***PROJECT TITLE: PREDICTING IMDB SCORES***

**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 DEFINITION**

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/>

**INNOVATION**

1**.Data Collection:**

- Obtain a dataset containing information about movies, including genre, premiere date, runtime, language, and their corresponding IMDb scores. You can use IMDb or other movie databases, APIs, or web scraping tools to collect this data.

***Innovation:*** The project begins with the collection of a comprehensive dataset about movies. While traditional methods involve scraping data from sources like IMDb, innovative approaches could include using advanced web scraping techniques, APIs from multiple movie databases, or even employing machine learning models to curate more up-to-date and extensive movie information. This innovation ensures that the dataset is richer and constantly updated, improving the accuracy of IMDb score predictions

***Libraries used:***

requests: For making HTTP requests and web scraping movie data from websites.

Beautiful Soup or Scrapy: Python libraries for parsing and navigating HTML/XML documents, essential for web scraping.

**2. Data Preprocessing:**

- Clean the dataset by handling missing values, duplicates, and outliers.

- Convert categorical variables like genre and language into numerical representations using techniques like one-hot encoding or label encoding.

- Normalize or scale numerical features like premiere date and runtime to ensure they have a similar scale.

***Innovation***: Data preprocessing is a crucial step in ensuring data quality. Beyond handling missing values, duplicates, and outliers, this stage could benefit from advanced techniques like natural language processing (NLP). NLP can be applied for sentiment analysis on movie reviews, thereby adding a sentiment-based feature to the dataset. Additionally, unstructured data, such as movie posters or plot summaries, can be processed to extract valuable information and enrich the dataset further.

***Libraries used:***

pandas: For data manipulation and cleaning, handling missing values, duplicates, and outliers.

NumPy: Used for numerical operations and array handling.

scikit-learn: Provides tools for data preprocessing, such as label encoding, scaling, and feature extraction.

**3. Feature Selection:**

- Analyze the importance of each feature to the target variable (IMDb score) and select the most relevant ones. You can use feature importance techniques or domain knowledge.

***Innovation:*** Determining which features are most relevant to predicting IMDb scores is essential. Innovative techniques can include automated feature selection methods that consider a broader range of interactions and dependencies between features. Advanced feature engineering through deep learning, such as creating neural network-based embeddings, can capture intricate relationships and improve prediction accuracy

***Libraries used:***

scikit-learn: Offers various feature selection techniques and tools for feature engineering.

Feature-engine: A library for feature engineering, including handling categorical data.

**4. Data Splitting:**

- Split the dataset into training and testing sets. A common split ratio is 80% for training and 20% for testing.

***Libraries used:***

scikit-learn: Provides functions for splitting data into training and testing sets.

**5. Model Selection:**

- Choose an appropriate machine learning regression algorithm for the task. Common choices include Linear Regression, Random Forest Regression, or Gradient Boosting Regression.

***Innovation:*** The choice of a regression model for IMDb score prediction can go beyond traditional methods like Linear Regression or Random Forest. Innovations might involve exploring state-of-the-art regression models or deep learning architectures like neural collaborative filtering, which can take into account user-item interactions and offer more accurate predictions.

***Libraries used:***

scikit-learn: Contains a wide range of machine learning models, including regression models like Linear Regression, Random Forest Regression, and Gradient Boosting Regression.

XGBoost, LightGBM, and CatBoost: Specialized libraries for gradient boosting algorithms.

**6. Model Training**

-Train the selected model on the training dataset. The model will learn the relationships between the features and the IMDb scores.

***Innovation:*** Training models on large datasets can be resource-intensive. Leveraging distributed computing, GPU acceleration, or cloud-based machine learning platforms can expedite training times and allow for the development of large-scale models efficiently.

***Libraries used:***

scikit-learn: Used for training machine learning models.

**7. Model Evaluation:**

- Evaluate the model's performance on the testing dataset using appropriate regression metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R2) to assess how well it predicts IMDb scores.

