A Baseline Naïve Bayes Movie Recommender System

Thanapar Leelasathapornkun
Araya Siriadun

Bachelor of Engineering in Software Engineering
International College
King Mongkut's Institute of Technology Ladkrabang
Academic Year 2018
KMITL-2019-IC-B-003-002

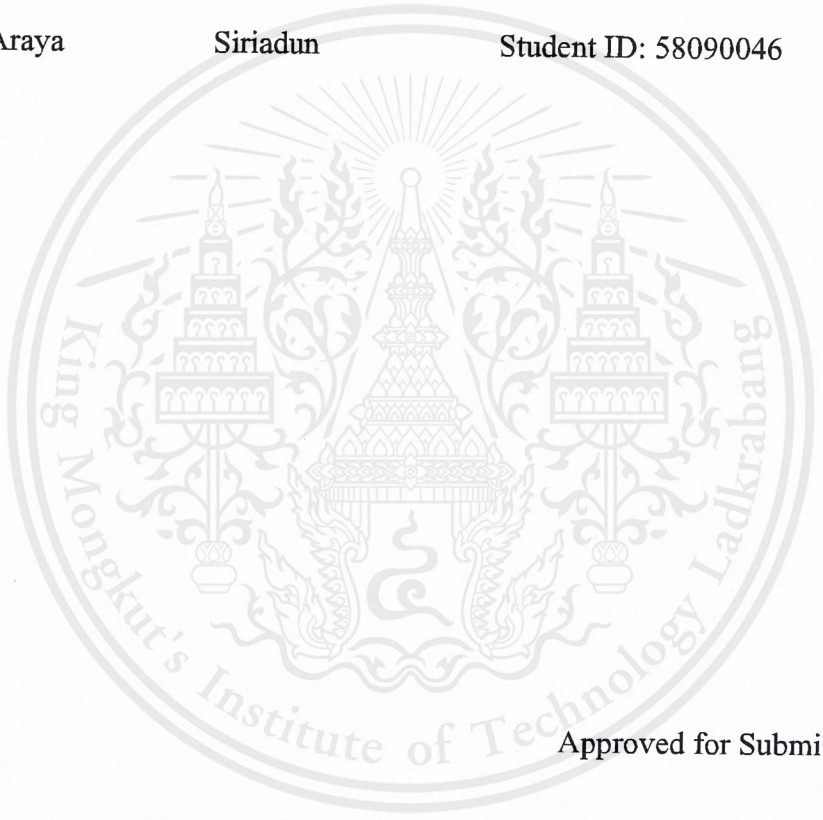
Thesis - Academic Year 2018

Bachelor of Engineering in Software Engineering
International College
King Mongkut's Institute of Technology Ladkrabang

Title: A Baseline Naïve Bayes Movie Recommender System

Authors:

1. Thanapar Leelasathapornkun Student ID: 58090019
2. Araya Siriadun Student ID: 58090046



Approved for Submission

.....Veera Boonjing.....

(Associate Professor Dr. Veera Boonjing)
Advisor

Date 24/6/19

Acknowledgments

This thesis could not be completed without a lot of advice from our advisor, Associate Professor Dr. Veera Boonjing. And we also would like to thank Dr. Ukrit Watchareeruetai and my colleagues for giving suggestions in presentations and useful discussions. Furthermore, we are thankful to our lecturers at the faculty of International College for support and knowledge. Also, we would like to give special thank to our families for encouragement, support, and motivations; however, people who are not mentioned above also play an essential role in supporting us. This project cannot be completed and get excellent outcomes without their co-operation.

Abstract

The movie application's industry currently increases prosperity in 2018, such as Netflix, iflix, and Hollywood HDTV. All of those web services use some recommendation engines to suggest movies. It is imperative that the movie recommendation system should be able to predict efficiently for customers attraction. Moreover, quickly finding one favorite movie from a huge amount of movies become a fundamental issue. Several algorithms have been used in movie recommendation world, including Naïve Bayes. Because of simplicity and effective of Naïve Bayes classification, it widely applies in many fields. Naïve Bayes has strong independent assumption among the attributes on given class attributes.

On the other hand, [30] its assumption can degrade the accuracy of classification. And, Naïve Bayes ignore local information such as user preference, so these two key points result that some case of predictions is incorrectly forecast which also affect the Naïve Bayes efficiency. We present Naïve Bayes classification with baseline algorithm to alleviate the independent assumption of Naïve Bayes [2] and to added the ignored information to Naïve Bayes using our method.

Keywords: Recommender System; Naive Bayes Classifier; Baseline Algorithm

Table of contents

List of figures

List of tables

Chapter 1
Introduction

1.1 Motivation

Everyone is rushing for their destination goals. Since the world is growing faster like never before, it results into the development of almost every sector in our life. One is the recommend sector in the online market as it grows exponentially, it's obvious that competition will enter in this field also. Thus, the recommender engines, e.g. recommend books, movies or articles based on our past actions, is one of the facilities that the owner use to attract users. In this article, we focus on the improved Naïve Bayes classifier in Movie recommendation system.

Movie recommendation system is facility that the businessman normally uses to attract users. Thus, the competition of Recommend engine grows dramatically. Its recommendation is one type of information filtering system, and, basically, it consists of three types of recommendation system, which is Collaborative recommender systems, Content-based recommender systems [7], Demographic recommender systems [7] [20], and Utility-based recommender systems[7]. And, also Naïve bayes also had been applied in movie recommendation system.

Naïve Bayes has been widely played an important role in the world of machine learning competing well with other complex classifiers. Its classifier has been applied effectively in the field of several practical applications such as text classification, spam filtering, and medical diagnosis, and also in the field of recommender system. Naive Bayes classifier, based on Bayes' theorem, is the simplest and effective classifier which having strong independent attribute assumption on given class attributes.

Naïve Bayes is very useful for three reasons [11]. First is simple and fast prediction, and also work well with multi-class prediction. Next, the performance of Naïve Bayes better when compare with other model such as logistic regression because Naïve

Bayes better use less data. And Naïve Bayes work better with categorical input variables than numerical variable because, in case of numerical variable, normal distribution is assumed. In contrast, Naïve Bayes have three drawbacks. For categorical variable, the data that had not seen in training data set. the model will assign the zero probability to that specific data, so it is unable to make a prediction because most of predictions are zero. It may be solved by smoothing techniques. Next, Naïve Bayes is called as bad estimator because the attribute independence assumption is patently false if you're trying to estimate the probability [10]. Lastly, the limitation of strongly assumption of independent predictors, so it hard to find set of completely independent predictors and apply as real life solution.

And, the strong independent assumption of Naïve Bayes ignore the joint probability between different classes. Because of this assumption, the Naïve Bayes cannot capture any information of between classes[30], and Naïve Bayes apply global information, but the local information is abandoned which this means that some waste ratings or unintended ratings are included in the model. In result, according to previous 2 weakness of Naïve Bayes, when we train and predict the data, some case of prediction is incorrectly forecast. From the past solution of alleviating independent assumption, some apply complicated algorithm such as neural network.

We believe that there is the solution to alleviate independent assumption and ignorance problem , and it is able to improve the accuracy of Naïve Bayes. Thus, we purpose "Combining Naïve Bayes Classifier with Baseline Algorithm" as a solution to improve Naïve Bayes efficiency.

1.2 Objectives

The targets of this thesis are to improve the accuracy of Naïve Bayes classifier using baseline algorithm in movie recommendation system. Using Combining NaBayes Classifier with Baseline Algorithm, it can relieve the strong independent assumption of Naïve Bayes, and reduce the gap of Naïve Bayes results in order to improve the result.

To evaluate the performance of recommendation system, mean absolute error (MAE) is used in this thesis. Therefore, we want to reduce the measurement score for the better movie recommendation system. The lower mean absolute error (MAE) value mean the better result. The movie recommendation system efficiency will be demon-

strate via those measurements with Movielens dataset [12] to analyze the behavior and accuracy of our proposed method.

1.3 Scope of Work

The scope of this project is:

- To develop a recommender system that has ability to generate predictions quickly consisting of thousands of users and items.
- To develop a recommender system that can accurately predict the rating of the non-rated user/item combination and recommend items based on these predictions.
- To develop a recommender system that should be able to produce recommendation for all the existing items and users regardless of the new user.
- To develop a recommender system that its performance should not degrade with sparsity.

1.4 Thesis Structure

This thesis consists of seven chapters which are arranged as follows:

- Chapter 1 Introduction – mentions the motivation, objectives, and scope of work.
- Chapter 2 Related works – describe the related researches.
- Chapter 3 Background knowledge – shortly inform the knowledge using in this research.
- Chapter 4 Our approach – present the our approach of Combining Naïve Bayes Classifier with Baseline Algorithm.
- Chapter 5 Implementation – is about the applied programming language and tools, and the detail of implementation.
- Chapter 6 Experimentation – describe the detail of data input, evaluation techniques, and the result of experiment.
- Chapter 7 Conclusion – talk about summary, and future work.

