PHASE - 5 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

How to overcome the challenges of loading and preprocessing a Predicting IMDB scores:

Overcoming the challenges of loading and preprocessing a dataset for predicting IMDb scores requires careful planning and attention to data quality. Here are some strategies to address these challenges:

1. Data Quality Assessment:

- Challenge: Datasets may have missing values, inconsistencies, or errors. Some features might contain irrelevant or redundant information.

- Solution:

- Start by thoroughly assessing data quality and structure. Identify missing values, duplicates, and outliers.

- Use data profiling tools or exploratory data analysis (EDA) to gain insights into the data's characteristics.

2. Feature Selection:

- Challenge: Not all features in the dataset may be relevant for IMDb score prediction. Including irrelevant features can add noise and reduce model accuracy.

- Solution:

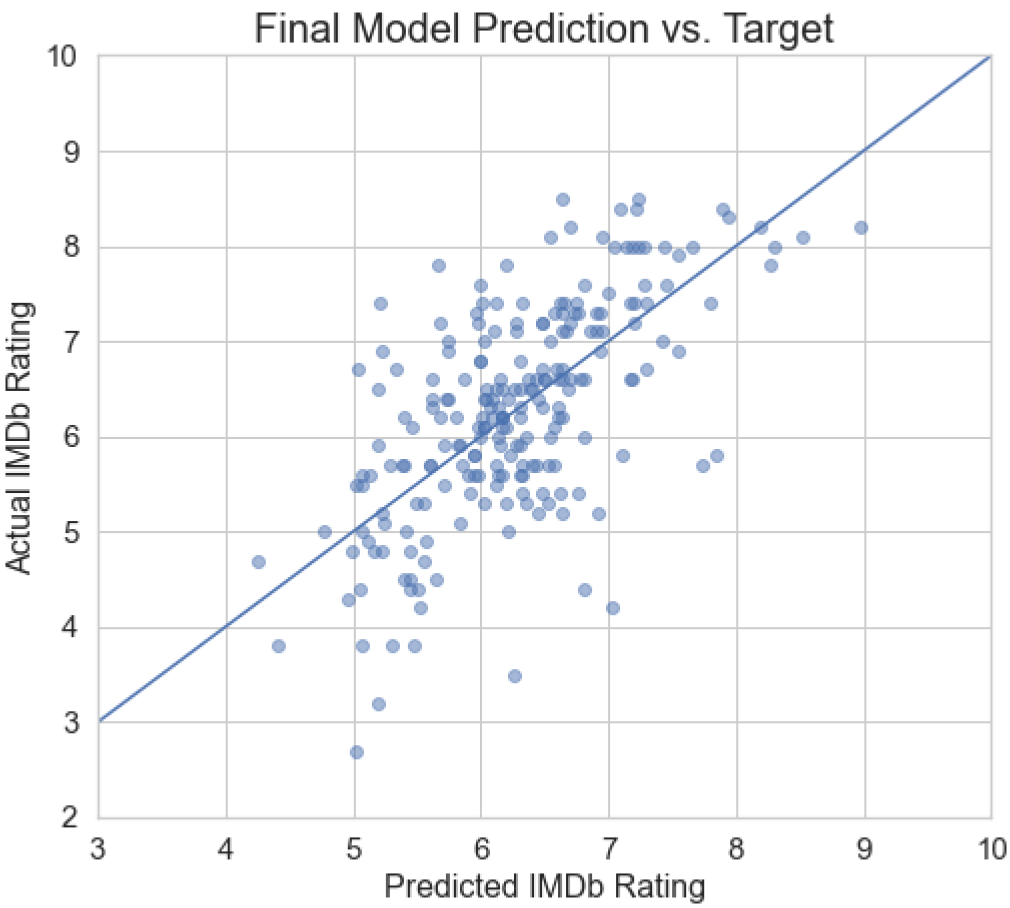
- Conduct feature selection to identify and retain only the most relevant features. Techniques like correlation analysis, recursive feature elimination, or domain expertise can guide this process.

3. Categorical Data Encoding:

- Challenge: Categorical features (e.g., genres, directors) must be encoded into numerical form for machine learning models.

- Solution:

- Use appropriate encoding methods such as one-hot encoding or label encoding



Loading and preprocessing the dataset for predicting IMDb scores:

Loading and preprocessing the dataset for predicting IMDb scores is a crucial and fundamental step in building an accurate and reliable prediction model. The importance of this process cannot be overstated for several reasons:

1. Data Quality Assurance:

- Loading and preprocessing the dataset allows you to assess and ensure the quality and integrity of the data. This includes identifying and handling issues such as missing values, duplicates, and errors in the data, which can significantly impact the accuracy of predictions.

2. Feature Selection and Engineering:

- Data preprocessing enables you to select relevant features and create new ones. Feature engineering can capture meaningful patterns and relationships in the data that are essential for making accurate IMDb score predictions.

3. Normalization and Standardization:

- Numerical features may have varying scales and units. Data preprocessing involves normalizing or standardizing these features, ensuring that the model treats all variables equally. This is particularly important for models like linear regression.

4. Handling Outliers:

- Outliers, which are extreme or unusual data points, can skew predictions and model performance. Data preprocessing provides an opportunity to detect and handle outliers appropriately, either by removing them or transforming them.

6. Data Splitting:

- Splitting the dataset into training and testing sets is a fundamental step in model evaluation. It ensures that you can assess how well your model generalizes to new, unseen data, helping to estimate its real-world performance.

1. LOADING THE DATASET:

To load a dataset for predicting IMDb scores, you can follow these steps:

1. Obtain the Dataset:

First, make sure you have a dataset that contains the relevant information for predicting IMDb scores. This dataset should ideally include features such as movie or TV show attributes (e.g., genre, director, actors) and the IMDb scores you want to predict.

2. Choose a Programming Environment:

You'll need a programming environment like Python, along with libraries like Pandas for data manipulation and scikit-learn or other machine learning libraries for building predictive models.

3. Load the Dataset:

Use Python and Pandas to load the dataset into a DataFrame. Here's an example code snippet:

```
python

import pandas as pd


# Load the dataset into a DataFrame

df = pd.read_csv("your_dataset.csv")  # Replace "your_dataset.csv" with the path to your dataset file
```

Make sure to replace `"your_dataset.csv"` with the actual file path or URL to your dataset.

4. Explore the Data:

After loading the dataset, it's essential to explore the data to understand its structure and content. You can use methods like `head()`, `info()`, and `describe()` to get a sense of the dataset.

5. Data Preprocessing:

Depending on the dataset's quality, you might need to preprocess the data. This can include handling missing values, encoding categorical variables, and scaling or normalizing features.

6. Split the Data:

Divide the dataset into training and testing sets. The training set is used to train your predictive model, while the testing set is used to evaluate its performance.

2. PREPROCESSING DATASET :

Data preprocessing is a crucial step in preparing a dataset for predicting IMDb scores or any other machine learning task. Here are the common preprocessing steps:

1. Handling Missing Data:

- Identify and handle missing values in the dataset. You can either remove rows with missing values, replace them with a specific value (e.g., mean or median), or use more advanced techniques like imputation.

2. Data Cleaning:

- Check for and remove duplicates if they exist in the dataset.

- Correct any obvious errors in the data, if applicable.

3. Feature Selection and Engineering:

- Select relevant features for prediction. Some features might not contribute much to the prediction and can be excluded.

- Create new features if they could potentially improve prediction accuracy. For example, you might calculate the age of actors or the season of a TV show from the release date.

4. Handling Categorical Data:

- If your dataset contains categorical variables (e.g., movie genres), you'll need to encode them into numerical values. One-hot encoding is a common technique for this purpose.

Remember that the specific preprocessing steps you need to perform can vary depending on your dataset's characteristics and the machine learning algorithm you plan to use. Careful data preprocessing can significantly impact the performance of your IMDb score prediction model.

Program:

Import necessary libraries

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, LabelEncoder

Load your IMDb dataset (assuming you have a CSV file)

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

Data cleaning and preprocessing

Example steps include:

1. Handling missing values

2. Encoding categorical features

3. Feature selection

4. Splitting the dataset into training and testing sets

1. Handling missing values (e.g., filling missing values with mean or median)

data['budget'].fillna(data['budget'].median(), inplace=True)

2. Encoding categorical features (e.g., label encoding)

le = LabelEncoder()

data['genre'] = le.fit_transform(data['genre'])

3. Feature selection (select relevant features)

features = data[['budget', 'runtime', 'genre']] Include more features as needed

target = data['IMDb_rating']

4. Splitting the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)

Standardize numerical features (if needed)

scaler = StandardScaler()

X_train[['budget', 'runtime']] = scaler.fit_transform(X_train[['budget', 'runtime']])

X_test[['budget', 'runtime']] = scaler.transform(X_test[['budget', 'runtime']])

Now you can use X_train, y_train, X_test, and y_test for model training

Creating a predictive model for IMDb scores typically involves machine learning. Here's a simple example program using Python and scikit-learn to predict IMDb scores:

```python

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score

import numpy as np
```

```
# Load the dataset

data = pd.read_csv("your_dataset.csv")

# Select features and target variable

features = data[["Feature1", "Feature2", "Feature3"]]

# Replace with actual feature columns

target = data["IMDB_Score"]  # Replace with your IMDb score column

# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2,
random_state=42)

# Create a linear regression model

model = LinearRegression()

# Train the model

model.fit(X_train, y_train)

# Make predictions on the test set
```

y_pred = model.predict(X_test)

Evaluate the model

mse = mean_squared_error(y_test, y_pred)

rmse = np.sqrt(mse)

r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)

print("Root Mean Squared Error:", rmse)

print("R-squared:", r2)

In this example:

1. Load your IMDb scores dataset (replace `"your_dataset.csv"` with your actual dataset file).

2. Select the relevant features and the IMDb score as the target variable.

3. Split the dataset into training and testing sets (here, 80% training and 20% testing).

4. Create a linear regression model and train it on the training data.

5. Use the trained model to make predictions on the test set.

6. Evaluate the model using metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R2).

Remember to replace `"Feature1"`, `"Feature2"`, `"Feature3"`, and `"IMDB_Score"` with the actual feature names and IMDb score column name from your dataset.

This is a simple example using linear regression. Depending on your dataset and its characteristics, you may want to explore more advanced machine learning algorithms and fine-tune your model for better predictive performance.