***Innovation:*** Utilizing custom evaluation metrics or ensemble techniques to improve the model's performance, and employing A/B testing to compare different model versions in a real-world scenario

***Innovation***: Evaluating the model's performance is typically done using standard metrics. Innovation can be introduced by defining custom evaluation metrics tailored to specific project goals. Additionally, employing A/B testing to compare different versions of the model in a real-world scenario can provide insights into real-world performance.

***Libraries used***:

scikit-learn: Offers various metrics for regression evaluation, such as MAE, MSE, and R-squared.

yellowbrick: A visualization library that can help evaluate model performance.

**8. Hyperparameter Tuning:**

- Optimize the model's hyperparameters to improve its performance. You can use techniques like grid search or random search.

***Innovation:*** Beyond traditional hyperparameter tuning methods like grid search or random search, more advanced techniques like Bayesian optimization or reinforcement learning-based approaches can be applied to efficiently fine-tune the model's parameters.

***Libraries used:***

scikit-learn: Grid Search and Random Search can be used for hyperparameter tuning.

BayesianOptimization: A library for Bayesian optimization.

Optuna: An optimization framework for machine learning hyperparameters.

**9. Model Interpretability :**

- If necessary, analyze the model's feature importances or coefficients to understand which features have the most influence on IMDb scores.

***Innovation:*** Understanding why the model makes specific predictions is vital. Implementing explainability techniques like LIME or SHAP values provides insights into the model's decision-making process, enhancing trust and transparency in its predictions.

***Libraries used:***

shap: A library for explaining the output of machine learning models.

lime: Used for local interpretable model-agnostic explanations.

**10. Deployment:**

- Once satisfied with the model's performance, deploy it as an application or API that users can interact with to predict IMDb scores for movies.

***Innovation:*** Deploying the model can involve containerization with technologies like Docker or serverless computing for scalability and flexibility. This ensures that the model can handle variable loads and remains accessible to users.

***Libraries used:***

Flask or FastAPI: Python web frameworks for building and deploying APIs.

Docker: For containerizing the application.

**11. User Interface (UI) Development:**

- Create a user-friendly interface where users can input movie details (genre, premiere date, runtime, language) and receive IMDb score predictions.

***Innovation:*** Creating a user-friendly interface with modern web development frameworks can enhance user engagement. Incorporating real-time visualization or interactive elements can make the UI more engaging and informative for users.

***Libraries used:***

HTML, CSS, and JavaScript: For building the front-end of the user interface.

Web development frameworks like React, Angular, or Vue.js: If you're building a complex front-end.

**12. Testing and Maintenance:**

- Continuously test the model in a real-world environment and monitor its performance. Retrain the model periodically with updated data to maintain its accuracy.

***Innovation:*** Employing automated testing, continuous integration, and deployment (CI/CD) pipelines ensures that the model's performance remains robust and up-to-date. Continuous testing in a real-world environment helps identify and resolve issues promptly.

***Libraries used:***

pytest: A testing framework for writing and running tests.

Continuous Integration (CI) and Continuous Deployment (CD) tools like Jenkins, Travis CI, or CircleCI.

**13. Feedback Loop:**

- Incorporate user feedback to improve the model's recommendations and predictions over time.

***Innovation:*** Building feedback mechanisms into the application to capture user feedback and usage patterns allows for further model improvement and personalization over time. This iterative feedback loop ensures that the model remains adaptive to changing user preferences and movie trends, enhancing its utility and accuracy.

***Libraries used:***

Custom code for capturing user feedback and integrating it into the model's improvement process.

**DETAILS OF COLUMNS:**

Below are the columns you would need from such a dataset:

1. **Title**: This column is essential for identifying the movies in the dataset.

2. **Genre**: Movie genre is crucial in understanding the type of content a movie belongs to, and it influences IMDb scores.

3. **Premiere Date**: Knowing when a movie was released is important, as the time of release can impact its IMDb score.

4. **Runtime**: The movie's runtime, in minutes, is a numerical feature that can affect audience engagement and perception.

5. **Language**: The primary language in which the movie is spoken or produced. Language can determine the accessibility of a movie to a wider or more niche audience.

6. **IMDb Score (or a Similar Rating)**: This is the target variable you aim to predict. You will use IMDb scores or a similar rating from the dataset as the outcome variable for your machine learning model.

**HOW TO TEST AND TRAIN:**

**Import Libraries:**

In this program, we start by importing the necessary Python libraries. These include:

pandas for data manipulation and handling the dataset.

train\_test\_split from sklearn.model\_selection for splitting the dataset into training and testing sets.

DecisionTreeRegressor from sklearn.tree to create a Decision Tree regression model.

mean\_absolute\_error, mean\_squared\_error, and r2\_score from sklearn.metrics to evaluate the model.

**Load the Dataset:**

The dataset containing information about movies, including features like Genre, Premiere Date, Runtime, Language, and IMDb Score, is loaded using pd.read\_csv('your\_dataset.csv'). Replace 'your\_dataset.csv' with the actual path to your dataset file.

**Define Features and Target:**

The features (independent variables) are selected and stored in X, and the target variable (IMDb Score) is stored in y.

**Data Preprocessing :**

Before splitting and modeling, you should perform any necessary data preprocessing, including handling missing values, feature encoding (if the data contains categorical variables), and feature scaling. The code assumes that you have already prepared your data.

**Split the Data:**

The dataset is divided into a training set and a testing set using train\_test\_split. In this code, the testing set contains 20% of the data, and random\_state is set for reproducibility.

**Create a Decision Tree Model:**

A Decision Tree regression model is created using DecisionTreeRegressor and assigned to the variable model.

**Train the Model:**

The model is trained on the training data using the fit method. It learns the patterns and relationships between the features (Genre, Premiere Date, Runtime, Language) and IMDb Scores from the training set.

**Make Predictions:**

After training, the model is used to make predictions on the testing data with the predict method. Predicted IMDb Scores are stored in the variable predictions.

**CODE:**

# Import necessary libraries

import pandas as pd

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

from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

# Load your dataset

# Replace 'your\_dataset.csv' with the path to your dataset file

data = pd.read\_csv('NetflixOriginals.csv')

# Define your features (X) and target (y)

X = data[['Genre', 'Premiere Date', '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)

# Create a Decision Tree Regression model

model = DecisionTreeRegressor(random\_state=42)

# Train the model on the training data

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

# Make predictions on the testing data

predictions = model.predict(X\_test)

**EVALUATION METRICS:**

**1**.**Mean Absolute Error (MAE):**

Definition: MAE measures the average absolute difference between the predicted IMDb scores and the actual IMDb scores in the testing set.

Interpretation: A lower MAE indicates that the model's predictions are closer to the actual values. MAE is easy to understand since it represents the average

prediction error in the original units (IMDb scores). For example, an MAE of 1.5 means, on average, the model's predictions are off by 1.5 IMDb points.

Formula: MAE = Σ(|actual - predicted|) / number of samples

**2.Mean Squared Error (MSE):**

Definition: MSE measures the average squared difference between the predicted IMDb scores and the actual IMDb scores in the testing set.

Interpretation: Like MAE, a lower MSE indicates that the model's predictions are closer to the actual values. However, MSE gives more weight to larger errors and can be sensitive to outliers.

Formula: MSE = Σ(actual - predicted)^2 / number of samples

**3.R-squared (R2):**

Definition: R2, also known as the coefficient of determination, measures the proportion of variance in the target variable (IMDb scores) explained by the model. It ranges from 0 to 1, where 0 indicates that the model explains none of the variance, and 1 indicates that the model explains all the variance.

Interpretation: An R2 value closer to 1 indicates that the model is a good fit for the data and explains a high percentage of the variance. An R2 value of 0 means that the model doesn't explain the variance and essentially predicts the mean IMDb score for all movies.

Formula: R2 = 1 - (MSE(model) / MSE(mean))