Using online textbook and in-class poll data to predict in-class performance

Noah Hunt-Isaak
Dickinson College
noahhuntisaak@gmail.com

Peter Cherniavsky
George Mason University
pcherniavsky@gmail.com

Mark Snyder
George Mason University
msnyder14@gmu.edu

Huzefa Rangwala
George Mason University
rangwala@cs.gmu.edu

ABSTRACT

National failure rates seen in undergraduate introductory CS courses are quite high. In this paper, we develop a predictive model for student in-class performance in an introductory CS course. The model can serve as an early warning system, flagging struggling students who might benefit from additional support. We use a variety of features from the first few weeks of the course such as scores on assignments, interaction with the online textbook, and participation with the in-class polling system in order to train our models. We compare the performance of a number of machine learning algorithms on predicting final exam scores as well as final course grade. We find that the Support Vector Machine and AdaBoost are the most effective, and that we can achieve increasingly accurate predictions as we use data from further into the course. The regression coefficients give us insights into which features are most correlated with student success, suggesting that certain types of assignments are more indicative of learning than others.

Keywords

education data mining, performance prediction, early warning system

1. INTRODUCTION

An enduring challenge in higher education is student dropout. National studies [1] have reported a relatively stable average six-year graduation of approximately 60% over the past decade. The problem is acute in STEM (Science, Technology, Engineering and Mathematics) fields where a well-trained and educated workforce is essential for national growth and economy. As the volume and variety of data collected in both traditional and online university offerings continue to expand, educational data mining [1, 20] provides the promise to assist students and improve overall student retention.

Noah Hunt-Isaak, Peter Cherniavsky, Mark Snyder and Huzefa Rangwala "Using online text books and in-class quizzes to predict in class performance" In: *Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020)*, Anna N. Rafferty, Jacob Whitehill, Violetta Cavalli-Sforza, and Cristobal Romero (eds.) 2020, pp. 438 - 443

Close to 30% of the students who enrolled in CS 112, the introductory CS course at George Mason University, during the Fall 17 and Fall 18 semesters failed the course. This is close to the national average pass rate in CS1 courses found by Watson and Li [23].

The focus of this paper is to develop a model for predicting student performance in a course that they are currently enrolled in based on performance in the first few weeks of the course. An accurate predictive model may serve as an early warning system which would alert the professor to students who are struggling, at which point additional support could be provided. Attempting to make predictions too early on in the course would result in inaccurate predictions, while waiting too long will mean it is too late to take any preventative action. Howard et al. suggest that the optimal time to employ such early warning systems is right around the midway point of the course, as this provides a good balance between reasonable predictive accuracy while still allowing time to make corrective changes [6].

In this paper we extract features that capture student behaviors such as consistency, dedication, and grit, and use these alongside the gradebook to make accurate predictions as early as possible. We investigate various machine learning (ML) regression algorithms including Ridge, Lasso, Elastic-Net, Support Vector Machine, AdaBoost, Gradient Boosting, Bagging, and Random Forest. Each algorithm is tuned by applying a grid search to the parameters space. We predict both final exam scores and final course scores, comparing the different models' prediction performance on both.

Finally, we examine the coefficient weights of the most successful models to determine which features are the most significant in making predictions. Discovering patterns which seem to help or hurt students could help the professor to better structure the course in the future, as well as discover more general trends would could be applied elsewhere.

2. PRIOR WORK

A number of attempts have been made at developing grade prediction models and early warning systems similar to what we wish to accomplish here. Most commonly, this has been done in the context of massive open online courses (MOOCs) such as in Ren et al. [19]. Li et al. [16] made an early

prediction model for a blended course. They had access to homework and test scores from the first 6 weeks of the course and used this data to predict students' final letter grades. However, they were only able to slightly improve upon a base model of predicting all A's (47.8% accuracy) by using an SVM (51.4% accuracy). This was probably a result of the course being too easy, which resulted in a fairly uniform data set. Elbadrawy et al. [11] looked at predicting performance on activities within a course, also using multi-regression models. They took into account a combination of features including demographic information, historical performance and course interaction data from the LMS. Another study done by Nam and Samson [17] investigated the impact of student behavioral signals in early warning system predictions. Costa et al. [4] investigated the effectiveness of four algorithms in predicting student failure in two computer science courses along with the the effect of preprocessing data and fine tuning the algorithms on their performance. The preprocessing and fine tuning was found to improve performance for the most part. Ren et al. [18] developed Additive Latent Effect Models to predict student performance in a future course. It used factors from the student, course, and instructor for the prediction. The model performed better than all the baselines. Crossley et al.[5] did a study that used click-stream, language, and demographic data to predict the performance of elementary school students in a math course. The click-stream and language data was from an online math tutoring system. They found that the linguistic, click-stream, and demographic factors explained 14% of the variance in the math score and random factors of the student explained 30% of the variance in the math score. Elbadrawy et al. [10] developed a personalized linear multiple regression (PLMR) and other models to predict grades in a future course and grades in a future assignment within regular courses and a massive open online course (MOOC). Matrix factorization (MF) and PLMR outperformed traditional models in predicting student success in a future course and PLMR was useful in predicting grades within regular courses and a MOOC.

Several papers have covered the usage of zyBooks in CS courses, the online interactive textbook used in the course here. Most students complete the textbook [8, 7, 9]. The relationship between completion rate of the textbook and percentage of the grade that the textbook work is worth levels off at a certain point [8, 9]. The acquisition rate of zyBooks is higher than traditional textbooks [7]. Students mostly do not cheat when using zyBooks [8, 9]. Students' use of zyBooks is stable throughout the semester [7].

Unlike most of the prior studies which have been done in this area, we are exclusively using data from within the context of the course itself (i.e. we don't consider any student demographic or background information). We do, however, use a wider range of interaction data for students.

3. DATASET

Our data is taken from two sections of George Mason's introductory CS course, CS112. CS112 is taught in Python and covers a range of basic programming concepts including variables, conditionals, functions, loops, dictionaries, files, classes, and recursion. The courses were taught during the Fall semesters of 2017 and 2018, for an overall enrollment of 1,197 students. It includes each student's grade book from the semester, including scores on homework, labs, projects, and tests. The grade book also contains additional information such as whether or not the student was flagged for an honor code violation and how many of their allotted late submissions were used. CS112 was taught with an accompanying interactive online textbook, zyBooks, from which we have submission logs. Additionally an online polling system, Pytania [21], was used routinely during lectures. The final grade composition for the course was computed as follows:

Category	Percent	Notes
Projects	40%	drop 1 lowest
Labs	10%	drop 2 lowest
Pytania Particip.	2%	up to 1% bonus for correct answers
zyBook readings	3%	(drop 3 lowest-completion sub-sections)
Tests	20%	(10% each test)
Final Exam	25%	(must pass final to pass class)

Note that students were required to pass the final in order to pass the course, so that even if a student's raw score would have given them a D or above, they ended up with an F if they failed the final (this impacted 82 out of the 1,197 students, or about 6.85%). The overall grade distribution across both semesters of the course can be seen in Figure 1, which uses the following letter grade assignment to map raw scores to letter grades.

Grade	Score	Grade	Score	Grade	Score	Grade	Score
A+	98%	B+	88%	C+	78%	D	60%
A	92%	B	82%	C	72%	F	0%
A-	90%	B-	80%	C-	70%		

4. METHODS

All of the students who were flagged with an honor code violation were removed from the data set and not considered when making predictions. We consider their data as inaccurate records of effort, ability, and expected outcomes.

4.1 Feature Engineering

New features were engineered based on the Pytania and zyBook data in an attempt to capture behavioral patterns of students. In particular, we aimed to represent qualities such as participation, consistency, and grit, all of which may be important factors in predicting success. Note that the Pytania data is essentially a measure of attendance and in-class participation.

The Pytania data in its raw form consisted of rows corresponding to a single question answered by a single user with a timestamp and whether or not the question was answered correctly. We needed to extract from this a set of features for

each user which could be fed into our predictive model. The first attempt was to simply add two new features for each student, the number of Pytania questions attempted and the number answered correctly. However we could capture more information from the data by also taking into account the chronology of when students were answering questions. For example, a student who missed the first week but has answered every question afterwards ought to be more highly considered than one who answered everything for the first couple of weeks but has since dropped off. To capture this information, binning was used to create multiple new features dependent on the number of questions answered within a certain time frame. In particular, a feature is added for the number of question answered in each 1 week period through the semester. For example, there is a bin for the most recent week, and another for the week before that, and so on until the start of the semester. In this way we expect the model to weight more recent bins more highly as they are more likely to predict future performance compared to using past performance as a predictor.

We also wanted to have some measure of consistency of student participation. This was measured by taking the standard deviation of the weekly bins. The result was added as a new feature to the model. Students with a high amount of inconsistency would perhaps be expected to perform worse, or at least be more difficult to predict accurately.

The zyBook data was similar in form to the Pytania data, recording individual attempts at a question by a student. Unlike with the Pytania data, students could have multiple attempts at each question. Across all users, the average number of submissions per question is 1.5. The features extracted from this data include total number of attempts (submissions), total number of correct submissions (max one per problem), and average earliness (measured as the difference in time between the problem due date and the first correct submission). The correct submissions feature is divided up by chapters from the textbook, and special 'challenge' problems, which are more difficult and involved, are handled separately from regular 'participation' problems.

4.2 Normalization

Normalization was applied to ensure that no one feature had too dominant of an impact. Each column of features was mapped to the range 0-1 such that the max observed value became a 1 and the min value became a 0. This had a minimal positive effect on results.

4.3 Algorithms

After the features were generated, we ran experiments to determine which algorithm does the best job at predicting student performance. We tested a wide variety of regression algorithms in our experiments including Linear, ElasticNet, Lasso, and Ridge regression models [24, 22, 15, 3]. We also used the support vector regression algorithm (SVR). Finally, we used a number of ensemble algorithms including AdaBoost, Bagging, Gradient Boost and a Random Forest approach [12, 2, 13, 14]. We employed the Python module `sk.learn` for implementations of each algorithm and each algorithm was additionally tuned for optimal parameters using GridSearch (the final parameters can be seen in appendix A).

4.4 Metrics

Two relevant outcomes for prediction were examined: predicting the final RAW score in the course and predicting the score on the final exam (test 3, "T3"). Final RAW score is arguably more important to predict as this is what impacts the student's GPA and is the most commonly used metric for success in a course. However, many of the features being used to predict RAW score are also directly correlated with RAW scores (e.g. projects make up 40% of the final course grade). This makes the prediction less meaningful as the regression weights may come to simply recreate the exact course weighting for assignment. Hence we also predict scores for the final exam, a distinct assessment which is not directly calculated from any of the other features. The final exam is also arguably a better metric for how much of the content a student was able to truly learn.

For regression tasks such as predicting the RAW score we used mean absolute error (MAE), root mean squared error (RMSE), and the coefficient of determination (R2) score as our metrics of evaluation (as defined in figure 2). Note that MAE and RMSE are in the range $[0, \infty)$, and lower scores are better. The RMSE is a commonly used metric for evaluation of regression models, which punishes big errors due to the squared error term. The MAE on the other hand weights all individual differences equally on a linear scale. The result is a more interpretable score, which represents the average error in our predictions. (e.g. a MAE of 10 while predicting final exam scores would mean that the mean prediction was 10 points off from guessing the true value). R2 is in the range $(-\infty, 1]$, where higher scores are better. The R2 score compares the effectiveness of a model to a simple baseline which predicts the mean value for each instance. A score of 0 implies a model which makes no improvement upon this baseline, while a negative score implies a model which is worse than the baseline. A score of 1 perfectly predicts each true value.

5. EXPERIMENTAL RESULTS

Figure 3 shows the resulting MAE, RMSE, and R2 scores for predicting final exam and raw scores using the first 9 weeks of data in a 16 week course. We see that the AdaBoost regressor achieved the lowest MAE score when predicting final exam scores, with an average error of 8.02. The SVR and Bagging regressors also performed well here. For predicting the raw course score, the Ridge regressor performed the best overall, with the top RMSE of 7.03, and an R2 score of 0.84. The Lasso regressor outperformed Ridge slightly in terms of MAE where it achieved a score of 5.42. For the sake of comparison, we also present a 'baseline' score which was calculated simply by predicting the mean value for each student (note that this baseline will have an R2 score of 0 by definition).

Using the optimal algorithms for each category from above, we ran experiments to determine how accurately we can make predictions at various points throughout the semester. In particular, we used the AdaBoost regressor for predicting final exam score and the Ridge regressor for predicting final RAW score. Figure 4 shows the results for this experiment, plotting MAE scores as a function of the number of weeks of data used. As expected, the accuracy of the prediction improves as more data is available for use through the first

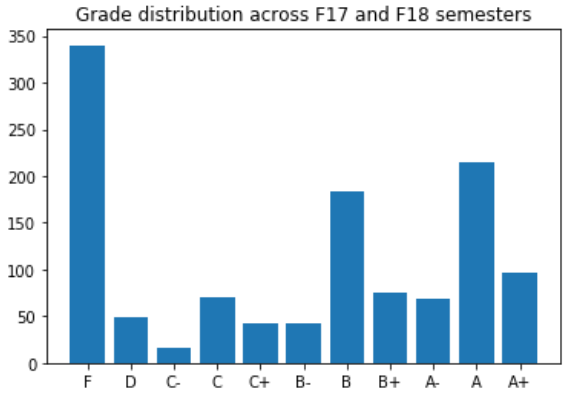

Figure 1: CS112 Grade Distribution

MAE	$\frac{1}{n}\sum \lvert y_{true} - y_{pred} \rvert$
RMSE	$\sqrt{\frac{1}{n}\sum \lvert y_{true} - y_{pred} \rvert^2}$
R^2	$1 - \frac{\sum((y_{true} - y_{pred})^2)}{\sum((y_{true} - y_{true_{mean}})^2)}$

Figure 2: Evaluation Metrics

(where y_{pred} and y_{true} correspond to the predicted value and the actual value respectively of a particular data point, n is the number of samples, and summations go over all n samples)

	Predict Final Exam			Predict RAW		
	MAE	**RMSE**	**R2**	**MAE**	**RMSE**	**R2**
Baseline	15.12	22.48	0.0	13.07	18.25	0.0
LinearRegression	9.98	13.51	0.47	5.72	7.55	0.82
Ridge	8.86	13.11	0.47	5.68	**7.03**	**0.84**
Lasso	9.95	13.11	0.50	**5.42**	7.07	0.84
ElasticNet	9.76	12.96	0.51	5.77	7.37	0.83
SVR	8.31	12.47	0.55	5.57	7.63	0.81
AdaBoost	**8.02**	**10.97**	**0.65**	6.11	9.20	0.73
GradientBoosting	9.32	14.11	0.43	7.97	12.76	0.49
Bagging	8.76	11.76	0.60	5.91	7.96	0.80
RandomForest	9.89	14.46	0.40	7.83	12.12	0.54

Figure 3: Regression Algorithm Comparison

10 weeks of the semester. We also see that predictions of raw score are consistently better than predictions of final exam score. This makes sense as the raw score is in fact calculated directly from some of the features we are using in our predictive model, while the final exam score is not.

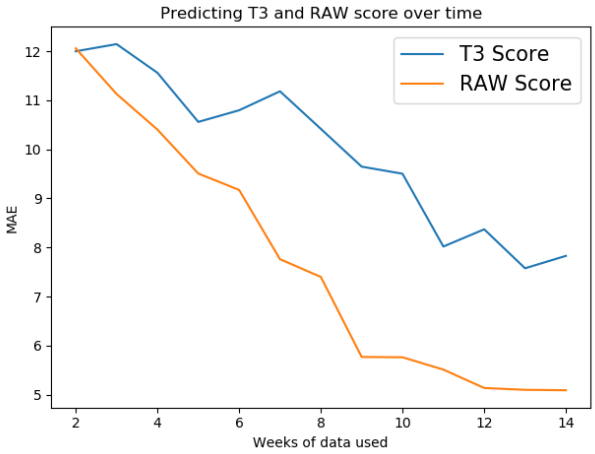

Figure 4: Predicting T3 score over time

Figures 6 and 7 show the values of the coefficient weights for each feature from the optimal final exam and raw score models run using 8 weeks of data (see figure 5 for coefficient definitions). For final exam prediction, it is not surprising to see that performance on the first exam, T1, is the best predictor. Next, other substantial graded assignments such as projects 2 and 3 and lab 5 also have fairly high weights. The Pytania features overall have small weights, suggesting that participation in the polling system is not strongly correlated with success. It may be the case that students did not take Pytania participation very seriously as it only accounted for 2% of their semester grade. However, we do observe that pwa:0 and pwa:1, which correspond to questions answered in the most recent weekly periods, are more significant than the earlier weeks, meaning that recent activity is a better indication of a student doing well than activity early on in the course. For the zyBook features we see that each chapter has a small positive weight. The strongest positive zyBook weights are chal_acp (the number of challenge problems successfully completed) and zy_early (the average earliness of submissions). Interestingly, zy_attempt (corresponding to the total number of attempted submissions) and extra_sub

Feature Name	Meaning
L1E	Lab 1 Exercise
L4T	Lab 4 Test
L6Q	Lab 6 Quiz
P1	Project 1
T1	Test 1
chal_acp	Challenge zyBook problems successfully submitted
extra_sub	Resubmissions to a previously correctly submitted problem
pytania_std	Standard deviation across Pytania bins
pwa:1	Pytania participation last week
pwa:2	Pytania participation two weeks ago
zychp1	zyBook correct submissions from chapter 1
zy_attempt	Total count of zyBook submissions
zy_early	Average earliness of zyBook submissions

Figure 5: Regression Coefficients

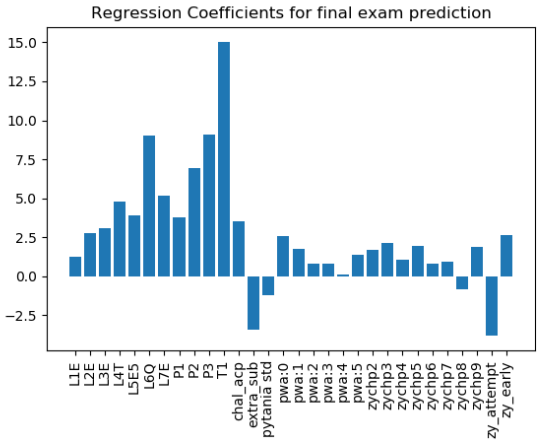

Figure 6: Regression coefficients T3

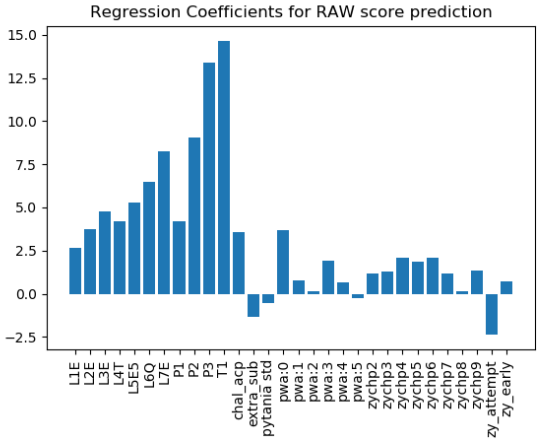

Figure 7: Regression coefficients Raw

(corresponding to the total number of additional submissions to a problem which had already been completed) both have negative weightings. The first could be explained by the fact that students who struggle and require multiple submissions to a problem are likely weaker students. The second is more difficult to account for, as one might expect subsequent attempts at a problem for which the student already has credit to be a sign of determination to learn the material.

6. DISCUSSION

In this paper we formulated an in-class predictive model for student performance in a CS1 course taught at George Mason University. Our model used a feature set solely derived from in-class activities and assignments rather than relying on past information or demographics. We employed a variety of machine learning algorithms and tested their accuracy as a function of the number of weeks of data used from the semester. Our results show that both final raw scores and final exam scores can be predicted with a high level of accuracy as early as 6 or 7 weeks into the course. Thus it could be effectively employed as an early warning system, such that students could have a good sense of what grade they will end up with if they remain on their current trajectory.

We note that often in cases where multiple features had equivalent direct grade contributions (due to course weighting) there was a discrepancy in the predictive power amongst these features. For example, we found that recent participation was a more relevant predictor of ultimate success than past participation, even though it is equivalent in terms of direct impact on a student's grade. Similarly, we found that engagement with certain textbook chapters, and success on certain labs and quizzes are more indicative of success than others. Thus there could be a situation in which two students have the same class grade, and yet one is flagged as a struggling student while the other is not. Herein lies the value of using such a model; it can discover nuances and patterns in student performance which an instructor (particularly in an introductory course with many students) would otherwise be unable to detect.

6.1 Future Work

In this work we considered only in-class information. It would be interesting to see how much improvement could be made to the model by also considering past information such as prior course scores and demographics. It would also be worth trying to make predictions of other CS courses, or non-CS courses, and comparing the accuracy of predictions as well as which features stand out as strong predictors.

We propose that our predictive model could serve as an early warning system, triggering intervention. However, we do not suggest exactly how or when this intervention should take place. Studies would have to be performed on different forms of intervention to determine which methods work best.

7. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation REU grant for Education Data Mining at George Mason University, award 1757064. We would also like to thank Dr. Rangwala (PI of the REU site) for his contributions to our work and his invaluable guidance throughout the research process, as well our fellow REU student participants for their continual support and encouragement.

8. REFERENCES

[1] Ryan Shaun Baker and Paul Salvador Inventado. Educational data mining and learning analytics. In *Learning analytics*, pages 61–75. Springer, 2014.

[2] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

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

[4] Evandro B Costa, Baldoino Fonseca, Marcelo Almeida Santana, Fabrísia Ferreira de Araújo, and Joilson Rego. Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses. *Computers in Human Behavior*, 73:247–256, 2017.

[5] Scott Crossley, Shamya Karumbaiah, Jaclyn Ocumpaugh, Matthew J Labrum, and Ryan S Baker. Predicting math success in an online tutoring system using language data and click-stream variables: A longitudinal analysis. In *2nd Conference on Language,*

Data and Knowledge (LDK 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[6] M. Meehan E. Howard and A. Parnell. Contrasting prediction methods for early warning systems at undergraduate level. *Internet Higher Educ.*, pages 66–75, 2018.

[7] Alex Edgcomb, Daniel de Haas, Roman Lysecky, and Frank Vahid. Student usage and behavioral patterns with online interactive textbook materials. In *International Conference of Education, Research and Innovation (ICERI), Spain*, 2015.

[8] Alex Edgcomb, Frank Vahid, Roman Lysecky, and Susan Lysecky. Getting students to earnestly do reading, studying, and homework in an introductory programming class. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 171–176. ACM, 2017.

[9] Alex Daniel Edgcomb, Frank Vahid, Roman Lysecky, and Susan Lysecky. An analysis of incorporating small coding exercises as homework in introductory programming courses. In *ASEE Annual Conference and Exposition, Conference Proceedings*, volume 2017, 2017.

[10] Asmaa Elbadrawy, Agoritsa Polyzou, Zhiyun Ren, Mackenzie Sweeney, George Karypis, and Huzefa Rangwala. Predicting student performance using personalized analytics. *Computer*, 49(4):61–69, 2016.

[11] Asmaa Elbadrawy, Scott Studham, and George Karypis. Personalized multi-regression models for predicting students performance in course activities. *UMN CS*, pages 14–011, 2014.

[12] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[13] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[14] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[15] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[16] Hengxuan Li, Collin F Lynch, and Tiffany Barnes. Early prediction of course grades: Models and feature selection. *arXiv preprint arXiv:1812.00843*, 2018.

[17] SungJin Nam and Perry Samson. Integrating students' behavioral signals and academic profiles in early warning system. In *International Conference on Artificial Intelligence in Education*, pages 345–357. Springer, 2019.

[18] Zhiyun Ren, Xia Ning, and Huzefa Rangwala. Ale: Additive latent effect models for grade prediction. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 477–485. SIAM, 2018.

[19] Zhiyun Ren, Huzefa Rangwala, and Aditya Johri. Predicting performance on mooc assessments using multi-regression models. *arXiv preprint arXiv:1605.02269*, 2016.

[20] Cristobal Romero and Sebastian Ventura. Educational data mining and learning analytics: An updated survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, page e1355, 2011.

[21] M Snyder. Large-lecture participation via (free) pytania student response system. In *Poster presented at Innovations in Teaching and Learning 2018, Fairfax, VA.*, 2018.

[22] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[23] Christopher Watson and Frederick W.B. Li. Failure rates in introductory programming revisited. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, ITiCSE '14, pages 39–44, New York, NY, USA, 2014. ACM.

[24] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.

APPENDIX

Appendices

A. GRIDSEARCH PARAMETERS

Grid search found the optimal combination of parameters for each machine learning algorithm. The following parameters were used (any unlisted used the default value from sklearn):

Ridge: alpha=1.0, solver='lsqr'
Lasso: alpha=0.1
ElasticNet: alpha=0.1, l1_ratio-0.9
SVR: C=1, gamma=0.001, kernel='linear'
AdaBoost: learning_rate=0.2, loss='exponential', n_estimators=80
GradientBoost: criterion='mae', learning_rate=0.01, loss='huber', max_depth=None, min_samples_leaf=1, min_samples_split=10, n_estimators=80, subsample=0.8
Bagging: bootstrap_features=True, max_features=20, n_estimators=80
RandomForest: criterion='mae', min_samples_leaf=0.1, min_samples_split=0.1, n_estimators=40