

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

Yifan Zhu | yifan.zhu@student.unisg.ch | 19-600-105

Jan Scheidegger | jan.scheidegger@student.unisg.ch | 15-055-411

Daniel Rüger | daniel.rueger@student.unisg.ch | 09-593-377

submitted on 14 May 2020

Prof. Dr. Anna-Lena Horlemann

**Abstract**

Today, platforms such as Netflix or Amazon offer tens of thousands of films which can be streamed online. With this huge amount, the user faces an information overload problem which makes the choice of a movie best-suited to their interests and needs time-consuming and complicated. In order to increase convenience and the quality of movie selection, we present three approaches for a moving recommendation system using machine learning algorithms. [*To be improved:* The implemented demographic filtering technique allows to suggest movies for unknown users, while the two content-based techniques we implement make use of the past consumer behavior. With a collaborative-filtering method, we exploit comparisons to other consumers and finally,..]

**List of figures**

**Table of contents**

[1 Introduction 3](#_Toc42374183)

[2 Data 5](#_Toc42374184)

[3 Application 5](#_Toc42374185)

[4 Conclusion 14](#_Toc42374186)

[5 List of references 14](#_Toc42374187)

[6 Appendix 15](#_Toc42374188)

# Introduction

The data available on the internet is constantly and quickly increasing (Bridle, 2010, p. 4). Consumers who inform themselves using the internet are confronted with a large amount of information which can easily become overstraining (Bawden & Robinson, 2009, p. 183). Information overload can result (Cui, 2017, p. 1). Consumers need to invest considerable time and resources in order to analyse information, and choosing a product can become complicated (Chan as cited in Li, 2017, p. 1; Keselmann, Rosemblat, & Kiligoclu as cited in Li, 2017, p. 1). A field which is affected by this evolution is movie selection (Cui, 2017, p. 1). Platforms such as Netflix or Amazon offer a huge number of films which customers can stream online (Peng, Liangshan, & Xiuran, 2013, p. 1). Thus, for consumers, it is difficult to identify the product best-suited to their needs.

Information overload can be reduced through recommendation systems, which screen information. These algorithms allow users to individualise recommendations with respect to consumer characteristics such as personal taste and needs. Recommendation systems have been successfully implemented in order to improve recommendation results, such as in the field of movies (Wei, Zheng, Chen, & Chen, 2016), music (Mao, Chen, Hu, and Zhang, 2016), and news (Wang & Shang, 2015). Netflix, for example, considers the films a client has watched and suggests further movies which are comparable to them (Reddy, Nalluri, Kunisetti, Ashok, & Venaktesh, 2019, p. 392).

Various methods can be used to implement a recommendation system. Three important categories are content-based, collaborative, and hybrid algorithms. Content-based systems analyse the user’s past consumption behaviour and suggest movies based on this. Collaborative filtering examines the past ratings and experiences of consumers and compares this with other clients’ ratings and experiences. Based on the choices of the users being the most comparable, suggestions are generated. In order to avoid the drawbacks of both methods, hybrid models have been proposed. (Reddy et al., 2019, p. 392)

The aim of this paper is to develop a movie recommendation system using machine learning algorithms. Our product will help consumers save time and resources when choosing a movie, make the process more convenient, and help consumers find films better-suited to their tastes and needs.

We propose three different approaches for overcoming the disadvantages of each of the above methods. We first implemented a demographic filtering algorithm, then a content-based algorithm, and finally several collaborative filtering algorithms. The demographic filtering algorithm allows to suggest films for persons with unknown characteristics. The content-based approach uses a k-NN algorithm and a plot-based recommender. Both use the consumer history of the user. In order to compare the experiences of different consumers, we used a collaborative filtering algorithm. Finally, a decision tree algorithm is used because… and why the last algorithm.,,

The subsequent parts of this paper are structured as follows. First, we describe the dataset in more detail. In the application section that follows, we describe the algorithms we used and their actual implementation. Finally, we end with the results and conclusion.

Due to the limited scope of this article, we do not test all the predictions made by our recommendation engine (Daniel: Is that correct?). Again due to the limited scope, we do not determine how the three approaches can be used together in a single recommendation. Instead, the user should assess the three individual algorithms on their own.

Many papers which present movie recommendation systems using machine learning methods have been published in recent years. A k-NN collaborative filtering algorithm, for instance, was used by Cui (2017). Reddy et al. (2019) applied a content-based method which used genre correlations. Wang, Sang, Zeng, and Hirokawa (2017) implemented a support vector machine method and improved particle swarm optimisation.

# Data

The data set used for our movie recommendation analysis was retrieved from the online community Kaggle[[1]](#footnote-1). It contains many variables such as revenues and ratings. This allows a large variety of machine learning algorithms to be implemented. Specifically, the details, credit and keywords of the movies in the data set were obtained from the TMDB Open API and the user ratings were scrapped from the MovieLens website.

The original full dataset contains metadata of 45’000 movies and 26 million ratings from 27’000 different users. Since we did not need such a large amount of data for our algorithms, we only used a subset and constructed two files in the data cleaning process.

The file movies\_data.R contains 4’960 observations of movies and 20 feature variables. 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. Additionally, we removed entries with NAN values to get a balanced data frame.

# Application

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 sharing certain attributes. Afterwards, we consider content-based filtering techniques which try to find movies similar to those a given user has enjoyed and propose them to the user. Lastly, we implement collaborative filtering methods. Rather than identifying similar films, the goal here is to group users based on their interests.

## 3.1. Demographic Filtering

A simple approach for making recommendations is to suggest the best rated or most popular (“trending”) movies in the dataset. 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 e.g. adult films may be suggested to children or French movies to English speakers.

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Figure 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 data set 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 and recommended to the user. Such content-based filtering methods include K-NN and plot-based recommenders which we will now consider in more detail.

### 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 and we first need to convert the data into a useable format by using the unlist function in R. In the following, we call the movie of interest “object movie” and denominate all other movies in the dataset as “target movies”. For each target movie, the similarity score is calculated as follows:

where

* j ε [cast, director, genres, keywords] are the four different movie features considered
* the object elements are the realizations for feature j of the object movie e.g. for j = genre, the elements of the object “Interstellar” would be “Adventure”, “Drama” and “Science Fiction”
* the target elements are the realizations for feature j of the target movie e.g. for j = genre, the elements of the target “Avatar” would be “Action”, “Adventure” and “Fantasy”
* shared elements are the elements that the object and target movie have in common e.g. for j = genre, the shared element of the object “Interstellar” and the target “Avatar” would be “Adventure”
* 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 data set, by assigning each feature an equal weight of 1. What becomes 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. 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.

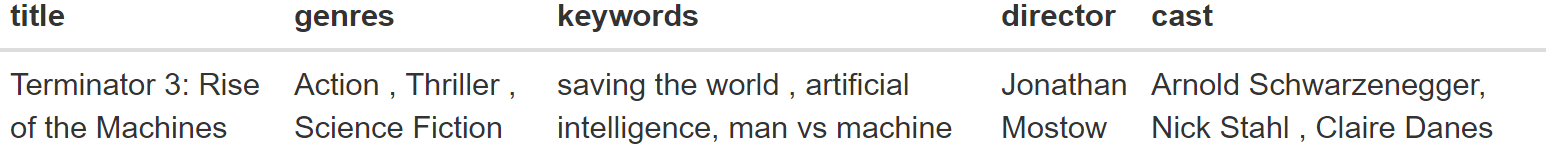


Figure 2: Movie features of Terminator 3 (2003)

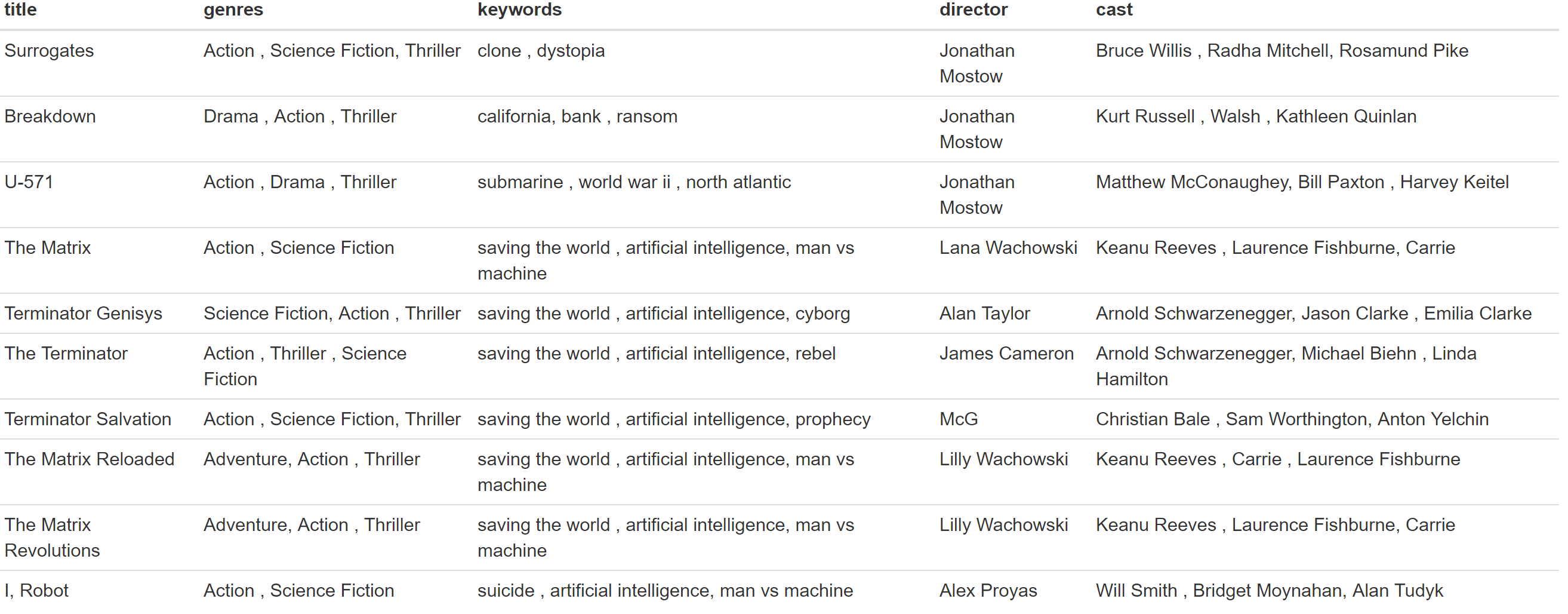


Figure 3: Movie recommendations for viewers of Terminator 3 based on k-NN with equal weights