Chapter 2
Related Works

2.1 Previous Recommender systems approach

Recommender systems are based on a variety of approaches such as content based approach, collaborative filtering approach, and hybrid approach.

The content-based approach [21] relies on the similarity of the recommended movie. The base idea is that if the user like a movie, then the user will also like the other "similar" movie as well. It commonly works well when it is obvious to determine the context or properties of each movie. This approach works with data that the user provides, specifically movie rating data. Based on that data, a user's profile is generated, which is using to provides suggestions to the user. If the user provides more inputs or takes actions on the recommendations, then the system will become more accurate. However, this approach cannot recommend a movie outside the user's profile and also unable to utilize the various recommendations of other users.

The collaborative filtering (CF) [36] is largely expanding in such a way that this approach influences most of the recommender systems. It is categorized into two primary types. First, the memory-based collaborative filtering approach [6] is based on past behavior and not on the context. Precisely, it is based on the similarity in preferences, tastes, and choices of between the users. It analyses how similar the preference of an individual user is to another and obtains recommendations from that. In common, This approach is the workhorse of the recommender system. The algorithm has a fascinating property of being able to do feature learning on its own, which indicates that it can begin to learn for itself what features to use. However, there are some drawbacks to this approach. They present serious scalability problems given that the algorithm has to process all the data to compute a single prediction. With a high number of users or items, these algorithms are not appropriate for online systems which recommend items in real time. It does not address the well-known cold-start problem, that is when a new

user or movie enters in the system. Moreover, It cannot deal with sparse data, indicating it is difficult to find users that have rated with the same movie [8] . Secondly, the model-based collaborative filtering approach [22] is based on matrix factorization (MF) which has received greater exposure, principally as an unsupervised learning method for underlying variable decomposition and dimensionality reduction. Matrix factorization is publicly used for recommender systems where it can deal better with scalability and sparsity than memory-based collaborative filtering [8].

Some research [17] seem to agree that movie recommendation systems are centered on collaborative filtering and clustering which focus on the orientation algorithm in collaborative filtering recommendation process that is the k-nearest neighbor (kNN) algorithm. Specifically, clustering algorithm with a bio-inspired algorithm called cuckoo search which is claimed to have better performance when compared with other algorithms such as genetic algorithms and particle swarm optimization. It was found that it takes less time than other algorithms applied to the same dataset with the strong convergent evidence which are mean absolute error (MAE) equal to 0.68 at 64 number of clusters.

2.2 Improved Naïve Bayes research

According to [27] author use Particle Swarm Optimization method to optimizate the Naïve bayes classifier. Particle Swarm Optimization (PSO) : is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. In this article,PSO technique is used to improve the efficiency and accuracy of Naïve Bayesian classification technique. The fitness function was modelled based on the structure of Naïve Bayes algorithm. The optimal value was determined by the PSO technique. Thus, Naïve Bayes accuracy was used as the fitness function. This research focus on the accuracy of Naïve Bayes and adding new method to optimize Naïve Bayes, and it overlooks the weakness of Naïve Bayes, the strongly independence assumption.

2.3 Improved Baseline research

From article [19], this paper is used to propose the way to extend the model to exploit both explicit and implicit feedback by the users using some methods including

baseline algorithm. Baseline is applied to provide an alternative way to learn user preferences, so this article modify the Baseline formula by add another set of weights to baseline estimates. It helps to emphasize the missing ratings. Thus, it will consider user opinion on both no ratings and existing ratings. For example, assume that user rate movie "Lord of the Rings part 3" high also gave high ratings to "Lord of the Rings 1–2." This will make the high weights from "Lord of the Rings 1–2" to "Lord of the Rings part 3.", and if a user did not rate "Lord of the Rings 1–2" at all, his predicted rating for "Lord of the Rings part 3" will be reduced the ratings. Thus, baseline algorithm is used in this article to increase the efficiency of baseline in movie recommender systems. This baseline article focus on using baseline to extract user preference which is able to improve Naïve Bayes in this article.

Chapter 3
Background Knowledge

3.1 Recommender Systems

Recommender systems began as an individual field of research about the mid-1990s and derived from different other research areas like cognitive science, approximation theory, information retrieval, forecasting theory, consumer modeling, and also management science [1]. Recommender systems have improved a lot since then. Nowadays, Recommender systems are aiming to produce these individualized outputs by learning about different user preferences and then predicting their unique needs. Due to varying conditions there exist various types or techniques of recommender systems, that all differ in the way a definite recommendation is given. It depends on the scenario what kind of system is chosen and applied. In the following the most well-known types are explained in detail.

3.1.1 Collaborative recommender systems

Collaborative recommender systems are one of the most commonly used systems. In general, these systems generate a user profile based on ratings of distinct objects and then compare these against a broader user group. The system recognizes similarities between users based on their ratings and then generates new recommendations based on this inter-user comparison. Numerous algorithms have been used in measuring user similarity or item similarity. For example, the k-nearest neighbor (k-NN) algorithm and the Pearson Correlation. Collaborative recommender systems can vary in the way a rating is determined. They can be binary, focused on opinion over time, model or memory-based [7]. A typical example is the additional product suggestions on Amazon (People who bought this item also bought that item).

3.1.2 Content-based recommender systems

Content-based recommender systems are based on the characteristics of the content in order to produce an individualized recommendation. The content is classified into different tags or features and can associate these to the ratings of the users. By doing this, the systems learn from the user profile and displays relevant recommendations [7]. To involved the features of the items in the system, an item presentation algorithm is used. A generally used algorithm is the term frequency-inverse document frequency. A typical example of this recommender system is the movie suggestions on Netflix. If the user watched a movie of the genre adventure once or even gave thumb up for that movie, the user will acquire suggestions with the corresponding label. How the content is labeled and matched against each other is regularly kept a secret.

3.1.3 Demographic recommender systems

The classification of various demographic attributes creates the baseline of demographic recommender systems. These attributes can include gender, age, cultural or other personal characteristics. The derived model of every combined personal attribute is then matched into a list of user stereotypes that were manually generated. In opposite, collaborative recommender systems use history of different ratings are not required [7]. Demographic recommender systems can be used in a wide field of applications but mainly benefits to solve the cold start problem or new user problem. This kind of problem appears when a model of a user does not exist yet, e.g. when the user uses a recommender system for the first time. Because the interests of users are undiscovered, suggestions can only be performed based on the available demographic information of the current users [20].

3.1.4 Utility-based recommender systems

Utility-based recommender systems are attempting to calculate the utility of each object with the user. In the following step, they try to increase this utility factor to satisfy the personal needs of each user. Consequently, a long-term evaluation of the user through ratings is not required because the system learns to recognize the user over time. The main problem thus is to define this usability method for each user which is regularly done through several user satisfaction methods [7].

3.1.5 Knowledge-based recommender systems

In a knowledge-based recommender system, there exists underlying data of the relation between the needs of a user and a particular item. This assumption between user needs and a particular item, that characterizes every recommendation types, can again then be stored in a user profile [5]. Fundamentally, every kind of expert systems can be considered a knowledge-based recommender system because the value of the system is formulated by the acquisition of the knowledge of the user [9].

3.2 Proposed algorithm in recommender data model

3.2.1 Naïve Bayes Classifier (NBC)

Naïve Bayes is first introduced by Monsteller and Wallace for text classification in 1964 [16]. Naïve Bayes is a technique which is simple and Naïve as its name, and it is applied for classifiers construction which builds Naïve Bayes probabilistic models. In equation (3.1), this is Naïve Bayes formula, and $\hat{A}$ will returns the maximum posterior probability of class $b \in B$.Hat notation means estimate of correct class. At first initiation, Naïve Bayesian classification is applied Bayes' rule, equation (3.2), to transform formula that have useful properties of Bayes' theorem. Bayes' rule is based on Bayes' theorem. Bayes' rule is the way to update based on the arrival of new pieces of evidence. Thus, conditional probability P(b | a) in , equation (3.1), can substituted by (P (a | b) · P(b)) / P(a), so the result is equation (3.3).

$$\hat{A} = \arg\max_{b \in B}(P(b \mid a)) \tag{3.1}$$

$$P(b \mid a) = \frac{P(a \mid b) \cdot P(b)}{P(a)} \tag{3.2}$$

$$\hat{A} = \arg\max_{b \in B}\left(\frac{P(a \mid b) \cdot P(b)}{P(a)}\right) \tag{3.3}$$

$$\hat{A} = \arg\max_{b \in B}(P(a \mid b) \cdot P(b)) \tag{3.4}$$

Next, Formula (3.3), whether the formula has the denominator P(a) or not, the P(a) value of same document will not change, so we can ignore P(a). Then, formula change into, equation (3.4). Naïve Bayes assume conditional independence strongly that probabilities $P(a \mid b)$ are independent on given class b. It means that if there are

any changes on the value of one feature, it will not have any impact on the value of other features. Using the following equation (3.5), it is substituted into Fig. 3.4 at $P(a \mid b)$, then its result is formula equation (3.6) which is called as Naïve Bayes classifier [16]. Naïve Bayes is easily scalable because Naïve Bayes is probabilistic model, and it uses simple algorithm that can respond request instantaneously, so good choice for real-world applications [14].

