**PHASE 3**

**DATA DEVELOPMENT**

To create a dataset for predicting IMDb scores, you'll need to gather information on various aspects of films. Some common sources for this data include IMDb itself, movie databases like The Movie Database (TMDb) or IMDb API, web scraping, and potentially crowdsourcing. The data you collect should include the following fields:

* **Title of the film:** The name of the movie.
* **Genre of the film:** The genre or genres that best describe the film (e.g., action, drama, comedy).
* **Original premiere date:** The date when the movie was first released in theaters or on a streaming platform.
* **Runtime in minutes:** The duration of the movie in minutes.
* **IMDb scores:** The IMDb ratings for the movie, which typically range from 1 to 10.
* **Languages currently available:** The languages in which the movie is available for viewing.

With reference to the link below :

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

**DATA PREPROCESSING**

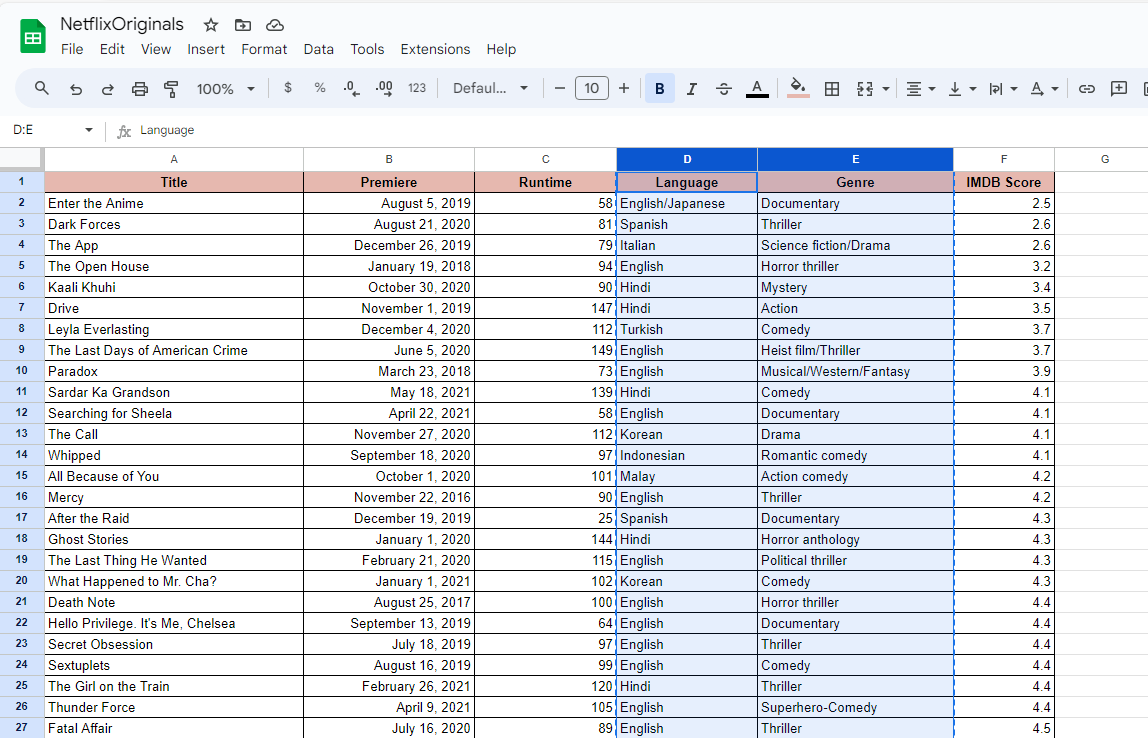
Once you have collected the data, you'll need to preprocess it to ensure it's clean and ready for analysis. This involves:

**Handling missing data:** You may need to deal with missing values in any of the fields, possibly by imputing them or removing the corresponding entries.

