Predicting movies viewers favour based on feature generation

Hung Nguyen VuDuy[1], Nhung Vuongthi[2], Thanh Trinh[1,3]*

[1] Faculty of Computer Science, Phenikaa University, Yen Nghia, Ha Dong, Hanoi, 12116, Vietnam
[2] Hanoi School of Business and Management, Vietnam National University, Xuan Thuy, Cau Giay, Hanoi, 11310, Vietnam
[3] Phenikaa Research and Technology Institute (PRATI), A&A Green Phoenix Group JSC, No.167 Hoang Ngan, Trung Hoa, Cau Giay, Hanoi 11313, Vietnam
20010866@st.phenikaa-uni.edu.vn, nhungvt@hsb.edu.vn, thanh.trinh@phenikaa-uni.edu.vn

Abstract. Social media provides people with amazing platforms to improve their experiments and offer interesting items. Predicting which movies users like or not has been addressed by several techniques; however, it is still a challenging problem. To address the problem of movie prediction, we propose a method to predict whether movies viewers like or not based on an analysis of movie ratings and user profiles. First, we consider each user as one data table, including features and a class label (*Like* or *Dislike*). Then, we select four machine learning techniques to classify the data table of the user. Experiments are conducted on the real movielens data. The empirical results show that bagging model predicts more accuracy than other algorithms. Moreover, this method can be implemented as an online tool to recommend appropriate films to viewers.

Keywords: Recommendation · Classification · Movie · Feature generation

1 Introduction

In the past decades, online platforms, such as online social networks and e-commerce websites, have become an important part of people. Moreover, recommendation systems are extremely used in social networks and digital entertainment, such as Netflix, Prime Video, and IMDB, to provide people with many interesting items [5]. A recommendation system is treated as a filtering system that offers suggestions to users based on their past behavior or preferences[14]. It helps determine if a user likes or dislikes a product, a service, and an opinion.

Amazon Prime and Netflix [20] are famous examples of movie recommendation systems to allow viewers to give feedback about movies and send star ratings. Moreover, many viewers can express their thoughts and opinions about

* Corresponding author

the movies. Hence, other viewers can know more about a particular movie with a lot of reviews and comments; hereby, they are easy to decide whether to watch or buy the movie.

Therefore, it would be helpful for both movie producers and viewers if movie reviews could be automatically mined and summarized. Movie producers and filmmakers can target their audience more effectively by identifying groups of movies that viewers like or do not like. Hence, the facts lead to a research problem: predicting whether movies users will like or not. In this paper, we study the problem within movielens data context. This research problem is formulated as follows. ***Given a new movie, our objective is to predict whether a user like the movie or not based on the list of watched movies of the user***.

To address this problem, we propose a method to generate features and class labels based on an investigation of movie and user profiles. In other words, genres of movies and movie tags are used to generate features. Then, we label each movie as *Like* or *Dislike* class, which depends on the star ratings that the user rate the movie. Then, we obtain a data table of each user. This table includes n samples (movies), each of which has a list of generated features and one class label. Finally, we select different classifiers (decision tree, naive bayes, bagging, and boosting) to predict which movies users like or not. We conduct experiments on Movielens data. An empirical study was conducted on the real movielens dataset. The experimental results have shown that bagging method outperforms other ones.

The paper is organized as follows. Section 2 discusses some related works. Next is the proposed method given in Section 3. In section 4, the empirical study was presented. Finally, the last session draws conclusion and the future work.

2 Related work

Recommendation systems are one of the most applications implemented in social networks and e-commercial websites [8]. The systems assist people in finding interests, friends, and groups. Social networks and social media pose many challenging problems for researchers, such as item recommendation, group recommendation, and popularity prediction problems [13]. Movie recommendation has been studied in several works [4, 1]. Movie recommendation systems can be classified into content-based recommendation, collaborative recommendation, and hybrid recommendation system[8].

