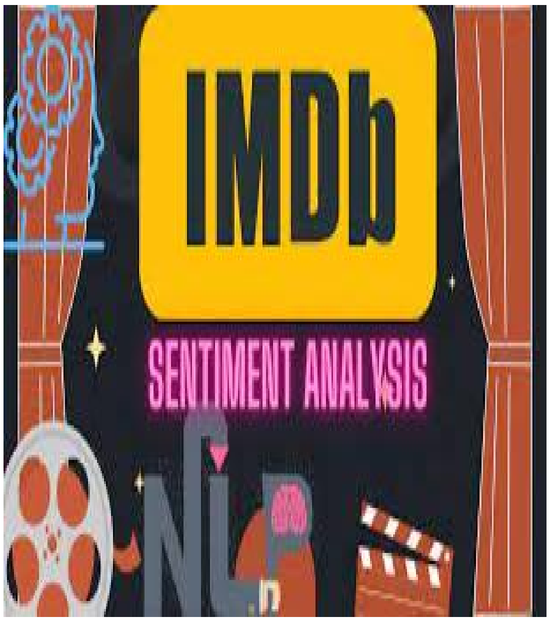
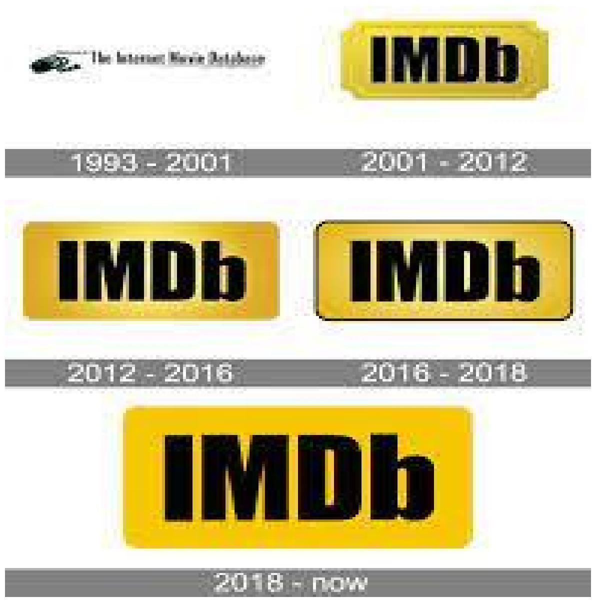
TITLE: IMDb Score Prediction using Data Science

PHASE 4: DEVELOPMENT PART 2





INTRODUCTION:

IMDb scores are determined by user ratings and can change over time as more users rate the movie or show.

The problem is to develop a machine learning model to predict the IMDb scores of movies available on Films based on their genre, premiere date, runtime, and language.

This project involves data collection, data prepossessing, feature engineering, clustering algorithms, visualization, and interpretation of results.

The model aims to accurately estimate the popularity of movies to assist users in discovering highly rated films that align with their preferences.

WORKS DONE IN PREVIOUS PHASES:

DEFINITION PHASE:

These phases can be executed using three parts

- Loading and Pre-processing data
- Training and Testing data
- Model testing and Displaying Output

These involves data preprocessing, feature engineering, model selection, training, and evaluation.

INNOVATION PHASE :

In the innovation phase of our IMDb scores prediction project, you can explore advanced techniques and methods to improve the accuracy of your IMDbPro uses proprietary algorithms that take into account several measures of popularity for people, titles and companies. The primary measure is who and what people are looking at on IMDb.

IMPORTING LIBRARIES:

We importing the necessary Python libraries, such as

- Pandas for data manipulation
- NumPy for analysis,
- Matplotlib for visualization.

Loading the dataset:

- To load data points from a file (e.g., a CSV file), you can use the **pd.read.csv()** function.

This dataset consists of all Netflix original films released as of June 1st, 2021. Additionally, it also includes all Netflix documentaries and specials. The dataset available on Kaggle.

Dataset consist of:

- Title
- Genre
- Premiere date
- Runtime
- IMDB scores
- Languages

Dataset link : https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores/

Here's the code for predicting the IMDb scores,

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px
df= pd.read_csv("/kaggle/input/netflix-original-films-imdb-scores/NetflixOriginals.csv",encoding = "ISO-8859-1")
df
```

Title	Genre	Premiere	Runtime	IMDB Score	Language	
0	Enter the Anime	Documentary	August 5, 2019	58	2.5	English/Japanese
1	Dark Forces	Thriller	August 21, 2020	81	2.6	Spanish
2	The App	Science fiction/Drama	December 26, 2019	79	2.6	Italian
3	The Open House	Horror thriller	January 19, 2018	94	3.2	English
4	Kaali Khuhi	Mystery	October 30, 2020	90	3.4	Hindi
...	...	...	...	...	...	...
579	Taylor Swift: Reputation Stadium Tour	Concert Film	December 31, 2018	125	8.4	English
580	Winter on Fire: Ukraine's Fight for Freedom	Documentary	October 9, 2015	91	8.4	English/Ukranian/Russian
581	Springsteen	One-man	December 16,	153	8.5	English

Title	Genre	Premiere	Runtime	IMDB Score	Language	
	on Broadway	show	2018			
582	Emicida: AmarElo - It's All For Yesterday	Documentary	December 8, 2020	89	8.6	Portuguese
583	David Attenborough: A Life on Our Planet	Documentary	October 4, 2020	83	9.0	English

```
df=describe()
```

Runtime	IMDB Score	
count	584.000000	584.000000
mean	93.577055	6.271747
std	27.761683	0.979256
min	4.000000	2.500000
25%	86.000000	5.700000
50%	97.000000	6.350000
75%	108.000000	7.000000
max	209.000000	9.000000

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

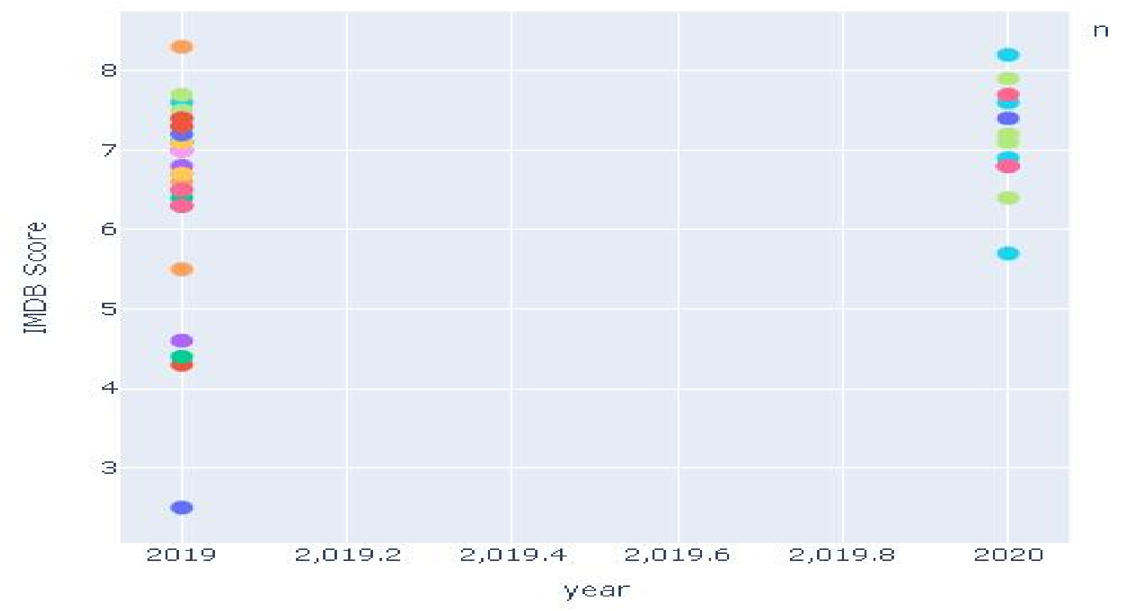
```
Title        0
Genre        0
Premiere     0
Runtime      0
IMDB Score   0
Language     0
dtype: int64
```

