ndutakanyora / Movie-Industry-Research Public

forked from learn-co-curriculum/dsc-project-template

Code Pull requests Actions Projects Wiki Security Insights Se

template-mvp

Movie-Industry-Research / dsc-phase1-project-template.ipynb

ndutakanyora commit

2 contributors

2525 lines (2525 sloc) | 232 KB

MOVIE INDUSTRY RESEARCH.

Authors: Susan Nduta Kanyora.

Overview

The project is based on the entertainment industry specifically movie production.The entertainment industry is fast evolving and it is only prudent for stakeholders such as production companies to keep up with the current trends.It has been seen in recent years that consumers are moving away from going to movie theatres to streaming movies.The creation of movies streaming sites as Netflix,Amazon Prime Video just to mention a few has made it possible to analyze the behaviour of consumers.The analysis helps production companies to make decisions based on which movie genres are popular,average ratings and how many minutes an average consumer spends on the streaming platforms.

Business Problem

The business problem being addressed is to provide insights into the success and popularity of movies, which can help production companies make informed decisions regarding the type of movies to produce and invest in.

The data questions that we plan to answer to solve this problem are:

1.What are the highest-rated movies of all time? 2.Which genres have the highest average rating? 3.What is the relationship between a movie's budget and its box office revenue? 4.Which directors have the highest average rating for their movies? 5.What is the trend in movie ratings and box office revenue over time?

These questions were selected based on their relevance to the business problem and their potential to provide valuable insights into the factors that contribute to the success of a movie. For instance, knowing the highest-rated movies and genres can help production companies understand consumer behaviour and what types of movies are likely to be successful. Understanding the relationship between a

movie's budget and its box office revenue can help production companies allocate their resources effectively. Knowing which directors have the highest average rating can help production companies identify talented filmmakers to work with. Finally, understanding the trend in movie ratings and box office revenue over time can help production companies anticipate changes in audience preferences and adjust their strategies accordingly.

Overall, answering these data questions can help production companies make data-driven decisions that lead to more successful and profitable movies.

Data Understanding

The data being used for this project comes from the zipped **IM.db** database.The data analysis questions are related to movie ratings, genres, budget, revenue, directors, and time, and the database contains information on all of these variables. The dataset includes a range of information related to movies, including movie title, start year, genres, average ratings and runtimes in minutes.

The data represent a sample of movies that have been released in recent years. The sample includes movies from various countries and in various languages, although there may be some bias towards English-language films. The variables included in the dataset are a mix of categorical and numerical variables, such as movie title (categorical), start year (numerical), and runtimes (numerical).

The target variable for this project will depend on the specific data analysis question being asked. For example, the highest-rated movies of all time question will use movie ratings as the target variable, while the relationship between a movie's genre and its average ratings question will use runtimes as the target variable.The range and distribution of values for each variable will also be important to consider when conducting the data analysis.

In [2]:

```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\susan\appdata\local\p
rograms\python\python311\lib\site-packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\susan\appdata\l
ocal\programs\python\python311\lib\site-packages (from matplotlib) (1.0.7)
Requirement already satisfied: cycler>=0.10 in c:\users\susan\appdata\local
\programs\python\python311\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\susan\appdata
\local\programs\python\python311\lib\site-packages (from matplotlib) (4.39.
0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\susan\appdata
\local\programs\python\python311\lib\site-packages (from matplotlib) (1.4.
4)
Requirement already satisfied: numpy>=1.20 in c:\users\susan\appdata\local
\programs\python\python311\lib\site-packages (from matplotlib) (1.24.2)Not
e: you may need to restart the kernel to use updated packages.

Requirement already satisfied: packaging>=20.0 in c:\users\susan\appdata\ro
aming\python\python311\site-packages (from matplotlib) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\susan\appdata\loca
l\programs\python\python311\lib\site-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\susan\appdata\l
ocal\programs\python\python311\lib\site-packages (from matplotlib) (3.0.9)
```

```
ocal\programs\python\python311\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\susan\appda
ta\roaming\python\python311\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\susan\appdata\roaming\p
ython\python311\site-packages (from python-dateutil>=2.7->matplotlib) (1.1
6.0)
```

In [3]:

```
pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\users\susan\appdata\local\prog
rams\python\python311\lib\site-packages (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in c:\users\susan\appda
ta\local\programs\python\python311\lib\site-packages (from seaborn) (1.24.
2)
Requirement already satisfied: pandas>=0.25 in c:\users\susan\appdata\local
\programs\python\python311\lib\site-packages (from seaborn) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in c:\users\susan\ap
pdata\local\programs\python\python311\lib\site-packages (from seaborn) (3.
7.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\susan\appdata\l
ocal\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=
3.1->seaborn) (1.0.7)
Requirement already satisfied: cycler>=0.10 in c:\users\susan\appdata\local
\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=3.1-
>seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\susan\appdata
\local\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,
>=3.1->seaborn) (4.39.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\susan\appdata
\local\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,
>=3.1->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\susan\appdata\ro
aming\python\python311\site-packages (from matplotlib!=3.6.1,>=3.1->seabor
n) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\susan\appdata\loca
l\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=3.1
->seaborn) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\susan\appdata\l
ocal\programs\python\python311\lib\site-packages (from matplotlib!=3.6.1,>=
3.1->seaborn) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\susan\appda
ta\roaming\python\python311\site-packages (from matplotlib!=3.6.1,>=3.1->se
aborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\susan\appdata\local
\programs\python\python311\lib\site-packages (from pandas>=0.25->seaborn)
(2022.7.1)
Requirement already satisfied: six>=1.5 in c:\users\susan\appdata\roaming\p
ython\python311\site-packages (from python-dateutil>=2.7->matplotlib!=3.6.
1,>=3.1->seaborn) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [4]:

```
# Import standard packages
import numpy as np
import pandas as pd
import sqlite3
import matplotlib.pyplot as plt
import seaborn as sns
import zipfile
```

In [5]:

```
#To explore the data
with zipfile .ZipFile('zippedData/im.db.zip') as my_zip:
    zipfile.ZipFile.extractall(my_zip,path='ZippedData')
```

In [6]:

In [6]:
```
#Connecting the database
conn =sqlite3.connect('ZippedData/im.db')
```

In [7]:
```
! ls zippedData
```

```
'ls' is not recognized as an internal or external command,
operable program or batch file.
```

Data Preparation

Data cleaning: Here, we identify and correct errors, inconsistencies, or inaccuracies in the data. For example, we can check for duplicate entries, correct misspellings, and remove irrelevant information such as empty fields or data that do not apply to our analysis.

Data transformation: Here, we transform the data to make it suitable for analysis. This includes converting data types, scaling variables, and creating new variables that might better represent the problem at hand. For example, we can create a new variable that indicates the genre of a movie or the director's name, which can be used in our analysis.

Data integration: In this case, we combine multiple datasets or sources of data to create a unified dataset for analysis. For example, we can integrate the IMDB database with other data sources, such as Box Office Mojo or Rotten Tomatoes, to enrich the dataset and gain additional insights.

Handling missing values or outliers: Missing values or outliers can affect the analysis results. We handle missing values by either deleting them or imputing them using various techniques such as mean or median imputation. Outliers can be handled by either removing them or transforming the data to reduce their impact. For example, we can replace extreme values with the median or use log transformation to reduce the effect of outliers on the analysis.

Data preparation is essential to ensure the quality and accuracy of the analysis. By dropping irrelevant variables and creating new ones, we can focus on the variables that are most important for the analysis goal. Handling missing values and outliers can improve the quality of the analysis by reducing bias and increasing the reliability of the results. By integrating data from multiple sources, we can create a more comprehensive dataset that provides a more complete view of the problem at hand.

In [8]:
```
# Here you run your code to clean the data
pd.read_sql("""
SELECT *
FROM sqlite_schema
WHERE type = 'table'
""" , conn)
```

Out[8]:

	type	name	tbl_name	rootpage	sql
0	table	movie_basics	movie_basics	2	CREATE TABLE "movie_basics" (\n"movie_id" TEXT...

1	table	directors	directors	3	CREATE TABLE "directors" (\n"movie_id" TEXT,\n...
2	table	known_for	known_for	4	CREATE TABLE "known_for" (\n"person_id" TEXT,\...
3	table	movie_akas	movie_akas	5	CREATE TABLE "movie_akas" (\n"movie_id" TEXT,\...
4	table	movie_ratings	movie_ratings	6	CREATE TABLE "movie_ratings" (\n"movie_id" TEX...
5	table	persons	persons	7	CREATE TABLE "persons" (\n"person_id" TEXT,\n ...
6	table	principals	principals	8	CREATE TABLE "principals" (\n"movie_id" TEXT,\...
7	table	writers	writers	9	CREATE TABLE "writers" (\n"movie_id" TEXT,\n ...

In [9]:

```
movies_ratings = None
movies_ratings = pd.read_sql("""
SELECT *
FROM movie_basics
LEFT JOIN movie_ratings
    USING(movie_id)
""" , conn)

movies_ratings
```

Out[9]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crin
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biograpl
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Come
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Dram
...	...	...	...	...	...	
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Doc
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Doc

146144 rows × 8 columns

In [10]:
```
movies_ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 8 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   movie_id         146144 non-null  object
 1   primary_title    146144 non-null  object
 2   original_title   146123 non-null  object
 3   start_year       146144 non-null  int64
 4   runtime_minutes  114405 non-null  float64
 5   genres           140736 non-null  object
 6   averagerating    73856 non-null   float64
 7   numvotes         73856 non-null   float64
dtypes: float64(3), int64(1), object(4)
memory usage: 8.9+ MB
```

In [11]:
```
#Inspecting for missing values
movies_ratings.isna().sum()
```

Out[11]:
```
movie_id               0
primary_title          0
original_title        21
start_year             0
runtime_minutes    31739
genres              5408
averagerating      72288
numvotes           72288
dtype: int64
```

In [12]:
```
#Eliminating titles with no votes or ratings
movies_ratings.dropna(subset=['numvotes'], inplace=True)
```

In [13]:
```
#Checking if the missing values were eliminated
print('Number of null ratings:', movies_ratings['averagerating'].isna().su
print('Number of null vote counts:', movies_ratings['numvotes'].isna().sum
```

```
Number of null ratings: 0
Number of null vote counts: 0
```

In [14]:
```
#Eliminating titles with no genre listed
movies_ratings.dropna(subset=['genres'], inplace=True)
```

In [15]:
```
#Checking for duplicated values
movies_ratings.duplicated().sum()
```

Out[15]:
```
0
```

In [16]:
```
movies_ratings.duplicated(subset='original_title').sum()
```

Out[16]:
```
2707
```

In [17]:
```
movies_ratings[movies_ratings.duplicated(subset=['original_title', 'runtim
```

Out[17]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	
2658	tt10275936	Raggarjävlar (Swedish Greasers)	Raggarjävlar (Swedish Greasers)	2019	70.0	
11830	tt1644694	The Gift	The Gift	2010	NaN	
12984	tt1674217	Transit	Transit	2010	80.0	Biography,D
19111	tt1825978	The Artist	The Artist	2011	100.0	
23887	tt1967651	Unconditional Love	Unconditional	2012	92.0	
24139	tt1977822	Inside	Inside	2012	85.0	
33380	tt2246595	Blood Money	Blood Money	2012	109.0	Ac
37698	tt2363471	The Summit	The Summit	2012	95.0	Adve
47280	tt2805202	Rise of the Undead	Rise of the Undead	2013	70.0	
50941	tt3019098	The Last Act	The Last Act	2012	NaN	
72877	tt4156972	Opening Night	Opening Night	2016	90.0	
80877	tt4649330	Eso que llaman amor	Eso que llaman amor	2015	NaN	
88715	tt5136180	A Courtship	A Courtship	2015	71.0	
103321	tt6052236	The Wonderful Digby	The Wonderful Digby	2016	82.0	Biography,D
103646	tt6073736	Almost Dead	Almost Dead	2016	85.0	
109186	tt6417762	Happy New Year	Happy New Year	2017	NaN	
116144	tt6896536	Foxtrot	Foxtrot	2017	113.0	
140322	tt9097086	Together	Together	2018	84.0	

In [18]:
```
#Sorting the dataset by vote count
movies_ratings.sort_values(by='numvotes', ascending=False, inplace=True)
```

In [19]:
```
#Eliminating the duplicates
movies_ratings.drop_duplicates(subset=['original_title', 'runtime_minutes'
```

In [20]:
```
movies_ratings.head()
```

Out[20]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	g
7066	tt1375666	Inception	Inception	2010	148.0	Action,Adventu
6900	tt1345836	The Dark Knight Rises	The Dark Knight Rises	2012	164.0	Action,T

311	tt0816692	Interstellar	Interstellar	2014	169.0	Adventure,Dran
20342	tt1853728	Django Unchained	Django Unchained	2012	165.0	Drama,W
356	tt0848228	The Avengers	The Avengers	2012	143.0	Action,Adventu

In [21]:

```
#Creating a new dataframe
clean_genres = movies_ratings.copy()
```

In [22]:

```
#Chenging from string to a list
clean_genres['genres'] = clean_genres['genres'].str.split(',')
clean_genres.head(6)
```

Out[22]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres	av
7066	tt1375666	Inception	Inception	2010	148.0	[Action, Adventure, Sci-Fi]	
6900	tt1345836	The Dark Knight Rises	The Dark Knight Rises	2012	164.0	[Action, Thriller]	
311	tt0816692	Interstellar	Interstellar	2014	169.0	[Adventure, Drama, Sci-Fi]	
20342	tt1853728	Django Unchained	Django Unchained	2012	165.0	[Drama, Western]	
356	tt0848228	The Avengers	The Avengers	2012	143.0	[Action, Adventure, Sci-Fi]	
545	tt0993846	The Wolf of Wall Street	The Wolf of Wall Street	2013	180.0	[Biography, Crime, Drama]	

In [23]:

```
#Cleaning a list of all unique genres,we can iterate through them.
genres_all = set()
genres_column = clean_genres['genres']

for glist in genres_column:
    for g in glist:
        genres_all.add(g)

print(genres_all)
```

```
{'Animation', 'Mystery', 'Action', 'Comedy', 'Crime', 'Romance', 'News', 'S
port', 'Thriller', 'History', 'Documentary', 'Sci-Fi', 'Music', 'War', 'Rea
lity-TV', 'Horror', 'Drama', 'Fantasy', 'Family', 'Game-Show', 'Adventure',
'Adult', 'Short', 'Musical', 'Biography', 'Western'}
```

In [24]:

```
print(f'There are {len(genres_all)} genres in our IMDb dataset.They are:\n
```

```
There are 26 genres in our IMDb dataset.They are:
```

```
 {'Animation', 'Mystery', 'Action', 'Comedy', 'Crime', 'Romance', 'News',
'Sport', 'Thriller', 'History', 'Documentary', 'Sci-Fi', 'Music', 'War', 'R
eality-TV', 'Horror', 'Drama', 'Fantasy', 'Family', 'Game-Show', 'Adventur
e', 'Adult', 'Short', 'Musical', 'Biography', 'Western'}.
```

In [25]:

```
# Using df.explode() to split each row so that it is a singular genre.
expl_clean_genres = clean_genres.explode('genres')
expl_clean_genres.head()
```

Out[25]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres	aver
7066	tt1375666	Inception	Inception	2010	148.0	Action	
7066	tt1375666	Inception	Inception	2010	148.0	Adventure	
7066	tt1375666	Inception	Inception	2010	148.0	Sci-Fi	
6900	tt1345836	The Dark Knight Rises	The Dark Knight Rises	2012	164.0	Action	
6900	tt1345836	The Dark Knight Rises	The Dark Knight Rises	2012	164.0	Thriller	

In [26]:

```
expl_clean_genres['genres'].value_counts()
```

Out[26]:

```
Drama          30784
Documentary    17748
Comedy         17289
Thriller        8212
Horror          7672
Action          6986
Romance         6586
Crime           4610
Adventure       3817
Biography       3807
Family          3411
Mystery         3038
History         2825
Sci-Fi          2206
Fantasy         2126
Music           1967
Animation       1742
Sport           1179
War              853
Musical          721
News             579
Western          280
Reality-TV        17
Adult              3
Game-Show          2
Short              1
Name: genres, dtype: int64
```

In [27]:

```
#Putting together 'movie_id' for each entry in four genres.
titles_in_genres = (expl_clean_genres[expl_clean_genres['genres'].isin(['G
```

In [28]:

```
#Using df.drop() to eliminate the entries of the genres listed above.
clean_genres.drop(index=clean_genres[clean_genres['movie_id'].isin(titles_

for dataset in [clean_genres, expl_clean_genres, movies_ratings]:
```

```
    dataset.drop(
        index=dataset[
            dataset['movie_id'].isin(titles_in_genres)
        ].index,
    inplace=True)
```

In [29]:

```
expl_clean_genres['genres'].value_counts()
```

Out[29]:

```
Drama          30779
Documentary    17738
Comedy         17285
Thriller        8211
Horror          7671
Action          6984
Romance         6586
Crime           4610
Adventure       3815
Biography       3806
Family          3411
Mystery         3038
History         2824
Sci-Fi          2206
Fantasy         2126
Music           1966
Animation       1742
Sport           1179
War              853
Musical          721
News             578
Western          280
Name: genres, dtype: int64
```

In [30]:

```
#Exploring the distributions of averagerating and numvotes.
mean_votes = movies_ratings['numvotes'].mean()
mean_votes
```

Out[30]: 3564.0968895098

In [31]:

```
median_votes = movies_ratings['numvotes'].median()
median_votes
```

Out[31]: 51.0

In [32]:

```
q90_votes = movies_ratings['numvotes'].quantile(0.90)
q90_votes
```

Out[32]: 1621.0

In [33]:

```
#Inspecting the bottom 90% of movies with regards to number of votes
movies_ratings.query(f"numvotes < {q90_votes}").sample(10)
```

Out[33]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	
37243	tt2354069	Hattrick	Hattrick	2012	104.0	Action,Com
63183	tt3628574	State Like Sleep	State Like Sleep	2018	104.0	
16195	tt1754765	Tom Toal: On the	Tom Toal: On the	2010	62.0	Comedy,Dc

		Scrapheap	Scrapheap			
52997	tt3125566	Boomtown	Boomtown	2013	NaN	
143938	tt9609726	A Journey of Happiness	Wan zhuan quan jia fu	2019	92.0	
50537	tt2996228	John Apple Jack	John Apple Jack	2013	89.0	Comedy,Dram
98677	tt5769560	Insomnia Lover	Shi mian nan nu	2016	95.0	
73579	tt4192830	A Wonderful Cloud	A Wonderful Cloud	2015	81.0	Comed
109016	tt6407390	Down on the Farm	Down on the Farm	2017	72.0	
34965	tt2290836	The Healing	The Healing	2012	107.0	Horror,Mys

In [34]:

```
#Dropping these movie titles from the dataset
for dataset in [
    clean_genres,
    expl_clean_genres,
    movies_ratings
]:
    dataset.drop(
        index=dataset.query(f"numvotes < {q90_votes}").index,
        inplace=True
    )
```

In [35]:

```
print(movies_ratings.shape)
```

```
(7304, 8)
```

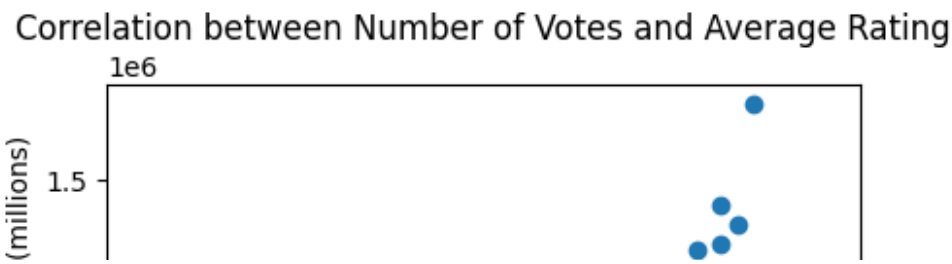
In [36]:

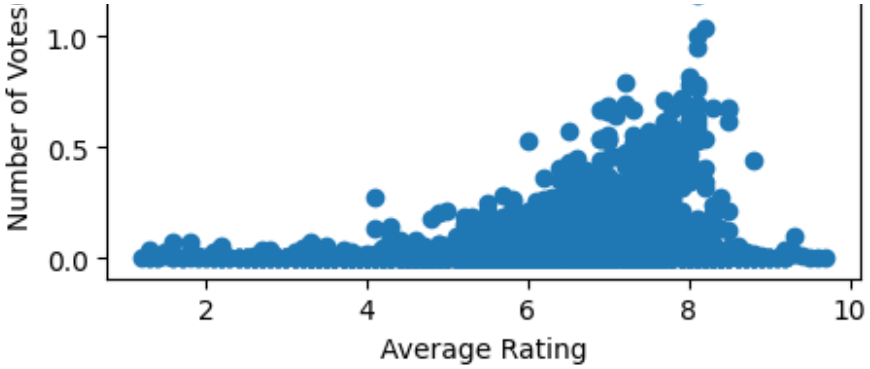
```
#Investigating the correlation between number of votes and average rating
# Create scatter plot
fig, ax = plt.subplots(figsize=(5,3))

ax.scatter(
    x=movies_ratings['averagerating'],
    y=movies_ratings['numvotes'],
)


ax.set_xlabel('Average Rating')
ax.set_ylabel('Number of Votes(millions)')
plt.title('Correlation between Number of Votes and Average Rating')

# Show the plot
plt.show()
```





In [38]:

```
#To model the data
#Using pivot tables showing the average of numvotes by genre

pivot_genres = pd.pivot_table(
    data=expl_clean_genres,
    values=['numvotes'],
    index='genres',
    aggfunc=np.median
).sort_values(by='numvotes', ascending=False).reset_index()

pivot_genres
```

Out[38]:

	genres	numvotes
0	Adventure	16484.0
1	Fantasy	10546.0
2	Sci-Fi	10067.0
3	Animation	9354.0
4	Mystery	8494.0
5	Western	8284.5
6	Action	7543.5
7	Crime	7414.0
8	Biography	6560.0
9	Romance	6540.5
10	Family	6322.0
11	Comedy	6293.0
12	Drama	6081.5
13	Thriller	5920.5
14	Music	5695.0
15	History	5536.0
16	Horror	5415.0
17	War	5347.5
18	Musical	4583.0
19	Sport	4454.0
20	News	3477.0
21	Documentary	3442.0

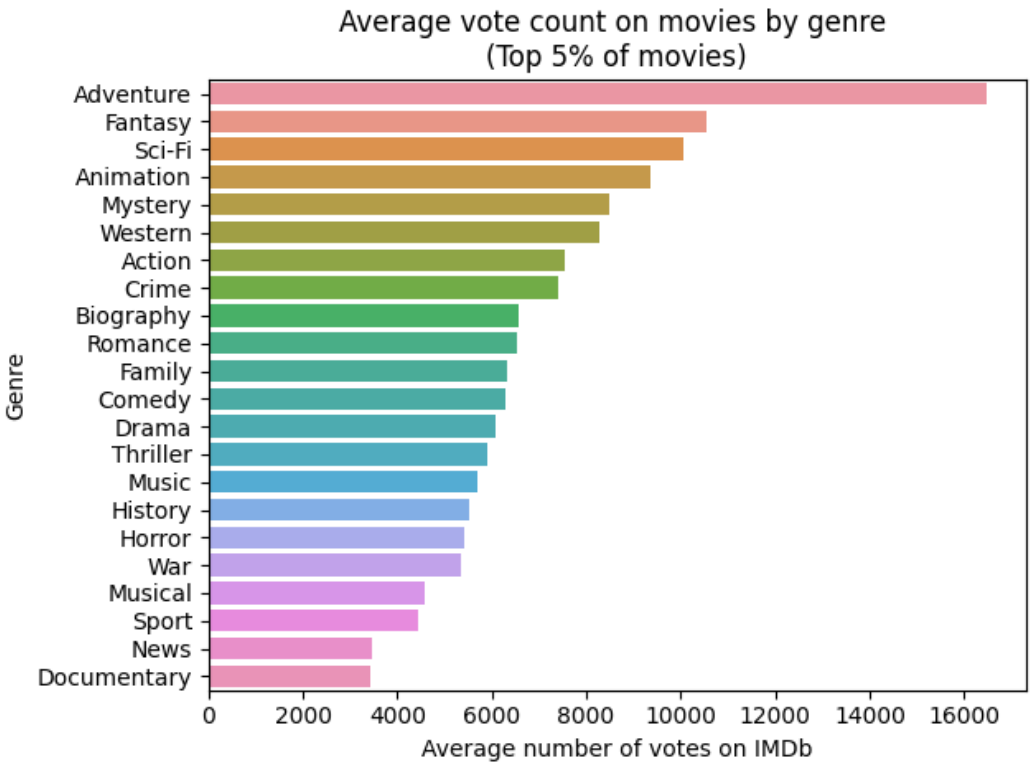
In [39]:

```
#Analyzing the most successful genres.

values = pivot_genres['numvotes']
labels = pivot_genres['genres']

genres_barplot = sns.barplot(
    x=values,
    y=labels,
    orient='h'
)
genres_barplot.set(
    xlabel='Average number of votes on IMDb',
    ylabel='Genre',
    title = 'Average vote count on movies by genre \n(Top 5% of movies)'
);

plt.savefig('./images/top_genres',dpi=150)
```



In [40]:

```
print('The top five genres in terms of average number of votes on IMDb are
for g in pivot_genres.iloc[:5]['genres']:
    print(g)
```

```
The top five genres in terms of average number of votes on IMDb are:
Adventure
Fantasy
Sci-Fi
Animation
Mystery
```

In [41]:

```
#Adding a column to clean_genres with a boolean value based on whether the

clean_genres['is_animation'] = ['Animation' in row for row in clean_genres
clean_genres.sample(7)
```

Out[41]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres
		The Haunting	The			

7822	tt1457765	in Connecticut 2: Ghosts of Georgia	Haunting in Connecticut 2: Ghosts of Georgia	2013	101.0	[Drama, Horror, Mystery]
5897	tt1210166	Moneyball	Moneyball	2011	133.0	[Biography, Drama, Sport]
		A Five Star				[Drama