sourcscode

June 12, 2025

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
df = pd.read_csv('netflix_titles.csv')
df.head()
```

[2]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm…
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban…	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi…	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act…	
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K…	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV …	

```
1  After crossing paths at a party, a Cape Town t…
2  To protect his family from a powerful drug lor…
3  Feuds, flirtations and toilet talk go down amo…
4  In a city of coaching centers known to train I…
```

[3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64 
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
```

[ ]:
```
# Checking Missing Values
df.isnull().sum()
```

[ ]:
```
show_id            0
type               0
title              0
director        2634
cast             825
country          831
date_added        10
release_year       0
rating             4
duration           3
listed_in          0
description        0
dtype: int64
```

[5]:
```
# Check unique values per column
df.nunique()
```

[5]:
```
show_id         8807
type               2
title           8807
director        4528
cast            7692
country          748
date_added      1767
release_year      74
rating            17
duration         220
listed_in        514
description     8775
dtype: int64
```

[12]:
```
# Data Cleaning
# Filling missing values

df['country'] = df['country'].fillna('Unknown')
df['rating'] = df['rating'].fillna(df['rating'].mode()[0])
df['duration'] = df['duration'].fillna(df['duration'].mode()[0])
df['date_added'] = df['date_added'].fillna(method='ffill')
```

```
C:\Users\Pranesh\AppData\Local\Temp\ipykernel_17152\321827693.py:7:
FutureWarning: Series.fillna with 'method' is deprecated and will raise in a
future version. Use obj.ffill() or obj.bfill() instead.
  df['date_added'] = df['date_added'].fillna(method='ffill')
```

[13]:
```
# Dropping the unwanted column

df.drop(['description', 'cast', 'director'], axis=1, errors='ignore',␣
 ↪inplace=True)
```

[19]:
```
df.columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   country       8807 non-null   object
 4   date_added    8807 non-null   object
 5   release_year  8807 non-null   int64
 6   rating        8807 non-null   object
 7   duration      8807 non-null   object
```

```
 8   listed_in     8807 non-null   object
dtypes: int64(1), object(8)
memory usage: 619.4+ KB
```

[20]:
```
np.random.seed(42)
df['user_rating'] = np.random.uniform(1.0,10.0,size = len(df))
```

[21]:
```
df.head()
```

[21]:

	show_id	type	title	country	date_added	release_year	rating	duration	listed_in	user_rating
0	s1	Movie	Dick Johnson Is Dead	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	4.370861
1	s2	TV Show	Blood & Water	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	9.556429
2	s3	TV Show	Ganglands	Unknown	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act…	7.587945
3	s4	TV Show	Jailbirds New Orleans	Unknown	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	6.387926
4	s5	TV Show	Kota Factory	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV …	2.404168

[22]:
```
df['user_rating'] = df['user_rating'].round(1)
```

[25]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   country       8807 non-null   object
 4   date_added    8807 non-null   object
 5   release_year  8807 non-null   int64
 6   rating        8807 non-null   object
 7   duration      8807 non-null   object
 8   listed_in     8807 non-null   object
```

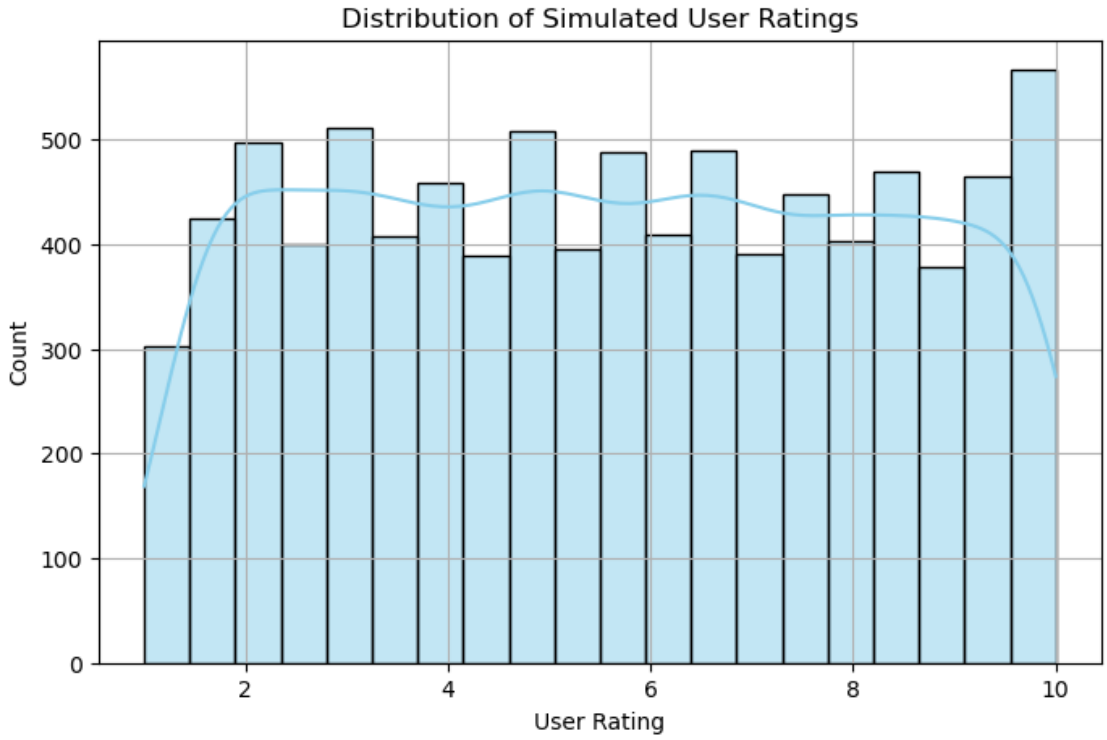
```
 9   user_rating   8807 non-null   float64
dtypes: float64(1), int64(1), object(8)
memory usage: 688.2+ KB
```

[26]:
```
# Boost TV Shows a bit
df.loc[df['type'] == 'TV Show', 'user_rating'] += 0.3

# Boost if the genre contains 'Drama'
df.loc[df['listed_in'].str.contains('Drama', case=False, na=False),␣
 ↪'user_rating'] += 0.2

# Reduce rating a bit if the release year is before 2000
df.loc[df['release_year'] < 2000, 'user_rating'] -= 0.4

# Clip values to stay in 1.0 - 10.0
df['user_rating'] = df['user_rating'].clip(1.0, 10.0)
```

[27]:
```
# Final check
df[['type', 'release_year', 'listed_in', 'user_rating']].head()
```

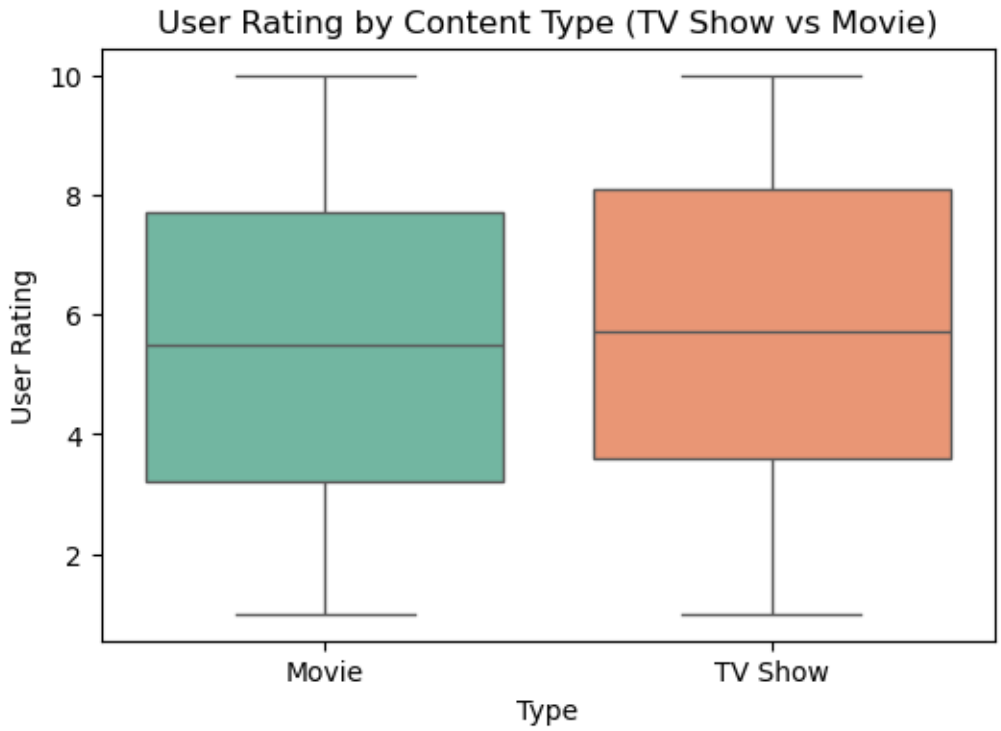
[27]:
```
      type  release_year                                          listed_in  \
0    Movie          2020                                      Documentaries
1  TV Show          2021    International TV Shows, TV Dramas, TV Mysteries
2  TV Show          2021  Crime TV Shows, International TV Shows, TV Act…
3  TV Show          2021                              Docuseries, Reality TV
4  TV Show          2021  International TV Shows, Romantic TV Shows, TV …

   user_rating
0          4.4
1         10.0
2          7.9
3          6.7
4          2.7
```

[30]:
```
# EDA Visualizations

plt.figure(figsize=(8, 5))
sns.histplot(df['user_rating'], bins=20, kde=True, color='skyblue')
plt.title('Distribution of Simulated User Ratings')
plt.xlabel('User Rating')
plt.ylabel('Count')
plt.grid(True)
plt.show()
```



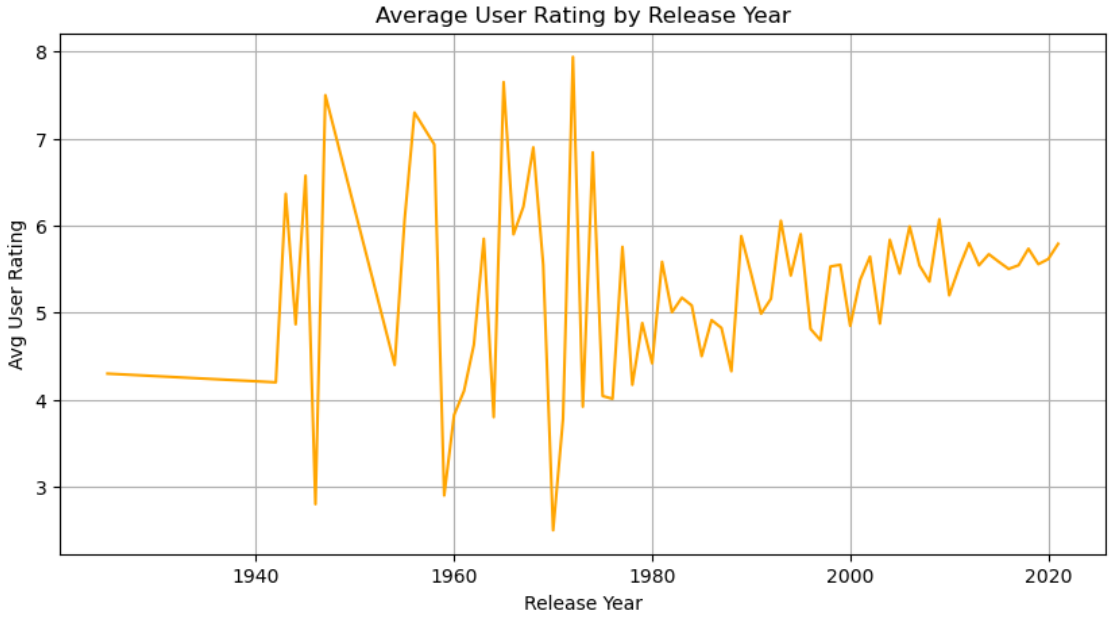
[31]:
```
plt.figure(figsize=(6, 4))
sns.boxplot(x='type', y='user_rating', data=df, palette='Set2')
plt.title('User Rating by Content Type (TV Show vs Movie)')
plt.xlabel('Type')
plt.ylabel('User Rating')
plt.show()
```

```
C:\Users\Pranesh\AppData\Local\Temp\ipykernel_17152\907826974.py:2:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='type', y='user_rating', data=df, palette='Set2')
```



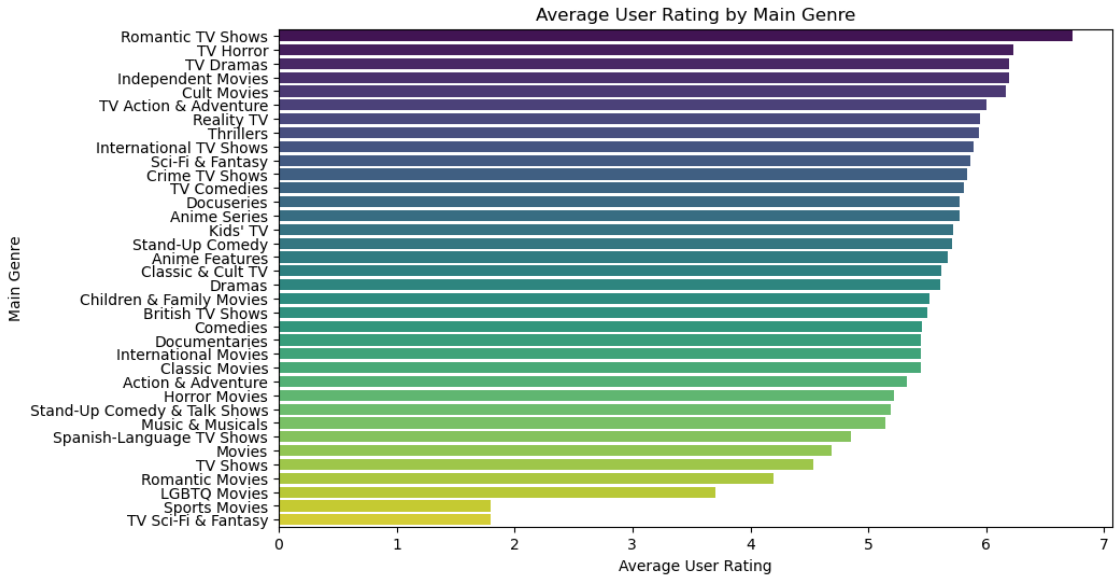
[32]:
```
plt.figure(figsize=(10, 5))
sns.lineplot(data=df, x='release_year', y='user_rating', estimator='mean',␣
↪ci=None, color='orange')
plt.title('Average User Rating by Release Year')
plt.xlabel('Release Year')
plt.ylabel('Avg User Rating')
plt.grid(True)
plt.show()
```

```
C:\Users\Pranesh\AppData\Local\Temp\ipykernel_17152\4020409434.py:2:
FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.lineplot(data=df, x='release_year', y='user_rating', estimator='mean',
ci=None, color='orange')
```



[33]:
```
# Extract the first genre only (to simplify)
df['main_genre'] = df['listed_in'].apply(lambda x: x.split(',')[0])

# Calculate average rating by main genre
genre_ratings = df.groupby('main_genre')['user_rating'].mean().
 ↪sort_values(ascending=False)

# Plot the bar chart
plt.figure(figsize=(10, 6))
sns.barplot(x=genre_ratings.values, y=genre_ratings.index, palette='viridis')
plt.title('Average User Rating by Main Genre')
plt.xlabel('Average User Rating')
plt.ylabel('Main Genre')
plt.show()
```

```
C:\Users\Pranesh\AppData\Local\Temp\ipykernel_17152\460016540.py:9:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=genre_ratings.values, y=genre_ratings.index, palette='viridis')
```



[34]:
```
#  PHASE II : FEATURE ENGINEERING
```

[36]:
```
y = df['user_rating']
X = df[['type', 'release_year', 'duration', 'main_genre']]
```

[37]:
```
y.head()
```

[37]:
```
0     4.4
1    10.0
2     7.9
3     6.7
4     2.7
Name: user_rating, dtype: float64
```

[38]:
```
X.head()
```

[38]:

	type	release_year	duration	main_genre
0	Movie	2020	90 min	Documentaries
1	TV Show	2021	2 Seasons	International TV Shows
2	TV Show	2021	1 Season	Crime TV Shows
3	TV Show	2021	1 Season	Docuseries
4	TV Show	2021	2 Seasons	International TV Shows

[42]:
```
X = df[['type', 'release_year', 'duration', 'main_genre']].copy()
X['type'] = X['type'].map({'Movie': 0, 'TV Show': 1})
```

[43]:
```
X['type'].unique()
```

[43]:
```
array([0, 1], dtype=int64)
```

[49]:
```
# Converting the "duration" to numeric values because ML model can't use text.

def convert_duration(val):
    if 'Season' in val:
        return int(val.split()[0]) * 90     # assuming ~90 mins per season
    else:
        return int(val.split()[0])          # just the minutes for movies
    return 0
X['duration'] = df['duration'].apply(convert_duration)
```

[50]:
```
X.head()
```

[50]:

	type	release_year	duration	main_genre
0	0	2020	90	Documentaries
1	1	2021	180	International TV Shows
2	1	2021	90	Crime TV Shows
3	1	2021	90	Docuseries
4	1	2021	180	International TV Shows

[51]:
```
genre_dummies = pd.get_dummies(X['main_genre'], prefix = 'genre')
X = pd.concat([X.drop('main_genre', axis = 1), genre_dummies], axis = 1)
```

[52]:
```
X.head()
```

[52]:

	type	release_year	duration	genre_Action & Adventure	\
0	0	2020	90	False	
1	1	2021	180	False	
2	1	2021	90	False	
3	1	2021	90	False	
4	1	2021	180	False	

	genre_Anime Features	genre_Anime Series	genre_British TV Shows	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	False	False	

	genre_Children & Family Movies	genre_Classic & Cult TV	\
0	False	False	
1	False	False	
2	False	False	
3	False	False	
4	False	False	

```
   genre_Classic Movies  …  genre_Sports Movies  genre_Stand-Up Comedy  \
0                 False  …                False                  False
1                 False  …                False                  False
2                 False  …                False                  False
3                 False  …                False                  False
4                 False  …                False                  False

   genre_Stand-Up Comedy & Talk Shows  genre_TV Action & Adventure  \
0                               False                         False
1                               False                         False
2                               False                         False
3                               False                         False
4                               False                         False

   genre_TV Comedies  genre_TV Dramas  genre_TV Horror  \
0              False            False            False
1              False            False            False
2              False            False            False
3              False            False            False
4              False            False            False

   genre_TV Sci-Fi & Fantasy  genre_TV Shows  genre_Thrillers
0                      False           False            False
1                      False           False            False
2                      False           False            False
3                      False           False            False
4                      False           False            False

[5 rows x 39 columns]
```

[53]:
```
X = X.astype(int)
```

[54]:
```
X.dtypes
```

[54]:
```
type                                int32
release_year                        int32
duration                            int32
genre_Action & Adventure            int32
genre_Anime Features                int32
genre_Anime Series                  int32
genre_British TV Shows              int32
genre_Children & Family Movies      int32
genre_Classic & Cult TV             int32
genre_Classic Movies                int32
genre_Comedies                      int32
genre_Crime TV Shows                int32
genre_Cult Movies                   int32
```

```
genre_Documentaries                    int32
genre_Docuseries                       int32
genre_Dramas                           int32
genre_Horror Movies                    int32
genre_Independent Movies               int32
genre_International Movies             int32
genre_International TV Shows           int32
genre_Kids' TV                         int32
genre_LGBTQ Movies                     int32
genre_Movies                           int32
genre_Music & Musicals                 int32
genre_Reality TV                       int32
genre_Romantic Movies                  int32
genre_Romantic TV Shows                int32
genre_Sci-Fi & Fantasy                 int32
genre_Spanish-Language TV Shows        int32
genre_Sports Movies                    int32
genre_Stand-Up Comedy                  int32
genre_Stand-Up Comedy & Talk Shows     int32
genre_TV Action & Adventure            int32
genre_TV Comedies                      int32
genre_TV Dramas                        int32
genre_TV Horror                        int32
genre_TV Sci-Fi & Fantasy              int32
genre_TV Shows                         int32
genre_Thrillers                        int32
dtype: object
```

[56]:
```
# PHASE III : MODEL BUILDING
```

[58]:
```
from sklearn.model_selection import train_test_split

# Target variable
y = df['user_rating']

# split
X_train, X_test , y_train, y_test = train_test_split(X,y, test_size=0.2,␣
 ↪random_state = 42)
```

[59]:
```
print(X_train.shape, X_test.shape)
```

```
(7045, 39) (1762, 39)
```

[79]:
```
df['type'] = df['type'].map({'Movie': 0, 'TV Show': 1}).fillna(0).astype(int)
```

[81]:
```
genre_dummies = genre_dummies.astype(int)
```

[82]:
```
X = pd.concat([df[['type', 'release_year', 'duration']], genre_dummies], axis=1)
```

[86]:
```
from sklearn.ensemble import GradientBoostingRegressor

# Re-initialize and train the model
gbr_model = GradientBoostingRegressor()
gbr_model.fit(X_train, y_train)

# Predict again
y_pred = gbr_model.predict(X_test)

# Evaluate performance
from sklearn.metrics import mean_squared_error, r2_score

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

print(f" Final Model (Gradient Boosting) MSE: {mse:.2f}")
print(f" Final Model (Gradient Boosting) R²: {r2:.2f}")
```

```
 Final Model (Gradient Boosting) MSE: 6.57
 Final Model (Gradient Boosting) R²: -0.01
```

[88]:
```
# Next Step: Save the Model for Streamlit

import joblib

# Save the trained model to a file
joblib.dump(gbr_model, 'netflix_rating_model.pkl')
```

[88]:
```
['netflix_rating_model.pkl']
```

[ ]:
```

```