```
df['Premiere'] = pd.to_datetime(df['Premiere'])
df['year']    = df['Premiere'].dt.year
df['month']   = df['Premiere'].dt.month_name()
df['weekday'] = df['Premiere'].dt.day_name()
df.head()
```

	Title	Genre	Premiere	Runtime	IMDB Score	Language	year	month	weekday
0	Enter the Anime	Documentary	2019-08-05	58	2.5	English/Japanese	2019	August	Monday
1	Dark Forces	Thriller	2020-08-21	81	2.6	Spanish	2020	August	Friday
2	The App	Science fiction/Drama	2019-12-26	79	2.6	Italian	2019	December	Thursday

	Title	Genre	Premiere	Runtime	IMDB Score	Language	year	month	weekday
3	The Open House	Horror thriller	2018-01-19	94	3.2	English	2018	January	Friday
4	Kaali Khuhi	Mystery	2020-10-30	90	3.4	Hindi	2020	October	Frida

```
df_temp=df.groupby(['Runtime','Title','Language']).mean().sort_
values(by='Runtime', ascending=False).reset_index().iloc[:,:3]
Fig= px.box(df, x= 'Runtime', hover_data = df[['Title','Language
']])fig.update_traces(quartilemethod="inclusive")fig.show()
df_doc = df[ ((df["year"]== 2019) |
            ((df["year"]== 2020) & ((df["month"] ==("January"))|
(df["month"] ==("February"))| (df["month"] ==("March"))| (df["
month"] ==("April")) | (df["month"] ==("May")) | (df["month"]
==("June")))) )
            & (df["Genre"]== "Documentary")  ]
 fig=px.scatter(df_doc, x='year', y='IMDB Score',color="month")
fig.update_traces(marker_size=10)
fig.show()
```



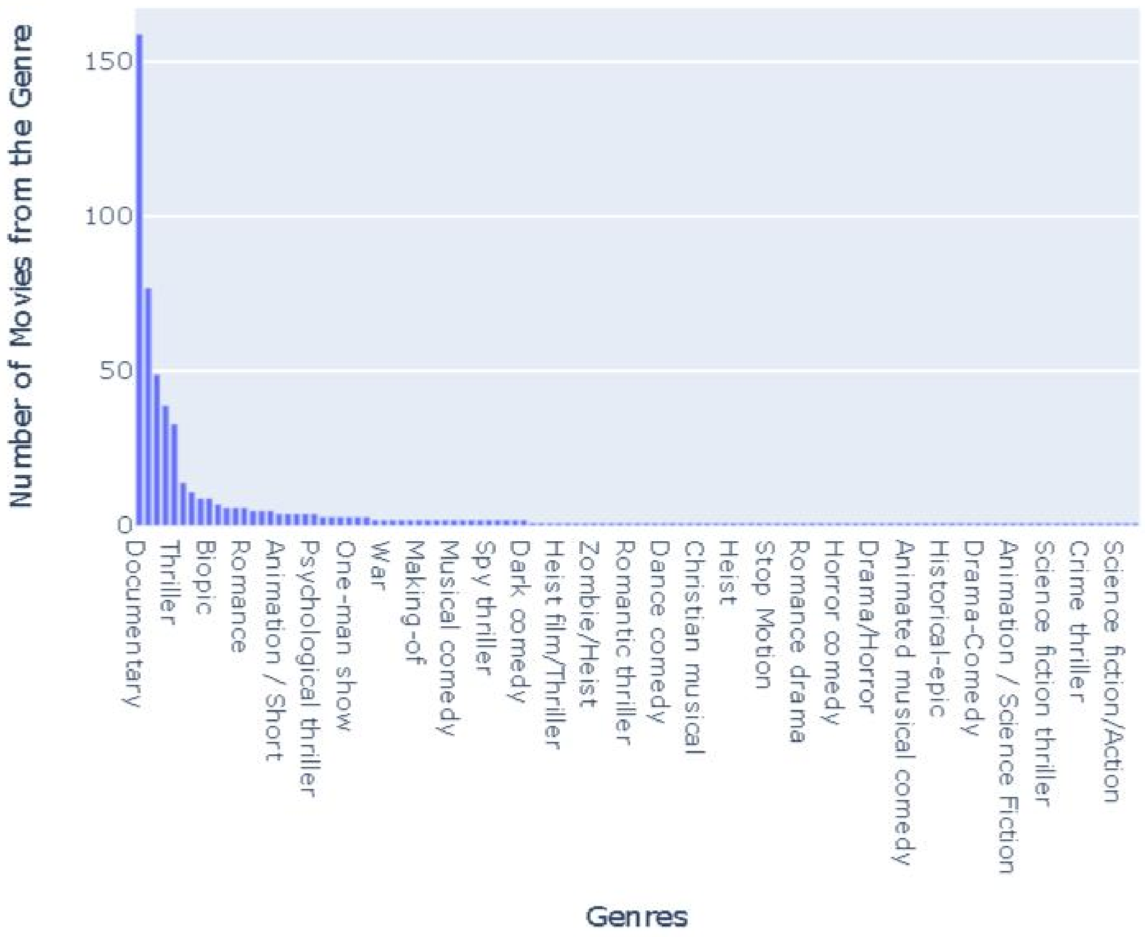
```
top_imdb_english= df[df['Language'] == "English"]
top_imdb_english = top_imdb_english.groupby(['Language','Genre','Title']).mean().sort_values(by=["IMDB Score"],ascending=False)[:10]
top_imdb_english
```

Language	Genre	Title	Runtime	IMDB Score	year
English	Documentary	David Attenborough: A Life on Our Planet	83.0	9.0	2020.0
	One-man show	Springsteen on Broadway	153.0	8.5	2018.0
	Concert Film	Ben Platt: Live from Radio City	85.0	8.4	2020.0

Language	Genre	Title	Runtime	IMDB Score	year
		Music Hall			
		Taylor Swift: Reputation Stadium Tour	125.0	8.4	2018.0
	Documentary	Cuba and the Cameraman	114.0	8.3	2017.0
		Dancing with the Birds	51.0	8.3	2019.0
		Seaspiracy	89.0	8.2	2021.0
	Animation/Christmas/Comedy/Adventure	Klaus	97.0	8.2	2019.0
	Documentary	Disclosure: Trans Lives on Screen	107.0	8.2	2020.0
		13th	100.0	8.2	2016.0

```
df_hindi = df[df["Language"] == "Hindi"]df_hindi.Runtime.value_counts()df_hindi.Runtime.mean()
115.78787878787878
```

```
df['Genre'].value_counts()
df['Genre'].value_counts().sum()
genre =df['Genre'].value_counts()

fig = px.bar(genre, x= genre.index, y=genre.values, labels={'y':'
Number of Movies from the Genre', 'index':'Genres'})
fig.update_layout(xaxis={'categoryorder':'total descending'})
fig.show()
```



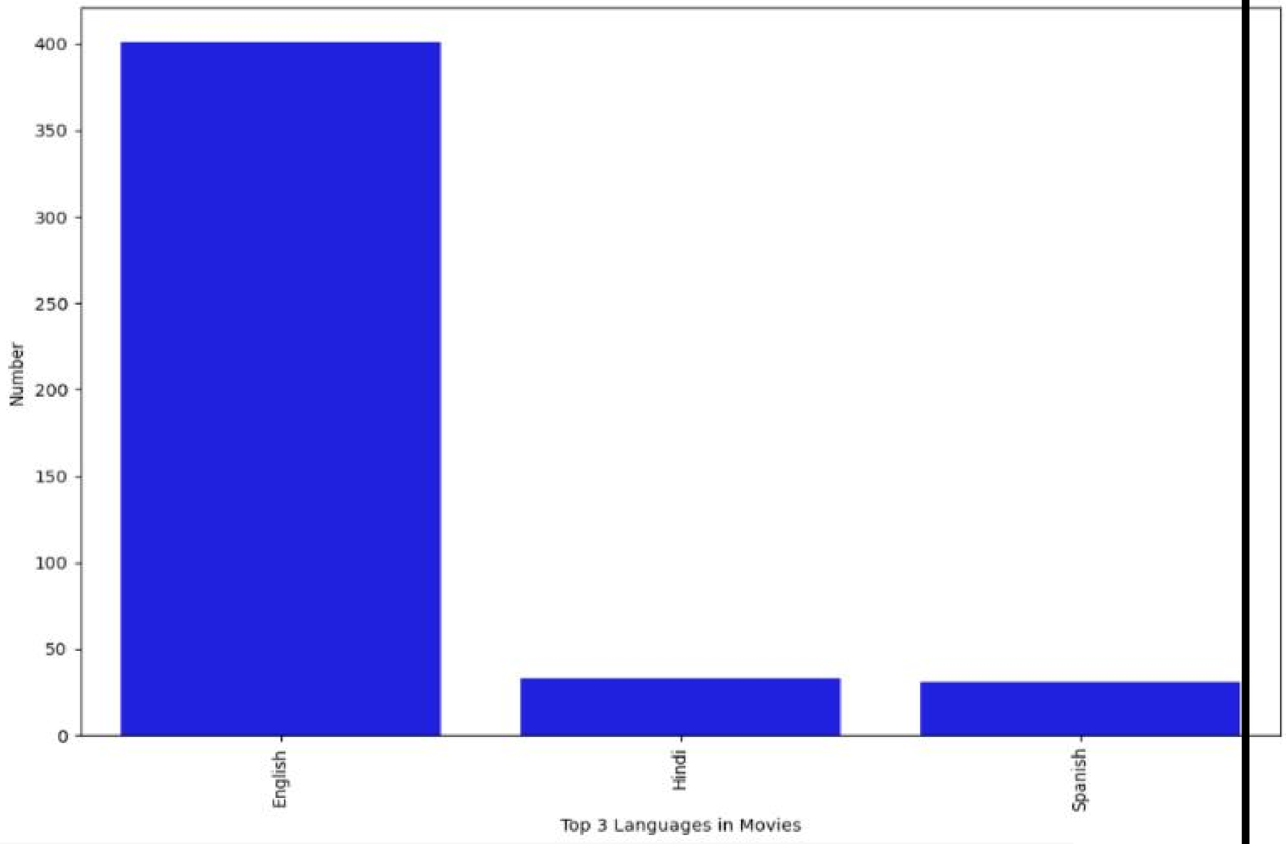
```
df.Language.unique()
df.Language.value_counts()
```

```
English                401
Hindi                   33
Spanish                 31
French                  20
Italian                 14
```

```
Portuguese                      12
Indonesian                       9
Japanese                         6
Korean                           6
German                           5
Turkish                          5
English/spanish                  5
Polish                           3
Dutch                            3
Marathi                          3
English/Hindi                    2
Thai                             2
English/Mandarin                 2
English/Japanese                 2
Filipino                         2
English/Russian                  1
Bengali                          1
English/Arabic                   1
English/ korean                  1
Spanish/English                  1
Tamil                            1
English/Akan                     1
Khmer/English/French             1
Swedish                          1
Georgian                         1
Thia/  english                   1
English/Taiwanese/Mandarin       1
English/Swedish                  1
Spanish/ catalan                 1
Spanish/Basque                   1
Norwagein                        1
Malay                            1
English/Ukranian/Russian         1
Name: Language, dtype: int64
```

```
df_top_lang = df.Language.value_counts().nlargest(3)
```

```
plt.figure(figsize=(12,8)
sns.barplot(x=df_top_lang.index,y=df_top_lang.values,data=df,
color='blue'
plt.xlabel('Top 3 Languages in Movies')
plt.xticks(rotation=90)
plt.ylabel('Number')
plt.show()
```



```
df_temp=df.sort_values(by='IMDB Score', ascending=False).r
eset_index().iloc[:13,:]

fig, ax = plt.subplots(1,1, figsize = (15, 6), constrained_layout
= True)ax = sns.barplot(x = 'Title', y = 'IMDB Score', data = df
_temp, hue = 'Genre');

for i in ax.patches:

   ax.text(x = i.get_x() + i.get_width()/2, y = i.get_height()+0.1,

          s = f"{i.get_height()}",

          ha = 'center', size = 14, weight = 'bold', rotation = 0, col
or = 'white',
            bbox=dict(boxstyle="circle,pad=0.5", fc='lightblue', ec
="lightblue", lw=2));
```

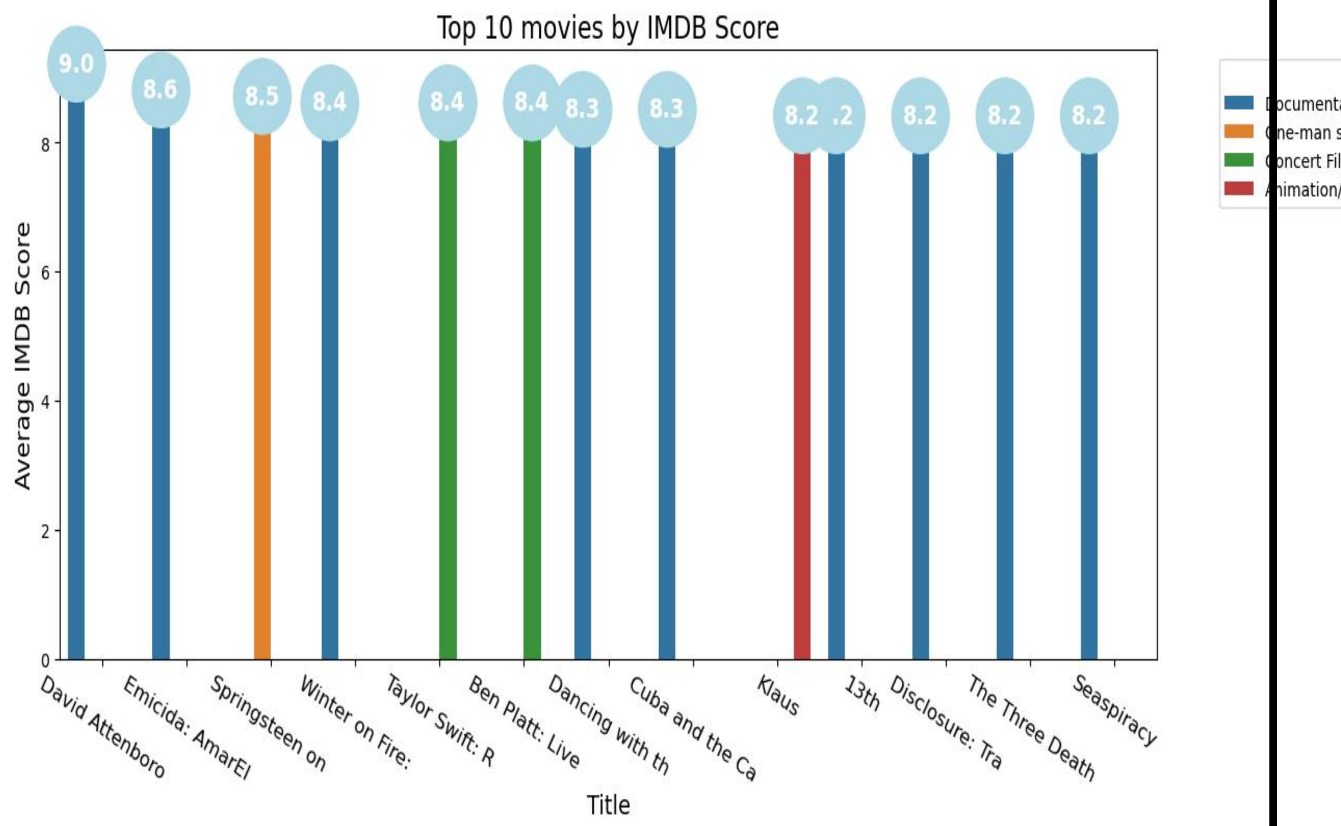
```
df[['IMDB Score','Runtime']].corr()
```

	IMDB Score	Runtime
IMDB Score	1.000000	-0.040896
Runtime	-0.040896	1.000000

```
fig = px.scatter(df, x='IMDB Score', y='Runtime')fig.show()
```

```
df_temp=df.groupby(['Genre']).mean(['IMDB
rating']).sort_values(by='IMDB Score',
ascending=False).reset_index().iloc[:10,:]
```

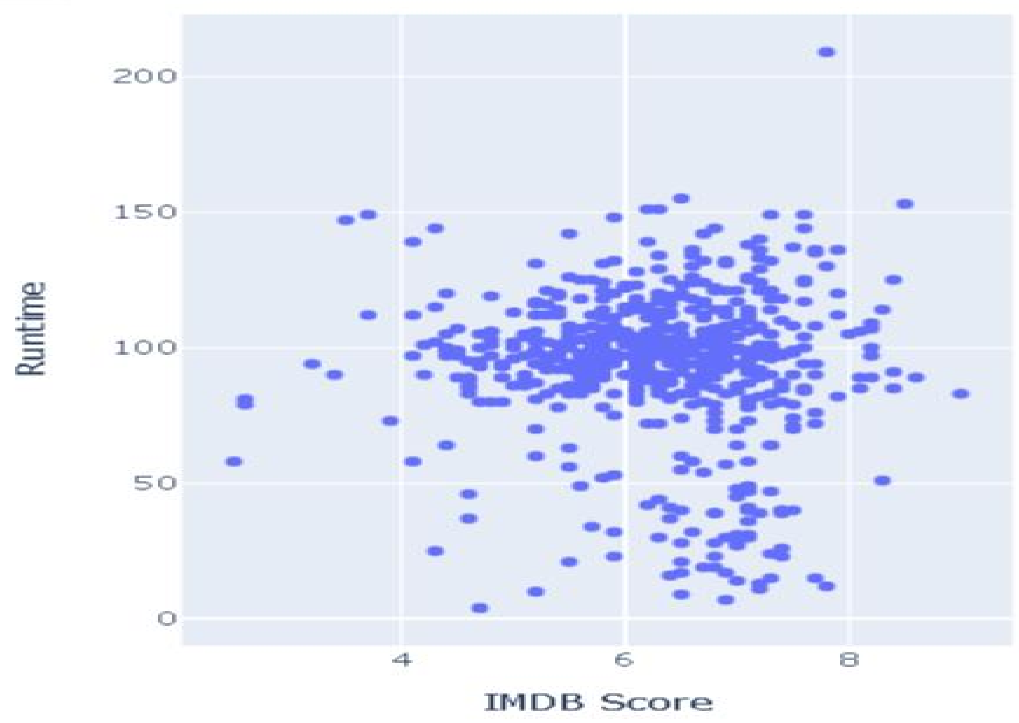
```
fig, ax = plt.subplots(1,1, figsize = (10, 6), constrained_layout
= True)ax = sns.barplot(x = 'Genre', y = 'IMDB Score', data = d
f_temp, color = 'violet')

for i in ax.patches:

    ax.text(x = i.get_x() + i.get_width()/2, y = i.get_height()/2,

            s = f"{round(i.get_height(),1)}",

            ha = 'center', size = 14, weight = 'bold', rotation = 0, col
or = 'white',

            bbox=dict(boxstyle="round,pad=0.5", fc='pink', ec="pi
nk", lw=2))
ax.set_xlabel('Title', fontsize=14)
ax.set_ylabel('Average IMDB Score', fontsize=14)ax.set_xtickla
bels([i[:15] for i in df_temp['Title'].unique()], fontsize=12, rotati
on = -30)
plt.title('Top 10 movies by IMDB Score', fontsize=16)
plt.legend(title='Gerne', bbox_to_anchor=(1.05, 1), loc='upper le
ft');
```



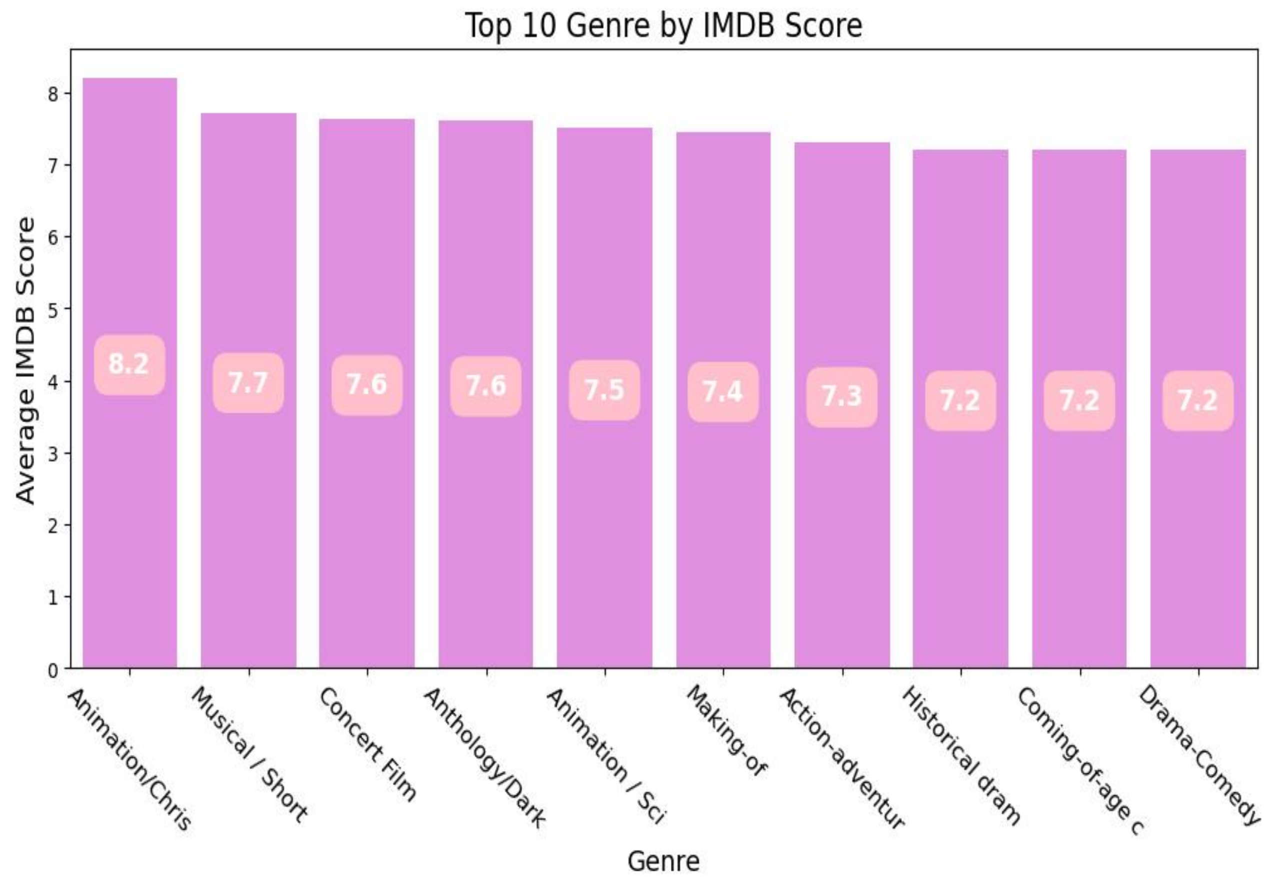
```
df[['IMDB Score','Runtime']].corr()
```

IMDB Score	1.000000	-0.040896
Runtime	-0.040896	1.000000

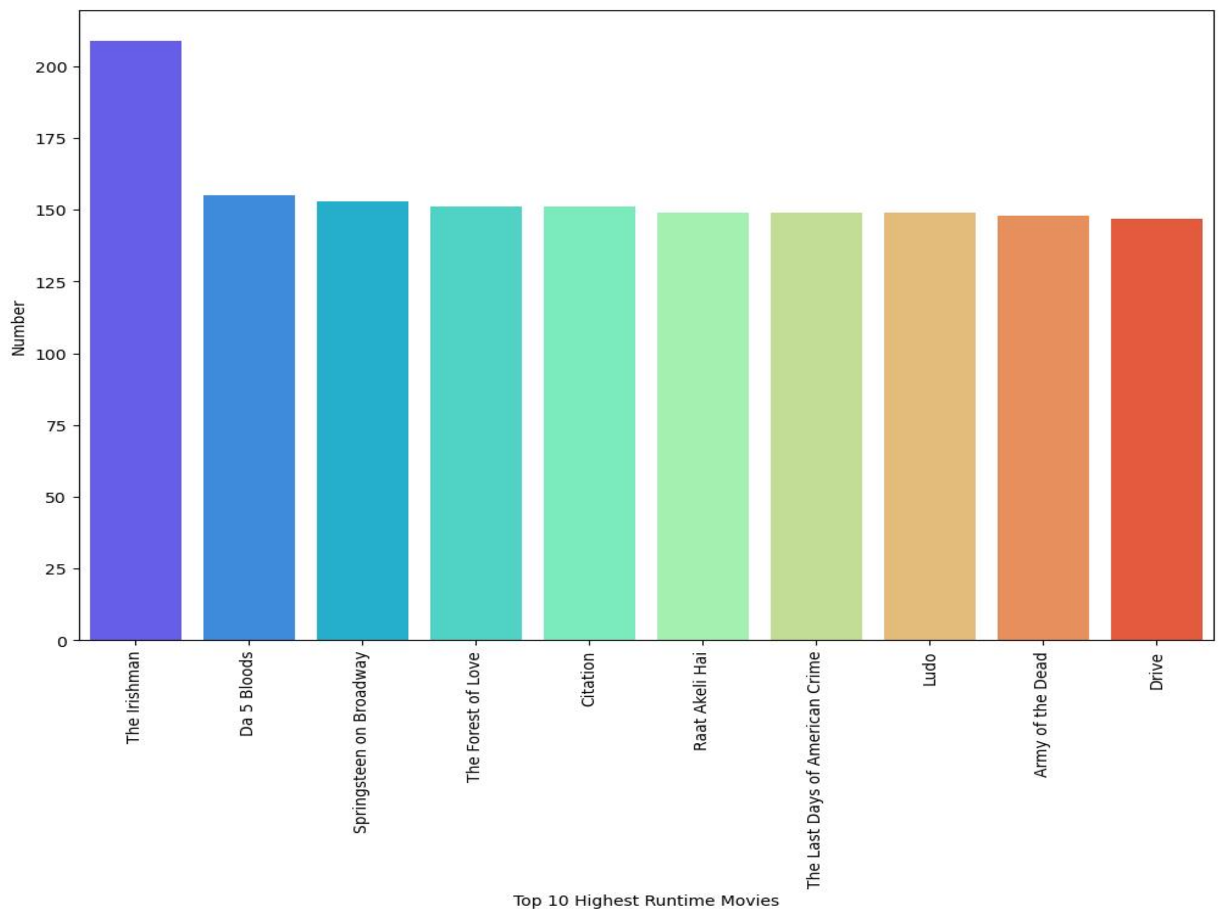
```
fig = px.scatter(df, x='IMDB Score', y='Runtime')
fig.show()
```



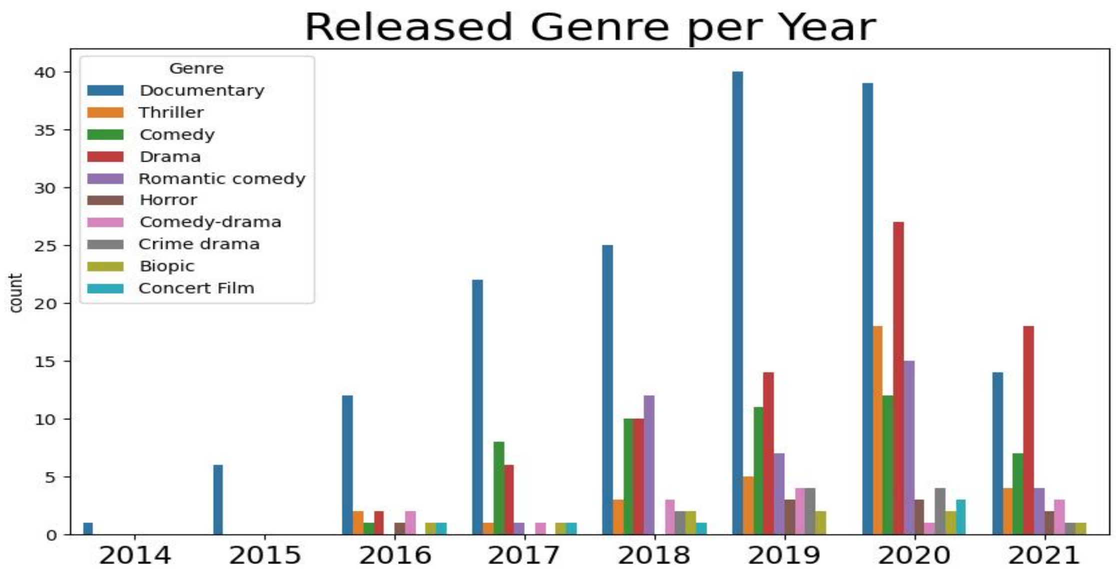
```
df_temp=df.groupby(['Genre']).mean(['IMDB rating']).sort_valu
es(by='IMDB Score', ascending=False).reset_index().iloc[:10,:]
fig, ax = plt.subplots(1,1, figsize = (10, 6), constrained_layout =
True)ax = sns.barplot(x = 'Genre', y = 'IMDB Score', data = df_t
emp, color = 'violet')
for i in ax.patches:
    ax.text(x = i.get_x() + i.get_width()/2, y = i.get_height()/2,
            s = f"{round(i.get_height(),1)}",
            ha = 'center', size = 14, weight = 'bold', rotation = 0, colo
r = 'white',
            bbox=dict(boxstyle="round,pad=0.5", fc='pin

