In [15]:
```
import pandas as pd
import numpy as np
```

In [16]:
```
import pandas as pd

# Load the CSV with a different encoding
data = pd.read_csv(r"C:\Users\vtu10\Downloads\IMDb Movies India.csv", encoding
```

In [17]:
```
data.head()
```

Out[17]:

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Ac
0		NaN	NaN	Drama	NaN	NaN	J.S. Randhawa	Manmauji	Birbal	Raj E
1	#Gadhvi (He thought he was Gandhi)	(2019)	109 min	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Ghamande	A J
2	#Homecoming	(2021)	90 min	Drama, Musical	NaN	NaN	Soumyajit Majumdar	Sayani Gupta	Plabita Borthakur	Ar
3	#Yaaram	(2019)	110 min	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Ishita Raj	Sid K
4	...And Once Again	(2010)	105 min	Drama	NaN	NaN	Amol Palekar	Rajat Kapoor	Rituparna Sengupta	A

In [18]:
```
#Handle missing values
```

In [19]:
```
import pandas as pd

# Load the dataset
data = pd.read_csv(r"C:\Users\vtu10\Downloads\IMDb Movies India.csv", encoding

# 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 conver
data['Duration'] = data['Duration'].astype(str).str.replace(' min', '').astype

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

In [20]:
```
from sklearn.preprocessing import OneHotEncoder
```

In [21]:
```
#Feature Engineering
```

In [22]:
```
# 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])
```

In [23]:
```
# Combine encoded categorical features with numerical features
numerical_features = ['Year', 'Duration']
X = np.concatenate([encoded_features, data[numerical_features].values], axis=1
```

In [24]:
```
# Target variable: 'Rating'
y = data['Rating'].values
```

In [25]:
```
#Data splitting
```

In [26]:
```
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, rando
```

In [27]:
```
#Model building
```

In [29]:
```
from sklearn.ensemble import RandomForestRegressor

# Initialize the RandomForestRegressor
model = RandomForestRegressor(n_estimators=100, random_state=42,n_jobs=-1)

# Train the model
model.fit(X_train, y_train)
```

Out[29]:
```
RandomForestRegressor(n_jobs=-1, random_state=42)
```

In [30]:
```
from sklearn.metrics import mean_squared_error, r2_score
```

In [31]:
```
y_pred=model.predict(X_test)

#Evaluate the model

mse=mean_squared_error(y_test, y_pred)
r2=r2_score(y_test, y_pred)

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

print(f"R-Squared: {r2}")
```

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

In [ ]:
```

```