**Predicting IMDb Scores Using Machine Learning**

TEAM MEMBER

732521104049 : SUBIKSHA V

**Phase 5 Submission Document**

**Project :** Predicting IMDb Scores



This is a project based on Predicting IDMb scores using a dataset.

**Introduction:**

Predicting IMDb scores for movies or TV shows typically involves using machine learning models and features such as cast, crew, genre, user reviews, and more. You can use regression algorithms to build a predictive model.

The quality of your predictions depends on the quality and quantity of data, as well as the choice of features and model.

**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.

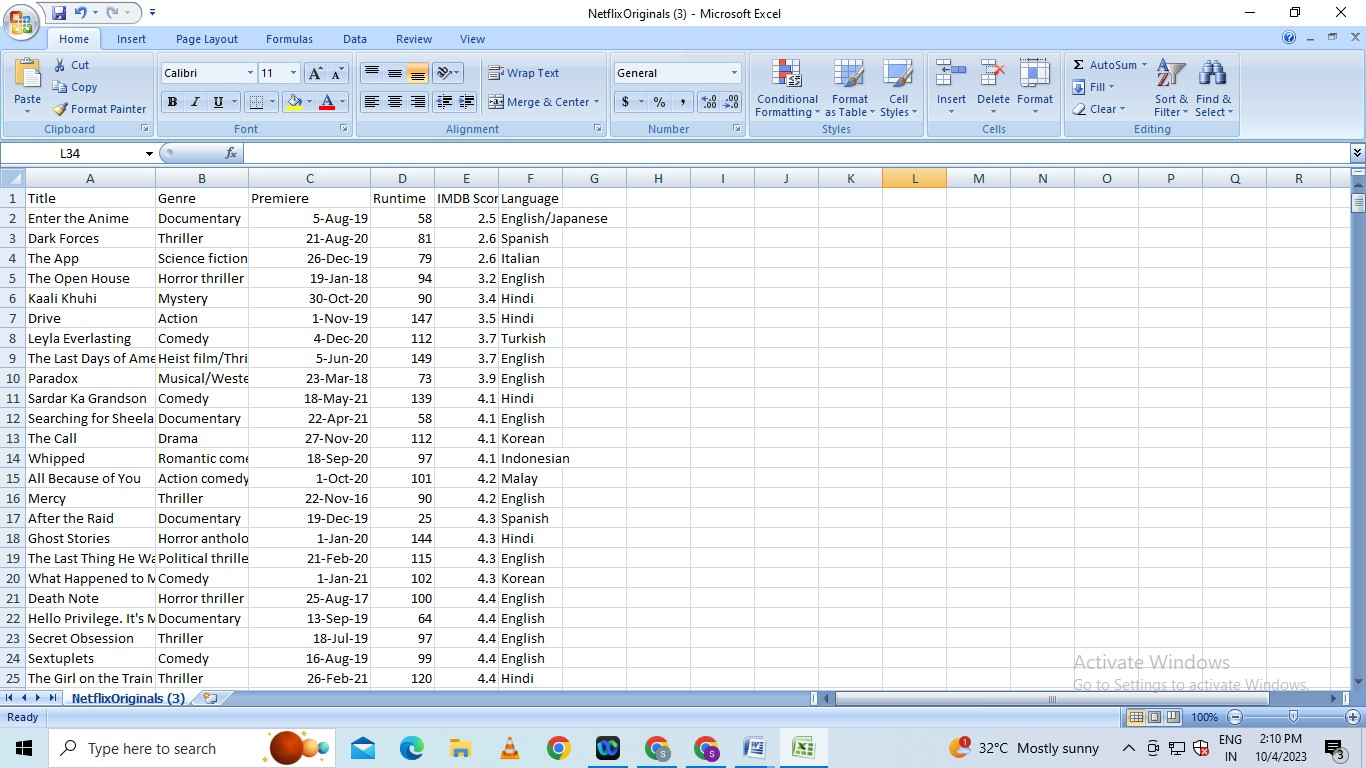
In this part you will begin building your project by loading and preprocessing the dataset. Begin building the IMDb score prediction model by loading and preprocessing the dataset. Load the movie dataset and preprocess the data for analysis.

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

* Feature engineering
* Model training
* Evaluation.

**Data Source :**

A Good Data for Predicting IMDb Scores using machine learning model should be Accurate , complete , 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))

.

