Towards Predicting Success in MOOCs: Programming Assignments

Kshitij Sharma1, Łukasz Kidzinski1, Patrick Jermann2, Pierre Dillenbourg1

1. Computer Human Interaction in Learning and Instruction

2. Center for Digital Education

École Polytechnique Fédérale de Lausanne

Email: <firstname>.<lastname>@epfl.ch

Abstract

Students of programming languages in massive on-line open courses (MOOCs) solve programming assignments in order to internalize the concepts. Programming assignments also constitute the assessment procedure for such courses. Depending on their motivation and learning styles, students pursue different strategies. We identify which approach to attempt these assignments results in better performance. We predict students’ success from their online behaviour; and identify different paths students chose in order to complete the MOOC. We also discuss how students resign from the course after having difficulties with assignments. Moreover, we also predict, when would a student give-up (or succeed) submitting solutions to a given assignment.

Keywords

MOOCs, Programming assignments, Predicting success, Student categorization.

# 1 Introduction

According to the current trends in MOOCs, assessments play a key role in defining success. In the case of programming courses, assessments become even more important, as programming assignments are the prime ways to assess students’ understanding about the content. In a preliminary analysis for 120,000 programming assessments, we observed that the main disengaging factor for the active students (who submit at least one assignment, Sharma et. al., 2015a) is failure in these programming assignments. Among all the active students, 27% fail the course, among which 74% left the course after failing in their last assignment submissions. This fact attracts a special concern for analysing the submission behaviour of the students.

In this contribution, we predict students’ success (final score for each assignment) in programming assignments. Our goal is to predict the success of the MOOC students at an early stage of the course. This might enable the MOOC teachers to support the students who have (or are predicted to have) problems in getting the correct submission for a programming assignment.

We also present the prediction results for every other submission attempt. We predict for each submission two quantities: 1) whether there will be another submission; and 2) what will be the next attempt score? Based on these predictions the MOOC instructors can have multiple support design guidelines, that reach beyond the simple error messages from the “online judge”.

The rest of the paper is organised as follows. The second section presents the related work. The third section explains the problem statement and the research questions. The fourth section introduces methods and different variables used in this contribution. The fifth section presents the results. Finally, the sixth section discusses the results and implications, and concludes the paper.

# 2 Related Work

The work on predicting the success in MOOC programming assignments is sparse. However, there have been numerous attempts to find the relationships between the academic success, the success in programming assignments and the success in programming courses in normal classroom settings; with students’ profiles (based on demographics, learning strategy and motivation). In this section, we provide a few examples on these findings.

**Academic success and student characteristics:** Students’ personal characteristics, learning styles, and learning strategies were correlated with their academic success. Conscientiousness (Chamorro-Premuzic and Furnham, 2003; Busato et. al., 2000), directed learning style (Busato et. al., 1998; Richardson 1999), intellectual ability (Busato et. al., 2000; Minnaert and Janssen, 1999), achievement motivation (Busato et. al., 1998), and strategic learning approach (Cassidy and Eachus, 2000) had positive correlations with students’ success. On the other hand, neuroctism (Chamorro-Premuzic and Furnham, 2003), undirected learning style (Busato et. al., 1998; Busato et. al., 2000), and apathetic learning approach (Cassidy and Eachus, 2000) were shown to have detrimental effects on students’ academic success.

**Programming assignments/ Success in computer science education:** Specifically, for university level programming courses, students’ success was positively correlated with their maths background (Wilson, 2002; Wilson and Shrock, 2001; Bennedsen and Carpersen, 2005; Byrne and Lyons, 2001), comfort level (Wilson, 2002; Wilson and Shrock, 2001), hard work (Bennedsen and Carpersen, 2005), and attribution to success/failure to luck (Wilson, 2002; Wilson and Shrock, 2001); while students’ attribution to success/failure to task difficulty (Wilson, 2002; Wilson and Shrock, 2001; Bennedsen and Carpersen, 2005) and a non-converging learning style (Byrne and Lyons, 2001) were negatively correlated with students’ success in programming courses.

**Experiences in MOOCs:** There had been other attempts to measure/improve the learning experiences in MOOCs. For example, Li et.al (2015) showed that students’ perceived content difficulty is reflected from their video navigation patterns. Building upon their results, Sharma el. al. (2015) showed that displaying teacher’s gaze in a MOOC video results in a behaviour that corresponds to low perceived content difficulty.

However, these studies do not focus on a particular type of courses. Moreover, either these studies focus on the student attributes (learning style, strategy, demographics, motivation) or they focus on learning experience. In this contribution, we focus on the success in programming courses, and we analyse the assignment part of these courses. We propose to use the students’ submission behaviour, which is also automatically logged in MOOC platforms, in order to predict success.

# 3 Problem Statement

A preliminary analysis shows that out of more than 120,000 students’ assessments (from 5 programming MOOCs) 13% students failed. Moreover, 45% submit the assignments only once, 16% of these students could not score passing marks for the given assignment. In order to support these students, predicting their level of success at an early stage is necessary. Furthermore, to help the students who failed (or as predicted, will fail), we need to find out strategies that the passing students chose. Specifically, we investigate the following research questions:

**Question 1:** Can we predict success in programming assignments using students' submission behaviour only? As we said before, this is necessary to know if a student will succeed or not at an early stage to provide support.

**Question 2:** Can we predict whether a given submission would be the last submission for a student? In our preliminary analysis, we observed that 13% of the students fail after attempting to solve the assignments and its necessary to predict when the students give up.

**Question 3:** What are the different paths to succeed in programming MOOCs? This is necessary to know how students succeed in programming assignments to provide support to the student who face problems.

**Question 4:** How is the dropout related to the failure in programming assignments? If there is a relation between students failing in a given assignment and them dropping out, we can take a few measures to encourage the re-engagement.

# 4 Methods and materials

## 4.1 Course descriptions

We used five programming courses, to test our methods for predicting success and engagement in programming assignments. For maintaining the ANONYMITY, we will give only generic information in this version. Complete details will appear in the final version of the paper.

The five courses were the first courses for their respective paradigms (procedural, object-oriented, functional). Here are the common details about the courses:

1. All the courses were 7 weeks long, in terms of instruction.
2. All the courses, except Scala that had Java as a prerequisite, had no prerequisites.
3. All the courses, except Scala that had five graded programming assignments, had four graded programming assignments, scores from which would contribute to the final grade and achievement level of the students.

## 4.2 Variables

**First score:** the score that the student gets after the first attempt to the given programming assignment. **Time difference between two successive submissions:** the time difference between the two successive attempts to the same programming assignment. **Improvement in score:** the difference in the score between the two successive attempts to the same programming assignment. **Change in the program:** the difference in the submitted programs, between the two successive attempts to the same programming assignment. This is the sum of the number of lines added and number of lines removed. **Number of attempts:** the total number of attempts done by a student for a given programming assignment. We observed that 95% students attempt to solve a given programming assignment up to 10 times. Hence, we limit our analysis to those students who attempt up to 10 times to solve a given assignment.

**Dependent Variables - Final score:** the score that the students get for each assignment. We normalised the score, dividing it by the maximum attainable score for each assignment. **Next score:** the score that the students get after each submission. We will test our methods to predict the score for the next submission attempt. We normalised the score, dividing it by the maximum attainable score for each assignment. **Last Attempt:** We distinguished between the attempts for the same assignment for each of the students, whether it was succeeded by another submission attempt (last attempt) or not (not a last attempt).

## 4.3 Methods

In this contribution, we propose two prediction methods. The two methods differ by one factor: the first method, uses the complete data from each assignment submission; while, the second method, uses the predicted output from the previous attempt as an input to predict the success in the current attempt.

**All-in-one prediction:** This method (as shown in the Figure 1, left pane) uses the data from each attempt in order to predict the success. This is an iterative method, where every iteration uses the data available from all the previous attempts.

**Staircase prediction:** This method is hierarchical and uses the data from each attempt in order to predict success. The input to each level “i” is the data from the “ith” attempt and the predicted values from the “(i-1)th” level. In the case of a continuous independent variable, the predicted values are the output value of the classifier; while, in the case of a categorical independent variable, the predicted value is the probability of being classified in one of the categories.



Figure 1: The two prediction methods: left- all-in-on, right- the staircase model.

In both aforementioned methods, we used artificial neural networks and generalised additive models as the prediction algorithms.

# 5 Results

In this section, we present the prediction results. First, we give the prediction results for the final score, the next score, and whether the current attempt is the last attempt. Then, we will provide different strategies chosen by the students and its effect on their success. Finally, we present the relation between failing in a given assignment and dropout. For the prediction sections there were two steps: 1) we selected the best predictors, from the list of variables presented in Section 4.2, using a forward feature selection method; and 2) we used variables selected in the first step to predict the score (or, whether the current submission is the last submission) after each assignment submission for a given assignment.

## 5.1 Final score prediction

First, we present the prediction results for the score that the students are awarded after each assignment submission. Using the forward feature selection, we observed that the model with the “first score” and the “improvement in the score” had the best results. Next, we predict the final score after each attempt for a given assignment. Figure 2 shows the prediction quality after each attempt for all the assignments. We observed that after the third attempt for a given assignment, we achieve decent prediction accuracy.

## 5.2 Last attempt prediction

Next, we will present the prediction results for the fact whether the current submission is the last submission for a given assignment. Using the forward feature selection, we observed that the linear model with “first score”, “improvement in the score” and “change in the program” had the best results. Next, we predict the current attempt to be the last attempt for every given submission. Figure 3 shows the progress of the prediction quality. We can see from Figure 3, that after the third attempt for a given assignment, we achieve decent prediction accuracy.

## 5.3 Next score prediction

Next, we will present the prediction results for the score for future (next) submission attempt for a given assignment. Using the forward feature selection, we observed that the linear model with “first score”, “improvement in the score” and “number of previous attempts” had the best results. Next, we predict the current attempt to be the last attempt for every given submission. Figure 4 shows the progress of the prediction quality. We can see from Figure 4, that after the fourth attempt for a given assignment, we achieve decent prediction accuracy.

## 5.4 Different paths for success

We identified three different classes of students submitting programing assignments: 1) those who submitted only once; 2) those who submitted more than once but only a few times (submitted twice); and 3) those who submitted more than twice but up to 10 times. Following are the details about each of these classes:

**Submitting once (one-timers):** Among all the students’ assessments; 45% of the students submitted only once, 16% of them failed; and rest of them passed.

**Submitting twice (thinkers):** Among all the students’ assessments; 21% of the students submitted twice for a given assignment, 9% of them failed; and rest of them passed. We observed a significant difference between the two groups (failed and passed) on the time difference (t(5008.33) = -9.85, p < .001) and program change (t(4488.637) = -8.29, p < .001). This shows that among those students, who submitted only twice, the students who did small and quick changes had a higher score. One plausible explanation for this could be that for the first submission, the students got a very small error but the ones who failed could not understand the automatic message and made bigger changes (for which they needed more time) to their previous programs and failed to score the necessary marks.

**More than 2 but up to 10 submissions (trial-and-error):** Among all the students’ assessments, 32.5% of the students submitted more than twice but up to 10 times for a given assignment; 11% of them failed; and rest of them passed. For the students in this class, we found a difference between the two groups (failed and passed) on program change (t(4738.717) = -5.38, p < .001). This shows that the students who introduced smaller changes than others had a higher score. This shows a typical “trial and error” strategy.



Figure 2: Prediction results for final score



Figure 3: Prediction results for last attempt

****

Figure 4: Prediction results for next score

****

Figure 5: Relation between failure and dropout

## 5.5 Effect on dropout

Next, we show the last activity of the students who failed in a given assignment (Figure 5). We observed that 36% of the students have their last activities as a submission to a programming assignment. We distinguished between the first submission (Assignment Submission, 12%) and submitting more than once (Assignment Re-submission, 24%). This shows that most of the “one-timers” quit the course after failing in one of the assignments. While most of the failed “thinkers” and “trial-and-error” students try to get help from the forums or the lectures and they re-submitted, but finally they quit the course.

# 6 Discussion and conclusions

We showed that, it is possible to detect students’ success in programming assignments at a very early stage. This is a key process if one chooses to provide any kind of support to the students who are predicted not to perform well, for example, suggesting extra course material, giving detailed feedback on the submitted programs and pointing the students to correct forum posts.

The facts that we were able to predict the next score and the last attempt to an assignment might give detailed guidelines for designing a support system that is beyond the mere error messages from an online judge. For example, if a student is predicted to give up submitting any further and his next score will not give him a passing grade, one can examine her previous submissions in more details than a set of unit tests; and provide appropriate support in terms of more learning material or motivating her for another submission with specific hints based on her mistakes in the previous submissions.

Moreover, since we can also predict the final score for an assignment after only a few submission attempts, one can use these predictions to enable the feedback systems to the teachers. Usually there are 2-3 weeks between the open time and submission deadline for an assignment. If many students are predicted to get lower grade, during these weeks the teacher can provide additional material to the students and/or discuss the problem statement in detail at the forums.

We have also shown the relationship between the students failing in an assignment and dropping out of the course. Figure 5 shows that 36% of the failed students (who attempt at least once) had last actions as assignment submissions (or re-submissions). This shows that failing in an assignment can have detrimental impact on students’ engagement levels. One could use the predictions, we proposed in this contribution, to design a support system (automatic or with the intervention from the teaching staff) to prevent the engagement levels from dropping.

Using the proposed variables, in this contribution, we were able to distinguish between the different paths, which the students chose in order to succeed in programming assignments. Most of these variables (all except the difference in the submitted program) are generalizable for the assessment in other type of courses as well.

Furthermore, we distinguished different approaches that the students chose to succeed in programming assignments. The three major strategies are: one-timers, thinkers and trial-and-error. We saw that one-timers have the most failure rate. This also reflects the relation between the disengagement and failure in one assignment. One can support these students by automatically analysing the program they submitted and providing more information about their mistakes than a simple error message from the online judge.

Finally, we observed that 32.5% students follow trial-and-error strategy to succeed in an assignment. These students might not learn the basic concepts even though they pass the course. For such students, one might suggest more learning material based on the predictions about their last attempt and their next score. If a student is predicted to have another attempts and not to achieve a passing score, the support system could suggest extra reading and re-watching a few lecture videos to help the student understand the programming concept and not to approach programming as a trial-and-error process.

# References

**Chamorro-Premuzic, T. and Furnham, A.** (2003). *Personality predicts academic performance: Evidence from two longitudinal university samples*. Journal of Research in Personality (37). Elsevier.

**Busato, V., Prins, F., Elshout, J., Hamaker, C.** (1998). *The relation between learning styles, the Big Five personality traits and achievement motivation in higher education*. Personality and Individual Differences (26). Elsevier.

**Busato, V., Prins, F., Elshout, J., Hamaker, C.** (2000). *Intellectual ability, learning style, personality, achievement motivation and academic success of psychology students in higher education*. Personality and Individual Differences (26). Elsevier.

**Wilson, B.** (2002). *A Study of Factors Promoting Success in Computer Science Including Gender Differences.* Computer Science Education (12).

**Bennedsen, J. and Caspersen, M.** (2005). *An Investigation of Potential Success Factors for an Introductory Model-Driven Programming Course*. ICER, ACM

**Wilson, B. and Shrock, S.** (2001). *Contributing to success in an introductory computer science course: A study of twelve factors*. SIGSCE. ACM

**Cassidy, S. and Peter Eachus, P.** (2000). *Learning Style, Academic Belief Systems, Self-report Student Proficiency and Academic Achievement in Higher Education*. Educational Psychology (20). Taylor & Francis.

**Richardson, J.** (1995). *Mature students in higher education: II. An investigation of approaches to studying and academic performance.* Studies in Higher Education (20). Taylor & Francis.

**Minnaert, A. and Janssena, P.** (1999). *The additive effect of regulatory activities on top of intelligence in relation to academic performance in higher education*. Learning and Instruction (9). Elsevier.

**Byrne, P. and Lyons, G.** (2001), *The Effect of Student Attributes on Success in Programming.* SIGCSE bulletin. ACM.

**Li, N., Kidziński, Ł., Jermann, P., Dillenbourg, P.** (2015). *How Do In-video Interactions Reflect Perceived Video Difficulty?*. In Proceeding to 3rd European MOOCs Stakeholders Summit, Mons, Belgium.

**Sharma, K., Jermann, P., & Dillenbourg, P.** (2015a).*Identifying Styles and Paths toward Success in MOOCs.*In Proceedings of the 8th International Conference on Educational Data Mining.

**Sharma, K., Jermann, P., Dillenbourg, P.** (2015b). *Displaying Teacher's Gaze in a MOOC: Effects on Students' Video Navigation Patterns.* In Proceedings of the 10th European Conference on Technology Enhanced Learning.