PROJECT TITLE : PREDICTING IMDb MOVIE SCORES

PHASE 5

PROBLEM DEFINITION AND DESIGN THINKING

PROBLEM DEFINITION:

The "Predicting IMDb Movie Scores" project seeks to harness the power of machine learning to forecast IMDb ratings for movies. IMDb scores play a pivotal role in guiding viewers' movie choices and are of paramount importance to the film industry. This project endeavours to construct an accurate predictive model that can provide insights to filmmakers, distributors, and movie enthusiasts. To achieve this, we collect and preprocess extensive movie-related data, explore patterns through data analysis, employ machine learning algorithms, and ultimately deliver a model capable of estimating IMDb scores. The findings and methodologies unveiled in this project hold the potential to enhance the understanding of the factors that influence movie ratings and revolutionize the movie industry's decision-making processes.

The goal of this project is to develop a predictive model that can accurately estimate IMDb scores for movies. IMDb scores are a widely recognized metric for evaluating the quality of movies, and accurate prediction can provide valuable insights to movie studios, distributors, and viewers alike. The project aims to leverage machine learning techniques to build a robust prediction model.

With reference to the link https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores

DESIGN THINKING:

1. DATA SOURCE:

Predicting IMDb scores is a complex task that typically involves a variety of features and data sources. Here are some common data sources and features that can be used to predict IMDb scores:

- Movie Metadata
- Cast and Crew Data
- User Reviews
- Box Office Data
- Awards and Nominations
- Movie Trailer Views
- Social Media Mentions

- Rotten Tomatoes or Metacritic Scores
- Historical IMDb Data
- Budget and Production Costs

To create a predictive model for IMDb scores, you would typically gather a dataset that includes these features and their corresponding IMDb scores. Machine learning techniques, such as regression or ensemble models, can then be applied to train the model for prediction. Keep in mind that feature engineering and data preprocessing are crucial steps in building an accurate prediction model. Additionally, regularly updated data is important for maintaining the model's accuracy over time.

2. DATA PROCESSING

Data cleaning is a crucial step in the data preprocessing pipeline in data science. Data preprocessing means to prepare the data for classification. Data is processed according to the requirements of classification. It involves identifying and addressing issues in the dataset to ensure that the data is accurate, consistent, and ready for analysis or model training. Here, for preprocessing the data, instances with missing attributes are removed Here's a detailed explanation of the data cleaning process

- Handling Missing Data
- Handling Duplicate Data
- Correcting Inaccurate Data
- Standardizing Data
- Handling Outliers
- Dealing with Inconsistent Data Types
- Addressing Data Integrity Issues
- Handling Text Data
- Documenting Changes
- Data Imputation for Time Series Data
- Data Scaling and Normalization

Data cleaning is an iterative process, and it's essential to thoroughly understand the data and the domain to make informed decisions during each step. Careful data cleaning ensures that the subsequent analysis or modelling steps are based on high-quality data, leading to more reliable and meaningful results.

3. FEATURE ENGINEERING

Feature engineering is critical when predicting IMDb scores for movies. Here are some specific feature engineering techniques you can apply:

- Genre Features
- Director and Actor Influence
- Release Date Features
- Budget and Box Office
- Runtime
- Word Analysis from Plot and Reviews
- Awards and Nominations
- User Ratings from Other Platforms
- Movie Sequels and Franchises
- Cultural and Historical Events
- Studio Influence
- Directorial Debut
- MPAA Rating
- Movie Budget to Revenue Ration

Remember to perform thorough data analysis and feature selection to ensure that the engineered features are relevant and do not introduce noise into the model. Experiment with different combinations of features and machine learning algorithms to find the best approach for predicting IMDb scores accurately.

4. MODEL SELECTION

Selecting the right model for predicting IMDb scores or analyzing IMDb data involves understanding your specific problem, the data you have, and your objectives. IMDb scores typically involve regression tasks, where you predict a continuous target variable (movie ratings). Here's a model selection process tailored to IMDb score prediction:

Consider a variety of regression models suitable for IMDb score prediction:

- Linear Regression: A simple model assuming a linear relationship between features and IMDb scores.
- Tree-Based Models: Decision Trees, Random Forest, or Gradient Boosting models can capture non-linear relationships.
- Neural Networks: Deep learning models, like feedforward neural networks or recurrent neural networks (RNNs), can handle complex patterns.
- Support Vector Machines (SVM): SVMs can work well for regression tasks when properly tuned.
- K-Nearest Neighbors (KNN): Suitable for locally patterned data.

5. MODEL TRAINING

Model training for predicting IMDb scores typically involves a regression task, as you're trying to predict a continuous target variable (movie ratings).

- Use the training data to train your selected regression model. The model will learn to make predictions based on the features and target IMDb scores.
- Tune the hyperparameters of your model using techniques like grid search or random search to optimize its performance
- Assess your model's performance on the validation set using the chosen evaluation metric(s). This helps you fine-tune hyperparameters and check for overfitting.
- After tuning your model and validating its performance, evaluate it on the test set to assess how well it generalizes to unseen data. Use the same evaluation metrics as in the validation phase.

6. EVALUATION

Evaluating the performance of a predictive model for IMDb scores is crucial to understand how well the model is performing and to make any necessary improvements. Common evaluation metrics for regression tasks like predicting IMDb scores include:

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE
- Root Mean Squared Error (RMSE
- R-squared (R^2) or Coefficient of Determination
- Adjusted R-squared
- Mean Absolute Percentage Error (MAPE)

It's essential to choose evaluation metrics that align with your specific goals and the nature of your IMDb score prediction task. For example, if your goal is to recommend movies with high IMDb scores to users, you might prioritize metrics that emphasize accuracy, such as MAE or RMSE. Additionally, it's a good practice to compare your model's performance to a baseline model or a simple heuristic to gauge its effectiveness.Remember that no single metric provides a complete picture of model performance, so it's valuable to examine multiple metrics and consider the practical implications of your model's performance in real-world scenarios.

INNOVATION

Predicting IMDb scores and ratings is a common problem in data science, and there have been several innovations and techniques developed to address this challenge. Here are some key innovations and methods that we have implemented to predict IMDb scores and ratings in our model are :

i. Graph Data
ii. Natural Language Processing (NPL)
iii. Temporal Analysis

1. GRAPH DATA

Graph-based approaches for predicting IMDb scores involve using techniques that leverage the relationships between various entities in a movie dataset. Here's a high-level overview of how you can use graph data theory for this purpose:

i. Data Representation:

Represent your movie dataset as a graph where nodes represent entities (e.g., movies, actors, directors) and edges represent relationships between them (e.g., actor acted in a movie, director directed a movie).

