In [7]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.impute import SimpleImputer
```

In [10]:
```
df=pd.read_csv('Movies India.csv',encoding='latin-1')
```

In [11]:
```
df.head()
```

Out[11]:

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

In [12]:
```
df.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
```

In [13]:
```
df.shape
```

Out[13]:
```
(15509, 10)
```

In [14]:
```
df.columns
```

Out[14]:
```
Index(['Name', 'Year', 'Duration', 'Genre', 'Rating', 'Votes', 'Director',
       'Actor 1', 'Actor 2', 'Actor 3'],
      dtype='object')
```

In [15]:
```
df.describe()
```

Out[15]:

	Rating
count	7919.000000
mean	5.841621
std	1.381777
min	1.100000
25%	4.900000
50%	6.000000
75%	6.800000
max	10.000000

In [16]:
```
df.isnull().sum() # shows the missing values
```

Out[16]:
```
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
```

In [17]:
```
# Data preprocessing which help to fixed the missing values
df = df.dropna(subset=['Rating', 'Genre', 'Director', 'Actor 1', 'Actor 2', 'Actor 3'])
```

In [18]:
```
# Define features and target variable
X = df[['Genre', 'Director', 'Actor 1', 'Actor 2', 'Actor 3']]
y = df['Rating']
```

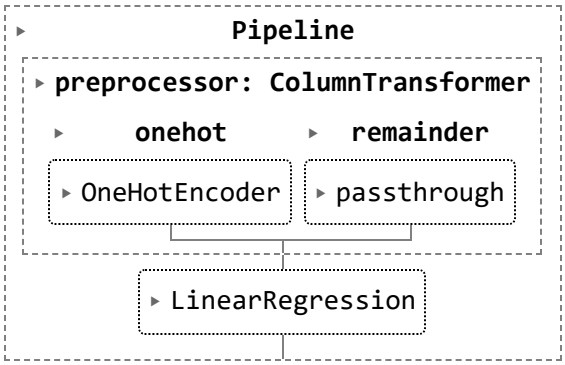
In [19]:
```
# Create a column transformer for one-hot encoding
column_transformer = ColumnTransformer(
    transformers=[
        ('onehot', OneHotEncoder(handle_unknown='ignore'), ['Genre', 'Director', 'Actor
    ],
    remainder='passthrough'
)
```

In [20]:
```
# Create a pipeline
pipeline = Pipeline([
    ('preprocessor', column_transformer),
    ('regressor', LinearRegression())
])
```

In [21]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4
```

In [22]:
```
# Fit the model
pipeline.fit(X_train, y_train)
```

Out[22]:



In [23]:
```
# Make predictions
y_pred = pipeline.predict(X_test)
```

In [24]:
```
# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"\nMean Squared Error: {mse:.4f}")
print(f"Root Mean Squared Error: {rmse:.4f}")
print(f"R^2 Score: {r2:.4f}")
```

```
Mean Squared Error: 6.6158
Root Mean Squared Error: 2.5721
R^2 Score: -2.5706
```

In [25]:
```
# R squared value is negative that means current features are not good predictors of mo
```

In [ ]: