**Predicting IMDb Scores Using Machine Learning**

TEAM MEMBER

732521104046 : SRI LAKSHMI T

**Phase 5 Submission Document**

**Project :** Predicting IMDb Scores

**Introduction:**

IMDb scores have become an important metric for evaluating the quality and popularity of movies and TV shows. IMDb (Internet Movie Database) is a popular online database that allows users to rate and review films and television series. Predicting IMDb scores can provide valuable insights to filmmakers, studios, and viewers.

Predicting IMDb scores is a challenging task that often relies on machine learning algorithms. These algorithms analyze various features of a movie or TV show, such as genre, cast, director, release year, runtime, and production budget. By training on historical data, the algorithms can learn patterns and relationships between these features and IMDb scores.

Machine learning models, such as regression models or ensemble methods, can be employed to predict the IMDb scores of movies and TV shows based on these features. These models are trained on a large dataset that contains information about previously released films and their corresponding IMDb scores.

However, it is important to note that predicting IMDb scores is not an exact science. IMDb scores are subjective, as they are based on the opinions and ratings of individual users. Various factors such as marketing, reviews from critics, and cultural biases can also influence IMDb scores. Therefore, while predictive models can provide reasonable estimates, they cannot guarantee the accuracy of future IMDb scores.

In summary, predicting IMDb scores is a complex task that involves machine learning algorithms analyzing different movie features to estimate the likelihood of a particular rating. While these predictions can be helpful, they should be interpreted with caution, as IMDb scores are subjective and subject to various influences.

**Problem Definition:**

The problem is to develop a machine learning model that predicts IMDb scores of movies available on Films based on features like genre, premiere date, runtime, and language. The objective is to create a model that accurately estimates the popularity of movies, helping users discover highly rated films that match their preferences. This project involves data preprocessing, feature engineering, model selection, training, and evaluation

In this part you will need to understand the problem statement and create a document on what have you understood and how will you proceed ahead with solving the problem. Please think on a design and present in form of the document.

In this part you will need to understand the problem statement and create a document on what have you understood and how will you proceed ahead with solving the problem. Please think on a design and present in form of a document.

In this part you will begin building your project by loading and preprocessing the dataset. Begin building the IMDb score prediction model by loading and preprocessing the dataset. Load the movie dataset and preprocess the data for analysis.

In this part you will Continue building the IMDb score prediction model by:

* Feature engineering
* Model training
* Evaluation.

**Design Thinking:**

1. **Data Source:** Utilize a dataset containing information about movies, including features like genre, premiere date, runtime, language, and IMDb scores.
2. **Data Preprocessing:** Clean and preprocess the data, handle missing values, and convert categorical features into numerical representations.
3. **Feature Engineering:** Extract relevant features from the available data that could contribute to predicting IMDb scores.
4. **Model Selection:** Choose appropriate regression algorithms (e.g., Linear Regression, Random Forest Regressor) for predicting IMDb scores.
5. **Model Training:** Train the selected model using the preprocessed data.
6. **Evaluation:** Evaluate the model's performance using regression metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared.

**Working Methodology:**

The working method for this work involves few steps. The methodology is shown in figure 1. The steps are described below.

• Data Extraction

• Data Preprocessing

• Applying Machine Learning Techniques

• Comparing the results of different algorithms

DATA EXTRACTION

DATA PREPROCESSING

MACHINE LEARNING TECHNIQUES

**Algorithm :**

Algorithm for developing the model

1: Prepare data set

2: Check Minority

3: If needed apply SMOTE algorithm until the minority class becomes equal to the size of it’s closest class 4: Classification

5: Accuracy ←− 0

6: while True do

7: Resample Data

8: Call (Classifier)

9: if % of correctly classified Instance >Previous Accuracy Measure then

10: Accuracy ←− % of correctly classif ied Instance

11: else

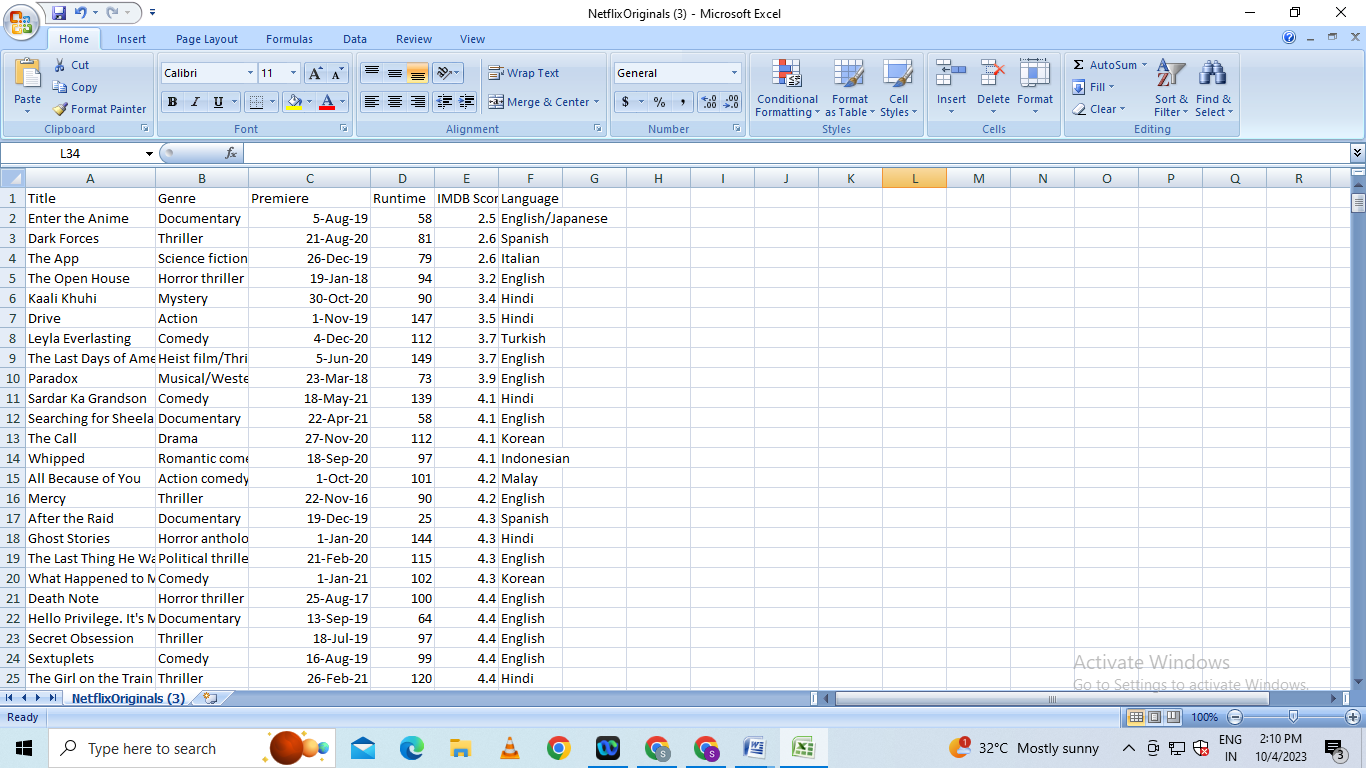
12: Break

13: end if

14: end while=0

**Data Source:**

A Good Data for Predicting IMDb Scores using machine learning model should be Accurate , complete , covering the geographic area of interest , accessible