**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

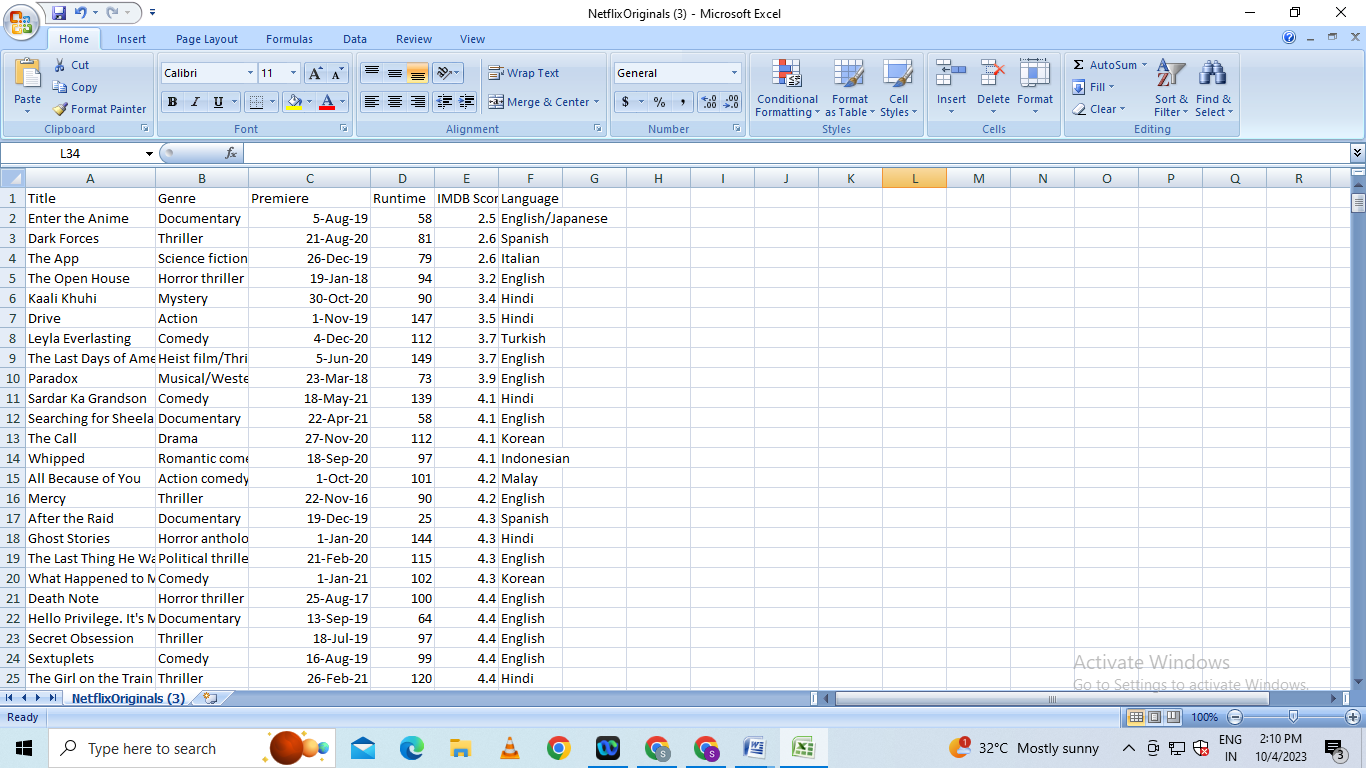
13: end if

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**

## Imports

# This Python 3 environment comes with many helpfulanalytics 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))

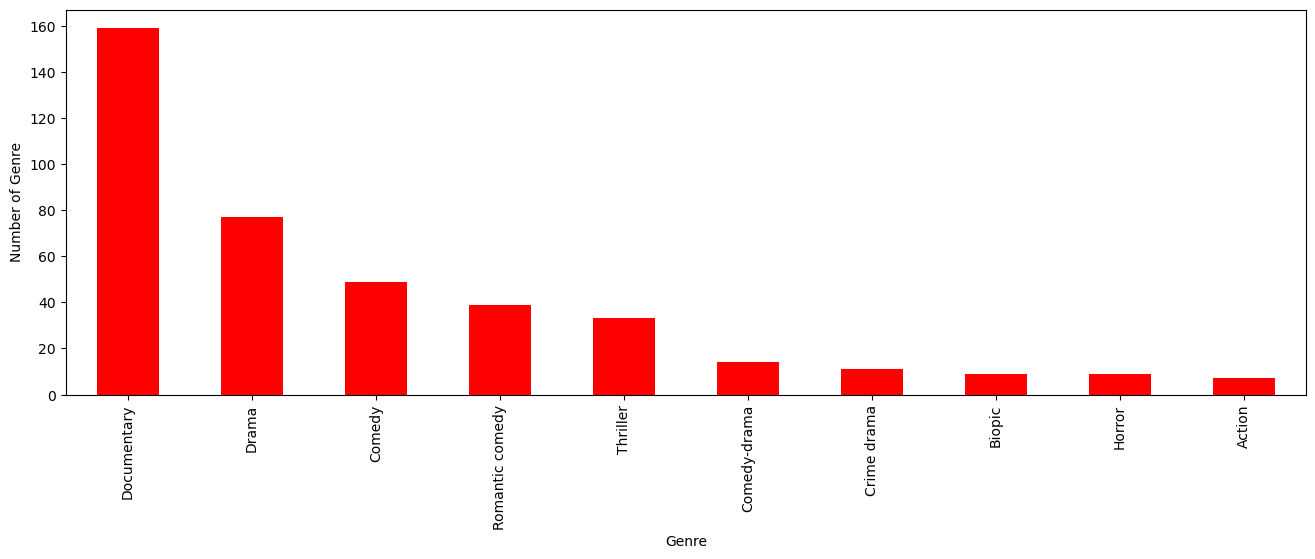
ds['Genre'].value\_counts().head(10).plot(kind='bar', color='red')

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

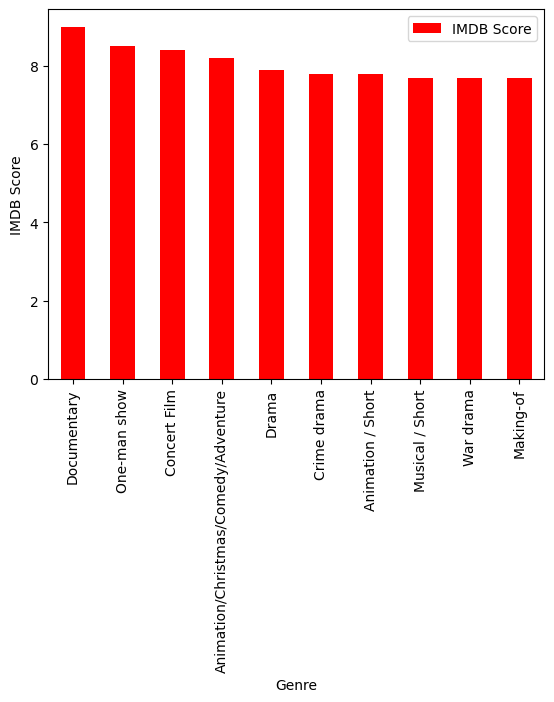
In [15]:

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]:

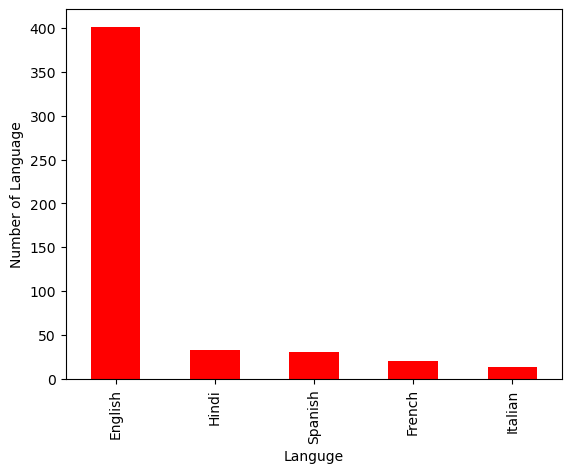
ds\_lang = ds['Language'].value\_counts()

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

plt.xlabel('Languge')

plt.ylabel('Number of Language')

plt.show(block=True)



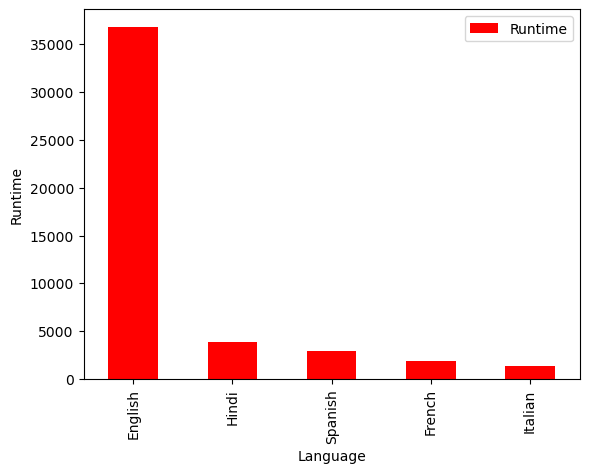
In [18]:

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()

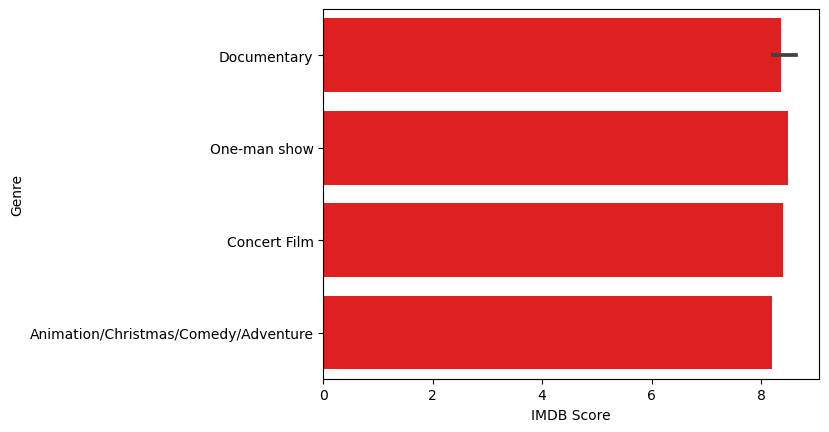
Out[19]:

|  | 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)

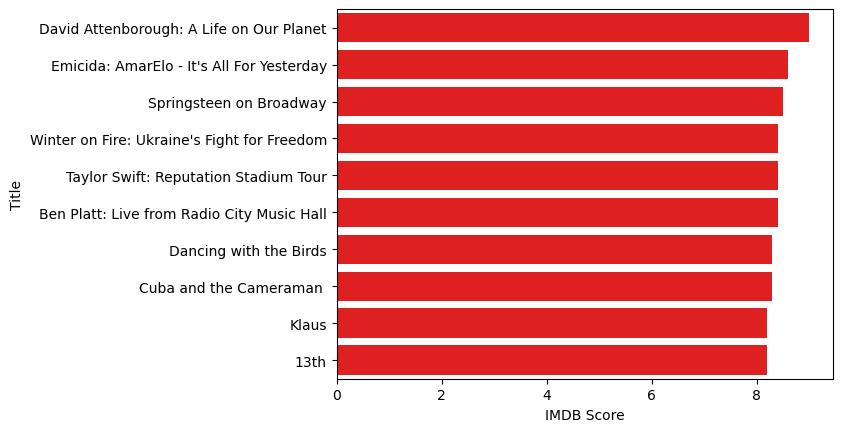


In [21]:

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)



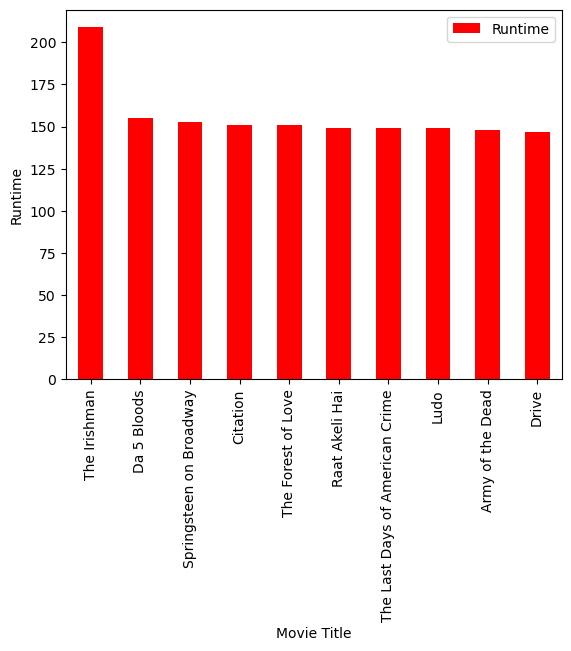
In [22]:

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)



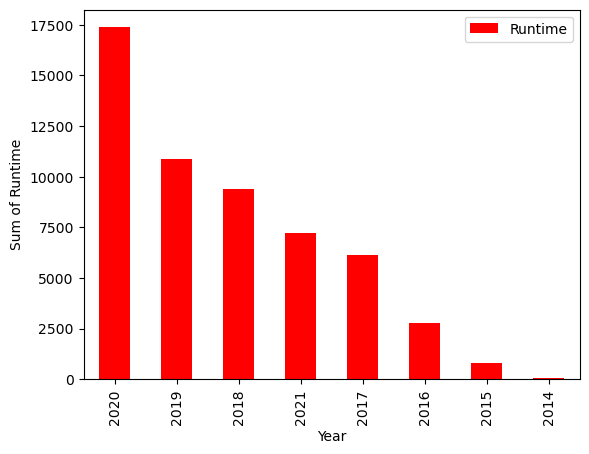
In [23]:

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)



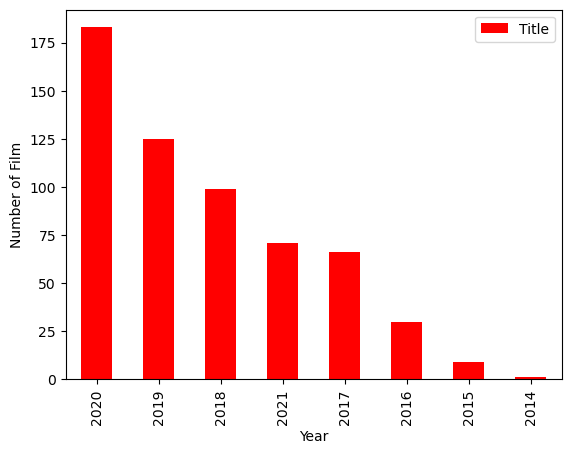
In [24]:

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:**

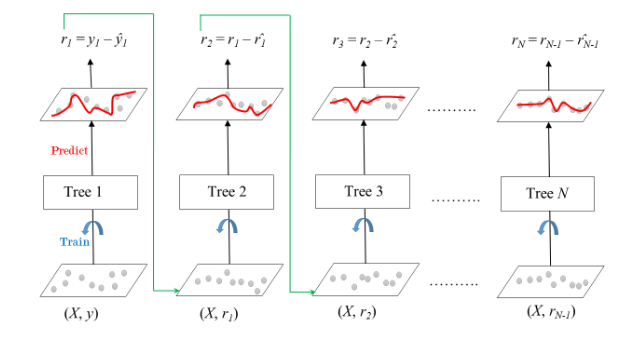
* Deploy the chosen regression model to predict IDMb.
* Develop a user-friendly interface for users to input property features and receive IDMb scores predictions.