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

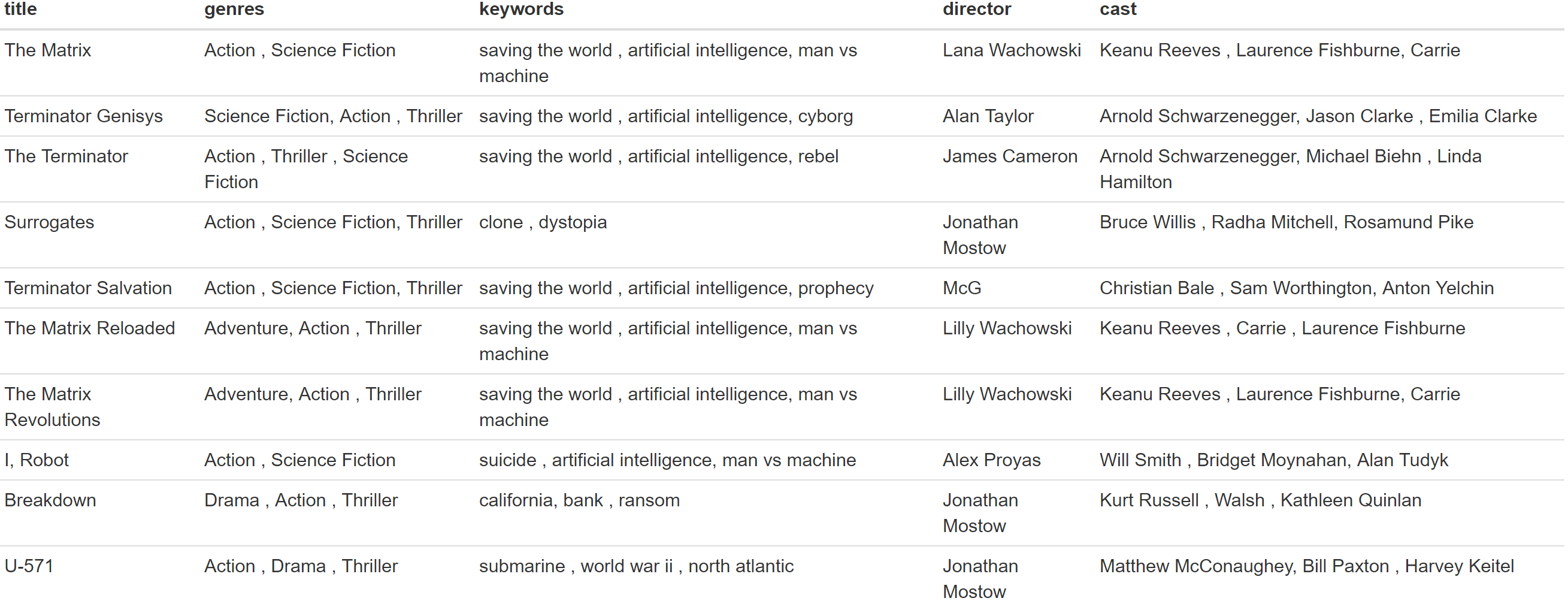


Figure 4: Movie recommendations for viewers of Interstellar based on k-NN with adjusted weights

The recommendations obtained under k-NN intuitively make sense. Furthermore, they are similar to the recommendations made by Amazon Prime. In order to test how well our recommendation system performs, we 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.

Ein Bild, das Foto, Ansicht, Mann, sitzend enthält.

Automatisch generierte Beschreibung

Figure 5: Amazon Prime recommendations for viewers of Terminator 3

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Figure 6: Average ratings for recommended movies under kNN

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. Moreover, this approach might recommend movies which a user has already watched and does not take user preferences into consideration.

### 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:”*

where:

* and 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 you can see, the suggestions are quite similar to those obtained under k-NN and include several movies from the Terminator series.



Figure 7: 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. 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).

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Figure 8: 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. However, ratings from users who have liked Terminator 3 are available only for 3 out of the 10 suggested movies.

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Figure 9: Average ratings for recommended movies under kNN

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 data set for similar movies, thereby completely ignoring individual user preferences with respect to genres, actors etc. and making the same recommendations to all users.

## 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 previous reviews by a user and try to find similar movies. Like the user-based approach it uses Pearson correlation or cosine similarity to derive the similarity. 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 watched 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 and scalability issues i.e. high computation time in large datasets by decreasing the number of dimensions, thereby mapping 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.

