Overview

Nowadays, we need recommender systems almost everywhere in our lives. Therefore, retailers are become more interested in recommender systems to analyze patterns of user interest in products and provide personalized recommendations. The first goal of this project is understanding, analyzing, and correlating the trend in average rating movies of different genres. The second goal is building recommender engines to provide recommendations to different users and build different machine learning models to predict the rating of each movie.

Business Objective

The recommender system is useful to any business that makes money via recommendations. Since we want to work with movie dataset, the client of this project could be Amazon, Netflix, Hulu, HBO, etc.

Giving good recommendations will help users spend less time searching for their type of movie and having a recommender platform. This will help the customer to continue with the service and having a good experience.

Data

I will use two datasets for this project as:

1. The first dataset is collected from a free, noncommercial movie recommender as MoviLens.org. (https://grouplens.org/datasets/movielens). This recommender is like Netflix, minus the ability to watch movies. The dataset which was released in April 2015 contains 20 million ratings and 465,000 tag applications applied to 27,000 movies by 138,000 users (movies.csv, ratings.csv, link.csv). The data has been created in October 2016. Users were selected at random for inclusion. All selected users had rated at least 20 movies.
2. The second dataset is collected from (https://www.kaggle.com/karrrimba/movie-metadatacsv). We have more features in this dataset like information about revenue, budget, release date, popularity,... .

Solution

I. How to get the insight about how people's rating change for different genres over the years?

Getting different distributions of the movie-ratings and movie-relevancy across title, genre, year, and tag of each movie can give the insight about how people's taste vary over the years. Different inferential statistics and visualization techniques can be implemented to deliver a better understanding of the trends.

II. How to recommend movies?

In this work different Collaborative filtering (CF) and Content-based filtering techniques will be used to analyze relationships between users and interdependencies among products to identify new user-item associations. Most CF algorithms are based on a user-item rating matrix where each row represents a user, each column an item. The entries of this matrix are ratings given by users to items. One of the primary areas of collaborative filtering is the latent factor models which are based on matrix factorization (Singular value decomposition (SVD)).

III. How to predict movie ratings?

In this work different machine learning techniques like Random Forest, Support Vector Machine, K-Nearest Neighbors,... will be implemented to predict the rating of the test set.

Workflow

- Collecting data and applying data wrangling methods
- Starting exploratory data analysis to find trends and storytelling
- Conduct further data analysis to identify relationships between different variables
- Perform in-depth analysis using collaborative filtering and machine learning techniques to recommend and predict
- Conclusion and Future works

First Dataset (MovieLens)

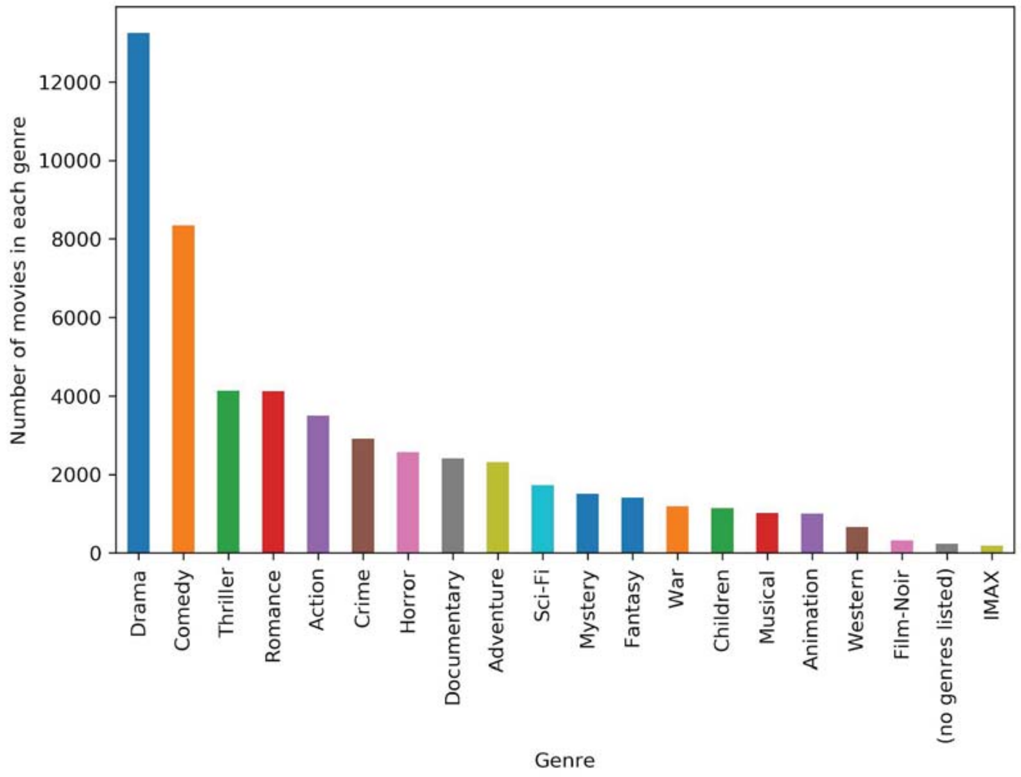
Data Wrangling

This dataset was clean. The ‘rating.csv’ has 20,000,263 rows and 4 columns. The ‘movie.csv’ dataset has 27,278 rows and 3 columns. I just removed the year from the titles and made ‘year’ and ‘decade’ as separate columns. I also prepared the data for EDA by separating different genres and merge it with the rating dataset for better EDA.

