Data-Driven Models to Predict Student Performance and Improve Advising in Computer Science

Varick L. Erickson
Computer Science
California State University, East Bay
varick.erickson@csueastbay.edu

Abstract—As enrollment in computer science (CS) majors continues to rise, CS departments are facing increased pressure to accommodate larger numbers of students, improve student graduation rates, and reduce time to graduate. As a result, effective student advising is critical to ensure the CS major is the right fit for students and to help students complete the degree in a timely and successful manner. Additionally, advisers play a crucial role in identifying struggling students and ensuring at-risk students either improve or quickly transition to a different major (saving resources both for the student and for the department). However, it is challenging for CS advisers to identify struggling students early. Often, by the time at-risk students are identified, these students have expended a considerable amount of time, effort, and money in the major when they could have been pursuing a major more suitable to their strengths. In the present work, we analyze actual enrollment data and develop models for predicting student performance. By leveraging these data-based models, we can identify at risk students with up to 77% accuracy. Based on these models, we also propose approaches to improve student advising in CS undergraduate programs.

Index Terms—Education, Success, Prediction, Classification, Advising, Modeling

I. INTRODUCTION

From 2010-2017, the number of computer science (CS) degrees conferred has increased 50.7% [1]. While many students are interested in pursuing a CS degree, not all students are prepared or suited to the degree. In fact, 28% of students who start in CS will change majors within the first three years [2]. Since CS programs are increasingly overcrowded, it is crucial to help students succeed and graduate in a timely manner. Similarly, students who are having difficulty with the computer science degree need to be offered support in order to quickly improve, or be advised to find another major before investing too heavily into the degree. The sooner a student realizes that CS is not the right fit, the sooner the student can transition to a better fit major, and the sooner the CS department can free up limited space and resources to help additional CS students succeed.

Given the increasing enrollment and high student turnover in CS degree programs, identifying and advising struggling students early in the students’ academic careers is invaluable. Many CS students fail to realize that CS is the wrong major for them until several semesters or even years have passed. Then, these students feel they are “too far along” to change majors. While some struggling students are able to complete the degree, it often takes far more time and repeating classes several times to complete the degree. With low GPAs, these students may also have difficulty succeeding after graduation. Another common reason for staying in the CS major too long is an issue called identity foreclosure. Identity foreclosure refers to situations where students establish goals without first doing thorough exploration of options and self reflection [3]–[5]. Students experiencing identity foreclosure can identify so strongly as CS majors that they refuse to consider alternate majors, even if these students are failing CS courses or not the right fit for the program. As a result, identity foreclosure causes some students to view the CS major as their only option, even if they are struggling.

Having an academic adviser objectively examine each student’s performance and advise appropriately has been shown to make a significant positive impact. For example, advising has been shown to enable persistence rates (retention rates) of 53% among underrepresented students [6], [7]. Therefore, it is critical to flag at-risk students as early as possible. Once these students are identified, an adviser can look more closely at a student’s situation and provide better guidance. Timely and effective adviser support can make the difference between having a struggling student stay in the program and ultimately fail or drop out, helping a student by providing access to support and resources, or helping a student move to a different major.

There are numerous advantages to an early major change. Arguably the most important advantage is decreasing the number of repeated classes. Students who stay too long in the CS major often need to repeat courses. This means affected students face a longer time required before graduation and potential financial challenges that could ultimately lead to abandoning an undergraduate degree. Students who repeat courses also tend to have lower GPAs even if they replace a grade or repeat a course. Based on our data, the average CS GPA of students who needed to repeat a CS course one or more times is 2.54 (on a 4.0 scale). In contrast, students who only needed to take CS courses once had an average GPA of 3.00.

Having struggling students repeat courses also negatively affects CS departments and successful students. When struggling students repeat a course, fewer seats are available for those taking the course for the first time. Again, having fewer seats available can lead to a longer amount of time spent in school and increased financial burden on successful CS students who want to be in the program. Thus, when advisers are able to...
identify and better support at-risk students, all students and the department can benefit as well.

In the present work, we aim to improve student advising by analyzing CS student data, developing predictive models for student performance, and proposing approaches to improve advising. First, the academic data for students at California State University, East Bay was compiled and analyzed. Various models were applied to identify and predict at-risk students. Based on these models, we are able to identify at-risk students with up to 77% accuracy. Based on these results and findings in the data, we also propose additional approaches to help CS departments better advise and support at-risk students.

We start with a survey of related work in Section II. We then discuss the choice of variables in Section III followed by details of data collection in Section IV. A preliminary data analysis is performed in Section V that motivates our model selection described in Sections VI, VII, and VIII. We compare these models in Section IX and then show how these models can be used in an advisory setting in Section X. Finally, we discuss future work in Section XI and summarize our findings in Section XII.

II. RELATED WORK

Researchers have studied predictive models for succeeding in CS for over 30 years. Many of these papers focus on predicting CS performance prior to entering a university [8]–[10]. Authors of [8] attempt to predict performance at the high school level before entering a university program. Similarly, authors of [10] look at a broad index of 62 items to determine learning traits described by Kolb and Pask [11] in order to predict how high school students will perform. Unlike these papers, our goal is to help advisers and current CS students predict success based on recent performance in CS courses.

In [12], authors use a linear regression approach to predict how well students will perform based on past performance. However, there are notable differences as compared with our approach. The authors found that a linear regression model was able to describe their population, which was CS majors at the U.S. Air Force Academy. In our case, we determined a linear regression approach was not suitable based on our residuals. It is likely that hidden factors are not being captured by our data. Additionally, the authors of [12] use a somewhat cumbersome approach, generating 99 models to capture each valid combination of previously taken classes. Our single model is able to capture all combinations of classes taken by treating variables as categorical data rather than continuous data. In the present work, we also introduce a "did not take" label to more accurately model cases where students did not complete a given course.

The authors of [13] have a similar aim as our paper and focus on identifying struggling students early. To predict student performance, they use an integrated development environment (IDE) extension that continuously logs student coding activities. This IDE classifies each student’s successes and failures and uses this data to dynamically refine a prediction model. The authors also compare their approach to a similar method described in [14], which also utilizes a similar dynamically adaptive classifier. In the present work, our approach is simpler, only requiring data already collected by universities and registrars without the use of specialized plugins. Despite the added complexity, both methods examined have a maximum of 75% accuracy. In contrast, the models proposed in the present work achieve up to 77% accuracy for identifying at-risk students.

III. INDICATOR AND RESPONSE VARIABLES

In this section, we discuss our choice of indicator and response variables. Our goal is to be able to predict student success in the major early using a model. To define a response variable, we must first start with our definition of success. There are many different possible metrics for success. One possible metric is overall GPA for CS courses. Another could be the time to complete the degree. For this paper, we decided to examine classes most closely associated with CS interviews. In particular, we wish to use student performance in the Data Structures course as our response variable since interviews often use concepts from this course [15]. Also, Data Structures is a key prerequisite to many courses required by the major. Our approach is also applicable to measuring success for Analysis of Algorithms, which is another popular topic for CS interviews. However, since our goal is to flag at-risk students as early as possible, we define success in Data Structures to be our response variable since Data Structures is typically taken earlier (during a student’s second year).

From an advisory perspective, our aim is to be able to predict Data Structures performance as early as possible, so we are able to identify and advise students early in the program. Thus, logical choices for indicator variables would be collected from required lower division courses that are taken prior to data structures. Table I shows these courses; for clarity, we define simplified course labels rather than use the official university course numbers. Our hypothesis is that performance in these earlier classes will be able to predict performance in Data Structures.

<table>
<thead>
<tr>
<th>Simplified Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS1</td>
<td>First quarter course on programming</td>
</tr>
<tr>
<td>CS2</td>
<td>Second quarter course on programming</td>
</tr>
<tr>
<td>DataS</td>
<td>Data Structures and Algorithms</td>
</tr>
<tr>
<td>Calc1</td>
<td>First course in calculus</td>
</tr>
<tr>
<td>Calc2</td>
<td>Second course in calculus</td>
</tr>
<tr>
<td>L</td>
<td>Linear Algebra</td>
</tr>
</tbody>
</table>

TABLE I: The courses used to predict student success. The rightmost column is the course name used by the university. Note that the courses in the dataset used the quarter system.

IV. DATA COLLECTION

To measure student success in these courses, anonymous registrar data was collected from 924 CS undergraduate students at California State University East Bay (CSUEB). The courses were taken between 2012-2018 and used the quarter system. Table II shows the demographics of CS students at CSUEB. It is important to note that many of the students...
TABLE II: Student Demographics [16] for CS Department 2017-2018.

At CSUEB are first-generation college students, and many of them also work full or part-time while attending university. While the data in the present work captures student courses taken, grades, and GPA, the data set used in the present work does not include any data for factors such as whether a student was working full or part-time, or whether the student was a first-generation college student.

V. PRELIMINARY DATA ANALYSIS

We start the analysis by examining the performance correlations among the lower division courses. Figure 1, shows the correlation matrix for the first attempt grades for Data Structures (DataS), CS1, CS2, Calculus 1 (Calc1), Calculus 2 (Calc2), and Discrete Math (DMath). The diagonal shows the histogram of the grade for the given course. Interestingly, only a weak correlation is seen between CS1 and Data Structures (0.09). However, we do see a stronger correlation between Data Structures and CS2 (0.27). A surprising observation is that Calculus 2 has a stronger correlation to Data Structures than CS1 or CS2. Perhaps most notable is that even the strongest correlations are fairly weak. Different combinations of interactions were also examined and also showed fairly weak correlations.

While these correlations are weaker than one might expect, there are some possible explanations for this result. As mentioned earlier, the data set used in the present work captures only student academic data and does not capture external factors such as whether a student was working full-time while taking classes. Stronger correlations may be possible if more data were available.

For our first model, we examine a linear regression model with the Data Structures grade as the response variable and the indicator variables from Table III using only performance from CS1 and CS2. Note that the course grade is represented by a number from 0 to 4, with 4 being an "A" or 4.0 grade and 0 being an "F". Unsurprisingly, none of the variables showed coefficients that were statistically significant. In addition, a plot of the residuals (see Figure 2) indicates that there is likely are one or more hidden indicator variables not being captured that are needed to accurately predict Data Structures grades. This could include variables such as how many hours the student works per week, how far the student has to commute, or other factors not present or easily determined with the existing data. These results show that a linear model is not suitable for predicting letter grades in the data structures course, at least not with the current data set.

Intuitively, we expect past performance in classes to offer some predictive measures. Rather than predict grades, we instead look at the probability of passing (achieving a grade of C- or better). Table IV shows several conditional probabilities of passing Data Structures given the student’s performance in past classes. For example, if a student fails CS1 in the past,
we see that the student has a probability of 0.2768 (about 28% chance) of passing Data Structures. We see a similar statistic for CS2 with a 0.2656 probability (about 27% chance) of passing Data Structures if the student previously failed CS1. Unsurprisingly, a student who failed both CS1 and CS2 in the past has an even lower probability of 0.1667 (about 17% chance) of passing Data Structures. Interestingly, Calc1 and Calc2 courses show similar conditional probabilities as compared with the CS1 and CS2 conditional probabilities. Students who fail Calc1 have a 0.2676 probability (about 27% chance) of passing Data Structures. Given a student failed Calc2, the student has a 0.3219 probability of passing Data Structures. If the student failed both Calc1 and Calc2, then the student has a 0.1951 probability (about 20% chance) of passing Data Structures. This initial result shows that early grades can indeed be a useful predictor of student performance in future courses.

VI. MODEL SELECTION

Rather than predict student grades for data structures, we instead choose to predict whether students are able to pass Data Structures. Since we are attempting to predict if a student will pass Data Structures based on past class performance, we can formulate this as a classification problem. In this section, we examine two different strategies for predicting whether a student will pass based on the grades of their previously taken courses. In particular, we examine logistic regression (LR) and support vector machine (SVM) classification models. For our indicator variables, we use the initial grades for CS1, CS2, Calculus 1, Calculus 2, and Linear Algebra. Our response variable is binary indicator whether or not the student would pass Data Structures given their initial grades.

One issue to consider is the different combinations of classes that students can take. Depending on the order that the classes are taken, we would need a model specific to the given permutation. For example, if a student takes CS1 only, then the model predicting the performance of Data Structures would be the single indicator variable using the CS1 grade. Similarly, if a student has taken Calc1 and CS1 but none of the other courses, then we need a model with Calc1 and CS1 indicator variables for prediction. While class prerequisites do reduce the number of possible permutations, this multiple model approach would require numerous models to capture each situation. To avoid this issue, we instead treat each variable as a categorical variable with each grade being a different classification label.

VII. LOGISTIC REGRESSION

The first model we will examine is a logistic regression model, which uses a logistic function to predict a binary value. For this model, we define the logistic function $l$ to be,

$$l = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$$

(1)

where $\beta_i$ are beta coefficients for the parameters of the model and $x_j$ is the value of the indicator variable $j$ for $1 \leq j \leq n$. Using $l$, we can calculate the probability $p$ of binary value of a response variable by calculating

$$p = \frac{1}{1 + e^{-l}}.$$ 

(2)

As previously discussed, instead of treating the class grade as a continuous variable and have a different models for each permutation, we instead interpret the grade as a categorical variable with 13 different categories: F, D-, D, D+, C-, C, C+, B-, B, B+, A-, A, and “did not take”. The advantage of this approach is that it allows for different combinations of classes to be captured by a single model. To construct this model, we create 12 “dummy” variables for each class. Thus, we define the model $L$ as

$$l = \beta_0 + \beta_1 CS1_F + \beta_2 CS1_{D-} + \cdots + \beta_{12} CS1_A + \beta_{13} CS2_F + \beta_{14} CS2_{D-} + \cdots + \beta_{24} CS2_A + \beta_{25} Calc1_F + \beta_{26} Calc1_{D-} + \cdots + \beta_{36} Calc1_A + \beta_{37} Calc2_F + \beta_{38} Calc2_{D-} + \cdots + \beta_{48} Calc2_A + \beta_{49} L_F + \beta_{50} L_{D-} + \cdots + \beta_{60} L_A$$

(3)

where $\beta_i$ are beta coefficients for the parameters of the model and $\beta_0$ is the intercept. $CS1_g$, $CS2_g$, $Calc1_g$, $Calc2_g$, and $L_g$ are 1 if they received grade $g$ or 0 if they did not. If the student did not take the class at all, then, each value is 0. For example, if a student did not take $Calc1$, then for each value of $Calc1_g$ is 0 for $g = F, D-, \ldots, A$. 

TABLE IV: Calculated conditional probabilities of passing Data Structures given different scenarios.

![Fig. 2: The residuals plot shows violation of linearity. The linear trend of residuals indicates there are likely hidden variables not accounted for by the linear regression model.](image-url)
VIII. SUPPORT VECTOR MACHINE

The next classification model tested is a Linear Support Vector Machine (SVM). This model is a supervised learning method that is able to make a classification to specific group given an input vector. Unlike the logistic regression model, it is a non-probabilistic and does not attach a probability to a classification. For this model, we define our training data as

\[ z = (x_1, y_1), \ldots, (x_n, y_n) \]  

(4)

where \( x_i \) is a vector of categorical inputs

\[ (CS1_F, \ldots, CS1_A, \]  

\[ CS2_F, \ldots, CS2_A, \]  

\[ Calc1_F, \ldots, Calc1_A, \]  

\[ Calc2_F, \ldots, Calc2_A, \]  

\[ LF, \ldots, LA) \]

with each position in vector \( x_i \) a 0/1 indicator whether the student received a given grade for a class. This is essentially the same input vector used by the previous LR model. For example, if the student received an A for CS1, then the value of \( CS1_A \) is 1 and all other values of \( CS1_g \) is 0 for \( g = F, \ldots, A \). Again, if the student did not take the class at all, then all the values for that particular class are 0. \( y_i \) is a 1 or -1 label specifying what class \( x_i \) belongs for data point \( i \).

Using the training data in equation 4, we find a boundary line satisfying

\[ w \cdot z - b = 0, \]

(5)

where \( w \) is a vector normal to the line. This boundary line defined by equation 5 allows us to classify new data points.

IX. MODEL COMPARISON

In this section, we examine and compare each model. In particular, we will be examining the ROC (Receiver Operating Characteristics) curve, the AUC (area under the curve), and the confusion matrix. The AUC and ROC curve are typical performance measurements for classifiers as they show how well the model is capable of distinguishing the different classes \[17\]; in our case, we are determining how well the models can determine if a student will pass or fail Data Structures. An AUC value of 1 indicates perfect classification. This metric is particularly useful for comparing two different classifiers. The other metric we use is positive predictive value and false discovery rate, which can be derived from a confusion matrix. The positive predictive value (PPV) is the probability that the student who is predicted to fail data structures does indeed fail the class. The false discovery rate (FDR) is the complement of the PPV \((FDR = 1 - PPV)\).

Figure 3a shows the results of the LR model. We see that the AUC is 0.67 and the PPV is 77% for failing and 51% for passing. This indicates that a student predicted to fail data structures will with 77% probability fail the course. Similarly, it show a student predicted to pass will pass with 51% probability. Figure 3b show the results of the SVM model. For this model, we see the AUC is slightly higher with an AUC of 0.73 as compare with the LR model; this indicates the SVM is a better classifier. Interestingly, the SVM model shows the same PPV as the LR model. The SVM is able to predict with 77% probability if students will indeed fail data structures. However, the SVM is better able to predict students passing data structures with a 57% PPV for predicting if students will pass data structures.

As our aim is to identify students likely to fail, we see that both LR and SVM have the same prediction probability for predicting a student failing data structures. As the SVM does slightly better classifying students who pass and also has a higher AUC, we consider the SVM a better prediction model.

X. MODEL UTILIZATION

In this section, we explore how the SVM prediction model is able to be utilized for advising. We first examine the impact of course repeats on GPA and the seats available for students. Table V shows several statistics of interest. The first column shows the average number of times students attempt a particular course. We see that on average students take CS1 1.143 times. This is significant since it means that approximately 14.3% of the seats in a given semester is occupied by someone repeating the course. We find similar results for CS2 where the average number of repeats is 1.149. Data structures, Linear Algebra, and Calc2 have average repeats of 1.160, 1.160, and 1.172 respectively. The highest average for course repeats is for Calc1 with an average of 1.207. These statistics support that a non-trivial number of course repeats are occurring and reinforce the need to reduce the number of students repeating a course through advising. The second column of Table V shows the average GPA of a course after repeating the course. This is calculated by finding the maximum grade from all attempts and averaging this across all students. The third column shows the average GPA of students who only take the course a single time. We see across all courses that students who need to repeat the course on average will earn a lower grade that students who only take the course once. This implies that even if a student repeats a course, it is likely they will earn a lower grade compared to a student who only needs to take the course once.

Our next analysis examines several different strategies for utilizing the models. We also examine how better advising

<table>
<thead>
<tr>
<th>Course</th>
<th>Average Repeats</th>
<th>Average GPA After Repeat</th>
<th>Average GPA No Repeat</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS1</td>
<td>1.143</td>
<td>2.88</td>
<td>3.09</td>
</tr>
<tr>
<td>CS2</td>
<td>1.149</td>
<td>2.42</td>
<td>2.85</td>
</tr>
<tr>
<td>DataS</td>
<td>1.160</td>
<td>2.40</td>
<td>2.82</td>
</tr>
<tr>
<td>Calc1</td>
<td>1.207</td>
<td>2.34</td>
<td>2.69</td>
</tr>
<tr>
<td>Calc2</td>
<td>1.160</td>
<td>2.23</td>
<td>2.65</td>
</tr>
<tr>
<td>L</td>
<td>1.172</td>
<td>2.29</td>
<td>2.64</td>
</tr>
</tbody>
</table>

TABLE V: This table shows the average number of repeats and the impact repeating classes has on student GPA. The first column shows the average number of repeats per course. The second column shows the average GPA of students if they choose to repeat a course. The third column shows the average GPA of students when they need to take the course only once.
(a) The first plot shows the ROC plot and AUC value for the logistic regression model. The second shows the confusion matrix for the logistic regression model and the positive predictive value.

(b) The first plot shows the ROC plot and AUC value for the SVM model. The second shows the confusion matrix for the SVM regression model and the positive predictive value.

could affect the average pass rate of data structures and the number of seats that would become available. The first proposed strategy is instituting a rule allowing students to repeat introductory courses a maximum of two times. In other words, if a student did not pass an introductory course after two attempts, they would be advised to pursue a different major or not to continue the program. Table VI shows the seats used and potentially saved applying this threshold. The first column shows the number of potentially saved seats. The second column shows the total number of seats occupied for a particular class. For CS1, we found 7 students who took CS1 two or more times meaning 7 seats would be saved in this situation. We found roughly twice as many potential seats in CS2 with 15 saved. Data structures we found 17 potentially saved seats. Calc1, we found the most number of potentially saved seat; we found 18 CS students repeating the class more than twice. For Calc2 and linear algebra, we found 12 and 13 potentially saved seats respectively. For CS1, two more or more times meaning 7 seats would be saved in this situation. We found roughly twice as many potential seats in CS2 with 15 saved. Data structures we found 17 potentially saved seats. Calc1, we found the most number of potentially saved seat; we found 18 CS students repeating the class more than twice. For Calc2 and linear algebra, we found 12 and 13 potentially saved seats respectively. The next to last row of Table VI shows that the maximum number of saved seats in total is 153 (or 3.3% of the total number of seats). Note that the maximum number of saved seats is higher than the sum of the classes individually. This is because students who repeat an earlier class would not take subsequent classes. For example, a student who unsuccessfully attempted CS1 twice would not attempt any other prerequisite courses. The last row shows a potential savings of $244,800 dollars in tuition cost over six years assuming an cost of $1,600 per class, which is a savings of approximately $40K per year. While this first strategy does improve the number of potential seats, it also does not address students who consistently repeat courses. For example, a who routinely takes courses twice also indicates a need for advising.

Our second strategy is expands the first strategy by also taking into account the total number of repeated courses. Table VII shows the results of imposing different thresholds allowed for maximum number of repeated courses while still imposing a two attempt maximum for any lower division course. The first row shows 137 students had to repeat 2 or more courses. If this threshold were applied, then 137 students would need advising and we would potentially save 369 seats (8% of the total seats) possibly saving $590,400; this is arguably too aggressive a threshold and not practical. The second row shows 40 students had to repeat 3 or more courses. Applying this threshold would lead to potentially saving 184 seats (4.0% of the total seats) and saving potentially $294,400; this rule adds 31 additional seats as compared with the two attempt strategy. This threshold seems to be the most reasonable. Thresholds of 4 only shows 8 students meeting this threshold. We also see in this rule only adds 2 additional seats over the two limit strategy and saving only $3,200 additional dollars. Similarly, only 2 students meet the 5 repeat threshold and no additional seats were gained as opposed to the two attempt strategy alone. Therefore, based on this data, we find that imposing a limit of two repeats per course and three repeats maximum across all
introducing courses would be a good first step to help reduce the number of repeated courses. Students at risk of hitting the limits on repeated courses could be flagged in the system and offered additional adviser support.

XI. Future Work

These predictive models offer a useful means to flag struggling students earlier in their academic careers. It is important to note that while the models are able to predict whether a student will fail Data Structures with up to 77% accuracy, the models were less accurate in predicting whether a student will pass Data Structures. These models therefore should be used as tools in combination with other information in order to make the best possible recommendations to students for advising. For example, if a student has a low probability of passing Data Structures given their past performance, an adviser could also look at whether the student is also working full-time, whether the student has access to supporting resources, and other factors in order to provide the best possible guidance.

Another interesting observation is that CS students tended to fail and repeat the calculus and linear algebra courses twice as often as the CS courses. As mentioned earlier, students who failed both calculus I and II had only a 20% chance of passing Data Structures. Future work could investigate additional indicators of success and identify approaches to help struggling students better perform in these classes.

While the focus of the current work was to predict at-risk students early on by focusing on the Data Structures course, future work could also examine whether a similar approach could be applied to the Analysis of Algorithms course and other upper division courses. Additionally, it could be interesting to attempt to capture other variables in the data set, such as whether a student was working full-time or was a first generation student, to attempt to improve the models.

Finally, future work should explore actions advisers and departments can take to better support students based on the predictions and data in this paper. For example, it could be beneficial to explore the impact of offering better access to tutoring or other support, or to implement department policies to more strictly enforce prerequisites and minimum grades for critical courses in CS. By combining the learning in the present work with improved advising and support, CS advisers and departments can make significant progress in improving time to graduate, reducing overcrowding, and improving graduation rates.

XII. Conclusion

In this paper, we explore two different classification based prediction models for predicting CS success and identifying students in need of advising. We show that we are able to identify students with high risk of failing with 77% accuracy. By utilizing these models and the recommended advising approaches, we show that we could potentially reduce the number of repeated courses by 4.0%, make 184 additional seats available for other students, and save approximately $250K for the 924 students in the study.

REFERENCES


