APPLIED DATA SCIENCE – GROUP 3

PROJECT 2 : PREDICTING IMDB SCORES

ABSTRACT

- PROBLEM DEFINITION
- DESIGN THINKING
- INNOVATION
- DATA PREPROCESSING
- Feature engineering
- Model training
- Evaluation

PREDICTING IMDB SCORES

DATASET : https://www.kaggle.com/datasets

Problem Statement and Design Thinking Problem

Problem Statement:

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.

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.

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

Based on the given dataset and the strategy planned, a machine learning model to predict IMDb scores of available movies by a certain criteria will be built such that it provides a better user expend improved.

About the Dataset

This dataset consists of all Netflix original films released as of June 1st, 2021. Additionally, it also includes all Netflix documentaries and specials. The data was web scraped from this Wikipedia page, which was then integrated with a dataset consisting of all of their corresponding IMDB scores.

Content

Included in the dataset is:

- Title of the film :

  It consist of the Title of the film

- Genre of the film

  It consist of the category of genre the film falls into

- Original premiere date

  It consist of the date, month and year the film was first premiered

- Runtime

  It consist of the runtime in minutes

- IMDB scores

  It consist of the IMDb rating of movies as of 06/01/21

- Languages

  It consist of the languages currently available as of 06/01/21

Implementation

Data preprocessing is a crucial step in any data analysis. This process involves cleaning and transforming the raw data to make it suitable for analysis. In this report, we will outline the key steps and techniques for data preprocessing in Python using various libraries, primarily Pandas and NumPy.

Code

1.Loading the required modules

```
[ ] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
```

2.Loading the Dataset

```
[ ] from google.colab import drive
    drive.mount('/content/drive')

     Mounted at /content/drive

[ ] df=pd.read_csv('/content/drive/MyDrive
```

3. Understanding the dataset

```
df.head()
```

	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

```
df.tail()
```

	Title	Genre	Premiere	Runtime	IMDB Score	Language
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

```
df.describe()
```

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

The number rows and columns

```
0s [7] df.shape

(584, 6)
```

```
df.info()
```

```
<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
```

4.Checking for null values

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

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

5.Checking for duplicate data

```
df.Title.duplicated().sum()
```

```
0
```

6.Performing analysis

The names of the columns present in the dataset

```
df.columns
```

```
Index(['Title', 'Genre', 'Premiere', 'Runtime', 'IMDB Score', 'Language'], dtype='object')
```

The number of movies in each genre

```
a=df.value_counts(['Genre'])
a
```

```
Genre
Documentary                   159
Drama                          77
Comedy                         49
Romantic comedy                39
Thriller                       33
                              ...
Coming-of-age comedy-drama      1
Comedy/Horror                   1
Comedy/Fantasy/Family           1
Comedy mystery                  1
Zombie/Heist                    1
Length: 115, dtype: int64
```

The number of films in each language

```
[14] df.value_counts(df['Language'])

Language
English                        401
Hindi                           33
Spanish                         31
French                          20
Italian                         14
Portuguese                      12
Indonesian                       9
Korean                           6
Japanese                         6
German                           5
Turkish                          5
English/Spanish                  5
Dutch                            3
Marathi                          3
Polish                           3
Filipino                         2
Thai                             2
English/Mandarin                 2
English/Japanese                 2
English/Hindi                    2
Tamil                            1
English/Akan                     1
Swedish                          1
Spanish/English                  1
Spanish/Catalan                  1
Thia/English                     1
Spanish/Basque                   1
English/Swedish                  1
Malay                            1
English/Arabic                   1
Norwegian                        1
English/Taiwanese/Mandarin       1
Khmer/English/French             1
English/Korean                   1
English/Russian                  1
Georgian                         1
English/Ukranian/Russian         1
Bengali                          1
dtype: int64
```

