In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
data=pd.read_csv("C:\\Users\\Suvarna\\Downloads\\DATA SETS\\IMDb Movies India.csv",enc
```

In [3]:
```
data
```

Out[3]:

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2
0		NaN	NaN	Drama	NaN	NaN	J.S. Randhawa	Manmauji	Birbal
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
3	#Yaaram	(2019)	110 min	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Ishita Raj
4	...And Once Again	(2010)	105 min	Drama	NaN	NaN	Amol Palekar	Rajat Kapoor	Rituparna Sengupta
...	...	...	...	...	...	...	...	...	...
15504	Zulm Ko Jala Doonga	(1988)	NaN	Action	4.6	11	Mahendra Shah	Naseeruddin Shah	Sumeet Saigal
15505	Zulmi	(1999)	129 min	Action, Drama	4.5	655	Kuku Kohli	Akshay Kumar	Twinkle Khanna
15506	Zulmi Raj	(2005)	NaN	Action	NaN	NaN	Kiran Thej	Sangeeta Tiwari	NaN
15507	Zulmi Shikari	(1988)	NaN	Action	NaN	NaN	NaN	NaN	NaN
15508	Zulm-O-Sitam	(1998)	130 min	Action, Drama	6.2	20	K.C. Bokadia	Dharmendra	Jaya Prada

15509 rows × 10 columns

In [4]:
```
# Fill missing values in 'Rating' with the median value
data['Rating'] = data['Rating'].fillna(data['Rating'].median())

# Ensure all entries in 'Year' are strings, then extract the numeric year
data['Year'] = data['Year'].astype(str).str.extract(r'(\d{4})').astype(float)

# Ensure all values in 'Duration' column are strings, remove ' min' and convert to flo
data['Duration'] = data['Duration'].astype(str).str.replace(' min', '').astype(float)

# Drop rows where critical features are missing (like 'Year' and 'Duration')
data.dropna(subset=['Year', 'Duration'], inplace=True)
```

In [5]:
```
from sklearn.preprocessing import OneHotEncoder
```

In [6]:
```
# One-hot encoding for 'Genre', 'Director', and 'Actors'
categorical_features = ['Genre', 'Director', 'Actor 1', 'Actor 2', 'Actor 3']
encoder = OneHotEncoder(handle_unknown='ignore', sparse=False)
encoded_features = encoder.fit_transform(data[categorical_features])
```

```
C:\Users\Suvarna\anaconda3\Lib\site-packages\sklearn\preprocessing\_encoders.py:972:
FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be rem
oved in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default valu
e.
  warnings.warn(
```

In [7]:
```
# Combine encoded categorical features with numerical features
numerical_features = ['Year', 'Duration']
X = np.concatenate([encoded_features, data[numerical_features].values], axis=1)
# Target variable: 'Rating'
y = data['Rating'].values
```

In [8]:
```
from sklearn.model_selection import train_test_split

# Split the data into 80% training and 20% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
from sklearn.ensemble import RandomForestRegressor
```

In [9]:
```
# Split the data into 80% training and 20% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
from sklearn.ensemble import RandomForestRegressor
```

In [10]:
```
# Initialize the RandomForestRegressor
model = RandomForestRegressor(n_estimators=100, random_state=42,n_jobs=-1)
```

In [11]:
```
# Train the model
model.fit(X_train, y_train)
```

Out[11]:
```
RandomForestRegressor(n_jobs=-1, random_state=42)
```

In [12]:
```
from sklearn.metrics import mean_squared_error, r2_score
```

In [13]:
```
# Predictions
y_pred = model.predict(X_test)
```

In [14]:
```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

In [15]:
```
print(f"Mean Squared Error: {mse}")
print(f"R-Squared: {r2}")
```

```
Mean Squared Error: 1.2684853176387911
R-Squared: 0.18345382130836552
```