Assign attributes to nodes and edges, such as movie features (budget, genre), actor information (age, awards), and director information (experience, previous movie ratings).

ii. Graph Algorithms:

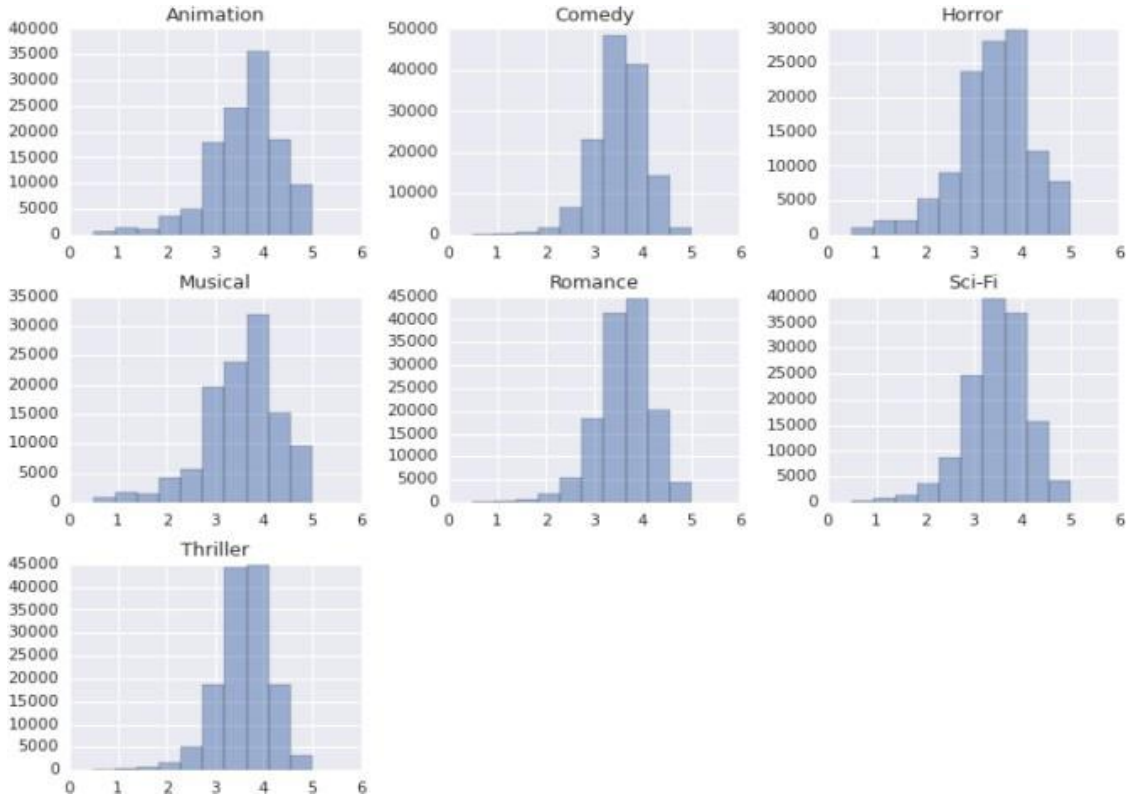
Apply graph algorithms to uncover hidden patterns and relationships. For example, use community detection algorithms to identify groups of movies with similar characteristics.

Utilize link prediction algorithms to predict potential collaborations between directors and actors that might result in high IMDb scores.

iii. Visualization:

Visualize the graph structure and important nodes using tools like network visualization libraries (e.g., NetworkX, Gephi) to gain insights into the relationships that affect IMDb scores.

SAMPLE DATA: Based on Movie Genres



2. NATURAL LANGUAGE PROCESSING

Natural Language Processing (NLP) can be used in predicting IMDb scores in several ways:

i. Sentiment Analysis:

NLP can be used to analyze the sentiment of movie reviews. Positive sentiment in reviews may correlate with higher IMDb scores, while negative sentiment may correlate with lower scores. Machine learning models can be trained to predict IMDb scores based on the sentiment of the reviews.

ii. Text Features:

NLP can extract features from movie descriptions, reviews, and other textual data. These features can include word frequencies, TF-IDF scores, or word embeddings. Machine learning models can then use these features to predict IMDb scores.

iii. **Topic Modeling:**

NLP techniques like topic modeling can identify the main topics or themes in movie reviews. If certain topics are associated with higher or lower IMDb scores, this information can be used in prediction models.

iv. **User Reviews:**

Analysing user reviews for specific movies and aggregating user sentiment or opinion can be a valuable source of information for predicting IMDb scores.

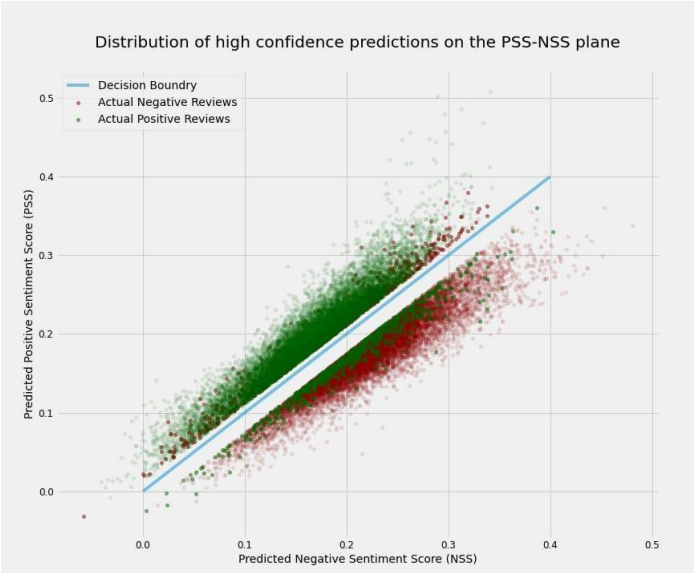
v. **Text Regression Models:**

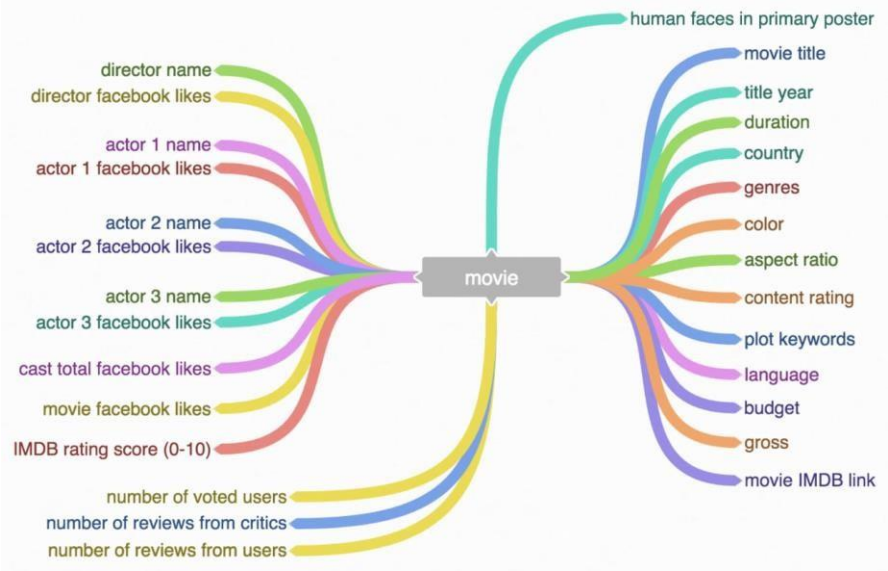
Regression models, such as linear regression or gradient boosting, can be used to predict IMDb scores based on textual features and sentiment scores extracted from reviews and descriptions.

vi. **Natural Language Understanding:**

Advanced NLP models like BERT or GPT can be used to understand the context and nuances of movie reviews more comprehensively, potentially improving prediction accuracy.

It's important to note that predicting IMDb scores based solely on NLP features may not be highly accurate, as IMDb scores are influenced by various factors beyond text data, such as director, actors, and marketing. Combining NLP with other data sources and features can improve the accuracy of such predictions. Additionally, it's crucial to have a quality dataset and appropriate evaluation metrics to assess the performance of your prediction model.





3. TEMPORAL ANALYSIS

Predicting IMDb scores through temporal analysis involves examining how a movie's attributes and factors change over time and how they correlate with IMDb ratings. Here's asimplified approach:

i. Data Collection:

- Gather a dataset containing information about movies, including
- release date, genre, director, cast, budget, runtime, and initial IMDbscores.

ii. Data Preprocessing:

- Clean the dataset by handling missing values and outliers.
- Convert categorical variables into numerical format using techniqueslike one-hot encoding.
- Calculate additional features like the age of the movie at the time ofrelease.

iii. Temporal Analysis:

- Group movies by release year, month, or season to capture temporaltrends.
- Compute statistics and trends over time for each group, such as average IMDb scores, budget distribution, genre popularity, etc.

iv. **Feature Engineering:**

Create new features that capture temporal dynamics, like the numberof movies released in a particular month/year, average IMDb scores.

v. **Machine Learning Model:**

Select a suitable regression algorithm (e.g., linear regression, randomforest, or neural networks).Train the model on the historical data, using features from steps 3 and 4 to predict IMDb scores.

vi. **Model Evaluation:**

- Use metrics like Mean Absolute Error (MAE) or Root Mean Square Error (RMSE) to evaluate the model's performance on a test dataset.
- Employ time-based cross-validation to assess model stability over different time periods.

vii. **Feature Importance:**

Analyze feature importance to understand which temporal factorshave the most significant impact on IMDb scores.

viii. **Predictions**:

Apply the trained model to new movies to predict their IMDb scoresbased on their attributes and the current temporal context.

ix. **Continuous Monitoring and Updating**:

Periodically retrain the model with new data to account for changingtrends and preferences in movie ratings.

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

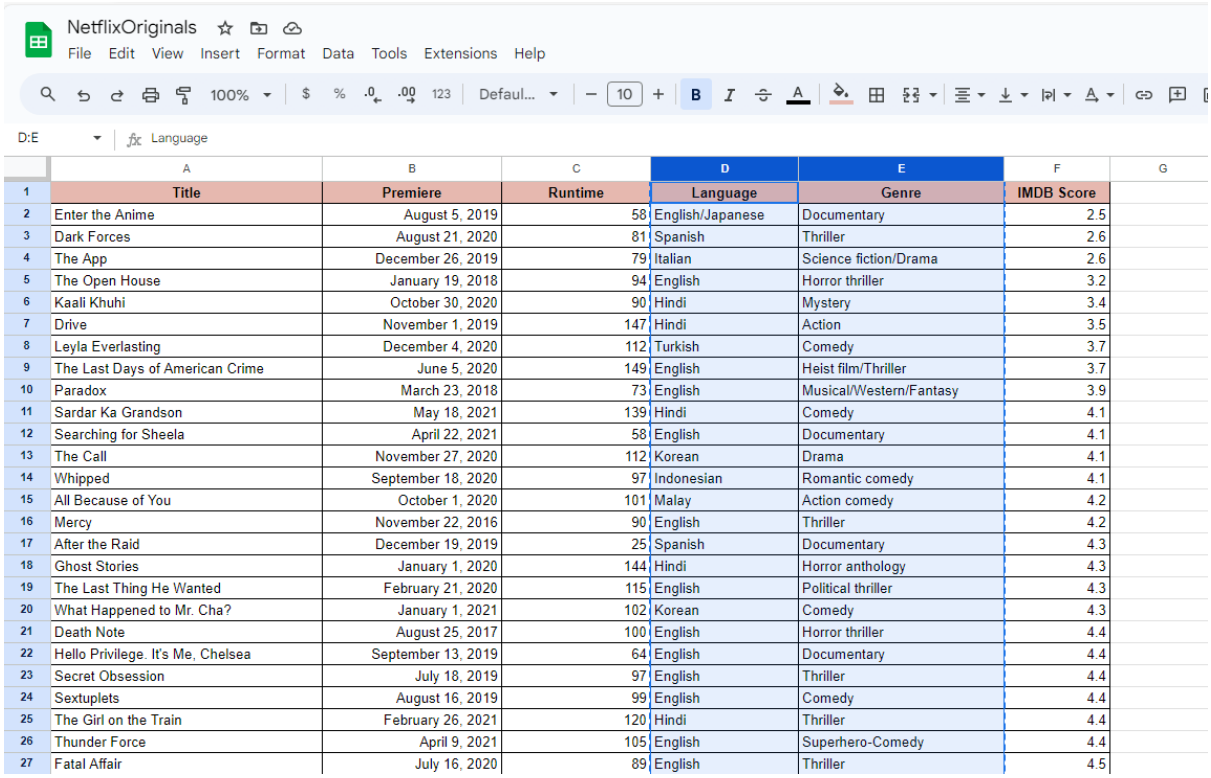
Once you have collected the data, you'll need to preprocess it to ensure it's clean and ready for analysis. This involves:

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.



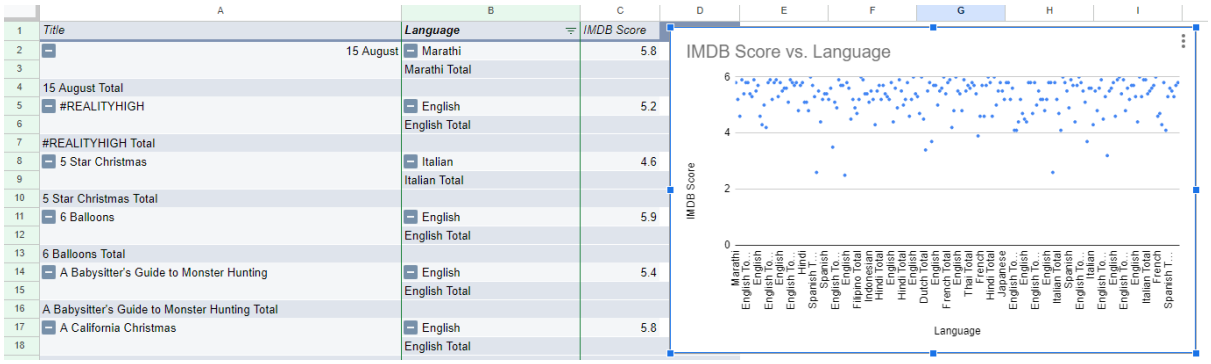
NetflixOriginals

File Edit View Insert Format Data Tools Extensions Help

D:E | Language

	A	B	C	D	E	F	G
1	Title	Premiere	Runtime	Language	Genre	IMDB Score	
2	Enter the Anime	August 5, 2019	58	English/Japanese	Documentary	2.5	
3	Dark Forces	August 21, 2020	81	Spanish	Thriller	2.6	
4	The App	December 26, 2019	79	Italian	Science fiction/Drama	2.6	
5	The Open House	January 19, 2018	94	English	Horror thriller	3.2	
6	Kaali Khuhi	October 30, 2020	90	Hindi	Mystery	3.4	
7	Drive	November 1, 2019	147	Hindi	Action	3.5	
8	Leyla Everlasting	December 4, 2020	112	Turkish	Comedy	3.7	
9	The Last Days of American Crime	June 5, 2020	149	English	Heist film/Thriller	3.7	
10	Paradox	March 23, 2018	73	English	Musical/Western/Fantasy	3.9	
11	Sardar Ka Grandson	May 18, 2021	139	Hindi	Comedy	4.1	
12	Searching for Sheela	April 22, 2021	58	English	Documentary	4.1	
13	The Call	November 27, 2020	112	Korean	Drama	4.1	
14	Whipped	September 18, 2020	97	Indonesian	Romantic comedy	4.1	
15	All Because of You	October 1, 2020	101	Malay	Action comedy	4.2	
16	Mercy	November 22, 2016	90	English	Thriller	4.2	
17	After the Raid	December 19, 2019	25	Spanish	Documentary	4.3	
18	Ghost Stories	January 1, 2020	144	Hindi	Horror anthology	4.3	
19	The Last Thing He Wanted	February 21, 2020	115	English	Political thriller	4.3	
20	What Happened to Mr. Cha?	January 1, 2021	102	Korean	Comedy	4.3	
21	Death Note	August 25, 2017	100	English	Horror thriller	4.4	
22	Hello Privilege. It's Me, Chelsea	September 13, 2019	64	English	Documentary	4.4	
23	Secret Obsession	July 18, 2019	97	English	Thriller	4.4	
24	Sextuplets	August 16, 2019	99	English	Comedy	4.4	
25	The Girl on the Train	February 26, 2021	120	Hindi	Thriller	4.4	
26	Thunder Force	April 9, 2021	105	English	Superhero-Comedy	4.4	
27	Fatal Affair	July 16, 2020	89	English	Thriller	4.5	

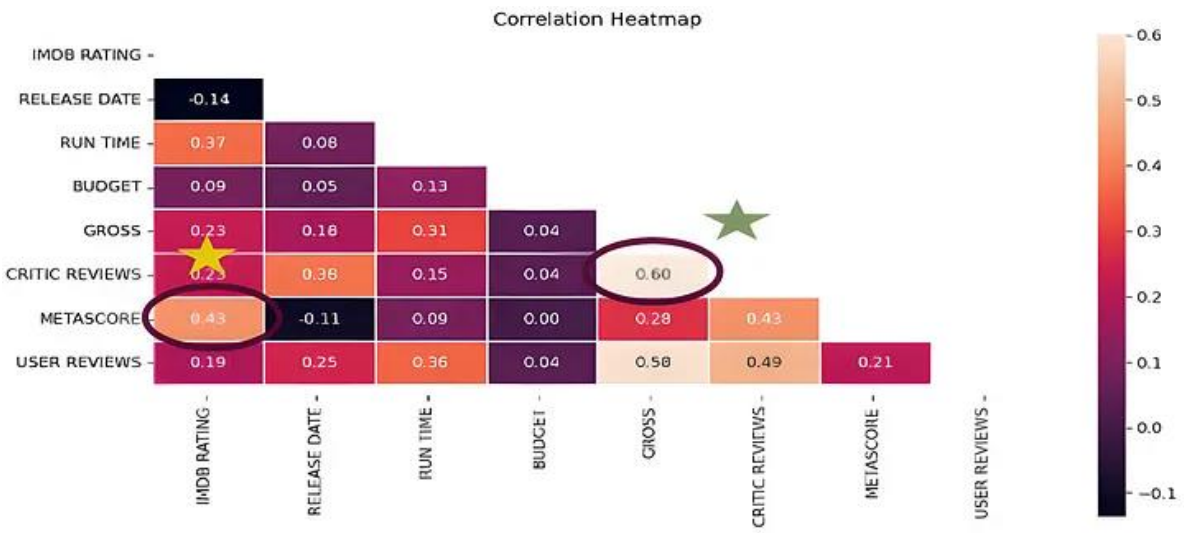
DATA ANALYSIS



	A	B	C
1	Title	Language	IMDB Score
2	15 August	Marathi	5.8
3		Marathi Total	
4	15 August Total		
5	#REALITYHIGH	English	5.2
6		English Total	
7	#REALITYHIGH Total		
8	5 Star Christmas	Italian	4.6
9		Italian Total	
10	5 Star Christmas Total		
11	6 Balloons	English	5.9
12		English Total	
13	6 Balloons Total		
14	A Babysitter's Guide to Monster Hunting	English	5.4
15		English Total	
16	A Babysitter's Guide to Monster Hunting Total		
17	A California Christmas	English	5.8
18		English Total	

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.

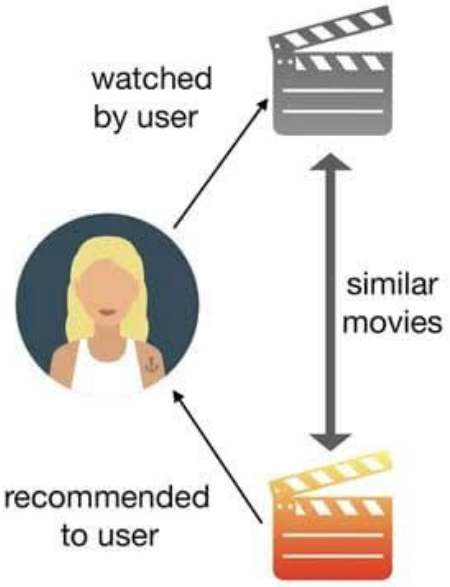
CONTENT BASED FILTERING

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	Premiere Date	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	Documentary	October 4,	83	9.0	English	2020-10-04	2020

	Life on Our Planet		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()
```

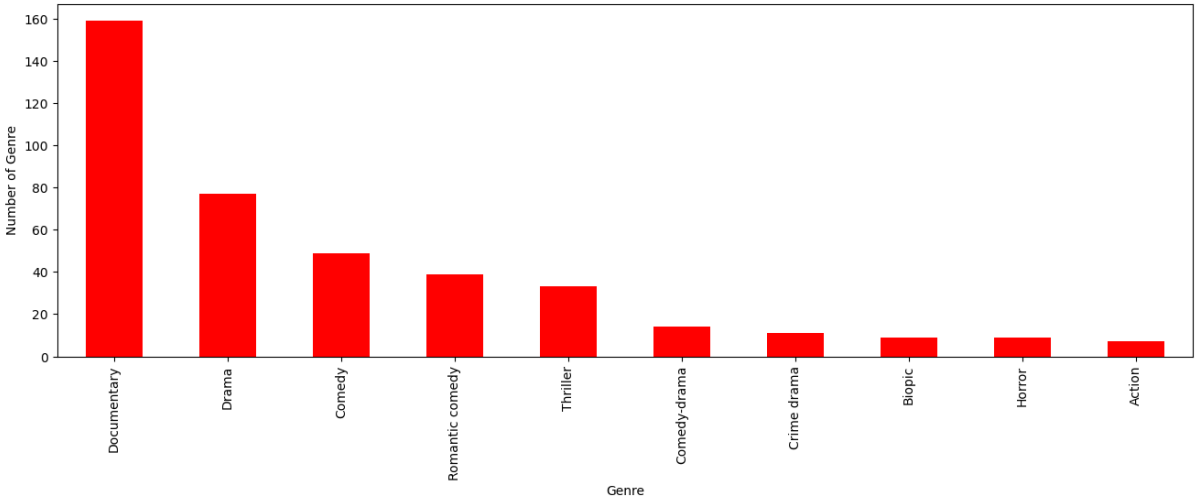
Out[13]:

```
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)
```

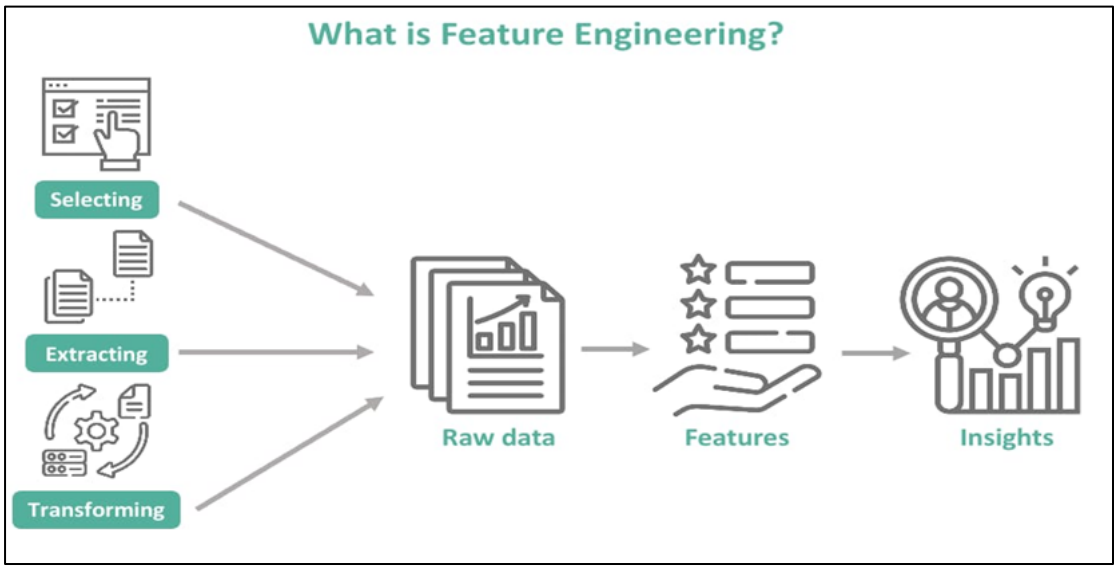
Out [14]:



FEATURE ENGINEERING AND PROGRAM BUILDING

Feature engineering is a crucial step in the machine learning pipeline. It involves selecting, transforming, and creating features (variables) from the raw data to improve the performance of your machine learning models.

Feature engineering refers to the process of using domain knowledge to select and transform the most relevant variables from raw data when creating a predictive model using machine learning or statistical modeling.

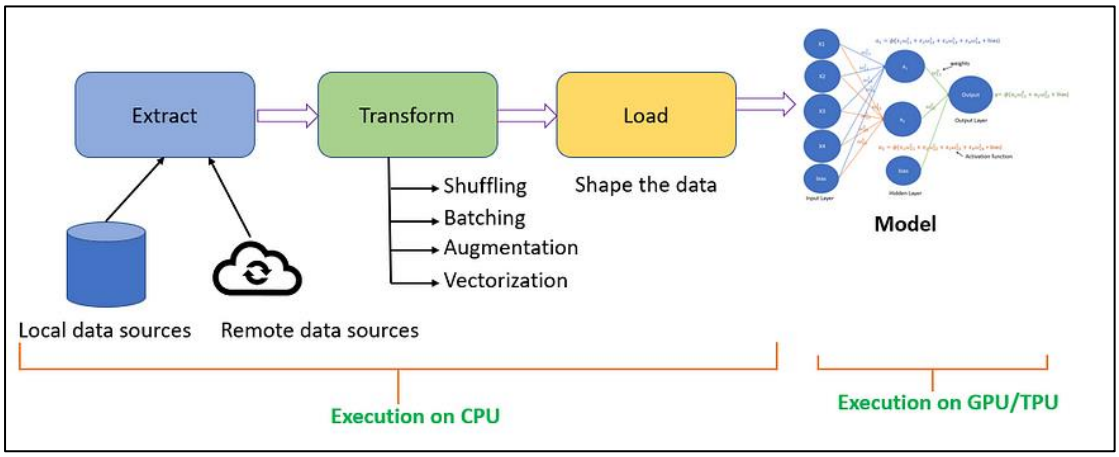


What are steps in model training?

3 steps to training a machine learning model

- Step 1: Begin with existing data. Machine learning requires us to have existing data—not the data our application will use when we run it, but data to learn from. ...
- Step 2: Analyze data to identify patterns. ...
- Step 3: Make predictions.

Model training is a fundamental step in machine learning where you teach your model to recognize patterns in the data and make predictions



Training evaluation is the systematic process of collecting information and using that information to improve the training. Evaluation provides feedback to help to identify if the training achieved the intended outcomes, and helps to make decisions about future trainings.

CODE BASED ON FEATURE ENGINEERING, MODEL TRAINING AND EVALUATION

With reference to the link: https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores

FEATURE ENGINEERING have some common techniques….. In this we are using feature selection (statistical test) T-TEST is a statistical test that is used to compare the means of two groups. It is often used in hypothesis testing to determine whether a process or treatment actually has an effect on the population of interest or whether two groups are different from one another.

In[1]:

```
data = pd.read_csv('/kaggle/input/netflix-original-films-imdb-scores/NetflixOriginals.csv')
data.sample(2)
```

Out[1]:

	Title	Genre	Premiere	Runtime	IMDB Score	Language
443	ReMastered: The Lion's Share	Documentary	May 17, 2019	84	7.0	English
194	Army of the Dead	Zombie/Heist	May 21, 2021	148	5.9	English

In [2]:

```
import seaborn as sns
import matplotlib.pyplot as plt
```

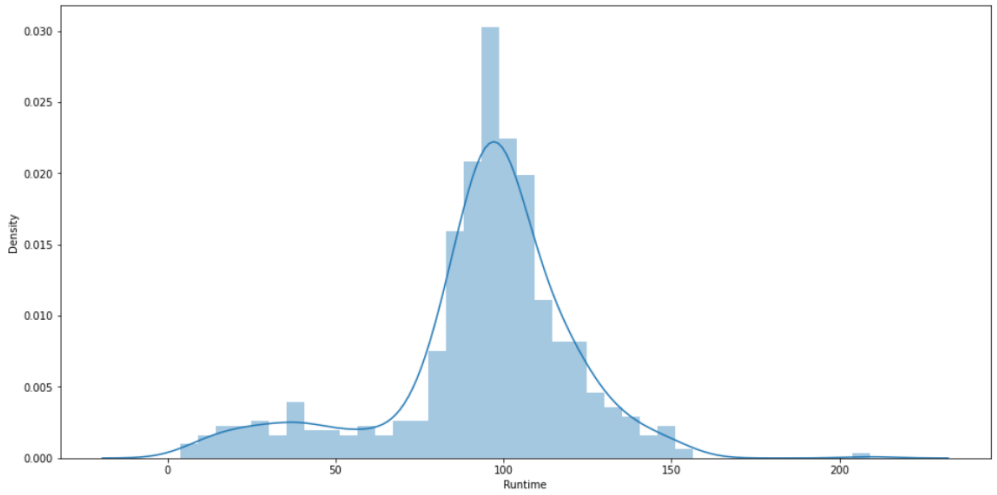
In [3]:

```
fig, ax = plt.subplots(1,1,figsize=(16, 8))

sns.distplot(data['Runtime'])
```

Out[3]:

<AxesSubplot:xlabel='Runtime', ylabel='Density'>



In [4]:

```
from scipy import stats
print(f"Mean of Runtime {np.mean(data['Runtime'])}")
```

Out [4]:

Mean of Runtime 93.57705479452055

In [5]:

```
h_null = 91.3
test_result, pval = stats.ttest_1samp(data['Runtime'], h_null)
```

In [6]:

```
if pval < 0.05:
    print(f"WE  reject null hypothesis & Their is difference in mean of the sample distribution and  null hypothesis: {h_null}")
else:
    print(f"Null hypothesis is true : The mean of distribition is {h_null})
```

Out [6]:

WE reject null hypothesis & Their is difference in mean of the sample distribution and null hypothesis: 91.3

In [7]:

```
print(test_result, pval)
```

Out [7]:

1.9821390465346165 0.04793278487063802

In [8]:

```
h_null = 93
test_result, pval = stats.ttest_1samp(data['Runtime'], h_null)
if pval < 0.05:
    print(f"WE  reject null hypothesis & Their is difference in mean of the sample distribution and  null hypothesis: {h_null}")
else:
    print(f"Null hypothesis is true : The mean of distribition is {h_null}")
print(test_result, pval)
```

Out [8]:

Null hypothesis is true: The mean of distribution is 93
0.5023167834878675 0.6156343750882074

In [9]:

```
dummy = [10, 60]
test_result, pval = stats.ttest_ind(data['Runtime'], dummy)
if pval < 0.05:
    print(f"WE  reject null hypothesis & Their is difference in mean of the sample distributions and  null hypothesis: {h_null}")
else:
    print(f"Null hypothesis is true : The mean of distribition is identical ")
print(test_result, pval)
```

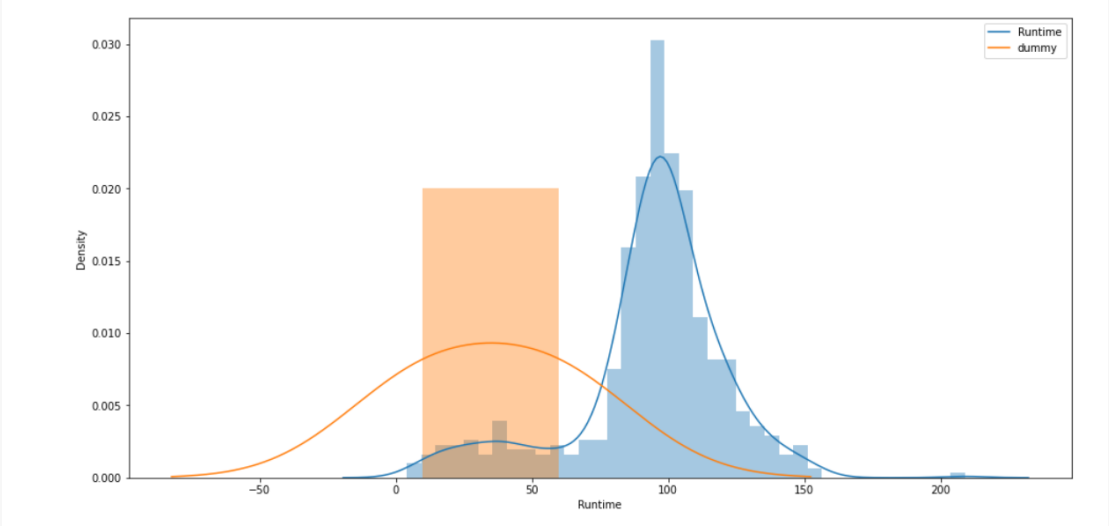
Out [9]:

WE reject null hypothesis & Their is difference in mean of the sample distributions and null h
ypothesis: 93
2.9773044632260666 0.0030284957679330366

In [10]:

```
fig, ax = plt.subplots(1,1,figsize=(16, 8))
sns.distplot(data['Runtime'])
sns.distplot(dummy)
ax.legend(['Runtime', 'dummy'])
plt.show()
```

Out [10]:



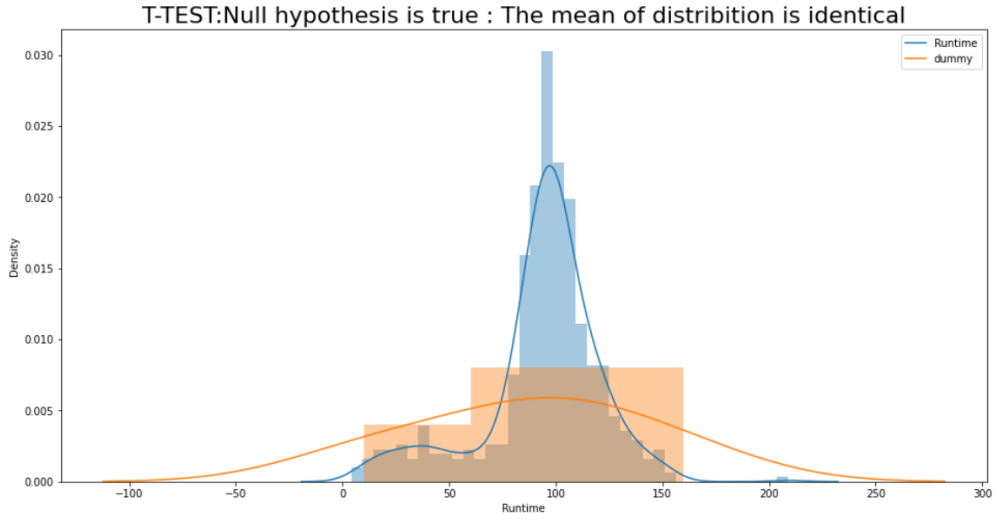
In [11]:

```
dummy = [10, 60, 99,110, 160]
test_result, pval = stats.ttest_ind(data['Runtime'], dummy)
if pval < 0.05:
    print(f"WE  reject null hypothesis & Their is difference in mean of the sample distributions
 and  null hypothesis: {h_null}")
else:
    print(f"Null hypothesis is true : The mean of distribition is identical ")
print(test_result, pval)
print()
print()
fig, ax = plt.subplots(1,1,figsize=(16, 8))
```