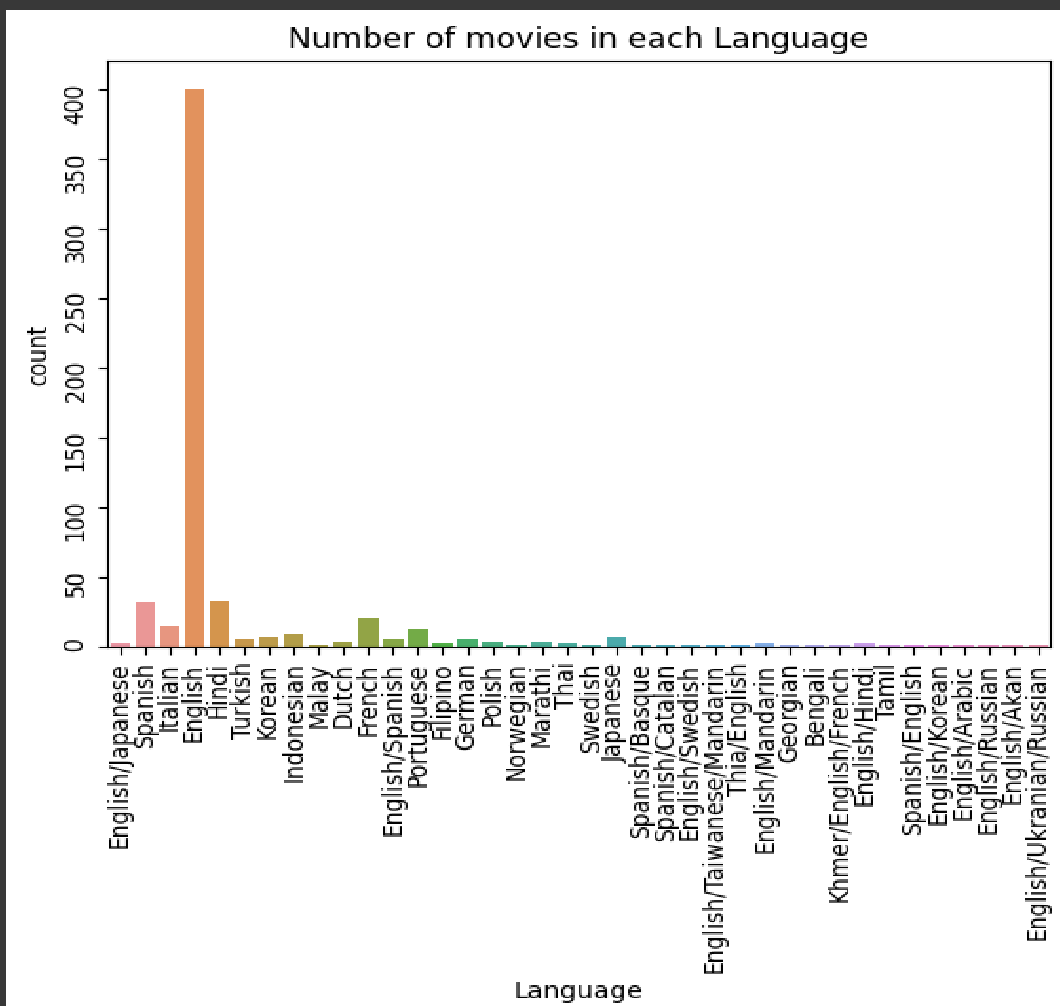
The highest number of films in a Language

```
df['Language'][[df.value_counts(df['Language']).max()]]
```

```
401    English
Name: Language, dtype: object
```

Countplot for number of films in each language

```
sns.countplot(x='Language',data=df)
plt.tick_params(rotation=90)
plt.title("Number of movies in each Language")
plt.show()
```

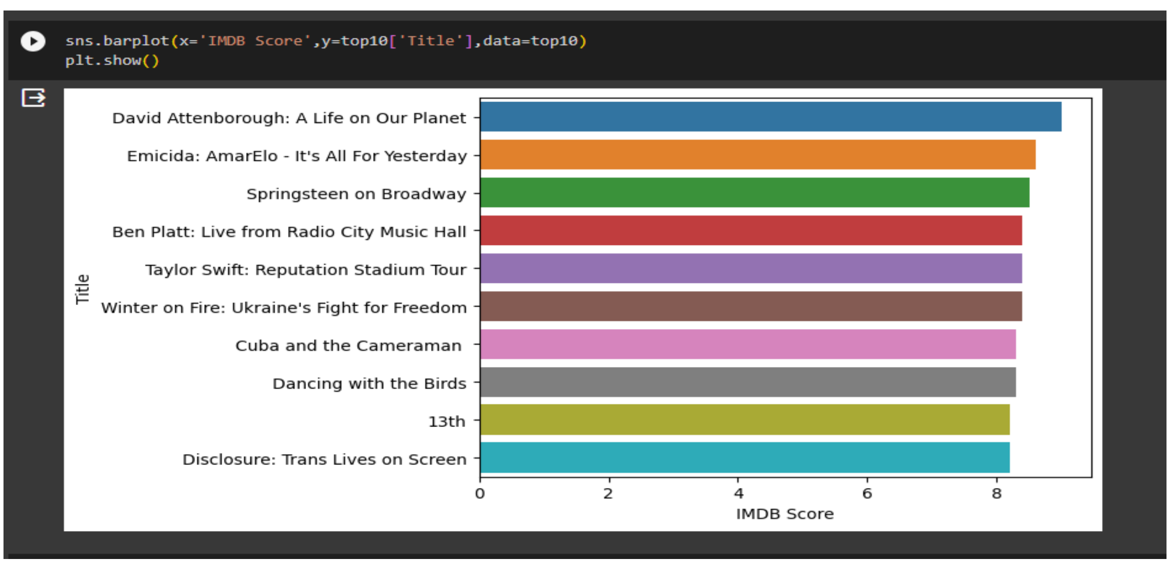


Grouping Top 10 films with highest rating

```python
[18] top10=df.nlargest(10,'IMDB Score')[['Title','IMDB Score']]
     top10
```

	Title	IMDB Score
583	David Attenborough: A Life on Our Planet	9.0
582	Emicida: AmarElo - It's All For Yesterday	8.6
581	Springsteen on Broadway	8.5
578	Ben Platt: Live from Radio City Music Hall	8.4
579	Taylor Swift: Reputation Stadium Tour	8.4
580	Winter on Fire: Ukraine's Fight for Freedom	8.4
576	Cuba and the Cameraman	8.3
577	Dancing with the Birds	8.3
571	13th	8.2
572	Disclosure: Trans Lives on Screen	8.2

Plot for the top 10 highest rated films with their titles



The rating of each film with their title in descending order

```
df.groupby('Title')['IMDB Score'].max().sort_values(ascending=False)
```

```
Title
David Attenborough: A Life on Our Planet      9.0
Emicida: AmarElo - It's All For Yesterday     8.6
Springsteen on Broadway                       8.5
Ben Platt: Live from Radio City Music Hall    8.4
Taylor Swift: Reputation Stadium Tour         8.4
                                             ...
Kaali Khuhi                                   3.4
The Open House                                3.2
The App                                       2.6
Dark Forces                                   2.6
Enter the Anime                               2.5
Name: IMDB Score, Length: 584, dtype: float64
```

The highest rating of the film

```
np.max(df['IMDB Score'])
```

```
9.0
```

The film which has the runtime equal to or more than 180 minutes

```
0s [21] df[df['Runtime']>=180]['Title']
```

```
561    The Irishman
Name: Title, dtype: object
```

Total number of genre in the dataset

```
[22] df.value_counts(df['Genre']).count()

     115
```

Total number languages in the dataset

```
[23] df.value_counts(df['Language']).count()

     38
```