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

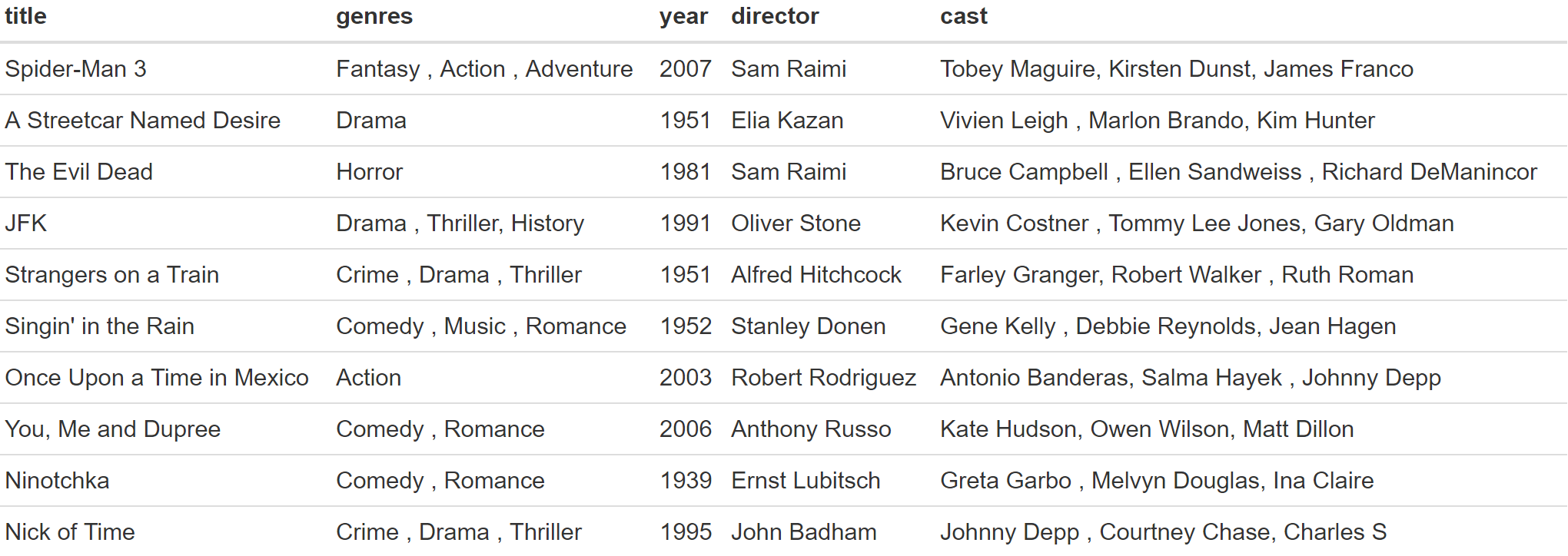


Figure 10: Recommendations based on collaborative filtering

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 recommender is learning constantly and recommendations should improve over time. However, this also means that the system will perform poorly for relatively new users. Moreover, the system is not transparent in how it makes recommendations.

## 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 loss function RSS is minimized. We build a decision tree for the most active user (user-Id 564). 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 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.

Ein Bild, das Screenshot, Vogel enthält.

Automatisch generierte Beschreibung  
Figure 10: 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 the user on average 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.

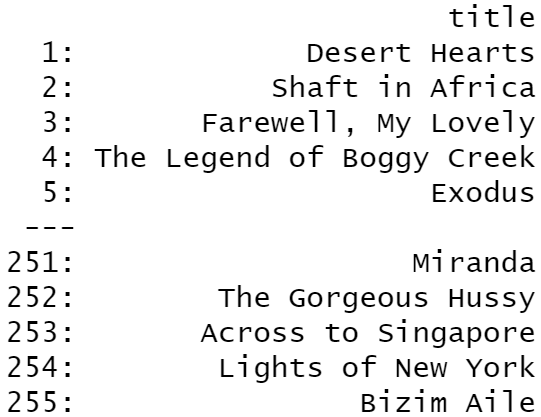


Figure 11: Decision Tree Recommendation

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%) data set and recompute our tree. Note that the resulting decision tree differs dramatically from the previous version. 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 data sets. Thus, we cannot make inferences as to whether our original tree makes good suggestions.

