
)

fig = dict( data = data5,layout = layout)
iplot(fig)
```

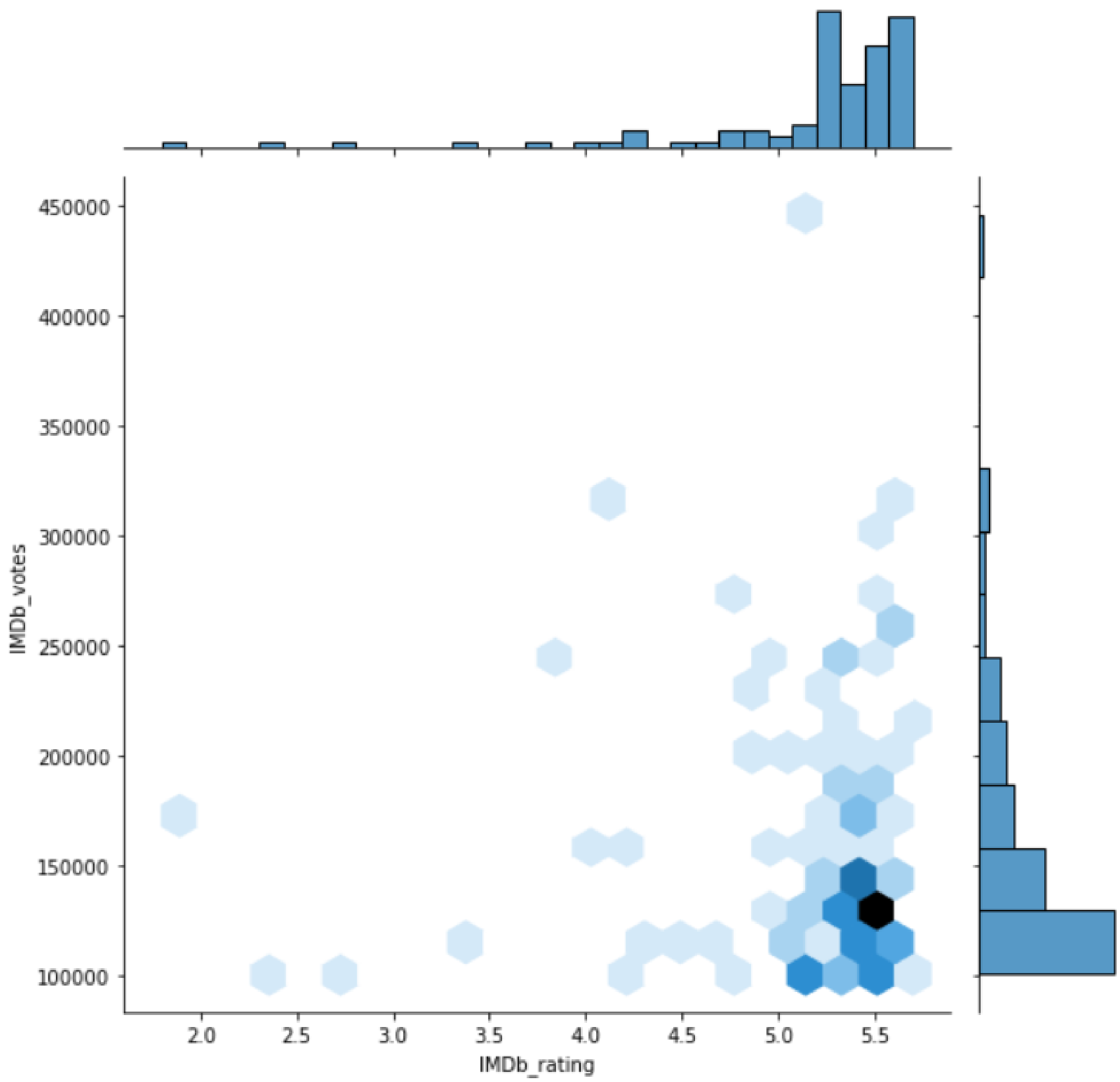


Combined plots

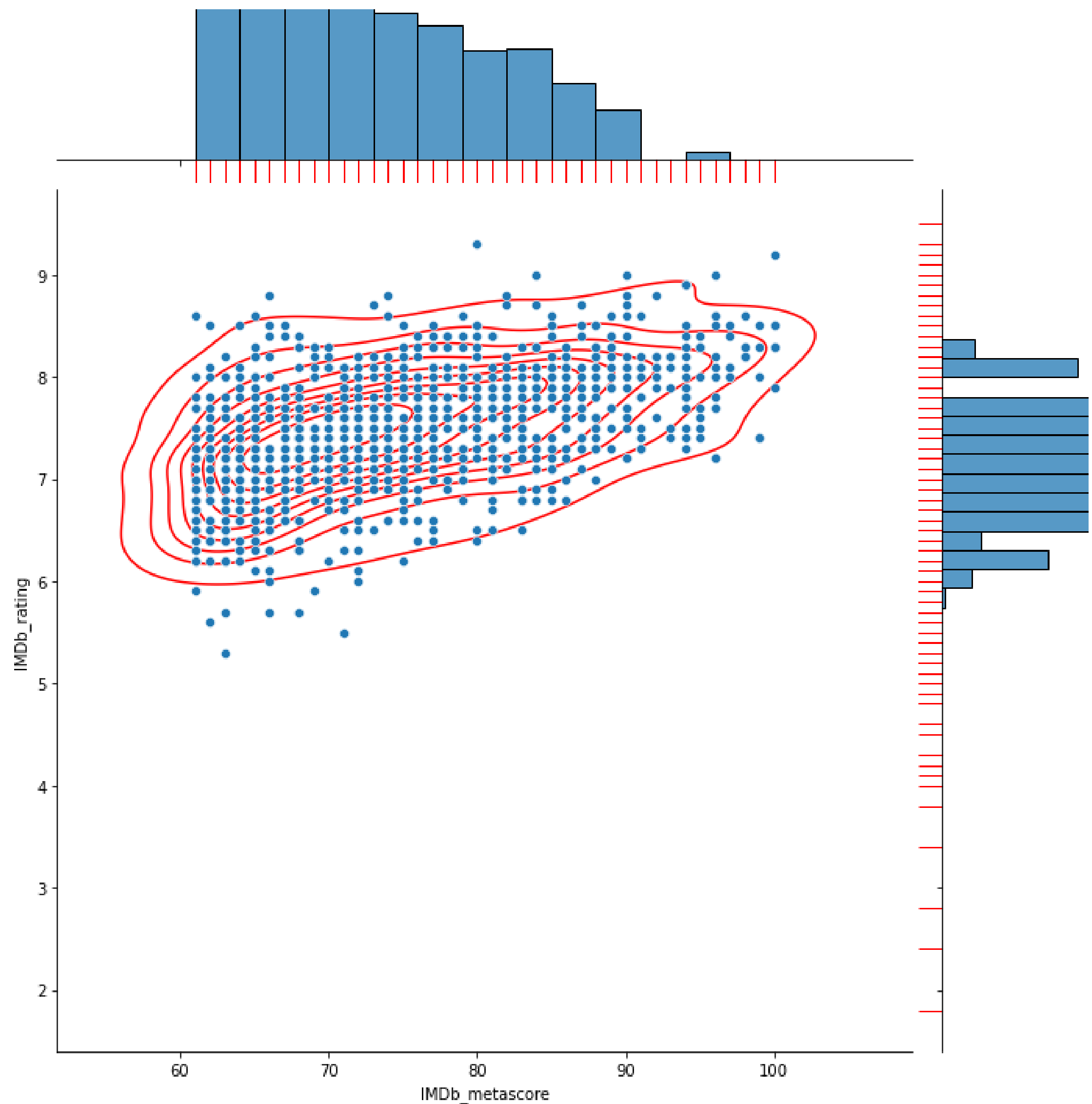
```
import scipy.stats as stats

