Are we meeting a deadline? Classification Goal Achievement in time in the presence of imbalanced data

Martin Hlosta[a,b,*], Zdenek Zdrahal[a,c], Jaroslav Zendulka[b]

[a] *Knowledge Media Institute, The Open University, Milton Keynes, UK*
[b] *Faculty of Information Technology, IT4Innovations Centre of Excellence, Brno University of Technology, Brno, Czech Republic*
[c] *CIIRC, Czech Technical University, Prague, Czech Republic*

Abstract

This paper presents the problem where objects out of a finite set are required to achieve a goal within a predefined deadline. For example, a group of students is supposed to submit a homework by a specified cutoff. Further, we are interested in predicting, which objects will achieve the goal within the deadline. The predictive models are built only based on the data from that population. The predictions are computed in various time instants by taking into account updated data about the objects. The first contribution of the paper is a formal description of the problem. The important characteristic of the proposed method for model building is the use of the properties of objects that have already achieved the goal. We call such approach Self-Learning. Since only a few of objects have achieved the goal at the beginning and their number gradually grows, the problem is inherently imbalanced. To mitigate the curse of imbalance, we improved the Self-Learning method by tackling the information loss and by several sampling techniques. The original Self-Learning and the modifications have been evaluated in the case study for predicting submission of the first assessment in the distance higher education courses. The results show that the proposed improvements outperforms the specified two base-line models, the original Self-Learner and also that the best results are achieved if domain-driven techniques are utilised to tackle the imbalance problem.

Keywords: Classification, Imbalanced Data, Learning Analytics, Educational Data Mining

1. Introduction

Student retention has been recognised as a common problem both in the distance Higher Education institutions and in Massive Open Online Courses (MOOCs) [1, 2]. Learning Analytics (LA) and Educational Data Mining (EDM) are research fields that are trying to tackle this issue by examining the available student data. They may include both static, e.g. mainly demographic data, and fluid data, e.g. data generated by students when interacting with a Virtual Learning Environment (VLE). These data are available for developing methods for identification of students who are at risk of failing the courses. If such students are identified early enough, a cost-effective support can be provided. Machine learning techniques are usually used to build models for predicting at-risk students. The predictions can be either made available directly to students [3] or mediated by tutors [4, 5] who may offer additional knowledge not captured by the data and take into account wider context, such as student's personal circumstances.

The standard way to train the predictive models is to take advantage of the information from the previous runs of the course. These models are applied to data of the current run. This approach is based on the assumption that the same or similar pattern of student behaviour prevails across the subsequent years. The existing approaches differ in (1) specification of who are at-risk students; (2) available data for predictions; and (3) the used machine learning algorithms. For example, the "at-risk student" could be defined as the one expected to achieve the final grade lower than C in [4]; or less than 60% in [6]; not submitting the following assessment in [7], or student likely not to submit any other following assessment [8]. In [7], it was shown that not submitting the first assessment is a strong predictor of the future failure.

For new courses, the data from the previous courses are not available and therefore, cannot be used to build predictive models. For such cases, we proposed the Self-Learning approach [9].

In this paper, we further develop the Self-Learning philosophy and demonstrate how to predict students likely to fail by not submitting the first assessment. In addition, we propose further generalisations and improvements.

1.1. Self-Learning in the educational domain

To overcome the lack of legacy data, Self-Learning utilises the behaviour of the students who submit the assessments in advance. We assume that the relevant patterns can be discovered in the VLE and demographics data. It is expected that the learners who are about to submit will follow a similar pattern as

*Corresponding author

Email addresses: martin.hlosta@open.ac.uk (Martin Hlosta), zdenek.zdrahal@open.ac.uk (Zdenek Zdrahal), zendulka@fit.vutbr.cz (Jaroslav Zendulka)

those who have already submitted and such a pattern is missing in the VLE data of students who will not submit.