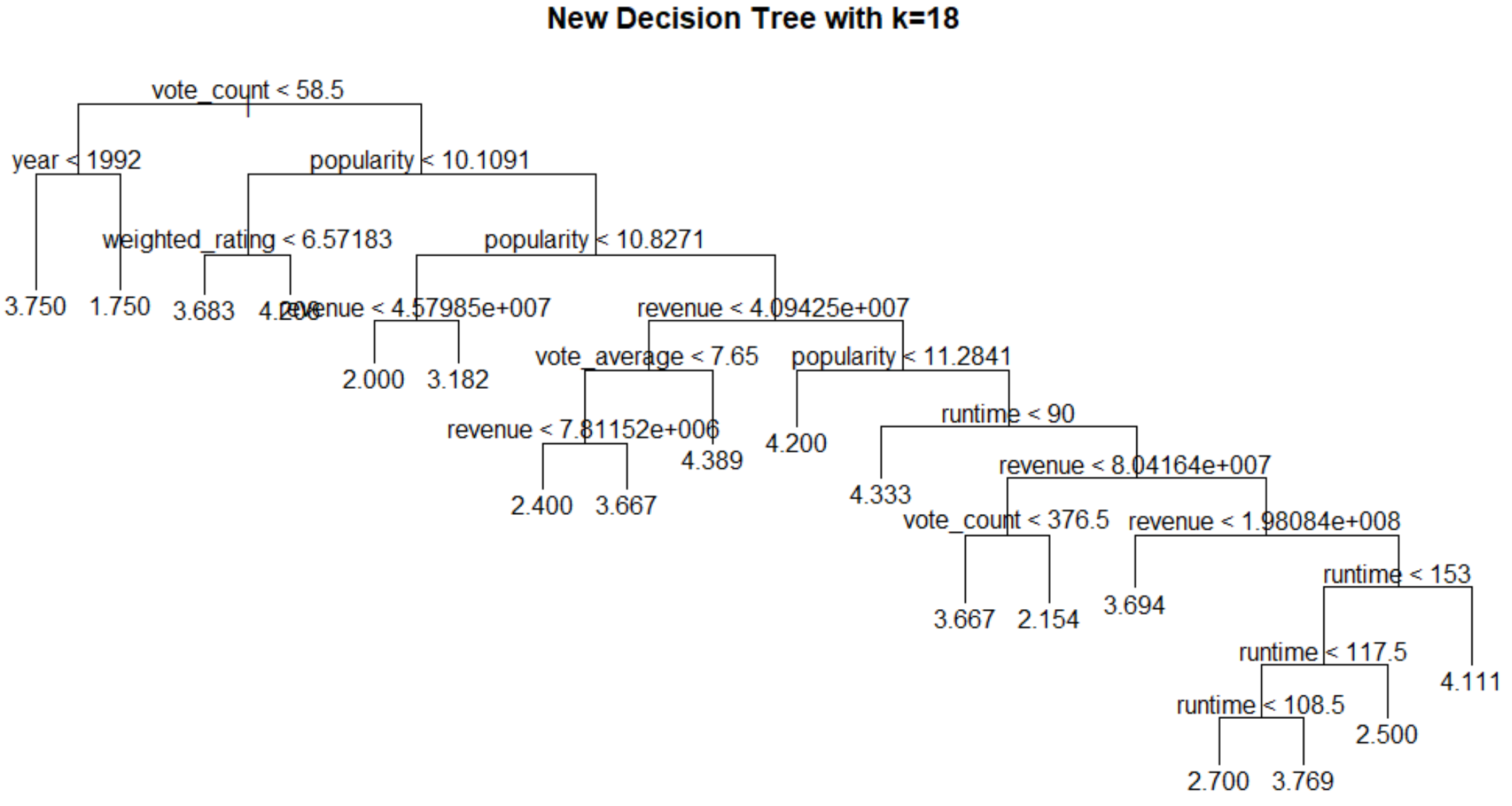


Figure 12: New decision tree w/o pruning

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 data set of only 1.14, the model performs poorly and one could more accurately predict the ratings in the test data set by simply always guessing the mean rating the user has assigned to movies from the training data set.

With 18 terminal nodes and 7 used features the new tree seems to be overfitting the training data. 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. 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 data set.

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Figure 13: 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.

# Conclusion

# List of references

Question Daniel: Shall I add weblinks?

* 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.
* Mao, K., Chen, G., Hu, Y., Zhang, I. (2016). Music recommendation using graph based quality model. *Signal Processing, 120,* 806-813.
* Peng, X., Liangshan, S., Xiuran, L. (2013). Improved Collaborative Filtering Algorithm In The Research And Application Of Personalized Movie Recommendations. *Fourth International Conference on Intelligent Systems Design and Engineering: Applications,* 349-352.
* 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.
* 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.
* Wang, Y. Shang, W. (2015). Personalized news recommendation based on consumer’s click behavior. *12th International Conference on Fuzzy Systems and Knowledge Discovery,* 634-638.
* Wei, S., Zheng, D., Chen, D., Chen, C. (2016). A hybrid approach for movie recommendation via tags and ratings. *Electronic Commerce Research and Applications, 18,* 83-94.

# Appendix

1. <https://www.kaggle.com/rounakbanik/movie-recommender-systems?select=movies_metadata.csv> [↑](#footnote-ref-1)