A Supervised Learning framework for Learning Management Systems

David Monllaó Olivé
The University of Western Australia
Crawley, WA, Australia
Moodle Pty Ltd
West Perth, WA, Australia
david.monllaoolive@research.uwa.edu.au

Du Q. Huynh
The University of Western Australia
Crawley, WA, Australia
du.huynh@uwa.edu.au

Mark Reynolds
The University of Western Australia
Crawley, WA, Australia
mark.reynolds@uwa.edu.au

Martin Dougiamas
Moodle Pty Ltd
West Perth, WA, Australia
martin@moodle.com

Damyon Wiese
Moodle Pty Ltd
West Perth, WA, Australia
damyon@moodle.com

ABSTRACT

Educational Data Mining (EDM) and Learning Analytics (LA) focus on data analysis of learners in the context of educational settings like Moodle, a Learning Management System (LMS). Both EDM and LA aim to understand learners and optimise learning processes. Predictive modelling serve a key role in optimising learning processes. Learning Analytics in an LMS covers many different aspects: finding students at risk of abandoning a course, predicting students failing a quiz or students not reaching the end of a lesson in less than 15 minutes. Thus, there are multiple prediction models that can be explored. The prediction models can target at the course also. For instance, will this course engage learners? Will this forum be useful to the students of this course? To ease the evaluation and usage of Supervised Learning prediction models in LMS, we abstract the key elements of prediction models and we build an analytics framework for Moodle, one of the most popular Learning management Systems available in the market. Our software framework manages the complete cycle that predictive models follow until they are used in production, which includes calculations of features and labels from the LMS database raw data, normalization, feature engineering, model evaluation and a production-ready mode to generate insights for users from predictions. Apart from the software framework we also present a use case that serves as an example: A prediction model which is able to identify students at risk of abandoning a course with a 92% in accuracy using past versions of the course as training data.

DATA '18, October 1–2, 2018, Madrid, Spain

ACM ISBN 978-1-4503-6536-9/18/10…$15.00
https://doi.org/10.1145/3279996.3280014

CCS CONCEPTS

• **Computing methodologies → Machine learning algorithms**; *Neural networks*; • **Applied computing → Learning management systems**; *Distance learning*; *E-learning*;

KEYWORDS

Supervised Learning, Learning Management Systems, Moodle, Learning Analytics, Educational Data Mining, Machine Learning, Neural Networks

1 INTRODUCTION

There are multiple *Learning Analytics* definitions. A popular definition from the *Call for Papers of the 1st International Conference on Learning Analytics & Knowledge (LAK 2011)* is that Learning Analytics is *"the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimizing learning and the environments in which it occurs"*. Educational Data Mining is a related field described in The Educational Data Mining website[1] as *"an emerging discipline, concerned with developing methods for exploring the unique and increasingly large-scale data that come from educational settings and using those methods to better understand students, and the settings which they learn in"*. A very popular educational setting is the Learning Management Systems (LMS). Moodle[2] is an open source Learning Management System with more than 130 million users around the world[3]. The back-end system of Moodle can be accessed via a web front-end and through a mobile application[4]. When an online course is managed by Moodle, the learning activities of students are logged and stored into the back-end database, which contains the detailed activity log. Predictive modelling allows us to model the relation between a target (also known as *label* or *dependent variable*) and a set of variables (also referred to as *independent variables* or *features*). Predictive modelling can be implemented using Supervised Learning algorithms which learn a function that maps this features-target

[1] http://educationaldatamining.org/
[2] https://moodle.org
[3] https://moodle.net/stats/
[4] https://download.moodle.org/mobile

relation. A large number of EDM and LA research papers are focused on predicting students' course grades or predicting students that might fail a course.

Although successfully passing a course is important, it is merely the result of a learning process rather than the learning process itself. In order to optimise learning processes, prediction models should also target at the learning processes themselves, in whatever shape they come. Learning processes in an LMS course can be part of an activity or they can be an activity or the course itself. Examples of activities that we can find in an LMS course are: assignments, quizzes or lessons. Examples of prediction model targets could be: learners failing a quiz, learners submitting assignments after the due date, or learners not being able to complete a lesson in less than 15 minutes. Examples of prediction models that target at the course itself rather than the learners are: will this course engage learners? Will this forum be useful to the students of this course?

The contribution of this paper: The presented software framework provides a solid foundation for the development of EDM and LA prediction models. The framework simplifies the implementation of new prediction models in online educational contexts. This is an advantage over developing and evaluating prediction models from scratch. By using this framework, EDM and LA researchers can focus on the educational aspects of their research instead of spending their time on evaluating models performance using basic statistical methods manually or tuning Machine Learning algorithms. Although the framework is developed for Moodle, the same design can be applied to other Learning Management Systems as the framework components are just Supervised Learning prediction model abstractions for an educational context. Our second contribution is a developed model for predicting students at risk that can be readily used within the framework. This model is able to accurately predict which students are at risk of abandoning a MOOC before it finishes. This allows educators to take appropriate intervention before the end of the course.

2 BACKGROUND

2.1 Learning Analytics

The term *Learning Analytics* includes different disciplines related to education and technology. There is no clear agreement on a single and clear definition. Different authors have extended the concept of Learning Analytics to a theoretical framework composed of different dimensions [14] [7]. Educational Data Mining has been compared with Learning Analytics and the main difference that has been identified is that the former contains Academic Analytics whereas the latter does not [24]. A survey on recent Educational Data Mining methods and applications is available in [2]. Other relatively recent surveys are [3] and [22]. Machine Learning techniques have been used in EDM and LA [20] [15] [6] [25] with good results from algorithms such as Neural Networks [4] [6] and Support Vector Machines [5].

Two of the main areas of interest of these two fields are: students' retention [17] [15] [6] and students' performance [25]. Other studied prediction models include assignment submissions [11] or students' learning style classification [1].

2.2 Moodle

By default, the Moodle API only provides basic reporting capabilities. Its reporting tools[5] allow users, mainly teachers, to access course activity logs, to group data by students or by activity, and to generate graphs for aggregated data. Analytics tools for extracting insights from activity logs and for visualization are not available in the default core system of Moodle; however, a wide range of plugins that are compatible to Moodle can be separately and easily installed. A few typical analytics tools for Moodle based on proprietary software are:

- X-Ray Learning Analytics[6], which provides predictive analytics;
- Intelliboard[7], which is a user-friendly reporting tool;
- Analytics Graphs[8], which provides graphs to illustrate students' profiles;
- Analytics local plugin[9], which provides site analytics based on the *Matomo*[10] (formerly known as *Piwik*) analytics platform.

There are two Educational Data Mining tools for Moodle worth mentioning: CVLA [11] which features an assignment submission prediction model and MDM, [16] an Educational Data Mining open source tool that helps to ease the whole knowledge discovery process. Some of these tools are open sourced and some of them have predictive analytics capabilities. However, none of them provides a complete framework that manages the entire cycle from hypothesis testing to actionable insights for the users, typically, the teachers.

3 FRAMEWORK ARCHITECTURE

The Supervised Learning framework presented in this paper is part of the Moodle core since its 3.4.0 version. A Moodle site prediction model is completely separated from other Moodle sites, so each Moodle site only uses the training data and obtains predictions for samples available in that same site. The framework manages the prediction model' life cycle and, for any given prediction model, the framework supports two different modes:

- It automatically extracts training and validation data from the finished courses according to the features and target defined for the prediction model. It then trains the prediction model and validates its prediction accuracy. This is referred to as the *testing mode*.
- It uses all the data from the finished courses as training data and the prediction is done for on-going courses. This is referred to as the *production mode*.

Moodle is written in PHP[11], which is not the best programming language for Machine Learning for multiple reasons. The memory overhead added by each PHP variable and the lack of GPU-acceleration support are two major concerns. Because of these, we separate the framework into two layers:

[5] https://docs.moodle.org/35/en/Course_reports
[6] https://www.moodlerooms.com/resource/x-ray-learning-analytics/
[7] https://moodle.org/plugins/local_intelliboard
[8] https://moodle.org/plugins/block_analytics_graphs
[9] https://moodle.org/plugins/local_analytics
[10] https://matomo.org/
[11] http://php.net

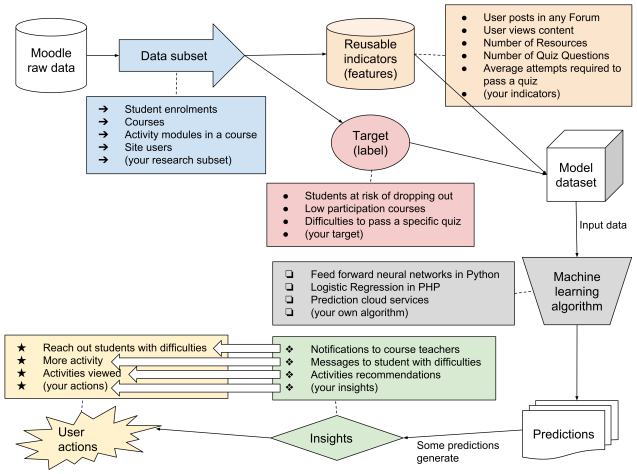

Figure 1: A Data flow diagram showing the steps that the framework goes through in the production mode. It includes examples of each component.

- The Moodle Analytics API[12] written in PHP is responsible for generating labelled and unlabelled CSV[13] files from Moodle's database contents. The CSV format is chosen because it is a simple text format that is portable on all platforms. For all prediction models, multiple CSV files are generated to capture information about the courses and the students.
- Machine Learning backends are responsible for processing these files. They process the labelled and unlabelled CSV files. These backends can be written in any programming language.

This 2-layer design allows us to speed up the training process as well as to allow researchers with Machine Learning experience to write their own Machine Learning backends if they are not satisfied with the default backends' performance.

3.1 Supervised Learning Abstraction

The framework is an abstraction of a Supervised Learning problem. In this section we list the main elements that compose the framework and how they are mapped to the typical elements needed in a Supervised Learning task.

Each prediction model is composed of a single *Analyser*, a single *Target*, a set of Indicators, one Time Splitting Method and a single Machine Learning backend. All these elements are implemented as PHP classes[14] although Machine Learning backends can be written in other programming language. The whole Supervised Learning framework has been designed so that the elements described below are reusable and extensible independently across different prediction models. The framework elements are described below following the order shown in Figure 1.

3.1.1 Analysers. Each Analyzer is responsible for defining the subject of the model. They select and pass all the Moodle data associated to these subjects to *Targets* and *Indicators* (described right below) as shown in Figure 1.

The following analysers are included in the framework and can be reused by researchers to create new prediction models:

- **Student enrolments:** The subject of the model are students in a course.
- **Users:** The subject of the model are site users.
- **Courses:** The subject of the model are courses.

3.1.2 Targets. They are the key element of a prediction model. They represent the **labels** of a Supervised Learning dataset and define the event of interest. Obviously, *Targets* depend on *Analysers*, because *Analysers* provide the subjects that *Targets* need for calculating the label. The framework includes an *identifying students at risk of Target*. Here are a few more examples of *Targets* in prediction models and their associated subjects:

- **Identifying spammer user:** The subjects of the model are site users.
- **Classifying ineffective course:** The subjects of the model are courses.
- **Assessing difficulties to pass a specific quiz:** The subjects of the model are quizzes.

3.1.3 Indicators. They represent the **features** in a Supervised Learning problem. Indicators are responsible for performing calculations on Moodle data. They are calculated for each subject using data available in the time period defined by the *Time splitting method* (described right below). They were designed to avoid normalisation issues so the CSV file features generated from indicators always have values in the range $[-1, 1]$. Indicators are in one of the following categories:

- **Linear:** The indicator values are floating point numbers in the range $[-1, 1]$. An example would be "the weight of quiz activities in a course".
- **Binary:** The indicator values are Boolean values $\in \{0, 1\}$. An example would be "has this student completed all activities in the course?".
- **Discrete:** The indicator values are a closed list of values. The framework one-hot encodes the list of values and generates N features with values $\in \{0, 1\}$. An example would be "How often do this student access the course?", with values "never", "in monthly basis" and "in weekly basis"

The framework includes a set of indicators that can be used in new prediction models. Not all indicators can be included in any model. E.g. student-related indicators could not be calculated if the predictions subjects are courses. Below are some of the indicators included in the framework:

- The number of clicks made by the student in a course. It is implemented as a *linear* indicator.
- Student posts to forum activities in a course. It is implemented as a *linear* indicator.
- Was the course accessed after the course end date? It is implemented as a *binary* indicator.
- Was the course accessed before the course start date? It is implemented as a *binary* indicator.

[12] https://docs.moodle.org/dev/Analytics_API
[13] http://www.ietf.org/rfc/rfc4180.txt#page-1
[14] http://php.net/manual/en/language.oop5.php

3.1.4 Time splitting methods. They define when the framework should generate predictions and the time period that should be used to calculate the indicators. The Moodle core includes a few time splitting methods that researchers can use in their prediction models. For example:

- Split the course duration into four parts and generate a prediction at the end of each part.
- Generate a prediction one week before each assignment's due date and generate a second prediction two days before the assignment due date.

3.1.5 Machine Learning backends. In the testing mode, the Machine Learning backends split the CSV files into training and testing sets and evaluate the prediction accuracy. This process is described in more detail in Sections 3.2 and 3.5 below. In the production mode, Machine Learning backends are trained with finished courses data and return predictions for ongoing courses.

The Moodle core includes two Machine Learning backends: A Feed-Forward Neural Network [9] written in Python[15] using the Tensorflow framework[16] and a Logistic Regression [8] classifier written in PHP for the Moodle sites where installing Python is not an option. New Machine Learning backends can be plugged on the framework.

3.2 Prediction model definition

Researchers can define prediction models by implementing a *Target* in PHP. Depending on the subject (see Section 3.1.2.) researchers can reuse an existing *Analyser* or they can create a new one. The next step that the researcher should perform is to select the Indicators that would have impact on the *Target* and select the Time splitting method from a list of choices provided by the framework.

3.3 CSV file preparation

In this section we describe the process the framework follows to generate a labelled CSV file from the Moodle site's database. The *Analysers* of the framework iterate through the site database, gathers the model subjects' data from the different analysable elements available in the system. The features in the CSV file are added according to the prediction model *Indicators* and *Time splitting method*. The interaction between the *Analyser* and the *Indicators* has been designed so that the framework is able to automatically add extra features. This allows us to perform feature engineering tasks while still having access to the LMS data. For each *Linear* indicator the framework automatically adds an extra feature whose value is the mean of all samples in that analysable element. The number of features in the resulting CSV file is determined by the following formula:

$$\mathrm{NF} = (\mathrm{NL} * 2) + \mathrm{NB} + \sum_{i=1}^{ND} \mathrm{NV}_i + \mathrm{TP}, \quad (1)$$

where NF is the number of features, NL the number of linear indicators, NB the number of binary indicators, superscript ND is the number of discrete indicators, NV the number of values in the i^{th} discrete indicator and TP the number of predictions generated by the time splitting method.

[15] https://www.python.org/
[16] https://www.tensorflow.org/

The values for *label* are also calculated according to the definition of the *Target*. The framework supports binary classification, multi-class classification and regression, but the Machine Learning backends included in Moodle do not yet support multi-class classification or regression, so only binary classifications are fully supported at this stage.

3.4 Machine Learning algorithms

The CSV files described in Section 3.3 above are consumed by Machine Learning algorithms. As mentioned in Section 3.1.5, the Machine Learning backends of the framework include two classifiers. These classifiers are described in detail in the following subsections.

3.4.1 Logistic Regression. The PHP Machine Learning backend uses a Logistic Regression binary classifier [8] to perform its predictions. In Logistic regression, a Logistic function $h_\theta(x) = 1/(1 + \exp(-\theta^T x))$ is applied to the feature vector x to produce a value in the range (0, 1). The parameter $\theta \in \mathbb{R}^n$ is a vector of weights that need to be learned. The function behaves like a thresholding function with a soft boundary. Because of the range the output value $h_\theta(x)$ can be interpreted as the probability whether x belongs to the target class in a two-class classification problem. Logistic Regression tries to find the best fitting model for the relation between features and their labels by optimising the following cross-entropy cost function:

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y_i \log(h_\theta(\boldsymbol{x_i})) + (1 - y_i) \log(1 - h_\theta(\boldsymbol{x_i}))], \quad (2)$$

where m is the total number of samples, $\mathrm{x_i}$ denotes the i^{th} sample and $\mathrm{y_i}$ denotes the corresponding ground truth label. The cost function gives the error between a set of weights where all samples' fit perfectly and the labels predicted by the algorithm. The parameter $\boldsymbol{\theta}$ is updated according to the gradients of the cost function. Gradient Descent [10] is the common algorithm used to optimise cost functions in Machine Learning. It is iteratively used to update the set of weights $\boldsymbol{\theta}$ using the following formula:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta), \quad \text{for } j = 1, \ldots, n, \quad (3)$$

where α is a constant called *learning rate* which defines how significant $\boldsymbol{\theta}$ updates are and $\frac{\partial}{\partial \theta_j} J(\theta)$ is the partial derivative of $J(\theta)$. Logistic Regression is an effective algorithm for binary classification although some problems would arise when the number of features increases: The algorithm becomes computationally very expensive and it is very easy for the algorithm to over fit the training data. Logistic Regression popularity is decreasing in favour of other Supervised Learning algorithms.

3.4.2 Feed-Forward Neural Network. Artificial Neural Networks (ANN) [23] mimic biological Neural Networks. They are networks containing relations between units, also called *neurons*. Neural Networks organise units into layers. The connections between neurons in different layers are represented as a set of weights $\boldsymbol{\theta}$. The Neural Network improve its accuracy during training by back propagating [9] the difference of the calculated label of a sample

versus the real value to the network's previous layers, updating its weights. The connections between different layer neurons become stronger or weaker after the back-propagation process.

Feed-forward Neural Networks contain a first layer of units called input layer, with as many units as features in the input CSV file. They then contain a number of hidden layers $l \geq 1$ and a final output layer with as many units as different labels the input CSV file has. The Neural Network included in the framework contains one hidden layer with ten hidden units. The network training process is composed of two different steps: feed-forward and back-propagation.

The feed-forward process computes the predicted labels for a given set of samples by multiplying input features values by the matrices that connect different layer units. The following formulas use a set of samples of size N. In $z_1 = \theta_1 \cdot x + b$ we multiply the input features matrix for the weights matrix θ_1 that connects the input features x with the first hidden layer units, adding a bias b. z_1 is multiplied by a non-linear activation function $g(z)$. Different activation functions can be used, some examples are $g(z) = \text{sigmoid}(z)$, $g(z) = \tanh(z)$ or $g(z) = \text{relu}(z)$. The activated matrix (one vector for each sample) in the following layer is therefore expressed as $a_1 = g(z_1)$. This calculation is repeated until the output layer is reached. It can be generalised as $a_l = g(\theta_l \cdot a_{l-1})$. The softmax function can be used as the output layer activation function as it returns a $\in [0, 1]$ value for each possible label. The softmax formula is expressed as follows:

$$\text{softmax}_\text{i} = \frac{e^{a_{li}}}{\sum_j e^{a_{lj}}}, \tag{4}$$

where i and j are each of the possible labels.

We finish the forward pass by calculating the error using the cross-entropy cost function $J(\theta)$ given in Eq. 2.

During back-propagation we minimise the cost function $J(\theta)$ by updating the weights that connect neurons in different layers. The updated value depends on the partial derivative of the error with respect to each of the network weights in the previous layer $\partial J(\theta)/\partial \theta ij$ and the learning rate α. Weights are updated using the Delta rule, expressed as follows:

$$\Delta \theta_l = -\alpha \frac{\partial J(\theta)}{\partial \theta_l}. \tag{5}$$

The partial derivatives calculations are based on the chain rule, which allows us to compute a derivative as the composition of two or more functions, in our case:

$$\frac{\partial J(\theta)}{\partial \theta_l} = \frac{\partial J(\theta)}{\partial g(z_l)} \cdot \frac{\partial g(z_l)}{\partial z_l} \cdot \frac{\partial z_l}{\partial \theta_l}, \tag{6}$$

where θ_l represents the vector of weights in layer l and $g(z)$ the activation function. $\partial z_j/\partial \theta_l = a_l$, the partial derivative of the activation output is the derivative of the activation function. E.g. Sigmoid function $g'(z_l) = g(z_l)(1 - g(z_l))$:

Finally, $\partial J(\theta)/\partial g(z_l)$ calculation depends on the value of l as we need to consider all the layers from l to the output layer. We calculate deltas δ_l for each layer starting from the output layer one, which is $\delta_{output} = (\hat{y} - y)$, where y and $\hat{y}$ are, respectively, the actual and predicted labels (both are vectors usually). With δ_{output} we can go backwards calculating the previous deltas until the first hidden layer with $\delta_l = {\theta_l}^T \delta_{l+1} \cdot g'(z_l)$. The partial derivative of any weight in the network is therefore represented as:

$$\frac{\partial J(\theta)}{\partial \theta_l} = \delta_{l+1} \cdot g(z_l). \tag{7}$$

3.5 Testing mode

In Section 3.3 we described that the framework generates a labelled CSV file from Moodle site contents based on the prediction model defined by the researcher. In this section we describe how the framework evaluates the defined prediction model using Machine Learning techniques.

The first thing the evaluation process detects are the CSV files with highly skewed classes. The provided Machine Learning backends do not cope well with really unbalanced classes. Even if the Machine Learning backend reports high accuracies the recall or the precision will probably not be high which would lead to a low predictive power. Prediction models with highly skewed classes are not further evaluated in the current framework.

The Machine Learning algorithm is trained with the training dataset and the test dataset is used to calculate the *Matthews' correlation coefficient* [18] of the prediction model. The *Matthews' correlation coefficient* is a good evaluation metric for binary classification problems because it takes into account both true positives and true negatives [21].

To calculate the *Matthews' correlation coefficient* we first fill out the confusion matrix with the predicted results and the test dataset labels. The confusion matrix looks like this:

$$\begin{bmatrix} \text{TN} & \text{FN} \\ \text{FP} & \text{TP} \end{bmatrix},$$

where TP = **True positives**, FP = **False positives**, TN = **True negatives** and FN = **False negatives**. The *Matthews' correlation coefficient* (MCC) formula is given by:

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP}) \times (\text{TP} + \text{FN}) \times (\text{TN} + \text{FP}) \times (\text{TN} + \text{FN})}}. \tag{8}$$

The entire process is automated. So this training and *MCC* calculation process described above is repeated multiple times and the *MCC* for each iteration is be recorded. The average *MCC* and standard deviation is later calculated from all iterations. An average *MCC* of 0 indicates that the model is no better than a random model, a negative value indicates an inverse relation and a positive value a positive relation. Therefore, the higher the *MCC* value is, the better the model is at predicting. The standard deviation of all the calculated *MCC* will be used to detect variations in the coefficients. Small variances in each iteration's *MCC* can be expected because the CSV file contents are shuffled before each iteration, but high variances are a good sign that the CSV file is not large enough to guarantee that the evaluation results are reliable.

The average *MCC* is automatically converted to a score in the range $[0, 100]$ using the following formula:

$$\text{score} = \frac{MCC + 1}{2}. \tag{9}$$

The computed score is provided to the researcher as a quality measure for the prediction model.

3.6 Production mode

The testing mode is useful for checking the hypotheses of the EDM / LA researchers and to verify existing educational literature theories. Once the researcher (or the site administrator if the researcher do not have enough permissions on the site) switches a model to production mode, these hypotheses and theories into optimised learning processes by generating actionable insights from predictions. This production model switch affects the whole site: Insights are generated for all stakeholders in the site, usually the teachers, in addition to the researchers who switch the framework to work in production mode. E.g. Arts teachers do nothing to receive predictions about their at-risk students. They would receive notifications about at-risk students once the prediction model is globally enabled for production mode.

As part of the *Target* definition, researchers can specify which predictions returned by the Machine Learning backend are worth observing and which predictions can be ignored. E.g. A model that predicts students at risk is only interested in at-risk students, not in students that progress as expected. Actionable insights can be specified as part of the *Target* definition as well. These actionable insights are provided as suggested actions to the site user that have access to predictions, usually a teacher in a course. Examples of actionable insights can be to message the at-risk student, to check the student activity log or to ignore the prediction. Figure 3 is an example of how insights are presented to teachers.

The state of the Machine Learning algorithms is saved once the training process finishes. The trained algorithm is restored every time Machine Learning backends need to generate predictions for ongoing courses. The CSV file generated by Moodle for ongoing courses is obviously unlabelled.

4 USE CASE

In this section, we report one of the implemented prediction models of the Supervised Learning framework. This prediction model classifies students without activity logs during the last quarter of a course as drop-outs and all other students as not-drop-outs. We implemented a *Time splitting method* to generate 3 predictions along the course duration. The first one is executed once the first quarter of the course is over, using data from the start of the course. The second one is executed after half course is completed and use activity data from the beginning of the course up to that point. The third prediction is executed after the third quarter of the course, also using all the data available from the start of the course up to that point in time.

A significant amount of Learning Analytics literature is focused on describing online students' engagement from different educational paradigms [19]. Some of these conceptual frameworks, like Community of Inquiry [12], are very popular among educators. Community of Inquiry started as an exploratory and descriptive framework, recognising later its limitations from an empirical point of view [13] The *Indicators* we used in the implemented at-risk students model are based on this paradigm. The adaptation of the Community of Inquiry paradigm to Moodle deserves a separate paper. As a brief summary, we implemented linear indicators based on how intensive the student's interactions with course activities are. Students with no interactions in activities result in negative values and students that interacted with teachers and other peers result in positive values.

Figure 2: Test results.

To test the prediction model we used 8 finished anonymised MOOCs with a total of 46,895 students. Results provided by the framework testing mode are shown in Table 1 and Figure 2.

Our test results show that the proposed at-risk students model gave an average prediction accuracy of 92.56% using the Neural Network described in Section 3.4.2 and an average prediction accuracy of 73.30% using the Logistic Regression classificator described in Section 3.4.1. The Neural Network appear to be better at modelling the relation between the input CSV file features and the label, probably due to the extra set of weights in the hidden layer which should allow the Neural Network to model more complex relations between features.

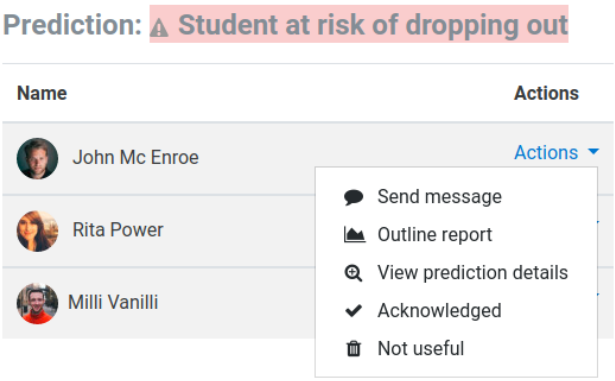

Figure 3: Actionable insights generated by an at-risk students model.

Once the production mode is enabled, predictions can be generated for ongoing courses. The actionable insights generated by this at-risk students model are shown in Figure 3.

5 CONCLUSIONS

The framework design covers prediction models like at-risk students and allows researchers to evaluate them to determine if they

Table 1: Prediction model evaluation results

	Score	MCC	Average accuracy	Average precision	Average recall
Neural Network	0.868	0.736	92.56%	94.66%	96.42%
Logistic Regression	0.646	0.292	73.30%	87.36 %	78.44%

are effective to be used in production. To create a prediction model using the presented framework is an advantage over creating a prediction model from scratch as the framework provides a set of base elements and a number of evaluation metrics to test prediction models accuracy. A prediction model to detect late assignment submissions [17] has also been used to check that the software framework design is adaptable to different prediction subjects and that the framework indicators can be reused across prediction models. The accuracy and recall of the presented prediction model for predicting at-risk students are good for a production system. The Supervised Learning framework presented in this paper is part of the Moodle core from version 3.4.0 onward, but is disabled by default as it requires sufficient computer power to run. Moodle is an open source Learning Management System so the exact number of unregistered users is unknown. Given that there are more than 130 million registered users around the world and given that approximately 50% of more than 100,000 registered Moodle sites use Moodle 3.4.0 or above [18] (June 2018), it would not be too unrealistic to claim that the framework is used by millions of users.

6 FUTURE WORK

There are multiple ideas worth exploring:

- The framework could be extended to cover Unsupervised Learning using the same 2-layer architecture. The framework has been designed keeping Unsupervised Learning in mind so the same set of indicators for Supervised Learning could be used for Unsupervised Learning.
- Learning Analytics literature can be reviewed and extra student engagement indicators can be extracted from the literature. These extra indicators can be adapted to the data available in the LMS, be implemented in PHP and added to the at-risk students model. As it is, the testing mode of the framework can be used to check which of these indicators result in accuracy improvements, supporting the results obtained in other research papers. It is important to mention that the Neural Network and the Logistic Regression classifier described in Section 3.4 and included in the Moodle core do not try to compete with the most accurate algorithms available nowadays and their performance can be easily surpassed by tuned Neural Networks and other Machine Learning techniques.
- Exploring how cross-validation can improve the adaptability of the Machine Learning algorithms included in the Moodle core is another option for the future. Cross-validation processes are useful to adapt Machine Learning algorithms to datasets so the trained algorithms are more accurate. To tune Machine Learning algorithms using cross-validation before training would be a very good option if our training datasets remain static. In our case though, new training data is available in regular basis (e.g. every time a course is finished) so cross-validation can be a double-edged sword.

[17] https://github.com/dmonllao/moodle-local_latesubmissions
[18] https://moodle.net/stats/

ACKNOWLEDGMENTS

We thank Moodle HQ for providing the dataset used in the use case. Also thanks to all Moodle HQ staff that participated in code reviews, testing and user interface and user experience design. Special thanks to Elizabeth Dalton, from Moodle HQ, for her involvement in the design of the indicators used in the framework use case.

REFERENCES

[1] Manal Abdullah, Asmaa Alqahtani, Jawhara Aljabri, Reem Altowirgi, and Ruqiah Fallatah. 2015. Learning Style Classification Based on Student's Behavior in Moodle Learning Management System. *Transactions on Machine Learning and Artificial Intelligence* 3, 1 (2015). https://doi.org/10.14738/tmlai.31.868

[2] Behdad Bakhshinategh, Osmar R. Zaiane, Samira ElAtia, and Donald Ipperciel. 2018. Educational data mining applications and tasks: A survey of the last 10 years. *Education and Information Technologies* 23, 1 (2018), 537–553. https://doi.org/10.1007/s10639-017-9616-z

[3] Alejandro Bogarín, Rebeca Cerezo, and Cristóbal Romero. 2018. A survey on educational process mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8, 1 (2018). https://doi.org/10.1002/widm.1230

[4] Sonja Išljamovi Ć, Milan Vuki Ć E V I Ć, and Milija Suknovi Ć. 2013. Metalurgia International Early Prediction of University Students ' Success Via Neural Networks. XVIII, 5 (2013), 120–126.

[5] J Camba, R E David, A Betan, A Lagman, and J D L Caro. 2016. Student analytics using support vector machines. In *2016 7th International Conference on Information, Intelligence, Systems Applications (IISA)*. 1–6. https://doi.org/10.1109/IISA.2016.7785425

[6] D S Chaplot, E Rhim, and J Kim. 2015. Predicting student attrition in MOOCs using sentiment analysis and neural networks. *CEUR Workshop Proceedings* 1432, June (2015), 7–12.

[7] Mohamed Amine Chatti, Anna Lea Dyckhoff, Ulrik Schroeder, and Hendrik Thüs. 2012. A reference model for learning analytics. *International Journal of Technology Enhanced Learning* 4, 5/6 (2012), 318. https://doi.org/10.1504/IJTEL.2012.051815

[8] D R Cox. 1958. the Regression Analysis of Binary Sequences. *Journal of the Royal Statistical Society. Series B (Methodological) Journal of the Royal Statistical Society. Series B Journal of the Royal Statistical Society SERIES B (METHODOLOGICAL)* 20, 2 (1958), 215–242. https://doi.org/10.2307/2983890

[9] Ronald J. David E. Ruineihart, Geoffrey E. Hinton Williams. 1986. Learning Internal Representations by Error Propagation. *Parallel distributed processing: Explorations in the microstructure of cognition* 1 (1986).

[10] Pierre-Gilles De Gennes. 1972. Méthode générale pour la résolution des systèmes d'équations simultanées. *Comptes rendus hebdomadaires des séances de l ' Académie des sciences, Série B* 275 (1972), 319–321.

[11] Giedre Dregvaite and Robertas Damasevicius. 2015. CVLA: Integrating Multiple Analytics Techniques in a Custom Moodle ReportInformation and Software Technologies: 21st International Conference, ICIST 2015 Druskininkai, Lithuania, October 15-16, 2015 Proceedings. In *Information and Software Technologies*, Vol. 538. 115–126. https://doi.org/10.1007/978-3-319-24770-0

[12] D.Randy Garrison, Terry Anderson, and Walter Archer. 1999. Critical Inquiry in a Text-Based Environment: Computer Conferencing in Higher Education. *The Internet and Higher Education* 2, 2-3 (1999), 87–105. https://doi.org/10.1016/S1096-7516(00)00016-6 arXiv:arXiv:1011.1669v3

[13] D Randy Garrison and J B Arbaugh. 2007. Researching the community of inquiry framework: Review, issues, and future directions. *The Internet and Higher Education* 10, 3 (2007), 157–172. https://doi.org/10.1016/j.iheduc.2007.04.001

[14] Wolfgang Greller and Hendrik Drachsler. 2012. Translating Learning into Numbers : A Generic Framework for Learning Analytics Author contact details :. *Educational Technology & Society* 15, 3 (2012), 42 – 57. https://doi.org/1820/4506

[15] Kate S. Hone and Ghada R. El Said. 2016. Exploring the factors affecting MOOC retention: A survey study. *Computers and Education* 98 (2016), 157–168. https://doi.org/10.1016/j.compedu.2016.03.016

[16] J. M. Luna, C. Castro, and C. Romero. 2017. MDM tool: A data mining framework integrated into Moodle. *Computer Applications in Engineering Education* 25, 1 (2017), 90–102. https://doi.org/10.1002/cae.21782

[17] Carlos Márquez-Vera, Alberto Cano, Cristobal Romero, Amin Yousef Mohammad Noaman, Habib Mousa Fardoun, and Sebastian Ventura. 2016. Early dropout prediction using data mining: A case study with high school students. *Expert Systems* 33, 1 (2016), 107–124. https://doi.org/10.1111/exsy.12135

[18] B. W. Matthews. 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *BBA - Protein Structure* 405, 2 (1975), 442–451. https://doi.org/10.1016/0005-2795(75)90109-9

[19] Katrina A. Meyer. 2014. Student Engagement in Online Learning: What Works and Why. *ASHE Higher Education Report* 40, 6 (2014), 1–114. https://doi.org/10.1002/aehe.20018

[20] Stuart Palmer. 2013. Modelling engineering student academic performance using academic analytics. *International Journal of Engineering Education* 29, 1 (2013), 132–138.

[21] David M W Powers. [n. d.]. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. ([n. d.]).

[22] C. Romero and S. Ventura. 2007. Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications* 33, 1 (2007), 135–146. https://doi.org/10.1016/j.eswa.2006.04.005 arXiv:1201.3417

[23] F Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* 65, 6 (1958), 386–408. https://doi.org/10.1037/h0042519 arXiv:arXiv:1112.6209

[24] George Siemens, Dragan Gasevic, Caroline Haythornthwaite, Shane Dawson, Simon Buckingham Shum, and Rebecca Ferguson. 2011. Open Learning Analytics : an integrated & modularized platform. *Knowledge Creation Diffusion Utilization* (2011), 1–20.

[25] Wanli Xing, Rui Guo, Eva Petakovic, and Sean Goggins. 2015. Participation-based student final performance prediction model through interpretable Genetic Programming: Integrating learning analytics, educational data mining and theory. *Computers in Human Behavior* 47 (2015), 168–181. https://doi.org/10.1016/j.chb.2014.09.034