1.1.1. Classification from imbalanced data

At the beginning, only a few students submit the assessment and the problem is inherently imbalanced. Classification from imbalanced data is a well-recognised problem in the machine learning field [10]. In many real-world supervised learning scenarios, a class exists that has significantly lower number of instances in the data than the other classes. It is not only the large disproportion between the number of instances representing different classes causing the problem. Intuitively, if the concept that separates the data is not complex and if, for example, one attribute discriminates between the two classes perfectly, the classifier would still be able to provide predictions with high accuracy. However, as the complexity of class characteristics grows, the higher imbalance ratio causes greater errors [11]. In the last decade, the impact of imbalance data in ML attracted significant attention of research community and hundreds of papers have been published that discuss, what are the sources of the imbalanced data or how to improve the performance under imbalanced data. As usual in machine learning, there is no guaranteed approach to all the problems and datasets and many of these solutions are domain-dependent. The most recent survey that covers many of the issues and also provides a taxonomy of the solutions comes from Branco et al. [12]. In the case of Self-Learner, the dataset evolves in time, more students submit and the imbalance ratio decreases.

Our experiments in [9] were focused on daily prediction analysis, and they compared various machine learning methods and their ability to deal with imbalanced data. The selected metric was the Area Under the Precision-Recall Curve (PR-AUC) because it is a convenient criterion when dealing with imbalanced data [13].

It was demonstrated that the performance rapidly decreases with the increasing time remaining to the deadline. The best performance was achieved by ensemble based classifiers, XGBoost [14] based on boosting followed by Random Forest based on bagging. Some algorithms, e.g. SVM or Logistic Regression, offer the provision to compensate the lower number of instances in the minority classes already in the training process. Such algorithms performed better than their original, uncompensated versions.

1.2. Generalisation of the concept

The proposed method was primarily targeted to identify students at risk of not submitting the first assessment. As suggested in [9, sec. Discussion], the same approach could predict the results of other milestones in the course, i.e. submission of further assessments. Given appropriate data, the application domain does not need to be limited to education. However, two conditions need to be satisfied: (1) the existence of the deadline within which the goal must be satisfied and also (2) the existence of students/objects that achieve this goal before the deadline. Motivated by this, we posed the first research question:

- **RQ1:** How can we formalise the problem of classification whether individuals in the population will satisfy the goal within the specified deadline?

1.3. Time in imbalanced data classification

Temporal changes of the class imbalance ratio have generated considerable research interest. The survey from 2016 by Krawczyk et al. [15] discussed open challenges in machine learning from imbalanced data and mentioned learning from imbalanced data streams among them. The usual problem of data streams is their dynamic nature: the distribution of the data can change. For example, the imbalance ratio between classes can change and also a different class can dominate as time progresses. In particular, the topic that was related to imbalanced data and still needed to be researched further is the problem of *new class emergence*[15]: the number of instances of the minority class is highly under-sampled in the beginning and then it grows over time.

Wang et al. [16] investigated the changes of imbalance ratio depending on the different speed of change. The experiments compared over-sampling and under-sampling bagging methods, with the over-sampling bagging being better. The performance, however, dropped immediately after the imbalance has changed. The results improved when combining both methods with adaptive weights. Together with the synthetic data, the results were examined on two real-world scenarios of fault detection. A similar task has been studied by Tan et al. in [17], where they focused on predicting "defect" changes in the source code from the versioning system of the open source projects. The goal was to detect changes of the source code that were later fixed and marked as bugs. Changes of code arrive permanently. The results showed the improved performance when using sampling methods against baseline and against *updatable classification methods*. Although four types of sampling have been used, the results presented in [17] did not provide sufficient details, e.g. which sampling performed best.

The specificity of the problem with students assessment submissions, or generally goal achievement as introduced above, lies in the presence of the deadline. Although the tasks presented in [16] and [17] generate imbalanced data by their nature, the absence of the deadline makes their problem different. Compared to their scenario, in our case, we receive new observations about the stable set of objects. Also, rather than an abrupt change, we expect a gradual increase of the submissions at the beginning followed by steep increase closer to the deadline. Consequently, most of the submissions usually occur close to the deadline. This has been also confirmed by our previous results in [9] and by other studies [18, 19, 20]. This phenomenon can be attributed to the well known psychological problem of procrastination, i.e. postponing or avoiding of starting, engaging in, or completing a task [21]. Since the models are constructed from the data of the same course that is being predicted, in the beginning, the methods suffer from the imbalanced data, i.e. the lack of information.

A concept similar to the Self-Learning framework is *Self-Training*, which has been used in some semi-supervised learning problems[22]. This technique utilises both labelled and unlabelled datasets to improve the performance of the classification. First, the model is trained solely on the labelled examples, and the unlabelled ones are then iteratively added until the performance of the classifier stops improving. Such a method has also been used in the imbalanced data processing. In [22], Stanescu and Caragea used the original Self-Training method with several modifications tailored to imbalanced data, achieving the best results when the training set was extended only with the examples predicted as a minority class. The difference between Self-Learning approach and the Self-Training in [22], and semi-supervised methods in general, is the absence of annotated objects of the negative class, *NotAchieve* in our case. In contrast, Self-Learning uses the temporal character of the data to construct the negative class examples from the pool of available objects, e.g. students in our case.

Our previous results [9] compared existing machine learning methods and methods for dealing with imbalanced data (sampling and algorithm based methods). In the beginning, the lack of information worsens the performance. The improvement due to the use of methods developed to tackle imbalanced data was negligible. This opens the potential for improvement, for instance using the domain knowledge. The dynamic nature of the *imbalanced problem* motivates the following research question.

- **RQ2:** How to modify the existing Self-Learning approach to improve the classification performance when applied to problems with time-dependent imbalanced ratio?

Based on the stated research questions, the paper is further structured as follows. First, the problem of goal achievement is formalised in Sec 2. Then, the Self-Learning method is briefly described in Sec 3 and followed by Sec 4, which analyses the issues of the method related to the imbalanced data and proposes new extensions. The experimental setup, the achieved results and discussion are provided in Sec 5. Further implications are summarised in the Conclusions 6.

2. Problem Description

Let us suppose we have a set of objects that are required to achieve a goal within the deadline. Some of these objects may have already done so. For all objects, we have information, which includes their behaviour, and for those that have already achieved the goal, when it happened. Based on such information, we would like to predict if they will have submitted before the deadline is due. The task is to construct a predictive model anytime after the first object has achieved the goal. Notice that we expect that no other legacy data that would guide the training of the predictions are available.

2.1. Goal Achievement Prediction Problem

Let D be a set of objects x_i, $D = \{x_1, x_2, \ldots, x_N\}$, where x_i is an object represented by an m-dimensional feature vector $x_i = (x_i^1, x_i^2, \ldots, x_i^m)$, i.e. an object described by values of m features (or attributes) $A^1, A^2, \ldots, A^m$. These attributes can be either of a numerical or categorical type and they are time dependent. Let g be a goal to be achieved and time be discrete starting at point t_0. The goal g can by achieved by the objects in D in time $t \in [t_0, t_d]$, where t_d is called the deadline. Let's denote achieving the goal g by the object x_i in time t by a predicate

$$Achieved(x_i, g, t). \tag{1}$$

For example, a customer Mark who made a purchase on 24th December 2010 would be denoted as $Achieved(Mark, Purchase, 24Dec2010)$; a student John, who submitted the first assessment on the 10th day of the course as $Achieved(John, SubmitA1, 10)$. [1]

To specify that the goal g was achieved by the object x before or at time t, let's define the predicate $AcBy$ (AchievedBy) as:

$$AcBy(x, g, t) = \begin{cases} True & \text{if} \quad \exists t_i : Achieved(x, g, t_i), t_0 \le t_i \le t \\ False & \text{otherwise} \end{cases} \tag{2}$$

Once an object has achieved the goal, it will be true until the deadline, i.e.:

$$AcBy(x, g, t) \Rightarrow AcBy(x, g, t_j), t \le t_j \le t_d \tag{3}$$

The set of objects that have achieved the goal before or at time t is defined as:

$$DA_{D,g}(t) = \{x | x \in D, AcBy(x, g, t) = True\}. \tag{4}$$

Analogously, the set of objects from D that have not achieved (unachieved) the goal at the time t is defined as:

$$\begin{aligned} DU_{D,g}(t) &= \{x | x \in D, AcBy(x, g, t) = False\} \\ &= D \setminus DA_{D,g}(t) \end{aligned} \tag{5}$$