Dataset Link :[**https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores**](https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores)

**Data Preprocessing:**

Data preprocessing is the critical first step in any machine learning project.It involves cleaning the data,removing outliers and handling missing values to prepare the dataset for model training. In the context of the predicting the IMDB scores project , let’s elaborate on the specific steps:

1. **Duplicate Removal:**

Duplicate rows can introduce bias into model.We will identify and remove duplicates,typically by sorting the dataset based on unique identifier and then eliminating consecutive rows with same identifiers

**b)Handling Missing Values:**

Missing data is common and needs to be addressed . We will utilize suitable methods such as :

* **Mean Imputation**
* **Median Imputation**

# This Python 3 environment comes with many helpful analytics libraries installed

# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python

# For example, here's several helpful packages to load

import numpy as np *# linear algebra*

import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

# Input data files are available in the read-only "../input/" directory

# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os

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

for filename **in** filenames:

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

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"

# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session

/kaggle/input/netflix-original-films-imdb-scores/NetflixOriginals.csv

In [2]:

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

from datetime import datetime,timedelta

**Dataset**

In [3]:

ds = pd.read\_csv("/kaggle/input/netflix-original-films-imdb-scores/NetflixOriginals.csv",encoding = "ISO-8859-1")

ds\_date = ds.copy()

ds.head(5)

Out[3]:

|  | 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 |

In [4]:

ds.describe().T

Out[4]:

|  | count | mean | std | min | 25% | 50% | 75% | max |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Runtime | 584.0 | 93.577055 | 27.761683 | 4.0 | 86.0 | 97.00 | 108.0 | 209.0 |
| IMDB Score | 584.0 | 6.271747 | 0.979256 | 2.5 | 5.7 | 6.35 | 7.0 | 9.0 |

insights: categorical of IMDB Score 5.7 > rendah 6.35 > sedang 7.0 > tinggi 9.0 > sangat tinggi

In [5]:

ds.info(verbose=True,show\_counts=True)

<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

In [6]:

ds.isna().sum()

Out[6]:

Title 0

Genre 0

Premiere 0

Runtime 0

IMDB Score 0

Language 0

dtype: int64

In [7]:

ds['Title'].value\_counts()

Out[7]:

Enter the Anime 1

Have a Good Trip: Adventures in Psychedelics 1

Tallulah 1

The Old Guard 1

Tony Robbins: I Am Not Your Guru 1

..

Cam 1

Earthquake Bird 1

Frankenstein's Monster's Monster, Frankenstein 1

Horse Girl 1

David Attenborough: A Life on Our Planet 1

Name: Title, Length: 584, dtype: int64

In [8]:

ds['Genre'].value\_counts()

Out[8]:

Documentary 159

Drama 77

Comedy 49

Romantic comedy 39

Thriller 33

...

Romantic comedy-drama 1

Heist film/Thriller 1

Musical/Western/Fantasy 1

Horror anthology 1

Animation/Christmas/Comedy/Adventure 1

Name: Genre, Length: 115, dtype: int64

In [9]:

ds['Premiere'].value\_counts()

Out[9]:

October 2, 2020 6

November 1, 2019 5

October 18, 2019 5

November 2, 2018 4

June 19, 2020 4

..

September 20, 2019 1

March 10, 2017 1

March 17, 2017 1

May 29, 2015 1

October 4, 2020 1

Name: Premiere, Length: 390, dtype: int64

In [10]:

ds\_date["Premiere"] = ds\_date["Premiere"].apply(lambda x: "".join(x for x **in** x.replace(".",",")))

ds\_date["PremiereDate"] = ds\_date["Premiere"].apply(lambda x: datetime.strptime(x, "%B **%d**, %Y").date())

ds\_date["Year"] = ds\_date["Premiere"].apply(lambda x: "".join(x for x **in** x.replace(",","").split()[-1]))

*#Convert object to date*

ds\_date["PremiereDate"] = pd.to\_datetime(ds\_date["PremiereDate"])

ds\_date

Out[10]:

|  | Title | Genre | Premiere | Run time | IMDB Score | Language | PremiereDate | Year |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | Enter the Anime | Documentary | August 5, 2019 | 58 | 2.5 | English/Japanese | 2019-08-05 | 2019 |
| 1 | Dark Forces | Thriller | August 21, 2020 | 81 | 2.6 | Spanish | 2020-08-21 | 2020 |
| 2 | The App | Science fiction/Drama | December 26, 2019 | 79 | 2.6 | Italian | 2019-12-26 | 2019 |
| 3 | The Open House | Horror thriller | January 19, 2018 | 94 | 3.2 | English | 2018-01-19 | 2018 |
| 4 | Kaali Khuhi | Mystery | October 30, 2020 | 90 | 3.4 | Hindi | 2020-10-30 | 2020 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 579 | Taylor Swift: Reputation Stadium Tour | Concert Film | December 31, 2018 | 125 | 8.4 | English | 2018-12-31 | 2018 |
| 580 | Winter on Fire: Ukraine's Fight for Freedom | Documentary | October 9, 2015 | 91 | 8.4 | English/Ukranian/Russian | 2015-10-09 | 2015 |
| 581 | Springsteen on Broadway | One-man show | December 16, 2018 | 153 | 8.5 | English | 2018-12-16 | 2018 |
| 582 | Emicida: AmarElo - It's All For Yesterday | Documentary | December 8, 2020 | 89 | 8.6 | Portuguese | 2020-12-08 | 2020 |
| 583 | David Attenborough: A Life on Our Planet | Documentary | October 4, 2020 | 83 | 9.0 | English | 2020-10-04 | 2020 |

584 rows × 8 columns

In [11]:

ds\_date.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 584 entries, 0 to 583

Data columns (total 8 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

6 PremiereDate 584 non-null datetime64[ns]

7 Year 584 non-null object

dtypes: datetime64[ns](1), float64(1), int64(1), object(5)

memory usage: 36.6+ KB

In [12]:

ds['Language'].value\_counts()

Out[12]:

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

Norwegian 1

Malay 1

English/Ukranian/Russian 1

Name: Language, dtype: int64

EDA

In [13]:

ds['Genre'].value\_counts()

genre = ds['Genre'].value\_counts()

genre.head()

