Capstone Project 2: Project Proposal

The goal of this project is to build a movie recommendation system for users based on the MovieLens data set (download link).

The datasets describe ratings and free-text tagging activities from MovieLens, a movie recommendation service. It contains 20000263 ratings and 465564 tag applications across 27278 movies. These data were created by 138493 users between January 09, 1995 and March 31, 2015. Users were selected at random for inclusion. All selected users had rated at least 20 movies.

It’s worth investing effort to build Recommendation system for ecommerce companies like Amazon, or streaming service companies like Netflix or Spotify. Since it can potentially increase its profits as it filters out recommended items which users are interested in.

In this project. I will be focusing on using a collaborative-filtering system for returning users which already have the watching history. Due to the limit time, this project won’t dig into “cold start” part which is also very important to any recommendation system regarding to new items and new users. But I’ll put some of my thoughts at the end and this could be a further research work to continue this project.

Capstone Project 2: Data preprocessing

1. Load original CSV files.
2. Split title and year into separate columns. Convert year to datetime.
3. Categorize genres properly: split strings into boolean columns per genre.
4. Modify the rating timestamp: from universal seconds to datetime year.
5. Check for NaN values.

Load original csv files:

```python
df_ratings.head(3)
```

	userId	movieId	rating	timestamp
0	1	2	3.5	2005-04-02 23:53:47
1	1	29	3.5	2005-04-02 23:31:16
2	1	32	3.5	2005-04-02 23:33:39

```python
df_movies.head(3)
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance

```python
df_tags.head(3)
```

	userId	movieId	tag	timestamp
0	18	4141	Mark Waters	1240597180
1	65	208	dark hero	1368150078
2	65	353	dark hero	1368150079

After preprocessing:

```
df_movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27278 entries, 0 to 27277
Data columns (total 24 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   movieId             27278 non-null  int32  
 1   title               27278 non-null  object 
 2   year                27256 non-null  float64
 3   title2              27278 non-null  object 
 4   (no genres listed)  27278 non-null  bool   
 5   Action              27278 non-null  bool   
 6   Adventure           27278 non-null  bool   
 7   Animation           27278 non-null  bool   
 8   Children            27278 non-null  bool   
 9   Comedy              27278 non-null  bool   
 10  Crime               27278 non-null  bool   
 11  Documentary         27278 non-null  bool   
 12  Drama               27278 non-null  bool   
 13  Fantasy             27278 non-null  bool   
 14  Film-Noir           27278 non-null  bool   
 15  Horror              27278 non-null  bool   
 16  IMAX                27278 non-null  bool   
 17  Musical             27278 non-null  bool   
 18  Mystery             27278 non-null  bool   
 19  Romance             27278 non-null  bool   
 20  Sci-Fi              27278 non-null  bool   
 21  Thriller            27278 non-null  bool   
 22  War                 27278 non-null  bool   
 23  Western             27278 non-null  bool   
dtypes: bool(20), float64(1), int32(1), object(2)
memory usage: 1.2+ MB
```