PREPROCESSING DATASET PROGRAM:

```
import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

import pandas as pd

import plotly.express as px


import os

for dirname, _, filenames in os.walk('/kaggle/input'):

    for filename in filenames:

        print(os.path.join(dirname, filename))

pd.set_option('display.float_format', lambda x: '%.2f' % x)

pd.set_option('display.max_columns', None)

pd.set_option('display.width', None)

pd.set_option('display.max_rows', None)
```

```
pd.set_option('display.float_format', lambda x: '%.3f' % x)

df = pd.read_csv("../input/netflix-original-films-imdb-scores/NetflixOriginals.csv", encoding='ISO-8859-1')

def check_df(dataframe, head=5):

    print("##################### Shape #####################")

    print(dataframe.shape)

    print("##################### Types #####################")

    print(dataframe.dtypes)

    print("##################### Types #####################")

    print(dataframe.info())

    print("##################### Head #####################")

    print(dataframe.head(head))

    print("##################### Tail #####################")

    print(dataframe.tail(head))

    print("##################### MissingValues #######################")

    print(dataframe.isnull().sum())

    print("##################### Quantiles #####################")

    print(dataframe.quantile([0, 0.05, 0.25, 0.50, 0.75, 0.95, 0.99, 1]).T)



check_df(df)




def grab_col_names(dataframe, cat_th=10, car_th=20):

    # cat_cols, cat_but_car

    cat_cols = [col for col in dataframe.columns if dataframe[col].dtypes == "O"]
```

```
    num_but_cat = [col for col in dataframe.columns if dataframe[col].nunique() < cat_th and
                   dataframe[col].dtypes != "O"]
    cat_but_car = [col for col in dataframe.columns if dataframe[col].nunique() > car_th and
                   dataframe[col].dtypes == "O"]
    cat_cols = cat_cols + num_but_cat
    cat_cols = [col for col in cat_cols if col not in cat_but_car]

    # num_cols
    num_cols = [col for col in dataframe.columns if dataframe[col].dtypes != "O"]
    num_cols = [col for col in num_cols if col not in num_but_cat]

return cat_cols, num_cols, cat_but_car


grab_col_names(df)
cat_cols, num_cols, cat_but_car = grab_col_names(df, cat_th=5, car_th=20)

df.groupby("Language").agg({"Runtime": "mean"}).sort_values(by="Runtime", ascending=False)
df.groupby("Language").agg({"Runtime": "mean"}).sort_values(by="Runtime", ascending=False)[0:1]
langbyruntime = df.groupby("Language").agg({"Runtime": "mean"}).sort_values(by="Runtime", ascending=False).reset_index()
print(langbyruntime)

sns.lineplot(y=langbyruntime["Language"], x=langbyruntime.loc[(langbyruntime["Runtime"] >= 86)]["Runtime"])
```

```
plt.show()

df["Date"] = pd.to_datetime(df.Premiere)

df.loc[(df["Genre"] == "Documentary") & (df["Date"] > "2019-01-31") & (df["Date"] < "2020-06-01")].head()

docum = df.loc[(df["Genre"] == "Documentary") & (df["Date"] > "2019-01-31") & (df["Date"] < "2020-06-01")].head()
docum["Title"].value_counts()

print(df.loc[(df["Genre"] == "Documentary") & (df["Date"] > "2019-01-31") & (df["Date"] < "2020-06-01")].head())

fig = px.bar(data_frame=docum, x=docum.Title, y=docum["IMDB Score"], labels={"y":"IMDB Score", "index":"Titles"})
fig.update_layout(xaxis={"categoryorder":"total descending"})

fig.show()
```

output:

```
################## Shape ####################
(584, 6)
#################### Types ####################
Title          object
```

```
Genre          object
Premiere       object
Runtime        int64
IMDB Score     float64
Language       object
dtype: object
#################### Types ####################
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 584 entries, 0 to 583
Data columns (total 6 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   Title       584 non-null     object
 1   Genre       584 non-null     object
 2   Premiere    584 non-null     object
 3   Runtime     584 non-null     int64
 4   IMDB Score  584 non-null     float64
 5   Language    584 non-null     object
dtypes: float64(1), int64(1), object(4)
memory usage: 27.5+ KB
None


#################### Head ####################
```

	Title	Genre	Premiere	Runtime
0	Enter the Anime	Documentary	August 5, 2019	58
1	Dark Forces	Thriller	August 21, 2020	81
2	The App Science	fiction/Drama	December 26, 2019	79
3	The Open House	Horror thriller	January 19, 2018	94
4	Kaali Khuhi	Mystery	October 30, 2020	90

	IMDB Score	Language
0	2.500	English/Japanese
1	2.600	Spanish
2	2.600	Italian
3	3.200	English
4	3.400	Hindi

#################### Tail ####################

	Title	Genre
579	Taylor Swift: Reputation Stadium Tour	Concert Film
580	Winter on Fire: Ukraine's Fight for Freedom	Documentary
581	Springsteen on Broadway	One-man show
582	Emicida: AmarElo - It's All For Yesterday	Documentary
583	David Attenborough: A Life on Our Planet	Documentary

