INTELLIGENT DEGREE PLANNING SYSTEMS: 
ADVANCEMENTS IN PERSONALIZED ACHIEVABILITY RATING

by

Mack Sweeney
A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Master of Science
Computer Science

Committee:

_____________________________ Dr. Huzefa Rangwala, Thesis Director

_____________________________ Dr. Carlotta Domeniconi, Committee Member

_____________________________ Dr. Kathryn Blackmond Laskey, Committee Member

_____________________________ Dr. Sanjeev Satia, Chairman, Department of Computer Science

_____________________________ Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering

Date: __________________________ Fall Semester 2016
George Mason University
Fairfax, VA
Intelligent Degree Planning Systems: Advancements in Personalized Achievability Rating

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

By

Mack Sweeney
Bachelor of Science
George Mason University, 2014

Director: Dr. Huzefa Rangwala, Professor
Department of Computer Science

Fall Semester 2016
George Mason University
Fairfax, VA
Dedication

I dedicate this thesis to my wife, Katie. Her support and encouragement have been fundamental to my success in my research. She has always been there to listen when I’ve needed to ramble on about ideas or frustrations. I also dedicate this thesis to my friends and family, who have provided support and feedback which have surely improved the quality of this document and the quality of my life while producing it.
Acknowledgments

I extend my sincere thanks to my thesis director, Dr. Rangwala. He has taught me much about the world of academic research. Under his tutelage, I have learned many valuable lessons which have shaped my understanding of computer science and sharpened my abilities as a researcher.

I would like to thank Drs. Kathryn Laskey and Carlotta Domeniconi for their valuable time in reviewing and providing feedback. Both have provided guidance which has helped me improve my thesis, advance my studies, and grow as a researcher and computer scientist. I would like to thank Dr. Laskey for exposing me to Bayesian methods and inference techniques. Her teaching and support were the foundation for an entire chapter of this thesis and a continuing interest in Bayesian methods and decision theory.

I would also like to thank Ryan Lucas for being my one true source of information regarding deadlines and requirements for thesis submission. Without his help, I suspect I would have spent a great deal more on unnecessary forms and registrations.

Finally, I would like to thank Dr. Zoran Duric for helping me understand the Masters thesis requirements. His advice and guidance kept me on track and likely saved me a great deal of time and frustration. For that, I am very grateful.

The data for this study was made available through a collaborative effort spearheaded by Office of Institutional Research and Reporting, and we would like to acknowledge Kris Smith, Kathryn Zora, Angela Detlev, Eve Dauer, Kathy Zimmerman, Joe Balducci, and Judy Lou at GMU.

This research was partially funded by NSF IIS grant 1447489.
## Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>vii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>viii</td>
</tr>
<tr>
<td>Abstract</td>
<td>ix</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 The Magnitude of the Graduation Rate Problem and the Importance of Advising</td>
<td>2</td>
</tr>
<tr>
<td>1.2 An Intelligent Degree Planning System</td>
<td>6</td>
</tr>
<tr>
<td>1.3 Performance Prediction for Personalized Achievability Rating</td>
<td>8</td>
</tr>
<tr>
<td>1.4 Purpose of this Thesis and Summary of Contributions</td>
<td>10</td>
</tr>
<tr>
<td>1.5 Thesis Layout</td>
<td>12</td>
</tr>
<tr>
<td>2. Background and Related Work</td>
<td>13</td>
</tr>
<tr>
<td>2.1 Intelligent Tutoring Systems (ITS)</td>
<td>13</td>
</tr>
<tr>
<td>2.2 Recommendation Systems</td>
<td>14</td>
</tr>
<tr>
<td>2.3 Performance Prediction Systems (PPS)</td>
<td>15</td>
</tr>
<tr>
<td>2.4 Probabilistic Mixture Models</td>
<td>16</td>
</tr>
<tr>
<td>3. The Next-Term Student Grade Prediction Problem</td>
<td>18</td>
</tr>
<tr>
<td>3.1 Problem Formulation</td>
<td>18</td>
</tr>
<tr>
<td>3.2 Dataset Description</td>
<td>19</td>
</tr>
<tr>
<td>3.2.1 Content Features</td>
<td>20</td>
</tr>
<tr>
<td>3.2.2 Cold Start Predictions</td>
<td>21</td>
</tr>
<tr>
<td>3.3 Evaluation Metrics</td>
<td>21</td>
</tr>
<tr>
<td>4. The Recommender Systems Approach</td>
<td>23</td>
</tr>
<tr>
<td>4.1 Methods</td>
<td>23</td>
</tr>
<tr>
<td>4.1.1 Simple Baselines</td>
<td>23</td>
</tr>
<tr>
<td>4.1.2 Matrix Factorization Methods</td>
<td>24</td>
</tr>
<tr>
<td>4.1.3 Common Regression Models</td>
<td>26</td>
</tr>
<tr>
<td>4.2 Experimental Results and Discussion</td>
<td>30</td>
</tr>
<tr>
<td>4.2.1 Feature Preprocessing and Selection</td>
<td>30</td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Cold Start Proportion By Academic Term</td>
<td>22</td>
</tr>
<tr>
<td>4.1</td>
<td>Non-Cold Start vs. Cold-Start Prediction RMSE</td>
<td>34</td>
</tr>
<tr>
<td>4.2</td>
<td>Native vs. Transfer Prediction RMSE</td>
<td>35</td>
</tr>
<tr>
<td>4.3</td>
<td>Cold-start vs. Non-Cold-start Error</td>
<td>36</td>
</tr>
<tr>
<td>4.4</td>
<td>RF: Top 10 Features by Gini Importance (GI)</td>
<td>40</td>
</tr>
<tr>
<td>5.1</td>
<td>Parameters for synthetic dataset distribution recapture experiments</td>
<td>51</td>
</tr>
<tr>
<td>5.2</td>
<td>PMixoR Sensitivity to $K$: True vs. Learned Values</td>
<td>52</td>
</tr>
<tr>
<td>5.3</td>
<td>MAE of the Various PMixoR Clusterings on Each Dataset</td>
<td>55</td>
</tr>
<tr>
<td>5.4</td>
<td>Mean Absolute Error (MAE) of Each Model on Each Dataset</td>
<td>56</td>
</tr>
<tr>
<td>5.5</td>
<td>PMixoR vs. PLMR MAE By Term on Engineering Dataset</td>
<td>58</td>
</tr>
<tr>
<td>5.6</td>
<td>MAE of PLMR and PMixoR on Synthetic Data</td>
<td>60</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>FM: Overall MADImp Feature Importance</td>
<td>37</td>
</tr>
<tr>
<td>4.2</td>
<td>PLMR: Feature Importance Evolution for All Data</td>
<td>38</td>
</tr>
<tr>
<td>4.3</td>
<td>PLMR: Feature Importance Comparison</td>
<td>39</td>
</tr>
<tr>
<td>4.4</td>
<td>Cohort by term number error visualization for non-transfer students.</td>
<td>43</td>
</tr>
<tr>
<td>5.1</td>
<td>Graphical Model for PMixoR with two entities, $s = \text{students}$ and $c = \text{courses}$.</td>
<td>47</td>
</tr>
<tr>
<td>5.2</td>
<td>Convergence of PmixoR Parameters to True Parameters of Data Distribution</td>
<td>51</td>
</tr>
<tr>
<td>5.3</td>
<td>Predicted vs. Actual Grades for RF and PmixoR</td>
<td>57</td>
</tr>
<tr>
<td>5.4</td>
<td>PMixoR vs. PLMR MAE BY Term on Engineering Dataset</td>
<td>59</td>
</tr>
<tr>
<td>5.5</td>
<td>MAE of PLMR and PmixoR on Synthetic Data With Increasing Observations</td>
<td>61</td>
</tr>
<tr>
<td>5.6</td>
<td>Highest Posterior Density (HPD) Intervals and Predictions on Simulated Data</td>
<td>62</td>
</tr>
<tr>
<td>5.7</td>
<td>Highest Posterior Density (HPD) Intervals and Predictions on CS Major-Only</td>
<td>63</td>
</tr>
<tr>
<td>5.8</td>
<td>Linear Effect of Chronological Student Term on Performance for Each Profile</td>
<td>64</td>
</tr>
<tr>
<td>5.9</td>
<td>Junior CS Student</td>
<td>67</td>
</tr>
<tr>
<td>5.10</td>
<td>Junior PHIL Student</td>
<td>67</td>
</tr>
</tbody>
</table>
Abstract

INTELLIGENT DEGREE PLANNING SYSTEMS: ADVANCEMENTS IN PERSONALIZED ACHIEVABILITY RATING

Mack Sweeney

George Mason University, 2016

Thesis Director: Dr. Huzefa Rangwala

Student retention to successful graduation is an enduring issue in higher education. National statistics indicate most higher education institutions have four-year degree completion rates around 50%, or just half of their student populations. While there are prediction models which illuminate factors that improve chances of student success, research has yet to clearly identify interventions that support course selections on a semester-to-semester basis. In this thesis, we highlight the potential of an ambitious academic advising program to improve student retention and learning outcomes. Given the complex demands of such an advising program, we posit that the development of an intelligent automated advising system is essential to its success. To further this goal, we develop a system to predict students’ grades in the courses they will enroll in during the next enrollment term. We take a data-driven approach, learning patterns from historical transcript data coupled with additional information about students, courses, and the instructors teaching them.

We explore a variety of classic and state-of-the-art techniques which have proven effective for recommendation tasks in the e-commerce domain. In our experiments, Factorization Machines (FM), Random Forests (RF), and the Personalized Multi-Linear Regression (PLMR) model achieve the lowest prediction error. We introduce a novel feature selection
technique that is key to the predictive success and interpretability of the FM. By comparing feature importance across populations and across models, we uncover strong connections between instructor characteristics and student performance. We also discover key differences between transfer and non-transfer students. Ultimately we find that a hybrid FM-RF method can be used to accurately predict grades for both new and returning students taking both newly introduced and well-established courses.

Unlike most e-commerce recommendations, academic advising can often have long-lasting impacts on the student, on institutions, and on society. National studies show students with a Bachelors degree earn an average of 62% more than those with a high school diploma. Meanwhile, students who start but fail to successfully complete a university degree program represent lost revenue for institutions and generate debt that burdens society; the cumulative losses from dropouts across the United States figure in the billions. In such high-impact advising scenarios, explainability of recommendations becomes essential for their adoption. To address this concern, we explored probabilistic techniques that compete with state-of-the-art methods but yield superior prediction explanations. In particular, we develop a novel method called Profiling Mixtures of Linear Regressions that matches the performance of PLMR. We derive an efficient Gibbs sampling inference algorithm to infer a full posterior distribution for this model. We then demonstrate through a variety of informative visualizations how this posterior distribution can be used to assist advisors in making academic degree planning recommendations that are clear and actionable for advisees.

The work in this thesis represents progress towards a truly intelligent degree planning system. Development of such a system holds promise for student degree planning, instructor interventions, and personalized advising, each of which could improve retention and student learning outcomes. Given the billion-dollar nature of the retention problem, successful application of the techniques in this thesis will bring significant gains for individuals, institutions, and society.
**Keywords.** matrix factorization; grade prediction; cold-start; recommender system; educational data mining; regression
Chapter 1: Introduction

An enduring issue in higher education is student retention to successful graduation [1]. The 2001 National Research Council report identified the critical need to develop innovative approaches to help higher-education institutions retain students, facilitate their timely graduation, and ensure they are well-trained and workforce ready in their field of study [2]. Despite all efforts made since that time, the six-year graduation rate has increased by only 2% from the roughly 57% number observed in 2001 [3]. Meanwhile the four-year graduation rate remains just below 40%. The inability to retain students to degree completion and to facilitate timely graduation represents a significant cost to students, universities, and society in general.

This is a billion dollar problem, so every bit of progress can represent significant gains for individuals, institutions, and society. We believe high-quality, comprehensive academic advising is one of the most effective means available for helping students meet learning objectives and improving the graduation rate. Our primary argument is that an ambitious advising agenda can be enabled by effective automation of key components of the advising process.

In this thesis, we situate this argument in the context of the leading educational literature. We discuss the magnitude of the graduation rate problem then lay out our argument for the importance of academic advising in addressing this problem. Next we highlight shortcomings in current advising programs and lay out an ambitious advising agenda which can help address these shortcomings. Following this formulation, we discuss how machine learning and predictive analytics can be used to support advisers in realizing such an ambitious agenda.

The core of this thesis is a demonstration of various techniques for predicting student
performance. We show how these techniques can be used to provide advisors with actionable, explainable insights which will help them improve the quality and personalization of student advising with little extra effort on their parts. These techniques successfully demonstrate the utility of such automation efforts. We believe the development of such advising automation systems and effective utilization by institutions will improve the graduation rate. Aided by these technologies, advisers will be better able to provide effective, personalized advising that helps students successfully navigate an academic program that meets their learning objectives and prepares them for success in the workforce.

1.1 The Magnitude of the Graduation Rate Problem and the Importance of Advising

We begin by examining some summary statistics to illustrate the magnitude of the graduation rate problem. We then discuss how academic advising is perhaps the most promising avenue for addressing the problem. This is due both to the critical importance of effective advising and to the current shortcomings of most advising programs.

In 2013, young adults (aged 25-34) with Bachelors degrees earned on average 62% more than those with only a high school diploma [3]. On top of that, those who went on to earn a Masters degree earned approximately 23% more than those with a Bachelors. Employment rates were also more favorable for those with a Bachelors degree, with 82% employed as compared to 72.6% for those with some college but no Bachelors and 67% for those with only a high school diploma [3]. The Educational Policy Institute put out a report in 2013 analyzing the average losses of postsecondary institutons due to dropout. According to the study, approximately $16.5 billion were lost by U.S. institutions in the 2010-2011 academic year. The average institution lost $9.9 million, with the average public institution losing $13.2 million and the average private institution losing $8.3 million [4]. A 2011 report by the American Institutes for Research estimated the lifetime earnings of a college graduate can exceed those of someone holding only a high school diploma by over a half
million dollars [5]. For full-time students starting in the Fall of 2002 and failing to graduate six years later, losses in 2008 consist of $3.8 billion in income, which would produce $566 million in federal taxes and $164 million in state taxes. Over a working life of 45 years for this single cohort, an estimated $158 billion in income would be lost. This represents approximately $32 billion in federal income taxes and $7 billion in state income taxes [5].

From the above summary statistics, we can conclude possible annual gains from improving retention rates sit comfortably at thousands of dollars for students, millions for institutions, and billions for society. Stretched over a lifetime, these gains become even more significant. Having established the importance of improving student retention, we next turn to determining a means of achieving this improvement. While there are a variety of minor theories of student retention, there are only six major theories recognized in the literature [6]. These include Spady’s Dropout Process [7], Tinto’s Student Integration Model [8], Bean’s Student Attrition Model [9], Pascarella’s Conceptual Model for Research on Student-Faculty Informal Contact [10], Astin’s Student Involvement Model [11], and Cabrera’s Integrated Model [12], which combines those of Tinto and Bean.

Pietras mapped the various factors studied by these six theories into a common vocabulary, resulting in 9 key factors [6]. These include Spady’s Dropout Process [7], Tinto’s Student Integration Model [8], Bean’s Student Attrition Model [9], Pascarella’s Conceptual Model for Research on Student-Faculty Informal Contact [10], Astin’s Student Involvement Model [11], and Cabrera’s Integrated Model [12], which combines those of Tinto and Bean.

Pietras mapped the various factors studied by these six theories into a common vocabulary, resulting in 9 key factors [6]. These include (1) grade performance, (2) intellectual development, (3) encouragement from family and friends, (4) social integration, (5) intent to persist, (6) institutional commitment/satisfaction, (7) faculty interaction, (8) environmental factors, and (9) pre-college achievement. Of these factors, 1, 4, 5, and 6 are agreed upon by all six theories. Among the rest, only environmental factors are agreed upon by a minority of the theories [6]. The goal of the present study is not to achieve a unified theory of retention. So we adopt the stance that all of these factors are important to address. An effort to improve retention should target as many of these factors as effectively as possible.

There is one activity present across almost all university degree programs which has the potential to influence all nine of these factors: academic advising. The National Academic Advising Association maintains a statement of core values for professional academic advisors [13]. These core values entail a broad array of responsibilities advisors have to their students.
This is emphasized by the following excerpt:

Effective advising requires a broad-based, holistic approach to working with students. Academic advisors develop crucial ties with others who assist students in such diverse areas as admissions, orientation, financial aid, housing, health services, athletics, course selection, satisfaction of academic requirements, special physical and educational needs (e.g., disabilities, study skills, psychological counseling), foreign study, career development, cocurricular programs, and graduation clearance.

Advisors can maintain regular interaction with students from the time they begin applying to the institution to the time they graduate. They can develop close relationships and serve as trusted confidants. They should also serve to help students achieve their academic goals, mediate conflicts in order to maintain satisfaction, and help students balance environmental factors by making appropriate accommodations when possible. A knowledgeable advisor will be able to connect students with campus events and resources to facilitate social integration. They will also be able to help students build an appreciation for the curriculum and mission of the institution while developing themselves intellectually.

Academic advising is positively associated with several variables that have a positive correlation with retention. These include student-faculty contact outside the classroom, student mentoring, utilization of campus support services, effective educational and career planning and decision-making, and student satisfaction with the college experience.

Drake continued to echo these same conclusions in 2011, citing several examples of more recent studies to support this claim. These repeated findings of positive correlations between advising and retention-raising factors represent strong evidence that improving advising is likely to improve retention. Having stated and supported this claim, we next turn to how advising can be improved.