```
df_ratings.info()
```

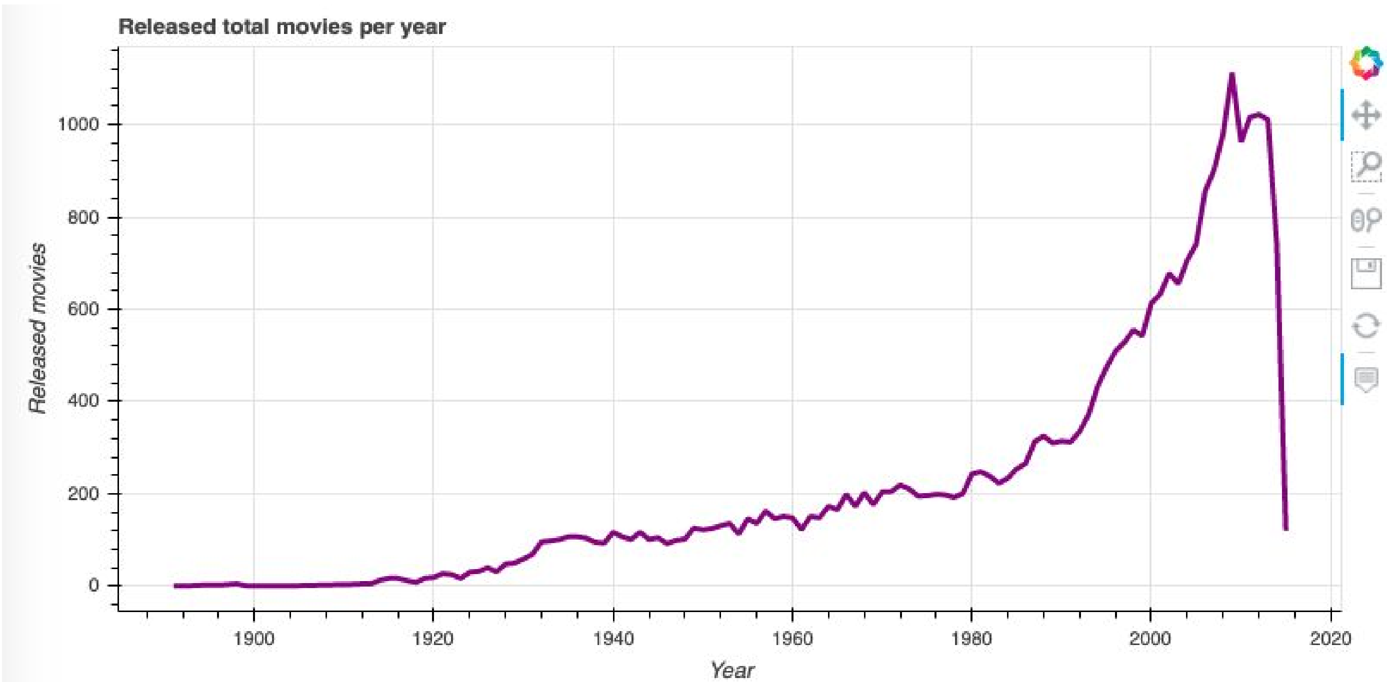
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20000263 entries, 0 to 20000262
Data columns (total 7 columns):
 #   Column        Dtype  
---  ------        -----  
 0   userId        int32  
 1   movieId       int32  
 2   rating        float32
 3   timestamp     object 
 4   title         object 
 5   rating_year   int64  
 6   rating_month  int64  
dtypes: float32(1), int32(2), int64(2), object(2)
memory usage: 991.8+ MB
```

```
df_tags.info()
```