In the work[4], movie genres were discussed and analyzed to construct genre correlation to improve the accuracy of movie recommendation. They presented a new specific measure to compute the correlation based on different numbers of movies and differences of decades in movies. To combine collaborative-based and content-based methods to address the problem of movie recommendation, Afoudi et al. [1] proposed a hybrid model. The proposed model used the self-organizing map to rank the recommended movie list. However, this model took much time to yield high precision. Wei et al. [16] proposed a hybrid model, which was a combination of social tags and ratings. In their work, social tags

(annotation) were first normalized and reconditioned based on user preferences. Then, the model was improved by using previous ratings. In another work[9], the authors designed a hybrid recommendation system based on content-based and collaborative-based filtering techniques. The model used movie genres to compute the Euclidean correlation between movies. To get a better understanding of movie similarity, Zhao et al. [20] proposed new visual features that were derived from pictures of movies. The visual features were used to construct a new matrix factorization, which recommends movies to users. Netflix and Douban datasets were used in their empirical study. Zamanzadeh et al.[19] integrated a group-based model with rating similarity and demographic information of users to present a new recommendation system. Autoencoder method was first adopted to extract features; then, these features were taken for clustering users. The experiments of this work were conducted on MovieLens dataset. The mood of users was studied in the movie recommendation problem [18]. Obviously, a user might rate a movie due to his/her mood. They investigated two studies about the positive and nervous moods of users who tend to rate movies. They designed a mood-aware collaborative filtering technique to predict more accurate results. Another study [6] investigated the changes of user interests. The authors generated features from movies and user interests. In which the problems of long-term and short-term interests were taken into account for experiments. Another aspect of users was a profile that has been studied in work [3]. In their work, they first split two kinds of users: positive profile and negative profile. Then, they proposed a new model to recommend similar movies to a suitable user. Triyanna et al. [17] used collaborative filtering to recommend the most suitable movies to users. In their study, they defined a new similarity computed based on movie genres and all information of users: age, gender, occupation, and location.

3 Methodology

This section presents our research design. First, we describe the movie data used in experiments. Second, we present feature generation from the movie data. And finally, we briefly present four machine learning classifiers.

3.1 Movie data

The movie data set [4] is used in several works. Each movie has a specific title and a list of genres. A movie is rated by several users, and each user can rate many movies. And each user gives a movie a star rating from 1 to 5. This rating is stored with a timestamp. A user can give some comments about a specific movie by giving tags.

[4] https://grouplens.org/datasets/movielens/

3.2 Feature generation

In this section, we propose a method to generate features and a class label. Given a user u and a list of movies, we create features of movies for this user u as follows.

Genres. First of all, we extract features from the genres of each movie. The genres of each movie fall into one of the 19 categories, i.e., *unknown, Action, Adventure, Animation, Children, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western.* Each genre is defined as one feature. A movie that belongs to one of the listed categories will be filled with 1; otherwise, it will be filled with 0. For example, a movie with the title *'Batman Forever (1995)'* has four genres: *Action, Adventure, Comedy, Crime.* We generated 19 features from this movie. In which the values of features *Action, Adventure, Comedy, Crime* are assigned to 1, the others are set to 0. Moreover, the number of genres that the movie falls into is considered one feature. The value of this feature is set to 4.

Positive or negative movie. The feature indicates whether a movie is positive or negative. In other words, each movie is tagged as negative or positive based on its tags. Each tag can have a positive or negative meaning. We use sentiment analysis [7] to compute the value of each tag of this movie for both positive and negative meanings. For example, the movie "Princess Bride, The (1987)" has 5 tags: *adventure, fantasy, funny, medieval, witty.* This movie will be assigned as positive or negative as follows. We first use NLTK to compute the positive and negative meanings of each tag. Hereby, negative values of the four words {*adventure, fantasy, funny, medieval, witty*} are {0, 0, 0, 0}. And, positive values of the four words {*adventure, fantasy, funny, medieval, witty*} are {1, 0, 1, 0}. Then, the average negative value is 0, and the positive is 0.4. Finally, this movie is set to positive.