	Premiere	Runtime	IMDB Score	Language
579	December 31, 2018	125	8.400	English

580	October 9, 2015	91	8.400	English/Ukranian/Russian
581	December 16, 2018	153	8.500	English
582	December 8, 2020	89	8.600	Portuguese
583	October 4, 2020	83	9.000	English

#################### MissingValues #####################

Title 0

Genre 0

Premiere 0

Runtime 0

IMDB Score 0

Language 0

dtype: int64

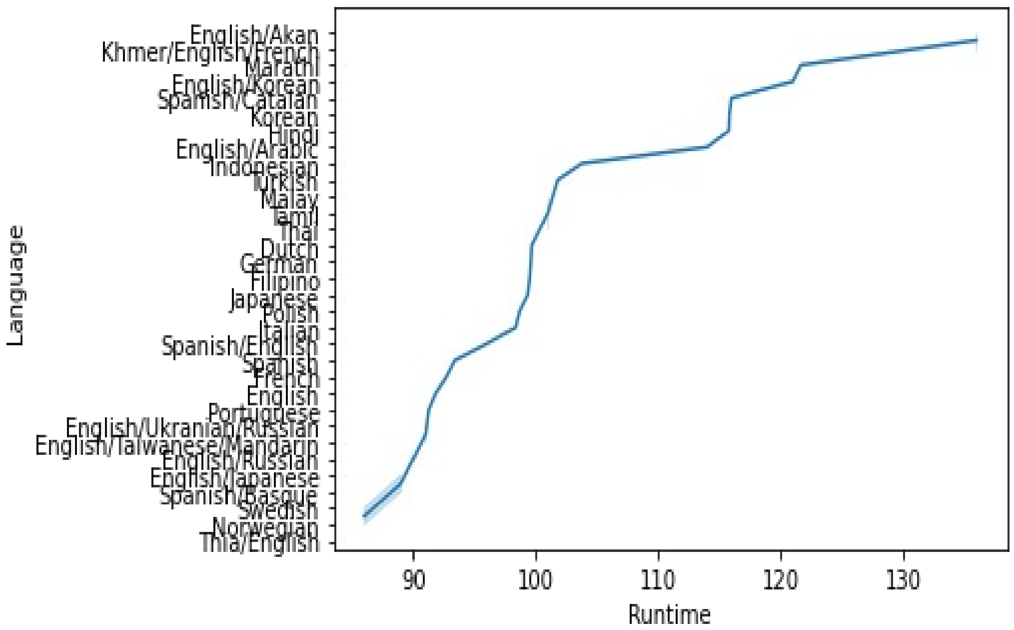
#################### Quantiles ####################

	0.000	0.050	0.250	0.500	0.750	0.950	0.990	1.000
Runtime	4.000	30.150	86.000	97.000	108.000	132.000	149.000	209.000
IMDB Score	2.500	4.600	5.700	6.350	7.000	7.700	8.317	9.000

	Language	Runtime
0	English/Akan	136.000
1	Khmer/English/French	136.000
2	Marathi	121.667
3	English/Korean	121.000
4	Spanish/Catalan	116.000
5	Korean	115.833
6	Hindi	115.788

7	English/Arabic	114.000
8	Indonesian	103.778
9	Turkish	101.800
10	Malay	101.000
11	Tamil	101.000
12	Thai	101.000
13	Dutch	99.667
14	German	99.600
15	Filipino	99.500
16	Japanese	99.333
17	Polish	98.667
18	Italian	98.357
19	Spanish/English	96.000
20	Spanish	93.387
21	French	92.700
22	English	91.818
23	Portuguese	91.250
24	English/Ukranian/Russian	91.000
25	English/Taiwanese/Mandarin	91.000
26	English/Russian	90.000
27	English/Japanese	89.000
28	Spanish/Basque	89.000
29	Swedish	86.000
30	Norwegian	86.000
31	Thia/English	80.000
32	English/Mandarin	59.000

33	Bengali	41.000
34	English/Swedish	40.000
35	English/Spanish	39.200
36	English/Hindi	32.500
37	Georgian	23.000



	Title	Genre	Premiere
0	Enter the Anime	Documentary	August 5, 2019
15	After the Raid	Documentary	December 19, 2019
20	Hello Privilege. It's Me, Chelsea	Documentary	September 13, 2019
30	After Maria	Documentary	May 24, 2019
111	Ghosts of Sugar Land	Documentary	October 16. 2019

	Runtime	IMDB Score	Language	Date
0	58	2.500	English/Japanese	2019-08-05

15	25	4.300	Spanish	2019-12-19
20	64	4.400	English	2019-09-13
30	37	4.600	English/Spanish	2019-05-24
111	21	5.500	English	2019-10-16

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",encoding = "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

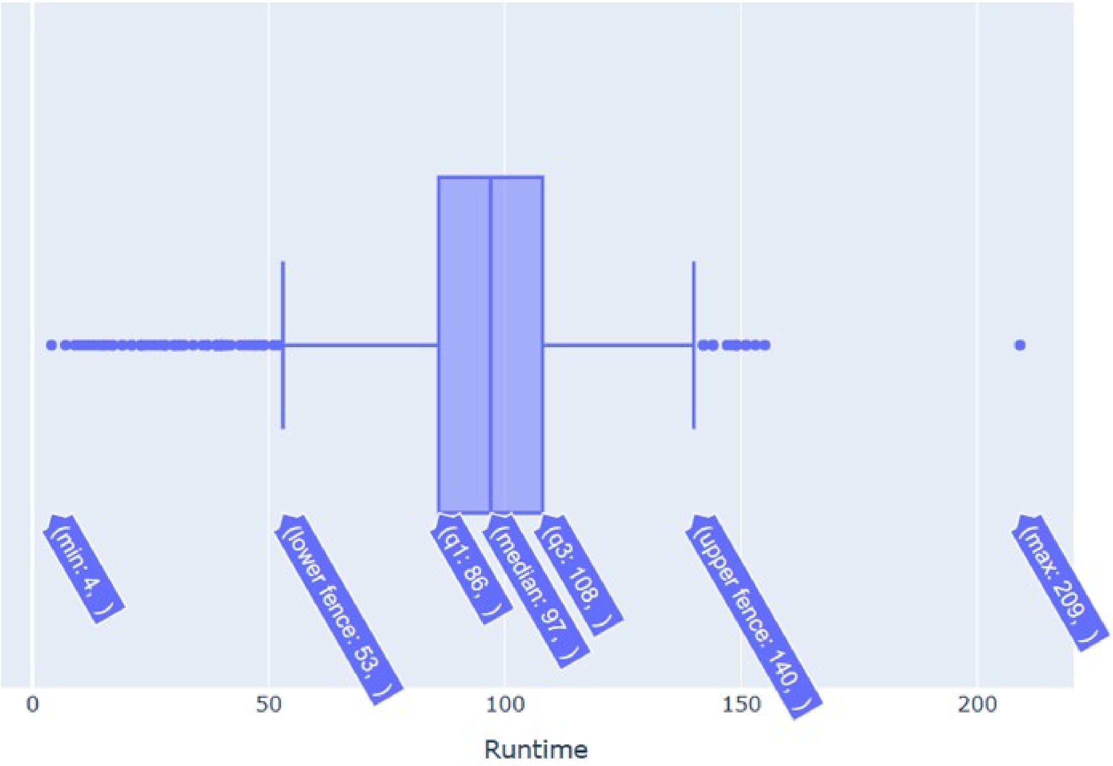
	Title	Genre	Premiere	Runtime	IMDB Score	Language	year	month	weekday
0	Enter the Anime	Documentary	2019-08-05	58	2.5	English/Japanese	2019	August	Monday
1	Dark Forces	Thriller	2020-08-21	81	2.6	Spanish	2020	August	Friday
2	The App	Science fiction/Drama	2019-12-26	79	2.6	Italian	2019	December	Thursday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

```python
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:

```python
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')
```

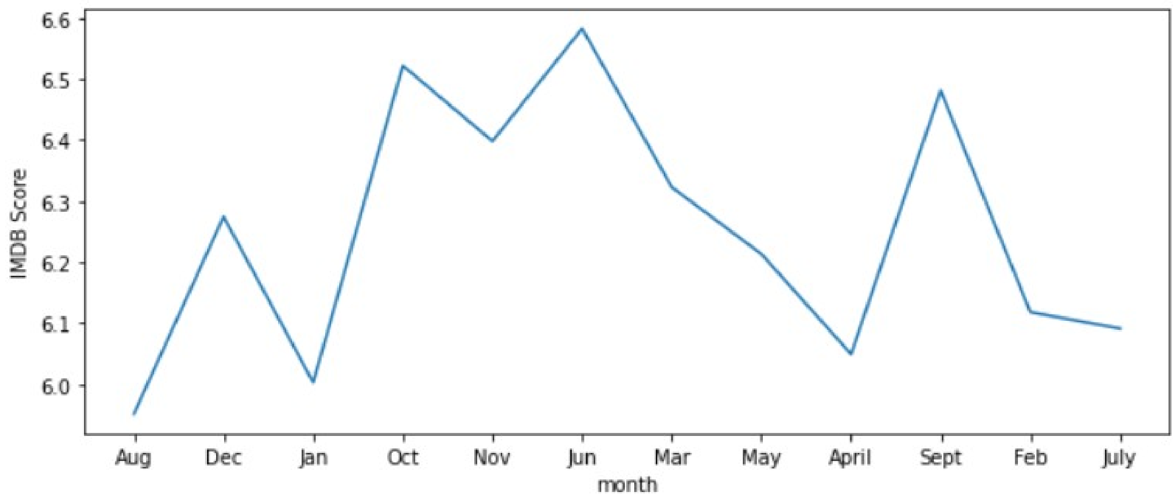
Output:

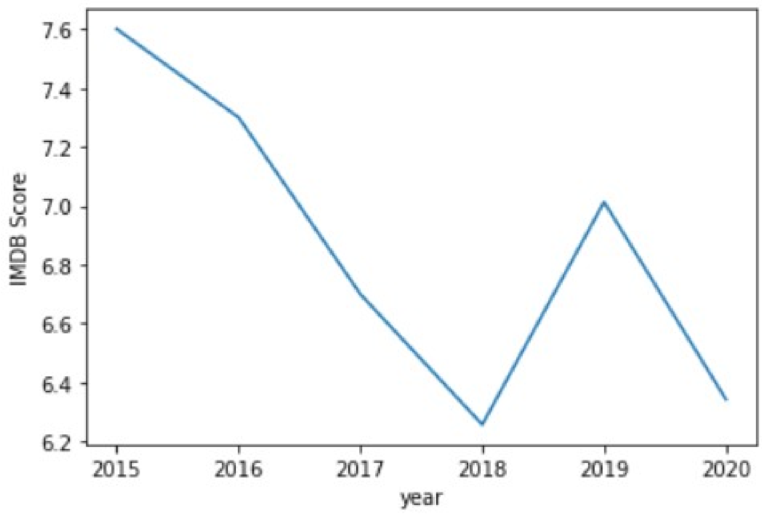
Mean Absolute Error (MAE): 0.51

Mean Squared Error (MSE): 0.42

Root Mean Squared Error (RMSE): 0.65

R-squared (R2): 0.75





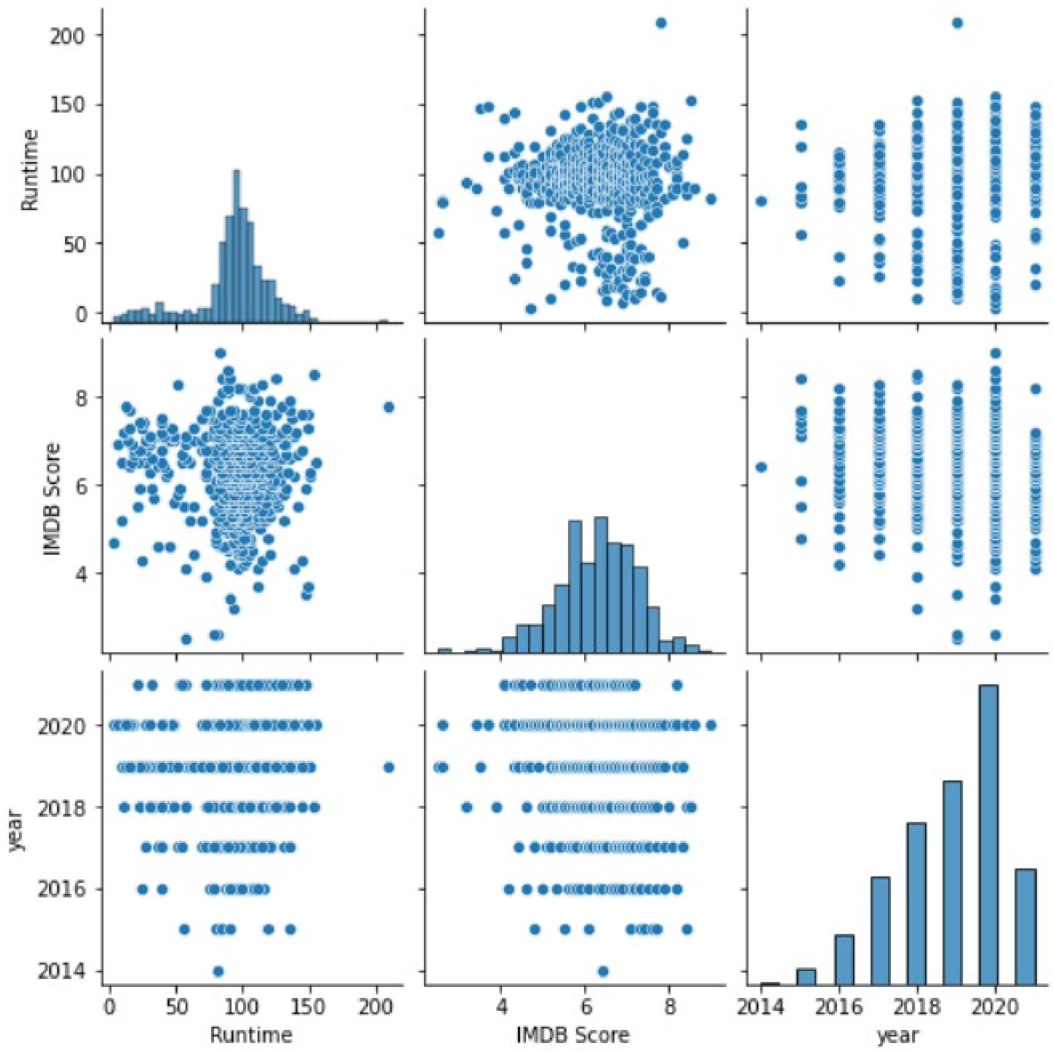
Here's a breakdown of what this code does:

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)
```

