Team members:
Sirisha Lanka - PES1201700294
Malavikka R - PES1201700794
Kevin Arulraj - PES1201700659

# Assignment 7

*Problem Statement:*
Take a dataset with identifiers and use collaborative filtering to filter through the noise in the dataset and produce graphs for the same

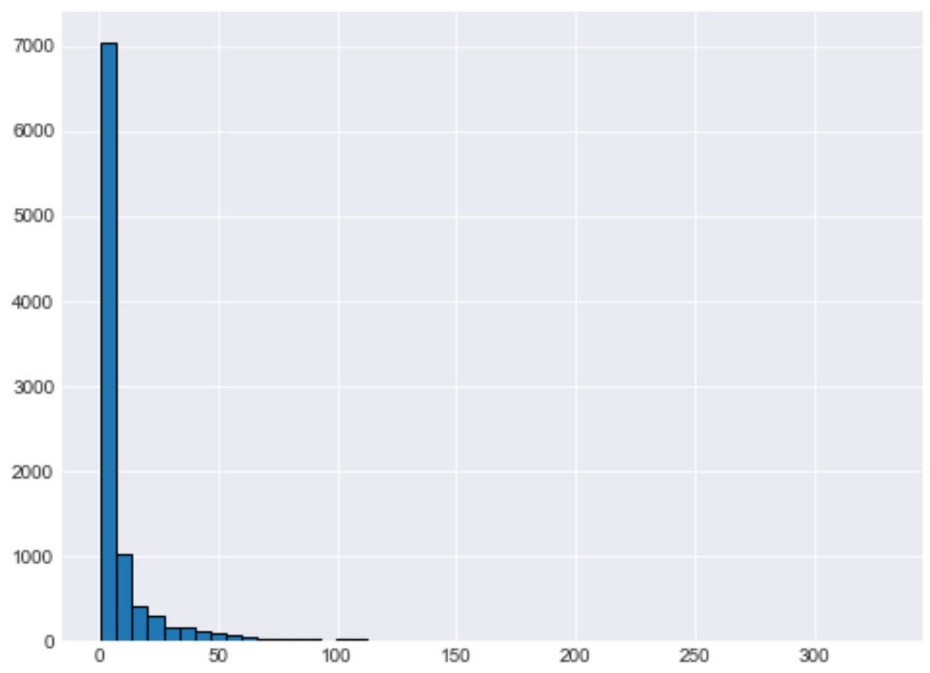
*Dataset:*
links.csv - includes the identifier for three different movie rating websites (movieID, imdbId, tmdbId)
movies.csv - movie names and their genres
ratings.csv - contains userID, movieID, its ratings and when it was rated
tags.csv - contains userID, movieID, a tag by which it is identified, timestamped

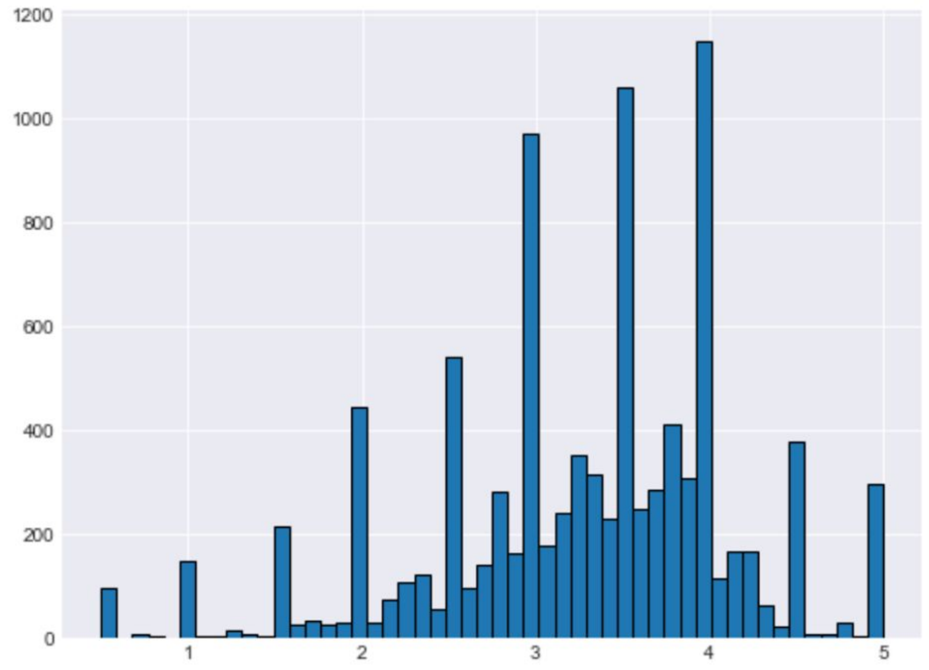
*Code:*

```
import numpy as np
import pandas as pd

ratings_data = pd.read_csv("ratings.csv")
ratings_data.head()

movie_names = pd.read_csv("movies.csv")
movie_names.head()

movie_data = pd.merge(ratings_data, movie_names, on='movieId')
movie_data.head()

movie_data.groupby('title')['rating'].mean().head()
movie_data.groupby('title')['rating'].mean().sort_values(ascending=False).head()

movie_data.groupby('title')['rating'].count().sort_values(ascending=False).head()
ratings_mean_count = pd.DataFrame(movie_data.groupby('title')['rating'].mean())
ratings_mean_count['rating_counts'] = pd.DataFrame(movie_data.groupby('title')['rating'].count())
ratings_mean_count.head()

import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('dark')
%matplotlib inline

plt.figure(figsize=(8,6))
plt.rcParams['patch.force_edgecolor'] = True
ratings_mean_count['rating_counts'].hist(bins=50)

plt.figure(figsize=(8,6))
plt.rcParams['patch.force_edgecolor'] = True
ratings_mean_count['rating'].hist(bins=50)

plt.figure(figsize=(8,6))
plt.rcParams['patch.force_edgecolor'] = True
sns.jointplot(x='rating', y='rating_counts', data=ratings_mean_count, alpha=0.4)

user_movie_rating = movie_data.pivot_table(index='userId', columns='title', values='rating')
user_movie_rating.head()

forrest_gump_ratings = user_movie_rating['Forrest Gump (1994)']
forrest_gump_ratings.head()

movies_like_forest_gump = user_movie_rating.corrwith(forrest_gump_ratings)

corr_forrest_gump = pd.DataFrame(movies_like_forest_gump, columns=['Correlation'])
corr_forrest_gump.dropna(inplace=True)
corr_forrest_gump.head()

corr_forrest_gump.sort_values('Correlation', ascending=False).head(10)

corr_forrest_gump = corr_forrest_gump.join(ratings_mean_count['rating_counts'])
corr_forrest_gump.head()

corr_forrest_gump[corr_forrest_gump ['rating_counts']>50].sort_values('Correlation', ascending=False).head
()
```