Highest similarity between tags and genres. The highest similarity that is computed based on tags and genres of each movie is also designed as a feature. For example, the movie '*Forrest Gump (1994)*' genres have been classified as *Comedy, Romance, Drama, and War.* This movie tags include *bittersweet, inspirational, Oscar, Best Actor, Oscar, Best Picture, psychology, quirky, Vietnam war.* The highest similarity of a pair (a tag and a genre) is 0.505, and this pair consists of *quirky* (tag) and *comedy* (genre).

Highest similarity between tags and name. Likewise, we calculate the similarity between tags and the name of the movie and get the largest similarity value. For example, the similarity between *Forrest Gump* (name)and *Oscar* (tag) is 0.35. And the value of this feature is set to 0.35.

Highest similarity among movie tags. This feature is designed to depict the highest similarity among tags in one movie. Word similarity is a value in a range of 0 and 1 that tells us how close two words are. This value is done by finding the similarity [12] between two words in the list of words. For instance, the movie '*Princess Bride, The (1987)*' includes five tags: *adventure, fantasy, funny, medieval, witty*. We first compute the similarity between each tag and the others based on Spacy library [5], then get the largest value. As a result, the similarity value between *witty* and *funny* is the highest, and the value is 0.641.

Highest similarity between tags of user u and all tags of the other users. This feature is used to represent the highest similarity between tags of user u and all tags of the other users for a specific movie. Each one of a user's tags and each one of all other users' tags is computed to obtain a similarity value. The highest value is obtained.

Day of week. Each user rates one movie with a specific timestamp; hence we obtain a specific day of the week from the timestamp. For example, the weekday of timestamp (1554071255) is Monday. Weekday is also treated as one feature.

Class Label. Each movie is rated in a range of 1 to 5 stars. A movie rated at least 4 stars is considered that the viewer likes this movie; otherwise, the viewer dislikes the movie. For example, User ID 4 in Movielens dataset rated '*movie ID 260*' with 3.5 stars; we label this movie as *Dislike* class. And, this user rated '*movie ID 296*' with 4 stars, the class of this movie is labeled with *Like*.

3.3 Classifiers

From feature generation, we obtain 26 features and one class label for each movie that is rated by a user. Hence, we present each user u as a data table $= \{X, Y\}^n$, where n is the number of movies rated by user u, X^i is a vector of 26 features and $Y^i \in \{Like, Dislike\}$ for $movie^i$. In this work, we select four well-known classifiers to predict which movies viewers like or not as follows.

Decision tree. Decision tree method [10] is a widely used and effective model in both regression and prediction problems of supervised learning. The decision tree model can be implemented by different node splitting methods, such as gini index and gain ratio.

Naive bayes. The Naïve Bayes model [15] is an algorithm based on Bayes' theorem in probability theory to make judgments and classify data based on observed and statistical data. This method assumes that all features are independent of each other and they have an equal effect on the outcome.

[5] https://pypi.org/project/spacy/

Bagging. Bagging (Bootstrap aggregating) that is proposed by Breiman [2] is an ensemble method. This method generates K datasets from a training dataset, and each generated dataset is used to train a learning model. Then we obtain an ensemble model of those K learning models.

Boosting. Different from bagging, the main idea behind boosting [11] is to create a strong learner from a set of weaker learners in a sequential iteration. In other words, a new model is built based on the previous model.

4 Empirical study

4.1 Data processing

We use MovieLens 25M dataset that includes 25 million ratings and around 1 million tags across 62423 movies. The data were generated by 162541 users between 1995 and 2019. In this work, we only select users who rate at least 40 movies with *Like* and 40 movies with *Dislike*, and each user comments more than 15 tags. Hence, three thousand users are chosen from MovieLens 25M. Each user is considered a data table. In other words, we have 3000 data tables in the experiments.

4.2 Experimental setting

We implemented six comparison models for a comparative study for 3000 data tables.

- **Decision Tree (DT)**. We used Gini index to build the tree model.
- **Naive Bayes (NB)**. We used Gaussian distribution in this classifier.
- **Bagging**. Decision tree (DT) and Naive bayes (NB) were adopted as a base classifier in Bagging, respectively. Hence, we obtained two models: **Bagging (DT)** and **Bagging (NB)**. The number of bootstraps for two bagging models was set to 100 (K=100).
- **Boosting**. Similar to Bagging, NB and DT were selected as a base classifier in Boosting; therefore, we achieved two boosting models: **Boosting (DT)** and **Boosting (NB)**. The number of base models in each boosting model is set to 100.

