Increasing the Prediction Power of Moodle Machine Learning Models with Self-defined Indicators

https://doi.org/10.3991/ijet.v16i24.23923

Tibor Fauszt[1](✉), László Bognár[2], Ágnes Sándor[1]
[1] University of Applied Sciences, Budapest, Hungary
[2] University of Dunaújváros, Dunaújváros, Hungary
fauszt.tibor@uni-bge.hu

Abstract—Starting with version 3.4 of Moodle, it has been possible to build educational ML models using predefined indicators in the Analytics API. These models can be used primarily to identify students at risk of failure. Our research shows that the goodness and predictability of models built using predefined core indicators in the API lags far behind the generally acceptable level. Moodle is an open-source system, which on the one hand allows the analysis of algorithms, and on the other hand its modification and further development. Utilizing the openness of the system, we examined the calculation algorithm of the core indicators, and then, based on the experience, we built new models with our own indicators. Our results show that the goodness of models built on a given course can be significantly improved. In the article, we discuss the development process in detail and present the results achieved.

Keywords—machine learning, learning analytics, online education

1 Introduction

Numerous studies have been conducted in recent decades on the construction of educational ML models and the evaluation of their predictive power [1], [2], [4], [9], [12], [15]. The number of available literatures on this topic is huge. At the same time, most of the research deals with the construction of models and the analysis of their efficiency, but there is a few research that deals with the predictive ability of ML models embedded in an LMS system, and with the improvement and optimization of this ability [7].

This issue is of particular importance as a Learning Analytics tool integrated into an LMS system may be suitable for building self-learning ML models. Over time, more and more students are attending a particular university course, so more and more data is available to teach the models, and so we can get better and better models.

One of the great novelties of Moodle version 3.4 is that such a self-learning system can be built using the Moodle Analytics API. The tool is a great initiative, especially for purely online MOOC courses, however, using it as a black box can come as a surprise to the user.

Based on educational ML models, predictions can be used to predict students' performance or identify students at risk of failure. [6], [8], [10], [11], [14], [16], [19]. The

instructor can warn these students of the danger of falling and can provide additional assistance in mastering the curriculum. However, in the case of an ML model with low predictive capabilities, several false alarms may occur. The system may also identify students at risk of failure who are actually working diligently and may not identify students who need help. It is likely that an ML model will never work with 100% accuracy, however, the goodness of the models can be significantly improved with proper design.

Using core indicators found in the Moodle Analytics API (MAA), we constructed different models based on data from a specific course and examined their predictive power. Different metrics (Accuracy, Recall, Fallout, F1 score, normalized Matthew Correlation coefficient, etc.) were used to test the predictive power of these core models. The results were published in a previous study and showed that the reliability of a Moodle core model with a small number of cognitive indicators may unfortunately fall far short of the desired level [3].

The question arose as to what the reason for the poor predictive power could be, and how ML models with higher reliability values could be built even in cases where the number of indicators that can be built into the model is small. In the first step, we examined the operation of the Analytics API, a model based on the calculation of core indicators, and its algorithm. Based on the experience, in the next step we built several different models with self-defined indicators for the same course and examined the predictive power of the new models. Our results show that the reliability values of the new models with self-made indicators have significantly improved compared to the core models. The improvement was basically achieved with two modifications. On the one hand, we introduced a new calculation method for calculating the values of the indicators, and on the other hand, we increased the number of indicators. In the following, we discuss in detail the process of analysis and modification, and present the results achieved. In the studies, Logistic regression was used to teach and evaluate the models. (MATLAB 2008, release 2018b).

2 Course description

The course on which we created our models basically included the following curriculum elements: Lecture videos, Minitab videos (videos for problem solving with statistical software), PDF lecture notes, Books of solved exercises, Quizzes for Self-testing. The individual Moodle resources and activities (components) that represented these curriculum elements were:

- **Page resource** for Lecture videos and Minitab videos,
- **File resource** for PDF lecture notes,
- **Book resource** for Books of solved exercises,
- **Quiz activity** for Quizzes for Self-testing.

The course was attended by 56 full-time students at the University of Dunaújváros. The course was a blended learning course, the form of education was fully online with additional teacher assistance. The content of the course was Applied Statistics and was

divided into 15 chapters. Not all chapters contained all the resources, some chapters had a Page-type resource only, and some chapters had several resources. Thus, in total, a Page type resource occurred in 15 chapters, a File type resource in 7 chapters, a Book type resource in 7 chapters, and a Quiz type activity in 14 chapters. During the course, students had to solve 4 midterm tests at specified times. They scored 25 points on each test. To complete the course, students had to achieve a total of at least 70 points.

3 How the analytics API works for core indicators

3.1 Classification and grouping of core indicators

The core indicators that are part of the Analytics API are the cognitive-depth and social-breath indicators. These indicators are defined for all Moodle resources and activities in the system. The schematic diagram of the model is shown in Figure 1.

Fig. 1. Cognitive Depth and Social Breadth (Source: Moodle documentation https://docs.moodle.org/310/en/Learning_analytics_indicators)

Moodle resources are elements in the system that denote some type of learning material or a tool for grouping learning materials. These can be files, folders containing files, pages displaying study material, URL links, and so on. Moodle activities are tools that support student activities. They facilitate communication, assignment, self-testing, creating your own student databases, etc. The terms resource and activity are hereinafter collectively referred to as components, in line with the terminology used in the Moodle database to identify these elements. The model places each component in two-di-

mensional space. The vertical dimension is cognitive depth, and the horizontal dimension is social breadth. The figure shows the levels of cognitive depth and social breadth of each component. For example, taking the chat and forum components, these components are in row 4 and column 2 of the two-dimensional table. Thus, the cognitive depth level of these components is 4, and the social breadth level is 2.

Cognitive Depth. There are 5 levels of cognitive depth from 1 to 5. 1 is the least deep and 5 is the deepest level. Each level is defined based on student activities. According to the model, learner activity belonging to cognitive depth level 1 is when the learner has only viewed the resource or activity details. Cognitive depth level 2 means when the learner has submitted content to the activity, cognitive depth level 3 when the learner has viewed feedback from an instructor or peer for the activity, cognitive depth level 4 when the learner has provided feedback to the instructor or a peer within the activity. Finally, the 5th, deepest cognitive depth level when the learner has revised and / or resubmitted content to the activity.

This flowchart of the conceptual model for the components and the type of student activities performed on them is shown in Figure 2. According to Moodle documentation, an algorithm for calculating each core cognitive-depth indicator is based on this model. Although a sophisticated calculation model appears from the figure, decoding the algorithm that computes the core cognitive-depth indicators, we have seen that the actual calculation is different, much simpler. The algorithm is analyzed in detail in the next section. Presumably, this conceptual model will be elaborated and further developed in later versions.

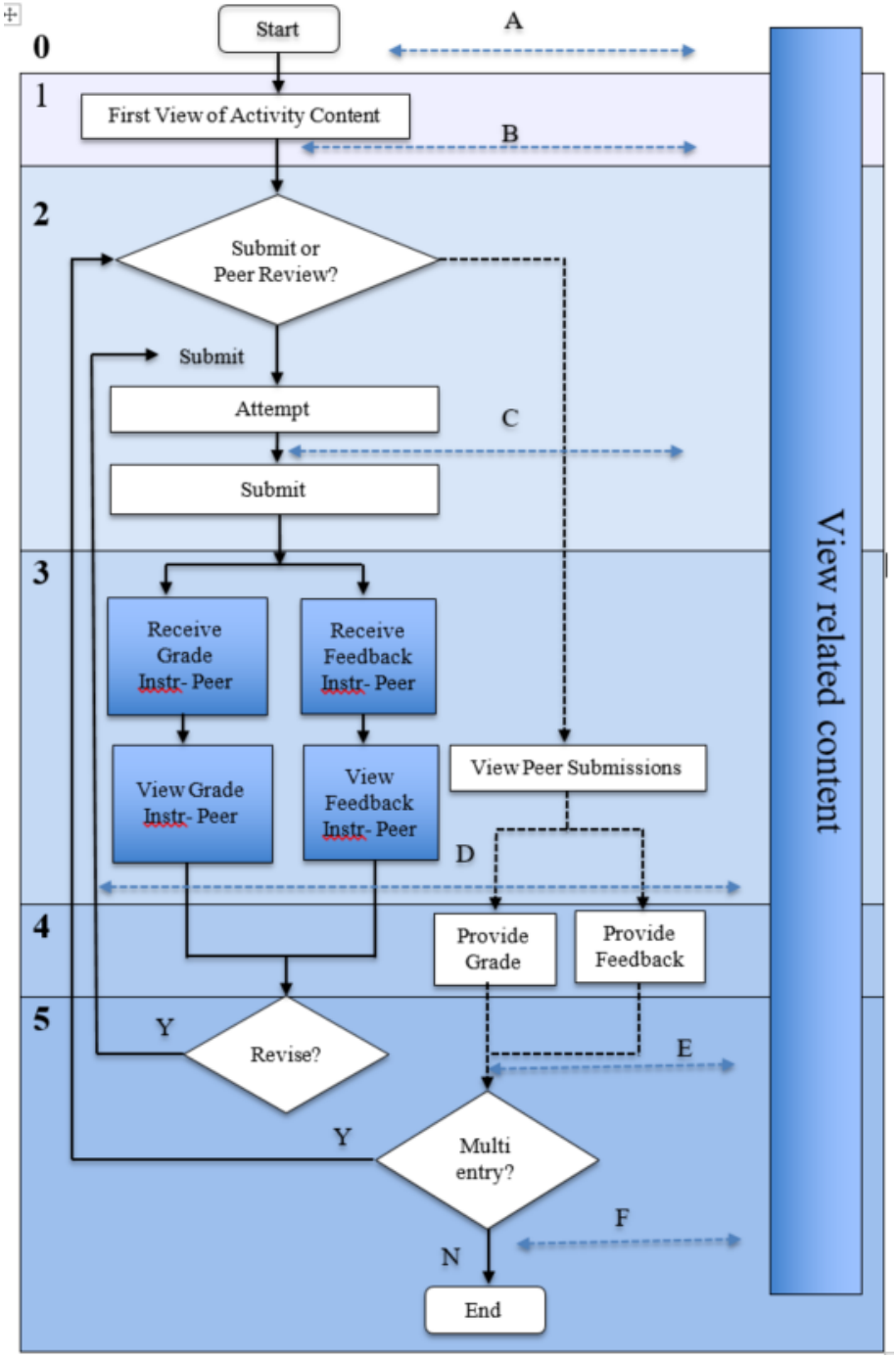

Fig. 2. Moodle Core Cognitive Indicators Calculation Conceptual Model Moodle Learning Analytics Online Workshop, Recording and Slides. Jan. 2020

Social Breadth. The two values of social breadth, based on student activities, are defined as follows: 1 if the learner has not interacted with any other participant in this activity, 2 if the learner has interacted with at least one other participant. The documentation also lists 3 more levels, but these have not yet been implemented in the system.

3.2 Calculation of core indicators

Moodle is an open-source system developed in PHP by the Moodle community. Each component has two core indicators in the system that are physically PHP source code files and they define two classes. The source code for each indicator class can be found in the *cognitive_depth.php* and *social_breadth.php* source codes in the different folders for that component. The *get_cognitive_depth_level* method of the class specifies the cognitive depth of the given component from 1 to 5, and the *get_social_breath_level* method specifies the social breadth value from 1 to 2. The calculation of both types of indicators is based on the same principle. The process of calculating the indicators is presented through the algorithm of the cognitive-depth type indicator.

A student activity refers to the use of a component. These activities are recorded in a *logstore_standard_log* table that is part of the Moodle database. Among other things, it records the time of the activity related to the component, the ID and type of the component, and the type of action. Three of the types of interactions play a key role in the calculation of indicators: *submitted, replied, and viewed*.

Other log entries that are important for the calculation are entries of *any write* and *any log* types. So, there are entries that are of the write type, i.e., the student has done some written activity, such as responding to a log entry. All other interactions belong to the type of *any log* entries.

The values of each core-cognitive indicator are basically calculated based on log entries and cognitive levels. The levels control the algorithm for calculating cognitive indicators, the code of which is found in the *cognitive_calculate_sample* method of the *community_of_inquiry_activity* class. Its flow chart is shown in Figure 3.

This algorithm, which determines the value of the indicator, essentially generates a ratio. Namely, it gives the ratio of the number of actual interactions performed by the learner on a given component to the number of possible interactions related to the component. The ratio is normalized by the algorithm between -1 and 1, because in the optimization algorithms used by Moodle, the values of the indicators must fall between these two values (*maxCognitiveLevel* = 1, *minCognitiveLevel* = -1).

The number of possible interactions is indicated by the number of components listed in the *useractivities* list. As an example of the process of calculating a *page_cognitive* indicator, which has a cognitive depth level of 1, the value of the indicator is as follows. The initial value of the indicator is *score* = -1. Assuming 4 page type components, the value of the *scoreperactivity* variable is:

$$scoreperactivity = \frac{maxCognitiveLevel - minCognitiveLevel}{count(Useractivities)} = \frac{1-(-1)}{4} = 0.5 \qquad (1)$$

The *useractivities* variable contains all possible page-type components that the learner can view. It takes each component in turn, retrieves the value of the cognitive

depth (*potentiallevel* = 1) for the page component defined in the page-cognitive indicator, and then calculates the *scoreperlevel* sub-score:

$$scoreperlevel = \frac{scoreperactivity}{(Page)\ potentiallevel} = \frac{0.5}{1} = 0.5 \quad (2)$$

It then examines whether the page resource (which has a cognitive level of 1) had any type of student activity and any related log entries (*any_log*). If so, the value of the indicator (*score*) increases by 0.5 *scoreperlevel*. If it was not, you get a sub-score of 0, the value of the indicator does not increase. Assuming there was an interaction:

$$score = score\ +\ scoreperlevel * 1 = -1 + 0.5 * 1 = -0.5 \quad (3)$$

In extreme cases, if there was an interaction for each possible page component (4 in our example), then the value of the indicator (*score*) will be the maximum 1, if there was no interaction at all on any component, then the value is the minimum value set for the initial value of the indicator. It will be -1. Assuming there was an interaction on all components:

$$score = -1 + 4 * scoreperlevel * 1 = -1 + 4 * 0.5 * 1 = 1 \quad (4)$$

A slightly more nuanced value can be obtained for components at a deeper level. The cognitive depth of a quiz-cognitive indicator for a quiz-type component is 5. Taking four possible components, the maximum sub-score that can be given will be as follows.

$$scoreperlevel = \frac{scoreperactivity}{(Quiz)\ potentiallevel} = \frac{0.5}{5} = 0.1 \quad (5)$$

If all quiz components had student activity and were of the *'submitted'* type in all cases (cognitive level 5 in the algorithm), the value of the indicator will be the same as for the page type, cognitive level 1 component.

$$score = -1 + 4 * scoreperlevel * 5 = -1 + 4 * 0.1 * 5 = 1 \quad (6)$$

However, other types of activity may occur with quiz-type components. It can be e.g., 'v*iewed'* which means the student did not pass the test, just viewed it. In this case, you will receive a reduced score for this activity, according to cognitive level 3 in the '*viewed'* branch of the algorithm. That is, the value of the indicator increases by 3/5 of the maximum possible sub-score. Based on four possible tests, if all tests have only been viewed by the student, the value of the indicator will be:

$$score = -1 + 4 * scoreperlevel * 3 = -1 + 4 * 0.1 * 3 = 0.2 \quad (7)$$

In the case of a test, however, the log entry can also be *'abandoned'*, which is an entry of type *any_log*. This entry has a level 1 branch, so the value of the indicator on this branch of the algorithm increases by 1/5 of the possible sub-score. Thus, in general, if there was an interaction with each possible component, but it did not always correspond to the cognitive level defined in the indicator defined for that component, the value of the indicator will be less than the maximum value. Similarly, the value of the

indicator will be less than the maximum value if there was no interaction with all components.

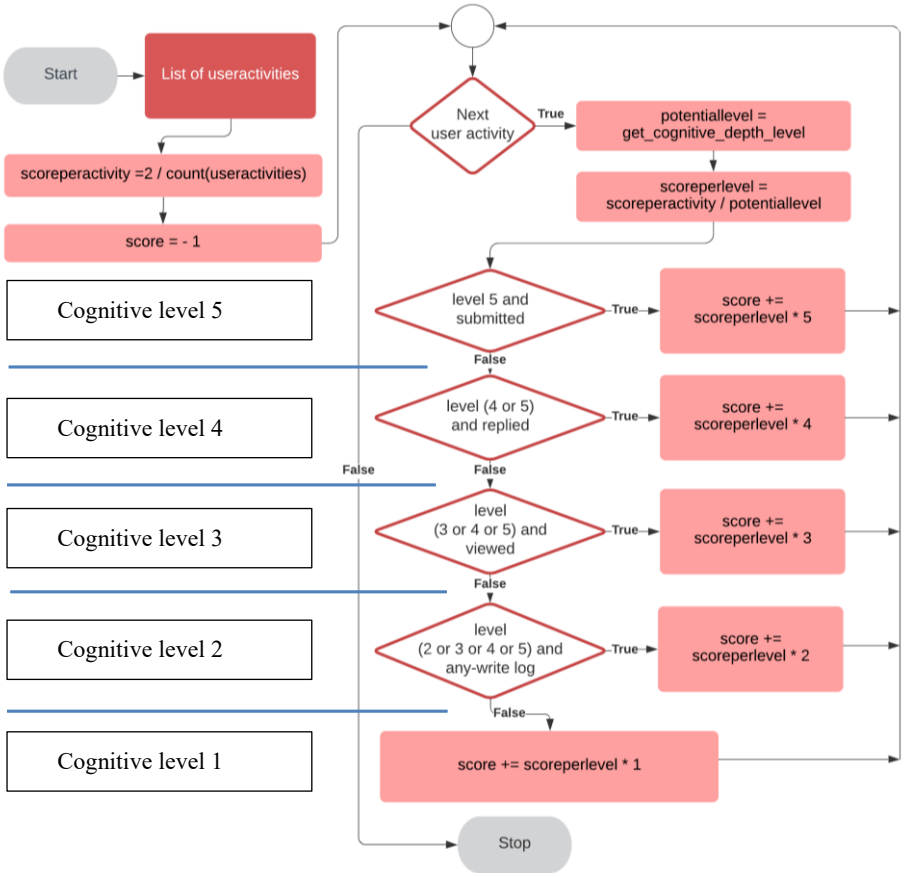

Fig. 3. Flowchart for calculating Moodle Core Cognitive indicators

4 Experiences, opportunities for improvement

Models that can be created in the API can be site-level or course-level models. For site-level models, the value of the indicator for the components is calculated based on the interactions performed in all courses taken by the student. For these models, a given component may represent different learning material in different courses. For example, a page resource is a common element in the system that can contain text, images, but can also display a video, or navigate to another page with links. Therefore, in the case of a site-level model, we cannot say exactly what type of learning material the interaction given to this type of component refers to, and how much that learning material contributed to the completion of the course. While these site-level models may provide an overall picture of student activity, we believe they are not suitable for forecasting.

However, it should be noted that the analysis of site-level models was not addressed in the present research.

Another question is how well the interactions performed on the page, URL, file type components correlate with the success of the course. A file can be downloaded, printed, and a URL can be bookmarked so that it can be viewed later without logging in. The value of the core indicator for such a component can only indicate in the model that the student has viewed these components. What we did with the content, such as printing a file and then learning from it or navigating to a page marked with a URL and what activities it did there, we no longer have information about. However, components that do not lead out of the system cover interactions that the student can only perform in the system have significant potential to improve the goodness of the model. Such components e.g., the tests. In one study, we showed that self-assessment tests play a prominent role in an online course [4]. They challenge students, give feedback on progress, motivate. In addition, all log entries related to their use are available in the log table. Therefore, when properly applied, they can play a significant role in improving the predictive power of models.

Another factor that fundamentally affects the goodness of the model is the determination of the values of the indicators. The calculation of the core indicators is basically the same for all components, so that if the student has made even a single interaction for each possible component, based on which he gets the maximum sub-score for the interactions, the value of the indicator will be the maximum 1. For a quiz-type indicator, this means that if you have submitted all the self-check tests once, you will receive a maximum indicator value of 1. Even if you gave incorrect answers to all the questions or did not view the questions. This calculation method does not seem logical for quiz-type indicators. The value of the indicator should be as closely related to the success function as possible.

This computational method is probably due to the fact that the system was designed to work on all the resources and activities in it, to provide results in some way based on the interaction on the component. However, too general operation seems to come at a price, it is not possible to build a reliable model in the system, the reliability of the models falls short of the expected level.

Another factor that has a significant impact on the goodness of the model is the number of indicators. For each course, we may work with few components. The components used are exhausted in the book, file, video pages, self-tests. Of course, these courses can be supplemented with components that help communication, but basically these components are the ones that support cognitive deepening, learning. The predictive power of a model with some (4-5) indicators is unlikely to be good.

5 Our model, the changes introduced

5.1 Calculation of indicators

As we have seen, the value of core indicators in the Moodle Analytics API is calculated using a general algorithm. This is based on whether there has been an interaction

with that component at least once. A more accurate picture of the degree of activity associated with a given component can be obtained by considering how many cases there were interactions on that component. Therefore, we created a new calculation in which, for each component, we determined the cut-off value of the number of interactions, above which we gave the maximum value of 1 as the value of the indicator belonging to the component. This number was the *Average of Total Attempt of User Activity* (*AVGTAUA*) for the component. For below-average interactions, the ratio of the number of interactions to the cut-off value was given as the indicator value. These indicators are called *ATTempt* (*ATT)* type indicators. For each component, we distinguish between indicators labeled *Page ATT, File ATT, Book ATT*, and *Quiz ATT (PATT, FATT, BATT, QATT)*. The flow chart of the calculation is shown in Figure 4.

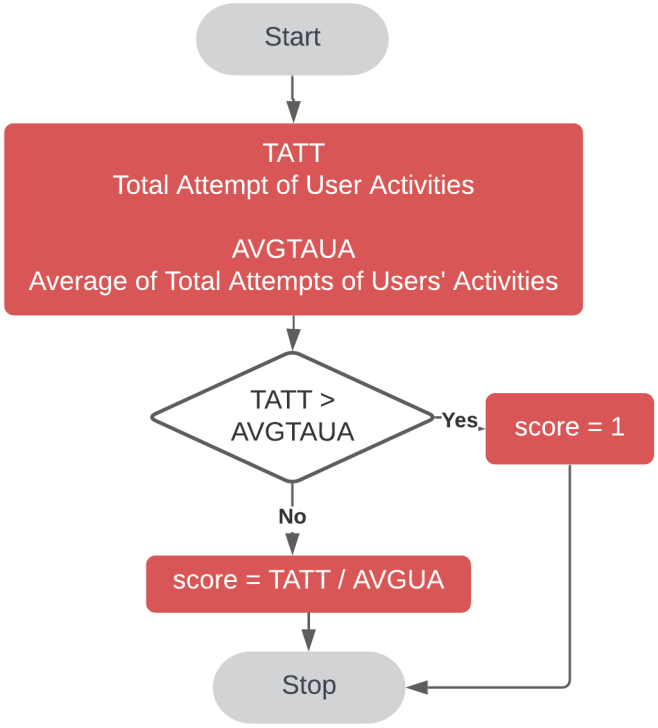

Fig. 4. Flowchart for the calculation of TATT indicators

In addition, the *Quiz MaxGRade (QMGR)* type indicator was introduced for the test type component. It is not enough information how many times a student has completed a self-assessment test. It is also very important how well he did this. In the defined calculation method, we considered the ratio of the sum of the student's *Total of Best Grades (TGG)* on the tests and the sum of the *Total of Achievable Maximum Grades (TAMG)* on the tests. The flow chart of the *QMGR* indicator calculation is shown in Figure 5:

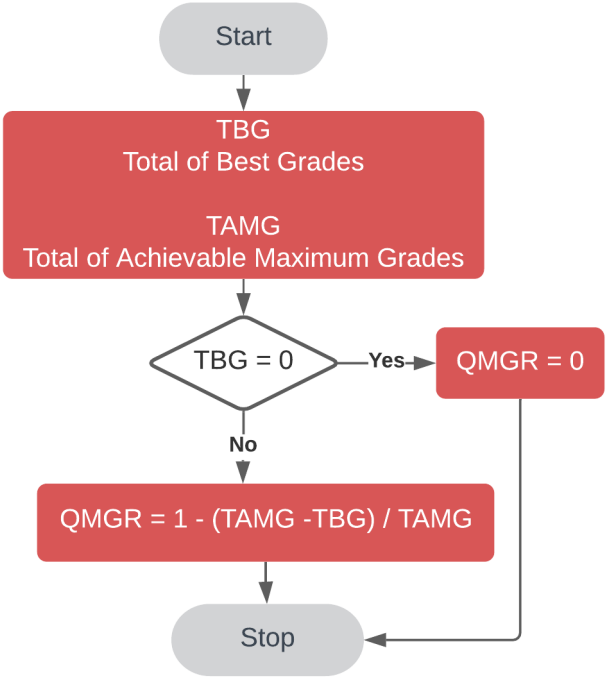

Fig. 5. Flowchart for the calculation of QMGR indicators

The result of comparing the values provided by the Moodle *QC* indicator and the *QATT* indicator we defined for the quizzes is shown in Figure 6. In this study, the value of the *QATT* indicator included all students' in-course test activities. The horizontal axis shows the *QATT* values determined from the students' quiz activities, and the vertical axis shows the *QC* values.

Although there appears to be some correlation between the values of the two indicators, the individual points are significantly scattered. The value of the *QATT* indicator is low (close to 0) if the student has tested little and high (close to 1) if tested a lot. The value of the *QC* indicator is low (close to -1) if the student submitted or viewed few of the available tests, and high (close to 1) if all tests were submitted, regardless of the result. *QC* values around 0 mean that approximately half of the possible tests were submitted/viewed by students. The figure shows that students with few views (*QATT* values close to 0) can also achieve *QC* values of 0.7-0.8, which is close to the maximum value of 1.

Of course, the values are somewhat related, since the algorithm for calculating *QC* indicators gives a partial score even if the student only viewed the test, so the value of the *QC* indicator increases with the number of views, like the *QATT* indicator. It is important to note that *QC* indicators also include a quality factor as opposed to *QATT* indicators. The figure shows that there are students who have a relatively high *QATT* value (0.5) but a low *QC* value (0.1). This means that the test is viewed many times but not submitted. We have no information in the system about how it was filled out. In

this case, the *QMGR* indicator does not necessarily reflect the student's knowledge of the quiz question.

It is likely that the correlation of the indicators for the quizzes with the target could be further improved by combining the calculation method of the two indicators.

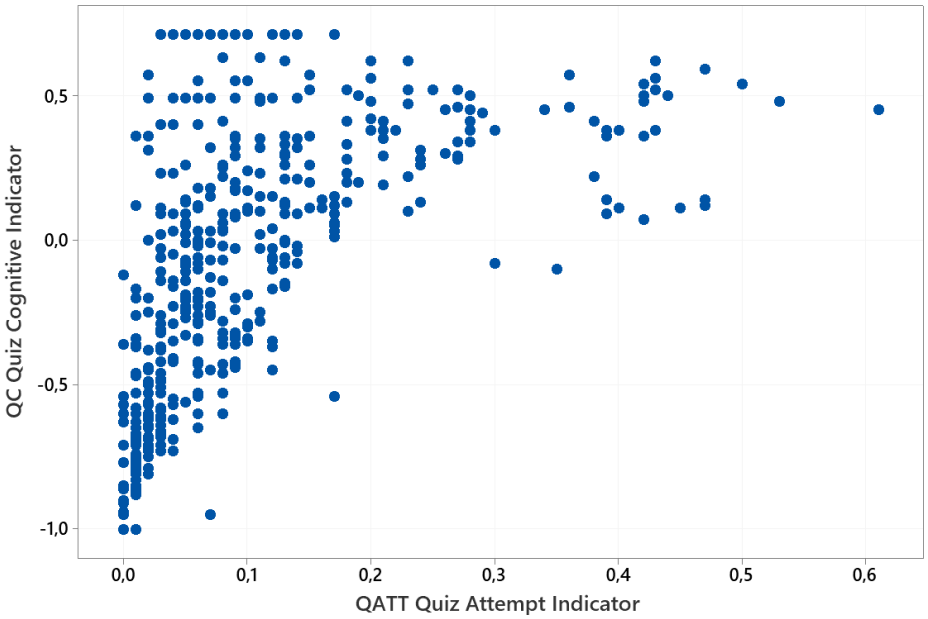

Fig. 6. Correlation between the values of QC and QATT indicators

5.2 Increasing the number of indicators

As mentioned earlier, we hypothesized that for a course with only a few (4-5) components, the number of core indicators that can be built into the model is too small to obtain reliable predictions. Based on the experience of the system analysis, we saw that the number of indicators can be effectively increased by assigning a separate indicator to each topic (section) within the given course. This also retains the flexibility of the system, as indicators defined at the topic level can be used in other courses.

In the case of the examined course, this was achieved with the modification that *ATT* and *QMGR* indicators were defined separately for each section. With this method, the number of *ATT*-type indicators was increased to 36 for the 4 different components and 15 chapters. With the introduction of section-level *QMGR* indicators, we were able to insert 7 additional indicators into the system. Thus, we created a total of 43 self-developed indicators based on the chapters of the course. Overall, the number of indicators is as shown in Table 1.

Table 1. Number of indicators

Indicator	Number of ATT indicator
File ATT	7
Page ATT	15
Book ATT	7
Quiz ATT	7
Quiz MGR	7

6 Results, comparison

To support our theoretical considerations, we built several different models that basically belonged to two groups. In the first group, the models included only Moodle Core Cognitive-type indicators. These are the models in Table 2.

Table 2. Moodle Core Cognitive indicators models

Model 1 *File Cognitive* NoI: 1	**Model 2** *Page Cognitive* NoI: 1	**Model 3** *Book Cognitive* NoI: 1	
Model 5 *File Cognitive + Page Cognitive + Book Cognitive* NoI: 3			**Model 4** *Quiz Cognitive* NoI: 1
Model 6 *File Cognitive + Quiz Cognitive + Page Cognitive + Book Cognitive* NoI: 4			

In the second group, the models contained only self-developed indicators. These models are shown in Table 3.

Table 3. Self-defined indicators models

Model 7 *File* NoI: 7	**Model 8** *Book* NoI: 7	**Model 9** *Page* NoI 15:	
Model 11 *File + Book + Page* NoI: 29			**Model 10** *Quiz Attempts + Quiz Max Grades* NoI: 14
Model 12 *File + Book + Page + Quiz Attempts + Quiz Max Grades* NoI: 43			

The tables include the names of each model, the type of indicators used in the models, and the number of indicators (NoI). Models with an increasing number of indicators embedded models with fewer indicators. The Accuracy, F1 Score, and nMCC values expressing the goodness of each model are shown in Table 4.

Table 4. Summary table with "goodness" values of the models

	Accuracy		F1 score		nMCC	
Goodness	***Moodle***	***Self***	***Moodle***	***Self***	***Moodle***	***Self***
File (M1 - M7)	0.84	0.87	NI	NI	NI	NI
Book (M2 – M8)	0.84	0.86	NI	NI	NI	NI
Page (M3- M9)	0.84	0.87	NI	0.34	NI	0.66
Quiz (M4 – M10)	0.84	0.82	NI	0.24	NI	0.51
All but quiz (M5 – M11)	0.85	0.89	NI	0.54	0.49	0.74
Full (M6 – M12)	0.86	0.9	0.07	0.66	0.56	0.8

The Accuracy values of all Moodle core models show a relatively high value. However, these results should be interpreted appropriately. The Accuracy values suggest a good model, however, for the F1 score and nMCC values, all models except Model 6, gave Not Interpretable (NI) results. Of the 56 students, 9 students failed the course, and these models, without exception, identified the failed students as successful. Even in the case of the full model (Model 6), very low F1 (0.07) and nMCC (0.56) values came out, which shows that these models are unsuitable for predictions.

Among the models made with self-made indicators, the models containing 7 indicators gave similarly incomprehensible results (Model 7, Model 8). Model 9 with 15 indicators, based on Page-type video display resources, and Model 10 with 14 indicators, which included only quiz-type *QATT* and *QMGR* indicators, have already yielded interpretable results. However, these results also indicate poor predictive power (Model 9: F1 = 0.34, nMCC = 0.66, Model 10: F1 = 0.24, nMCC = 0.51). Model 11, which included all indicators except *QATT* and *QMGR* indicators, had a total of 29 indicators with F1 = 0.54 and nMCC = 0.74.

The most spectacular improvement was for the Model 12 with all 43 indicators. For this model, F1 is 0.66 and nMCC is 0.8. This represents a significant improvement in the predictive power of the models. The values for Model 6 containing only Moodle core indicators were F1 = 0.07 and nMCC = 0.56.

Comparative plots of F1 score and nMCC values are shown in Figures 7 and 8.

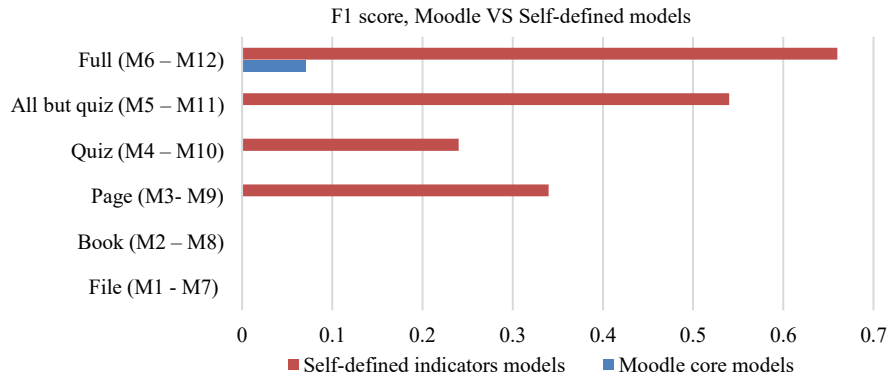

Fig. 7. F1 score values

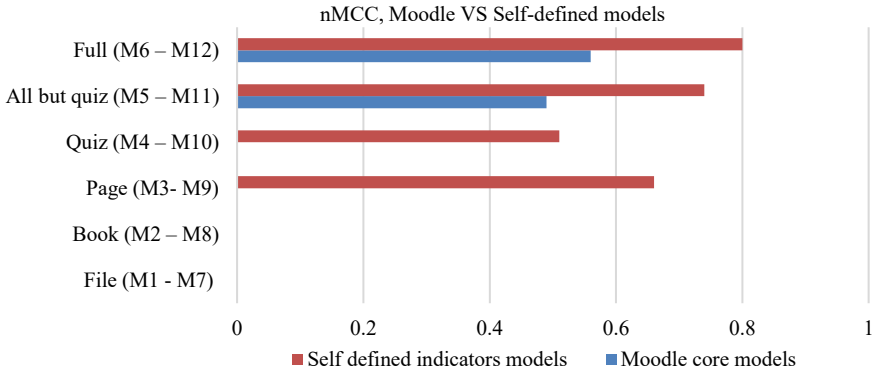

Fig. 8. nMCC values

7 Conclusions

A Learning Analytics tool integrated into an LMS system can be an excellent help, especially for online courses, to analyze learning processes. Self-learning Machine Learning models can be built, which can then be used for different predictions. The Analytics API integrated into the Moodle system is one such tool, a great initiative, but when used as a black box, the system may be unusable for forecasting. The goodness of an ML model is affected by several factors. The number of participants in the course, the structure of the course, the proportion of failed and successful students, the learning habits, the proportion of students and predictors, the correlation of each predictor with student success, etc. If we take these factors into account when building the model, we can significantly improve its predictive power. In the present studies, we highlighted two factors, namely the method of calculating the indicators and the number of indicators. Even by optimizing these two factors alone, the goodness of models built on the same course can be significantly improved. By considering additional aspects, the predictive power of the models can be further improved.

8 References

[1] Abu-Naser, S. S., Zaqout, I. S., Abu Ghosh, M., Atallah, R. R., & Alajrami, E. (2015). Predicting student performance using artificial neural network: In the faculty of engineering and information technology. https://doi.org/10.14257/ijhit.2015.8.2.20

[2] Agrawal, H., & Mavani, H. (2015). Student performance prediction using machine learning. International Journal of Engineering Research and Technology, 4(03), 111-113. https://doi.org/10.17577/IJERTV4IS030127

[3] Bognár, L., Fauszt, T., & Nagy, G. Z. (2021). Analysis of Conditions for Reliable Predictions by Moodle Machine Learning Models. International Journal of Emerging Technologies in Learning, 16(6). https://doi.org/10.3991/ijet.v16i06.18347

[4] Bognár, L., Fauszt, T., & Váraljai, M. (Under press). The impact of online quizzes on student success. International Journal of Engineering Research and Technology.

[5] Buschetto Macarini, L. A., Cechinel, C., Batista Machado, M. F., Faria Culmant Ramos, V., & Munoz, R. (2019). Predicting students success in blended learning—evaluating different interactions inside learning management systems. Applied Sciences, 9(24), 5523. https://doi.org/10.3390/app9245523

[6] Chung, J. Y., & Lee, S. (2019). Dropout early warning systems for high school students using machine learning. Children and Youth Services Review, 96, 346-353. https://doi.org/10.1016/j.childyouth.2018.11.030

[7] D. Monllaó, D.Q. Huynh, M. Reynolds, M. Dougiamas and D. Wiese, "A Supervised Learning framework for Learning Management Systems" In: Proceedings of the First International Conference on Data Science, E-Learning and Information Systems, p. 18. ACM, October 2018.

[8] Dalipi, F., Imran, A. S., & Kastrati, Z. (2018, April). MOOC dropout prediction using machine learning techniques: Review and research challenges. In 2018 IEEE Global Engineering Education Conference (EDUCON) (pp. 1007-1014). IEEE. https://doi.org/10.1109/EDUCON.2018.8363340

[9] Gray, C. C., & Perkins, D. (2019). Utilizing early engagement and machine learning to predict student outcomes. Computers & Education, 131, 22-32. https://doi.org/10.1016/j.compedu.2018.12.006

[10] Hamim, T., Benabbou, F., & Sael, N. (2021). Survey of Machine Learning Techniques for Student Profile Modelling. International Journal of Emerging Technologies in Learning, 16(4). https://doi.org/10.3991/ijet.v16i04.18643

[11] Kloft, M., Stiehler, F., Zheng, Z., & Pinkwart, N. (2014, October). Predicting MOOC dropout over weeks using machine learning methods. In Proceedings of the EMNLP 2014 workshop on analysis of large scale social interaction in MOOCs (pp. 60-65). https://doi.org/10.3115/v1/W14-4111

[12] Kotsiantis, S. B. (2012). Use of machine learning techniques for educational proposes: a decision support system for forecasting students' grades. Artificial Intelligence Review, 37(4), 331-344. https://doi.org/10.1007/s10462-011-9234-x

[13] Mduma, N., Kalegele, K., & Machuve, D. (2019). A survey of machine learning approaches and techniques for student dropout prediction. https://doi.org/10.5334/dsj-2019-014

[14] Prenkaj, B., Velardi, P., Stilo, G., Distante, D., & Faralli, S. (2020). A survey of machine learning approaches for student dropout prediction in online courses. ACM Computing Surveys (CSUR), 53(3), 1-34. https://doi.org/10.1145/3388792

[15] Rastrollo-Guerrero, J. L., Gomez-Pulido, J. A., & Duran-Dominguez, A. (2020). Analyzing and predicting students' performance by means of machine learning: a review. Applied sciences, 10(3), 1042. https://doi.org/10.3390/app10031042

[16] Sokout, H., Usagawa, T., & Mukhtar, S. (2020). Learning Analytics: Analyzing Various Aspects of Learners' Performance in Blended Courses. The Case of Kabul Polytechnic University, Afghanistan. International Journal of Emerging Technologies in Learning (iJET), 15(12), 168-190. https://doi.org/10.3991/ijet.v15i12.13473

[17] Tan, M., & Shao, P. (2015). Prediction of student dropout in e-Learning program through the use of machine learning method. International Journal of Emerging Technologies in Learning, 10(1). https://doi.org/10.3991/ijet.v10i1.4189

9 Authors

Tibor Fauszt is the associate professor of Information Technology at the Budapest Business School University of Applied Sciences. Buzogány u. 10-12. Budapest, Hungary, H-1149. https://uni-bge.hu/en. He is a member of the Informatics Sciences Committee of the Hungarian Rectors' Conference. http://www.mrk.hu/bizottsagok/informatikai-tudomanyok-bizottsaga/

László Bognár is the professor of Applied Statistics at the University of Dunaújváros, Táncsics M. u. 1. Dunaújváros, Hungary H-2400, www.uniduna.hu. He also works for APAVE Hungary and holds trainings in different fields of industrial statistics, www.apave.hu (email: bognar.laszlo@uniduna.hu).

Ágnes Sándor is an assistant lecturer at Budapest Business School Department of Business Information Technology. Buzogány u. 10-12. Budapest, Hungary, H-1149. https://uni-bge.hu/en. She is currently a PhD student at the Doctoral School of Entrepreneurship and Business of the Budapest Business School. Her research interests focus on digitalization, digital maturity (email: sandor.agnes@uni-bge.hu).

Article submitted 2021-05-25. Resubmitted 2021-07-28. Final acceptance 2021-08-21. Final version published as submitted by the authors.