Out[13]:

Documentary 159

Drama 77

Comedy 49

Romantic comedy 39

Thriller 33

Name: Genre, dtype: int64

In [14]:

plt.figure(figsize=(16, 5))

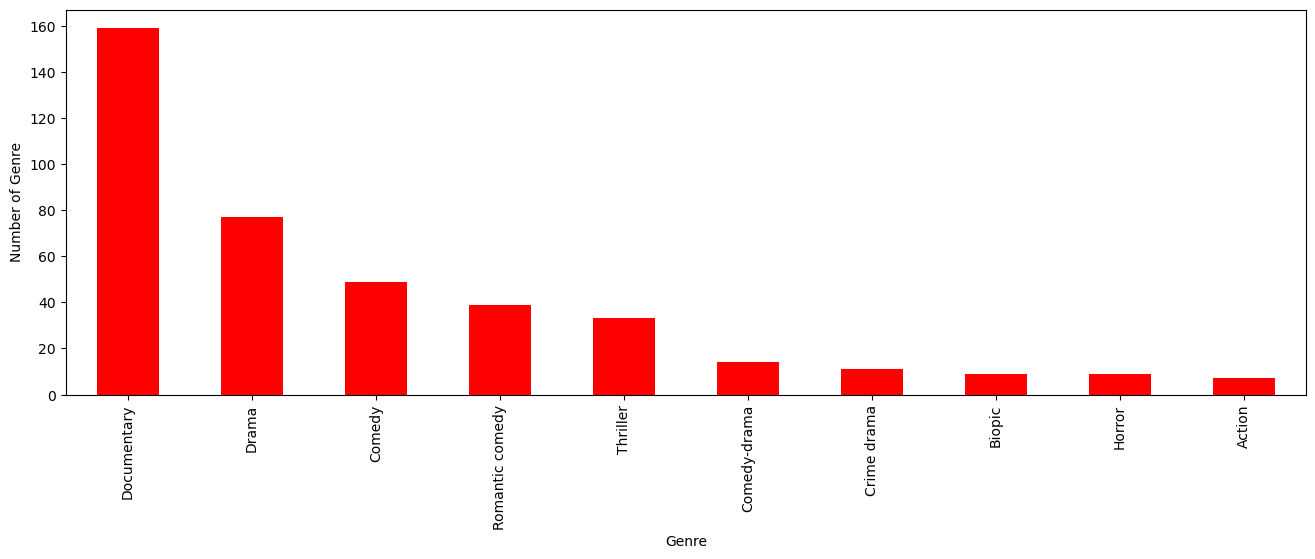
ds['Genre'].value\_counts().head(10).plot(kind='bar', color='red')

plt.xlabel('Genre')

plt.ylabel('Number of Genre')

plt.xticks(rotation=90)

plt.show(block=True)



insights: the most popular movies from genre is documentary

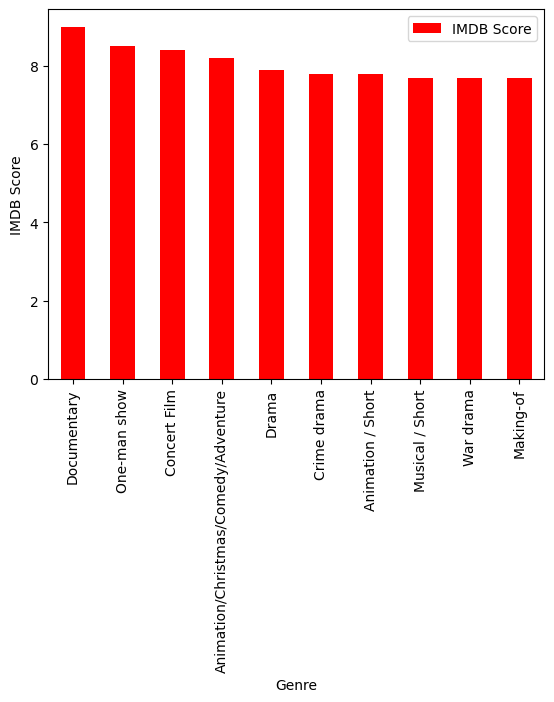
In [15]:

ds[['Genre', 'IMDB Score']].sort\_values('IMDB Score', ascending=False).drop\_duplicates('Genre').head(10).plot(x='Genre', y='IMDB Score', kind='bar', color='red')

plt.xlabel('Genre')

plt.ylabel('IMDB Score')

plt.show(block=True)



In [16]:

ds['Language'].value\_counts()

Out[16]:

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

Norwegian 1

Malay 1

English/Ukranian/Russian 1

Name: Language, dtype: int64

In [17]:

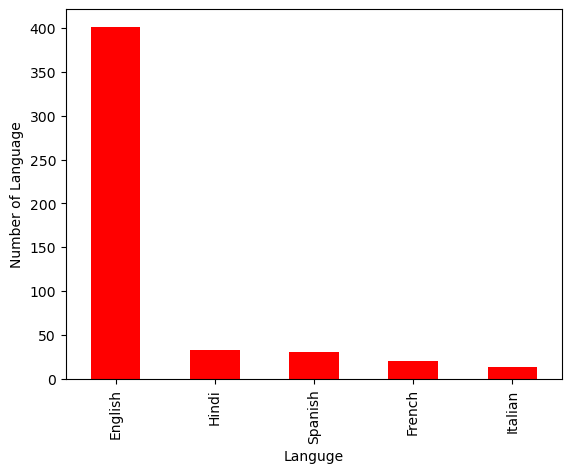
ds\_lang = ds['Language'].value\_counts()

ds\_lang.head(5).plot(kind='bar', color='red')

plt.xlabel('Languge')

plt.ylabel('Number of Language')

plt.show(block=True)



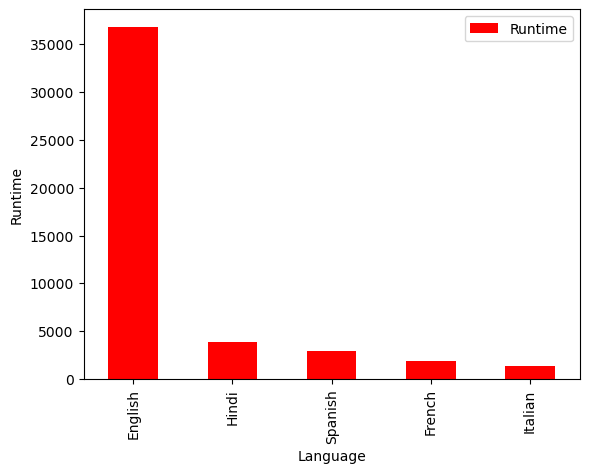
In [18]:

ds.groupby('Language').agg({'Runtime': 'sum'}).sort\_values('Runtime', ascending=False).head(5).plot(kind='bar',color='red')

plt.xlabel('Language')

plt.ylabel('Runtime')

plt.show(block=True)