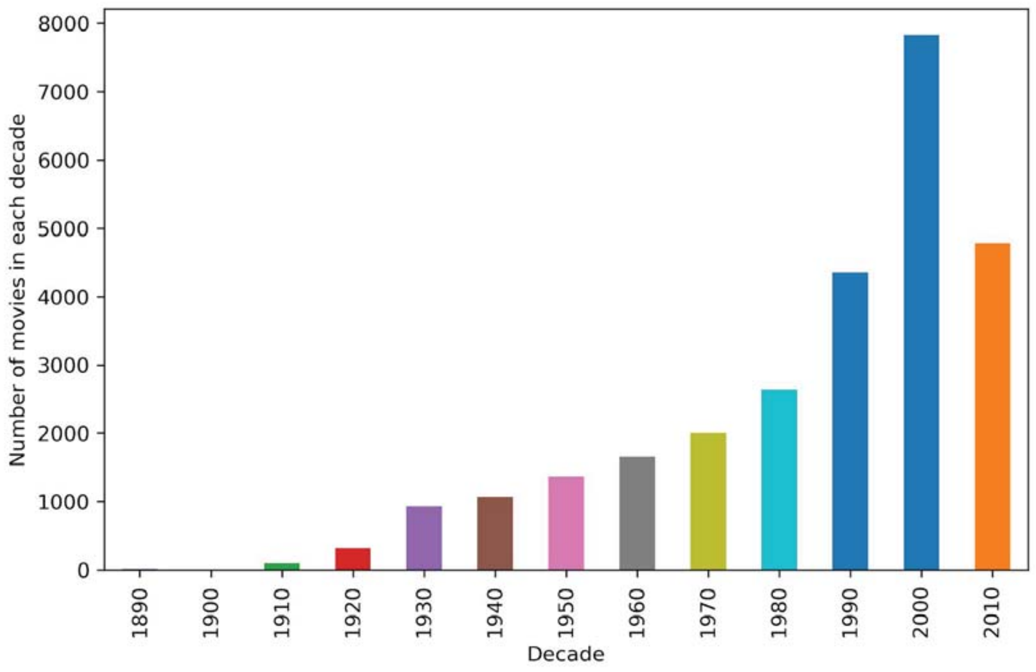
Exploratory Data Analysis (EDA) and Storytelling

In this section, the various insights produced through descriptive statistics and data visualization is presented.

Genres: **Drama, Comedy**, and **Thriller** have the most production.

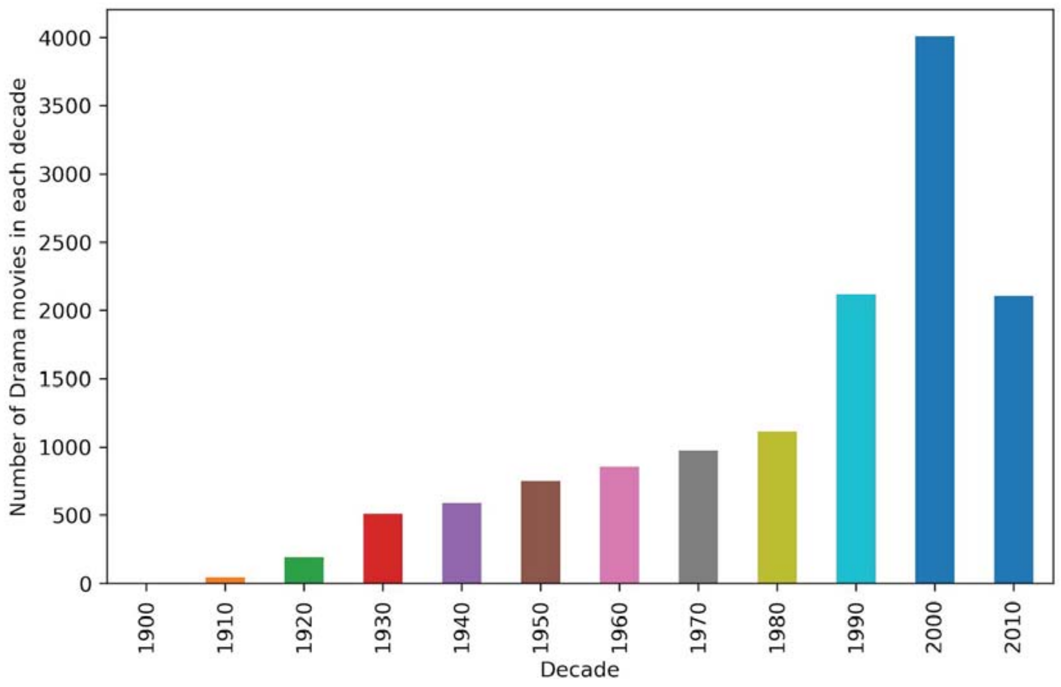


Total number of movies in each decade is presented below. Most production was in the decade **2000.**

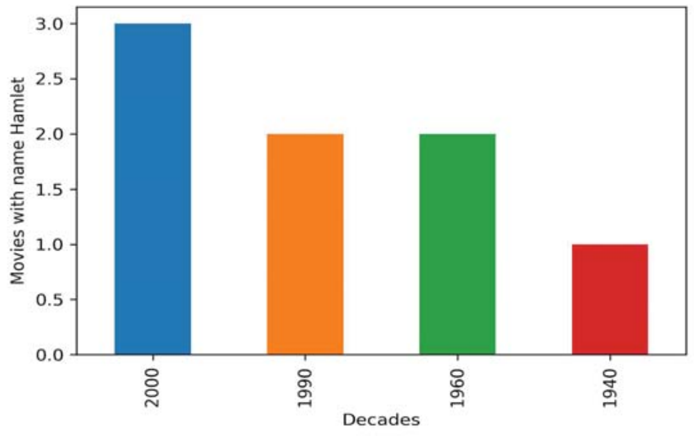


Most **drama** movies has been produced in years **2009, 2007,** and **2008** respectively.

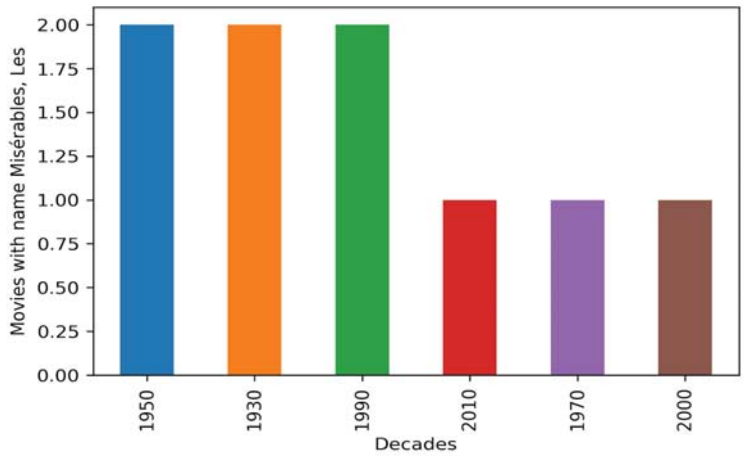
Most **drama** movies produced in the decade **2000**.



8 movies have been produced with the title **'Hamlet'**, The following barplot shows the number of 'Hamlet' movies in each decade.



