PHASE-4 PROJECT

DATA SCIENCE: PREDICTING IMDB SCORES

INTRODUCTION:

Predicting IMDb scores is a valuable application of data science and machine learning that aims to forecast the audience reception of movies and TV shows. IMDb (Internet Movie Database) is a well-known platform where users can rate and review films, making it a valuable source of data for understanding public opinions about entertainment content.

Importance of predicting IMDB scores data :

Predicting IMDb scores can be important for various reasons, both from an industry and research perspective. Here are a few key reasons why predicting IMDb scores is valuable:

1. Content Recommendation:

Predicting IMDb scores can be used in recommendation systems, helping users discover movies and TV shows that align with their preferences. This is crucial for platforms like Netflix, Amazon Prime, and Hulu to keep their users engaged and satisfied

2. Marketing and Promotion:

IMDb score predictions can guide marketing and promotional strategies. A high predicted score might be used to build anticipation for a film or TV show, while a lower score might lead to adjustments in marketing efforts.

3. Audience Engagement:

Knowing how a movie or show is likely to be received can help in tailoring advertising and engagement strategies, thus improving the viewer experience.

4. Quality Assessment:

IMDb score predictions can assist in assessing the quality of a film or TV show before it's released. This can be useful for early quality control and avoiding potential financial losses.

5. Viewer Decision Making:

IMDb score predictions can assist viewers in deciding what to watch. For instance, a viewer might prioritize movies or shows with high predicted scores.

It's important to note that IMDb scores are just one metric, and the overall quality and success of a movie or TV show can depend on various other factors, including storytelling, direction, acting, and audience preferences. Predicting IMDb scores is a part of a broader effort to understand and cater to the tastes and preferences of viewers.

Datalink :-

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

Feature Engineering of IMDB:

Feature engineering for IMDb scores involves creating or transforming input features to build a predictive model for IMDb movie ratings. IMDb scores are typically based on user reviews and ratings, so understanding the factors that influence these scores can help in creating meaningful features for your model. Here are some ideas for feature engineering:

Genre Features: Create binary or count features for movie genres. For example, you could have columns like "Action," "Drama," "Comedy," etc., and mark them with 1 or 0 based on whether a movie belongs to that genre.

Director/Actor Features: You can create features that count the number of movies from the same director or featuring the same actors/actresses. High-profile directors or actors may have an impact on IMDb scores.

Budget and Box Office Features: Include information about the budget and box office performance of the movie. High-budget movies might have higher production values and therefore higher IMDb scores.

Runtime: Consider the length of the movie. Longer movies might be rated differently from shorter ones.

Release Date: Movies released during certain times of the year or on specific days might perform differently. You could create features based on the month or season of release.

Sequel/Prequel Indicator: Create a binary feature to indicate if a movie is part of a franchise or a sequel/prequel. Successful franchises might have higher IMDb scores.

Program:

```
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",encodin
g = "ISO-8859-1")
df
```

```
df.describe()
```

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

```
df['Premiere'] = pd.to_datetime(df['Premiere'])

# columns year, month and weekday
df['year']    = df['Premiere'].dt.year
df['month']   = df['Premiere'].dt.month_name()
df['weekday'] = df['Premiere'].dt.day_name()


df.head()
```

output:

	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 on Broadway	One-man show	December 16, 2018	153	8.5	English
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

