**PROJECT TITLE: PREDICTING IMDB SCORES**

**(PHASE 5)**

# Team Members:

* *GEETIKA.R-715521106014*
* *MIRUTHULA.P.V-715521106029*
* *SRI KARTHICK RAJAN.N-715521106050*
* *THANUJ.R-715521106052*
* *THARUN SRIPAL.A.D-715521106053*
* *HARSHAN.B-715521106304*

# PROBLEM STATEMENT:

The problem at hand is to develop a machine learning model that can accurately predict the IMDb scores of movies based on several key attributes, including genre, premiere date, runtime, and language. IMDb scores represent the perceived quality and popularity of movies, making this prediction task valuable for assisting users in discovering high-rated films that align with their preferences.

# DATASET:

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

The data set includes details about the film's title, genre, Premiere, runtime (the length of the film), IMDb ratings, and language.

# DESCRIPTION

## importing necessary libraries:

* + numpy for numerical operations.
  + pandas for data manipulation and analysis.
  + matplotlib for data visualization.
  + seaborn for additional data visualization capabilities.
  + datetime for working with date-related data.
  + Import specific modules like lines and gridspec from matplotlib.

## Loading the dataset:

* + Reads a CSV file called "NetflixOriginals.csv" using pandas and stores it in a DataFrame named ds. The file is assumed to be encoded in "ISO-8859-1".

## Data Preprocessing:

* + A copy of the original DataFrame (ds) is created as ds\_date for further processing.

## Initial Data Exploration:

* + Displays the first few rows of the dataset using ds.head().
  + Provides summary statistics for the dataset using ds.describe().T.
  + Shows information about the dataset, including data types and non-null counts, using ds.info(verbose=True, show\_counts=True.

## Handling Missing Data:

* + Calculates the count of missing values in each column using ds.isna().sum().

## Exploring Categorical Columns:

* + Counts the occurrences of unique values in the 'Title', 'Genre', and 'Premiere' columns using ds['Title'].value\_counts(), ds['Genre'].value\_counts(), and ds['Premiere'].value\_counts().

## Date Manipulation:

* + Transforms the 'Premiere' column into a more suitable date format. It replaces dots with commas and extracts the premiere date and year.
  + Converts the 'PremiereDate' column to a proper datetime format and stores the year information in a new column 'Year'.

## Further Data Exploration:

* + Displays information about the updated ds\_date DataFrame using ds\_date.info().

## More Categorical Exploration:

* + Counts the occurrences of unique values in the 'Language' and 'Genre' columns again using ds['Language'].value\_counts().

## Filtering English Language Shows:

* Filters the dataset to include only English-language shows and sorts them by 'IMDB Score' in descending order using ds\_english.

## Machine Learning Preparation:

* Sets up the features (X) and the target variable (y) for a machine learning model.
* Splits the data into training and testing sets using train\_test\_split.

## Feature Scaling:

* Standardizes the numeric features using StandardScaler.

## Categorical Data Transformation:

* Defines the categorical columns ('Genre' and 'Language') that need one-hot encoding.
* Creates a ColumnTransformer to apply one-hot encoding to the categorical columns and leaves other columns as-is.
* Transforms the training and testing data using the defined preprocessor.

## Model Building and Training:

* Creates a Random Forest Regressor model with 100 trees.
* Fits the model to the training data.

## Model Evaluation:

* Makes predictions on the test data.
* Calculates Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2) to evaluate the model's performance. Results are printed to the console.

### Source code:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import matplotlib.lines as lines

import matplotlib.gridspec as gridspec

import seaborn as sns

ds = pd.read\_csvds = pd.read\_csv("NetflixOriginals.csv",encoding = "ISO-8859-1")

ds\_date = ds.copy()

ds.head()

ds.describe().T

ds.info(verbose=True,show\_counts=True)

ds.isna().sum()

ds['Title'].value\_counts()

ds['Genre'].value\_counts()

ds['Premiere'].value\_counts()

from datetime import datetime

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

ds\_date.info()

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

ds['Genre'].value\_counts()

genre = ds['Genre'].value\_counts()

genre.head()

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

ds\_english = ds[ds['Language'] == 'English'].sort\_values('IMDB Score', ascending=False)

ds\_english.head()

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

# Specify your features (X) and target variable (y)

# Replace 'X' and 'y' with your actual data

X = ds\_date[['Title','Genre','Premiere','Runtime','Language']] # Replace with your features

y = ds\_date['IMDB Score'] # Replace with your target variable

# Split the data into training and testing sets (e.g., 80% train, 20% test)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

print('X\_train',X\_train)

print('X\_test',X\_train)

print('y\_train',X\_train)

print('y\_train',X\_train)

from sklearn.preprocessing import StandardScaler

# Select only the numeric features (excluding 'Title')

numeric\_features = X\_train.select\_dtypes(include=[np.number])