$$P(a_1, a_2, ..., a_i \mid b) = P(a_1 \mid b) \cdot P(a_2 \mid b) \cdot ... \cdot P(a_i \mid b) \tag{3.5}$$

$$c_{NBC} = \underset{\mathbf{b} \in B}{\arg\max} \left(P(b) \cdot \prod_{a \in A} P(a \mid b) \right) \tag{3.6}$$

Naïve Bayes classifier is supervised machine learning algorithm and based on bayes' rule, equation (3.2). It employs the probabilistic relationships between the class attributes and the features set. This classifier is a family of machine learning algorithms. There are four steps to build Naïve Bayes classifier which is ,first is calculate the distribution over the data inputs for each value C, so we got the probabilistic tables. Then, using tables in recent step can give $P(a_1, a_2, .., a_i \mid b = V_j)$ value. Next, it need to calculate $P(b = V_j)$ using existing records with $B = V_j$. Lastly, Naïve Bayes classifier calculate new prediction using equation (3.6) formula which b is class attributes and a_1 to a_i is other features. Typically, it can be applied in wide range of classification fields such as filtering spam, classifying documents, sentiment prediction etc [14].

For the example of Naïve bayes classifier, firstly, the data should be process into manageable format Fig. data table 3.1. According to Fig. 3.1, first row is Movie A to F, and First column is User 1 to 7, and the data of table are ratings. Some of missing data use as 0 ratings, no ratings.

To train the data using Naïve Bayes classifier, the $P(a = a_i \mid b = b_j)$ is the conditional probability need to find for each specific user i and movie j, and assigned to each value to the table 3.3 for each other Movies with specific class attribute movie F such as probability table of Movie A Fig. 3.4.

To calculate probabilistic relationship ,Let assume to find class rating X with user rating Y for specific movie A, the probabilistic value of row Y and column X of probability table can be calculated from table Fig. 3.1. If counting number of the specific user rating of movie A equal to Y and movie of class attribute F rating equal to

X, counting number will be equal to 1 otherwise equal to 0. And counting number must be divided with the counting value of class attribute of F equal to X, so the process is repeated to retrieve all probability tables.

In this example, to find value of row 1 and column 0 of table movie A, Fig. 3.4, counting value where user ratings of movie A equal to 1 and also the user has rating on the movie class attribute F equal to 0. class attribute F at user 4 equal to 0 ,but user 4 rate movie A is not equal to 1, so it is ignored. Next, User 17 rate movie F as 0, and also user 17 rate movie A equal to 1 which is correct criteria, so it is counted as 1. Then, it is divided with counting value of class F which equal to number 0, counting as 2. Therefore, the value of row 1 and column 0 of table movie A , Fig. 3.4, equal to 1 divided by 2 ,so result is 0.5 . After that, this method will be applied to each row Y and column X with specific movie A to E because the probabilistic tables can be calculated for each movies except the movie F that is assigned as column of class attributes. So in this case, After the data was trained using Naïve Bayes classifier, we got 5 probabilistic model or probabilistic tables for specific class attribute movie F Fig. 3.4 , 3.5 , 3.6 , 3.7 , 3.8.

Another probabilistic values for each specific ratings, in this case, there are possible five ratings 0 to 5, so the counting values from P(f=0) to P(f=5), where ratings in rows of class movie F, is divided by the total number of users or possible user ratings of class F. In this example of Fig. data table 3.1, P(f=0) value is the user ratings for movie F equal to 0 divided by total possible user ratings of class F, so the result is 2 divided by 20 which equal to 0.1 . Then, it will be applied the same way to calculate for ratings from 0 to 5 as shown in table Fig. 3.2.

User\Movie	A	B	C	D	E	F
1	1	2	5	3	2	4
2	1	2	3	2	5	1
3	2	2	2	3	1	2
4	5	1	3	4	1	0
5	1	3	5	3	2	4
6	1	2	4	3	0	1
7	1	2	2	3	1	1
8	4	3	2	3	2	5
9	4	5	2	5	2	3
10	2	5	2	1	2	4
11	3	3	4	4	4	5
12	4	3	3	2	1	4
13	1	2	2	3	1	3
14	1	5	0	1	1	3
15	2	2	4	2	1	2
16	2	0	3	2	1	1
17	1	2	2	3	4	0
18	4	2	0	4	5	2
19	0	3	4	1	2	2
20	1	5	2	1	3	2

Figure 3.1: Data table

Naïve Bayes Prior probability of class F	
P(F = 0)	0.1
P(F = 1)	0.2
P(F = 2)	0.25
P(F = 3)	0.15
P(F = 4)	0.2
P(F = 5)	0.1

Figure 3.2: Prior probability of class F

		Class attribute B					
		0	1	2	3	4	5
Attribute A	0	P(A=0 | B=0)	P(A=0 | B=1)	P(A=0 | B=2)	P(A=0 | B=3)	P(A=0 | B=4)	P(A=0 | B=5)
	1	P(A=1 | B=0)	P(A=1 | B=1)	P(A=1 | B=2)	P(A=1 | B=3)	P(A=1 | B=4)	P(A=1 | B=5)
	2	P(A=2 | B=0)	P(A=2 | B=1)	P(A=2 | B=2)	P(A=2 | B=3)	P(A=2 | B=4)	P(A=2 | B=5)
	3	P(A=3 | B=0)	P(A=3 | B=1)	P(A=3 | B=2)	P(A=3 | B=3)	P(A=3 | B=4)	P(A=3 | B=5)
	4	P(A=4 | B=0)	P(A=4 | B=1)	P(A=4 | B=2)	P(A=4 | B=3)	P(A=4 | B=4)	P(A=4 | B=5)
	5	P(A=5 | B=0)	P(A=5 | B=1)	P(A=5 | B=2)	P(A=5 | B=3)	P(A=5 | B=4)	P(A=5 | B=5)

Figure 3.3: Probabilities table

Attribute [Movie A]	Class attribute [Movie F]					
	0	1	2	3	4	5
0	0	0	0.2	0	0	0
1	0.5	0.75	0.2	0.6667	0.5	0
2	0	0.25	0.4	0	0.25	0
3	0	0	0	0	0	0.5
4	0	0	0.2	0.3333	0.25	0.5
5	0.5	0	0	0	0	0

Figure 3.4: Probabilities table of movie A

Attribute [Movie B]	Class attribute [Movie F]					
	0	1	2	3	4	5
0	0	0.25	0	0	0	0
1	0.5	0	0	0	0	0
2	0.5	0.75	0.6	0.3333	0.25	0
3	0	0	0.2	0	0.5	1
4	0	0	0	0	0	0
5	0	0	0.2	0.6667	0.25	0

Figure 3.5: Probabilities table of movie B

Attribute [Movie C]	Class attribute [Movie F]					
	0	1	2	3	4	5
0	0	0	0.2	0.3333	0	0
1	0	0	0	0	0	0
2	0.5	0.25	0.4	0.6667	0.25	0.5
3	0.5	0.5	0	0	0.25	0
4	0	0.25	0.4	0	0	0.5
5	0	0	0	0	0.5	0

Figure 3.6: Probabilities table of movie C

		Class attribute [Movie F]					
		0	1	2	3	4	5
Attribute [Movie D]	0	0	0	0	0	0	0
	1	0	0	0.4	0.3333	0.25	0
	2	0	0.5	0.2	0	0.25	0
	3	0.5	0.5	0.2	0.3333	0.5	0.5
	4	0.5	0	0.2	0	0	0.5
	5	0	0	0	0.3333	0	0

Figure 3.7: Probabilities table of movie D

		Class attribute [Movie F]					
		0	1	2	3	4	5
Attribute [Movie E]	0	0	0.25	0	0	0	0
	1	0.5	0.5	0.4	0.6667	0.25	0
	2	0	0	0.2	0.3333	0.75	0.5
	3	0	0	0.2	0	0	0
	4	0.5	0	0	0	0	0.5
	5	0	0.25	0.2	0	0	0

Figure 3.8: Probabilities table of movie E

Naïve Bayes classifier use formula Fig. 3.6 for prediction. To find $P(A = movie_a, movie_b, ..., movie_i \mid B = movie_j)$ value using conditional independence assumption, $P(A = movie_a, movie_b, ..., movie_i \mid B = movie_j)$, the joint model can be expressed as

$$P(A = movie_a \mid B = movie_j) \cdot P(movie_b \mid B = movie_j) \cdot ... \cdot P(movie_i \mid B = movie_j)$$

so $P(movie_i \mid B = movie_j)$, and for each $P(movie_i \mid B = movie_j)$, it can be retrieved for each specific movie Fig. 3.4, 3.8 such as movie 2 using table Fig. 3.5. And using the $P(B = movie_b)$ value from table Fig. 3.2.