Next, the number of objects that achieved or unachieved the goal g up to time t is:

$$\begin{aligned} NrAcBy_{D,g}(t) = |DA_{D,g}(t)| \\ NrUnacBy_{D,g}(t) = |DU_{D,g}(t)|. \end{aligned} \tag{6}$$

Let us assume, that in the beginning, $t = t_0$, none of the objects has achieved the goal, i.e. $NrAcBy_{D,g}(t_0) = 0$ and $NrUnacBy_{D,g}(t_0) = |D|$.

and the time of the first achievement t_{first} for set D and goal g with the deadline t_d is defined as:

$$t_{first} = \min\{t | t \in [t_0, t_d] \wedge NrAcBy_{D,g}(t) > 0\} \tag{7}$$

Example 2.1 In the rest of the paper, we will use the running example of students submitting their assessment in a course to support the description of the problem definition. Let us have a set of seven students $D = \{s_1, s_2, \ldots, s_7\}$ with the goal of submitting the assessment denoted as g having the deadline in $t_d = 10$. The time is measured since time $t_0 = 0$. The student s_1 submits the assessment in $t = 3$, s_2 and s_3 in $t = 7$, the students

[1] This notation was used for simplifying the explanation. Formally, it would be an object x_i, where one of the attributes in x_i is the name.

s_4, s_5, s_6 in the deadline $t = 10$. The student s_7 does not submit at all. Then $t_{first} = 3$, and for $t \in \{7, 8\}$

$$
\begin{aligned}
DA_{D,g}(7) = DA_{D,g}(8) &= \{s_1, s_2, s_3\}, \\
DU_{D,g}(7) = DU_{D,g}(8) &= \{s_4, s_5, s_6, s_7\} \\
NrAcBy_{D,g}(7) = NrAcBy_{D,g}(8) &= 3 \\
NrUnacBy_{D,g}(7) = NrUnacBy_{D,g}(8) &= 4
\end{aligned}
$$

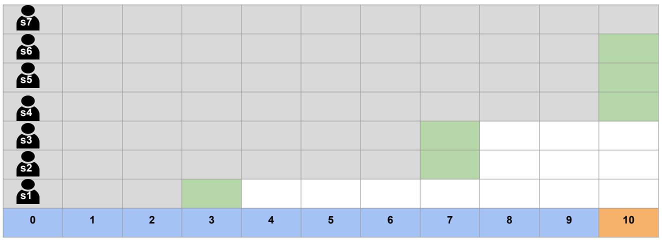

Figure 1: Running example showing the submission time of students $s_1, s_2, \ldots, s_6$ and the student s_7.

Let's suppose that the objects achieve the goal independently on each other. The number of objects that have achieved the goal before or on time (i.e. $NrAcBy(D, g, t)$ is a non-decreasing function of time with the maximum reaching in the deadline t_d:

$$
\begin{aligned}
\forall t_i, t_j \in [t_0, t_d], \quad & t_i \leq t_j : NrAcBy_{D,g}(t_0) \\
& \leq NrAcBy_{D,g}(t_i) \leq NrAcBy_{D,g}(t_j) \\
& \leq NrAcBy_{D,g}(t_d)
\end{aligned} \tag{8}
$$

Analogously, the $NrUnacBy$ is a non-increasing function, as each object, after achieving the goal, is moved from DU to DA. The imbalance ratio IR between two sets is defined as a ratio between the majority and the minority set,

$$
IR(D, g, t) = \frac{\max[NrAcBy_{D,g}(t), NrUnacBy_{D,g}(t)]}{\min[NrAcBy_{D,g}(t), NrUnacBy_{D,g}(t)]} \tag{9}
$$

In the beginning, the majority set is the $DU_{D,g}(t)$ until the moment where the number of achieved objects reaches half of the objects in D. Let's denote this time as t_{eq}. Let's also expect that not all objects will achieve the goal by the deadline. Then the IR function is defined in $[t_{first}, t_d]$, where t_{first} denotes the first achievement of the goal[2]. For $t < t_{first}$, the function is undefined. The function is non-increasing in $[t_{first}, t_{eq}]$ and non-decreasing in $[t_{eq}, t_d]$. Hence, $DU_{D,g}(t)$ and $DA_{D,g}(t)$ can exchange their roles, i.e. $DU_{D,g}(t)$ becomes minority set and $DA_{D,g}(t)$ becomes majority set. However, depending on the domain, such a case might not happen, especially if the majority of the objects achieve the goal at the last minute before the deadline. The problem becomes more interesting before the number of achievers and non-achievers is small, as less information about the reasons for achievement is available.

[2] If we expect all objects to achieve the goal before the deadline, the function would be defined in $[t_{first}, t_{last})$ with t_{last} being the last achievement time.

2.1.1. *Partial Goal Achievement Prediction Problem*

For the goal g, the set of objects D, start time t_0, the deadline time t_d and the prediction time $t_p \in [t_0, t_d)$, we define the task as a binary classification problem of achieving the goal before or at the deadline, *Partial Goal Achievement Prediction Problem at time t_p* as:

$$
GP^{nat}_{tp} = (D, g, t_d, t_0, cpm), \tag{10}
$$

, where *nat* denotes the natural order of time. Later, we will also use time running backwards from the deadline.

The first four parts of the tuple have been defined earlier and cpm is a classification performance measure that is optimised, defined as a function:

$$
cpm(y_{pred}, y_{true}) \tag{11}
$$

y_{pred} denotes the vector of predictions for the objects and y_{true} their true class labels. The examples of cpm are Accuracy, ROC AUC [3], PR AUC[4] and others. The objects that have not achieved the goal before or at time t_p are subject to predictions, defined by the function $DU_{D,g}(t_p)$, see Eq. 5. Once the object achieves the goal, its prediction is not interesting anymore. For such objects $x \in DU_{D,g}(t_p)$ at time t_p, the target classes are defined as:

$$
class(x, g, t_p, t_d) = \begin{cases} Achieve & \text{if} \quad AcBy(x, g, t_d) \\ NotAchieve & \text{if} \quad \neg AcBy(x, g, t_d) \end{cases} \tag{12}
$$

In other words, the goal is to find the model approximating the $class$ function, i.e. predicting goal achievement within the deadline for the objects that have not achieved the goal by the prediction time. Notice that for t_p, the available data are known and the first time for which we predict achieving of the goal in t_p is $t_p + 1$. The true values of the classes are known just after the deadline passes and at this time it is possible to evaluate the problem.

Example 2.2 Following the running example 2.1 with the start t_0 and the deadline $t_d = 10$, the performance measure we will use in is *ROC AUC* (now shortened as *AUC*). The partial problem for $t_p = 7$ is depicted in Figure 2, i.e. $GP^{nat}_7 = (D, g, 10, 0, AUC)$. The predictions are computed for days 0 - 7, i.e. the last prediction day is $t_p = 7$. The number of days to the deadline for which the predictions are made is $t_d - t_p = 10 - 7 = 3$.

Moreover, the last day for prediction can be $t = t_d - 1 = 9$, the partial prediction problem is defined as $GP^{nat}_9 = (D, g, 10, 0, AUC)$ and the predictions are computed for only one day, i.e. one day before the deadline t_d.

[3] Area Under ROC Curve
[4] Area under Precision-Recall Curve

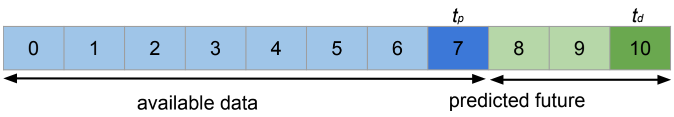

Figure 2: Time line for the partial problem GP_7 in the 7th day, with the deadline denoted as t_d (dark green), the day of prediction as t_p (dark blue). Known past (training) data are highlighted in blue, predicted future is in green.

2.1.2. Backward Aligned Problem

$$GP^R_{\tau p} = (D, g, t_d, t_0, cpm) \tag{13}$$

, where τ_p is the prediction time relative to the deadline such as $0 < \tau_p \leq |t_d - t_0|$. τ will be used from here on to emphasise that the time is counted relatively from the deadline. As from now on we will only refer to the relative partial problem and we use the notation $GP_{\tau p}$. The other parts are the same as in the definition 10. Also, $\tau_{first} = t_d - t_{first}$ denotes the first day with the goal achievement in a relative manner. Similarly, $\tau_d = 0$ will denote the deadline and $\tau_0 = t_d - t_0$ the time t_0 of the partial problem. In the rest of the paper, we will use this relative counting of time.