In [19]:

ds\_english = ds[ds['Language'] == 'English'].sort\_values('IMDB Score', ascending=False)

ds\_english.head()

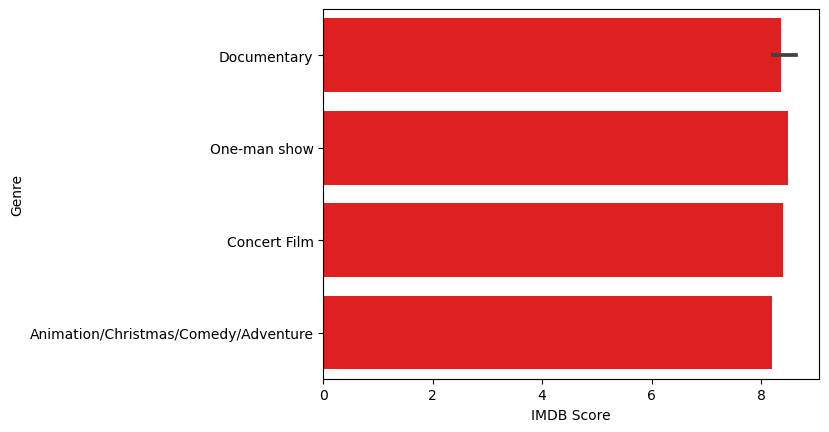
Out[19]:

|  | Title | Genre | Premiere | Runtime | IMDB Score | Language |
| --- | --- | --- | --- | --- | --- | --- |
| 583 | David Attenborough: A Life on Our Planet | Documentary | October 4, 2020 | 83 | 9.0 | English |
| 581 | Springsteen on Broadway | One-man show | December 16, 2018 | 153 | 8.5 | English |
| 579 | Taylor Swift: Reputation Stadium Tour | Concert Film | December 31, 2018 | 125 | 8.4 | English |
| 578 | Ben Platt: Live from Radio City Music Hall | Concert Film | May 20, 2020 | 85 | 8.4 | English |
| 577 | Dancing with the Birds | Documentary | October 23, 2019 | 51 | 8.3 | English |

In [20]:

sns.barplot(y=ds\_english['Genre'].head(10), x=ds\_english['IMDB Score'], color='red')

plt.show(block=True)

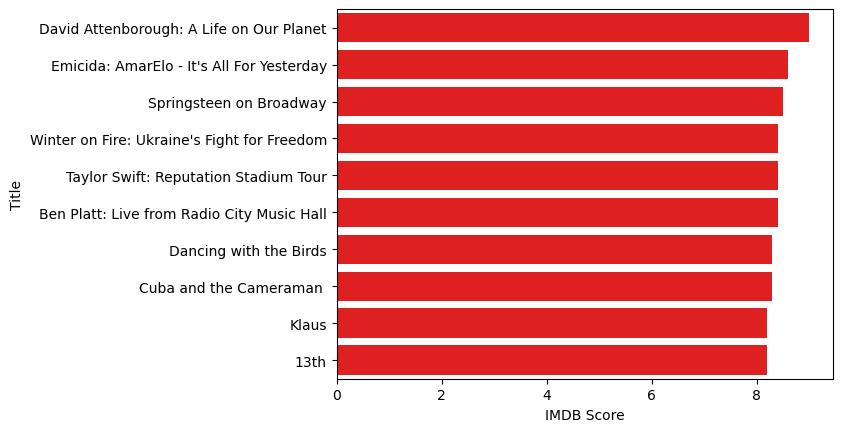


In [21]:

ds\_movie = ds[['Title', 'IMDB Score']].sort\_values('IMDB Score', ascending=False).head(10)

sns.barplot(y='Title', x='IMDB Score', data=ds\_movie, color='red')

plt.show(block=True)



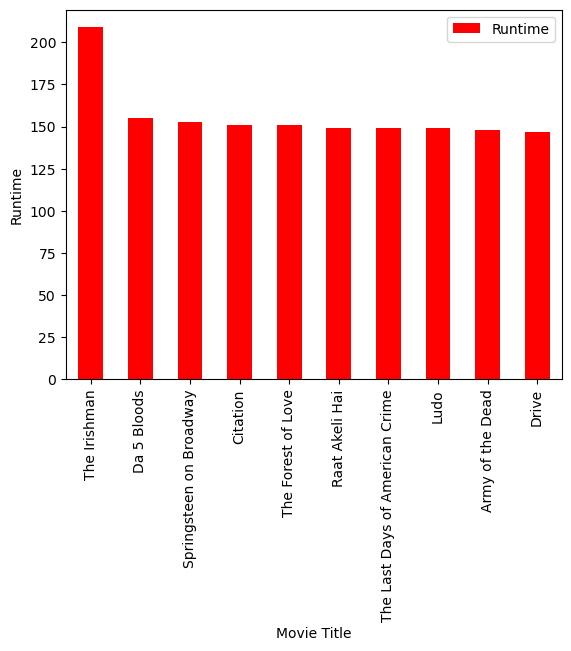
In [22]:

ds[['Title', 'Runtime']].sort\_values('Runtime', ascending=False).head(10).plot(x='Title', y='Runtime', kind='bar', color='red')

plt.xlabel('Movie Title')

plt.ylabel('Runtime')

plt.show(block=True)



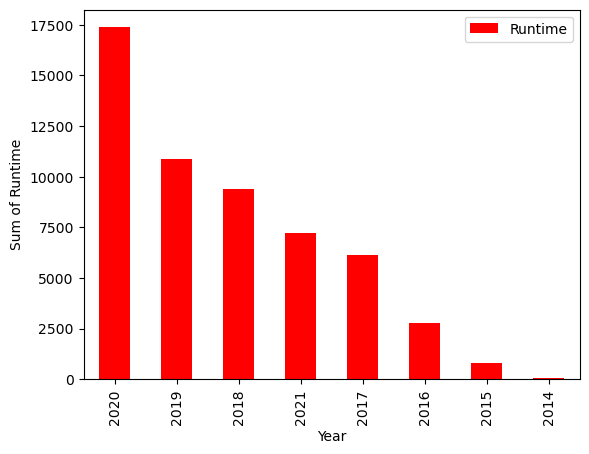
In [23]:

ds\_date.groupby('Year').agg({'Runtime': 'sum'}).sort\_values('Runtime', ascending=False).plot(kind='bar', color='red')

plt.xlabel('Year')

plt.ylabel('Sum of Runtime')

plt.show(block=True)



