Models and Algorithms for Performance Prediction and Course Recommendation in Higher Education

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Agoritsa Polyzou

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Dr. George Karypis

August, 2020

Acknowledgements

When I first started my graduate studies, I could not have imagined how things would turn out to be. As time passed, I enjoyed every year, met many people, learned how to do research, and here I am now, writing my Ph.D. thesis! This has been an incredible journey for me, and I cannot imagine myself doing anything else. I was lucky enough to be surrounded by many people that helped me and supported me, whom I would like to thank from the bottom of my heart.

First and foremost, I would like to thank my advisor, George Karypis. He has been an excellent advisor to me that supported me from the very beginning. Research-wise, he gave me direction when I needed it. He taught me how to do responsible and high-quality research, how to explore the data and interpret the results observed. Every meeting and interaction with him was an opportunity for me to learn. He also showed me how to keep my cool in situations that might be unfavorable or unexpected and only focus on what I could do better. I was always considering myself lucky to be close to such a caring, talented, smart, enthusiastic, and inspiring advisor, teacher, and person. I would like to thank him for believing in me, giving me the opportunity to work with him, and pushing me to become a better researcher.

I would like to thank Professors Nikolaos Sidiropoulos, Maria Gini, Lana Yarosh, John Carlis, and Claudia Neuhauser for serving on my committee for my thesis, thesis proposal, and preliminary exams. They have such deep knowledge in the field that I was fortunate enough to have them on my committees. Their comments and suggestions have been invaluable in shaping my research. Additionally, I would like to thank professors Huzefa Rangwala and Bodong Chen with whom I had the opportunity to collaborate and discuss my research.

Special thanks go to my big Greek family for their continuous love and support. My mother, Vaso, and my father, Kostas, always believed in me and tried their best to keep me safe and give me the best chance in life. I love them, and I wish to make them proud.

Of course, my sister Ioanna has a special and unique place in my heart and my life. She always supports me and believes in me. I feel that she is my number one fan, and she has such a heart-warming way of showing that. I am glad that I got to spend some of my graduate years next to her. I would also like to thank my aunt, Theoni, and my uncle, Giorgos. They were the people that motivate me to pursue a doctoral degree in Computer Science in the first place. They introduced me to the countless possibilities for forming a career path in the States, and they opened up Greece's borders for me. If it were not for them, I would have never thought about applying for a Ph.D. in the States. They made me believe that I can do whatever I set my mind on if I work hard enough. Through most of my studies, I consider myself lucky to have my boyfriend (and now fiance), Andreas, next to me. He always has my back; he celebrates my successes and helps me to go through some of my most difficult times. At the end of the day, I know that we have each other, and that makes me smile. We explored Minnesota together, and I look forward to our new adventures.

My Ph.D. journey would be very different without our amazing research lab (sometimes called "the Karypis crew") consisted of bright students, that I now get to call my friends. We stick together through this challenging time of our lives; we support each other both in research and on a personal level. I have kept great memories of my time in the Karypis lab that I will never forget, which include: Ancy Tom, Maria Kalantzi, Saurav Manchanda, Zeren Shui, Kostas Mavromatis, Athanasios Nikolakopoulos, Evangelia Christakopoulou, Sara Morsy, Mohit Sharma (my awesome office-mate for many years that supported me from the very beginning), Dominique Lasalle, David Anastasiu, Shaden Smith, Jeremy Iverson, Prableen Kaur, and Rohit JV.

During my years in Minnesota, I made many friends that I would like to thank: Ben, Evangelia, Konstantina, Foteini, Dimitris, Panos, Vasilis, Fateme, Dimitris, Donghoon, Yanning, Katerina, Giota, Katrina, Konstantinos, Giorgos, Nikos, Charilaos, and Nikos. I met some of them from the beginning and others, I met a little bit later, but I am very happy to have spent time with all of you. I believe you are great people, and I will miss you when we will be apart. I would also like to thank my friends Takis, Nikos, Mitsos, Sofi, Maria, and Maria, who are back in Greece, and know me for a long time.

Lastly, I would like to thank the staff at the Department of Computer Science, the Digital Technology Center, and the Minnesota Supercomputing Institute at the University of Minnesota for providing me the necessary resources for my research.

Abstract

Educational institutions need to use supporting tools in order to reduce the high student drop-out rates and ensure their students' timely graduation. *Educational data mining* involves the development of such methods that leverage student data. Their purpose is to generate warnings about students that are at risk of failing, identify the enrollment practices that are associated with academic success, and allow for suitable interventions. The application of these models in a higher education institution can improve advising and degree planning.

The objective of this thesis is to develop methods that will support students to make informed decisions regarding their course registration in the context of higher education. We also explore fairness concerns that might arise in a course recommendation system. We want to create models that can be used before the semester starts in order to allow the students to make any necessary adjustments in their plans. Instructors can also benefit from them, as it will allow them to shape their expectations about the enrolled students in their class, and adjust the syllabus and course material accordingly. For the purpose of this work, we will focus on traditional higher education datasets and develop models that do not need any information for the structure and the syllabus of a particular course. All we use is the students' transcript data.

In this setting, we first introduce next-term grade prediction methods to estimate the grades that a student is expected to receive for the courses they plan on taking the following semester. These algorithms are based on sparse linear and low-rank matrix factorization models that are specific to each course or student-course tuple. These methods identify the predictive subsets of prior courses on a course-by-course basis.

Second, we delve into the factors that influence students' performance. We extract informative features from the students' transcript data and form the problem as a binary classification one, to identify the students that have poor performance. We consider a student's performance both in the absolute sense (in terms of grades achieved), but also in the relative sense (compared to the student's past performance). We present a comprehensive study to answer the following questions: which features are good indicators of a student's performance? which features are the most important? The findings are interesting, as different features are the most important for different classification tasks.

Next, we addressed the problem of course recommendation. Recommender systems have been extensively used in various domains to support decision making and empower user choices. Within the educational domain, recommenders can generate a personalized list of courses for each student to consider taking in the next semester. Towards that direction, we propose Scholars Walk, a random-walk-based approach that captures the sequential relationships between the different courses. Based on the "wisdom of the crowd" and the students' prior courses, we recommend a short list of courses for next semester. Our framework is very efficient, easily interpretable, while also being able to take into consideration important aspects of the educational domain.

Finally, we explore fairness in the context of course recommendation. It is important to ensure that the models that we develop and apply in an educational institution do not discriminate against particular groups of students. All students should be treated fairly and offered similar opportunities and quality of services based on the notion of group fairness. Based on this idea, we examine the fairness of the opportunities offered to the students by the system's recommendations. We formulate our approach as a multi-objective optimization problem, and we study the trade-offs between equal opportunity and quality. The algorithm first makes an initial assignment of courses to recommend to the students, which we assume is of optimal quality. The model then refines the initial solution to improve the overall fairness. The experimental evaluation with synthetic and real datasets showcase that we can promote equality of opportunity in the recommendations without significantly weakening their quality.

Contents

List of Tables

List of Figures

List of Abbreviations

CRS Course Recommendation Rystem.

CSMF Course-Specific Matrix Factorization.

CSR Course-Specific Regression.

FaiREO Fairness in Recommendation for Equality of Opportunity.

GHC Greedy Hill Climbing.

GPA Grade Point Average.

LMS Learning Management Systems.

LSTM Long Short Term Memory.

MF Matrix Factorization.

RNN Recurrent Neural Networks.

SSR Student-Specific Regression.

Chapter 1

Introduction

Higher educational institutions constantly try to improve the retention and success of their enrolled students. According to the US National Center for Education Statistics [55], 60% of undergraduate students on four-year degrees will not graduate at the same institution where they started within the first six years. At the same time, 30% of college freshmen drop out after their first year of college.

The general purpose of higher education is to offer programs, which will help learners to gain knowledge throughout their studies. Students enjoy a plethora of offerings. However, course selection can be "messy and unorganized" [5] as it depends on many factors that students need to consider. Students have to balance personal preferences (interests, objectives, and career goals) and general education and degree program requirements. As a result, course selection can be a non-trivial task. Decisions can be made based on manual guides offered from each department, but these are not tailored to individual cases [26] in a higher education setting. Personalized assistance can be given by academic advisers, however this is not scalable with large cohorts with thousands of students. The ratio of student to advisor may be very high [45], limiting the adviser-advisee discussion time. Additionally, college students take on average up to 20% more courses than required [22]. Better advising can help alleviate these problems.

Colleges look for ways to serve students more efficiently and effectively. Student experience and success are the prioritized factors for educational institutions. Educational data mining and learning analytics have been developed to provide tools for supporting and understanding the learning process, like monitor and measure student progress, but also, predict success or guide intervention strategies. We need predictive models that can be employed to enable strategic action and attain better results. Existing approaches

focus on identifying students at risk who could benefit from further assistance in order to successfully complete a course or their studies.

Within the context of educational data mining, we explore the following questions:

- **How reliable can we predict student performance?** A fundamental task is to predict a student's performance. We need methods to accurately estimate how well students will perform (as measured by their grade) on courses that they have not yet taken. Being able to accurately estimate students' grades in future courses is important as it can be used by them (and/or their academic advisers) to identify the appropriate set of courses to take during the next term, and create personalized degree pathways that enable them to successfully and effectively acquire the required knowledge to complete their studies in a timely fashion.

- **Which are the most predictive factors for identifying students at-risk of failing a course?** Grade prediction is just the first step towards understanding student performance. In order to better support students, we need to study the factors that might influence their performance. It is important to gain insights in the learning process and when a student is more likely to perform well or not. That will enable us to make meaningful, informed and more targeted interventions.

- **Which are the courses that a student needs to take next semester?** Students register ahead of time for the courses that they will take the upcoming semester. The student has to be prepared for these courses, and these courses need to match this interests. A system that recommends courses is very useful as it help the students decide what to choose. Recommender systems can be used in this domain to tailor a set of courses that best matches the students' progress, interests and background.

- **How can we ensure that all students treated similarly from a recommendation system regardless their protected attributes?** All students have access to the same tools offered from an institution, but we need to ensure that they are also treated fairly. That is not always the case, because the recommendation systems may propagate the biases found on the input data towards future users. Specifically, historical and social biases may introduce disparity in the quality, usefulness, and impact of a system for students in different protected groups. We need to measure the extent of such system behavior and account for that while generating recommendations.

1.1 Contributions

We develop various **next-term grade prediction** methods that utilize approaches based on sparse linear models and low-rank matrix factorizations. Regression and matrix factorization have been applied before in related work with a variety of data, but our methods rely entirely on the performance that the students achieved in previously taken courses. A unique aspect of our methods is that their associated models are either specific to each course or specific to each student-course tuple. This allows them to identify and utilize the relevant information from the prior courses that are associated with the grade for each course and better address problems associated with the reliable estimation of the low-rank models and the not-missing-at-random nature of the student-course historical grade data.

The next body of work focuses on **understanding poor-performing students**, who are the ones that need these systems the most. The objective of this work is two-fold. First, we explore if poorly performing students can be more accurately predicted by formulating the problem as binary classification. Second, in order to gain insights as to which are the factors that can lead to poor performance, we engineered a number of human-interpretable features that quantify these factors. We identify two complementary groups of students, the ones that are likely to successfully complete a course, and the ones that seem to struggle. However, good performance can be relative or not. For example, a B− grade might be considered a bad grade for an excellent student, while being a good grade for a very weak student. Similarly, a B+ for an easy class can be a rather bad grade, whereas a B in a hard class can be a good grade. We investigated both absolute and relative ways to define groups of students, which are interesting to predict from an intervention standpoint. In order to gain more insight into the learning process and its most important characteristics, we have identified factors and extracted features that might affect student performance. Using these features, we present a comprehensive study that is designed to answer the following questions: which features are good indicators of a student's performance? which features are the most important? The findings are interesting, as different features are the most important for different classification tasks.

For the **course selection problem**, we introduced Scholars Walk, a random-walk based approach. It describes a personalized model that takes advantage of the sequential nature of course selection. We assume that students' choices for the next term depend on the courses they have taken so far. In our approach, we build a Markov chain for

each degree program over the courses taken consecutively. Then, we perform a random walk, starting from the courses that students took in the previous semester. We evaluate the proposed approach on a number of different departments with different subjects and characteristics. Scholars Walk overall outperforms other competing approaches in all the metrics considered, while still being simple and interpretable.

When studying **fairness in course recommendation**, we first considered ways that such a system can be unfair towards its primary users, the students. We introduce a notion of fairness for recommender systems and define a metric called fairness in recommendation for equality of opportunity (FaiREO). We assume that course recommendation involves a notion of opportunity; when students receive a particular course suggestion, they have the opportunity to review its contents and consider taking it next semester, something that they might not have done otherwise. We operationalize this notion by ensuring that a course is recommended at a similar rate across student groups formed based on one or more protected attributes. To that end, our work introduced five greedy hill-climbing algorithms, GHC(Gc), GHC(G), GHC(NoNe), GHC-Inc, and GHC-Tabu, that work in two phases. First, they make an initial assignment of recommended courses to users, and then, they refine the initial solution in order to improve the overall fairness according to a multi-objective function. This function captures and balances two fairness-related but often conflicting goals: equality in recommendation quality and opportunity offered to students across different protected groups. The experimental evaluation with synthetic and real data shows that even by giving a little importance to fairness, we can significantly improve the behavior of the system to offer equal opportunities to its users.

1.2 Related Publications

The work presented in this thesis has been published in a variety of conferences and journals. The related publications with this thesis are presented in the following list:

- **A. Polyzou** and G. Karypis. Grade prediction with course and student specific models. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), pages 89–101. Springer, 2016. [72]
- **A. Polyzou** and G. Karypis. Grade prediction with models specific to students and courses. International Journal of Data Science and Analytics, 2(3-4):159–171, 2016. [73]

- A. Elbadrawy, **A. Polyzou**, Z. Ren, M. Sweeney, G. Karypis, and H. Rangwala. Predicting student performance using personalized analytics. Computer, 49(4):61–69, 2016. [31]
- Q. Hu, **A. Polyzou**, G. Karypis, and H. Rangwala. Enriching course-specific regression models with content features for grade prediction. In IEEE International Conference on Data Science and Advanced Analytics (DSAA), pages 504–513. IEEE, 2017. [41]
- **A. Polyzou** and G. Karypis. Feature extraction for classifying students based on their academic perfor- mance. In Proceedings of the 11th International Conference on Educational Data Mining (EDM), pages 356–362. ERIC, 2018. [74]
- **A. Polyzou** and G. Karypis. Feature extraction for next-term prediction of poor student performance. IEEE Transactions on Learning Technologies, 12(2):237–248, 2019. [75]
- **A. Polyzou**, A. N. Nikolakopoulos, and G. Karypis. Scholars walk: A markov chain framework for course recommendation. In Proceedings of the 12th International Conference on Educational Data Mining (EDM), pages 396–401. ERIC, 2019. [76]
- **A. Polyzou**, M. Kalantzi, and G. Karypis. FaiREO: User Group Fairness for Equality of Opportunity in Course Recommendation. 2020. *Under review.*

1.3 Thesis Outline

This thesis proposal is organized as follows:

- Chapter 2 introduces the notation that we will use in the rest of the report.
- Chapter 3 discusses the previous research work that is related to grade prediction, feature importance, course recommendation problems and fairness.
- Chapter 4 presents the proposed course-specific approaches for next-term grade prediction.
- Chapter 5 presents the study for the factors indicating poor student performance in future courses.

- Chapter 6 presents Scholars Walk, a system that recommends to students courses that better match their academic level and progress.
- Chapter 7 describes our latest work in considering fairness in the task of course recommendation.
- Chapter 8 summarizes the conclusions drawn from the work in this thesis and discusses future directions.

Chapter 2

Definitions and Notation

Boldface uppercase letters will be used to represent matrices (e.g., $\mathbf{G}$) and boldface lowercase letters to represent row vectors, (e.g., $\mathbf{r}$). The ith element of vector $\mathbf{r}$ is represented as r_i. The ith row of matrix $\mathbf{R}$ is represented as $\mathbf{r}_i$. The entry in the i-th row and j-th column of matrix $\mathbf{G}$ is denoted as $g_{i,j}$. An estimated value is denoted by having a hat over it (e.g., $\hat{y}$). Calligraphic letters will be used to denote sets (e.g., $\mathcal{S}$, with cardinality $|\mathcal{S}|$).

The sets of students and courses are indicated by $\mathcal{S}$ and $\mathcal{C}$. For their cardinalities, the letters n and m are used, respectively. A subscript, if existing, describes the sub-population it refers to. The historical student-course grade information will be represented by a sparse matrix $\mathbf{G} \in \mathbb{R}^{n \times m}$ and $g_{i,j}$ is the grade in the range of [0,4] that student i achieved in course j. If a student has not taken a course, the corresponding entry will be missing.

The course and student whose grades need to be predicted or the student groups we need to examine will be called *target student*, *course*, and *student group*, respectively, in order to seperate them from the rest. All the courses that a student took in past semesters, before the target semester t, are the *prior courses*, denoted by $\mathcal{C}_{i,all\_prior}$. The set of courses for a single semester t is denoted as $\mathcal{C}_{s,t}$. Any course j worths a specified number of credits, cr_j. A student is active in the program for a number of terms, $\text{nterms\_active}_{i,t}$, before target semester. Additionally, for a course j there might exist a stated set of courses that are required for a student to take before attempting j. We refer to this set as the *prerequisite courses*.

Chapter 3

Related Work

3.1 Performance Prediction

In recent years, there has been a lot of research activity in using data analysis approaches to understand, support, and enhance student learning. This research addresses a variety of problems at different environment settings. The key problems include student modeling, selection of educational material that match student's learning needs, and student's performance prediction at tasks, course activities, homework questions, exams and final grades, either during a course or after its completion. Some other studies focus on the problem of predicting the students' term and final GPA [3, 63, 65].

The need of tools that understand and support student learning has led to the development of intelligent "early warning systems" that monitor the students' performance during the term [4, 56, 86]. The data collected by Learning Management Systems (LMS) have also been exploited [80, 81]. Text mining of written comments has been applied for performance prediction by [51, 85], while [59, 79] apply classification and genetic algorithms with features about student interaction and the use of the LMS. In order to analyze the student's past performance and interaction with the LMS and predict how well he/she will perform in course activities, multi-regression models have also been proposed [32]. Various approaches for modeling and predicting the success or failure of students in the context of intelligent tutoring systems have been developed. In that case, the goal is to predict the correctness of a student's attempt to solve a single or a sequence of problems/tasks/exercises. These approaches include regression models [6, 17, 35], HMMs and bagged decision trees [69], collaborative filtering techniques and their combination (k-NN, SVD, RBM) [94], matrix completion [44, 91, 92] and tensor factorization [93].

Recently, research efforts aim to predict the grade that a student will obtain in a future course he/she will take. Within the context of developing methods to predict the next-term grades, most existing approaches [16, 23, 24, 77] rely on neighborhood-based collaborative filtering methods. For each student whose grade needs to be predicted, a set of *similar* students are identified that have already taken that course and their grade is used to estimate the desired grade via some similarity-weighted aggregation function. Despite their relative simplicity, the estimations obtained by these methods are reasonably accurate indicating that there is sufficient information in the historical student-course grade data to make the estimation problem feasible. Influenced by the area of recommender systems, the authors of [88, 89] examine grading prediction as rating prediction using a matrix completion approach. In [89], the features are about the student, course and instructor, while in [88], the matrix contains information about the grades of past courses. The matrix will be estimated by the product of lower rank matrices. The authors also point out that bias terms are important and quite informative for the models.

The models we developed are based on linear regression and matrix factorization, but they utilize only a course and student-course specific subset of the data. According to our results, focusing on specific models per course improves the prediction accuracy and enable more reliable model estimation, while models are capable to fit the data better.

3.2 Feature Selection

When there is a significant number of features without any quality guarantees, feature selection approaches are applied to gain insights on the underlying concepts. Supervised selection methods are of interest to us since we have the label information available. These methods are divided into three categories: filter, wrapper, and embedded models [90]. Filter models evaluate the features based only on the characteristics of the data, e.g., mutual information [71], relief [78]. The best features are selected first, and then, they are used to build any classifier. Wrapper models chose first the classifier, and then determine the quality of a subset of features by the accuracy achieved on the predetermined classifier [48]. Lastly, the embedded models combine the two models by performing feature selection and classifier building at the same time [50]. In our case, we will use an approach based on wrapper models, with predefined subset of features to examine.

3.3 Course Recommendation

Recommender systems have been broadly applied within the context of student learning [53]. We will further review the different approaches developed to help students select a subset of courses to register for an upcoming semester. The first course recommender systems are based on constraint satisfaction [67]. The sequence-based recommender [98] also considers complex constraints to improve the expected time-to-degree and GPA. A related body of work involves mining of association rules. Al-Badarenah et al. [2] cluster the students based on their grades first. Nguyen et al. [61] apply sequential rule mining in (course, grade) pairs and recommend the courses with the best performance. A different CRS was proposed by Esteban et al. [33], where there is available information about students' satisfaction after taking a course.

Recently, Recurrent Neural Networks (RNN) have been successfully applied within the educational domain. Long Short Term Memory (LSTM) networks have been used for grading prediction [42, 64]. In terms of course recommendation, a combination of LSTM and skip-gram models has also developed to balance implicit and explicit student preferences [68]. Morsy et al. [60] have also used RNN to recommend courses which are expected to help maintain or improve students' GPA. Other approaches include a Markov-based model [47], that represents the sequence of courses taken as a stochastic process. Garner et al. [39] build a co-enrollment network and extract features for a network-based structural model. Finally, Elbadrawy et al. [30] propose using the academic features to improve the recommendation performance.

3.4 Fair Recommender Systems

We have identified three main classes of problems closely related to our problem and fairness issues, and we present briefly representative related work in the following subsections.

3.4.1 Fair Recommender Systems

Fairness in recommender systems is relatively new and each work presents its own point of view on the subject. A body of work studies fairness in ranking lists [1, 8, 9, 84, 99, 100], where the goal is to provide diverse representation of the items, in the sense of equal exposure of different groups of items. A recommendation ranking is considered unfair

when specific protected groups of items are under-ranked and as a result they receive lower visibility in the system. This corresponds to fairness with respect to the items. Beutel et al., [8] account also for the user engagement. They measure the differences in accuracy across the groups of items based on pairwise comparisons. According to their definition of pairwise fairness, assuming that two items have received the same user engagement, then both protected groups should have the same likelihood of a clicked item being ranked above another relevant unclicked item. The work [100] studies fairness in search engines of people, such as job recruiting, companionship or friendship search. In such cases, an outcome is unfair if members of one protected group are systematically under-ranked than those of another protected group. The recommended candidates are determined by a ranking algorithm. The proposed method to remove the bias is a post-processing process. All the above works are different from ours as we do not account for diversity in the recommendation lists; we are not interested in recommending courses from all different categories. In the course recommendation domain, we recommend a limited number of courses which should be aligned with the students' interests and capabilities in order to maximize their performance and achieve their goals. Moreover, item diversity does not guarantee group fairness with respect to equality of opportunity for the student groups. Additionally, we aim at ensuring user-side fairness in a CRS and as such we consider protected groups of users/students and not items.

Another body of work studies fairness across multiple stakeholders in recommendations: the system, the suppliers/vendors and the users [15, 52, 58]. While these works study the tradeoff between the different stakeholders' benefits, we are interested in user-side fairness only. We consider ways to fix the fairness while harming as little as possible the relevance of recommended items, both benefits for a single stakeholder that compete with each other.

Apart from the algorithmic fairness, issues may also arise from biases in the input data which the recommender system amplifies [34, 99]. Tsintzou et al., [95] proposed a metric called bias disparity to measure the difference between the bias towards different movie genres in user profiles (input) and in resulted recommendations (output). A similar work proposed a group-based metric to compare the preference ratio in the input and output data (recommendation lists) and quantify the degree to which recommendation algorithms may propagate any biases [49].

3.4.2 Fair Allocation Problem

In the fair allocation or division problem, we want to fairly divide a resource or *goods* to *agents* with different preferences in the resource [11]. In the course recommendation context, we could consider the courses as the resources, the students as the agents and the recommendation scores as the expressed preferences of the agents towards the goods.

We focus on the relevant body of work that addresses group fairness in the context of fair division. Group fairness was initially introduced for the problem of divisible goods, e.g., the cake-cutting problem, in the form of envy-freeness [7, 43]. In certain works [54, 87], the notion of group fairness deals with settings in which the members of each group are allocated the same set of resources, which does not apply in our case of protected groups, as each student of one group can receive different recommendations from the others in the same group.

In summary, in all the above cases, the notion of group fairness is defined with respect to the preferences of the agents over the goods. In other words, a group is treated fairly when the assignment of the goods is aligned with the members' preferences, in the sense that *everyone gets what he or she values the most.* In our case, this is achieved by the recommendation algorithm, and then, in order to ensure group fairness, we want to refine this initial solution and allocate/recommend different courses equally across the protected groups.

3.4.3 Fair Course Allocation

According to the Course Allocation problem [27], we have a set of students with preferences to courses, a set of courses with preferences to students (priority orderings over the students from the course administrator), and each course has a specific predefined capacity; the goal is to allocate students to seats of courses. Course Allocation is an instance of the combinatorial assignment problem if we consider no preferences on the courses side (one-sided preferences) [14]. In this domain, a highly unfair outcome could lead to some students assigned to their most preferable courses and some other students assigned to their least ones or even to zero courses [14]. Diebold et al., [28] evaluate multiple matching mechanisms with real data in the context of course allocation with indifferences-ties in school preferences (combinatorial assignment problem). This notion of fairness corresponds to individual and not group fairness. Additionally, in the Fair

Course Recommendation problem, we do not consider any restrictions (such as the capacity of a course) other than offering high-quality (in terms of recommendation scores) and fair recommendations to all students.

Chapter 4

Next-Term Grade Prediction

4.1 Introduction to the Problem

We studied next-term grade prediction in the context of higher education. Before the start of a semester, students register for the courses they will take. Our work focuses on developing methods that utilize historical student-course grade information to accurately estimate how well students will perform (as measured by their grade in the end of the semester) on courses that they plan on taking the upcoming semester. We are interested in predicting students' grades before the start of the semester.

Being able to accurately estimate students' grades in future courses is important as it can be used by them (and/or their academic advisers) to assess a student's readiness for a course, identify the appropriate set of courses to take during the next term, and create personalized degree pathways that enable them to successfully and effectively acquire the required knowledge to complete their studies in a timely fashion.

4.2 Proposed Approaches

4.2.1 Course-Specific Regression (CSR)

Undergraduate degree programs are structured in such a way that courses taken by students provide the necessary knowledge and skills for them to do well in future courses. As a result, the performance that a student achieved in a subset of the earlier courses can be used to predict how well he/she will perform in future courses. Motivated by this, we developed a grade prediction method, called *course-specific regression* (CSR)

that predicts the grade that a student will achieve in a specific course as a sparse linear combination of the grades that the student obtained in past courses.

In order to estimate the CSR model for course c, we extract from the overall student-course matrix $\mathbf{G}$ the set of rows corresponding to the students that have taken c. For each of these students (rows), we keep only the grades that correspond to courses taken prior to course c. Let $\mathbf{G}^c \in \mathbb{R}^{n_c \times m}$ be the matrix representing that extracted information, where n_c is the number of students that took course c. In addition, let $\mathbf{y}^c \in \mathbb{R}^{n_c}$ be the grades that the students in $\mathbf{G}^c$ obtained in course c (y_i^c is the grade that was obtained by the student of the ith row of $\mathbf{G}^c$). Given this, the CSR model $\mathbf{w}^c \in \mathbb{R}_+^m$ for c is estimated as:

$$\underset{\mathbf{w}^c \succeq 0}{\text{minimize}} \ \left\|\mathbf{y}^c - \mathbb{1} w_0^c - \mathbf{G}^c \mathbf{w}^c\right\|_2^2 + \lambda_1 \left\|\mathbf{w}^c\right\|_2^2 + \lambda_2 \left\|\mathbf{w}^c\right\|_1, \tag{4.1}$$

where w_0^c is a bias term, $\mathbb{1} \in \mathbb{R}^{n_c}$ is a vector of ones, and λ_1, λ_2 are regularization parameters to control overfitting and promote sparsity. The model is non-negative because we assume that prior courses can only provide knowledge to future courses. The individual weights of $\mathbf{w}^c$ indicate how much each prior course contributes to the prediction and represent a measure of the importance of the prior course within the context of the estimated model. Using this model, the grade that a student will obtain in course c is given by:

$$\hat{y}^c = w_0^c + \mathbf{s}^T \mathbf{w}^c, \tag{4.2}$$

where $\mathbf{s} \in \mathbb{R}^m$ is the vector of the student's grades in the courses he/she has taken so far.

In this approach, prior to estimating the model using Equation 4.1, we first subtract from each $g_{i,j}^c$ grade the GPA of the i-th student (GPA is calculated based on the information in $\mathbf{G}^c$). This centers the data for each student and takes into consideration a notion of student bias as it predicts the performance with respect to the current state of a student. Note that in the case of GPA-centered data, we remove the non-negativity constraint on $\mathbf{w}^c$. We found that by centering each student's grades around his/hers GPA leads to more accurate predictions (see Section 4.4.1).

4.2.2 Student-Specific Regression (SSR)

Depending on the major, the structure of different undergraduate degree programs can be different. Some degree programs have limited flexibility as to the set of courses that a

student has to take and at which point in their studies they can take them (i.e., specific semester). Other degree programs are considerably more flexible and are structured around a fairly small number of core courses and a large number of elective courses.

For the latter type of degree programs, a drawback of the CSR method is that it requires the same linear regression model to be applied to all students. However, given that the set of prior courses taken by students in such flexible degree programs can be quite different, there can be cases in which many of the most important courses that were identified by the CSR model were simply not be taken by some students, even though these students have acquired the necessary knowledge and skills by taking a different set of courses. To address this limitation, we developed a different method, called *student-specific regression* (SSR), which estimates course-specific linear regression models that are also specific to each student.

The student specific model is derived by creating a student-course specific grade matrix $\mathbf{G}^{s,c}$ for each target student s and each target course c from the $\mathbf{G}^{c}$ matrix used in the CSR method. $\mathbf{G}^{s,c}$ is created in two steps. First, we eliminate from $\mathbf{G}^{c}$ any grades for courses that were not taken by the target student. Second, we eliminate from $\mathbf{G}^{c}$ the rows that correspond to the students that have not taken a sufficient number of courses that are in common with the target student s. Specifically, if $\mathcal{C}_s$ and $\mathcal{C}_i$ are the set of courses for student s and i, respectively, we compute the overlap ratio $(\text{OR}) = |\mathcal{C}_s \cap \mathcal{C}_i|/|\mathcal{C}_s|$ and if $\text{OR} < t$, then student i is not included in $\mathbf{G}^{s,c}$. The value of t is a parameter of the SSR method and high values ensure that the set of students forming $\mathbf{G}^{s,c}$ have taken many courses in common with s and have followed similar degree plans. Given $\mathbf{G}^{s,c}$, the SSR method proceeds to estimate the model using Equation 4.1 (with $\mathbf{G}^{s,c}$ replacing $\mathbf{G}^{c}$), and uses Equation 4.2 for prediction.

4.2.3 Methods Based on Matrix Factorization

Low rank matrix factorization (MF) approaches have been shown to be very effective for accurately estimating ratings in the context of recommender systems [46]. These approaches can be directly applied to the problem of predicting the grade that a student will achieve on a particular course by treating the student-course grade matrix $\mathbf{G}$ as the user-item rating matrix.

The use of such MF-based approaches for grade prediction is postulated on the fact that there is a low dimensional latent feature space that can jointly represent both students and courses. Given the nature of the domain, this latent space can correspond to

the space of knowledge components. Each course vector is the set of components associated with a course and each student vector represents the student's level of knowledge across these knowledge components.

By applying the common approaches of MF-based rating prediction to the problem of grade prediction, the grade that student i will obtain on course j is estimated as

$$\hat{g}_{i,j} = \mu + sb_i + cb_j + \mathbf{p}_i \mathbf{q}_j{}^T, \tag{4.3}$$

where μ is a global bias term, sb_i and cb_j are the student and course bias terms, respectively, and $\mathbf{p_i}$ and $\mathbf{q_j}$ are the latent representations for student i and course j, respectively. The parameters of the MF method ($\mu, \mathbf{sb} \in \mathbb{R}^n, \mathbf{cb} \in \mathbb{R}^m, \mathbf{P} \in \mathbb{R}^{n \times l}$, and $\mathbf{Q} \in \mathbb{R}^{n \times l}$) are estimated following a matrix completion approach that considers only the observed entries in $\mathbf{G}$ as

$$\begin{aligned} \underset{\mu,\mathbf{sb},\mathbf{cb},\mathbf{P},\mathbf{Q}}{\text{minimize}} & \sum_{g_{i,j} \in \mathbf{G}} \left(g_{i,j} - \mu - sb_i - cb_j - \mathbf{p}_i \mathbf{q}_j^T\right)^2 \\ & + \lambda(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2 + \|\mathbf{sb}\|_2^2 + \|\mathbf{cb}\|_2^2), \end{aligned} \tag{4.4}$$

where λ is a regularization parameter and l is the dimensionality of the latent space, which is a parameter to this method.

The accurate recovery of the low rank model (when such a model exists) from a set of partial observations depends on having a sufficient number of observed entries, and on these entries be randomly sampled from the entries of the target matrix $\mathbf{G}$ [18]. However, in the context of student grade data, the set of courses that students take is not a random subset of the courses being offered as they need to satisfy their degree program requirements. As a result, such an MF approach may lead to suboptimal prediction performance.

In order to address this problem we developed a *course specific matrix factorization* (CSMF) approach that estimates an MF model for each course by utilizing a course specific subset of the data that is denser (in terms of the number of observed entries and the dimensions of the matrix). As a result, it contains a larger number of randomly sampled subsets of sufficient size. The denser course specific matrix will allow a more reliable estimation of the low rank models. At the same time, the sub-matrix will be more homogeneous, as the students included are likely to be more similar (i.e., with more common prior courses) compared to all the students. That will allow the model to

fit the data better.

Given a course c and a set of students $\mathcal{S}^c$ for which we need to estimate their grade for c (i.e., the students in $\mathcal{S}^c$ have not taken this course yet), the data that CSMF utilizes are the:

1. the students and grades of the $\mathbf{G}^c$ matrix and $\mathbf{y}^c$ vector of the CSR method (Section 4.2.1), and

2. the students in $\mathcal{S}^c$ and their grades.

This data is used to form a matrix $\mathbf{X}^c \in \mathbb{R}^{(n_c+n_t)\times(m_c+1)}$, where n_c is the number of students in $\mathbf{G}^c$, $n_t = |\mathcal{S}^c|$, and m_c is the number of distinct courses that have at least one grade in $\mathbf{G}^c$ or $\mathcal{S}^c$. The values stored in $\mathbf{X}^c$ are the grades that exist in $\mathbf{G}^c$ and $\mathcal{S}^c$. The last column of $\mathbf{X}^c$ stores the grades $\mathbf{y}^c$ for the course c that were obtained from the students in $\mathbf{G}^c$. Thus, $\mathbf{X}^c$ contains all the prior grades associated with the students who have already taken course c and the students for which we need to have their grade on c predicted. Matrix $\mathbf{X}^c$ is then used in place of matrix $\mathbf{G}$ in Equation 4.4 to estimate the parameters of the CSMF method, which are then used to predict the missing entries of the last column of $\mathbf{X}^c$, which are the grades that need to be predicted.

4.3 Experimental Design

4.3.1 Dataset

The student-course-grade dataset that we used in our experiments was obtained from the University of Minnesota which has a very flexible degree program. It contains the students that have been part of the Computer Science and Engineering (CS&E) and Electrical and Computer Engineering (ECE) programs from Fall of 2002 to Spring of 2014. Both of these degree programs are part of the College of Science & Engineering. Students have to take a common set of core science courses during the first 2–3 semesters, but they can select more courses from different levels and departments.

Because of the nature of these departments, the curriculum coherence tends to be vertically aligned, i.e., what students learn in one lesson, course, or grade level is most likely going to be used by the next lesson, course, or grade level. Students select courses in order to learn the knowledge and skills that will progressively prepare them for more challenging, higher-level topics. However, we need to point out that this might not

always be the case, as there are departments that are more horizontally aligned, where there do not exist such strong dependencies across different courses and levels.

While preprocessing the dataset, we removed any courses that are not part of those offered by departments in the college, as these correspond to various liberal arts and physical education courses, which are taken by few students and in general do not count towards degree requirements. Furthermore, we eliminated any courses that were taken as pass/fail. The initial grades were in the A–F scale, which was converted to the 4–0 scale using the standard letter-grade to GPA conversion. The resulting dataset consists of 2,949 students, 2,556 different courses, and 76,748 student-course grades.

We used this dataset to assess the performance of the different methods for the task of predicting the grades that the students will obtain in the last semester (i.e., the most recent semester for which we have data). For this reason, the dataset was further split into two parts, one containing the students that are still *active*, i.e., have taken courses in the last semester (D_{active}) and one that contains the remaining students ($D_{inactive}$). D_{active} contains 876 students, 19,089 grades, out of which 3,427 grades are for the 475 distinct classes taken in the last semester. $D_{inactive}$ contains 2,073 students and 57,659 grades.

These datasets were used to derive various training and testing datasets for the different methods that we developed. Specifically, for the CSR method we extracted the course specific training and testing datasets as follows. For each course c that was offered in the last semester, we extracted course-specific training and testing sets ($D_{train}^{c,\geq k}$ and $D_{test}^{c,\geq k}$) by selecting from $D_{inactive}$ and D_{active}, respectively, the students that have taken c, and prior to taken c, they also took at least k other courses. The reason that these datasets were parametrized with respect to k is because we wanted to assess how the methods perform when different amount of historical student performance information is available. In our experiments we used k in the set $\{5, 7, 9\}$. That information creates the grade matrix $\mathbf{G}^c$, where $g_{i,j}^c$ is the grade of the ith student on the jth course from the training set $D_{train}^{c,\geq k}$. Table 4.1 shows statistics about the various course-specific datasets for different values of k.

For the CSMF method, the training dataset for course c was obtained by combining $D_{train}^{c,\geq k}$ and $D_{test}^{c,\geq k}$ into a single matrix after removing the grades that the target students achieved in course c.

For the MF method, the matrix $\mathbf{G}$ is constructed using data from all $\mathbf{X}^c$ matrices. It refers to the union of the sets $D_{train}^{c,\geq k}$ and $D_{test}^{c,\geq k}$ for every course to be predicted,

Table 4.1: Statistics for Course-Specific datasets.

	CS&E courses			ECE courses		
Prior courses	5	7	9	5	7	9
Avg number of students in training set	386	325	258	414	377	332
Avg number of students in test set	41	37	29	34	33	32
Avg number of prior courses	178	176	173	158	156	155
Avg number of grades	5,671	5,186	4,484	7,084	6,804	6,366
Courses predicted	24	24	24	25	25	25
Grades predicted	1,004	910	712	858	841	800

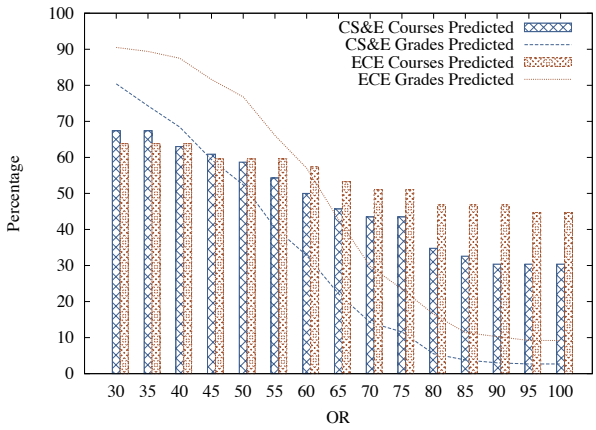


Figure 4.1: Statistics of the datasets used in SSR w.r.t. overlap ratio.

after removing the grades that the active students achieved in the courses we want to predict. We formulated the dataset in this way in order to provide the same information for training and testing to all our models. Moreover, since we predict the grades for a specific semester, matrix $\mathbf{G}$ does not contain any grading information regarding following semesters.

In the SSR, the grade matrix $\mathbf{G}^{s,c}$ is created by selecting from $D_{train}^{c,\geq k}$ the set of courses that were also taken by student s and the set of students whose OR with s is at least t. Figure 4.1 shows some statistics about these datasets as a function of t, and Figure 4.2 shows only the common subsets that can be predicted by both course specific and SSR datasets. When the OR is more than 0.8, we cannot predict many grades because there are not enough students that had followed the same degree plan as the selected student.

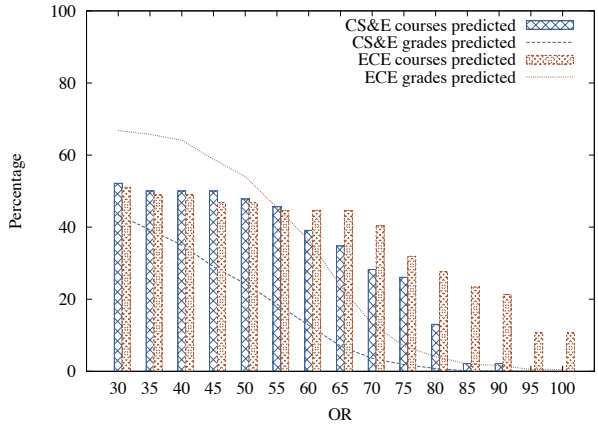


Figure 4.2: Statistics of the common subset of datasets used in SSR and in course specific approaches w.r.t. overlap ratio.

Finally, we did not consider the courses that have less than 20 students in their corresponding dataset, as we consider them to have too few training instances for reliable estimation, or less than 4 test students, as we might not get valid results.

4.3.2 Competing Methods

In our experiments, we compared our methods with the following competing approaches.

1. **BiasOnly.** We only took into consideration local and global biases to predict the students' grades. These biases were estimated using Eqn. 4.4 by setting $l = 0$.

2. **Student-Based Collaborative Filtering (SBCF).** This method implements the approach described in [16]. For a target course c, every student i is represented by a vector whose non-zero entries are the grades that the student obtained on the courses taken prior to c. We compare the vector of a target student s against the vectors of the other students that have taken course c using the Pearson's correlation coefficient. We perform grade prediction while taking into consideration the positively similar students to s according to

$$\hat{g}_{s,c} = \bar{g}_s + \frac{\min(r, nbr)}{r} \frac{\sum_{i=1}^{nbr} (g_{i,c} - \bar{g}_i)\ \mathrm{sim}_{s,i}}{\sum_{i=1}^{nbr}\ \mathrm{sim}_{s,i}}, \tag{4.5}$$

where nbr is the number of students selected, r is a confidence lower limit for significance weighting, $\bar{g}_i$ is the average grade of the student prior taking c, and

$\text{sim}_{s,i}$ represents the similarity of target student s with i.

Table 4.2: The performance achieved by Linear Course-Specific Regression per department.

	CS&E courses					
	RMSE			AvgRMSE		
Prior courses	5	7	9	5	7	9
CSR	0.928	0.958	0.990	0.994	1.034	1.082
CSR-RC	0.727	0.725	0.722	0.726	0.726	0.716
	ECE courses					
	RMSE			AvgRMSE		
Prior courses	5	7	9	5	7	9
CSR	0.717	0.693	0.704	0.702	0.685	0.699
CSR-RC	0.634	0.632	0.634	0.651	0.646	0.651

The performance of the models trained on the different datasets were evaluated on the $D_{test}^{\geq 9}$ test set, which is the common subset among their respective test sets.

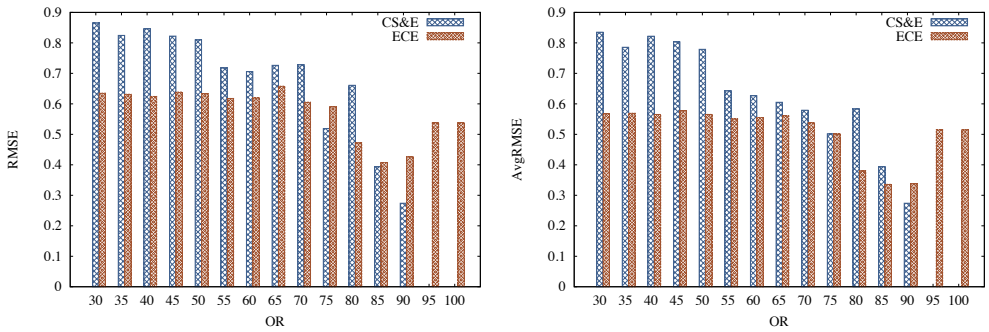


Figure 4.3: The performance achieved by the SSR model w.r.t. overlap ratio.

4.3.3 Parameters and Model Selection

For CSR, we let λ_1 take values from 0 to 40 in increments of 1 and λ_2 from 0 to 50 in increments of 1. For SSR, we let λ_1 take values from 0 to 10 in increments of 1 and λ_2 from 0 to 14 in increments of 2. For BiasOnly, MF and CSMF, we let λ take values from 0 to 16 in increments of 0.05. For SSR, the range of the tested values for overlap ratio is 0.3 to 1, in increments of 0.04 and for the confidence lower limit is 10 to 100,

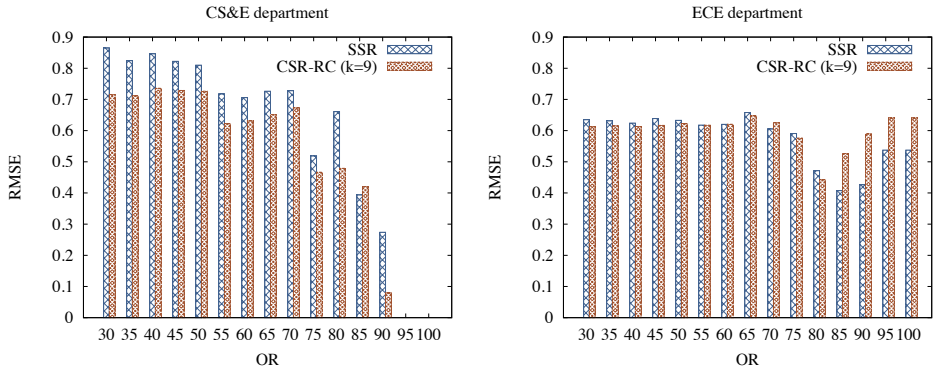


Figure 4.4: Comparison of SSR model and CSR-RC with 9 prior courses w.r.t. overlap ratio. The other options for number of prior courses have similar behavior.

in increments of 10. For SBCF, we tested the number of neighbors to be from 10 to 100 with increments of 10. For MF and CSMF methods we tested the number of latent dimensions with the values 2, 5 and 8.

For SBCF, CSR and SSR, we used the semester before the target semester to estimate and select the best parameters. For BiasOnly, MF and CSMF, model selection was based on the performance of the validation set, which was a randomly selected 10% subset of the training data. For the CSMF model, the best-performing parameters were selected for each course.

4.3.4 Evaluation Methodology & Performance Metrics

We evaluated the performance of the different approaches by using them to predict the grades for the last semester in our dataset using the data from the previous semesters for training. We report the results for the courses belonging to CS&E and ECE departments.

We assessed the performance using the root mean square error (RMSE) between the actual grades and the predicted ones. Since the courses whose grades are predicted have different number of students, we computed two RMSE-based metrics. The first is the overall RMSE in which all the grades across the different courses were pooled together, and the second is the average RMSE obtained by averaging the RMSE values for each course. We will denote the first by RMSE and the second as AvgRMSE.

In order to get a better understanding of the quality of the predictions, we also report the distribution of the actual vs predicted letter grades. The grading system used by the University of Minnesota has 11 letter grades (A, A-, B+, B, B-, C+, C, C-, D+, D, F)

that correspond to grades from 4 to 0 (4, 3.667, 3.333, 3, 2.667, 2.333, 2, 1.667, 1.333, 1, 0). After converting the predicted grades to their closest letter grade, we compute the percentage of grades that are within or more than x ticks away from their actual grades. A tick is defined as the difference between two successive letter grades (e.g., B vs B+ is one tick, A vs B is 3 ticks).

4.4 Experimental Results

4.4.1 Course-Specific Regression

Table 4.2 shows the performance achieved by the CSR and CSR-RC models when trained using the three different data sets discussed in Section 4.3.1. These results show that between the two models, CSR-RC, which operates on the GPA-centered grades, leads to considerably lower errors both in terms of RMSE and AvgRMSE, especially for the CS&E courses.

In terms of the sensitivity of their performance on the amount of historical information that was available when estimating these models (i.e., the minimum number of prior courses), we can see that the performance of the models does not change significantly for the CSR-RC method. CSR predicts CS&E courses better when using 5 prior courses, while it predicts better the ECE courses with 9 prior courses. This indicates that the model benefits from increased number of students that increased number of prior courses, because the students with 9 prior courses are only 67% of the students with 5 prior courses. The ECE department does not suffer from such low number of students left with 9 prior courses, as the corresponding percentage is 80% (statistics according to Table 4.1).

4.4.2 Student-Specific Regression

As one of the parameters for this problem was the overlap ratio between the courses of the target student and other students, Figure 4.3 presents the behavior of the model's RMSE (left) and AvgRMSE (right) as we vary the OR for $D_{test}^{c,\geq 9}(k = 9)$. When the OR is increased, the selected students have more courses in common with the target user and that leads to better performance.

In order to compare the performance of SSR against CSR-RC, Figure 4.4 shows the RMSE of the best CSR-RC and SSR models. The RMSE values were computed on

Table 4.3: Errors per department for Matrix Factorization methods.

			CS&E courses		ECE courses	
Prior courses	Latent Factors		MF	CSMF	MF	CSMF
5	2	RMSE	0.740	0.734	0.603	0.606
	5		0.753	0.731	0.605	0.616
	8		0.735	0.734	0.596	0.602
	2	AvgRMSE	0.726	0.716	0.614	0.615
	5		0.732	0.717	0.608	0.628
	8		0.721	0.714	0.605	0.612
7	2	RMSE	0.741	0.739	0.606	0.615
	5		0.750	0.735	0.611	0.607
	8		0.744	0.734	0.598	0.601
	2	AvgRMSE	0.726	0.729	0.610	0.626
	5		0.720	0.711	0.607	0.617
	8		0.727	0.728	0.604	0.609
9	2	RMSE	0.740	0.735	0.604	0.603
	5		0.746	0.723	0.600	0.601
	8		0.751	0.733	0.597	0.598
	2	AvgRMSE	0.726	0.732	0.611	0.617
	5		0.721	0.714	0.601	0.611
	8		0.735	0.725	0.607	0.610

the subsets of the test set that was predicted by both models for $D_{test}^{c_i \geq 9}(k = 9)$. These results show that SSR leads to consistently worse predictions for the CS&E courses than the CSR-RC model. However, in the case of the ECE courses, SSR does better than CSR-RC when the OR is greater than 0.8. That might be related to the fact that the degree program of ECE is more structured than the CS&E degree program, giving some advantage to the SSR method. As shown in Figure 4.1, at such high OR values, the number of grades that can be predicted by SSR is small. For example, when OR is 0.8, the SSR model can predict less than 10% of the grades in the target semester.

4.4.3 Methods Based on Matrix Factorization

The performance of the methods based on matrix factorization (Section 4.2.3) is shown in Table 4.3.

These results show that for the CS&E courses, CSMF performs the best in terms of RMSE and AvgMSE, for any number of prior courses. That confirms that by building

matrix factorization models on smaller but denser course-specific sub-matrices, we can derive low-rank models that lead to more accurate matrix completion. On the other hand, the performance of the ECE courses does not vary a lot. For that department, the best predictions are performed by MF, followed by CSMF with a RMSE difference of 0.002. A potential explanation for these results is that the ECE courses are part of a stricter degree program, whose structure is present even in the more general setting of MF. As a result, by selecting the course-specific sub-matrices does not provide any further insight to the data, as happens for the CS&E courses.

In order to see how the size of the training set associated with the different courses impacts the performance of the MF and CSMF methods, Figure 4.5 shows the cumulative AvgRMSE over the courses with increasing training size and the RMSE per course achieved from each method. Cumulative AvgRMSE is used to provide some insight to the impact that the training size has on the performance of our models. We can notice that for the ECE courses, MF model has an advantage against CSMF for relatively smaller courses. MF performs better for eight out of the ten smallest courses, indicating that it gains its accuracy by utilizing other data that are not included in the course specific datasets in order to compute better biases. Moreover, from the bottom part of the figure, we can confirm that the performance of both MF and CSMF is similar for the ECE courses in comparison to the CS&E courses.

In terms of the number of latent factors, we see that when we are using the smallest dataset for training (the one with 9 prior courses), the best performance is achieved for smaller number of latent factors compared to the datasets with 5 or 7 prior courses. In that case, the average number of grades per course is lower, which might not support a large number of latent factors.

4.4.4 Comparison with Other Methods

We compare the performance of the baseline approaches described in Section 4.3.2 (BiasOnly and SBCF) with the best-performing course-specific regression method (CSR-RC), the MF and CSMF methods. A summary of the comparison between every pair of methods tested can be found on Table 4.4. For each method, we count the courses for which a method wins, ties and losses in terms of RMSE against each other method tested. This analysis shows that for the CS&E courses, CSR-RC outperforms the other methods, except SBCF that is very close, in the majority of the courses, whereas for the ECE courses, the CSMF outperforms each one of the other methods (even MF method that

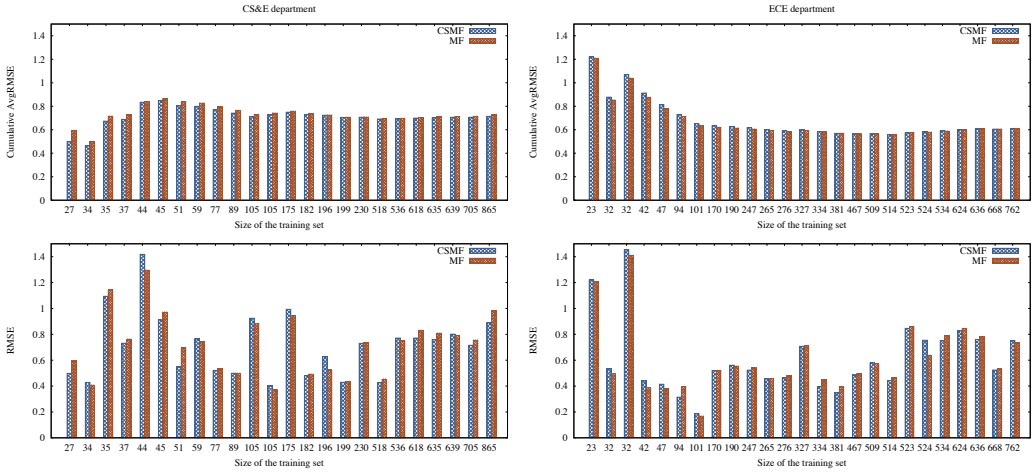


Figure 4.5: Cumulative AvgRMSE w.r.t. increasing training size (top) and RMSE achieved per course (bottom) of CSMF and MF models for $D_{test}^{c,\geq 9}(k = 9)$.

has slightly better RMSE) in the majority of the courses.

4.4.5 Fine Grain Analysis of the Predictions

In order to gain a better understanding as to the types of errors generated by the different methods and the real-world implication of the predictions Table 4.5 analyzes the performance achieved by the different methods by focusing on grade ticks as opposed to RMSE values.

Table 4.5 shows the percentage of predicted grades that were close to the true grades, over all the instances predicted by a model. For the CS&E department, CSMF is the model with the most grades that are predicted to be within two ticks from their true values, while CSR-RC is the best model when focusing on exact predictions. For the ECE department, MF has the highest percentages, and CSMF can be better only for the case of 9 prior courses, within two letter grades from the actual grades.

Table 4.6 analyzes the performance of the models on the instances that they fail to accurately predict. We examine the difference between the grades over or under predicted, i.e., they are predicted to be more or less than their real values respectively. In this case, the lower the percentage, the better the model is, as there are less inaccurate predictions. These results show that, compared the CS&E, ECE has less under predictions, but higher number of over predictions of more than one tick. Moreover, the best methods for the CS&E courses are the CSR-RC and CSMF, and for the ECE courses,

Table 4.4: Wins/Ties/Losses for every pair of methods tested.

	CS&E courses				
	OnlyBias	SBCF	CSR-RC	MF	CSMF
OnlyBias		7/1/16	7/1/16	6/5/13	7/2/15
SBCF	16/1/7		12/1/11	11/3/10	13/2/9
CSR-RC	16/1/7	11/1/12		13/2/9	12/4/8
MF	13/5/6	10/3/11	9/2/13		9/2/13
CSMF	15/2/7	9/2/13	8/4/12	13/2/9	
	ECE courses				
	OnlyBias	SBCF	CSR-RC	MF	CSMF
OnlyBias		8/2/15	10/3/12	7/6/12	6/2/17
SBCF	15/2/8		13/4/8	8/4/13	8/4/13
CSR-RC	12/3/10	8/4/13		7/4/14	6/3/16
MF	12/6/7	13/4/8	14/4/7		8/3/14
CSMF	17/2/6	13/4/8	16/3/6	14/3/8	

The cell (i, j) refers to the wins/ties/losses of the i-th method compared to the corresponding j-th method.

are the MF and CSMF. Another finding is that CSR-RC has the highest percentages of under prediction errors for the ECE department. The reason this is happening is because a student might have not taken an important course, and its corresponding regressor will be missing while estimating their grade. As a result, we can see that in the case of this department, that has a stricter degree program, CSR-RC (that is a linear model) cannot handle the absence of an important prior course. However, CSR-RC is the only model that manages to lower the over prediction error while using more dense data (case of 9 prior courses).

Table 4.5: Analysis of the accuracy of the predictions in terms of letter grades.

		5 prior courses				
		BiasOnly	SBCF	CSR-RC	MF	CSMF
CS&E	no error	23.73	23.87	**26.40**	25.84	24.85
	within one tick	62.63	63.59	63.45	**65.02**	62.47
	within two ticks	81.58	82.95	83.81	81.98	**84.22**
ECE	no error	30.38	28.38	26.37	**30.13**	26.38
	within one tick	66.62	65.39	64.89	**67.36**	66.24
	within two ticks	87.75	88.26	86.50	**90.12**	88.86
		9 prior courses				
		BiasOnly	SBCF	CSR-RC	MF	CSMF
CS&E	no error	25.84	24.16	**26.68**	24.58	24.43
	within one tick	62.08	63.32	63.31	63.04	**63.74**
	within two ticks	81.46	83.25	84.37	81.98	**84.52**
ECE	no error	30.98	28.11	26.27	**30.88**	27.40
	within one tick	65.74	64.98	65.52	**67.51**	66.00
	within two ticks	87.59	88.09	86.26	88.62	**89.73**

These numbers correspond to the percentage of the predicted grades that were exactly, within one tick or two ticks away from the true letter grade. One tick corresponds to a letter grade away from the true grade, i.e., we predict a grade of B while the student took B- in a course.

While comparing models, the higher the percentage the better it is for the grades predicted exactly, or less than one or two ticks away. For each case, the best percentage is in bold.

Table 4.6: Analysis of the error severity of the predictions in terms of letter grades.

		5 prior courses				
		BiasOnly	SBCF	CSR-RC	MF	CSMF
CS&E	underpredict (>1 tick)	20.34	18.93	**18.38**	18.79	21.34
	overpredict (>1 tick)	16.96	17.38	18.08	**16.10**	16.11
	underpredict (>2 ticks)	9.53	**7.56**	7.98	8.97	8.00
	overpredict (>2 ticks)	8.82	9.39	8.12	8.96	**7.70**
ECE	underpredict (>1 tick)	14.74	15.38	15.22	**13.73**	15.74
	overpredict (>1 tick)	18.60	19.16	19.83	18.84	**17.96**
	underpredict (>2 ticks)	4.88	3.75	4.99	**2.73**	4.50
	overpredict (>2 ticks)	7.33	7.92	8.45	7.08	**6.58**
		9 prior courses				
		BiasOnly	SBCF	CSR-RC	MF	CSMF
CS&E	underpredict (>1 tick)	20.76	19.22	**18.94**	19.37	19.22
	overpredict (>1 tick)	17.10	17.38	17.66	17.51	**16.96**
	underpredict (>2 ticks)	9.10	7.28	7.14	8.28	**7.00**
	overpredict (>2 ticks)	9.38	9.39	**8.40**	9.66	**8.40**
ECE	underpredict (>1 tick)	15.21	15.87	15.08	**12.60**	14.73
	overpredict (>1 tick)	**18.96**	19.04	19.31	19.83	19.22
	underpredict (>2 ticks)	4.86	3.75	5.47	**3.61**	3.63
	overpredict (>2 ticks)	7.86	8.05	8.18	7.71	**6.59**

These numbers correspond to the percentage of the predicted grades that were one or two ticks away from the true letter grade. One tick corresponds to a letter grade away from the true grade, i.e., we predict a grade of B while the student took B- in a course.

A model under or over predicts when the grade predicted is lower or higher, respectively, than the actual one.

While comparing models, the lower the percentage the better it is for the grades predicted more than one or two ticks away. For each case, the best percentage is in bold.

Chapter 5

Understanding Student Performance

5.1 Introduction to the Problem

Course-specific approaches utilize the student's grades from courses taken prior to that course to predict a student's grade. However, there are a lot of factors other than student's historical grades that influence his/her performance, such as the difficulty of the courses, the academic level of the students when taking the courses and so on. We consider a course-specific regression model enriched with features about students and courses. This model not only utilizes the students' grades, but also a number of features extracted from the historical data that capture student and course features. Considering a specific term for which a student has taken a course, we extracted their GPA of the previous term, the accumulative GPA as of last term, the GPA over only courses from their own departments, as well as, the students' academic level. The features relating to a course include its discipline, the credit hours and course level. We use the GPA of the course from last term to represent the difficulty of the course. We still compute the students' grades using Eq. 4.2, but this time, $\mathbf{s}$ includes both the grades and the extracted features for every student. We evaluate this hybrid model using the data from Spring 2014 as the test set, and the rest data as the training set. The experimental results showed that incorporating content features can boost the performance of the course-specific model. However, based on Fig. 5.1, we over-predict students grades' and we completely miss the students failing courses. For students received the letter grade

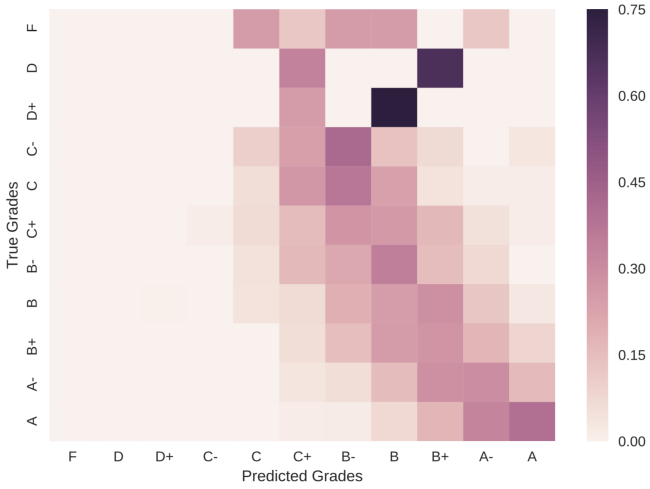


Figure 5.1: Distribution of letter grades predicted for each true (ground truth) letter grade (rows sum up to one).

F, we predict at least a grade of C. This serves as motivation that we need to better understand and predict students at-risk.

In this section, we focus on the poor-performing students, who are the ones that need these systems the most. The task of grade prediction is formulated as a classification task, where two groups of students are formed according to their course performance. We identify two complementary groups of students, the ones that are likely to successfully complete a course or activity, and the ones that seem to struggle. After identifying the latter group, we can provide additional resources and support to enhance their likelihood of success. We also need to gain more insight into the learning process and its most important characteristics that influence the student performance.

5.2 Dataset

An undergraduate student enrolled to a college or university has to take some courses each semester, and receive a satisfactory grade on an A–F basis in order to successfully complete them. Depending on the student's degree program, these courses might be required, electives, or simply courses that the student takes for his/her own advancement, intellectual curiosity, or enjoyment.

The original dataset was obtained from the University of Minnesota and it spans 13 years. We removed any instances that received a letter grade not in the A–F grading

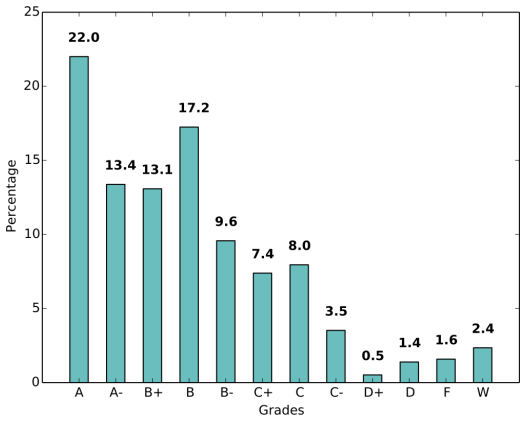


Figure 5.2: Percentage of each letter grade with respect to the total grades.

scale (A, A−, B+, B, B−, C+, C, C−, D+, D, F). If a student withdraws from a course after the first two weeks of classes, it is denoted by the letter 'W' in the student's transcript. Statistics about the grades in the dataset are shown in Fig. 5.2. In our dataset, the letter grade A is the most common. We extract features for the instances occurring during the last 10 fall and spring semesters. Given a semester and a target course, we utilize all the students that had taken the course before, and for each student taking a course, we extract a set of features.

Additionally, we generate features for the instances awarded with the letter W, but we do not utilize them in any other way during the feature extraction process. These will be used only when trying to predict the students that drop-out from a course.

5.3 Classification Problems

5.3.1 Classification Tasks

Our motivation was to identify groups of students that need further assistance and guidance in order to successfully complete a course. These students could benefit from informed interventions. We consider this to be a binary classification problem, where these students form one of the classes and the remaining students form the other class.

We consider different ways of measuring when a student does not do well in a course to deal with the performance measurement challenges we mentioned earlier. Unsatisfactory performance can occur when the earned grade represents a performance that is below the student's potential. We considered the following four ways for labelling, resulting to

Table 5.1: Statistics for the different classification tasks.

Task	Fgr	Wgr	RelF	RelCF	Agr
Total instances	94,364	96,941	94,364	94,364	94,364
Positive instances	3,139	2,577	20,398	21,724	20,851
% of positive class	3.33	2.70	21.62	23.02	22.10

four different classification tasks:

1. Failing student performance, i.e., letter grades D and F (denoted as the **Fgr** task).
2. The letter grade W (denoted as the **Wgr** task). This represents the instances when the student dropped the course. This behavior is worrisome as it shows that either the student was not interested in the course anymore or he/she expects to perform poorly.
3. Student performance that is worse than expected, i.e., the grade achieved is more than two letter grades lower than the student's GPA (denoted as the **RelF** task).
4. Student performance that is worse than expected while taking into consideration the difficulty of the course (denoted as the **RelCF** task). The difficulty of a course is expressed by the average grade achieved by the students that took the course in prior offerings. A positive instance is when the grade achieved is more than two letter grades lower than the average of the student's GPA and the course's prior average grade.

While the first two ways of labeling are absolute, the last two are relative to the student and the course of each instance. Statistics for the different classification tasks can be found at Table 5.1.

As discussed at the related work section, it is easier to predict the successful students. In order to have a better understanding of the relative difficulty of this task compared with the four tasks mentioned above, we also examined the task of predicting the students that completed a course with the grade A (denoted as the **Agr** task).

5.3.2 Extracted Features

Having as input the historical grading data, we want to study what factors can discriminate students with different performance. We derived from the original data different

features that capture different possible factors for a student's poor performance. The features can be separated into three distinct categories: the student-specific (they are independent from course c), course-specific features (they are independent from student s) and student- and course-specific features (they are a function of both s and c). All extracted features are described in the following list, where related features are grouped together into eight different subcategories. The keywords on bold are used to indicate the corresponding group of features later.

Note that for each $\{s, t, c\}$, where student s took course c in semester t, we generate a different set of features. Every set of features characterize a student's attempt to take course c at the specific point of his/her studies.