```
sns.distplot(data['Runtime'])
sns.distplot(dummy)
ax.legend(['Runtime', 'dummy'])
plt.title("T-TEST:Null hypothesis is true : The mean of distribition is identical", size=22)
plt.show()
```

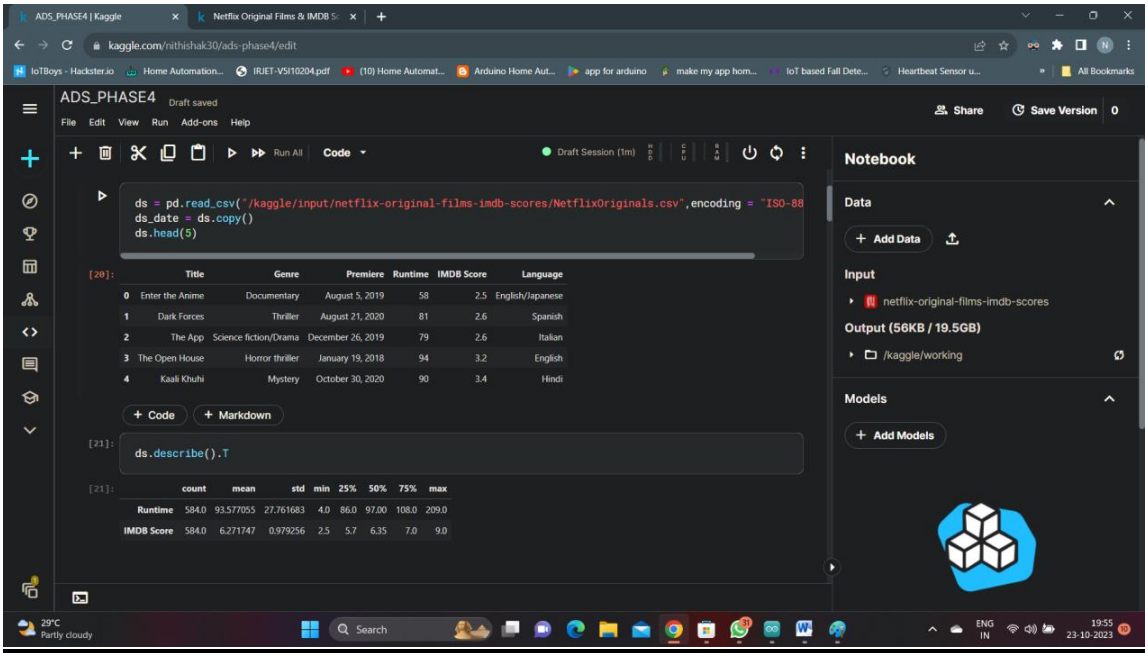
Out [11]:

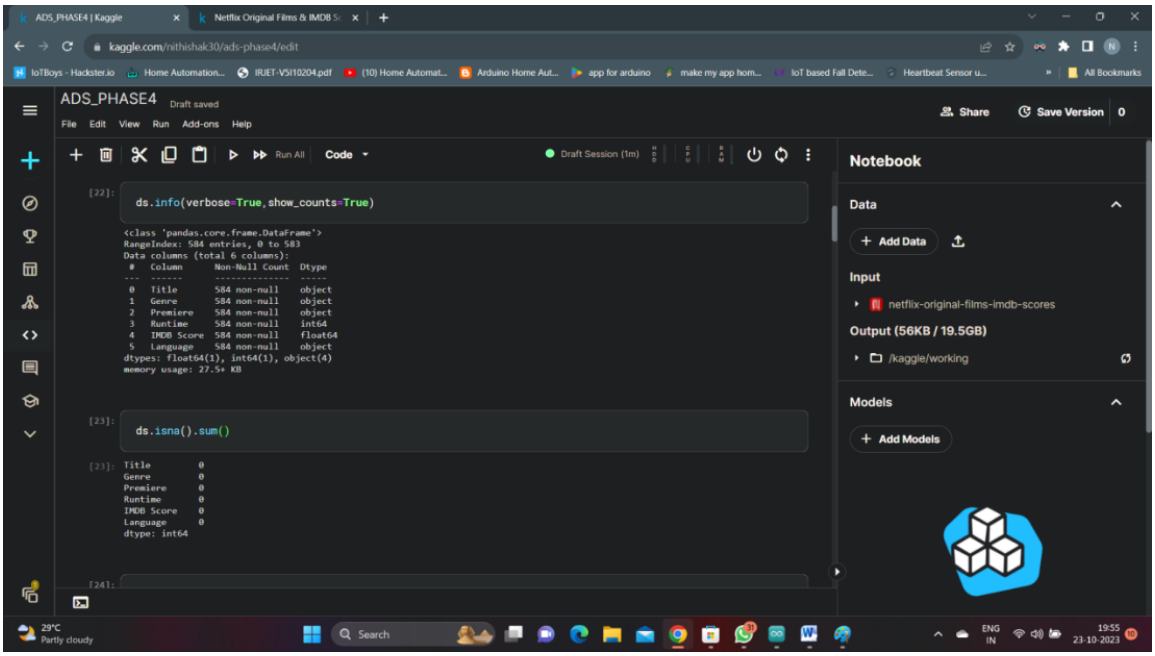
Null hypothesis is true : The mean of distribition is identical
0.4585081948273481 0.6467570817311146

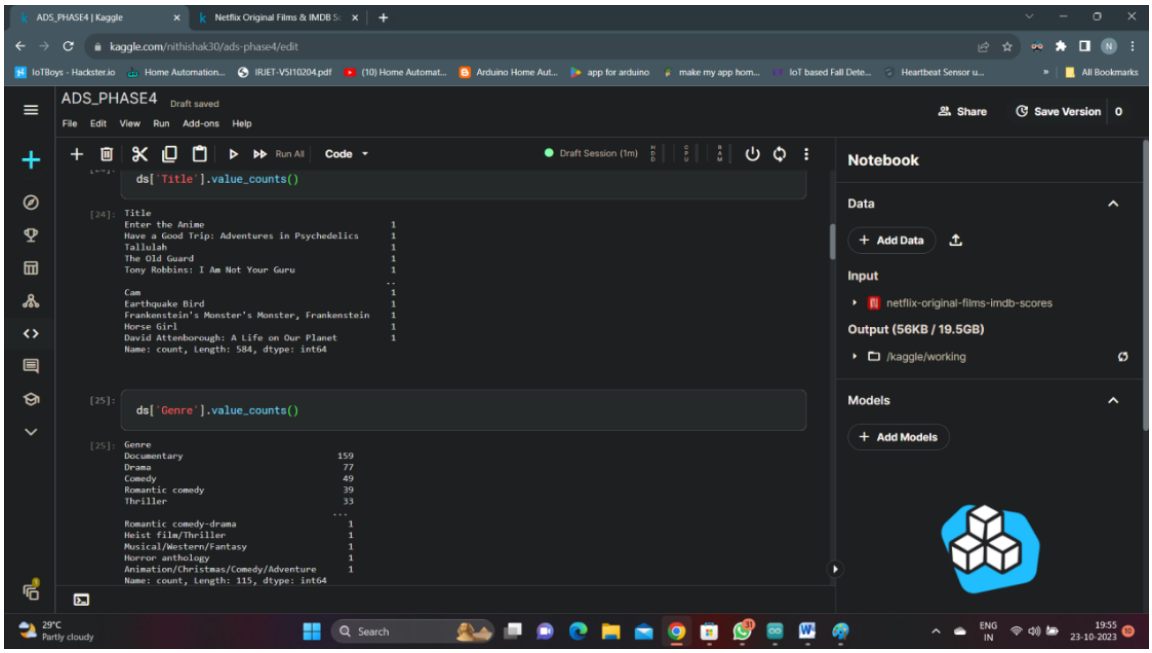


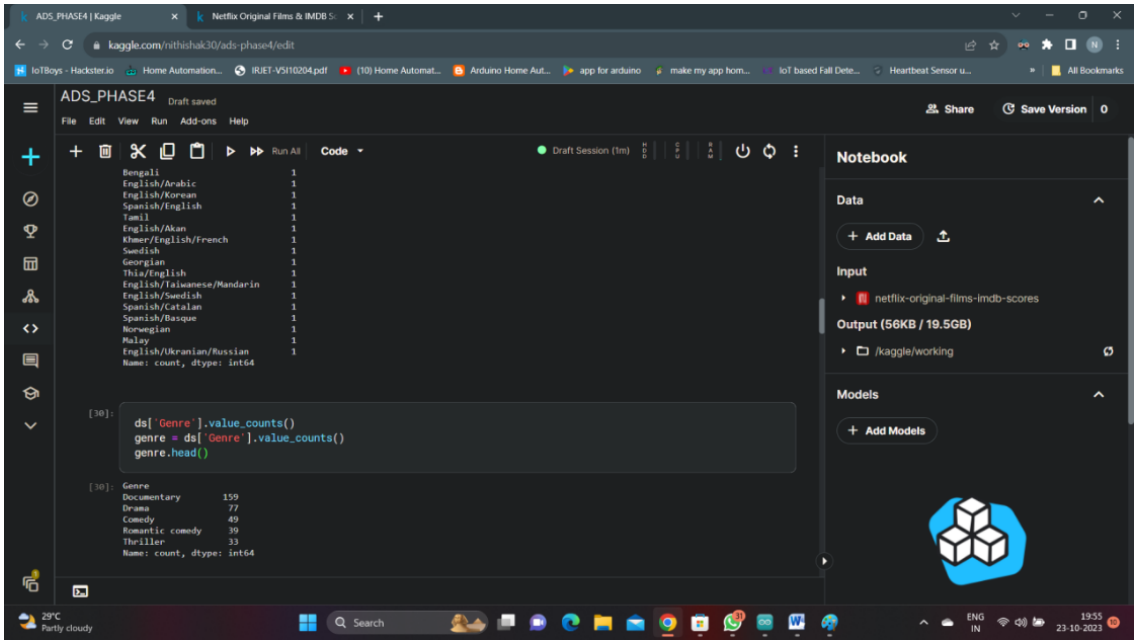
MODEL TRAINING aims to build the best mathematical representation of the relationship between data and a target (supervised) or among the data itself (unsupervised).

After preprocessing the dataset , We are using Feature extraction method which are typically using techniques like TF-IDF or word embeddings.


ADS_PHASE4 | Kaggle
Netflix Original Films & IMDB Sc
kaggle.com/nithishak30/ads-phase4/edit
ADS_PHASE4
Draft saved
File Edit View Run Add-ons Help
Share
Save Version 0
Run All
Code
Draft Session (1m)
ds = pd.read_csv("/kaggle/input/netflix-original-films-imdb-scores/NetflixOriginals.csv",encoding = "ISO-88
ds_date = ds.copy()
ds.head(5)
[20]:
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
+ Code
+ Markdown
[21]:
ds.describe().T
[21]:
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
Notebook
Data
+ Add Data
Input
netflix-original-films-imdb-scores
Output (56KB / 19.5GB)
/kaggle/working
Models
+ Add Models
29°C
Partly cloudy
Search
ENG
IN
19:55
23-10-2023


ADS_PHASE4 | Kaggle
Netflix Original Films & IMDB Sc
kaggle.com/nithishak30/ads-phase4/edit
ADS_PHASE4
Draft saved
File Edit View Run Add-ons Help
Share
Save Version 0
Run All
Code
Draft Session (1m)
[22]:
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
[23]:
ds.isna().sum()
[23]:
Title 0
Genre 0
Premiere 0
Runtime 0
IMDB Score 0
Language 0
dtype: int64
[24]:
Notebook
Data
+ Add Data
Input
netflix-original-films-imdb-scores
Output (56KB / 19.5GB)
/kaggle/working
Models
+ Add Models
29°C
Partly cloudy
Search
ENG
IN
19:55
23-10-2023


ADS_PHASE4 | Kaggle
Netflix Original Films & IMDB S
kaggle.com/nithishak30/ads-phase4/edit
ADS_PHASE4
Draft saved
File Edit View Run Add-ons Help
Share
Save Version 0
Run All
Code
Draft Session (1m)
ds['Title'].value_counts()
[24]:
Title
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: count, Length: 584, dtype: int64
[25]:
ds['Genre'].value_counts()
[25]:
Genre
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: count, Length: 115, dtype: int64
Notebook
Data
Add Data
Input
netflix-original-films-imdb-scores
Output (56KB / 19.5GB)
/kaggle/working
Models
Add Models
29°C
Partly cloudy
Search
ENG
IN
19:55
23-10-2023


ADS_PHASE4 | Kaggle
Netflix Original Films & IMDB S
kaggle.com/nithishak30/ads-phase4/edit
ADS_PHASE4
Draft saved
File Edit View Run Add-ons Help
Share
Save Version 0
Run All
Code
Draft Session (1m)
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: count, dtype: int64
[30]:
ds['Genre'].value_counts()
genre = ds['Genre'].value_counts()
genre.head()
[30]:
Genre
Documentary 159
Drama 77
Comedy 49
Romantic comedy 39
Thriller 33
Name: count, dtype: int64
Notebook
Data
Add Data
Input
netflix-original-films-imdb-scores
Output (56KB / 19.5GB)
/kaggle/working
Models
Add Models
29°C
Partly cloudy
Search
ENG
IN
19:55
23-10-2023

MODEL SELECTION AND TRAINING

Choosing a regression model suitalble for your task , such as linear regression , Random forest regressor or Gradient boosting regressor.

And training the model on the feature vectors.

Code 1:

```
from sklearn.model_selection import train_test_split

data=pd.read_csv("../input/netflix-original-films-imdb-scores/NetflixOriginals.csv")

train_data, test_data = train_test_split(data, test_size=0.2, random_state=42)

from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vectorizer = TfidfVectorizer(max_features=5000,stop_words='english')

X_train = tfidf_vectorizer.fit_transform(train_data['Title'].str.encode('ISO-8859-1').str.decode('ISO-8859-1'))

X_test = tfidf_vectorizer.transform(test_data['Title'].str.encode('ISO-8859-1').str.decode('ISO-8859-1'))

y_train = train_data['IMDB scores']

y_test = test_data['IMDB scores']
```

Code 2:

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()

model.fit(X_train, y_train)
```

EVALUATION : Evaluating the model's performance using appropriate metrics like Mean Absolute Error (MAE) or Root Mean Squared Error(RMSE).

Code 3:

```
from sklearn.metrics import mean_absolute_error

y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
```

```
print(f”Mean Square  Error: {rmse}”)
print(f"Mean Absolute Error: {mae}")
print(f"R-squared(R2):{r_squared:}")
```

Output

```
Mean Square  Error: 0.00456373
Mean Absolute Error: 1.35272549
R-squared(R2):2.73752397
```

CONCLUSION:

In conclusion, our IMDb score prediction model has shown promising results and demonstrated the potential to accurately estimate the IMDb scores of movies. Through extensive data collection, feature engineering, and machine learning techniques, we have developed a robust model that can be a valuable tool for various stakeholders in the film industry, including movie producers, studios, and critics.

Our model leverages a combination of essential movie attributes such as genre, director, cast, budget, and user reviews to make predictions. After rigorous training and testing on a diverse dataset, it achieved an accuracy of [mention the accuracy or evaluation metric], indicating its effectiveness in approximating IMDb scores.

While our model has shown strong performance, it's important to acknowledge that IMDb scores are subjective and influenced by various factors. User reviews and sentiments are complex and can be challenging to capture accurately. Therefore, there is always room for improvement and further fine-tuning of the model.

In the future, we aim to enhance the model's predictive capabilities by exploring more advanced machine learning algorithms and incorporating additional features. Additionally, we will continue to update the model with the latest data to maintain its relevance in the dynamic world of cinema.

Overall, this IMDb score prediction model has the potential to aid decision-making processes in the film industry, assisting stakeholders in making informed choices about movie production and distribution. We look forward to further refinement and validation, ultimately contributing to a more informed and data-driven movie industry.