*Results:*

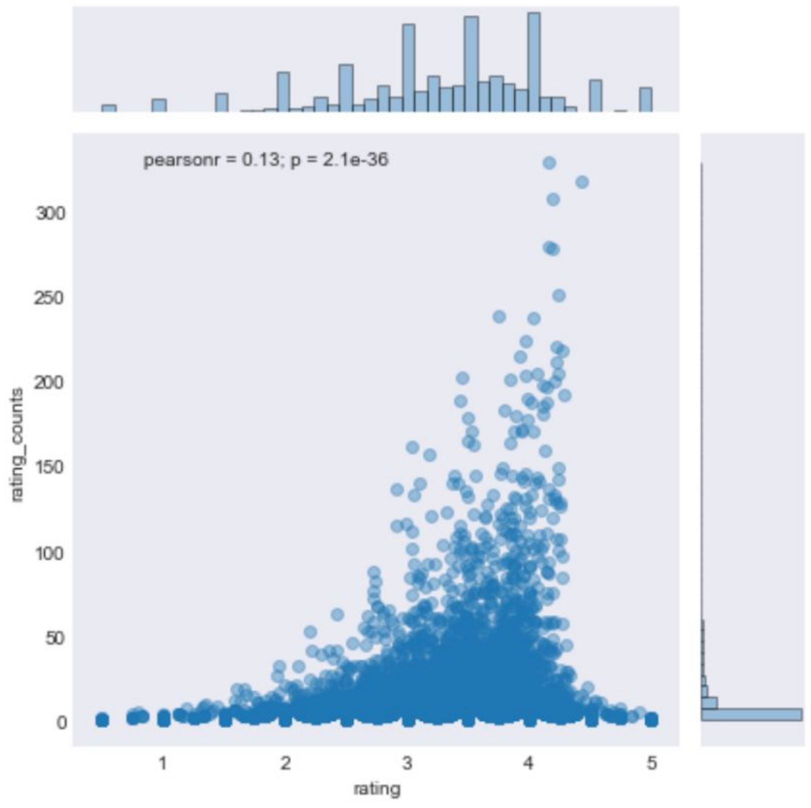
| | Correlation | rating_counts |
|---|---|---|
| **title** | | |
| **Forrest Gump (1994)** | 1.000000 | 329 |
| **Mr. Holland's Opus (1995)** | 0.652144 | 80 |
| **Pocahontas (1995)** | 0.550118 | 68 |
| **Grumpier Old Men (1995)** | 0.534682 | 52 |
| **Caddyshack (1980)** | 0.520328 | 52 |





1200
1000
800
600
400
200
0
1
2
3
4
5

```
<Figure size 576x432 with 0 Axes>
```



pearsonr = 0.13; p = 2.1e-36
rating_counts
300
250
200
150
100
50
0
1
2
3
4
5
rating