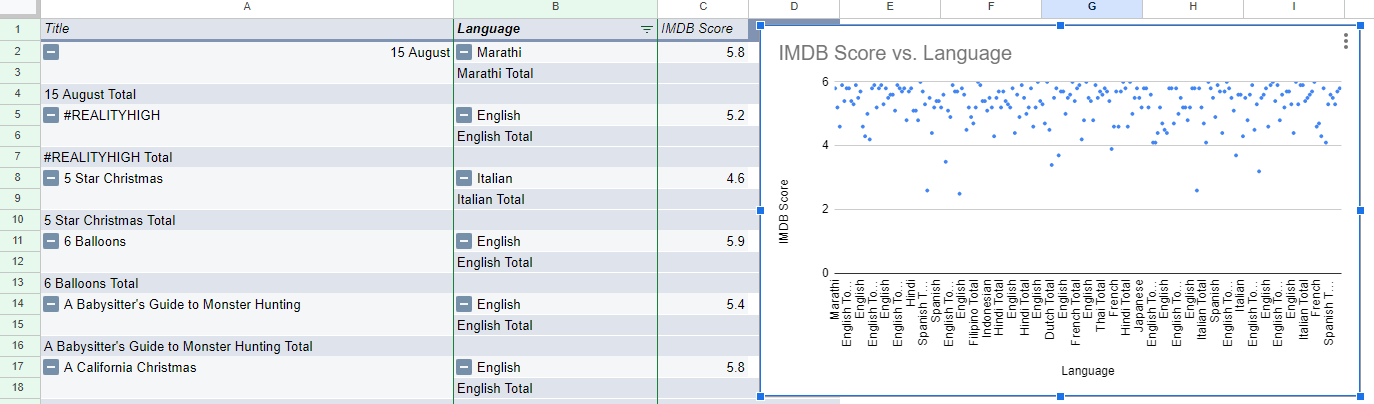
**Data encoding:** Categorical data like genre and languages need to be encoded into a numerical format. This can be done using techniques like one-hot encoding or label encoding.

**Feature scaling**: You might need to scale numeric features like runtime to have similar ranges, especially if you plan to use algorithms that are sensitive to feature scales, such as gradient descent-based methods.

**Outlier detection**: Identify and handle outliers in IMDb scores or runtime that could skew the predictions.

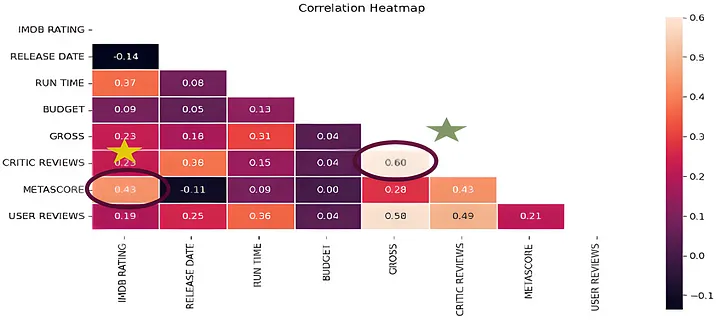


**DATA ANALYSIS**



An analysis is done on the given Kaggle Dataset of the project Predicting IMDb scores here the relation between the **Language and the IMDb rating** of the respective dataset is exaggerated in depth with pivot option. It allows you to summarise, aggregate, and rearrange data in a way that makes it easier to understand and draw insights from your dataset.

**FEATURE ENGINEERING**



Some new variables can be created in the dataset in order to improve model accuracy. In this perspective, we can see importance of domain knowledge. If we do not have any information about our study area, we can find important parameters with EDA or some investigations related to this area.In this study, we prefer to use “correlation heatmap” analysis to find effective parameter.

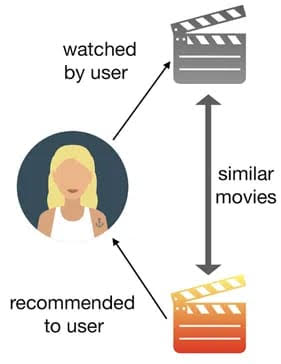
**CONTENT BASED FILTERING**

Content-Based filtering doesn’t involve other users, but based on our preference, the algorithm will simply pick items with similar content to generate recommendations for us.

This algorithm offers less diverse recommendations, but it will work regardless of the fact of whether user rates things or not.

For example, there can be a situation where a user potentially likes sci-fi action movies, but he might never know unless he decides to give it a try autonomously. So, what this filter does, it will keep on recommending superhero movies or similar. We can calculate this similarity on many attributes, but in our case, we build this recommender system based on the following key features:

* TITLE
* PREMIERE
* RUNTIME
* GENRE
* IMDb SCORES
* LANGUAGE



Now, moving forward to the term that we keep on mentioning, similarity, and what it means in our context. It might not seem like something we could quantify, but it can be measured. Before we proceed to the methodologies, we used to build our content-based recommender system, let’s briefly discuss the concept of cosine-similarity, one of the metrics that can be used to calculate the similarity between users or contents.

**DATA DEVELOPMENT WITH PYTHON**

#Load csv file

import numpy as np *# linear algebra*

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

import os

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

    for filename **in** filenames:

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

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)

out [5]:

<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 | Runtime | 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()

out [11]:

<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

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)

Out [14]:

