

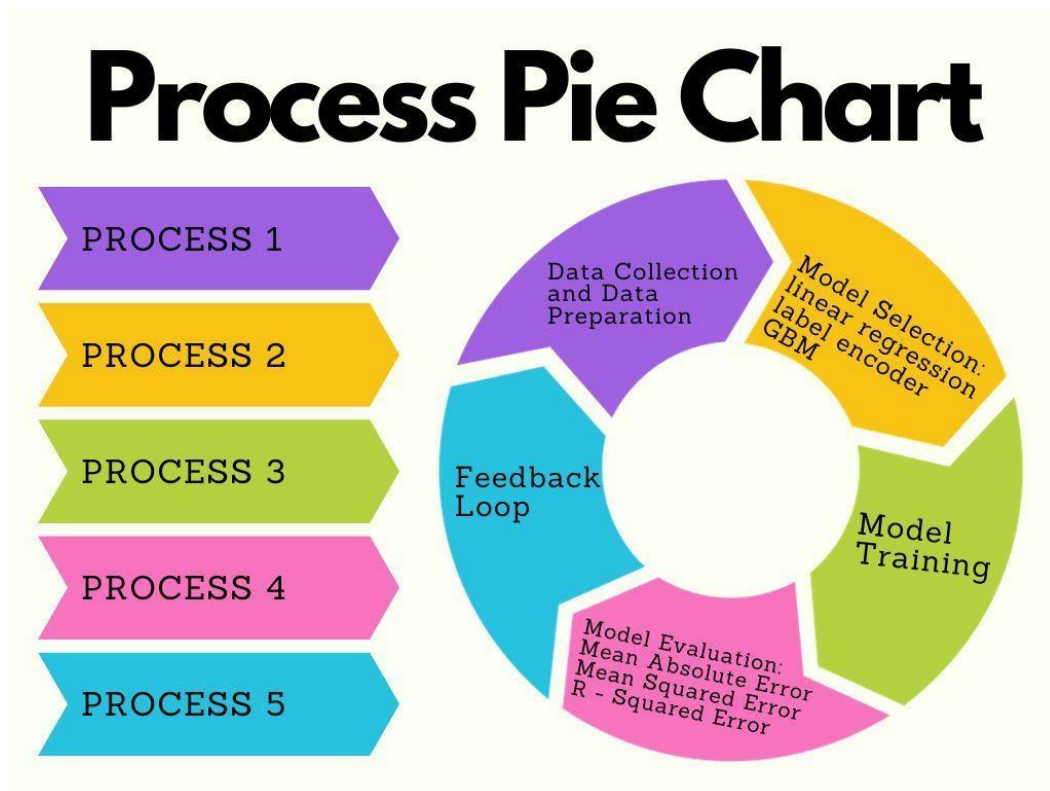
# IMDb SCORE PREDICTION

## (PHASE-5)

### Introduction:

Predicting IMDb scores for movies or TV shows is a complex task that often involves a combination of factors, including the quality of the content, marketing, critical reception, and audience reception. Estimate the popularity of movies. This estimation is based on certain features, including genre, premiere date, runtime, and language. In other words, to build a model that can predict how well a movie is likely to be rated on IMDb.

### Pie chart:



## **Data Collection:**

- **Movie/TV Show Information:** List all relevant details about the project, including title, release date, director, writer, cast, genre, budget, box office performance (if applicable), production company, and any notable awards or nominations.
- **Historical IMDb Data:** Mention any historical IMDb scores for similar movies or TV shows that will be used as benchmarks.

## **Dataset Link:**

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

## **Factors Affecting IMDb Scores**

- **Cast Significance:**  
Discuss the importance of the cast, their star power, and acting talent.
- **Genre Impact:**  
Explain how the chosen genre may influence IMDb scores.
- **Budget and Production Values:**  
Describe the relationship between budget and production values and how they can impact audience perception.
- **Marketing and Promotion:**  
Discuss the role of effective marketing and promotion in generating audience interest.
- **Early Reviews and Previews:**  
Explain how critical and audience reactions from early reviews and preview screenings can provide insights.
- **Box Office Performance:**  
Mention whether box office performance is a relevant factor and how it correlates with IMDb scores.

## **Data preparation:**

Clean and preprocess the data, handling missing values, outliers, and encoding categorical variables

### **1. Handling Missing Values:**

- Decide on a strategy for dealing with missing data. Common options include:
- Removing rows or columns with too many missing values.
- Imputing missing values with the mean, median, mode, or a predictive model.
- Using domain knowledge to infer missing values

### **2. Data Cleaning:**

- Address data inconsistencies and errors. This may involve:
- Correcting typos and spelling mistakes.
- Standardizing categorical variables.
- Handling duplicates.
- Removing irrelevant or redundant features.

## **Model selection:**

Once you have identified your features, you need to select a machine learning model to train. Some popular models for predicting IMDb scores include:

- Linear regression
- Label encoder
- Gradient boosting machines

## **Linear Regression:**

Linear regression is a widely used statistical method for modeling the relationship between a dependent variable (also called the target or outcome variable) and one or more independent variables (also called predictors or features). It is a fundamental technique in the field of machine learning and statistics, often employed for tasks like predicting numerical values, understanding correlations, and making forecasts.

