Creating a Student Success Predictor Using Statistical Learning

Rex Gandy
Provost
Austin Peay State University
gandyr@apsu.edu

Daniel Kasper
Data Scientist
Decision Support and Institutional Research
Austin Peay State University
kasperd@apsu.edu

Andrew Luna
Executive Director
Decision Support and Institutional Research
Austin Peay State University
lunaa@apsu.edu

Abstract: Using decision tree models, this study utilizes the strength of statistical learning to predict student success during the first two years of college. The model pulls from data easily acquired from the student information systems of most colleges and universities and establishes probabilities of success based upon key student components. As a student progresses beyond the first year, new data is used to update success probabilities based on the student's academic performance. Results from the models in this study indicate that both the assignment and ranking of student success probabilities were strong.

keywords: statistical learning; predictive analytics; student success

Introduction

Statistical or machine learning has been used successfully in business and medicine to determine outcomes based on large amounts of data. The basic concept behind statistical learning is to use past data to train a model to learn patterns and configurations within the data in order to understand or predict behavior. Once a model has been established, computer processors can then quickly scan through large, live data sets to pick up these patterns and make predictions that are regularly more accurate than those of traditional statistical models (Greenwell, 2017).

These machine learning models differ from traditional statistical models in that they only look at patterns of data and do not make inferences, find relationships, or structure models based on normalized assumptions.

Using decision tree models, this study utilized the strength of statistical learning to predict student success, including fall-to-fall retention, during the first two years of college. The model pulls from data easily acquired from the student information systems of most colleges and universities and establishes probabilities of success based upon key student components. As students progress beyond the first year, new data are used to update success probabilities based on their academic performance. From the beginning of their first year through the second year, a probability number will be generated from the model and assigned to each new student. As student tracking continues, resources and intervention strategies will also be developed for those students deemed by the model to be marginal or below success thresholds. It is believed that such intervention strategies will positively impact student retention and success during the first two years.

Established during the 1980s, machine learning is an area of computer science where prediction systems were built from system-learned data rather than the use of specifically programed instructions. With the onset of datasets exponentially increasing in size to the point that regular data analysis was difficult if not impossible to perform, computer scientists decided to utilize the power of the newer and faster computers to allow the computer itself to develop the models using one of many algorithms (James, Witten, Hastie, & Tibshirani, 2013).

Decision trees are a class of machine learning tools that build top down tree-like models based on the attributes of a given set of data. More importantly, decision trees are good predictive modeling techniques that are used to classify or categorize given data objects on the basis of a previously generated learning model (Bhargava, Bhargava, & Mathuria, 2013). The decision tree can predict either a categorical or continuous response variable. Specifically, the decision tree is made up of nodes in the data from which splits occur. The initial split is made using the predictor variable, creating two or more nodes. At this point, the nodes are continually split until no more divisions can occur. It is at these terminal nodes that predictions are made.

This recursive process is tedious and consumes processing power. Within this process the computer sifts through large data sets to find patterns through general-purpose learning algorithms and training (historic) data. Once patterns have been set by the training data, predictions can then be made on new data.

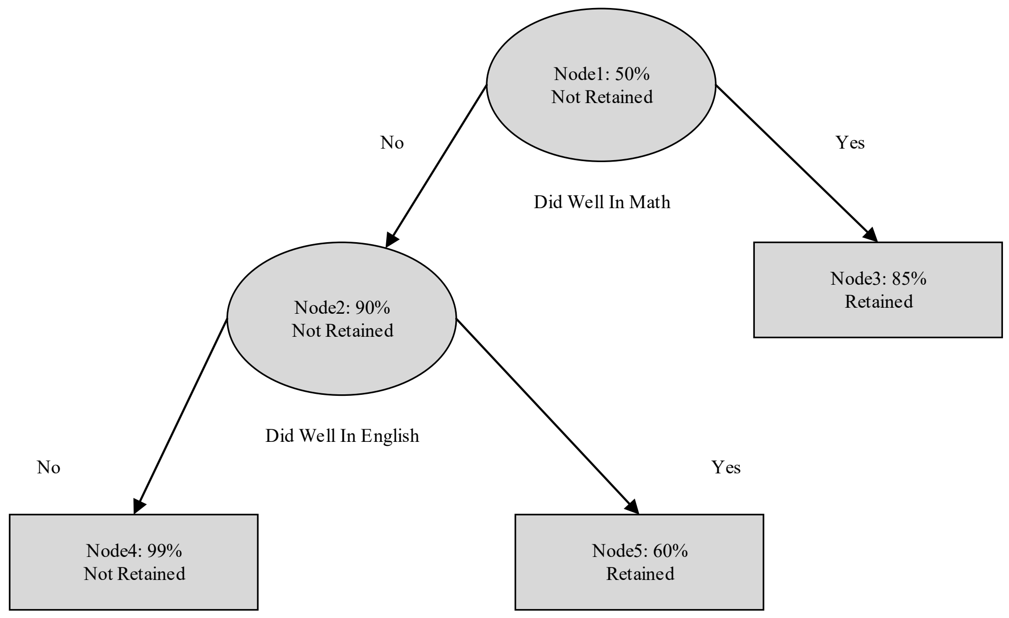

Figure 1: Example of Decision Tree Methodology

To understand the method of using decision trees, it may be helpful to think of a common organizational chart where the highest position within the organization is at the top and the positions ancillary to it are set below by order of importance. An example of a simple decision tree is provided in Figure 1. In the first (root) node within this example, it is determined that 50% of a group of students were retained while the other 50% were not. When observing how well the student did in math the model created another degree of separation. Nodes 2 and 3 produce better classifications than Node 1 to predict students who are not retained. When the model looked at how well the student performed in English, an even better prediction was created. Those nodes that do not have any descendant nodes are considered terminal nodes and are called leaves; in Figure 1 these leaf nodes are expressed as rectangles while the other nodes are expressed as ovals. In this example, 99% of the students who performed poorly in both math and English were not retained in school (Node 4) while 60% of the students returned who did poorly in math but not poorly in English (Node 5). A total of 85% of students who did well in math were retained (Node 3).

Machine learning is very useful when datasets are exceptionally large, because computers can quickly isolate data patterns without getting bogged down into statistical inference or meeting general statistical assumptions.

Statistics, whether descriptive or inferential, rely on the measures of central tendency in order to determine the stability (normality) of a set of data. Therefore, statistical models tend to be sensitive to non-normal variations in the dataset. This sensitivity to non-normalized data is the main reason statistical models are built upon prescriptive assumptions. These assumptions for traditional linear regression include:

1. Linear relation between independent and dependent variables,
2. Homoscedasticity (variance around the regression line is the same for all predictor variables),
3. Mean of error at zero for every dependent value,
4. Independence of observations, and
5. Error should not significantly deviate from a normal distribution for each value of the dependent variable.

Furthermore, unlike the traditional statistical model where the function is estimated:

$$\text{Dependent Variable (Y)} = f\ \text{(Independent Variable)} + \text{Error Function}$$

Machine learning finds pockets of X in n dimensions (where n is the number of attributes) where occurrence of Y is significantly different:

$$Output\ (Y) - -\rightarrow Input\ (X)$$

Therefore, machine learning models do not use relationships or inference to make predictions (James, et al., 2013). Rather, they scroll through large amounts of data to find patterns in relation to a particular outcome.

Traditional statistical models, however, may be more useful and adaptive when the population significantly changes over a period of time. In machine learning, training (or historical) data is used to develop a prediction model which can be reasonably accurate if the population does not shift. If the population does significantly change, the machine model will be ineffective in its predictive power (James, et al., 2013).

Machine learning can be particularly attractive to enrollment managers because they can be adapted to fluctuating environments and data as time, policies, and students change. The models are based on finding patterns between student characteristics and retention, and the predictive value of them ultimately depends on whether these patterns continue to occur in the future. However, since machine learning depends on being trained by the researcher, as circumstances change, so can the models.

The literature is rich with examples of how machine and statistical learning models can be used in higher education. Goyal and Vohra (2012) supported using machine learning and data mining within higher education to improve student performance, increase retention and graduation, and help to better manage financial resources within the institution. Even though higher education data systems are significantly smaller than those in business and industry, the authors contend that, once machine learning algorithms have been created, they can be successfully used with On-Line Analytical Processing (OLAP) data that are stored within college and university data systems to create predictions on up-to-date data. Kumar & Vijayalakshmi (2011) and Hamoud, Hashim, and Awadh (2017) used learning models, specifically decision trees, to predict student performance in higher education.

Likewise, Herzog (2006) studied student retention and degree completion time utilizing decision trees, neural networks, and regression. It was determined that data-mining tools offered distinct advantages over tradition statistical models, particularly when working with large data sets. Likewise, Luan (2002) demonstrated the use of decision trees in predicting the transfer of community college students to four-year institutions.

Methodology

Within the arsenal of predictive analytics, machine learning is a powerful tool to rapidly scan large sums of data in order to detect patterns that the researcher previously established within the model. The noteworthy attributes of these models is that, once patterns have been established with existing historical data, these models can quickly scan large amounts of new data in order to predict a phenomenon.

This paper focuses on the use of trees and neural networks and how they can be used to create student success predictors at a southeastern masters/comprehensive public university. Decision trees are comprised of splits to the data. Trees start with training or historical data residing in the first node or branch. An initial split is made using a predictor variable, separating the data into two more branches or nodes. More splits can be made on these nodes until the terminal node is reached where no more splits are made.

Three related prediction problems were examined: Predicting fall-to-spring retention of first-time, full-time freshmen using demographic data, predicting fall-to-fall retention of first-time freshmen using demographic data, and predicting fall-to-fall retention of first-time, full-time freshmen with the addition of first fall end-of-term coursework data.

Variables

The data used in this study came from the institution's student data system. For the first two problems, 50 variables were used from the student's first semester before the first semester's final grades become available. These variables included basic information such as gender and race, academic information, economic information, information about status as a student at the university, high school information, information about the student's appointment date for orientation, and general information about the county and zip code of the student's permanent residence. For the purposes of conciseness, the above 50 variables are referred to as the students' demographic data to distinguish it from end-of-term data.

Note that there is multicollinearity (a high degree of correlation between the predictor variables), which is an issue for inference but less so for prediction, as long as the same pattern is present in the data for which a prediction is desired. Because the decision tree algorithms essentially perform their own variable selection during the fitting process (selecting one variable at each split), multicollinearity is not an issue for tree-based models the way it is for linear models.

In the third problem (predicting fall-to-fall retention of first-time, full-time freshmen with the addition of fall end-of-term data), the above pieces of information were used as predictors as well as the following variables: 1st fall semester undergraduate GPA, 1st fall semester credit hours attempted, grades for 10 courses that first-semester freshmen commonly have taken, and averages of each student's grades for seven categories in the university's general education requirements.

Before constructing the models, the data for the 2013-2017 first-time, full-time freshman cohorts were divided into three groups: a training set and two holdout sets. The training set was used to train the models. The first holdout set consisted of a random sample of 1,000 students from the 2013-2016 cohorts. The training set consisted of the remaining 4,350 students in the 2013-2016 cohorts. The second holdout set consisted of the 2017 cohort (1,755 students).

The purpose of the first holdout was set to assess the ability of the models to rank students and to assess whether the probabilities were well-calibrated. Note that there is a somewhat subtle distinction between these two properties of a model: A model could be very good at ranking whether students are more or less likely to return, yet produce probabilities that are generally too high, too low, too close to the extremes, or not extreme enough. Such a model is said to be poorly calibrated, and can be re-calibrated using a data set for which the outcomes are known (such as this first holdout set) without having to refit the entire model.

The second holdout set (2017 cohort data) was reserved to evaluate the quality of calibrations produced using the first holdout set. This is because it would not be appropriate to evaluate the quality of a calibration using the same data used to generate the calibration. However, the models were deemed to give sufficiently well-calibrated probabilities without re-calibration, so this second holdout set was not needed for this purpose.

Algorithms

According to Vandamme, Meskens, and Superby (2007), the type of algorithm used is instrumental in determining how the data will be split. The three machine learning algorithms used for prediction were conditional inference forest (cforest) models, Extreme Gradient Boosting (XGBoost) models, and neural network (nnet) models from the corresponding R packages, party (Hothorn, Hronik, & Zeileis, 2018), xgboost (Chen et al., 2018), and nnet (Ripley & Venables, 2016).

Using ensembles or resampled data for each prediction or classification in a decision tree, otherwise known as boosting and bagging, may improve model prediction by as much as 80% over standard algorithms (Opitz and Maclin, 1999). The XGBoost and cforest models used are both ensembles of many individual trees, yet there are differences between the two model types. One difference is the manner in which splits are generated in the two models. The cforest model is a forest of conditional inference trees, and conditional inference trees differ from more traditional decision tree algorithms in the measure used to generate the splits. Essentially, any tree algorithm needs some measure to determine whether it is preferable to split on one variable versus another. Conditional inference trees use a statistically-based measure for this purpose that has theoretical benefits over the node impurity-based information measure used in more traditional trees (Hothorn et al., 2018).

The other major difference between the cforest and XGBoost models is the relationship, or lack thereof, between the individual trees in the ensemble. In the cforest models, there is no particular relationship between the individual trees. The idea is that combing the output from a collection of trees with variation will provide better predictions than just one tree. The cforest algorithm guarantees that the generated trees are not all identical to each other by using a random fraction of data to train each tree and by randomly sampling variables at each node (to limit the variables available to be used for splitting). More information about the cforest algorithm can be found in the publications of the authors of the package (Hothorn, Buehlmann, Dudoit, Molinaro, & Van Der Laan, 2006; Strobl, Boulesteix, Kneib, Augustin, & Zeileis, 2008; Strobl, Boulesteix, Zeileis, & Hothorn, 2007).

XGBoost, known as extreme gradient boosting, involves sequentially building a series of relatively simple models, which all contribute to the output of the model. In each step, a new simple model is added to the collection of contributing of models in order to improve on the shortcomings of the collection. In this study, each of these simple models is a single tree. Because each tree is built specifically to improve on the preceding ones, the generation of a single tree within an XGBoost model depends on all of the trees before it. Because boosting generates very flexible models, it has the potential to generate models that overfit the training data; for this reason, the XGBoost algorithm has a number of hyperparameters to control overfitting ("Notes," 2016). XGBoost has been highly successful in machine learning competitions (Chen & Guestrin, 2016).

The other type of model used was an artificial neural network model, which is not tree-based. Neural networks are somewhat analogous to the brains of animals. A neural network consists of a number of nodes with connections between them, which are similar to the neurons in the brain. Like neurons, the nodes take input from other nodes, aggregate that input, and send the output to other nodes. In this way, the neural networks are able to approximate complex mathematical functions, given training data with known result values. Neural networks have been successful in complex tasks such as image recognition.

Typically, neural networks are structured into layers: they have one input layer (with each node corresponding to one of the input variables), one output layer (with one or more nodes corresponding to the response of the model), and one or more hidden layers in between. The neural network fitted by the nnet package had exactly one hidden layer and was a feedforward neural network, meaning that the connections only go in one direction, without cycles in the network. Each node takes the inputs of the previous layer, aggregates them, and then outputs to the next layer.

The connections between the nodes have weights that affect their contribution during aggregation. These weights are the parameters of the neural network, much like the coefficients in linear regression, and it is possible to express a neural network as an equation of the inputs and connection weights. When a neural network is being trained, these weights are varied in an iterative process to better fit the data. The weight decay hyperparameter in the nnet algorithm was used in order to prevent overfitting.

Hyperparameters are values that must be chosen before a model is trained and affect how the model-fitting process proceeds; they are the knobs and levers of machine learning algorithms. For example, the XGBoost algorithm has a numeric hyperparameter that defines the maximum depth that can be attained by a tree, preventing additional splits from occurring in a tree when that depth is reached, thereby affecting the complexity of each tree. Similarly, the nnet algorithm has a hyperparameter that defines the number of nodes in the hidden layer. Hyperparameters contrast with parameters, values that are set during the fitting process and define the resulting model, such as the coefficients in a linear regression and the connection weights in a neural network. One method of selecting good hyperparameters for a task is tuning them via cross-validation. The hyperparameters of each of the three types of model were tuned via ten-fold cross validation, using the mlr R package (Bischl et al., 2016). In 10-fold cross-validation, the training data is split into ten groups and then a model using a single combination of hyperparameters is built using the data from nine of the groups, and this model's performance is evaluated on the 10th group. This process is repeated for each possible choice of 10th group, using the same combination of hyperparameters. The 10 evaluation results are combined to produce one evaluation measure for that combination of hyperparameters. This is repeated on the 10 groups for each combination of hyperparameters to be evaluated. The hyperparameters with the best evaluation measure are then selected to build a model using the entire training data set.

In tuning hyperparameters, there are two choices to be made: the combinations of hyperparameters to be evaluated and the evaluation measure. A traditional method of generating hyperparameter combinations involves choosing a discrete number of values for each hyperparameter, e.g., at equally spaced intervals. The possible combinations of these values are evaluated to find the best combination. These combinations define a multidimensional grid with a dimension for each hyperparameter, so the method is called a grid search. Unfortunately, the size of the space to be searched increases exponentially with the number of hyperparameters to be tuned, making it computationally expensive, and the method also misses the spaces between the points on the grid.

Instead of a grid search, the mlrMBO R package (Bischl et al., 2018) was used to tune the hyperparameters via model-based optimization. In this method, the performance of previous combinations of hyperparameters affects the combination of hyperparameters to be evaluated next.

This leaves the choice of evaluation measure. For these models, logarithmic loss was used as the evaluation measure during cross-validation. Logarithmic loss is closely related to the concepts of cross-entropy, logistic loss, information content, self-information, surprisal, negative log-likelihood, and the logarithmic scoring rule in fields linked to machine learning. Logarithmic loss is given by

$$\text{logarithmic loss} = \text{mean}(\ln(p_i))$$

for the sample being evaluated, where p_i is the probability assigned by the model to the true outcome for each observation in the sample (Bischl et al., n.d.). The desire is for the logarithmic loss to be small.

The advantage of using logarithmic loss over a more generally familiar performance measure, accuracy, is that while accuracy only takes into account whether a prediction was correct or incorrect, logarithmic loss additionally takes into account the reasonableness of the predicted probability. With logarithmic loss, incorrect predictions will be penalized more heavily the more brash they are in their wrongness. Likewise, logarithmic loss favors a model that gives correct predictions and assigns high probabilities to those correct outcomes over a model that gives correct predictions but assigns only modest probabilities to those outcomes. Accuracy does not differentiate between these cases.

Variable Transformations

Because these machine learning algorithms lack the linearity requirement of traditional linear regression models, it is not necessary to transform variables to achieve linearity.

Generally, the variable transformations used depended on the model. The XGBoost and nnet algorithms are only able to handle numeric predictor variables, so, when training via these two algorithms, categorical variables had to be converted. Binary categorical variables were converted to numeric indicator variables. Categorical variables with more than two categories were coded via 1-of-n coding, also known as one-hot encoding, where every possible value of the categorical variable is given its own dummy variable, each dummy variable taking on a value of 0 or 1 depending on whether the student was a member of that category. Note that this differs from the traditional reference coding used in linear models, where there is a reference category that does not have an associated dummy variable.

Because the mlr package was used to manage the model tuning process, the outcome variables for all models were coded as factors. Instead of using R's default action, which reads text data as factor variables and automatically chooses the levels, categorical predictor variables were loaded into R as character variables and then converted to factors, with the levels of these factors explicitly hard-coded to ensure consistency in application to new data. Student orientation appointment dates were converted to a number relative to the Tuesday before the start of the corresponding fall semester, in order to align days of the week for data from different years.

Imputation (assignment of values to missing values of the predictor variables) was performed on all variables with missing values for nnet and on categorical variables with missing values for cforest. The XGBoost algorithm natively handles missing values, so imputation was not performed in constructing the XGBoost models. Much of the transformations and data cleaning were done using functions in the mlr package and the dplyr package (Wickham, Francois, Henry, & Miller, 2018). The ggplot2 package was used for graphs (Wickham, 2016).

Results

Results of the three models indicated very similar results. Therefore, to make reporting of the results less cumbersome, the XGBoost model only will be discussed since it produced slightly better-calibrated probabilities than the other two models, particularly for the two prediction problems involving fall-to-fall retention.

The primary method used for evaluating model performance was the receiver operating characteristic curve (ROC curve). The ROC curves give insight into how well the models are able to rank students by chance of retention. ROC curves were so named because of their use during World War II to analyze the ability of radar operators to distinguish whether a signal represented a true threat (Institute of Medicine (US) and National Research Council (US) Committee on New Approaches to Early Detection and Diagnosis of Breast Cancer, 2005).

The construction of a ROC curve involves applying a model to a separate data set and using the model's predictions to classify students into "retained" or "not retained" groups. This classification is based on whether the predicted probability of retention from that model exceeds some threshold value, and the process is repeated with a large number of thresholds. For each threshold, a true positive rate (the fraction of returning students who were correctly classified as returning) and a false positive rate (the fraction of non-returning students who were falsely classified as returning) is calculated. The true positive rates for the model are plotted on the y-axis against the corresponding false positive rates on the x-axis, and the points are connected to generate the ROC curve. A greater Area Under the ROC Curve (AUC) indicates that the model is better at ranking students. The AUC may be interpreted as the probability that a randomly selected student who was actually retained would be assigned a higher chance of retention than a randomly selected student who was not actually retained (Hajian-Tilaki, 2013). For example, an area of 0.7 would correspond to 70% probability of ranking these two randomly selected students in the correct order. ROC curves were generated by applying the models (which were trained on the training data set) to obtain predictions for the separate set of students in the first hold-out set.

The ROC curves and AUC values for the three modeling algorithms (XGBoost, nnet, and cforest) were very similar when comparing ROC curves for the same prediction problem, which indicates that the three algorithms performed similarly at ranking the students by chance of returning. The ROC curves for the three XGBoost models are displayed in Figure 2.

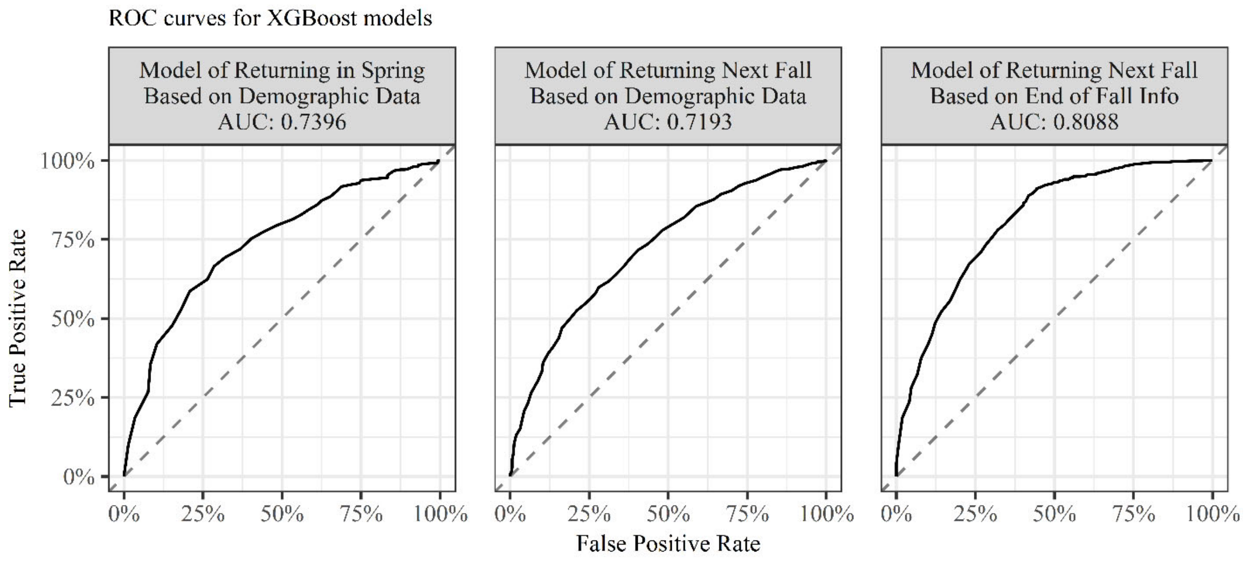

Figure 2: ROC curves for the three XGBoost models with one plot per prediction problem. The "Model of Returning Next Fall Based on End of Fall Info" incorporates both the end of fall information and the demographic data.

From the AUC values, it is clear that the addition of the end-of-term data resulted in an 8-9% improvement in the ability of the model to rank the students by fall-to-fall retention probability.

While ROC curves assess how well the models perform at ranking students, they do not give any insight into how accurate the model's retention probabilities are. For assessing the quality of the probabilities returned by the model, reliability plots were used.

The construction of reliability plots involved sorting the students in the holdout set by predicted probability, dividing them into 10 groups of 100 students based on those predicted probabilities. Each group's proportion of students who returned is plotted against the group's mean predicted probability of returning. For a perfectly calibrated model, we would expect these predicted probabilities and observed frequencies to closely match, such that all of the plotted points would fall on a diagonal line.

The reliability plots for the three XGBoost models are displayed in Figure 3.

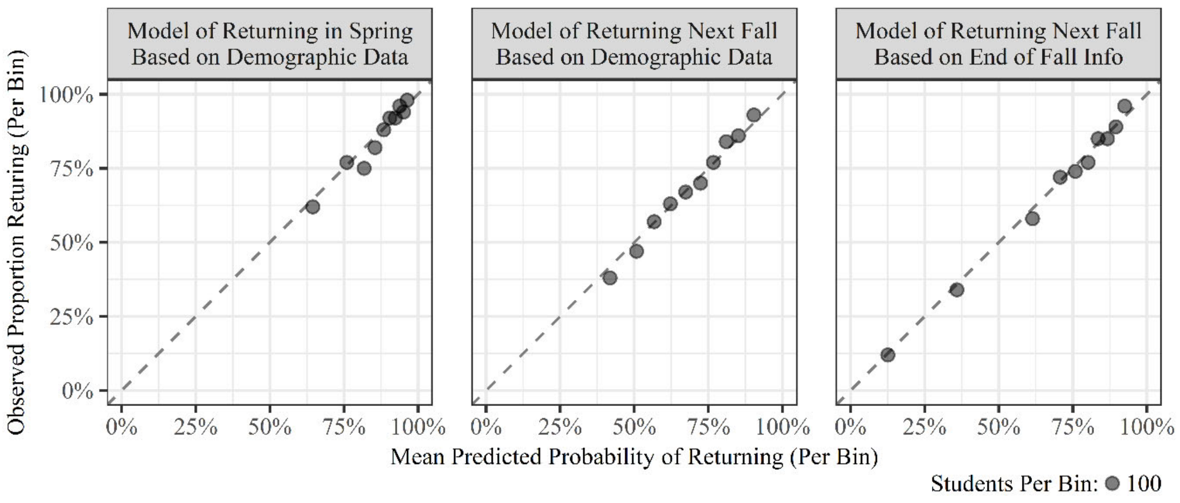

Figure 3: Reliability plots for the three XGBoost models with one plot per prediction problem. The "Model of Returning Next Fall Based on End of Fall Info" incorporates both the end of fall information and the demographic data.

Because fall-to-spring retention is relatively high, the plotted points are clustered near the top right hand corner of the graph for the model predicting fall-to-spring retention.

The rightmost two plots assess the reliability of the two fall-to-fall prediction XGBoost models. For predicting fall-to-fall retention, the inclusion of fall end-of-term data improves the ability of the model to assign low retention probabilities to the students, as seen by the fact that there are now points for low probabilities on the diagonal line in the rightmost plot. Part of this improvement in prediction ability may be due to students who left during the course of their first Fall Semester, affecting their end-of-term academic information.

Conclusions

Identifying the most appropriate model was seminal to the student success predictor project. In theory, a model could be created to adequately identify students who may falter during their first two years of school. In practice, this study was able to demonstrate the creation of a series of models that can reasonably predict success probabilities of entering freshmen and adjust those probabilities as they progress beyond the freshman year.

The models used were trained on historical student data. Now that the models have been established, data from the new freshman class will be entered and probability numbers calculated. Once freshmen complete the first semester, a new model will be used and success probabilities recalculated, taking into consideration how successful students performed in various college-level courses.

The probability numbers calculated from the model can range from 0% probability of success to 100% probability of success (Figure 4). In reality, the actual success probability numbers will fall between these two extremes. With these success probabilities calculated, the university administration can establish a Threshold of Success where incoming students who fall above the threshold require less help or intervention.

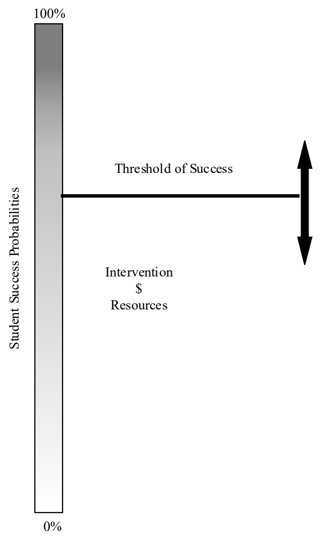

Figure 4: Student success probabilities and the Threshold of Success.

Below that threshold, however, lie opportunities for administrators to assign significant resources and intervention strategies in order to increase the success rate among these types of students. Based on the results of the model, the institution has plans to implement strategies to support students who fall below the threshold. Such strategies include:

- The institution is currently in the process of hiring new academic counselors to be in place by fall semester.
- The institution is currently partnering with College Possible to provide additional resources to new freshmen.
- The institution plans to update or create special classes and/or learning communities.
- The institution is currently working on plans to encourage and increase freshman participation and socialization within the campus community, especially among those students who fall below the threshold.

Also, as shown in Figure 4, the Threshold of Success can be adjusted as iterations of the student success predictor are run for each new freshman class and the success probabilities of the model are calibrated with actual data on student success. Therefore, if an institution fosters a more nurturing environment where marginal or conditional students have greater support and are encouraged to enroll, the Threshold of Success indicated may need to be lowered slightly. Conversely, if an institution raises its admission standards and becomes a more academically challenging institution, administrators may need to raise the Threshold of Success in order to account for an increase in at-risk students.

As iterations of this model are run and success probabilities calculated, the institution will continue to compare results of the models to actual student success and test whether institutional interventions can significantly increase the success of marginally and below average students.

References

Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Jones, Z., … Schratz, P. (n.d.) *Implemented performance measures.* Retrieved from https://mlr.mlr-org.com/articles/tutorial/measures.html

Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., … Jones, Z. (2016). mlr: Machine learning in R. *Journal of Machine Learning Research, 17*(170), 1-5. Retrieved from http://jmlr.org/papers/v17/15-066.html

Bischl B., Richter J., Bossek J., Horn D., Thomas J., & Lang M. (2018). *mlrMBO: A modular framework for model-based optimization of expensive black-box functions*. Manuscript submitted for publication. Retrieved from https://arxiv.org/pdf/1703.03373v3.pdf

Bhargava, G.S., Bhargava, R., & Mathuria, M. (2013). Decision tree analysis on J48 algorithm for data mining. *International Journal of Advanced Research in Computer Science and Software Engineering, 3*(6).

Chen, T. & Guestrin, C., (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, 785-794. doi:10.1145/2939672.2939785

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., … Li, Y. (2018). xgboost: Extreme gradient boosting [Computer software]. Retrieved from https://CRAN.R-project.org/package=xgboost

Goyal, M. & Vohra, R. (2012). Applications of data mining in higher education. *International Journal of Computer Science, 9*(2).

Greenwell, B. M. (2017). pdp: An R package for constructing partial dependence plots. *The R Journal, 9*(1), 421-436. doi:10.32614/RJ-2017-016

Hajian-Tilaki, K. (2013). Receiver operating characteristic (ROC) curve analysis for medical diagnostic test evaluation. *Caspian Journal of Internal Medicine, 4*(2), 627-635. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3755824/

Hamoud, A., Hashim, S., & Awadh, A. (2018). Predicting student performance in higher education institutions using decision tree analysis. *International Journal of Interactive Multimedia and Artificial Intelligence*, (in press). Retrieved from http?//ds.doi.org/10.9781/ijimai.208.02.004

Herzog, S. (2006). Estimating student retention and degree-completion time: Decision trees and nural networkds vis-à-vis regression. *New Directions for Institutional Research*, (131), 17-33. San Francisco: Jossey-Bass. Retrieved from https://doi.org/10.1002/ir.185

Hothorn, T., Buehlmann, P., Dudoit, S., Molinaro, A., & Van Der Laan, M. (2006). Survival ensembles. *Biostatistics, 7*(3), 355-373. doi:10.1093/biostatistics/kxj011

Hothorn, T., Hornik, K., & Zeileis, A. (2018, August 8). party: A laboratory for recursive partitioning [PDF file]. Retrieved from https://cran.r-project.org/web/packages/party/vignettes/party.pdf.

Institute of Medicine (US) and National Research Council (US) Committee on New Approaches to Early Detection and Diagnosis of Breast Cancer (2005). Appendix C, ROC analysis: Key statistical tool for evaluating detection technologies. In J. E. Joy, E. E. Penhoet, & D. B. Petitti (Eds.). *Saving women's lives: Strategies for improving breast cancer detection and diagnosis.* Washington, DC: National Academies Press. Available from: https://www.ncbi.nlm.nih.gov/books/NBK22319/

James, G., Witten, D., Hastie, T, & Tibshirani, R. (2013). *An introduction to statistical learning with applications in R*. New York, NY: Springer.

Luan, J. (2002). Data mining and its applications in higher education. In A. M. Serban and J. Luan (Eds.), *Knowledge management: Building a competitive advantage in higher education*. New Directions for Institutional Research, no. 113. San Francisco: JosseyBass.

Kumar, A., & Vijayalakshmi, M. (2011). Efficiency of decision trees in predicting student's academic performance. *Computer Science & Information Technology, 2*, 335-343.

Notes on parameter tuning (2016). Retrieved from https://xgboost.readthedocs.io/en/latest/tutorials/param_tuning.html

Opitz, D., & Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research, 11*, 169-198.

Ripley, B. & Venables, W. (2016). nnet: Feed-forward neural networks and multinomial log-linear models [Computer software]. Retrieved from https://CRAN.R-project.org/package=nnet

Strobl, C., Boulesteix, A., Kneib, T., Augustin, T., & Zeileis, A. (2008). Conditional variable importance for random forests. *BMC Bioinformatics, 9*(307). doi:10.1186/1471-2105-9-307

Strobl, C., Boulesteix, A., Zeileis, A., & Hothorn, T. (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics, 8*(25). doi:10.1186/1471-2105-8-25

Vandamme, J., Meskens, N., & Superby, J. (2007). Predicting academic performance by data mining methods. *Education Economics*, *15*(4), 405-419.

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis [Computer software]*. New York: Springer-Verlag.

Wickham, H., François, R., Henry, L., & Müller, K. (2018). *dplyr: A grammar of data manipulation [Computer software].* Retrieved from https://CRAN.R-project.org/package=dplyr