Book Recommender System Using Scikit-Surprise

Noreen Naz
Riphah International University
Lahore, Pakistan
noreennazmrd123@gmail.com

Umair Ashraf
Riphah International University
Lahore, Pakistan
umairashraf7566@gmail.com

Hadiqa Ilyas
Riphah International University
Lahore, Pakistan
hadiqailyas15@gmail.com

Abstract— A Recommender System (RS) is used to recommend/suggest a product to its user on the basis of its popularity, similarity, content and services etc. It filters the useful patterns from a given dataset and tries to predict those products which can be of user's interest. Recommender Systems are widely used in different areas like Movie Recommendations, Music Recommendations; Research articles Recommendations, Book Recommendations etc.
In this paper, different prediction algorithms from Python-Scikit Library are evaluated on the basis of Root Mean Square Error(RMSE) values, in order to provide a basis for implementation of a Book Recommender System.

Keywords—Book Recommender System, RMSE, Surprise-Scikit Library, SVD, SVDpp, NMF, BaselineOnly, NormalPredictor, KNNBasic, KNNWithMeans, KNNWithZScore, KNNBaseline, SlopeOne, CoClustering

I. INTRODUCTION

Recommender system is basically an algorithm, which analyses the data patterns in a data set and gives valuable suggestions to a user regarding any product of his/her choice etc. Recommender systems are very helpful in this age of Information Technology and E-Commerce, where online users have been given a lot of choices and opportunities either to buy a product or watch a movie. So it is strongly needed by online businesses that some valuable suggestions and recommendations regarding any product should be provided to its user so that he/she may do the correct decisions.
Recommender systems can be categorized into three main types:

- Simple Recommendation System:
  It works on the popularity or type of product mechanism. It recommends most popular top products or belonging to a specific category on need basis to its user.
- Content-based Recommendation System:
  These recommender systems provide suggestions regarding its user's past behavior and choices.
- Collaborative filtering Recommendation System:
  These systems works on the correlation mechanism between different users, i.e. for a given user, find other similar users whose ratings strongly correlate with the current user. Recommend items rated highly by these similar users, but not rated by the current user. Almost all existing commercial recommenders use this approach (e.g. Amazon).

In this paper, collaborative filtering approach has been applied using Python Scikit-Surprise Library [1] algorithms on a book rating dataset in order to get the recommendations for a typical Book Recommender System. RS is then evaluated using RMSE [2], error rate and test time measurements.

II. RELATED WORK

[3] have done a Research work on a movie and book recommendation system using Surprise Library. Performance is then evaluated using RMSE values.[4] have illustrated a comparison of product-based and user based collaborative Filtering approaches.
[5] have suggested hybrid approaches for Recommendation Systems very well. They have proposed a Book RS for mobile platform using hybrid approach. [6] have used three RS platforms LensKit, Mahout, and MyMediaLite. They have compared these platforms on the basis of prediction accuracy of a Book RS. [7] have worked on Music Recommender System. They have improved the performance of Music RS by applying both collaborative filtering technique and deep learning. [8] have proposed an improved Recommender system giving more true predictions using collaborative filtering with Jaccard Similarity. [9] have used multiple recommendation techniques after deeply analyzing different available datasets like Amazon book reviews etc. [10] have proposed a book RS using collaborative and association rule mining techniques.

III. RESEARCH METHODOLOGY

A. Surprise Recommendatin Kit and dataset used

This paper uses Python Scikit-Surprise Library ready-to-use prediction algorithms on a Book rating dataset in order to get the best possible algorithm for future book recommender system implementation. Following prediction algorithm are used:

1. BaselineOnly
   It is used to guess the baseline estimated value for any user/Product.
2. NormalPredictor
   It randomly generates prediction value i.e rating on the basis of training data.
3. KNNBasic
   This algorithm works on collaborative filtering approach.
4. KNNWithMeans
   This algorithm also works on collaborative filtering approach but also considering the average user's ratings.
5. KNNWithZScore
   This algorithm also works on collaborative filtering approach but also considering the normalized user's ratings.

6. KNNBaseline
This algorithm also works on collaborative filtering approach but also considering the baseline rating user's ratings.
7. SVD
SVD algorithm is basically a Probabilistic Matrix Factorization algorithm. Here baseline value is not used.
8. SVDpp
The SVDpp is same as SVD but it considers implicit ratings.
9. SlopeOne
SlopeOne is also a collaborative filtering algorithm doing accurate predictions.
10. NMF
NMF algorithm is basically a collaborative filtering algorithm but it takes into account Non-negative Matrix Factorization approach.
11. CoClustering
It is also a collaborative filtering algorithm but taking into account co-clustering approach too.

Above mentioned algorithms from Surprise Library have been used, checking each one's Root Mean Square Error(RMSE) values and compared accordingly to choose the best one for further processing.
Nicolas Hug [11] build a Recommendation System Library called Surprise, in order to facilitate the development of a typical recommendation system.
Surprise is a part of Python-Scikit package having a number of different algorithms in order to develop and explore the Recommendation Systems. The Surprise scikit package installation is done very easily through a simple pip command.
Book-Crossing[12] is a book rating dataset being developed by Cai-Nicolas Ziegler. This dataset has three tables: BX-Books, BX-Books-Ratings and BX-Users. Book ratings are from 1 to 10. There are almost 1.1 million book ratings of 270K books, being rated by 90K users.

B. Software Used

Kaggle is a platform to perform analytics and predictive modeling, arrange competitions related to Data Science around the world. Moreover it has powerful tools and resources to test our projects. Python language is used for project coding.

C. Project Steps explained:

Main purpose of this project was to evaluate the working of chosen prediction algorithms from Scikit-surprise library for Book-crossing dataset and then select the best one based on RMSE value. All Steps of the project explained here:

Step 1: Import required libraries and Load the DATASET

Step 2: Data preprocessing (Filling missing values, check the shape, description, a number of unique value, columns and analyze to get more insights from our data.

Step 3: Exploratory Data Analysis & Data Pre-processing

Rating Distribution Graph:

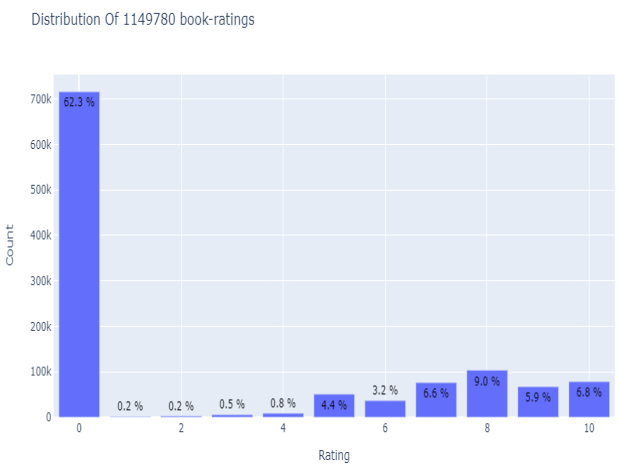

Fig. 1. Rating Distribution Graph

We can see that over 62% of all ratings in the data are 0, and very few ratings are 1 or 2, or 3. It can be seen in Fig-1 graph, that most of book ratings are 0 and a small number of book ratings are 1, 2, and 3.

Book-wise Rating Distribution Graph:

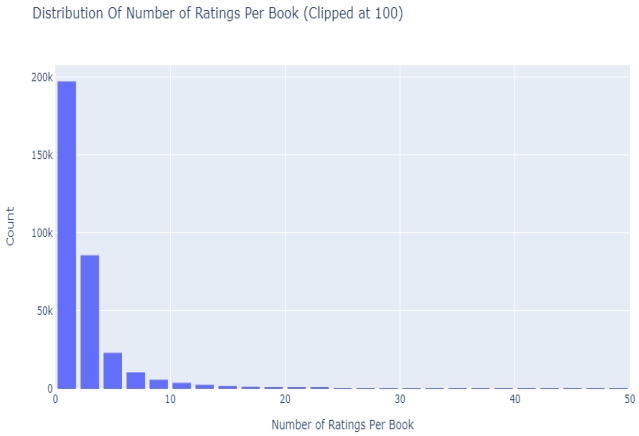

Fig. 2. Book-wise Rating Distribution Graph

Books ISBN wise rating sorted list:

	ISBN	bookRating
247408	0971880107	2502
47371	0316666343	1295
83359	0385504209	883
9637	0060928336	732
41007	0312195516	723
101670	044023722X	647
166705	0679781587	639
28153	0142001740	615
166434	067976402X	614
153620	0671027360	586

Fig. 3. ISBN-wise rating sorted list

It is obvious from Fig. 2 and Fig.3 that, users have given a huge number of per-book ratings < 5. A small number of books have been rated good and highest value for a book rating is 2502.

User-wise Rating Distribution Graph:

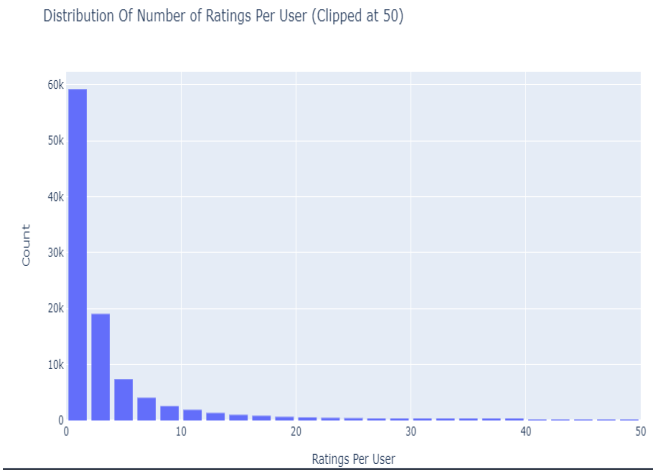

Fig. 4. User-wise Rating Distribution Graph

User-wise Book rating sorted list:

	userID	bookRating
4213	11676	13602
74815	198711	7550
58113	153662	6109
37356	98391	5891
13576	35859	5850
80185	212898	4785
105111	278418	4533
28884	76352	3367
42037	110973	3100
88584	235105	3067

Fig. 5. User-wise Rating Sorted List

It is obvious from Fig.4 and Fig. 5 too that, very few book readers have given many book ratings.
Fig. 1 and Fig. 2 clearly show a significant decline of rating values.
Setting 50 as minimum threshold for both book-rating and user-number-of-rating. And then filtering data accordingly, we got following data head, as shown in Fig. 6.

	userID	ISBN	bookRating
394	243	0060915544	10
395	243	0060977493	7
401	243	0316601950	9
405	243	0316776963	9
406	243	0316899984	7

Fig. 6. Merged Data Head of User and Book Rating

Step4: Applying Surprise Library Prediction Algorithms

IV. RESULTS AND COMPARATIVE ANALYSIS

The Fig.7 shows the RMSE values for the various prediction algorithms applied for modeling the Book RS.

Algorithm	test_rmse	fit_time	test_time
SVD	1.546933	3.394856	0.221207
SVDpp	1.558233	30.586062	1.193151
BaselineOnly	1.559250	0.131542	0.129299
CoClustering	1.644062	1.683903	0.147371
KNNWithMeans	1.698192	0.554907	1.032278
KNNBaseline	1.701614	0.674156	1.139271
KNNWithZScore	1.702145	0.701493	1.164115
SlopeOne	1.818636	0.425736	0.886646
KNNBasic	1.837948	0.532462	1.048465
NormalPredictor	2.237659	0.082705	0.250159
NMF	2.547487	4.048031	0.155086

Fig. 7. RMSE values table for Book RS

Best predictions Error rate values are shown in Fig.8, depicting a value of zero.
Worst predictions Error rate values are shown in Fig.9.

	uid	iid	rui	est	details	lu	Ui	err
2399	164581	0590353403	9.0	9.0	{'was_impossible': False}	7	51	0.0
6969	30735	0070212570	9.0	9.0	{'was_impossible': False}	38	35	0.0
693	220688	0446608955	9.0	9.0	{'was_impossible': False}	27	32	0.0
9155	98391	0316666343	9.0	9.0	{'was_impossible': False}	75	173	0.0
9712	22625	0786868716	9.0	9.0	{'was_impossible': False}	38	66	0.0
1827	144318	0312971346	9.0	9.0	{'was_impossible': False}	24	37	0.0
4197	110912	0060199652	9.0	9.0	{'was_impossible': False}	46	13	0.0
1821	2033	0590353403	9.0	9.0	{'was_impossible': False}	9	51	0.0
4966	11676	0385504209	9.0	9.0	{'was_impossible': False}	741	145	0.0
5470	23571	0609608444	9.0	9.0	{'was_impossible': False}	9	12	0.0

Fig. 8. Best Prediction Error Rate Table

	uid	iid	rui	est	details	Iu	Ui	err
605	66443	0142001740	2.0	8.842595	{'was_impossible': False}	5	96	6.842595
8626	162738	0440224675	1.0	7.876985	{'was_impossible': False}	25	31	6.876985
6862	11676	0440213991	1.0	7.911695	{'was_impossible': False}	741	10	6.911695
6569	124322	0743456866	1.0	7.980996	{'was_impossible': False}	1	5	6.980996
2497	8067	042518630X	2.0	9.000000	{'was_impossible': False}	29	15	7.000000
5900	251843	0316603287	1.0	8.033492	{'was_impossible': False}	33	18	7.033492
233	204474	1558743316	1.0	8.073827	{'was_impossible': False}	3	10	7.073827
9720	243930	0349101779	1.0	8.172049	{'was_impossible': False}	21	9	7.172049
8213	11676	0451207521	1.0	8.183357	{'was_impossible': False}	741	7	7.183357
8644	59269	0451161343	1.0	8.412019	{'was_impossible': False}	9	11	7.412019

Fig. 9. Worst Prediction Error Rate Table

Figure 4- Book RS Evaluation (RMSE) Graph

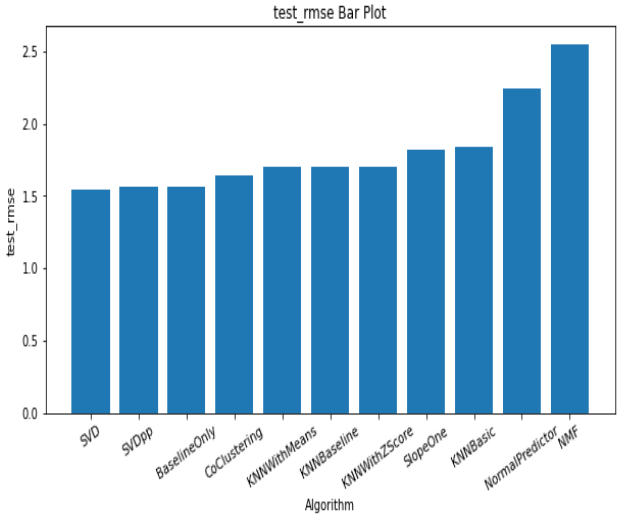

Fig. 10. Book RS Evaluation(RMSE) Graph

The comparative analysis concludes that SVD outperforms the other prediction algorithms.

V. CONCLUSION & FUTURE WORK

Recommender system applications are growing very fast, facilitating the online users to get the best suggestions regarding any required product. There are a lot of different approaches used in order to develop a typical recommender system. In this paper, python Scikit-Surprise library ready to use algorithms have been used and compared on the basis of RMSE values. Filtering the book rating patterns from Book-Crossing data set was made quite easy with the help of Surprise Recommender Library.

It is concluded that SVD algorithm had lowest RMSE rate and good test time. The error rates of SVD algorithm were found, for best predictions 0 and for worst predictions less than 10.

With the help of Surprise library, a researcher can easily find out best possible prediction algorithm for a typical recommender system to further apply on, hence saving his/her time and efforts significantly.

In the future, we will try to build a Recommender System using its three different approaches i.e Simple RS, Content-based RS, and Collaborative Filtering RS and then compare the performance /results of all these techniques.

REFERENCES

[1] Surprise Library, A Python scikit for recommender systems, "surpriselib.com"

[2] Evaluation Metrics", https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python

[3] Ananth G S, Raghuveer K: A Movie and Book Recommender System using Surprise Recommendation Kit

[4]. PETER BOSTROM, MELKER FILIPSSON:Comparison of User Based and Item Based Collaborative Filtering Recommendation Services

[5]. Muhammad Jabbar, Qaisar Javaid, Muhammad Arif, Asim Munir, Ali Javed: An Efficient and Intelligent Recommender System for Mobile Platform

[6]. Araek Tashkandi, Lena Wiese, Marcus Baum: Comparative Evaluation for Recommender Systems for Book Recommendations

[7]. Anand Neil Arnold, Vairamuthu S: Music Recommendation using Collaborative Filtering and Deep Learning

[8]. Avi Rana, K. Deeba Online Book Recommendation System using Collaborative Filtering (With Jaccard Similarity)

[9]. Jinny Cho Ryan Gorey Sofia Serrano Shatian Wang JordiKai Watanabe-Inouye: Book Recommendation System

[10]. Abhay E. Patil ; Simran Patil; Karanjit Singh; Parth Saraiya; Aayusha Sheregar: ONLINE BOOK RECOMMENDATION SYSTEM USING ASSOCIATION RULE MINING AND COLLABORATIVE FILTERING

[11]. Surprise, a Python scikit for building and analyzing recommender Systems(2017): http://nicolas-hug.com/project/surprise

[12] Book-Crossing dataset, http://www2.informatik.uni-freiburg.de/~cziegler/BX/