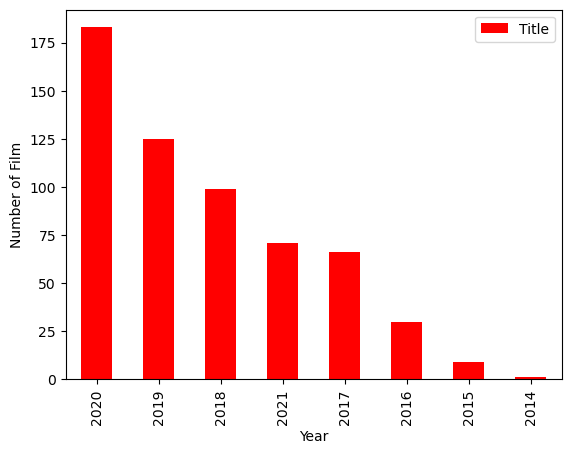
In [24]:

ds\_date.groupby('Year').agg({'Title': 'count'}).sort\_values('Title', ascending=False).plot(kind='bar', color='red')

plt.xlabel('Year')

plt.ylabel('Number of Film')

plt.show(block=True)



**Model Evaluation and Selection:**

* Split the dataset into training and testing sets.
* Evaluate models using appropriate metrics (e.g., Mean Absolute Error, Mean Squared Error, R-squared) to assess their performance.
* Use cross-validation techniques to tune hyperparameters and ensure model stability. Compare the results with traditional linear regression models to highlight improvements.
* Select the best-performing model for further analysis.

**Model Interpretability:**

* Explain how to interpret feature importance from Gradient Boosting and Neural Networks.
* Discuss the insights gained from feature importance analysis and their relevance to IDMb scores prediction.
* Interpret feature importance from ensemble models like Random Forest and Gradient
* Boosting to understand the factors influencing IDMb.

**Deployment and Prediction:**

* Deploy the chosen regression model to predict IDMb.
* Develop a user-friendly interface for users to input property features and receive IDMb scores predictions.

**Program:**

**IDMb Score Prediction**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import r2\_score,mean\_absolute\_error, mean\_squared\_error

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import Lasso

from sklearn.ensemble import RandomForestRegressor

from sklearn.svm import SVR

import warnings

warnings.filterwarnings("ignore")

warnings.warn(" A NumPy version>={np\_minversion) and <{np\_maxversion}")

dataset = pd.read\_csv("C:/ibm:/Netflix.csv”)

**Model 1 - Linear Regression**

**In [1]:**

model\_Ir-LinearRegression()

**In [2]:**

model\_Ir.fit(X\_train\_scal, Y\_train)

**Out[2]:**

Linear Regression

Linear Regression()

**Predicting Prices**

**In [3]:**

Prediction= model\_Ir.predict(X\_test\_scal)

**Evaluation of Predicted Data**

**In [4]:**

plt.figure(figsize=(12,6))

plt.plot(np.arange(len(Y\_test)), Y\_test, label=Actual Trend)

plt.plot(np.arange(len(Y\_test)), Prediction1, label="Predicted Trend)

plt.xlabel(”Data”)

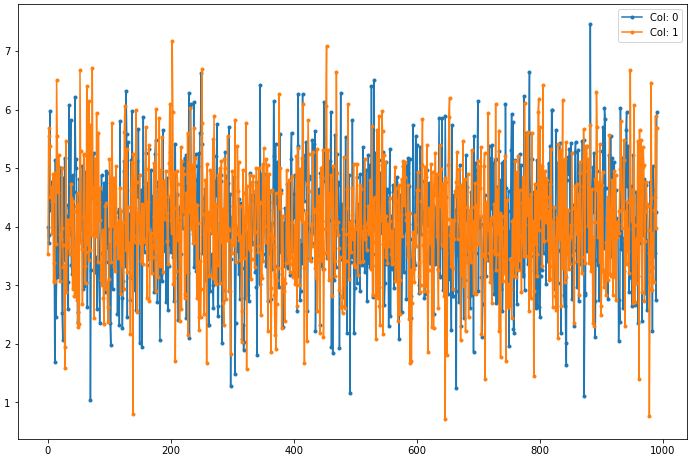
plt.ylabel("Trend”)

plt.legend()

plt.title(“Actual vs Predicted”)

**Out[4]:**

Text(0.5, 1.0, 'Actual vs Predicted')

Actual vs Predicted

**DATA LOADING IN MACHINE LEARNING :**

To load data for machine learning, you typically need to follow these steps:

1. Import the necessary libraries: Start by importing the required libraries such as NumPy, Pandas, or TensorFlow, depending on your specific needs.

2. Obtain the data: Get the dataset that you want to use for machine learning. This data can come from various sources, such as CSV files, databases, or APIs.

3. Load the data into memory: Use the appropriate functions provided by the libraries to read and load the data into memory. For example, you can use Pandas' read\_csv() function to load data from a CSV file into a DataFrame.

4. Explore and preprocess the data: Explore the loaded dataset to get a better understanding of its structure and contents. Perform data preprocessing steps like handling missing values, normalizing or standardizing features, and encoding categorical variables.

5. Split the data: Split the dataset into training and test sets. The training set is used to train the machine learning model, while the test set is used to evaluate its performance.

6. Further split the data (optional): Additionally, you can split the training set into training and validation sets. This allows you to monitor the model's performance during training and tune the hyperparameters accordingly.

7. Convert the data: Convert the data into a suitable format for model training. This might involve transforming text data into numerical representations using techniques like tokenization or one-hot encoding.

8. Finally, feed the data into your machine learning model for training or evaluation.

Here's an example Python program that outlines the steps mentioned earlier for loading data for machine learning using the Pandas library:

import pandas as pd

Step 1: Import necessary libraries

Step 2: Obtain the data

# Assuming the data is in a CSV file named 'data.csv' in the same directory

data = pd.read\_csv('data.csv')

Step 3: Load the data into memory

Step 4: Explore and preprocess the data

# Example: Handling missing values

data = data.dropna() # Remove rows with missing values

# Example: Splitting the data into features (X) and target variable (y)

X = data.drop('target', axis=1) # Assumes 'target' is the column containing the target values

y = data['target']

Step 5: Split the data into training and test sets

Step 6: Further split the data (optional)

Step 7: Convert the data

Step 8: Feed the data into your machine learning model for training or evaluation

**Program:**

To predict IMDb scores using Python, you can use various machine learning algorithms. Here's a basic example using linear regression:

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

**Step 1:** Load the dataset

data\_path = "path/to/dataset.csv"

# Replace with the actual path to your dataset

data = pd.read\_csv(data\_path)

**Step 2:** Split the data into features (X) and target variable (y)

X = data.drop('IMDb\_Score', axis=1) # Assuming 'IMDb\_Score' is the column to predict

y = data['IMDb\_Score']

**Step 3:** Split the data into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=123)

**Step 4:** Create and train the model

model = LinearRegression()

model.fit(X\_train, y\_train)

**Step 5:** Predict using the trained model

y\_pred = model.predict(X\_test)

**Step 6:** Evaluate the model

mse = mean\_squared\_error(y\_test, y\_pred)

print("Mean squared error:", mse)

# You can further assess the model's performance using other metrics or perform hyperparameter tuning as needed.

This is a basic example using linear regression. Depending on the nature of your dataset and the specific problem, these might want to use different algorithms or apply additional preprocessing and feature engineering techniques.

**PREPROCESSING OF DATASET :**

* Preprocessing of a dataset refers to the steps taken to transform and prepare the data for further analysis or modeling.
* It involves handling missing values, removing irrelevant or noisy data, transforming categorical variables, and scaling numeric data, among other tasks.
* The preprocessing steps vary depending on the nature of the dataset and the specific requirements of the problem at hand.

**SYNTAX FOR PREPROCESSING THE DATASET :**

The Syntax for preprocessing data depends on programming language and libraries you are using. Here we are using more libraries such as pandas etc..

**Here are some commonly performed preprocessing steps:**

1. Handling missing data: This involves identifying and dealing with missing values in the dataset. Options include removing rows or columns with missing values, imputing missing values with the mean or median, or using advanced imputation techniques.

2. Removing irrelevant data: If the dataset contains unnecessary columns or attributes that won't contribute to the analysis or modeling, those columns can be dropped or removed from the dataset.

3. Handling categorical variables: Categorical variables, such as text labels or nominal values, need to be converted into a numerical representation for machine learning algorithms to process them. This process is known as encoding or one-hot encoding and can be done using techniques like label encoding or creating dummy variables.

4. Scaling numeric data: Numeric variables that have different scales or units can be standardized or normalized to ensure they are on a similar scale. Common techniques include min-max scaling or standardization using mean and standard deviation.

5. Handling outliers: Outliers, which are extreme or unusual values in the dataset, can impact the modeling process. They can be treated by either removing them, transforming them, or using robust algorithms that are less sensitive to outliers.

6. Feature engineering: This involves creating new features from existing ones that may improve the performance of the model. It could include operations like polynomial expansion, log transforms, or creating interaction terms.

7. Splitting into training and testing sets: The dataset is typically split into a training set and a testing set to assess the model's performance. The training set is used to train the model, while the testing set is used to evaluate its performance on unseen data.

These are general steps in preprocessing a dataset, and the specific preprocessing required may vary depending on the dataset and the machine learning task being undertaken.

**Steps Involved in Data Preprocessing:**

**1. Data Cleaning:**

The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

**(a). Missing Data:**

This situation arises when some data is missing in the data. It can be handled in various ways.

Some of them are:

Ignore the tuples: This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.

Fill the Missing values: There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

**(b). Noisy Data:**

Noisy data is a meaningless data that can’t be interpreted by machines.It can be generated due to faulty data collection, data entry errors etc.

It can be handled in following ways :

Binning Method: This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task. Each segmented is handled separately. One can replace all data in a segment by its mean or boundary values can be used to complete the task.

Regression:Here data can be made smooth by fitting it to a regression function.The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).

Clustering: This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

**2. Data Transformation:**

This step is taken in order to transform the data in appropriate forms suitable for mining process. This involves following ways:

Normalization: It is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)

Attribute Selection: In this strategy, new attributes are constructed from the given set of attributes to help the mining process.

Discretization: This is done to replace the raw values of numeric attribute by interval levels or conceptual levels.

Concept Hierarchy Generation: Here attributes are converted from lower level to higher level in hierarchy. For Example-The attribute “city” can be converted to “country”.

**3. Data Reduction:**

Data reduction is a crucial step in the data mining process that involves reducing the size of the dataset while preserving the important information. This is done to improve the efficiency of data analysis and to avoid overfitting of the model. Some common steps involved in data reduction are:

Feature Selection:This involves selecting a subset of relevant features from the dataset. Feature selection is often performed to remove irrelevant or redundant features from the dataset. It can be done using various techniques such as correlation analysis, mutual information, and principal component analysis (PCA).

Feature Extraction: This involves transforming the data into a lower-dimensional space while preserving the important information. Feature extraction is often used when the original features are high-dimensional and complex. It can be done using techniques such as PCA, linear discriminant analysis (LDA), and non-negative matrix factorization (NMF).

Sampling:This involves selecting a subset of data points from the dataset. Sampling is often used to reduce the size of the dataset while preserving the important information. It can be done using techniques such as random sampling, stratified sampling, and systematic sampling.

Clustering: This involves grouping similar data points together into clusters. Clustering is often used to reduce the size of the dataset by replacing similar data points with a representative centroid. It can be done using techniques such as k-means, hierarchical clustering, and density-based clustering.

Compression:This involves compressing the dataset while preserving the important information. Compression is often used to reduce the size of the dataset for storage and transmission purposes. It can be done using techniques such as wavelet compression, JPEG compression, and gzip compression.

**FEATURE ENGINEERING :**

Feature engineering is a process in machine learning and data analysis that involves creating new, derived features from the existing data to improve the performance of a predictive model. It involves transforming and enhancing the raw data by extracting useful information or creating new representations that can capture the underlying patterns and relationships.

**Here are some common techniques used in feature engineering:**

1. Encoding Categorical Variables: Convert categorical variables into numerical representations that machine learning models can understand. This can be done using techniques like one-hot encoding, ordinal encoding, or label encoding.

2. Handling Missing Values: Deal with missing data by imputing values based on strategies like mean, median, mode, or using more complex methods like k-nearest neighbors or predictive models.

3. Scaling and Normalization: Scale numerical features to bring them to a similar range or distribute them normally. Common techniques include standardization (subtracting mean and dividing by standard deviation) or min-max scaling (scaling between a specified range).

4. Binning: Group continuous numeric features into bins or intervals to convert them into categorical variables. This can help capture non-linear relationships or handle outliers.

5. Feature Interaction: Create new features by combining or interacting existing features. For example, combining height and weight to calculate BMI or creating interactions between different categorical variables.

6. Time-based Features: Extract relevant information from date or time variables, such as day of the week, month, or time duration since a particular event.

7. Textual Features: Process and extract features from text data, such as word counts, TF-IDF vectors, n-grams, or word embeddings. These transformations can be useful for natural language processing tasks.

8. Feature Selection: Use techniques like correlation analysis, mutual information, or regularization methods to select the most informative features and eliminate redundant or irrelevant ones. This helps reduce overfitting and improve model performance.

9. Domain-Specific Features: Incorporate domain knowledge by creating features that capture specific characteristics or patterns related to the problem at hand. This often requires expertise in the particular application area.

10. Feature Extraction from Images or Audio: If the data involves images or audio, techniques like convolutional neural networks (CNNs) or Mel-frequency cepstral coefficients (MFCC) can be used to extract meaningful features.

It's important to note that feature engineering is an iterative process and requires experimentation to find the most effective features for a given problem. Additionally, it is essential to balance the complexity of features with the size and quality of the dataset to avoid overfitting or introducing unnecessary noise.

**Procedure :**

To build an IMDb score prediction model, feature engineering plays a crucial role in extracting relevant information from the available data. Here's a general outline of the feature engineering process:

1. Data Preparation:

Start by collecting and cleaning the IMDb dataset, removing any duplicates or missing values, and handling outliers appropriately.

2. Textual Features:

* Title: Extract information from the movie title, such as the number of words or characters, presence of certain keywords (e.g., "adventure,""romance").
* Plot Summary: Analyze the plot summary using techniques like tokenization, stop-word removal, and stemming/lemmatization. Generate features like bag-of-words, TF-IDF, or word embeddings to capture the essence of the movie's storyline.
* Genre: Convert categorical genres into binary features (one-hot encoding).

3. Numeric Features:

* Runtime: Transform the movie's runtime into a numerical feature by converting it into minutes or seconds.
* Production Year: Extract the production year from the release date and convert it into a numerical feature.
* Budget and Revenue: Utilize financial information such as budget and revenue (if available) as numeric features.

4. Cast and Crew:

* Actors and Directors: Consider the involvement of popular actors and directors as binary features, indicating whether they are associated with the movie.
* Awards and Nominations: Extract information about awards and nominations received by the cast and crew as numerical features.

5. External Data:

* Incorporate external data sources like sentiment analysis on reviews or social media mentions to capture the public sentiment around the movie.

6.User Interaction:

* Leverage existing user ratings and reviews to create features like average user rating, counts, or sentiment analysis of reviews.

7.Scaling and Normalization:

* Scale and normalize the features to ensure they're on a similar scale.

8.Feature Selection:

* Utilize techniques like correlation analysis, mutual information, or feature importance scores to select the most relevant features for the prediction model.

9.Model Building:

* Choose a machine learning algorithm suitable for regression tasks, such as linear regression, decision trees, random forests, or gradient boosting. Train and fine-tune the model using appropriate evaluation metrics (e.g., mean squared error) and cross-validation techniques.

Remember, feature engineering is an iterative process that requires experimentation with different techniques and domain knowledge to create the most informative features for the IMDb score prediction model.

**MODEL TRAINING :**

Model training is the process of teaching a machine learning model to learn patterns and relationships from the available data. Here's a general outline of the steps involved in model training:

1. Data Preparation: Prepare your dataset for training by splitting it into two sets: a training set and a validation/test set. The training set is used to teach the model, while the validation/test set is used to evaluate its performance.

2. Feature Selection/Extraction: Select the relevant features that will be used as inputs to the model. This may involve applying feature engineering techniques as discussed earlier to extract informative representations from the data.

3. Model Selection: Choose an appropriate machine learning algorithm or model architecture based on the nature of your problem. This could be a linear regression model, decision tree, support vector machine, neural network, or any other algorithm suitable for your specific task.

4. Model Initialization: Initialize the model with random or predefined parameters. This step is crucial for models that require initial weights or biases.

5. Training Loop: Iterate over the training data multiple times and update the model's parameters to minimize the difference between predicted outputs and true labels. This process is known as optimization or learning. The most commonly used optimization algorithms are gradient descent and its variations.

6. Loss Function: Define a loss function that quantifies the discrepancy between predicted outputs and true labels. The goal is to minimize this loss during the training process. The choice of loss function depends on the nature of the problem (e.g., mean squared error for regression, cross-entropy for classification).

7. Backpropagation: For models with trainable parameters (e.g., neural networks), perform backpropagation to compute the gradients of the loss with respect to the model's parameters. This allows for updating the parameters in the direction that reduces the loss.

8. Hyperparameter Tuning: Adjust the hyperparameters of the model, such as learning rate, regularization strength, batch size, and number of hidden layers, to optimize performance. This can be done through techniques like grid search, random search, or more advanced methods like Bayesian optimization.

9. Validation: Periodically evaluate the model's performance on the validation/test set during training to monitor its progress and detect any overfitting or underfitting issues. Adjust the model or hyperparameters accordingly.

10. Final Evaluation: Once the model training is complete, assess its performance on an independent test set to get an unbiased estimate of its generalization ability. This step helps determine how well the model will perform on unseen data.

11. Model Deployment: After the model has been trained and tested successfully, it can be deployed to make predictions on new and unseen data.

Remember that the model training process often requires iterations and experimentation to achieve the best performance.

**Procedure :**

To build an IMDb score prediction model, you can follow these steps for model training:

1. Data Preparation: Preprocess the IMDb dataset, handle missing values, convert categorical variables to numerical representations (one-hot encoding, label encoding), and split the data into training and test sets.

2. Feature Engineering: Create relevant features from the available data as discussed earlier. This step helps in capturing important information for predicting IMDb scores.

3. Model Selection: Choose a suitable machine learning algorithm for regression, such as linear regression, decision trees, random forests, support vector machines, or gradient boosting models. Consider the characteristics of your dataset and the assumptions of each algorithm to make an informed choice.

4. Model Training: Fit your chosen algorithm to the training data using the features and IMDb scores. This process involves estimating the model's parameters and finding the best fit to the training data.

5. Optimization and Hyperparameter Tuning: Optimize your model by fine-tuning its hyperparameters. This process involves adjusting parameters like learning rate, regularization strength, depth of trees, or the number of hidden layers to improve performance on the validation set. Techniques like grid search, random search, or Bayesian optimization can be used for hyperparameter tuning.

6. Model Evaluation: Evaluate your trained model using the test set. Calculate evaluation metrics appropriate for regression tasks, such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), or R-squared (coefficient of determination). Assessing these metrics helps understand how well your model performs in predicting IMDb scores.

7. Model Refinement: Analyze the model's performance and iteratively refine it by repeating steps 4 to 6. This involves adjusting the feature engineering process, trying different algorithms, or re-tuning hyperparameters to improve prediction accuracy.

8. Final Model Deployment: Once you're satisfied with the model's performance, deploy it to make predictions on new, unseen data. Be mindful of any necessary preprocessing steps needed for the input data to match the format used during training.

It's essential to note that the process described above is a general guideline, and the specific steps may vary based on the chosen algorithm, dataset characteristics, and domain knowledge. Experimentation, fine-tuning, and careful evaluation are the key to building an accurate IMDb score prediction model

**EVALUATION :**

Evaluation is a crucial step in the machine learning workflow to assess the performance and effectiveness of a trained model. It helps determine how well the model generalizes to new, unseen data and provides insights into its strengths and weaknesses. Here are some common evaluation techniques:

1. Train-Test Split: Divide the available dataset into two subsets, the training set and the test set. Train the model on the training set, and then evaluate its performance on the test set. This helps estimate how well the model will perform on unseen data.

