2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems

An Apache Spark-based Platform for Predicting The Performance of Undergraduate Student

Thong Le Mai, Phat Thanh Do, Minh Thanh Chung, Nam Thoai
Faculty of Computer Science & Engineering
Ho Chi Minh City University of Technology, Vietnam
Email: 1513293,1512400,ctminh,namthoai@hcmut.edu.vn

***Abstract*—Nowadays, Education Data Mining (EDM) plays a very important role in higher education institutions. Plenty of algorithms have been employed to measure student's GPA in the next semester's courses. The results can be used to early identify dropout students or help students choose the elective courses which are appropriate for them. The most widely used methods are machine learning, however, the problem is the accuracy which can be changed from dataset to dataset. More importantly, the performance of prediction models can be affected by the characteristic of dataset associated with the applied model. In this paper, we build a distributed platform on Spark to predict missing grades of elective courses for undergraduate students. The paper compares several methods that are based on the combination of Collaborative Filtering & Matrix Factorization (namely Alternative Least Square). We evaluate the performance of these algorithms using a dataset provided by Ho Chi Minh University of Technology (HCMUT). The dataset consists of information about undergraduate students from 2006 to 2017. Depending on the characteristics of our dataset, the paper highlights that Alternative Least Square with non-negative constraint achieves the better results than others in comparison.**

***Index Terms*—Educational Data Mining, Spark, prediction, student performance, distributed system, machine learning**

1. Introduction

Education Data Mining (EDM) is a research field which concerns data mining techniques to analyze patterns from data in educational context [1]. Currently, there are many learning systems that gather a large amount of educational data such as Learning Management Systems (LMS), Massive Open Online Courses (MOOCs). Following that, applications and tasks in EDM can be divided into different categories, depending on different properties. Based on the literature review in 2013 [2], we consider two main groups: "Student Modeling" and "Decision Support Systems" in terms of EDM. Concretely, predicting student's performance belongs to the "Student Modeling" group. The most widely used methods for this problem are regression and classification, but other methods have also been used such as clustering and feature selection.

The prediction has been one of the most dominant fields in EDM since 1995 [3]. Related researches usually exploit influential factors from the university's data to build a prediction model such as GPA, age, sex, etc. A lot of Machine Learning algorithms are used to solve these problems including Decision Tree, Random Forest, Regression, Neural Network [4] [5] [6]. Some others techniques frequently based on Recommender System (e.g., Collaborative Filtering, Matrix Factorization) are also found a lot of successes [7] [8] [9] [10] [11]. Furthermore, some studies focus on whether different types of predictor variables rather than the explicit ratings such as ages, sex, online time, response efficiency in improving the accuracy [6] [12]. However, the problems, associated with the ease of collecting educational data today, are the scale of prediction models and the characteristic-aware of each dataset to applied prediction methods.

In this paper, we propose a distributed platform based on Spark to predict the scores of future courses for undergraduate students in our university. We obtain different prediction algorithms from combining techniques based on Recommender System such as Collaborative Filtering (namely User-based Collaborative Filtering and Item-based Collaborative Filtering), Matrix Factorization (using Alternative Least Square) and Nonnegative Matrix Factorization (added non-negative constraint to Alternative Least Square). The original dataset is grouped by different faculties: Industrial Maintenance, Chemical Engineering, Civil Engineering, ... For the evaluation, the paper performs our proposed methods to the data of four faculties. They are Computer Science and Engineering (MT), Environment and Natural Resources (MO), Chemical Engineering (HC), Civil Engineering (XD). Then, our experiment highlights that the best prediction model depends on the characteristic of the dataset, the performance can be changed from dataset to dataset. Furthermore, the accuracy of each prediction model is also affected by the way of dividing or choosing dataset associated with training the prediction model. In some cases, the experiment shows the difference when the prediction models are trained from student's data of the whole university and the separate faculty.

The rest of this paper is organized as follows. Section 2 shows related work about methods for predicting student performance. Section 3 is the background of algorithms and techniques that are used. We describe specifically the dataset

978-1-7281-2058-4/19/$31.00 ©2019 IEEE
DOI 10.1109/HPCC/SmartCity/DSS.2019.00041

provided by Ho Chi Minh City University of Technology in Section 4. Section 5 shows the proposed platform and methods. In Section 6, we present the experiment results, and highlight the conclusions as well as future work in Section 7.

2. Related Work

In terms of Educational Data Mining (EDM), one of the most common tasks is filtering out information that can be used to predict the student's performance [3] [1]. Generally, many studies have been conducted in order to predict student's grade as well as identify risky students. Currently, machine learning algorithms are used to solve the prediction problem in this field. Some of them are based on techniques frequently used in Recommender System [13].

Romero et al. have used classification algorithms such as Decision Tree, Rule Induction, Neural Network,... to predict students' final marks based on information in an e-learning system [5]. Random forest was employed in [4] to examine the statistical relationship between students' graduate-level performance and undergraduate achievements. García et al. have applied association rule mining to discover interesting information through students' usage data in the form of IF-THEN recommendation rules, which is to build a system helping teachers to continually improve and maintain adaptive and non-adaptive e-learning courses [14]. Nurjanah et al. proposed a technique for recommending learning materials which combine content-based filtering and collaborative filtering [7], [8]. In detail, content-based filtering is first applied to filter out relevant materials. Then, collaborative filtering is used to select good students. This technique aims to reduce the drawback of classic collaborative filtering which recommends materials based on the similarity between students and not take into account student's competence. The resulting model achieved MAE score of 0.96 for a scale of 1 to 10 and 0.73 for a scale of 1 to 5. In 2017, Iqbal et al. have applied and evaluated Collaborative Filtering, Matrix Factorization and Restricted Boltzmann Machine (RBM) to predict student grade in a dataset consists of 225 students and 24 courses with 1736 available grades and 3664 missing grades (grades are given in scale 0-4). They concluded that the RBM model gives the best result with 0.3 RMSE nearly half of the second best model (Non-negative Matrix Factorization) which had RMSE of 0.57 [9]. Otherwise, Conijn et al. analyzed 17 blended courses with 4,989 students using 23 predictor variables collected from Moodle Learning Management Systems (LMS) [6], They found that there is a larger improvement in prediction when those grades are unavailable in the case of in-between assessment grades are available. Thus, the LMS data in this dataset are substantially smaller predictive value compared to the midterm grades.

Concerning another technique, Thai-Nghe et al. used matrix factorization to predict student performance on the Knowledge Discovery and Data Mining Challenge 2010 dataset. They showed that matrix factorization could improve prediction results compared to traditional regression methods such as logistic/linear regression [10]. Furthermore, in their follow up paper [11], they extended the research, using tensor-based factorization to take the temporal effect into account when predicting student performance. Feng et al. [12] have taken the advantage of the student-system interaction information that is normally not available in traditional practice tests such as the time students take to answer questions and the time they take to correct an answer which they got wrong. The addition of this information is shown to make better predictions than tradition models which only uses correctness of test question. Elbadrawy et al. attempted to use Personalized Multiregression and Matrix Factorization to forecast students' grades on in-class assessments [15]. The results revealed that these methods can achieve a lower error rate than traditional methods.

Furthermore, a lot of researches focus reviews on existing types of educational systems and methods applied in EDM. Romero et al. have categorized EDM researches into 2 large groups [16]:

- Statistics and visualization
- Web mining

Web mining is considered a prominent group of EDM because many methods revolve around the analysis of logs of student-computer interaction. [1] examined three hundred published papers until 2009 grouped by task/category such as recommendation, predicting performance, detecting behavior, analysis, and visualization, etc. [3] investigated what some of the major trends are in EDM research. They found that in 43% papers which was examined in [16] published between 1995 and 2005 centered around relationship mining methods. However, in 2008 and 2009, relationship mining slipped to fifth place with only 9% papers. On the other hand, prediction, which was in second place between 1995 and 2005, moved to the dominant position in 2008-2009.

3. Background

3.1. Spark cluster overview

Today, Spark [17] is used at a wide range of areas to analyzing large-scale datasets. Figure 1 gives an overview of how Spark performs tasks on clusters. Applications or user-defined programs run as independent sets of processes, this is coordinated by the definition of *SparkContext* object in the code (called the Driver Program). Then, *SparkContext* plays a role in allocating resources across programs. Once connected, Spark finds *executors* on worker nodes in the cluster where the processes of user's programs will be run and stored data. Next, Spark sends the program code (written by Java or Python) to the executors. Finally, *SparkContext* sends *tasks* to the executors to run.

3.2. Data mining techniques

3.2.1. Collaborative Filtering. Collaborative Filtering is the popular algorithm which is commonly used in recommender systems. A recommender system focuses on suggesting the set of items for users based on their existing

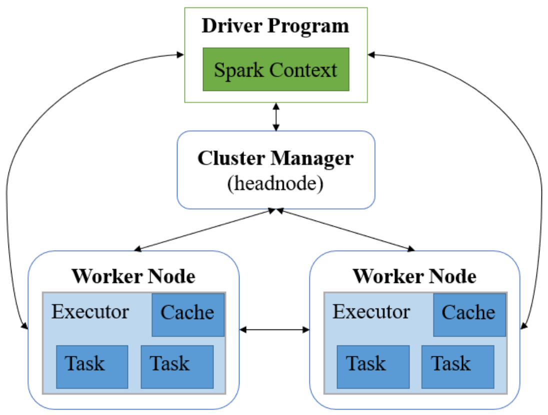

Figure 1: Spark architecture with the cluster mode

item. To do this function, firstly, we determine the user's rating for each item. In this paper, the users are students, the items are courses associated with the user's ratings are grades. There are two kinds of Collaborative Filtering: User-based Collaborative Filtering (UBCF) [18] and Item-based Collaborative Filtering (IBCF) [19].

User-based Collaborative Filtering: As regards, the student's grade in a course can be predicted by identifying similar students. The predictions are performed by selecting and aggregating the grades of other students. There is a list of n students S = $\{S_1, S_2, ..., S_n\}$ and a list of m courses C = $\{C_1, C_2, ..., C_m\}$. Each student has a list of courses which represents student GPA. To predict the student's grade in a course:

- Firstly, the UBCF algorithm calculates the similarity matrix to determine how similar each student in the database to the active student.
- After that, the algorithm select the most similar students by using k-nearest neighbors algorithm [20].
- The prediction results are generated by aggregating the GPAs of the most similar students. In the simple case, the aggregation can be mean or weighted average by taking similarity between students into account.

Item-based Collaborative Filtering: IBCF is used in the case that the courses have been rarely changed. In this algorithm, the student's grade in a course can be predicted by identifying similar courses which have learned by the current student. Instead of identifying the most similar students in UBCF, the IBCF algorithm determines the most similar courses from the set of courses that the current student have learned. The predictions are made by selecting and aggregating the grades of other courses. To predict the student's grade in a course:

- Firstly, the IBCF algorithm calculates the similarity matrix between the courses to determine how similar each course in the database to the course that needs to be predicted.
- Then, the algorithm will select the most similar courses which are learned by the active student based on using k-nearest neighbors algorithm.
- Similar to UBCF, the prediction result is made by aggregating the GPAs of the most similar courses.

3.2.2. Matrix Factorization. Matrix Factorization is the basis for some of the most successful realizations in the latent factor model which tries to characterize students and courses on k factors to explain the grades patterns. For courses, these factors can correspond to the amount of math, difficulty, number of equations. For student, these factors correspond to the student affinity toward those latent factor. Matrix Factorization is a method that decomposes a matrix. In this case, it is the utility matrix which represents all student's grade into two or more matrix.

Singular Value Decomposition: Singular Value Decomposition tries to decompose the utility matrix G into two matrix, U and V:

$$G \approx U \times V \tag{1}$$

where:

- U is a $m \times r$ matrix, where m is the number of students, r is the number of latent factors. Each student u is associated with vector p_u of the length r. Each element in this vector corresponds to the affinity of student u for the corresponding latent factor. Vector p_u can be viewed as a row in matrix U where U_{uk} represents the affinity of student u for the latent factor k.
- V is a $r \times n$ matrix, where n is the number of courses. V_{ik} represents the affinity of course i for latent factor k. Each course i is associated with a vector q_i of the length r. Each element in this vector corresponds to the affinity of course i for the corresponding latent factor. Vector q_i can be viewed as a row in matrix V where V_{ik} represents the affinity of course i for the latent factor k.

The dot product $p_u q_i^T$ will be the estimated grade $\hat{g}_{ui}$ of student u in course i:

$$\hat{g}_{ui} = p_u q_i^T \tag{2}$$

To learn matrix U and V, we will minimize the cost function:

$$\sum_{u,i \in H} \left(r_{ui} - p_u q_i^T\right) + \lambda \left(p_u^2 + q_i^2\right) \tag{3}$$

where H is the set of (u, i) pair where g_{ui} is in the training set; λ is the regularization parameter.

Using gradient descent, for each given value of g_{ui} in the training set, to update vector p_u and q_i:

$$\begin{aligned} p_u &= p_u + \gamma \left(\left(r_{ui} - p_u q_i^T\right) \cdot q_i - \lambda p_u\right) \\ q_i &= q_i + \gamma \left(\left(r_{ui} - p_u q_i^T\right) \cdot p_u - \lambda q_i\right) \end{aligned} \tag{4}$$

where γ is the learning rate.

Alternative Least Square (ALS): This method is one of the optimization for the Singular-Value Decomposition (SVD) [21] method. Recall Equation 3 where both p_u, q_i are unknown and tied with each other in a multiplication operation which makes this non-convex. The idea of ALS is: when we fix one of the unknown variables which is either p_u or q_i, the cost function becomes a quadratic problem. In each iteration, ALS first fixes U (all vectors p_u) and solves for V, then it fixes V (all vectors p_i) and solves for U. The process is repeated until there is a convergence. In ALS, each p_u is independent with other $p_{u'!=u}$, and each q_i is independent with other $q_{i'!=i}$. This algorithm can be massively parallelized.

Non-negative Matrix Factorization: Another type of matrix factorization, where the non-negative constraint is added. A given non-negative matrix G contains all observed grades, then we need to find the non-negative matrix factors, W and H [22].

$$G^{(n)} \approx W \times H \tag{5}$$

- W is a non-negative $m \times r$ matrix.
- H is a non-negative $r \times n$ matrix.

With normal matrix factorization, we can obtain negative affinity between a student u and a latent factor k which can be hard to interpret (e.g., the difficulty of latent factors can be negative). Non-negative matrix factorization can give us a better representation of the latent factors by guaranteeing non-negative value, thus a course can be described as a collection of smaller knowledge domains. For example, Computer Graphic Course can be interpreted as a collection of 60% algebra +20% math +20% art +0% literature.

4. Dataset

The educational dataset is collected from Ho Chi Minh City University of Technology in Vietnam. The dataset contains data of 61271 undergraduate students with over three million records about student's grade. There are 35 columns in the dataset, however, it is reprocessed with influence fields. In the context of this work, we focus on the prediction of average grades in some unfinished courses of students. Therefore, the basic foundation of our proposed module is the relation between finished courses and unfinished courses. Especially, this also depends on the educational program of each university.

Table 1: The statistics of the dataset

Number of faculties	14
Number of courses	2389
Number of students	61271
Number of student's grades	2270045
Sparsity	0.9845

In HCMUT, there are totally 14 faculties, 2389 courses and the number of students in the given dataset (from 2006 until 2017) is 61271, as Table 1 shown. The undergraduate program will finished in 4 - 4.5 years and the courses are divided into three groups: *Basis Courses*, *Compulsory Courses* and *Core&Elective Courses*. The courses correspond to the level of years that students are studying. Figure 2 shows the program of undergraduate students in the Faculty of Computer Science & Engineering. This highlights that our module just predicts the scores of unfinished and elective courses. Fundamentally, the information from finished courses (such as basic and compulsory courses) will be the basic foundation for the prediction of unfinished courses.

Besides 35 columns of information in the dataset, we reprocess with 4 main fields, as Table 2. In the core of our prediction module, the proposed algorithms will use these fields to train the prediction model. In detail, we have Student ID, name of faculty, course ID and the average grade from component grades on each course.

Table 2: Educational dataset

Student	Faculty	Course	Grade
29081892	Computer Science and Engineering	Data Mining	7.5
28193782	Chemical Engineering	Calculus	9.0
32876719	Mechanical Engineering	Physical Education	7.5
...	...	...	...

Each record is information about the course student's grade with the corresponding course. The grades are scaled from 0 to 10 by the *double* type. The distribution of student's grades is shown in Figure 3. The popular range of undergraduate grade is from 5 to 8.5, and the sparsity of the dataset is 0.9845. The sparsity is calculated by the following formula - (6):

$$S = 1 - \frac{G}{N \cdot C} \tag{6}$$

where, N, G are C are the total number of student's grades, students, and courses respectively. Furthermore, we also show the overview of information in all faculties in our university from the dataset, as Table 3 shown, e.g., the number of courses, students, or grades.

5. Methods

Apache Spark [17] has a well-defined layer architecture where all components are loosely coupled. This architecture can be further integrated with various extensions and libraries. The core of Spark is the base engine for large-scale parallel and distributed data processing. Our platform is built on the top of the core, which allows modifying as well as improve the predictor with different demands further.

In our platform, there are 4 main modules: preprocessing, training, deployed & recommender. Figure 4 shows the working flow and overview of our prediction platform, the white boxes are input, output or the definition of state in related modules. The paper aims at predicting student's score of unfinished & elective courses in the third-fourth year. The main idea is the machine learning core with the insights from

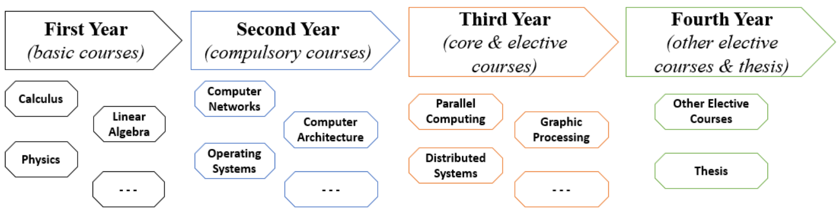

Figure 2: An example of the educational program in Faculty of Computer Science & Engineering.

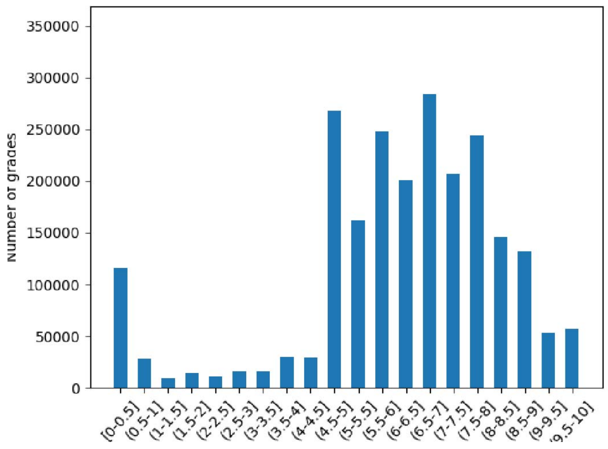

Figure 3: Distribution of student's grades

Table 3: The detail statistics of each faculty in the dataset

Faculty	Nota-tion	# courses	# stu-dents	# stu-dent's grades	Spar-sity
Computer Science and Engineering	MT	168	5158	155574	0.8205
Industrial Maintenance	BD	116	1958	54976	0.7580
Mechanical Engineering	CK	435	9233	351539	0.9125
Geology & Petroleum Engineering	DC	207	2476	93516	0.8175
Electrical and Electronic Engineering	DD	325	9391	360546	0.8819
Transportation Engineering	GT	230	2323	88510	0.8343
Chemical Engineering	HC	322	6117	222478	0.8870
Environment and Natural Resources	MO	177	2401	90633	0.7867
Energy Engineering	PD	89	565	16912	0.6637
Industrial Management	QL	137	3577	104514	0.7867
Applied Sciences	UD	192	2099	78986	0.8040
Materials Technology	VL	183	2910	109612	0.7942
Training Program of Excellent Engineers in Vietnam (PFIEV)	VP	309	1515	92040	0.8039
Civil Engineering	XD	445	11691	450209	0.9135

the history of finished courses that undergraduate students have had. According to the working flow:

- Firstly, the raw dataset is analyzed and preprocessed by the preprocessing module. Afterward, there are two sets: a training set and a test set.
- Secondly, the training set is used for building the prediction model with the machine learning techniques. Especially, the training module includes several core methods for training the prediction model. Following this module is the testing phase, it is to improve the prediction model across a number of loops.
- Next, the best module for predicting student's score will be deployed, and it is used in practice with the student's input (including student's information & scores of finished courses).
- Finally, the result of the prediction model will be used for the recommender module. This module focuses on the recommendation for students which courses they should choose to achieve a high result. Association rule is used here for the recommender module.

As mentioned above, the training module is the core of our prediction platform. We use several machine learning techniques to build the module. These techniques can be modified or improved with further methods. Depending on the feature of each dataset as well as the structure of education programs in different universities that we have a suitable method in the training module. The proposed machine learning techniques include:

5.1. Baseline Method

This method simply takes an average of all available grades of the student to predict missing course's grades. It is used as a baseline in comparison among different prediction models. For example, in Table 4, the estimated grade of Course 2 for student 1511000 will be calculated as $\frac{9+8}{2} = 8.5$.

5.2. Matrix Factorization

We apply Matrix Factorization in Spark as the form of Alternative Least Square. Spark also supports Non-negative Matrix Factorization by adding the non-negative constraints when solving least squares. In detail, the data in Table 4 can be interpreted as a matrix like Table 5 shown. This matrix is then factorized by using Alternative Least Square approach and the result matrix will be used as a prediction for missing values.

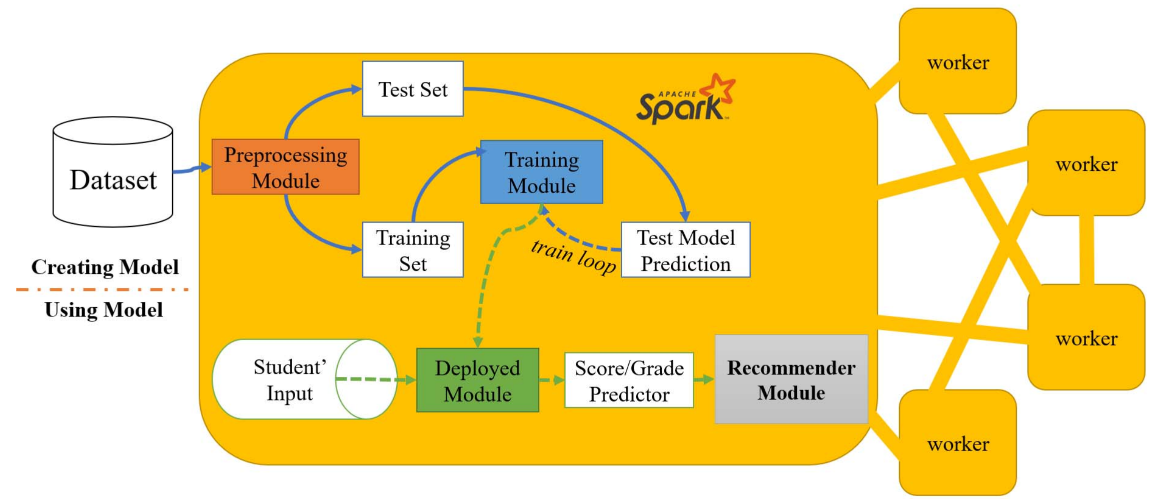

Figure 4: The overview of the score-prediction platform based on Spark for undergraduate level

Table 4: An example about dataset

Student ID	Course ID	Grade
1511000	0	9
1511000	1	8
1512000	1	7
1512000	0	8
1512000	2	7.5
1512000	3	8.5
1513000	0	7.5
1513000	2	7.5
1513000	1	8.5
1511000	**2**	**?**

Table 5: Representing the student's grades as a matrix

	0	**1**	**2**	**3**
1511000	9	8	?	?
1512000	8	7	7.5	8.5
1513000	7.5	8.5	7.5	?

5.3. User-based and Item-based Collaborative Filtering

Combining with Dataframe API [23] supported in Spark, we implement the prediction models that based on the principle of UBCF & IBCF. To illustrate how this method works, from the sample data in Table 4, to predict grade of student 1511000 in course 2 with UBCF, this algorithm will calculate the similarity between student 1511000 and others student who learned course 2 (using cosine similarity [19]), as Table 6 shown. Each student's feature contains student's grades in [course 0, course 1, course 2, course 3]. Finally, the UBCF algorithm will predict the grade of student 1511000 in course 2 by combining grade of student 1512000 & 1513000 with the corresponding similarity:

$$\text{predicted score} = \frac{0.68402 \cdot 7.5 + 0.82787 \cdot 7.5}{0.68402 + 0.82787} = 7.5$$

For IBCF, it will calculate the similarity between the predicted course (course 2) and other courses which are learned by student 1511000 instead of calculating the similarity of students. Each course's feature contains the grade of students [1511000, 1512000, 1513000] in that course. Finally, the IBCF algorithm will predict grade of student 1511000 in course 2 by combining the grade of student 1512000 in course 0 and course 1 with corresponding similarity:

$$\text{predicted score} = \frac{0.77259 \cdot 9 + 0.80526 \cdot 8}{0.77259 + 0.80526} = 8.48960$$

Table 6: Similarity Score between student 1511000 and other students

1511000's feature	[9, 8, 0, 0]
1512000's feature	[8, 7, 7.5, 8.5]
1513000's feature	[7.5, 8.5, 7.5, 0]
Similarity(1511000, 1512000)	0.68402
Similarity(1511000, 1513000)	0.82787

Table 7: Similarity Score between course 2 and other courses

course 0's feature	[9, 8, 7.5]
course 1's feature	[8, 7, 8.5]
course 2's feature	[0, 7.5, 7.5]
Similarity(course 2, course 0)	0.77259
Similarity(course 2, course 1)	0.80526

5.4. Item-based Collaborative Filtering on Course Factor matrix of Matrix Factorization

The resulting course factor matrices produced by ALS algorithms can be used to calculate the similarity among courses. Course factor matrix represents the relation between courses and hidden features in ALS algorithms, we can use those hidden features as inputs in IBCF.

6. Results

6.1. Testing environment

Our experiments are performed on the cluster named SuperNode-XP which is a heterogeneous cluster with 24 compute nodes. There are 2 CPU sockets - Intel Xeon E5-2680 v3 @ 2.70 GHz, 2 Intel Xeon Phi 7120P (Knight Corners) cards, and 128GB RAM per node. The Spark cluster in this paper is built on 4 nodes: 1 master & 3 workers. In addition, the software stack consists of:

Table 8: Software Specification on the testing environment

No.	Software	Description
1	Operating System	Red Hat Enterprise Linux 7.2
2	Spark	Apache Spark ver 2.4.0
3	Python	Version 2.7.15

6.2. Grade prediction

For undergraduate students, the dataset is divided separately into groups for training and testing the prediction model. This paper evaluates two situations for the experiments of predicting student's grades with the real dataset. Concretely, there are 2 case studies:

1) Only use the data of a specific faculty to train & test the prediction for that faculty (Locality Case - **LC** or **The experiment 1**).
2) Use the data of the whole university to train & test the prediction for students in a faculty (Global Case - **GC** or **The experiment 2**).

Students will be split into 2 groups: students are used in the train set, and students are used in the test set (80% and 20% respectively). Therefore, we will have a group of students that we have all the information (train students) in the test sets. In detail, the tested students are considered as new students with missing grades. In short:

- In the train set: occupies 80% of the whole dataset.
- In the test set: occupies 20% of the whole dataset.

We fit models using the train sets, then calculate Root Mean Square Error (RMSE), Mean Square Error (MSE), Mean Absolute Error (MAE) score of predictions on the test set. RMSE and MSE score will show us if there are any unusually high errors in some predictions. Meanwhile, MAE score give a more average error in the experiment. Each method is run and tested 10 times, with the train set and the test set is randomized again after each time. Then we get the average for the final results. Concretely, we use seven algorithms for the prediction model, as Table 9 shown.

When UBCF is used in a dataset with a large number of users, the run time can be too long. Users must be first clustered together, and we apply the algorithms to each cluster. Therefore, experiment 2 will not be run for UBCF. The results of 4 faculties are shown in Figures 5 to 8. Table 10 highlights the detail values of error for the evaluation of two experiments, **(1)** and **(2)**.

Table 9: The detail information of proposed algorithms

Name	Algorithms
Baseline	Taking average of all visible grades on each student
IBCF	Item-based Collaborative Filtering
UBCF	User-based Collaborative Filtering
ALS	Alternative Least Square
ALS_NN	Alternative Least Square with nonnegative constraint
ALS_NN_IBCF	Item-based Collaborative Filtering on Non-negative ALternative Least Square's Course Factor Matrix
ALS_IBCF	Item-based Collaborative Filtering on Normal ALternative Least Square's Course Factor Matrix

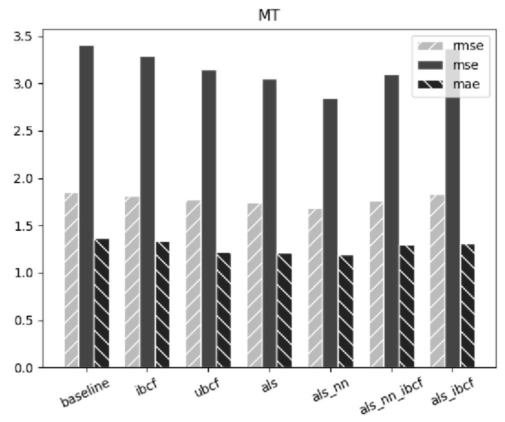

Figure 5: The error scores of different prediction models in the experiment 1 for the faculty of Computer Science & Engineering (MT)

Overall, all methods perform better than the baseline method. ALS models with the non-negative constraint achieve the best score when comparing to other models across all four faculties. The models with the non-negative constraint perform better than models with the negative constraint. For example, as Figure 5 shown, ASL_NN (the Non-negative ALS model) is obviously better than

Table 10: Detail results

Faculty	Metric	**Baseline**	**IBCF**	**UBCF**	**ALS**	**ALS_NN**	**ALS_NN_IBCF**	**ALS_IBCF**
MT(1)	RMSE	1.85	1.81	1.78	1.75	**1.69**	1.76	1.83
	MSE	3.40	3.29	3.15	3.05	**2.85**	3.10	3.37
	MAE	1.37	1.33	1.22	1.21	**1.19**	1.31	1.30
MO(1)	RMSE	1.61	1.57	1.54	1.50	**1.44**	1.56	1.58
	MSE	2.59	2.48	2.37	2.24	**2.08**	2.44	2.50
	MAE	1.21	1.18	1.05	**1.02**	1.03	1.18	1.18
HC(1)	RMSE	1.68	1.64	1.55	1.50	**1.47**	1.61	1.64
	MSE	2.82	2.69	2.41	2.25	**2.17**	2.60	2.71
	MAE	1.24	1.20	1.02	1.05	**1.01**	1.19	1.18
XD(1)	RMSE	1.91	1.84	**1.73**	1.76	1.73	1.79	1.88
	MSE	3.64	3.39	2.99	3.11	**2.98**	3.22	3.54
	MAE	1.41	1.35	**1.20**	1.26	1.25	1.33	1.33
MT(2)	RMSE	1.85	1.78	_	1.80	**1.72**	1.78	2.10
	MSE	3.44	3.18	_	3.24	**2.95**	3.16	4.64
	MAE	1.37	1.30	_	1.26	**1.22**	1.31	1.31
MO(2)	RMSE	1.62	1.55	_	1.46	**1.45**	1.56	1.59
	MSE	2.61	2.41	_	2.12	**2.11**	2.44	2.53
	MAE	1.22	1.15	_	1.04	**1.04**	1.17	1.16
HC(2)	RMSE	1.68	1.62	_	2.28	**1.49**	1.61	1.62
	MSE	2.82	2.61	_	7.36	**2.23**	2.60	2.63
	MAE	1.24	1.19	_	1.21	**1.05**	1.19	1.18
XD(2)	RMSE	1.90	1.82	_	1.74	**1.71**	1.79	1.88
	MSE	3.60	3.32	_	3.03	**2.93**	3.21	3.56
	MAE	1.41	1.34	_	1.24	**1.24**	1.33	1.33