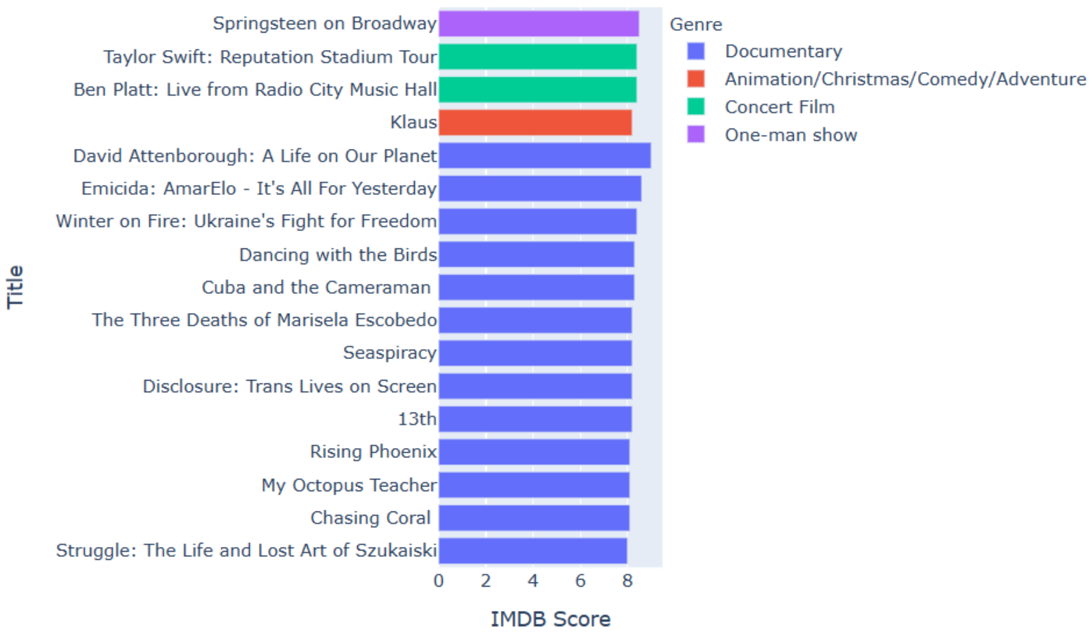
```
fig = px.bar(genre_low, x='Genre', y= 'IMDB Score', color = 'Title')
fig.show()
```

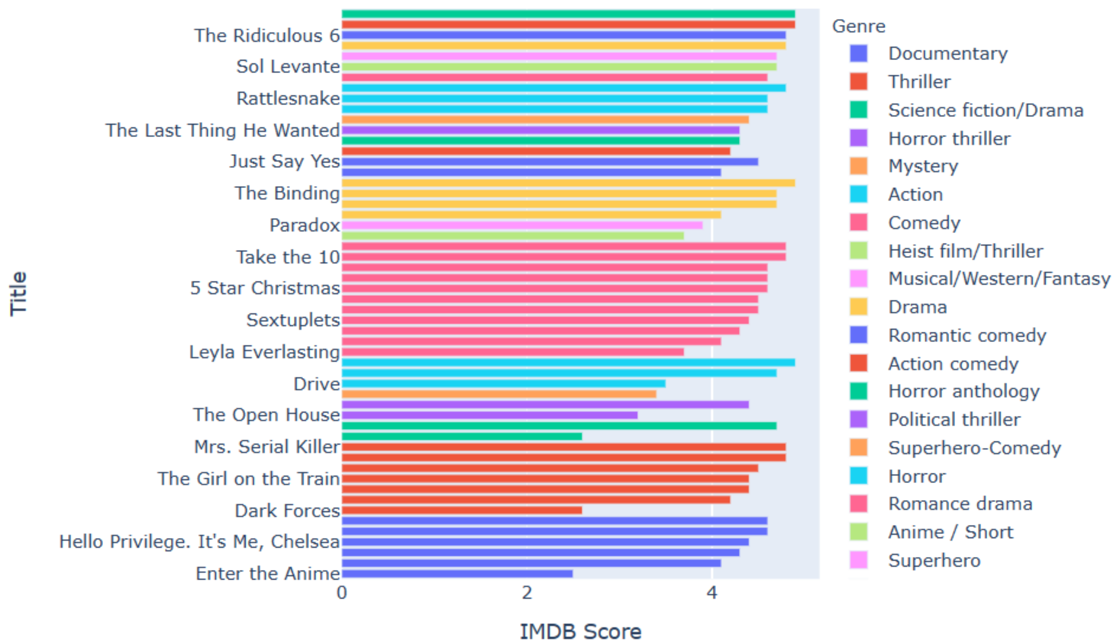
```
plt.figure(figsize = (28,9))
df.Genre.value_counts().plot(kind='bar')
```

output:

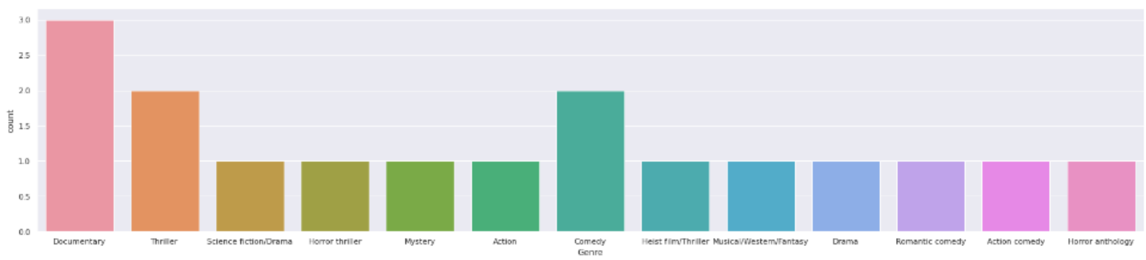
	Title	Genre	Premiere	Runtime	IMDB Score	Language
0	Enter the Anime	Documentary	August 5, 2019	58	2.5	English/Japanese
1	Dark Forces	Thriller	August 21, 2020	81	2.6	Spanish
2	The App	Science fiction/Drama	December 26, 2019	79	2.6	Italian
3	The Open House	Horror thriller	January 19, 2018	94	3.2	English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

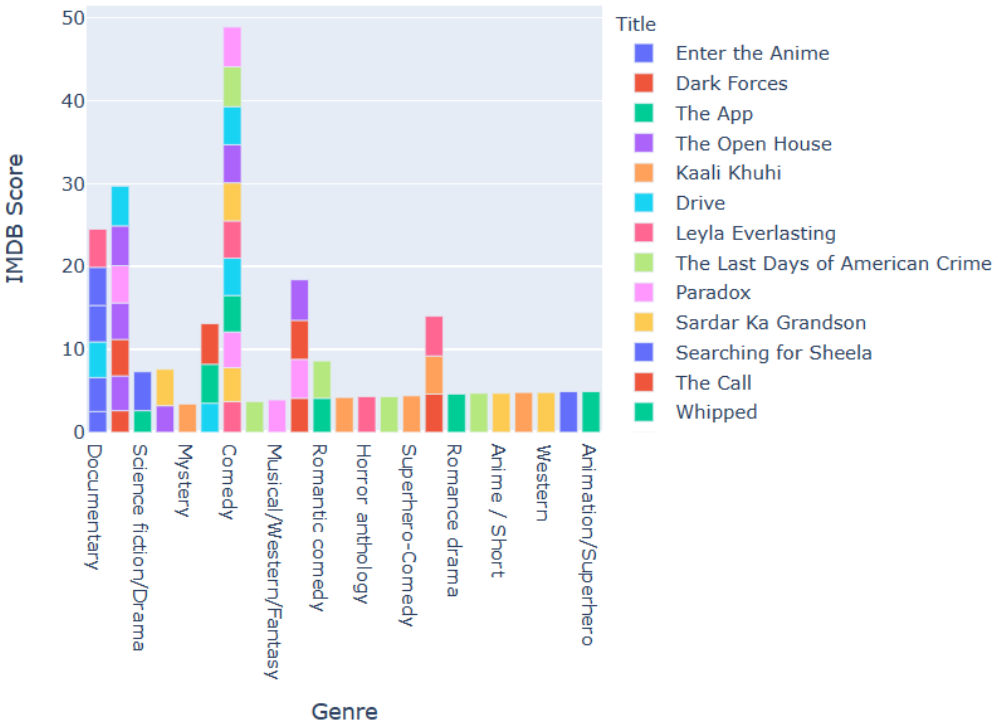
```
Documentary              159
Drama                     77
Comedy                    49
Romantic comedy           39
Thriller                  33
                        ...
Family/Comedy-drama        1
Historical drama           1
Musical / Short            1
Christmas musical          1
War-Comedy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

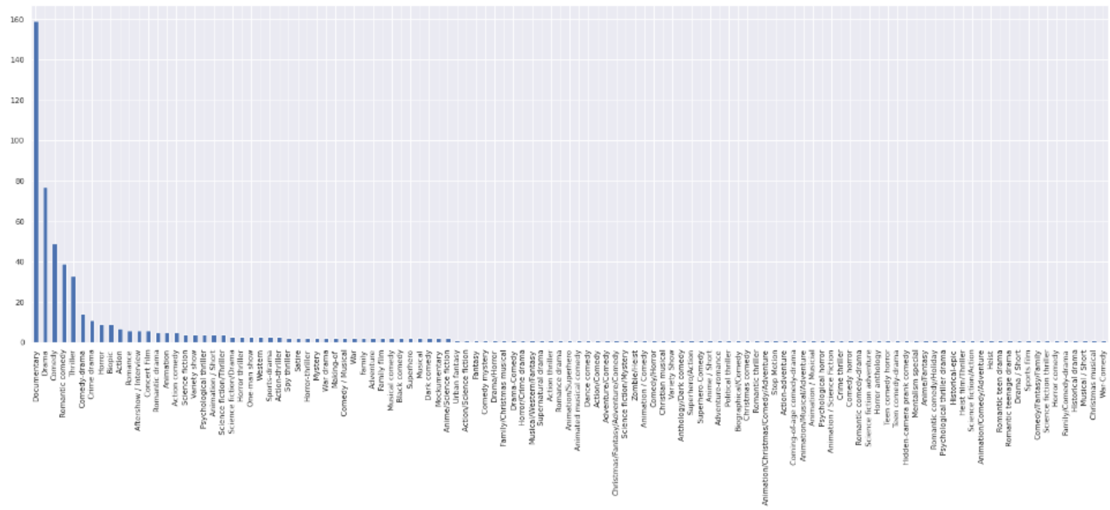




	Genre	Title	IMDB Score	Language
0	Documentary	Enter the Anime	2.5	English/Japanese
1	Thriller	Dark Forces	2.6	Spanish
2	Science fiction/Drama	The App	2.6	Italian
3	Horror thriller	The Open House	3.2	English
4	Mystery	Kaali Khuhi	3.4	Hindi
5	Action	Drive	3.5	Hindi
6	Comedy	Leyla Everlasting	3.7	Turkish
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.

Conclusion:

- Summarize the key achievements and contributions of your machine learning model in predicting IMDb scores for movies.