For example, using Naïve Bayes classifier to predict the possible ratings that the user 21 give to class F. It takes existing ratings of user 21 into account 3.9, and, for each $P(movie_{a,...,e} \mid B = movie_f)$ and $P(B = movie_f)$ can be retrieved from the previous probabilistic tables such as $P(movie_a = 1 \mid movie_f = 0)$ using table Fig. 3.4 where row is 1 and column is 0, so $P(movie_1 = 1 \mid movie_f)$ equal to 0.5, and $P(movie_f) = 0$ equal to 0.1 from Fig. 3.2. Therefore, Naïve Bayes classifier apply existing inputs using formula Fig. 3.6 with every possible ratings, so it results in equation 3.9. As a result, the ratings 0 can be calculated from

$$P(movie_a = 2 \mid movie_f = 0) \cdot P(movie_b = 3 \mid movie_f = 0) \cdot$$
$$P(movie_c = 2 \mid movie_f = 0) \cdot P(movie_d = 3 \mid movie_f = 0) \cdot$$
$$P(movie_e = 1 \mid movie_f = 0) \cdot P(movie_f = 0) = (0)(0)(0.5)(0.5)(0.5)(0.1) = 0$$

then we applied the same way with other rating. For rating 4, probability rating is 0.00078125 which is the highest probabilistic value when it is compared to other ratings, so it can predict that the most possible ratings that user 21 give to movie class F is 4. However, Naïve Bayes can predict incorrectly when larger amount of data is applied.

Naïve Bayes Prediction [**predicted value** for user 21 of movie F.]						Predicted result	real result
User \ movie	A	B	C	D	E	F	F
user 21	2	3	2	3	1	4	4

Figure 3.9: Naïve bayes classifier prediction

3.2.2 Baseline Algorithm

For rating prediction, baseline estimates or baselines for predictive models is performance evaluation of given problem in statistics [40]. To applied baseline estimates [19], using formula , equation (3.7) , unknown rating of baseline estimates is defined by b_{ui}, μ is the mean of overall rating, b_u and b_i define the average of existing evidence of specific user u and item i sequentially.

$$b_{ui} = \mu + b_u + b_i \tag{3.7}$$

$$\mu = \frac{\sum_{r_{ui} \in R_{Train}} r_{ui}}{|R_{Train}|} \tag{3.8}$$

$$b_i = \frac{\sum_{r_{ui} \in U_i} (r_{ui} - \mu)}{|U_i|} \tag{3.9}$$

$$b_u = \frac{\sum_{r_{ui} \in I_i} (r_{ui} - \mu - b_i)}{|I_u|} \tag{3.10}$$

For equation μ (3.8), r_{ui} means The actual rating of user u for item i, and R_{train} is The training set. For equation b_i(3.9), r_{ui} is The actual rating of user u for item i, and U_i is The set of all users that have rated item i. For equation b_u (3.10), I_u is the set of all items rated by user u.

For example, to apply baseline estimates in previous example, mean,μ, in equation (3.7), can be calculated using formula equation (3.8) , first is to find average of whole data in table or summation of data in table and then it is divided by total number of data without considering "0" data. Thus, total bias of whole table for this example, mean is 2.5841 as shown in Fig. 3.11. Then, for b_i, bias of item i can be calculated using equation b_i (3.9) by the average rating of item i divided by total number of possible ratings of item i which each rating is subtracted by μ. Next, to calculate b_u, bias of user can be computed using equation b_u (3.10) with existing ratings of user predictor. It use the summation of rating of user u which each rating is subtracted by μ and bias of rating of user predictor sequentially. Then, it is divided by total number of possible ratings of user u, and then the result is subtracted by mean.

In this example, to find bias of movie A, in this example, ratings at column of movie A is 1, ... , 4, 1 so summation ratings of movie A is 41, note that zero number is not counted, and total number of possible ratings is 19. Therefore,

41 is divided by 19 which the result is 2.15789. Then, result is subtracted by mean, 2.15789, so its result is -0.42621 as shown in table Fig. 3.10 at second column last row.

Next, for prediction, formula equation (3.7) is applied for baseline prediction. mean of total table in Fig. 3.11 and b_i, bias of item i and b_u, bias of user u can be substituted into formula (3.7) , and the result is b_{ui} which is the possible value of rating item i of user u. To find unknown rating that user 21 probably give to movie F, in this example, To find bias value of user 21, it use the equation 3.10 and use the existing rating information of user 21 as shown in Fig. 3.12.

$$b_u = [(2 - 2.5841 - (-0.4262)) + (3 - 2.5841 - 0.258) + (2 - 2.5841 - 0.4159) + (3 - 2.5841 - 0.0659) + (1 - 2.5841 - (-0.4262))]/5$$

and the result is $= (-1.8079)/5 = -0.36158$. Then, for prediction, mean from Fig. 3.11 is 2.5841, from Fig. 3.10 , bias value of movie F is 0.1381. Those value is substituted into equation (3.7) and the result is $2.5841 + (-0.36158) + 0.1381 = 2.36062$ as shown Fig. 3.12, so possible rating is 2.36062;however, in this example, some result are incorrect prediction because the amount of input information may affect the efficiency of baseline prediction, for this example, predicted value of user 21 probably give to movie F is 2.36062 which actual ratings is 4.

User\Movie	A	B	C	D	E	F
1	1	2	5	3	2	4
2	1	2	3	2	5	1
3	2	2	2	3	1	2
4	5	1	3	4	1	0
5	1	3	5	3	2	4
6	1	2	4	3	0	1
7	1	2	2	3	1	1
8	4	3	2	3	2	5
9	4	5	2	5	2	3
10	2	5	2	1	2	4
11	3	3	4	4	4	5
12	4	3	3	2	1	4
13	1	2	2	3	1	3
14	1	5	0	1	1	3
15	2	2	4	2	1	2
16	2	0	3	2	1	1
17	1	2	2	3	4	0
18	4	2	0	4	5	2
19	0	3	4	1	2	2
20	1	5	2	1	3	2

	bais movieA	bais movieB	bais movieC	bais movieD	bais movieE	bais movieF
Bias value	-0.4262	0.258	0.4159	0.0659	-0.4262	0.1381

Figure 3.10: Data table and bias value of each movies

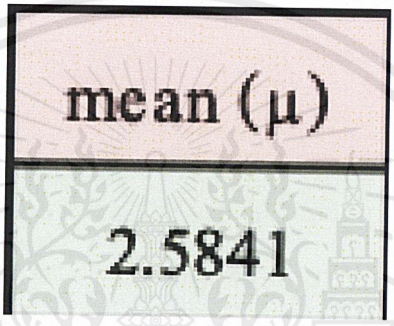


Figure 3.11: Mean of overall rating

Example baseline prediction [**predicted value** for user 21 of movie F.]						Predicted result	real result
User\Movie	A	B	C	D	E	F	F
user 21	2	3	2	3	1	2.36062	4

Figure 3.12: Result of baseline prediction

Chapter 4
Our Approach

We propose a framework for combining the Naïve Bayes Classifier with the Baseline Algorithm. Let $\hat{R}_{NBC}$, $\hat{R}_{BL}$ and $\hat{R}_{Final}$ represent the predictions rating generated by the Naïve Bayes Classifier, Baseline Algorithm, and the final prediction we are confident to be accurate, respectively. The proposed approach is outlined in Algorithm 1 as follow.

Algorithm 1 Combining Naïve Bayes classifier with baseline algorithm

```
1: procedure PREDICT(R̂_NBC, R̂_BL)
2:     if ( |R̂_NBC − R̂_BL| < β ) then
3:         R̂_Final ← R̂_NBC.
4:     else
5:         R̂_Final ← R̂_BL.
6:     return R̂_Final
```

The threshold parameter β tells us if the difference between the predictions given by the individual recommender systems is small enough, then we are confident that Naïve Bayes Classifier can predict a rating correctly. This is a kind of heuristic approach learned from the prediction behavior of Baseline Algorithm and Naïve Bayes Classifier. Baseline Algorithm gives $\hat{R}_{BL}$ in a floating point scale, and Naïve Bayes Classifier gives $\hat{R}_{NBC}$ in an integer point scale. Baseline recommender systems give an accurate recommendation, but regularly they do not predict actual value, for example, if the actual value of an unknown rating is 3, then $\hat{R}_{BL}$ might be 2.9 or 3.1 (or some other value). Conversely, $\hat{R}_{NBC}$ can give actual value, for example, in the case, as mentioned earlier, it might give us 3. However, Naïve Bayes Classifier might result in the prediction that is not close to the actual one, e.g., 2, 1. We take the difference of individual recommender's predictions, and if it is less than a threshold β, then we use $\hat{R}_{NBC}$ assuming that it has been predicted correctly. Otherwise, we use the $\hat{R}_{BL}$ given by the Baseline Algorithm.

Chapter 5
Implementation

In this chapter, the implementation of our system for the proposed approach are discussed.

5.1 Programming Language and Tools

To test our proposed approach. We tested using the JupyterLab [18, 25] that run on the Google Cloud Platform (GCP). The tools and technology used are as follows.

