The Future of Engineering Education
2024 Annual Conference & Exposition
Oregon Convention Center
Portland, OR . June 23 - 26, 2024
ASEE
Paper ID #41475

Board 289: From Logs to Learning: Applying Machine Learning to Instructor Intervention in Cybersecurity Exercises

Aubrey Nicholas Birdwell, Georgia Institute of Technology

Aubrey Birdwell is currently pursuing a Master's degree in Artificial Intelligence and Machine Learning at the Georgia Institute of Technology, and previously completed a Bachelor of Science degree in Computer Science at The Evergreen State College. His research is focused on computing education, particularly in the domain of cybersecurity. Aubrey has worked extensively on developing an application aimed at teaching cybersecurity content. This project explores data processing and the application of machine learning techniques to provide dynamic hints, evaluate user progress, and enhance learning through visualization.

Jack Cook, The Evergreen State College

Lead Developer for the EDURange cybersecurity training platform.

Dr. Richard S Weiss, The Evergreen State College

Richard Weiss is currently a Member of the Faculty at The Evergreen State College and has been teaching security and information assurance since 2003. He received an A.B. in mathematics from Brandeis University and a Ph.D. in mathematics from Harvard University.

Dr. Jens Mache, Lewis & Clark College

Jens Mache is an educator and researcher at Lewis & Clark College in Portland, Oregon.

From Logs to Learning: Applying Machine Learning to Instructor Intervention in Cybersecurity Exercises

Aubrey Birdwell
Georgia Institute of Technology
aubrey.birdwell@gmail.com

Jack Cook
The Evergreen State College
cookjackc@gmail.com

Richard Weiss
The Evergreen State College
weissr@evergreen.edu

Jens Mache
Lewis & Clark College
jmache@lclark.edu

Abstract

Hands-on exercises provide students with practical skills and abilities. However, for exercises to be effective, students may need timely feedback while they are engaged to prevent them from getting stuck or frustrated. The goal of this project is to use machine learning to help identify such students such that timely and contextually appropriate hints can be given.

We are building a system that identifies students who are potentially in the most need of help, and suggests hints that the instructor could provide. The instructor can reject hints that they do not find appropriate. The hint system will be integrated into the EDURange cybersecurity education platform and will also be compatible with other platforms.

We are collecting data that will be analyzed to determine the efficacy of the tool, and to develop new hints and strategies for helping students. This project plans to use our machine learning system to create, test, and deploy semi-automated hints in a timely manner.

Keywords: Computer Science Education, Cybersecurity, Active Learning, Laboratory Experience, Learning Environment, Experience Report, Artificial Intelligence, Machine Learning

Introduction

EDURange is an open source computer science education platform. It is a joint effort by students and faculty at multiple universities. Our primary goal is to develop educational tools for computer science and cybersecurity education. While the platform itself has been in development for over a decade, in 2019 we began developing a machine learning system to facilitate instructor-student interaction by attempting to detect struggling participants. The current version includes an instructor dashboard and chat that connects students with instructors during the exercise, which is when students need help the most. Our goal is to develop an early warning system that detects students who are not making progress and connect them with immediate feedback.

Our research groups have also been actively engaged in research in educational data visualization, and machine learning, and user interface design.

Related Work

A systematic survey of 357 articles related to predicting performance in computer science course work was published by Hellas et al in 2018 [1]. Most of the research surveyed concerned predicting grades for programming courses. This paper is different. It concerns giving rapid feedback to students who are engaged in cybersecurity exercises. Beyond the work mentioned in that article, there have been a few attempts to predict student outcomes specifically for cybersecurity training and exercises [2, 3, 4]. One of the first examples used the support vector machine to predict whether a student would complete an exercise based on their success in the beginning of the exercise [5]. Our current work explores a number of machine learning algorithms and compares their accuracy for determining levels of exercise completion with the hope of identifying participants who are having difficulties, as well as helping to locate where in the scenario they are struggling.

Research Methods

While developing our machine learning system we worked closely with colleagues from other institutions who are simultaneously developing their own platform. Building from and extending some of their practices, we adopted a stratified k-fold nested cross-validation loop to split our data. We then train eight classifier models using a Logistic Regression classifier for estimation and feature selection. The classifiers we use are XGBOOST, Random Forest, Decision Tree, K-Nearest Neighbor, Support Vector Machine (RBF), Support Vector Machine (Linear), Naive Bayes, and Logistic Regression. The 67 data points we used for this comparison were all from the same command-line exercise which involved the students completing 13 tasks using bash commands. Our study involves the early detection of struggling students so that we can identify students who need intervention or hints. Ideally, the machine learning system will help to determine what a student is working on and whether they are making progress.

EDURange uses what we call ”milestones” to tag command-line bash history activity where students achieve a sub-goal of the scenario. Milestones are compound regular expressions that are written to match particular objectives that we wish to detect students achieving. A scenario is a collection of milestones or tasks and can be thought of as a complete exercise. Our objective is to identify students that complete at least 50% of a scenario and it uses features based on command line data only. While our current study evaluates the entire dataset, we would like to implement a system that can identify that a student will reach that 50% threshold given a partial data set. We would also like to create a multiclass classification system that can detect what task in the scenario a student is working on either by using discretized percentage thresholds directly or by running them as individual classifiers in parallel. This is meant to work in concert with our hint delivery system. We are continuing to explore additional methods to enhance our machine learning such as reinforcement learning and how we could incorporate LLMs eventually.

Data Collection

Our data is collected during scenario deployment when students are participating in the exercises. After receiving IRB approval, the students are informed of the data collection and the exercises are offered both for credit in university level course work, as well as purely outside of the academic world in workshops. The system for data collection occurs on multiple levels. Most of our exercises take place on the command-line, and we were able to build on top of existing research that explored monitoring students command-line bash history [6]. In detail, the pure TTY text streams are captured and formatted into CSV files using automated scripts that clean up the text streams and use parsing to determine the nature of the log events, marking them for additional processing and use with machine learning. The answers to questions in the student interface are captured and stored in a database, as well as interactions between students and instructors. Unique identifiers combined with system timestamps allow us to integrate the logs from multiple sources.

System Architecture

Our training platform is composed of four major systems: a web application, a control framework for provisioning virtual environments and collecting data during exercises, a database for storing and organizing this data, and a post-processing tool chain to analyze this data further. The web application is a basic portal which allows instructors to create and organize student groups and exercises. Additionally, students interact with the web application to access tutorials for the exercise they're playing, answer assessment questions, and interact with the instructor via a real time chat. We also provide connection information for the virtual environments and an interactive in-browser terminal. The control framework for exercise deployment is managed via Terraform to allow instructors to deploy pre-designed Docker containers on demand, depending on which exercise they are running. This also allows designing new exercises to be a very flexible and streamlined process, since any Docker image can easily be turned into a new exercise [7]. As a result of using Docker as our only infrastructure tool, the platform is cloud-agnostic, highly portable, lightweight, and could even be installed on an instructor's personal computer.

Our data collection tool chain consists of command-line logging, app interface data, and chat data collected from student-instructor interactions. Command-line logs are initially collected using a TTY logger that attaches to the students TTY session and pipes the raw text into a file. Additional processing is performed using Python scripts that run periodically to filter out control character sequences, parse, and organize the input and output streams into several columns of comma separated data. The polished data is then parsed and relevant milestones or identifiable goals are labelled. Once the data is labelled, several additional scripts extract features and create a dataframe so the data can be processed using machine learning. In addition to the command-line logs, we are storing the chat sessions generated by student-instructor interactions, and the web interface data generated by students submitting solutions to the app. The latter two data sets have not yet been used for machine learning but we are working to integrate them.

Results

We chose a basic 50% threshold of completion as a target label because our goal is to demonstrate that command-line logs are feasible for use in our machine learning auto detection system. In practice, a parallel system of classifiers, each detecting a different threshold or a multi-class system would be used. Since we only have collected 67 data points at this time, our system needed to be stripped down a little while we focused on feature engineering and creating a broad table of results for comparison. While for this study we employ 8 different classifiers, only one will ultimately be chosen and we are in the process of developing a reinforcement learning system that may end up performing better than traditional classifiers. The table below represents the average performance of 8 classifiers that were trained using a stratified kfold cross validation loop. The features for each of the underlying models were chosen automatically using a Logistic Regression estimator and grid search. While we developed our own custom feature and log extraction modules, the code for setting up the machine learning was adapted from previous work, and is available as open source for anyone to use [8]. Below is the table representing the average performance of several kinds of classifiers we have been testing.

Classifier	Sensitivity	Specificity	Balanced Accuracy	AUC
XGBoost	0.466	**0.925**	0.696	0.832
Random Forest	0.500	0.877	0.693	0.884
Decision Tree	0.399	0.794	0.597	0.648
K Nearest Neighbors	0.666	0.850	0.758	0.860
SVM with RBF	**0.933**	0.869	0.901	0.955
SVM with linear	0.766	0.850	0.808	0.904
Naive Bayes	0.533	0.780	0.656	0.702
Logistic Regression	1.0	0.850	**0.925**	**0.973**

Table 1: Comparison of performance of 8 classifiers on our 67 data points.

It is presented both as a work in progress and as a proof of concept. For the classification, we separate the students who complete 50% of the scenario as a minimum threshold for completion. Our aim is merely to demonstrate that pure command line logs, and features derived from them, can be used for our machine learning system.

Discussion

Our initial work with machine learning has been successful. It is important to recognize that the success we have had is not a complete system and that our choices have influenced what classifiers appear to be performant. For instance, perhaps a different estimator function would emphasize other classifiers. We have a proof of concept that can facilitate advancing our project but it needs development. We are able to clearly demonstrate that command-line data can be used for machine learning and other research teams have used both command-line and web application data in successful machine learning systems. The system thus far is able to identify how complete a student's progress is just based on command line features such as number of commands per minute, time gaps in commands being issued, and other similarly abstract features. Our system needs to be able to help us detect exactly where in the scenario a student is and whether they are making progress on their current task.

Our next steps are to develop a more robust machine learning system incorporating our web app data and to train models on more complex labels. This will take the form of a multi-class learning system that will specify what task students are working on and whether they are making progress or not with that task. This is important since we hope to use classification to inform and trigger our hint system. The hints will then be supplied to students who are in the process of completing the exercises. While currently we are exploring a broad spectrum of machine learning algorithms, it is our hope to narrow this to the best performing features and machine learning techniques.

A second aspect of our research is collecting student-instructor interactions through a custom chat interface built into our platform. These chats will eventually serve as training data for hint generation. Currently we have successfully deployed the chat system in the classroom but we have not collected enough data to effectively use it for training a complete hint system. This will take more time. The interactions take the form of labelled text interactions which can be matched up with our command line logs so that the issues in the command line logs may be associated with the instructor interventions. Our plan is to extract useful tips and hints from the intervention chat data and automatically apply them via a nudge from our app interface when struggling students are detected.

Initially, when we conceived of the system, large language model technology was not widespread. We had thought we would use the text from the chats to generate hints using more traditional natural language processing techniques to extract relevant text snippets. Given the rapid success of LLMs, we have begun to consider using open source LLMs in the system.

It may be possible to use our chat logs to fine tune an LLM to generate custom hints based on the models provided by student-instructor interactions, and potentially, to analyze some of the failed command line behaviors we are able to label using machine learning. With or without the use of LLMs, combining our early detection system using machine learning with our chat text data, we hope to develop a complete system to detect participants having difficulties and offer them custom suggestions.

While the previous two use cases for LLMs might consume a lot of overhead, there is a more practical manner that LLMs could be used off-line during the scenario design process. Having scenario designers enumerate all of the potential solutions to command-line exercises can be a bottleneck for new scenario design. Using LLMs to enumerate potential solutions that can be vetted by experts and encoded in our json format for the parser would speed up scenario deployment.

As a research group we do typically focus on teaching cybersecurity, but it is important to acknowledge that we feel this system could be used in other domains of computing education and that we are interested in extending the platform to cover more material.

Conclusion

We have successfully operated our chat in the classroom and students seem to like it. In addition to the machine learning system, we are continually improving and experimenting with different presentations of the student interface and chat to improve accessibility and student engagement. We have demonstrated success using machine learning to classify student progress through a scenario. These are first steps but also proof of concept that our constituent systems are working. We will continue to develop and integrate the student interface for chat and receiving hints, and our machine learning system. Another important goal is improving our ability to accurately localize students within a scenario, as that is central to our ability to then provide relevant hints.

References

[1] A. Hellas, P. Ihantola, A. Petersen, V. V. Ajanovski, M. Gutica, T. Hynninen, A. Knutas, J. Leinonen, C. Messom, and S. N. Liao, "Predicting academic performance: A systematic literature review," in *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE 2018 Companion. New York, NY, USA: Association for Computing Machinery, 2018, p. 175–199. [Online]. Available: https://doi.org/10.1145/3293881.3295783

[2] V. Švábenský, R. Weiss, J. Cook, J. Vykopal, P. Čeleda, J. Mache, R. Chudovský, and A. Chattopadhyay, "Evaluating two approaches to assessing student progress in cybersecurity exercises," in *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education (SIGCSE)*, 2022.

[3] Y. Deng, D. Lu, C.-J. Chung, D. Huang, and Z. Zeng, "Personalized learning in a virtual hands-on lab platform for computer science education," in *2018 IEEE Frontiers in Education Conference (FIE)*. New York, NY, USA: IEEE, 10 2018, pp. 1–8.

[4] A. R. Silva, J. T. McClain, B. R. Anderson, K. S. Nauer, R. Abbott, and J. C. Forsythe, "Factors impacting performance in competitive cyber exercises," Sandia National Lab, Tech. Rep., 2014. [Online]. Available: https://www.osti.gov/servlets/purl/1315132

[5] Q. Vinlove, J. Mache, and R. Weiss, "Predicting student success in cybersecurity exercises with a support vector classifier," *J. Comput. Sci. Coll.*, vol. 36, no. 1, p. 26–34, 10 2020. [Online]. Available: https://dl.acm.org/doi/10.5555/3447051.3447055

[6] J. Mirkovic, A. Aggarwal, D. Weinman, P. Lepe, J. Mache, and R. Weiss, "Using terminal histories to monitor student progress on hands-on exercises," in *Proceedings of Special Interest Group on Computer Science Education Symposium (SIGCSE)*, 2020.

[7] J. Cook, R. Weiss, J. Mache, C. G. Morán, and J. Wang, “An authoring process to construct docker containers to help instructors develop cybersecurity exercises,” *Journal of Computing Sciences in Colleges*, vol. 37, no. 10, pp. 37–47, 2022.

[8] K. Tkáčik, “Predicting student success in cybersecurity training,” https://is.muni.cz/th/dkm2u/, 2023, online.