Movie Recommendation Model

Netflix Movie Recommendation System: Data-Driven Insights and Foundations for Personalized Viewing

1. Overview:

This project focuses on analyzing Netflix movie data to extract meaningful insights that could inform a content-based recommendation approach. After thorough data cleaning handling missing values, duplicates, and inconsistent formats. I conducted exploratory data analysis (EDA) to understand user engagement and content trends. Key areas of analysis included:

- **Correlation between popularity, vote count, and rating**
- **Average popularity across genres**
- **Content volume trends over the years**

The insights highlight how popularity and viewer ratings relate to different genres and release periods. While not building a full recommendation engine, this project sets a strong analytical foundation by identifying the key features that influence user interest an essential step for designing effective, data-informed recommendations.

2. Data Cleaning:

To ensure the dataset was ready for accurate analysis and insights, I performed a series of cleaning steps. These helped improve data consistency, remove noise, and prepare the features for meaningful exploration.

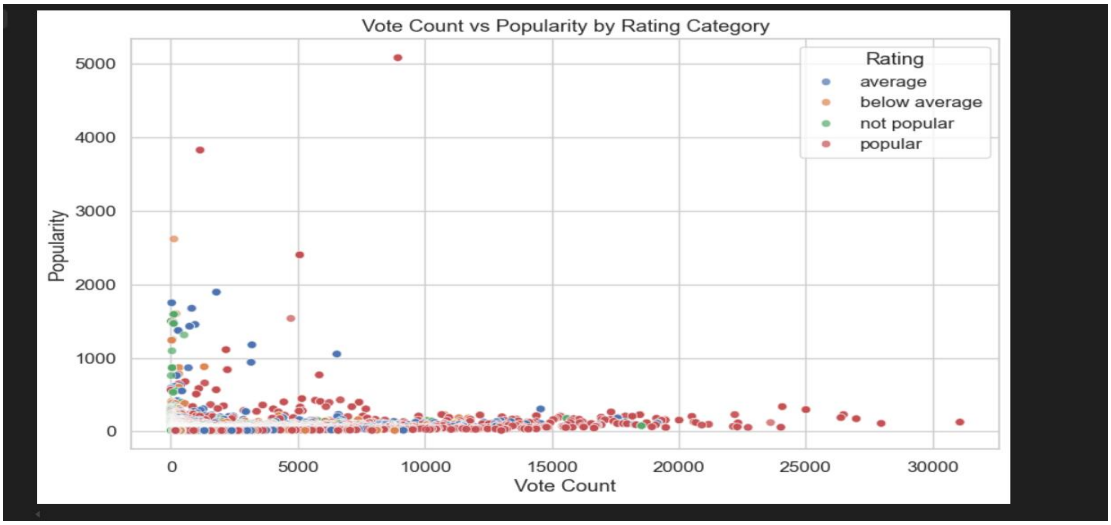
1. Handled Missing Values:
    - Checked for nulls across all columns.
    - Dropped rows with critical missing information (e.g., missing titles, genres, or popularity scores).
2. Removed Duplicates:
    - Identified and removed duplicate entries.
3. Data Type Conversions:
    - Converted release_date to datetime format for time-based analysis.
    - Ensured if numerical fields like popularity, vote_average, and vote_count were of the correct data type (float or integer).
4. Standardized Categorical Data:
    - Cleaned and standardized text fields such as genre for consistency.
    - Split multi-genre entries to analyze genre-wise patterns more effectively.
5. Dropped Irrelevant Columns:
    - Removed columns not necessary for the analysis or recommendation-related insights (e.g., URLs, Original Language).