Example 2.3 Figure 3 depicts the same problem as in Example 2.2 but with relatively defined prediction time $\tau_p = 3$. Times relative to the prediction day will be more intuitive for defining how to compute the predictions. Such problem is defined as $GP^R_3 = GP_3 = (D, g, 10, 0, AUC)$.

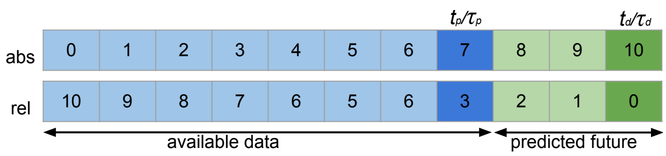

Figure 3: Time line for the partial problem 3 days before the deadline, depicted in both natural (top) and relative (bottom) time counting. The deadline t_d is in day 10 (dark green), prediction time is t_p for absolute counting, τ for relative counting.

2.1.3. Goal Prediction Problem Formulation

The definition of prediction problem integrates all partial problems for all of the available prediction times, for which relative counting is $0 < \tau_p \leq |t_d - t_0|$. Let us define the problem as *Goal Prediction Problem GP* as:

$$GP \quad = \quad (D, g, t_d, t_0, cpm) \tag{14}$$

From now on, when used in the same context, $GP_{\tau p}$ will refer to the partial problem of the problem GP for the prediction time τp and with the same D, g, t_d, t_0, cpm.

Example 2.4 Following the running example, the problem is defined as $GP = (D, g, 10, 0, AUC)$. The problem is defined in times $\tau_p \in [1, |t_d - t_0|]$, i.e. in $[1, 10]$. The first prediction will be 10 days before the deadline and the last one the day before.

The goal is to find a method which constructs a predictive model in each time after the first goal achievement, i.e. for the *Goal Prediction Problem GP*.

The key question is the selection of a performance measure for the prediction problem. This issue is highly related to the presence of imbalanced data and will be discussed later in Sec 5. For a selected classification performance measure cpm and a trained predictive model m, the problem performance measure (ppm) for the partial problem $GP_{\tau p}$ is computed by: (1) retrieving the predictions by applying the model to the testing data and (2) calculating the classification performance measure cpm using the predicted and true values (labels) of the classes. This can be denoted as:

$$ppm(GP_{\tau p}, m) \tag{15}$$

Theoretically, it is possible to provide predictions at any time, but the meaningful models can be computed only after the first object has achieved the goal in τ_{first}. In the running example, the first student submitted on day $t = 3$, i.e. in $\tau_{first} = 10 - 3 = 7$, so it is makes sense to evaluate the problem only in relative times $[1, \tau_{first}] = [1, 7]$.

Thus, ppm for a problem GP we define as the mean over all the prediction times $\tau \in [1, \tau_{first}]$ as:

$$ppm(GP, \boldsymbol{m}) \quad = \quad \frac{1}{\tau_{first}} \sum_{\tau=1}^{\tau_{first}} ppm(GP_\tau, m_\tau) \tag{16}$$

, where $\boldsymbol{m}$ denotes the vector of trained models. For each prediction time $\tau \in [1, \tau_{first}]$, there is a model m_τ. Recall that the time τ is measured backwards from the deadline t_d. Usually, we are interested in the performance of one learning algorithm type, e.g. logistic regression, which calculates in each time τ a different instance of the same model type, denoted m_τ.

