data-analysis-report

December 11, 2023

1 Data Analysis Report

1.1 Importing required libraries

[1]:
```
# !pip install surprise
# !pip install wordcloud
# !pip install pandoc
from surprise import SVD
import numpy as np
import surprise
from surprise import Reader, Dataset
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
```

1.2 Description of dataset

I will be loading each giving dataset in the start

1.2.1 ratings

ratings dataframe has four columns userId, movieId, ratings, timestamp, on a first glance it gives the following insights,

1. Each row in the dataframe represents a user(userId) A, rated movie (movieId) B with rating(rating) X
2. It has total number of 100836 rows and 4 columns

1.2.2 movies

movies dataframe has three columns movieId, title, genres, first glance insights,

1. movieId is an integer and seems to be coherent with movieId of ratings table -> (gives me idea that I can join these two dataframes and there are chances I might be able to get what Genres interests a user more, and it might contribute to my recommendation)
2. title is nothing but a string i.e. movie name
3. genres are separated by pipe | and are closely associated with the movieId and title

1.2.3 tags

tags dataframe has four columns userId, movieId, tag, timestamp, first glance insights

1. userId, and movieId are of integer type and seems to be in conherence with movies and ratings dataframe
2. tag is a new column which is user-generated meta data, I would say a user's perspective for a particular movie, it is also a string (object) datatype

links links dataframe has three columns movieId, imdbId, tmdbId, based on the information provided in the assessment readme, I am understanding it as the identifiers for movies, and might be helpful once I build recommender system, I can route user to these links.

tmdb_metadata tmdb_metadata provides meta data for the movies

1.3 Converting JSON files in tmdb folder in a CSV file

[ ]:
```
# Commenting out the code as I have already generated csv file using this script
"""
import os
import json
import pandas as pd

# Specify the path to the folder containing JSON files
json_folder_path = 'tmdb'

# List all JSON files in the folder
json_files = [f for f in os.listdir(json_folder_path) if f.endswith('.json')]

# Initialize an empty list to store data from JSON files
data = []

# Read each JSON file and append its data to the list
for json_file in json_files:
    with open(os.path.join(json_folder_path, json_file), 'r', encoding='utf-8')␣
↪as f:
        file_content = f.read()
        if file_content.strip():
            try:
                json_data = json.loads(file_content)
                data.append(json_data)
            except json.JSONDecodeError as e:
                print(f"Error loading {json_file}: {e}")

# Convert the list of dictionaries to a Pandas DataFrame
df = pd.DataFrame(data)

# Specify the path for the output CSV file
```

```
csv_output_path = 'tmdb_metadata.csv'

# Write the DataFrame to a CSV file
df.to_csv(csv_output_path, index=False)

print(f"CSV file created at: {csv_output_path}")
"""
```

```
CSV file created at: tmdb_metadata.csv
```

1.4 Load all the csv data

[3]:
```python
ratings = pd.read_csv("ratings.csv")
```

[4]:
```python
print("Ratings dataset")
ratings.head()
```

```
Ratings dataset
```

[4]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

[5]:
```python
movies = pd.read_csv("movies.csv")
```

[6]:
```python
print("Movies dataset")
movies.head()
```

```
Movies dataset
```

[6]:

	movieId	title	\
0	1	Toy Story (1995)	
1	2	Jumanji (1995)	
2	3	Grumpier Old Men (1995)	
3	4	Waiting to Exhale (1995)	
4	5	Father of the Bride Part II (1995)	

	genres
0	Adventure|Animation|Children|Comedy|Fantasy
1	Adventure|Children|Fantasy
2	Comedy|Romance
3	Comedy|Drama|Romance
4	Comedy

[7]:
```python
tags = pd.read_csv("tags.csv")
```

[8]:
```
print("tags dataset")
tags.head()
```

```
tags dataset
```

[8]:
```
   userId  movieId               tag   timestamp
0       2    60756             funny  1445714994
1       2    60756   Highly quotable  1445714996
2       2    60756      will ferrell  1445714992
3       2    89774      Boxing story  1445715207
4       2    89774               MMA  1445715200
```

[9]:
```
links = pd.read_csv("links.csv")
```

[10]:
```
print("links dataset")
links.head()
```

```
links dataset
```

[10]:
```
   movieId  imdbId   tmdbId
0        1  114709    862.0
1        2  113497   8844.0
2        3  113228  15602.0
3        4  114885  31357.0
4        5  113041  11862.0
```

[11]:
```
links.dtypes
```

[11]:
```
movieId      int64
imdbId       int64
tmdbId     float64
dtype: object
```

1.4.1 Merge movies and links dataset on movieId

Once we will merge movies with links data, we would be able to merge tmdb_metadata with this merged data, in the next few steps I will be merging these two datasets.

Reason I am doing this for data exploration, to get better understanding of movies dataset, things like, how movies are distributed among genres, popularity of movies, etc.

[12]:
```
movies_links = pd.merge(movies, links, how='inner', on='movieId')
```

[13]:
```
movies_links.head()
```

[13]:
```
   movieId                               title  \
0        1                    Toy Story (1995)
1        2                      Jumanji (1995)
```

```
2        3              Grumpier Old Men (1995)
3        4             Waiting to Exhale (1995)
4        5   Father of the Bride Part II (1995)

                                        genres  imdbId   tmdbId
0  Adventure|Animation|Children|Comedy|Fantasy  114709    862.0
1                   Adventure|Children|Fantasy  113497   8844.0
2                               Comedy|Romance  113228  15602.0
3                         Comedy|Drama|Romance  114885  31357.0
4                                       Comedy  113041  11862.0
```

Let's check for the NaN values if any in the dataset

[14]:
```
nan_counts = movies_links.isna().sum()
print(nan_counts)
```

```
movieId    0
title      0
genres     0
imdbId     0
tmdbId     8
dtype: int64
```

It seems column tmdbId have 8 NA values, I think it would be ok to drop all those rows from the dataset, that would not be huge loss

Dropping NaN rows

[15]:
```
movies_links = movies_links.dropna()
```

[16]:
```
movies_links.head()
```

[16]:
```
   movieId                                title  \
0        1                     Toy Story (1995)
1        2                       Jumanji (1995)
2        3              Grumpier Old Men (1995)
3        4             Waiting to Exhale (1995)
4        5   Father of the Bride Part II (1995)

                                        genres  imdbId   tmdbId
0  Adventure|Animation|Children|Comedy|Fantasy  114709    862.0
1                   Adventure|Children|Fantasy  113497   8844.0
2                               Comedy|Romance  113228  15602.0
3                         Comedy|Drama|Romance  114885  31357.0
4                                       Comedy  113041  11862.0
```

Change datatype Also changing the datatype of column tmdbId to int type, because we want to join two datasets `tmdb_metadata` and `movies_links` on tmdbId, and `tmdbId` in the `tmdb_metadata` dataframe is of type `int64`

[17]:
```
movies_links['tmdbId'] = movies_links['tmdbId'].astype('int64')
movies_links.head()
```

[17]:
```
   movieId                               title  \
0        1                    Toy Story (1995)
1        2                      Jumanji (1995)
2        3             Grumpier Old Men (1995)
3        4            Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)

                                        genres  imdbId  tmdbId
0  Adventure|Animation|Children|Comedy|Fantasy  114709     862
1                   Adventure|Children|Fantasy  113497    8844
2                               Comedy|Romance  113228   15602
3                         Comedy|Drama|Romance  114885   31357
4                                       Comedy  113041   11862
```

Bringing in tmdb_metadata in the jupyter notebook cell

[18]:
```
tmdb_metadata = pd.read_csv("tmdb_metadata.csv")
```

[19]:
```
tmdb_metadata.head()
```

[19]:
```
                                            overview  popularity  \
0  The second "visual album" (a collection of sho…       8.738
1  Set in 1929, a political boss and his advisor …      17.518
2  A student's premonition of a deadly rollercoas…      40.900
3  On Christmas Eve, three homeless people living…      21.095
4  A wily old codger matches wits with the King o…      12.456

                     original_title  runtime release_date  vote_average  \
0                          Lemonade       65   2016-04-23         8.497
1                  Miller's Crossing      115   1990-09-21         7.455
2                Final Destination 3       93   2006-02-09         6.081
3                                 93   2003-12-29         7.895
4  Darby O'Gill and the Little People       93   1959-06-24         6.700

   vote_count    status                                           tagline  \
0         147  Released                                               NaN
1        1496  Released  Up is down, black is white, and nothing is wha…
2        3549  Released             This ride will be the death of you.
3        1076  Released          Meet the ultimate dysfunctional family.
4         130  Released  A touch O'Blarney… a heap O'Magic and A LOAD…

  spoken_languages                                               cast      id
0               en  Beyoncé|Jay-Z|Serena Williams|Zendaya|Quvenzha…  394269
```

```
1      en|ga|it|yi  Gabriel Byrne|Albert Finney|Jon Polito|Marcia …    379
2               en  Mary Elizabeth Winstead|Ryan Merriman|Kris Lem…   9286
3         en|ja|es  Aya Okamoto|Yoshiaki Umegaki|Tohru Emori|Satom…  13398
4            ga|en  Albert Sharpe|Janet Munro|Sean Connery|Jimmy O…  18887
```

Renaming a column changing the name of the id column to tmdbId so that we can easily join movies_links and tmdb_metadata together

[20]:
```python
tmdb_metadata.rename(columns={'id': 'tmdbId'}, inplace=True)
```

[21]:
```python
tmdb_metadata.head()
```

[21]:
```
                                            overview  popularity  \
0  The second "visual album" (a collection of sho…       8.738
1  Set in 1929, a political boss and his advisor …      17.518
2  A student's premonition of a deadly rollercoas…      40.900
3  On Christmas Eve, three homeless people living…      21.095
4  A wily old codger matches wits with the King o…      12.456

                      original_title  runtime release_date  vote_average  \
0                           Lemonade       65   2016-04-23         8.497
1                  Miller's Crossing      115   1990-09-21         7.455
2                Final Destination 3       93   2006-02-09         6.081
3                                     93   2003-12-29         7.895
4  Darby O'Gill and the Little People       93   1959-06-24         6.700

   vote_count    status                                        tagline  \
0         147  Released                                            NaN
1        1496  Released  Up is down, black is white, and nothing is wha…
2        3549  Released             This ride will be the death of you.
3        1076  Released         Meet the ultimate dysfunctional family.
4         130  Released  A touch O'Blarney… a heap O'Magic and A LOAD…

  spoken_languages                                               cast  tmdbId
0               en  Beyoncé|Jay-Z|Serena Williams|Zendaya|Quvenzha…  394269
1      en|ga|it|yi  Gabriel Byrne|Albert Finney|Jon Polito|Marcia …    379
2               en  Mary Elizabeth Winstead|Ryan Merriman|Kris Lem…   9286
3         en|ja|es  Aya Okamoto|Yoshiaki Umegaki|Tohru Emori|Satom…  13398
4            ga|en  Albert Sharpe|Janet Munro|Sean Connery|Jimmy O…  18887
```

[22]:
```python
movies_metadata = pd.merge(movies_links, tmdb_metadata, on='tmdbId',␣
↪how='inner' )
```

[23]:
```python
movies_metadata.head()
```

[23]:
```
   movieId                               title  \
0        1                    Toy Story (1995)
```

```
1       2                      Jumanji (1995)
2       3             Grumpier Old Men (1995)
3       4            Waiting to Exhale (1995)
4       5  Father of the Bride Part II (1995)

                                        genres  imdbId  tmdbId  \
0  Adventure|Animation|Children|Comedy|Fantasy  114709     862
1                   Adventure|Children|Fantasy  113497    8844
2                               Comedy|Romance  113228   15602
3                         Comedy|Drama|Romance  114885   31357
4                                       Comedy  113041   11862

                                            overview  popularity  \
0  Led by Woody, Andy's toys live happily in his …     100.954
1  When siblings Judy and Peter discover an encha…      13.981
2  A family wedding reignites the ancient feud be…      12.686
3  Cheated on, mistreated and stepped on, the wom…      11.945
4  Just when George Banks has recovered from his …      19.558

                original_title  runtime release_date  vote_average  \
0                    Toy Story       81   1995-10-30         7.970
1                      Jumanji      104   1995-12-15         7.239
2             Grumpier Old Men      101   1995-12-22         6.494
3            Waiting to Exhale      127   1995-12-22         6.183
4  Father of the Bride Part II      106   1995-12-08         6.239

   vote_count    status                                            tagline  \
0       17277  Released  Hang on for the comedy that goes to infinity a…
1        9891  Released           Roll the dice and unleash the excitement!
2         350  Released  Still Yelling. Still Fighting. Still Ready for…
3         142  Released  Friends are the people who let you be yourself…
4         665  Released  Just when his world is back to normal… he's …

  spoken_languages                                               cast
0               en  Tom Hanks|Tim Allen|Don Rickles|Jim Varney|Wal…
1            en|fr  Robin Williams|Kirsten Dunst|Bradley Pierce|Bo…
2               en  Walter Matthau|Jack Lemmon|Ann-Margret|Sophia …
3               en  Whitney Houston|Angela Bassett|Loretta Devine|…
4               en  Steve Martin|Diane Keaton|Martin Short|Kimberl…
```

1.5 Data exploration on the Movies metadata

Movies metadata provides information about various features of movies, such as genres, release dates, popularity, votes, cast, etc. Understanding these features is crucial for building recommendation models that can capture user preferences. Genres play a significant role in user preferences. Analyzing movie genres helps in creating genre-based recommendation systems. Users often have specific genre preferences, and recommending movies based on these preferences can enhance user satisfaction. Collaborative filtering is a popular recommendation technique that relies on user-item interactions. Movies metadata, including user ratings and reviews, is crucial for collaborative filtering models. Analyzing user behavior helps identify patterns and similarities, enabling accurate recommendations.