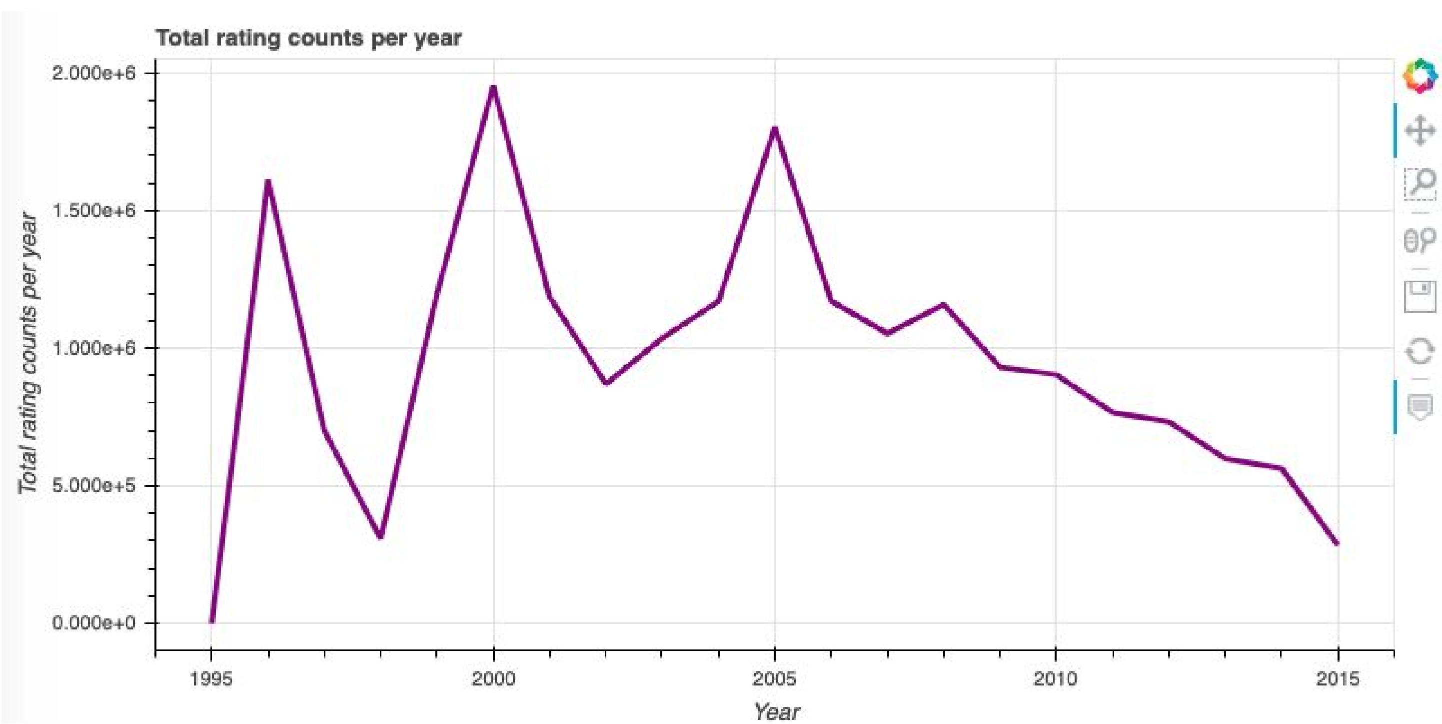
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 465564 entries, 0 to 465563
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype 
---  ------     --------------   ----- 
 0   userId     465564 non-null  int64 
 1   movieId    465564 non-null  int64 
 2   tag        465548 non-null  object
 3   timestamp  465564 non-null  int64 
dtypes: int64(3), object(1)
memory usage: 14.2+ MB
```

Exploratory Data Analysis

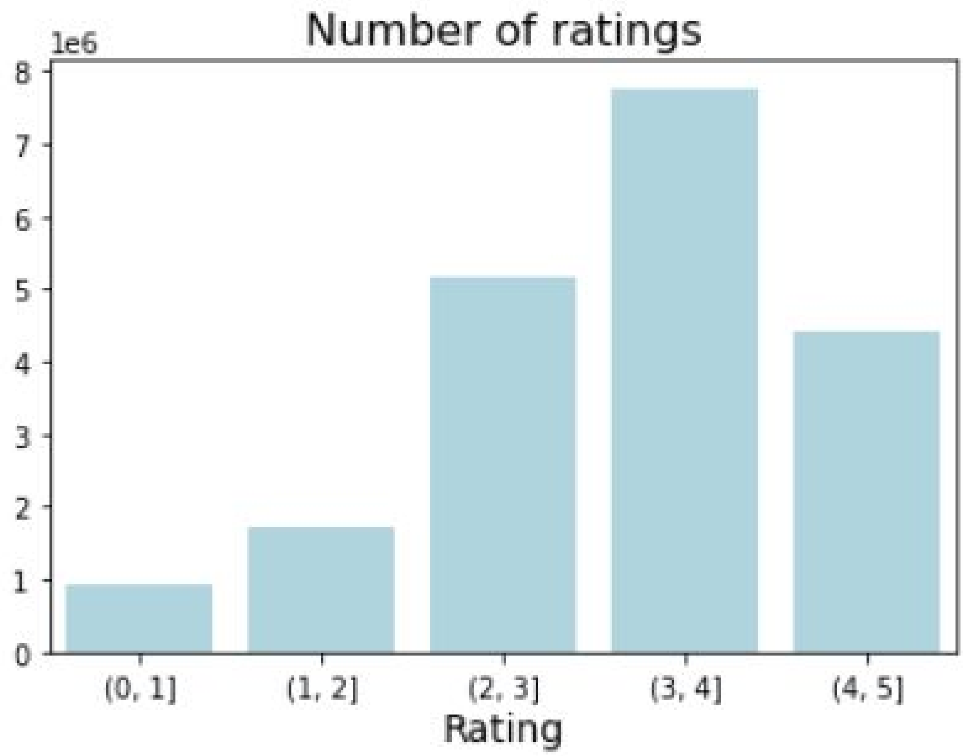
Yearly trending of released movies



Yearly trending of total ratings



Bar chart of rating buckets



Percentage table:

```
# pecentage
df_ratings['rating_bins'].value_counts().sort_index(ascending=False)/len(df_ratings)
```

```
(4, 5]    0.221671
(3, 4]    0.388099
(2, 3]    0.258726
(1, 2]    0.085511
(0, 1]    0.045992
Name: rating_bins, dtype: float64
```

This is good news that the majority of the ratings fall into range 3+ ~ 5. This means most of the audience love the movies on the website.

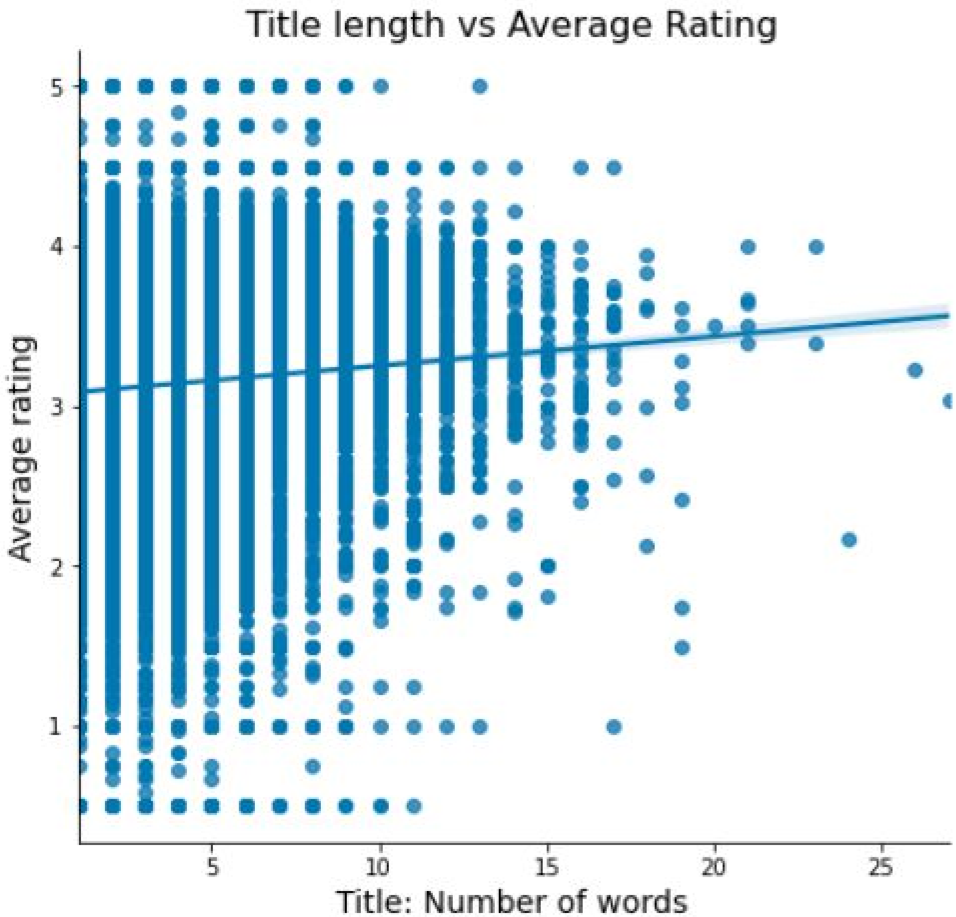
What are the top 10 movies with the highest average rating

title	avg_rating
Prom Queen: The Marc Hall Story (2004)	5.0
The Garden of Sinners - Chapter 5: Paradox Par...	5.0