# Create a StandardScaler instance

scaler = StandardScaler()

# Fit the scaler on the training data and transform both training and testing data

X\_train\_scaled = scaler.fit\_transform(numeric\_features)

X\_test\_scaled = scaler.transform(X\_test[numeric\_features.columns])

from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler

from sklearn.compose import ColumnTransformer

categorical\_cols = ['Genre', 'Language']

# Create a ColumnTransformer to apply one-hot encoding

preprocessor = ColumnTransformer(

transformers=[

('cat', OneHotEncoder(), categorical\_cols),

],

remainder='passthrough' # Include other columns as-is

)

# Fit and transform the training data

X\_train = preprocessor.fit\_transform(X\_train)

# Transform the test data using the same preprocessor

X\_test = preprocessor.transform(X\_test)

# Define the target variable

y = ds\_date['IMDB Score']

# Split the data into training and testing

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create a Random Forest Regressor model

model = RandomForestRegressor(n\_estimators=100, random\_state=42)

# Train the model on the training data

model.fit(X\_train, y\_train)

# Make predictions on the test data

y\_pred = model.predict(X\_test)

# Evaluate the model

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f"Mean Absolute Error (MAE): {mae}")

print(f"Mean Squared Error (MSE): {mse}")

print(f"R-squared (R^2): {r2}")

# Phases of development:

## 1. Data Exploration and Preprocessing:

- In this initial phase, the code loads the dataset, examines its structure, and performs data preprocessing tasks. This includes looking at basic statistics, checking for missing values, and exploring categorical variables such as 'Genre,' 'Language,' and 'Premiere.' The code also transforms the date information from the 'Premiere' column into a more usable format and handles missing data. Data quality and initial understanding are crucial in this phase.

## 2. \*\*Data Analysis and Visualization:

- Although not explicitly mentioned in the provided code, this phase typically involves data visualization and analysis to gain insights into the dataset. It can include creating various plots and charts to visualize relationships between variables, distribution of IMDB scores, and other relevant aspects. Visualization libraries like `matplotlib` and `seaborn` can be used for this purpose.

## 3. \*\*Data Preparation for Machine Learning:\*\*

- In this phase, the code prepares the data for machine learning. It selects features and a target variable ('IMDB Score') and splits the data into training and testing sets. Numeric features are scaled using `StandardScaler` to ensure that features have the same scale. Categorical data is transformed, specifically one-hot encoding for 'Genre' and 'Language' columns, making them suitable for machine learning models.

## 4. \*\*Model Building and Training:\*\*

- The code defines a machine learning model, specifically a Random Forest Regressor, and trains it using the training data. This phase involves setting up the machine learning model, specifying hyperparameters, and fitting the model to the training data.

## 5. \*\*Model Evaluation and Reporting:\*\*

- In the final phase, the code evaluates the machine learning model's performance on the test data. It calculates and reports key evaluation metrics, including Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2). These metrics provide an assessment of how well the model is performing in predicting IMDB scores. The evaluation results help in understanding the model's effectiveness and potential areas for improvement.

# Choice of a Machine Learning Algorithm:

## 1. \*\*Random Forest Regressor:\*\*

- Random forests are a good starting point for regression tasks. They are an ensemble method that combines multiple decision trees to make predictions. They often perform well without much hyperparameter tuning and are less prone to overfitting.

## 2. \*\*Gradient Boosting Regressor (e.g., XGBoost, LightGBM, CatBoost):\*\*

- Gradient boosting methods, like XGBoost, LightGBM, and CatBoost, are powerful for regression tasks. They sequentially build decision trees, allowing for improved predictive accuracy. They are highly customizable and often deliver state-of-the-art results.

## 3. \*\*Linear Regression:\*\*

- Linear regression can be a simple yet effective choice if the relationships between features and IMDB scores are largely linear. It provides interpretable coefficients for each feature.

## 4. \*\*Support Vector Regression (SVR):\*\*

- SVR can be useful if you believe that the relationship between features and IMDB scores is nonlinear and you want to find a hyperplane that best fits the data in a high-dimensional space.

## 5. \*\*Neural Networks (e.g., Multi-layer Perceptron):\*\*

- If your dataset is large and complex, neural networks can be employed. They can capture intricate patterns and relationships in the data. However, they often require more data and extensive hyperparameter tuning.

## 6. \*\*Ridge or Lasso Regression:\*\*

- Ridge and Lasso are regularization techniques used in linear regression to handle multicollinearity and overfitting. They can be beneficial when you have a lot of features.