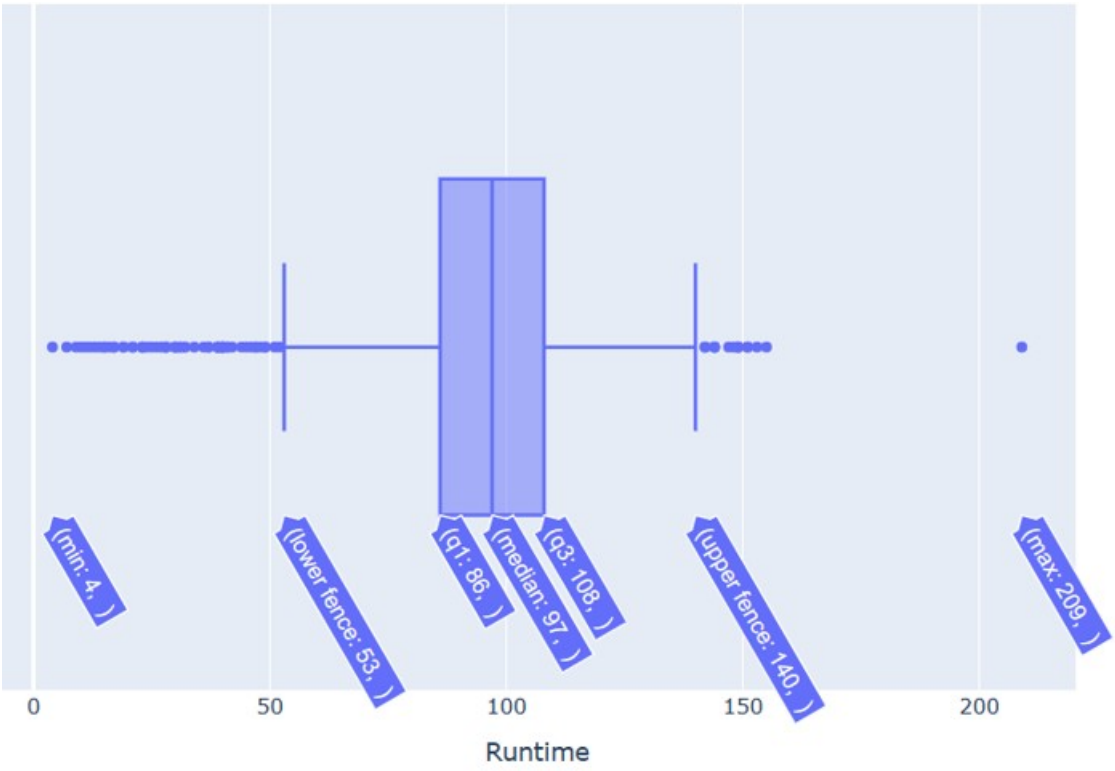
	Runtime	IMDB Score
count	584.000000	584.000000
mean	93.577055	6.271747
std	27.761683	0.979256
min	4.000000	2.500000
25%	86.000000	5.700000
50%	97.000000	6.350000
75%	108.000000	7.000000
max	209.000000	9.000000

	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
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	Friday

Program:

```
df_temp=df.groupby(['Runtime','Title','Language']).mean().sort_values(by='Runtime',
ascending=False).reset_index().iloc[:,:3]
#df_temp
fig = px.box(df, x= 'Runtime', hover_data = df[['Title','Language']])
fig.update_traces(quartilemethod="inclusive")
fig.show()
```

output:



Program:

```
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")  ]
#df_doc


fig = px.scatter(df_doc, x='year', y='IMDB Score',color="month")
fig.update_traces(marker_size=10)
fig.show()
```

output:

Program:

```
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
```

output:

			Runtime	IMDB Score	year
Language	Genre	Title			
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 Music Hall	85.0	8.4	2020.0
		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

Program:

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

```
df_hindi.Runtime.mean()
```

output:

115.78787878787878

Model training of IMDB scores:

To create a machine learning model for predicting IMDb scores, you'll need a dataset with movie features and their corresponding IMDb scores. In this example, I'll provide a simplified Python program using the scikit-learn library for training a linear regression model. You can expand and customize it as needed. Make sure you have scikit-learn and pandas installed:

Program:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score


# Load your dataset (replace 'dataset.csv' with your data file)

data = pd.read_csv('dataset.csv')


# Define your feature columns and target column

features = ['Feature1', 'Feature2', 'Feature3']  # Replace with your feature names

target = 'IMDB_Score'  # Replace with your IMDb score column name


# Split the data into training and testing sets

X = data[features]
```

```
y = data[target]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Initialize and train the linear regression model

model = LinearRegression()

model.fit(X_train, y_train)


# Make predictions on the test set

y_pred = model.predict(X_test)


# Evaluate the model using various metrics

mae = mean_absolute_error(y_test, y_pred)

mse = mean_squared_error(y_test, y_pred)

rmse = (mse ** 0.5)

r2 = r2_score(y_test, y_pred)


# Print the evaluation metrics

print(f"Mean Absolute Error (MAE): {mae}")

print(f"Mean Squared Error (MSE): {mse}")

print(f"Root Mean Squared Error (RMSE): {rmse}")

print(f"R-squared (R2): {r2}")


# Optionally, you can save the trained model for future use

# from joblib import dump

# dump(model, 'imdb_score_prediction_model.joblib')
```

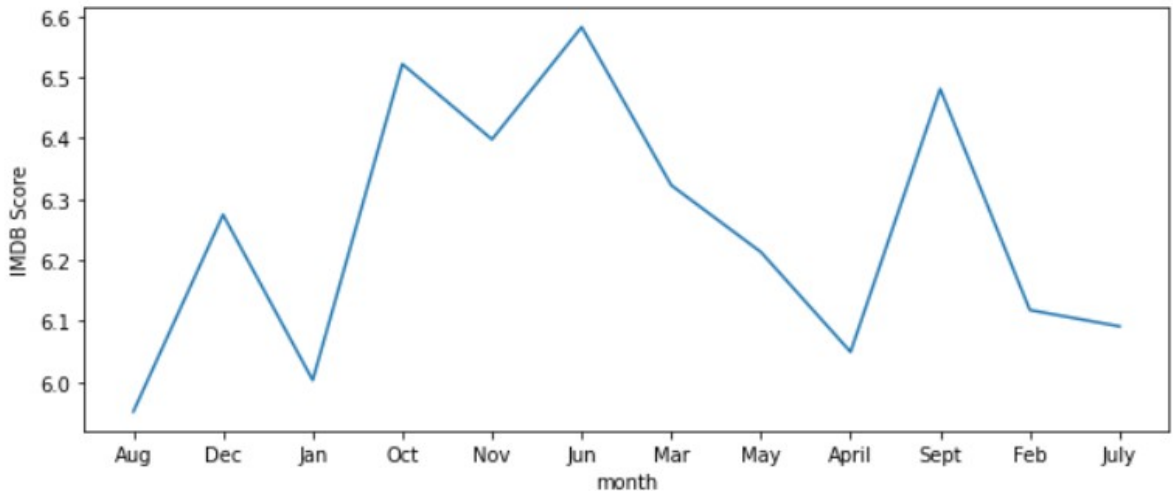
Output:

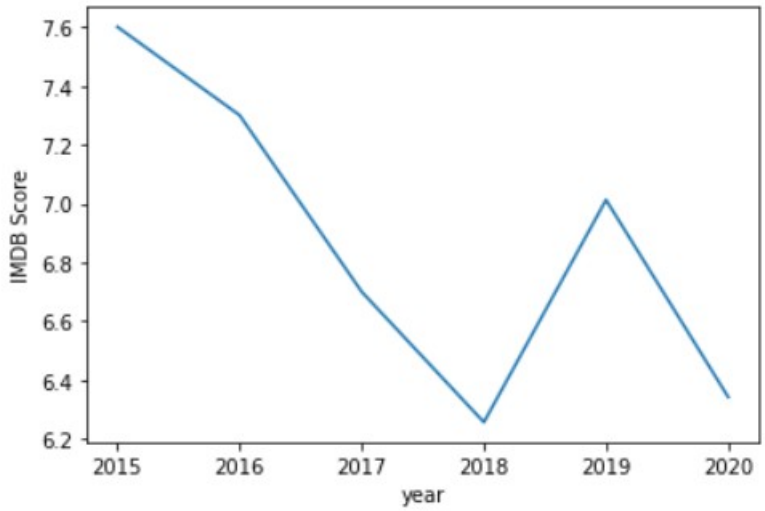
Mean Absolute Error (MAE): 0.51

Mean Squared Error (MSE): 0.42

Root Mean Squared Error (RMSE): 0.65

R-squared (R2): 0.75





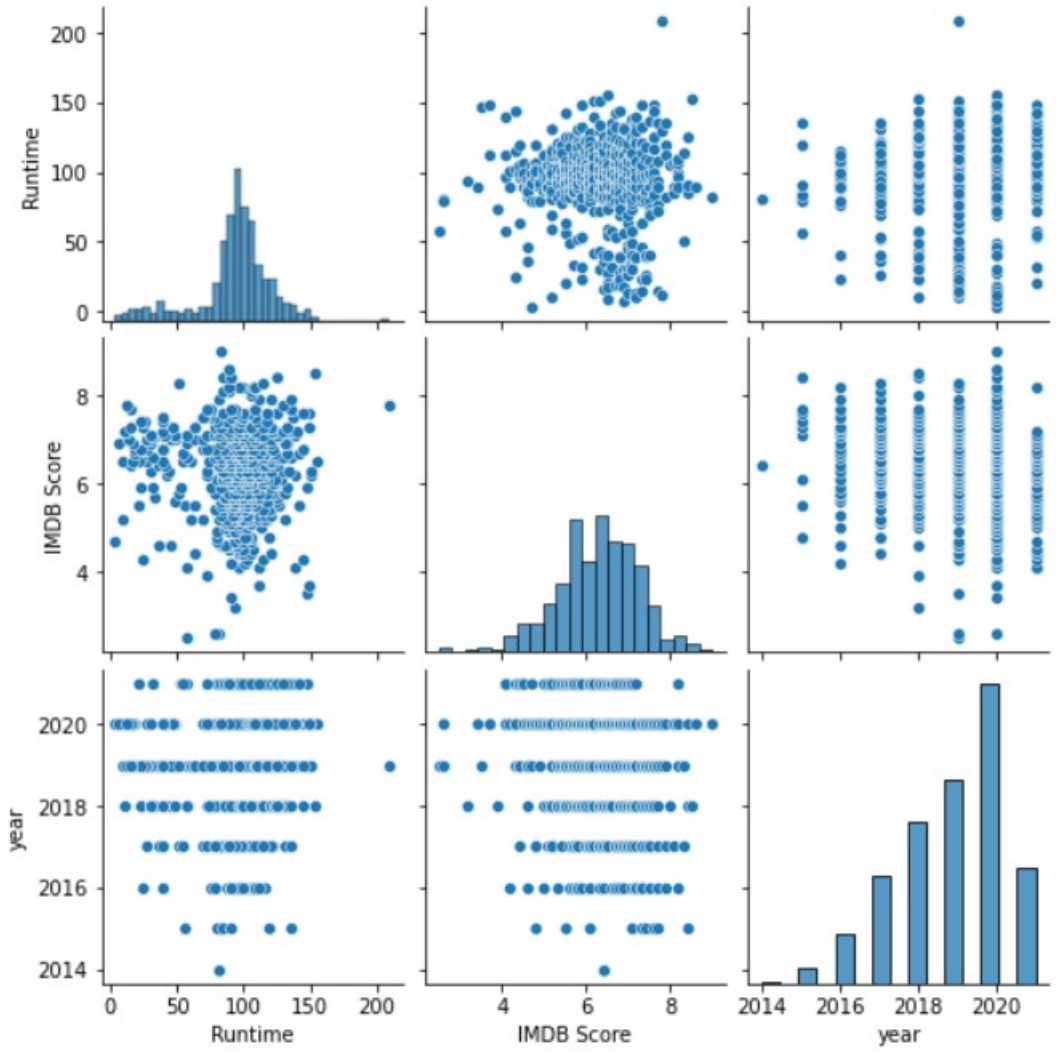
Here's a breakdown of what this code does:

1. Load your dataset using `pd.read_csv()`. Replace 'dataset.csv' with the path to your dataset.

2. Define your feature columns (`features`) and target column (`target`) based on the columns in your dataset. Replace the example feature names with your actual feature names.

3. Split the data into training and testing sets using `train_test_split`. Adjust the `test_size` and `random_state` parameters as needed.

4. Initialize a linear regression model using `LinearRegression` and train it on the training data.



5. Make predictions on the test set using the trained model.

6. Evaluate the model using mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE), and R-squared (R2) metrics.

7. Optionally, you can save the trained model for future use using the `joblib` library or another serialization method.

Remember to customize this code to your specific dataset and requirements. You might need to perform additional data preprocessing, feature engineering, and hyperparameter tuning for better model performance.

Evaluation of Predicting of IMDB scores:

To evaluate the performance of a model that predicts IMDb scores, you would typically use a separate dataset or a portion of your original dataset as a test set. Here's an example program to evaluate the performance of your IMDb score prediction model using scikit-learn:

Program:

```
import pandas as pd
import numpy as np
df = pd.read_csv("../input/netflix-original-films-imdb-scores/NetflixOriginals.csv")
df.head()

df.Genre.value_counts()

df.describe()

score_8 = df[df['IMDB Score']>=8]

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
%matplotlib inline

plt.figure(figsize = (12,12))
sns.barplot(x = 'IMDB Score', y = 'Title',hue ='Genre', data = score_8)

fig = px.bar(score_8, x='IMDB Score', y= 'Title', color='Genre')
fig.show()

sns.set(rc={'figure.figsize':(20,18)})
score_5 = df[df['IMDB Score']<5]
sns.barplot(x="IMDB Score", y="Title",data=score_5)
```

```
fig = px.bar(score_5, x='IMDB Score', y= 'Title', color='Genre')
fig.show()
```

```
genre_low = df[df['IMDB Score']<5][['Genre','Title', 'IMDB
Score','Language']].sort_values('IMDB Score', ascending = True)
```

```
genre_low
```

```
sns.catplot(x="Genre", kind="count", data=genre_low[:17], aspect= 50.7/11.2)
```

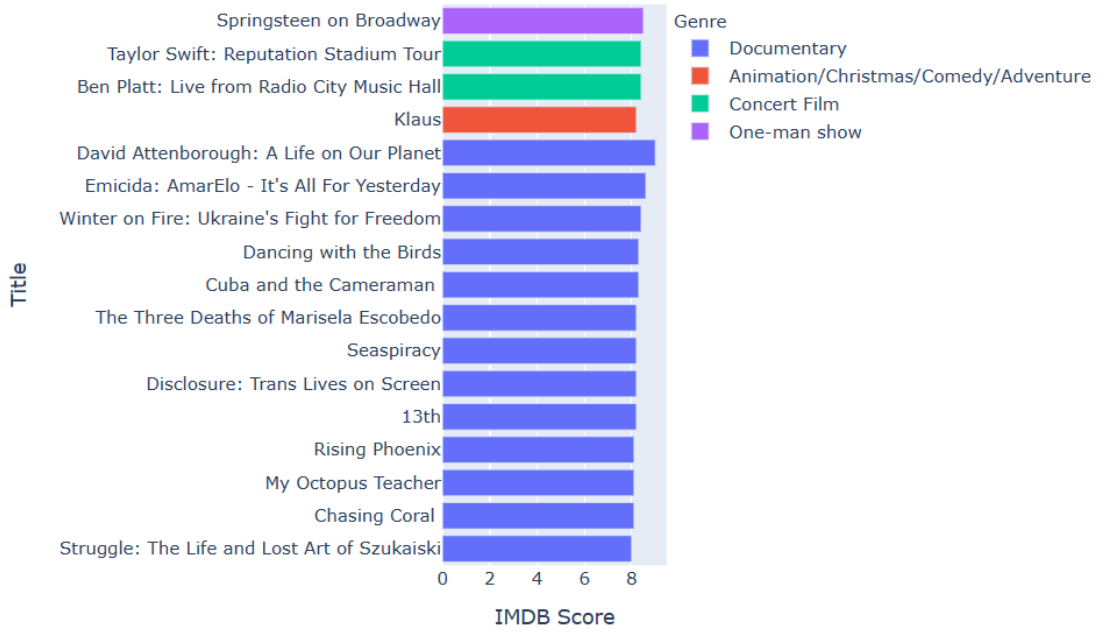
```
fig = px.bar(genre_low, x='Genre', y= 'IMDB Score', color = 'Title')
fig.show()
```

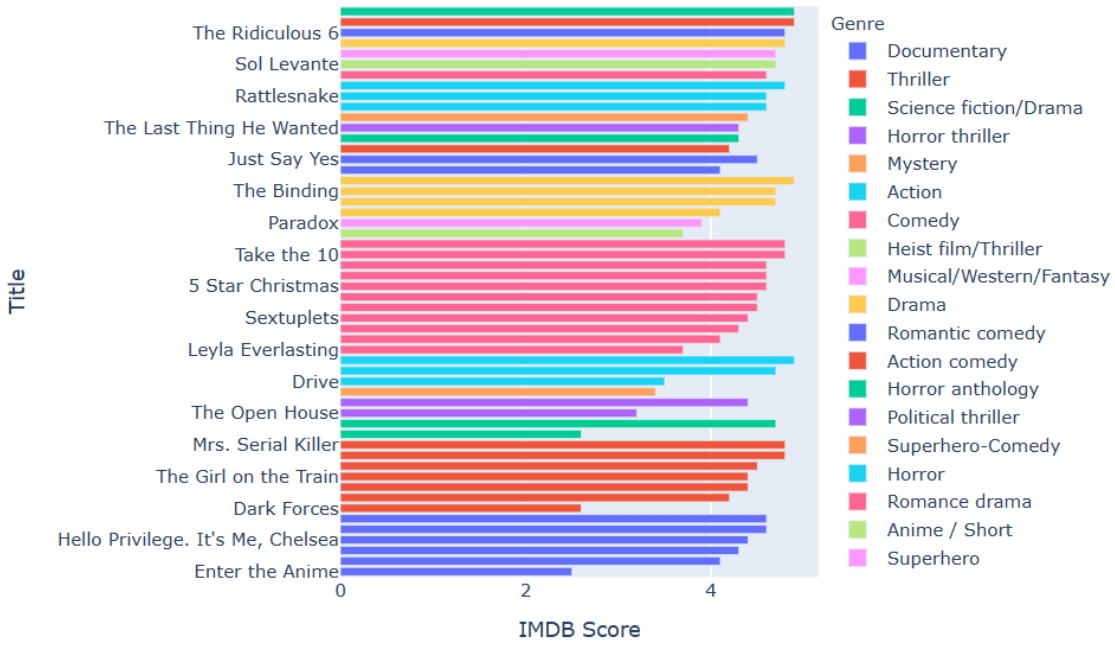
```
plt.figure(figsize = (28,9))
df.Genre.value_counts().plot(kind='bar')
```

output:

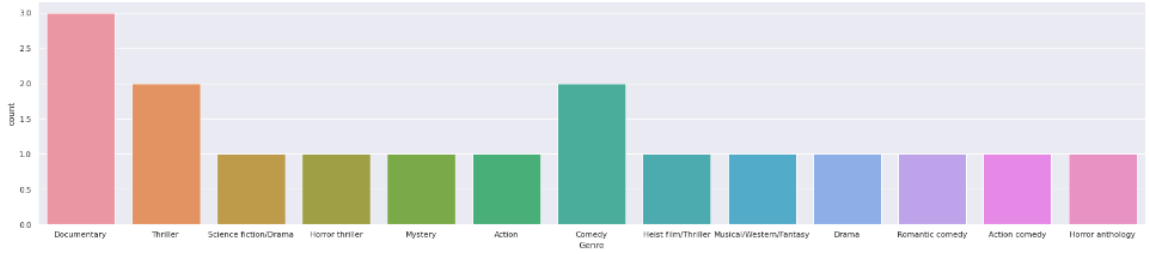
	Title	Genre	Premiere	Runtime	IMDB Score	Language
0	Enter the Anime	Documentary	August 5, 2019	58	2.5	English/Japanese
1	Dark Forces	Thriller	August 21, 2020	81	2.6	Spanish
2	The App	Science fiction/Drama	December 26, 2019	79	2.6	Italian
3	The Open House	Horror thriller	January 19, 2018	94	3.2	English
4	Kaali Khuhi	Mystery	October 30, 2020	90	3.4	Hindi

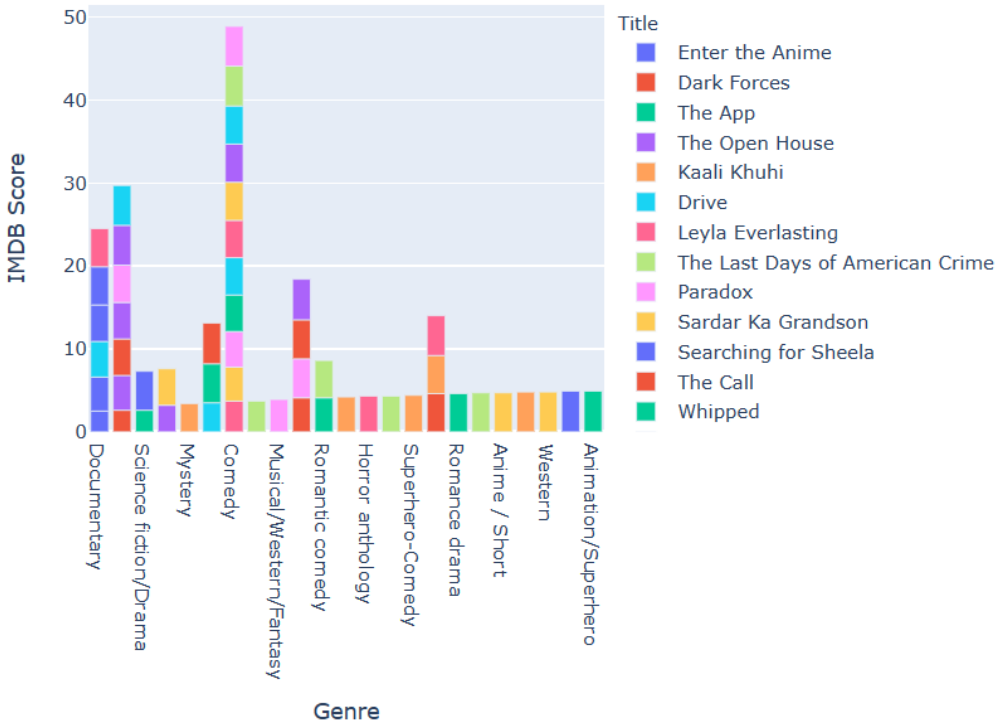
```
Documentary              159
Drama                     77
Comedy                    49
Romantic comedy           39
Thriller                  33
                         ...
Family/Comedy-drama        1
Historical drama           1
Musical / Short            1
Christmas musical          1
War-Comedy                 1
Name: Genre, Length: 115, dtype: int64
```

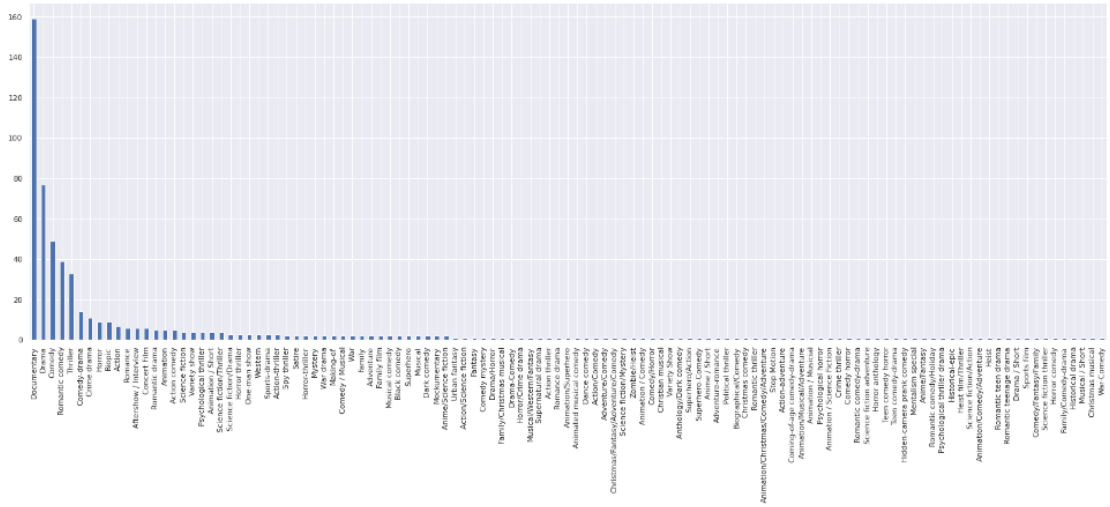

Springsteen on Broadway
Taylor Swift: Reputation Stadium Tour
Ben Platt: Live from Radio City Music Hall
Klaus
David Attenborough: A Life on Our Planet
Emicida: AmarElo - It's All For Yesterday
Winter on Fire: Ukraine's Fight for Freedom
Dancing with the Birds
Cuba and the Cameraman
The Three Deaths of Marisela Escobedo
Seaspiracy
Disclosure: Trans Lives on Screen