2. Cross-Validation: Split the dataset into multiple subsets (folds). Train the model on a combination of these subsets and evaluate it on the remaining fold. This process is repeated several times, with each fold serving as a test set once, and then average the evaluation results across all folds. Cross-validation provides a more robust estimate of the model's performance compared to a single train-test split.

3. Evaluation Metrics: Choose appropriate evaluation metrics based on the problem type. For classification tasks, metrics like accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC) are commonly used. For regression tasks, metrics like mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), or R-squared (coefficient of determination) are typically used. The choice of metric depends on the specific problem requirements and can help measure the model's performance accurately.

4. Confusion Matrix: For classification tasks, a confusion matrix provides a detailed breakdown of model predictions and actual labels for each class. It helps understand the model's performance in terms of true positives, true negatives, false positives, and false negatives. From the confusion matrix, additional metrics such as precision, recall, and F1-score can be derived.

5. ROC Curve and Precision-Recall Curve: These curves provide a graphical representation of the performance of a binary classifier across different probability thresholds. The ROC curve shows the trade-off between sensitivity (recall) and specificity (1 - false positive rate), while the precision-recall curve illustrates the trade-off between precision and recall.

6. Bias and Fairness Analysis: Evaluate the model's fairness and bias by examining its performance across different subgroups (e.g., demographic groups). This helps identify potential disparities and biases in predictions, which is important for ensuring ethical and fair deployment of the model.

7. Error Analysis: Analyze specific instances where the model makes mistakes or produces unexpected results. This helps gain insights into areas where the model may be underperforming or where the training data may contain discrepancies or biases.

Remember that evaluation is an iterative process, and it may involve refining the model, adjusting hyperparameters, or exploring different feature engineering techniques based on the evaluation results. Additionally, it's important to carefully consider the limitations and scope of the evaluation metrics chosen to ensure they align with the specific problem and goals of the model.

**Evaluation in machine learning :**

In machine learning, evaluation refers to the process of assessing the performance and quality of a trained model. It involves measuring how well the model can generalize its predictions to unseen data. The evaluation process helps to determine if the model is effectively learning patterns and making accurate predictions.

**Here are some common evaluation techniques used in machine learning:**

1. Training and Test Sets: The most basic evaluation technique involves splitting the available dataset into two sets: a training set and a test set. The model is trained on the training set and then evaluated on the test set. The performance on the test set provides an estimate of how well the model generalizes to unseen data. This approach helps identify if the model is overfitting (performing well on training data but not on test data) or underfitting (performing poorly on both training and test data).

2. Cross-Validation: Cross-validation is a more advanced evaluation technique that helps overcome limitations in the train-test split method, especially when the dataset is small. It involves dividing the data into multiple subsets or "folds." The model is trained on a combination of these folds (usually k-1 folds) and evaluated on the remaining fold. This process is repeated k times, with each fold serving as the test set once. The performance results are then averaged to provide a more reliable estimate of the model's performance.

3. Evaluation Metrics: Different evaluation metrics are used based on the type of machine learning task. For classification tasks, metrics like accuracy, precision, recall, F1-score, and area under the ROC curve (AUC-ROC) are commonly used. For regression tasks, metrics like mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and R-squared are often used. The choice of evaluation metrics depends on the specific problem and the desired performance criteria.

4. Validation Set: In addition to the test set, a separate validation set can be used during the model training process to evaluate the model's performance at different stages. This helps in monitoring and fine-tuning the model to prevent overfitting.

5. Noisy Data Analysis: Evaluation also involves analyzing the model's performance on noisy or mislabeled data. This helps in understanding the model's robustness and potential limitations in real-world scenarios.

It's important to remember that the evaluation process should be performed on data that the model has not seen during training. This ensures a fair assessment of the model's performance on unseen data and its generalization ability. Additionally, evaluation results should be interpreted with domain knowledge and context in mind to make meaningful conclusions about the model's effectiveness and reliability.

**Procedure :**

To build an IMDb score prediction model using evaluation, you can follow these steps:

1. Data Preparation: Preprocess the IMDb dataset by handling missing values, converting categorical variables into numerical representations (such as one-hot encoding or label encoding), and splitting the data into training and test sets. Make sure to shuffle the data to remove any inherent ordering.

2. Feature Selection/Engineering: Select relevant features from the dataset that can help in predicting IMDb scores. This can include attributes such as movie genre, director, cast, runtime, release year, etc. Perform any necessary feature engineering steps like scaling or normalization to ensure consistency in the data.

3. Model Selection: Choose an appropriate model for IMDb score prediction. This can include regression algorithms like linear regression, decision trees, random forests, support vector machines (SVM), or gradient boosting models. Consider the characteristics of your dataset and the requirements of your problem when selecting a model.

4. Model Training: Train the selected model on the training dataset using the chosen features and IMDb scores as the target variable. Adjust the model's hyperparameters as necessary based on experimentation or cross-validation results. Fit the model to the data to estimate its parameters and optimize its performance.

5. Model Evaluation: Evaluate the trained model using the test dataset. Calculate evaluation metrics appropriate for regression tasks, such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), or R-squared (coefficient of determination). Assess how well the model predicts IMDb scores and compare its performance against baseline or benchmark models.

6. Hyperparameter Tuning: Fine-tune the model's hyperparameters to improve its performance. This can involve techniques like grid search, random search, or Bayesian optimization. Iterate on different combinations of hyperparameters to find the best configuration that maximizes the evaluation metrics.

7. Model Refinement: Analyze the model's performance and potential shortcomings. Assess any biases, errors, or areas of improvement. Iterate on the feature engineering process, try different algorithms, adjust hyperparameters, or consider additional data sources if necessary. This iterative refinement process help improve the model's prediction capabilities.

8. Final Model Deployment: Once you are satisfied with the model's performance, deploy it to make predictions on new, unseen data. Use the entire dataset (including the previously held-out test set, if desired) to train the final model before deployment, to utilize all available data for training.

Remember that evaluation is an ongoing process, and it's important to continuously monitor and assess the model's performance as new data becomes available. This helps ensure that the model remains accurate and effective over time.

**CONCLUSION :**

In conclusion, predicting IMDb scores can be a complex task that involves considering various factors such as user reviews, movie metadata, and external variables. Machine learning models, particularly regression and recommendation systems, can be valuable tools in this endeavor.

However, the accuracy of predictions depends on the quality and quantity of data, as well as the model's design and training.

It's essential to continually refine and validate these models to improve their predictive capabilities.

Keep in mind that IMDb scores are subjective and can be influenced by various factors, so while predictions can provide valuable insights, they may not always perfectly reflect a movie's true quality.