For each user, we split the number of movies into two parts: 80% for training and 20% for testing. Training part is trained in each comparative model, then testing part is used to assess the performance of six comparative models. This process is run 100 times to yield 100 results of accuracy for the six models. The models and processing data were implemented in Python.

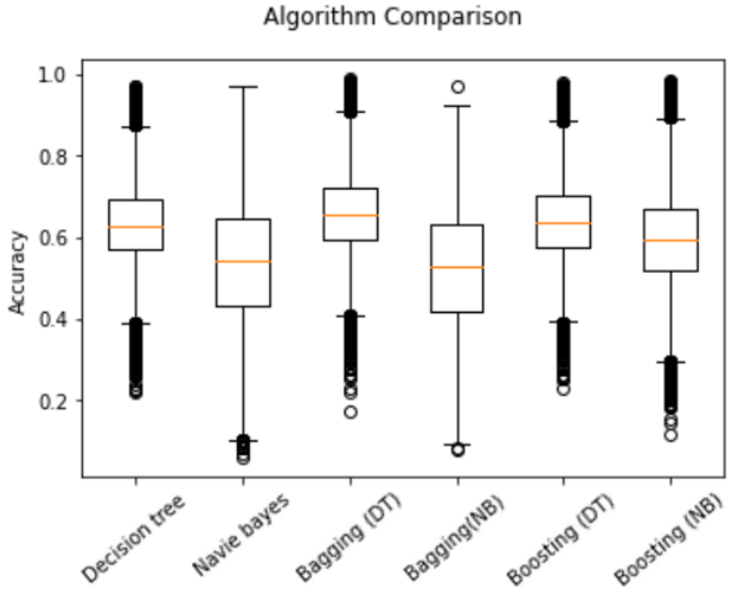
The accuracy of each model is computed as follows.

$$Accuracy = \frac{\Sigma_i^n U_{acc}^i}{n} \tag{1}$$

where n is the number of users (n = 3000), U_{acc} is the average accuracy of user U^i for 100 results yielded by the model.

Table 1: Average accuracy of 6 six models.

	Accuracy	Standard deviation
Decision tree	0.63	0.09
Naive Bayes	0.53	0.14
Bagging (DT)	0.65	0.09
Bagging (NB)	0.52	0.14
Boosting (DT)	0.64	0.09
Boosting (NB)	0.59	0.14



(a)

Fig. 1: The accuracy distributions for six comparative models.

4.3 Experimental results

Table 1 presents the average accuracy of six models for 3000 users. Figure 1 demonstrates the distribution of accuracy for the six comparison models. We observe that ensemble methods outperform single classifiers. And bagging is the best method, while naive bayes is the worst.

Each user has a data table with at least 80 samples, each of which has 26 generated features and a class label (*Like* or *Dislike*). We have 19 features extracted from genres. And 7 features are yielded by an analysis of tags, genres, titles, etc. Hence, each user has a lot of features with the same value 0 or 1; these features are relevant to movie genres. Therefore, the features are highly

correlated. Naive bayes model is constructed based on an assumption about independent features. So, navie bayes yields the worst results.

Moreover, decision tree model is built based on the Gini index method; hence, this model can choose a group of informative features. Therefore, decision tree can yield better accuracy than naive bayes. **Bagging (DT)** is the best because this model is a combination of 100 tree models. It is seen that **Boosting (NB)** yields better results than **Bagging (NB)**. It can be explained as follows. 1) Each model in boosting model is learned from the previous one; hence the next one can be stronger than the previous one. 2) And each model in bagging is independent of others. Therefore, the results of **Bagging (NB)** and **Boosting(NB)** are reasonable.

It is observed that there are a lot of outliers yielded by comparative models, demonstrated in Figure 1. Each model runs 3000 data tables, and users have different interests in rating or commenting on movies. Hence, the outliers can not be avoidable.