The languages in the dataset

```
#total no of languages
df['Language'].unique()

array(['English/Japanese', 'Spanish', 'Italian', 'English', 'Hindi',
       'Turkish', 'Korean', 'Indonesian', 'Malay', 'Dutch', 'French',
       'English/Spanish', 'Portuguese', 'Filipino', 'German', 'Polish',
       'Norwegian', 'Marathi', 'Thai', 'Swedish', 'Japanese',
       'Spanish/Basque', 'Spanish/Catalan', 'English/Swedish',
       'English/Taiwanese/Mandarin', 'Thia/English', 'English/Mandarin',
       'Georgian', 'Bengali', 'Khmer/English/French', 'English/Hindi',
       'Tamil', 'Spanish/English', 'English/Korean', 'English/Arabic',
       'English/Russian', 'English/Akan', 'English/Ukranian/Russian'],
      dtype=object)
```

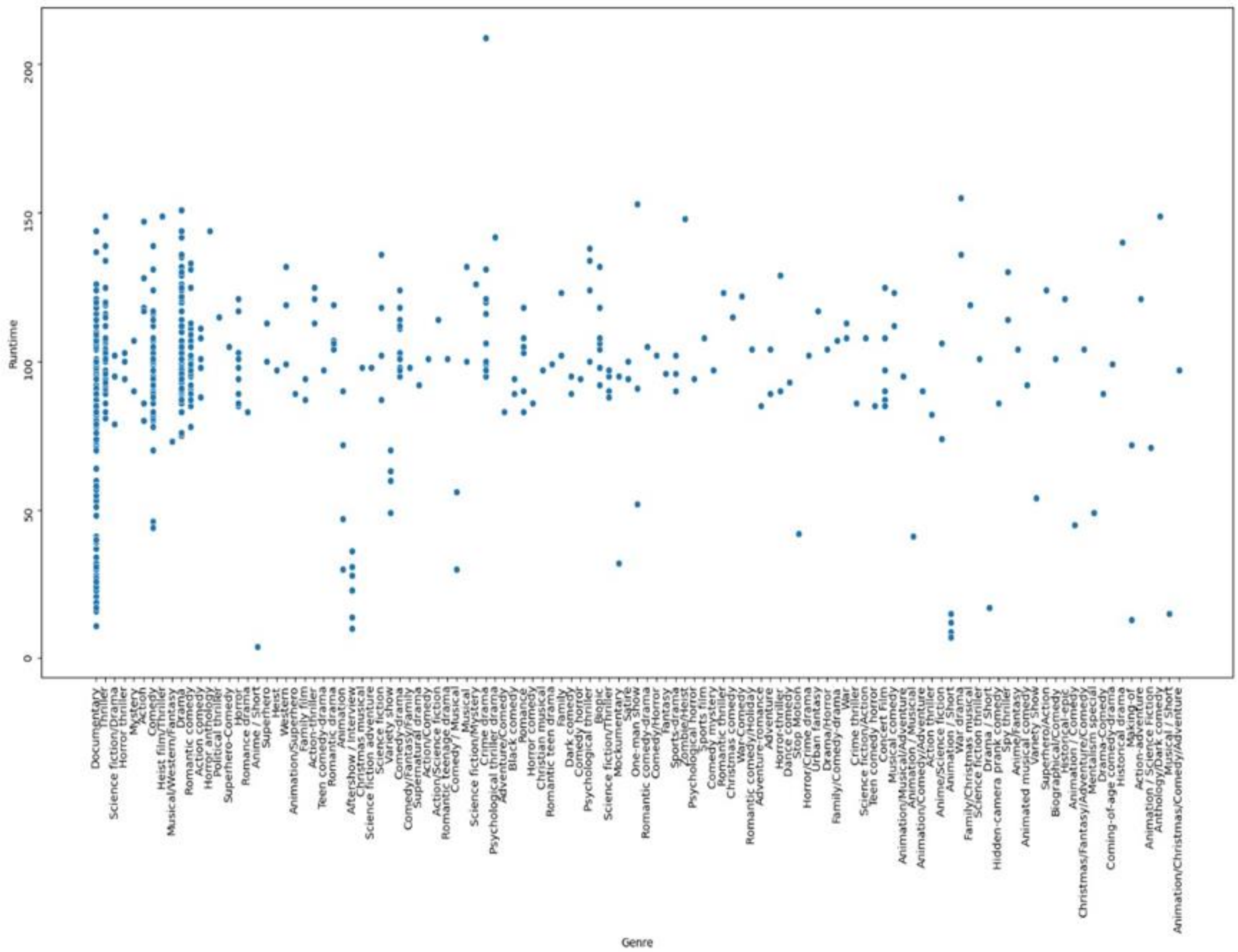
Which language has the highest number of films in a genre

```
df.groupby('Genre')['Language'].max()
```