This exploration will be there with code cell by cell

Shape of our `movies_metadata` dataframe is `9622*16`, 9622 rows and 16 columns

1.5.1 Number of unique movie names

[24]:
```
print("Number of unique movie names: \n")
movies_metadata['title'].nunique()
```

```
Number of unique movie names:
```

[24]:
```
9619
```

1.5.2 Duplicate data

Looks like there are some duplicates in the movie title (total number of rows 9622 and unique number of titles are 9619), let's find out those duplicate rows

Checking for duplicate titles

[25]:
```
dup_titles = movies_metadata[movies_metadata.duplicated('title', keep=False)]

print("repeated titles: \n")
dup_titles
```

```
repeated titles:
```

[25]:
```
      movieId                                  title  \
649       838                            Emma (1996)
4161     6003  Confessions of a Dangerous Mind (2002)
4162   144606  Confessions of a Dangerous Mind (2002)
5581    26958                            Emma (1996)
5903    34048                 War of the Worlds (2005)
6891    64997                 War of the Worlds (2005)

                                      genres  imdbId  tmdbId  \
649                     Comedy|Drama|Romance  116191    3573
4161             Comedy|Crime|Drama|Thriller  290538    4912
4162     Comedy|Crime|Drama|Romance|Thriller  270288    4912
5581                                 Romance  118308   12254
5903       Action|Adventure|Sci-Fi|Thriller  407304      74
6891                           Action|Sci-Fi  449040   34812
```

```
                                               overview  popularity  \
649   Emma Woodhouse is a congenial young lady who d…      15.265
4161  Television made him famous, but his biggest hi…      15.898
4162  Television made him famous, but his biggest hi…      15.898
5581  Emma Woodhouse has a rigid sense of propriety …      11.745
5903  Ray Ferrier is a divorced dockworker and less-…      54.283
6891  In this modern retelling of H.G. Wells' classi…       7.492

                        original_title  runtime release_date  vote_average  \
649                               Emma      121   1996-08-02         6.679
4161  Confessions of a Dangerous Mind      113   2002-12-31         6.710
4162  Confessions of a Dangerous Mind      113   2002-12-31         6.710
5581                              Emma      107   1996-10-02         6.700
5903                 War of the Worlds      117   2005-06-28         6.500
6891    H.G. Wells' War of the Worlds      100   2005-06-28         5.519

      vote_count    status                                  tagline  \
649          545  Released            Cupid is armed and dangerous!
4161        1044  Released  Some things are better left top secret.
4162        1044  Released  Some things are better left top secret.
5581         115  Released                                      NaN
5903        7790  Released                    They're already here.
6891          53  Released                                      NaN

     spoken_languages                                              cast
649                en  Gwyneth Paltrow|Toni Collette|Alan Cumming|Ewa…
4161               en  Sam Rockwell|Drew Barrymore|George Clooney|Jul…
4162               en  Sam Rockwell|Drew Barrymore|George Clooney|Jul…
5581               en  Kate Beckinsale|Mark Strong|Samantha Morton|Ra…
5903               en  Tom Cruise|Dakota Fanning|Justin Chatwin|Miran…
6891               en  C. Thomas Howell|Rhett Giles|Jake Busey|Peter …
```

1.6 Observation

1.6.1 Duplicate titles

1. It seems there are three movie title which are exactly same so does their release year (except for movie Emma which has two different release date)
2. I closely observed the genres of these three movies and found some similarity over there
3. But it seems popularity value differs for these movies
4. I will be sorting the movies based on their popularity because I think for a recommender system popularity of the movie can influence a movie rating, so I will keep the movies with high popularity and will drop the duplicates

[26]:
```
movies_metadata = movies_metadata.sort_values(by='popularity', ascending=False)
```

[27]:
```
movies_metadata = movies_metadata.drop_duplicates('title', keep='first')
movies_metadata.reset_index(drop=True).head()
```

[27]:
```
   movieId                               title  \
0    61350                  Babylon A.D. (2008)
1   177765                          Coco (2017)
2    99721             Texas Chainsaw 3D (2013)
3   122912  Avengers: Infinity War - Part I (2018)
4   112171                Equalizer, The (2014)

                         genres   imdbId  tmdbId  \
0  Action|Adventure|Sci-Fi|Thriller   364970    9381
1      Adventure|Animation|Children  2380307  354912
2           Horror|Mystery|Thriller  1572315   76617
3           Action|Adventure|Sci-Fi  4154756  299536
4             Action|Crime|Thriller   455944  156022

                                            overview  popularity  \
0  A veteran-turned-mercenary is hired to take a …     679.514
1  Despite his family's baffling generations-old …     406.240
2  A young woman learns that she has inherited a …     212.727
3  As the Avengers and their allies have continue…     202.813
4  McCall believes he has put his mysterious past…     164.860

           original_title  runtime release_date  vote_average  vote_count  \
0            Babylon A.D.      101   2008-08-20         5.600        1874
1                    Coco      105   2017-10-27         8.217       18096
2       Texas Chainsaw 3D       92   2013-01-03         5.444        1521
3  Avengers: Infinity War      149   2018-04-25         8.252       27907
4           The Equalizer      132   2014-09-24         7.257        8288

     status                            tagline spoken_languages  \
0  Released                   Kill or be Killed.            en|ru
1  Released           The celebration of a lifetime            en|es
2  Released                Evil wears many faces.               en
3  Released  An entire universe. Once and for all.            en|xh
4  Released   What do you see when you look at me?               en

                                                cast
0  Vin Diesel|Michelle Yeoh|Mélanie Thierry|Lambe…
1  Anthony Gonzalez|Gael García Bernal|Benjamin B…
2  Alexandra Daddario|Dan Yeager|Trey Songz|Tania…
3  Robert Downey Jr.|Chris Hemsworth|Mark Ruffalo…
4  Denzel Washington|Marton Csokas|Chloë Grace Mo…
```

[28]:
```
movies_metadata = movies_metadata.sort_values(by='movieId', ascending=True)
movies_metadata.reset_index(drop=True).head()
```

[28]:

```
   movieId                               title  \
0        1                    Toy Story (1995)
1        2                      Jumanji (1995)
2        3             Grumpier Old Men (1995)
3        4            Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)

                                        genres  imdbId  tmdbId  \
0  Adventure|Animation|Children|Comedy|Fantasy  114709     862
1                   Adventure|Children|Fantasy  113497    8844
2                               Comedy|Romance  113228   15602
3                         Comedy|Drama|Romance  114885   31357
4                                       Comedy  113041   11862

                                            overview  popularity  \
0  Led by Woody, Andy's toys live happily in his …     100.954
1  When siblings Judy and Peter discover an encha…      13.981
2  A family wedding reignites the ancient feud be…      12.686
3  Cheated on, mistreated and stepped on, the wom…      11.945
4  Just when George Banks has recovered from his …      19.558

                original_title  runtime release_date  vote_average  \
0                    Toy Story       81   1995-10-30         7.970
1                      Jumanji      104   1995-12-15         7.239
2             Grumpier Old Men      101   1995-12-22         6.494
3            Waiting to Exhale      127   1995-12-22         6.183
4  Father of the Bride Part II      106   1995-12-08         6.239

   vote_count    status                                      tagline  \
0       17277  Released  Hang on for the comedy that goes to infinity a…
1        9891  Released           Roll the dice and unleash the excitement!
2         350  Released  Still Yelling. Still Fighting. Still Ready for…
3         142  Released  Friends are the people who let you be yourself…
4         665  Released  Just when his world is back to normal… he's …

  spoken_languages                                               cast
0               en  Tom Hanks|Tim Allen|Don Rickles|Jim Varney|Wal…
1            en|fr  Robin Williams|Kirsten Dunst|Bradley Pierce|Bo…
2               en  Walter Matthau|Jack Lemmon|Ann-Margret|Sophia …
3               en  Whitney Houston|Angela Bassett|Loretta Devine|…
4               en  Steve Martin|Diane Keaton|Martin Short|Kimberl…
```

1.6.2 Redundant data

1. My first observation of column `title` and `original_title` says, these two columns are essentially same except `original_title` also have information in regional language, I think dropping this column would be safe

2. Observation on column `title` and `release_date` gives me insight that column `title` possess the information regarding the release year of the movie, my hunch says dropping this column would be safe too (by safe I mean, we will not lose any important information by dropping it)

1.6.3 Information from other columns

1. **One another column is `spoken_languages`, this column can would certainly make sense when we would develop community recommendation engine. For example a user from Asian community would like to get recommendation based on some asian languages like Japanese, Korean, Hindi, Chinese etc.** Otherwise, I feel it would not be play a vital role for a generic movie recommendation engine.
2. Column `status` can be dropped cause it all the datapoints have a value Released, is certainly not contributing much for the recommendation system
3. Other columns at first glance looks helpful, I will be exploring more with the each column.

[29]:
```
movies_metadata['status'].unique()
```

[29]:
```
array(['Released'], dtype=object)
```

[30]:
```
movies_metadata.drop(columns=['original_title', 'release_date', 'status',␣
↪'spoken_languages'], inplace=True)
```

After doing cleaning the dataframe would look like

[31]:
```
movies_metadata.head()
```

[31]:
```
   movieId                               title  \
0        1                    Toy Story (1995)   
1        2                      Jumanji (1995)   
2        3             Grumpier Old Men (1995)   
3        4            Waiting to Exhale (1995)   
4        5  Father of the Bride Part II (1995)   

                                        genres  imdbId  tmdbId  \
0  Adventure|Animation|Children|Comedy|Fantasy  114709     862   
1                   Adventure|Children|Fantasy  113497    8844   
2                               Comedy|Romance  113228   15602   
3                         Comedy|Drama|Romance  114885   31357   
4                                       Comedy  113041   11862   

                                            overview  popularity  runtime  \
0  Led by Woody, Andy's toys live happily in his …     100.954       81   
1  When siblings Judy and Peter discover an encha…      13.981      104   
2  A family wedding reignites the ancient feud be…      12.686      101   
3  Cheated on, mistreated and stepped on, the wom…      11.945      127   
4  Just when George Banks has recovered from his …      19.558      106   
```

```
   vote_average  vote_count  \
0         7.970       17277
1         7.239        9891
2         6.494         350
3         6.183         142
4         6.239         665

                                             tagline  \
0  Hang on for the comedy that goes to infinity a…
1          Roll the dice and unleash the excitement!
2  Still Yelling. Still Fighting. Still Ready for…
3  Friends are the people who let you be yourself…
4  Just when his world is back to normal… he's …

                                                cast
0  Tom Hanks|Tim Allen|Don Rickles|Jim Varney|Wal…
1  Robin Williams|Kirsten Dunst|Bradley Pierce|Bo…
2  Walter Matthau|Jack Lemmon|Ann-Margret|Sophia …
3  Whitney Houston|Angela Bassett|Loretta Devine|…
4  Steve Martin|Diane Keaton|Martin Short|Kimberl…
```