5.1.1 Python

Python is a general-purpose, interpreted high-level programming language whose design philosophy emphasizes code readability [31]. Its syntax is clear and expressive. Python has an extensive and comprehensive standard library and more than ten thousand extension modules. We use Python version 3 for developing every part of the systems. The other Python modules are discussed as follows.

5.1.2 NumPy

NumPy is a software library for written the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to run on these arrays [28, 37].

5.1.3 Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis [23]. Precisely, it offers data structures and operations for handling numerical tables and time series. It gives a wide range of features including DataFrame object for data manipulation with integrated indexing, tools for reading and

writing data between in-memory data structures and various file formats, data alignment and integrated handling of missing data and further.

5.1.4 NLTK

The Natural Language Processing Toolkit (NLTK) is an open source language processing module of human language in Python [3]. Created in 2001 as a part of computational linguistics course in the Department of Computer and Information Science at the University of Pennslyvania. NLTK provides inbuilt support for easy-to-use interfaces over fifty lexicon corpora. Along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. For our project, we use `NaiveBayesClassifier` module to train and classify the dataset. NLTK was designed with four goals in mind.

1. Simplicity: Provide and intuitive framework along with substantial building blocks, giving users a practical knowledge of NLP without getting bogged down in the tedious house-keeping usually associated with processing annotated language data.
2. Consistency: Provide a uniform framework with consistent interfaces and data structures and simply guessable method names.
3. Extensibility: Provide a structure into which new software modules can easily be accommodated, including alternative implementations and competing approaches on the same task.
4. Modularity: Provide components that can be used independently without requiring to understand the rest of the toolkit.

5.1.5 Surprise

Surprise is a Python scikit-learn building and analyzing recommender systems [15, 29]. The name SurPRISE stands for Simple Python RecommendatIon System Engine. It was designed with the following four purposes in mind:

1. Give perfect control over the experiments. To this point, a strong emphasis is laid on documentation, which they have tried to make as clear and precise as possible by pointing out every detail of the algorithms.
2. Alleviate the pain of Dataset handling. Users can use both built-in datasets and their custom datasets.

3. Make it simple to implement new algorithm ideas.

4. Provide tools to evaluate, analyze, and compare the performance of the algorithms. Cross-validation procedures can be run very easily using powerful CV iterators (inspired by scikit-learn great tools), as well as exhaustive search over a set of parameters.

5.2 Implement the Model

5.2.1 Data Preprocessing

We perform data preprocessing to create the ratings matrix (see Fig. 5.1) and prepare it for the training and testing. First, we load the data from a delimited text file. Each row of the file after the header row represents one rating of one movie by one user. Next, we establish a zero-indexed set of unique IDs for users and movie items. This guarantees that a unique ID corresponds to a specific row and column indexes of the ratings matrix. The MovieLens 100k dataset uses one-based IDs, where the lowest index of the unique set is one. To normalize, we subtract one from each index. Next, we create a Pandas DataFrame matrix that contains the user and item IDs and ratings. Finally, we randomly split the ratings matrix into a training matrix and test matrix using K-fold cross-validation that describes in chapter: Experimentation.

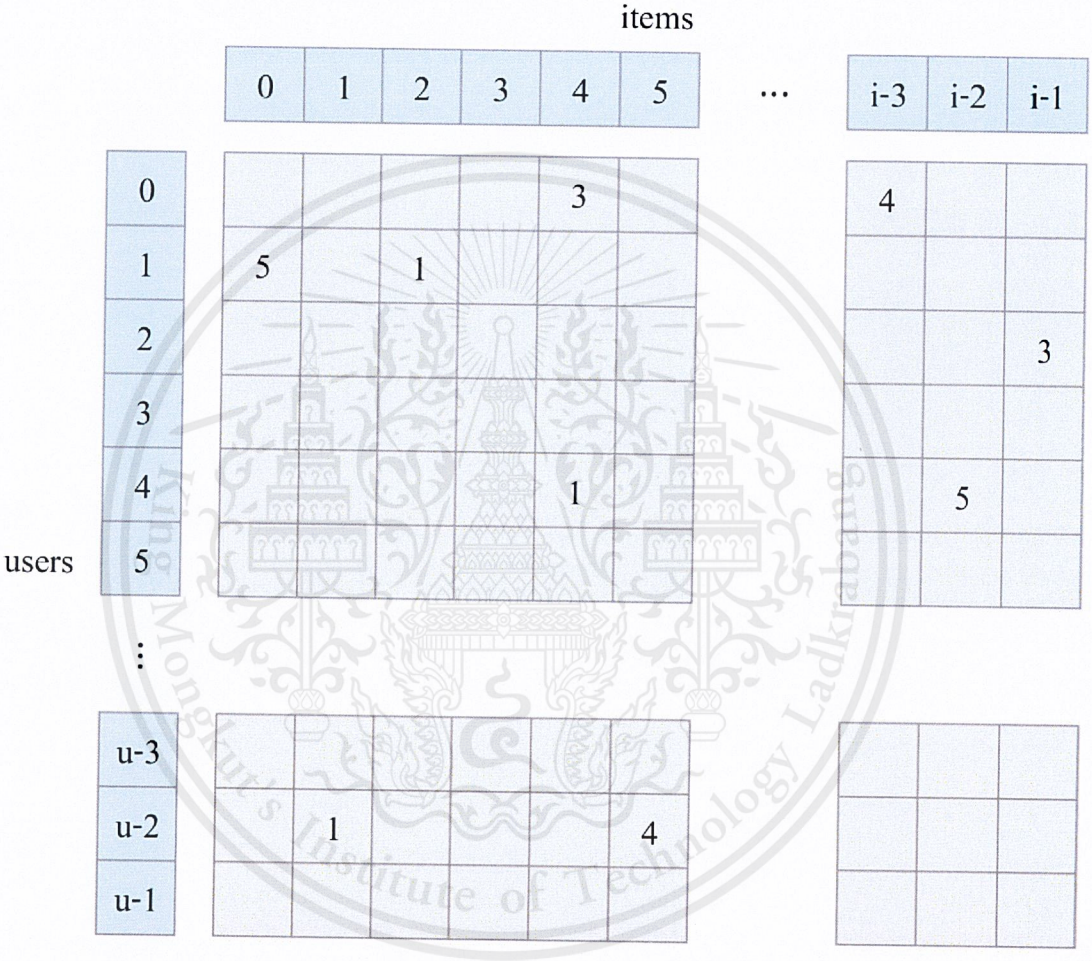


Figure 5.1: The MovieLens ratings matrix

5.2.2 Naïve Bayes Classifier

Training the model

To start, we use `NaiveBayesClassifier`, which is NLTK module, written in Python, and based on the Naïve Bayes algorithm. NLTK has many supported method and function, which helps to work with human language data.

In our recommender system, we have to predict the rating that can belong to several movie items. Due to every movie can be chosen as a class. So, we have to choose each movie items as a class. First, we start by choosing the first movie item to be a class (assume that first movie item = j), labeling the training dataset (we describe the labeling method below). Next, we fed the list of labeled featuresets that obtain from labeling method into the Naïve Bayes algorithm to generate a model. To this step we will get the model according to Fig. 5.2. The model in Fig. 5.2 contain the probability that can use to predict the rating for only single class (item j).

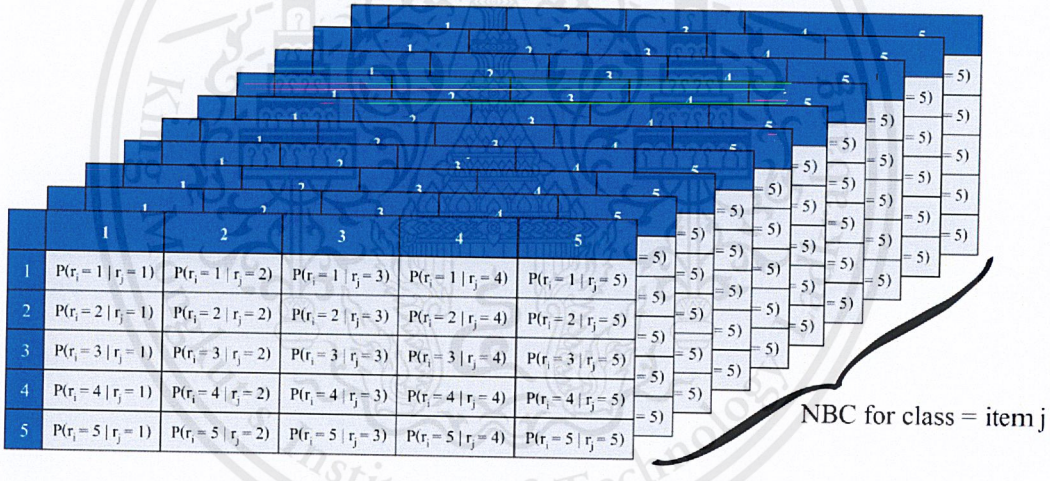


Figure 5.2: NBC model for class = item j

Next, we choose the next movie items to be a class and repeat the following step until the last move item. After all of this is done, we will have derived the Naïve Bayes predictive model that contain all trained Naïve Bayes classifier belong to each class, see Fig. 5.3, which it can use to predict the rating of all movie items in the training dataset. We can test our model with the test set.

