Task 02 : MOVIE RATING PREDICTION WITH PYTHON

Description: I have used the movie rating dataset to build a model that predicts the rating of a movie based on genre, director, and actors.

FLOW ANALYSIS:

- Importing Libraries
- Data loading
- Data Understanding
- Data Pre-Processing (Cleaning)
  - Replacing missing values
  - Extracting numerical values
- Spliting training and test data
- Model training -LinearRegression
- Model Evaluation - Prediction

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Importing all the required libraries
import numpy as np
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn.metrics import mean_squared_error, r2_score
```

```
# Dataset Loading
movie_review = pd.read_csv('/content/drive/MyDrive/CodSoft/IMDb Movies India.csv', encoding='latin-1')
```

Data Understanding

```
# Displaying the first 5 rows of the dataset
movie_review.head()
```

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2
0		NaN	NaN	Drama	NaN	NaN	J.S. Randhawa	Manmauji	Birba
1	#Gadhvi (He thought he was Gandhi)	(2019)	109 min	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Ghamande
2	#Homecoming	(2021)	90 min	Drama, Musical	NaN	NaN	Soumyajit Majumdar	Sayani Gupta	Plabita Borthakur

```
# Displaying total rows and columns of the dataset
movie_review.shape
```

```
(15509, 10)
```

```
# It will calculate and display count, mean, std, min, max, 25%, 50% and 75% of numeric columns here only "Rating" column.
movie_review.describe()
```

	Rating
count	7919.000000
mean	5.841621
std	1.381777
min	1.100000

```
# Displaying information regarding datatype, null values of every column
movie_review.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15509 entries, 0 to 15508
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      15509 non-null  object
 1   Year      14981 non-null  object
 2   Duration  7240 non-null   object
 3   Genre     13632 non-null  object
 4   Rating    7919 non-null   float64
 5   Votes     7920 non-null   object
 6   Director  14984 non-null  object
 7   Actor 1   13892 non-null  object
 8   Actor 2   13125 non-null  object
 9   Actor 3   12365 non-null  object
dtypes: float64(1), object(9)
memory usage: 1.2+ MB
```

```
# Checking for null values
movie_review.isna().sum()
```

```
Name           0
Year         528
Duration    8269
Genre       1877
Rating      7590
Votes       7589
Director     525
Actor 1     1617
Actor 2     2384
Actor 3     3144
dtype: int64
```

Data Cleaning

```
movie_review['Duration'] = movie_review['Duration'].str.extract('(\d+)').astype(float)

# Fill missing values with the median value
median_duration = movie_review['Duration'].median()
movie_review['Duration'].fillna(median_duration, inplace=True)


movie_review['Votes'] = pd.to_numeric(movie_review['Votes'], errors='coerce')

# Fill missing values with the mean value
mean_votes = movie_review['Votes'].mean()
movie_review['Votes'].fillna(mean_votes, inplace=True)


# Handling missing values with mean for numerical columns and mode for categorical columns
movie_review['Genre'].fillna(movie_review['Genre'].mode()[0], inplace=True)
movie_review['Rating'].fillna(movie_review['Rating'].mean(), inplace=True)

movie_review['Actor 1'].fillna('Unknown', inplace=True)
movie_review['Actor 2'].fillna('Unknown', inplace=True)
movie_review['Actor 3'].fillna('Unknown', inplace=True)
movie_review['Director'].fillna('Unknown', inplace=True)


# Extract numeric year from the 'Year' column and convert to float
movie_review['Year'] = movie_review['Year'].str.extract('(\d+)').astype(float)

# Fill missing values with the median year or another appropriate strategy
median_year = movie_review['Year'].median()
movie_review['Year'].fillna(median_year, inplace=True)
```

```
movie_review.isnull().sum()
```

```
Name        0
Year        0
Duration    0
Genre       0
Rating      0
Votes       0
Director    0
Actor 1     0
Actor 2     0
Actor 3     0
dtype: int64
```

One Hot Encoding

```
movie_review = pd.get_dummies(movie_review, columns=['Genre'], prefix='Genre')
```

Label Encoding

```
label_encoders = {}
columns_to_label_encode = ['Director', 'Actor 1', 'Actor 2', 'Actor 3']

for column in columns_to_label_encode:
    label_encoder = LabelEncoder()
    movie_review[column] = label_encoder.fit_transform(movie_review[column])
    label_encoders[column] = label_encoder
```

```
# Splitting the training and testing dataset
X = movie_review[['Year', 'Duration'] + list(movie_review.filter(regex='Genre_').columns) + columns_to_label_encode]
y = movie_review['Rating']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Model Training
model = LinearRegression()
model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
#Model Evaluation
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

```
Mean Squared Error: 0.9287783110540945
R-squared: 0.03501283596284133
```