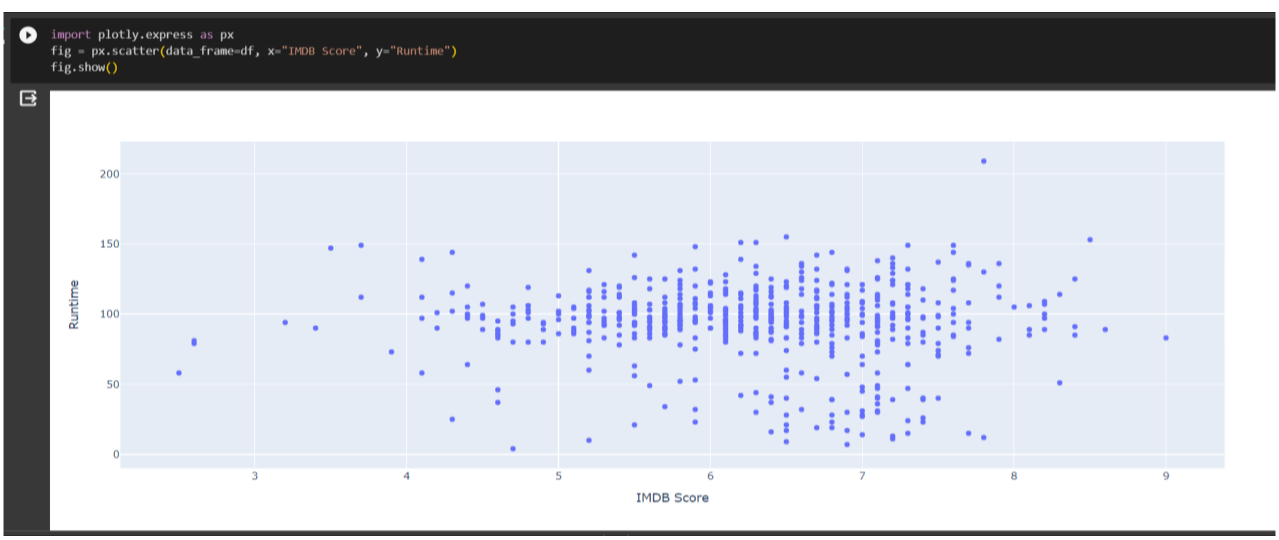
```
Genre
Action                         Hindi
Action comedy                  Malay
Action thriller              English
Action-adventure      English/Korean
Action-thriller           Indonesian
                             ...
War                          English
War drama               English/Akan
War-Comedy                   English
Western                   Portuguese
Zombie/Heist                 English
Name: Language, Length: 115, dtype: object
```

A plot to analyze the relationship between Runtime and Genre

```
plt.figure(figsize=(20,10))
sns.scatterplot(x='Genre',y='Runtime',data=df)
plt.tick_params(rotation=90)
plt.show()
```



A plot to analyze the relationship between IMDb Score and Runtime



Data preprocessing ensures that the data is clean, consistent, and suitable for the tasks at hand. By following the steps outlined in this report and using the appropriate techniques and libraries, the given dataset has been pre-processed in python for accurate and meaningful analysis.

Feature engineering

Feature engineering is the process of transforming raw data into features that are suitable for machine learning models. In other words, it is the process of selecting, extracting, and transforming the most relevant features from the available data to build more accurate and efficient machine learning models.