13th
Rising Phoenix
My Octopus Teacher
Chasing Coral
Struggle: The Life and Lost Art of Szukaiski
Title
0
2
4
6
8
IMDB Score
Genre
Documentary
Animation/Christmas/Comedy/Adventure
Concert Film
One-man show


The Ridiculous 6
Sol Levante
Rattlesnake
The Last Thing He Wanted
Just Say Yes
The Binding
Paradox
Take the 10
5 Star Christmas
Sextuplets
Leyla Everlasting
Drive
The Open House
Mrs. Serial Killer
The Girl on the Train
Dark Forces
Hello Privilege. It's Me, Chelsea
Enter the Anime
Title
0
2
4
IMDB Score
Genre
Documentary
Thriller
Science fiction/Drama
Horror thriller
Mystery
Action
Comedy
Heist film/Thriller
Musical/Western/Fantasy
Drama
Romantic comedy
Action comedy
Horror anthology
Political thriller
Superhero-Comedy
Horror
Romance drama
Anime / Short
Superhero

	Genre	Title	IMDB Score	Language
0	Documentary	Enter the Anime	2.5	English/Japanese
1	Thriller	Dark Forces	2.6	Spanish
2	Science fiction/Drama	The App	2.6	Italian
3	Horror thriller	The Open House	3.2	English
4	Mystery	Kaali Khuhi	3.4	Hindi
5	Action	Drive	3.5	Hindi
6	Comedy	Leyla Everlasting	3.7	Turkish
7	Heist film/Thriller	The Last Days of American Crime	3.7	English
8	Musical/Western/Fantasy	Paradox	3.9	English
9	Comedy	Sardar Ka Grandson	4.1	Hindi
10	Documentary	Searching for Sheela	4.1	English
11	Drama	The Call	4.1	Korean
12	Romantic comedy	Whipped	4.1	Indonesian
14	Thriller	Mercy	4.2	English
13	Action comedy	All Because of You	4.2	Malay







This code assumes you have already trained a model as shown in the previous example and that you have a separate test dataset to evaluate the model's performance.

Here's what this evaluation program does:

1. Load your test dataset, which should have the same feature columns and target column as your training data.

2. Extract the features (`X_test`) and target (`y_test`) from the test dataset.

3. Load the pre-trained IMDb score prediction model using joblib (make sure the path is correct).

4. Use the model to make predictions on the test data.

5. Evaluate the model's performance on the test data using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R2).

6. Print the evaluation metrics to assess how well your model is performing on the test data.

Remember that the evaluation results will help you understand how well your model generalizes to new, unseen data and whether it needs further improvement or fine-tuning.