2.1.4. Multi-Goal Problem

Let's consider n prediction problems $GP_1, \ldots, GP_n$, with their datasets denoted as $GP_i.D, i \in [1, n]$. The problems have the same goal g, and we want to evaluate the performance of models on all of the problems using cpm. First, in order to align the problems, the minimum of τ_{first} time instances over all the problems is selected. Let's denote this as τ_{minf}. Then a Multi-Goal Problem is defined as a matrix of partial problems

$$\boldsymbol{MGP} := (pgp_{d,\tau})_{n \times \tau_{minf}} \tag{17}$$

The rows index the partial problems by the datasets, and the columns by the prediction times. Let's suppose a matrix of trained models $\boldsymbol{MO} := (m_{d,\tau})_{n \times \tau_{minf}}$ for these problems, where the model $m_{d,\tau}$ refers to the model trained for the partial problem $pgp_{d,\tau}$, i.e. on the dataset $GP_d.D$ in the relative prediction time τ. Then, the performance measure is defined as:

$$ppm(\boldsymbol{MGP}, \boldsymbol{MO}) = \frac{1}{n \cdot \tau_{minf}} \sum_{d=1}^{n} \sum_{\tau=1}^{\tau_{minf}} ppm(pgp_{d,\tau}, m_{d,\tau}) \tag{18}$$

Example 2.5 In case of students submitting assessments, we might be interested in the average performance measure for the first assessment (denoted as a goal g) of three courses C_1, C_2, C_3 described by datasets D_1, D_2, D_3. Each course is described by a dataset D_i, i.e. $= R = \{D_1, D_2, D_3\}$. If the first submission occurs in times $\tau = 7$ for the course D_1, $\tau = 5$ for D_2 and $\tau = 6$ for D_3, then $\tau_{minf} = 5$. Hence, the performance measure would be mean of $3 \times 5 = 15$ values, i.e. mean over 15 models. As mentioned, these models will usually be trained using one type of learning algorithm, such as logistic regression.

2.2. Comparison with Gold Standard

The mean absolute value, however, might be biased towards the more accurate models closer to the deadline, for example 1 day before. The bias can also be observed in case one dataset D_i has significantly different performance than the others. The resulting measure would correctly order the models according to the performance, but the value might be difficult to interpret.

In some cases, we might have available a performance of a *gold standard* and compare the solution with that. In such cases, we define the performance in terms of loss of performance to this *gold standard*. We define it as the best model out of those trained on the testing data. This approach captures the variability and prediction power of features with respect to the predicted target. Let us define the loss of the model m to the best model m_{best} for a partial goal achieving problem $GP_{\tau p}$ as:

$$ppmLoss(GP_{taup}, m) \quad = ppm(GP_{\tau p}, m_{best}) - ppm(GP_{\tau p}, m) \tag{19}$$

Then, the performance loss for the prediction problem is defined analogously to Eq. 16 as the average across the partial problems as:

$$ppmLoss(GP, \boldsymbol{m}) = \frac{1}{\tau_{first}} \sum_{\tau=1}^{\tau_{first}} ppmLoss(GP_{\tau p}, m_\tau) \tag{20}$$

Analogously, for the Multi-Goal Problem MGP, $ppmLoss$ is defined in the same way as in the Eq. 18. Only ppm for the inner partial problem would be replaced by $ppmLoss$ (Eq. 19).

2.3. Summary of the problem definition

This section first formally defined the partial problem of achieving the goal by the deadline in prediction time t before the deadline Eq. 10. Using the backward alignment from Eq. 13 allowed us to define the Goal Prediction Problem GP in all available prediction times (Eq. 14), which is the main problem we focus to optimise in this paper. To achieve this, a problem performance metric was defined in 16 using the average across the partial prediction problems. If we have a performance of a gold standard to compare with, we propose to use the performance loss measure instead (Eq. 20). Moreover, both metrics can be used to compare across several datasets with the similar and comparable goal achievement problem, we refer to them as Multi-Goal Problem.

3. Materials and Methods: Self-Learner

This section briefly describes the generalised principle of the Self-Learning presented in [9]. The goal of the method is to learn the predictive model in all the specified time instants τ. The key aspect is the existence of behavioural features for the given population and using only the features from this population, especially of the early goal achievers. To allow this, we assume that the behaviour of objects which achieve the goal closer to the deadline follows a similar pattern as those who have already achieved the goal in advance; and also differs from the objects that will not achieve the goal within the deadline.