5 Conclusions

This paper presents a study to deal with the problem of movie prediction within the real movielens dataset. Machine learning techniques are applied to predict whether movies viewers like or not based on movies and users' profiles. For each user, we first collected all movies rated by this user. We generated 26 features of movies, and we labeled a movie as *Like* and *Dislike* based on the star ratings of this user. Therefore, we obtained a data table of each user. Finally, we used this data table to train machine learning models to predict whether movies the user likes or not. Experimental results have shown that bagging model yields the best accuracy compared to other models. Moreover, this study can be implemented as an online tool for e-commerce and social media systems in order to recommend suitable films to movie watchers.

References

1. Y. Afoudi, M. Lazaar, and M. Al Achhab. Hybrid recommendation system combined content-based filtering and collaborative prediction using artificial neural network. *Simulation Modelling Practice and Theory*, 113(April):102375, 2021.
2. L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.
3. Y.-L. Chen, Y.-H. Yeh, and M.-R. Ma. A movie recommendation method based on users' positive and negative profiles. *Information Processing & Management*, 58(3):102531, may 2021.
4. S. M. Choi, S. K. Ko, and Y. S. Han. A movie recommendation algorithm based on genre correlations. *Expert Systems with Applications*, 39(9):8079–8085, 2012.
5. M. Eirinaki, J. Gao, I. Varlamis, and K. Tserpes. Recommender Systems for Large-Scale Social Networks : A review of challenges and solutions. *Future Generation Computer Systems*, 78:413–418, 2018.
6. J. Li, W. Xu, W. Wan, and J. Sun. Movie recommendation based on bridging movie feature and user interest. *Journal of Computational Science*, 26:128–134, may 2018.

7. NLTK. Sentiment Analysis, 2022.
8. S. Rajarajeswari, S. Naik, S. Srikant, M. K. Sai Prakash, and P. Uday. Movie Recommendation System. pages 329–340. 2019.
9. S. Reddy, S. Nalluri, S. Kunisetti, S. Ashok, and B. Venkatesh. *Content-based movie recommendation system using genre correlation*, volume 105. Springer Singapore, 2019.
10. S. L. Salzberg. C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993. *Machine Learning*, 16(3):235–240, sep 1994.
11. R. E. Schapire. The Boosting Approach to Machine Learning: An Overview. pages 149–171. 2003.
12. Spacy. a library for advanced Natural Language Processing, 2023.
13. T. Trinh and N. Vuongthi. A Predictive Paradigm for Event Popularity in Event-Based Social Networks. *IEEE Access*, 10:125421–125434, 2022.
14. T. Trinh, D. Wu, R. Wang, and J. Z. Huang. An effective content-based event recommendation model. *Multimedia Tools and Applications*, apr 2020.
15. G. I. Webb, J. R. Boughton, and Z. Wang. Not So Naive Bayes: Aggregating One-Dependence Estimators. *Machine Learning*, 58(1):5–24, jan 2005.
16. S. Wei, X. Zheng, D. Chen, and C. Chen. A hybrid approach for movie recommendation via tags and ratings. *Electronic Commerce Research and Applications*, 18(February):83–94, 2016.
17. T. Widiyaningtyas, I. Hidayah, and T. B. Adji. User profile correlation-based similarity (UPCSim) algorithm in movie recommendation system. *Journal of Big Data*, 8(1):52, dec 2021.
18. P. Winoto and T. Y. Tang. The role of user mood in movie recommendations. *Expert Systems with Applications*, 37(8):6086–6092, aug 2010.
19. Z. Zamanzadeh Darban and M. H. Valipour. GHRS: Graph-based hybrid recommendation system with application to movie recommendation. *Expert Systems with Applications*, 200:116850, aug 2022.
20. L. Zhao, Z. Lu, S. J. Pan, and Q. Yang. Matrix factorization+ for movie recommendation. In *IJCAI International Joint Conference on Artificial Intelligence*, volume 2016-Janua, pages 3945–3951, 2016.