k', ec="pink", lw=2))

ax.set_xlabel('Genre', fontsize=14)
ax.set_ylabel('Average IMDB Score', fontsize=14)
ax.set_xticklabels([i[:15]
for i in df_temp['Genre'].unique()], fontsize=12, rotation = -45 )
plt.title('Top 10 Genre by IMDB Score', fontsize=16);
```



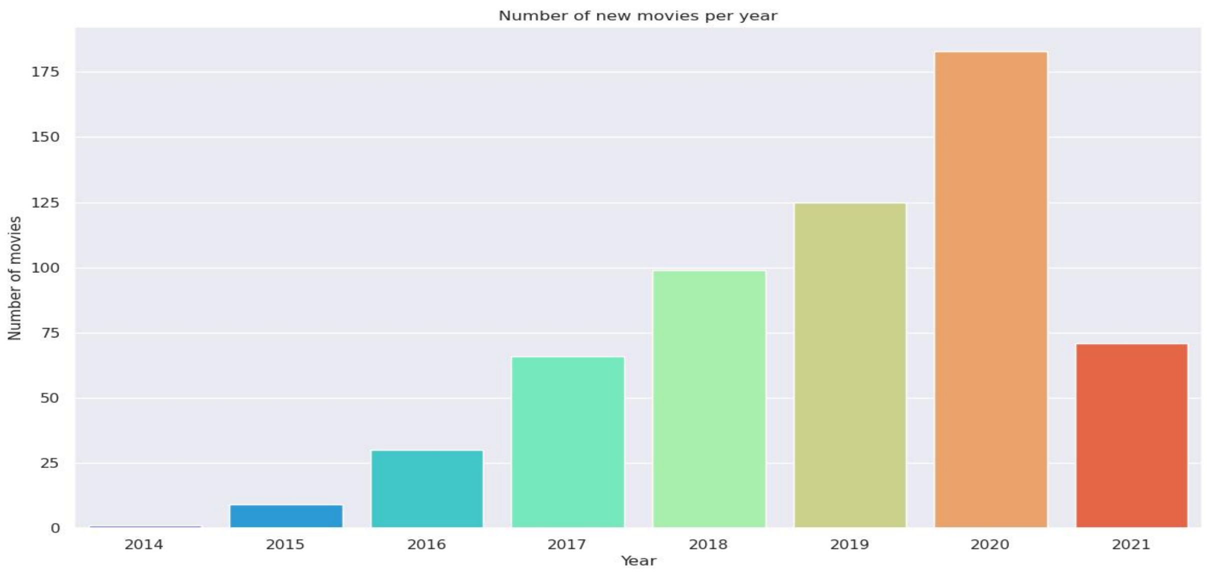
```
df_temp=df.groupby(['Title']).mean(['Runtime rating']).sort_val
ues(by='Runtime', ascending=False).reset_index().iloc[:10,:2]
plt.figure(figsize=(12,8))sns.barplot(x=df_temp["Title"],y=df_te
mp["Runtime"],data=df,palette='rainbow')
plt.xlabel('Top 10 Highest Runtime Movies')
plt.xticks(rotation=90)
plt.ylabel('Number')
plt.show()
```



```
top_genres = df.loc[df['Genre'].isin(df.groupby('Genre').sum().
sort_values(by='IMDB Score', ascending=False).reset_index()
['Genre'][:10])].groupby('Genre').mean().sort_values(by='IMD
B Score', ascending=False).reset_index()['Genre']
plt.figure(figsize= (10, 6))
sns.countplot(x = df.loc[df['Genre'].isin(top_genres)]['year'],
            hue= df.loc[df['Genre'].isin(top_genres)]['Genre'])
plt.title('Released Genre per Year', size= 25)
plt.xlabel(None)
plt.xticks(size= 16)
plt.show()
```

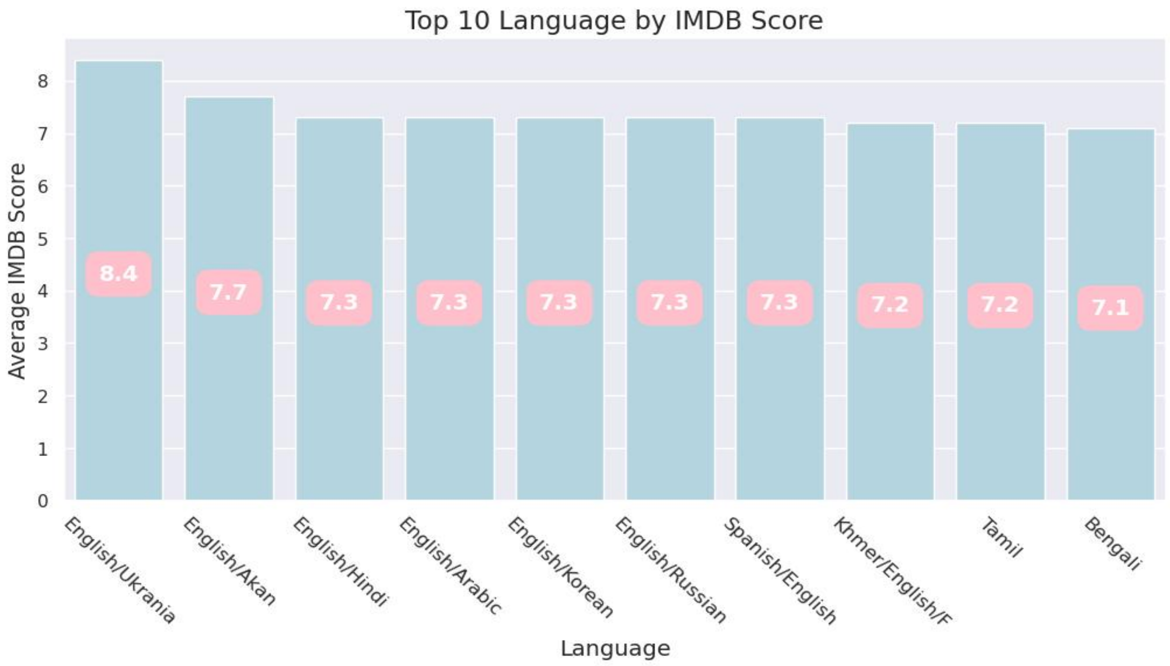