We have implemented certain Feature extraction process in the given dataset like extracting the columns ‘Date’ , ‘Year’ and ‘Month’ from the Premiere column that was already existing in the given dataset 2 3 The Feature extraction process in the given dataset like extracting the columns ‘Date’ , ‘Year’ and ‘Month’ from the Premiere column that was already existing in the given dataset

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

```
0      2019-08-05
1      2020-08-21
2      2019-12-26
3      2018-01-19
4      2020-10-30
          ...
579    2018-12-31
580    2015-10-09
581    2018-12-16
582    2020-12-08
583    2020-10-04
Name: Date, Length: 584, dtype: datetime64[ns]
```

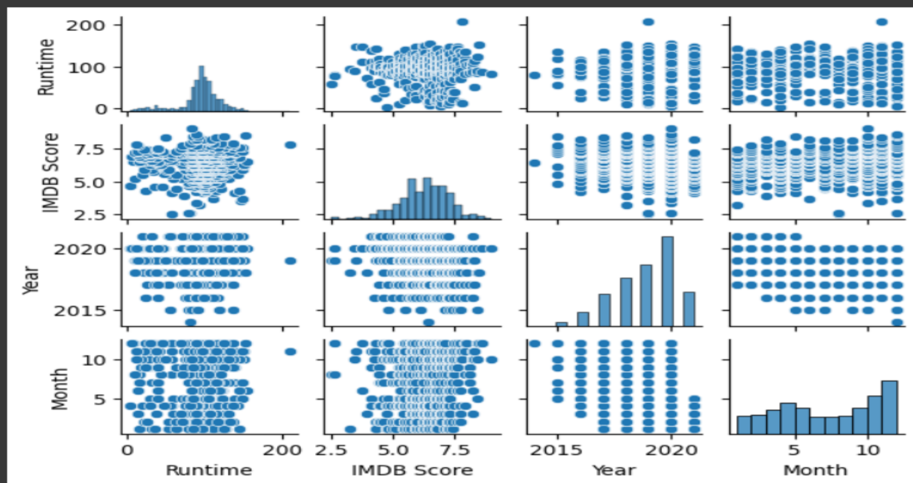
```
df["Year"] = df["Date"].dt.year
df["Month"] = df["Date"].dt.month
print(df.head())
```

```
             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       Date  Year  Month
0         2.5  English/Japanese 2019-08-05  2019      8
1         2.6           Spanish 2020-08-21  2020      8
2         2.6           Italian 2019-12-26  2019     12
3         3.2           English 2018-01-19  2018      1
4         3.4             Hindi 2020-10-30  2020     10
```

Pairplot

```
sns.pairplot(df, height=1.2, aspect=1.25)
plt.savefig('pariplot.png');
```




```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 584 entries, 0 to 583
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Title       584 non-null    int64
 1   Genre       584 non-null    int64
 2   Premiere    584 non-null    int64
 3   Runtime     584 non-null    int64
 4   IMDB Score  584 non-null    float64
 5   Language    584 non-null    int64
 6   Date        584 non-null    datetime64[ns]
 7   Year        584 non-null    int64
 8   Month       584 non-null    int64
dtypes: datetime64[ns](1), float64(1), int64(7)
memory usage: 41.2 KB
```

The *Feature transformation* was implemented by the use of Label encoder to convert the categorical values into numeric values.

```
[382] from sklearn.preprocessing import LabelEncoder
      cols=['Title','Genre','Runtime','Premiere','Language','Date','Year','Month']
      df[cols]=df[cols].apply(LabelEncoder().fit_transform)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 584 entries, 0 to 583
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Title       584 non-null    int64
 1   Genre       584 non-null    int64
 2   Premiere    584 non-null    int64
 3   Runtime     584 non-null    int64
 4   IMDB Score  584 non-null    float64
 5   Language    584 non-null    int64
 6   Date        584 non-null    int64
 7   Year        584 non-null    int64
 8   Month       584 non-null    int64
dtypes: float64(1), int64(8)
memory usage: 41.2 KB
```

The *Feature selection* is used to choose the most relevant features to include in the model while eliminating irrelevant or redundant ones. Therefore, we have excluded ‘Premiere’ and ‘Month’.

```
[383] x=df.drop(['Premiere','Month','IMDB Score'], axis=1)
      y=df['IMDB Score']

      print(x)
      print(y)

          Title  Genre  Runtime  Language  Date  Year
0           147     45       42         6   182     5
1           120    106       56        29   281     6
2           433     93       54        20   219     5
3           500     63       69         2    85     4
4           243     73       65        18   312     6
..          ...    ...      ...       ...   ...   ...
579         425     40      100         2   140     4
580         575     45       66        13     6     1
581         410     74      121         2   138     4
582         145     45       64        28   331     6
583         121     45       58         2   299     6

[584 rows x 6 columns]
0       2.5
1       2.6
2       2.6
3       3.2
4       3.4
       ...
579     8.4
```

Splitting of data - The dataset is divided into training and test sets. The training set is used to train the model and the test set is used to evaluate the model's generalization performance

```
[387] from sklearn.model_selection import train_test_split
      x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=42)

[313] print(x_train.shape)
      print(y_train.shape)
      print(x_test.shape)
      print(y_test.shape)

      (467, 6)
      (467,)
      (117, 6)
      (117,)
```

Model Building

By choosing a machine learning algorithm a model is build for Predicting the IMDb Scores for the given dataset

Why Random Forest Regressor?

Since we need to build a predictive model for continuous data , we have to go for a Regressor model. Random Forest Regressor is an ensemble model and hence the accuracy will be better and since it can work with non linear data and reduces outliers and overfitting we have chosen Random forest Regressor. And among all the models we tried Random Forest Regressor gave a better predictive model with less error value.

Model Training

Train the model on the training dataset. The model learns the patterns and relationships in the data during this phase.

```
[451] from sklearn.ensemble import RandomForestRegressor
      from sklearn.metrics import mean_squared_error
```

```
[452] rf = RandomForestRegressor(n_estimators=15,max_depth=15,random_state=0,criterion='squared_error')
      rf.fit(x_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(max_depth=15, n_estimators=15, random_state=0)
```

Model Validation

Validate the model on the test set to assess its generalization performance. This step ensures that the model can make accurate predictions on unseen data.

```
y_pred=rf.predict(x_test)
rmse = float(format(np.sqrt(mean_squared_error(y_test, y_pred)), '.3f'))
print("\nRMSE: ", rmse)
```

```
RMSE:  0.845
```

Model Evaluation

Evaluation metrics for regression models are used to assess the performance of models that predict continuous numeric values. These metrics help to understand how well the regression model is making predictions and are crucial for model selection, hyperparameter tuning, and comparing different regression algorithms. Here are some common evaluation metrics for regression models:

1. Mean Absolute Error (MAE):

   - Measures the average absolute difference between actual and predicted values.

   - Calculation: (1/n) Σ |actual - predicted|

2. Mean Squared Error (MSE):

   - Measures the average of the squared differences between actual and predicted values.

   - Calculation: (1/n) Σ (actual - predicted)^2

3. Root Mean Squared Error (RMSE):

   - It is the square root of the mean squared error.

   - Calculation: √MSE

4. R-squared (R2) Score:

   - Measures the proportion of the variance in the dependent variable that is predictable from the independent variables.

   - Calculation: 1 - (MSE(model) / MSE(mean))

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

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}")
```

```
Mean Absolute Error (MAE): 0.6575585906821325
Mean Squared Error (MSE): 0.7138135847776027
Root Mean Squared Error (RMSE): 0.8448748929738666
R-squared (R2): 0.31227950251748593
```

Visualization of Random Forest Regression

```
df_range=df.index[-len(y_test):]

plt.figure(figsize=(12,6))
plt.plot(df_range,y_test,label='Actual Rating ',linewidth=2)
plt.plot(df_range,y_pred,label='Predicted Rating ',linestyle='--',linewidth=2)
plt.title("Actual vs. Predicted IMDb scores")
plt.legend()
plt.xlabel('Movie')
plt.ylabel('IMDb Score')
plt.grid()
plt.show()
```