pearsonreg = sns.jointplot(x = data.IMDb_rating , y = data.IMDb_votes,kind ="reg",color="DarkSlateGrey",height=10)

pearsonreg = sns.despine()

plt.show()
```



```
sns.jointplot(x = data.IMDb_rating.tail(100),y =
data.IMDb_votes.tail(100),kind ="hex",height=8)

plt.show()
```

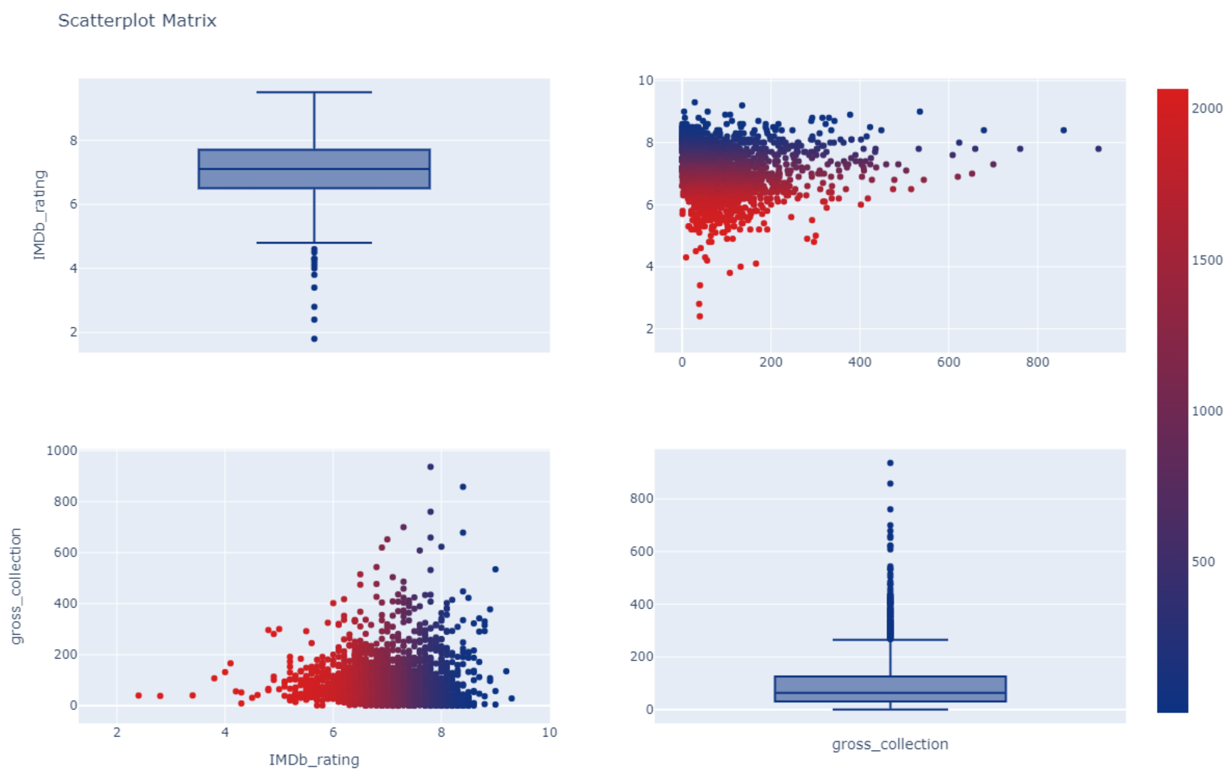



```
g = sns.jointplot(data=data, x="IMDb_metascore",
y="IMDb_rating",height=10)

g.plot_joint(sns.kdeplot, color="r", zorder=0, levels=10)

g.plot_marginals(sns.rugplot, color="r", height=-.15, clip_on=False)
```




```
import plotly.figure_factory as ff

data1 = data.loc[:,["IMDb_rating","gross_collection"]]

data1["index"] = np.arange(1,len(data1)+1)



fig = ff.create_scatterplotmatrix(data1,diag= "box", index =
"index",colormap = "Portland",colormap_type = "cat",
                                height = 800,width=1200)
```

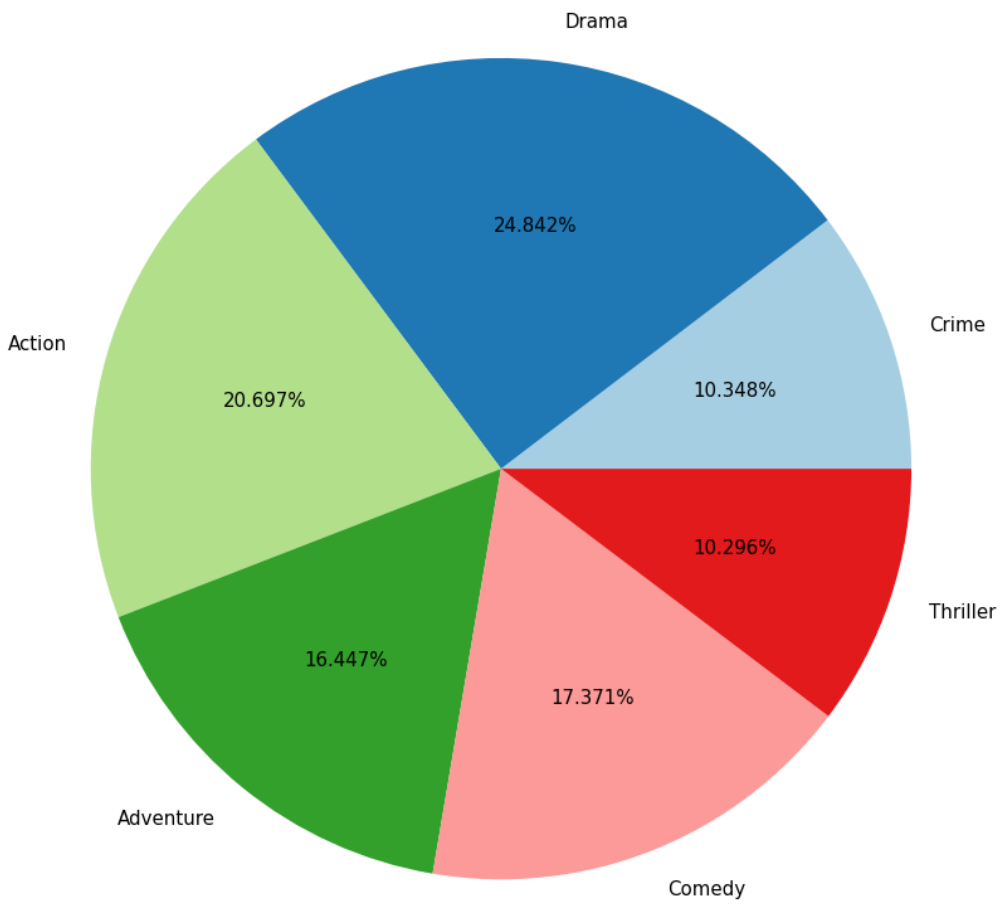
```
iplot(fig)
```



Pie Chart

```
df = data.genres.copy()

list_kind = df.str.split(", ")
a = []
for each in list_kind:
    for i in each:
        a.append(i)
```

```
keys=[]
values=[]
c=[]

f= dict(Counter(a))

for key,value in f.items() :
    if value > 300 and key != "":
        keys.append(key)
        values.append(value)

labels = keys
colors = sns.color_palette("Paired",6)
explode =[0,0,0,0,0,0]
sizes= values