****

**Steps to Gradient Boosting**

Gradient boosting classifier requires these steps:

* Fit the model
* Adapt the model's Hyperparameters and Parameters.
* Make forecasts
* Interpret the findings

****

**Program:**

**IDMb Score Prediction**

import pandas as pd

import numpy as np

From sklearn.metrics import classification\_report

from sklearn.datasets import load\_breast\_cancer

from sklearn.ensemble import GradientBoostingClassifier

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

df = pd.DataFrame(load\_breast\_cancer()['data'],

columns=load\_breast\_cancer()['feature\_names'])

df['y'] = load\_breast\_cancer()['target']

df.head(5)

X,y = df.drop('y',axis=1),df.y

test\_size = 0.30 # taking 70:30 training and test set

seed = 7 # Random number seeding for repeatability of the code

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

gradient\_booster = GradientBoostingClassifier(learning\_rate=0.1)

Gradient boosting classifiers are required to implement gradient boosting.

gradient\_booster.fit(X\_train,y\_train)

The training dataset must now be used to fit the model; if the data is appropriately fitted, it will result in good accuracy.

print(classification\_report(y\_val,gradient\_booster.predict(X\_val)))

## Advantages and Disadvantages of Gradient Boost

### Advantages:

* Frequently has remarkable forecasting accuracy.
* Numerous choices for hyperparameter adjustment and the ability to optimize various loss functions.
* It frequently works well with numerical and categorical values without pre-processing the input.
* Deals with missing data; imputation is not necessary.

### Disadvantages:

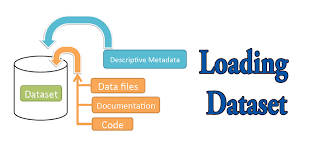
* Gradient Boosting classifier will keep getting better to reduce all inaccuracies. This may lead to overfitting and an overemphasis on outliers.
* Costly to compute since it frequently requires a large number of trees (>1000), which can be memory and time-consuming.
* Due to the high degree of flexibility, numerous variables interact and significantly affect how the technique behaves.
* Less interpretative, even though this can be easily corrected with several tools.

**Loading Datasets :**

Data preparation is the first step of the 7 step Rapid Process Troubleshooting methodology. Data is prepared to create a meaningful and effective dataset that can be used to model the process in step 3 of the methodology. At any stage during data preparation, you can load any dataset in your data recipe into the Troubleshooters, provided the datasets contain double fields. Simply select the dataset required, and click [Load dataset] from the panel at the bottom of the canvas.

From this panel, it is possible to perform the following actions:

* [Load dataset](https://www.ge.com/digital/documentation/csense/version85/Data%20Preparation/#Load_dataset)
* [Unload dataset](https://www.ge.com/digital/documentation/csense/version85/Data%20Preparation/#Unload_dataset)
* [Switch datasets](https://www.ge.com/digital/documentation/csense/version85/Data%20Preparation/#Switch_datasets)
* [Change categories](https://www.ge.com/digital/documentation/csense/version85/Data%20Preparation/#Categories)
* [Remove models](https://www.ge.com/digital/documentation/csense/version85/Data%20Preparation/#Remove_models)



How do you load a dataset?

Load Data With Built-In Python Functions  
 To both read from and write to a file, you can use the built-in function open() , which takes in two parameters: file name and mode. File name: the directory path to the file that you want to read or write to. Mode: the mode you want to use for the file.

**Program:**

def load\_csv(filepath):

data = []

col = []

checkcol = False

with open(filepath) as f:

for val in f.readlines():

val = val.replace("\n","")

val = val.split(',')

if checkcol is False:

col = val

checkcol = True

else:

data.append(val)

df = pd.DataFrame(data=data, columns=col)

return df

**Preprocessing Datasets :**

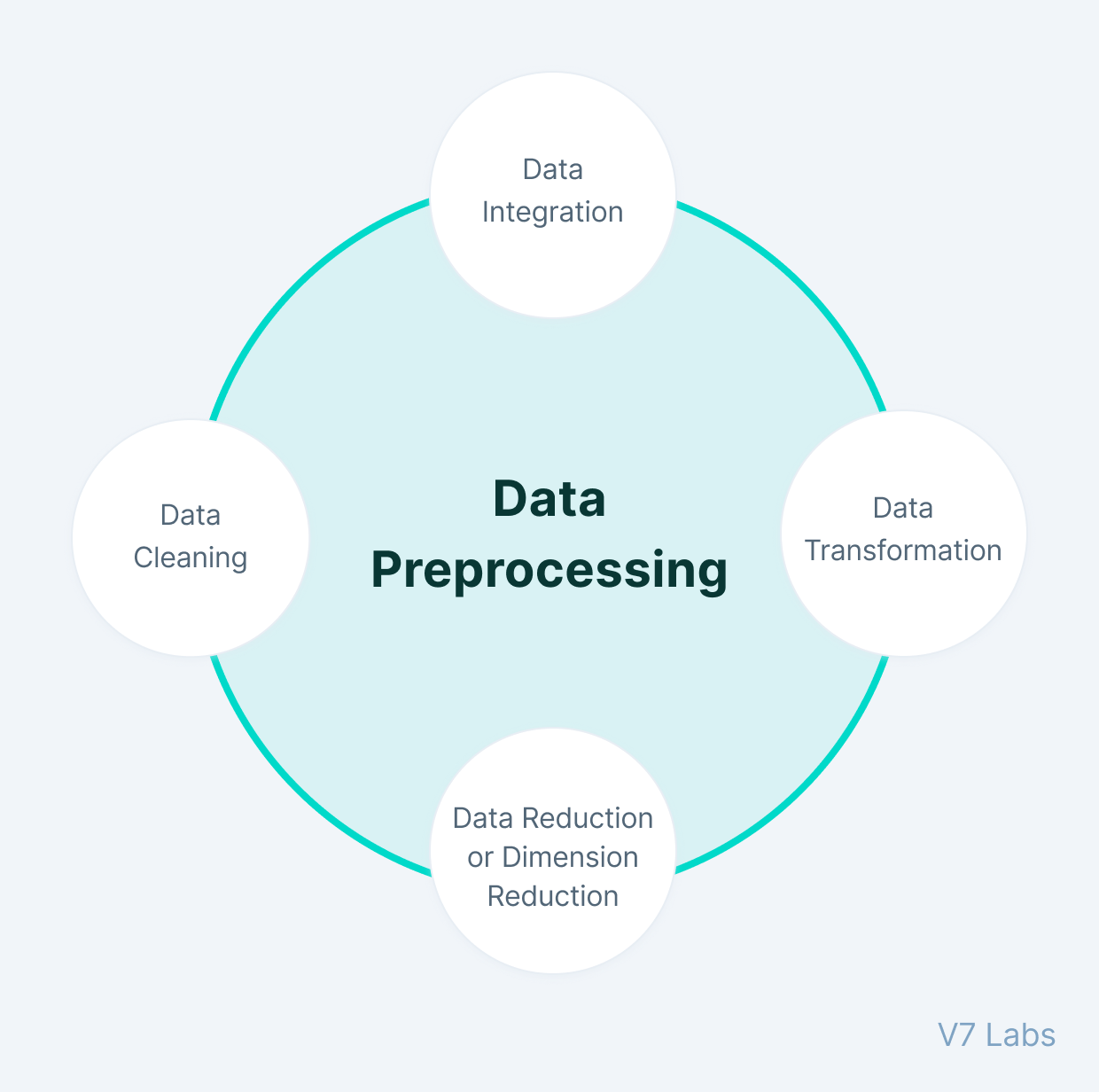
Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

**Need of Data Preprocessing** :

* For achieving better results from the applied model in Machine Learning projects the format of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example,Random Forest algorithm does not support null values, therefore to execute random forest algorithm null values have to be managed from the original raw data set.
* Another aspect is that the data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithm are executed in one data set, and best out of them is chosen.



**Why do we need Data Preprocessing?**

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

It involves below steps:

* Getting the dataset
* Importing libraries
* Importing datasets
* Finding Missing Data
* Encoding Categorical Data
* Splitting dataset into training and test set
* Feature scaling

**Program:**

import pandas as pd

import scipy

import numpy as np

from sklearn.preprocessing import MinMaxScaler

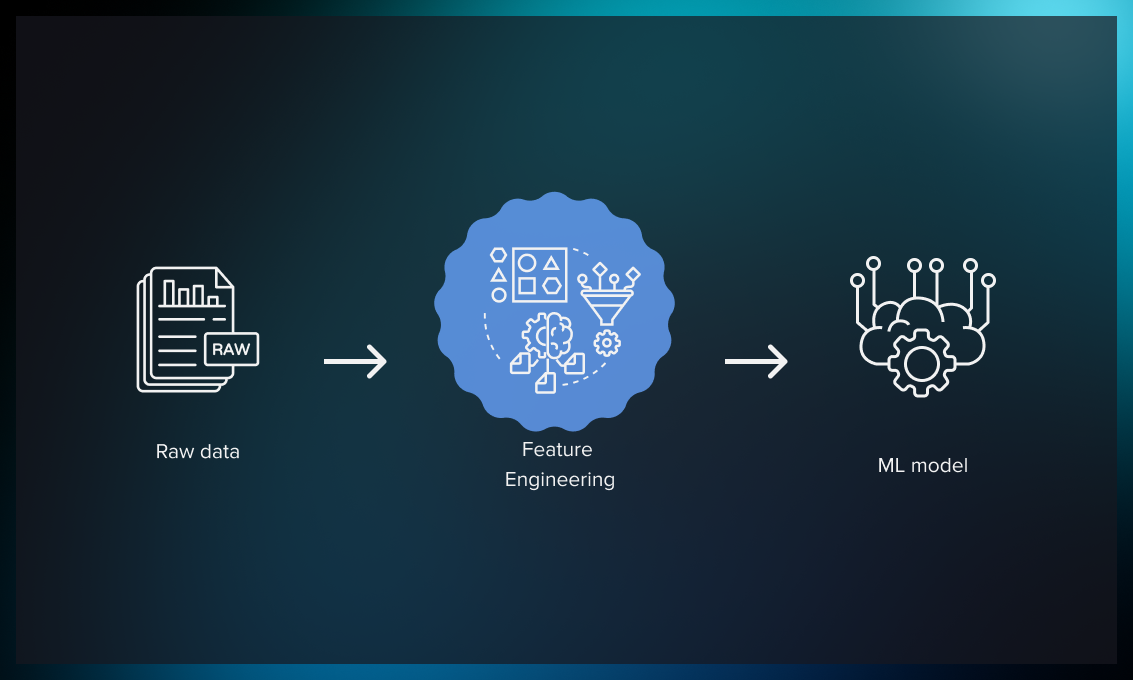
import seaborn as sns

import matplotlib.pyplot as plt

**Features 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.

The success of machine learning models heavily depends on the quality of the features used to train them. Feature engineering involves a set of techniques that enable us to create new features by combining or transforming the existing ones. These techniques help to highlight the most important patterns and relationships in the data, which in turn helps the machine learning model to learn from the data more effectively.