## **Label Encoder:**

A label encoder is a preprocessing technique used in machine learning and data analysis to convert categorical or textual data into numerical values. Categorical data consists of labels or categories that do not have any inherent numerical meaning, such as colors, car brands, or job titles. Machine learning algorithms often require numerical inputs, so label encoding is used to represent categorical data in a format that can be used for model training.

## **Gradient boosting machines:**

Gradient Boosting Machines (GBM) is a powerful machine learning technique used for both regression and classification tasks. It's an ensemble learning method that builds a predictive model by combining the predictions of multiple individual models, typically decision trees. GBM has gained popularity in the field of machine learning due to its high predictive accuracy and robustness.

## **Model Training:**

Split our data into training and testing sets. Train our model on the training data. - Split our dataset into training and testing subsets (typically 70-80% for training and the rest for testing). This separation helps evaluate the model's generalization performance. - Train each selected model on the training data, using techniques like cross-validation to prevent overfitting.

## **Model Evaluation :**

Assess the performance of each model using appropriate regression evaluation metrics:

- **Mean Absolute Error (MAE):** Measures the average absolute difference between predicted and actual IMDb scores. Mean Absolute Error (MAE) is a commonly used metric in statistics and machine learning to evaluate the accuracy of a predictive model. It quantifies

the average absolute difference between predicted values and actual values. In other words, it measures how close the model's predictions are to the true values in terms of magnitude, without considering the direction (overestimation or underestimation).

- **Mean Squared Error (MSE):** Measures the average squared difference between predicted and actual scores, giving more weight to large errors. Mean Squared Error (MSE) is a widely used metric in statistics and machine learning to measure the average squared difference between predicted values and actual values. It quantifies the accuracy of a predictive model, with larger errors receiving more weight due to the squaring operation.

- **R-squared ( $R^2$ ):** Evaluates how well the model fits the data. A higher  $R^2$  indicates a better fit. R-squared, also known as the coefficient of determination, is a statistical measure used to evaluate the goodness of fit of a regression model. It provides information about how well the model explains the variation in the dependent variable. R-squared is a value between 0 and 1, with higher values indicating a better fit of the model to the data.

## **Feedback Loop :**

- Incorporate user feedback into the model's improvement cycle, allowing for adjustments to the feature set, model selection, and hyperparameters.

- Maintain a feedback loop to gather user ratings and incorporate them into the dataset for continuous model improvement.

## **Data Cleaning:**

Data cleaning, also referred to as data cleansing and data scrubbing, is one of the most important steps

Data cleansing is a subprocess of the data science process that focuses on removing errors in the data so that the data becomes a true and consistent representation of the processes it originates from.

The first type is the interpretation error, such as incorrect use of terminologies.

The second type of error points to inconsistencies between data sources or against your company's standardized values.

Data cleaning is the process of fixing or removing

- Corrupted data
- Incorrectly formatted
- Duplicate
- Incomplete data

If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. There is no one absolute way to prescribe the exact steps in the data cleaning process because the processes will vary from dataset to dataset.

### **Joining Tables:**

Joining tables is used to combine the information of one observation found in one table with the information that was found in another table.

### **Appending Tables:**

Appending or stacking tables is effectively adding observations from one table to another table.

### **Transforming Data:**

Certain models require their data to be in a certain shape. So the data can be changed according to our need. Data transformation is the process of converting data from one format or structure into another. Transformation processes can also be referred to as data wrangling, or data munging, transforming and mapping data from one "raw" data form into another format for warehousing and analyzing.

When the dataset is summarized range index, total number of columns, data types, column heading and memory usage are viewed.

### **Feature engineering:**

Feature engineering refers to manipulation — addition, deletion, combination, mutation — of the data set to improve machine learning model training, leading to better performance and greater accuracy.

Creating new features gives a deeper understanding of the data and results in more valuable insights. When done correctly, feature engineering is one of the most valuable techniques of data science, but it is also one of the most challenging.

Feature engineering in ML contains mainly four processes:

- Feature Creation
- Transformations
- Feature Extraction
- Feature Selection

### **Feature Creation:**

Feature creation is finding the most useful variables to be used in a predictive model. The process is subjective, and it requires human creativity and intervention.

### **Transformations:**

The transformation step of feature engineering involves adjusting the predictor variable to improve the accuracy and performance of the model.

### **Feature Extraction:**

Feature extraction is an automated feature engineering process that generates new variables by extracting them from the raw data. The main aim of this step is to reduce the volume of data so that it can be easily used and managed for data modelling. Feature extraction methods include cluster analysis, text analytics, edge detection algorithms, and principal components analysis (PCA).

### **Feature Selection:**

Reducing the input variable to your model by using only relevant data and getting rid of noise in data. It is the process of automatically choosing relevant data and getting rid of noise in data.

### **Model training:**

Model training is the phase in the data science development lifecycle where practitioners try to fit the best combination of weights and bias to a machine learning algorithm to minimize a loss function over the prediction range. Train the selected model using the training dataset. The model learns the relationships between the input features (movie attributes) and the target variable (IMDb scores).

There are 3 steps to train a machine learning model

Step 1: Begin with existing data.

Step 2: Analysis data to identify patterns.

Step 3: Make predictions.

### **Evaluation:**

To evaluate the model's performance, we can use metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE) or R-squared. These metrics measure how well your model's predictions match the actual IMDb scores.

### **PROCEDURE:**

Step 1: The required libraries are imported.

Step2: The given dataset is loaded in the program.

Step 3: To check whether the uploaded dataset is correct, a sample data is displayed in the output.

Step 4: The information from the dataset is summarized for better understanding.

Step 5: Then preprocessing is done, in which missing data, duplicate data are checked.

Step 6: If any modify the data or print False.

Step7:Then the features are extracted from the given data sheet.

Step 8:Split the data into training and testing sets.

Step 9:Initialize the regression algorithm.

Step 10:The set is trained and evaluated.

Step 11:The model is completed.

### **PROGRAM:**

```
import pandas as pd
```

```
import numpy as np
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.ensemble import GradientBoostingRegressor

# loading the data set
data = pd.read_csv("NetflixOriginals.csv",encoding='latin-1')
print('\n')

print('DEVELOPMENT PART-1:')
print('-----')
print(data.columns)
print('\n INFORMATION:')
print('-----\n')
print(data.head(10))

# getting the information about the data for better understanding
print('\nSUMMARY')
print('----- \n')
print(data.info())
print('\nDUPLICATE DATA:')
print('-----')

# checking for any duplicate data in the data set
print(data.duplicated().any())

# encoding the categorical features
label_encoder = LabelEncoder()
```



```

data['Genre'] = label_encoder.fit_transform(data['Genre'])

data['Language'] = label_encoder.fit_transform(data['Language'])

# encoding the premiere data since they contain month date, year and month date. year
data['Premiere_comma'] = pd.to_datetime(data['Premiere'], format='%B %d, %Y',
errors='coerce')

data['Premiere_period'] = pd.to_datetime(data['Premiere'], format='%B %d. %Y',
errors='coerce')

# Merge the two columns into a single datetime column
data['Premiere'] = data['Premiere_comma'].combine_first(data['Premiere_period'])

# Drop the temporary columns
data.drop(['Premiere_comma', 'Premiere_period'], axis=1, inplace=True)

# Convert valid datetime values to timestamps and handle NaT values
data['Premiere'] = data['Premiere'].apply(lambda x: x.timestamp() if not pd.isna(x) else
None)

data.head()

print('\n MISSING DATA:')
print('-----')

# checking for any missing data in the premiere
print(data.isnull().values.any())

print('\n')
print('DEVELOPMENT PART 2')
print('-----\n')

# Split data into features (X) and target (y)
X = data[['Genre','Premiere','Runtime', 'Language']]
y = data['IMDB Score']

```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print('Linear Regression model')
print('-----\n')

# Choose a regression algorithm
model = LinearRegression()

# Initialize the Gradient Boosting Regressor
gb_regressor = GradientBoostingRegressor()

# Model Training
model.fit(X_train, y_train)

# Fit the model to your training data
gb_regressor.fit(X_train, y_train)

# Model Evaluation
y_pred = model.predict(X_test)

# Make predictions
predictions = gb_regressor.predict(X_test)

# by using LinearRegression method
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Absolute Error:", mae)
```

```
print("Mean Squared Error:", mse)
```

```
print("R-squared:", r2)
```

```
print('\nGradient Boosting Regressor')
```

```
print('-----\n')
```

```
# by using Gradient Boosting Regressor
```

```
mae = mean_absolute_error(y_test, predictions)
```

```
mse = mean_squared_error(y_test, predictions)
```

```
r2 = r2_score(y_test, predictions)
```

```
print(f'Mean Absolute Error: {mae}')
```

```
print(f'Mean Squared Error: {mse}')
```

```
print(f'R-squared: {r2}')
```

**OUTPUT:**

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

DEVELOPMENT PART-1:

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

INFORMATION:

	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
5	Drive	Action	November 1, 2019	147	3.5	Hindi
6	Leyla Everlasting	Comedy	December 4, 2020	112	3.7	Turkish
7	The Last Days of American Crime	Heist film/Thriller	June 5, 2020	149	3.7	English
8	Paradox	Musical/Western/Fantasy	March 23, 2018	73	3.9	English
9	Sardar Ka Grandson	Comedy	May 18, 2021	139	4.1	Hindi

SUMMARY

<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  
None

DUPLICATE DATA:

False

MISSING DATA:

False

DEVELOPMENT PART 2

Linear Regression model

Mean Absolute Error: 0.8155336353328344  
Mean Squared Error: 0.994440830285098  
R-squared: 0.04191044117818454

Gradient Boosting Regressor

Mean Absolute Error: 0.7311737159548151  
Mean Squared Error: 0.8821936141014356  
R-squared: 0.1500545182888865