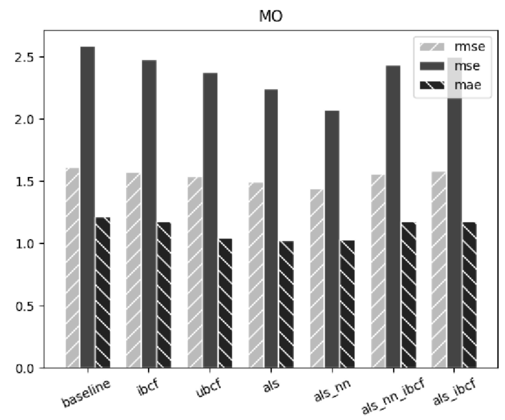

Figure 6: The error scores of different prediction models in the experiment 1 for the faculty of Environment and Natural Resources (MO)

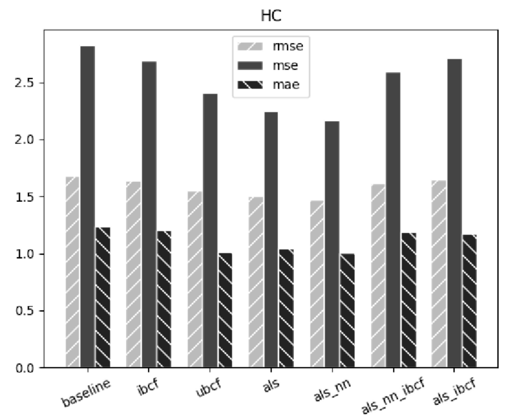

Figure 7: The error scores of different prediction models in the experiment 1 for the faculty of Chemical Engineering (HC)

ALS about RMSE, MSE, with the gain of $\approx$ 3.5% & 7%. Likewise, ALS_NN_IBCF also has lower errors than ALS_IBCF, corresponding to the gain of $\approx$ 4% for RMSE and 8% for MSE. For the evaluation on the data of other faculties as MO, HC & XD in Figure 6, 7, 8 respectively, we also get the same trend of results. In most scenarios, ALS_NN achieves the best results about the accuracy when predicting student's grades for missing courses. At the other view of prediction models, $UBCF$ always has a lower error compared to $IBCF$ across all 4 faculties (with average RMSE score of 1.65 and 1.72 respectively). Furthermore, another interesting point is: using $IBCF$ on the raw dataset results in the worse scores rather than applying $IBCF$ on Non-negative ALS's Course Factor Matrix (ALS_NN_IBCF).

These two experiments (**(1)** and **(2)**) also show that using a big train dataset with all of the faculties does not affect much when comparing to the prediction models which are trained by the dataset of a specific faculty. Table 10 shows that MT and XD faculties have a much higher base error than MO and HC faculties although HC has far higher sparsity (89%) and the number of courses (322) compared to MT which has sparsity of 82% and 168 courses. When running the experiment 2 of models without non-negative constraint, sometimes RMSE and MSE scores will be much higher abnormally despite the fact that MAE scores are very consistent among all running trials. For example, the ALS model in the faculty of HC has a pretty bad RMSE score (2.28) - worse than the RMSE of the baseline model. However, with the MAE score, it gets a better MAE score compared to the baseline model. This behavior is also seen in $IBCF$ and ALS models when running the dataset of MT faculty. The MAE score is very close to $IBCF$ but RMSE and MSE errors are very high. In summary, ALS models with the non-negative constraint can give the best accuracy out of all tested models in the case of our dataset in Ho Chi Minh City University of Technology. The accuracy can be changed from data to data and also depend on the prediction model that we apply.

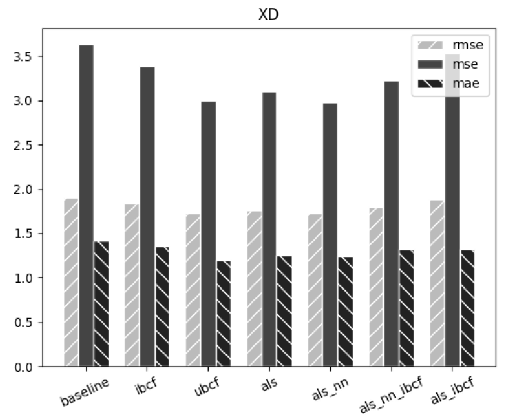

Figure 8: The error scores of different prediction models in the experiment 1 for the faculty of Civil Engineering (XD)

7. Conclusions and Future Work

Predicting GPA of future courses can be a valuable source of information to determine student's performance. These predicted GPA can assist instructors in identifying risky students and help students find their strength. In this study, we proposed a distributed platform built in Spark to accommodate all important components of a prediction and recommender system. Furthermore, our work also analyzes and compares various techniques commonly based on the combination of two theories Matrix Factorization and Collaborative Filtering. The experiments are performed with two cases of training the prediction models: Locality Case and Global Case. In the Locality Case (1), the prediction models are trained with student's data in a specific faculty and then we use it to predict student's grade in that faculty. Otherwise, the Global Case tries to train the prediction models with student's data of the whole university, then the model will be used to predict student's grade in one of the requested faculties. The result shows that there is little difference in performance between those two cases but not much. Furthermore, the prediction models with the non-negative constraint are better than others without. Especially, the Non-negative Alternative Least Square models (ALS_NN) achieve the best accuracy in most scenarios.

