TopMovies_popularity_vote-average

June 5, 2026

1 Objective: Given vote average, find / predict Popularity

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

[6]:
```
df = pd.read_csv("Top_Rated_Movies.csv")
df
```

[6]:
```
     popularity release_date                                title  \
0       174.522    9/23/1994                The Shawshank Redemption
1       165.677    3/14/1972                            The Godfather
2       174.522    9/23/1994                The Shawshank Redemption
3       165.677    3/14/1972                            The Godfather
4        47.916   12/20/1997                       Life Is Beautiful
..          …          …                                    …
959      23.525    5/17/2018                             Slender Man
960      13.800    3/23/2016              The Visitors: Bastille Day
961      12.215    3/12/1999                           Baby Geniuses
962      20.010    1/25/2007                              Epic Movie
963      32.961    5/22/2015  The Human Centipede 3 (Final Sequence)

     vote_average
0           8.706
1           8.690
2           8.706
3           8.690
4           8.449
..            …
959         4.301
960         4.100
961         4.109
962         3.853
963         3.750

[964 rows x 4 columns]
```

[7]:
```
df.describe()
```

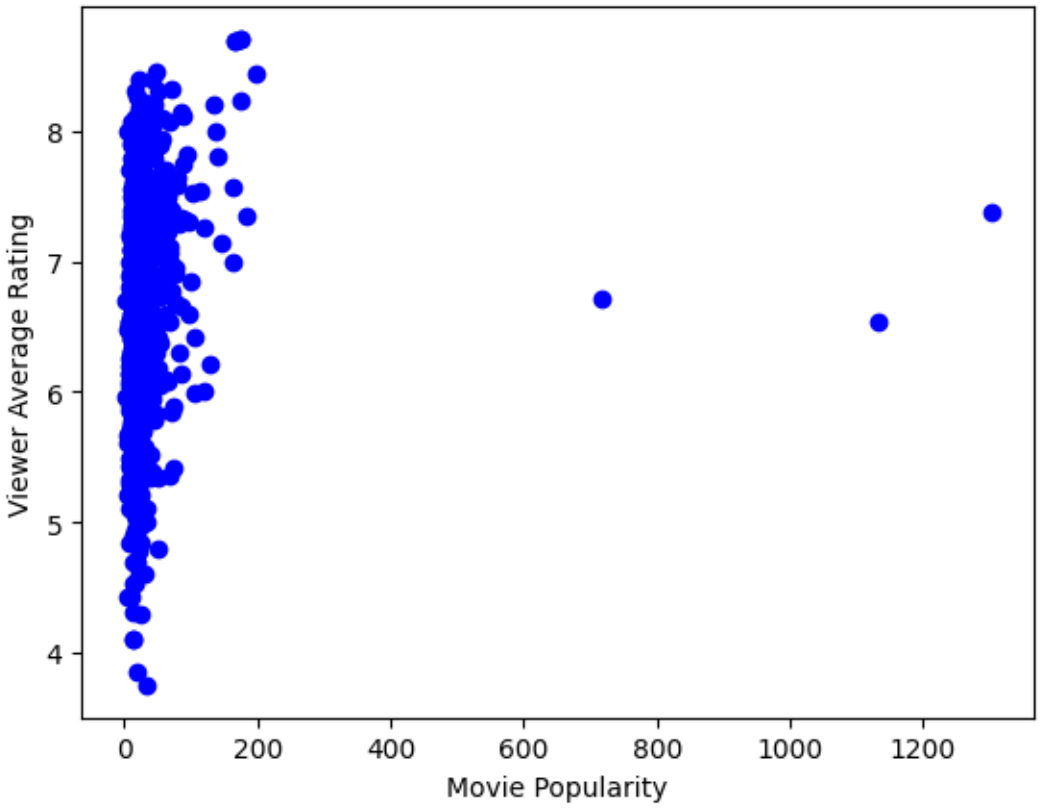
[7]:
```
         popularity  vote_average
count   964.000000    964.000000
mean     29.787525      6.641201
std      63.242456      0.788626
min       0.579000      3.750000
25%      14.530500      6.117750
50%      19.298000      6.664500
75%      29.353750      7.200000
max    1301.432000      8.706000
```

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 964 entries, 0 to 963
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   popularity    964 non-null    float64
 1   release_date  964 non-null    object 
 2   title         964 non-null    object 
 3   vote_average  964 non-null    float64
dtypes: float64(2), object(2)
memory usage: 30.3+ KB
```

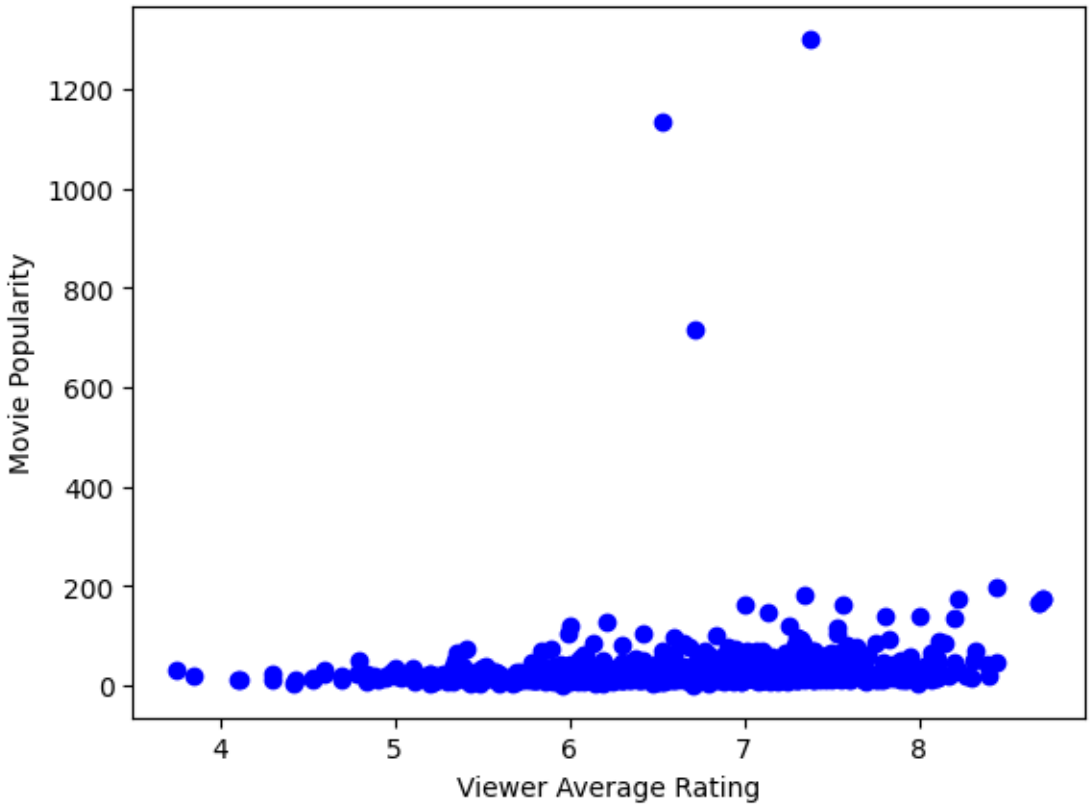
[12]:
```
plt.scatter(df['popularity'], df['vote_average'], color = 'blue')
plt.xlabel("Movie Popularity")
plt.ylabel("Viewer Average Rating")
```

[12]:
```
Text(0, 0.5, 'Viewer Average Rating')
```



[13]:
```
plt.scatter( df['vote_average'], df['popularity'], color = 'blue')
plt.ylabel("Movie Popularity")
plt.xlabel("Viewer Average Rating")
```

[13]: Text(0.5, 0, 'Viewer Average Rating')



[19]:
```
X = df.iloc[:, -1].values
y = df.iloc[:, 0].values
```

[54]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,␣
↪random_state = 0)
```

[55]:
```
print(f"X_train contains {len(X_train)} items")
print(f"\ty_train contains {len(y_train)} items")
print(f"X_test contains {len(X_test)} items")
print(f"\ty_test contains {len(y_test)} items")
```

```
X_train contains 674 items
        y_train contains 674 items
X_test contains 290 items
        y_test contains 290 items
```

[56]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
X_train = X_train.reshape(-1, 1)
regressor.fit(X_train, y_train)
```

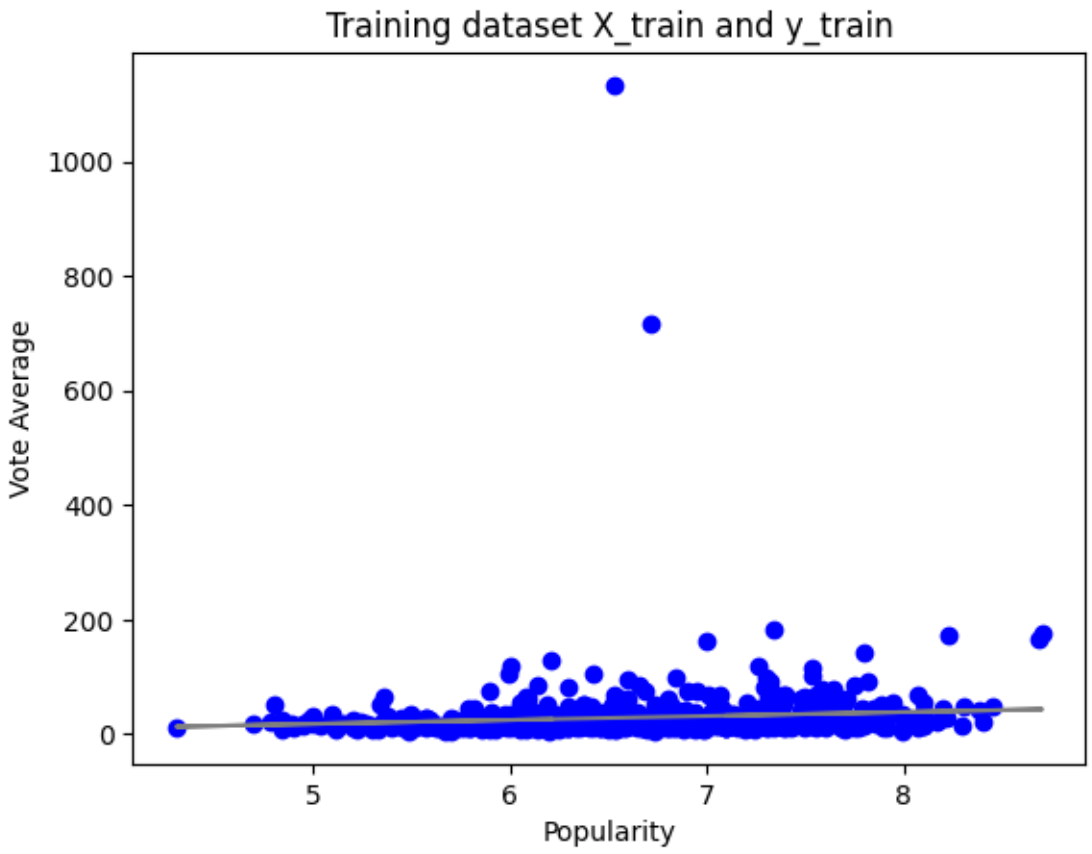
[56]:
```
LinearRegression()
```

[58]:
```
X_test = X_test.reshape(-1, 1)
y_pred = regressor.predict(X_test)
```

[59]:
```
plt.scatter(X_train, y_train, color = 'blue')
plt.plot(X_train, regressor.predict(X_train), color = 'grey')

plt.title ("Training dataset X_train and y_train")
plt.xlabel("Popularity")
plt.ylabel ("Vote Average")
```

[59]:
```
Text(0, 0.5, 'Vote Average')
```



[52]:
```
print(regressor.predict([[4], [2]]))
```

```
[  5.02659687 -14.3843978 ]
```

[60]:
```
slope = f"{regressor.coef_[0]: .3f}"
yint = f"{regressor.intercept_: .3f}"
```

```
print(f"The slope of line is {slope} and its y-intercept is at {yint}")
```

```
The slope of line is  6.891 and its y-intercept is at -16.509
```

[61]:
```
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
import numpy as np
# y_test = actual values, y_pred = your model's predictions
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"\tR2 Score (Variance Explained): {r2:.2f}")
print(f"\tMean Absolute Error: {mae:.2f}")
print(f"\tRoot Mean Squared Error: {rmse:.2f}")
```

```
        R2 Score (Variance Explained): 0.02
        Mean Absolute Error: 19.14
        Root Mean Squared Error: 78.22
```

[ ]:
```

```