plt.figure(figsize = (15,15))
plt.pie(sizes,explode = explode,labels=labels,colors =
colors,autopct='%.3f%%',textprops= {"fontsize": 15},shadow = False)
plt.title=("Top 6 categories of Genres")
plt.show()
```

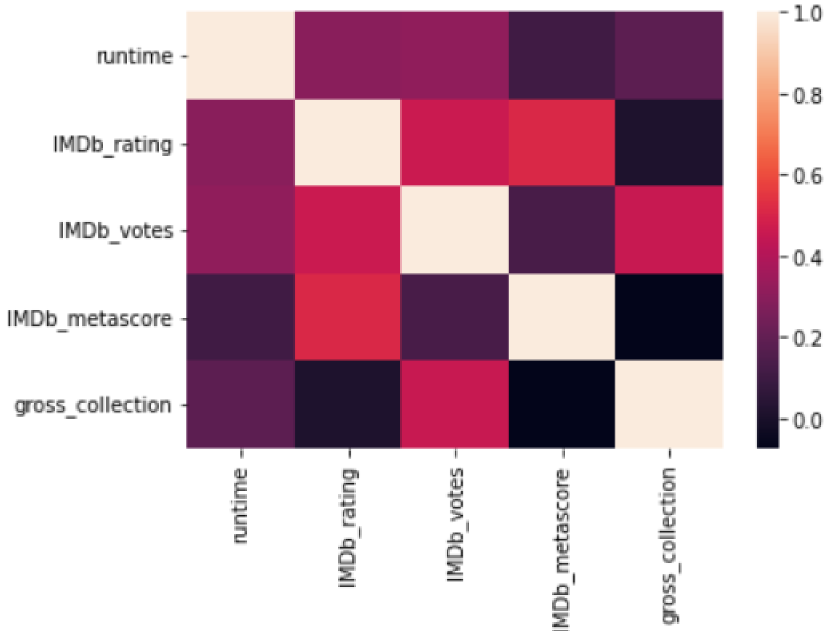


Heatmaps

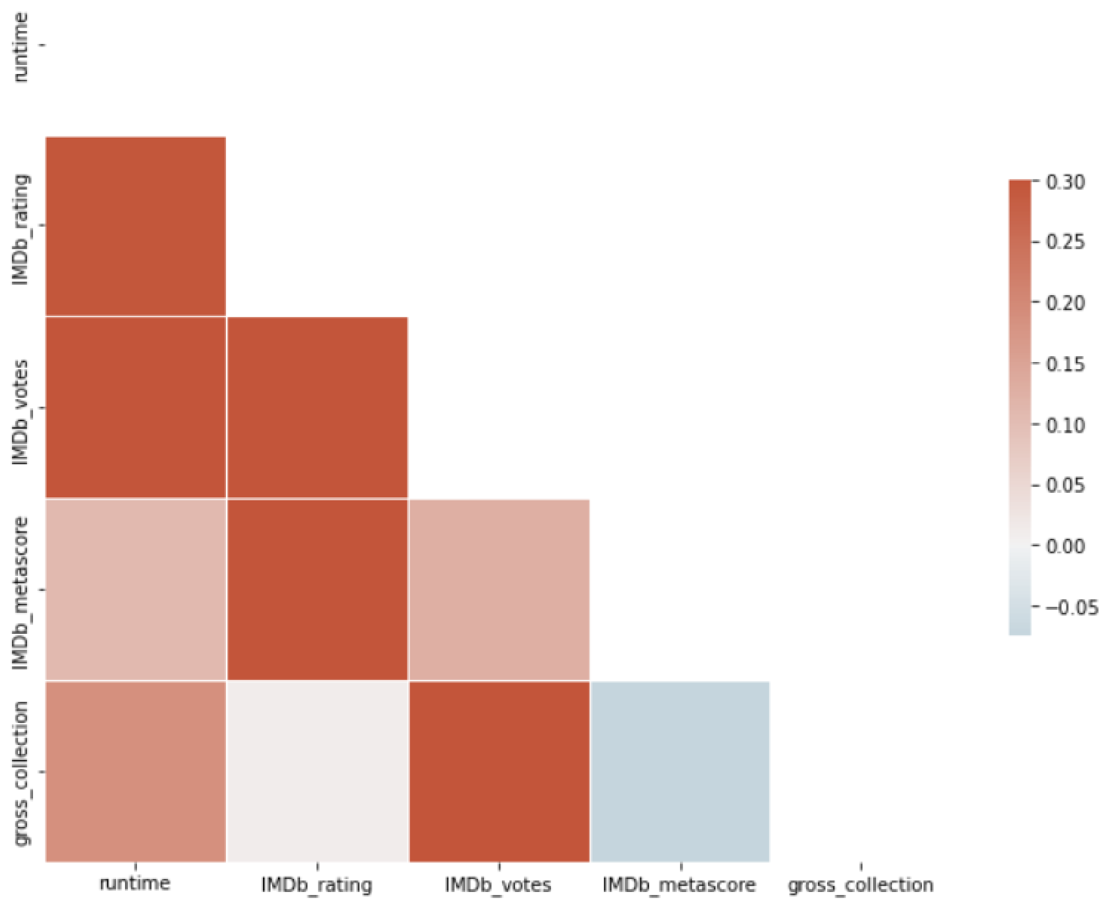
```
data.corr()
```

```
...                   runtime  IMDb_rating  IMDb_votes  IMDb_metascore  \
    runtime           1.000000     0.296527    0.314704        0.108212
    IMDb_rating       0.296527     1.000000    0.462967        0.513270
    IMDb_votes        0.314704     0.462967    1.000000        0.128917
    IMDb_metascore    0.108212     0.513270    0.128917        1.000000
    gross_collection  0.185052     0.012312    0.452807       -0.073461

                      gross_collection
    runtime                   0.185052
    IMDb_rating               0.012312
    IMDb_votes                0.452807
    IMDb_metascore           -0.073461
    gross_collection          1.000000
```