These features are either numerical, categorical or indicator variables. For indicator features, we use the values of 0 or 1. The categorical features are encoded via a numerical value. For example, the feature about the current semester is categorical, and the values {fall, spring, summer} are transformed to {0,1,2}, respectively.

Feature groups describing the target student s in the target semester t:

1. Student's status in terms of grades. (**grades**)

   - Average grade of s in prior courses C_{s,all_prior}. $\sum_j g_{s,j}/|C_{s,all\_prior}|$, for j in C_{s,all_prior}.
   - GPA of s, i.e., weighted average of the grades in prior courses w.r.t. the credits worth. $\sum_j g_{s,j} cr_j / \sum_j cr_j$, for j in C_{s,all_prior}. cr_j is the number of credits of course j.
   - GPA of s over the prior courses that belong in his/her major.
   - GPA of s over the prior courses that do not belong in his/her major.
   - GPA of s over the courses taken the previous semester, i.e., at the semester (t-1).
   - GPA of s over prior courses taken the past two semesters, i.e. at semesters (t-1) and (t-2).
   - GPA of s over prior courses taken on fall, spring and summer semesters. Essentially, here there are 3 features, one for each semester type.
   - Average grade of courses that s took with the same corresponding credit. There are 6 different features, each corresponding to prior courses that worth 1, 2, 3, 4, 5 or 6 credits.

- GPA of courses that s took with different course levels. There are 6 levels (1xxx, 2xxx, 3xxx, 4xxx, 5xxx, or 8xxx). Higher level courses are more advanced.

2. Other info indicating a student's status. (**status**)

   - The number of prior courses, $|C_{s,all\_prior}|$.
   - Student's major. Included majors: Aerospace Engineering, Biomedical Engineering, Chemical Engineering, Chemistry, Civil Engineering, Computer Science, Electrical Engineering, Materials Science, Mathematics, Mechanical Engineering, Physics, and Statistics.
   - The total credits that s has earned in prior courses $\sum_j cr_j$, for j in C_{s,all_prior}.
   - Indicator whether target semester t is a fall, spring, or summer semester.
   - Indicator whether the student has ever registered for the summer semester. This is an indicator of the past behavior of the student.
   - The number of semesters that the student is active, nterms_active$_{s,t}$.
   - The number of years student s is in the program.
   - The number of transferred credits. It is quite common for students to transfer some credits from other institutions, or from qualified courses they took at high school.

3. Student's course load. (**load**)

   - Average credits s earned per semester. $\sum_j cr_j$/nterms_active$_{s,t}$, for j in C_{s,all_prior}.
   - The number of credits s earned in the past semester. $\sum_j cr_j$, for j in $C_{s,t-1}$.
   - The number of credits earned in the current semester. $\sum_j cr_j$, for j in $C_{s,t}$.
   - The number of courses taken the current semester. $|C_{s,t}|$.
   - Ratio of s's course load in the current semester to his/her average course load over the past semesters. This is a way to compare the usual load of the student with the load for the target semester. $(\sum_i cr_i)/(\sum_j cr_j/\text{nterms\_active}_{s,t})$, for i in $C_{s,t}$ and j in C_{s,all_prior}.

4. Course's difficulty and popularity. (**c-diff**)

- Relative course load in semester t (when s took c) w.r.t. the average credits of past students at the semester they had taken c. For each past student, compute the number of credits earned on that semester. Then, compute the fraction of $\sum_j cr_j$, for j in $C_{s,t}$, divided by the average number credits earned from past students on the same semester that they took course c. Values greater than 1 indicate heavier load than other students.
- Average grade in c earned by past students.
- Average grade in c of past students within the same major as the s. Now, filter the students in order to keep only the students that are in the same department as s.
- Average grade in c of students belonging to c's major or not. This describes two features, by separating the past students to the ones that are in the same major as the department of c, and the ones that are out-of-the-department.

5. Performance / Familiarity with the course's background and department. (**c-backgr**)

   - Fraction of students in the same major as s that have taken the c. This feature measures how popular is course c across the students on the department of student s.
   - Fraction of students from s's major that took c. This feature shows how common is c in s's major.
   - Number of courses that s took and belong to c's department. Absolute measurement of how familiar is s with the department of the course c.
   - Ratio of courses that s took and belong to c's department. Relative measure of how familiar is s with the department of the course c.
   - Ratio of credits that s took and belong to c's department. Relative measurement of how familiar is s with the department of the course c, in terms of credits.
   - Ratio of credits that s took and belong to c's department and the average credits that past students took and belonged to c's department. This is a relative measurement of how familiar is s with the department of the course c, in comparison with past students.

- GPA over the courses that s took and belong to c's department. This feature is a quantitative measure of student's performance in the c's department.

6. Information about the prerequisites.(**prerequ**)

   - GPA of the prerequisite and non-prerequisite courses that s has taken. Two features that show the performance of the student in prerequisite and other courses.
   - Number of the prerequisite courses taken by s, an absolute measurement.
   - Ratio of prerequisite courses taken by s. Relative measure to show how well-prepared the student is, in terms of the stated prerequisites.
   - Average terms past since prerequisite courses were taken by s. This feature will indicate how recently the prerequisite knowledge was acquired.

7. Performance relative to the course's level.(**c-perform**)

   - The number of lower, same and higher level courses w.r.t. the level of c.
   - GPA over lower, same, higher level courses w.r.t. the level of c.

8. Course-specific features. (**c-spec**)

   - Course level that c belongs to.
   - Indicator whether c is in the student's major or not.
   - Average grade earned by past students.

In this list of features, the target student, course, semester are denoted as s, c, and t, respectively. The set C_{s,all_prior} represents the courses that the student took before the target semester t. For semester t, $C_{s,t}$ represents the set of courses that student s took on semester t. The term nterms_active$_{s,t}$ refers to the number of semesters that student s was active, i.e., taking courses, before semester t. The term cr_j refers to the credits that course j worths.

5.3.3 Methods Compared

In order to support students that need help to successfully complete a course, we will use classification techniques to identify them from the rest of the students. The instances of interest will be labeled as 1, and the rest as 0. The problem can be described as

follows. We are given a set of training examples that are in the form $(\mathbf{x}, y)$ and we want to learn their structure. We assume that there is some unknown function $y = f(\mathbf{x})$, that corresponds the feature vector $\mathbf{x}$ to a value y. In our case, $y = \{0, 1\}$. A classifier is an hypothesis about the true function f. Given unseen values of $\mathbf{x}$, it predicts the corresponding y values.

We tested the following classifiers [37], using scikit-learn library in Python [70]: Decision Tree (DT) [13] and Linear Support Vector Machine (SVM) [20] as base classifiers, and Random Forest (RF) [12] and Gradient Boosting (GB) [38] as ensemble classifiers.

While using Decision Trees, the classification process is modeled as a series of hierarchical decisions on the features, forming a tree-like structure. In other words, we ask a series of questions about the features of an instance, and based on the answer, we may ask more questions, until we reach to a conclusion about the class label of that instance. Each question, which is the splitting criterion, corresponds to an internal node of the tree, that separates the data into two or more parts. The goal is to get a split that results to as pure subsets (in terms of class labels) as possible, in order to make a confident prediction.

Consider the m-dimensional space that is defined by the feature vectors $\mathbf{x}$, of length m. There, every training instance corresponds to a single point. A Linear SVM looks for a decision boundary between two classes, a hyperplane that bisects the data with the largest possible margin between the two different classes. The margin on each side of the hyperplane is the area with no data points in it. Decision boundaries with large margins are expected to generalize well on previously unseen instances as they are robust to input uncertainty and noise.

Ensemble methods try to increase the prediction accuracy by combining the results from multiple weak classifiers. Assuming that base classifiers are more accurate than random guessing and that they are diverse, i.e., they make different errors on new entries, their combination will result in a more accurate classifier. Random Forest is a class of ensemble methods that uses decision trees as week learners. Randomness has been explicitly inserted in the model building process, as in each decision tree, every splitting criterion considers only a subset of features, randomly selected from the feature vector of $\mathbf{x}$, to select the best split. This is repeated for each iteration, till the tree is fully grown, without any pruning. Once we build all the trees, the majority class from the various predictions of a test instance is reported. The strategy of this method is to reduce the correlation among the constructed decision trees so that the strength of the ensemble

will improve.

In boosting, a weight is associated with each training instance, and the classifiers are constructed using these weights. In this iterative procedure, the weights are updated according to the classifier performance. Using the same algorithm, classifiers are trained on a weighted training set to focus on hard-to-classify instances. At the end of each iteration, the weights of instances with high misclassification error are relatively increased for future iterations. Each model created is applied to test instances and the final prediction is obtained by aggregating the predictions from the base classifiers. In Gradient Boosting for binary classification, a single regression tree is built, where in each splitting criterion, only a subset of the features is considered. Once the tree is built, then, the corresponding weight of the classifier in the current iteration is estimated.

5.4 Experimental Design

5.4.1 Training and Testing Methodology

The models constructed are global, i.e., a single model predicts the performance of all students over all the courses. We used a 5-fold cross-validation approach to assess the performance of the different models. The data are partitioned into 5 disjoint subsets. For each fold, we test on one partition and use the remaining ones for training. As randomization takes part in the models while sampling and/or initialization, we run the same model with 3 different seeds and average out the performance achieved. The average of the performance achieved over the 5 folds was used as the performance of this 5-fold cross-validation experiment.

Note that our evaluation methodology can lead to scenarios in which students from the same cohort are used for both training and testing. However, we did not include the test instances during the feature extraction phase of any student that needed information from other students taking the target course. Given this, the protocol still ensures that no information is shared across training and test data, directly or indirectly, because no information was passed from the test to the training set during feature extraction.

Additionally, over the years and during our very long dataset, it does occur some shifting in course characteristics, like course content, difficulty, and popularity. As a result, two versions of a course offered now and before 10 years will not be the same. The current experimental setting does not take that into account, as it uses all the data for feature extraction, and learns a single model for all years. This can be considered by

updating our models every semester or year and discarding k-fold validation. We could break the dataset into smaller subsets, that each includes a smaller range of years. As time progresses, we will utilize only a number of the most recent semesters for feature extraction and training. In this way, we can take into consideration these changes.

5.4.2 Metrics

Since the classes are unbalanced and the class of interest is the positive class (i.e., the students that fail, withdrawn from a class, etc.), the performance assessment metrics that we used focused on evaluating the performance on the positive class. We used three different metrics to compute the performance, which are the maximum F_1 score for the positive class, the area under the ROC curve, and the precision achieved by assigning a small percentage of test instances to the positive class.

All of the above metrics require that the classifier being evaluated assigns a score to each test instance indicating its prediction strength towards the positive class (i.e., the highest the score, the stronger the prediction towards the positive class is). Such a prediction score is naturally generated by the DT, RF and GB classifiers. For DT and GB, the predicted class probability is the fraction of samples of the same class in a leaf. For Random Forests, the prediction score of an input sample is computed as the mean predicted class probabilities of all the trees in the forest. Generating this for the SVM classifier is less obvious. During training time, we fit the data while computing and keeping track of the probability information. In this way, we can compute the probability of positive class assigned to a sample.

At a threshold θ, instances with prediction scores above θ are assigned to the positive class, otherwise to the negative class. For a given θ, we define $F_1(\theta)$ as:

$$F_1(\theta) = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{5.1}$$

where, recall measures the ratio of true positives to all actual positives and precision measures ratio of true positives to all predicted positives. The max F_1 score is the highest F_1 score achieved for any threshold θ, according to the following steps:

1. Sort the prediction scores in non-increasing order.

2. For each point L in this sorted sequence, compute the F_1 score, using Eq. 5.1, by assuming that any instances that have a prediction score that is greater than that

of the L-th instance are classified as positive and everything else is classified as negative.

3. The *max* F_1 score is the maximum F_1 value obtained.

The second metric is the area under the receiver operating characteristic (ROC) curve, AUC. The ROC curve plots the true positive rate against the false positive rate, at various thresholds. AUC corresponds to the probability that the classifier will rank a random positive instance higher than a negative one.

We also report precision@k%, which assigns the highest ranked k% of the test instances to the positive class and then measures their correctly classified fraction. Moreover, if there is a tie among some instances with the lowest prediction score that is considered to be positive, then all these instances are labeled as positive. An advantage of this metric is that it provides an easy to interpret quantity as to the accuracy of the classifier at different values of k.

5.4.3 Model Selection

We will present results for two ways of model selection. Firstly, we perform model selection and parameter tuning based on the max F_1 score, obtained as described above. As an alternative way of evaluation, we utilize the precision@k% measure. In that case, precision is evaluated at a given threshold, considering as positive only the top k% of the instances with the highest prediction score.

Since the percentage of positive instances varies across the different classification tasks, the value of k depends on the task we have at hand, to match the expected percentage of positive instances, assuming that the same trend will be followed by the test data as well. In our experiments we report the results for two values of k. The first value of k% refers to the percentage of instances that are at least as much as the percentage of observed positive instances, i.e., 5%, 5%, 24%, and 24% for Fgr, Wgr, RelF, and RelCF tasks, respectively. In this way, we make sure that a perfect model would be able to capture all the positive instances. We also evaluate precision@$k/2$%, where $k/2$ covers at least half of the observed percentage of positive instances in each classification problem, i.e., 2%, 2%, 12%, and 12% for Fgr, Wgr, RelF, and RelCF tasks, respectively. With this approach, we hope to more accurately predict a smaller set of instances. We believe that, even though precision at k% and at $k/2$% assess the performance at two discrete points, they provide insights as to how the classification methods and problems

discussed can be used in an operational setting (i.e., deployed at a University).

5.4.4 Overlap Between Two Models

In order to compare how much the predictions of two models (i and j) agree, we assume that TP_i and TP_j are their corresponding sets of instances that were correctly identified as positive, with cardinalities $|TP_i|$ and $|TP_j|$. The percentage of overlap between TP_i and TP_j is computed as: $\text{Overlap}(i,j) = 100 \times |TP_i \cap TP_j|/|TP_i \cup TP_j|$. If we take into consideration that $|TP_i| \neq |TP_j|$, we can scale overlap by the maximum possible overlap: $\text{Max Overlap}(i,j) = 100 \times \min(|TP_i|, |TP_j|)/\max(|TP_i|, |TP_j|)$, and obtain the *Scaled Overlap*:

$$\text{Scaled Overlap}(i,j) = \frac{\text{Overlap}(i,j)}{\text{Max Overlap}(i,j)}. \tag{5.2}$$

Scaled overlap takes values from 0 to 1. When it is 0, there is no overlap. When it is 1, there is as much overlap as possible, i.e., the positive instances of the one model are included in the positive predictions of the other model.

5.5 Results

5.5.1 Performance Analysis

Table 5.2 shows the performance of the various methods for the classification tasks, in terms of the AUC and the max F_1 score, when performing parameter tuning based on the max F_1 score.

Based on both metrics presented, GB is the best performing method, closely followed by the RF classifier. As expected, DT, which is the simplest method, has the lowest performance. These results are better compared to the performance of grade prediction methods for any classification task. When using Course-Specific Regression for predicting the failing students, we get a max F_1 score of 0.118, which is lower than any of the other methods we discuss.

While comparing the classification tasks, we can see that the tasks that predict relative performance have lower AUC values than when predicting absolute performance. In terms of max F_1 scores, we can see clearly that the A-students are the most accurately predicted. The F_1 scores of the different tasks are related to the percentage of positive instances in each task. If we consider the performance of a random classifier (which randomly predicts as positive the percentage of data equal to the number of positives),

Table 5.2: Performance of the various classifiers when model selection is based on F_1 scores.

Area under the ROC curve					
Classifier	Fgr	Wgr	RelF	RelCF	Agr
Random	0.500	0.500	0.501	0.501	0.500
DT	0.834	0.710	0.689	0.716	0.820
SVM	0.853	0.736	0.690	0.718	0.819
RF	0.873	0.778	0.748	0.759	0.850
GB	**0.877**	**0.780**	**0.755**	**0.765**	**0.854**
max F_1 score					
Classifier	Fgr	Wgr	RelF	RelCF	Agr
Random	0.034	0.027	0.232	0.222	0.216
DT	0.255	0.123	0.450	0.466	0.573
SVM	0.276	0.171	0.452	0.469	0.570
RF	0.317	0.165	0.499	0.502	0.604
GB	**0.319**	**0.181**	**0.506**	**0.507**	**0.610**

then we can achieve 9.5, 6.5, 2.3 and 2 times improvement for the tasks Fgr, Wgr, RelF and RelCF, respectively. The tasks Fgr and Wgr, that are highly unbalanced, have significantly lower F_1 scores, in a problem that is way more difficult though. Moreover, as there is 81% overlap between the students that are positive for both RelF and RelCF, the tasks of RelF and RelCF have very similar performance.

Table 5.3 presents the performance achieved in terms of precision, when model selection is based on precision@k% and precision@$k/2$%. Looking at the performance of the different models in terms of precision at k% and $k/2$% we see that GB tends to achieve the best performance, which is consistent with that previous observations.

These precision numbers also provide some insights as to what are the implications of actually using them to flag the students that may need to reconsider taking a particular course and potentially notifying them and/or their academic advisors. In general, the precision improves when we only consider the highest scoring predictions. If we look at the precision@k%, we can see that for the Fgr class, the models will be able to correctly flag 26% of the students, for the Wgr class precision is at 13.4%, whereas for RelF and RelCF, this number increases to around 46%. The corresponding numbers for the $k/2$% results are 35.5%, 19% and around 55%. As a result, when we use precision@$k/2$%, we also get fewer false positive instances.

Table 5.3: Performance of the various classifiers when model selection is based on precision scores.

Precision@k%				
Classifier	Fgr	Wgr	RelF	RelCF
DT	0.196	0.086	0.402	0.408
SVM	0.222	0.127	0.402	0.414
RF	**0.261**	0.123	0.468	0.459
GB	0.260	**0.134**	**0.475**	**0.466**
Precision@$k/2$%				
Classifier	Fgr	Wgr	RelF	RelCF
DT	0.249	0.111	0.456	0.451
SVM	0.296	0.178	0.456	0.452
RF	**0.365**	0.170	0.554	0.532
GB	**0.365**	**0.189**	**0.564**	**0.544**

Using the top k or $k/2$ percent of predictions (or even a smaller percentage) will be an academic unit's specific decision and needs to balance the costs associated with failing to warn a student about a course that he/she will potentially fail or not perform well in, versus having a high false positive rate. The latter case can either increase the workload of academic advisors (if they need to decide if the student will be warned or not) or increase the false positive rate to the point that students end up ignoring any warnings that were directly sent to them.

5.5.2 Feature Importance Study

In the problem that we examine, we do not know which are the most relevant influencing factors. In order to better represent the students, extracted features are introduced. Some of them may be irrelevant to the problem or repetitive. Our goals is to study which factors are important indicators of a student's performance. We performed the following experiment, that uses a wrapper method to evaluate the quality of the features[90]. We select a subset of features and use that subset to build a predetermined classifier. Evaluate the selected subset by the performance of the classifier, and repeat as much as possible in order achieve some desired quality.

Since the number of candidate features is quite high, the exhaustive search of all possible subsets is impractical. As a result, we shrank the search space by defining

the subsets ourselves. We categorize the extracted features to the 8 groups we formed. Afterwards, for each classification task, we built RF classifiers using only the features belonging to one of the above groups. We selected to use RF over GB, as they achieve similar performance in less training time. Model selection is performed based on the F_1 score.

The accuracy achieved for a model using a single group of features is expected to be less than the accuracy when using all the features. The percentage of accuracy that a model, using only the features belonging to one group, manages to achieve, in terms of the max F_1 score, are presented on Fig. 5.3. In this bar chart, we can see the percentage of accuracy achieved from all the different feature groups for all the discussed classification tasks. The higher the percentage achieved by a single group of features, the more predictive ability these features have.

From this figure, we can get many insights on the factors that affect student performance. For example, the features related to the students' grades have a very good predictive capability in almost all the tasks, except the task of predicting the W grades. In this task, features related with the course's difficulty and popularity as well as features that are course-specific, manage to achieve the same accuracy as when using all the features. This indicates that the reasons that a student drops a course are related more to the course, rather than to the students themselves. The next best indicator is the feature group about the student's course load during the semester.

On the other hand, this is not the case for predicting the failing students, in the absolute sense, i.e., receive a D or F. When using only course-related groups (c-diff, c-spec) for predicting the students likely to fail a course (Fgr task), we manage to recover half or less from the max F_1 score. As a result, these factors do not influence the absolute failing performance of a student, indicating that the reasons for that are mostly related with the student. As the students' grades manage to recover almost the same performance as when using all the features, they are the ones that affect the Fgr prediction the most. When using the other groups, it is very difficult to achieve comparable performance, as they recover 80% or less of the max F_1 score.

The feature groups are behaving similarly for RelF and RelCF. However, we notice that for the RelCF task, the feature groups that are related with student-course specific features have slightly better performance, while the student-specific groups have slightly worst performance, compared to the task of RelF. This is happening because, for RelCF, we take into consideration how other students usually perform on the target course.

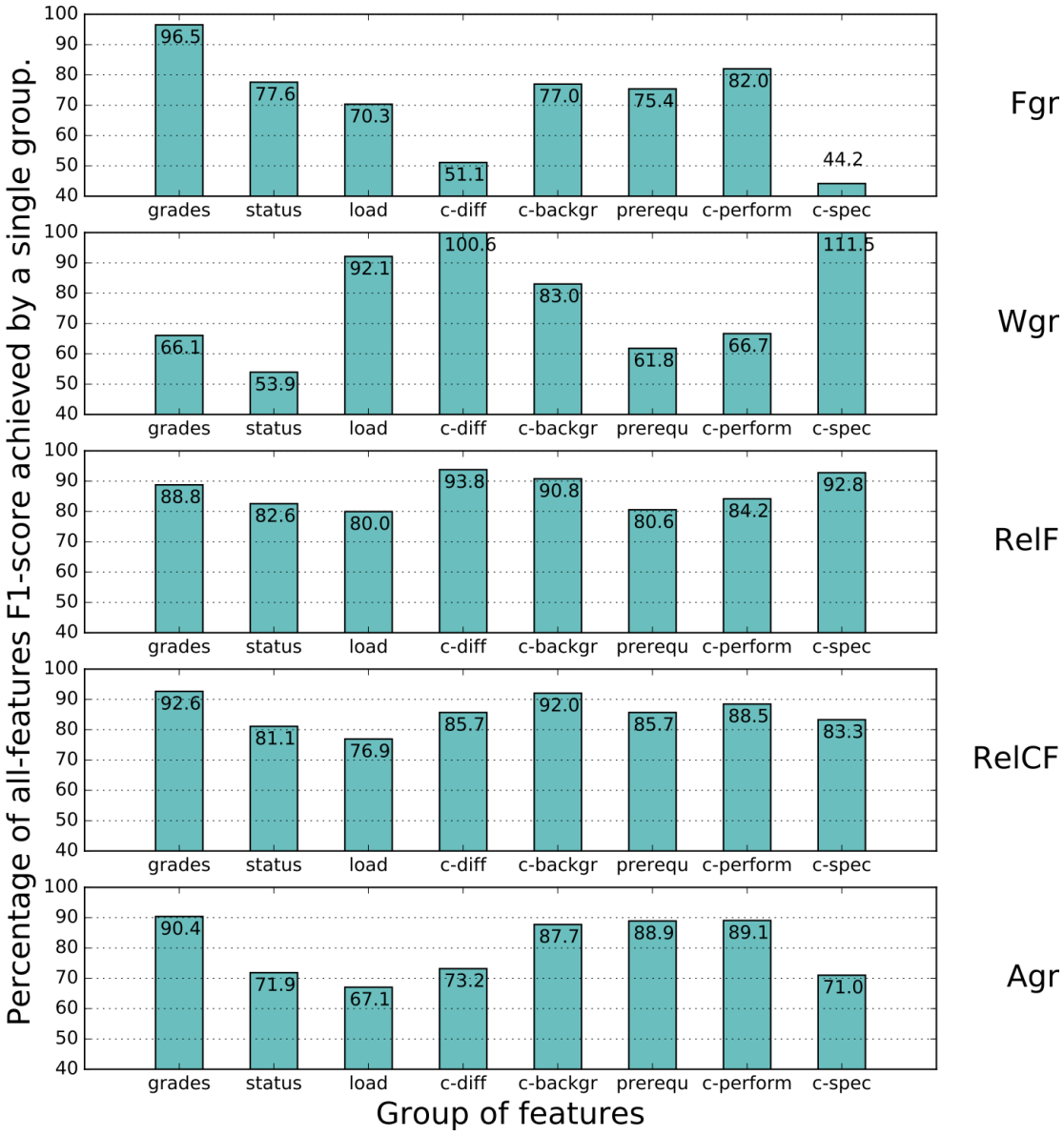


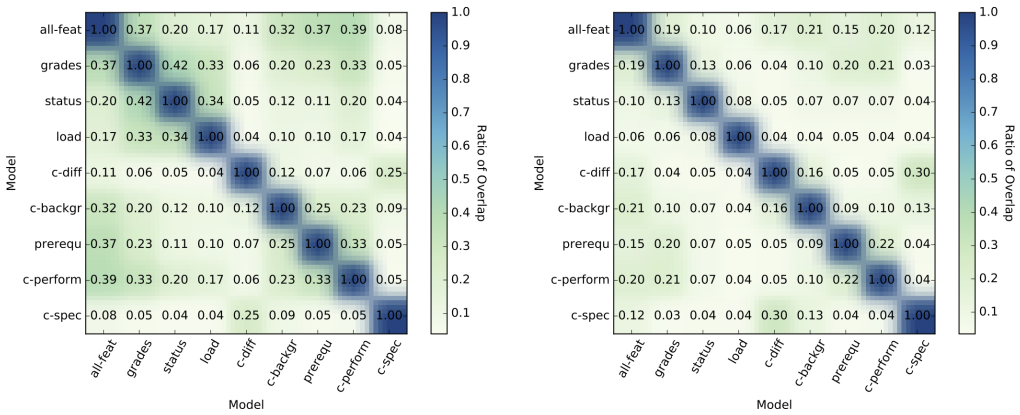
Figure 5.3: Percentage of performance managed to recover using only one group of features.

Every single group has enough information for the RF to utilize to achieve performance which is as good as 75% of the best case, i.e., when using all the features.

Finally, for identifying the A-students, the feature groups 1, 5, 6, 7 (grades, c-backgr, prerequ, c-perform) are the ones that manage to have the best performance. These groups are related with students' grades in general, but also, with their grades relative to the target course's background, prerequisites and level. Using only one of them can provide us with the information we need in order to recover around 90% of the performance while using all the features.

5.5.3 Comparing the Predictions of Different Models

Many of the models shown in Fig. 5.3 tend to achieve similar performance for the classification tasks. To see if that performance similarity is due to the fact that all of them identify the same set of students or not, we compare how similar are the positive top predictions of these models. We compare every pair of models in Section 5.5.2, and the best performing RF model with all the features. Specifically, we analyze the top k and $k/2\%$ of their predictions and use the scaled overlap metric (Eq. 5.2) to compute the overlap between the correctly predictive positive classes of these models (Fig. 5.4). Every (i, j) element in the heatmap is the average scaled overlap when combining models i and j, over the different seeds and folds. The heatmaps for precision@$k/2\%$ have similar patterns, but smaller overlap values. Surprisingly, for many of these models, the overlap between the best performing models is relatively low, suggesting that these models actually identify different subset of problematic student-course instances.



(a) Fgr (b) Wgr

(c) RelF (d) RelCF

Figure 5.4: Scaled Overlap of the models that use a single feature group. The two columns correspond to the experiments that are based on precision@k%.

Chapter 6

Course Recommendation

6.1 Introduction to the Problem

This chapter presents our work in course recommendation that facilitates course selection for undergraduate students. In a higher education setting, students register for the courses they will take next semester in advance. Students have the freedom to select at least a number of elective courses out of a plethora of courses from their major or any other department in the university. Students shape their degree plan according to their personal preferences, goals and interests. Based on past course selections, we need to generate a list of courses for the next semester that best matches the student's degree progress.

6.2 Proposed Method

6.2.1 Assumptions

In the context of course recommendation for higher education, we make the following assumptions:

1. Time is discrete and moves in steps, from one semester to the next.
2. There is a relative ordering of the courses in terms of course levels, difficulty or material covered.
3. Learning is seldom non-sequential; each course completed provides some knowledge and experience that can be used in future courses. As a consequence, sequence matters in course selection.

Table 6.1: Notation.

n, m	number of courses, students
i, i'	indexes for courses
j, j'	indexes for students
t_j	number of semesters that j has taken courses
t	index for semesters
$\mathcal{C}$, $\mathcal{S}$	set of courses, courses
$\mathcal{A}$	set of states in Markov chain $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$
$\mathcal{H}_j$	enrollment history of student j
$\mathcal{C}_{j,t}$	course that student j took in semester t
$\mathbf{T}$, $\mathbf{F}$	matrices $(n \times n)$
$\mathbf{T}_{k,l}$	the (k, l) element of matrix $\mathbf{T}$
$\mathbf{T}_k$	the k-th row of matrix $\mathbf{T}$ $(1 \times n)$
$\mathbf{T}_{:,l}$	the l-th column of matrix $\mathbf{T}$ $(n \times 1)$
$\mathbf{u}$	personalization vector $(1 \times n)$
$\mathbf{p}^{(k)}$	state vector $(1 \times n)$ at timestep k

4. In the absence of enough domain experts, the order in which courses are taken by students historically can reveal useful information on the curriculum and degree requirements.
5. We know the number of courses that the student will take next semester.

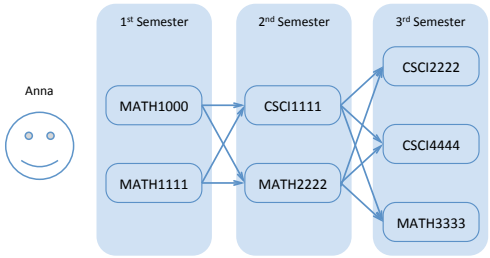
When we use the word *target* we will refer to the student/course/semester for which we want to generate recommendation.

Student j has an enrollment history $\mathcal{H}_j$, that is an ordered set of courses, which we denote by $\{\mathcal{C}_{j,1}, \ldots, \mathcal{C}_{j,t}, \ldots, \mathcal{C}_{j,t_j}\}$, where $\mathcal{C}_{j,t}$ is the set of courses taken in semester t and t_j is the last semester that the student took courses. Table 6.1 presents the symbols we used.

6.2.2 Building the Markov Chain

Markov models satisfy the Markov property, i.e., the conditional probability distribution of future states depends only on the current state. In the simplest Markov model, known as first-order, each state is formed by a single action, i.e., a student took a course. In the case of K-th-order models, the state-space will correspond to all possible sequences of K actions. As the available data could not adequately support the number of states of higher-order chains, these models would suffer from reduced coverage and possibly

Figure 6.1: Example: Anna's enrollment history.



worse overall performance [25]. Therefore, we adopted a first-order Markov chain. We assume that the next-semester courses depend only on the courses that the student is taking the current semester.

Markov models are represented by the parameters $\langle \mathcal{A}, \mathbf{T} \rangle$, where $\mathcal{A}$ is the set of states for which the Markov model is defined; and $\mathbf{T}$ is an $(n \times n)$ transition probability matrix (TPM), where n is the number of states (i.e., courses). In this context, state $\mathcal{A}_i$ is associated with the fact that the student took the course i. Each entry $\mathbf{T}_{i,i'}$ corresponds to the probability of moving to state $\mathcal{A}_{i'}$ when the process is in state $\mathcal{A}_i$, i.e., taking course i' after course i. Note that this matrix is not symmetric, i.e., $\mathbf{T}_{i,i'} \neq \mathbf{T}_{i',i}$, as the order in which the courses are taken matters.

Based on the historical enrollment information of the students, we first compute $\mathbf{F}$, an $(n \times n)$ matrix that holds the counts of every pair of consecutive courses. Every pair of courses (i, i') that a student has taken consecutively is used to estimate the entry $\mathbf{F}_{i',i}$, i.e., the frequency of the event that state $\mathcal{A}_{i'}$ follows the state $\mathcal{A}_i$. For example, consider student Anna in Fig. 6.1. The entry corresponding to the course pair of (MATH1000, CSCI1111) will be updated. Similarly, every line connecting two courses will equally contribute in the corresponding element of matrix $\mathbf{F}$.

After we compute the frequencies of matrix $\mathbf{F}$, we need to normalize it to get $\mathbf{T}$, a row stochastic matrix, so that the total transition probability from state i to any other state will sum up to 1:

$$\mathbf{T}_i = \mathbf{F}_i / \sum_{i'=1}^{n} \mathbf{F}_{i,i'}, \text{ if } \sum_{i'=1}^{n} \mathbf{F}_{i,i'} > 0.$$

Additionally, it is possible that the sum of some rows to be zero. This occurs when a course is taken at the last semester of every student, so there are no courses after that to

pair it with. In that case, we set the diagonal elements of the zero rows to one; $\mathbf{T}_{i,i} = 1$ and $\mathbf{T}_{i,i'} = 0$ for $i \neq i'$, if $\sum_{i'=0}^{n} \mathbf{F}_{i,i'} = 0$.

6.2.3 Walking Over Courses

We can view the Markov chain in the context of random walk on a course-to-course graph that is governed by the transition probability matrix. A random walk on a directed graph will form a path of vertices generated from a start vertex by selecting an edge, making a step by traversing the edge to a new vertex, and repeating the process [10]. This concept has been applied to many scientific fields. Closer to this work, random walks have recently been used for top-n item recommendation [62], and they are also known to empower systems used in production at major social media platforms [29, 40].

A random walk starts with any probability distribution $\mathbf{u} \in \mathcal{R}^{1\times n}$. $\mathbf{u}_i$ is the probability of starting at vertex i. If one starts at a vertex i, then $\mathbf{u}_i = 1,$ else $\mathbf{u}_{i'} = 0$ for $i' \neq i$. In our setting, the random walk for student j will equally start from any course in the student's last semester, so the personalization vector will be:

$$\mathbf{u}_i = \begin{cases} 1/|\mathcal{C}_{j,t_j}| & \text{if } i \in \mathcal{C}_{j,t_j}, \\ 0 & \text{otherwise.} \end{cases} \tag{6.1}$$

Let $\mathbf{p}^t \in \mathcal{R}^{1\times n}$ be a row vector with an element for each vertex specifying the probability of being there at time t. Before we start the walk, $\mathbf{p}^0 = \mathbf{u}$. After the first step, the probability of being at vertex i' is the sum over each adjacent vertex i of starting at i and taking the transition from i to i'. In matrix notation, when we are at state k and we take a step, we will get the following probability distribution:

$$\mathbf{p}^{k+1} = \mathbf{p}^k \mathbf{T}, \tag{6.2}$$

where the i-th entry of the $\mathbf{p}^{k+1}$ is the probability of the walk after $k+1$ steps to land at vertex i. This can be written as a function of the starting probability vector as:

$$\mathbf{p}^{k+1} = \mathbf{u}\mathbf{T}^k. \tag{6.3}$$

The probability of the walker to reach the vertices after K steps provides an intuitive measure that can be used to rank the courses and offer personalized recommendations to the student accordingly.

Scholars Walk

To introduce an additional way for personalization in our model, we perform a *random walk with restarts* [66]. We introduce a parameter α, $0 < \alpha \leq 1$ that controls if the walk will take the step described above, or if the walk will restart. In the latter case, we use the personalized probability distribution as the restarting distribution. The probability distribution now is defined as:

$$\begin{aligned} \mathbf{p}^{k+1} &= \alpha \mathbf{p}^k \mathbf{T} + (1-\alpha)\mathbf{u} \\ &= \mathbf{p}^k(\alpha \mathbf{T} + (1-\alpha)\mathbb{1}\mathbf{u}) \\ &= \mathbf{u}(\alpha \mathbf{T} + (1-\alpha)\mathbb{1}\mathbf{u})^k, \end{aligned} \tag{6.4}$$

where $\mathbb{1}$ is a column vector $(n \times 1)$ of ones. The product of $\mathbb{1}\mathbf{u}$ will give us an $(n \times n)$ matrix where every row will have the probability that the walk will start at the corresponding course. Scholars Walk will perform a random walk governed by the matrix $\alpha\mathbf{T} + (1-\alpha)\mathbb{1}\mathbf{u}$.

The exact steps we followed are shown in Alg. 1. We can specify the number of steps to perform, or we can allow the algorithm to converge. If the number of steps is very small, the walk might not explore enough courses. If the number of steps is large, the walk might travel too far, and the recommendations might not be so relevant for the student. Additionally, to limit the domination of popular courses, we penalize the probabilities with the term $\text{pop}_i^{-\beta}$ [19], where pop_i is the popularity of the course. The parameter $\beta, 0 < \beta \leq 1$ shows how harsh we need to be with the penalty term.

Scholars Walk allows us to consider direct, as well as, transitive relations between the courses. It also provides a considerable degree of personalization, in order to recommend courses that are relevant to each particular student.

6.3 Experimental Design

6.3.1 Dataset

This work focuses on the undergraduate students in a traditional educational institution. We used a dataset from the University of Minnesota that spans more than 10 years. The A–F grading scale (A, A-, B+, B, B-, C+, C, C-, D+, D, F) is used. Courses in which a student receives less than a C- do not count toward satisfying degree requirements.

We extracted the degree programs that have at least 500 graduated students from 23

Algorithm 1 SCHOLARS WALK

Input: Model $\mathbf{T}$, student's personalization vector $\mathbf{u}$, parameters α, β, number of steps K.
Output: Recommendation vector $\mathbf{p}^{rec}$.
$\mathbf{p}^0 \leftarrow \mathbf{u}$, $k \leftarrow 0$
repeat
  $k \leftarrow k + 1$
  $\mathbf{p}^k \leftarrow \alpha \mathbf{p}^{k-1}\mathbf{T} + (1-\alpha)\mathbf{u}$ ▷ Take a step.
  $\mathbf{p}^k \leftarrow \mathbf{p}^k / \|\mathbf{p}^k\|_1$ ▷ Normalize $\mathbf{p}^k$.
until $\|\mathbf{p}^k - \mathbf{p}^{k-1}\|_2 < tol$ or $k \geq K$
for $i \leftarrow 1$ to n **do**
  $\mathbf{p}^k_i \leftarrow \mathbf{p}^k_i * \text{pop}_i^{-\beta}$ ▷ Penalize popular courses.
$\mathbf{p}^{rec} \leftarrow \mathbf{p}^k$

different majors. We only kept students that actually received their degree and had at least three consecutive semesters with valid courses. We selected the 40 most frequent courses and the courses that belonged to frequent subjects. A subject is considered frequent if students have taken at least three courses that belong to that subject on average. We removed instances without an A–F grade, and non-academic courses, like independent/directed study or field study. We did not consider offerings in the summer semester. As these are less common, they would distort the course sequence of students not enrolled in summer. Basic statistics for each degree program are shown in Table 6.2. The average course popularity (%pop) for course i is the percentage of students that have taken i at least once during their studies. The degree flexibility (flex) is a measure of how different are the course selections that students make. It is one minus the average Jaccard similarity coefficient for every pair of students. The Jaccard similarity is computed as the number of courses that two students have in common divided by the minimum courses that student has taken them.

6.3.2 Competing Approaches

The baselines are two group popularity approaches, on the department level (**Pop1**) and the academic level (**Pop2**) of the student measured by the number of years in the program [30]. For Pop1, we recommend the most popular courses in the major. For Pop2, we recommend the most common courses on the major and the academic level of the student ("freshmen", "sophomores", "juniors", and "seniors"). Students after their

Table 6.2: Statistics for each major.

Major	n	m	grades	%pop	flex
Accounting	846	53	22,524	45.9	0.28
Aerospace Engr	532	109	16,259	25.7	0.10
Biology	1,275	146	28,084	14.9	0.11
Biol Society Env	709	57	14,597	31.4	0.31
Biomedical Engr	644	131	19,748	23.8	0.16
Chemical Engr	826	108	24,825	26.3	0.11
Chemistry	724	145	18,292	17.4	0.14
Civil Engr	651	112	19,189	26.5	0.12
Communication	1,333	95	22,421	15.4	0.19
Computer Sc	998	161	24,899	13.7	0.11
Electrical Engr	740	164	22,191	17.1	0.12
Elementary Ed	770	49	16,527	40.4	0.31
English	1,176	153	17,736	9.5	0.11
Finance	1,234	83	32,255	29.7	0.20
Genetics Cell	680	93	15,385	23.1	0.19
Journalism	2,306	100	40,519	17.1	0.20
Kinesiology	1,176	164	33,622	14.8	0.16
Marketing	1,291	69	29,901	30.8	0.20
Mechanical Engr	1,369	132	39,436	18.9	0.11
Nursing	819	86	25,136	31.2	0.27
Nutrition	554	87	15,591	29.7	0.19
Political Science	1,307	171	19,260	8.1	0.12
Psychology	1,894	115	31,141	13.4	0.15

n, m are the number of students and courses.
%pop is the course popularity (percentage of students that took a course at least once).
The last column (flex) is the degree flexibility.

forth year are considered seniors.

We also compared against Basic Markov model (**Markov**) and Basic Markov model with skip (**MarkovSkip**) [47]. In these models, for a target student, the set of courses that other students have taken after taking a course that the target student took are the possible courses to recommend. We consider the combination of courses during the last two semesters to build and test the model. Each course is assigned a recommendation score that is the sum of all the conditional probabilities that lead to that course starting from the student's enrollment in the last semester. While the counts used in this case are the same with the ones computed in our matrix **F**, the conditional probabilities are computed differently. In order to produce recommendations for students whose set of prior courses did not have a match, the skip model was introduced. In that case, we find other students that have similar course history with the target student, and weight their corresponding probabilities by a parameter λ.

Last, we train an **LSTM**-based course prediction model similar to [60, 68]. LSTMs can learn temporal dependencies with additional gates to retain and forget selected information. As input, we use a multi-hot representation of course enrollments per semester which are mapped to a predicted sequence of vectors. Once the LSTM has been learnt, we feed the network with a binary vector that indicates the courses that the target student has taken the past semester. The weights at the output of the model are used to rank the courses.

6.3.3 Evaluation Metrics

Like in prior work [30, 47, 60, 68], we used **Recall@**n_s as the primary evaluation metric for the predictions, where n_s is the number of courses that the student took in the target semester. This is the percentage of actual enrolled courses that were contained in the recommendation list. The reported metrics are averaged out across all students predicted. Note that recall and precision are equivalent in our setting, since we recommend exactly as many courses as the student will take the upcoming semester.

We also compute the percentage of queries for which we were able to retrieve at least one of the courses that the student took in the target semester (**%rel**). It measures for how many cases we were able to recommend at least one course that was relevant.

6.3.4 Experimental Setting

Model selection. Using the dataset described in Sect. 6.3.1, we split it into train, validation and test sets as follows. All semesters before 2013 (about 10 years) were used for training, courses taken during 2013 and in Spring 2014 were used for validation, and courses taken afterwards (Fall 2014 to Spring 2017) were used for test purposes, to report the results. The training set was used for building the models, whereas the validation set was used to select the best performing parameters in terms of the highest Recall@n_s. Based on the best set of parameters for the validation set, we computed the test set results in Sect. 6.4.

Parameters. For parameter α, we tried the following set of values: {1e-4, 1e-3, 1e-2, 1e-1, 0.2, 0.4, 0.6, 0.7, 0.8, 0.85, 0.9, 0.99, 0.999}. For parameter β, we tested values from 0 to 0.8, in increments of 0.025. In terms of the number of steps that we allowed for our walker, we tested the values 1, 3, and 1000. The last value corresponds to no limit for the number of steps.

Additional filtering. We build a different model for each major for all the approaches we tested. After we generate a ranked list of the courses using any method, we filter out courses that are not offered the target semester. We also remove courses that the student has taken in the past and achieved a grade above C-, as they do not count towards any degree requirements, as mentioned in Sect. 6.3.1. In the end, we return a list with as many recommendations as the number of courses, n_s, that the student took next semester, based on assumption 5.

6.4 Results

In this section, we will try to answer the following questions: 1) How do the parameters in our models affect the overall performance? Specifically, how does the number of steps affect recommendation performance? 2) What is the performance of our approach compared to the state-of-the-art approaches?

6.4.1 The Effect of the Number of Steps

The performance of our models in terms of the metrics computed for different values of K is shown in Table 6.3. For each model and selection of K, we see the values of the parameters α and β that were used. These parameters were selected based on the recall on the validation set. The parameter α controls the restarting probabilities, while β is

Table 6.3: Results for Scholars Walk w.r.t. K.

K	Recall@n_s	%rel	α	β	avg#steps
1	0.466	75.1	0.955	0.047	1
3	0.460	74.6	0.088	0.053	1.95
1000	0.461	74.6	0.075	0.051	2.32

K is the number of steps that we allow to our walker.
α, β columns show the average values of these parameters over the models of all the majors.
The last column shows the actual average number of steps the Scholars Walk made before convergence.

used to re-weight the probability distribution before recommending its highest-weighted courses. The column avg#steps shows the average number of steps that the Scholars Walk actually made before convergence.

In this domain, we need only a few steps, as we can understand from Table 6.3: not only when we set $K = 1$ we get the best performance, but also, when we allow the walk to take many steps, the parameter α gets smaller values. This forces the walk to go back to the student's personalized starting vector with higher probability, indicating that the starting distribution is very important. Additionally, even if we do not put any constraints in K, the number of steps that the Scholars Walk takes is quite small. There is a small increase when increasing K from 1 to 3, but after that, the number of steps actually taken is not that high.

It is worth pointing out that, while setting $K = 1$ gives us the best overall performance, this is not the case for all the departments. The right value for K depends on the dataset used. In our data, there are four departments that need these extra steps. We observed that these departments have low average course popularity, which is average percentage of students that have taken a course at least once at some point during their studies, over all the courses. The average value for the departments with $K > 1$ was $16.7 \pm 9.7\%$, while for the rest of the models the corresponding number is $24.1 \pm 7.2\%$. A stronger signal is present in the metric of the degree flexibility, which is the average Jaccard distance between the courses that any pair of students took, as defined in the end of Sect. 6.3.1. The departments with $K > 1$ have 0.118 ± 0.005 degree flexibility against 0.184 ± 0.066 of the rest of the departments. This is an indicator that for stricter degrees, the walk depends on the extra steps to explore more courses. In

Table 6.4: Performance comparison.

Model	Recall@n_s	%rel
Pop1	0.336	62.5
Pop2	0.338	64.6
Markov	0.456	73.0
MarkovSkip	0.400	69.6
LSTM	0.406	69.6
Scholars Walk	**0.466**	**75.1**

these departments, students will take overall very similar sets of courses. On the other hand, if the degree program offers more freedom to the students, they select a wider range of courses, and there are more connections within courses.

6.4.2 Performance Comparison

By comparing the best Scholars Walk model against five competing approaches, we get the results on Table 6.4. Our model performs the best, both in terms of recall, and in the percentage of cases for which it manages to be return some relevant recommendations.

Popularity approaches are having considerably satisfactory performance. However, specifying the academic level of the student does not help much. They can recommend relevant courses to more than 60% of the cases. The two Basic Markov models have quite different performance. The Markov model with skips performs poorly, compared to the Basic model. Additionally, it is worth mentioning that the Skip model was performing better and better as the parameter λ was getting smaller. The weight of the cases that do not completely match the target student's history, have as weight a power of λ. Consequently, when $\lambda \to 0$, the Skip model becomes the Basic Model. For that reason, the smaller value of λ that we report results for, is 0.4.

While comparing the Basic Markov model with Scholars Walk, it may seem that they have similar performance. However, that might be misleading, as the Basic Markov model utilizes longer course enrollment history than the Scholars Walk. It looks back two semesters on the student's courses, which corresponds to a second-order Markov chain. Moreover, the model uses data from two semesters not only for computing the associated probabilities, but also to make predictions. This leads to increased complexity because of the larger state-space with no benefit in recommendation quality. In the same boat are the LSTMs as well. Their increased complexity might lead to the overfitting of the

model, when the data are not sufficient for training. Our approach, which is a first-order Markov chain, manages to perform better than the higher-order models and LSTMs.

Scholars Walk can accurately predict the course selection of the students, by taking advantage of the "breadth and depth" of the data. In terms of time complexity, once we build the transition probability matrix, walking through the courses is trivial. As a result, it scales well with the number of students, while providing them personalized recommendations. At the same time, it is a white-box model, where the recommendations are easily explainable.

Chapter 7

Fair Course Recommendation

7.1 Introduction to the Problem

An optimal recommendation system provides high-quality suggestions to all students that best align with their degree progress. If that CRS is not fair, we can carefully alter the recommendation lists in order to balance the system's behavior towards students' from different protected groups. The challenge that arises in this context is that we need to improve fairness without significantly altering the quality of the recommendations, but also, ensure that the system still offers equally good recommendations to the students in the different protected groups.

We studied the problem of fairness in course recommendation motivated by work on equality in educational opportunity [83]. This chapter's contributions include: 1) a new definition of fairness in recommender systems, FaiREO, that captures the equality of opportunity, 2) a multi-objective optimization problem formulation to consider FaiREO in the recommendation, and 3) a set of steepest-ascent hill climbing algorithms to solve this problem.

7.2 Problem Formulation

7.2.1 Assumptions

In this work, we make the following assumptions:
(Assumption 1) The student body has at least one protected attribute based on which we can form protected groups of students (or simply, student groups). When there are

Table 7.1: Notation used within the context of fair course recommendation.

i, j	Index for students, courses.
p, q	Index for student groups, course buckets.
$\mathcal{S}, \mathcal{C}$	Set of all students, courses.
n, m	Number of all students, courses.
g_s, g_c	Number of student, course subsets formed.
$\mathcal{C}_{1,\ldots,g_c}$	Subset of courses.
$\mathcal{S}_{1,\ldots,g_s}$	Subset of students.
$\mathbf{R}$	The recommended courses for all students.
$\mathbf{Y}$	The $(n \times m)$ recommendation score matrix of the optimal solution.
$\mathcal{R}_i$	Set of recommended courses for student i, $\mathcal{R}_i \subset \mathcal{C}$.
k	Number of courses we recommend, i.e., $\lvert\mathcal{R}_i\rvert$.
n^j	Number of students to whom we recommend course j, i.e., $\lvert\{i \text{ s.t. } j \in \mathcal{R}_i\}\rvert$.
n_p	Number of students that belong in the $\mathcal{S}_p$ group.
n_p^j	Number of students that belong in $\mathcal{S}_p$ group to whom we recommend course j, i.e., $\lvert\{i \in \mathcal{S}_p \text{ s.t. } j \in \mathcal{R}_i\}\rvert$.
$y_{i,j}$	Recommendation score for student i taking course j.

more than two protected attributes that we need to consider, we create a protected group for each combination of values that they take.

(Assumption 2) We have access to a method that computes the recommendation scores for all the courses a student might register in the next semester. The scores accurately capture how well a course matches the student's academic level, background, and knowledge. Any courses that the student has already taken receive zero recommendation score. We will refer to the solution that recommends the k highest scored courses for each student as the **HSC** solution.

(Assumption 3) When students receive a recommendation from a CRS, they will have the chance to at least consider taking that particular course next semester. As a result, by recommending courses to students, we are giving them the opportunity to explore courses that they might not have considered otherwise.

Notation. As a reminder, capital calligraphic letters will be used for sets. Lower bold case letters will indicate vectors, e.g., $\mathbf{f}$, and their elements will be denoted by regular lower case letters, e.g., f_p. Table 7.1 presents the symbols we use in this chapter. If there is a subscript, it indicates the particular subset it refers to.

7.2.2 Fairness in Recommendation with Equality of Opportunity

In higher education, students can register for courses that best match their interests, strengths, and goals. However, their judgment is often affected by already existing biases and socio-technical issues, as well as other people's actions and opinions. As a result, course enrollment data exhibit historical and social biases [57]. A course recommendation system will use that input data to generate recommendations for the users. Even though a CRS will not purposefully differentiate its users, it may propagate the biases found on the input data to the output. For example, such a system would rarely recommend heavy-coding classes to female students in a computer science department. This could be a female student's, Anna's, only chance to consider such a class since most of her female friends (stereotypically) view computer programming as a male nerd field. Such a programming class could have provided Anna with the experience needed to successfully apply for a software engineering job. She will now have a disadvantage against her male classmates that have taken similar courses, and she might have trouble finding an equally well-paid job. A fair CRS would not discriminate against Anna, simply because she is female. We consider that a recommendation is an opportunity for the student to consider taking the corresponding course. A fair CRS would ensure that students in all protected groups are offered the same opportunities (in terms of course recommendations).

Our interest is on equal opportunity, but we still need to consider our initial goal in a CRS: support students by offering them recommendations of good quality. While these two aspects add on the value of a recommendation system, they can be conflicting as well. We assume that the HSC solution offers the highest quality output, but there are no guarantees for the equality of the opportunities it offers. On the other hand, if we modify the recommendation lists to satisfy equality of opportunity, some recommendations will be of lower quality. As a result, there is the need to balance these two goals. At the same time, we need to ensure that the equality of opportunity does not come at the expense of recommending bad courses to a protected group. A recommendation system is fair when it offers the same quality of service to users, irrespective of their inherited or acquired characteristics captured by the protected attributes.

Motivated by the above discussion, we introduce a new type of group fairness, referred to as *fairness of equality of opportunity* (FaiREO), which is defined as follows:

Definition 7.2.1. Let $\mathcal{S}$ be a population of students, that can be divided based on the value of one or more protected features into g_s groups, $\mathcal{S}_1, \ldots, \mathcal{S}_{g_s}$, with cardinalities $n_1, \ldots, n_{g_s}$, respectively. A course recommendation system satisfies **fairness for**

equality of opportunity, FaiREO, when:

1. Each student group gets a share of each course's recommendations relative to its size. Let n_p^j be the number of students in group $\mathcal{S}_p$ to whom we recommend course j. Recommendations w.r.t. course j offer equal opportunities when:

$$\frac{n_p^j}{n^j} \approx \frac{n_p}{n}, \quad \forall p \in \{1, \ldots, g_s\}. \tag{7.1}$$

2. All student groups equally receive recommendations of high quality with respect to the courses' recommendation scores, i.e.,

$$\sum_{i \in \mathcal{S}_p} \sum_{j \in \mathcal{R}'_i} y_{i,j} \approx \sum_{i \in \mathcal{S}_p} \sum_{j \in \mathcal{R}_i} y_{i,j}, \quad \forall p \in \{1, \ldots, g_s\}, \tag{7.2}$$

where $\mathcal{R}'_i$ is a set of recommended courses for student i, $\mathcal{R}_i$ the set of courses recommended based on the HSC solution, $y_{i,j}$ denotes the score of student i in course j, and $\mathcal{S}_p$ denotes the students in the protected group p.

We will refer to the first and second condition as the *fairness conditions* and *quality conditions*, respectively.

7.3 Methods

7.3.1 Objective Functions

Fair course recommendation according to FaiREO is a multi-objective optimization problem that simultaneously tries to satisfy both conditions of equal opportunity and quality, as described in Definition 7.2.1. We define two different objective functions (F and Q) to capture each condition, and then we linearly combine them into our overall objective function.

Opportunity Condition

We quantify the first condition of fairness in opportunity by using the mismatch between the two quantities of the Eq. 7.1, i.e., the distance of course j from the fair ratio that corresponds to the protected group p ($= n_p/n$). We compute the *fraction of the*

recommendations that introduce unfairness in the protected group p as:

$$f_p = \frac{1}{n_p k} \sum_{j=1}^{m} \left(n^j \left| \frac{n_p^j}{n^j} - \frac{n_p}{n} \right| \right), \tag{7.3}$$

where n is the number of students, m is the number of courses, n_p is the number of students belonging in group p, k is the number of courses we recommend to the student, n^j is the number of students to whom we recommend course j, and n_p^j is the number of students in group p to whom we recommend course j. The term of the absolute difference captures how far away we are (measured by the ratio of recommendations) from balancing the opportunities offered in group p regarding course j. The term in the parenthesis corresponds to the number of students that introduce this unbalance in the recommendations of j to group p. Note that the overall sum is normalized with the number of recommendations generated for group p (k courses for every one of the n_p students) in order for f_p to be invariant of the group size. The opportunity objective function targets to minimize the unfairness in opportunity existing in the recommended lists of courses:

$$F = \min \|\mathbf{f}\|_x, \text{ where } \mathbf{f} = [f_1, \ldots, f_{g_s}]. \tag{7.4}$$

Quality Condition

To quantify the quality objective, we use the recommendation scores of the courses. We measure the quality of the suggested courses by the summation of their recommendation scores. We will capture how different is the quality of a solution compared to the solution that recommends the highest scored courses to the students (HSC solution). We formulate the *fraction of quality loss* for each group p as:

$$q_p = \frac{\sum_{i \in \mathcal{S}_p} \left(\sum_{j \in \mathcal{R}_i} y_{i,j} - \sum_{j \in \mathcal{R}'_i} y_{i,j} \right)}{\sum_{i \in \mathcal{S}_p} \sum_{j \in \mathcal{R}_i} y_{i,j}}, \tag{7.5}$$

where $\mathcal{R}'_i$ is a set of recommended courses for student i, and $\mathcal{R}_i$ the set of courses recommended based on the HSC solution. $y_{i,j}$ is the recommendation score of student i in course j, and $\mathcal{S}_p$ is the subset of students in group p. The summation in the numerator is the difference in the quality of the two solutions. We normalize it by the quality of the HSC solution that recommends the top scored courses for every student to make it invariant of the size of the protected groups and their quality. The quality objective

function that minimizes the quality loss is:

$$Q = \min \|\mathbf{q}\|_x, \text{ where } \mathbf{q} = [q_1, \ldots, q_{g_s}]. \tag{7.6}$$

Combined Objective Function

There is a trade-off between the two objectives, as optimizing for equality in opportunity will replace the highest-scored courses with others that have the same or lower scores. This will probably result in recommendations with lower-scored courses than the HSC solution, which will incur quality loss. The combined objective function is:

$$V = aF + (1 - a)Q. \tag{7.7}$$

The parameter α weights the importance of each objective, and it can be adjusted by the system administrator. We can use any norm greater than 1, which penalize high values, to aggregate $\mathbf{f}, \mathbf{q}$ for all student groups.

7.3.2 Greedy Hill Climbing (GHC) Algorithms

The fair recommendation of courses to students is a multi-objective, combinatorial optimization problem described by Eq. 7.7, where the process of searching for minima of the objective function involves a discrete but large configuration space. That space cannot be exhaustively searched, as there are $\binom{m}{n}^n$ possible combinations to examine, where n, m, and k are the number of students, courses, and recommended courses per student, respectively.

Our approach uses the steepest ascent hill climbing technique with a greedy strategy for performing local search. It includes two phases: 1) the assignment of an initial solution, 2) the refinement of this solution in order to reach a solution that better minimizes the objective function, V. The refinement is a series of moves that the algorithm makes towards a fairer solution. Iteratively, it considers a neighborhood of solutions that it can reach by making a single move from the current solution and greedily selects the move that minimizes V. The algorithm terminates when it cannot find a single move to improve V. The algorithm will reach one local minimum out of many that might exist in such a combinatorial optimization problem. We describe our method in detail at Alg. 2. Additional details about these steps are provided in the subsequent sections.

Algorithm 2 GHC

Input / Output: $\mathbf{R}$ (Recommended course lists for every student), $\Gamma(\mathbf{R})$ (Definition of the neighhorhood).

best_val $\leftarrow V(\mathbf{R})$
obj_val $\leftarrow 0, steps \leftarrow 0$
$\mathcal{V}_t \leftarrow \emptyset, \mathcal{V}_T \leftarrow \emptyset$ ▷ Visited course and student groups while not finding an improved solution.
while $|\mathcal{V}_t| < m$ **do**
    best_val $\leftarrow V(\mathbf{R})$
    Select neighborhood $\Gamma(\mathbf{R})$.
    for $(i, j_{\text{out}}, j_{\text{in}}) \in \Gamma(\mathbf{R})$ **do** ▷ Check every neighboring solution.
        obj_val $\leftarrow$ best_val
        $\mathbf{R}' \leftarrow \mathbf{R}$
        $\mathcal{R}'_i \leftarrow \mathcal{R}_i - \{t\} + \{j\}$
        temp_val $\leftarrow V(\mathbf{R}')$
        if temp_val $\leq$ obj_val **then**
            obj_val $\leftarrow$ temp_val
            $(i', j'_{\text{out}}, j'_{\text{in}}) \leftarrow (i, j_{\text{out}}, j_{\text{in}})$
    if best_val $>$ obj_val **then** ▷ Make a move, update $\mathbf{R}$.
        $\mathcal{R}_{i'} \leftarrow \mathcal{R}_{i'} - \{j'_{\text{out}}\} + \{j'_{\text{in}}\}$
        $\mathcal{V}_t \leftarrow \emptyset, \mathcal{V}_T \leftarrow \emptyset$
    else
        Update $\mathcal{V}_t, \mathcal{V}_T$, as needed. ▷ Add the examined target course/student group in the visited list(s).
return R

Initial solution

We first need to decide the initial solution for our refinement algorithm. A common practice is to start from a good solution and try to improve it. Since we have access to the recommendation scores of a CRS model (assumption 2, Sect. 7.2.1), we can use HSC as the initial assignment. By design, HSC achieves $Q = 0$, which is the global minimum with respect to the Q objective. We start from the HSC assignment and refine it to support the notion of FaiREO fairness.

Moves

A fundamental element of search methods is the type of moves allowed to transition from a given feasible solution to a new one. Given a solution, we remove a course from

the recommendation list of a single student, and replace it with another course. This move can be fully described by a triplet $(i, j_{\text{out}}, j_{\text{in}})$, where j_{out} and j_{in} are the courses we remove from, and introduce to the recommendation list of student i, respectively. We say that a move is *positive* when it results in a solution with improved (i.e., lower) objective function V, and *negative*, otherwise.

Neighborhood of Solutions

We need to specify the neighborhood of solutions that the algorithm will evaluate in order to make a move towards the one that improves V the most. The simplest solution is to examine all possible one-step-away solutions from the existing solution. This is a full-blown search that will consider changing all the student-course pairs $\{(i, j_{\text{out}})\}$, where $i \in \mathcal{S}$, and $j_{\text{out}} \in \mathcal{R}_i$. For each pair of $\{(i, j_{\text{out}})\}$, we will examine all courses that are not currently recommended to the student i as j_{in} (the course that will take the place of j_{out}). We will be referring to the corresponding method as **GHC(NoNe)**, since, essentially, there is no neighborhood specified.

We also use some heuristics for defining a neighborhood in order to avoid searching all the space every time. Rather, we examine a smaller set of moves, hoping that the next best move will belong there. First, we limit a neighborhood of solutions by considering moves altering the recommendations of students in a specific (target) protected group T. In this method, **GHC(G)**, we will search the pairs $\{(i, j_{\text{out}})\}$, where $i \in \mathcal{S}_T$, and $j_{\text{out}} \in \mathcal{R}_i$. We choose the *target protected group* T to be the one that exhibits the highest unfairness in opportunity, i.e., the most severe unbalance in recommendations:

$$T = \underset{p \in \{1,\ldots,g_s\}}{\arg\max} f_p.$$

We can further limit the neighborhood for the local search by additionally selecting a specific (target) course t to replace. The method **GHC(Gc)** will search the pairs $\{(i, j_{\text{out}})\}$, where $i \in \mathcal{S}_T$, and $j_{\text{out}} = t$. The *target course* t is the one that is over-recommended the most among the students of group T, i.e.,

$$t = \underset{j \in \mathcal{C}}{\arg\max} f_{T,j}, \quad \text{where } f_{T,j} = n^j \left(\frac{n_T^j}{n^j} - \frac{n_T}{n} \right).$$

$f_{T,j}$ is the term in the parenthesis in Eq. 7.3 without the absolute value. We select i, j_{out} in such a way assuming that they have the most room for improvement during

refinement.

When the algorithm searches but cannot find a positive solution within the neighborhood we defined, it moves to the next closest neighborhood. There are lists of visited protected groups and courses which we do not consider while selecting the next neighborhood to explore, and they reset every time we find a positive move. For the GHC(Gc) algorithm, we first examine different courses within the target protected group and then explore users in other groups.

In terms of computational complexity, each variation of the algorithm is different. For the case of GHC(NoNe), we need to examine the whole search space every time we make a move, which includes $nk(m-k)$ solutions. To perform one move with GHC(Gc), we need to first find the target course t and protected group T, and then examine all possible moves within the specified neighborhood. In total, the complexity is $n_g+m+n_T^t(m-k)$.

7.3.3 Incremental GHC Algorithm (GHC-Inc)

We also propose another algorithm to optimize the objective function in Eq. 7.7, based on GHC(Gc). In the **GHC-Inc** algorithm, instead of optimizing for the given parameter α, we start optimizing the objective function with a small value of $\alpha' \leftarrow \alpha_0$. Once we reach a local minimum, we increment alpha by a parameter α_{step}, and we further improve the current solution for the updated value of $\alpha' \leftarrow \alpha' + \alpha_{\text{step}}$. We repeat this until we reach the value of given parameter $\alpha' = \alpha$. The intuition behind this process is that we will *gradually increase* the importance of the quality objective, so we hope to reach an assignment with a lower value for the Q objective.

7.3.4 Tabu-based GHC Algorithm (GHC-Tabu)

So far, the discussed algorithms have one common characteristic: they all stop exploring the solution space when there are no positive moves left. This indicates that the algorithm has reached a local minimum, i.e., there is no single move that would improve the current solution. However, in such a huge solution space of this combinatorial optimization problem, this might not be the global minimum. To further explore the search space after this point, we incorporate the idea of Tabu search in algorithm GHC(Gc). Whenever there are no improving moves, the **GHC-Tabu** algorithm performs the move that degrades the objective function the least. We hope that by taking a negative move, we will get into a different neighborhood of solutions that will drive us to a better local

minimum. In order for the algorithm to terminate, we control the *number of negative moves* that we allow it to make.

We also need to ensure that the algorithm will not make the reverse move on the next step, and return to the local minimum already visited. We use the tabu list, a short-term memory list structure with the first-in-first-out property, to store a characterization of every move we made in order not to consider it and reverse it. A parameter controls the *tabu list size*. We store in the tabu list the pair of student-course (i, j_{in}) that we just updated and that we do not allow to take back. However, the tabu list may forbid moves that would lead to a better solution. We introduce an *aspiration criterion* which allows us to make moves forbidden by the tabu list if they lead us to a solution with lower objective function than the lowest objective achieved so far.

7.4 Experimental Setup

7.4.1 Synthetic Datasets

We generated synthetic datasets to evaluate our approaches since we do not have data regarding both students' registration history and their protected attributes. The kind of data that that we need to generate are:

1. the student-course recommendation score matrix $\mathbf{Y} \in \mathbb{R}^{n \times m}$, where $y_{i,j}$ represents the recommendation score of course j for student i estimated by a CRS, and
2. the partitioning of students into protected groups, $\mathcal{S}_1, \ldots, \mathcal{S}_{g_s}$.

The parameters that we need to specify is the number of students n, the number of courses m, and the number of student groups g_s.

We want to create synthetic datasets whose characteristics align with these of real-world datasets. Assume a matrix $\mathbf{Y}$ obtained from a CRS with real data and the corresponding HSC solution when we recommend the highest scored courses for each student. The fairness of this solution depends on the existence of courses whose recommendation scores tend to be higher for a specific group of students. In that case, we will introduce unfairness to our recommendation system because some courses will be good candidates and systematically suggested more times to one particular group than the rest. We model these factors into our dataset generator, by introducing the notion of *course buckets*, which correspond to a partition of the set of courses $\mathcal{C}$. The number of course buckets is controlled by the parameter g_c.

We model the relation between student groups and course buckets via a matrix $\mathbf{M} \in \mathbb{R}^{g_s \times g_c}$, such that $m_{p,q}$ is the average value of the recommendation scores of students in group p for courses in bucket q. We will fill the first row of $\mathbf{M}$ by sampling a normal distribution $N(\mu_{\mathrm{M}}, d_{\mathrm{M}})$ with mean value μ_{M}, and standard deviation d_{M}. To ensure that all students will have initial recommendations of similar quality, we will fill the remaining rows of $\mathbf{M}$ with a permutation of the initial vector, $\mathbf{M}_0$. Once we have generated matrix $\mathbf{M}$, we can finally fill the matrix $\mathbf{Y}$ by sampling the recommendation scores for students in group p and courses in bucket q from a normal distribution $N(\mu_{\mathrm{Y}} = \mu_{p,q}, d_{\mathrm{Y}})$. In order to generate a dataset based on this process, we need to specify the following parameters: number of course buckets g_c, $\mu_{\mathrm{M}}, d_{\mathrm{M}}$ for the initial vector of means $\mathbf{M}_0$, and the standard deviation d_{Y} for the generation of $\mathbf{Y}$.

This dataset generator is parameterized in order to create datasets of different difficulty levels. This allows us to evaluate how our algorithm will operate under different settings. Given this framework, the difficulty of a dataset is controlled by how close to each other are the mean values in $\mathbf{M}_0$, i.e., the standard deviation d_{M}. The further away the means are, the further away the scores of different protected groups for a course bucket will be (and the less likely it will be to recommend courses from this bucket to all student groups).

We set the parameters as follows: $n = 600, m = 60, g_s = \{2, 4\}, g_c = 4, d_{\mathrm{Y}} = 0.3$. We generated datasets of three difficulty levels: easy (`Uni`), medium (`Gauss(1,0.1)`), and hard (`Gauss(1,0.3)`). For the datasets `Gauss(1,0.1)` and `Gauss(1,0.3)` we set $\mu_{\mathrm{M}} = 1.0$ and the parameter $d_{\mathrm{M}} = \{0.1, 0.3\}$, respectively. The easiest dataset is `Uni`, where all recommendation scores $y_{i,j}$ are sampled from a uniform distribution in $[0, 1]$, and there are not courses with high scores by design. In total, we create six families of datasets; for three difficulty levels, and for two or four protected groups. For every type of dataset, we create five versions of them, by using different seeds to generate the matrices $\mathbf{M}$ and $\mathbf{Y}$. In this way, we get to examine how sensitive are our models to input data with similar characteristics. We show the variation in performance of the different versions with the standard error bars.

7.4.2 Real Datasets

We collected data from the Computer Science and Engineering department of a large public University. The data include the grades of undergraduate students and span a period of 10 years, until the fall semester of 2015. We only considered full-time students

that actually graduated with a bachelor's degree. We used the last three semesters as the test set, and we trained a random-walk based course recommendation system [76] using the rest of the data. We keep the recommendation scores of the students in the test set, and use them as the matrix $\mathbf{Y}$. We set $k = 5$ in our experiments. We treat every semester as a different dataset and we end up with the following datasets: fall 2014, spring 2015 and fall 2015, with $188, 170, 112$ students and $59, 54, 55$ courses, respectively. We will be referring to these datasets as the `CompSci` datasets.

The available data did not include any protected attributes, so we used other student-related information (entry registration status and the number of credits transferred) to simulate the socioeconomic status of the students. This led to three protected groups: high school students with less than 15 credits transferred (HS), high school students with more than 15 credits transferred (HSAP), and those coming from other institutions/colleges (NAS). High school students can take Advanced Placement (AP) courses and transfer the credits earned to their undergraduate program. Minorities and low-income students are underrepresented in AP classes, and a low percentage of them actually take and pass the AP exams every year [96, 97]. Regarding NAS students, we do not have information about the institution where they transferred from. However, reports statistically show that almost half of NAS students come from 2-year colleges [82] which are considered a major access point to 4-year institutions for minority and low-income students [21]. The fraction of students belonging in the [HS, HSAP, NAS] protected group is $[0.19, 0.76, 0.06]$ for the fall 2014, $[0.18, 0.76, 0.05]$ for the spring 2015, and $[0.21, 0.73, 0.06]$ for the fall 2015.

7.4.3 Model Parameters

Regarding the norm base, x, we explore the use of $L2$ norm and $L\infty$ norm. The $L2$-norm penalizes elements with high values, and promotes all entries having small values, which is exactly what we are trying to achieve. We also consider $L\infty$ norm what considers only the $\mathbf{f}$ and $\mathbf{q}$ vectors' highest elements. The $L\infty$ norm is a stricter norm, that will force the student group with the highest objective values to get as low as possible. This will limit the worst case scenario for the protected groups.

The models have two additional parameters that we need to specify: the number of courses to recommend k, and the value of α that controls the trade off between opportunity and quality loss objectives. We set $k = 5$, and $\alpha = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. For the algorithm GHC-Inc, we set $a_0 = 0.1, a_{\text{step}} = 0.1$. For the GHC-Tabu, we set the

number of negative moves to 150 and the tabu list size to 50.

7.5 Experimental Results

7.5.1 Neighborhood for Local Search

The first aspect of the proposed methods we need to evaluate the breadth of neighborhood for the local search as presented in Sect. 7.3.2. The methods GHC(Gc), GHC(G), and GHC(NoNe) have increasing larger neighborhoods to search in order to make one move. We select the target student group T and course t with the highest f_p and $f_{T,j}$, respectively. Figs. 7.1, 7.2, 7.3 present these results for the synthetic datasets and $L\infty$ norm.

These figures include the method GHC(G-V), a version of GHC(G) where the target student group is selected to be the one that contributes the most at the combined objective function V. We cannot evaluate the corresponding GHC(Gc-V), as we cannot compute the contribution of a student group and single course to the overall objective function (and more specifically, the quality objective). We can easily tell from the Fig. 7.1 that GHC(G-V) perform worse or about the same than the rest of the approaches. In Fig. 7.2, we can see that GHC(G-V) introduces higher Q objective when α increases. It does better for 2 student groups (first row) and the hardest dataset (`Gauss(1,0.3)`), but it is still worse than GHC(NoNe). As a result, we see that considering the overall objective, instead of only the opportunity objective, to select the target protected group does not offer any performance improvements.

For the rest of the methods, we see that GHC(NoNe) does perform the best for the datasets with 2 protected groups. In this case, the algorithm manages to significantly improve the equality of opportunity in the recommendations, even for small values of α, without introducing additional quality loss. On the other hand, GHC(Gc) and GHC(G) also have very similar performance that is worse than GHC(NoNe), especially for higher values of α. From Fig. 7.2, we can see that all of these methods manage to achieve similar opportunity objective. The difference is on the quality loss that they introduce when α is closer to 1.

The performance of the models is different for the datasets with 4 student groups (second row in Figs.7.1, 7.2, 7.3) with $L\infty$ norm. Here, GHC(Gc) performs the best in terms of the combined objective function and the opportunity objective F. Even though the method GHC(NoNe) is free to search the whole space every time before

making a move, it cannot improve the opportunity objective as effectively as the other methods. When we select the target student group T and course t, we force the models to make a move that will likely improve the corresponding opportunity objective. We force the GHC(Gc) algorithm to stay within a specific path towards the final solution, that manages to minimize the opportunity objective. On the other hand, GHC(NoNe) does not have this direction, and gets stuck at a local optimum. We can observe that in the percentage of recommendations that were affected by the algorithms (Fig. 7.3). In the second row that corresponds to 4 protected groups, we see that GHC(NoNe) makes fewer changes and stops as it cannot find other moves that further improve the current combined objective value.

We performed similar analysis for the $L2$ norm. GHC(NoNe) outperformed the rest of models, whose performance was similar, for both 2 and 4 protected groups. For the rest of the experimental result section, we will only present results for GHC(NoNe) and GHC(Gc), as these are the two methods with the best performance achieved for the datasets tested and $L2$ and $L\infty$ norm. Note that, even for cases where GHC(Gc) performs worse than GHC(NoNe), it does manage to reach the same fairness levels and it does so 10 times faster than GHC(NoNe).

7.5.2 Comparison of Methods Developed

In this section, we compare the GHC(Gc) and GHC(NoNe) with GHC-Inc and GHC-Tabu with Figs. 7.4, 7.5, 7.6, 7.7, 7.8, and 7.9. These figures come in pairs, showing:

- the overall objective function V w.r.t. α,
- the Pareto curve showing a scatter plot between the quality objective Q and the opportunity objective V.

These figures show the results for:

- the synthetic datasets, with $L2$ norm,
- the synthetic datasets, with $L\infty$ norm,
- the `CompSci` datasets, with $L2$ and $L\infty$ norms.

The first thing that we notice is that GHC-Tabu has very close to the performance of the GHC(Gc) method for all the cases and especially for $L2$ norm and for the `CompSci`

datasets. When there is a noticeable difference, it is for values of α around of larger than 0.5. In these cases, GHC-Tabu manages to make some further moves after GHC(Gc) to improve the quality objective. However, it does not make many additional moves, so it looks similar to GHC(Gc). In general, either the GHC(Gc)/GHC-Tabu or the GHC(NoNe) are the algorithms that still perform the best across all the datasets and norms we tested.

The GHC-Inc is one of the worst-performing algorithms. For the `CompSci` datasets, it performs very poorly, and that is the reason why it is not included in Figs. 7.6, 7.7. For the synthetic datasets and $L2$ norm, the objective function V is high because of the quality objective Q. Even though it manages to improve fairness, it sacrifices the quality of the recommendations considerably. However, for the $L\infty$ norm and the the exception of the `Uni` dataset, GHC-Inc manages to be the second best performing method and it may be better than GHC(NoNe) and GHC-Tabu.

GHC-Inc takes more time for higher values of α, as it needs to build all the models with increasing α. However, even in the worst case, it is still faster than GHC(NoNe). Overall, we could potentially replace GHC(NoNe) with GHC-Tabu or GHC-Inc as needed in order to save some computational time.

Regarding the `CompSci` datasets, the performance of the GHC(Gc), GHC(Tabu) and GHC(NoNe) is very similar. However, we can notice in Fig. 7.9 that for small values of α (e.g., $\alpha = 0.1$) GHC-Tabu achieves lower opportunity objective F. Even with such small weight on fairness, GHC-Tabu considerably improves the opportunity objective without introducing a high quality objective. For example, consider the term of Spring '15 for $L2$. The initial value of the opportunity objective is 0.1719. By assigning $\alpha = 0.1$, we get 0.1278 and 0.1318 opportunity objective for GHC-Tabu and GHC(NoNe), respectively. When $\alpha = 0.2$, these numbers correspond to 0.0937 and 0.0978, and the quality objective is still below 0.0075. That was an example where using GHC-Tabu, we can almost improve the opportunity objective by 50% while sacrificing the quality of the recommendations just a little.

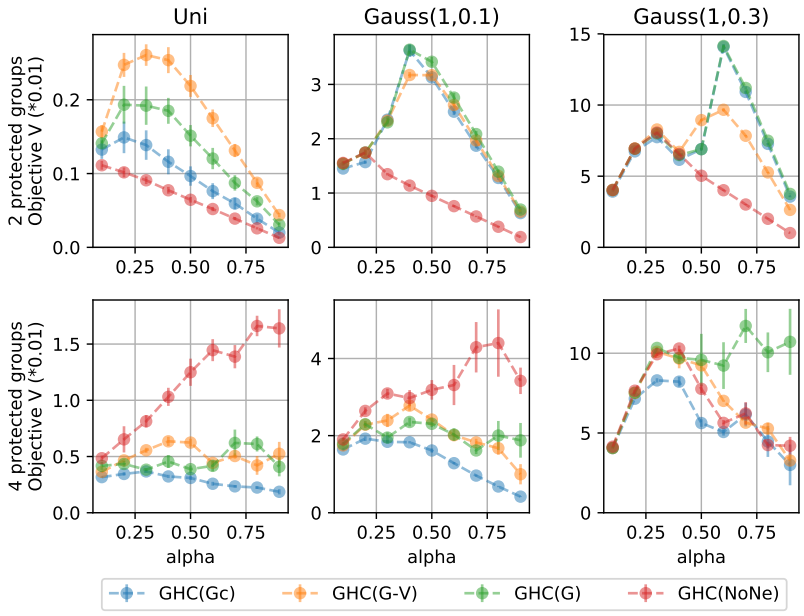


Figure 7.1: Objective function V, for synthetic datasets with $L\infty$ norm.

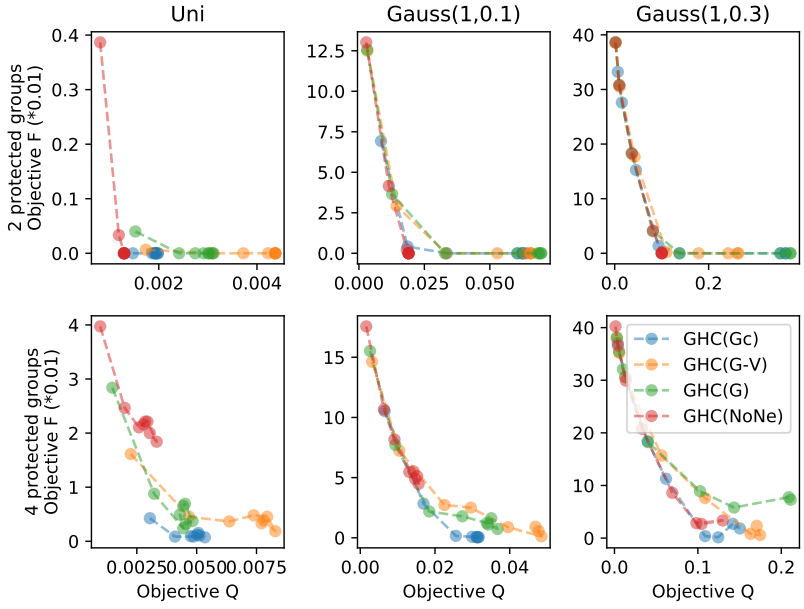


Figure 7.2: Objective functions Q vs F, for synthetic datasets with $L\infty$ norm.

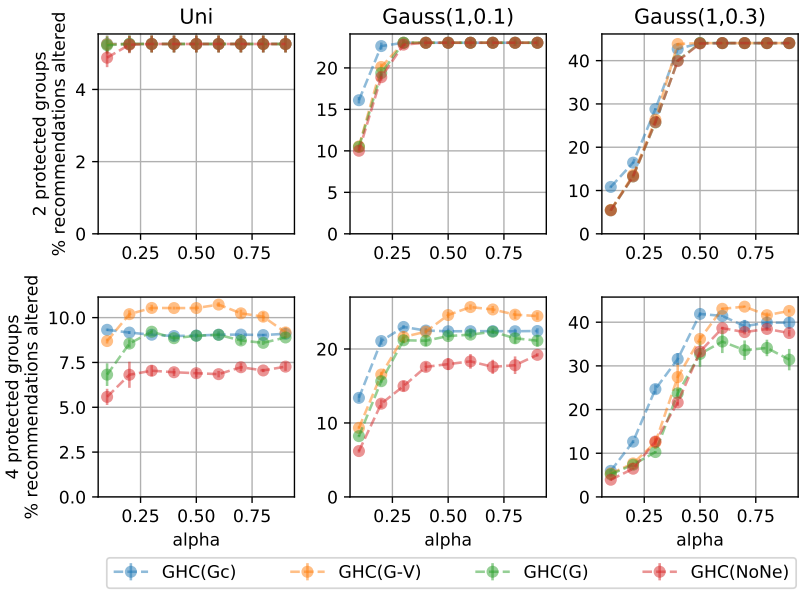


Figure 7.3: Percentage of recommendations that were affected by our algorithms with $L\infty$ norm for the synthetic datasets.

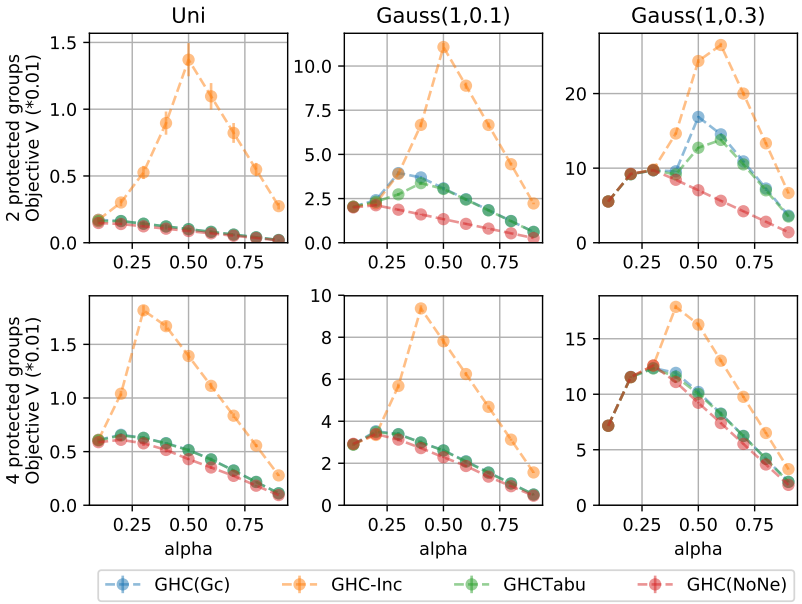


Figure 7.4: Objective function V, for synthetic datasets with $L2$ norm.

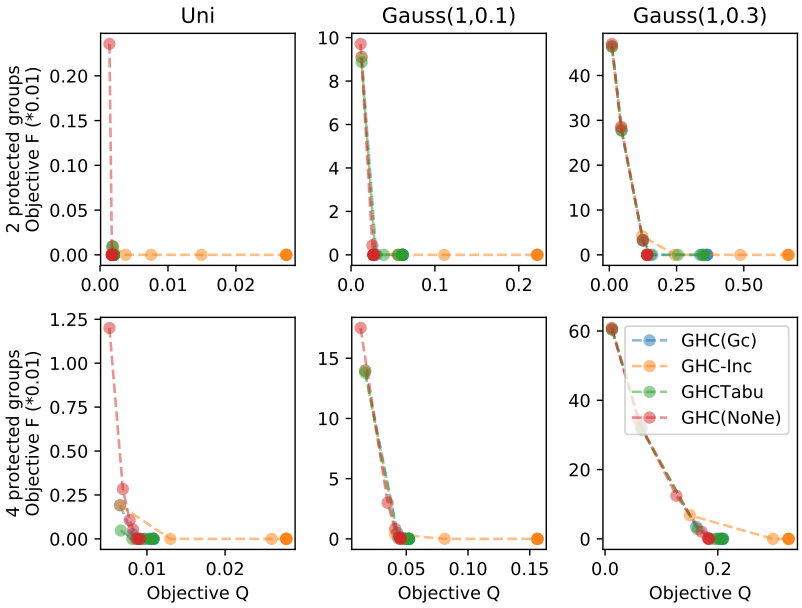


Figure 7.5: Objective functions Q vs F, for synthetic datasets with $L2$ norm.

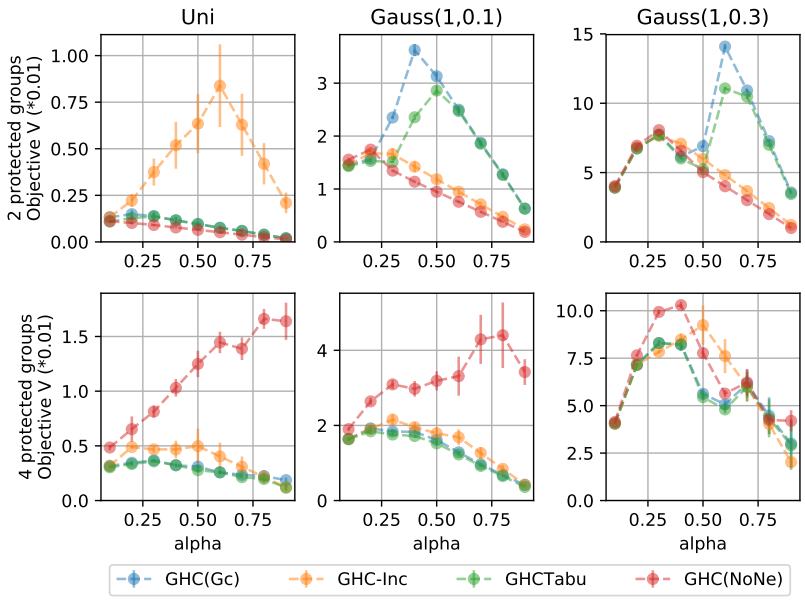


Figure 7.6: Objective function V, for synthetic datasets with $L\infty$ norm.

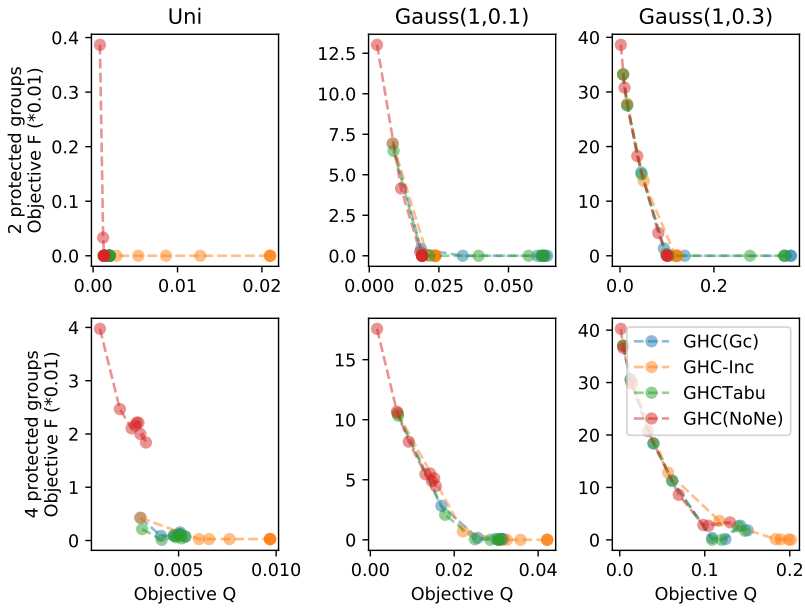


Figure 7.7: Objective functions Q vs F, for synthetic datasets with $L\infty$ norm.

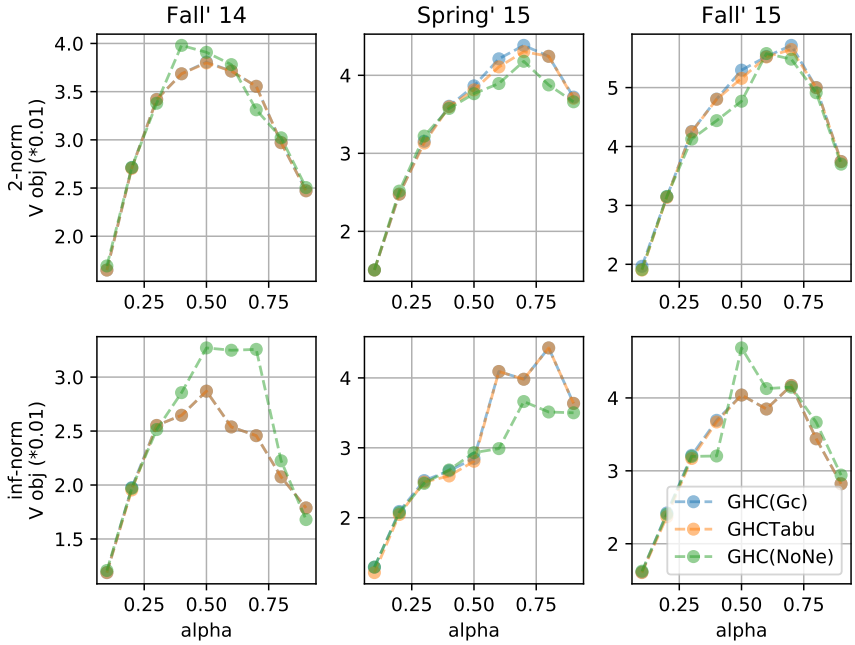


Figure 7.8: Objective function V for `CompSci` datasets.

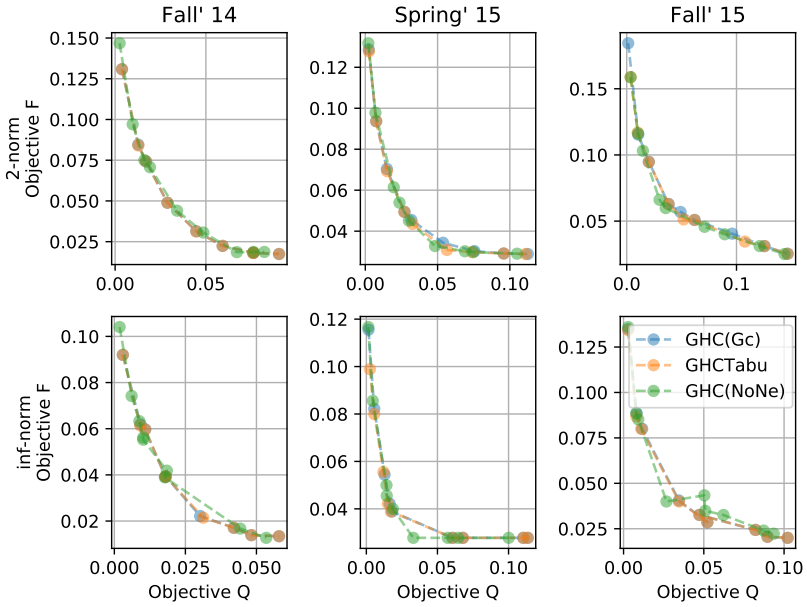


Figure 7.9: Objective functions Q vs F for `CompSci` datasets.

Chapter 8

Summary and Future Work

In this dissertation we presented novel techniques towards predicting and understanding student performance, as well as, supporting course selection process in higher education. Motivated and inspired by the needs of users in undergraduate institutions, we developed machine learning models and algorithms that account for the characteristics and restrictions of this particular domain. We validated our models using data collected over more than 12 years from the University of Minnesota. Our work sets the basis for future work on these problems.

The following subsections summarize the contributions introduced in this work, and discuss future research avenues to explore.

8.1 Thesis Summary

We presented three course-specific approaches based on linear regression and matrix factorization that perform better than existing approaches based on traditional methods, assuming that the degree programs involved have a vertical structure. In that case, focusing on a course specific subset of the data can result in more accurate predictions. Moreover, the performance for different departments can significantly vary, as they may have different characteristics and structures. Overall, our approaches can improve the performance of grade prediction over other methods tested for our dataset, while the degree of improvement depends on the department.

We also focused on accurately identifying students that are at risk before they even take a class. These students might fail the class, drop it, or perform worse than they usually do. We can notify the student's instructor, advisor and department to take the

appropriate steps to assist that particular student before the student registers for a course for which he/she seems to fail or drop later, after spending valuable time and effort. We successfully extracted features from historical grading data, in order to test different simple and sophisticated classification methods. By performing our feature importance study and creating meaningful feature groups, we also got interesting findings that can explain student performance in a concise way. When combining the predictions from different models, we can further improve the performance achieved by a single model, since different feature groups capture different characteristics.

For the problem of course selection in particular, we propose Scholars Walk, a novel method designed to harvest the sequential patterns arising from past course enrollment data in order to recommend a short list of personalized course suggestions for the next semester. The proposed method relies on a random walk-based scheme on a course-to-course graph and personalization is achieved by a student-adapted starting distribution reflecting the current student's past enrollment. When compared with five competing models, from popularity-based to LSTMs and Basic Markov models, Scholars Walk achieves the best performance. It manages to be a successful, scalable approach that provides personalized recommendations for every student.

Course selection not only plays an important role in students' progress towards graduation, but also in the career path they will follow afterwards. To ensure that all students are given the same opportunities when using a recommender system, we examined group fairness in the context of course recommendation and introduces the definition of fairness FaiREO. A multi-objective problem balances the fairness in opportunity and quality objectives. We developed greedy algorithms that iteratively improve the combined objective function. The results indicate that, even by assigning little importance to fairness, we can considerably improve the recommender system to balance its recommendations across the protected groups.

8.2 Future Research Directions

In this work, we provided data mining-based solutions for the problems of performance prediction and course recommendation. Here we outline some future research directions that stem from our work.

- Our Scholar's Walk algorithm, presented in Ch. 6, utilizes the course registration only from one past semester to generate recommendations for a student. That

possibly limits the personalization achieved by the model. We need to find an effective way to include more historical data as input to the recommendation system at the time of testing.

- The GHC approaches that we presented in Ch. 7 might get stuck at a local minimum. To overcome this, we proposed GHC-Tabu, which slightly improved the overall objective. We can try a different family of algorithms to solve this problem similar to [36] that work on phases and allow negative steps that may temporarily decrease the objective function hoping to reach better local minima.

- In the context of course recommendation, we can study how the personalization relates to fairness.

- In Ch. 7, we proposed Fairness in Recommendation for Equality of Opportunity, and designed algorithms that improve this notion on a recommender's output. We can further explore more sources of unfairness in problems related to higher education, such as grade prediction.

- A significant problem of the research conducted in education data mining is data sharing among institutions. As the students enrolled in any university at any point of time is very specific and limited, the research community would benefit from accessing data from other institutions to build more accurate and robust models. However, that cannot happen because of privacy concerns. We need to find a way to effectively transfer learning from one dataset and institution to another, without violating data privacy.

References

[1] H. Abdollahpouri, R. Burke, and B. Mobasher. Managing popularity bias in recommender systems with personalized re-ranking. In *The Thirty-Second International Flairs Conference*, 2019.

[2] A. Al-Badarenah and J. Alsakran. An automated recommender system for course selection. *Intl. Journal of Advanced Computer Science and Applications*, 7(3):1166–1175, 2016.

[3] M. A. Al-Barrak and M. Al-Razgan. Predicting students final gpa using decision trees: A case study. *International Journal of Information and Education Technology*, 6(7):528, 2016.

[4] K. E. Arnold and M. D. Pistilli. Course signals at purdue: using learning analytics to increase student success. In *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge*, pages 267–270. ACM, 2012.

[5] E. Babad and A. Tayeb. Experimental analysis of students' course selection. *British Journal of Educational Psychology*, 73(3):373–393, 2003.

[6] J. E. Beck and B. P. Woolf. High-level student modeling with machine learning. In *International Conference on Intelligent Tutoring Systems*, pages 584–593. Springer, 2000.

[7] M. Berliant, W. Thomson, and K. Dunz. On the fair division of a heterogeneous commodity. *Journal of Mathematical Economics*, 21(3):201–216, 1992.

[8] A. Beutel, J. Chen, T. Doshi, H. Qian, L. Wei, Y. Wu, L. Heldt, Z. Zhao, L. Hong, E. H. Chi, et al. Fairness in recommendation ranking through pairwise comparisons. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2212–2220, 2019.

[9] A. J. Biega, K. P. Gummadi, and G. Weikum. Equity of attention: Amortizing individual fairness in rankings. In *The 41st international acm sigir conference on research & development in information retrieval*, pages 405–414, 2018.

[10] A. Blum, J. Hopcroft, and R. Kannan. Random walks and markov chains. In *Foundations of data science*. Vorabversion eines Lehrbuchs, 2016.

[11] S. Bouveret, Y. Chevaleyre, and N. Maudet. Fair allocation of indivisible goods., 2016.

[12] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[13] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[14] E. Budish. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. *Journal of Political Economy*, 119(6):1061–1103, 2011.

[15] R. D. Burke, H. Abdollahpouri, B. Mobasher, and T. Gupta. Towards multi-stakeholder utility evaluation of recommender systems. In *UMAP (Extended Proceedings)*, 2016.

[16] H. Bydžovská. Are collaborative filtering methods suitable for student performance prediction? In *Progress in Artificial Intelligence*, pages 425–430. Springer, 2015.

[17] H. Cen, K. Koedinger, and B. Junker. Learning factors analysis–a general method for cognitive model evaluation and improvement. In *Intelligent tutoring systems*, pages 164–175. Springer, 2006.

[18] Y. Chen, S. Bhojanapalli, S. Sanghavi, and R. Ward. Coherent matrix completion. *arXiv preprint arXiv:1306.2979*, 2013.

[19] F. Christoffel, B. Paudel, C. Newell, and A. Bernstein. Blockbusters and wallflowers: Accurate, diverse, and scalable recommendations with random walks. In *9th ACM Conf. on Recommender Systems*, pages 163–170, New York, NY, USA, 2015. ACM.

[20] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[21] G. Crisp and A.-M. Nuñez. Understanding the racial transfer gap: Modeling underrepresented minority and nonminority students' pathways from two-to four-year institutions. *Review of Higher Education*, 37(3):291–320, 2014.

[22] L. DeAngelo, R. Franke, S. Hurtado, J. H. Pryor, and S. Tran. Completing college: Assessing graduation rates at four-year institutions. *Los Angeles: Higher Education Research Institute, UCLA*, 2011.

[23] T. Denley. Course recommendation system and method. `http://www.google.com/patents/US20130011821`.

[24] T. Denley. Austin peay state university: Degree compass. *EDUCAUSE Review Online. Available: http://www.educause.edu/ero/article/austin-peay-state-university-degree-compass*, 2012.

[25] M. Deshpande and G. Karypis. Selective markov models for predicting web page accesses. *ACM Trans. on Internet technology (TOIT)*, 4(2):163–184, 2004.

[26] A. Diamond, J. Roberts, T. Vorley, G. Birkin, J. Evans, J. Sheen, and T. Nathwani. Uk review of the provision of information about higher education: advisory study and literature review: report to the uk higher education funding bodies by cfe research. *Higher Education Funding Council for England*, 2014.

[27] F. Diebold, H. Aziz, M. Bichler, F. Matthes, and A. Schneider. Course allocation via stable matching. *Business & Information Systems Engineering*, 6(2):97–110, 2014.

[28] F. Diebold and M. Bichler. Matching with indifferences: A comparison of algorithms in the context of course allocation. *European Journal of Operational Research*, 260(1):268–282, 2017.

[29] C. Eksombatchai, P. Jindal, J. Z. Liu, Y. Liu, R. Sharma, C. Sugnet, M. Ulrich, and J. Leskovec. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *World Wide Web Conf.*, pages 1775–1784, 2018.

[30] A. Elbadrawy and G. Karypis. Domain-aware grade prediction and top-n course recommendation. In *10th ACM Conf. on RecSys*, pages 183–190, 2016.

[31] A. Elbadrawy, A. Polyzou, Z. Ren, M. Sweeney, G. Karypis, and H. Rangwala. Predicting student performance using personalized analytics. *Computer*, 49(4):61–69, 2016.

[32] A. Elbadrawy, R. S. Studham, and G. Karypis. Collaborative multi-regression models for predicting students' performance in course activities. In *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge*, pages 103–107. ACM, 2015.

[33] A. Esteban, A. Z. Gómez, and C. Romero. A hybrid multi-criteria approach using a genetic algorithm for recommending courses to university students. In *11th Intl. Conf. on Educational Data Mining*, 2018.

[34] G. Farnadi, P. Kouki, S. K. Thompson, S. Srinivasan, and L. Getoor. A fairness-aware hybrid recommender system. *arXiv preprint arXiv:1809.09030*, 2018.

[35] M. Feng, N. T. Heffernan, and K. R. Koedinger. Looking for sources of error in predicting student?s knowledge. In *Educational Data Mining: Papers from the 2005 AAAI Workshop*, pages 54–61, 2005.

[36] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Design Automation Conference*, pages 175–181. IEEE, 1982.

[37] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[38] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Ann. of statistics*, pages 1189–1232, 2001.

[39] J. P. Gardner, C. Brooks, and W. Li. Learn from your (markov) neighbor: Coenrollment, assortativity, and grade prediction in undergraduate courses. *Journal of Learning Analytics*, 5(3):42–59, 2018.

[40] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh. Wtf: The who to follow service at twitter. In *22Nd Intl. Conf. on World Wide Web*, New York, NY, USA, 2013. ACM.

[41] Q. Hu, A. Polyzou, G. Karypis, and H. Rangwala. Enriching course-specific regression models with content features for grade prediction. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 504–513. IEEE, 2017.

[42] Q. Hu and H. Rangwala. Course-specific markovian models for grade prediction. In *Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, pages 29–41. Springer, 2018.

[43] F. Husseinov. A theory of a heterogeneous divisible commodity exchange economy. *Journal of Mathematical Economics*, 47(1):54–59, 2011.

[44] C.-S. Hwang and Y.-C. Su. Unified clustering locality preserving matrix factorization for student performance prediction. *IAENG International Journal of Computer Science*, 42(3), 2015.

[45] A. Kadlec, J. Immerwahr, and J. Gupta. Guided pathways to student success perspectives from indiana college students and advisors. *New York: Public Agenda*, 2014.

[46] P. B. Kantor, L. Rokach, F. Ricci, and B. Shapira. *Recommender systems handbook*. Springer, 2011.

[47] E. S. Khorasani, Z. Zhenge, and J. Champaign. A markov chain collaborative filtering model for course enrollment recommendations. In *Big Data (Big Data), IEEE Intl. Conf. on*, pages 3484–3490. IEEE, 2016.

[48] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

[49] K. Lin, N. Sonboli, B. Mobasher, and R. Burke. Crank up the volume: preference bias amplification in collaborative recommendation. *arXiv preprint arXiv:1909.06362*, 2019.

[50] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. on knowledge and data engineering*, 17(4):491–502, 2005.

[51] J. Luo, E. Sorour, K. Goda, and T. Mine. Predicting student grade based on free-style comments using word2vec and ann by considering prediction results obtained in consecutive lessons. *International Educational Data Mining Society*, pages 396–399, 2015.

[52] E. C. Malthouse, K. A. Vakeel, Y. K. Hessary, R. Burke, and M. Fudurić. A multistakeholder recommender systems algorithm for allocating sponsored recommendations. In *Workshop on Recommendation in Multi-stakeholder Environments (RMSE'19), in conjunction with the 13th ACM Conference on Recommender Systems, RecSys*, volume 19, page 2019, 2019.

[53] N. Manouselis, H. Drachsler, R. Vuorikari, H. Hummel, and R. Koper. Recommender systems in technology enhanced learning. In *Recommender systems handbook*, pages 387–415. Springer, 2011.

[54] P. Manurangsi and W. Suksompong. Asymptotic existence of fair divisions for groups. *Mathematical Social Sciences*, 89:100–108, 2017.

[55] J. McFarland, B. Hussar, C. de Brey, T. Snyder, X. Wang, S. Wilkinson-Flicker, S. Gebrekristos, J. Zhang, A. Rathbun, A. Barmer, et al. Undergraduate retention and graduation rates. In *The Condition of Education 2017. NCES 2017-144.* ERIC, 2017.

[56] T. McKay, K. Miller, and J. Tritz. What to do with actionable intelligence: E 2 coach as an intervention engine. In *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge*, pages 88–91. ACM, 2012.

[57] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635*, 2019.

[58] R. Mehrotra, J. McInerney, H. Bouchard, M. Lalmas, and F. Diaz. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. In *Proceedings of the 27th acm international conference on information and knowledge management*, pages 2243–2251, 2018.

[59] B. Minaei-Bidgoli and W. F. Punch. Using genetic algorithms for data mining optimization in an educational web-based system. In *Genetic and evolutionary computation conference*, pages 2252–2263. Springer, 2003.

[60] S. Morsy and G. Karypis. Learning course sequencing for course recommendation, 2018.

[61] H.-Q. Nguyen, T.-T. Pham, V. Vo, B. Vo, and T.-T. Quan. The predictive modeling for learning student results based on sequential rules. *Intl. Journal of Innovative Computing, Information and Control (IJICIC)*, 14(6):2129–2140, 2018.

[62] A. N. Nikolakopoulos and G. Karypis. Recwalk: Nearly uncoupled random walks for top-n recommendation. In *12th ACM Intl. Conf. on Web Search and Data Mining*, pages 150–158. ACM, 2019.

[63] A. Ogunde and D. Ajibade. A data mining system for predicting university students? graduation grades using id3 decision tree algorithm. *Journal of Computer Science and Information Technology*, 2(1):21–46, 2014.

[64] F. Okubo, T. Yamashita, A. Shimada, and H. Ogata. A neural network approach for students' performance prediction. In *Seventh Intl. Learning Analytics & Knowledge Conf.*, pages 598–599. ACM, 2017.

[65] E. Osmanbegović and M. Suljić. Data mining approach for predicting student performance. *Economic Review*, 10(1), 2012.

[66] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[67] A. Parameswaran, P. Venetis, and H. Garcia-Molina. Recommendation systems with complex constraints: A course recommendation perspective. *ACM Trans. on Information Systems (TOIS)*, 29(4):20, 2011.

[68] Z. A. Pardos, Z. Fan, and W. Jiang. Connectionist recommendation in the wild: on the utility and scrutability of neural networks for personalized course guidance. *User Modeling and User-Adapted Interaction*, pages 1–39, 2019.

[69] Z. A. Pardos and N. T. Heffernan. Using hmms and bagged decision trees to leverage rich features of user and skill from an intelligent tutoring system dataset. *Journal of Machine Learning Research W & CP*, 2010.

[70] 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, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[71] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[72] A. Polyzou and G. Karypis. Grade prediction with course and student specific models. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 89–101. Springer, 2016.

[73] A. Polyzou and G. Karypis. Grade prediction with models specific to students and courses. *International Journal of Data Science and Analytics*, 2(3-4):159–171, 2016.

[74] A. Polyzou and G. Karypis. Feature extraction for classifying students based on their academic performance. In *Proceedings of the 11th International Conference on Educational Data Mining (EDM)*, pages 356–362. ERIC, 2018.

[75] A. Polyzou and G. Karypis. Feature extraction for next-term prediction of poor student performance. *IEEE Transactions on Learning Technologies*, 12(2):237–248, 2019.

[76] A. Polyzou, A. N. Nikolakopoulos, and G. Karypis. Scholars walk: A markov chain framework for course recommendation. In *Proceedings of the 12th International Conference on Educational Data Mining (EDM)*, pages 396–401. ERIC, 2019.

[77] S. Ray and A. Sharma. A collaborative filtering based approach for recommending elective courses. In *International Conference on Information Intelligence, Systems, Technology and Management*, pages 330–339. Springer, 2011.

[78] M. Robnik-Šikonja and I. Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine learning*, 53(1-2):23–69, 2003.

[79] C. Romero, M.-I. López, J.-M. Luna, and S. Ventura. Predicting students' final performance from participation in on-line discussion forums. *Computers & Education*, 68:458–472, 2013.

[80] C. Romero, S. Ventura, P. G. Espejo, and C. Hervás. Data mining algorithms to classify students. In *Educational Data Mining 2008*, 2008.

[81] C. Romero, S. Ventura, and E. García. Data mining in course management systems: Moodle case study and tutorial. *Computers & Education*, 51(1):368–384, 2008.

[82] D. Shapiro, A. Dundar, F. Huie, P. K. Wakhungu, X. Yuan, A. Nathan, and Y. Hwang. Tracking transfer. *NSC Research Center*, 2019.

[83] L. Shields, A. Newman, and D. Satz. Equality of educational opportunity. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, 2017.

[84] A. Singh and T. Joachims. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2219–2228, 2018.

[85] S. E. Sorour, T. Mine, K. Goda, and S. Hirokawa. A predictive model to evaluate student performance. *Journal of Information Processing*, 23(2):192–201, 2015.

[86] Starfish: Earlyalert. `http://www.starfishsolutions.com/home/student-success-solutions/`.

[87] W. Suksompong. Approximate maximin shares for groups of agents. *Mathematical Social Sciences*, 92:40–47, 2018.

[88] M. Sweeney, J. Lester, and H. Rangwala. Next-term student grade prediction. In *Big Data (Big Data), 2015 IEEE International Conference on*, pages 970–975. IEEE, 2015.

[89] M. Sweeney, H. Rangwala, J. Lester, and A. Johri. Next-term student performance prediction: A recommender systems approach. *arXiv preprint arXiv:1604.01840*, 2016.

[90] J. Tang, S. Alelyani, and H. Liu. Feature selection for classification: A review. In C. C. Aggarwal, editor, *Data classification: algorithms and applications*, chapter 2, pages 37–64. CRC Press, 2014.

[91] N. Thai-Nghe, L. Drumond, T. Horváth, and L. Schmidt-Thieme. Using factorization machines for student modeling. In *UMAP Workshops*, 2012.

[92] N. Thai-Nghe, L. Drumond, A. Krohn-Grimberghe, and L. Schmidt-Thieme. Recommender system for predicting student performance. *Procedia Computer Science*, 1(2):2811–2819, 2010.

[93] N. Thai-Nghe, T. Horváth, and L. Schmidt-Thieme. Factorization models for forecasting student performance. In *Educational Data Mining 2011*, 2010.

[94] A. Toscher and M. Jahrer. Collaborative filtering applied to educational data mining. *KDD cup*, 2010.

[95] V. Tsintzou, E. Pitoura, and P. Tsaparas. Bias disparity in recommendation systems. *arXiv preprint arXiv:1811.01461*, 2018.

[96] A. Tugend. Who benefits from the expansion of ap classes. *The New York Times Magazine*, 7, 2017.

[97] G. W. Whiting and D. Y. Ford. Multicultural issues: Black students and advanced placement classes: Summary, concerns, and recommendations. *Gifted Child Today*, 32(1):23–26, 2009.

[98] C. Wong. Sequence based course recommender for personalized curriculum planning. In *Intl. Conf. on Artificial Intelligence in Education*, 2018.

[99] K. Yang and J. Stoyanovich. Measuring fairness in ranked outputs. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, pages 1–6, 2017.

[100] M. Zehlike, F. Bonchi, C. Castillo, S. Hajian, M. Megahed, and R. Baeza-Yates. Fa* ir: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1569–1578, 2017.