[32]:
```
nan_counts = movies_metadata.isna().sum()
print(nan_counts)
```

```
movieId            0
title              0
genres             0
imdbId             0
tmdbId             0
overview           9
popularity         0
runtime            0
vote_average       0
vote_count         0
tagline         1568
cast              44
dtype: int64
```

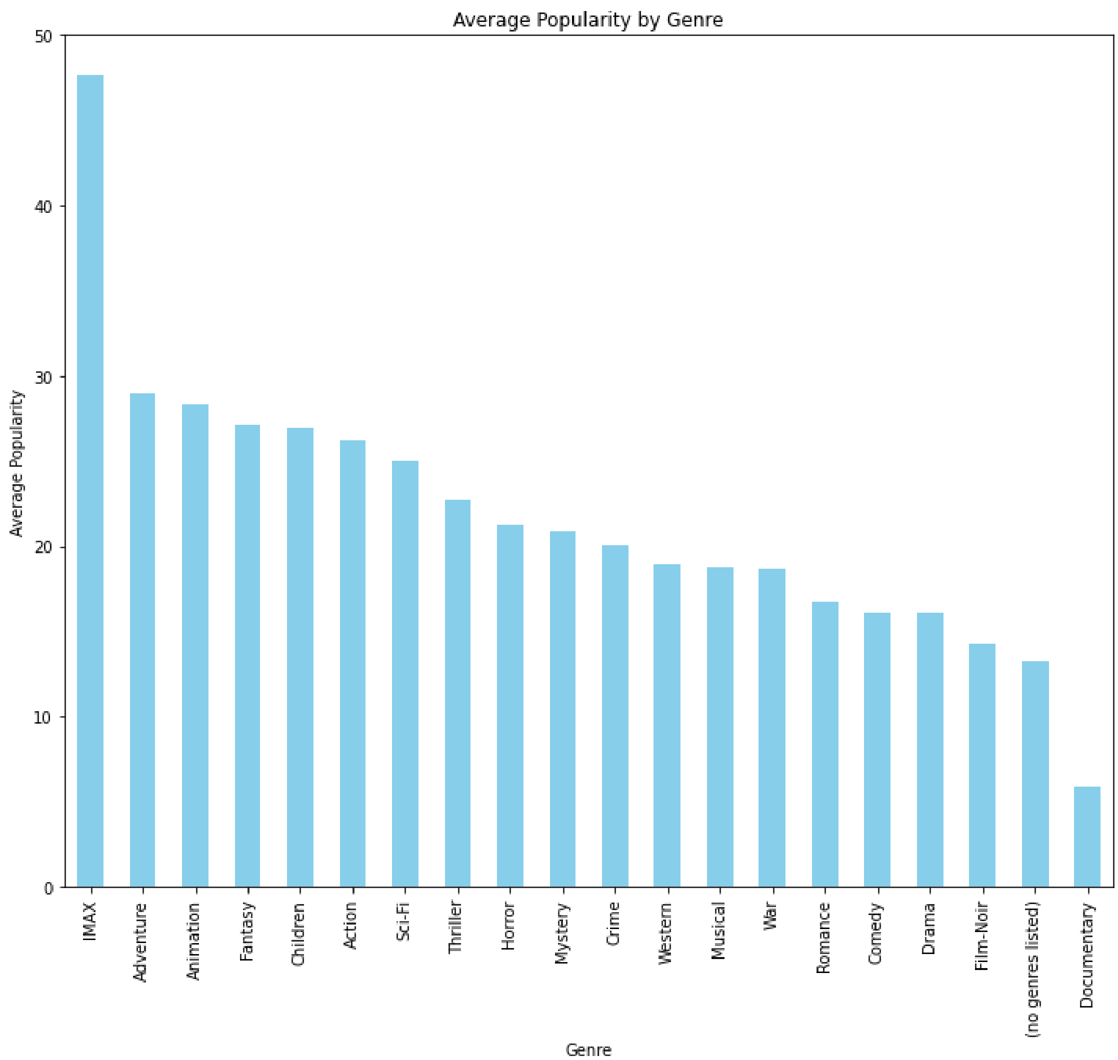
Dataset looks more cleaner and concise now.

1.7 Genre based analysis

In next steps I would like to explore genre's association with a couple of other features

1.7.1 Bar plot based on the popularity of Genres

[33]:
```
# Convert the genres column to a list
df = movies_metadata.copy()
df['genres'] = df['genres'].apply(lambda x: x.split('|'))

# Create a new DataFrame with exploded genres
df_genres = df.explode('genres')

# # # Calculate the average popularity for each genre
avg_popularity_by_genre = df_genres.groupby('genres')['popularity'].mean()
sorted_genres = avg_popularity_by_genre.sort_values(ascending=False)

# Plot the results
sorted_genres.plot(kind='bar', color='skyblue', figsize=(12, 10))
plt.title('Average Popularity by Genre')
plt.xlabel('Genre')
plt.ylabel('Average Popularity')
plt.show()
```



Above chart shows that IMAX genre is the most popular one, but in the dataset additional information, IMAX genre does not make any sense. It says, Genres are a pipe-separated list, and are selected from the following:

- Action
- Adventure
- Animation
- Children's
- Comedy
- Crime
- Documentary
- Drama
- Fantasy
- Film-Noir
- Horror

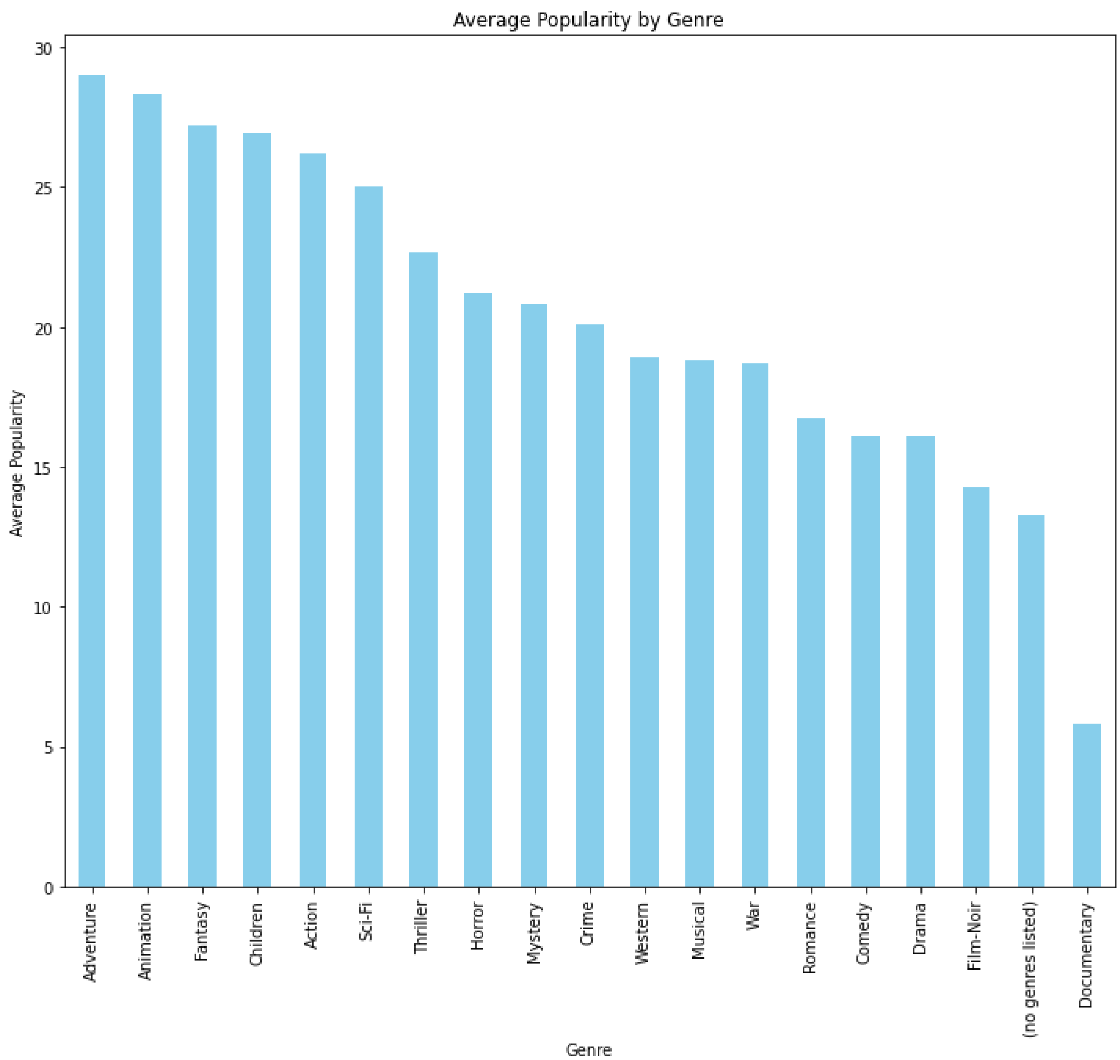
- Musical
- Mystery
- Romance
- Sci-Fi
- Thriller
- War
- Western
- (no genres listed)

This gives me idea that I can simply remove IMAX from the genre column where values are | separated

[34]:
```
# Remove "IMAX" from all rows in the "genres" column
movies_metadata['genres'] = movies_metadata['genres'].apply(lambda x: '|'.
↪join(filter(lambda genre: genre != 'IMAX', x.split('|'))))

# df = movies_metadata.copy()
movies_metadata['genres'] = movies_metadata['genres'].apply(lambda x: x.
↪split('|'))

# Create a new DataFrame with exploded genres
df_genres = movies_metadata.explode('genres')

print("unique movie genres: ", df_genres['genres'].unique())
```

```
unique movie genres:  ['Adventure' 'Animation' 'Children' 'Comedy' 'Fantasy'
'Romance' 'Drama'
 'Action' 'Crime' 'Thriller' 'Horror' 'Mystery' 'Sci-Fi' 'War' 'Musical'
 'Documentary' 'Western' 'Film-Noir' '(no genres listed)']
```

[35]:
```
avg_popularity_by_genre = df_genres.groupby('genres')['popularity'].mean()
sorted_genres = avg_popularity_by_genre.sort_values(ascending=False)

# Plot the results
sorted_genres.plot(kind='bar', color='skyblue', figsize=(12, 10))
plt.title('Average Popularity by Genre')
plt.xlabel('Genre')
plt.ylabel('Average Popularity')
plt.show()
```



[36]:
```
# Get the top 5 popular genres
top_5_genres = sorted_genres.head(5)
print("Top 5 most popular genres: \n")
print(top_5_genres)
```

```
Top 5 most popular genres:

genres
Adventure    28.979274
Animation    28.327928
Fantasy      27.174074
Children     26.935398
Action       26.179208
Name: popularity, dtype: float64
```

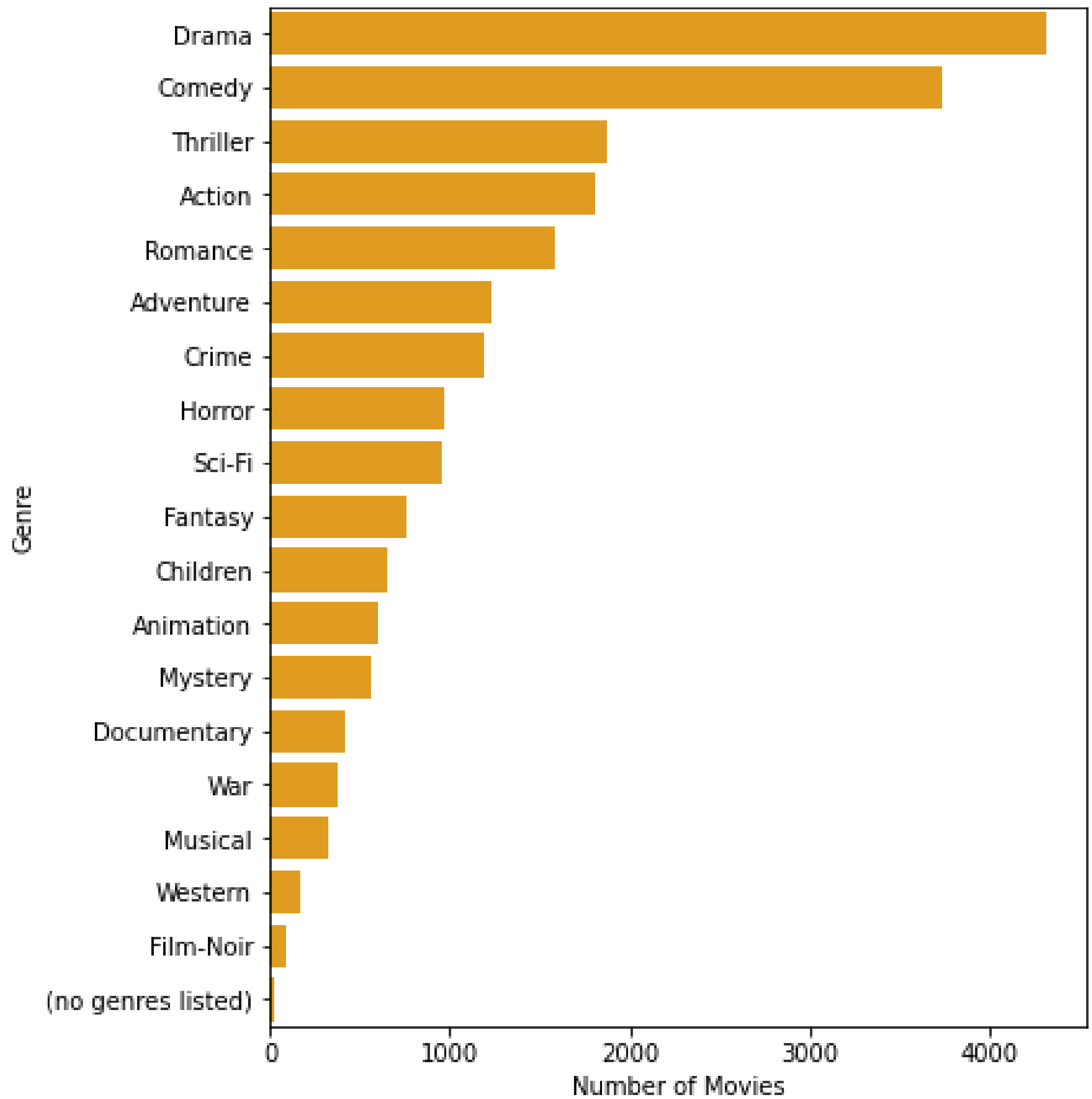
MOST POPULAR GENRES Top 5 popular Genres after doing analysis are, 1. Adventure 2. Animation 3. Fantasy 4. Children 5. Action

And most unpopular Genre is `Documentary`

1.7.2 Number of movies per genre barplot

[37]:
```
# Extract unique genres and their counts
genre_counts = (
    movies_metadata['genres']
    .explode()
    .value_counts()
    .to_frame()
    .reset_index()
)
genre_counts.columns = ['genre', 'count']
```

[38]:
```
# Inset bar plot for number of genres
ax2 = plt.axes([1, 0.5, 0.8, 1.5])  # [left, bottom, width, height]
sns.barplot(x='count', y='genre', data=genre_counts, color='orange', ax=ax2)
ax2.set_xlabel('Number of Movies')
ax2.set_ylabel('Genre')
plt.show()
```



Exploration of numerical columns There are four numerical columns I am interested to know more with respect to Genre, as I know none of these column has a NaN value, I would like to see, mean, max, and min value for each of these columns

1. Popularity
2. runtime
3. vote_average
4. vote_count

I would like to use pandas describe method for this

[39]:
```
movies_metadata_numeric_data = movies_metadata[['popularity','runtime',␣
↪'vote_average', 'vote_count']]
```

[40]:
```
movies_metadata_numeric_data.describe()
```

[40]:

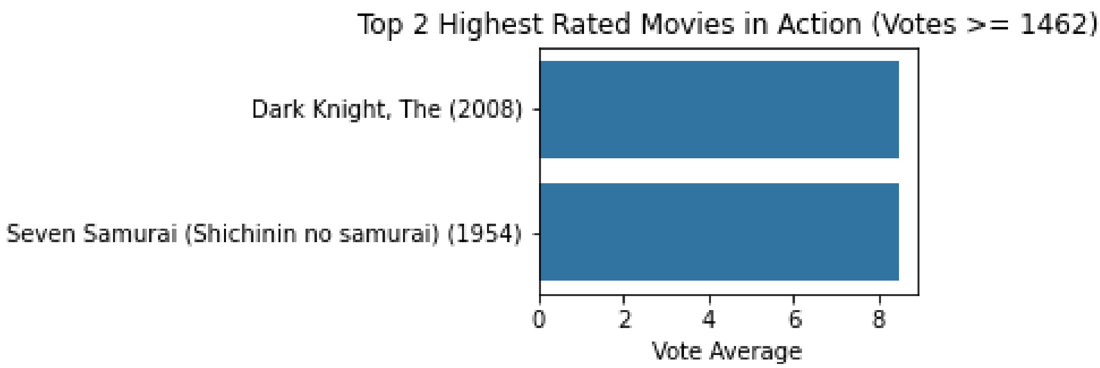
	popularity	runtime	vote_average	vote_count
count	9619.000000	9619.000000	9619.000000	9619.000000
mean	17.823707	104.337665	6.511115	1462.350244
std	17.638178	24.451975	0.866306	2923.100811
min	0.600000	2.000000	0.000000	0.000000
25%	8.769000	92.000000	6.000000	128.000000
50%	13.506000	102.000000	6.564000	410.000000
75%	21.085000	115.000000	7.115000	1374.500000
max	679.514000	583.000000	8.917000	34704.000000

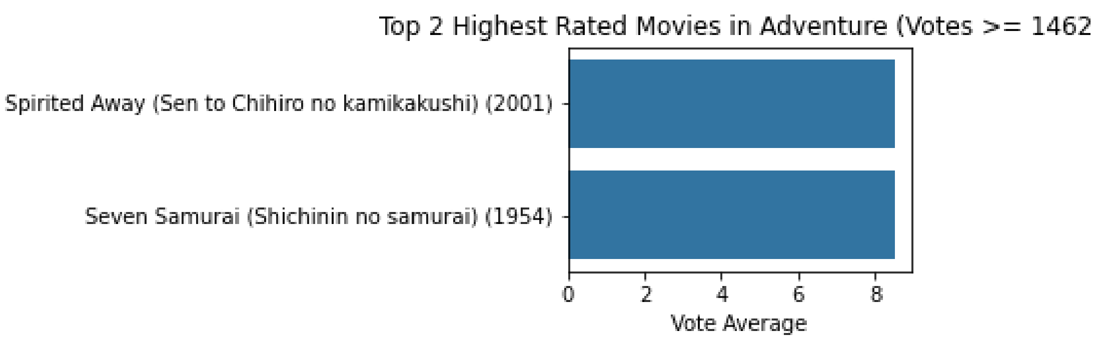
Nice statistics, let me now explore all the genres where `vote_count` is greater then the mean vote_count value and I will then plot top 2 movies from each genre

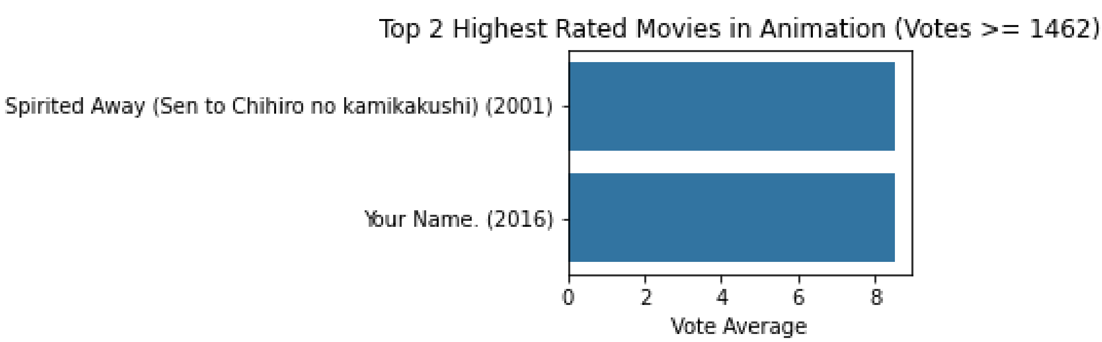
1.7.3 Barplot of each genre by vote_average

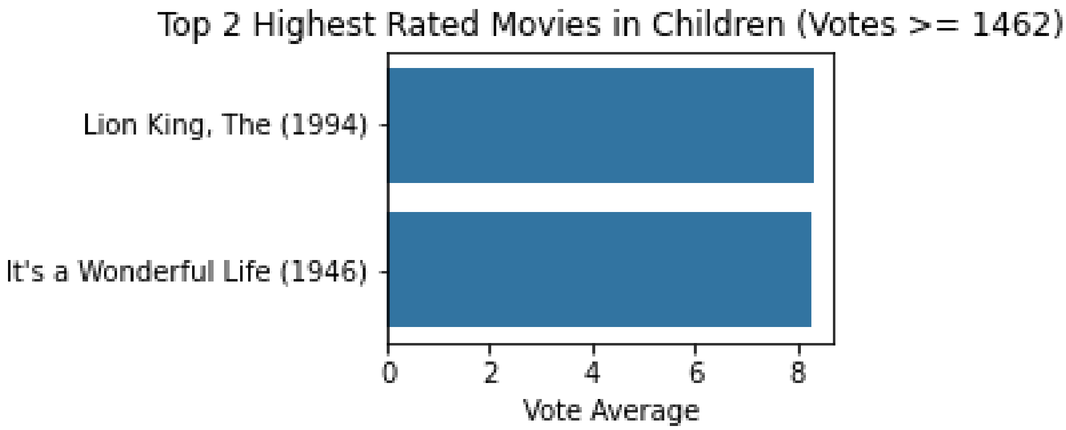
I am excluding two genres '(no genres listed)', 'Film-Noir' as not much movies are there in these two genres

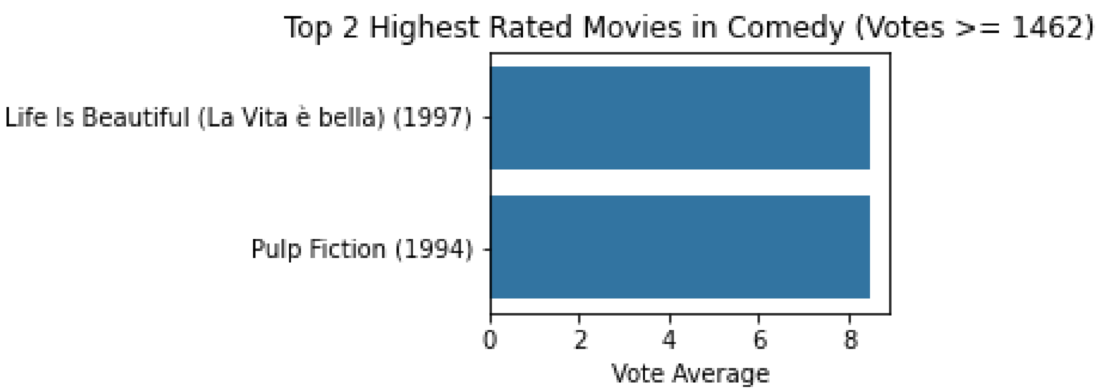
[41]:
```
filtered_movies = df_genres[df_genres['vote_count'] >= 1462]
filtered_movies = filtered_movies[~filtered_movies['genres'].isin(['(no genres␣
↪listed)', 'Film-Noir'])]

# Group by genre, sort within each group by vote_average, and select the top 2␣
↪for each genre
top_movies_by_genre = filtered_movies.groupby('genres').apply(lambda x: x.
↪nlargest(2, 'vote_average')).reset_index(drop=True)

# Create separate bar plots for each genre
for genre, data in top_movies_by_genre.groupby('genres'):
    plt.figure(figsize=(3, 2))
    sns.barplot(x='vote_average', y='title', data=data, errorbar=None)
    plt.xlabel('Vote Average')
    plt.ylabel('')
    plt.title(f'Top 2 Highest Rated Movies in {genre} (Votes >= 1462)')
    plt.show()
```

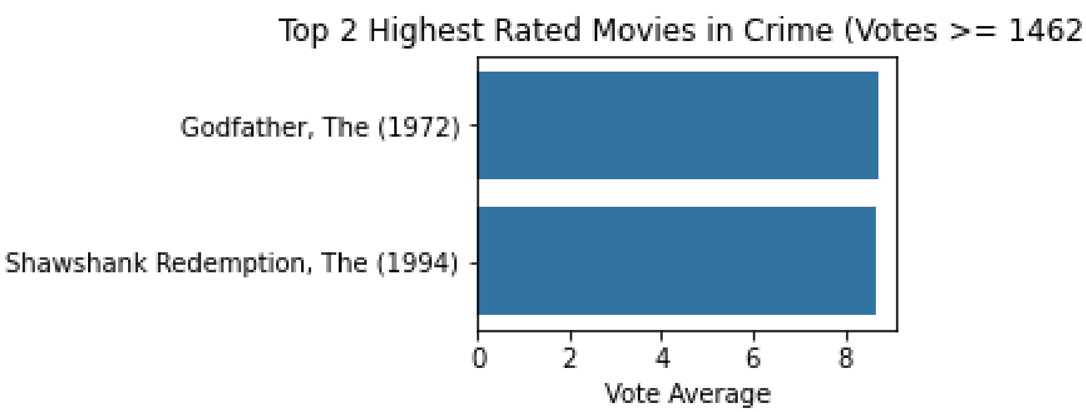

Top 2 Highest Rated Movies in Action (Votes >= 1462)
Dark Knight, The (2008)
Seven Samurai (Shichinin no samurai) (1954)
0
2
4
6
8
Vote Average

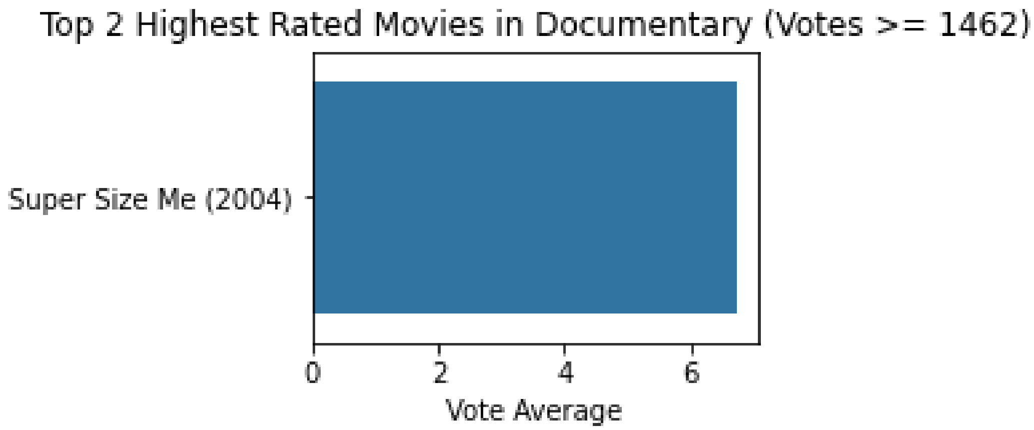

Top 2 Highest Rated Movies in Adventure (Votes >= 1462)
Spirited Away (Sen to Chihiro no kamikakushi) (2001)
Seven Samurai (Shichinin no samurai) (1954)
0
2
4
6
8
Vote Average

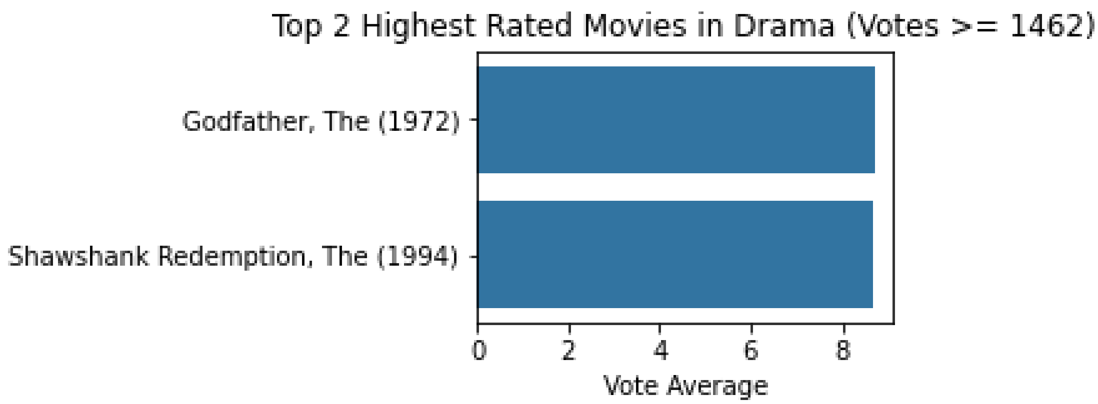

Top 2 Highest Rated Movies in Animation (Votes >= 1462)
Spirited Away (Sen to Chihiro no kamikakushi) (2001)
Your Name. (2016)
0
2
4
6
8
Vote Average

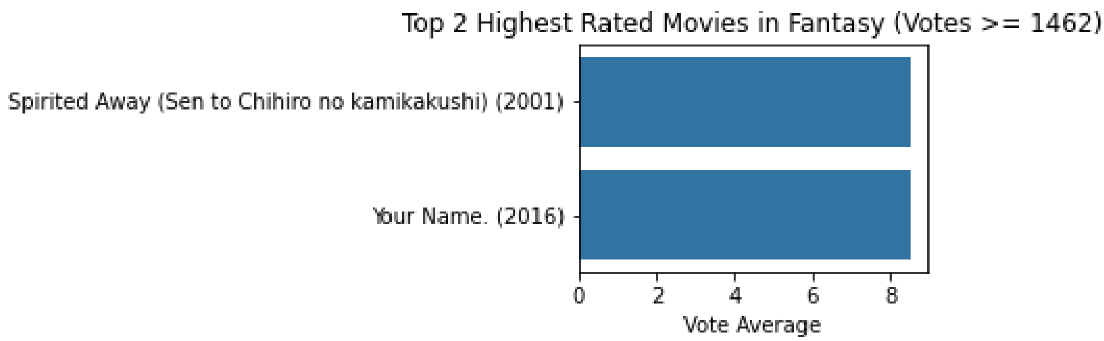

Top 2 Highest Rated Movies in Children (Votes >= 1462)
Lion King, The (1994)
It's a Wonderful Life (1946)
0
2
4
6
8
Vote Average

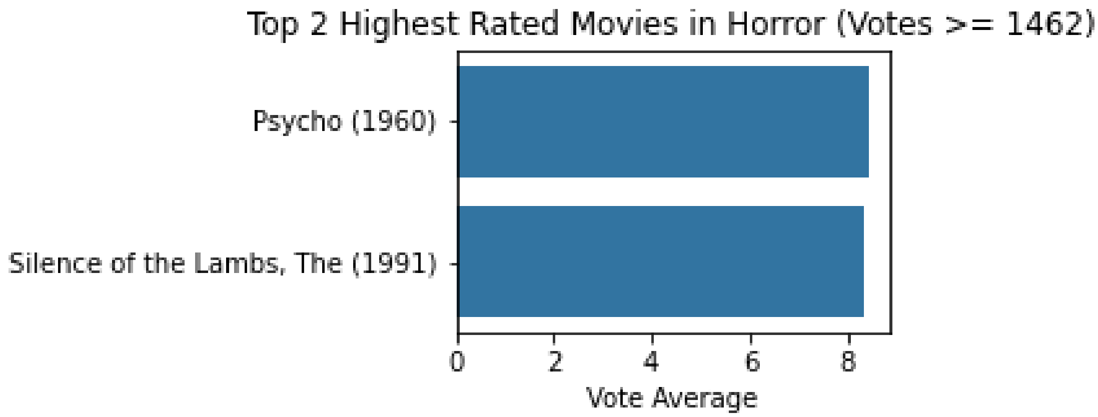

Top 2 Highest Rated Movies in Comedy (Votes >= 1462)
Life Is Beautiful (La Vita è bella) (1997)
Pulp Fiction (1994)
0
2
4
6
8
Vote Average

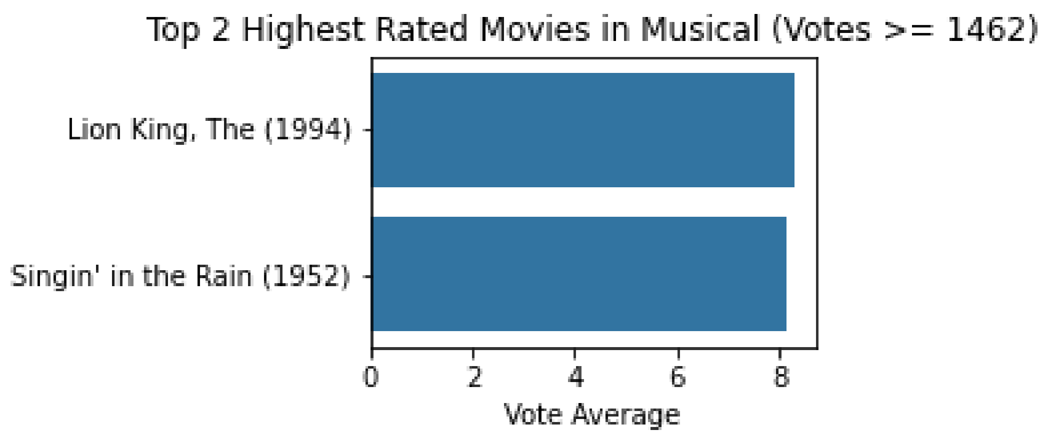

Top 2 Highest Rated Movies in Crime (Votes >= 1462)
Godfather, The (1972)
Shawshank Redemption, The (1994)
0
2
4
6
8
Vote Average

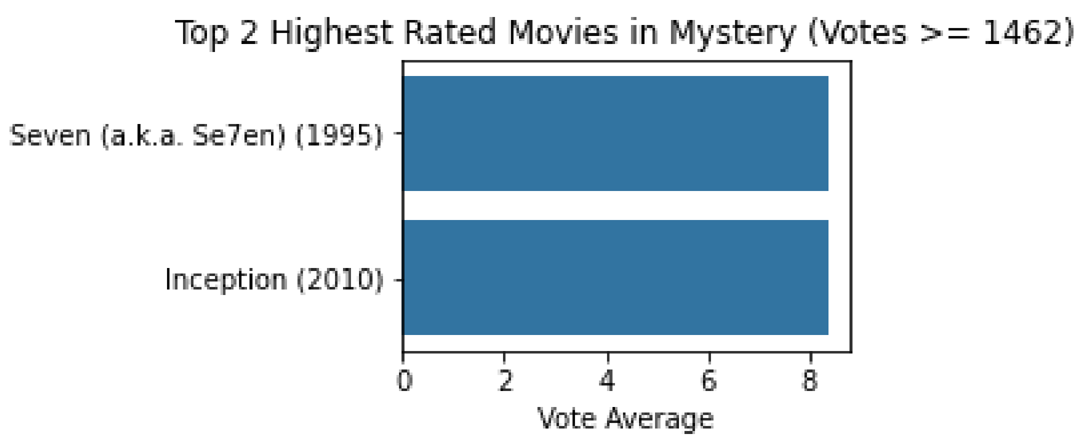

Top 2 Highest Rated Movies in Documentary (Votes >= 1462)
Super Size Me (2004)
0
2
4
6
Vote Average

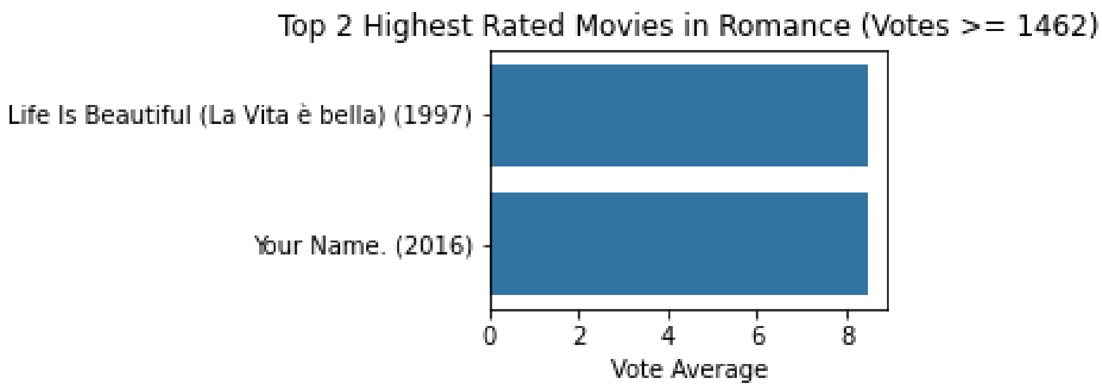

Top 2 Highest Rated Movies in Drama (Votes >= 1462)
Godfather, The (1972)
Shawshank Redemption, The (1994)
0
2
4
6
8
Vote Average

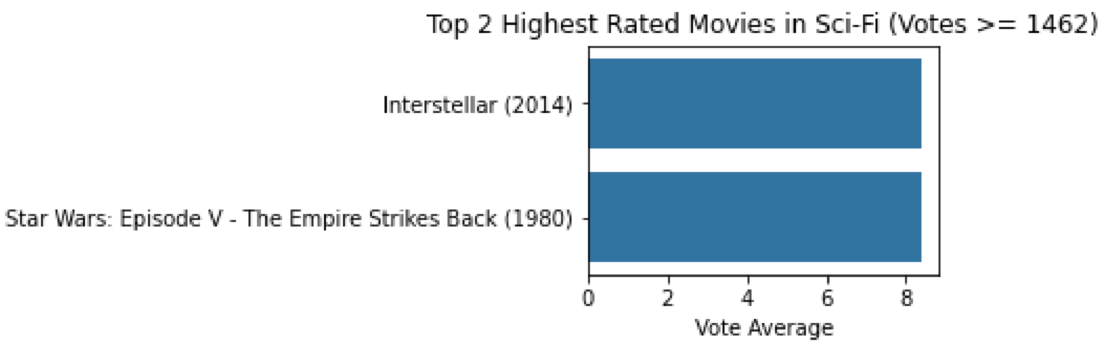

Top 2 Highest Rated Movies in Fantasy (Votes >= 1462)
Spirited Away (Sen to Chihiro no kamikakushi) (2001)
Your Name. (2016)
0
2
4
6
8
Vote Average

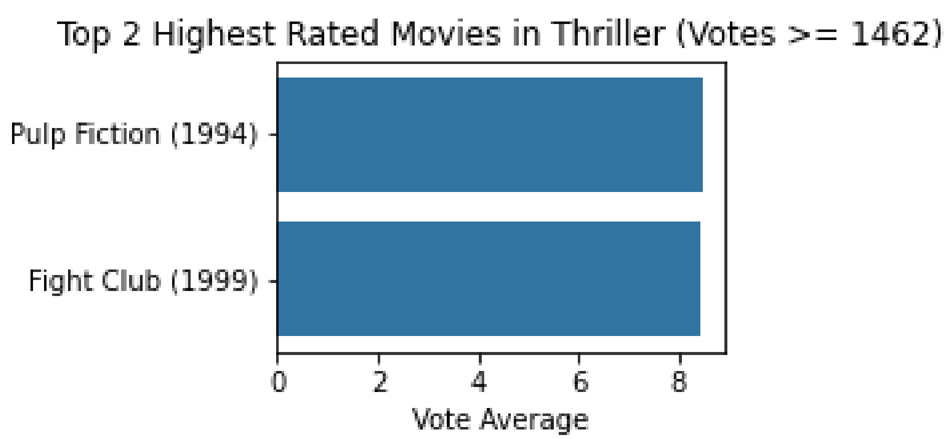

Top 2 Highest Rated Movies in Horror (Votes >= 1462)
Psycho (1960)
Silence of the Lambs, The (1991)
0
2
4
6
8
Vote Average

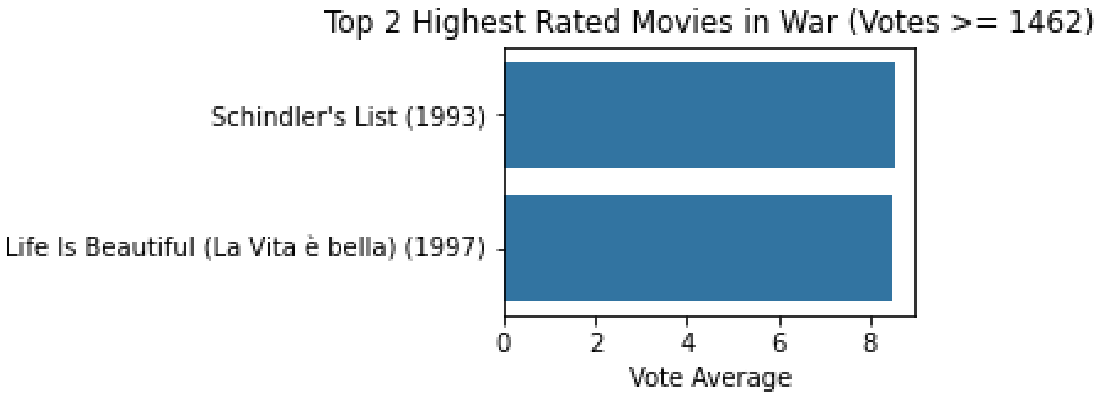

Top 2 Highest Rated Movies in Musical (Votes >= 1462)
Lion King, The (1994)
Singin' in the Rain (1952)
0
2
4
6
8
Vote Average

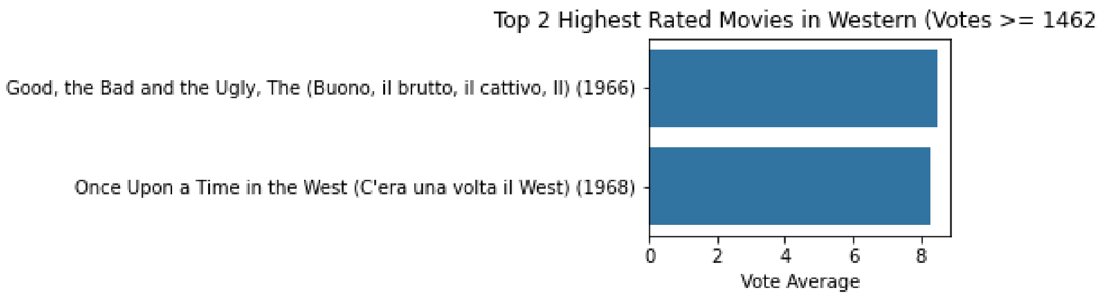

Top 2 Highest Rated Movies in Mystery (Votes >= 1462)
Seven (a.k.a. Se7en) (1995)
Inception (2010)
0
2
4
6
8
Vote Average


Top 2 Highest Rated Movies in Romance (Votes >= 1462)
Life Is Beautiful (La Vita è bella) (1997)
Your Name. (2016)
0
2
4
6
8
Vote Average


Top 2 Highest Rated Movies in Sci-Fi (Votes >= 1462)
Interstellar (2014)
Star Wars: Episode V - The Empire Strikes Back (1980)
0
2
4
6
8
Vote Average

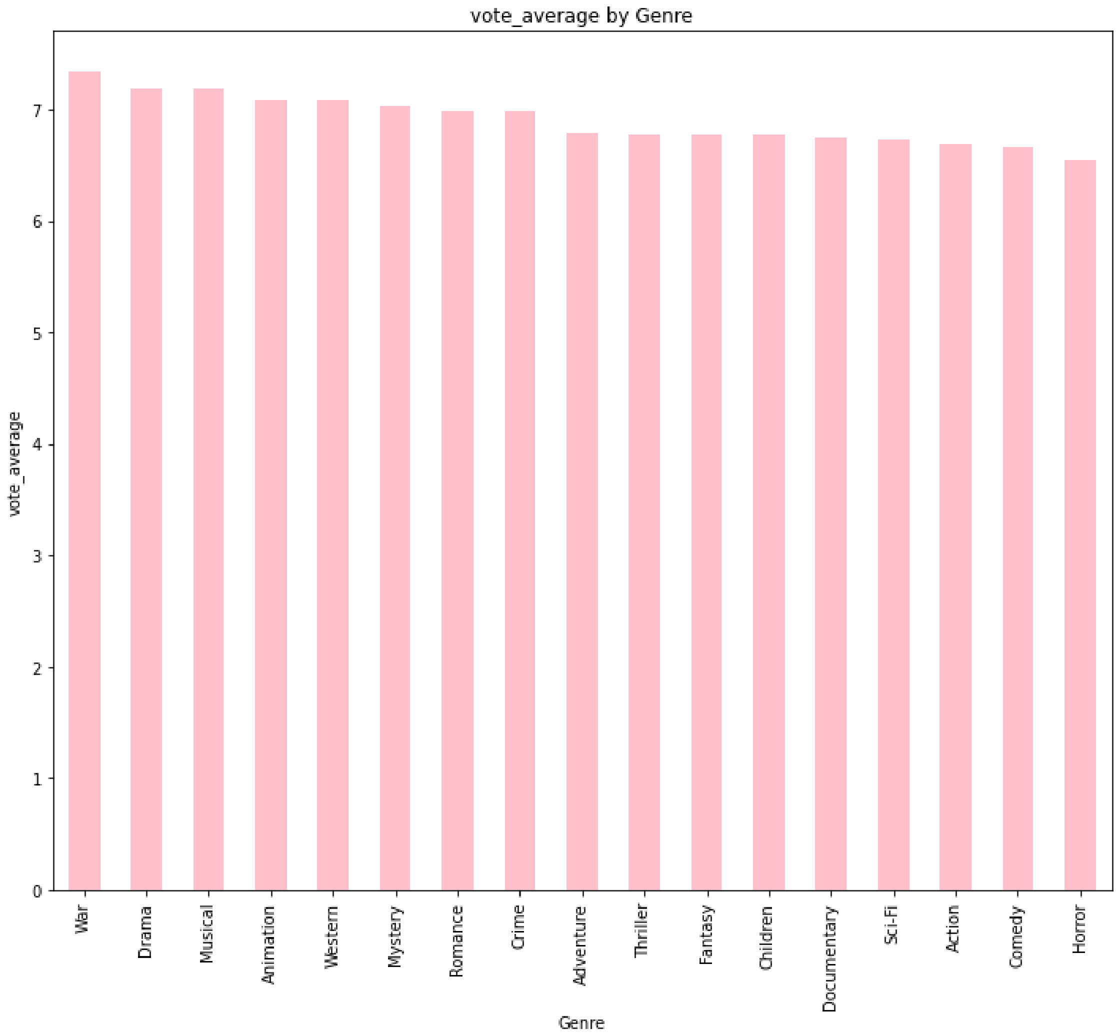

Top 2 Highest Rated Movies in Thriller (Votes >= 1462)
Pulp Fiction (1994)
Fight Club (1999)
0
2
4
6
8
Vote Average





1.7.4 Bar plot based on the average vote for each Genres

[42]:
```
# Convert the genres column to a list
# # # Calculate the average popularity for each genre
filtered_movies = filtered_movies[~filtered_movies['genres'].isin(['(no genres␣
↪listed)', 'Film-Noir'])]
avg_vote_count_by_genre = filtered_movies.groupby('genres')['vote_average'].
↪mean()
sorted_genres = avg_vote_count_by_genre.sort_values(ascending=False)

# Plot the results
sorted_genres.plot(kind='bar', color='pink', figsize=(12, 10))
plt.title('vote_average by Genre')
plt.xlabel('Genre')
plt.ylabel('vote_average')
plt.show()
```



Looks like almost all the genres were voted equally

1.8 Genre-based recommendation

To create a genre-based recommendation system, I am going to use using two dataframes (`movies_metadata` and `ratings`), `ratings` and `movies_metadata` dataframe can be joined to understand what kind of specific Genre a person would have admired the most

1. Data Preprocessing:
    - I will merge the two dataframes based on the movieId to create a unified dataframe.
    - Extract relevant information such as movie titles, genres, userId, ratings.
2. Genre-Based Recommendations:
    - I will calculate the average rating given by the user for each genre.
    - will identify genres that the user has liked based on their ratings.
    - Filter movies that a user haven't yet and

3. Recommendation Generation:

   - Recommend movies from genres that the user has liked, considering high average ratings.

[43]:
```
movies_metadata.head()
```

[43]:
```
   movieId                        title  \
0        1             Toy Story (1995)
1        2               Jumanji (1995)
2        3      Grumpier Old Men (1995)
3        4     Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)

                                             genres  imdbId  tmdbId  \
0  [Adventure, Animation, Children, Comedy, Fantasy]  114709     862
1                    [Adventure, Children, Fantasy]  113497    8844
2                                 [Comedy, Romance]  113228   15602
3                          [Comedy, Drama, Romance]  114885   31357
4                                          [Comedy]  113041   11862

                                            overview  popularity  runtime  \
0  Led by Woody, Andy's toys live happily in his …     100.954       81
1  When siblings Judy and Peter discover an encha…      13.981      104
2  A family wedding reignites the ancient feud be…      12.686      101
3  Cheated on, mistreated and stepped on, the wom…      11.945      127
4  Just when George Banks has recovered from his …      19.558      106

   vote_average  vote_count  \
0         7.970       17277
1         7.239        9891
2         6.494         350
3         6.183         142
4         6.239         665

                                             tagline  \
0  Hang on for the comedy that goes to infinity a…
1          Roll the dice and unleash the excitement!
2  Still Yelling. Still Fighting. Still Ready for…
3  Friends are the people who let you be yourself…
4  Just when his world is back to normal… he's …

                                                cast
0  Tom Hanks|Tim Allen|Don Rickles|Jim Varney|Wal…
1  Robin Williams|Kirsten Dunst|Bradley Pierce|Bo…
2  Walter Matthau|Jack Lemmon|Ann-Margret|Sophia …
3  Whitney Houston|Angela Bassett|Loretta Devine|…
4  Steve Martin|Diane Keaton|Martin Short|Kimberl…
```

1.8.1 Merge two dataframes

After merging I will be try to get what is the minimum and max value in userId, movieId and rating column

[44]:
```
ratings_movies = pd.merge(ratings, movies_metadata[['movieId', 'title',␣
↪'genres']], on='movieId')
```

[45]:
```
minValuesObj = ratings_movies[['userId', 'movieId', 'rating']].min()
print(minValuesObj)
```

```
userId     1.0
movieId    1.0
rating     0.5
dtype: float64
```

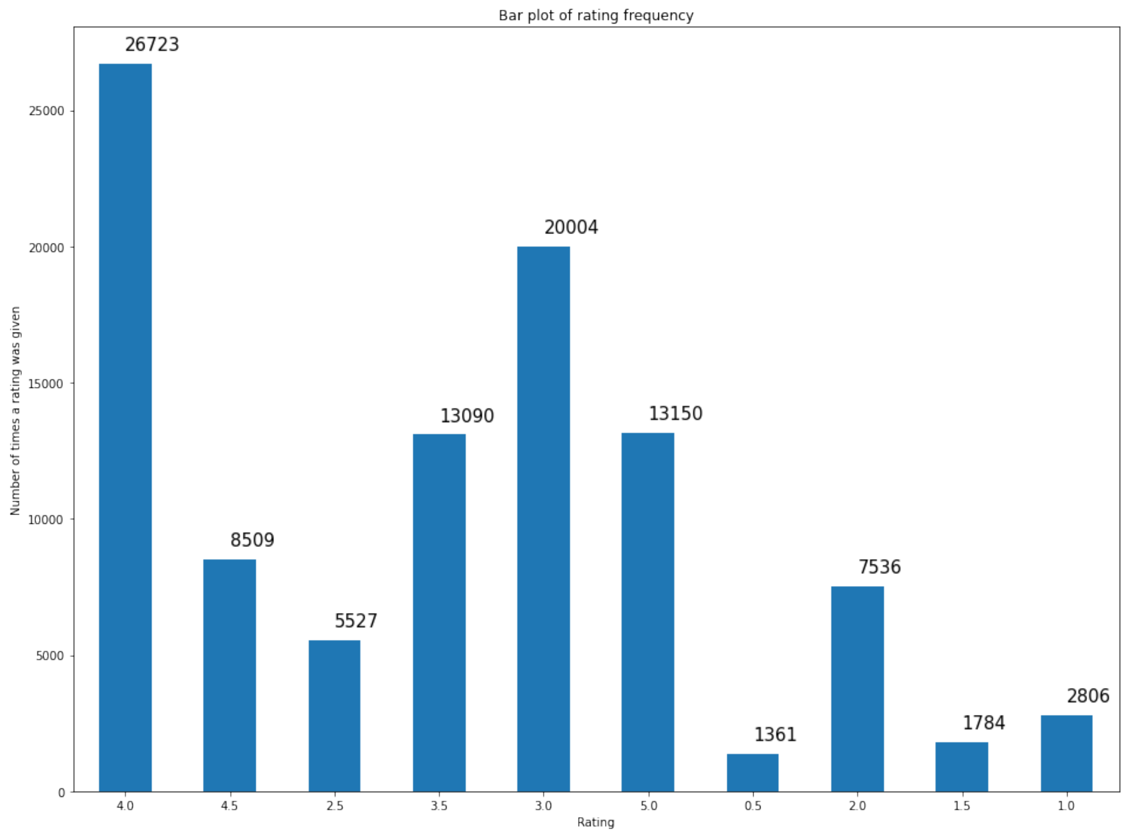
[46]:
```
maxValuesObj = ratings_movies[['userId', 'movieId', 'rating']].max()
print(maxValuesObj)
```

```
userId        610.0
movieId    193609.0
rating          5.0
dtype: float64
```

1.8.2 Data visualization

I will plot a bar graph that can show what is the total number of count a movie is being rating between 0.5-5

[47]:
```
# Count the frequency of each rating
rating_counts = ratings_movies['rating'].value_counts(sort=False)

# Plotting the bar chart
rating_counts.plot(kind='bar', figsize=(16, 12), use_index=True, rot=0)
plt.title('Bar plot of rating frequency')
plt.xlabel('Rating')
plt.ylabel('Number of times a rating was given')

# Adding labels to the bars
for i, v in enumerate(rating_counts):
    plt.text(i , v + 500, str(v), size=15)

plt.show()
```



The above bar graph gives me an idea, 1. most frequent ratings are in either 4 or 3, 2. frequency of getting movie rating 3.5 and 5.0 is almost similar 3. Very few chances are there for someone to rate a movie 0.5 or 1.5 or even 1.0

[48]:
```
def genre_based_recommendation(user_id):

    # Step 1: Filter movies not seen by the user
    user_movies = ratings_movies[ratings_movies['userId'] == user_id]['title']
    movies_not_seen = ratings_movies[~ratings_movies['title'].isin(user_movies)]


    movie_genres = movies_not_seen.explode('genres')


    # Step 2: Calculate average ratings for each genre
    genre_ratings = movie_genres.groupby('genres')['rating'].mean().
↪reset_index()

    # Step 3: Identify user's liked genres
    """
```

```
    I did analysis and found that mostly all the genres were rated euqally, and␣
↪rating range was 3-4
    so I decided to take rating threshold value as 3.5
    """
    user_liked_genres = genre_ratings[genre_ratings['rating'] >= 3.5]['genres']

    # print(user_liked_genres)
    # Step 4: Filter movies from liked genres
    liked_movies = movie_genres[movie_genres['genres'].isin(user_liked_genres)]


    # Step 5: Sort by ratings (descending order)
    liked_movies_sorted = liked_movies.groupby('title')['rating'].mean().
↪sort_values(ascending=False).reset_index()

    # Step 6: Recommend top movies
    top_recommendations = liked_movies_sorted.head(5)

    return top_recommendations
```

Let's invoke the function `genre_based_recommendation` with randomly generated user_ids

[49]:
```
import random

user_id = random.randint(1, 610)

print(f"Top 5 movie Recommendations for User {user_id}: \n",␣
 ↪genre_based_recommendation(user_id))
```

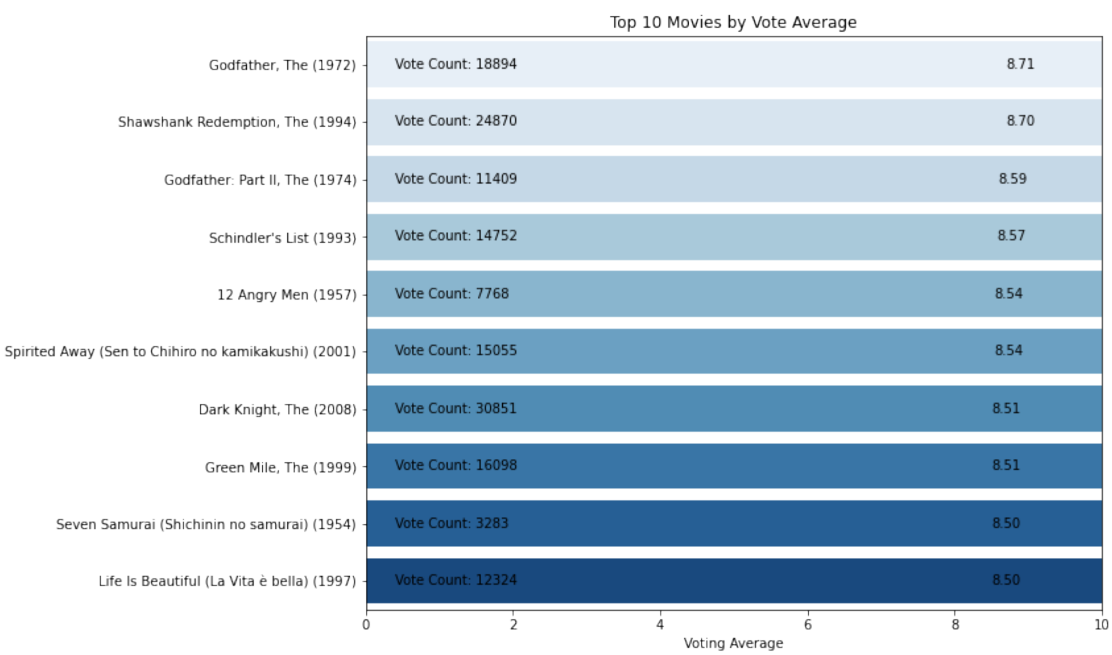
```
Top 5 movie Recommendations for User 287:
                                                 title  rating
0                           Gena the Crocodile (1969)     5.0
1                                           PK (2014)     5.0
2                                      Buzzard (2015)     5.0
3  Battle Royale 2: Requiem (Batoru rowaiaru II: …     5.0
4                         Battle For Sevastopol (2015)     5.0
```

1.9 Movie Cast based Analysis

1. Next, I am displaying the top 10 movies with the highest `vote_average` that received at least mean votes meaning `vote_count > 1462`.
2. I will again use `movies_metadata` dataframe and this time I will plot some bar graphs based on the cast of the movies

[50]:
```
# Filter movies with vote_count greater than 250
filtered_movies = movies_metadata[movies_metadata['vote_count'] > 1462]
```

```python
# Select the top 20 movies based on vote_average
top_movies = filtered_movies.nlargest(10, 'vote_average')

# Plot the results using seaborn
plt.figure(figsize=(10, 8))
sns.barplot(x='vote_count', y='title', hue='title', data=top_movies,␣
↪palette='Blues', dodge=False, legend=False)
plt.title('Top 10 Movies by Vote Average')
plt.xlabel('Voting Average')
plt.ylabel('')
plt.xlim(0, 10)

# Annotate with vote_average values
for index, value in enumerate(top_movies['vote_average']):
    plt.text(value, index, f'{value:.2f}', va='center', fontsize=10,␣
↪color='black', ha='left')

# Annotate with vote_count values
for index, value in enumerate(top_movies['vote_count']):
    plt.text(0.4, index, f'Vote Count: {value}', va='center', fontsize=10,␣
↪color='black')

plt.show()
```



1.9.1 Plots based on the cast of the movie

[51]:
```
df = movies_metadata.copy()

df.dropna(subset=['cast'], inplace=True)
```

[52]:
```
df['cast'] = df['cast'].str.split('|')

movies_metadata = df.copy()

def clean_cast(cast_list):
    cleaned_cast = []
    for name in cast_list:
        cleaned_name = ''.join(char for char in name if char.isalpha() or char.
↪isspace())
        cleaned_cast.append(cleaned_name.strip())
    return cleaned_cast

# Apply the clean_cast function to each element in the 'cast' column
df['cast'] = df['cast'].apply(clean_cast)
```

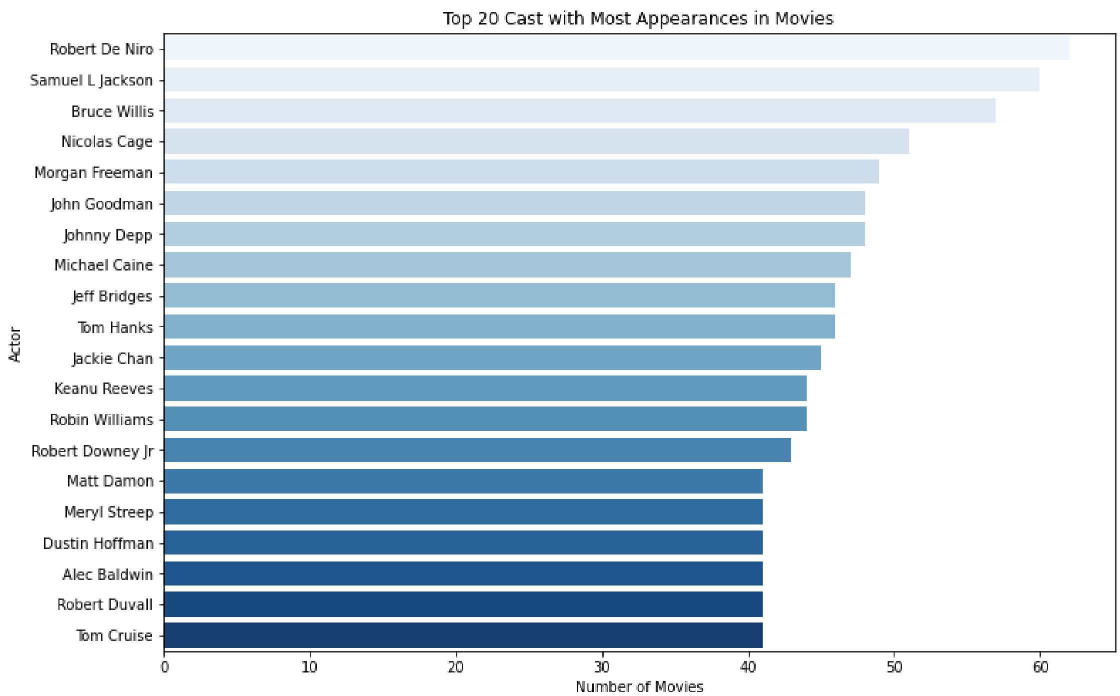
[53]:
```
# Create a new DataFrame with exploded genres
df_cast = df.explode('cast').reset_index(drop=True)
df_cast.head()
```

[53]:
```
   movieId             title  \
0        1  Toy Story (1995)
1        1  Toy Story (1995)
2        1  Toy Story (1995)
3        1  Toy Story (1995)
4        1  Toy Story (1995)

                                               genres  imdbId  tmdbId  \
0  [Adventure, Animation, Children, Comedy, Fantasy]  114709     862
1  [Adventure, Animation, Children, Comedy, Fantasy]  114709     862
2  [Adventure, Animation, Children, Comedy, Fantasy]  114709     862
3  [Adventure, Animation, Children, Comedy, Fantasy]  114709     862
4  [Adventure, Animation, Children, Comedy, Fantasy]  114709     862

                                            overview  popularity  runtime  \
0  Led by Woody, Andy's toys live happily in his …     100.954       81
1  Led by Woody, Andy's toys live happily in his …     100.954       81
2  Led by Woody, Andy's toys live happily in his …     100.954       81
3  Led by Woody, Andy's toys live happily in his …     100.954       81
4  Led by Woody, Andy's toys live happily in his …     100.954       81

   vote_average  vote_count  \
```

```
0          7.97       17277
1          7.97       17277
2          7.97       17277
3          7.97       17277
4          7.97       17277

                                               tagline            cast
0  Hang on for the comedy that goes to infinity a…      Tom Hanks
1  Hang on for the comedy that goes to infinity a…      Tim Allen
2  Hang on for the comedy that goes to infinity a…    Don Rickles
3  Hang on for the comedy that goes to infinity a…     Jim Varney
4  Hang on for the comedy that goes to infinity a…  Wallace Shawn
```

[54]:

```python
# Assuming df_cast is your DataFrame

# Group by cast and count the number of appearances
cast_appearances = df_cast['cast'].str.split('|').explode().value_counts()

# Select the top 20 cast with most appearances
top_cast = cast_appearances.nlargest(20)

# Plot the results using Seaborn
plt.figure(figsize=(12, 8))
sns.barplot(x=top_cast.values, y=top_cast.index,hue=top_cast.index,␣
 ↪palette='Blues', legend=False)
plt.xlabel('Number of Movies')
plt.ylabel('Actor')
plt.title('Top 20 Cast with Most Appearances in Movies')
plt.show()
```



[55]:
```
top_movies['cast']=top_movies['cast'].str.split('|')
```

[56]:
```
top_movies.reset_index(drop=True)
```

[56]:
```
   movieId                                              title  \
0      858                             Godfather, The (1972)
1      318                   Shawshank Redemption, The (1994)
2     1221                    Godfather: Part II, The (1974)
3      527                           Schindler's List (1993)
4     1203                               12 Angry Men (1957)
5     5618  Spirited Away (Sen to Chihiro no kamikakushi) …
6    58559                            Dark Knight, The (2008)
7     3147                              Green Mile, The (1999)
8     2019        Seven Samurai (Shichinin no samurai) (1954)
9     2324          Life Is Beautiful (La Vita è bella) (1997)

                          genres  imdbId  tmdbId  \
0                [Crime, Drama]   68646     238
1                [Crime, Drama]  111161     278
2                [Crime, Drama]   71562     240
3                  [Drama, War]  108052     424
4                       [Drama]   50083     389
5  [Adventure, Animation, Fantasy]  245429     129
6         [Action, Crime, Drama]  468569     155
```

```
7                  [Crime, Drama]  120689     497
8       [Action, Adventure, Drama]   47478     346
9   [Comedy, Drama, Romance, War]  118799     637

                                            overview  popularity  runtime  \
0  Spanning the years 1945 to 1955, a chronicle o…     147.845      175
1  Framed in the 1940s for the double murder of h…     121.554      142
2  In the continuing saga of the Corleone crime f…      77.298      202
3  The true story of how businessman Oskar Schind…      63.432      195
4  The defense and the prosecution have rested an…      48.956       97
5  A young girl, Chihiro, becomes trapped in a st…      97.582      125
6  Batman raises the stakes in his war on crime. …     115.589      152
7  A supernatural tale set on death row in a Sout…      71.236      189
8  A samurai answers a village's request for prot…      42.812      207
9  A touching story of an Italian book seller of …      40.478      116

   vote_average  vote_count  \
0         8.710       18894
1         8.704       24870
2         8.591       11409
3         8.571       14752
4         8.542        7768
5         8.540       15055
6         8.512       30851
7         8.508       16098
8         8.500        3283
9         8.500       12324

                                             tagline  \
0                        An offer you can't refuse.
1  Fear can hold you prisoner. Hope can set you f…
2       All the power on earth can't change destiny.
3    Whoever saves one life, saves the world entire.
4   Life is in their hands. Death is on their minds.
5                                                NaN
6                  Welcome to a world without rules.
7                               Miracles do happen.
8  The Mighty Warriors Who Became the Seven Natio…
9  An unforgettable fable that proves love, famil…

                                                cast
0  [Marlon Brando, Al Pacino, James Caan, Robert …
1  [Tim Robbins, Morgan Freeman, Bob Gunton, Will…
2  [Al Pacino, Robert Duvall, Diane Keaton, Rober…
3  [Liam Neeson, Ben Kingsley, Ralph Fiennes, Car…
4  [Martin Balsam, John Fiedler, Lee J. Cobb, E.G…
5  [Rumi Hiiragi, Miyu Irino, Mari Natsuki, Takas…
```

```
6   [Christian Bale, Heath Ledger, Michael Caine, …
7   [Tom Hanks, Michael Clarke Duncan, David Morse…
8   [Toshirō Mifune, Takashi Shimura, Yoshio Inaba…
9   [Roberto Benigni, Nicoletta Braschi, Giorgio C…
```

1.10 Results

Based on the Data Analysis, there are couple of factors that can influence movie recommendation for a user, I can see movie Genre, vote_count, popularity and ratings given by a user himself contributes most to the movie recommendation.

`genre-based recommendation` is an example of content-based recommendation. Content-based recommendation systems make recommendations based on the features or content of the items and the preferences expressed by the user. In the case of movies, the genre is one of the key features or content attributes.

In a genre-based recommendation system, the system recommends items that are similar in terms of their genre to the ones the user has liked or interacted with. For example, if a user has shown a preference for movies in the "Action" genre, the system might recommend other movies that also fall into the "Action" genre.

But there are some limitations of Content-based systems recommend items similar to what a user has already liked. This can result in a "filter bubble" where users are exposed to a narrow set of recommendations, limiting the discovery of new and diverse items. Over time, user preferences may change, but content-based systems might not adapt quickly. If the system doesn't update user profiles effectively, recommendations may become less accurate.

I will applying the **Collaborative Filtering Algorithms to build movie recommender system**

1.11 Collaborative Filtering

I will be using powerful Surprise Library to build a collaborative filter based on single value decomposition.

[57]:
```python
from surprise import Dataset, Reader
from surprise.model_selection import train_test_split
from surprise import SVD
from surprise.model_selection import cross_validate, GridSearchCV
```

1.11.1 Load data

[58]:
```python
# Load data
ratings = pd.read_csv('ratings.csv')
movies = pd.read_csv('movies.csv')

# Create a Surprise dataset
reader = Reader(rating_scale=(1, 5))
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)
```

1.11.2 Select best parameters for model training

[59]:
```
# param_grid = {
#     'n_factors':[5, 10, 20, 100],
#     'reg_all': [0.001, 0.01, 0.1],
#     'n_epochs': [10,20,100]
# }
```

[60]:
```
# gs = GridSearchCV(SVD, param_grid, measures=['rmse', 'mae'], cv=5)
```

[61]:
```
# gs.fit(data)
```

Evaluation metric would be rmse root mean squared error, lesser the rmse value better the performance of the model

[62]:
```
# print(gs.best_score['rmse'])
```

[63]:
```
# print(gs.best_params['rmse'])
```

[64]:
```
data = data.build_full_trainset()
```

1.11.3 Choosing best parameters to train the model

[65]:
```
# Use the SVD algorithm for collaborative filtering
algo = SVD(n_factors=100, n_epochs=100, reg_all=0.1)
algo.fit(data)
```

[65]: <surprise.prediction_algorithms.matrix_factorization.SVD at 0x1774beef0>

[66]:
```
def get_movie_recommendations(user_id, model, n=5):
    # Get a list of all movie IDs
    all_movie_ids = ratings['movieId'].unique()

    # Get movie IDs not rated by the user
    user_unrated_movies = [movie_id for movie_id in all_movie_ids if movie_id␣
 ↪not in [item_inner_id for (item_inner_id, _) in data.ur[data.
 ↪to_inner_uid(user_id)]]]

    # Predict ratings for unrated movies
    predictions = [model.predict(user_id, movie_id) for movie_id in␣
 ↪user_unrated_movies]

    # Sort predictions by estimated rating
    predictions.sort(key=lambda x: x.est, reverse=True)

    # Get the top N recommendations
    top_recommendations = predictions[:n]
```

```
    top_movies_info = [(movies[movies['movieId'] == prediction.iid]['title'].
↪values[0], prediction.est) for prediction in top_recommendations]

    return top_movies_info

# Example: Get recommendations for user_id 1
```

[67]:
```
user_id = random.randint(1, 610)

recommendations = get_movie_recommendations(user_id, algo, n=5)
print(f"Top 5 movie recommendations for user {user_id}:\n{recommendations}")
```

```
Top 5 movie recommendations for user 396:
[('Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)',
4.395290572382037), ('Trial, The (Procès, Le) (1962)', 4.334291289686203),
('Holy Mountain, The (Montaña sagrada, La) (1973)', 4.3233374926438914), ('Grand
Day Out with Wallace and Gromit, A (1989)', 4.293576636850696), ('Jetée, La
(1962)', 4.273299147150454)]
```

1.11.4 Convert model in deployable format

[68]:
```
# Example for scikit-learn
from joblib import dump

# Assuming `model` is your trained model
dump(algo, 'recommendation_model.joblib')
```

[68]: ['recommendation_model.joblib']

[ ]: