Analyzing Computer Science Students' Performance Data to Identify Impactful Curricular Changes

Hui Yang
Department of Computer Science
San Francisco State University
San Francisco, USA
huiyang@sfsu.edu

Thomas W. Olson
Department of Computer Science
San Francisco State University
San Francisco, USA
two@mail.sfsu.edu

Arno Puder
Department of Computer Science
San Francisco State University
San Francisco, USA
arno@sfsu.edu

Abstract—**This full research paper presents a systematic analysis of 10 years' student performance data of Computer Science (CS) majors at San Francisco State University, a public 4-year degree-granting university, aiming to address the ongoing challenges of early dropouts and low graduation rate. The main objective is two-fold: (1) gain a comprehensive understanding of how the existing curriculum has been supporting (or hindering) students' progress towards graduation; and (2) suggest data-informed curricular changes.**

To this end, we utilize both explorative statistical analysis and data mining/machine learning approaches to first learn how individual courses and the prescribed course sequences influence a student's dropout/graduation status, and then build machine learning models to interpret/validate the observed inter-dependency among key courses in the current curriculum. Such patterns/models are consequently utilized to suggest impactful curricular changes towards reducing early dropouts and improving the overall student success as measured by graduation with a CS degree.

One main finding of this research is that a successful CS student needs to excel in both critical thinking and core CS skills. To help students gain critical thinking skills, it is essential to strengthen the presence of mathematics and physics courses in the CS curriculum. Furthermore, our results suggest that CS students without a solid math foundation before starting their college career should complete a remedial math course earlier than putting it off for later. Moreover, before students advance to the second half of their CS study to gain core CS knowledge/skills (e.g., operating systems), they should complete the required physics class. Finally, we observe that it is necessary to introduce new prerequisite requirements among upper-level CS courses, for example, Operating Systems as a prerequisite to an upper-level CS core course on programming theories.

Keywords—engineering curriculum, computer science curriculum, statistical analysis, data mining, machine learning

I. INTRODUCTION

The demand for a college degree in STEM, particularly in computer science (CS), continues to rise in recent years. We however also witness ongoing challenges of early dropouts and late graduation if graduating at all. A recent study reports that the percentage of students who enter college and do not finish consistently runs to around 40-50% [1]. For students who eventually graduate with a bachelor's degree, it often takes longer than the expected 4 years. According to the most recent National Center for Educational Statistics report [2], the nationwide 6-year graduation rate of first-time, full-time undergraduate students who began seeking a bachelor's degree at 4-year degree-granting institutions in fall 2012 was only 62 percent. This rate however dropped to 34% for colleges with an open admission policy.

We in the computer science department at San Francisco State University, a 4-year degree-granting public university, have been facing the same challenges, in addition to a lack of a diverse student body [3]. For example, between 2009 and 2018, 30+% of the 3000+ CS students dropped out during their first two years of college. Excluding students who are still in progress at the time of conducting this research, the 6-year graduation rate was around 57%.

To address such challenges, we have adopted a data-driven approach, commonly referred to as educational data mining [4][5][6][7][8]. Specifically, we have applied statistical data analysis, data mining and machine learning approaches to analyze the academic performance data of 3,200+ computer science students between 2009 and 2018, along with limited demographic information such as gender and ethnicity. Note that, as summarized in [18], a college student's success is the result of a range of concerted constructs such as learning experience, academic advising, social connectedness, and staff approachability. This research primarily focuses on improving a student's learning experience.

Our study focuses on answering the following questions:

- What are the main characteristics of our CS students? In addition to characterizing our student body's demographic composition in the context of dropout and graduation, we focus on studying how a student's college-readiness can impact her college success, as this issue can potentially be addressed through revising the existing curriculum.
- What are the critical courses that frequently "make or break" a student in the context of college dropout, changing to a different major, and graduation with a CS degree? Answering this question will help anchor the subsequent analysis. This will also allow the department to make data-informed decision to allocate its resources.

- How has the current curriculum's prescribed sequence of study, realized by imposing mandatory prerequisites before enrolling a given course, been preparing our students towards successful graduation? We are also interested in identifying courses that should have a prerequisite relation.
- Finally, what are the high-impact courses that can serve as good indicators of a student's future graduation/dropout status at different stages of her college career? Furthermore, what prior courses in the curriculum are most influential in preparing a student to succeed in CS core courses? The answers to these questions will help us gain insights of the inter-dependency among the 21 required courses in the current curriculum.

To tackle the first two questions, we apply the Exploratory Data Analysis (EDA) approach and philosophy [9]. For the third question, we perform a comprehensive correlation analysis while respecting the curriculum-prescribed course ordering. We also investigate how course ordering, that is, taking course A before, after, or concurrently with course B, can positively or negatively affect a student's performance in course B. Finally, we cast the last task as a classification problem and apply a host of classification algorithms including Logistic Regression [9], Support Vector Machine (SVM) [11], and Random Forests classifiers [12]. We then adopt an ensemble method utilizing the feature importance scores or weights generated by all the above classifiers to identify the high-impact courses.

Before summarizing the main findings, we would like to point out the two main premises followed in this research: (1) a student's academic performance in a given class is largely a faithful reflection of her learning outcome; and (2) variances introduced by different instructors of the same course can be effectively smoothed out due to our relatively large number of course sessions within the 10 years under study.

The results of the above analysis clearly demonstrate that a successful CS student needs to excel in both critical thinking and core CS skills. To help students gain critical thinking skills, it is essential to strengthen the presence of courses in disciplines such as mathematics and physics in the CS curriculum. This finding corroborates a similar conclusion from other researchers concerning computer science education [14][15][16][17]. This finding already played an important role in the authors' department when evaluating whether to exclude one of the two physics courses from the current CS curriculum. Our analysis on unprepared students (those who did not demonstrate a solid math foundation) show that this group of students have less than half the likelihood to graduate with a CS degree when compared with their prepared peers. With the help of the course-ordering analysis, we suggest to implement a simple curricular change that can potentially help these students graduate with a higher likelihood, that is, requesting these unprepared students to complete a remedial math course before taking their first CS programming language class. Finally, we observe that it is necessary to introduce new prerequisite requirements among upper-level CS courses, for example, Operating Systems as a prerequisite to an upper-level CS core course on Programming Theories.

These insights and findings demonstrate the great potential of making data-informed recommendations to improve students' learning experience. Our findings may not be readily applicable to other universities, our methodology however is generalizable.

II. Related Works

This research is directly related to the following areas: curricular planning, graduation prediction, and performance prediction. It is also related to works that emphasize the importance of critical thinking skills in computer science education. Here, we briefly discuss a few representative works that are most germane to this work. For comprehensive surveys on educational data mining, readers are referred to [6][7][8][24][27].

Al-Radaideh *et al.* present a model that relies on the use of decision trees to predict student performance in future courses [19]. The model draws on past academic performance and demographic data. Ibrahim *et al.* utilize demographic data and a student's first semester cumulative GPA to predict academic performance and found that neural networks outperformed decision trees and linear regression [20]. Zimmermann *et al.* build a model to predict graduate student success based on undergraduate studies and find that the third year of undergraduate contains the most information relating to computer science success [21].

Researchers have also used both performance and demographic data to predict students' performance. For example, Saa includes a large variety of demographic data such as parent's occupation, mode of transportation, income, number of friends, together with performance data, to construct a decision tree for grade prediction [22]. The results indicate gender, high school grades, mother occupation, and scholarships provided the most information in terms of scores prediction.

Existing studies also look into what makes certain CS students succeed while others fail. For example, Lahtinen et al. demonstrate in their study that novice computer science students struggle with knowing when to apply certain computer science principles, rather than with learning the principles themselves [23]. Lister et al. find in their study novice CS students often struggle to understand the importance of a line of code in global context [17]. Both of these earlier studies emphasize the importance of problem solving and critical thinking skills in CS students' success, which agrees with our findings in this work.

Another line of work directly related to this research is concerned with the important role math or other science courses play in CS students' success. For example, Katz *et al.* have found through surveys of first year CS students that high school math scores are critical to success in computer sciences [16]. They also find that any benefit of early introduction of computer science courses is outweighed when those courses are taken at the expense of foundational math or science courses. Our analysis agrees with this decade-old finding.

Finally, studies have been conducted for curriculum planning. For instance, Morsy and Karypis show that the timing of courses has a strong correlation with a student's graduation GPA and time to degree [5]. Our study here looks into the ordering of courses and its impact on a student's performance. We also strive to make data-informed curriculum-level changes.

III. METHODOLOGY

Before we present the methods adopted in this study, we first given an overview of the underlying dataset. We then briefly discuss the data preprocessing strategies.

A. The Computer Science Curriculum

The Bachelor of Science Degree program in our Computer Science Department is one of the most popular undergraduate programs at San Francisco State University. Its curriculum consists of 63 major units or 21 mandatory courses, in addition to a number of General Education units and upper-division elective courses. TABLE I. presents a topic-based summary of these 21 courses. For a full list of all these courses, please refer to https://cs.sfsu.edu/undergrad.

TABLE I. A SUMMARY OF THE CS CURRICULUM

Category	Main topics
Math & Physics (8 courses)	M226-Calculus I; M227-Calculus II; M324- Probability and Statistics; M325-Linear Algebra; Ph220-General Physics with Calculus 1; Ph222-Ph220's lab; Ph230-General physics with Calculus II; Ph232-Ph230's lab.
Core CS I (6 courses)	CS210-Intro to programming; CS211-Intro to software lab; CS220-Data structures; CS230-Discrete math for CS; CS256-Machine structures; CS300-CS Ethics
Core CS II (2 courses)	CS340-Programming methodology; CS412-Software lab; CS413-Software development
Adv. CS (4 courses)	CS415-Operating system; CS510-Algorithm analysis; CS600-Programming theories; CS648-software engineering.

B. Dataset Description

The underlying dataset contains 3240 CS major students' course performance data of the above 21 courses between 2009 and 2018, together with limited demographic data. TABLE II. enumerates the list of data fields of each individual student.

TABLE II. DATA DESCRIPTION OF EACH STUDENT

Type of Info	List of attributes
Demographic info	Deidentified ID, Age, Gender, Ethnicity
Student status data	Admission basis (freshman or transfer), Major, Year of addmission, Graduation or not, Year of graduation, Graduation major
Academic performance data	Term GPA, Cumulutive GPA, a list of courses with each course described as (course number, course title, semester taken, grade)

C. Data Preprocessing

TABLE III. CRITERIA ON LABELING EACH STUDENT

Label	Labeling criteria
CS in Progress	not graduated and taking CS courses in 2019
Dropout	not graduated and no progress in their last year.
Graduated in CS or non-CS	Available in the original student record
Non-CS in Progress	not graduated, taking non-CS courses in their last year
Unprepared	must enroll in a pre-calculus course before enrolling in the required lower-division math or physics course

Recall that in this research, we measure a student's success by her dropout and graduation status and use such status to drive the following data analysis. The original dataset does not provide up-to-date status information, except for those who have already graduated from the university. Additionally, we are interested in studying students who joined the department unprepared. We therefore preprocess the data to label each student correspondingly according to the criteria specified in TABLE III. Note that each student can have multiple labels, for instance, "Unprepared" and "Graduated in CS".

D. Explorative Data Analysis to Uncover Trends/Patterns

As described in Section I, we apply the EDA approach and philosophy [9] to tackle the first two issues: (1) What are the main characteristics of our CS students? And (2) What are the critical courses that frequently "make or break" a student?

The approaches behind EDA may seem simple and straightforward, as they largely rely on aggregating, averaging, hypothesis testing (e.g., t-test) and simple visualization tools such as histograms and boxplots. They however enable us to "listen to the data" without pre-judgement. Furthermore, EDA provides evidence-based data insights for model building to address the proposed fourth question. For example, it is due to the application of EDA that guides us to the path of investigating the importance of math and physics courses in a CS curriculum.

To address the first issue above, we generate a collection of pivot tables [25]. The values of such pivot tables include the total number and ratio of students labeled as Dropout, CS_In_Progress, Graduated in CS, and Graduated in non-CS. These values are conditioned upon the following list of single dimensions or their combinations (i.e., rows and columns of a pivot table):

- Admission status: (1) *freshman_start*, where a student was admitted as a freshman; and (2) *transfer_start*, where a student transfers to the department from a different college. Students normally have completed a 2-year associate degree at the time of transfer.
- Year of entry: 2009 to 2018
- Gender: female or male
- Ethnicity: American Indian, Asian, Black, Hispanic, Pacific islander, White, Two or more races, and Unknown. We also include International Students here to simplify the analysis.
- Preparedness: yes or no

Additionally, we also look into (1) the time to graduation and dropout measured in number of semesters; and (2) the top program choices after students forgo their CS major.

For the second issue, we identify the list of courses containing a large number of failures (a grade of C- or lower) over the years. We also infer the list of courses, referred to as *critical courses*, that likely lead to a student's dropout or change of major by checking (1) whether a course is the last one a student failed as a CS major; and (2) the correlation of passing/failing a given class and students' final graduation or dropout status. Chi-square statistics is used for the latter.

E. Correlation and Course-order Impact Analysis

To address the proposed third question: How has the current curriculum's prescribed course sequence been preparing our students towards successful graduation? We first investigate how closely students have been following the recommended 4-year long curriculum. Not surprisingly, given the department's relatively low 4-year graduation rate (15~17% for freshmen who began seeking a B.Sc. degree between 2009-2012), majority of

the CS students deviate from the recommended sequence after their freshman year. Furthermore, we observe that among the students who have graduated with a CS degree, they follow many different pathways to reach their end goal. Moreover, they do not necessarily respect the mandatory prerequisite requirements at times. (This is likely a result of some instructors not reinforcing the mandatory prerequisite requirements).

All these observations put together, we decide to carry out the following two types of analysis to quantify how earlier courses prepare students in subsequent, more advanced courses:

- Correlation analysis between two courses A and B, where A is required to be completed before B according to the curriculum. Pearson's correlation coefficient is calculated using students' normalized performance scores in courses A and B, respectively. A t-test is then performed to identify the course pairs that are significantly correlated [26].
- Course-order impact analysis: Pearson's correlation coefficients are intuitive and easy to interpret; however, they can be easily confounded by the fact that our students do not always follow the required course order. For example, we have observed in our dataset, given any pair of courses (A, B), we can always locate three groups of students: students who take A before B, A after B, and A and B concurrently in the same semester. Let us denote these three groups as $G_{A\rightarrow B}$, $G_{B\rightarrow A}$, and $G_{A||B}$, respectively. To calculate the course order impact between A and B, we first require $G_{A\rightarrow B}$, $G_{B\rightarrow A}$, and $G_{A||B}$ to consist of at least 50 students each, to reduce bias. We then compare the mean score in course B of all the students in $G_{A\rightarrow B}$ with that of all the students in $G_{B\rightarrow A}$, and $G_{A||B}$. If the former is significantly higher, this indicates that students benefit from taking A prior of B. Two-sample t-tests [13] are adopted to infer whether the difference between two such mean scores is statistically significant. We conduct this exhaustive order impact analysis for all the course pairs that meet the above size requirement.

F. Ensemble Methods To Identify High-impact Courses

A host of classification algorithms, namely, Logistic Regression, SVM and Random Forests, are employed to address the last task in this study. We choose these algorithms for two main reasons: (1) they have been regularly adopted in prior work to predict students' learning outcome with satisfying performance [27]. Additional algorithms such as gradient boosting machines [28] were also used, but without significant performance gain; and (2) all these algorithms include a built-in ranking mechanism to measure the importance of each attribute (i.e., a course) in predicting the class label. This in turn enables us to build robust ensemble methods that are described below.

The first subtask uses the learned classification models to identify high-impact courses that help explain a student's dropout and graduation status at different stages of her college career. Specifically, we use the above three classifiers to predict a student's graduation/dropout after she has finished the first 3, 7, 9, and 13 courses, respectively. See TABLE IV. for the specific courses in each list. These five lists of courses are formed by considering both the current curriculum's structure (TABLE I.) and the findings in the previous tasks. They approximately correspond to the completion of 2, 3, 4, and 5 terms, respectively, if this student follows the department's 4-year curriculum. Students with known graduation/dropout records were used to train and evaluate these classifiers. We next outline the main steps to identify the high-impact courses.

TABLE IV. COURSES USED FOR PREDICTING LEARNING OUTCOME

3-course list	M226, M227, CS210
7-course list	M226, M227, CS210, Ph220, Ph222, CS220, CS230
9-course list	M226, M227, CS210, Ph220, Ph222, CS220, CS230, Ph230, Ph232
13-course list	M226, M227, CS210, Ph220, Ph222, CS220, CS230, Ph230, Phy232, M324, M325, CS256, CS340

Let us take the first stage, i.e., having completed the first three courses, as an example. Given the students' performance data in these courses and their demographic data (TABLE II.), we first perform recursive feature selection to retain the most relevant attributes for each of the three classifiers [29]. We then train and evaluate a Logistic Regression model, SVM, and Random Forests using 5-fold cross validation. A grid-search method is utilized to tune each classifier's hyper parameters (e.g., the number of trees for the Random Forests) [30]. Finally, we observe each model's overall performance and their corresponding ranked list of features. If a course is selected by all these three models, we identify them as high-impact courses. Note that intuitively, these courses should also overlap with the critical courses identified earlier using the EDA approach.

The second subtask is to identify which prior courses have the most influence over a later course's performance. We consider a total of four CS-core-skills building courses: one mid-division course on programming methodology, and 3 upper-division courses on operating systems, algorithm analysis, and programming theories, respectively. We apply the same three steps above with two major differences: (1) the dataset is considerably much larger. For each of these four target courses, we construct a separate dataset including all the students who have taken this course. Take the first mid-division course as an example. The class label of this dataset is the letter grade (A, B, C, D, and F) of a student who has taken this course. The predicting attributes are this students' performance in the courses prior to taking this target course; and (2) the above binary classifiers Logistic Regression and SVM adopt the "one vs. one" method to address the multiclass problem here [10].

IV. RESULTS

In this section, we present the major results obtained from addressing each of the aforementioned four questions. We will also identify their curricular implications.

A. Major Characteristics of CS Students

TABLE V. summarizes the year-by-year student data according to a student's admission type, i.e., freshman vs. transfer, and their corresponding rate of unpreparedness. The last three columns list the annual dropout, CS graduation, non-CS graduation rates including both admission types. Note that the graduation rates are calculated for students who did not drop out.

From TABLE V. one can observe the following: First, the number of students admitted to the CS department almost doubled in year 2014 and has been stable since then. This is largely aligned with the national trend [31]. Second, the CS-graduation rates amongst those who did not drop out are constantly below 50% except 2009; while the dropout rates are

as high as 38.84% in 2011. Finally, the unprepared rates of our freshmen-entry students are markedly higher at 41.24~54.04% (except 2009) than those of the transfer-entry students (1.97~16.87%). This is likely due to the facts (1) we are a public university accepting a diverse student body; and (2) transfer students are normally surer of their choice of major, thereby working more diligently to meet the math/physics requirements during their college years before transferring to our department.

We further analyze how a student's unpreparedness in math/physics can affect their final graduation outcome. As shown in TABLE VI. , the CS-graduation rates for unprepared students are less than half of their prepared counterparts, regardless of the entry method. *This is a strong indicator that lack of a solid math or physics foundation can hinder/prevent CS students from reaching their academic goals.* We demonstrate later that by adjusting the current curriculum in a relatively minor way, we can potentially help these unprepared students to succeed with a CS degree.

TABLE V. UNPREPARED, DROPOUT, AND GRADUATION RATES

Year	Total# freshmen	%unprep freshmen	Total # transfers	%unprep transfer	% Dropout	% GradCS	% Grad Non CS
2009	129	18.60	122	6.56	27.49	52.19	19.52
2010	60	41.67	63	9.52	34.96	43.09	21.14
2011	60	50.00	61	11.48	38.84	42.15	16.53
2012	97	41.24	83	16.87	30.56	48.33	16.67
2013	119	47.06	95	7.37	35.51	44.39	12.62
2014	239	45.19	159	9.43	35.43	35.93	10.30
2015	235	54.04	180	10.56	35.18	22.89	3.61
2016	274	48.91	207	10.63	30.77	15.18	0.21*
2017	262	47.71	264	*6.82*	*21.10**	*1.52*	*0.00**
2018	277	44.77	254	*1.97*	*9.79**	*0.00*	*0.00**

*: These low percentages are a result of many students who were still early in their college life.

TABLE VI. COMPARISON OF PREPARED AND UNPREPARED STUDENTS

Admission Type	Prepared		Unprepared	
	Graduated CS	Graduated	Graduated CS	Graduated
Freshman	38.34%	56.07%	16.67%	29.10%
Transfer	53.15%	59.35%	23.75%	45.00%

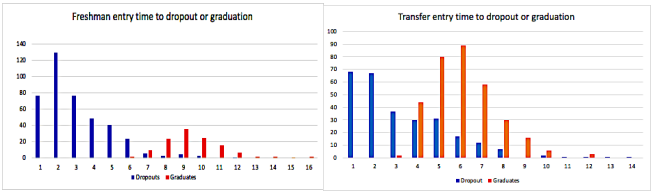
TABLE VII. GENDER VS. UNPREPARED/GRADUATION RATES

Admission Status	Totals	Unprepared%	Graduated CS %	College Grad Rate
Freshman	1752	45.26%	29.21%	44.70%
Female	364	50.27%	24.26%	49.70%
Male	1388	43.95%	30.36%	43.54%
Transfer	1488	8.13%	50.64%	58.12%
female	206	9.71%	44.17%	54.17%
male	1282	7.88%	51.59%	58.70%

To investigate who these unprepared students are, we conduct further analyses along the gender or ethnicity dimension. TABLE VII. and TABLE VIII. show that the CS student body exhibits a severe lack of diversity in the past 10 years in both gender and ethnicity. Specifically, it is dominated by male students. The top three ethnic groups are Asian, White, and Hispanic. Furthermore, more female freshmen-students join the department unprepared than male students (50.27% vs. 43.95%). For transfer students, this gap of unpreparedness between females and males shrinks to less than 2%. With respect to ethnicity, 69.12% of black freshman-entry students came unprepared, followed by 55.62% Hispanic students. As for transfer students, regardless of their ethnicity, around 90% or more join the department prepared. *This contrast of freshman- and transfer-entry students in terms of preparedness provides us evidence that gender or ethnicity has little bearing over a student's success in CS.* Instead, we should focus on supporting such unprepared students via curricular innovations and orchestrated support at the institutional level.

TABLE VIII. ETHNICITY VS. UNPREPARED/GRADUATION RATES

Admission Status	Ethnicity	Totals	Unprepared%	Graduated CS %	College Grad Rate
Freshman	Asian	695	40.58%	45.35%	54.63%
	Black	68	**69.12%**	**25.00%**	**41.94%**
	Hispanic	371	**55.26%**	**10.81%**	**22.81%**
	International	176	41.48%	47.31%	53.33%
	TwoMore	87	**47.13%**	**18.42%**	**32.61%**
	Unknown	60	30.00%	40.00%	48.57%
	White	281	42.35%	36.05%	45.98%
Transfer	Asian	564	6.21%	60.42%	62.57%
	Black	64	**10.94%**	**36.36%**	**41..67%**
	Hispanic	179	8.38%	46.15%	**48.62%**
	International	173	6.94%	**44.90%**	**47.06%**
	TwoMore	75	**10.67%**	**45.24%**	52.08%
	Unknown	103	5.83%	63.33%	68.12%
	White	323	**11.46%**	57.51%	62.21%



(a) freshman entry (b) Transfer entry

Fig. 1. Time to dropout (blue) and graduation (orange). (a) The number of freshmen-entry students dropped out or graduated after a given term; and (b) Number of transfer-entry students dropped or graduated after a given term.

Fig. 1 characterizes the timing of CS students' dropout and graduation. For freshman-entry students, the most common graduation timeline is 9 semesters. Amongst these who graduated, around 70% took 9 or more semesters (or >=4.5 academic years). For transfer students who normally enter as juniors, their time to graduation resembles a normal distribution centered at 6 semesters. Out of these who graduated, more than 85% took more than 2 academic years to graduate. *Such data suggest that it is necessary for the department to reevaluate its recommended 4-year curriculum, for example, offering a 4.5-year or 5-year long version of the current curriculum.*

In terms of dropouts, 80% of freshman-entry dropouts occur in the first 4 semesters, with most occurring after the second semester. For transfer students, the first two semesters are the most vulnerable to dropouts, where 73% of the dropouts occur. Further analysis of the freshman-entry dropouts indicates that nearly 60.00% of the dropouts in the first 4-semesters joined the department unprepared. *This further reinforces an earlier observation that a student's lack of math and physics skillsets before joining CS may play a vital role in deterring her from making further progress.*

Finally, about 15% (478) CS students during 2009-2018 changed their major. The most commonly chosen destination major is Business Administration Information Systems. This is not surprising due to its overlap with computer science.

B. List of Critical Courses in the CS Curriculum

In this section, we report the critical courses that mostly likely result in a student's dropout. Chi-Square independence test is performed for each of the 21 mandatory courses (see TABLE I.) and confirms a statistically significant correlation between passing/failing a class and a student's dropout/ graduation status (all the *p*-values are less than 0.001 except CS648 at 0.02). A further investigation shows that failing some course are more detrimental to students than failing others. Note that the minimum letter grade to pass these courses is C.

Specifically, for freshman-entry students, if a student fails one of the following courses with a grade of C- or below, she has a 0.70 or higher probability to drop out or change her major: CS210, M226, CS220, M227, Ph220, Ph222, CS230 (essentially a math course). For transfer students, this list of courses includes to CS220, M324, CS230, CS340, and M325.

These two lists of critical courses agree with our earlier dropout timing analysis, as the former list falls into the first two years of a freshman-entry student, whereas the latter the first two semesters of a transfer-entry student. We also observe that students become more resistant to course failures after they have finished the first two years (or two semesters for transfer students) of college. Specifically, the dropout ratio among all the students who failed any of the mid- or high-level CS course sharply decreases to 31% or lower. Observing these two lists, one notices again that *failing math/physics courses plays an important role in pushing students out of CS in their early college career*. One may ask: *is it really necessary to include math or physics courses in a CS curriculum?*

C. Correlation and Course-order Impact Analysis Results

The answer to the above question is: *yes, it is necessary and important to include math and physics courses in a CS curriculum*. TABLE IX. lists the top-5 correlated courses for each of the core CS courses. One notices that math/physics courses are correlated with each and every CS course, including the final high-level core courses such as CS510, CS600 and CS648. We understand that correlation does not equate causality. Combining the results shown in TABLE VI. and TABLE IX. , one can't dismiss the important role such foundational math and physics courses have played in helping CS students succeed. *Our main hypothesis of this phenomenon is that math and physics courses help train a student's analytical and problem-solving skills. This in turn helps CS students grasp CS core concepts and knowledge at a deeper level within a larger context, thus building an ever larger and longer-lasting knowledge base as they progress through the CS curriculum.*

TABLE IX. TOP 5 CORRELATED COURSES (P-VAL<0.01)

Course	Top 5 correlated prior courses (Pearson's correlation coefficent)
CS210	**Ph220** (.56), **M227** (.50), **M226** (.32)
CS220	**M324** (.49), CS230 (.49), **M227** (.41), **Ph222** (.37), **Ph220** (.35)
CS230	**M324** (.61), **M325** (.56), **Ph220** (.46), CS220 (.42), **M227** (.35)
CS256	CS340 (.49), **Ph230** (.44), **M324** (.36), CS230 (.34), CS220 (.31)
CS340	**Ph230** (.39), **Ph232** (.31), **M227** (.26), **M324** (.26), CS220 (.25)
CS413	CS510 (.42), **Ph232** (.33), **Ph230** (.30), **Ph220** (.27), **M324** (.25)
CS415	CS648 (.54), CS600 (.41), CS510 (.35), **Ph232** (.30), CS413 (.29)
CS510	CS600 (.77), CS648 (.64), CS415 (.42), CS413 (.41), **Ph232** (.39)
CS600	**Ph230** (.35), CS510 (.35), **M325** (.32), CS648 (.28), **Ph220** (.28)
CS648	**Ph232** (.37), CS600 (.33), CS415 (.27), **Ph230** (.24), CS510 (.20)

Next, we present the main results of course-order impact analysis. As summarized in TABLE X. , taking a course too early or too later can have different effects. The last column is the p-value of the two-sample t-test performed to verify whether the difference between the two mean scores (columns 4 and 5, respectively) is statistically significant.

TABLE X. FRESHMAN-ENTRY STUDENTS COURSE-ORDER IMPACT

Course A	Rel	CourseB	MeanScore CourseB (#stduents)	MeanScore CourseB if ¬(A rel. B) (#students)	p-value
Ph230	→	M324	**76.39** (184)	68.64 (125)	<0.01
Ph232	→	M325	**78.71** (148)	70.77 (154)	<0.01
M324	→	Ph230	72.80 (125)	**79.15** (184)	<0.01
CS340	→	Ph230	73.26 (128)	**80.43** (181)	<0.01
CS412	→	Ph230	71.72 (95)	**80.19** (183)	<0.01
CS256	→	Ph230	73.06 (133)	**79.68** (186)	<0.01
CS210	→	M226	72.53 (105)	**78.62** (240)	<0.01
CS220	→	Ph220	71.62 (102)	**77.52** (218)	0.02
CS230	→	Ph222	81.15 (81)	**87.05** (255)	0.03
CS256	→	Ph232	83.20 (130)	**86.42** (187)	0.04
CS230	‖	Ph230	**80.76** (155)	75.48 (300)	<0.01
CS220	‖	Ph230	**80.24** (134)	75.79 (316)	<0.01
Ph230	‖	CS220	**84.70** (134)	81.22 (316)	<0.01
CS340	→	M324	**76.89** (144)	70.67 (137)	0.01
CS256	→	CS340	68.13 (85)	**75.28** (61)	0.04
CS600	‖	CS415	**83.28** (109)	75.64 (215)	<0.01
CS510	‖	CS600	69.55 (88)	**75.70** (239)	<0.01
CS600	‖	CS648	83.68 (102)	**86.71** (168)	0.01

One observes the following from TABLE X. : First, taking the two physics courses Ph230 and Ph232 before the last two math courses M324 and M325, respectively, are helpful at improving a student's performance in these two math courses. *Given that these are the critical courses for transfer students, we may recommend our transfer students to review Ph230 and Ph232 materials before taking M324 and M325.* Second, pushing math and physics courses after having taken CS courses (e.g., CS340→Ph230) often has an adverse effect on the corresponding math or physics class. *Considering the importance of math and physics courses, this leads us to recommend CS students take the required math and physics courses earlier rather than later.* Third, taking CS220 or CS230 concurrently with Ph230 helps improve students' Ph230 performance. Also, it is beneficial for students to take M324 after CS340, which is not required in the current CS curriculum. These suggest that after a student has succeeded in their first math and physics course, *CS and Math/Physics courses start to support each other*. Finally, *among the CS courses, we may need to reconsider their prerequisite relationships*. For instance, the data in TABLE X. suggest to make CS340 a prerequisite of CS256 (which is currently recommended to take place concurrently with or before CS340), and CS600 the last course instead of the current CS648.

TABLE XI. COURSE-ORDER IMPACT FOR UNPREPARED STUDENTS

CourseA	Rel.	CourseB	MeanScore CourseB	MeanScore CourseB if ¬(A rel. B)	p-val
CS210	‖	M199	77.96 (143)	**81.51 (275)**	0.07
M199	→	CS210	**76.39 (230)**	70.22 (45)	0.08

TABLE XII. IMPACT OF COURSE-ORDER OVER UNPREPARED STUDENTS IN THEIR DROPOUT AND GRADUATION STATUS

	#students	#in-prog	%dropout	%GradCS	%Grad
M199→ CS210	230	129	29.57% (68)	17.82% (18)	32.67% (33)
M199 ‖ CS210	133	80	36.84% (49)	7.55% (4)	7.55% (4)

We end this section by refocusing on the unprepared students. As shown in TABLE XI. , if a student takes CS210 concurrently with M199 (a required pre-calculus course for unprepared students), she will earn a lower score in M199. On

the other hand, if she takes M199 before CS210, she will do much better in CS210, which in turn increases her chance to stay in the CS program. TABLE XII. further validates the positive impact of M199→CS210. Even though the number of graduated students is relatively small, it suggests that by *requiring unprepared freshman-entry students to take M199 prior to CS210 can potentially decrease their chance of early dropout, thereby increasing their chance of graduation in CS or non-CS*.

Note that results in TABLE X. TABLE XI. and TABLE XII. also provide further evidence that math and physics courses may help improve students' analytical and problem-solving skills: students generally achieve better grades in a CS core course (e.g., CS210) if she takes this course after having completed a math/physics course (e.g., M199 and M226).

D. List of High-impact Courses Based on Outcome Prediction

In Sections IV.A, IV.B, and IV.C, we summarize the main findings by observing how 2 or 3 attributes (e.g., two courses, preparedness vs. graduation/dropout) interact with each other. In this section, we present the list of high-impact courses at both the curricular and course level by employing an ensemble of three classification algorithms: Logistic Regression, Random Forests and SVM.

TABLE XIII. summarizes the performance of these three classification models using 3, 7, 9, and 13 courses, respectively, (see TABLE IV.) to predict a student's graduation/dropout status. Five-fold cross validation is employed. The last column lists the high-impact course selected by all three models, math and physics courses are in bold for clarity.

TABLE XIII. GRADUATION PREDICTION & HIGH-IMPACT COURSES

#Classes used	Algo.	Acc.	Prec.	Rec.	F1	High-impact courses
3	L.R.	0.69	0.69	0.80	**0.74**	**M227**
	R.F.	0.76	0.69	0.75	0.72	
	SVM	0.69	0.60	0.64	0.61	
7	L.R.	0.81	0.81	0.94	**0.87**	CS230 (math intensive), **Ph220, Py222**, **M227**, CS220
	R.F.	0.81	0.83	0.91	0.86	
	SVM	0.79	0.78	0.95	0.86	
9	L.R.	0.77	0.77	0.60	0.67	**Ph230, Ph232**
	R.F.	0.81	0.75	0.80	**0.77**	
	SVM	0.77	0.77	0.60	0.67	
13	L.R.	0.85	0.78	0.85	0.81	CS340, **Ph230, Ph232, M324, M325,** CS256, CS220, **Ph220, Ph222**
	R.F.	0.87	0.83	0.84	**0.83**	
	SVM	0.83	0.76	0.84	0.79	

From TABLE XIII. , we observe that these three models deliver the best F1 score using 7 courses (approximately corresponding to the list of courses recommended to finish in the first 3 semesters), followed by using 13 courses (approximately corresponding to the list of courses recommended to finish in the first 5 semesters). Examining the corresponding high-impact courses, the presence of math and physics courses is unmistakable. It is also important to notice that important CS courses such as CS220-Data Structure, CS256-Machine Structures, and CS340-Programming Methodology start to play an important role in helping explain a student's graduation/dropout outcome. The fact M227 instead of CS210 is selected by all the models when using the first 3 courses further validates the critical role math skills assume in supporting CS students' success. Moreover, when using the first 9 courses for prediction, Ph230 and Ph232 are the only two courses chosen by these three models. This is a bit surprising. On the one hand, it shows that failing these two courses often end a student's pursuit of a CS degree. On the other hand, succeeding in these courses often bring a student closer to graduate with a CS degree.

In summary, *these results reveal the dual nature of the CS curriculum: one part analytical and critical thinking, and the other part CS foundation and theories.* During the first stage, roughly corresponding to finishing the first year's recommended courses, math and physics are strong indicators to explain a student's final academic outcome. During the second stage, roughly corresponding to finishing the second year's recommended courses, CS courses (e.g., CS220-Data Structure) start to emerge and help explain a student's final outcome. Finally, during the last stage, roughly corresponding to having finished the first 2.5 year' recommended courses, CS core courses together with math/physics ones come together to help explain a student's final graduation/dropout outcome.

This above observation becomes even more noticeable in TABLE XIV. , which identifies the prior influential courses in terms of predicting the letter-grade performance in one of the following four CS core courses: CS340-Programming Methodology, CS415-Operating Systems, CS510-Algorithm analysis, and CS600-Programming theories.

TABLE XIV. INFLUENTIAL PRIOR COURSES OF CS CORE COURSES

Target course	**#prior classes**	**Algo.**	**Acc.**	**Prec.**	**Rec.**	**F1**	**Influential prior courses**
CS340	9	L.R.	0.45	0.45	0.45	**0.45**	CS230 (math intensive), **Ph230, Py232, M227**
		R.F.	0.39	0.38	0.41	0.39	
		SVM	0.43	0.43	0.43	0.43	
CS415	13	L.R.	0.40	0.40	0.40	0.40	CS220, **Ph230, Ph232**, CS256, CS340, **M324, M325,** CS230, **M226**, CS210, **M227**
		R.F.	0.47	0.47	0.45	**0.48**	
		SVM	0.42	0.42	0.42	0.42	
CS510	13	L.R.	0.51	0.51	0.51	0.51	**Ph220, Ph222 Ph230, Ph232**
		R.F.	0.51	0.53	0.50	**0.52**	
		SVM	0.51	0.51	0.51	0.51	
CS600	17	L.R.	0.45	0.45	0.45	0.45	CS510, CS415, **M325, M324**
		R.F.	0.44	0.48	0.46	0.46	
		SVM	00.47	0.47	0.47	**0.47**	

Comparing with the performance data in TABLE XIII. and TABLE XIV. , all three models exhibit much lower performance scores at predicting the letter grade of each course using prior courses' performance data. This is likely due to the following two main reasons: (1) this is a multi-classification problem of 4-class labels (i.e., letter grades A, B, C, D, and F); (2) as shown in our own research **Error! Reference source not found.**, knowledge in prior courses can only partially explain a student's performance in a given course. There are other more important factors such as a student's understanding in specific topics covered in a given course, which can be captured by a student's course activities such as assignments and examinations.

The resulting lists of influential prior courses in TABLE XIV. clearly highlight the strong presence of math courses in CS340, and physics courses in CS510. Furthermore, we also *observe a synergy between CS and math/physics courses in influencing the performance of CS415-Operating Systems and CS600-Programming Theories*.

To further investigate the influence of math/physics courses over a CS core course. We conduct a comparative study to predict CS510 performance by excluding physics courses from the list of prior courses. (An ideal comparative study would

require certain students not to take these physics courses. This of course is unrealistic.) The results are shown in TABLE XV. One can observe here that the performance of all three classification models decreases, especially for Random Forests and SVM with a 13% drop and 5% in F1 score, respectively.

TABLE XV. IMPACT OF PHYSICS COURSES FOR CS510 PERFORMANCE PREDICTION (W/: INCLUDING PHYSICS CLASSES; W/O: EXCLUDING PHYSICS CLASSES)

	Accuracy		Precesion		Recall		F1	
Alg.	w/	w/o	w/	w/o	w/	w/o	w/	w/o
L.R.	**0.51**	0.49	**0.51**	0.49	**0.51**	0.49	**0.51**	0.49
R.F.	**0.51**	0.41	**0.53**	0.39	**0.50**	0.43	**0.52**	0.39
SVM	**0.51**	0.46	**0.51**	0.46	**0.51**	0.46	**0.51**	0.46

In summary, these classification models help shed lights in understanding and explaining the academic outcome of CS students in the context of graduation/dropout and CS core courses using primarily academic performance data. The resulting lists of high-impact and influential courses further strengthen the dualistic nature of a CS curriculum. It is worth noting that based on the main findings of this research, the authors' department has decided to reconsider an earlier motion to replace Ph230 and Ph232 with two additional CS courses.

V. CONCLUSION AND FUTURE WORK

We conduct a systematic study of 10 years' students' performance data in the computer science department at a 4-year degree-granting public university by employing both explorative data analysis and data mining approaches. This study focuses on identifying and learning patterns embedded in the current CS curriculum that either support a CS student to graduate with a CS degree or deter one from achieving this goal. This will help us make data-informed curricular changes.

From the curricular perspective, this study validates the necessity and importance of including math and physics courses towards nurturing CS students' analytical and problem-solving skills. Such courses take a more prominent role at explaining students' success during the first stage of a student's college journey. We are pleased to observe that gradually both CS and math/physics courses work synergistically to support a student's CS journey. Finally, this study also points out a few immediately actionable curricular changes, including (1) require unprepared CS students to finish a remedial math course before taking the first CS core course in programming; (2) suggest students to take the first set of math and physics courses earlier other than later; and (3) introduce additional prerequisite requirements such as CS340→M324, CS340→CS256, and CS415→CS600.

This research naturally leads to the following directions: How indispensable are math/physics courses in a CS curriculum? Can they be replaced by a different set of courses that are designed to train and foster critical thinking skills? How generalizable are the findings in this research, for example, in other STEM fields? Finally, what measures should be taken to perform fair data analysis in the face of heavily skewed underlying data that are biased against female and underrepresented minority students? We invite our colleagues to explore these directions with us in the future.

REFERENCES

[1] Webber, D. A. "Are college costs worth it? How ability, major, and debt affect the returns to schooling." *Econ. of Edu. Review* 53 (2016): 296-310.

[2] "Undergraduate Retention and Graduation Rates", National Center for Educationa Statistics, U.S. Department of Education, April 2020. https://nces.ed.gov/programs/coe/indicator_ctr.asp

[3] Institutional Research, San Francisco State University, https://ir.sfsu.edu/content/students-data

[4] Elbadrawy, A. and Karypis, G. Domain-aware grade prediction and top-n course recommendation. 10th ACM Conf. on Rec. Sys. 183–190, 2016.

[5] Morsy, S. and Karypis, G. 2019. A study on curriculum planning and its relationship with graduation gpa and time to degree.In Proc. of the 9th Intern'l Conf. on Learning Analytics & Knowledge. ACM, 26–35.

[6] Hu, Qian and Huzefa Rangwala. 2019. Reliable Deep Grade Prediction with Uncertainty Estimation. In Proc. of the 9th Int. Conf. on Learning Analytics & Knowledge. ACM,76–85.

[7] Peña-Ayala, A. "Educational data mining: A survey and a data mining-based analysis of recent works." *Exp. sys. w. App.* 41.4 (2014): 1432-62.

[8] Dutt, Ashish, Maizatul Akmar Ismail, and Tutut Herawan. "A systematic review on educational data mining." *Ieee Access* 5 (2017): 15991-16005.

[9] Tukey, John (1977), *Exploratory Data Analysis*, Addison-Wesley.

[10] Christopher M. B.: Pattern Recognition and Machine Learning, Ch. 4.3.4

[11] B.E. Boser I.M. Guyon and V.N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," Proc. Fifth Ann. Workshop Computational Learning Theory, ACM Press, New York, 1992, pp. 144-152.

[12] Leo Breiman. 2001. Random Forests. *Mach. Learn.* 45, 1 (October 1 2001), 5–32. DOI:https://doi.org/10.1023/A:1010933404324

[13] Snedecor, George W. and Cochran, William G. (1989), *Statistical Methods*, Eighth Edition, Iowa State University Press.

[14] Feder, Toni "Convincing US states to require physics." Physics Today 64, 7, 29 (2011); https://doi.org/10.1063/PT.3.1163

[15] Wilson, Brenda Cantwell. "A study of factors promoting success in computer science including gender differences." *Computer Science Education* 12.1-2 (2002): 141-164.

[16] Katz, Sandra, *et al.* "Gender, achievement, and persistence in an undergraduate computer science program." *ACM SIGMIS Database: the DATABASE for Advances in Information Systems* 37.4 (2006): 42-57.

[17] Lister, Raymond, *et al.* "Not seeing the forest for the trees: novice programmers and the SOLO taxonomy." *ACM SIGCSE Bulletin* 38.3 (2006): 118-122.

[18] Hanover Research, Strategies for Improving Student Retention, Sep. 2014

[19] Al-Radaideh, Qasem A., *et al.*. "Mining student data using decision trees." *Intern'l Arab Conf. on Info. Tech. (ACIT'2006), Jordan*. 2006.

[20] Ibrahim, Zaidah, and Daliela Rusli. "Predicting students' academic performance: comparing artificial neural network, decision tree and linear regression." *21st Annual SAS Malaysia Forum, 5th September*. 2007.

[21] Zimmermann, Judith, et al. "A Model-Based Approach to Predicting Graduate-Level Performance Using Indicators of Undergraduate-Level Performance." *Journal of Educational Data Mining* 7.3 (2015): 151-176.

[22] Saa, Amjad Abu. "Educational data mining & students' performance prediction." Inter. J. of Adv. Comp. Sci. & App. 7.5 (2016): 212-220.

[23] Lahtinen, Essi *et al.*. "A study of the difficulties of novice programmers." *Acm sigcse bulletin* 37.3 (2005): 14-18.

[24] A. Hicham *et al.*, "A Survey on Educational Data Mining [2014-2019]," *2020 ISCV*, pp. 1-6, doi: 10.1109/ISCV49265.2020.9204013.

[25] Jelen, Bill; Alexander, Michael (2006). Pivot table data crunching. Indianapolis: Que. pp. 274. ISBN 0-7897-3435-4.

[26] https://opentextbc.ca/introbusinessstatopenstax/chapter/testing-the-significance-of-the-correlation-coefficient/

[27] K. L. Ang *et al.*, "Big Educational Data & Analytics: Survey, Architecture and Challenges," in *IEEE Access*, vol. 8, pp. 116392-116414, 2020.

[28] Tianqi Chen, Carlos Guestrin, "XGBoost: A Scalable Tree Boosting System", SIGKDD 2016.

[29] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

[30] https://scikit-learn.org/stable/modules/grid_search.html#grid-search

[31] CRA Taulbee Survey, https://cra.org/data/generation-cs/phenomenal-growth-cs-majors-since-2006/

[32] Kiriti Upadhyaya, Analyze the factors that help students succeed in the Programming Methodology course, Master's Thesis, Department of Computer Science, San Francisco State University, 2021.