```
corr = data.corr()
sns.heatmap(corr)
plt.show()
```

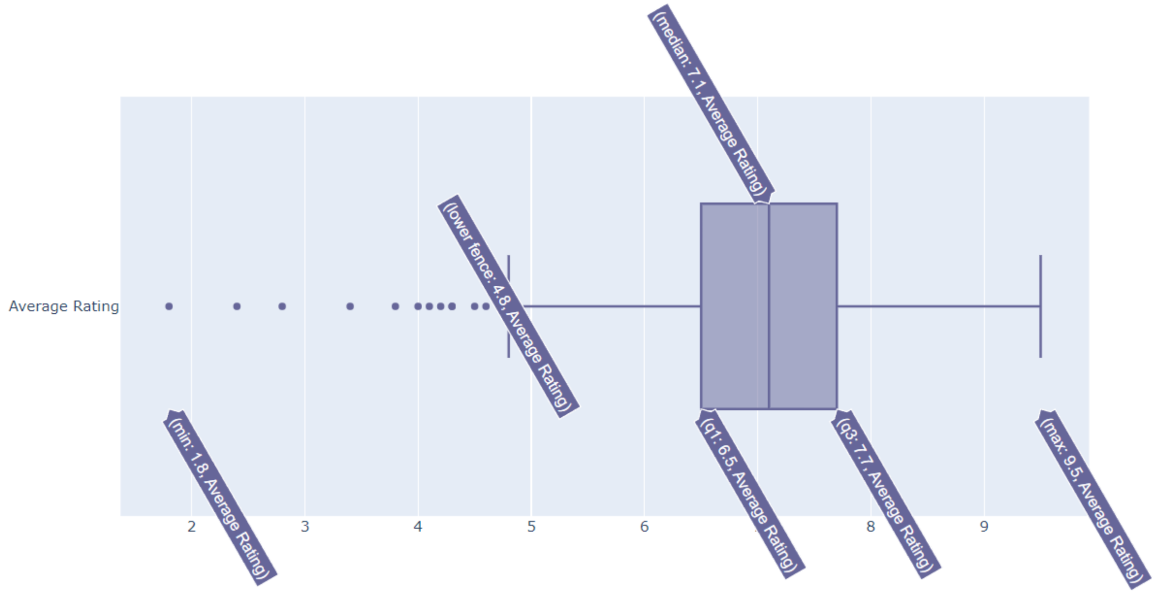



```
mask = np.triu(np.ones_like(corr, dtype=bool))
f, ax = plt.subplots(figsize=(11, 9))
cmap = sns.diverging_palette(230, 20, as_cmap=True)

sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

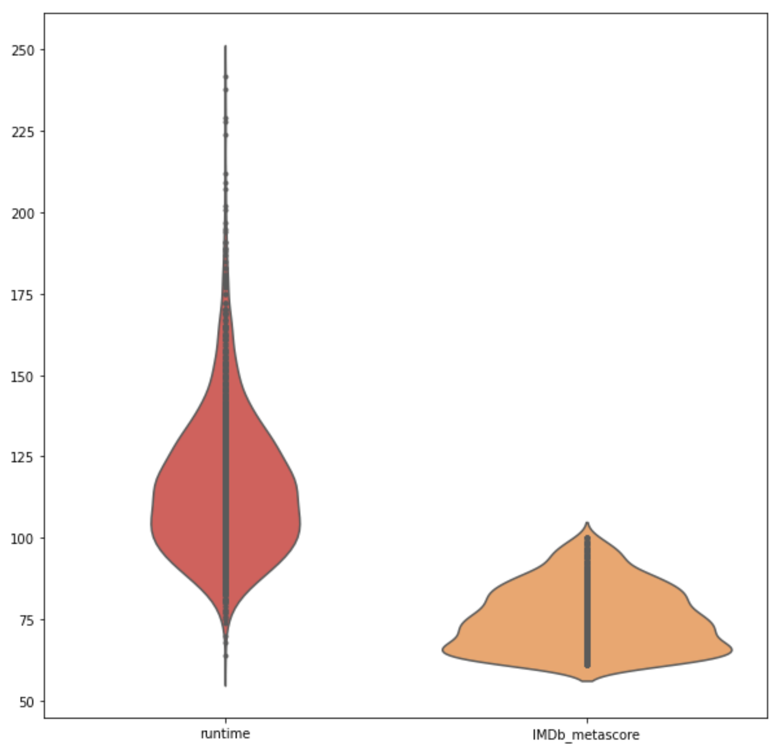


Box Plot and Violin Graphs

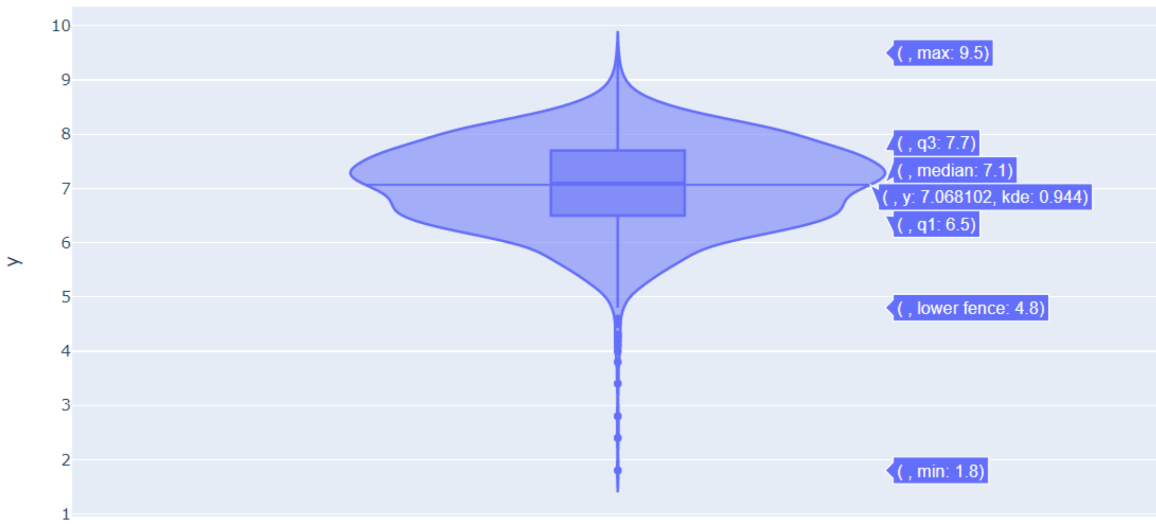
```
trace1= go.Box(
x =data.IMDb_rating,
name = "Average Rating",
marker = dict ( color = "#666699"),

)

iplot([trace1])
```




```
df = data.loc[:,["runtime","IMDb_metascore"]].copy()
 plt.figure( figsize = (10,10))
 sns.violinplot(data=df , palette =
 sns.color_palette("Spectral"),inner ="points")
 plt.show()
```



```
import plotly.express as px

df4 = data["runtime"]
fig = px.violin(df4, y=data["IMDb_rating"],box=True)
fig.show()
```

