PREDICTING IMDb SCORE

TEAM MEMBER

732121104053:RANJANADEVI C



# Problem Statement

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.

# Procedure:

1. \*Import Required Libraries\*:

First, import the necessary libraries:

Python

import pandas as pd

import numpy as np

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

from sklearn.preprocessing import LabelEncoder, StandardScaler

2. \*Load the Dataset\*:

Load the dataset into a Pandas DataFrame:

python

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

3. \*Data Exploration\*:

Before preprocessing, it's a good practice to explore the dataset to understand its structure. You can use functions like `data.head()` to see the first few rows of the dataset, and `data.info()` to get information about the data types of each column.

4. \*Data Preprocessing\*:

Preprocessing is crucial for preparing the data for analysis and modeling. Here are some common preprocessing steps:

a. \*Handling Missing Values\*:

Check for missing values and decide how to handle them. You can use `data.isnull().sum()` to check for missing values in each column and decide whether to fill them in or drop rows with missing values.

b. \*Feature Selection\*:

Decide which features (columns) to use as input for the IMDb score prediction model. You might want to remove columns that are not relevant.

c. \*Encoding Categorical Variables\*:

If your dataset contains categorical variables (e.g., "Rating" or "Genre"), you'll need to encode them. For example, you can use `LabelEncoder` from Scikit-Learn to convert categorical variables into numerical values.

d. \*Feature Scaling\*:

If your dataset has numerical features with different scales, it's a good idea to standardize them using `StandardScaler` from Scikit-Learn.

e. \*Splitting the Data\*:

Split the dataset into training and testing sets for model training and evaluation.

Here's a code snippet for these preprocessing steps:

python

# Handling missing values (if needed)

data = data.dropna()

# Feature selection

selected\_features = ["Feature1", "Feature2", ...] # Replace with the actual feature names you want to use

# PROGRAM:

# Specify the encoding as 'ISO-8859-1' when reading the CSV file

data = pd.read\_csv('NetflixOriginals.csv', encoding='ISO-8859-1')

# Now, you can proceed with your data analysis

print(data.head())

print(data.info())

print(data.describe())

pd.set\_option('display.max\_rows', 100)

pd.set\_option('display.max\_columns', 100)

# Display your DataFrame

data

# Check for missing values in each column

missing\_values = data.isnull().sum()

# Display columns with missing values

print(missing\_values[missing\_values > 0])

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

import pandas as pd

# Check the size of your dataset

print(data.shape)

# Split the data into train and test

train\_data, test\_data = train\_test\_split(data, test\_size=0.2, random\_state=42)

# Display the first few rows of the training dataset

print("Training Dataset:")

print(train\_data.head())

# Display the first few rows of the testing dataset

print("\nTesting Dataset:")

print(test\_data.head()) # Encoding categorical variables

label\_encoders = {}

categorical\_columns = ["Rating", "Genre"] # Replace with the actual categorical columns

for col in categorical\_columns:

label\_encoders[col] = LabelEncoder()

data[col] = label\_encoders[col].fit\_transform(data[col])

# Feature scaling

scaler = StandardScaler()

data[selected\_features] = scaler.fit\_transform(data[selected\_features])

# Splitting the data

X = data[selected\_features]

y = data['IMDB Score']

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

# OUTPUT:

# WhatsApp Image 2023-10-25 at 3.39.26 PM.jpegWhatsApp Image 2023-10-25 at 3.28.47 PM.jpegWhatsApp Image 2023-10-25 at 3.39.26 PM (1).jpeg

# Conclusion:

In this simplified example, we loaded the dataset, preprocessed it by handling missing values and selecting features, split the data into training and testing sets, built and trained a linear regression model, and evaluated its performance.

The conclusion for this basic IMDb score prediction model is as follows:

The model's Mean Squared Error (MSE) and R-squared (R2) values give an indication of its predictive performance.

The model's accuracy depends on the features selected, data quality, and the choice of machine learning algorithm. You can further improve the model by considering more features and experimenting with different algorithms.

For a more robust IMDb score prediction model, you may need to explore more advanced regression techniques, feature engineering, and hyperparameter tuning. Additionally, you might consider other factors such as the movie's genre, director, and cast to enhance the predictive power of your model.