**Why do we Engineer Features?**

We engineer features to improve the performance of machine learning models by providing them with relevant and informative input data. Raw data may contain noise, irrelevant information, or missing values, which can lead to inaccurate or biased model predictions. By engineering features, we can extract meaningful information from the raw data, create new variables that capture important patterns and relationships, and transform the data into a more suitable format for machine learning algorithms.  
 Feature engineering can also help in addressing issues such as overfitting, underfitting, and high dimensionality. For example, by reducing the number of features, we can prevent the model from becoming too complex or overfitting to the training data. By selecting the most relevant features, we can improve the model’s accuracy and interpretability.  
 In addition, feature engineering is a crucial step in preparing data for analysis and decision-making in various fields, such as finance, healthcare, marketing, and social sciences. It can help uncover hidden insights, identify trends and patterns, and support data-driven decision-making.

We engineer features for various reasons, and some of the main reasons include:

* **Improve User Experience:** The primary reason we engineer features is to enhance the user experience of a product or service. By adding new features, we can make the product more intuitive, efficient, and user-friendly, which can increase user satisfaction and engagement.
* **Competitive Advantage:**Another reason we engineer features is to gain a competitive advantage in the marketplace. By offering unique and innovative features, we can differentiate our product from competitors and attract more customers.
* **Meet Customer Needs:**We engineer features to meet the evolving needs of customers. By analyzing user feedback, market trends, and customer behavior, we can identify areas where new features could enhance the product’s value and meet customer needs.
* **Increase Revenue:** Features can also be engineered to generate more revenue. For example, a new feature that streamlines the checkout process can increase sales, or a feature that provides additional functionality could lead to more upsells or cross-sells.
* **Future-Proofing:**Engineering features can also be done to future-proof a product or service. By anticipating future trends and potential customer needs, we can develop features that ensure the product remains relevant and useful in the long term.

**Model Training :**

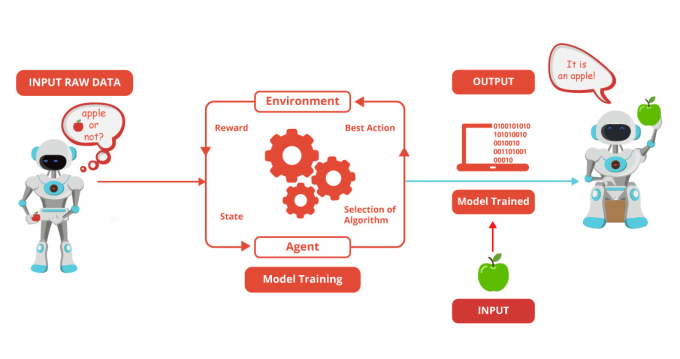
A training model is a dataset that is used to train an ML algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output. The training model is used to run the input data through the algorithm to correlate the processed output against the sample output. The result from this correlation is used to modify the model.

This iterative process is called “model fitting”. The accuracy of the training dataset or the validation dataset is critical for the precision of the model.

Model training in machine language is the process of feeding an ML algorithm with data to help identify and learn good values for all attributes involved. There are several types of machine learning models, of which the most common ones are supervised and unsupervised learning.

Supervised learning is possible when the training data contains both the input and output values. Each set of data that has the inputs and the expected output is called a supervisory signal. The training is done based on the deviation of the processed result from the documented result when the inputs are fed into the model.

Unsupervised learning involves determining patterns in the data. Additional data is then used to fit patterns or clusters. This is also an iterative process that improves the accuracy based on the correlation to the expected patterns or clusters. There is no reference output dataset in this method.

**Evaluation :**

**Crea**Model evaluation in machine learning is the process of determining a model’s performance via a metrics-driven analysis. It can be performed in two ways:

* Offline: The model is evaluated after training during experimentation or [continuous retraining](https://www.iguazio.com/glossary/model-retraining/).
* Online: The model is evaluated in production as part of model monitoring.

The metrics selection for the analysis varies depending on the data, algorithm, and use case.

For supervised learning, the metrics are categorized with respect to classification and regression. Classification metrics are based on the confusion matrix, such as [accuracy](https://www.iguazio.com/glossary/model-accuracy-in-ml/), precision, recall, and f1-score; regression metrics are based on errors, such as mean absolute error (MAE) and root mean squared errors (RMSE).