3. Data Insights:

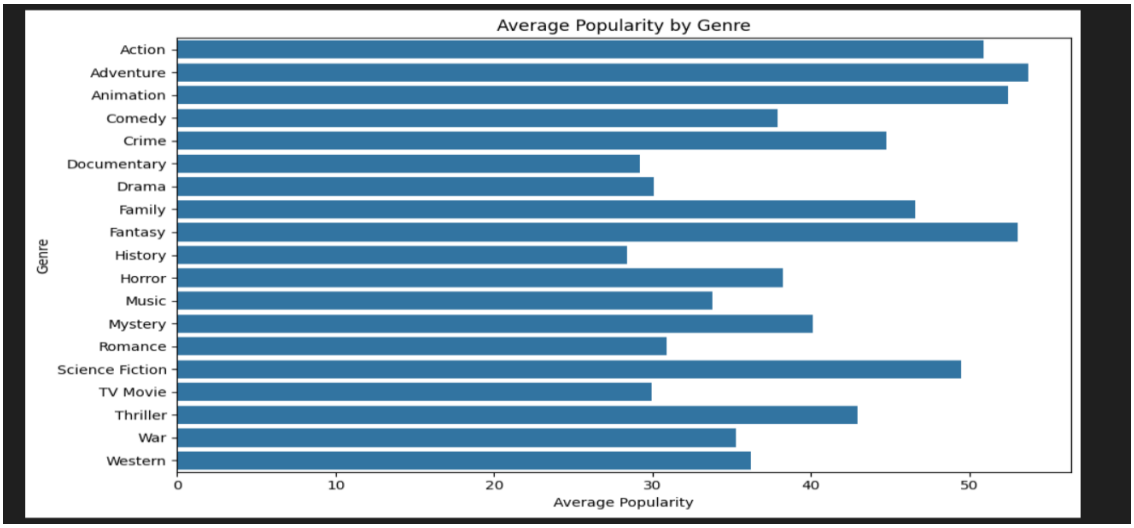
After cleaning the data, I analyzed it to find patterns in movie popularity, ratings, genres, and how Netflix's content has changed over time.

One key finding was that more votes usually mean higher popularity, but not always. The "popular" rating group contains many movies with high vote counts and popularity, showing wide reach. Some low-rated movies still have high popularity, meaning popularity doesn't always equal quality. The data is spread out a few movies have extremely high vote counts and popularity, but most don't.

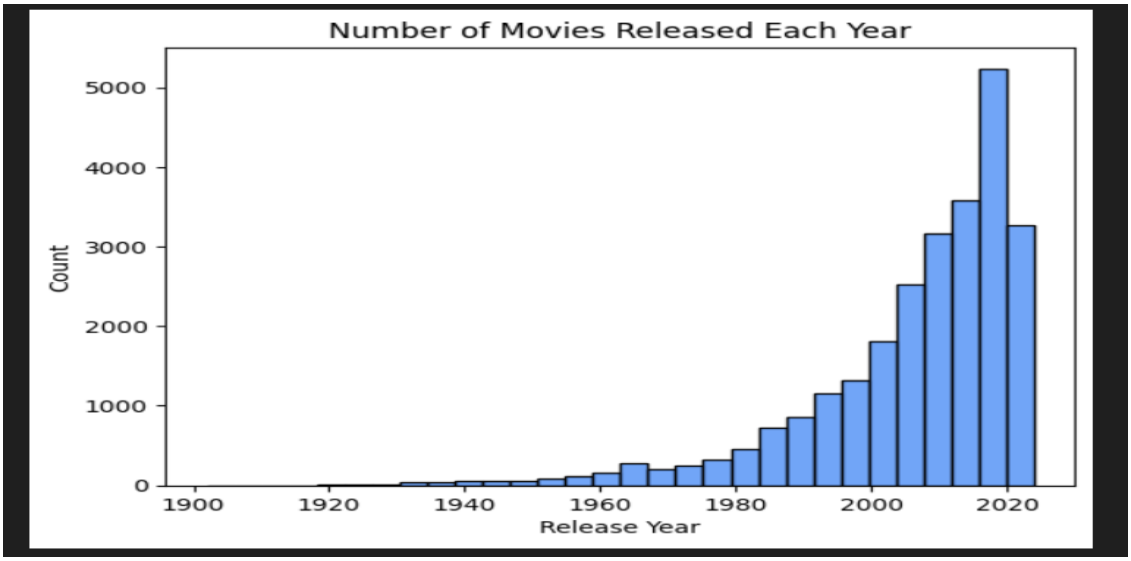
Although we expected movies with higher vote counts to be more popular, the data shows that popularity doesn't directly correlate with vote count. Some movies with tens of thousands of votes still have low popularity possibly because they are older, no longer trending, or received votes over a long time.



