Master of Arts HSG
in Banking and Finance

University of St.Gallen

Machine Learning

Master of Arts in Banking and Finance

Spring term 2020

Development of a Movie Recommendation Engine

Group term paper

Nils Kaufmann | nils.kaufmann@student.unisg.ch | 19-602-796
Daniel Rüger | daniel.rueger@student.unisg.ch | 09-593-377
Jan Scheidegger | jan.scheidegger@student.unisg.ch | 15-055-411
Yifan Zhu | yifan.zhu@student.unisg.ch | 19-600-105

submitted on May 14, 2020

Prof. Dr. Anna-Lena Horlemann

Abstract

Today, streaming platforms such as Netflix, Amazon Prime or Disney Plus offer tens of thousands of movies. Given this huge amount, users face an information overload problem which makes the choice of a movie best-suited to their interests time-consuming and complicated. In order to increase convenience and the quality of movie selection for users, we present various approaches for making movie recommendations. We find that the presented strategies are complementary in the sense that they apply to different situations. Demographic and content-based filtering techniques allow to make movie recommendations to unknown users but do not improve over time. In contrast, recommendation systems based on collaborative filtering, logistic regressions or decision trees continuously learn from the watching behaviour of users on the platform. However, they require sufficient training and cannot be applied to make recommendations to relatively new users.

Table of Contents

List of Figures

1 Introduction

The amount of data available on the internet is increasing rapidly (Bridle, 2010). Internet users are thus confronted with a large amount of information which can easily become overstraining (Bawden & Robinson, 2009). They need to invest considerable time and resources in order to analyse information, and choosing the right product can become complicated (Li, 2017). A field which is affected by this evolution is movie selection (Cui, 2017). Platforms such as Netflix or Amazon Prime offer their customers a large number of movies. Thus, for users of those platforms, it is difficult to identify the movies best-suited to their interests and mood.

This information overload can be reduced through recommendation systems, which screen information to help users make optimal decisions. Algorithms allow streaming platforms to personalize recommendations with respect to user characteristics and learn from the user's watching behaviour over time. Moreover, an appropriate recommendation system can substantially raise revenues of a firm (Zhou, Wilkinson, Schreiber & Pan, 2008). This is because customers tend to spend more time on the platform, buy bigger plans and are less likely to cancel their subscriptions. For instance, according to YouTube's Chief Product Officer Neal Mohan, recommendation engines account for more than 70% of the time users spend on the company's website.[1]

Various methodologies can be used to implement recommendation systems. Three important categories are content-based and collaborative filtering, as well as hybrid models. Content-based filtering approaches group items (e.g. movies) based on their features and recommend items that are similar to those a user has liked in the past. In contrast, collaborative filtering methods group users based on their characteristics. Recommendations under this model depend on which items users from the same group have watched and enjoyed. In order to avoid the drawbacks of both methods, hybrid models have been proposed (Reddy et al., 2019).

Many papers which present movie recommendation systems based on machine learning methods have been published in recent years. For instance, a k-NN collaborative filtering algorithm has been used by Cui (2017). Reddy et al. (2019) have applied a content-based method which uses genre correlations. Wang, Sang, Zeng,

[1] https://www.cnet.com/news/youtube-ces-2018-neal-mohan/

and Hirokawa (2017) have implemented a support vector machine method and improved particle swarm optimisation.

Recommendation systems are also applied to improve recommendations in multiple other fields like music and news. For example, Spotify considers contextual information such as demographic context, user location, type of the device and time of day to improve its music recommendation system (Volokhin & Agichtein, 2018).

The aim of this paper is to develop a movie recommendation system. For this purpose, we use both machine learning and non-machine learning algorithms. Our recommendation system helps users to save time and makes the process of finding movies that match their individual preferences more convenient. To build the recommendation system, we consider demographic, content-based and collaborative filtering techniques as well as a decision tree algorithm and a logistic regression. Considering multiple approaches enables us to identify the models which are suited best for making movie recommendations. As we will see, recommendation systems should combine different techniques, as the optimal model depends on the amount of information that is available on a user.

The paper is structured as follows. First, we describe the considered dataset in more detail. In the application section that follows, we implement various movie recommendation algorithms and discuss their scope of application, advantages and drawbacks. Finally, we elaborate how to best combine the algorithms discussed and provide an outlook for future analyses in our conclusion.

2 Data and Methodology

2.1. Data Source and Methodology

The dataset we use for our movie recommendation analysis was retrieved from the online community Kaggle[2]. Specifically, the details, credits and keywords of the movies in the dataset were obtained from the TMDB Open API and the user ratings were scrapped from the MovieLens website. The original dataset comprises metadata of 45’000 movies and 26 million ratings from 27’000 users. For each movie, the dataset contains information on more than 20 features. This allows for a large variety of

[2] https://www.kaggle.com/rounakbanik/movie-recommender-systems?select=movies_metadata.csv

machine learning algorithms to be implemented. Since, for our use case, we do not need such a large number of ratings and as we want to keep computation times low, we use a smaller subset which contains 100'000 ratings.

We have two files in the data cleaning process. The movies_data.R file contains 4'960 movies and 20 feature variables (Figure 2.1). The second ratings_data.R file contains 21'561 ratings from 671 different users. The columns of the data in the csv files were json formatted and needed to be reformatted during the data cleaning process so that they could be used individually in our analysis. Due to the large amount of observations, we were able to remove entries with NAN values to get a balanced data frame.

features	descriptions
movie_id	A unique identifier for each movie
cast	The main actors in the movie
budget	The movie's budget in USD
genre	The genre(s) of the movie
keywords	Keywords that describe the movie
overview	A brief description of the movie
popularity	Numeric value that shows whether a movie is trending
year	The movie's release year
revenue	The revenue generated by the movie
runtime	The duration of the movie in minutes
title	The title of the movie
vote_average	The avg. rating the movie received on a scale from 0.5 to 5
vote_count	The number of reviews for the movie
weighted_rating	IMDB weighted rating calculated from vote_count and vote_average

Figure 2.1: Overview of the variables used in our analysis

For our movie recommendation engine, we use the methodology proposed by Ibtesam Ahmed[3]. We extend her analysis by implementing additional algorithms and testing the adequacy of the recommendations based on the different approaches.

2.2. Descriptive Statistics

Ahead of discussing the implemented algorithms, let us have a glance at the univariate relationship between the numeric movie features. This is especially important for the

[3] https://www.kaggle.com/ibtesama/getting-started-with-a-movie-recommendation-system

feature selection in the logistic regression model. As you can see from Figure 2.2, most variables have a positive correlation. For instance, weighted rating is highly correlated with vote count and vote average which are used for its construction. The high correlation between budget and revenue is also reasonable, since movie productions with a high budget can spend more on marketing and thus generate higher revenues. In order to avoid multicollinearity in our logistic regression, we therefore drop vote count, vote average and budget.

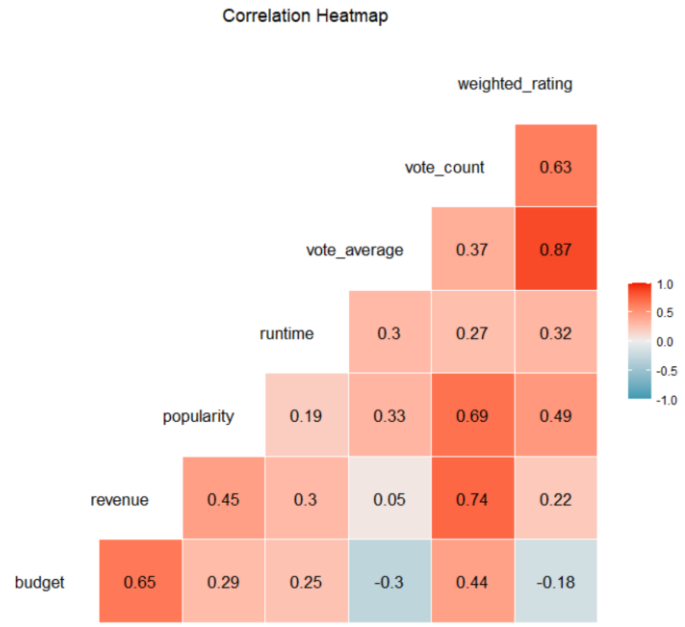


Figure 2.2: Correlations of the numeric variables

The user ratings variable is used as the target variable in our algorithms and has a slightly left skewed distribution (Figure 2.3). The users were able to rate the movies with scores between 0.5 and 5. Most of the ratings appear in the middle and upper range with a mean rating of 3.55.

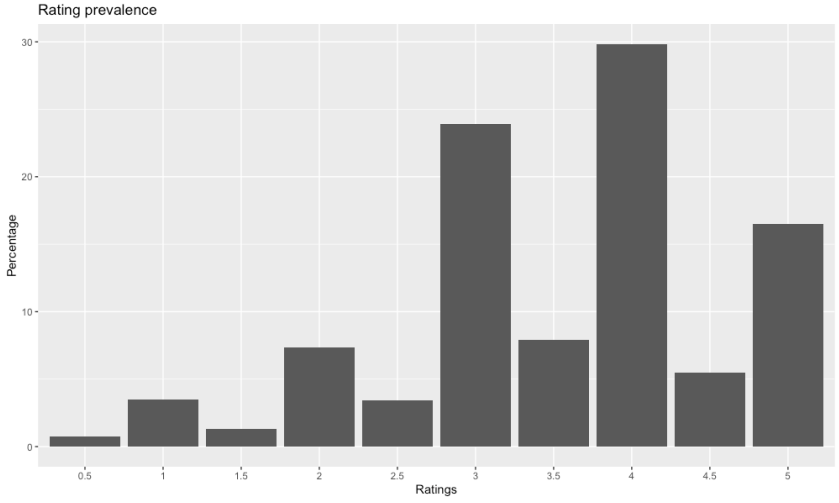


Figure 2.3: Distribution of user ratings

3 Movie Recommendation System

We now turn to the core of our project. In this section we apply various techniques for making movie recommendations. First, we look at simple demographic filtering approaches which utilize demographic user characteristics such as age or gender and do not involve any machine learning. Instead, those techniques make general recommendations to all users that share certain attributes. Afterwards, we consider content-based filtering techniques which try to identify movies similar to those a given user has enjoyed. Moreover, we implement collaborative filtering methods. Rather than identifying similar films, the goal here is to group users based on their interests. Lastly, we build recommendation systems for individual users based on a decision tree algorithm and a logistic regression.

We proceed as follows: At first, we briefly describe the theoretical background of the individual algorithm. Afterwards, we apply the algorithm to the dataset and interpret the results. Finally, we discuss advantages and disadvantages of each model.

3.1. Demographic Filtering

A simple approach for making movie recommendations is to suggest the best rated or most popular (“trending”) movies in the dataset (Figure 3.1). While this approach is associated with low computation cost and effort, it completely ignores differences between users. Hence, this method is only appropriate for making suggestions to new users with unknown characteristics. However, there are obvious drawbacks associated with this strategy. For instance, adult films may be suggested to children or French movies to English speakers.

title	weighted_rating
The Shawshank Redemption	8.2928
The Godfather	8.2226
The Dark Knight	8.1677
Fight Club	8.1353
Pulp Fiction	8.1179
Forrest Gump	8.0170
Inception	7.9954
Schindler's List	7.9727
Interstellar	7.9704
Whiplash	7.9690

title	popularity
Minions	547.4883
Wonder Woman	294.3370
Beauty and the Beast	287.2537
Baby Driver	228.0327
Big Hero 6	213.8499
Deadpool	187.8605
Guardians of the Galaxy Vol. 2	185.3310
Avatar	185.0709
John Wick	183.8704
Gone Girl	154.8010

Figure 3.1: Best rated and most popular movies in the dataset

A better tactic would therefore be to ask users for their age, gender, education, spoken languages etc. and tailor the recommendation to their demographic group. This way, differences in terms of demographics could be captured. However, differences between users within a demographic group would still be ignored. Unfortunately, the considered dataset does not contain any demographic information on the respective users. Thus, demographic filtering beyond suggesting trending or best rated movies is not possible.

3.2. Content-Based Filtering

A more sophisticated approach than demographic filtering is to group movies based on features like genre, budget, cast, director, runtime or average rating. Movies similar to those a given user has liked in the past can then be identified. Such content-based filtering methods include k-NN and plot-based recommenders which we will now consider in more detail.

3.2.1. k-NN Algorithm

First, we develop a k-NN based recommendation system. For a given movie, the idea is to find the k most similar movies ("nearest neighbors") in terms of four features: cast, director, genre and keywords. Except for the director, this information is stored in lists that contain multiple elements (see Figure 3.2) and we first need to convert the data into a useable format by using the unlist function in R.

title	cast	director	genres	keywords
Minions	Sandra Bullock, Jon Hamm , Michael Keaton	Kyle Balda	Family , Animation, Adventure	assistant , aftercreditsstinger , duringcreditsstinger
Wonder Woman	Gal Gadot , Chris Pine , Robin Wright	Patty Jenkins	Action , Adventure, Fantasy	dc comics , hero , greek mythology
Beauty and the Beast	Emma Watson, Dan Stevens, Luke Evans	Bill Condon	Family , Fantasy, Romance	france, magic , castle
Baby Driver	Ansel Elgort, Lily James , Kevin Spacey	Edgar Wright	Action, Crime	robbery, atlanta, music
Big Hero 6	Scott Adsit , Ryan Potter , Daniel Henney	Chris Williams	Adventure, Family , Animation	brother brother relationship, hero , talent
Deadpool	Ryan Reynolds , Morena Baccarin, Ed Skrein	Tim Miller	Action , Adventure, Comedy	anti hero , mercenary , marvel comic

Figure 3.2: Glance at the features of interest in the movie dataset

In the following, we calculate a similarity score between a given movie of interest which we will call our "object movie" and all other movies in the dataset which we denominate as "target movies". For each target movie, the similarity score is calculated as follows:

$$\text{target similarity score} = \sum_{j=1}^{4} \frac{n_j}{a_j * b_j} * w_j$$

where

- j ε [cast, director, genres, keywords] are the four different movie features considered
- a_j is the number of realizations for feature j of the object movie e.g. for j = genre, the realizations for the object "Minions" would be "Family", "Animation" and "Adventure", thus a = 3

- b_j is the number of realizations for feature j of the target movie e.g. for j = director, the realization for the target “Avatar” would be “James Cameron”, thus b = 1
- n_j is the number of elements that the object and the target movie have in common e.g. for j = genre, the shared element of the object “Minions” and the target “Avatar” would be “Adventure” i.e. n = 1
- w_j is the arbitrary weight of feature j

As an example, let us identify the 10 most similar items for “Terminator 3: Rise of the Machines”, the most rated movie in the dataset, by assigning each feature an equal weight of 1. The relevant features for our object movie are depicted in Figure 3.3.

title	genres	keywords	director	cast
Terminator 3: Rise of the Machines	Action , Thriller , Science Fiction	saving the world , artificial intelligence, man vs machine	Jonathan Mostow	Arnold Schwarzenegger, Nick Stahl , Claire Danes

Figure 3.3: Movie features of Terminator 3 (2013)

What becomes immediately apparent when looking at the resulting recommendations, is that all movies in the dataset which were directed by Jonathan Mostow come out at the top of the list (Figure 3.4). This is due to the fact that the director feature consists of a single element, whereas the other features typically consist of three elements. Thus, the magnitude of the similarity score for this component is larger.

title	genres	keywords	director	cast
Surrogates	Action , Science Fiction, Thriller	clone , dystopia	Jonathan Mostow	Bruce Willis , Radha Mitchell, Rosamund Pike
Breakdown	Drama , Action , Thriller	california, bank , ransom	Jonathan Mostow	Kurt Russell , Walsh , Kathleen Quinlan
U-571	Action , Drama , Thriller	submarine , world war ii , north atlantic	Jonathan Mostow	Matthew McConaughey, Bill Paxton , Harvey Keitel
The Matrix	Action , Science Fiction	saving the world , artificial intelligence, man vs machine	Lana Wachowski	Keanu Reeves , Laurence Fishburne, Carrie
Terminator Genisys	Science Fiction, Action , Thriller	saving the world , artificial intelligence, cyborg	Alan Taylor	Arnold Schwarzenegger, Jason Clarke , Emilia Clarke
The Terminator	Action , Thriller , Science Fiction	saving the world , artificial intelligence, rebel	James Cameron	Arnold Schwarzenegger, Michael Biehn , Linda Hamilton
Terminator Salvation	Action , Science Fiction, Thriller	saving the world , artificial intelligence, prophecy	McG	Christian Bale , Sam Worthington, Anton Yelchin
The Matrix Reloaded	Adventure, Action , Thriller	saving the world , artificial intelligence, man vs machine	Lilly Wachowski	Keanu Reeves , Carrie , Laurence Fishburne
The Matrix Revolutions	Adventure, Action , Thriller	saving the world , artificial intelligence, man vs machine	Lilly Wachowski	Keanu Reeves , Laurence Fishburne, Carrie
I, Robot	Action , Science Fiction	suicide , artificial intelligence, man vs machine	Alex Proyas	Will Smith , Bridget Moynahan, Alan Tudyk

Figure 3.4: Movie recommendations for Terminator 3 based on k-NN with equal weights

We therefore adjust the recommendation system by decreasing the weight of the director feature in our calculation to 1/3, while keeping the weights of the other features constant. This way, the calculation of the similarity score is more balanced between the different features. While the new recommendation list still contains several movies by Jonathan Mostow, director is no longer the dominant criterion (Figure 3.5).

title	genres	keywords	director	cast
The Matrix	Action , Science Fiction	saving the world , artificial intelligence, man vs machine	Lana Wachowski	Keanu Reeves , Laurence Fishburne, Carrie
Terminator Genisys	Science Fiction, Action , Thriller	saving the world , artificial intelligence, cyborg	Alan Taylor	Arnold Schwarzenegger, Jason Clarke , Emilia Clarke
The Terminator	Action , Thriller , Science Fiction	saving the world , artificial intelligence, rebel	James Cameron	Arnold Schwarzenegger, Michael Biehn , Linda Hamilton
Surrogates	Action , Science Fiction, Thriller	clone , dystopia	Jonathan Mostow	Bruce Willis , Radha Mitchell, Rosamund Pike
Terminator Salvation	Action , Science Fiction, Thriller	saving the world , artificial intelligence, prophecy	McG	Christian Bale , Sam Worthington, Anton Yelchin
The Matrix Reloaded	Adventure, Action , Thriller	saving the world , artificial intelligence, man vs machine	Lilly Wachowski	Keanu Reeves , Carrie , Laurence Fishburne
The Matrix Revolutions	Adventure, Action , Thriller	saving the world , artificial intelligence, man vs machine	Lilly Wachowski	Keanu Reeves , Laurence Fishburne, Carrie
I, Robot	Action , Science Fiction	suicide , artificial intelligence, man vs machine	Alex Proyas	Will Smith , Bridget Moynahan, Alan Tudyk
Breakdown	Drama , Action , Thriller	california, bank , ransom	Jonathan Mostow	Kurt Russell , Walsh , Kathleen Quinlan
U-571	Action , Drama , Thriller	submarine , world war ii , north atlantic	Jonathan Mostow	Matthew McConaughey, Bill Paxton , Harvey Keitel

Figure 3.5: Movie recommendations for Terminator 3 based on k-NN with adjusted weights

The recommendations obtained under k-NN intuitively make sense. Furthermore, they are similar to the recommendations made by Amazon Prime (Figure 3.6). In order to test how well our recommendation system performs, we also look at the ratings that users who have watched and enjoyed Terminator 3 (i.e. assigned it a rating of at least 3.5) have given to the recommended target movies. The test results confirm our initial assessment, as all recommended films receive good average ratings between 2.9 and 4.2 (Figure 3.7).

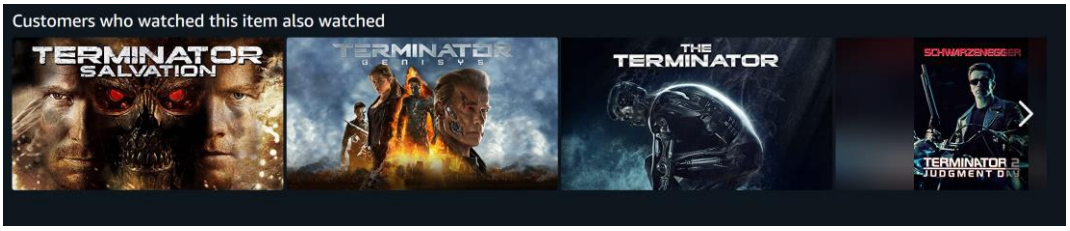


Figure 3.6: Amazon Prime recommendations for viewers of Terminator 3

title	rating
The Terminator	4.20
Terminator Salvation	3.50
The Matrix Revolutions	3.40
I, Robot	3.35
Breakdown	3.25
U-571	2.89

Figure 3.7: Average ratings for recommended movies under k-NN

We conclude that the k-NN algorithm works reasonably well and could certainly be applied to users who have searched for or liked a particular movie. However, a major downside of this algorithm is that it has to be recomputed for every movie and movie recommendations under this setting are thus always based on a single item, meaning that no learning happens. In fact, we are really just screening the dataset for movies similar to the object movie. Moreover, this approach does not account for individual

user preferences but makes the same recommendation to all users and might recommend movies which a user has already watched.

3.2.2. Plot-Based Recommender

This approach is very similar to the k-NN recommendation system. For our plot-description-based recommender, we also calculate pairwise similarity scores for all movies. Those scores are now based on the terms used in the movie descriptions. Term Frequency-Inverse Document Frequency (TF-IDF) vectors - a standard tool from text processing - allow us to convert movie descriptions into a useable format (a matrix of all movies as columns and words used in the description as rows). In her tutorial Ibtesam Ahmed defines TF-IDF as follows:

*"[…] term frequency is the relative frequency of a word in a document given as (term instances/total instances). Inverse Document Frequency is the relative count of documents containing the term given by log(number of documents/documents including the term). The overall importance of each word to the documents in which they appear is equal to TF * IDF."*

The similarity scores for any pair of movies are then calculated by applying cosine similarity:

$$similarity = \cos(\theta) = \frac{A * B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

where:

- A_i and B_i are components of the TF-IDF vectors A and B
- similarity ε [0,1]

Once again, we identify the 10 most similar movies for "Terminator 3: Rise of the Machines". As one can see, the suggestions are quite similar to those obtained under k-NN and include several movies from the Terminator series (Figure 3.8).

title	genres	director	cast
Terminator 2: Judgment Day	Action , Thriller , Science Fiction	James Cameron	Arnold Schwarzenegger, Linda Hamilton , Robert Patrick
Terminator Salvation	Action , Science Fiction, Thriller	McG	Christian Bale , Sam Worthington, Anton Yelchin
Highlander	Adventure, Action , Fantasy	Russell Mulcahy	Christopher Lambert, Roxanne Hart , Clancy Brown
Highlander: Endgame	Action , Fantasy , Science Fiction	Douglas Aarniokoski	Christopher Lambert, Bruce Payne , Adrian Paul
Terminator Genisys	Science Fiction, Action , Thriller	Alan Taylor	Arnold Schwarzenegger, Jason Clarke , Emilia Clarke
The Terminator	Action , Thriller , Science Fiction	James Cameron	Arnold Schwarzenegger, Michael Biehn , Linda Hamilton
The Boondock Saints II: All Saints Day	Action , Thriller, Crime	Troy Duffy	Sean Patrick Flanery, Norman Reedus , Billy Connolly
Premam	Comedy , Music , Romance	Alphonse Putharen	Nivin Pauly , Sai Pallavi , Madonna Sebastian
It's Kind of a Funny Story	Comedy, Drama	Ryan Fleck	Keir Gilchrist , Emma Roberts , Zach Galifianakis
Hitman: Agent 47	Action , Crime , Thriller	Aleksander Bach	Rupert Friend , Zachary Quinto, Hannah Ware

Figure 3.8: Movie recommendations by the plot-based recommender

Let us have a glance at the movie plots of the top 3 recommendations (which unsurprisingly are all from the Terminator series) and compare them to the description of Terminator 3 (Figure 3.9). Apart from the title “Terminator” and the main characters John and Sarah Connor, the movie plots share several characteristic terms like “Skynet” and “back” (as part of Arnold Schwarzenegger’s legendary “I’ll be back” catchphrase).

title	overview
Terminator 3: Rise of the Machines	It's been 10 years since John Connor saved Earth from Judgment Day, and he's now living under the radar, steering clear of using anything Skynet can trace. That is, until he encounters T-X, a robotic assassin ordered to finish what T-1000 started. Good thing Connor's former nemesis, the Terminator, is back to aid the now-adult Connor … just like he promised.
Terminator Genisys	The year is 2029. John Connor, leader of the resistance continues the war against the machines. At the Los Angeles offensive, John's fears of the unknown future begin to emerge when TECOM spies reveal a new plot by SkyNet that will attack him from both fronts; past and future, and will ultimately change warfare forever.
Terminator 2: Judgment Day	Nearly 10 years have passed since Sarah Connor was targeted for termination by a cyborg from the future. Now her son, John, the future leader of the resistance, is the target for a newer, more deadly terminator. Once again, the resistance has managed to send a protector back to attempt to save John and his mother Sarah.
The Terminator	In the post-apocalyptic future, reigning tyrannical supercomputers teleport a cyborg assassin known as the ""Terminator"" back to 1984 to kill Sarah Connor, whose unborn son is destined to lead insurgents against 21st century mechanical hegemony. Meanwhile, the human-resistance movement dispatches a lone warrior to safeguard Sarah. Can he stop the virtually indestructible killing machine?

Figure 3.9: Comparison of Terminator movie plots

Looking at the average rating that users who have enjoyed Terminator 3 have given to the recommended movies, we find that the reviews are very positive (Figure 3.10). However, ratings from users who have liked Terminator 3 are available only for 3 out of the 10 suggested movies.

title	rating
Terminator 2: Judgment Day	4.20
The Terminator	4.03
Terminator Salvation	3.40

Figure 3.10: Average ratings for recommended movies under k-NN

Thus, like the k-NN algorithm, the plot-based strategy seems to lead to good results. In fact, the outputs are almost identical. But the approach is also subject to the same disadvantages, namely that recommendations have to be computed separately for each movie and that no learning happens. Instead, this approach is basically screening the dataset for similar movies, thereby completely ignoring individual user preferences with respect to genres, actors etc. and making the same recommendations to all users.

3.3. Collaborative Filtering

In order to overcome those issues, we now implement collaborative filtering techniques which can be divided into user- and item-based filtering methods. The former method groups similar users with the help of Pearson correlation or cosine similarity and proposes movies based on what other users in the same category have liked. In contrast, item-based filtering methods take positive reviews by a given user and try to find similar movies. Like the user-based approach, item-based filtering uses Pearson

correlation or cosine similarity to derive the similarity between two items. Both techniques require detailed information on the user for whom the recommendation is made and cannot be applied to new users. In this paper we will focus on item-based collaborative filtering.

In a first step, we construct a matrix of all users (rows) and movies (columns) in the dataset which we fill with the ratings assigned by the users to the respective movies were possible. To the remaining movie-user combinations, (i.e. the movies that have not been reviewed by an individual user yet) we assign the average movie rating. Using a latent factor model via the built-in svd function in R allows us to address sparsity issues (i.e. the occurrence of extreme values due to a small number of observations) as well as scalability issues (i.e. high computation times in large datasets) by decreasing the number of dimensions. By applying a latent feature model, we map users and movies into a latent feature space. In this feature space, we only use the first three dimensions, as they typically contain almost all of the relevant information. We now have an optimization problem where the goal is to minimize RMSE, the average prediction error, in terms of the rating which a user will assign to a given movie.

We apply the recommendation system to the most active user in the dataset with ID 564 and get the following recommendations:

title	genres	year	director	cast
Spider-Man 3	Fantasy , Action , Adventure	2007	Sam Raimi	Tobey Maguire, Kirsten Dunst, James Franco
A Streetcar Named Desire	Drama	1951	Elia Kazan	Vivien Leigh , Marlon Brando, Kim Hunter
The Evil Dead	Horror	1981	Sam Raimi	Bruce Campbell , Ellen Sandweiss , Richard DeManincor
JFK	Drama , Thriller, History	1991	Oliver Stone	Kevin Costner , Tommy Lee Jones, Gary Oldman
Strangers on a Train	Crime , Drama , Thriller	1951	Alfred Hitchcock	Farley Granger, Robert Walker , Ruth Roman
Singin' in the Rain	Comedy , Music , Romance	1952	Stanley Donen	Gene Kelly , Debbie Reynolds, Jean Hagen
Once Upon a Time in Mexico	Action	2003	Robert Rodriguez	Antonio Banderas, Salma Hayek , Johnny Depp
You, Me and Dupree	Comedy , Romance	2006	Anthony Russo	Kate Hudson, Owen Wilson, Matt Dillon
Ninotchka	Comedy , Romance	1939	Ernst Lubitsch	Greta Garbo , Melvyn Douglas, Ina Claire
Nick of Time	Crime , Drama , Thriller	1995	John Badham	Johnny Depp , Courtney Chase, Charles S

Figure 3.11: Recommendations based on collaborative filtering

Based on the recommendations, it seems that the user has a preference for old movies and likes movies from various genres.

Different than the content-based filtering approaches in the previous sections, this recommendation system is learning constantly and recommendations should improve over time. However, this also means that the system might perform poorly for relatively

new users. Moreover, the system is not fully transparent in how it makes recommendations.

3.4 Decision Tree Algorithm

Decision tree algorithms repeatedly split the feature space (in our case the set of all movies in the dataset) into two different spaces. The splitting criterion at each node is chosen in such a way that the overall RSS loss function is minimized.

We build a decision tree for the most active user with ID 564 (Figure 3.12). In order to construct the tree, we use all 463 observed ratings by this particular user on movies he/she has already watched and use all numeric movie features in the dataset as potential decision criteria. For each resulting subset of similar movies, the algorithm thus predicts the mean rating which the user will assign to movies from this group.

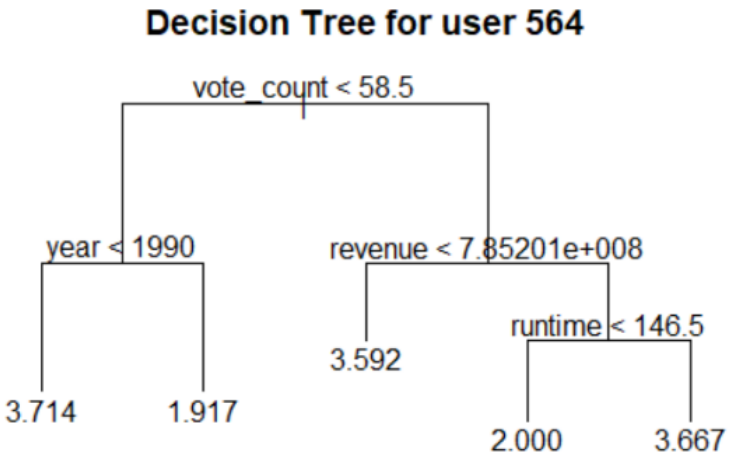


Figure 3.12: Decision Tree for user 564

Overall, four criteria are used to form the tree (vote count, year, revenue and runtime). Interestingly, popularity and weighted rating are not used as factors, indicating that public opinion does not have a strong impact on what this user likes. The decision tree shows that, on average, the user assigns the highest rating to older movies that were released before 1990 and have received less than 59 votes. We use the decision tree to predict the rating the user would give to those movies in the dataset he/she has not watched yet and recommend the movies with the highest predicted rating. One might also say that we search for movies with the two characteristics described above which the user seems to like the best. From the total set of 4497 unwatched movies we obtain a list of 255 movies for which we predict a rating of 3.714 (Figure 3.13).

```
                        title
  1:             Desert Hearts
  2:           Shaft in Africa
  3:       Farewell, My Lovely
  4: The Legend of Boggy Creek
  5:                    Exodus
 ---
251:                   Miranda
252:        The Gorgeous Hussy
253:       Across to Singapore
254:        Lights of New York
255:                Bizim Aile
```

Figure 3.13: Decision Tree Recommendation

But how accurate are those recommendations? In order to answer this question, we randomly split the observed ratings by the selected user into a training (70%) and a testing (30%) dataset and recompute our tree. Note that the resulting decision tree differs dramatically from the previous version (Figure 3.14). One of the major downsides of using decision trees is that they are extremely sensitive to changes in the training data and may look entirely different even for only slightly modified training datasets. Thus, we cannot make inferences as to whether our original tree makes good suggestions.

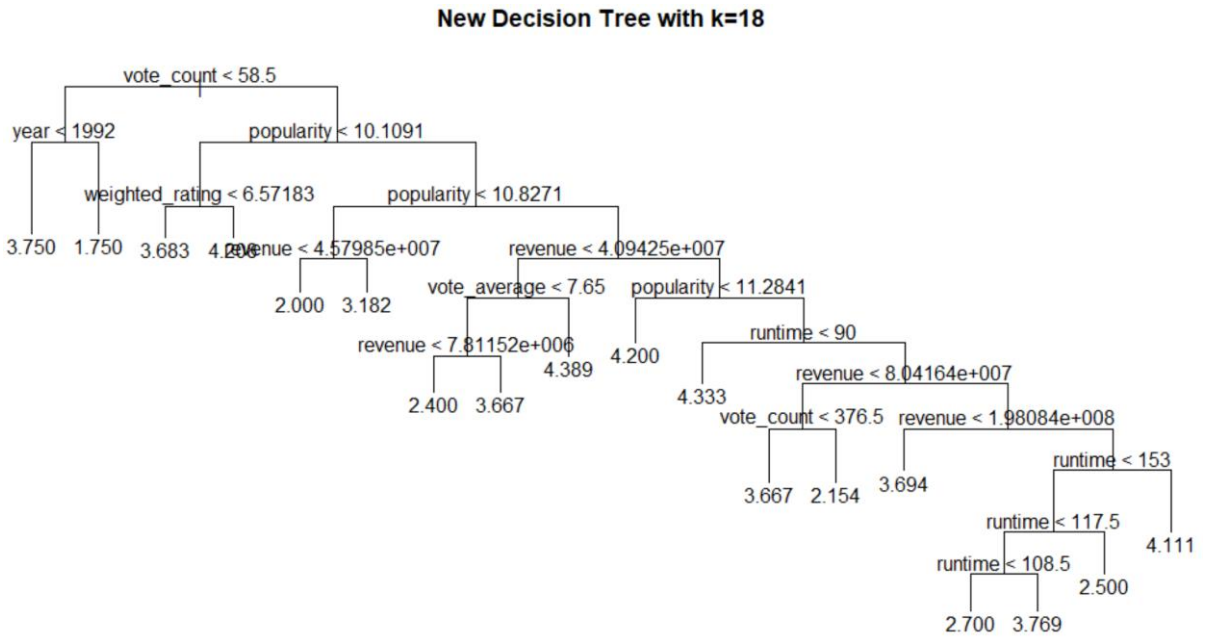


Figure 3.14: New decision tree w/o pruning

However, we can investigate whether decision trees in general are a good tool for making recommendations. The root-mean-square error (RMSE) for the testing data, i.e. the average absolute prediction error based on the new decision tree, amounts to 1.41. Given a standard error of the ratings in the test dataset of only 1.14, the model performs poorly and one could more accurately predict the ratings in the test dataset by simply always guessing the mean rating the user has assigned to movies from the

training dataset. With 18 terminal nodes and 7 used features, the new tree seems to be overfitting the training dataset. Therefore, we prune the tree to find the number of terminal nodes which minimizes the root-mean-squared error for the testing data. We find that a decision tree with three terminal nodes is optimal (Figure 3.15). However, with an RMSE of 1.13, the strategy performs only slightly better than the strategy of always guessing the mean rating observed on the training dataset.

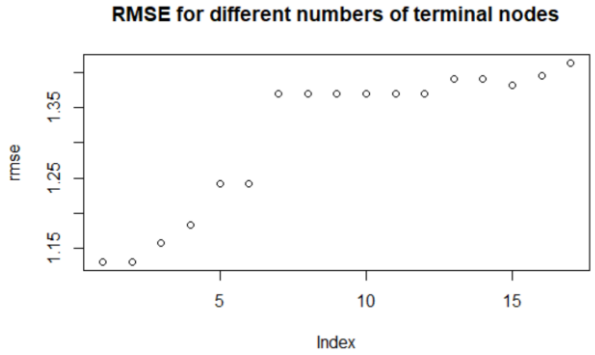


Figure 3.15: RMSE for different numbers of terminal nodes

We conclude that decision trees are very intuitive and easy to compute. Also, they take into account user preferences with respect to movie characteristics. However, they appear to be overly simplistic and lead to inaccurate predictions. Moreover, decision trees may overfit the training data and are extremely sensitive to changes in the training dataset.

3.5 Logistic Regression

Another approach for making movie recommendations is to use a logistic regression and predict the likelihood with which a given user will like a movie based on its features. Once again, we consider the most active user with ID 564 and train our regression model using all 463 reviews by this user. Since logistic regressions can only work with binary dependent variables, we introduce a like variable which equals 1, for perfect ratings (i.e. 5 out of 5) and 0 otherwise. As our independent variables we use the numeric movie features: budget, popularity, revenue, runtime, vote average, vote count and year. The regression summary statistics show that only revenue and vote count are statistically significant predictors of whether a user will like a movie at the 10% significance level (Figure 3.16). Interestingly, revenue has a negative coefficient, indicating that the selected user does not have a preference for blockbusters. Moreover, the negative coefficient for “year” – even though it is not statistically significant - confirms our finding from the decision tree recommendation system that the user seems to like older movies. In order to eliminate the effect of multicollinearity,

we also perform a second regression where we drop budget and vote count which - as we have shown above - are highly correlated with revenue. We observe that the statistical significance of the revenue variable disappears. However, the negative coefficients for revenue and year persist.

```
glm(formula = likes ~ budget + popularity + revenue + runtime +
    vote_average + vote_count + year, data = user_rating)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.4071  -0.2045  -0.1719  -0.1283   0.9031

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.517e+00  2.757e+00   1.638   0.1020
budget        7.685e-10  6.409e-10   1.199   0.2311
popularity   -2.716e-03  2.404e-03  -1.130   0.2591
revenue      -3.693e-10  1.569e-10  -2.354   0.0190 *
runtime       3.649e-04  9.785e-04   0.373   0.7094
vote_average -3.897e-03  3.272e-02  -0.119   0.9052
vote_count    4.351e-05  2.249e-05   1.935   0.0536 .
year         -2.177e-03  1.345e-03  -1.618   0.1064
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.1549074)

    Null deviance: 71.892  on 462  degrees of freedom
Residual deviance: 70.483  on 455  degrees of freedom
AIC: 460.41

Number of Fisher Scoring iterations: 2
```

```
glm(formula = likes ~ popularity + revenue + runtime + vote_average +
    year, data = user_rating)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.3243  -0.1960  -0.1808  -0.1491   0.8743

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.866e+00  2.678e+00   1.070    0.285
popularity    3.572e-04  1.898e-03   0.188    0.851
revenue      -8.464e-11  1.006e-10  -0.841    0.401
runtime       5.284e-04  9.471e-04   0.558    0.577
vote_average  7.183e-03  2.739e-02   0.262    0.793
year         -1.394e-03  1.308e-03  -1.066    0.287

(Dispersion parameter for gaussian family taken to be 0.1561368)

    Null deviance: 71.892  on 462  degrees of freedom
Residual deviance: 71.355  on 457  degrees of freedom
AIC: 462.1

Number of Fisher Scoring iterations: 2
```

Figure 3.16: Logistic regression summary statistics for all numeric features (left) and when accounting for multicollinearity (right)

Using the coefficients obtained in the regressions above, we predict the “like” probabilities for all unwatched movies in the dataset. Our recommendation system makes movie suggestions by identifying the movies with the highest like probabilities. The recommendations based on the two regressions differ substantially. Only “The birth of a nation” from 1915 is proposed under both models. Notably, the second model only proposes older movies from before 1962 and the predicted probabilities of a perfect rating are much lower than for the first model (Figures 3.17, 3.18).

title	likes	genres	year	director	cast
Inception	0.5151	Action , Thriller , Science Fiction	2010	Christopher Nolan	Leonardo DiCaprio, Joseph Gordon , Ellen Page
Django Unchained	0.4807	Drama , Western	2012	Quentin Tarantino	Jamie Foxx , Christoph Waltz , Leonardo DiCaprio
Mad Max: Fury Road	0.4602	Action , Adventure , Science Fiction	2015	George Miller	Tom Hardy , Charlize Theron, Nicholas Hoult
Interstellar	0.4393	Adventure , Drama , Science Fiction	2014	Christopher Nolan	Matthew McConaughey, Jessica Chastain , Anne Hathaway
Batman Begins	0.4004	Action, Crime , Drama	2005	Christopher Nolan	Christian Bale, Michael Caine , Liam Neeson
The Lone Ranger	0.3972	Action , Adventure, Western	2013	Gore Verbinski	Johnny Depp , Armie Hammer , William Fichtner
The Birth of a Nation	0.3804	Drama , History, War	1915	Griffith	Lillian Gish, Mae Marsh , Henry B
X-Men: First Class	0.3784	Action , Science Fiction, Adventure	2011	Matthew Vaughn	James McAvoy , Michael Fassbender, Jennifer Lawrence
The Matrix	0.3656	Action , Science Fiction	1999	Lana Wachowski	Keanu Reeves , Laurence Fishburne, Carrie
Now You See Me	0.3591	Thriller, Crime	2013	Louis Leterrier	Jesse Eisenberg, Mark Ruffalo , Woody Harrelson

Figure 3.17: Movie recommendation for a logistic regression with all numeric features

title	likes	genres	year	director	cast
The Birth of a Nation	0.3464	Drama , History, War	1915	Griffith	Lillian Gish, Mae Marsh , Henry B
The Thief of Bagdad	0.3182	Action , Adventure, Drama	1924	Raoul Walsh	Douglas Fairbanks, Snitz Edwards , Charles Belcher
Seven Samurai	0.3168	Action, Drama	1954	Akira Kurosawa	Toshir , Takashi Shimura, Yoshio Inaba
Gone with the Wind	0.3164	Drama , Romance, War	1939	Victor Fleming	Vivien Leigh , Clark Gable , Olivia de Havilland
Ben-Hur: A Tale of the Christ	0.3117	Adventure, Drama	1925	Charles Brabin	Ramon Novarro, Francis X , May McAvoy
Cleopatra	0.3069	Drama , History, Romance	1963	Joseph L	Elizabeth Taylor, Richard Burton , Rex Harrison
The Ten Commandments	0.3042	Drama , History	1956	Cecil B	Charlton Heston, Yul Brynner , Anne Baxter
Giant	0.3014	Action , Drama , Western	1956	George Stevens	Elizabeth Taylor, Rock Hudson , James Dean
Lawrence of Arabia	0.3002	Adventure, Drama , History	1962	David Lean	Alec Guinness, Anthony Quinn, Jack Hawkins
The Best Years of Our Lives	0.3000	Drama , History, Romance	1946	William Wyler	Fredric March, Myrna Loy , Dana Andrews

Figure 3.18: Movie recommendation for a logistic regression when accounting for multicollinearity

In order to test the predictive power of our logistic regression model, we again randomly split the reviews by the most active user into a training (70%) and a testing dataset (30%). When looking at the coefficients and their statistical significance, one realizes that the regression model obtained for the training data subset is fairly similar to the regression model before which was based on the whole dataset for user 564 (Figure 3.19). Thus, unlike for the decision tree algorithm, we can draw conclusions with respect to the accuracy of our original model.

```
glm(formula = likes ~ budget + popularity + revenue + runtime +
    vote_average + vote_count + year, data = Data.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.3963  -0.2110  -0.1626  -0.1042   0.9091

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.849e+00  3.343e+00   1.749   0.0812 .
budget        6.902e-10  7.691e-10   0.897   0.3702
popularity   -2.721e-03  2.581e-03  -1.054   0.2925
revenue      -4.480e-10  1.971e-10  -2.273   0.0237 *
runtime       1.405e-03  1.222e-03   1.150   0.2511
vote_average -2.270e-02  3.880e-02  -0.585   0.5589
vote_count    5.118e-05  2.625e-05   1.950   0.0520 .
year         -2.841e-03  1.636e-03  -1.736   0.0835 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.1532564)

    Null deviance: 50.172  on 324  degrees of freedom
Residual deviance: 48.582  on 317  degrees of freedom
AIC: 322.63

Number of Fisher Scoring iterations: 2
```

```
glm(formula = likes ~ popularity + revenue + runtime + vote_average +
    year, data = Data.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.3895  -0.1987  -0.1722  -0.1285   0.9036

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.128e+00  3.252e+00   1.269    0.205
popularity    6.369e-04  1.977e-03   0.322    0.748
revenue      -1.199e-10  1.248e-10  -0.961    0.337
runtime       1.591e-03  1.170e-03   1.359    0.175
vote_average -5.273e-03  3.265e-02  -0.162    0.872
year         -2.046e-03  1.591e-03  -1.287    0.199

(Dispersion parameter for gaussian family taken to be 0.1546444)

    Null deviance: 50.172  on 324  degrees of freedom
Residual deviance: 49.332  on 319  degrees of freedom
AIC: 323.6

Number of Fisher Scoring iterations: 2
```

Figure 3.19: Summary statistics of the logistic regressions based on the training data for all numeric features (left) and when accounting for multicollinearity (right)

We predict the probability of a perfect rating for each movie in the testing dataset based on the two regression models. To compute the error rate of our regression model, we need to convert those probabilities into a binary like variable. We choose 16.5%, the fraction of five-star ratings overall, as our decision criterion. Whenever the predicted probability exceeds this threshold, we assume that the user will give a movie a perfect score. The error rate is then calculated as the percentage of false predictions. For the first regression model which includes all numeric features, we obtain a high error rate of 56.5%. The second model performs even worse and produces an error rate of

61.6%. However, note that the error rate drops substantially when we increase the decision criterion and converges to 21.7% (i.e. the fraction of perfect ratings in the testing dataset), as for high values of decCrit we always predict a score worse than 5 (Figure 3.20).

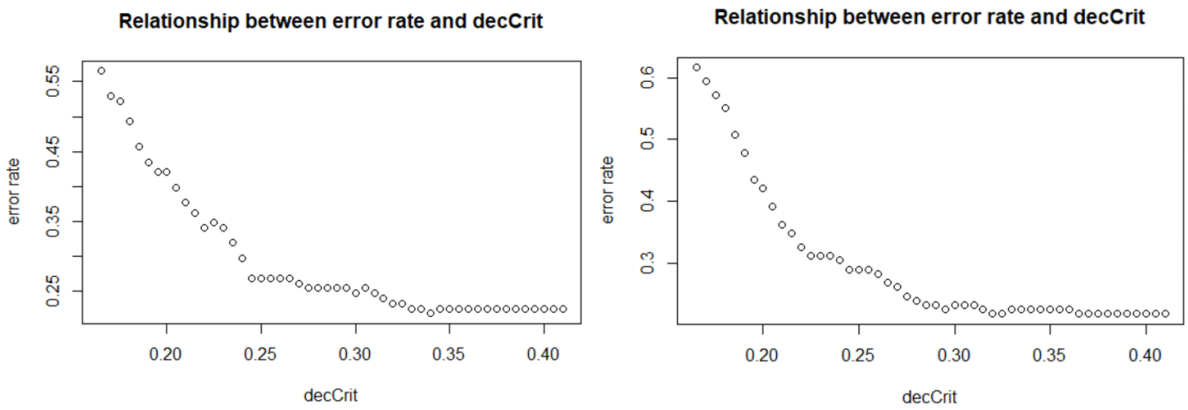


Figure 3.20: Relationship between the error rate and the decision criterion

Contrary to the k-NN and plot-based recommendation systems, the logistic regression model is constantly learning and should improve over time. It cannot be applied to new users as it requires at least one observation for training purposes. The approach is also highly susceptible to outliers when there are only a few observations. Moreover, for the observed user, we find that the model performs poorly even when a large number of observations is available. Thus, the logistic regression specified above might not be an adequate tool for making movie recommendations. It is questionable whether features like runtime, revenue or release year are good indicators for whether a user will like a movie or not. Perhaps the regression model could be improved by including other independent dummy variables for genres, directors, main actors etc.

4. Conclusion

The recommendation systems we have discussed in this paper are quite complementary. Demographic and content-based filtering techniques are suitable tools for making recommendations to users for whom no or very little information is available, as they do not tailor recommendations to individual user preferences and therefore do not require any user data for training purposes.

Hence, if a new user registers on a streaming platform like Netflix or Amazon Prime, a good start is to recommend the most popular or best-rated movies in the dataset. An even better approach – and in fact standard practice e.g. on Netflix (Figure 4.1) – is to ask users for their favorite movies when they access the platform for the first time. This

makes more sophisticated recommendations based on k-NN algorithms and plot-based filtering feasible. Our analysis has shown that both models lead to good results. Thus, it makes sense to apply both algorithms to make initial recommendations to new users.

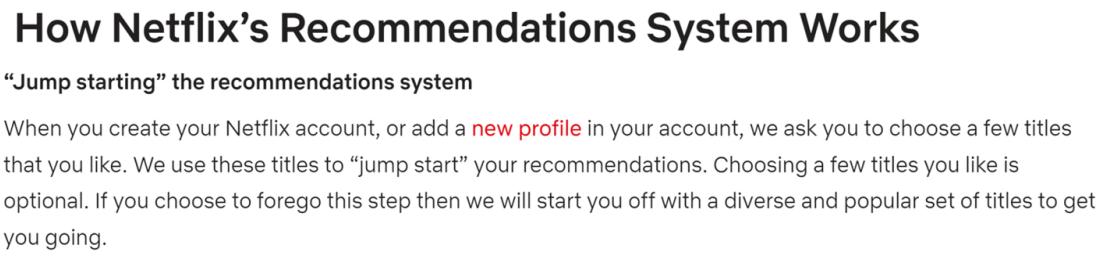


Figure 4.1: Initial recommendations to new users on Netflix[4]

Alternatively, demographic information collected upon a user's registration could be used to design a demographic filtering algorithm and make proposals to new users. As we have pointed out earlier, the dataset we considered does not contain any information on the users other than their ID. Hence, demographic filtering is not feasible for this dataset. In contrast, big streaming platforms ask new users to provide various information such as their address, payment information and the preferred plan after their free trial. Thus, they possess large amounts of user-specific data even before users watch the first movie on their platform. For instance, a user's address could be used to derive their income and age. This would theoretically allow them to assign users to certain demographic groups. However, Netflix claims not to use general demographic information like age or gender in its recommendation system.

As soon as a user starts to watch movies on the platform, similar movies can be identified via k-NN or plot-based recommenders and proposed to the user. However, content-based filtering should only be one element of a recommendation system for known users. Ultimately the goal is to customize recommendations to the individual taste of a user and to constantly learn from the user's watching behavior, rather than to simply look for movies with similar characteristics.

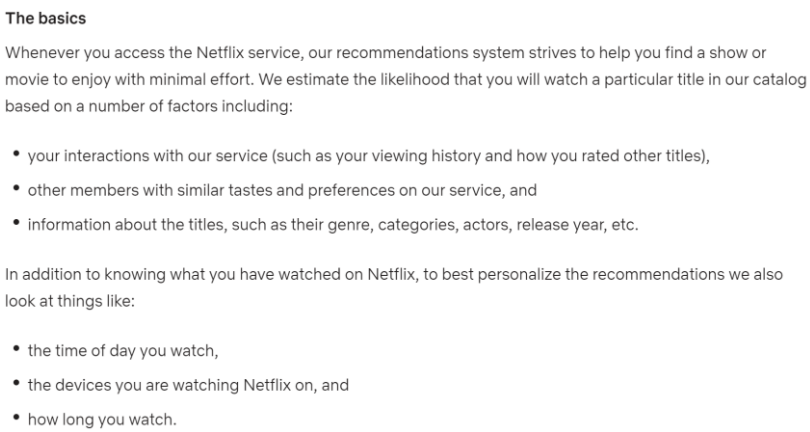
We have presented three such machine learning models: collaborative filtering methods, decision tree and logistic regression-based recommendation systems. Both the decision tree and the logistic regression model have worked poorly for the user we have studied. This could have three reasons:

[4] Source: https://help.netflix.com/en/node/100639

- The movie features we used to build the tree and train the regression model respectively are inadequate predictors for movie recommendation purposes
- More explanatory variables would be required in our models
- The models themselves are not suited well to make recommendations.

Our work could therefore be extended by analyzing whether decision trees and logistic regressions are adequate tools for making movie recommendations and which movie features should be included in the models to make the best recommendations.

While we have solely focused on movie features in training our recommendation engines, large streaming platforms like Disney Plus or Netflix can also rely on what they have learned about the users themselves to personalize their recommendations:



The basics

Whenever you access the Netflix service, our recommendations system strives to help you find a show or movie to enjoy with minimal effort. We estimate the likelihood that you will watch a particular title in our catalog based on a number of factors including:

- your interactions with our service (such as your viewing history and how you rated other titles),
- other members with similar tastes and preferences on our service, and
- information about the titles, such as their genre, categories, actors, release year, etc.

In addition to knowing what you have watched on Netflix, to best personalize the recommendations we also look at things like:

- the time of day you watch,
- the devices you are watching Netflix on, and
- how long you watch.

Figure 4.2: Overview of the principles of Netflix's recommendation engine

Certainly, datasets that contain more detailed information on the users allow for even better recommendations. Future research could therefore use datasets with more variables such as the "Stanford AI Lab Large Movie Review Dataset".

Another interesting addition to our work would be to make the k-NN algorithm learn the optimal weights of the movie features for a given user by minimizing an error function. This way, instead of using arbitrary weights and making the same recommendation to all users, k-NN would personalize recommendations.

5. List of References

- Bawden, D., Robinson, I. (2009). The dark side of information: overload, anxiety and other paradoxes and pathologies. Journal of Information Science, 35(2), 180-191.
- Bridle, P. (2010). Information contextualizer: making sense of the information overload. Development and Learning in Organizations, 24, 4-5.
- Cui, B.B. (2017). Design and Implementation of Movie Recommendation System Based on Knn Collaborative Filtering Algorithm. ITM Web of Conferences, 12, 1-5.
- Li, Ch.-Y. (2017). Why do online consumers experience information overload? An extension of communication theory. Journal of Information Science, 43(6), 835-851.
- Reddy, S., Nalluri, S, Kunisetti, S. Ashok, S., Venkatesh, B. (2019). Content-Based Movie Recommendation System Using Genre Correlation. Smart Intelligent Computing and Applications: Smart Innovations, Systems and Technologies, 105, 391-397.
- Volokhin, S., Agichtein, E. (2018). Towards Intent-Aware Contextual Music Recommendation: Initial Experiments. The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 1045-1048.
- Wang, X., Luo, F. Sang, Ch., Zeng, J., Hirokawa, S. (2017). Personalized Movie Recommendation System Based on Support Vector Machine and Improved Particle Swarm Optimization. Institute of Electronics, Information and Communication Engineers: Transactions on Information and Systems, 100(2), 285-293.
- Zhou, Y., Wilkinson, D., Schreiber, R., Pan, R. (2008). Large-Scale Parallel Collaborative Filtering for the Netflix Prize. In: Fleischer, R., Xu, J. (eds), Algorithmic Aspects in Information and Management (337-348). Heidelberg: Springer-Verlag.