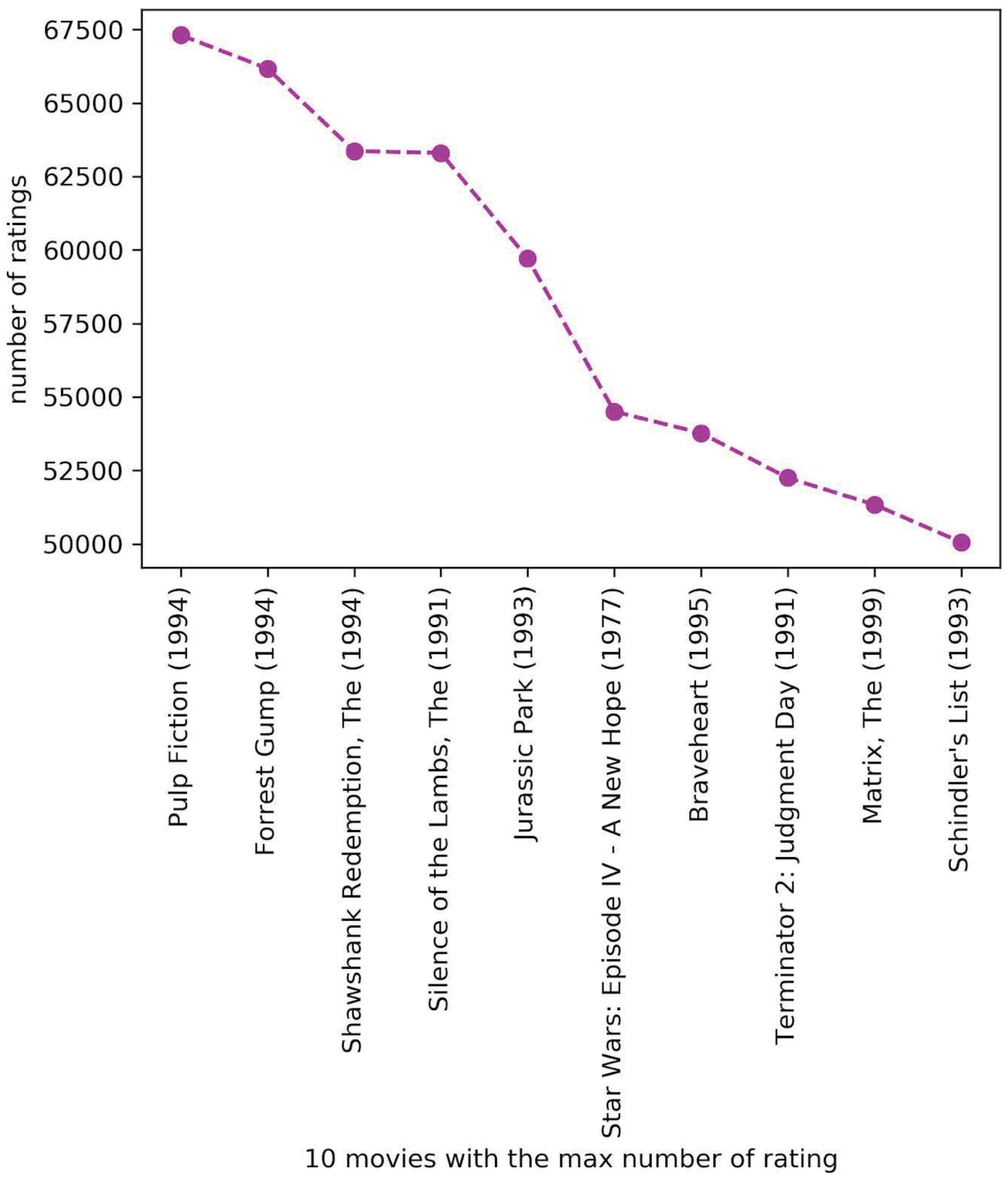
9 movies have been produced with title **'Misérables, Les'.** The following barplot shows the number of 'Misérables, Les' movies in each decade.



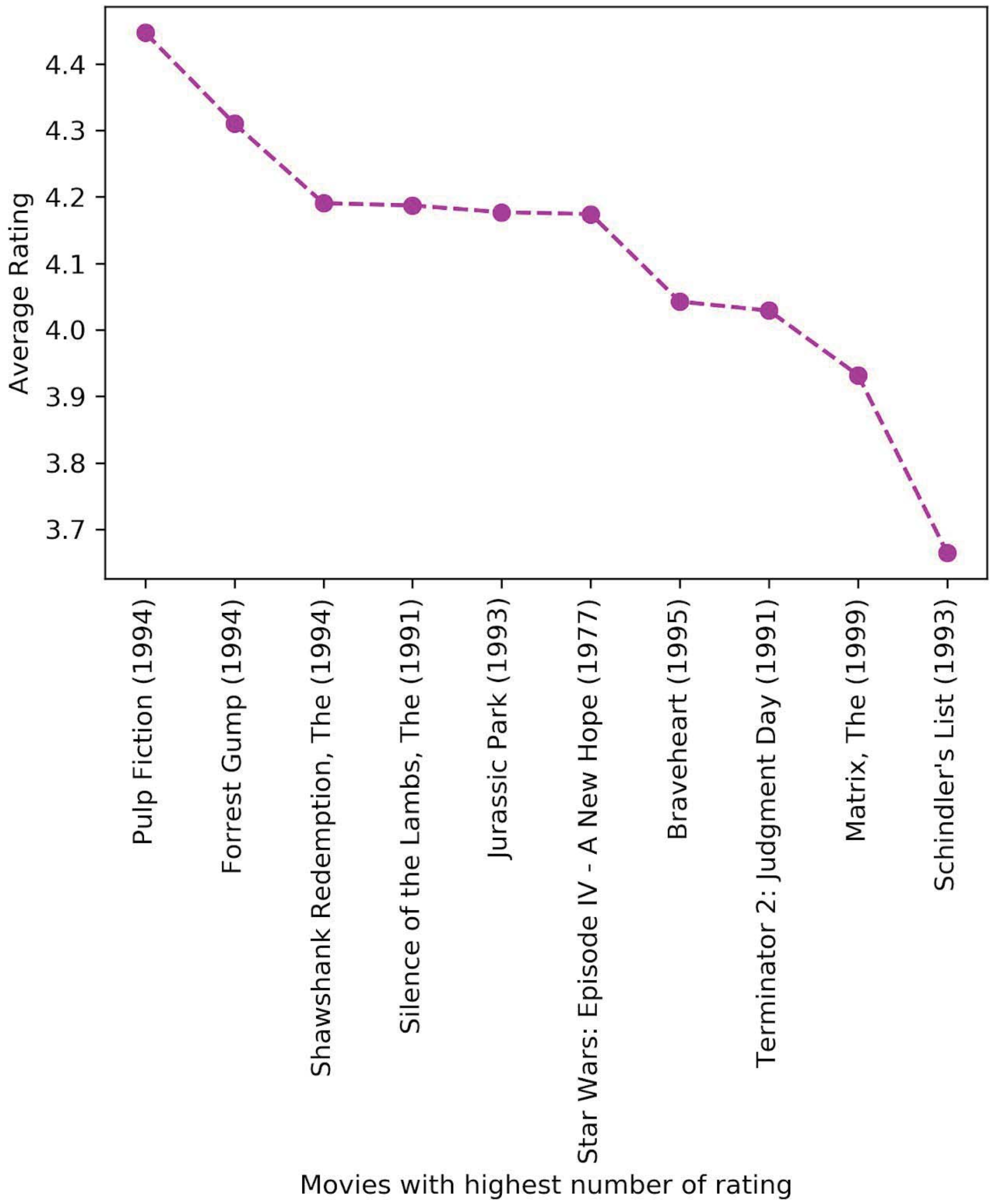
Most favorable movies are:

- As it was expected, **Shawshank Redemption, Pulp Fiction,** and **Silence of the Lambs** are the first three with most ranking equal to 5 respectively.
- **Shawshank Redemption, Matrix**, and **Pulp Fiction** are the first three with most ranking equal to 4.5 respectively.
- The first three movies with most ranks equal to 4 are **Silence of the Lambs, Fugitive**, and **Jurassic**.
- The average rating for the most favorable movie **Shawshank Redemption** is: 4.447
- The average rating for the second favorable movie **Pulp Fiction** is: 4.174
- The movies with the max number of ratings are **Pulp Fiction, Forrest Gump, Shawshank redemption** respectively.

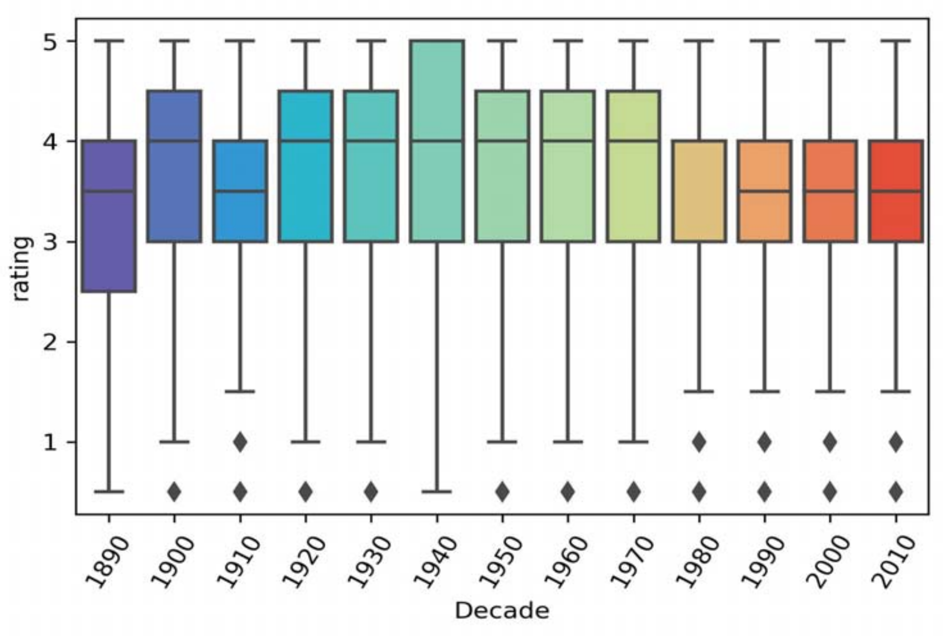
- The graph for the ten movies with the maximum number of the rating is presented below.



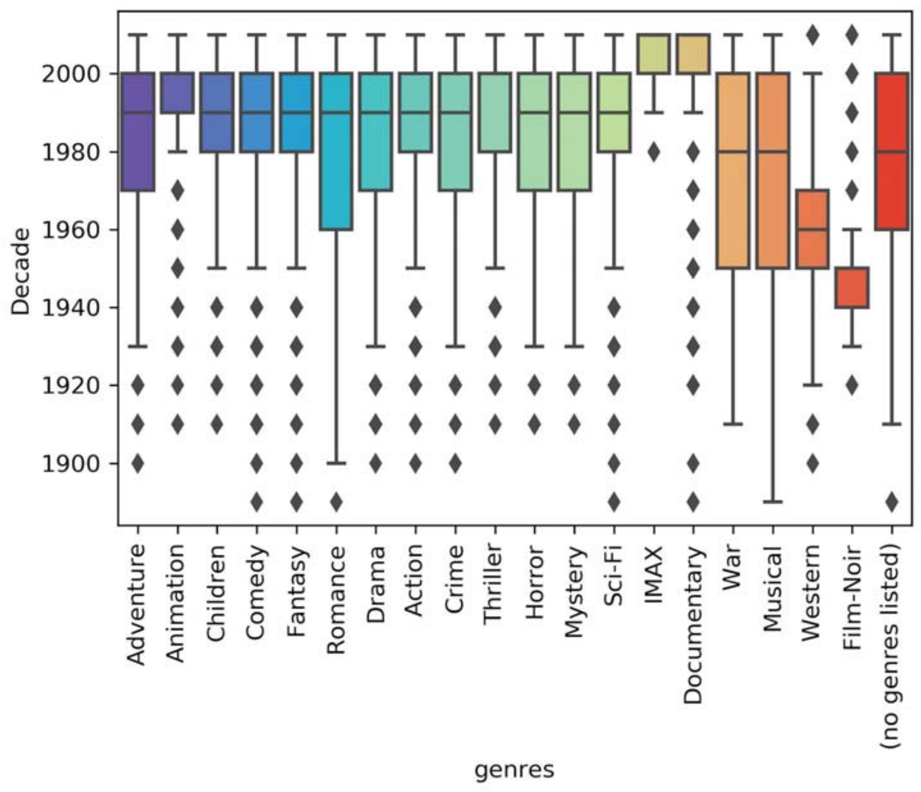
- The graph of the movies with the largest average rating is presented below.