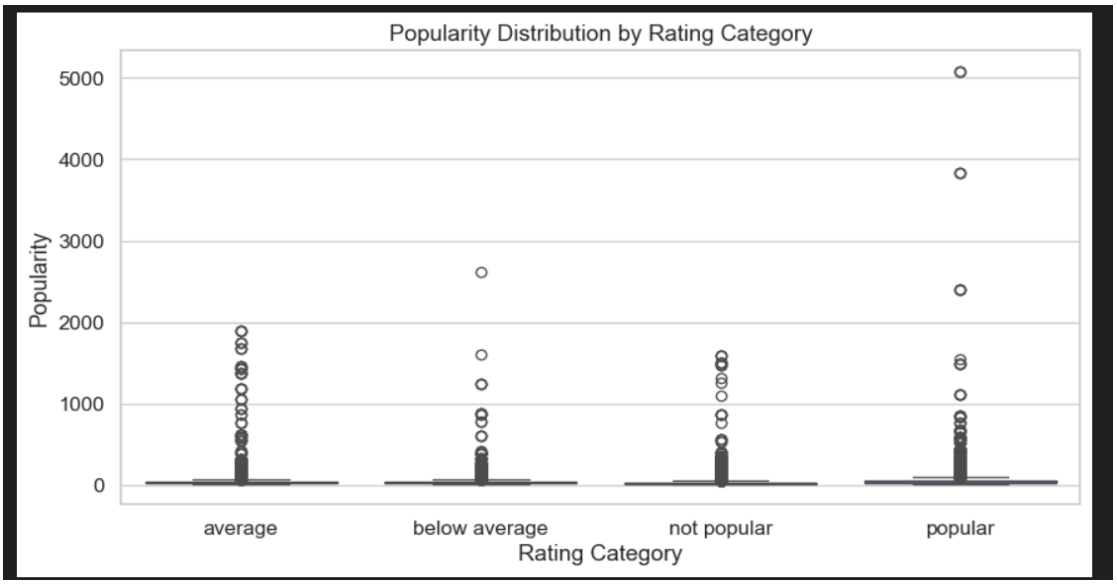
When looking at **Genres**, I found that **Action, Adventure, Animation,** and **Fantasy** are the most popular and also have the most movies on Netflix. These genres seem to be the ones Netflix focuses on to attract more viewers.



I also checked how many movies were added over the years. There was a big **increase from 2017 to 2020**, likely because Netflix was growing and creating more original content. But after 2020, there was a **drop in the number of movies added**, most likely because of the **COVID-19 pandemic**, which slowed down film production around the world.



This below boxplot shows the range of popularity scores in each rating category. Most movies in all categories have low popularity scores. The "popular" category includes movies with the highest popularity overall. All categories have outliers meaning a few movies in every group became unexpectedly popular. Popularity and rating don't always match a movie can be well-rated and unknown, or badly rated and widely talked about.