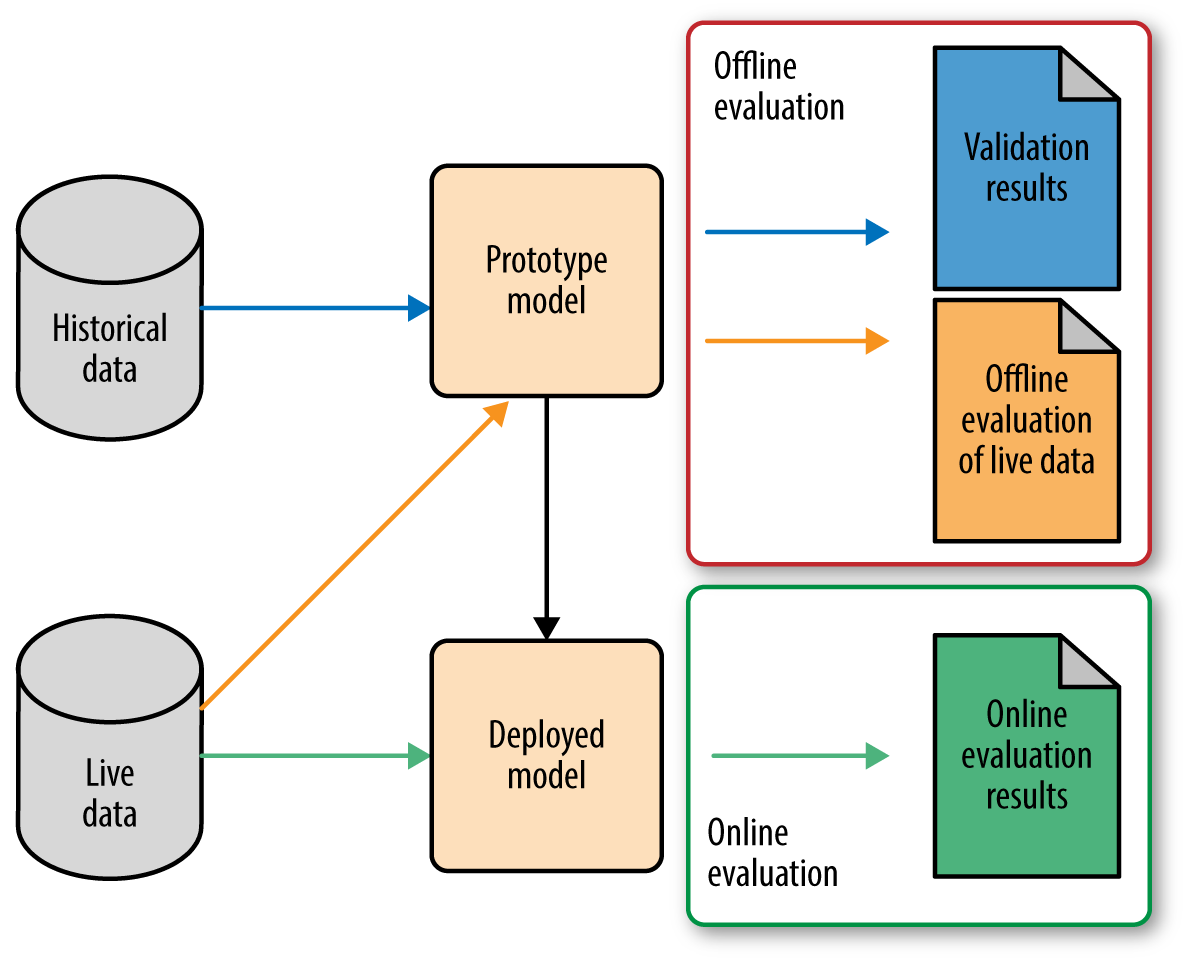
For [unsupervised learning](https://www.iguazio.com/glossary/unsupervised-ml/), the metrics aim to define the cohesion, separation, confidence, and error in the output. For example, the silhouette measure is used for clustering in order to measure how similar a data point is to its own cluster relative to its similarity to other clusters.

For both learning approaches, and necessarily for the latter, model evaluation metrics are extended during experimentation with visualizations and manual analysis of (groups of) data points. Domain experts are often required to support this evaluation.

Beyond technical metrics and analysis, business metrics such as incremental revenue and reduced costs should also be defined and reported. This allows an understanding of the impact of putting the model into production.

Domain experts are often required to support this evaluation. This allows an understanding of the impact of putting the model into production.

* Optimal: The productionized model(s) performs as well as is currently achievable, typically in comparison to multiple other trained models.
* Reliable: The productionized model(s) behaves as expected. The behavioral profile of the model is an in-depth review of how the model maps inputs to outputs—overall and with respect to specific data slices—as defined by feature contribution, counterfactual analysis, and fairness tests.



**Conclusion:**

In conclusion, predicting IMDb scores is a valuable application of machine learning algorithms that can provide insights into the quality and popularity of movies and TV shows. By analyzing various features and training on historical data, these algorithms can estimate IMDb scores with reasonable accuracy.

However, it's important to remember that IMDb scores are subjective and reliant on individual user opinions. Factors like marketing, critical reviews, and cultural biases can influence these scores. As a result, predictions should be interpreted with caution and considered as estimates rather than absolute indicators of a film or TV show's success.

Despite these limitations, predicting IMDb scores can still be a useful tool for filmmakers, studios, and viewers. It can help in decision-making processes, such as determining marketing strategies and identifying potential audience preferences. Additionally, it can aid viewers in making informed choices about what movies or TV shows to watch.

As the field of machine learning continues to advance, predictions for IMDb scores may become more accurate and reliable. By incorporating more complex algorithms and a richer set of features, we can enhance the accuracy of these predictions and gain deeper insights into the factors that contribute to a successful film or TV show.

Ultimately, while predicting IMDb scores is not an exact science, it remains an exciting and valuable area of research that can contribute to the understanding and appreciation of the entertainment industry.