Predicting Student Success in Cybersecurity Exercises With a Support Vector Classifier*

Quinn Vinlove[1], Jens Mache[1], Richard Weiss[2]
[1]Lewis & Clark College
{quinnvinlove, jmache}@lclark.edu
[2]The Evergreen State College
weissr@evergreen.edu

Abstract

In this paper we explore if we can detect whether students are struggling to complete simple hands-on cybersecurity exercises based on their command line history. These exercises are becoming more popular, especially with the increase in remote instruction. However, students may struggle for many reasons, including lack of some skills, confusion by what is being asked, or confusion about how the testbed works.

Using a small collection of annotated log files from a sample exercise on DeterLab, we were able to generate three features and construct a support vector classifier to predict with 80% accuracy if students would complete the remaining parts of the exercise. Our work could be applied to early detection of students who likely will have difficulty completing the exercise, and offer them hints to boost engagement and learning.

1 Introduction

Capture the flag (CTF) exercises have been used in cybersecurity for fun and for training and education for many years. In our own experience, students often become frustrated when they do not make progress. This has also been reported by others [2]. Even with instructor office hours and TA support,

many students have trouble and struggle with completing hands-on exercises. It can be especially problematic with cybersecurity because it draws on many aspects of computer science, and some of the concepts are difficult. Hands-on exercises can be both more engaging than written homework and can also be more frustrating. Yet, hands-on exercises are very useful for teaching critical thinking and how to apply theory to practice. Our goal is to provide tools to help students be more engaged by automatically detecting when they are struggling, and to potentially offer a hint when that happens.

We used the 'Introduction to DETER and Unix', or 'Intro' exercise [6] on DeterLab [4], completed by 25 students in a small cybersecurity class, and developed features based on the resulting annotated bash history files produced by ACSLE [3], an automated reporting and log file collection program for testbeds. While the first class we tested our methods on was small, the data collection and annotation system would scale to larger classes and potentially give even more precise results with more data.

One of the important features of the ACSLE framework is the detection of milestones. For each of the exercises we tested, we relied on a well-defined set of milestones that measure student completion of different tasks in the exercise. They are defined in terms of the student's input, the command output, and the context in which it was used, e.g. the host or directory. The state of the student's accomplishments at any given time is measured as the set of milestones that have been completed.

We evaluated the classifier using primarily the number of milestones completed, but also the log files themselves with both student input and command line output. We did not have ground truth information on each student as to whether they struggled or not. By constructing three features from this data: the number of related commands between each new milestone achieved, number of repeated commands, and distance from a few 'known good' commands, and using these features as inputs into a multi-dimensional support vector classifier, we were able to predict exercise completion with 80% accuracy. Our work will provide a good basis for a new model to suggest hints automatically.

2 Related Work

Our work provides a useful addition to the field of cybersecurity education research. Švábenskỳ et al. [9] argued in their meta-analysis of all security education papers that while there are many papers on education methods, there are few new methods beyond surveying students directly. We hope that this paper motivates the use of machine learning models to make evaluation succinct and useful for both students and instructors.

For teaching introductory programming, Piech et al. [7] described a learner's

path through an exercise as a Markov chain, and also sought to build effective generalizations of the types of ways that new programmers completed a simple assignment in Java. Their work featured more data, more robust statistical analysis, and validation from midterm and final grades, which was difficult for us because our research was mainly conducted after the school year. Rafferty et al. [8] used a more complex hidden Markov model (HMM) for their cognitive science paper, which has a broad focus on how to apply HMMs to model student learning in a variety of contexts. In both cases, a HMM may prove to be a compelling model, and while we looked into it, we were unable to generalize our exercises as a model with a finite number of hidden states.

3 Methodology

To construct inputs for our classifier, we first created three features, computed them for every student, and checked our work by evaluating each log file by hand. These features are described below.

The first metric was the average number of commands between new milestones reached. This could possibly describe how much effort was spent for each learning objective met. Initial analysis showed that students who completed more milestones either entered relatively few commands for each milestone that they reached, or took their time and entered more, possibly indicating that they didn't know the answer at first, but worked hard to complete the exercise.

The second metric was the longest string of repeats of any single command. To count commands that were close, but not identical, we would compute the edit distance (or Levenshtein distance) from each new command to the last. With long periods of time where the edit distance was low (≤ 3 characters), we could determine that a similar command with few variations was tried a lot, with little variation. We anticipated that users who achieved more milestones, and subsequently understood the exercise better, would have repeated commands less, but the opposite was true: users who did better generally had a wider variation in the length of their longest repeat sequence than users who didn't do as well, showing that frustration may not entirely be indicated by students trying the same thing over and over again.

The last metric we used was the smallest edit distance between a list of 'known good' commands and bash history. This operated on the initial assumption that while the REGEX search method of seeing if a command matches a milestone may be good, it is binary, and sometimes a continuous metric can produce better results. In some cases, like $find$, a command can be missing an option and not work, but still be close enough to indicate that a student knows what they're doing. We computed this minimum edit distance for each com-

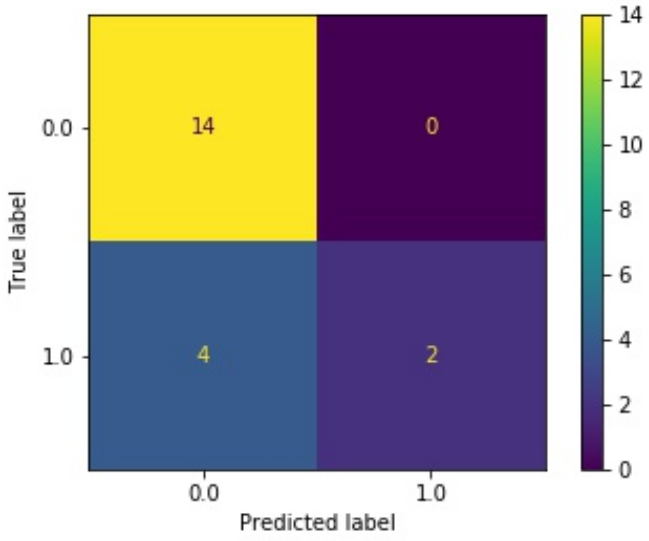

Figure 1: Confusion matrix for the evaluation dataset after training our SVC. Note here that 0.0 represents a 'will not complete' prediction and 1.0 represents a 'will complete' prediction. Each point in the matrix represents a measured state and its outcome.

mand, and summed them. We found that most high-achieving students had a sum that was small, indicating that what they did was pretty close to a known good solution, while students who got few milestones had bash commands with a large edit distance between these predefined commands.

The exercise log data from two classes at USC and one at LC were loaded into a Jupyter notebook with Pandas, processed with the help of numpy, then normalized with a min-max scalar, shuffled, and split 80-20 between the testing set and validation set. For desired tags, we assigned a true to each student if they got 6 or 7 milestones (all of them) and a false if they didn't. Then, we placed this data into a scikit-learn support vector classifier [5].

4 Results

After training a support vector classifier with the training set, we evaluated our work with the validation set and found that the classifier was able to accurately predict true or false values for 80% of the data. The confusion

matrix is shown in Figure 1. Note that the classifier is pessimistic, which is good, since we want to minimize false negatives. There were no false negatives with the sample data. There were some false positives, i.e. it predicted that 4 out of 20 students would not finish but they did.

5 Discussion

Drawing conclusions based on this data set was limited by not having a direct measure of which students actually struggled; nevertheless, we were able to infer this in many cases by reading the bash history logs and counting the milestones achieved. For a machine learning project, we had relatively little data: the 'biggest' exercise was the 'intro' exercise, with only around 75 samples between all schools involved in the study. Inferring any pattern from this data wasn't exactly straightforward.

One of the simplest trends we observed was that more persistent students would score higher on the exercises. Encouraging persistence may increase exercise completion rate. In the future, we plan to interview students afterwards to see what they struggled with and correlate that with their command line history. It's important to note that because our model does not account for overall clock time to completion, persistence simply means trying many new things repeatedly. If a student started the exercise and walked away from the keyboard, our model wouldn't consider that to be persistent.

Some preliminary work explored how well this model applies when students are just beginning the exercise, not just at the end of it. A variation of this model trained only on the snippets of bash history between new milestones achieved still maintains 80% accuracy (Fig. 3), and points toward increasing completion rates as students persist longer (Fig. 2). To construct this dataset and generate more data, we sliced the data for each student several times progressively, so that a single student who, for example, achieved three new milestones, had three data points, each representing what their bash history data would look like if they stopped working at that point.

ACSLE can be used with EDURange [1, 10, 11], and we plan to integrate an improved version of ASCLE that uses string-edit distance, so that instructors can receive feedback about students in real time. We hypothesize that the string-edit distance can distinguish trial and error guessing from knowledge-based exploration.

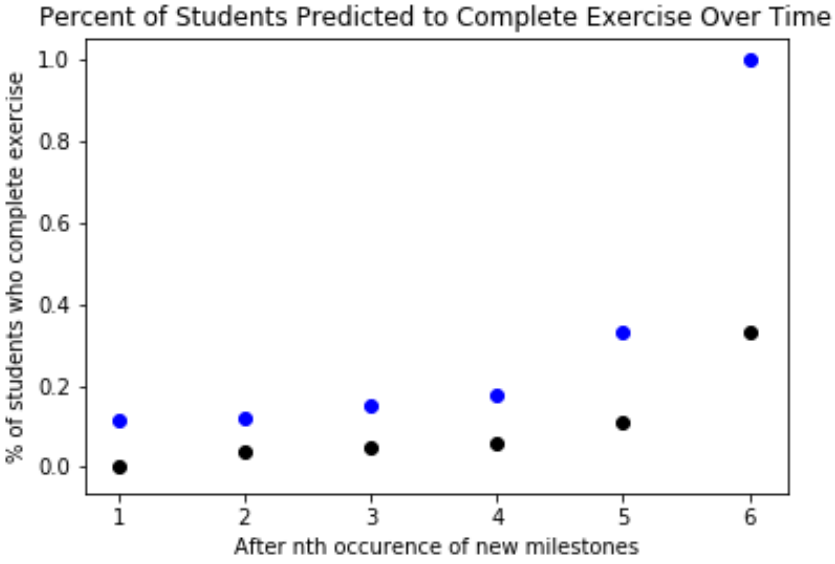

Figure 2: This figure shows the percentage of students who achieved one or more new milestones, then went on to complete the exercise. As students progress and continue to reach new milestones, more stop working. As fewer continue, the ones who do end up with a higher likelihood of completing. The blue dots represent the actual value, and the black dots represent the predicted value.

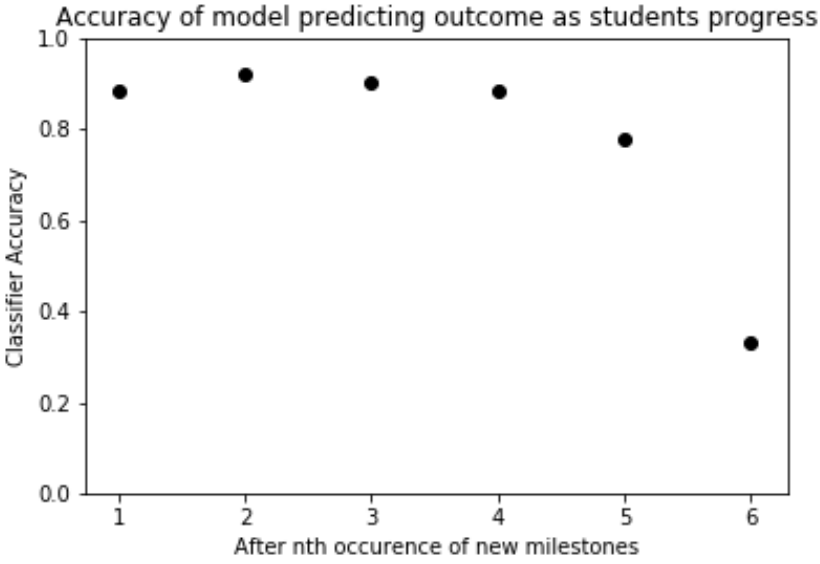

Figure 3: Classifier accuracy for the scale on the x axis described by Fig. 2. Our classifier is able to accurately predict exercise completion with 80 percent accuracy or better for the first five occurrences of new milestones. Accuracy goes down near the end because we had few students who made it that far, but the model is not needed at this point because students who did ended up completing the exercise anyway.

6 Conclusion and Future Work

Given a small set of log files with limited information, the system was able to construct a model that estimates whether a student will complete the exercise. This was run using information taken at different times during the exercise, and could potentially indicate when a student was struggling and should be offered a hint. Our work will be useful in extending EDURange, and might be useful in application to other areas where generalized models of student success could be built with very little data.

Since the ACSLE system shows the failed attempts at a milestone, it makes it easier for exercise authors to identify common misconceptions and then produce hints for each of them. One way to use this would be to alert the instructor when a student is struggling, and the instructor could choose one of the pre-recorded hints based on a digest of the student's failed attempts.

In the future, instructors will be able to write a single file that describes commands that students could use in a solution. The string-edit distance between what they typed and those commands would be used to assess progress. Potentially, 'snippets' of them could be used as a hint to assist stuck students after our system detects that they may not finish. After implementing this system, we also plan on evaluating its effect on exercise scores.

Acknowledgments

We would like to thank Jelena Mirkovic from USC. This work was partially supported by National Science Foundation grants 1723714 and 1723705.

Research Artifacts

Datasets, along with the accompanying Jupyter Notebook, can be found on GitHub at https://github.com/edurange/predicting-student-success.

References

[1] Stefan Boesen, Richard Weiss, James Sullivan, Michael E Locasto, Jens Mache, and Erik Nilsen. EDURange: meeting the pedagogical challenges of student participation in cybertraining environments. In *7th Workshop on Cyber Security Experimentation and Test (CSET)*, 2014.

[2] Kevin Chung and Julian Cohen. Learning obstacles in the capture the flag model. In *2014 {USENIX} Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, 2014.

[3] Jelena Mirkovic, Aashray Aggarwal, David Weinman, Paul Lepe, Jens Mache, and Richard Weiss. Using terminal histories to monitor student progress on hands-on exercises. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 866–872, 2020.

[4] Jelena Mirkovic and Terry Benzel. Teaching cybersecurity with DeterLab. *IEEE Security & Privacy*, 10(1):73–76, 2012.

[5] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[6] Peter A. H. Peterson and Peter Reiher. *Introduction to DETER and Unix*, (accessed August 27, 2020). `https://www.isi.deterlab.net/file.php?file=/share/shared/LinuxandDeterLabintro`.

[7] Chris Piech, Mehran Sahami, Daphne Koller, Steve Cooper, and Paulo Blikstein. Modeling how students learn to program. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 153–160, 2012.

[8] Anna N Rafferty, Michelle M LaMar, and Thomas L Griffiths. Inferring learners' knowledge from their actions. *Cognitive Science*, 39(3):584–618, 2015.

[9] Valdemar Švábenskỳ, Jan Vykopal, and Pavel Čeleda. What are cybersecurity education papers about? a systematic literature review of sigcse and iticse conferences. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 2–8, 2020.

[10] Richard Weiss, Michael E. Locasto, and Jens Mache. A reflective approach to assessing student performance in cybersecurity exercises. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, page 597–602, New York, NY, USA, 2016. Association for Computing Machinery.

[11] Richard Weiss, Franklyn Turbak, Jens Mache, and Michael E. Locasto. Cybersecurity education and assessment in EDURange. *IEEE Security & Privacy*, 15(03):90–95, May 2017.