Labeling the training dataset

The labeling training dataset is used to convert the training set to a list of labeled featuresets. Suppose we choose item j as the class. Each labeled featureset is pairs of featuresets and class labels, where featureset represented by a vector $u = (r_{u1}, r_{u2}, \ldots, r_{un})$ where $r_{uj} \notin u$, but r_{uj} is a class label of vector u.

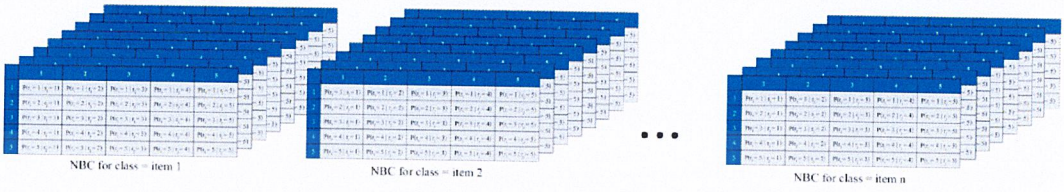


Figure 5.3: Naïve Bayes predictive model

Testing the model

The classification is based on a probability distribution between each given labeled featureset and class label. The Naïve Bayes generally use a decision rule as the equation (3.6). In which it will return the most appropriate label for the given featureset. But in our model, we have applied the logarithmic identities to the original equation (3.6), then we get the sums of logarithmic probabilities rather than products of probabilities to avoid underflow errors.

5.2.3 Baseline Algorithm

Training the model

In the classification, we have to classify on the unseen user rating data, so we compute just two parameters μ and b_i in the equation (3.7). First, the parameter μ defined as being equal to the sum of every rating values divided by the total number of rating values in the training dataset according to equation (3.8). Second, the parameter b_i, we initialize an empty list of size equal to the number of movie item in the training dataset. For each movie compute baseline by using equation (3.9) and then store the computed value in this list. We can use these two parameters together with parameter b_u that will compute in classification.

Testing the model

For each unseen user rating data, we compute the remaining parameter bu indicate the observed deviations of user u from the average, which it can calculate by the equation (3.10). Then we can predict the rating by substitute value of parameter μ and b_i that computed in the training step and parameter b_u into the equation (3.7) to estimate the predicted rating.

5.2.4 Combining Naïve Bayes Classifier with Baseline Algorithm

Training the model

We can train the model by using both methods that was mentioned in Section: training the model of Naïve Bayes Classifier and Baseline Algorithm. Both models will use together in the testing part.

Testing the model

We use both models individually to predict the unknown rating according to Section: testing the model of Naïve Bayes Classifier and Baseline Algorithm. Then we will obtain two predicted rating values. We pass these predicted rating as a parameter of Algorithm 1 and set the threshold parameter β. Finally, we follow the procedure of Algorithm 1 to get the final prediction value.

Chapter 6
Experimentation

6.1 Dataset

The MovieLens (ML) datasets was released by MovieLens web site [12]. GroupLens is a research lab at the Department of Computer Science and Technology at the University of Minnesota which specializes in recommender systems, online communities, mobile and ubiquitous technologies, digital libraries and local geographic information systems.

There are several sizes of the MovieLens datasets that are available. The version that is used for evaluation of the proposed approach is the stable benchmark MovieLens 100k dataset was released in the year 1998, in April. It contains exactly 100,000 ratings from 943 users on 1682 movie on the an integer scale 1 to 5 and each user rated at least 20 movie. MovieLens data set has been used in many research projects [34, 38, 39]. The sparsity of this dataset is approximately 93.7%, which makes it challenging to obtain optimal decision boundaries for the classifier [4]. It is computed as follows:

$$1 - \frac{non\ zero\ entries}{all\ entries} = 1 - \frac{100,000}{943 \times 1,682} \approx 0.937$$

6.2 Accuracy Metrics

Accuracy metrics measure to what extent a recommender system can predict ratings of users. These metrics are especially useful for systems that display the predicted ratings to their users. Since rated items have an order, accuracy metrics can also be used to measure a system's ability to rank items [13].

Let r_{ui}, $\hat{r}_{ui}$ and $\hat{R}$ represent the actual rating of user u for item i, the predicted rating of user u for item i and the set of predicted ratings, respectively.

6.2.1 Mean Absolute Error (MAE)

A widely used accuracy metric for recommender system is the mean absolute error or MAE. Mean absolute error takes the sum of the absolute difference between the user's rating and the predicted rating and divides it by the number of items considered as shown in equation (6.1).

$$MAE = \frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} |r_{ui} - \hat{r}_{ui}| \tag{6.1}$$

6.2.2 Root Mean Squared Error (RMSE)

Many variations of mean absolute error exist, such as mean squared error (MSE), root mean squared error (RMSE) and normalized MAE (NMAE). The mean squared error measure emphasizes large errors by squaring each individual error, as shown in equation (6.2). RMSE is simply defined as the root of MSE.

$$RMSE = \sqrt{\frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2} \tag{6.2}$$

These metrics include Mean Absolute Error (MAE), Mean Square Error (MSE), Root Mean Square Error (RMSE), and Normalized Mean Absolute Error (NMAE) have been used in research projects such as [32, 33, 34, 35]

6.3 Evaluation Methodology

In order to estimate the quality of a recommender system, we need to properly partition the dataset into a training dataset and a test dataset. It is imperative that performance is estimated on data which take no part in the formulation of the model. Some learning schemes also need a validation set in order to optimize the model parameters. Our dataset is split according to k-fold cross-validation methods.

K-fold cross-validation consists in dividing the dataset into k independent folds (so that folds do not overlap). In turn, each fold is used exactly once as a test dataset and the remaining folds are used for training the model, see Fig. 6.1. According to [26, 41], the suggested number of folds is 10. This technique is suitable to evaluate the recommending capability of the model when new users (i.e., users do not already belong to the model) join the system. By choosing a reasonable number of folds, we can compute mean, variance, and confidence interval.

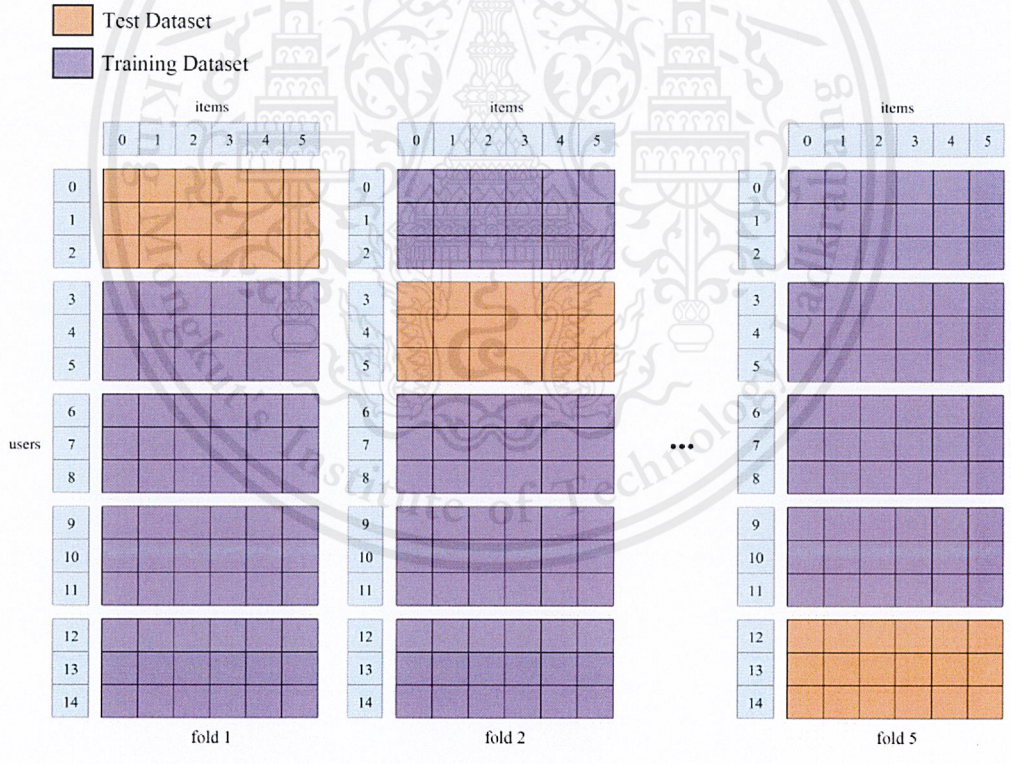


Figure 6.1: K-fold cross-validation where K = 5

6.4 Experimental Setup

We evaluated the results using the Mean Absolute Error (MAE) metric and as we mentioned in the section: Evaluation Methodology, we conducted a 10-fold cross validation of our experiments by randomly choosing different training and test sets each time and taking the average of the MAE.

6.5 Experimental Results

We compared our approach (CB_{NBCBL}) with Naïve Bayes Classifier (NBC) and Baseline Algorithm (BL). The results are shown in the Table 6.1.

Table 6.1: A comparison of the proposed approach in terms of Mean Absolute Error (MAE) using 10-folds cross-validation

Fold	**NBC**	**BL**	**CB_{NBCBL}**
1^{st}	0.835 360	0.746 788	0.721 512
2^{nd}	0.832 879	0.736 853	0.714 260
3^{rd}	0.918 512	0.727 034	0.703 234
4^{th}	0.832 722	0.733 981	0.704 408
5^{th}	0.841 066	0.714 013	0.684 198
6^{th}	0.836 206	0.750 463	0.719 692
7^{th}	0.920 627	0.772 334	0.744 782
8^{th}	0.867 565	0.790 459	0.757 416
9^{th}	0.928 492	0.720 074	0.688 012
10^{th}	0.912 114	0.744 279	0.712 199
Mean MAE	0.872 554	0.743 628	0.714 971

6.6 Optimal Value of Parameter β

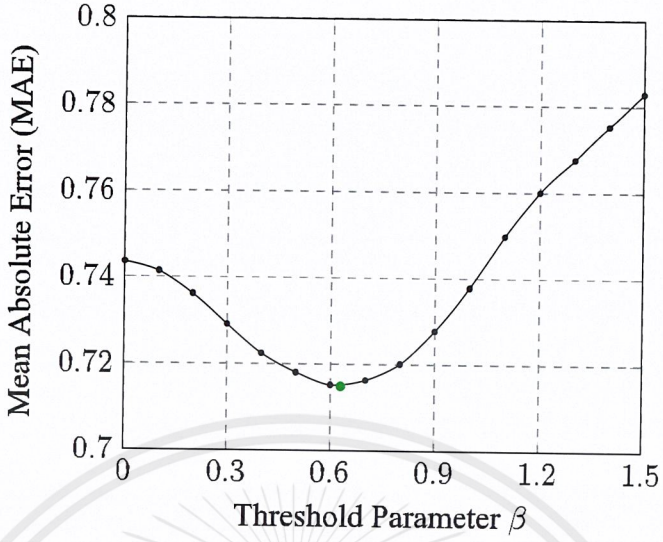


Figure 6.2: Determining the optimal value of parameter β

For determining the optimal value of β, we varied the value of β from 0 to 1.5 with a difference of 0.1. Then we found that the lowest MAE is in the range from 0.6 to 0.7. So again, we varied with the value of β from 0.6 to 0.7 with a difference of 0.01. The results are shown in Fig. 6.2 shows that $\beta = 0.63$ gave the lowest MAE for 100k MovieLens dataset

6.7 Using Confidence Intervals to Compare the Recommender Models

A fair comparison among competing recommender systems should ideally test on the same data, with a sample size that is large enough to produce reasonably robust verification scores. In many studies, Confidence Intervals (CI) [24] have been computed on the scores and examined to see whether they overlap to measure the significance of scores at the confidence interval. So we compute the confidence interval of our approach, Naïve Bayes Classifier (NBC), and Baseline Algorithm (BL) to check whether our recommender system performs significantly better than other using equation (6.3).

$$\bar{x} - t_{\alpha/2,n-1}\frac{s}{\sqrt{n}} \leq \mu \leq \bar{x} + t_{\alpha/2,n-1}\frac{s}{\sqrt{n}} \tag{6.3}$$

$\bar{x}$ and s are the mean and sample standard deviation of a random sample from a normal distribution with unknown variance σ^2, $100(1-\alpha)\%$ confidence interval on μ

is given by equation (6.3) where where $t_{\alpha/2,n-1}$ is the upper $100\alpha/2$ percentage point of the t distribution with n–1 degrees of freedom.

We decided to compute 90% Confidence Interval ($\alpha = 0.1$), since n = 10, we have n – 1 = 9 degrees of freedom for t distribution and using MAE value from Table 6.1

6.7.1 CI on the mean MAE value for the NBC model

The sample mean is $\bar{x} \approx 0.872554$, and the sample standard deviation is $s \approx 0.042162$, The resulting CI is computed as follows:

$$\bar{x} - t_{\alpha/2,n-1}\frac{s}{\sqrt{n}} \leq \mu \leq \bar{x} + t_{\alpha/2,n-1}\frac{s}{\sqrt{n}}$$

$$0.872554 - t_{0.1/2,10-1}\frac{0.042162}{\sqrt{10}} \leq \mu \leq 0.872554 + t_{0.1/2,10-1}\frac{0.042162}{\sqrt{10}}$$

$$0.872554 - 0.024441 \leq \mu \leq 0.872554 + 0.024441$$

$$0.848114 \leq \mu \leq 0.896995$$

Therefore, we are 90% CI that the mean MAE of NBC model is between 0.848114 and 0.896995

6.7.2 CI on the mean MAE value for the BL model

The sample mean is $\bar{x} \approx 0.743628$, and the sample standard deviation is $s \approx 0.023382$, The resulting CI is computed as follows:

$$\bar{x} - t_{\alpha/2,n-1}\frac{s}{\sqrt{n}} \leq \mu \leq \bar{x} + t_{\alpha/2,n-1}\frac{s}{\sqrt{n}}$$

$$0.743628 - t_{0.1/2,10-1}\frac{0.023382}{\sqrt{10}} \leq \mu \leq 0.743628 + t_{0.1/2,10-1}\frac{0.023382}{\sqrt{10}}$$

$$0.743628 - 0.013554 \leq \mu \leq 0.743628 + 0.013554$$

$$0.730074 \leq \mu \leq 0.757182$$

Therefore, we are 90% CI that the mean MAE of BL model is between 0.730074 and 0.757182

6.7.3 CI on the mean MAE value for the CB_{NBCBL} model

The sample mean is $\bar{x} \approx 0.714971$, and the sample standard deviation is $s \approx 0.022820$, The resulting CI is computed as follows:

$$\bar{x} - t_{\alpha/2,n-1}\frac{s}{\sqrt{n}} \leq \mu \leq \bar{x} + t_{\alpha/2,n-1}\frac{s}{\sqrt{n}}$$

$$0.714971 - t_{0.1/2,10-1}\frac{0.022820}{\sqrt{10}} \leq \mu \leq 0.714971 + t_{0.1/2,10-1}\frac{0.022820}{\sqrt{10}}$$

$$0.714971 - 0.013228 \leq \mu \leq 0.714971 + 0.013228$$

$$0.701743 \leq \mu \leq 0.728199$$

Therefore, we are 90% CI that the mean MAE of CB_{NBCBL} model is between 0.701743 and 0.728199

6.7.4 Comparing Confidence Intervals

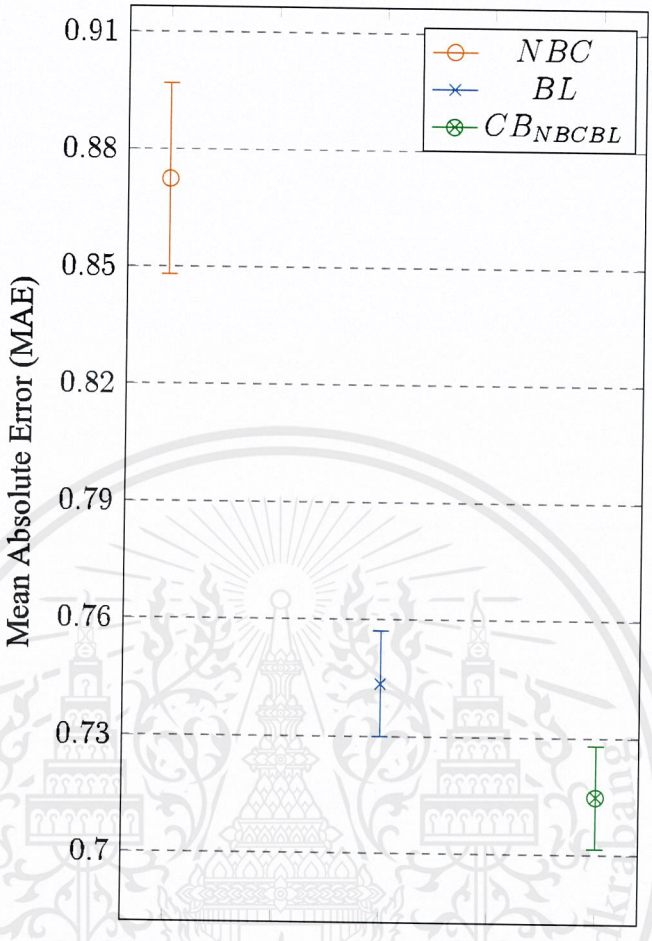


Figure 6.3: 90% confidence interval plot for the mean MAE of each model

The difference between the three mean values of MAE, our model having the best value (lowest). The 90% confidence intervals on the mean MAE values for the three models shown that our model did not overlap with the other two models, see Fig 6.3.(range of 0.848114 to 0.896995 for NBC model, range of 0.730074 to 0.757182 for BL model and range of 0.701743 to 0.728199 for our model). We can conclude that the MAE of our model is statistically significantly improved from the MAE of both NBC model and the BL model at the 90% significance level.

Chapter 7
Conclusion

In this thesis, we have proposed the new recommendation approach by combining Naïve Bayes Classifier with Baseline Algorithm. We empirically show that our recommendation approach outperforms others in terms of accuracy, and is more scalable. Our results are well performed when it is compared with Naïve Bayes results, and this result can fulfill the gap of lacking local information problem as discussed in this article and alleviate the strong independent assumption. Thus, the result meets our objective, which is to improve the efficiency of Naïve Bayes using baseline algorithm.

As future work, we would like to apply Support Vector Machines over features vectors for generating recommendations. Furthermore, we would like to evaluate our algorithms on the dataset of domains other than movies, such as Book-Crossing Dataset [42].

Bibliography

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions", *IEEE Trans. on Knowl. and Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005, issn: 1041-4347, doi: 10.1109/TKDE.2005.99.

[2] R. Bergmann, K.-D. Althoff, M. Minor, M. Reichle, and K. Bach, "Case-based reasoning - introduction and recent developments", *Künstliche Intelligenz*, vol. 1, pp. 5–11, Jan. 2009.

[3] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, 1st. O'Reilly Media, Inc., 2009, isbn: 0596516495, 978-0596516499.

[4] J. Bissmark and O. Wärnling, "The sparse data problem within classification algorithms: The effect of sparse data on the naïve bayes algorithm", May 2017.

[5] Y. Blanco-Fernández, J. J. Pazos-Arias, A. Gil-Solla, M. Ramos-Cabrer, M. López-Nores, J. García-Duque, A. Fernández-Vilas, R. P. Díaz-Redondo, and J. Bermejo-Muñoz, "A flexible semantic inference methodology to reason about user preferences in knowledge-based recommender systems", *Know.-Based Syst.*, vol. 21, no. 4, pp. 305–320, May 2008, issn: 0950-7051.

[6] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey", *Knowl.-Based Syst.*, vol. 46, pp. 109–132, 2013.

[7] R. Burke, "Hybrid recommender systems: Survey and experiments", *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, Nov. 2002, issn: 1573-1391, doi: 10.1023/A:1021240730564.

[8] F. Cacheda, V. Carneiro, D. Fernández, and V. Formoso, "Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems", *ACM Trans. Web*, vol. 5, no. 1, 2:1–2:33, Feb. 2011, issn: 1559-1131, doi: 10.1145/1921591.1921593.

[9] W. Carrer-Neto, M. L. Hernández-Alcaraz, R. Valencia-García, and F. García-Sánchez, "Social knowledge-based recommender system. application to the movies domain", *Expert Systems with Applications*, vol. 39, no. 12, pp. 10 990–11 000, 2012, issn: 0957-4174, doi: 10.1016/j.eswa.2012.03.025.

[10] P. Domingos and M. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss", *Machine Learning*, vol. 29, no. 2, pp. 103–130, Nov. 1997, issn: 1573-0565, doi: 10.1023/A:1007413511361.

[11] C. Gaurav. All about naive bayes, [Online]. Available: https://towardsdatascience.com/all-about-naive-bayes-8e13cef044cf (visited on May 2019).

[12] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context", *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, 19:1–19:19, Dec. 2015, issn: 2160-6455, doi: 10.1145/2827872.

[13] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems", *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, Jan. 2004, issn: 1046-8188, doi: 10.1145/963770.963772.

[14] How naive bayes algorithm works?, [Online]. Available: https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-works-with-example-and-full-code (visited on May 2019).

[15] N. Hug, *Surprise, a Python library for recommender systems*, http://surpriselib.com, 2017.

[16] D. Jurafsky and J. Martin, "Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition", unpublished, [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf.

[17] R. Katarya and O. P. Verma, "An effective collaborative movie recommender system with cuckoo search", *Egyptian Informatics Journal*, vol. 18, no. 2, pp. 105–112, 2017, issn: 1110-8665, doi: 10.1016/j.eij.2016.10.002.

[18] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter notebooks – a publishing format for reproducible computational workflows", in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. 2016, pp. 87–90, doi: 10.3233/978-1-61499-649-1-87.

[19] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering", *ACM Trans. Knowl. Discov. Data*, vol. 4, no. 1, 1:1–1:24, Jan. 2010, issn: 1556-4681, doi: 10.1145/1644873.1644874.

[20] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, "Facing the cold start problem in recommender systems", *Expert Systems with Applications*, vol. 41, no. 4, Part 2, pp. 2065–2073, 2014, issn: 0957-4174, doi: 10.1016/j.eswa.2013.09.005.

[21] P. Lops, M. de Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends", in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Boston, MA: Springer US, 2011, pp. 73–105, isbn: 978-0-387-85820-3, doi: 10.1007/978-0-387-85820-3_3.

[22] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments", *Decis. Support Syst.*, vol. 74, no. C, pp. 12–32, Jun. 2015, issn: 0167-9236.

[23] W. McKinney, "Data structures for statistical computing in python", in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 51–56.

[24] D. C. Montgomery and G. C. Runger, "Applied statistics and probability for engineers", in, 6th ed. John Wiley Sons Inc., Nov. 2013, ch. 8, pp. 271–304.

[25] A. Morin, B. Eisenbraun, J. Key, P. C. Sanschagrin, M. A. Timony, M. Ottaviano, and P. Sliz, "Cutting edge: Collaboration gets the most out of software", *eLife*, vol. 2, Sep. 2013, doi: 10.7554/eLife.01456.

[26] R. O Duda, P. E Hart, and D. G.Stork, "Pattern classification", in. Jan. 2001, vol. xx, isbn: 0-471-05669-3.

[27] G. Obunadike, A. Isah, and J. Alhassan, "Optimized naïve bayesian algorithm for efficient performance", Jan. 2018.

[28] T. E. Oliphant, *Guide to NumPy*, 2nd. USA: CreateSpace Independent Publishing Platform, 2015, isbn: 151730007X, 9781517300074.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, and G. Louppe, "Scikit-learn: Machine learning in python", *Journal of Machine Learning Research*, vol. 12, Jan. 2012.

[30] I. Rish, "An empirical study of the naïve bayes classifier", *IJCAI 2001 Work Empir Methods Artif Intell*, vol. 3, Jan. 2001.

[31] G. van Rossum, "Python reference manual", Amsterdam, Netherlands, Tech. Rep., 1995.

[32] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithm for collaborative filtering", *UAI*, Jan. 2013.

[33] B. Sarwar, G. Karypis, J. Konstan, and J. T. Riedl, "Application of dimensionality reduction in recommender system – a case study", Aug. 2000.

[34] B. Sarwar, G. Karypis, J. Konstan, and J. T. Riedl, "Item-based collaborative filtering recommendation algorithmus", Jan. 2001.

[35] B. Sarwar, G. Karypis, J. Konstan, and J. T. Riedl, "Recommender systems for large-scale e-commerce scalable neighborhood formation using clustering", vol. 50, Jan. 2002.

[36] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques", *Adv. in Artif. Intell.*, vol. 2009, 4:2, Jan. 2009, issn: 1687-7470.

[37] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: A structure for efficient numerical computation", *Computing in Science Engineering*, vol. 13, no. 2, pp. 22–30, Mar. 2011, issn: 1521-9615, doi: 10.1109/MCSE.2011.37.

[38] M. G. Vozalis and K. G. Margaritis, "On the enhancement of collaborative filtering by demographic data", *Web Intelligence and Agent Systems*, vol. 4, pp. 117–138, Jan. 2006.

[39] M. G. Vozalis and K. G. Margaritis, "Using svd and demographic data for the enhancement of generalized collaborative filtering", *Information Sciences*, vol. 177, no. 15, pp. 3017–3037, 2007, issn: 0020-0255, doi: 10.1016/j.ins.2007.02.036.

[40] What is a baseline predictor model?, [Online]. Available: https://www.quora.com/What-is-a-baseline-predictor-model (visited on May 2019).

[41] I. H. Witten, E. Frank, and M. A. Hall, "Chapter 5 - credibility: Evaluating what's been learned", in *Data Mining: Practical Machine Learning Tools and Techniques*, ser. The Morgan Kaufmann Series in Data Management Systems, Third Edition, Boston: Morgan Kaufmann, 2011, pp. 147–187, isbn: 978-0-12-374856-0, doi: 10.1016/B978-0-12-374856-0.00005-5.

[42] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification", in *Proceedings of the 14th International Conference on World Wide Web*, ser. WWW '05, ACM, 2005, pp. 22–32, isbn: 1-59593-046-9, doi: 10.1145/1060745.1060754.