Netflix Movie Recommendation System

Fatima Soytemiz

June 22, 2021

Contents

1 Introduction

Nowadays, recommender systems are ubiquitous. Companies depend on them to keep their users engaged and keep their sales high. Giving good recommendations help users spend less time searching for their type of product and increase their engagement with the platform. This will help customers to continue with the service and have a good experience. For a company like Netflix in particular, good recommendations keep users engaging with new shows and movies they enjoy, and can therefore be the difference between users staying on the platform or switching to another streaming service.

The purpose of this project is to create a personalized movie scoring and recommendation system based on the user's previous movie ratings through machine learning training. The recommender system will be able to predict users rating for a new movie they haven't seen. Different people have different taste in movies, and this is not reflected in a single score that we see when we Google a movie. Our movie scoring system helps users instantly discover movies to their liking, regardless of how distinct their tastes may be.

2 What is a Recommendation System?

A recommendation system is a platform that provides its users with content based on their preferences. A recommendation system takes the information about the user as an input. This information can be in the form of the past usage of the product or the ratings that were provided to the product. It then processes this information to predict how much the user would rate or prefer the product. A recommendation system makes use of a variety of machine learning algorithms.

3 Data Acquisition

In this project, the dataset was used from one of the most famous competitions launched by Netflix with a one-million-dollar price. It is called the Netflix Grand Prize. It was an open competition for the best collaborative filtering algorithm to predict user ratings for films, based on previous ratings without any other information about the users or films. This competition started in October 2006 and lasted till 2009. The main goal of this competition was to find a more accurate recommendation system by reducing root mean square error (RMSE) to replace Netflix's own algorithm, called Cinematch.

The dataset contains a total of 100,480,507 ratings, based on 17,700 movies which come from a total of 480,189 users from the United States. The data can be reached from the Kaggle link.

The data is separated into two sets. The first dataset used is the movie list which has movie ID, the title of movie and corresponding year of release. The data table is as below.

	movie_id	year	name
0	1	2003	Dinosaur Planet
1	2	2004	Isle of Man TT 2004 Review
2	3	1997	Character
3	4	1994	Paula Abdul's Get Up & Dance
4	5	2004	The Rise and Fall of ECW

The first set consists of a list of movies with their rating by individual users. The ratings are on a range from 1 to 5. To protect customer privacy, each customer id has been replaced with a randomly-assigned id. The date of each rating and the title and year of release for each movie id are also provided. We represent this data as a matrix where one dimension represents users and the other dimension represents movies, the matrix is very sparse since most users have rated only a small portion of all the movies. The data table is as below.

	movie_id	user_id	rating	date
100480502	17770	1790158	4	2005-11-01
100480503	17770	1608708	3	2005-07-19
100480504	17770	234275	1	2004-08-07
100480505	17770	255278	4	2004-05-28
100480506	17770	453585	2	2005-03-10

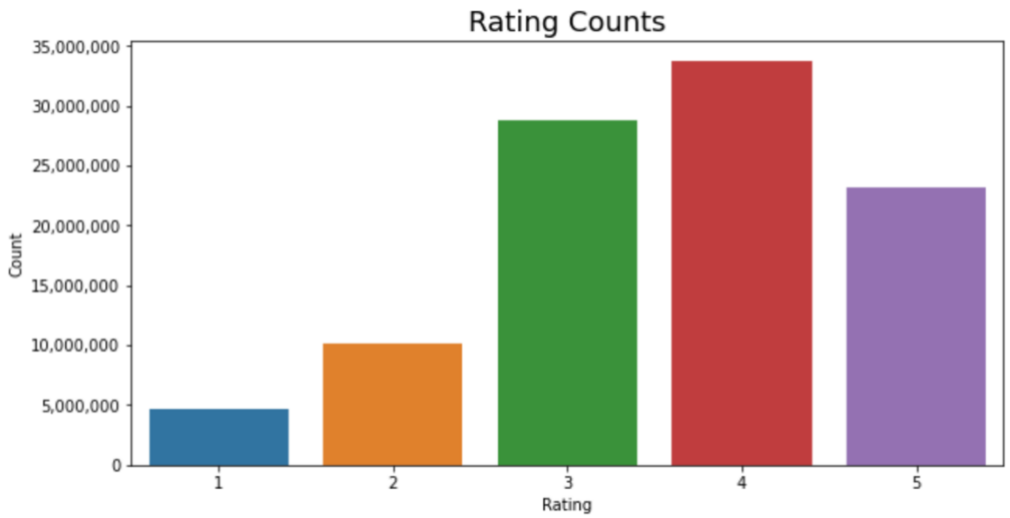
4 Exploratory Data Analysis

In statistics, exploratory data analysis is an approach of analyzing data sets to summarize their main characteristics, often using statistical graphs and other data visualization methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modelling or hypothesis testing task. Exploratory data analysis was promoted by John Tukey to encourage statisticians to explore the data, and possibly formulate hypotheses that could lead to new data collection and experiments. EDA is different from initial data analysis. which focuses more narrowly on checking assumptions required for model fitting and hypothesis testing, and handling missing values and making transformations of variables as needed. EDA encompasses IDA.

Before one is able to make an initial recommendation based on historic ratings, one must get more insight of the data. Therefore, a data analysis is done to get more acquainted and familiar with the reduced and selected data.

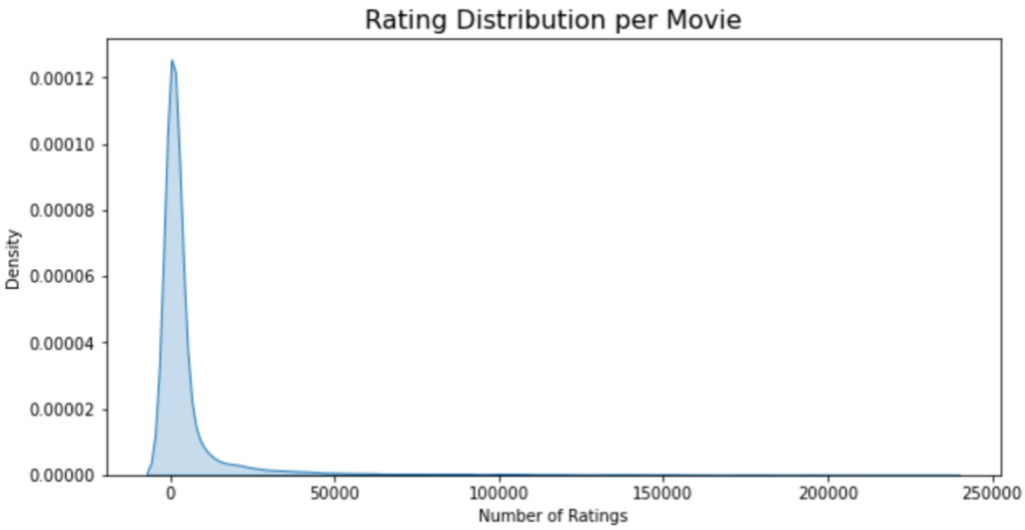
Distribution of Movie Ratings

The distribution of ratings is important for the recommender system. As can be obtained in the first histogram below, most common ratings are 3 or 4 stars. We can see that people tend to be relatively positive who rate 3 or more. Few people gave rating 1. Low rating movies mean they are generally really bad.



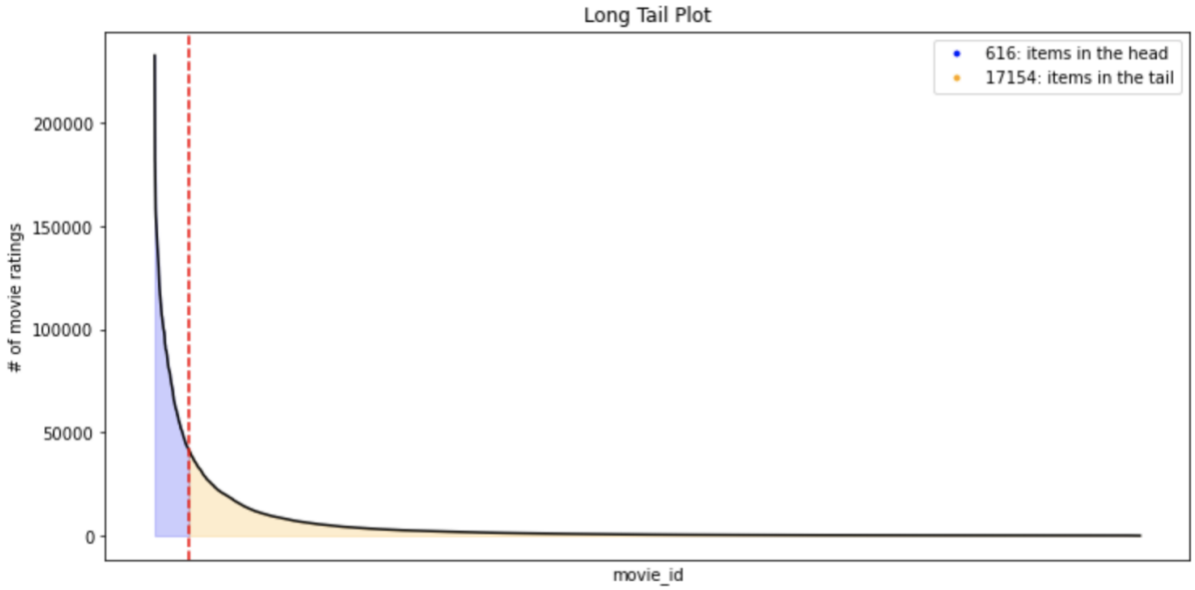
Distribution of Ratings Per Movie

As it is clearly seen in the chart below, most of the movies in the data received less than 5,000 ratings. Few movies have many ratings. The most rated movie has received 240K ratings.



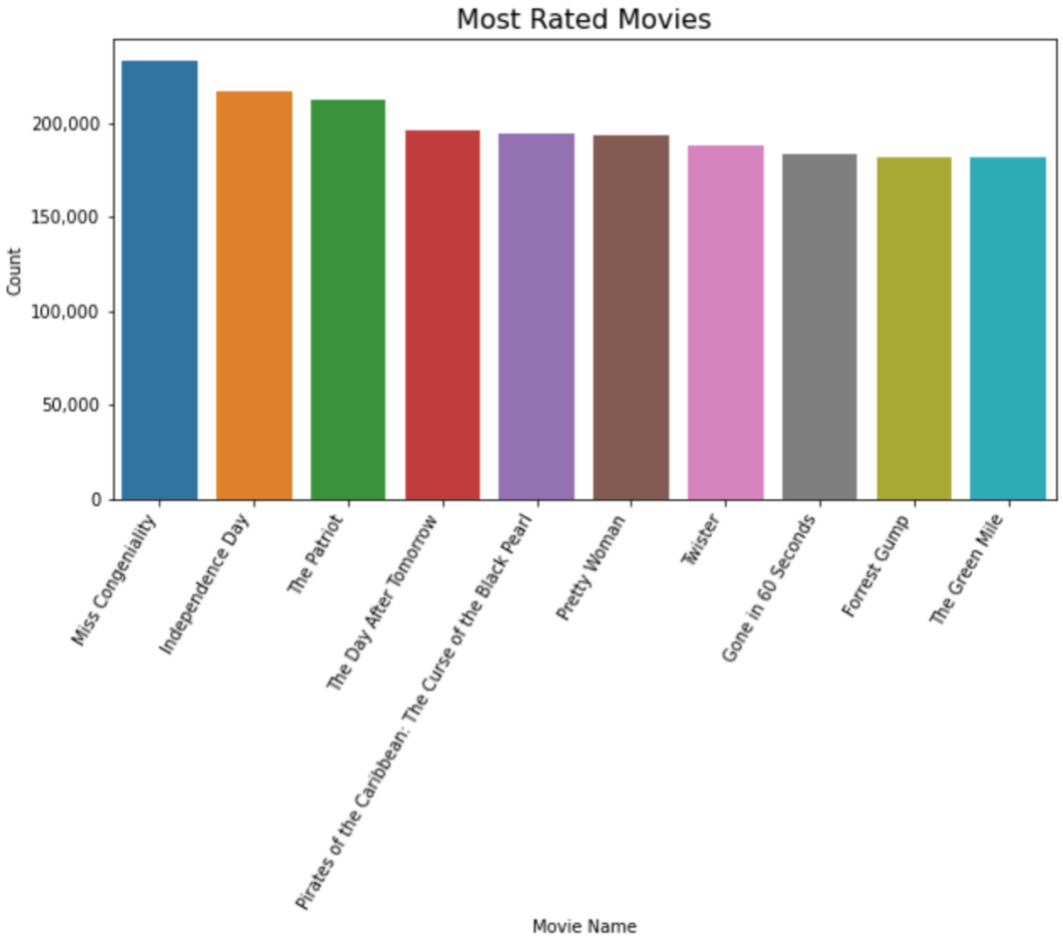
Long Tail Plot

We know that we have sparse data. The following chart clearly shows that. We see that 616 movies out of 17,700 movies received 50% of the ratings. It means that only 4% of movies got half of all ratings.



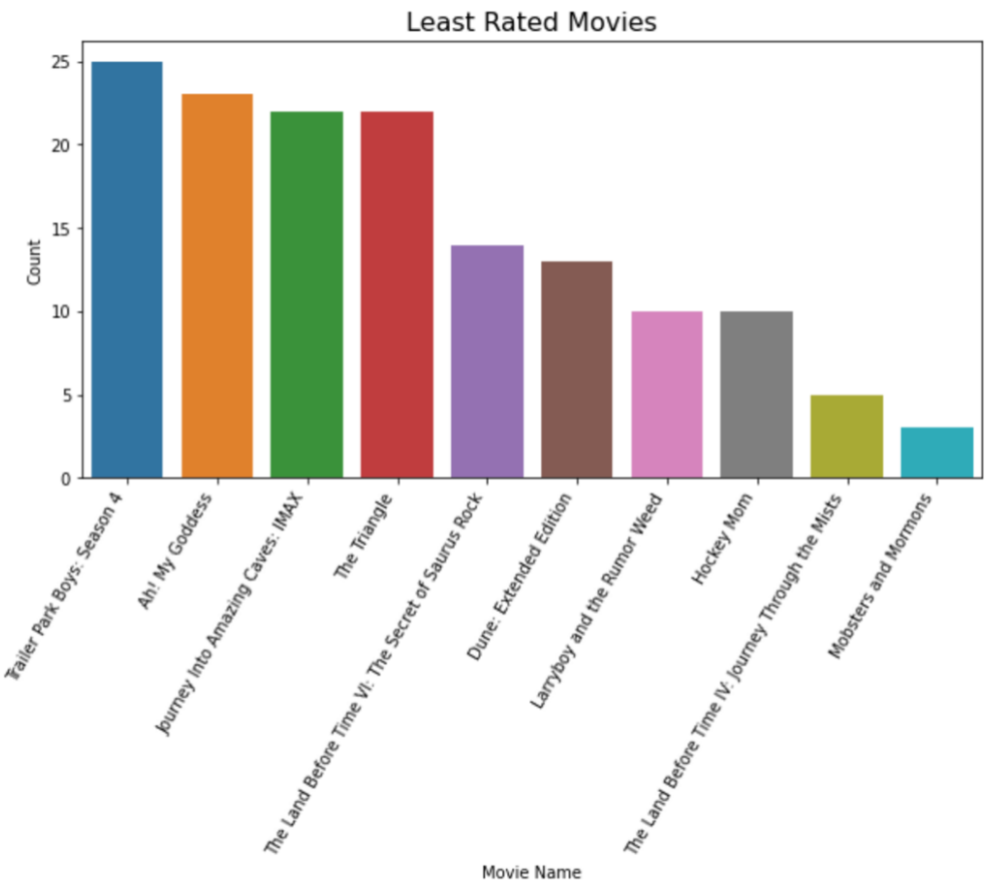
Most Common Rated Movies

It is clearly seen that the movie Miss Congeniality received the greatest number of ratings. It has 240K ratings.



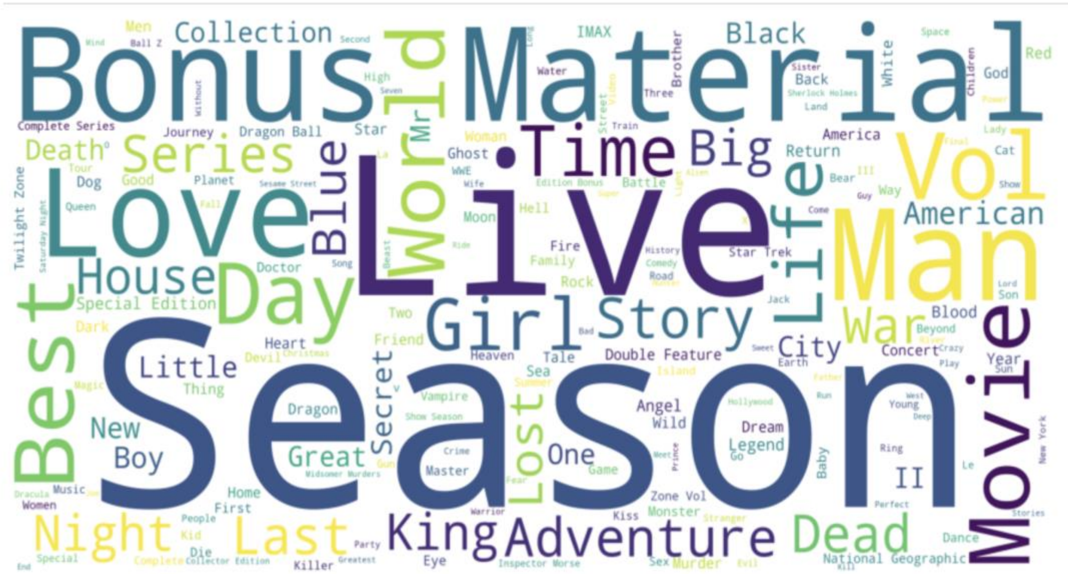
Least Common Rated Movies

In the chart below, we see that the movie Mobsters and Mormons received the lowest number of ratings. It has 3 ratings.



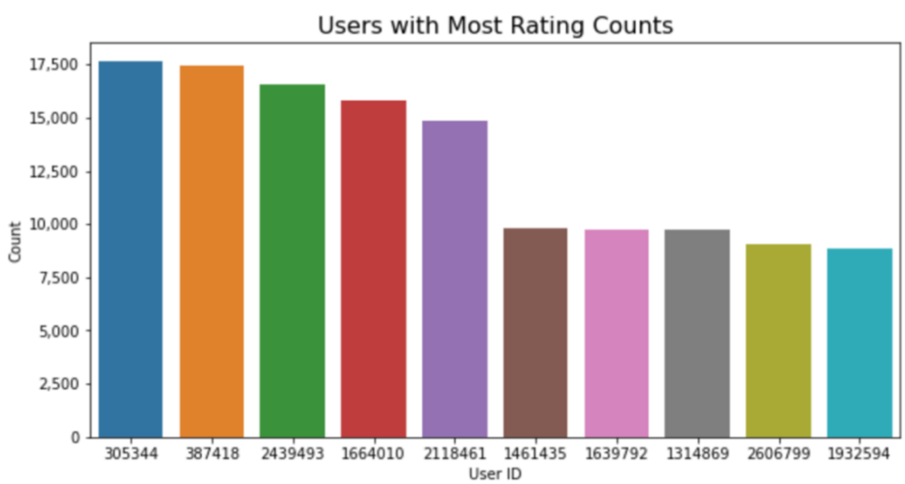
Title Word Cloud

The words Season, Live, Material, Bonus and Love are the most commonly used words in movie titles. Movie, Man and Girl are also among the most commonly occurring words. I think the word Season shows that episodes predominantly exist in the movie list.
The word Season is the most commonly used word in the movie list. I think it shows that episodes predominantly exist in the movie list. Live, Material, Bonus, Love, Movie, Man and Girl are also among the most commonly occurring words. I think it indicates the idea of the ubiquitous presence of romance in movies pretty well.



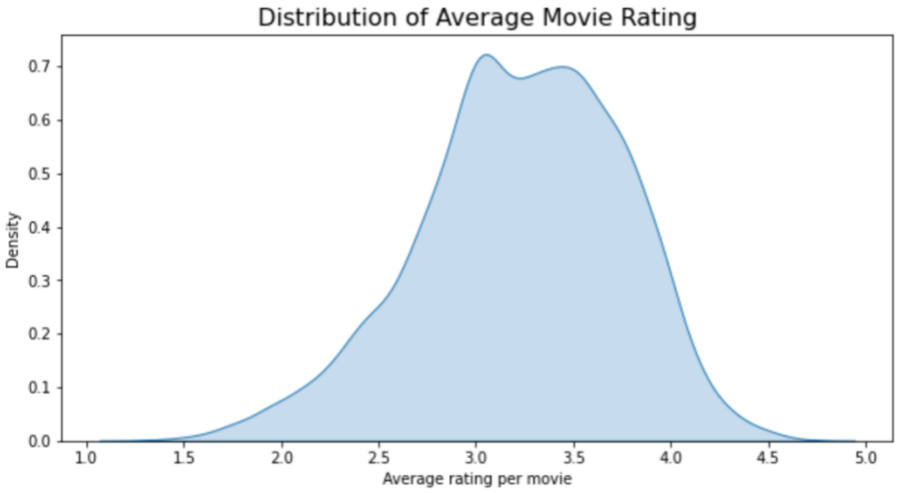
User Rating Counts

As it is seen in the chart below, one of the users rated 17,500 movies.



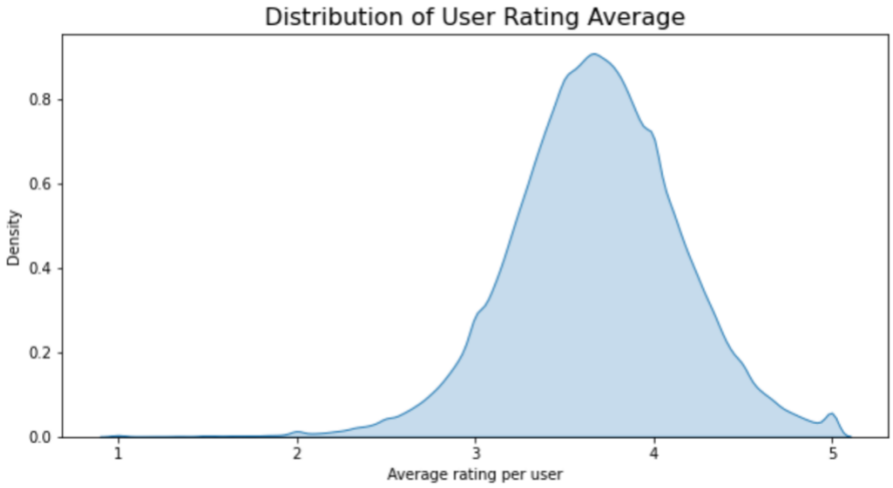
Average Movie Rating

In the chart below, we see that most of the movies are rated 3.2 on average.



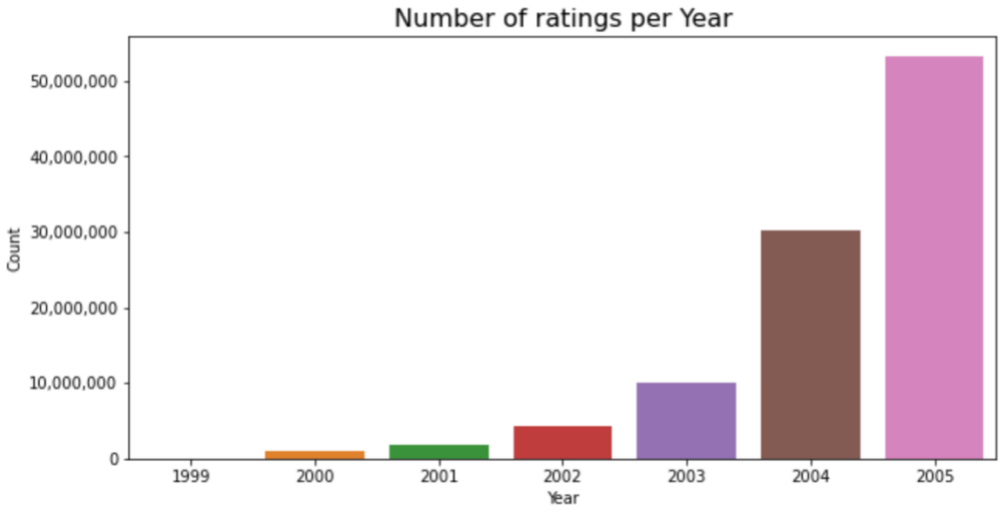
User Rating Average

In the chart below, we understand that most of the users gave 3.7 on average.



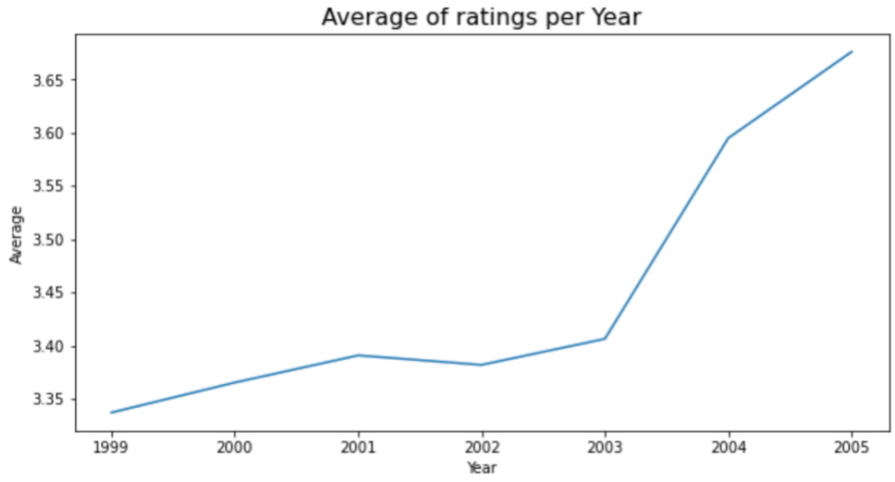
Ratings per Year

As it is seen in the chart below, the number of ratings increases exponentially in time. One of the reasons may be that movies are more realistic and creative in visual and editorial perspective and people love it.



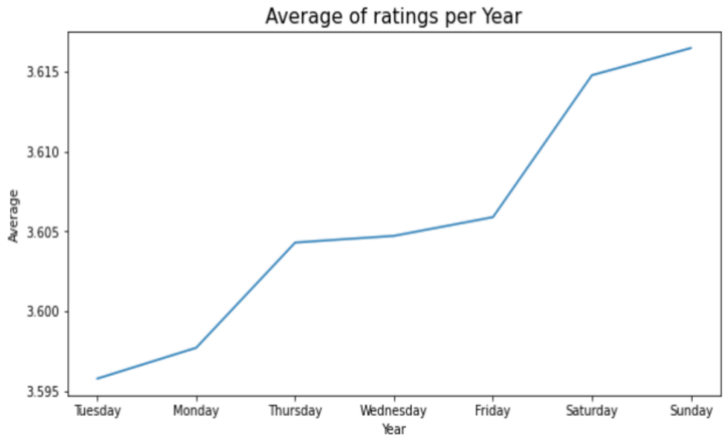
Average Rating per Year

The data shows that people give slightly high ratings to movies over time. One of the reasons may be that people like movies over time because of the developing technology in the movie industry that produces greater quality visuals.



Average Ratings per Day

Based on the data, people give high rates to movies on Sundays, even though they usually rate movies on Tuesdays.



5 Recommendation Systems

There are two main types of recommendation systems.

5.1 Content-based Recommendation Systems

The Content-Based Recommender relies on the similarity of the items being recommended. The basic idea is that if you like an item, then you will also like a “similar” item. It generally works well when it’s easy to determine the context/properties of each item. For example, if you have watched a movie in a sci-fi genre, the content-based recommendation system will provide you with suggestions for similar movies that have the same genre and you could potentially watch.

A content-based recommender system works with data that the user provides. Based on that data, a user profile is generated, which is then used to make suggestions to the user. As the user provides more inputs or takes actions on the recommendations, the engine becomes more and more accurate.

5.2 Collaborative-based filtering Recommendation Systems

Unlike the content-based filtering, collaborative-based filtering is entirely based on the past behavior and not on the context. More specifically, it is based on the similarity in preferences, tastes and choices of two users. It analyses how similar the tastes of one user is to another and makes recommendations on the basis of that. One key advantage of collaborative filtering is that it is independent of the product knowledge. Rather, it relies on the users with a basic assumption that what the users liked in the past will also like in the feature. For example, if person A watches crime, sci-fi and thriller genres and person B watches sci-fi, thriller and action genres then A will also like action and B will like crime genre.

There are two major approaches in collaborative filtering: user to user method and item to item method.

5.2.1 User-based Collaborative Filtering

User-based collaborative filtering (UBCF) recommends items by finding similar users by using cosine similarity. The main idea behind user-based collaborative filtering is that people with similar characteristics share similar taste. The procedure is to first find other users that are similar to a given user, then find the top-rated items purchased by those users. Those items are then recommended for the given user. The advantage of user-based collaborative filtering is the sparsity and scalability. Many recommender systems use data with lots of users and items, but with relatively few numbers of actual ratings. User-based collaborative filtering only uses necessary data, which reduces the run time.

5.2.2 Item-based Collaborative Filtering

The item-based collaborative filtering (IBCF) is a well-known technique and widely used in recommender systems. The basic idea behind this technique is to look for similar items based on items users have already rated. Item similarity is calculated by using cosine similarity. Item's rating is predicted based on how similar items have been rated by that user. The ratings are predicted using the user's own ratings on close neighboring items. Since the ratings are predicted using the ratings of the user herself, the predicted ratings tend to be much more consistent with the other ratings of that user.

In this project, item-based collaborative filtering will be used in building recommendation systems because we have less movies than users and it would be effective based on computation.

5.3 Cosine Similarity

How can item similarity be calculated? Cosine Similarity is a measurement that quantifies the similarity between two items. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The output value ranges from 0–1. 0 means no similarity, whereas 1 means that both the items are 100% similar.

The first step of this technique is to calculate the $n \times n$ similarity matrix S that contains all item-to-item similarities. The Cosine similarity is defined by the following formula, where $\vec{x}$ and $\vec{y}$ are rating vectors of two items:

$$S = Similarity_{cosine} = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \, \|\vec{y}\|}$$

6 Data Pre-Processing

6.1 Sparse Data

We create a matrix by representing users as rows and movies as columns. However, since we had 17,700 movies and 480,189 users, the size of the matrix will be extremely big (17,700 x 480,189). There are a total of 100,480,507 ratings in the data. Therefore, the sparsity of the matrix is 99%. There are two major constraints on computing, time and memory. We need to make sure our program doesn't consume all our memory. If we store a full matrix (dense matrix) it is simply inefficient. This is because a full array occupies a block of memory for each entry, so a n x m array requires n x m blocks of memory. It doesn't make sense to store so many zeros!

Sparse data structure allows us to store only non-zero values assuming the rest of them are zeros. This approach saves a lot of memory and computing time. Instead of storing all values in a dense matrix, a sparse matrix stores the non-zero values by just using their row and column indices. In this way, we will speed up the data process.

6.2 Train Test Split

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. First, we separated features and response variables as X and y. Then, we divided the dataset into the training and test sets. We have used 20% of the data for testing and 80% of the dataset for the training. Splitting our dataset is essential for an unbiased evaluation of prediction performance. We use the training set to build and train the model. Once the model is ready, we will test it on the testing set to see how well it performs.

7 Modeling

In this part, I will use different collaborative and machine learning base models to recommend movies. Basically, they will be predicting a user rating.

7.1 Surprise Package Models

Surprise is a recommendation system library, one of the Python scikit series. It is easy to use and supports multiple recommendation algorithms. for building and analyzing recommender systems that deal with explicit rating data. It provides various ready-to-use predictive

algorithms such as baseline algorithms, neighbourhood methods, matrix factorization and many others. It provides tools to evaluate, analyze and compare the algorithms' performance. Cross-validation procedures can be run very easily using powerful CV iterators. I have used the algorithms listed below.

7.1.1 Basic Algorithms

NormalPredictor: This is an algorithm that predicts a random rating based on the distribution of the training set, which is assumed to be normal. This is one of the most basic algorithms that do not do much work.

BaselineOnly: This algorithm predicts the baseline estimate for a given user and item.

7.1.2 K-NN Algorithms

KNNBasic: This is a basic collaborative filtering algorithm.

KNNWithMeans: KNNWithMeans is a basic collaborative filtering algorithm, taking into account the average value of each user's rating.

KNNWithZScore: This is a basic collaborative filtering algorithm, taking into account the z-score normalization of each user.

KNNBaseline: KNNBaseline is a basic collaborative filtering algorithm taking into account a baseline rating.

7.1.3 Matrix Factorization-Based Algorithms

SVD: This algorithm is equivalent to Probabilistic Matrix Factorization. In general, SVD is used to reduce the number of features of a data set.

SVDpp: This algorithm is an extension of SVD that takes into account implicit ratings.

NMF: This is a collaborative filtering algorithm based on Non-negative Matrix Factorization. It is very similar to SVD.

SlopeOne: This is a simple but accurate collaborative filtering algorithm

Co-clustering: This is a collaborative filtering algorithm based on collaborative clustering.

7.2 Linear Regression

Linear regression is one of the supervised Machine learning algorithms that observes continuous features and predicts an outcome. Therefore, the outcome will be a predicted rating of users for each movie. The data set has variables user_id, movie_id and rating. I have added more variables. I added top 5 similar users, top 5 similar movies from item-item similarity by using cosine similarity. I also added a user global rating average and a movie global rating average then the new data set became as below:

user_id	movie_id	rating	sm1	sm2	sm3	sm4	sm5	su1	su2	su3	su4	su5	userGlobalAverage	movieGlobalAverage
1032298	7745	3	4	4	2	5	4	4	2	3	4	3	3.684647	4.082576
792958	10906	4	5	5	2	0	0	4	4	3	3	3	3.109756	3.496262
2518644	2452	5	4	5	3	3	4	5	5	4	5	5	3.625731	4.434708
1995860	10162	3	5	5	3	4	3	2	5	5	5	4	3.080985	3.985408
904391	12299	2	3	5	4	3	5	4	4	2	3	4	3.198358	3.646641

8 Model Evaluation

In this project, root mean square error is used to evaluate models. Root mean square error is the average difference between the predicted value and the actual value. Root mean square estimation is used as a protective measure to reduce the individual impact from the errors of each individual measurement, so that no one majorly faulty estimation will skew the result too much. Root mean square estimation is calculated according to the following formula:

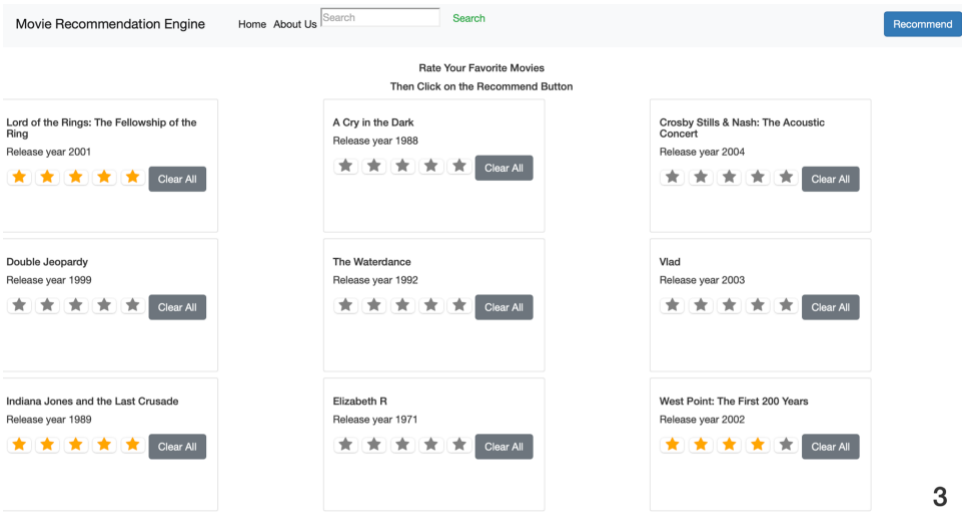
$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

In this formula, n is the number of estimated data points, y_i is the actual value and $\hat{y}_i$ is the predicted value. RMSE is a good measure of accuracy. The lower the RMSE, the better the model. Based on the model comparison table below, Linear Regression has lowest RMSE, 0.8913. Therefore, Linear Regression is used to predict user ratings to be able to recommend movies that users will love. Recommendation system will be built based on a machine learning based model.

	test_rmse
LinearRegression	0.8913
SVDpp	0.9576
BaselineOnly	0.9677
SVD	0.9732
KNNBaseline	0.9782
KNNBasic	1.0245
SlopeOne	1.0333
KNNWithZScore	1.0500
KNNWithMeans	1.0530
NormalPredictor	1.4560

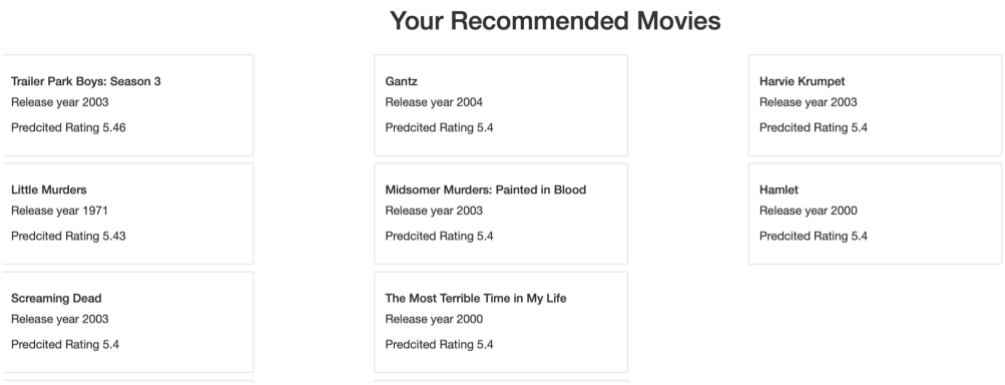
9 Web Application

The webpage needs to look very simple yet informative enough for everyone to understand. The movie recommendation web application for the Netflix movies is built by using the flask framework of python language. The web application home page consists of Netflix movies. First, users should rate movies they have watched before. Users can select movies from the list on the home page or they can search the movie by using the search engine. A user rating is shown below:



3

Next, they should click on the “Recommend” button on the right side of the homepage. Upon submission, the movie/s are captured at the backend and further processed. Then the recommendation system will provide a user’s personalized recommended movie list as shown below:



The recommendation system is using a machine learning based model. Based on our model evaluation results, we decided to pick Linear Regression since it gave us the lowest RMSE. The application runs the recommendation algorithm to predict the movie that the user potentially likes. The more the rated movies, the better the recommendation.
The application runs the recommendation algorithm to predict movies by feeding the model similar users and movies from cosine similarity matrix.

10 Conclusion

Recommender systems are a powerful technology for extracting additional value for a business from its user databases. The importance of these recommender systems is increasing along with the increase in the volume of user data. New technologies are needed that can dramatically improve the scalability of recommender systems.

In this project, I presented and experimentally evaluated item-item based collaborative filtering using basic algorithms, K-NN algorithms and matrix factorization-based algorithms from Surprise package and Linear Regression by adding top similar movies and users as a new feature by using cosine similarity. I have evaluated these models by calculating RMSE for each model. Results show that Linear Regression gave the lowest RMSE. With these insights, Linear Regression is picked to recommend movies to the users who potentially love them.

The movie recommender system is useful to any business that makes money via recommendation. This includes Netflix, DisneyPlus, Hulu, etc. These companies could be potential clients of this recommendation engine. The system will give a good recommendation

since the model gives minimal error compared to other methods. Giving good recommendations directly entails one or many of the following:

1. Customers buy a particular product or service leading to increased revenue or sales.
2. Customers use the platform more frequently due to the quality and relevance of content shown to them.
3. Better User Experience. Customers spend less time searching and more time watching. The pain of discovery is eliminated.