In summary, Machine Learning techniques applied in this study only take the advantage of available grades to predict the future performance and missing out other variables such as age, social standing, hobbies, preliminary test performance, etc. These predictor variables can be used to estimate the GPA of first-year courses where the dataset is extremely sparse. Our future research will explore deeply the relation of more influence variables in the prediction models, then evaluate the feasibility when applying to the graduate student's performance.

Acknowledgments

This research was conducted within the "Studying Tools to Support Applications Running on Powerful Clusters & Big Data Analytic (HPDA phase I 2018-2020)" funded by Ho Chi Minh City Department of Science and Technology (under grant number 46/2018/HĐ-QKHCN).

References

[1] C. Romero and S. Ventura, "Educational data mining: A review of the state of the art," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, pp. 601–618, Dec. 2010.

[2] K. Chrysafiadi and M. Virvou, "Student modeling approaches: A literature review for the last decade," *Expert Systems with Applications*, vol. 40, no. 11, pp. 4715–4729, 2013.

[3] R. S. Baker and K. Yacef, "The state of educational data mining in 2009: A review and future visions," *Journal of Educational Data Mining*, vol. 1, pp. 601–618, Dec. 2009.

[4] J. Zimmermann, K. H. Brodersen, J.-P. Pellet, E. August, and J. Buhmann, "Predicting graduate-level performance from undergraduate achievements," Jul. 2011, pp. 357–358.

[5] P. G. E. Cristóbal Romero, Sebastián Ventura and C. Hervás, "Data mining algorithms to classify students," *1st International Conference on Educational Data Mining*, p. 8–17, Jun. 2008.

[6] R. Conijn, C. Snijders, A. Kleingeld, and U. Matzat, "Predicting student performance from lms data: A comparison of 17 blended courses using moodle lms," *IEEE Transactions on Learning Technologies*, 2017.

[7] D. Nurjanah, "Good and similar learners' recommendation in adaptive learning systems," *Conference on Computer Supported Education*, vol. 1, pp. 434–440, 2016.

[8] R. Turnip, D. Nurjanah, and D. Kusumo, "Hybrid recommender system for learning material using content-based filtering and collaborative filtering with good learners' rating," Nov. 2017, pp. 61–66.

[9] Z. Iqbal, J. Qadir, A. N. Mian, and F. Kamiran, "Machine learning based student grade prediction: A case study," Aug. 2017.

[10] N. Thai-nghe, L. Drumond, A. Krohn-Grimberghe, and L. Schmidt-Thieme, "Recommender system for predicting student performance," *Procedia Computer Science*, vol. 1, p. 2811–2819, 2010.

[11] N. Thai-nghe, L. Drumond, R. Nanopoulos, and L. Schmidt-thieme, "Recommender system for predicting student performance," in *In Proceedings of the 3rd International Conference on Computer Supported Education (CSEDU)*, 2011.

[12] M. Feng, N. Heffernan, and K. Koedinger, "Addressing the assessment challenge with an online system that tutors as it assesses," *User Model. User-Adapt. Interact.*, vol. 19, pp. 243–266, Aug. 2009.

[13] P. Resnick and H. R. Varian, "Recommender systems," *Communications of the ACM*, vol. 40, no. 3, pp. 56–59, 1997.

[14] E. García, C. Romero, S. Ventura, and C. D. Castro, "An architecture for making recommendations to courseware authors using association rule mining and collaborative filtering," *User Modeling and User-Adapted Interaction*, vol. 19, no. 1-2, pp. 99–132, Feb. 2009.

[15] A. Elbadrawy, A. Polyzou, Z. Ren, M. Sweeney, G. Karypis, and H. Rangwala, "Predicting student performance using personalized analytics," *IEEE Computer.*, p. 61—69, Apr. 2016.

[16] C. Romero and S. Ventura, "Educational data mining: A survey from 1995 to 2005," *Expert Systems with Applications*, vol. 33, pp. 135–146, Jul. 2007.

[17] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets." *HotCloud*, vol. 10, no. 10-10, p. 95, 2010.

[18] Z.-D. Zhao and M.-S. Shang, "User-based collaborative-filtering recommendation algorithms on hadoop," in *2010 Third International Conference on Knowledge Discovery and Data Mining*. IEEE, 2010, pp. 478–481.

[19] B. M. Sarwar, G. Karypis, J. A. Konstan, J. Riedl *et al.*, "Item-based collaborative filtering recommendation algorithms." *Www*, vol. 1, pp. 285–295, 2001.

[20] K. Fukunage and P. M. Narendra, "A branch and bound algorithm for computing k-nearest neighbors," *IEEE transactions on computers*, no. 7, pp. 750–753, 1975.

[21] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," in *Linear Algebra*. Springer, 1971, pp. 134–151.

[22] D. Lee and H. Seung, "Algorithms for non-negative matrix factorization," in *Proceedings of the 13th International Conference on Neural Information Processing Systems*, vol. 13, Jan. 2000, pp. 535–541.

[23] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi *et al.*, "Spark sql: Relational data processing in spark," in *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. ACM, 2015, pp. 1383–1394.