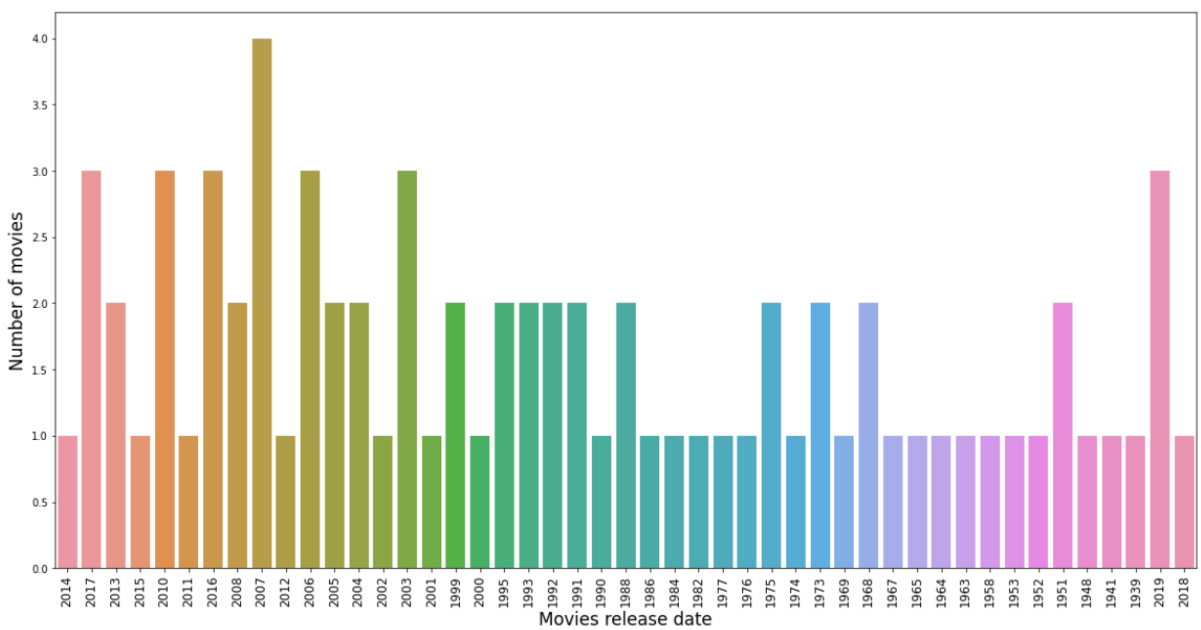


Histograms

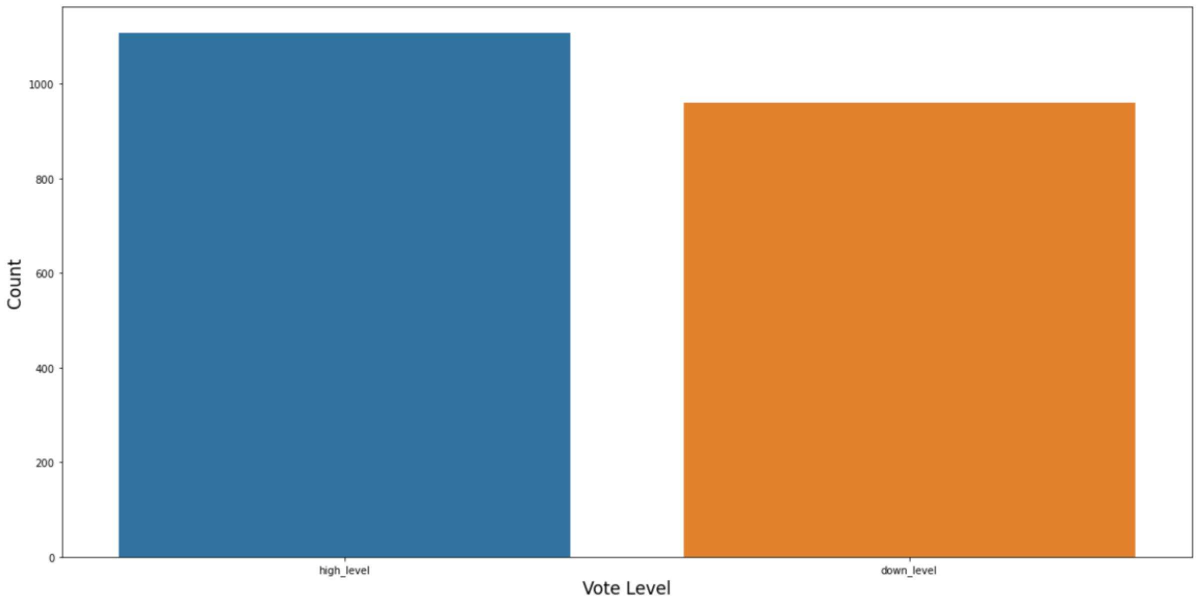
```
data5 = pd.read_csv("IMDb_movies_dataset.csv")
# Index of moives from 237 to 314(random)
last_ten_year_release_analysis = data5['year'].iloc[237:313]
plt.figure(figsize = (20,10))
print(last_ten_year_release_analysis)
sns.countplot(x = last_ten_year_release_analysis)
plt.xticks(rotation= 90,fontsize = 12)
plt.xlabel("Movies release date",fontsize =17)
plt.ylabel("Number of movies",fontsize = 17)
```

```
plt.show()
```

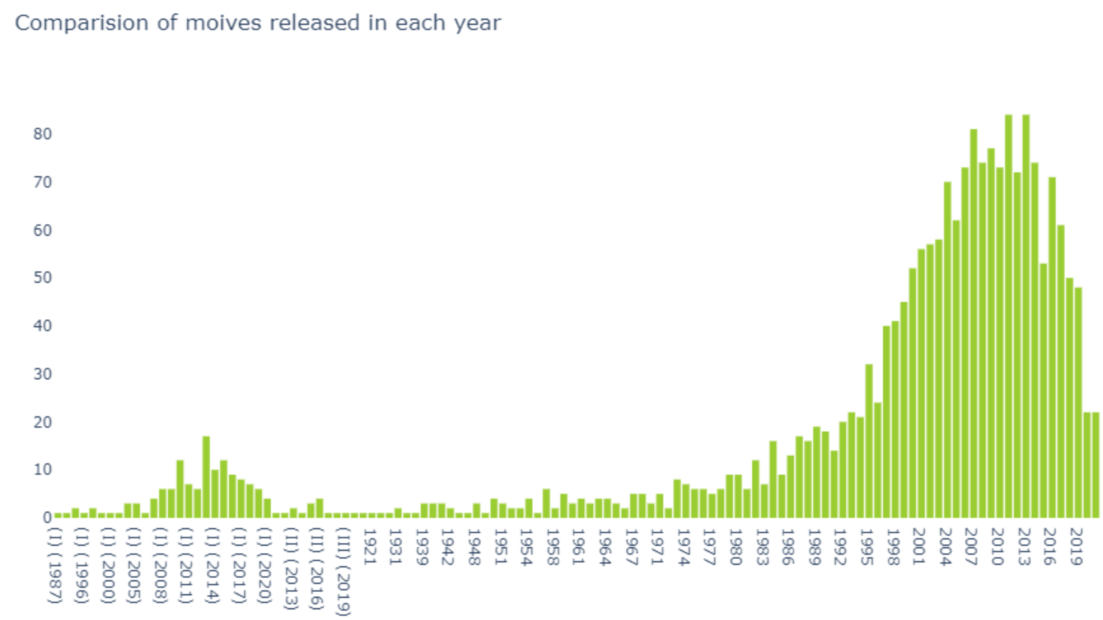
```
... 237    2014
    238    2017
    239    2013
    240    2015
    241    2010
           ...
    308    2018
    309    2016
    310    2016
    311    2017
    312    2019
    Name: year, Length: 76, dtype: object
```




```
plt.figure(figsize = (20,10))
sns.countplot(x = data.vote_level)
plt.xlabel("Vote Level",fontsize =17)
plt.ylabel("Count",fontsize = 17)
plt.show()
```



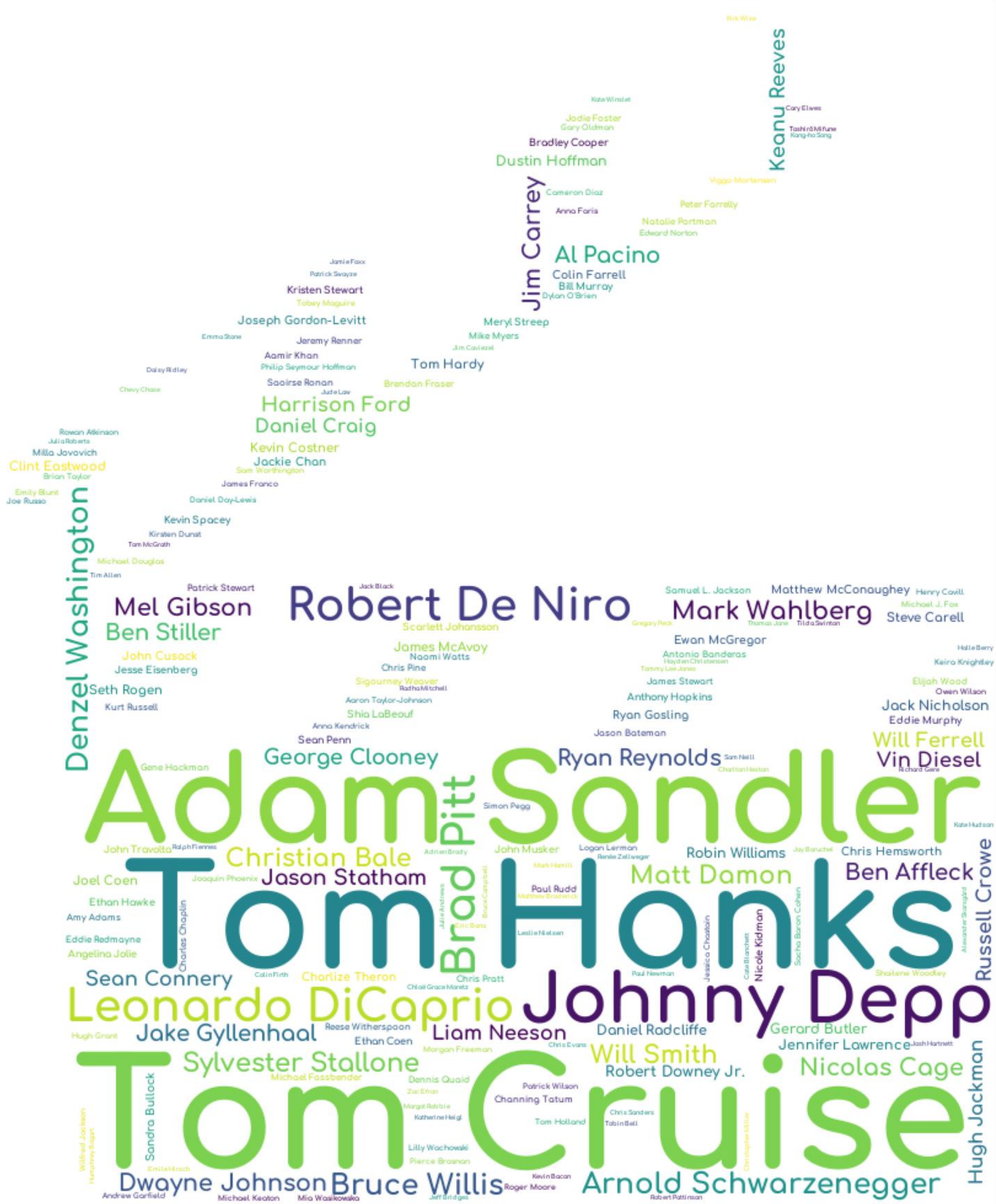

```
trace1= go.Histogram(
    x = data5['year'].sort_values(ascending=True),
    marker = dict ( color = "yellowgreen")
)


layout1 = dict( title = "Comparision of moives released in each
year",hovermode = "x",
              xaxis = {'showgrid' : False},yaxis = {'showgrid' :
False},

paper_bgcolor='rgba(0,0,0,0)',plot_bgcolor='rgba(0,0,0,0)')


data2 = [trace1]
fig =go.Figure( data = data2,layout = layout1)
iplot(fig)
```



Word cloud

```
from PIL import Image

names = data5['lead_actor'].value_counts()


mask = np.array(Image.open('movie-clip-icon.png'))

font_path =
'C:/Users/ekave/AppData/Local/Microsoft/Windows/Fonts/Comfortaa-
Bold.ttf'

plt.subplots( figsize = (32,32))

wordcould = WordCloud(

    background_color= "white", mask = mask, font_path = font_path ,

    width= 2160,

    height = 720,

).generate_from_frequencies(names)

plt.imshow(wordcould)
```

```
plt.axis("off")

plt.show()
```



Movie Recommendation

Content Based Recommendation

```
import numpy as np

import pandas as pd

import difflib

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine_similarity


smallmoviesdataset = pd.read_csv('movies.csv')
```

Selecting the required features from the dataset

```
select_features = ['genres','keywords','tagline','cast','director']

print(select_features)


['genres', 'keywords', 'tagline', 'cast', 'director']
```

Replacing the null values with null strings

```
for feature in select_features:

    smallmoviesdataset[feature] =
smallmoviesdataset[feature].fillna('')
```

Combining the selected features

```
combined_features = smallmoviesdataset['genres']+'
'+smallmoviesdataset['keywords']+' '+smallmoviesdataset['tagline']+'
'+smallmoviesdataset['cast']+' '+smallmoviesdataset['director']
```

Converting the text data to feature vectors

```
vectorizer = TfidfVectorizer()


feature_vectors = vectorizer.fit_transform(combined_features)
```

Getting the similarity scores using cosine similarity

```
similarity = cosine_similarity(feature_vectors)


movie_name = input(' Enter your favourite movie name : ')


list_of_all_titles = smallmoviesdataset['title'].tolist()


find_close_match = difflib.get_close_matches(movie_name,
list_of_all_titles)


close_match = find_close_match[0]


index_of_the_movie = smallmoviesdataset[smallmoviesdataset.title ==
close_match]['index'].values[0]


similarity_score = list(enumerate(similarity[index_of_the_movie]))


sorted_similar_movies = sorted(similarity_score, key = lambda
x:x[1], reverse = True)


print('Movies suggested for you : \n')
```

```python
i = 1

for movie in sorted_similar_movies:
    index = movie[0]
    title_from_index = smallmoviesdataset[smallmoviesdataset.index==index]['title'].values[0]
    if (i<11):
        print(i, '.',title_from_index)
        i+=1
```

```
...   Enter your favourite movie name : var
     Movies suggested for you :

     1 . War
     2 . Punch-Drunk Love
     3 . The One
     4 . The Limey
     5 . The X Files: I Want to Believe
     6 . Lone Wolf McQuade
     7 . The Forbidden Kingdom
     8 . Journey 2: The Mysterious Island
     9 . The Mummy: Tomb of the Dragon Emperor
     10 . Punisher: War Zone
```

CONCLUSION:

Making graphs is about audiences and reviews, and it involves standardizing results that are useful for discovering and comparing patterns. Furthermore, we discover that there are some consistent relationships between audiences and reviews.

Our approach could successfully demonstrate the existence phenomenon with visualization techniques. We successfully examined a small dataset of movies from the IMDb web portal as part of this project. We've also used diversified graphs to illustrate data relationships. Furthermore, we have accomplished content-based filtering in order to recommend movies to the user.

FUTURE WORK:

As previously indicated, we concentrated on content-based filtering. In the future, we may work on collaborative filtering and hybrid filtering to improve the results and encourage users to explore other popular films, genres, and categories apart from the movies they like.

REFERENCES:

https://www.researchgate.net/publication/331966843_Content-Based_Movie_Recommendation_System_Using_Genre_Correlation

https://ieeexplore.ieee.org/document/4272022?arnumber=4272022

https://ieeexplore.ieee.org/document/6741434

https://www.raco.cat/index.php/ELCVIA/article/download/373942/467477/

https://www.researchgate.net/publication/282133920_Predicting_Movie_Success_Based_on_IMDB_Data

https://www.ijrte.org/wp-content/uploads/papers/v7i4s/E2008017519.pdf

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0158423

https://seaborn.pydata.org/tutorial.html