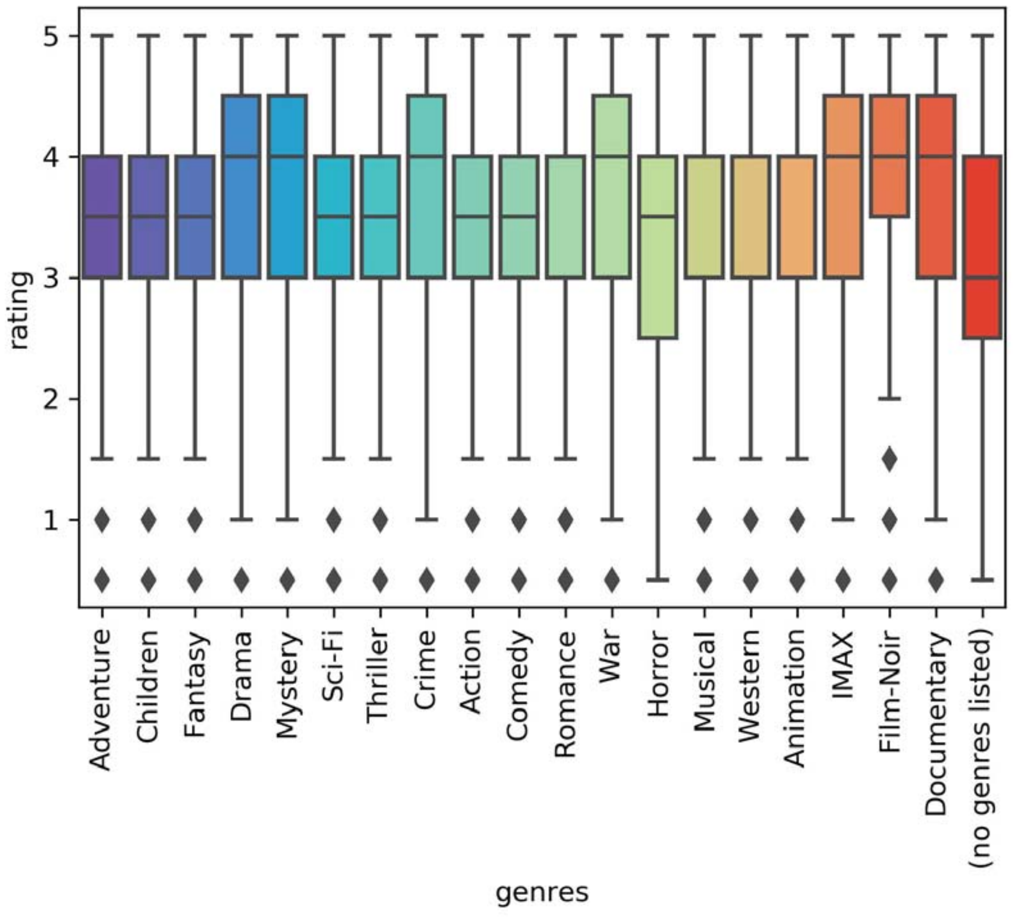
- The Boxplot of the **'Decade'** vs **'Rating'** is illustrated below. The movies of the **40th** decade have the highest rating.



- Boxplot of the **'Decade'** vs **'Genres'** is presented below:

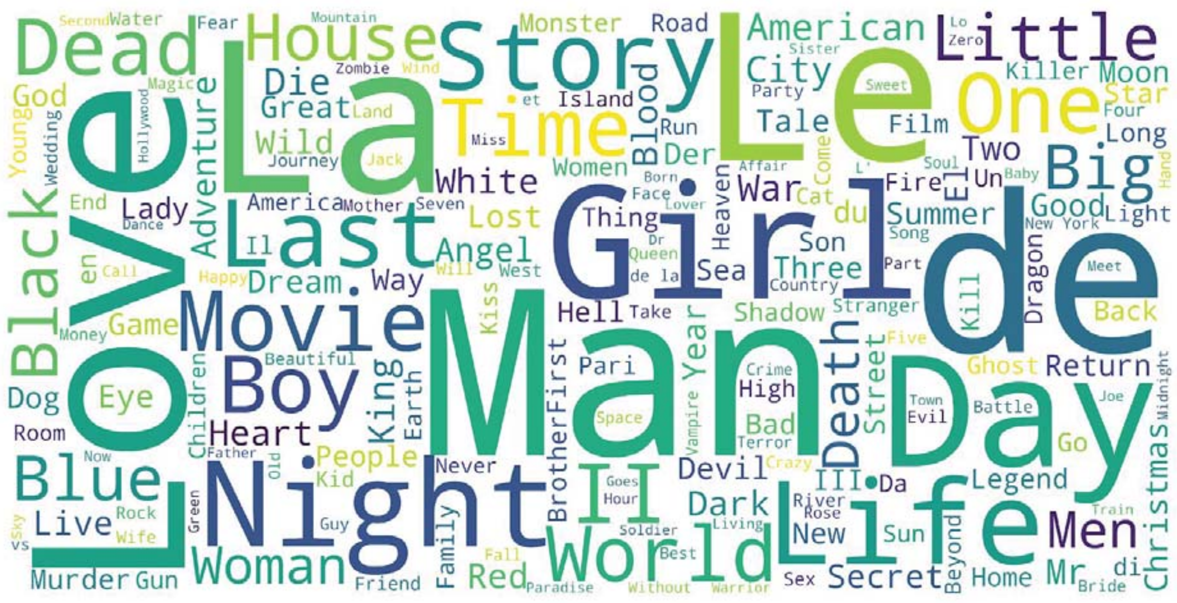


- Boxplot of the **'Genre'** vs **'Rating'**
  Drama, Mystery, Crime, War, Imax, documentary have the highest ratings.
  Horror is the least favorite.



Title Word Clouds

There are certain words that use more often in titles. I use the WordCloud library to find out what are these words. The word **Love** is the most commonly used word in movie titles. **Girl, Day** and **Man** are also among the most commonly occurring words.



Recommender Systems

Before applying different recommender engines, I filtered the movies with more than 100 vote counts. Because we don't want movies with small number of votes, affect our recommender. Also, I use the sklearn library to set the train and test data.

1. Simple Recommender

The Simple Recommender offers generalized recommendations to every user based on movie popularity, ratings, and genre. The basic idea behind this recommender is that movies that are more popular and more critically acclaimed will have a higher probability of being liked by the average audience. This model does not give personalized recommendations based on the user.

So I sort the movies based on their genre, and I recommend the ones with the highest number of votes and the higher average rankings to the new audience.

The list of the **10 best Drama** movies to recommend are:

1 : Pulp Fiction (1994)
2 : Shawshank Redemption, The (1994)
3 : Schindler's List (1993)
4 : American Beauty (1999)
5 : Fargo (1996)
6 : Godfather, The (1972)
7 : Fight Club (1999)
8 : Lord of the Rings: The Return of the King, The (2003)
9 : One Flew Over the Cuckoo's Nest (1975)
10 : Godfather: Part II, The (1974)

The list of the **10 best Romance** movies to recommend are:

1 : Forrest Gump (1994)
2 : Beauty and the Beast (1991)
3 : Princess Bride, The (1987)
4 : Groundhog Day (1993)
5 : Shrek (2001)
6 : Sleepless in Seattle (1993)
7 : Good Will Hunting (1997)
8 : Four Weddings and a Funeral (1994)
9 : Crouching Tiger, Hidden Dragon (Wo hu cang long) (2000)
10 : There's Something About Mary (1998)

The list of the **10 best Action** movies to recommend are:

1 : Star Wars: Episode IV - A New Hope (1977)
2 : Braveheart (1995)
3 : Matrix, The (1999)
4 : Star Wars: Episode VI - Return of the Jedi (1983)
5 : Star Wars: Episode V - The Empire Strikes Back (1980)
6 : Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)
7 : Fight Club (1999)
8 : Saving Private Ryan (1998)
9 : Princess Bride, The (1987)
10 : Lord of the Rings: The Return of the King, The (2003)

2. IMDB Weighted Rating Formula

Another technique could be IMDB's **weighted rating** formula which is mathematically represented as follows:

$$WR = \frac{v}{v+m}R + \frac{m}{v+m}C$$

where,

v is the number of votes for the movie

m is the minimum votes required to be listed in the chart

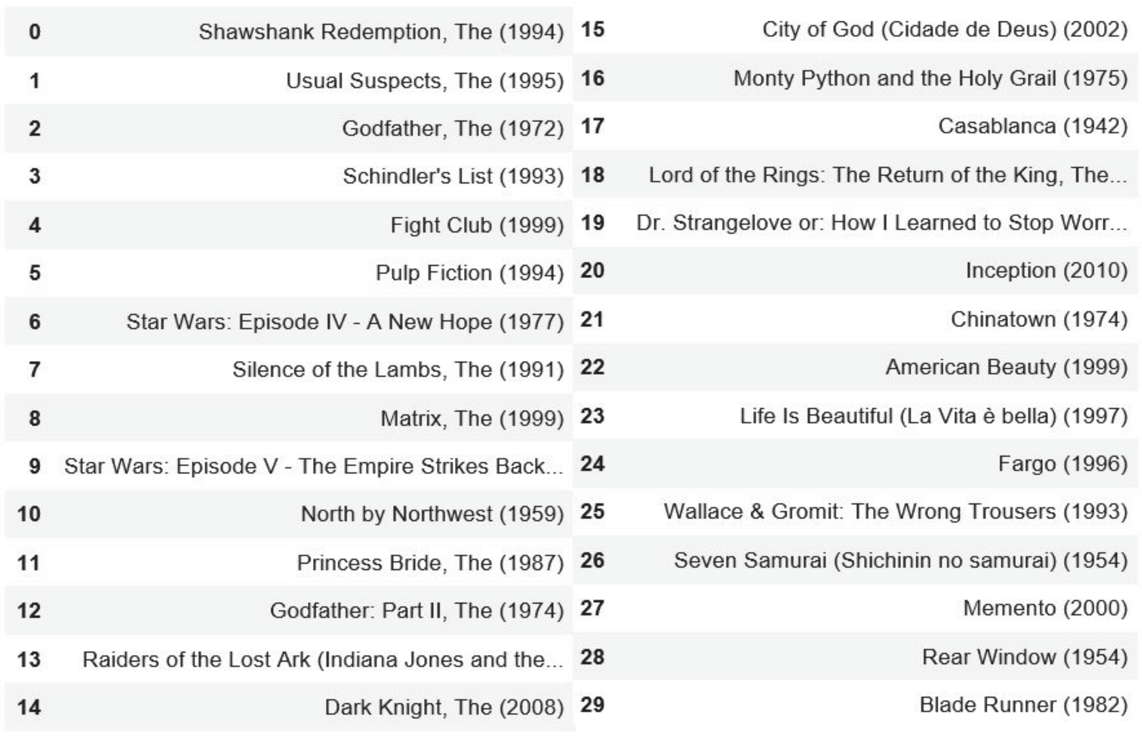
R is the average rating of the movie

C is the mean vote across the whole report

The next step is to determine an appropriate value for m, the minimum votes required to be listed in the chart. We will use 9**5th percentile** as our cutoff. In other words, for a movie to feature in the charts, it must have more votes than at least 95% of the movies in the list.

- mean vote across the whole report (C): **3.13**
- minimum votes required to be listed in the chart (m): **8463.75**
- **1061** movies are qualified

  The list of the 30 best movies to recommend are:



0	Shawshank Redemption, The (1994)
1	Usual Suspects, The (1995)
2	Godfather, The (1972)
3	Schindler's List (1993)
4	Fight Club (1999)
5	Pulp Fiction (1994)
6	Star Wars: Episode IV - A New Hope (1977)
7	Silence of the Lambs, The (1991)
8	Matrix, The (1999)
9	Star Wars: Episode V - The Empire Strikes Back...
10	North by Northwest (1959)
11	Princess Bride, The (1987)
12	Godfather: Part II, The (1974)
13	Raiders of the Lost Ark (Indiana Jones and the...
14	Dark Knight, The (2008)
15	City of God (Cidade de Deus) (2002)
16	Monty Python and the Holy Grail (1975)
17	Casablanca (1942)
18	Lord of the Rings: The Return of the King, The...
19	Dr. Strangelove or: How I Learned to Stop Worr...
20	Inception (2010)
21	Chinatown (1974)
22	American Beauty (1999)
23	Life Is Beautiful (La Vita è bella) (1997)
24	Fargo (1996)
25	Wallace & Gromit: The Wrong Trousers (1993)
26	Seven Samurai (Shichinin no samurai) (1954)
27	Memento (2000)
28	Rear Window (1954)
29	Blade Runner (1982)

3. Content-Based Filtering

The recommender we built in the previous section suffers some severe limitations. For one, it gives the same recommendation to everyone, regardless of the user's personal taste. If a person who loves romantic movies (and hates action) had to look at our Top 10 Chart, s/he wouldn't probably like most of the movies. If s/he were to go one step further and look at our charts by genre, s/he wouldn't still be getting the best recommendations.

So in this section, I set an engine to recommend based on the user's rating history. A simple example is using the mean as an aggregation function. For implementation step check the Ipython notebook.

The RMSE for content-based filtering is 0.9969 which is a acceptable value for this large dataset.

The total elapsed time for this evaluation was 61.474 sec.

4. Collaborative Filtering

Collaborative Filtering is based on the idea that users similar to a me can be used to predict how much I will like a particular product or service those users have used/experienced but I have not.

1- Simple Collaborative filtering based on mean

Recommend based on other user's rating histories. A simple example is using the mean as an aggregation function.

The results for some random user and movies Id is presented below:

```
Collaborative mean filter for userId: 1 and movieId: 151 is: [3.31060606]
Actual rating value is: 4.0

Collaborative mean filter for userId: 1 and movieId: 29 is: [3.80232558]
Actual rating value is: 3.5

Collaborative mean filter for userId: 138493 and movieId: 55269 is:
[4.06666667]
Actual rating value is: 5.0
```

2-Matrix Factorization-based algorithms

The idea is basically to take a large (or potentially huge) matrix and factor it into some smaller representation of the original matrix. You can think of it in the same way as we would take a large number and factor it into two much smaller primes. We end up with two or more lower dimensional matrices whose product equals the original one.

When we talk about collaborative filtering for recommender systems we want to solve the problem of our original matrix having millions of different dimensions, but our “tastes” not being nearly as complex. Even if i’ve viewed hundreds of items they might just express a couple of different tastes. Here we can actually use matrix factorization to mathematically reduce the dimensionality of our original “all users by all items” matrix into something much smaller that represents “all items by some taste dimensions” and “all users by some taste dimensions”. These dimensions are called latent or hidden features and we learn them from our data.

Doing this reduction and working with fewer dimensions makes it both much more computationally efficient and but also gives us better results since we can reason about items in this more compact “taste space”.

If we can express each user as a vector of their taste values, and at the same time express each item as a vector of what tastes they represent. You can see we can quite easily make a recommendation. This also gives us the ability to find connections between users who have no specific items in common but share common tastes.

SVD is a matrix factorization technique that is usually used to reduce the number of features of a data set by reducing space dimensions from N to K where K < N. For the purpose of the recommendation systems however, we are only interested in the matrix factorization part keeping same dimensionality. The matrix factorization is done on the user-item ratings matrix. From a high level, matrix factorization can be thought of as finding 2 matrices whose product is the original matrix.

(https://medium.com/@m_n_malaeb/singular-value-decomposition-svd-in-recommender-systems-for-non-math-statistics-programming-4a622de653e9)

I use **Surprise** library which has several powerful algorithms like **Singular Value Decomposition (SVD), Non-negative Matrix Factorization (NMF), K Nearest Neighbor (KNN), and Co-Clustering** to minimize RMSE (Root Mean Square Error) and give recommendations.

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9165  0.9119  0.9194  0.9091  0.9220  0.9158  0.0047
MAE (testset)     0.7058  0.7010  0.7093  0.7029  0.7101  0.7058  0.0035
Fit time          16.00   15.48   14.18   17.57   15.86   15.82   1.09
Test time         0.25    0.25    0.25    0.23    0.36    0.27    0.05
Total Elapsed time with the model is: 81.90


Evaluating RMSE, MAE of algorithm NMF on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9560  0.9636  0.9502  0.9585  0.9592  0.9575  0.0044
MAE (testset)     0.7364  0.7387  0.7284  0.7336  0.7390  0.7352  0.0039
Fit time          15.98   15.27   15.72   12.91   13.61   14.70   1.22
Test time         0.27    0.22    0.34    0.20    0.31    0.27    0.05
Total Elapsed time with the model is: 76.32 sec

Evaluating RMSE, MAE of algorithm SlopeOne on 5 split(s).
```

```
                Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9487  0.9463  0.9371  0.9449  0.9491  0.9452  0.0043
MAE (testset)     0.7264  0.7261  0.7208  0.7232  0.7278  0.7248  0.0025
Fit time          9.84    10.11   11.68   10.86   11.21   10.74   0.68
Test time         11.49   13.06   12.45   10.91   8.92    11.37   1.43
Total Elapsed time with the model is: 112.29 sec
```

Evaluating RMSE, MAE of algorithm **KNNBaseline** on 5 split(s).

```
                Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9148  0.9180  0.9152  0.9191  0.9149  0.9164  0.0018
MAE (testset)     0.7050  0.7066  0.7044  0.7064  0.7034  0.7052  0.0012
Fit time          1.50    1.56    1.59    1.61    1.58    1.57    0.04
Test time         5.85    5.47    5.25    6.03    5.73    5.67    0.28
Total Elapsed time with the model is: 37.747 sec
```

Evaluating RMSE, MAE of algorithm CoClustering on 5 split(s).

```
                Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9785  0.9831  0.9782  0.9906  0.9765  0.9814  0.0051
MAE (testset)     0.7587  0.7585  0.7542  0.7660  0.7567  0.7588  0.0039
Fit time          12.22   6.37    6.58    6.59    6.59    7.67    2.28
Test time         0.44    0.17    0.47    0.17    0.47    0.34    0.14
Total Elapsed time with the model is: 41.57
```

From the results**, SVD** and **KNN** provide the lowest RMSE for the test set.

Now let's compare the predicted rating values of some samples with their actual values:

UserId MovieId	**SVD**	**NMF**	**Slope One**	**KNN**	**CoClutering**	**Actual**
userId=1 movieId=151	3.68	3.4	3.49	3.77	3.28	4

userId=1 movieId=29	3.87	3.86	3.72	3.94	3.97	3.5
userId=138493 movieId=55269	3.84	3.48	3.48	3.82	3.48	5

5. ALS Implicit Collaborative Filtering

Alternating Least Squares (ALS) is a the model we'll use to fit our data and find similarities.

ALS is an iterative optimization process where we for every iteration try to arrive closer and closer to a factorized representation of our original data.

We have our original matrix R of size u x i with our users, items and some type of feedback data. We then want to find a way to turn that into one matrix with users and hidden features of size u x f and one with items and hidden features of size f x i. In U and V we have weights for how each user/item relates to each feature. What we do is we calculate U and V so that their product approximates R as closely as possible: R ≈ U x V.

By randomly assigning the values in U and V and using least squares iteratively we can arrive at what weights yield the best approximation of R. The least squares approach in it's basic forms means fitting some line to the data, measuring the sum of squared distances from all points to the line and trying to get an optimal fit by minimising this value.

With the alternating least squares approach we use the same idea but iteratively alternate between optimizing U and fixing V and vice versa. We do this for each iteration to arrive closer to R = U x V.

For more info read article:

https://medium.com/radon-dev/als-implicit-collaborative-filtering-5ed653ba39fe

So now let's check ALS for some well known movies, and see how does it recommend:

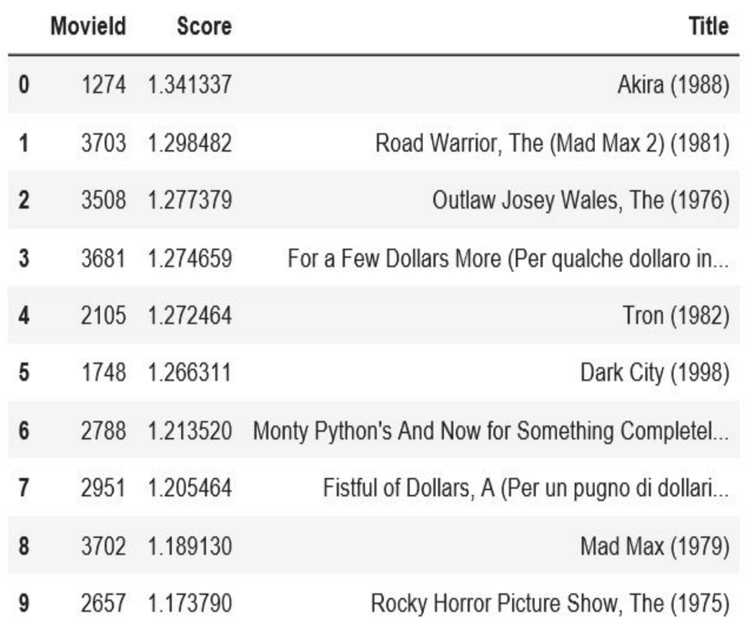
Similar items to 'Shawshank Redemption':

	MovieId	Score	Title
0	318	0.310394	Shawshank Redemption, The (1994)
1	527	0.309371	Schindler's List (1993)
2	593	0.308900	Silence of the Lambs, The (1991)
3	296	0.308599	Pulp Fiction (1994)
4	50	0.308179	Usual Suspects, The (1995)
5	356	0.307320	Forrest Gump (1994)
6	110	0.307277	Braveheart (1995)
7	47	0.306583	Seven (a.k.a. Se7en) (1995)
8	480	0.305257	Jurassic Park (1993)
9	589	0.303026	Terminator 2: Judgment Day (1991)

Find similar items to 'Pulp Fiction':

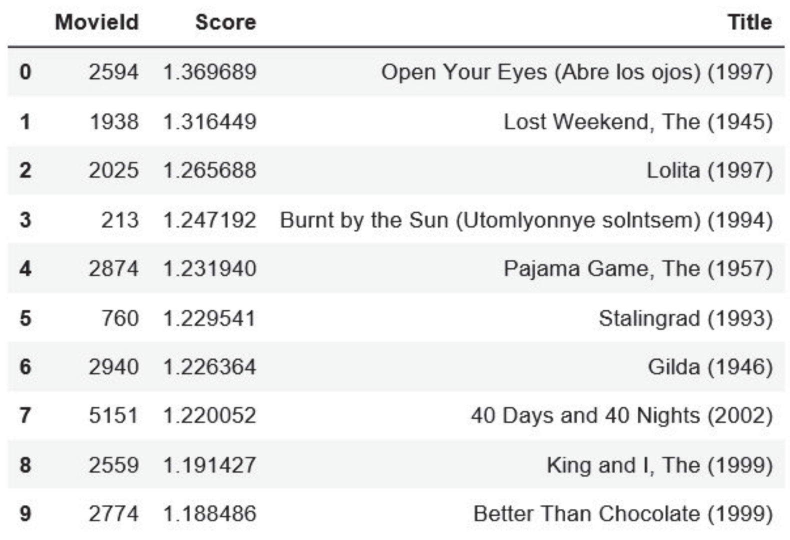
	MovieId	Score	Title
0	296	0.306645	Pulp Fiction (1994)
1	50	0.305682	Usual Suspects, The (1995)
2	593	0.305319	Silence of the Lambs, The (1991)
3	318	0.304871	Shawshank Redemption, The (1994)
4	527	0.304766	Schindler's List (1993)
5	47	0.304278	Seven (a.k.a. Se7en) (1995)
6	110	0.303050	Braveheart (1995)
7	356	0.302907	Forrest Gump (1994)
8	480	0.302016	Jurassic Park (1993)
9	589	0.300631	Terminator 2: Judgment Day (1991)

Recommend movies to userId=1:



	MovieId	Score	Title
0	1274	1.341337	Akira (1988)
1	3703	1.298482	Road Warrior, The (Mad Max 2) (1981)
2	3508	1.277379	Outlaw Josey Wales, The (1976)
3	3681	1.274659	For a Few Dollars More (Per qualche dollaro in...
4	2105	1.272464	Tron (1982)
5	1748	1.266311	Dark City (1998)
6	2788	1.213520	Monty Python's And Now for Something Completel...
7	2951	1.205464	Fistful of Dollars, A (Per un pugno di dollari...
8	3702	1.189130	Mad Max (1979)
9	2657	1.173790	Rocky Horror Picture Show, The (1975)

Recommend movies to userId=903:



	MovieId	Score	Title
0	2594	1.369689	Open Your Eyes (Abre los ojos) (1997)
1	1938	1.316449	Lost Weekend, The (1945)
2	2025	1.265688	Lolita (1997)
3	213	1.247192	Burnt by the Sun (Utomlyonnye solntsem) (1994)
4	2874	1.231940	Pajama Game, The (1957)
5	760	1.229541	Stalingrad (1993)
6	2940	1.226364	Gilda (1946)
7	5151	1.220052	40 Days and 40 Nights (2002)
8	2559	1.191427	King and I, The (1999)
9	2774	1.188486	Better Than Chocolate (1999)

6. Deep Learning