Some Additional Insights:

Which genre has the lowest Vote and Count Avg?

```
lowest_rated_genre = df.groupby("Genre", observed=True)[["Vote_Average", "Vote_Count"]].mean().sort_values("Vote_Average").head(1)
print(lowest_rated_genre)
```

[43] Python

```
        Vote_Average  Vote_Count
Genre
Horror      5.940068  985.364626
```

Which genre has the Highest Vote and Count Avg?

```
highest_rated_genre = df.groupby("Genre", observed=True)[["Vote_Average", "Vote_Count"]].mean().sort_values("Vote_Average",ascending=False).head(1)
print(highest_rated_genre)
```

[44] Python

```
Highly Voted but poorly rated:
                                  Title  Vote_Count  Vote_Average      Genre
0                                  2012       10288           5.8     Action
1                      Assassin's Creed        6719           5.4     Action
2      Batman v Superman: Dawn of Justice       15596           5.9     Action
3                        Fantastic Four        7911           5.8     Action
4  Fantastic Four: Rise of the Silver Surfer       6657           5.6  Adventure


Low Votes but highly rated:
                          Title  Vote_Count  Vote_Average        Genre
0                  12 Angry Men        6227           8.5        Drama
1       1987: When the Day Comes          95           8.0        Drama
2                     3 Idiots        1789           8.0        Drama
3  50 Greatest Harry Potter Moments          49           8.0  Documentary
4                            8¾        1733           8.2      Fantasy
```

These insights help us understand what kind of movies are popular on Netflix, how viewer behaviour works, and how global events can affect streaming content. This kind of analysis is helpful when thinking about how to recommend the right content to viewers.

4. ML Algorithms:

- **K-Nearest Neighbours (KNN)** is a simple and effective algorithm for making recommendations based on similarity. It helps find movies that are most similar to a selected one using features like:

  - Popularity
  - Number of Votes
  - Average Rating

  KNN doesn't require training a complex model it simply compares feature values and finds the "closest" movies. This is similar to how platforms like Netflix suggest “more like this” recommendations.

5. Code for KNN:

```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import NearestNeighbors

df_unique = df.drop_duplicates(subset='Title').reset_index(drop=True)

# Prepare features
features = df_unique[['Popularity', 'Vote_Count', 'Vote_Average']]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(features)

# Fit KNN model
knn_model = NearestNeighbors(n_neighbors=6, algorithm='auto')
knn_model.fit(X_scaled)

#input
movie_title = "pursuit".strip().lower()

# Case-insensitive match
df_unique['Title_lower'] = df_unique['Title'].str.lower()

if movie_title in df_unique['Title_lower'].values:
  movie_index = df_unique[df_unique['Title_lower'] == movie_title].index[0]

# Find similar movies
  distances, indices = knn_model.kneighbors([X_scaled[movie_index]])
  print("\nSelected Movie:", df_unique.loc[movie_index, 'Title'])
  print("Vote Average:", df_unique.loc[movie_index, 'Vote_Average'])
  print("\nRecommended Movies:")

 for idx in indices[0][1:]:  # Skip the first (same movie)
    title = df_unique.loc[idx, 'Title']
    vote_avg = df_unique.loc[idx, 'Vote_Average']
    genre = df_unique.loc[idx, 'Genre']
    print(f"{title} (Rating: {vote_avg}) : {genre}")

else:
  print("Sorry, movie not found.")
```

6. Summary:

The code builds a simple recommendation system using the K-Nearest Neighbors (KNN) algorithm. It uses features like popularity, vote count, and average rating to measure how similar one movie is to another. All features are scaled to ensure fair comparison. When a user enters a movie title, the model finds that movie's data and looks for the closest matches in the dataset. It then suggests 5 similar movies along with their average ratings and Genre.

7. Tools and Technologies:

- Python: Used as the primary programming language for data manipulation, visualization, and implementing the KNN algorithm.
- NumPy & Pandas: Used for loading, cleaning, and analyzing movie dataset in tabular format.
- Matplotlib & Seaborn: Used for creating visualizations like scatter plots and boxplots to explore the relationship between popularity, vote count, and ratings.
- Scikit-learn (sklearn): For feature normalization to bring all numeric values to a similar scale. For applying the K-Nearest Neighbours algorithm to recommend similar movies based on selected features.
- VS code/Jupyter Notebook: Used for interactive development, combining code, output, and explanations in a single document.
- Dataset: From https://www.kaggle.com/