Death of a Nation - The Timor Conspiracy (1994)	5.0
Poison (1951)	5.0
Sun Kissed (2012)	5.0
Giorgino (1994)	5.0
Schmatta: Rags to Riches to Rags (2009)	5.0
De la servitude moderne (2009)	5.0
The Encounter (2010)	5.0
Best of Ernie and Bert, The (1988)	5.0

What are the most popular top 10 movies with highest mean ratings?

title	total_cnt
Pulp Fiction (1994)	67310
Forrest Gump (1994)	66172
Shawshank Redemption, The (1994)	63366
Silence of the Lambs, The (1991)	63299
Jurassic Park (1993)	59715
Star Wars: Episode IV - A New Hope (1977)	54502
Braveheart (1995)	53769
Terminator 2: Judgment Day (1991)	52244
Matrix, The (1999)	51334
Schindler's List (1993)	50054

Does the length of the title influence the average rating?



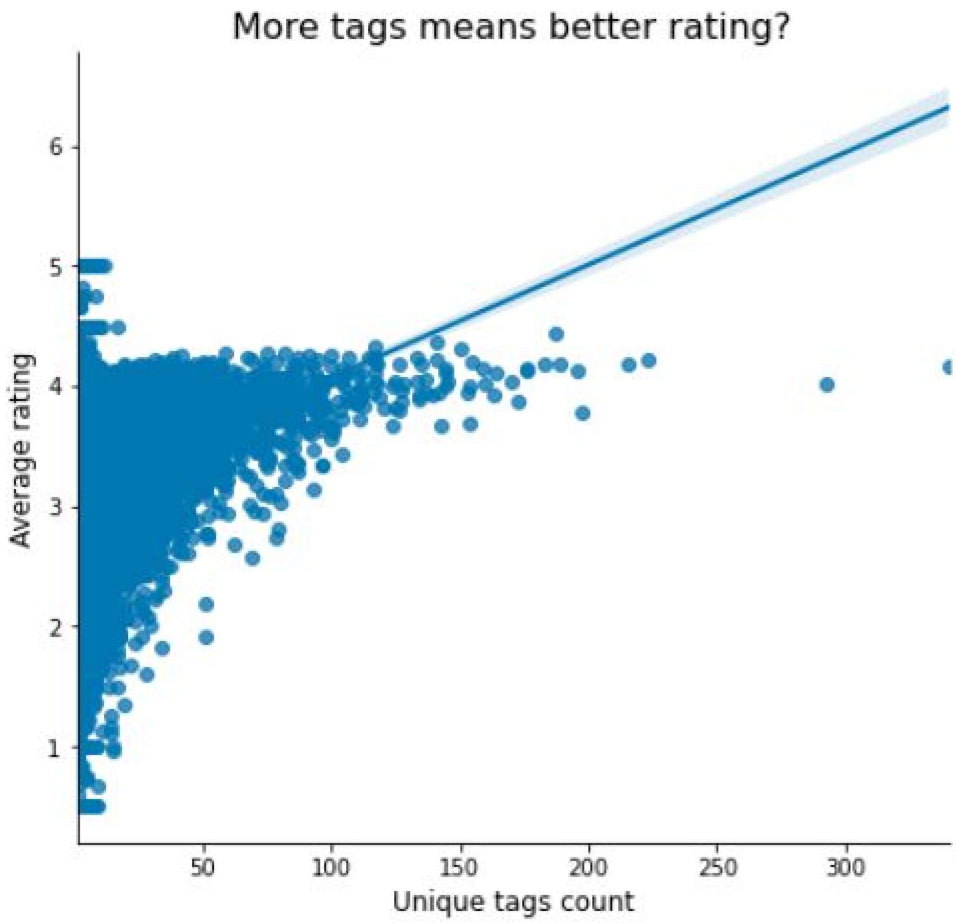
Correlation coefficient: 0.066
We got a very small positive correlation coefficient which means **a**s the number of words in the title increases, so does the average rating.

How does the average rating of a movie relate to the number of ratings it has received?

Correlation coefficient: 0.066

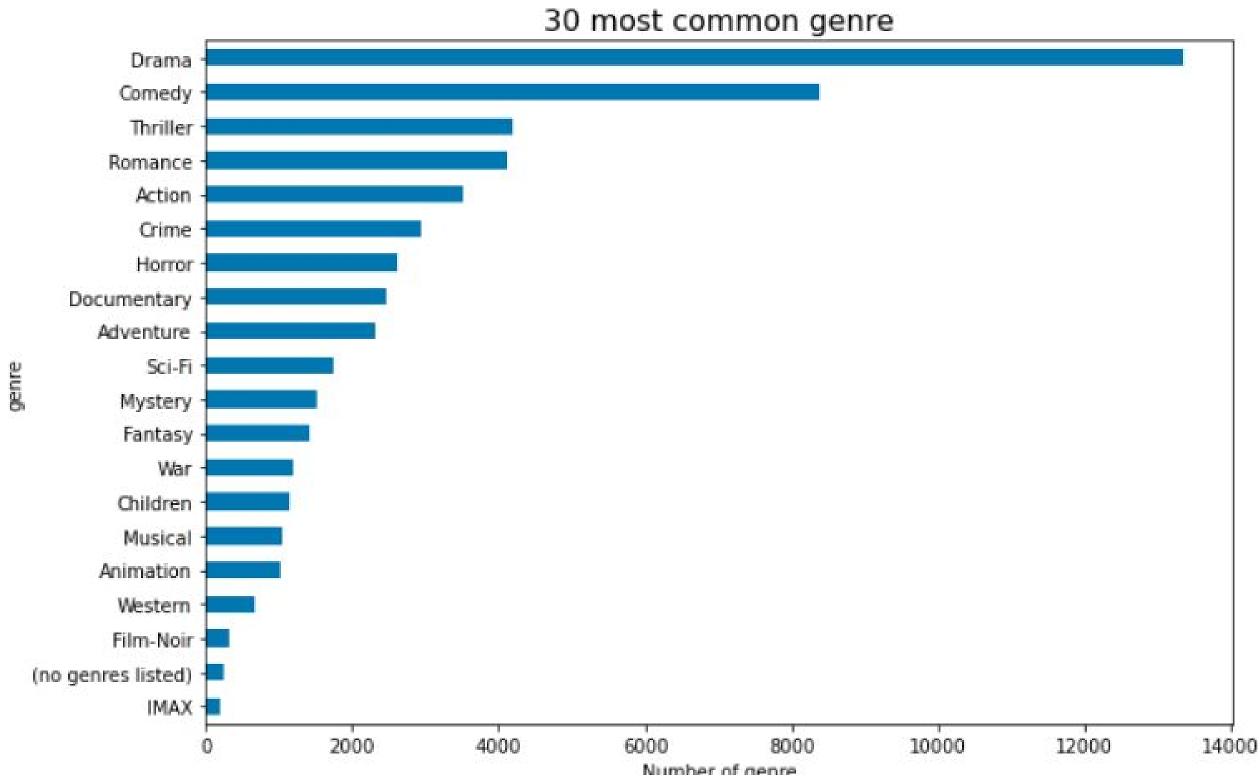
We got a very small positive correlation coefficient which means as the number of ratings received increases, so does the average rating.

How does more unique tags per movie mean the movie having better ratings?

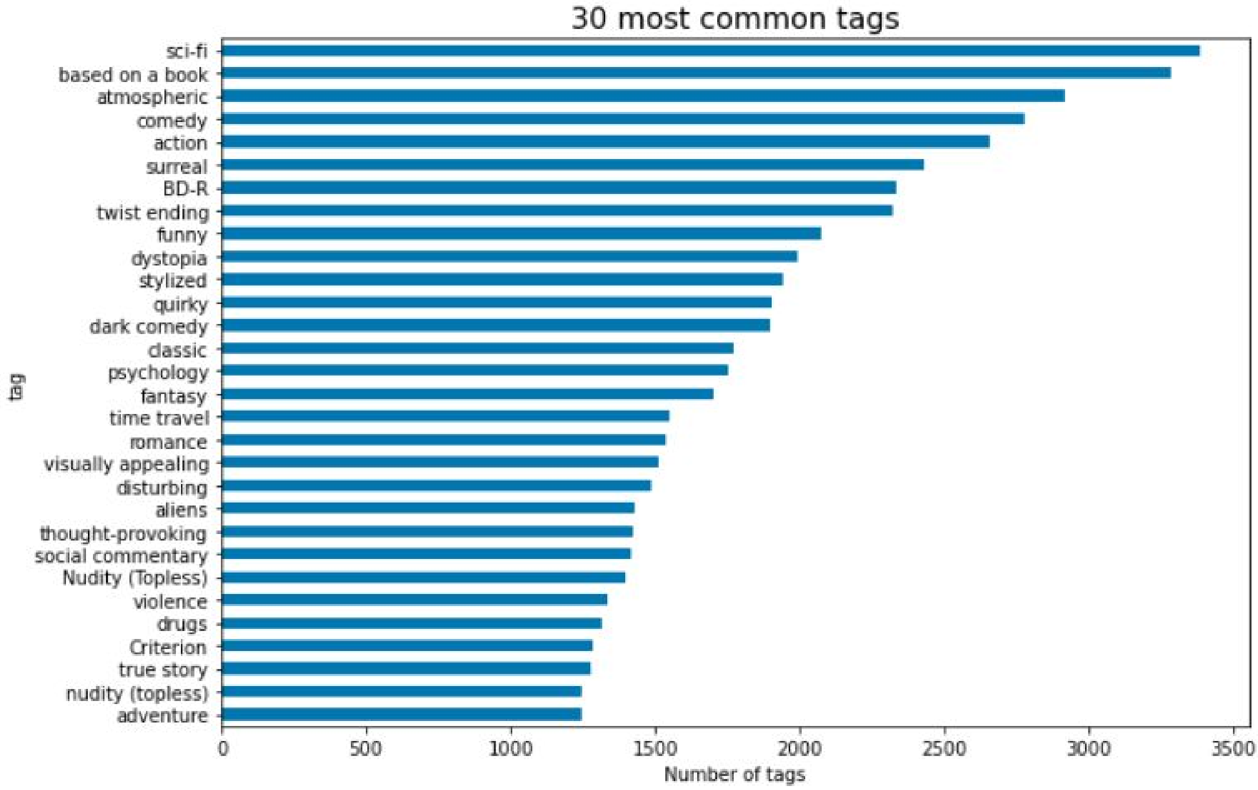


Correlation coefficient: 0.26637361301113666

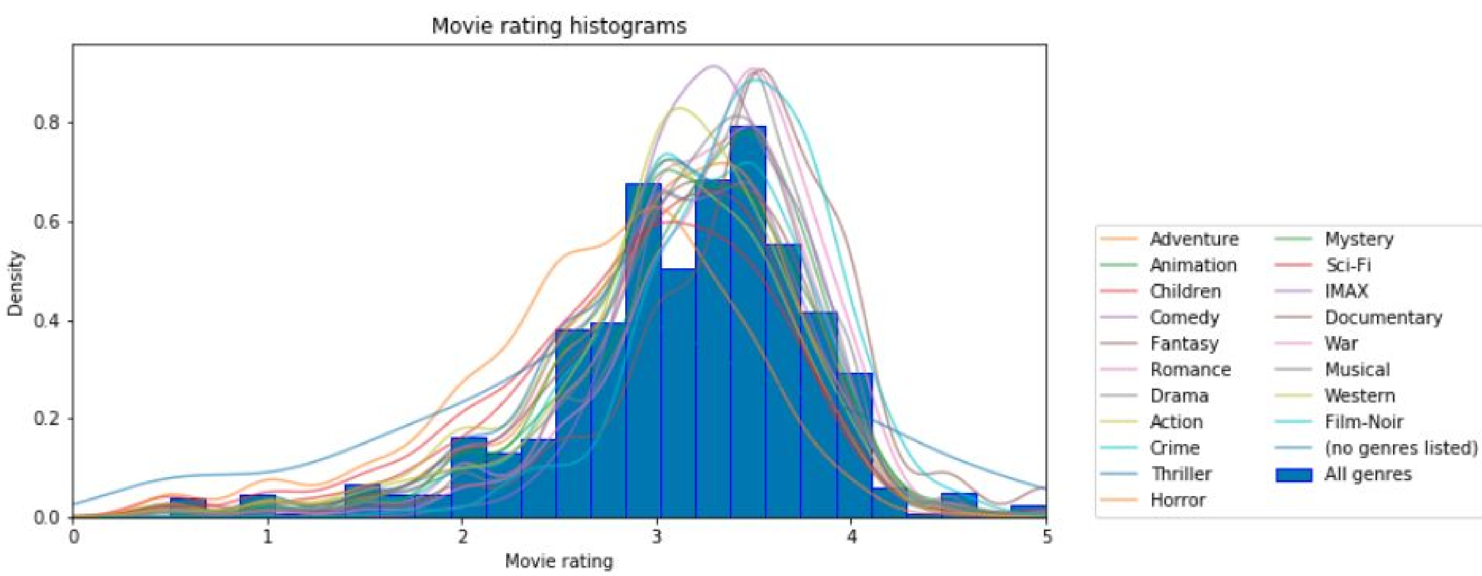
What are the top 30 most common genres?



What are the top 30 most common tags?



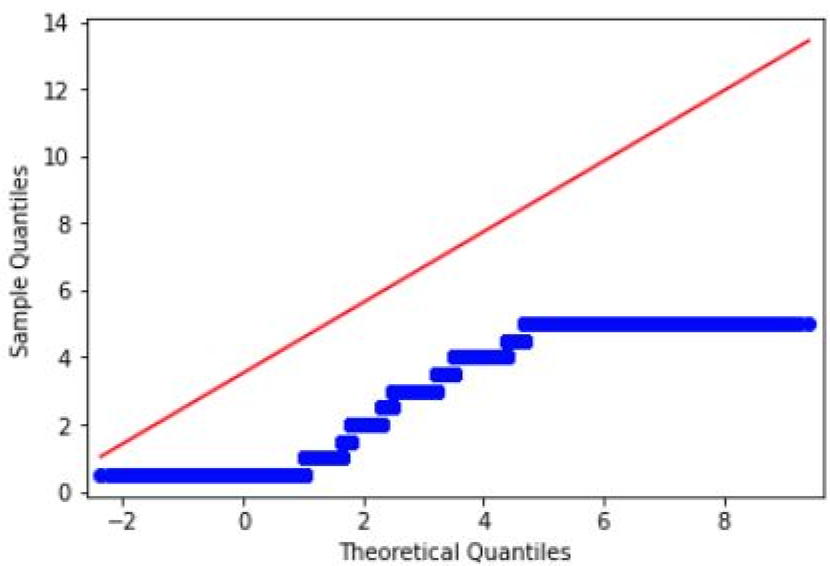
Rating distribution by genres



The plot shows us a left-skewed distribution for all genres and each genre follows the left-skewed shape as well.

Check Normality

```
import pylab
import statsmodels.api as sm
mu=np.mean(df_ratings['rating'].dropna())
sigma=np.var(df_ratings['rating'].dropna())
sm.qqplot(df_ratings['rating'], loc = mu, scale = sigma, line='s')
pylab.show()
```




```
from scipy.stats import kurtosis
from scipy.stats import skew
print("mean : ", np.mean(df_ratings['rating'].dropna()))
print("var  : ", np.var(df_ratings['rating'].dropna()))
print("skew : ",skew(df_ratings['rating'].dropna()))
print("excess kurtosis : ",kurtosis(df_ratings['rating'].dropna()))
```

```
mean :  3.5255287
var  :  1.1066805
skew :  -0.6553115248680115
excess kurtosis :  0.13746752211657665
```

1. we got negative skewness which means the mass of th distribution is concentrated on the right which fits the plog above

2. Kurtosis is a measure of the thickness of the tails of a distribution. Excess kurtosis = kurtosis - 3. Excess kurtosis is not ZERO telling us that the data is not normally distributed.

We are able to tell that the data is not normal by plot QQ plot and calculate skewness and excess kurtosis. However, let's still run a normality test to double check our conclusion.

By conducting Kolmogorov–Smirnov test with alpha = 0.05

```
stats.kstest(df_ratings['rating'].dropna(),'norm', args=(mu, sigma))
```

```
KstestResult(statistic=0.16570544657699904, pvalue=0.0)
```

We rejected the null hypothesis due to p value < .05