Educational Data Mining

Students time consuming in programming course

Introduction

Programming courses have a lot of exercises to do. When doing exercises most of students will get stuck at some point. Our research team was willing to discover can we find out a time that is optimal to use in one exercise. During this educational data mining course we noticed that there has been done research about difficulty, how to improve learning etc. and couple of articles about time consuming. This is partly how we decided that we want to know how students use time in programming courses. Of course we also were interested in how much time per exercise is optimal because we all study physics or computer science and have exercises that can vary from easy to very tedious.

The data, that we use, is from online programming course and was originally published in **Automatically Detectable Indicators of Programming Assignment Difficulty** that was published at the 15th Annual Conference on Information Technology Education SIGITE'14. The first 4 columns are generic information entries containing the user id, gender (M/F), year of birth, and whether the user had previous programming experience (true/false). After the generic information entries, there are 11 columns for each of the assignment given in the course. The other columns are:

- WORKED_ON_{EXERCISE}, whether student worked on the exercise (Y/N)
- SUBMITTED_{EXERCISE}, whether student submitted the exercise (Y/N)
- SECONDS_SPENT_ON_{EXERCISE}, number of seconds spent on the exercise
- PERCENTAGE_COMPILES_{EXERCISE}, percentage of states that the student was in that compiled
- DIFFICULTY_{EXERCISE}, difficulty of that assignment as indicated by the student
- EDUCATIONAL_VALUE_{EXERCISE}, educational value of that assignment as indicated by the student (each student responded to the question "How Educational Was This Assignment" on a scale from 1 to 5 -- this was not used in the original study, but may be interesting for you :));
- NUM_OF_STATES_{EXERCISE}, number of different source code states that the student visited (can be thought of as "steps taken")
- SECONDS_IN_COMPILING_STATE_{EXERCISE}, how many seconds did the student spend in a state that did compile
- SECONDS_IN_NON_COMPILING_STATE_{EXERCISE}, how many seconds did the student spend in a state that did not compile
- LINES_OF_CODE_{EXERCISE}, how many lines of code did the students last solution have
- FLOW_CONTROL_ELEMENT_COUNT_{EXERCISE}, amount of flow control elements (if, while, for, ...) in the solution how many lines of code did the student's last solution have

Research

When doing work on problems in programming and in general it is said that it is better to take short breaks once a while. We wanted to know if there could be found a relationship between progress in solving a problem and time. A quite known technique is the Pomodoro technique [1] there one works for 25 min and then takes a brake for 5 min and then one repeats the cycle. The question that comes to mind is are the times chosen only to fit 2 cycles in a hour or is there a deeper understanding in the 25 to 5 relation. Therefore we hypothesized that there should be found a limit in time, when solving problems, where the contribution given to solving problems did start to decelerate and a break could be in place. The reason to this technique to have a solid ground could lie in the way the working memory works[2]. The brain can not hold all information in the working memory or short time memory as it is also called. The brain transfers the information to the long term memory in rest. When solving problems one faces many new things, then to be able to solve a large problem one must usually have clear all the details. If no break is taken new things will occupy the working memory and few can be transferred to the long term memory.

Approach methods

We approached this problem by mining our data with Python. We used mainly Python Pandas package and Matplotlib to plot our results. We also used polynomial fitting to get the predictive model. We started by gathering the time usage data. After that we mined the difficulty and educational value data. We also made time, difficulty and educational value histograms for every single exercise that the course had. This is where we started:

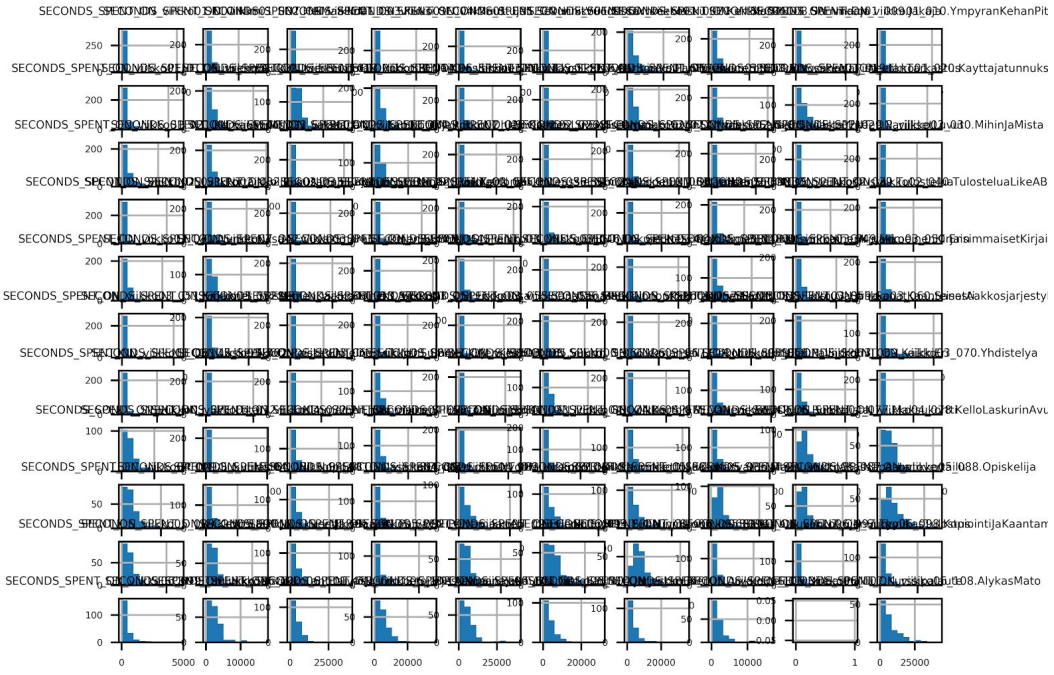

Figure 1. All the exercises and their time usage of the programming course

First results

From the exercise data we could see some similarity between the mean values of time used and times that students used. Here is an example:

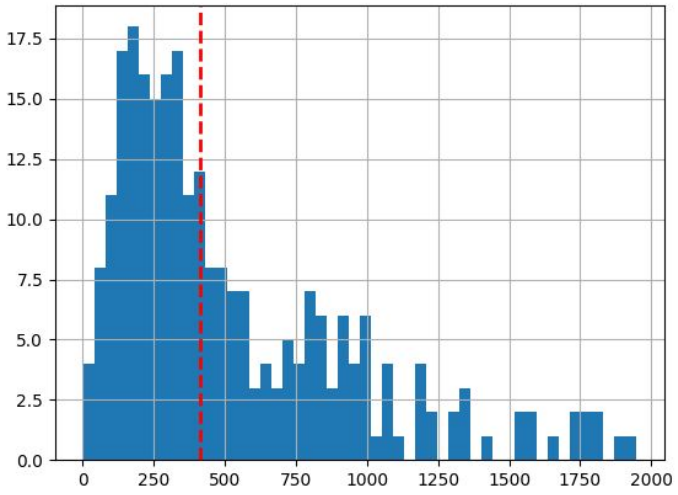

Figure 2. Seconds spent in exercise 58. Toistuva sana

This is from exercise 58. “Toistuva sana”. You can see that most of students did the exercise under the median. From the same exercise we can clearly see that students “liked” it and it was rather easy, difficulty value was given by students between 1-5:

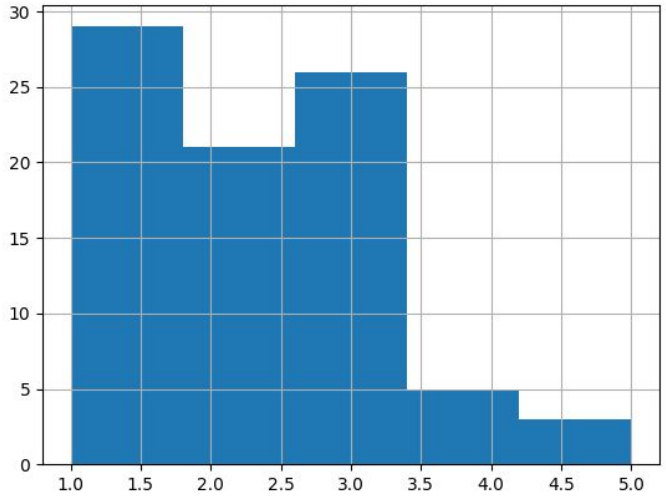

Figure 3. Difficulty for exercise 58. Toistuva sana

Students also gave the educational value for exercises between 1-5. We noticed that exercises like this that was rather easy the students gave usually 3.0 for educational value.

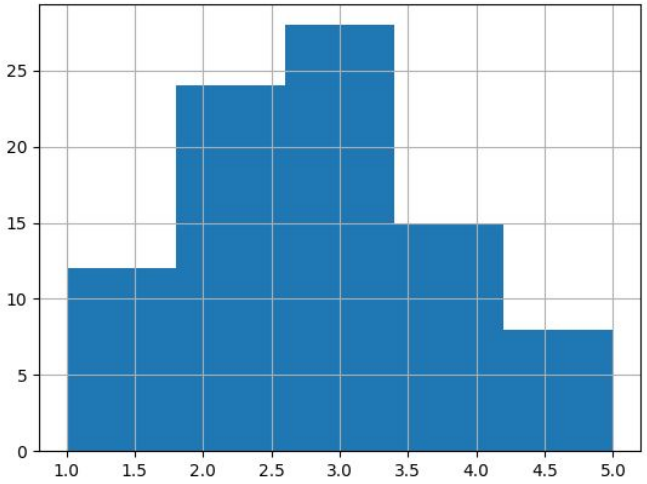

Figure 4. Educational value for exercise 58. Toistuva sana

Main results

Our results

After mining all the data we took all the mean values of time that were used in exercises and the educational + difficulty values that were given for the exercises (one dot represents one exercise). The results were quite surprising, there is a clear point where most of the students made the exercises:

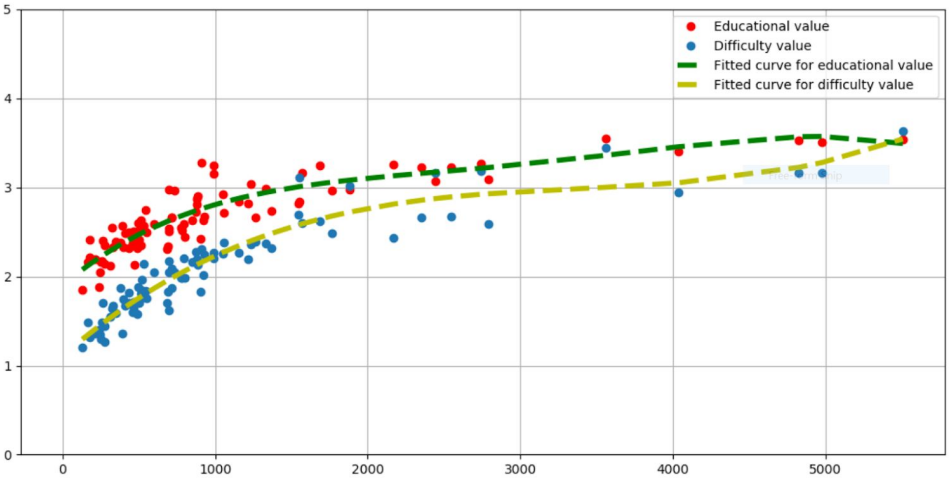

Figure 5. Main result

As one can see from Figure 5 there is a clear pattern, students think that the learning and difficulty increases almost linearly until 1500s. After 1500s the rate is slowed down and putting more effort to a problem does not seem to increase. It is good to keep in mind the fact that these are factors that are given by the students. We presume that the data is so large and this being the mean of the results the individual differences disappear.

Statistics of the result

We modeled students' reported educational value and difficulty to the time they spent on the exercise with polynomial regression. The mean squared error on the educational value was 0.15 and for the difficulty value 0.30.

For an exercise with a target completion time of 5 minutes the expected educational value would be 2.3 and the expected difficulty 1.5. For an exercise that should be completed in 25 minutes the educational value is expected to be 3.0 and difficulty 2.6. After 25 minutes the growth in difficulty or educational value grows the expected time used significantly and the model is not very valuable. For an hour long exercise the expected educational value is 3.4 and difficulty 3.0.

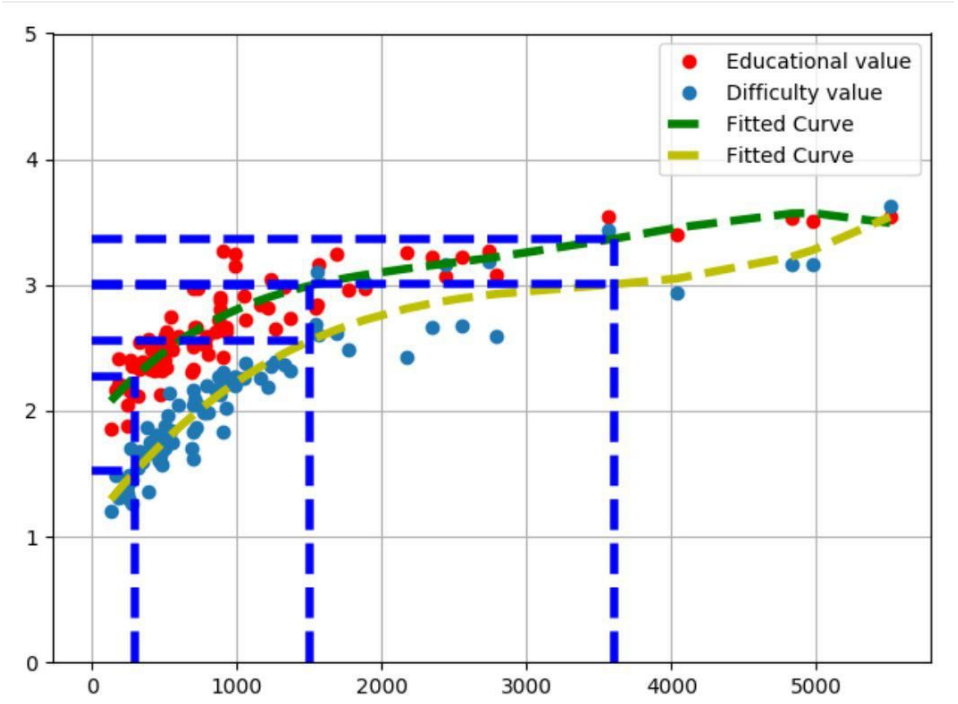

Figure 6. Statistics

Also we can see that most of the students made the exercises in under 25 minutes. Therefore our hypothesis of the Pomodoro method where you make work 25 minutes and keep 5 minutes break is similar what we got.

After 80 minutes the educational value starts turning down and difficulty grows. So most valuable exercises take around 80 minutes to make, but we assume that this includes the breaks what we mentioned.

Discussion

We were happy to see that our hypothesis was right. From our result you can see that it is better to have a break after 25 minutes so that you work most efficient way. Also we can see that harder the exercise is the more valuable for educational use it is until certain point.

It would be better to actually require the students to give educational value and difficulty value and also to gather data from many courses. These results are rather good taking account that not every student gave educational value or difficulty value. With bigger datasets we could probably confirm better the 25 minute “Pomodoro” break or we could confirm that exercise that takes 80 minutes is the most valuable for educational use.

The time data doesn’t really tell was the exercise done in five minutes and submitted after 20 minutes break. If we could confirm that these times were really times that the students were writing code then we could be more sure of the results.

A interesting thing would be to investigate deeper into typing when doing exercises. When considered that a person types fast it probably means that the text is produced from previous knowledge [2] and this is not loading the working memory. Then when the pattern slows down, the person is most likely thinking more, and it it this face in the process of solving problems that we have seen that a approximate time of 25 min is the most efficient. If gathered this data one could improve ways in learning to be more efficient for the students.

All of our code that we used can be found from Github. The first visualisation of the data is visualisation.py, all the exercise plots are made with groupedViz.py. The main results are in file groupedValues.py.

References

[1] https://francescocirillo.com/pages/pomodoro-technique
[2] http://adsabs.harvard.edu/abs/2008Sci...321..851B

Appendix

Our Github repository: https://github.com/pihvi/edutime
Our all plottings of the data: https://github.com/pihvi/edutime/tree/master/plots
Our results as pictures:
https://github.com/pihvi/edutime/blob/master/reports/seconds_spent.md