```
sns.set(rc={'figure.figsize':(14, 8)})
ax = sns.countplot(x = df['year'], palette='rainbow')
ax.set_title('Number of new movies per year')
plt.xlabel('Year')plt.ylabel('Number of movies')
plt.show()
```

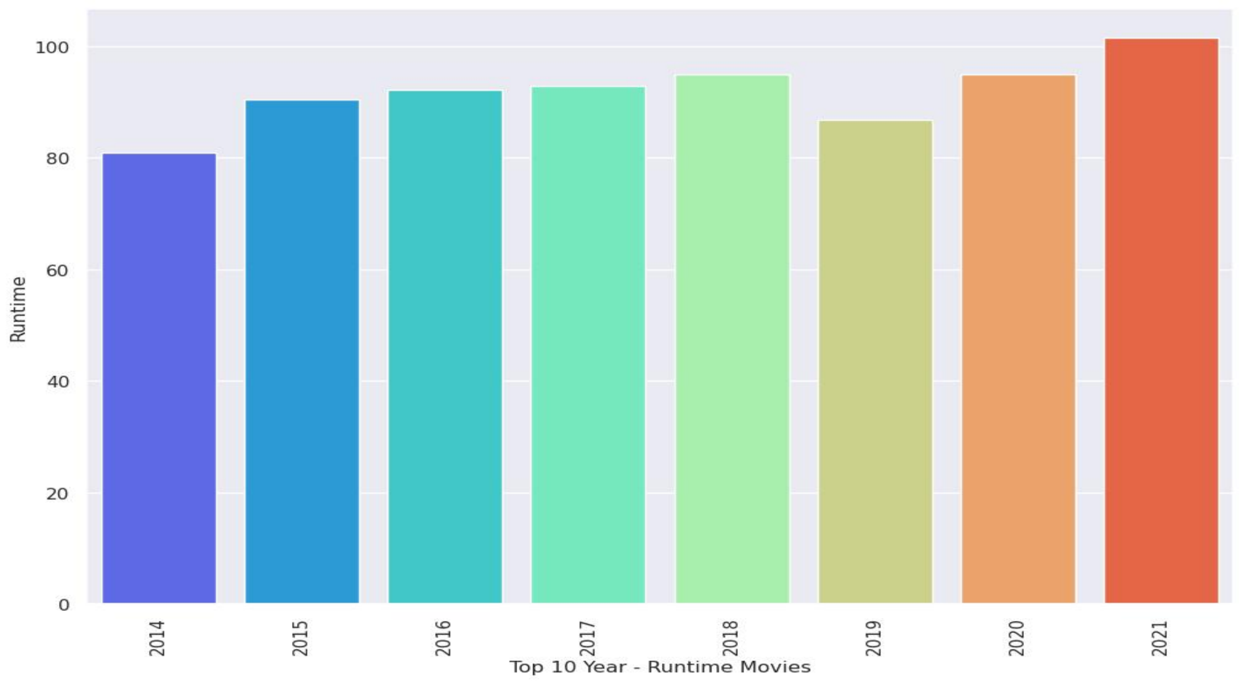


```
df_temp=df.groupby(['Language']).mean(['IMDB rating']).sort_
values(by='IMDB Score', ascending=False).reset_index().iloc[:1
0,:]
fig, ax = plt.subplots(1,1, figsize = (10, 6), constrained_layout =
True
)ax = sns.barplot(x = 'Language', y = 'IMDB Score', data = df_te
mp, color = 'lightblue')
```

```
for i in ax.patches:
    ax.text(x = i.get_x() + i.get_width()/2, y = i.get_height()/2,
            s = f"{round(i.get_height(),1)}",
            ha = 'center', size = 14, weight = 'bold', rotation = 0, colo
r = 'white',
            bbox=dict(boxstyle="round,pad=0.5", fc='pink', ec="pin
k", lw=2))

ax.set_xlabel('Language', fontsize=14)
ax.set_ylabel('Average IMDB Score', fontsize=14)
ax.set_xticklabels([i[:15] for i in df_temp['Language'].unique()],
 fontsize=12, rotation = -45 )
plt.title('Top 10 Language by IMDB Score', fontsize=16);
```



```
df_temp=df.groupby(['year']).mean(['Runtime rating']).sort_valu
es(by='Runtime', ascending=False).reset_index().iloc[:10,:2]
plt.figure(figsize=(12,8))
sns.barplot(x=df_temp["year"],y=df_temp["Runtime"],data=df,p
alette='rainbow')
plt.xlabel('Top 10 Year - Runtime Movies')
plt.xticks(rotation=90)
plt.ylabel('Runtime')
plt.show()
```



```
df_run= df[df["year"] ==2021]
df_run.Runtime.mean()
```

```
101.6056338028169
```

```
genre_lang =[]for i in df.Language.unique():
    df_lang =df[df["Language"]==i]
    df_lang_genre =df_lang.Genre.value_counts().nlargest(1)
    genre_lang.append((i,df_lang_genre))
```

```
df_lang = pd.DataFrame(genre_lang, columns = ['Language', '
Genre'])
```

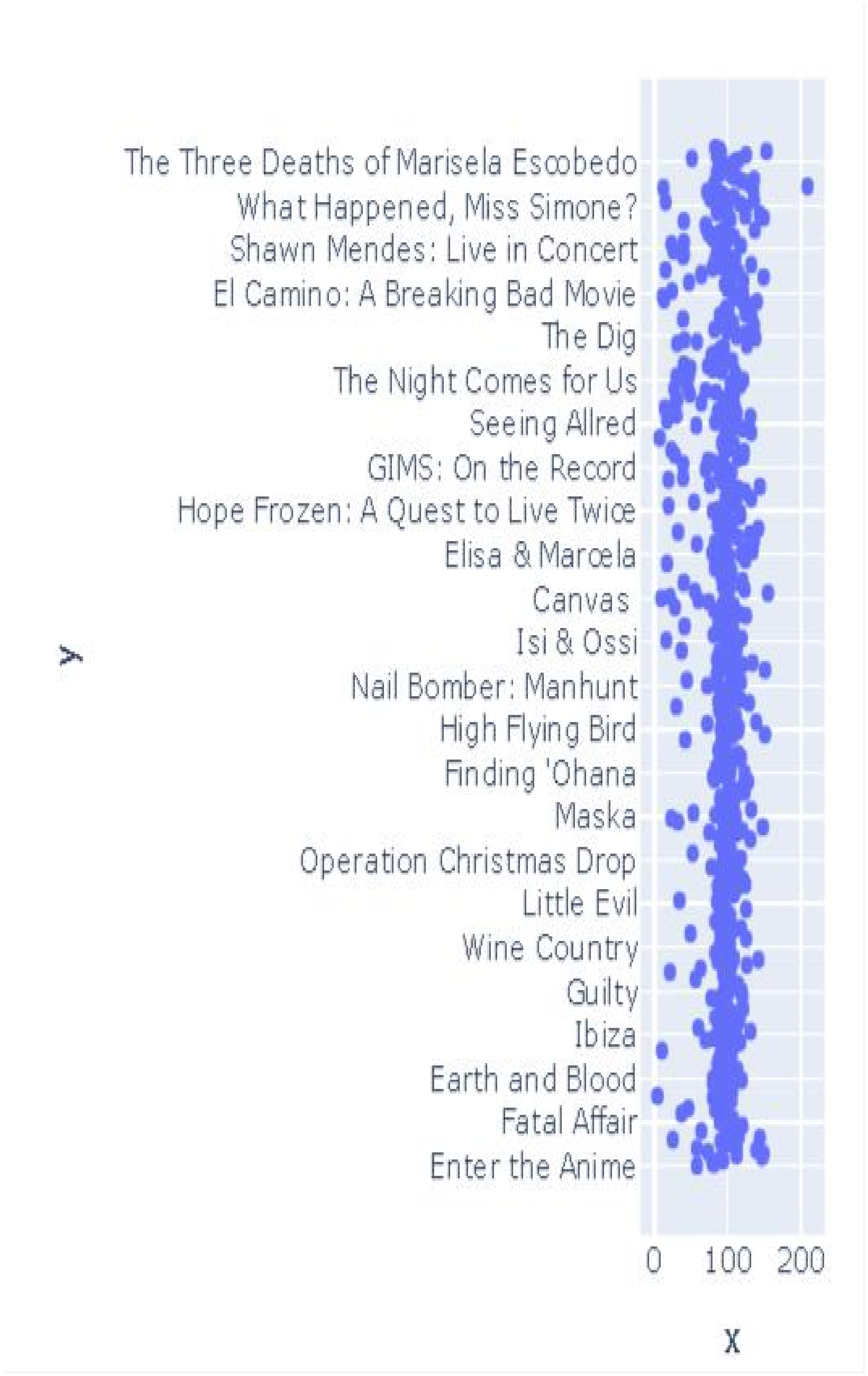
```
df_lang.sort_values(by=['Language'],ignore_index=True)
```

Language	Genre	
0	Bengali	Documentary 1 Name: Genre, dtype: int64
1	Dutch	Romantic comedy 1 Name: Genre, dtype: int64
2	English	Documentary 120 Name: Genre, dtype: int64
3	English/Akan	War drama 1 Name: Genre, dtype: int64
4	English/Arabic	Documentary 1 Name: Genre, dtype: int64
5	English/Hindi	Documentary 2 Name: Genre, dtype: int64
6	English/Japanese	Documentary 1 Name: Genre, dtype: int64
7	English/Korean	Action-adventure 1 Name: Genre, dtype: int64
8	English/Mandarin	Documentary 2 Name: Genre, dtype: int64
9	English/Russian	Documentary 1 Name: Genre, dtype: int64
10	English/Spanish	Documentary 5 Name: Genre, dtype: int64
11	English/Swedish	Documentary 1 Name: Genre, dtype: int64
12	English/Taiwanese/Mandarin	Drama 1 Name: Genre, dtype: int64
13	English/Ukranian/Russian	Documentary 1 Name: Genre,

Language	Genre	
		dtype: int64
14	Filipino	Drama 1 Name: Genre, dtype: int64
15	French	Documentary 6 Name: Genre, dtype: int64
16	Georgian	Documentary 1 Name: Genre, dtype: int64
17	German	Thriller 1 Name: Genre, dtype: int64
18	Hindi	Drama 13 Name: Genre, dtype: int64
19	Indonesian	Drama 3 Name: Genre, dtype: int64
20	Italian	Drama 4 Name: Genre, dtype: int64
21	Japanese	Anime/Science fiction 2 Name: Genre, dtype:...
22	Khmer/English/French	Drama 1 Name: Genre, dtype: int64
23	Korean	Drama 2 Name: Genre, dtype: int64
24	Malay	Action comedy 1 Name: Genre, dtype: int64
25	Marathi	Drama 2 Name: Genre, dtype: int64
26	Norwegian	Horror 1 Name: Genre, dtype: int64

Language	Genre	
27	Polish	Horror 1 Name: Genre, dtype: int64
28	Portuguese	Comedy 6 Name: Genre, dtype: int64
29	Spanish	Documentary 8 Name: Genre, dtype: int64
30	Spanish/Basque	Black comedy 1 Name: Genre, dtype: int64
31	Spanish/Catalan	Documentary 1 Name: Genre, dtype: int64
32	Spanish/English	Documentary 1 Name: Genre, dtype: int64
33	Swedish	Thriller 1 Name: Genre, dtype: int64
34	Tamil	Drama 1 Name: Genre, dtype: int64
35	Thai	Horror 1 Name: Genre, dtype: int64
36	Thia/English	Documentary 1 Name: Genre, dtype: int64
37	Turkish	Comedy 2 Name: Genre, dty

```
fig = px.scatter(x=df['Runtime'], y=df['Title'])

fig.show()
```



The Three Deaths of Marisela Escobedo
What Happened, Miss Simone?
Shawn Mendes: Live in Concert
El Camino: A Breaking Bad Movie
The Dig
The Night Comes for Us
Seeing Allred
GIMS: On the Record
Hope Frozen: A Quest to Live Twice
Elisa & Marcela
Canvas
Isi & Ossi
Nail Bomber: Manhunt
High Flying Bird
Finding 'Ohana
Maska
Operation Christmas Drop
Little Evil
Wine Country
Guilty
Ibiza
Earth and Blood
Fatal Affair
Enter the Anime
y
0
100
200
x

Conclusion :

In conclusion, predicting IMDb scores is a complex task that involves various factors and challenges.IMDb scores are influenced by a multitude of subjective and contextual factors, and no model can perfectly capture all of these nuances.

To improve IMDb score predictions, it's crucial to consider factors such as user reviews, genre, director, actors, and release date, among others. However, it's essential to remember that IMDb scores are ultimately a reflection of audience opinions, and these opinions can change over time.

Therefore, any prediction model should be periodically updated and validated against new data.