Based on 25 years of ACT research, Cuseo concludes that four factors are essential to but often lacking from academic advising programs. First, every advising program should have a clearly articulated mission statement, but only 54% do. Second, sufficient incentives,
recognition, and rewards should be provided. Only 12% of programs offer incentives or rewards and roughly half of all faculty contracts make no mention of advising at all. Third, the program should establish criteria for recruitment, selection, and deployment of advisors, but over 2/3 of all programs fail to do so. Fourth, substantive orientation, training, and development opportunities should be required for all advisers. Only 1/3 of programs provide this and only 1/4 require it. Several reports separate from the ACT surveys identified advising as the weakest or one of the weakest links in the education of college students [14].

We argue that advising should address two primary goals: (1) maximize probability of timely graduation and (2) maximize number and quality of life-enriching learning opportunities. To help maximize the probability of timely graduation, advisers should:

- remove all guesswork from course selection and scheduling
- minimize potential for overwhelming schedules by helping students spread high-difficulty courses across semesters
- help students account for insufficient prerequisite knowledge at a personalized level

To maximize the number and quality of learning opportunities, advisers should:

- help students filter courses by interests
- match students to instructors based on interests
- balance course difficulty with expected time constraints
- balance students’ declared interests with subject novelty and diversity

Sub-goals which attribute to both objectives include:

- connect student goals and interests with university resource allocation strategies
- establish trust relationship with students to promote adoption and cooperation with university systems
- facilitate and encourage student-faculty interactions
• connect students with similar interests
• identify opportunities for new programs and services to improve student educational experiences

This ambitious advising agenda addresses the first limitation identified by ACT – providing a clearly articulated mission statement. However, the expansiveness of this mission would seem to exacerbate the other limitations identified. Our primary argument is that this expansive set of academic advising objectives can be achieved and greatly improved through the development and adoption of automated advising support. With the increasing automation of these services, the number of advisers required and the administrative demands on their time will be reduced significantly. If the complex constraint satisfaction and interest-matching problems can be partially or completely automated, advisers would have significantly more time to focus on forming trusting, supportive relationships with students. The cooperation between such an intelligent degree planning system and motivated advisors bears great potential for increasing quality of advising, and therefore for increasing student retention.

1.2 An Intelligent Degree Planning System

The development of an intelligent degree planning system that can perform the functions stated above is a complicated and ambitious undertaking. The system must be able to represent and reason about complex degree requirements. Each major a student will enroll in has separate requirements, so this task may involve reasoning about hundreds of different sets of requirements. Furthermore, students may choose to change majors, so an intelligent advising system should anticipate possible changes and advise courses that might count towards both sets of requirements. The requirement satisfaction problem is a preliminary yet necessary step. Not every student has the same interests, the same skills, or the same knowledge. So the decision to pursue a particular degree plan must be personalized to the student. It is easy to recommend the same program to thousands of students, but some
students will find the program easy and uninteresting, while others might possibly find it unachievable given their time constraints and existing knowledge and skills. So the degree plan must also be achievable for the student it is advised for.

On top of meeting the necessary requirements for a degree and being achievable by the student, a degree must also be valuable. Students enroll in universities and pursue degrees because doing so increases earning potential and provides learning opportunities that align with a student’s interests and career goals. So the real value of a degree comes from its subjective value to the student and its market value during the student’s working lifetime. An intelligent advising system should be able to estimate this value and avoid advising degree plans which are valuable in neither sense. Every buyer making a purchase wishes to maximize the return on his or her investment. The purchase of university services to facilitate degree acquisition is no different in this regard. So the system should be capable of estimating the return on investment a particular student will receive from a particular degree. Both the subjective and market value components of this estimation are challenging problems in their own right.

Finally, it is necessary to recognize that a university brings to bear a collection of resources that are limited and valuable. There are a fixed number of instructors available for teaching, a fixed number of faculty available to run the facilities and support university functions, and a fixed number of physical resources. It is misleading and unhelpful to recommend programs to students which the university cannot support. If the instructors aren’t available, the courses aren’t being taught, there is insufficient parking or transportation, or there are no rooms or spaces to teach the courses in, students cannot hope to execute on their degree plans.

So we have four essential components of an intelligent degree planning system. First, it must recommend plans that meet degree requirements, anticipating changes of major and helping students avoid setbacks in those cases. Second, it must ensure recommended plans are achievable by the student the plan is recommended to. Third, it must ensure the plan has an acceptable return on investment for the student. By providing accurate estimates
of reward, an intelligent system can help students make critical decisions regarding which major and subsequently which career to pursue. Fourth, the system must account for limited university resources, recommending only plans which, taken as a whole, can be accommodated by the university. To be sure, a system which has all four of these functionalities would be both useful and valuable. It is also clear that the development of such a system is a challenging and time-consuming undertaking. We believe progress in all four areas are required for a truly intelligent advising system, but progress in any of them represents progress towards improving advising and thereby the graduation rate. In this thesis, we focus our efforts upon the second task – ensuring a recommended degree plan is achievable for a particular student.

1.3 Performance Prediction for Personalized Achievability Rating

The ability to predict student grades in future enrollment terms provides valuable information to aid students, advisors, and educators in achieving the mutually beneficial goal of increased student retention. This information can be used to help students choose the most suitable majors, properly blend courses of varying difficulty in a semester’s schedule, and indicate to advisors and educators when students need additional assistance. Early identification of at-risk students is a key aspect of preventing them from becoming discouraged and dropping out [16]. Incorporation of next-term grade predictions in early-warning systems can increase the number of such students correctly identified, potentially increasing retention rates significantly. Furthermore, successful prediction models will yield valuable insights into what factors impact student success across various subpopulations. These insights can inform policy decisions to increase student engagement to further increase retention.

In this thesis, we develop a system to predict students’ course grades for the next
enrollment term in a traditional university setting. Students take courses over a sequence of academic terms. During each term, students enroll in one or more courses and earn letter grades in the range A-F for each. Our dataset consists of grades from previous terms, which we call historical transcript data, coupled with student demographics, course characteristics, and instructor information. Given this data, our task is to predict the grades students will obtain in the courses they will take in the next term.

With this problem formulation, the next-term student grade prediction problem becomes quite similar to recommendation problems such as rating prediction and next-basket recommendation. Throughout the years, a variety of methods have been used for recommendation in the e-commerce domain. These recommender methods typically fall into one of two categories, depending on the attributes exploited for prediction: collaborative filtering (CF) and content-based (CB) methods. CF models use only the user-item rating matrix for predictions. In our setting, this is the user-course grade matrix with grades for each user in all courses he or she has completed. CB methods also have access to the rating matrix but additionally incorporate either user profiles, item descriptions, or both. For grade prediction, these features consist of student demographics, performance history, and course features. These are collectively referred to as content features.

The present effort explores a variety of recommender system methods that leverage content features, blending CF and CB modeling ideas in powerful hybrid models. We also add data from students who transfer to the university and use various feature importance metrics to explore the different student characteristics with regards to course performance. This final analysis yields several interesting insights regarding the importance of instructors and the differing factors that affect transfer and non-transfer (native) student performance.

The task of predicting students’ grades in the coming term is complicated by the ever-expanding volume of data from increasing student enrollments and by the continually shifting characteristics of the overall student population. Furthermore, recommender systems

\footnote{Source available at: \url{https://github.com/macks22/ntsgp}}
typically have issues predicting for previously unseen users and items; these issues are collectively called the *cold-start problem*. Our experiments show that Factorization Machines (FMs), Random Forests (RFs), and the Personalized Multi-Linear Regression (PMLR) model achieve the lowest prediction error. While these methods also struggle with the cold-start problem, we found that incorporation of key content features results in improved predictions from at least one of the top three methods for each cold-start scenario. Application of a novel feature selection metric for the FM model greatly improves its accuracy. With the improved FM model, we were also able to devise a hybrid FM-RF method that outperforms all three individual methods, exploiting the strengths of both models to overcome the cold-start problem.

While the methods explored and developed in these initial efforts succeed in making accurate predictions for new and returning students, they are often difficult to interpret. Students have a lot riding on successful completion of their degree programs, so they are often unlikely to act upon advice which they cannot understand. We sought to address this problem by developing a probabilistic method similar to PLMR. We developed a fully Bayesian method called Profiling Mixtures of Linear Regressions (PMixoR) that matches the performance of PLMR while producing far more explainable predictions. We achieve this improved explainability using a probabilistic model and full posterior inference via an efficient Gibbs sampler. Given the posterior distribution of this model, we produce a variety of visualizations and demonstrate how an advisor could use them to make clear and actionable recommendations to students.

### 1.4 Purpose of this Thesis and Summary of Contributions

As we have discussed above, the development of an intelligent degree planning system can be employed by academic advisors to improve student learning outcomes and graduation rates. Progress in this effort will yield financial gains for students, universities, and society measuring cumulatively in the billions annually. While the financial gain is easier to measure, progress towards this goal can also yield significant quality-of-life improvements for...
students who are thereby gainfully employed in professions that match their goals. We have identified the four key functionalities of such a system. In this thesis, we focus specifically on the second task – ensuring a recommended degree plan is achievable for a particular student. We explore requirements for this task and in doing so arrive at the next-term grade prediction problem. We investigate existing methods for this task, identify their limitations, develop new methodologies to tackle the problem, and then enumerate future directions to continue these efforts. Our contributions are summarized as follows:

- Identify the major components of an intelligent degree planning system and formulate how such a system might be developed and employed to increase student retention in university degree programs.

- Investigate existing methods for the next-term grade prediction problem, whose solution is a key step in personalized achievability rating for recommended degree plans.

- Develop state-of-the-art methods for making accurate and explainable next-term grade predictions.

- Identify several promising areas for future work based on an analysis of characteristics and limitations of the methodologies employed in this study.

The work in this thesis synthesizes two years of work. We first addressed the next-term grade prediction problem from a collaborative filtering perspective; this work is published in our paper Next-Term Student Grade Prediction [18]. In our work titled Next-Term Student Performance Prediction: A Recommender Systems Approach, we expanded the amount of data and number of features analyzed. To handle this increased data size and variety, we brought to bear more varied and powerful regression models. [19]. A summarizing report of our findings and those of our collaborators at the University of Minnesota were published in the IEEE Computer article Predicting Student Performance Using Personalized Analytics [20]. Finally, we intend to submit the work in chapter 5 of this thesis to AAAI this fall.
1.5 Thesis Layout

The rest of this document is organized as follows. Chapter 2 explores related work in the areas of intelligent tutoring systems, recommender systems, and performance prediction systems. This work is related to the current effort in order to provide a cohesive view of our foundations and our contributions. Chapter 3 formulates the next-term performance prediction problem in more detail, describes the dataset used in this study, and lays out experimental procedures, objectives, and metrics. In chapter 4, we experiment with a variety of existing techniques from recommender systems and academic literature to develop an effective next-term student grade prediction system. We also develop a novel feature importance technique and a hybrid state-of-the-art model which uses this technique. Finally, we examine the importance of various features to demonstrate how the predictions of these models can be used to gain insight into student performance characteristics and aid in the advising effort.

In chapter 5, we develop a novel probabilistic method called Profiling Mixtures of Linear Regressions. We compare this method to the best performers from chapter 4 to demonstrate its competitive performance. Then we perform several case studies to illustrate its interpretability; these studies indicate that full posterior inference of probabilistic models can provide particularly useful information for advising. In chapter 6, we make observations about our results, discussing the performance of our methods and the beneficial applications that might be achieved with them. Chapter 7 concludes with a summary of our findings and discusses several promising areas for future work. We hope the ideas in this final section spark further research into this area and hence progress towards the worthy goal of increasing student retention in university degree programs.
Chapter 2: Background and Related Work

2.1 Intelligent Tutoring Systems (ITS)

Despite the differences, there are some interesting similarities between the present work and recent work in predicting correct first attempt (CFA) in ITS. CFA is a binary indicator: 0 if a student gets the problem/task correct on the first try and 1 otherwise. These binary targets are aggregated to produce real-valued "ratings" in the [0, 1] range. Thai-nghe utilizes standard MF and slightly customized tensor factorization (TF) models [21]. Results demonstrate that MF and TF methods achieve good predictive performance when compared with logistic regression and simple baselines. The MF models used in that study are subsumed by the FM model used in the present study. The TF model was employed to capture 3-way user-item-time interactions. While we do explore the utility of 1- and 2-way interaction terms, we do not explore 3-way interactions in this study. This would be an interesting direction for future work.

Both [22] and [23] apply multi-relational matrix factorization (MRMF) models to CFA prediction. This method simultaneously performs MF on several "rating" matrices for complementary relations, such as "student solves a problem", "problem requires a skill", and "student has a skill." Jointly decomposing several matrices into partially shared factorized matrices is empirically shown to more effectively regularize the entities in the primary relation, yielding improved predictions. Applying unequal weights (WMRMF) to these joint learning objectives can further improve results. Adding bias terms to both the MRMF and WMRMF models can achieve even further improvements. Thai-Nghe et. al. also apply the FM model to the task of CFA prediction [24]. However, they limit their data to the user-item matrix and learn the model with SGD. In this setting, the FM model reduces to a regularized Singular Value Decomposition (SVD) with bias terms and a global intercept.
We studied the ability of this and other similar models for predicting grades on a subset of the present dataset in a prior work [18].

ITS researchers have also been very interested in student modelling and knowledge tracing. In a seminal paper, Corbett and Anderson presented an analysis whose task was modeling the changing knowledge states of students during skill acquisition [25]. These student models often form core components of ITS systems. Desmarais and Baker provide a broad review of ITS systems in [26]. One example of recent state-of-the-art includes Xu and Mostow’s logistic regression dynamic bayes net (LR-DBN) model for tracing multiple student subskills over time [27]. They also recently conducted a morphological analysis of prominent student models, surveying recent methods and identifying gaps to be filled [28]. Barnes and Stamper demonstrated another interesting direction with their use of Markov Decision Processes (MDPs) for automatically generating hints for an intelligent logic proof tutor [29]. Baker et. al also developed systems that can adapt and intervene based on predictions of future student performance, and they were able to show these interventions were effective in improving students’ experiences [30]. It would be very interesting to explore the modeling techniques leveraged in these studies for the task of grade prediction in future work. The present study makes no attempt to model learner knowledge states or to provide interactive feedback during activities.

2.2 Recommendation Systems

Recommendation methods include nearest neighbor approaches [31] [32] [33], matrix factorization (MF) for collaborative filtering [34] [35], restricted boltzmann machines [36], and topic modeling methods [37], among others. Much of the research pertaining to recommendation tasks has been conducted in the domain of e-commerce. In particular, the task of movie recommendation was widely studied during the Netflix Prize competition [17]. Many companies have come to rely heavily on large-scale recommender systems for reducing information overload and targeting advertisements in industrial settings [31] [38].
In this paper we study the application of recommender system technology to student grade prediction within a traditional university setting. Similar work deals mostly with online learning environments, such as Massive Open Online Courses (MOOCs) \cite{40,41,42} and Intelligent Tutoring Systems (ITS) \cite{21,22,23,24}. In contrast, the goal in the present study is to predict grades in a traditional university learning environment. Online learning environments provide a variety of detailed student behavior data that is not available in traditional learning environments. The timeline of those studies is also more granular. In MOOCs, studies on predicting student performance (PSP) usually seek to predict grades for homeworks, quizzes, and exams within a single course. Some studies do span multiple courses, but the timelines are often a single year or less \cite{42}.

\subsection{Performance Prediction Systems (PPS)}

While most recent research is focused on online learning, there are some recent studies which perform grade prediction in university settings as we do. In one such study, comments from learning reflection assignments in a course are used as features for grade prediction \cite{43}. Latent Semantic Analysis (LSA) is used to extract topics and reduce dimensionality of the comments. Then k-means is used to cluster the comments into per-grade clusters. Rhodes et. al. also leverage lexical features from student self-reflection for performance prediction \cite{44}. By analyzing these features together with student effort features (time spent writing solutions), they find that self-explanation vocabulary correlates with amount of effort expended. The features in these studies consist only of bag-of-words term vectors. The present study does not leverage any such text features, since none were readily available in our dataset.

In our previous work, we studied the next-term grade prediction task without incorporating content features \cite{18} (only CF methods). The present effort expands upon this work to explore a variety of methods that leverage content features, blending CF and CB
modeling ideas in powerful hybrid models. In another recent study by Elbadrawy et. al. \cite{45}, the authors devise a custom mixed-membership multi-linear regression model (PMLR – mentioned above) to predict student grades in a traditional university setting. They use a variety of data, including grades and learning management system (LMS) features. Use of LMS data allows prediction of grades at the "activity" level – individual assessments within a given course. The present study does not use data from or predict grades for individual activities. Instead, grades are predicted only at the course level using only past student grades. We do, however, compare against the PMLR model in our study and find it to be one of the top three performers.

### 2.4 Probabilistic Mixture Models

Mixture models have been studied for many years, and mixtures of regression models for at least 40 years \cite{46,48}. Initial mixtures of linear regressions were explored under the names switching regression and clusterwise linear regression. More recently, Gaffney and Smith \cite{49} developed an Expectation-Maximization algorithm for a finite mixture of linear regressions for clustering trajectories of observations. This model involved linear regression components and trajectory-specific membership weights.

Around that time, Radford Neal \cite{50} published his far-reaching paper on sampling techniques for Dirichlet Process Mixture Models. Since these early trailblazing works, there has been a variety of research in this area. Kottas et al. \cite{51} explored mixtures of regressions in a model that explicitly considers ordinal output data. They present a cutoff technique which we use in this work. There has also been a variety of work exploring various technical issues in dealing with these models \cite{52,55}. Finally, Chen and Shao \cite{56} present a method for computing highest posterior density (HPD) intervals from Markov Chain Monte Carlo (MCMC) samples we use in this paper.

In chapter 5, we develop a probabilistic model (PMixoR) that can be viewed as an extension on the combined work of Gaffney and Smith \cite{49} and Kottas et al. \cite{51}. We built
on the work of Neal [50] while developing our collapsed Gibbs sampling algorithm and that of Hennig [54] to help address the identifiability problems that can arise in mixture models.
Chapter 3: The Next-Term Student Grade Prediction

Problem

3.1 Problem Formulation

Given a database of (student, course) dyads (i.e., pairs) with associated content features for the course, student, and course instructor, our goal is to predict grades for each student for the next enrollment term. More formally, we have $n$ students and $m$ courses, comprising an $n \times m$ sparse grade matrix $G$, where $\{G_{ij} \in \mathbb{R} \mid 0 \leq G_{ij} \leq 4\}$ is the grade student $i$ earned in course $j$. This is the primary source of information leveraged by matrix factorization techniques, such as Singular Value Decomposition (SVD). For those methods, the task is often cast as a matrix completion problem. However, we do not wish to complete the entire matrix. The work laid out in this paper assumes the student(s) have already selected a set or several possible sets of courses to take in the coming term. Our goal is to provide feedback to students, advisors, and educators regarding the selection(s) made or proposed. Recent research investigates how to predict the courses students will select in the coming term [57] and how to identify selections that satisfy degree requirements [58]. These efforts could be combined with ours to recommend additional courses or detect when students have made selections which will put them behind in their degree plan.

We consider each cell to be a (student, course) dyad and represent it as a feature vector $X_{ij} \in \mathbb{R}^{1 \times p}$. The first $n$ variables are a 1-of-$n$ vector representing the one-hot-encoded student IDs and the next $m$ are a 1-of-$m$ vector for the course IDs. The rest of the $p$ total features consist of content features generated by student $i$ taking course $j$. We train our models on all feature vectors $X_{ij}$ preceding the current term and predict grades $\hat{G}_{ij}$ for all feature vectors $X_{ij}$ in the current term. This setup is critical for avoiding data leakage – a
severe experimental misstep that can inflate real-world significance of performance metrics [59]. For fair and effective evaluation of the proposed methods, we train one model per academic term in the dataset, and we use that model to predict only for this term. All results presented in this paper represent an aggregate of predictions obtained with this sequential train-predict loop.

### 3.2 Dataset Description

The data used for this study comes from a public university with an enrollment of 33,000 students as of Fall 2014 [60]. Observations begin in Summer 2009 and continue until Spring 2014. Each year there is a Spring, Summer, and Fall term; these terms span from January to May, May to August, and August to December. Hence our dataset contains a total of 15 terms. All students whose cohorts pre-date Summer 2009 were excluded from the data. After preprocessing, there are 30,754 students declared in one of 144 majors, each of which belongs to one of 13 colleges. 584,179 (65.29%) of these are transfer students and the other 310,557 are non-transfer (34.71%). During this time period, these students have taken 9,085 unique courses, each of which is classified as one of 161 disciplines and taught by one of 6,347 instructors. After discarding records with no grades or grades which do not translate to the A-F scale (such as withdrawals and audits), we have 894,736 dyads. All this data was collected and anonymized in accordance with Institutional Review Board (IRB) policies.

The A-F letter grades are discrete, but they have ordinal equivalents in the range 0-4. This mapping is common in American universities, but for those unfamiliar, it proceeds as follows. An F is mapped to 0, a D to 1, a C- to a 1.67. From there, we have C, C+, B-, B, B+, A-, and A, which correspond to the numbers from 2 to 4 in increments of $\frac{1}{3}$. As a result of the ordinal mapping, this problem could be cast as either classification or regression. We explored both options in our preliminary experiments. In particular, we looked at using kNN, a Gaussian Mixture Model (GMM), Support Vector Machine (SVM),

---

1George Mason University (GMU)
boosted decision trees, random forests, standard decision trees, and the FM. All of these classification methods performed poorly compared to the regression techniques; we believe this is because classification models fail to capture the ordinal nature of the data. In order to properly capture this ordering information, we cast the problem as a regression task.

3.2.1 Content Features

For each dyad, we have a variety of student, course, and instructor features, either categorical or real-valued. For each of these three entities, we have unique identifiers and a variety of other information. For students, we have demographics data, such as age, race, sex, high school CEEB code and GPA, zip code, and 1600-scale SAT scores. For each dyad, we have the declared major of the student and the grade earned. For each term, we have the GPA from the previous term as well as the cumulative GPA. We have the number of credit hours the student is enrolled in during the current term and the number of credit hours attempted up to the current term. We also have an academic level obtained from credit hours attempted. Finally, we annotate each term for each student with that student’s relative term number. This feature reflects the number of terms the student has taken courses for.

There is also a variety of data for courses. Each course belongs to a particular discipline, is worth a fixed number of credit hours, and is assigned a particular course level. For each term, we have the aggregate GPA of the course from the previous term as well as the cumulative aggregate GPA over all terms the course has previously been offered (in our dataset). We have the number of students enrolled in all sections during this term, as well as the total number of students enrolled for all prior terms the course has been offered. In addition to information specific to the course itself, the course side information also includes instructor information. For each instructor, we have his or her classification (Adjunct, Full time, Part time, Graduate Research Assistant, Graduate Teaching Assistant), rank (Instructor, Assistant Professor, Associate Professor, Eminent Scholar, University Professor), and tenure status (Term, Tenure-track, Tenured). Transfer course records are mapped to GMU
equivalents. Instructor information for these records is mostly absent. So we use the ID of the institution of origin to substitute for an instructor ID and leave out the other instructor features. A more detailed listing of the features used in this study is given in Appendix A.

### 3.2.2 Cold Start Predictions

In the context of the next-term prediction task, cold-start records are defined as (student, course) dyads for which either or both the student and course appear in a term (prediction phase) but do not appear in any of the previous terms (training phase). In our dataset, 447,378 (50.00%) dyads are non-cold-start and 447,358 (50.00%) are cold-start. 41,843 (9.35%) of these are dyads for which both the student and the course are cold-start. 389,449 (87.06%) are student-only cold-start, and 16,066 (3.59%) are course-only cold-start. Table 3.1 breaks down the proportion of cold-start records by academic term. For instance, 98.69% of the dyads are cold-start for the Fall 2009 cohort. The only previous enrollment was in Summer 2009, so we know the other 1.31% of the dyads represent students who enrolled in the previous summer.

### 3.3 Evaluation Metrics

Evaluations are performed in terms of two common regression metrics: Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE).

\[
RMSE = \sqrt{\frac{\sum_{i=1}^{N} (\hat{G}_{ij} - G_{ij})^2}{N}} \quad MAE = \frac{\sum_{i=1}^{N} |\hat{G}_{ij} - G_{ij}|}{N}
\]

RMSE penalizes severe prediction errors more heavily than small ones. Given our task, we would prefer not to declare a method the best if it performs very well for half the students and very poorly for the other half. Hence RMSE is the metric we use to compare methods. MAE allows us to understand the range of grades we might actually be predicting. If a student is actually going to get a B, and the MAE is 0.33, we expect our model to predict
Table 3.1: Cold Start Proportion By Academic Term

<table>
<thead>
<tr>
<th>Term</th>
<th>Term #</th>
<th>Dyads</th>
<th>NCS</th>
<th>CS</th>
<th>% CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>'09 Summer</td>
<td>0</td>
<td>252</td>
<td>0</td>
<td>252</td>
<td>100.00</td>
</tr>
<tr>
<td>'09 Fall</td>
<td>1</td>
<td>62,074</td>
<td>813</td>
<td>61,261</td>
<td>98.69</td>
</tr>
<tr>
<td>'10 Spring</td>
<td>2</td>
<td>41,075</td>
<td>17,545</td>
<td>23,530</td>
<td>57.29</td>
</tr>
<tr>
<td>'10 Summer</td>
<td>3</td>
<td>3,577</td>
<td>3,165</td>
<td>412</td>
<td>11.52</td>
</tr>
<tr>
<td>'10 Fall</td>
<td>4</td>
<td>80,287</td>
<td>21,685</td>
<td>58,602</td>
<td>72.99</td>
</tr>
<tr>
<td>'11 Spring</td>
<td>5</td>
<td>64,295</td>
<td>37,358</td>
<td>26,937</td>
<td>41.90</td>
</tr>
<tr>
<td>'11 Summer</td>
<td>6</td>
<td>7,043</td>
<td>6,734</td>
<td>309</td>
<td>4.39</td>
</tr>
<tr>
<td>'11 Fall</td>
<td>7</td>
<td>98,673</td>
<td>36,166</td>
<td>62,507</td>
<td>63.35</td>
</tr>
<tr>
<td>'12 Spring</td>
<td>8</td>
<td>80,200</td>
<td>53,587</td>
<td>26,613</td>
<td>33.18</td>
</tr>
<tr>
<td>'12 Summer</td>
<td>9</td>
<td>8,880</td>
<td>8,546</td>
<td>334</td>
<td>3.76</td>
</tr>
<tr>
<td>'12 Fall</td>
<td>10</td>
<td>107,511</td>
<td>51,130</td>
<td>56,381</td>
<td>52.44</td>
</tr>
<tr>
<td>'13 Spring</td>
<td>11</td>
<td>100,342</td>
<td>64,720</td>
<td>35,622</td>
<td>35.50</td>
</tr>
<tr>
<td>'13 Summer</td>
<td>12</td>
<td>10,678</td>
<td>10,388</td>
<td>290</td>
<td>2.72</td>
</tr>
<tr>
<td>'13 Fall</td>
<td>13</td>
<td>125,312</td>
<td>58,901</td>
<td>66,411</td>
<td>53.00</td>
</tr>
<tr>
<td>'14 Spring</td>
<td>14</td>
<td>104,537</td>
<td>76,640</td>
<td>27,897</td>
<td>26.69</td>
</tr>
</tbody>
</table>

CS: Cold-Start, NCS: Non Cold-Start
Term # denotes chronological ordering of academic terms in dataset.
Cold-start dyads have either a new student, a new course, or both.

either a B-, B, or B+. All MAE measurements are accompanied by the standard deviation of the absolute errors. Note that both of these metrics measure error, so a smaller number is better.
Chapter 4: The Recommender Systems Approach

4.1 Methods

We explore three classes of methods for the next-term student grade prediction task. These are (1) simple baselines, (2) MF-based methods, and (3) common regression models. The first two classes are methods which do not incorporate content features. The last consists of traditional regression methods which must necessarily incorporate content features. Since the FM model can use any features, it falls into (3) when incorporating content features and (2) otherwise.

4.1.1 Simple Baselines

We devised three simple baselines to better understand the effect of leveraging three types of central tendencies in the data.

- **Uniform Random (UR):** Randomly predict grades from a uniform distribution over the range [0, 4].
- **Global Mean (GM):** Predict grades using the mean of all previously observed grades.
- **Mean of Means (MoM):** Predict grades using an average of the global mean, the per-student mean, and the per-course mean.

The UR method illustrates the result of making predictions by randomly guessing. The GM method illustrates the informative value of the overall central tendency of the data, often called the *global intercept*. The MoM method takes this strategy one step further, also incorporating the per-student and per-course (row and column) averages. When compared to the GM method, this illustrates the added benefit of a small level of personalization and
historical knowledge about the course difficulty. In the cold-start setting, we allow the MoM method to use whatever information is available. For a particular cold-start dyad, either the student or course may be present, but not both. If neither is present, it reduces to the GM method.

4.1.2 Matrix Factorization Methods

We look at three methods based on Matrix Factorization (MF):

- **Singular Value Decomposition (SVD).**
- **SVD-kNN**: SVD post-processed with kNN [61].
- **Factorization Machine (FM) [62].**

Each of these models attempt to capture the pairwise interaction of features by decomposing the feature space into a $k$-rank reduced subspace. This results in two sets of latent vectors: one for the courses ($v_j \in \mathbb{R}^k$), and one for the students ($v_i \in \mathbb{R}^k$), which can be thought of as latent course characteristics and latent student competencies or knowledge states for each of these general course characteristics, respectively. These latent feature vectors can be considered a less noisy, condensed representation of the student and course information. For SVD, each grade is simply predicted as the dot product of the latent student and course feature vectors. We call this the factorized 2-way interaction of the student with the course. This differs from a simple linear regression in its use of latent feature vectors to deal with sparsity in the data. It also differs in its use of 2-way interactions, which capture the effects that interactions of two predictor variables have on the target variable (the grade).

$$\hat{G}_{ij}^{(SVD)} = \sum_{f=1}^{k} v_{i,f} v_{j,f} = v_i^T v_j.$$ (4.1)

Paterek [61] showed that post-processing SVD with k-nearest neighbors (kNN) yields improved predictive performance. In essence, the predicted grade for a course is replaced with
the predicted grade for the most similar course. Similarity is calculated as cosine similarity in the latent feature space. Though this method has not, to our knowledge, been analyzed formally, it has been shown to be effective in practice. It can be thought of as a cluster-based smoothing, where the clusters are computed from latent course characteristics.

Pure collaborative filtering (CF) methods such as SVD and SVD-kNN are unable to make predictions for cold-start records. Without any previous observations for a student or a course, no latent student competencies or latent course characteristics can be learned. Content-based (CB) methods handle course cold-start by incorporating features describing each item and/or student profiles such as demographics [63]. FMs can incorporate arbitrary content features while also leveraging the sparse student-course grade matrix. This capability means the FM is a hybrid model which can incorporate the data traditionally used by CB and CF methods in one model. Combined with suitable feature engineering, this allows FMs to subsume most state-of-the-art factorization-based recommender system models developed up to this point [62].

We observe that the FM model is able to capture all of the information captured by the simple baselines as well as the information captured by SVD. In general, it captures the global central tendency, 1-way (linear) relationships between the predictors and the grade (bias terms), and 2-way factorized interactions between each predictor and the grade:

\[
\hat{G}_{ij}^{(FM)} = w_0 + \sum_{l=1}^{p} w_l x_l + \sum_{l=1}^{p} \sum_{l'=l+1}^{p} x_l x_{l'} (v_l^T v_{l'})
\] (4.2)

The FM model can be understood as an adaptation of second order polynomial regression (PR) for sparse data. This is particularly evident from comparing the FM formula with that of the second order PR model:

\[
\hat{G}_{ij}^{(PR)} = w_0 + \sum_{l=1}^{p} w_l x_l + \sum_{l=1}^{p} \sum_{l'=l}^{p} x_l x_{l'} w_{l,l'}
\] (4.3)
We use overloaded notation here, representing both the 1-way interaction terms and the 2-way interaction terms with $w$ and differentiating by the number of subscripts. There are two main differences between the FM and PR model. First, the 2-way interactions are not factorized – note the replacement of $v_i^T v_p$ with $w_{l,l'}$. This means the model will be less able to deal with data sparsity and hence unlikely to learn patterns for categorical data effectively. Each interaction can only be learned if it occurs in the data. Consider the student and course interaction features. We can only learn a pattern for a student in a particular course if we observe the outcome of the student taking the course. Even if we were to keep outcomes from students retaking courses, this would not happen for the majority of courses. So by the time we have learned the interaction, it is unlikely to be useful to us; students already know what grade they got in courses they have completed.

The second difference is the inclusion of squared terms, i.e. 2-way interactions between each term and itself. These terms were not included in the FM model for *properness* and *multilinearity*; for more formal details, see [64].

We compare SVD and SVD-kNN to the FM model trained using only the student-course grade matrix $G$. With this data, the model reduces to the sum of global ($w_0$), student ($w_i$), and course ($w_j$) bias terms and the factorized interaction of the student with the course ($v_i^T v_j$). This last term is the same dot product seen in the SVD model. The other terms capture information modeled by the naive baselines.

$$\hat{G}_{ij}^{(FM)} = w_0 + w_i + w_j + v_i^T v_j. \quad (4.4)$$

We use the fastFM library [65] for a fast implementation of the FM algorithm. The model parameters are learned with Gibbs sampling, which is a common Markov Chain Monte Carlo (MCMC) learning algorithm.

### 4.1.3 Common Regression Models

We tested four different regression models:
• Random Forest (RF)
• Stochastic Gradient Descent (SGD) Regression
• k-Nearest Neighbors (kNN)
• Personalized Linear Multiple Regression (PLMR)

We leveraged the scikit-learn library \[66\] for the first three, which are common regression models. Parameter settings for each method were found using grid search on a sampled held-out set. These are listed in Instructor Features. We briefly describe each of these models.

Random Forest

The Random Forest algorithm combines a group of random decision trees in a bagging ensemble \[67\]. A single non-random decision tree is constructed by discovering the most informative questions to ask in order to split all samples into groups with similar target attribute values. Each question splits the data into two or more groups by thresholding some feature of the samples. So we end up with a tree of decision nodes. For regression, the most informative questions are those that produce leaf nodes whose mean squared error (MSE) are minimal among all possible splits.

Tree construction stops once an additional split would not reduce MSE. Since this usually overfits, an early-termination criterion is often specified. This is usually maximum tree depth or minimum number of nodes at each leaf; we used the former for our experiments. Once built, the tree can be used for regression of new data samples. A sample is run through the decision-making sequence defined by the structure of the tree until reaching a leaf. Then the prediction is the mean of the grades of the samples at that node when training finished.

A random decision tree results from learning a decision tree on a bootstrap data sample and considering a random subset of features for each split. This “feature bagging” is done to reduce correlation between the trees. The Random Forest then combines many of these trees in a weighted averaging approach to make decisions regarding unseen data. This reduces the variance of the individual trees while retaining the low-bias.
Stochastic Gradient Descent (SGD) Regression

The SGD regression method learns a least squares linear regression fit under an L1 regularization penalty (absolute norm). The least squares fit minimizes the squared difference between actual and predicted grades, while the L1 regularization penalty encourages feature sparsity. In particular, unimportant parameters have a tendency to be pushed towards 0, so the L1 penalty operates as a kind of online feature selection. In contrast to ordinary least squares (OLS), this method is an approximate best fit, learned using stochastic gradient descent (SGD). SGD is a gradient-based optimization technique that updates the model parameters incrementally, rather than on the entire training set at once (which is what normal gradient descent would do). This reduces overfitting and significantly improves training time.

k-Nearest Neighbors

The k-Nearest Neighbors (kNN) algorithm is a classic method for clustering samples based on similarity. These clusterings can be used for regression in the following manner. First a pairwise distance metric is used to identify the $k$ most similar neighbors among all dyads in the training set. The grade is then predicted using local interpolation of the targets associated with these neighbors. Many different distance metrics can be used; for our experiments, we use standard Euclidean distance:

$$Euclidean(X_{i,j}, X_{i',j'}) = \sqrt{\sum_{f=1}^{p} (X_{i,j,f} - X_{i',j',f})^2}.$$

(4.5)

To predict a grade for a new dyad $(i, j)$, the Euclidean distance from $(i, j)$ to every dyad in the training set is computed. The $k$ dyads $(i', j')$ with the smallest distance are selected and placed into a set of neighbors $N_{i,j}$. The grade for student $i$ in course $j$ is then predicted as the uniformly weighted average of these neighbors’ grades.
Personalized Linear Multiple Regression

We also employ the more recently developed Personalized Linear Multiple Regression model from [45]. The original model predicts a missing grade $G_{ij}$ for student $i$ in course $j$ using:

$$\hat{G}_{ij}^{(PLMR)} = s_i + c_j + \sum_{l=1}^{k} P_{il} \sum_{f=1}^{p} W_{lf} X_{ijf}$$

(4.6)

$$= s_i + c_j + P_i W X_{ij},$$

(4.7)

where $s_i$ is a bias term for student $i$, $c_j$ is a bias term for course $j$. PLMR uses a linear combination of $k$ regression models weighted on a per-student basis. $P_i$ is the $1 \times k$ vector of model weights for student $i$ and $W$ is the $k \times p$ matrix of regression coefficients. $X_{ij}$ is the feature vector generated by student $i$ taking course $j$. The model is learned using the following objective function. Root Mean Squared Error (RMSE) is used as the loss function $\mathcal{L}(\cdot)$, $P$, and $W$ are regularized using the squared Frobenius norm, and all parameters are constrained to be non-negative.

$$\minimize_{P, W, s, c} \mathcal{L}(P, W, s, c) + \lambda_W (\|P\|_F^2 + \|W\|_F^2)$$

(4.8)

We implemented our own version of the PLMR model, adding a global intercept term $w_0$ and a regularization on the bias terms controlled by parameter $\lambda_b$. This global intercept captures sample-wide trends, just as in the GM method. Preliminary experimentation showed the regularization term improves this method (particularly for cold-start scenarios) by avoiding overconfident bias term learning when observing small sample sizes. The resulting model and objective function are shown below. All parameters are constrained to be
non-negative, as in the original model.

\[
\hat{G}_{ij}^{(PLMR)} = \text{new}_{ij} = w_0 + s_i + c_j + P_1 W X_{ij}
\]  \hspace{1cm} (4.9)

\[
\min_{P,W,s,c} \mathcal{L}(P,W,s,c) + \lambda_w(\|P\|_f^2 + \|W\|_f^2) + \lambda_b(\|s\|_f^2 + \|c\|_f^2)_{\text{new}}
\]  \hspace{1cm} (4.10)

4.2 Experimental Results and Discussion

4.2.1 Feature Preprocessing and Selection

For preprocessing, missing values for each real-valued feature were filled in using the medians. This was not done for the FM model, since it handles missing data without loss of performance and performs worse with the median-value imputation. After this step, the real-valued attributes were scaled using Z-score scaling. Finally, the predictions of all methods are post-processed to bound the grade predictions to the range \([0, 4]\); all predictions below 0 are set to 0 and all those above 4 are set to 4.

Unlike any of the other models used, FMs are capable of learning effectively from both categorical and real-valued features. We would like to maximize the amount of differentiating information which can be captured by the 2-way interaction factors of the model. For this purpose, when we have the choice between encoding a feature as categorical (one-hot encoded) or real-valued (single feature with ordinal value), we choose the categorical encoding. This allows unique 2-way interactions to be learned for each combination of categories for all categorical features. Among the techniques which can incorporate content features, the FM model is the only one which leverages MF techniques to learn these sparse 2-way interactions effectively. With this in mind, we do not include highly sparse features (such as the instructor IDs) as training data for the other models. This reduces computational overhead at an insignificant loss to performance.
FM Feature Selection: A Novel Importance Metric

In our experiments, we found that the FM model has particularly poor feature selection capabilities, unlike the L-1-regularized regression model and the decision-tree-based models. We introduce a new feature importance metric called Mean Absolute Deviation Importance (MADImp).

Inspired by the work of [45], MADImp is a method for computing the importance of each feature in any generalized linear model (GLM). Elbadrawy et. al. compute importance for a model term as the proportion of the overall prediction accounted for by that term, normalized over all records. This is greatly simplified by the non-negativity constraints enforced in the particular linear model used by those authors. Since not all GLMs have non-negativity constraints, we cannot use simple averaging. Instead, we use a quantity computed from the mean absolute deviation from a global intercept term.

For each dyad, the active features (non-zero) are involved in one or more additive interaction terms which move the prediction away from the global intercept \( w_0 \). To measure importance, we sum the absolute values of these additive terms for a particular feature and divide by the total absolute deviations caused by all features. Importance is measured as the proportion of the absolute deviation from \( w_0 \) accounted for by a particular feature. We average this over all records – hence the name Mean Absolute Deviation Importance (MADImp). For a practical example that solidifies these ideas, see Instructor Features.

We now define MADImp more formally. Throughout this section we use \( d \) as shorthand for some dyad indices \( i, j \). The importance of each feature for a single dyad is calculated as:

\[
I(X_{d,f}) = \frac{\sigma_1(X_{d,f}) + \sigma_2(X_{d,f})}{T_d},
\]

where \( \sigma_1(X_{d,f}) \) is the deviation from 1-way interactions of feature \( f \) in dyad \( d \), \( \sigma_2(X_{d,f}) \) is the deviation from 2-way interactions of feature \( f \) in dyad \( d \), and \( T_d \) is the total deviation.
from the global intercept $w_0$ in the estimation for dyad $d$.

$$\sigma_1(X_{d,f}) = |w_f X_{d,f}| \quad (4.12)$$

$$\sigma_2(X_{d,f}) = X_{d,f}^2 \sum_{f' = 1}^{p} \frac{|X_{d,f} Z_{f,f'}|}{|X_{d,f}| + |X_{d,f'}|} \quad (4.13)$$

$$T_d = \sum_{f=1}^{p} |w_f X_{d,f}| + \sum_{f=1}^{p} \sum_{f'=f+1}^{p} |X_{d,f} X_{d,f'} Z_{f,f'}| \quad (4.14)$$

The sum of the importance measures of all features for a dyad equals the total deviation for that dyad: $\sum_{f=1}^{p} I(X_{d,f}) = T_d$. We can calculate the overall importance of a feature in our training dataset through:

$$I(X_{*,f}) = \sum_{d=1}^{p} I(X_{d,f}) \quad (4.15)$$

In general, to calculate MADImp for $X_{*,f}$ for any GLM, group all terms $X_{*,f}$ is involved in and take the absolute value. Do the same for all other $X_{*,f}$. Normalize by the sum of all term deviations $\sum_{f} X_{*,f}$. Finally, take the mean across all training records.

After training the FM model on all academic terms, we average these importance metrics across terms, weighting by the number of records predicted in each term. By applying this metric, we found that the student bias, course bias, and instructor bias were most informative, followed by the course discipline, major, student race, and instructor rank. The cohort, instructor class and tenure, student term, transfer indicator, and sex were found to be less important but still informative. Training the FM with only these features greatly improves the results when compared with training on all features. RMSE dropped from $1.0587$ to $0.7832$ – an improvement of $26\%$. This demonstrates that using MADImp for FM feature selection can greatly improve results and significantly reduce the amount of effort required when searching for the most suitable features. For the other models, we relied on their inherent feature selection capabilities.
4.2.2 Grade Prediction Results

The prediction results are broken down by non-cold-start vs. cold-start in Tables 4.1. We first discuss results from methods not incorporating content features. On non-cold-start records, we see the UR and GM methods perform worst, as expected. It is somewhat surprising to see SVD perform worse than the MoM method. This is likely due to overfitting, which is a common issue with SVD [61]. While the post-processing is able to improve the SVD results, they are not nearly as good as those produced by the MoM method. This indicates the per-student and per-course biases are quite informative. The FM model outperforms all the others by a wide margin, indicating 2-way feature interactions play an important role in predicting performance. This also indicates that proper regularization noticeably improves the generalization of learned patterns for prediction in future terms.

All methods except the MoM method actually show improved results when predicting on cold-start records. Since the GM method improves, we can conclude the grades are simply less spread among cold-start dyads, making the task simpler than is usually the case. This is likely an artifact of the large number of transfer credits. These are all cold-start dyads since transfer students are all previously unseen in the dataset. Since only passing grades can transfer from other universities, these transfer credits shift the grade distribution upwards and make it more centered around the mean. The MoM method goes from being the best to being the worst. This indicates that relying on only one or two of the bias terms leads to over-confidence in predictions. In this setting, the FM model is relying on essentially the same information as the MoM method, but it is able to avoid overconfident learning patterns through the use of Bayesian complexity control (integrating out the regularization hyperparameters) for proper regularization.

We next discuss results for prediction with content features included (see Table 4.1). The FM model continues to produce the lowest-error predictions. The next best method is PLMR, followed by the RF. For non-cold start dyads, all of these methods outperform MoM, but only the FM with content features outperforms the FM without. The same thing cannot be said for the FM on cold-start records. In this setting, the RF model is the clear
Table 4.1: Non-Cold Start vs. Cold-Start Prediction RMSE

<table>
<thead>
<tr>
<th>Method</th>
<th>Without Content Features</th>
<th>With Content Features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NCS RMSE MAE</td>
<td>CS RMSE MAE</td>
</tr>
<tr>
<td>FM</td>
<td>0.7792 0.56 ± 0.54</td>
<td>0.7980 0.61 ± 0.51</td>
</tr>
<tr>
<td>MoM</td>
<td>0.8643 0.64 ± 0.58</td>
<td>1.4788 1.30 ± 0.70</td>
</tr>
<tr>
<td>SVD kNN</td>
<td>0.9249 0.68 ± 0.62</td>
<td>0.8123 0.67 ± 0.46</td>
</tr>
<tr>
<td>SVD</td>
<td>0.9370 0.69 ± 0.63</td>
<td>0.8095 0.67 ± 0.46</td>
</tr>
<tr>
<td>GM</td>
<td>0.9448 0.72 ± 0.61</td>
<td>0.8144 0.68 ± 0.45</td>
</tr>
<tr>
<td>UR</td>
<td>1.8667 1.54 ± 1.06</td>
<td>1.8977 1.57 ± 1.07</td>
</tr>
<tr>
<td>FM</td>
<td>0.7423 0.52 ± 0.53</td>
<td>0.8111 0.61 ± 0.53</td>
</tr>
<tr>
<td>PLMR</td>
<td>0.7886 0.57 ± 0.55</td>
<td>1.0207 0.75 ± 0.70</td>
</tr>
<tr>
<td>RF</td>
<td>0.7936 0.58 ± 0.54</td>
<td>0.7475 0.59 ± 0.46</td>
</tr>
<tr>
<td>kNN</td>
<td>0.8061 0.59 ± 0.55</td>
<td>0.8101 0.65 ± 0.49</td>
</tr>
<tr>
<td>SGD</td>
<td>0.8207 0.60 ± 0.56</td>
<td>1.0297 0.75 ± 0.71</td>
</tr>
</tbody>
</table>

The FM model performs just slightly worse than the kNN regressor, and the SGD model performs worst.

Of these models, the SGD regressor is the only one limited to a linear fit, since it uses only 1-way interaction terms. Its relatively poor performance indicates there are definite non-linear trends correlated with performance outcomes. The kNN model performs slightly worse than the RF model on non-cold-start records but does not degrade significantly on cold-start records. This shows that kNN is able to capture somewhat more general trends, which is expected of a clustering technique that infers patterns from shared membership in a subpopulation. It may be that kNN fails to capture the trends present in a large set of historical data because it breaks the data up into smaller subsets and therefore loses out on more general trends that emerge over time. It may also be the case that it fails to identify patterns in ordinal data that would be further delineated by examining a larger population.

It is particularly interesting to notice that the FM with content features performs worse than the FM without on cold-start prediction. This indicates the FM model is learning patterns from prior terms that no longer hold in the current (prediction) term. It is able
Table 4.2: Native vs. Transfer Prediction RMSE

<table>
<thead>
<tr>
<th>Method</th>
<th>Native RMSE</th>
<th>Native MAE</th>
<th>Transfer RMSE</th>
<th>Transfer MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Content Features</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FM</td>
<td>0.8054 ± 0.56</td>
<td>0.58 ± 0.57</td>
<td>0.7805 ± 0.48</td>
<td>0.62 ± 0.48</td>
</tr>
<tr>
<td>SVD-kNN</td>
<td>0.9343 ± 0.63</td>
<td>0.69 ± 0.63</td>
<td>0.8345 ± 0.50</td>
<td>0.67 ± 0.50</td>
</tr>
<tr>
<td>SVD</td>
<td>0.9519 ± 0.65</td>
<td>0.70 ± 0.65</td>
<td>0.8325 ± 0.49</td>
<td>0.67 ± 0.49</td>
</tr>
<tr>
<td>GM</td>
<td>0.9576 ± 0.63</td>
<td>0.72 ± 0.63</td>
<td>0.8391 ± 0.49</td>
<td>0.68 ± 0.49</td>
</tr>
<tr>
<td>MoM</td>
<td>1.0076 ± 0.65</td>
<td>0.77 ± 0.65</td>
<td>1.3064 ± 0.74</td>
<td>1.08 ± 0.74</td>
</tr>
<tr>
<td>UR</td>
<td>1.8679 ± 1.06</td>
<td>1.54 ± 1.06</td>
<td>1.8999 ± 1.07</td>
<td>1.56 ± 1.07</td>
</tr>
<tr>
<td>With Content Features</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FM</td>
<td>0.7879 ± 0.55</td>
<td>0.55 ± 0.57</td>
<td>0.7807 ± 0.52</td>
<td>0.58 ± 0.52</td>
</tr>
<tr>
<td>RF</td>
<td>0.8062 ± 0.55</td>
<td>0.59 ± 0.55</td>
<td>0.7515 ± 0.47</td>
<td>0.59 ± 0.47</td>
</tr>
<tr>
<td>kNN</td>
<td>0.8346 ± 0.57</td>
<td>0.61 ± 0.57</td>
<td>0.7936 ± 0.49</td>
<td>0.62 ± 0.49</td>
</tr>
<tr>
<td>PLMR</td>
<td>0.8860 ± 0.64</td>
<td>0.62 ± 0.64</td>
<td>0.9258 ± 0.63</td>
<td>0.68 ± 0.63</td>
</tr>
<tr>
<td>SGD</td>
<td>0.9137 ± 0.66</td>
<td>0.65 ± 0.66</td>
<td>0.9402 ± 0.69</td>
<td>0.69 ± 0.64</td>
</tr>
</tbody>
</table>

to capture 2-way interactions between features while other methods are not. So these patterns seem to be shifting significantly as new students enroll. This problem, where the test distribution differs significantly from the training distribution, is commonly known as covariate shift [68]. Uncovering inability to deal with covariate shift as a limiting factor in one of our best methods allows us to target future research at overcoming this problem. We discuss these implications in more detail in Section 6.

Table 4.2 provides an alternative breakdown of the prediction results, comparing native students to transfer students. We observe significantly better prediction results for transfer students than for native students and note the gap is narrowed when incorporating content features. This observation is another reflection of the reduced spread of the transfer grade distribution as compared to native grades. Notice that the best performing methods closely align with the non-cold-start vs. cold-start results from Table 4.1: FM performs best on native students while RF performs best on transfer students. This largely reflects the proportion of cold-start dyads from each sample: only 7.37% of native dyads are cold-start while 42.62% of transfer dyads are cold-start. The greater diversity of grades among the
Table 4.3: Cold-start vs. Non-Cold-start Error

<table>
<thead>
<tr>
<th>Group</th>
<th>Dyad %</th>
<th>Method</th>
<th>RMSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCS</td>
<td>48.60</td>
<td>FM</td>
<td>0.7423</td>
<td>0.5187 ± 0.5310</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PLMR</td>
<td>0.7890</td>
<td>0.5635 ± 0.5522</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RF</td>
<td>0.7936</td>
<td>0.5837 ± 0.5377</td>
</tr>
<tr>
<td>CSS</td>
<td>42.31</td>
<td>RF</td>
<td>0.7381</td>
<td>0.5867 ± 0.4478</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FM</td>
<td>0.8112</td>
<td>0.6114 ± 0.5331</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PLMR</td>
<td>0.9917</td>
<td>0.7321 ± 0.6689</td>
</tr>
<tr>
<td>CSC</td>
<td>1.75</td>
<td>FM</td>
<td>0.7456</td>
<td>0.5293 ± 0.5252</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RF</td>
<td>0.7776</td>
<td>0.5695 ± 0.5295</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PLMR</td>
<td>1.1771</td>
<td>0.7489 ± 0.9081</td>
</tr>
<tr>
<td>CSB</td>
<td>4.55</td>
<td>RF</td>
<td>0.8203</td>
<td>0.6603 ± 0.4867</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FM</td>
<td>0.8337</td>
<td>0.6614 ± 0.5075</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PLMR</td>
<td>1.2060</td>
<td>0.8829 ± 0.8215</td>
</tr>
</tbody>
</table>

CS = cold-start, NCS = non-CS, CSS = CS student-only, CSC = CS course-only, CSB = CS both

native students makes the content features much more important for accurate predictions.

Table 4.3 lays out the results for the top three methods (FM, PLMR, and RF) with a more detailed cold-start vs. non-cold-start error breakdown. The FM model is outperformed when student information is absent; in these cases, the RF model performs best. PLMR only performs well on non-cold-start records. These results indicate that we can improve our next-term grade prediction system by swapping out RF for FM whenever we are lacking prior student information. Doing so gives an overall RMSE of 0.7443 compared to 0.7709 for FM and 0.7775 for RF. This demonstrates that such a hybrid is a viable solution to overcoming the cold-start problems suffered by FMs.

4.2.3 Feature Importance

It is useful to have some notion of which features are most informative for next-term grade prediction. This understanding can eventually help us to uncover the relationships between student performance and the various predictor variables available to us. These features
differ between the FM, RF, and PLMR models. While FMs can leverage sparse categorical features effectively via matrix factorization, decision trees are typically unable to discover useful patterns in such data. The PLMR model can leverage 1-way interactions from categorical features, but they must rely on real-valued features to learn useful trends in the absence of the 2-way interactions of the FM model.

We start by examining the importance of features in the FM model. Figure 4.1 shows the importance of features broken down by 1-way and 2-way interactions. The top 3 features make up 97.3% of the importance. These are the student bias (sid), course bias (cid), and instructor bias (iid). Roughly two thirds of the importance comes from 2-way interactions, and the other third comes from the 1-way interactions. Of the non-bias terms, the course discipline (cdisc) and major are the only features which have any notable importance, accounting for 1.1% and 0.8%. This indicates that these high-level interactions can account
The PLMR model does not learn 2-way interactions between features, so there is much more variety in the importance metrics. Figure 4.2 shows the evolution of feature importance across terms, excluding the summer terms (which have very few records). The individual lines represent the percent of the overall importance accounted for by a particular feature in a particular term. Each segment/chunk in a line represents 2% of the overall importance. The X-axis represents the terms, moving from Fall 2009 to Spring 2014.

We see a clear shift from the first term to the second and less drastic shifts moving forward from there. By the last term, the instructor bias, classification, rank, and tenure status, the number of terms a student has been enrolled, and the course discipline emerge as the most important features. We see that last-term GPA features gain in importance as more data is acquired. In contrast, the number of students enrolled in a course during the current term and in prior terms both decline in importance.

Figure 4.3 shows the overall PLMR feature importances for the transfer vs. the non-transfer data. These were obtained by training separate models on the transfer data and the
non-transfer data. The most noticeable difference is the massive importance of the course bias (accounting for nearly 60% of overall bias) in the non-transfer data compared to the complete lack of course bias importance in the transfer data. This is most likely due to the mapping from transfer credits to their GMU equivalents. A variety of courses from many different universities can be mapped to the same GMU course. So the mapping process dilutes the usefulness of this feature. This also explains the lack of importance of the course discipline in the transfer data. For the transfer population, the lack of useful course bias terms causes the PLMR model to learn larger weights for the other features. Discounting this general trend, there are still marked differences in importance for the number of terms a student has been enrolled, instructor classification, and the student and instructor bias terms. Finally, we note that the importance of the iid feature reflects the institution id in all dyads obtained from transfer records. Since this is one of the more useful features
Table 4.4: RF: Top 10 Features by Gini Importance (GI)

<table>
<thead>
<tr>
<th>Feature</th>
<th>GI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lterm_gpa</td>
<td>0.4164</td>
<td>Student’s last-term GPA</td>
</tr>
<tr>
<td>lterm_cum_cgpa</td>
<td>0.2484</td>
<td>Course’s last-term cumulative GPA</td>
</tr>
<tr>
<td>lterm_cum_gpa</td>
<td>0.0645</td>
<td>Student’s last-term cumulative GPA</td>
</tr>
<tr>
<td>prior_gpa</td>
<td>0.0539</td>
<td>Student’s high school or transfer GPA</td>
</tr>
<tr>
<td>lterm_cgpa</td>
<td>0.0349</td>
<td>Course’s last-term GPA</td>
</tr>
<tr>
<td>total_chrs</td>
<td>0.0197</td>
<td>Credit hours attempted by student</td>
</tr>
<tr>
<td>term_chrs</td>
<td>0.0175</td>
<td>Credit hours student is taking this term</td>
</tr>
<tr>
<td>total_enrolled</td>
<td>0.0170</td>
<td>Total historical course enrollment</td>
</tr>
<tr>
<td>zip</td>
<td>0.0153</td>
<td>Student zip code</td>
</tr>
<tr>
<td>iclass</td>
<td>0.0147</td>
<td>Instructor classification</td>
</tr>
</tbody>
</table>

learned for the transfer performance prediction, it seems substitution of instructor id with institution id is a reasonable preprocessing step when predicting performance in these cases.

The RF model performs binary splits on the input data using one particular feature at a time. The Gini Coefficient is used to determine the most informative splits at each level of the tree. Gini Importance (GI) is computed as the normalized total reduction in mean squared error (MSE) brought by each feature over all splits. Due to the normalization, this can also be thought of as the percent of the MSE reduction achieved. We compute GI for each term, including predictions for cold-start records, then average them to get the final GI proportions. The top 10 most important features of the RF in order of highest to lowest GI are shown in Table 4.4.

For the RF model, the most informative are the grade, credit hour, and course enrollment features. Together, these features characterize how competent a student generally is and how difficult and well-established a course generally is. The prior GPA gives some notion of historical competency, but this weighs in less heavily than more recent college grades. The zip code is the one demographic feature that shows up here, indicating that demographics are important when no student-specific bias terms can be learned. The instructor classification shows up again here as well.

It is particularly interesting to note that the instructor features were important for all
three methods. This indicates that the particular instructor and his or her rank, classification, and tenure status have detectable effects on student grades in particular courses. To our knowledge, no previous studies have explored the effect of instructor biases or interactions for grade prediction. Having clearly characterized their importance here, we hope further research in this field can capitalize on the information present in such features to improve educational data mining applications.

4.2.4 Error Analysis

Fig. 4.4a lays out the distribution of error for the FM model in terms of RMSE in a cohort by term number matrix for the non-transfer students. The cohort is the term the student was admitted to the university, while the term number is the objective chronological ordering of the academic terms in the dataset. Each cell represents the aggregate RMSE for all grade predictions for the students admitted in a particular cohort taking courses in a particular academic term. The bar chart on top of the heatmap shows the per-term RMSE, and the bar chart on the right shows the per-cohort RMSE. These errors are from FM with the best settings using content features and predicting for all records, including cold-start. Note that we have excluded Summer terms, which account for only 2.67% of the predictions. This clarifies the trends observed. Also note that Spring cohorts account for only 1.10% of the total predictions after removing summer terms. Hence they have much more variability than the Fall terms. Fig. 4.4b shows a heatmap of the dyad counts for each of the cohort/term cells.

Looking down the diagonal, we see that predictions for each cohort for the first term are generally poorer than predictions for subsequent terms. This reflects the increased difficulty of cold-start predictions, since these are all cold-start students and may also be cold-start courses (new courses). Following each row from left to right, we expect the predictions to decrease in error as we accumulate more information about the students in this cohort. With the exception of the Fall of 2013 and Spring of 2014, we do see this trend. So in general, our predictions improve as we accumulate more historical grade data.
We hypothesize that the degradation of performance moving into the last two terms is indicative of covariate shift in the dataset. The first large cohort in our dataset enrolled in Fall 2009. So those students are beginning to graduate in significant numbers in 2013. Nearly 1000 students graduated in the Spring of 2013, followed by another 500 or so in the summer and Fall. Then another approximately 1200 students graduate in the Spring of 2014. These transitions represent a significant shift in the characteristics of the dataset and also remove from the dataset our most well-known students. These effects taken together seem to largely explain the performance degradation moving into these terms and provide further evidence for covariate shift in such university data.
(a) Error breakdown for FM predictions, excluding summer terms.

(b) Dyad counts for each cell in error breakdown.

Figure 4.4: Cohort by term number error visualization for non-transfer students.
Chapter 5: Profiling Mixtures of Linear Regressions: A Novel Probabilistic Method

In the absence of specific information about entities in a system, it is often useful to profile them into subpopulations. Only once we make observations about individuals can we begin to differentiate on trends specific to them. These ideas are particularly pertinent to the domain of secondary education. When students enroll, universities have little information to differentiate them. Yet advisors must make recommendations that will help students succeed. Initially, seasoned advisors draw on experience to profile new students based on similarities to past advisees. After the first few semesters, the advisor will get to know each student and observe performance measurements from courses taken. This student-specific information will allow more personalized recommendations.

We envision an automated advising assistance system which will help students and advisors more effectively use both profiling and personalization to hone in on the most suitable programs of study. We model the profiling aspect using a mixture of Bayesian linear regressions and personalize using bias terms that combine additively with the mixture. We predict performance for new students with a generic mixture of past performance trends. Once we gather more information about a student, personalized trends are captured by the bias terms and membership weights. We can also add bias terms for courses and instructors to capture general course and instructor difficulty.

We derive an efficient collapsed Gibbs sampling algorithm for full posterior inference and demonstrate competitive performance on real-world and synthetic data. Finally, we demonstrate how the interpretability of PMixoR can be used to generate highly explainable predictions. This characteristic is critical for gaining trust from users and makes our method very promising for both recommendation and data exploration.
5.1 Solution Modeling

We consider a dataset consisting of polyadic relationships between entities \( e = 1, \ldots, E \) interacting with each other. For each type of entity \( e \), we have distinct counts \( n_1, \ldots, n_e \). Each of the \( N \) interactions generates some features \( f = 1, \ldots, F \) and produces an output, which becomes our regression target. For interaction \( n \), we store the features in \( X_n \), the output in \( y_n \), and the list of entities in \( I_n \). We consider the first entity \( I_{n,1} \) to be the primary for the purpose of learning entity-specific membership weights and the others to be non-primary entities. For convenience, we define a matrix \( D_n = (I_n, X_n, y_n) \), whose \( n^{th} \) row contains the entity ids, features, and target.

We model this data through the following generative process. Each primary entity belongs to one of \( k = 1, \ldots, K \) latent profiles; for the \( n^{th} \) interaction, \( z_n = k \) indicates entity \( I_{n,1} \) belongs to profile \( k \). The \( k^{th} \) profile consists of a Bayesian linear regression with regression coefficients \( W_k \) and variance \( \sigma_k^2 \). Each entity, including non-primary entities, has an associated bias term \( b^{(e)} \), where \( b^{(e)}_{n} \) is the bias term corresponding to the entity with ID \( I_{n,e} \). Each of the entity bias vectors is drawn from a univariate normal distribution. The output for the \( n^{th} \) record is given by:

\[
y_n | X_n, I_n, z_n \sim \mathcal{N} \left( X_n W_k + \sum_{e=1}^{E} b^{(e)}_n, \sigma_k^2 \right)
\]  

(5.1)

Our observations do not include the profiles or the bias terms, so we assume these are latent parameters. We place a Multinomial prior on the membership indicator and a Dirichlet prior on that. We place a Normal-Inverse-Gamma (NIG) prior on each profile with hyperparameters \( \Theta = (\mu_0, V_0, \alpha_0, \beta_0) \). Finally, we place Normal-Inverse-Gamma distributions on each of the \( E \) bias term normal distributions, with hyperparameters \( \Theta_e = (m_e, \kappa_e, \alpha_e, \beta_e) \).
So we have likelihoods:

\[ p(W_k|\mu_0, V_0, \sigma_k^2) = \mathcal{N}_F(\mu_0, V_0 \sigma_k^2) \] (5.2)

\[ p(b^{(e)}|\mu_e, \sigma^2_e) = \mathcal{N}(\mu_e, \sigma^2_e) \] (5.3)

\[ p(\mu_e, \sigma^2_e|\Theta_e) = \text{NIG}(m_e, \kappa_e, \alpha_e, \beta_e) \] (5.4)

Note the NIG prior for the profiles involve a multivariate normal with diagonal covariance, while the NIG for the bias terms is univariate. For convenience, we define the set \( \mathcal{B} = \{b^{(1)}, \ldots, b^{(E)}\} \).

### 5.1.1 Inference

We now develop a collapsed Gibbs sampling algorithm to infer our latent parameters given the data \( D \). Our model is semi-conjugate, so we can perform conjugate updates on each parameter given its Markov Blanket. Without \( z_n \), we have:

\[ y_n|X_n, I_n, W, \sigma, \mathcal{B} \sim \mathcal{N}\left(X_n W^T \pi_k + \sum_{e=1}^{E} b^{(e)} n, \sigma^2_k\right) \] (5.5)

The posteriors resulting from the parameter and hyperparameter conjugate updates are included in Instructor Features. These were derived using [69] and [70] as reference.

Since the parameters are estimated with some error, we must incorporate our uncertainty in those estimates when making predictions. Therefore all predictions must come from the posterior predictive distribution, which is a students-t. Given the posterior profile parameters \( \Theta^* = (\mu_0^*, V_0^*, \alpha_0^*, \beta_0^*) \):

\[ p(y_n|\text{rest}) = \int \mathcal{N}(X_n W_k, \sigma_k^2) \text{NIG}(\Theta^*) \, dW_k \, d\sigma_k^2 = \text{MVSt}_{2\alpha_0^*} \left( X_n \mu_0^*, \frac{\beta_0^*}{\alpha_0^*}(I + X_n V_0^* X_n^T) \right) \] (5.6)
To sample $z_n$, we consider each instance of the primary entity in isolation, treating the others as given. The posterior predictive for an instance belonging to profile $k$ is:

$$p(z_n = k | z_{\setminus n}, X, W, \sigma^2) = H_{n,k} \propto p(z_n = k | z_{\setminus n}, \alpha)p(y_n | X_n, I_n, z_n = k, z_{\setminus n}, W, \sigma^2) \quad (5.7)$$

$$= \frac{N_{k\setminus n} + \alpha/K}{N + \alpha - 1} p(y_n | \text{rest}) \quad (5.8)$$

With PMixoR, we predict student grades using linear relationships between the covariates and the grade. We also obtain a probabilistic clustering of students. Analysis of learned parameters for each profile can yield useful insights about the students. However, we still face one significant problem; the observed grades are in a bounded numerical range. We resolve this problem by sampling the bias terms and $y_n$ from truncated normal distributions, which we denote $N_{[a,b]}$, where $a$ is the lower and $b$ the upper bound.
Algorithm 1 Sample(s=num-samples)

1: Initialize bias terms to 0.
2: Initialize $W[0], \sigma^2[0], z[0]$ using K-means($X$).
3: for $i = 1, \ldots, s$ do
   4: $b_e(i) \sim \mathcal{N}_{a,b}(\mu^e[i-1], \sigma^2_e[i-1]) \quad \forall e, i$
   5: for $n = 1, \ldots, N$ do
      6: Remove $I_{n,1}$ interactions from profile $k$.
      7: Refit posterior predictive for profile $k$.
      8: Compute $H_{n}[i]$ using (5.8).
   9: Assign $I_{n,1}$ interactions to $\max_k(H_{n}[i])$.
10: $W_k[i], \sigma^2_k[i] \sim \text{NIG}(\Theta^*[i-1]) \quad \forall k$

Algorithm 2 Predict(T=trace, b=burnin, t=thin-step)

1: Discard $b$ samples as burnin.
2: Thin the trace by keeping only every $t^{th}$ sample.
3: Compute number samples left: $S = (s - b)/t$.
4: for $s = 1, \ldots, S$ do
   5: $\hat{Y}_s \sim \mathcal{N} \left(X_nW_k[s] + \sum_{e=1}^{E} b_{n,e}[s], \sigma^2_k[s] \right)$
6: $\hat{y} = \text{colsum}(\hat{Y})$.
7: Compute HPD interval from $T$ (see [56]).

5.2 Data Description

The real-world data is the same university dataset used in the recommender system approach experiments. We model the data slightly differently in this section. Our real-world dataset consists of polyadic relationships resulting from students taking courses over a sequence of semesters. Each record is uniquely identified by the student id (sid), course id (cid), and a chronological semester number (term). We can organize these triads into a sparse tensor $Y \in \mathbb{R}^{N \times M \times T}$. $N$ is the number of unique students in the dataset, $M$ the number of unique courses, and $T$ the number of terms. So ordinal entry $\{Y_{n,m,t} \in \mathbb{R} \mid 0 \leq Y_{n,m,t} \leq 4\}$ represents the grade student $n$ earned in course $m$ during semester $t$. We extract $F$ features from the content information and place these into a sparse tensor $X \in \mathbb{R}^{N \times M \times T \times F}$.

We choose eight subsets of this data for our PMixoR experiments. We select three majors from three colleges: (1) computer science (CS) from Engineering, (2) chemistry (CHEM) from Science, and (3) philosophy (PHIL) from Humanities. We first pull out
all students declared in the major, giving three datasets. We next additionally include all students taking courses in the same discipline (e.g. CS students and non-CS students taking CS courses). This gives three more datasets which are supersets of the first three. Next, we combine the first three datasets to produce our seventh. Finally, we pull data from several engineering disciplines to look at a large slice of the engineering students. These include information technology, CS, civil engineering, computer engineering, software engineering, and electrical engineering.

5.2.1 Synthetic Data

For our synthetic experiments, we generate data from the generative process of PMixoR, laid out in Algorithm 3

<table>
<thead>
<tr>
<th>Algorithm 3 PMixoR-Generative-Process($N, F, K, n$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: α = $1^{1 \times K}$.</td>
</tr>
<tr>
<td>2: β₀, α₀ = 1.</td>
</tr>
<tr>
<td>3: µ₀ = $0^{1 \times F}$.</td>
</tr>
<tr>
<td>4: π $\sim$ Dirichlet(α).</td>
</tr>
<tr>
<td>5: z $\sim$ Multinomial(π).</td>
</tr>
<tr>
<td>6: $\sigma^2_k$ = Inverse-Gamma($\alpha_0, \beta_0$) ∀k.</td>
</tr>
<tr>
<td>7: $W_k = \mathcal{N}_F(\mu_0, \sigma^2_k)$ ∀k.</td>
</tr>
<tr>
<td>8: $X = \text{Uniform}(1, 10)^{N \times F}$.</td>
</tr>
<tr>
<td>9: p = 0 \ldots (n_1 - 1).</td>
</tr>
<tr>
<td>10: Assign first $n_1$ entries of $I$ to $p$ to ensure each user has ≥ 1 obs</td>
</tr>
<tr>
<td>11: Fill remaining entries of $I$ by sampling ids uniformly from $p$.</td>
</tr>
<tr>
<td>12: for $e = 1 \ldots E$ do</td>
</tr>
<tr>
<td>13: αₐ, βₐ, κₐ = 1.</td>
</tr>
<tr>
<td>14: $m_e = 0$.</td>
</tr>
<tr>
<td>15: $\mu_e, \sigma^2_e \sim \text{NIG}(m_e, \kappa_e, \alpha_e, \beta_e)$.</td>
</tr>
<tr>
<td>16: for $i = 1 \ldots n_n$ do</td>
</tr>
<tr>
<td>17: $b^{(e)}_i \sim \mathcal{N}(\mu_e, \sigma^2_e)$</td>
</tr>
<tr>
<td>18: for $n = 1 \ldots N$ do</td>
</tr>
<tr>
<td>19: $i = I_{n,1}$.</td>
</tr>
<tr>
<td>20: $k = z_i$.</td>
</tr>
<tr>
<td>21: $y_n \sim \mathcal{N}(X_n W^T_k + \sum_{e=1}^E b^{(e)}_i, \sigma^2_k)$.</td>
</tr>
</tbody>
</table>
5.3 Validation Experiments

We conducted several experiments on synthetic data to validate the model and its implementation.

5.3.1 Recapturing Original Parameters

The first validation experiment attempted to recapture the true parameters used to generate the data. We trained the model on several synthetic datasets of various sizes. We were interested in measuring convergence to the true parameter values. To do this, we keep moving expectations of all PMixoR parameters during Gibbs sampling. Starting from the first sample, we begin calculating Euclidean distance between the moving averages for each parameter and the true values. To calculate Euclidean distance, we concatenate all parameters into a flat vector and take the distance between the vectors. In order to compare convergence across datasets with distributions of differing complexity, we normalize by the number of free parameters. In order to better understand this idea, we next analyze the model complexity of PMixoR.

PMixoR has model complexity linear in the number of profiles, features, and entities. For each profile, we learn a population membership weight, a variance, a vector of $F$ regression coefficients, and a vector of membership weights for each of the $n_1$ primary entities. For each entity, we learn a bias term and a prior mean and variance for that bias term. This gives us an overall free parameter count of $K \ast (2 + n_1 + F) + 3 \sum_{e=1}^{E} n_e$.

We performed experiments with five synthetic datasets generated using increasing numbers of parameters. Hence, each dataset has an underlying distribution with increasing complexity. Table 5.1 lays out the parameters used to generate the five synthetic datasets and the total count of free parameters involved in each. For these experiments, we necessarily set $K$ to be the true $K$. Then we learn using 1000 Gibbs samples and discard the first 400 as burn-in. Since this is a stochastic inference procedure, we repeat it five times in order to obtain more reliable trends for comparison.

By normalizing by the number of free parameters, we capture trends for individual
Table 5.1: Parameters for synthetic dataset distribution recapture experiments.

<table>
<thead>
<tr>
<th>Exp. #</th>
<th>N</th>
<th>F</th>
<th>K</th>
<th>(\mathbf{n})</th>
<th>Free Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>3</td>
<td>2</td>
<td>5,3</td>
<td>54</td>
</tr>
<tr>
<td>2</td>
<td>500</td>
<td>5</td>
<td>3</td>
<td>10,5</td>
<td>96</td>
</tr>
<tr>
<td>3</td>
<td>1000</td>
<td>10</td>
<td>4</td>
<td>20,15</td>
<td>201</td>
</tr>
<tr>
<td>4</td>
<td>2000</td>
<td>12</td>
<td>5</td>
<td>100,20</td>
<td>702</td>
</tr>
<tr>
<td>5</td>
<td>5000</td>
<td>15</td>
<td>6</td>
<td>150,100</td>
<td>1,251</td>
</tr>
</tbody>
</table>

Figure 5.2: Convergence of PmixoR Parameters to True Parameters of Data Distribution.

Parameter convergence, Figure 5.2 shows these results. We can see that as the dataset becomes more complex, PmixoR must be trained for more iterations to converge to the true distribution. However, the convergence rate for each parameter remains nearly linear for each experiment. Given enough iterations, it always converged successfully to approximately the original distribution in these experiments. This shows that PmixoR will successfully recapture the original distribution if \(K\) is chosen properly and the data fits its modeling assumptions. In the next experiment, we look at the sensitivity of PmixoR to selection of the parameter \(K\).
Table 5.2: PMixoR Sensitivity to $K$: True vs. Learned Values

<table>
<thead>
<tr>
<th>True $K$</th>
<th>$K = 2$</th>
<th>$K = 3$</th>
<th>$K = 4$</th>
<th>$K = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

5.3.2 Sensitivity to $K$

The second validation experiment explored the sensitivity to the parameter $K$. For this experiment, we used five datasets generated with values of $K$ ranging from 1 to 5. For each dataset, we set the model parameter $K$ to values ranging from 2 to 5. For each setting, we trained the model and saved the expected value of $\pi$ obtained. We then measured “how many models were used”, $\hat{K}$. We threshold $\pi$ s.t. all values below $\tau$ are set to 0 and all those above are set to 1. $\hat{K}$ is then calculated as the sum of these thresholded values. For each dataset, we used $N = 5000, F = 10, n = (100, 20)$. We set $\tau = 0.05$, which means that any profile to which more than 5 entities are assigned is counted.

The result of this experiment is a $5 \times 5$ matrix. The row number is the true values of $K$ used to generate the data; the column number is the parameter setting of $K$ used to train the model; the cell value is the number of models actually used, $\hat{K}$. This matrix is shown in Table 5.2. We see that the model is generally robust to the parameter setting of $K$. When the true $K$ is 1, PMixoR uses only 1 model, even at settings of $K$ from 2 to 5. For a true $K$ of 2, PMixoR uses 2 models at all settings except $K = 5$, where it uses 3 models. For a true $K$ of 3, it uses as many models as allowed until $K = 3$, then uses at most 3. The pattern for a true $K$ of 4 is similar to that for a true $K$ of 2. So in this experiment, PMixoR never underestimates $K$ but may slightly overestimate $K$.

Theoretically, PMixoR can overestimate or underestimate the true $K$ depending on the nature of the mixture. For instance, if there are many highly similar profiles, PMixoR
is likely to learn a single more dispersed profile from the data that subsumes these. The opposite can also happen. If there is one highly disperse profile, PMixoR may learn a few less dispersed profiles if it is given the flexibility to do so. So given a finite sample from an underlying distribution, we cannot be confident of recovery of the true value of $K$. However, for practical purposes, we can see that setting $K$ to be slightly higher than the expected number of profiles will yield better results than setting it lower.

5.4 Real-World Comparison to State-of-the-Art

We conducted eight large-scale experiment using our real-world data to compare PMixoR with the Personalized Linear Mixture Regression (PLMR) model developed by Elbadrawy et al. [45] and extended by us [18]. We are particularly interested in comparing to PLMR because it is a linear model which has demonstrated state-of-the-art performance on the next-term grade prediction problem, and because PLMR has similar modeling assumptions but is non-probabilistic. We would like to discover if a probabilistic model such as PMixoR can achieve similar or better performance. If so, its superior interpretability would make it a better candidate for certain recommendation tasks.

For context, we also compare to the Factorization Machine (FM) [62], and a Random Forest (RF) [67]. In our prior experiments, we found that these models outperformed PLMR, so we expect they will also outperform PMixoR since it shares many modeling assumptions with PLMR. We use our own Stochastic Gradient Descent (SGD) implementation for PLMR, the Gibbs sampler from the fastFM library [65] for FM, and scikit-learn [66] for the RF model. For each model, we learn the best parameters with a grid search and cross-validation on a hold-out set.

5.4.1 Train/Test Procedure

We define $Y^{(t)}$ recursively as the cumulative grade matrix up to and including term $t$:

1. $Y^{(0)} = Y_{*,*,t,0}$
2. \( Y^{(t)} = \text{join}(Y^{(t-1)}, Y_{s^*t}) \) \( \forall t > 0 \),

where \text{join} is a full outer join on the course id, student id, and term fields that uniquely identify an observation. Conflicts are resolved by keeping the values from the more recent term. So if a student takes a course more than once, we keep the most recent attempt. Stated without the join concepts: we combine all records observed so far into a single matrix and drop duplicates by keeping the cell from the matrix most recently observed.

We define \( X^{(t)} \) to be the cumulative feature vector matrix, recursively accumulated in the same manner. Our goal is to make predictions for the next (unobserved) term \( t \) given data from all terms prior to \( t \). In terms of the cumulative data matrices just defined, we train our models on \( Y^{(t-1)}, X^{(t-1)} \) and make predictions for the missing grades \( Y_{s^*t} \) given data \( X_{s^*t} \). Using this train/test procedure, we train and predict with one model for each of the \( T = 15 \) terms in our dataset. The results presented below are aggregates from all these terms.

Missing values for each real-valued feature were filled in using the medians. This was not done for the FM model, since it handles missing data without loss of performance and performs worse with the median-value imputation. After this step, the real-valued attributes were scaled using Z-score scaling. Finally, the predictions of all methods are post-processed to bound the grade predictions to the range \( \{ \hat{y} \in \mathbb{R} \mid 0 \leq \hat{y} \leq 4 \} \); all predictions above 4 are set to 4 and all those below 0 are set to 0. For each dataset, we learn the best parameters for PMixoR using a grid search and cross-validation on a hold-out set.

### 5.4.2 Experimental Results

We begin by examining the mean absolute error (MAE) of the various clustering entity combinations of PMixoR. We cluster on students (sid), courses (cid), and instructors (iid). For sid and cid, we include student and course bias terms. For iid, we include bias terms for instructors and students – here we found including course bias terms degraded performance. Table 5.3 shows these results. We see that profiling on instructors only produces good predictions on the small PHIL datasets but is worst of the three on all the others. For
"-mcd" stands for "major and course discipline" and includes students of the major and those taking courses in that discipline; the "-m" datasets only include students of the major.

the major-only datasets, profiling on students consistently produces the best results. However, once we include other students taking courses in the discipline, profiling on courses is consistently the best option. The major-only datasets contain less students than courses, while the “mcd” datasets are the opposite. This seems to indicate we are able to learn more useful profiles when we have more observations for each entity instance.

Next we keep the best result from the three PMixoR clusterings for each dataset and compare to the state-of-the-art models. Table 5.4 shows these results. As expected from our prior work [19], either the RF or FM model perform best on every dataset. Since we are primarily interested in determining if PMixoR is able to outperform PLMR, our performance compared to PLMR is of greater interest here. Even so, we did some high-level analysis of the performance differences.

Since the RF model performs best, we looked at several views of the predictions made by RF and PMixoR. The most interesting overarching trend we found is illustrated in Figure 5.3. PMixoR is generally more pessimistic than RF; it predicts lower on average than RF. Since most of the dyads have higher grades, this provides a high-level explanation for the performance differences. Even though PMixoR actually tends to predict closer to the true grade on lower grades, it loses out in aggregate performance due to the skew towards
### Table 5.4: Mean Absolute Error (MAE) of Each Model on Each Dataset

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PMixoR</th>
<th>PLMR</th>
<th>FM</th>
<th>RF</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>phil-m</td>
<td>0.93 ± 0.63</td>
<td>1.11 ± 0.77</td>
<td>0.95 ± 0.99</td>
<td>0.83 ± 0.69</td>
<td>275</td>
</tr>
<tr>
<td>phil-mcd</td>
<td>0.81 ± 0.50</td>
<td>0.99 ± 0.71</td>
<td>0.81 ± 0.93</td>
<td>0.64 ± 0.59</td>
<td>3,837</td>
</tr>
<tr>
<td>cs-m</td>
<td>0.77 ± 0.55</td>
<td>0.74 ± 0.63</td>
<td>0.70 ± 0.63</td>
<td>0.69 ± 0.61</td>
<td>8,199</td>
</tr>
<tr>
<td>cs-mcd</td>
<td>0.76 ± 0.54</td>
<td>0.79 ± 0.68</td>
<td>0.75 ± 0.67</td>
<td>0.73 ± 0.64</td>
<td>11,574</td>
</tr>
<tr>
<td>chem-m</td>
<td>0.70 ± 0.49</td>
<td>0.68 ± 0.55</td>
<td>0.64 ± 0.59</td>
<td>0.62 ± 0.54</td>
<td>4,603</td>
</tr>
<tr>
<td>chem-mcd</td>
<td>0.69 ± 0.55</td>
<td>0.76 ± 0.61</td>
<td>0.64 ± 0.59</td>
<td>0.67 ± 0.57</td>
<td>10,989</td>
</tr>
<tr>
<td>combined</td>
<td>0.73 ± 0.57</td>
<td>0.71 ± 0.61</td>
<td>0.67 ± 0.61</td>
<td>0.66 ± 0.59</td>
<td>13,077</td>
</tr>
<tr>
<td>engr</td>
<td>0.75 ± 0.50</td>
<td>0.76 ± 0.70</td>
<td>0.70 ± 0.66</td>
<td>0.69 ± 0.60</td>
<td>40,975</td>
</tr>
</tbody>
</table>

"-mcd" stands for “major and course discipline” and includes students of the major and those taking courses in that discipline; the “-m” datasets only include students of the major.

Having gained some insight into the performance differences between PMixoR and the highest performing model – the RF – we next look more closely at how it compares to PLMR. PMixoR outperforms PLMR on the PHIL datasets, which are smallest. It also outperforms PLMR on the datasets involving students from both the major and the course discipline (the “-mcd” datasets). It is outperformed on the major-only datasets, with the exception of PHIL. Both models perform similarly for the combined and engineering datasets (the last two in the table). The most striking trend arising from these results is that PMixoR appears to make better predictions when few observations are available, as is the case on the small PHIL datasets. This could be due to its superior regularization mechanism. While PLMR must rely on a sub-optimal grid search, PMixoR is able to regularize directly from the data using Bayesian complexity control. This could also be due to PLMR’s use of a global bias term. In prior experiments, we found that including the global bias improves results, but this was on much larger datasets [19]. It is possible predicting on this global performance pattern is over-confident on smaller datasets. Since both models use similar modeling assumptions and perform similarly, it seems likely the data aligns with these modeling assumptions to some extent.
These results are from the CS major and course discipline dataset. We chose it because the performance discrepancy between RF and PMixoR was greatest on this dataset. We introduced some jitter into the actual grades in order to better visualize the densely packed predicted grade points. The actual grade points are categorical along the letter grade discretizations.

Figure 5.3: Predicted vs. Actual Grades for RF and PMixoR

5.5 Detailed Comparison to PLMR

5.5.1 Comparison on Real-World Data

The results from the previous section seem to indicate that PMixoR outperforms PLMR on smaller datasets and roughly matches its performance on larger datasets. In this section we seek to better understand the comparative performance of PMixoR relative to PLMR. For this purpose, we perform a more detailed analysis of the results on the real-world data and perform an additional experiment on synthetic data. For the in-depth analysis of the real-world data results, we focus on the engineering dataset, since it is the largest.

Table 5.5 shows term-by-term results of PMixoR and the results of PLMR and Figure 5.4 shows the same visually. As in Table 5.4, we used the best clustering decision for PMixoR – clustering on courses. Term 0 is left out of these results since it is too small and noisy. We see that both models perform quite similarly overall, but PMixoR shows slightly better performance in the first few and last few terms. Its predictions also seem to be more consistent across the dyads, as evidenced by the consistently reduced spread of error. These
Table 5.5: PMixoR vs. PLMR MAE By Term on Engineering Dataset

<table>
<thead>
<tr>
<th>Term</th>
<th>PLMR</th>
<th>PMixoR-cid</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.42 ± 0.87</td>
<td>2.11 ± 0.67</td>
<td>1,404</td>
</tr>
<tr>
<td>2</td>
<td>0.79 ± 0.65</td>
<td>0.76 ± 0.60</td>
<td>1,515</td>
</tr>
<tr>
<td>3</td>
<td>0.62 ± 0.43</td>
<td>0.58 ± 0.38</td>
<td>86</td>
</tr>
<tr>
<td>4</td>
<td>0.72 ± 0.58</td>
<td>0.72 ± 0.63</td>
<td>2,684</td>
</tr>
<tr>
<td>5</td>
<td>0.72 ± 0.58</td>
<td>0.72 ± 0.55</td>
<td>2,654</td>
</tr>
<tr>
<td>6</td>
<td>0.66 ± 0.46</td>
<td>0.66 ± 0.45</td>
<td>179</td>
</tr>
<tr>
<td>7</td>
<td>0.71 ± 0.55</td>
<td>0.71 ± 0.44</td>
<td>3,880</td>
</tr>
<tr>
<td>8</td>
<td>0.66 ± 0.56</td>
<td>0.66 ± 0.51</td>
<td>3,803</td>
</tr>
<tr>
<td>9</td>
<td>0.67 ± 0.52</td>
<td>0.66 ± 0.49</td>
<td>315</td>
</tr>
<tr>
<td>10</td>
<td>0.66 ± 0.57</td>
<td>0.66 ± 0.46</td>
<td>5,346</td>
</tr>
<tr>
<td>11</td>
<td>0.66 ± 0.59</td>
<td>0.66 ± 0.47</td>
<td>5,179</td>
</tr>
<tr>
<td>12</td>
<td>0.68 ± 0.55</td>
<td>0.68 ± 0.54</td>
<td>501</td>
</tr>
<tr>
<td>13</td>
<td>0.72 ± 0.66</td>
<td>0.72 ± 0.56</td>
<td>6,822</td>
</tr>
<tr>
<td>14</td>
<td>0.77 ± 0.71</td>
<td>0.76 ± 0.61</td>
<td>6,607</td>
</tr>
<tr>
<td>all</td>
<td>0.76 ± 0.70</td>
<td>0.75 ± 0.50</td>
<td>40,975</td>
</tr>
</tbody>
</table>

observations provide further evidence that PMixoR’s performance advantage comes from its superior regularization. When both models are working with less data in the earlier terms, PMixoR is able to make noticeably better predictions.

5.5.2 Comparison on Synthetic Data

To better understand the performance of each model as the number of observations increase, we next compare performance trends of both models with increasing training set sizes. To do this, we perform the experiment on synthetic data. With this dataset, we can freely vary the training set size since it does not have the same temporal data leakage concerns as the real-world data. In particular, we do not need to keep the data split up by terms. We created the synthetic dataset for this experiment using the generative process of PMixoR. Theoretically, PMixoR should be able to capture the underlying distribution of the data exactly as it observes more samples generated from it. Since PLMR is unconstrained by an underlying probabilistic model, it should also be able to approximate the underlying distribution more and more accurately as it observes more samples from it. So using the
PMixoR generative process puts both models on a level playing field. We generated 5000 samples with 10 features and three true underlying profiles. We set $K = 3$ for both models and learned suitable PLMR parameters using grid search.

Table 5.6 and Figure 5.5 show the results of this experiment. These results show that both models manage to perform well on this dataset, but PMixoR exclusively outperforms PLMR. Initially, the performance gap is wider. PLMR quickly improves from 10% to 25% observed. From 25% to 45% observed, both models improve at roughly the same linear rate. From this point on, PLMR starts to level out in its increases while PMixoR continues to improve at a roughly linear rate. So we see that on data that fits its assumptions, PMixoR is able to outperform PLMR and continue to improve as it makes additional observations. The results of this experiment offer further evidence for our theory that PMixoR is superior due to its superior regularization when the training set size is small. It also demonstrates that this regularization improvement may help it adjust better as more data becomes available.

### 5.6 Interpretability Analysis

Having demonstrated competitive performance compared to state-of-the-art methods, we next examine a few figures produced from the PMixoR output to demonstrate its interpretability. The visualization in this section show the power of using a probabilistic model...
Table 5.6: MAE of PLMR and PMixoR on Synthetic Data

<table>
<thead>
<tr>
<th>Train %</th>
<th>PLMR</th>
<th>PMixoR</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10.04 ± 8.67</td>
<td>6.13 ± 5.93</td>
<td>4,453</td>
</tr>
<tr>
<td>15</td>
<td>9.05 ± 8.76</td>
<td>5.85 ± 5.74</td>
<td>4,249</td>
</tr>
<tr>
<td>20</td>
<td>6.40 ± 6.37</td>
<td>5.24 ± 5.11</td>
<td>3,999</td>
</tr>
<tr>
<td>25</td>
<td>5.62 ± 5.33</td>
<td>4.71 ± 4.38</td>
<td>3,749</td>
</tr>
<tr>
<td>30</td>
<td>5.09 ± 4.89</td>
<td>4.32 ± 4.02</td>
<td>3,499</td>
</tr>
<tr>
<td>35</td>
<td>4.53 ± 4.15</td>
<td>3.88 ± 3.72</td>
<td>3,249</td>
</tr>
<tr>
<td>40</td>
<td>4.23 ± 3.83</td>
<td>3.49 ± 3.31</td>
<td>2,999</td>
</tr>
<tr>
<td>45</td>
<td>3.89 ± 3.45</td>
<td>3.20 ± 3.09</td>
<td>2,749</td>
</tr>
<tr>
<td>50</td>
<td>3.87 ± 3.54</td>
<td>2.71 ± 2.19</td>
<td>2,499</td>
</tr>
<tr>
<td>55</td>
<td>3.59 ± 3.27</td>
<td>2.51 ± 1.97</td>
<td>2,249</td>
</tr>
<tr>
<td>60</td>
<td>3.28 ± 2.85</td>
<td>2.23 ± 1.73</td>
<td>1,999</td>
</tr>
<tr>
<td>65</td>
<td>3.12 ± 2.74</td>
<td>1.77 ± 1.50</td>
<td>1,749</td>
</tr>
<tr>
<td>70</td>
<td>3.07 ± 2.69</td>
<td>1.25 ± 1.42</td>
<td>1,499</td>
</tr>
</tbody>
</table>

and inferring a full posterior distribution when it comes to explaining results. This interpretability makes PMixoR very promising for applications in automated advising systems.

5.6.1 Actual vs. Predicted with HPD Intervals

We first look at two graphs of actual vs. predicted targets with highest posterior density (HPD) intervals. The first comes from a synthetic dataset and the second from one of our real-world datasets. Figure 5.6 shows the results on the synthetic dataset. Our predictions are shown as blue dots, the HPD as a light blue vertical line through the dot, and the actual target as a green dot. This synthetic data was generated with three profiles, three features, 20 primary entities, and three secondary entities. 400 interactions were observed and 450 Gibbs samples analyzed after burn-in and thinning. Figure 5.7 shows results from the dataset with only CS majors. This dataset had 8,199 dyads. Since it is quite difficult to visualize this many results, we randomly sampled 5% of them. For each graph, we sorted the samples from highest predicted grade to lowest.

From the synthetic data visualization, we can see the majority of the actual targets fall within the 95% HPD interval. In repeated experiments, approximately 95% repeatedly
Figure 5.5: MAE of PLMR and PMixoR on Synthetic Data With Increasing Observations

fell within the intervals. Confirming the model produces this expected behavior provides validation of its reliability on data that meets PMixoR’s modeling assumptions. We can see by examining these intervals that there are two distinct sets of targets. The first includes samples whose target variables are less dispersed with a smaller mean. The other includes samples whose targets have a larger mean and higher variance. There were less samples from the latter cluster, which is reflected in the width of the HPD intervals. PMixoR is more certain about predictions for entities in the cluster with more observations.

The real-world data visualizations are shown in Figure 5.7. Looking closely, we can pick apart the three distinct profiles from the widths of the HPD intervals. The first profile tends to consist mostly of high performers and is the largest. The second is spread across the range of actual grades, with slightly less membership among the highest grades. The last is a small profile that consists mostly of low-performers. We can see this distinctly by the
very wide HPD intervals near the right of the graph. This also reflects a trend we’ve seen in prior experiments; it is harder to characterize performance trends as grades drop and most predictions are above 2.0. Notice that the actual grades are discrete within the 0-4 range. This is an artifact of the letter grade mappings. We experimented with post-processing PMixoR predictions to try to recapture these actual grades. However, doing so yields the same error; it tends to move just as many grades away from the true grade as it moves toward the true grade. In reality, students obtain real-valued grades before that grade is thresholded to a letter grade, so predicting in the real-valued range is a truer reflection of this.

5.6.2 Visualization of Profiles

We next look at a few visualizations of the profiles learned by PMixoR; these provide some qualitative confirmation that the profiles learned are sensible and useful. There are two
Figure 5.7: Highest Posterior Density (HPD) Intervals and Predictions on CS Major-Only

types of profiles we get with the model: population-wide and personalized. The population-wide profiles are those $K$ profiles corresponding to the model’s mixture components. The personalized profiles are learned through the student-specific profile membership weights.

We first look at an interesting characteristic of the population-wide profiles learned on the engineering dataset (engr). We were interested in identifying interesting factors across the profiles. Through visual inspection, we found the patterns between chronological student term number (stern) and performance particularly interesting. For a student who takes no summer classes, these terms correspond to only spring or fall, but for a student who takes classes every summer, we could see as many as 12 “sterms” in four years. Figure 5.8 shows a plot of the coefficients learned for these variables for each of the three profiles. The mean is shown as a point and the 95% confidence interval as a vertical line through the mean point.

We can see three distinct performance trends. In the green profile, students tend to start
Figure 5.8: Linear Effect of Chronological Student Term on Performance for Each Profile

out performing higher than average. Then their performance drops in the Junior year (5th term). From there, they steadily improve until the likely graduation terms (8-10). In the red profile, we see students start out performing worse than average, then their performance rises steadily until the likely graduation terms, dipping slightly near the end. In the blue profile, we see more or less steady performance throughout all terms, with just slightly higher in the first four terms and just slightly lower from the fifth until likely graduation terms. All three profiles tend to show widely varying behavior in the later terms. We have less data for students persisting into these later terms. In most cases, this means students are moving into their sixth and seventh years. These may be part-time students or they may be students who are having difficulty graduating. They could also be students who take classes in many summer terms.

Our hypothesis is that these trends reflect the interplay of two factors: varying extracurricular activities and increasingly difficult coursework. The first four terms are typically those terms where students are taking general education requirements (gen-eds) and easing into their major coursework. The fifth to eighth terms are usually those consisting primarily of more difficult major-specific coursework. Under this hypothesis, the green profile would be students who perform well on gen-eds and experience a bit of a shock while transitioning
to the more difficult major-specific coursework. From there, performance improves. So either these students adjust or we see a survivorship bias where the students who struggled most dropped out. In contrast, we expect students in the red profile have almost the opposite experience. They either underestimate the difficulty of gen-eds or simply become involved in many extra-curriculars and are unable to balance grades with activities effectively. The red profile trend may also be explained by a survivorship bias. Meanwhile, the students in the blue profile maintain steady performance characteristics throughout their college enrollment. On this dataset, the blue profile was the largest, accounting for 73.3% of students, the red was next largest, accounting for 19.2%, and the green was smallest, accounting for the remaining 7.5% of students.

Next we examine a few examples of personalized profiles. Once we have fitted the PMixoR model to a particular dataset, we have $K$ population profiles. When clustering on students, we also have per-student memberships in each of those profiles. We can use these to calculate student-specific profiles. We call these personalized profiles. Algorithm 4 specifies how to compute them, and Figures 5.9 and 5.10 show two examples.

Both personalized profiles have distinct and interesting performance trends. We see that the CS student typically performs best in English and CS classes and worst in Biology classes. From his student bias of roughly 0.4, we see that he performs about a third of a letter grade higher than average. He does particularly well in courses taught by instructors and associate professors, as opposed to full professors. He also seems to perform better in courses taught by tenure-track professors than in courses taught by tenured professors. Finally, we see that his fifth term was characterized by performance uncharacteristically low for him.

While the CS student profile shows higher than average performance, the PHIL profile shows performance more than an entire point lower than average. This student performs best in Philosophy and History courses and worst in Math courses. Interestingly, she performs particularly well in courses taught by tenured professors. She has more distinct temporal performance patterns; her first two terms were characterized by poor performance.
Algorithm 4 Personalized-Profile \((T = \text{trace after burnin and thinning})\)

1: \(\hat{\mu}_k = \frac{1}{|T|} \sum_{s=1}^{|T|} T.W_k[s].\) \hspace{1cm} \triangleright \text{\(T\) contains the Gibbs samples for each parameter.}

2: \(\hat{\Sigma}_k = \frac{1}{|T|} \sum_{s=1}^{|T|} T.\Sigma_k[s].\) \hspace{1cm} \triangleright \text{\(\Sigma_k = V_0^\star \sigma_k^2.\)}

3: \(\hat{\sigma}_k^2 = \text{diag}(\hat{\Sigma}_k).\)

4: \bf{for} \(n = 1, \ldots, N_1\) \bf{do}

5: \(H_{n,k} = \frac{1}{|T|} \sum_{s=1}^{|T|} T.H_{n,k} \forall K.\) \hspace{1cm} \triangleright \text{Personalized memberships.}

6: \(\hat{\mu}_n = \sum_{k=1}^K H_{n,k} \hat{\mu}_k.\) \hspace{1cm} \triangleright \text{Unconditional coefficient means.}

7: \(\hat{\sigma}_n^2 = \sum_{k=1}^K H_{n,k} \hat{\sigma}_k^2 + \sum_{k=1}^K H_{n,k} \hat{\mu}_k^2 - \mu_n^2.\) \hspace{1cm} \triangleright \text{Unconditional coefficient variances.}

while her third and fourth show marked improvement. Finally, we can see that her performance tends to decrease by about 0.2 grade points per course for each additional credit hour she enrolls in.

Some of the information provided by these profiles is more a curiosity than an actionable insight. However, several of these characteristics would provide an advisor with useful information for making recommendations. For instance, we might advise the CS student to take more lab and interactive courses, which are usually taught by instructors. We might also advise him to pick up a minor in technical writing, which would complement his CS major well and fit well with his strengths in English courses. In addition, we might advise him against taking Biology courses or pursuing a minor in one of the natural sciences. If he is required to take additional Biology courses, we might advise him to visit a Biology tutor once or twice a week.

For the Philosophy student, we can see she started out performing poorly and has increased her performance over time. A discussion of how and why this occurred might be useful if she is not aware of this trend. In addition, we might advise her to take less courses in terms when she must take required Math courses. Since she performs better with a lighter class load, taking her Math courses over the summer or moving other courses to summer terms might be a good option for her. From her student bias, we can also see that this student might need special advising or instruction to help her increase performance in her coursework.
Personalized Student Profiles

Figure 5.9: Junior CS Student

Figure 5.10: Junior PHIL Student

ENGL = English, Instr = Instructor, TenTrack = Tenure Track, AssocProf = Associate Professor, FTFaculty = Full Time Faculty, Ten = Tenured, BIOL = Biology, HIST = History, sbias = student bias
5.7 Summary of PMixoR Modeling and Results

We built a fully Bayesian model (PMixoR) that (1) accurately predicts next-term grades given data from previous terms, (2) clusters students to identify common learner and student-specific profiles, and (3) is highly interpretable. We derived an efficient collapsed Gibbs sampling algorithm for inference and used the model for prediction and posterior analysis of a real-world university dataset. We also developed several informative visualizations of the profiles learned by the model and demonstrated how they can provide insight for academic advising. Our results show that developing probabilistic models around existing state-of-the-art techniques and using Bayesian inference can improve prediction performance and interpretability.

In the future, we would like to develop a variational inference technique for faster inference. We would also like to develop additional informative visualizations using the wealth of information provided by the posterior parameter distributions. Finally, we would like to work towards developing a mixture of factorization machines model, whose profiles are Bayesian FM mixture components.
Chapter 6: Discussion

6.1 Predictive Performance and Explainability

We found that a FM-RF hybrid is an effective method of overcoming the cold-start limitations of FMs and is the most effective method in general. However, we also note that more sophisticated methods have been developed to improve performance of MF-based methods on cold-start prediction tasks. One notable example is the attribute-to-latent-feature mapping developed by [71]. Such an approach might produce better results and/or be computationally cheaper than the FM-RF hybrid.

We also uncovered covariate shift as a likely source of error when using FMs for cold-start prediction. When the test distribution differs from the training distribution, the FM model is liable to learn overconfident 2-way interactions that reduce its ability to generalize. The RF model does not seem to suffer from this problem, but it is also clearly unable to capture 2-way interactions. Discovering new ways to handle covariate shift while learning 2-way interactions might be another viable way to overcome cold-start issues in MF-based methods.

We found that PLMR was one of the top 3 performing models in our initial experiments and also one of the most interpretable. Motivated by these findings, we developed a probabilistic model with many of the same modeling assumptions called Profiling Mixtures of Linear Regressions (PMixoR). We found that using Bayesian inference to obtain a full posterior distribution, we were able to obtain superior performance. We believe this is due to the improved regularization capabilities of probabilistic models. Combined with the much-improved interpretability of PMixoR, these results demonstrate that probabilistic models are quite promising for automated advising applications.
6.2 Beneficial Applications

Using our system, we can predict student grades in the next enrollment term much more accurately than a random guessing strategy or the regression models commonly used in the higher education literature would. The main thrust of this thesis is that automating a variety of advising tasks will facilitate superior academic advising and improve the graduation rate. This information can also be used to aid students, educators, and advisors in a variety of ways.

For students, we can incorporate this information into a degree planning system. Such systems already have ways of determining which courses meet which degree requirements. Given several sets of possible course selections for a semester, our system can be used to maximize the expected grade. We could also provide students with a personalized difficulty rating for each course based on their expected grade. This would help students prioritize studies for each course and prepare properly for particularly challenging courses.

For educators, knowledge of which students have the lowest expected grades could provide opportunities to increase detection of at-risk students. While students can seek help and have access to many resources, research shows that at-risk students perform better if instructors are proactive in identifying and reaching out to them. A great deal of research has gone into identifying characteristics of effective interventions based on information provided by learning analytics. The Open Source Analytics Initiative (OSAI) conducted a survey and analysis of this body of research, identifying several key characteristics. Interventions that increase student communication with instructors, connect students to available university services, and provide students with an accurate assessment of how they are currently performing can be effective at increasing both retention and academic performance. Our grade prediction system can be used as a component of an early-warning system that can equip instructors to provide interventions with these crucial characteristics.

The information provided by our system would also be invaluable for advisors. As discussed in the introduction to this thesis, advising has the potential to influence all nine of the key factors that impact student retention. In order to improve student retention,
advisors must perform many tasks, some of them quite complex. In order to achieve the ambitious advising agenda we believe is necessary for an ideal advising program that will improve retention, we maintain it is necessary to adopt effective advising automation systems. The work in this thesis represents a component of such a system.

Anticipating student performance and recommending the best plan of action is a critical component of advisor responsibilities[74]. Any additional information that helps advisors personalize their advice to each student could potentially help thousands of students. For instance, imagine a particular course is known to be a challenging bottleneck course and another course is known to be particularly easy in general. Without additional information, an advisor might always recommend students take these two courses in the same semester. However, some students might have trouble with exactly the type of material taught or exactly the type of teaching style used in the “easy” course. As shown by our results, a system employing models such as the FM-RF hybrid and PMixoR can help advisors determine the achievability of various course selections for both new and returning students. To help convince students, advisors could employ informative visualizations from a probabilistic model like PMixoR to explain both general and personalized trends that form the basis for their recommendations. The combination of accurate models with explainable visualizations is very promising for advising students toward achievable degree plans.
Chapter 7: Conclusions and Future Work

Motivated by the need for institutions to retain students, ensure timely graduation, and ensure students are well-trained and workforce-ready in their field of study, we have leveraged state-of-the-art recommender systems techniques to develop a system for accurate next-term student grade prediction. After experimenting with a wide variety of regression and factorization models, we determined that a hybrid of the Random Forest model and the MF-based Factorization Machine is best-suited to this task. Key to the success of this hybrid model is the application of MADImp to FM feature selection. Using this hybrid, we can predict grades for both new and returning students and for both new and existing courses. While this system significantly outperforms random guess predictions, and noticeably outperforms the other regression models tested, it still has limitations we seek to address in future works. The main limitations are poorer performance for grades at the extremes and difficulty dealing with covariate shift as students graduate and new students enroll.

These methods also produce results that are difficult to explain. To help address this last concern, we developed Profiling Mixtures of Linear Regressions (PMixoR), a fully probabilistic model that builds on modeling ideas from the Personalized Linear Mixture Regression (PLMR) model. We found that using Bayesian inference with this probabilistic model allowed us to improve upon the predictive performance of PLMR while also greatly increasing interpretability. These results demonstrate the promise of Bayesian methods for high-impact recommendations like degree planning, where the people receiving the recommendations are likely to need an explanation in order to feel confident acting on the recommendations.
7.1 Directions for Future Work

Of the methods surveyed, tensor factorization, multi-relational matrix factorization, and custom tree-based methods seem most promising for improving predictive performance. The first two methods are inspired by similar work in the ITS community. Both of these methods hold promise for effectively leveraging the valuable instructor features identified in this study. The idea of using more customized tree-based methods is motivated by the desire for interpretable decision-making rules. Easily understandable decision rules can speed student, educator, and advisor adoption of new practices motivated by our findings. Additional feature engineering and custom methods of constructing and pruning decision trees seem promising areas for further work.

As in other MF applications, we observed the cold-start problem is a limiting factor and complementary methods were required to overcome it. We overcame this problem by combining FMs with RFs to compose a hybrid model. We have also discussed using attribute-to-latent-feature mappings and identifying means of overcoming covariate shift while learning 2-way interactions as two possible augmentations to help FMs overcome the cold-start problem. Further research in this area could greatly benefit many fields where MF-based methods are applicable.

We also identified probabilistic methods and Bayesian inference techniques as promising areas for future work. In particular, the combination of matrix factorization techniques with regression mixtures is a very promising future direction. Combining the multi-way covariate interactions with the interpretability of mixture models may yield a model with performance superior to a mixture of linear models that is more interpretable than a lone factorization model.

The core experiments in this thesis address the problem of next-term student grade prediction. However, a real degree-planning system will need to predict further than the next term. An important area for future work is the prediction for several terms, and eventually for an entire degree plan. When embarking on this path, it will be important to consider the likelihood of degree plan adherence and feedback loops. To truly understand
the utility of these techniques, it will be necessary to deploy this system for live use as a component of a degree planning and/or early-warning system for students, instructors, and advisors. Once live, A/B testing would yield better understanding of its performance, the effect it has on the decision-making of its various users, and its ability to improve student retention and learning success outcomes.

Beyond the grade prediction performance and interpretability concerns, it is also necessary for an effective advising system to balance resource constraints. As discussed in the introduction, it is necessary to recognize that a university brings to bear a collection of resources that are limited and valuable. There are a fixed number of instructors available for teaching, a fixed number of faculty available to run the facilities and support university functions, and a fixed number of physical resources. It is misleading and unhelpful to advise programs to students which the university cannot support. If the instructors aren’t available, the courses aren’t being taught, there is insufficient parking or transportation, or there are no rooms or spaces to teach the courses in, students cannot hope to execute on their degree plans. While some progress has been made in this area, it remains a hard and complex problem. Further work in this direction will certainly yield benefits for many students.

Finally, we conclude with some directions we believe to be very challenging but very interesting. These include student modeling and effective maximization of student return-on-investment. The prediction methods analyzed and developed in this thesis made no attempt to model student knowledge, course topics, or instructor proficiencies. Further, we also made no attempt to connect skills learned or taught to those desired by the economic market place students are graduating into. Academic pursuits are ultimately for a mixture of two objectives: increase earning potential and learn more about topics of interest. A truly intelligent automated advising system will focus directly on these objectives and treat the goals of achievability and satisfiability as intermediate steps towards these objectives. Progress in this pursuit will surely push the boundaries of what is currently possible in machine learning and artificial intelligence more generally.
Appendix A: Feature Descriptions

This section provides detailed descriptions of each feature in the real-world dataset used for experiments in this thesis.

A.1 Student Features

- **sid**: Unique identifier of the student. When used in training data, the student ID is one-hot encoded to learn student bias terms.

- **grdpts**: [0, 4] grade the student has obtained for a particular course.

- **major**: Declared major during current term. Students may change majors, so the same student may take courses with a different declared major at different times.

- **race**: Self-reported race of student; may be unspecified.

- **sex**: Self-reported gender; may be unspecified.

- **age**: Age determined from birth date in admissions records.

- **zip**: Zip code, or postal code for students from outside the US.

- **sat**: 1600-scale SAT score, if available. The SAT is a standardized test commonly used for college admissions in the United States. The reading and math sections are often considered separately from the writing. Both are scored from 200-800, giving a max score of 1600.

- **hs**: High school CEEB code. These are standardized ID numbers given to high schools, colleges, and universities by the Educational Testing Service (ETS). They are used to uniquely identify high schools in our dataset.

- **hsgpa**: High school GPA. For transfer students, this feature contains the GPA from the institution the student is transferring from.
• \textit{lterm\_gpa}: Grade Point Average (GPA) from the previous term.

• \textit{lterm\_cum\_gpa}: Cumulative GPA as of the previous term. The cumulative GPA is an average of course grades weighted by the course credit hours.

• \textit{term\_chrs}: Number of credit hours the student is enrolled in during the current term.

• \textit{total\_chrs}: Number of credit hours student has taken (not passed) up to the previous term.

• \textit{alevel}: Academic level of the student. Obtained by binning total\_chrs: \([0,30)=0, [30,60)=1, [60,90)=2, [90,120]=3, (120+)\)=4. This is a measure of experience, rather than of progress in the program. It reflects total credit hours attempted, not just those from courses passed.

• \textit{sterm}: Chronological numbering of terms relative to this student. The student’s first term is 0, his second is 1, and so on.

\section*{A.2 Course Features}

• \textit{cid}: Unique identifier of the course. When used in training data, the student ID is one-hot encoded to learn course bias terms.

• \textit{edisc}: Course discipline.

• \textit{chrs}: Number of credit hours this course is worth. This is a measure of expected work required by the course and is used to determine the importance of the course in GPA calculations.

• \textit{clevel}: Course level \([1, 7]\).

• \textit{termnum}: Number of the term this course was offered in. These are relative to the dataset only. Term 0 is the first term for which we have data, and they are numbered chronologically onwards from there.
• **num_enrolled**: Number of students enrolled in this course for the current term, across all sections.

• **total_enrolled**: Total number of students enrolled in this course since its first offering, including the current term.

• **lterm_cgpa**: The aggregate [0, 4] GPA of all students who took this course during the previous term.

• **lterm_cum_cgpa**: The aggregate [0, 4] GPA of all students who have ever taken this class up to the previous term.

### A.3 Instructor Features

• ** iid**: Unique identifier of the instructor. When used in training data, the student ID is one-hot encoded to learn instructor bias terms.

• **iclass**: Classification (Adjunct, Full time, Part time, GRA, GTA).

• **irank**: Rank (Instructor, Assistant Professor, Associate Professor, Eminent Scholar, University Professor).

• **itenure**: Tenure status (Term, Tenure-track, Tenured).
Appendix B: Model Parameter Settings

• **Factorization Machine**: $k = \text{rank} = 8$, $i = \text{number of iterations} = 200$, $s = \text{initial standard deviation} = 0.2$

• **Personalized Linear Multiple Regression**: $k = \text{number of models} = 4$, $l_w = \text{regularization on } P \text{ and } W = 0.01$, $l_b = \lambda_B = 0.5$, $l_r = \text{SGD learning rate} = 0.001$

• **Random Forest**: $n = \text{number of trees} = 100$, $m = \text{max depth} = 10$

• **Boosted Decision Trees**: $n = \text{number of trees} = 100$, $m = \text{max depth} = 11$

• **Stochastic Gradient Descent (SGD) Regression**: $l_r = \text{learning rate} = 0.001$, $r = \text{regularization term} = 0.001$, $i = \text{number of iterations} = 15$

• **Ordinary Least Squares (OLS) Regression**: no parameters

• **k-Nearest Neighbors (kNN)**: $k = \text{number of neighbors} = 20$

• **Decision Tree**: $m = \text{max depth} = 4$
Appendix C: MADImp Example

Let us consider a particular example to better understand the idea of Mean Absolute Deviation Importance (MADImp). We proceed with the FM model but note again that MADImp can be used with any generalized linear model (GLM).

Assume we have three features: a user ID, an item ID, and a season (Spring, Summer, Fall, Winter). We one-hot encode all features, so for a particular record we have three features with a value of 1 and all the rest have values of 0. Now the goal is to predict the response a particular user gives to a particular item during a particular season. Let us now fixate upon a single user. This user generally tends to give all items a slightly positive response. The responses are more positive in the Spring and Fall, and less positive in the Summer and Winter. Next we fixate upon a single item. This item usually generates very positive responses, with the positivity spiking in the Summer. The other seasons have no effect on users’ responses to the item. The user has no particular preferences for or against the item. We also observe the overall pattern of responses is slightly positive and the range of ratings is from 0 to 4. Finally, we do not observe any general seasonality effects. We only find seasonal preferences of and for particular users and items.

Given this setting, we can now assign hypothetical values to our parameters. We have 10 users and 20 items, and there are 4 seasons. So after one-hot encoding, we have a total of 34 features. Let our user be feature 1, our item be feature 11, and the seasons are then the following feature numbers: Spring = 31, Summer = 32, Fall = 33, Winter = 34. As is typical with regression models, we set feature 0 to 1 for all records to learn the global intercept term.

- \( w_0 = 0.5 \), a positive global intercept term.
- \( w_1 = 0.5 \), a positive response trend for this user.
- \( w_{11} = 2.0 \), a very positive response trend for this item.
- \( Z_{1,31} = Z_{1,33} = 0.2 \), this user has slightly more positive responses in Spring and Fall.
• $Z_{1,32} = Z_{1,34} = -0.2$, this user has slightly less positive responses in Summer and Winter.

• $Z_{11,32} = 0.2$, this item has slightly more positive responses in the Summer.

With these parameter values, a prediction for this user-item combo in the Summer would be calculated as:

$$\hat{y}(X_d) = w_0 + w_1 + w_{11} + Z_{1,32} + Z_{11,32}$$

$$= 0.5 + 0.5 + 2.0 - 0.2 + 0.2$$

$$= 3.0$$

Recall that we have three features set to 1 for each dyad feature vector $X_d$. So we calculate the importance of these three features using (4.11).
\[ T_d = |w_1| + |w_{11}| + |Z_{1,32}| + |Z_{11,32}| \]

\[ = |0.5| + |2.0| + |-0.2| + |0.2| = 2.9 \]

\[ I(X_{d,1}) = \frac{|w_1| + |Z_{1,32}|/2}{T_d} \]

\[ = \frac{|0.5| + \frac{|-0.2|}{2}}{2.9} = \frac{0.6}{2.9} \approx 0.2069 \]

\[ I(X_{d,11}) = \frac{|w_{11}| + |Z_{11,32}|/2}{T_d} \]

\[ = \frac{|2.0| + \frac{|0.2|}{2}}{2.9} = \frac{2.1}{2.9} \approx 0.7241 \]

\[ I(X_{d,32}) = \frac{|Z_{1,32}|/2 + |Z_{11,32}|/2}{T_d} \]

\[ = \frac{|-0.2|/2 + |0.2|/2}{2.9} = \frac{0.2}{2.9} \approx 0.0690 \]
Appendix D: Conjugate Updates for PMixoR

Hyperparameters

Note that precision is the inverse of variance, and we denote precisions with \( \tau_e \). The posteriors for our parameters and hyperparameters are detailed below.

\[
p(\mu_e, \sigma_e^2 | \Theta^*) = \text{NIG}(\mu_e^*, \kappa_e^*, \alpha_e^*, \beta_e^*) \quad (D.1)
\]

\[
p(W_k, \sigma_k^2) = \mathcal{N}(\mu_0^*, \sigma_k^2 V_0^*) \text{IG}(\alpha_0^*, \beta_0^*) \quad (D.3)
\]

\[
p(b^{(e)}_i | \mu_i^*, \sigma_i^{2*}) = \mathcal{N}(\mu_i^*, \sigma_i^{2*}) \quad (D.2)
\]

where

\[
\mu_i^* = \frac{\tau_e \mu_e + \tau_k \sum_{n=1}^N \delta(I_{n,e} = i) \left( y_n - X^T W_k - \sum_{e' = 1, e' \neq e}^E b^{(e')}_n \right)}{\tau_i^*} \quad (D.4)
\]

\[
\tau_i^* = \tau_e + N_i \tau_k \quad (D.5)
\]

\[
\kappa_e^* = \kappa_e + n \quad (D.6)
\]

\[
V^* = \left[ V_0^{-1} + X_k^T X_k \right]^{-1} \quad (D.10)
\]

\[
\mu_e^* = \frac{\kappa_e \mu_e + n \bar{b}^{(e)}}{\kappa_e^*} \quad (D.7)
\]

\[
\mu_k^* = V^* \left[ V_0^{-1} \mu_0 + X_k^T r_k \right] \quad (D.11)
\]

\[
\alpha_e^* = \alpha_e + \frac{n}{2} \quad (D.8)
\]

\[
\alpha^* = \alpha_0 + N_k / 2 \quad (D.12)
\]

\[
\beta_e^* = \beta_e + \frac{1}{2} \sum_{n=1}^{N_k} (b^{(e)} - \bar{b}^{(e)}) \quad (D.9)
\]

\[
\beta^* = \beta_0 \quad (D.13)
\]

\[
\frac{1}{2n} \left( \bar{b}^{(e)} - \mu_e^* \right)^2 + \frac{1}{2} \left( r_k r_k^T + \mu_0 V_0^{-1} \mu_0 - \mu^* V^* \mu^* \right)
\]

\[
+ \frac{N_k}{n + \kappa} \left( \bar{b}^{(e)} - \mu_e^* \right)^2
\]

Note that \( r_n \) is the residual resulting from \( y_n - \sum_{e=1}^E b^{(e)}_n \). So we first fit the bias terms and then fit the profiles on the residual. We also introduce \( X_k = \{X_n\}_{I_n = k}, r_k = \{r_n\}_{I_n = k} \), and \( N_k = \sum_{n=1}^N \delta(I_n = k) \) for notational convenience. These are the feature vectors, residuals, and interaction counts for profile \( k \).
Bibliography


88
Mack Sweeney is currently a Masters student in Computer Science at George Mason University (GMU), in the D.C. Metropolitan area. After completing his Masters degree, he plans to continue at George Mason for his Ph.D. in Computer Science. Before beginning graduate studies, Mack earned his Bachelor of Science in Computer Science from GMU. In addition to his academic pursuits, Mack works full-time as a Data Scientist at Research Innovations, Inc. on the Live Data Collection and Analytics Team. His research interests are wide and varied; they include machine learning, decision theory, information retrieval, Bayesian inference, reinforcement learning, scalable and high-performance computing architectures, and effective software engineering practices.