3.1. Extending Labelling Window

Given the partial problem $GP_{\tau p} = (D, g, t_d, t_0)$, to be able to create the prediction model for n days to the deadline it is essential to have labelled examples to be used as the training data. The true label in the training data is known only for the objects that have already achieved the goal. Because of that, a virtual labelling interval is created to measure the goal satisfaction until the prediction time. Only features from objects before the start of that interval are used. To simulate the problem as occurring in the prediction time τ_p, the window of the same size as the time remaining to the deadline was selected [9].

The method can be summarised as follows. In each instance of time τ_p remaining to the deadline τ_d:

1. In the training phase, the behavioural features are moved backwards by τ_p time units. This creates the *virtual deadline* ($virt\_\tau_d$) in the current prediction time and also the *virtual prediction time* ($virt\_\tau_p$), which is moved τ_p time units back. Recall that τ_d denotes the deadline in a relative manner.
2. For training, keep only the objects that have not achieved the goal by the virtual prediction time, i.e. this will exclude early achievers.
3. Create the labels, Achieved/NotAchieved, by looking if the objects in the training set achieved the goal by the virtual deadline.
4. Optional step: Apply a sampling method to remove the imbalance in the training data,
5. Use a selected machine learning algorithm to train the model,
6. Apply the model to all the objects in the testing set, i.e. the objects that have not achieved the goal by the prediction time τ_p.
7. Evaluate the predictions, once the deadline is due.

Example 3.1 To better illustrate the principle, Figure 4 shows an example for predicting 3 days before the deadline. After training, the model is capable to predict for the individuals that have not achieved the goal if they will succeed in the following days 2,1 or 0. The day $\tau_p = 3$ denotes the current prediction day, the green area depicts the predicted interval until the deadline, and the blue area the days from which are extracted values of the features. This view shows the shift in the training and testing data. For example, values of the features for day

$\tau_p = 3$ in the training data relate to the day $\tau_p = 6$ in the testing data, because the data for training are aligned towards the virtual deadline $virt\_\tau_d = 3$.

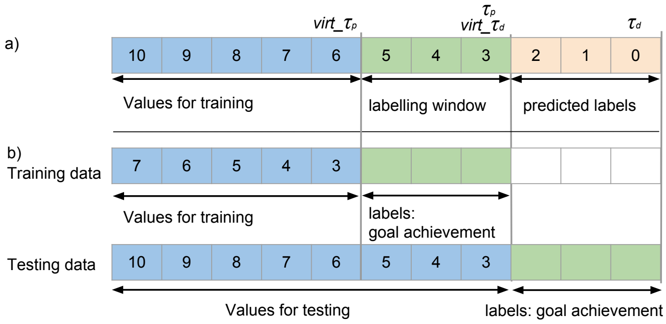

Figure 4: Classification framework for Self-Learning and testing predictions of at-risk students. The available features denote from which days the features can be used for training or testing data.

3.2. Case Study in Learning Analytics

In [9], this method has been evaluated on 4 courses from The Open University Learning Analytics Dataset (OULAD) [23]. The features used for learning include both static information such as demographic data or the date of the course registration; as well as fluid daily aggregated data from the VLE. VLE data are grouped by the activity type such as reading the course content, downloading the PDF resources or participation in forums.

The following machine learning algorithms were used to train the models: Logistic Regression (LR), Weighted Logistic Regression (LR-W) - weighted by the relative cardinality of classes; Support Vector Machines with the radial basis (SVM), Weighted SVM (SVM-W), Random Forest (RF), XGBoost (XGB), Naive Bayes (NB) and two baseline models B[NS] and B[NA]. These are defined as:

- **Base[NotSubmit]** or **B[NS]** – this assigns all the students to NotSubmit class. The model will have maximum *Recall* = 1, but it is expected to have low Precision and $Specificity = 0$.

- **Base[NotAccessed]** or **B[NA]** – it reflects the simple belief that students that have not logged into the system since its opening are not showing effort to submit the assessment. The model classifies all these students as NotSubmit and the others as Submit. The model should be able to capture the most critical students, but it is not expected to identify all of them.

The PR AUC measure was selected as a classification performance measure. It is suitable for imbalanced data, it provides a probabilistic view of the classifications and in contrast to ROC AUC, it is more focused towards the target class. In this case, they were the students at risk of failing the course. The results reported daily performances averaged across all the courses and they were compared with training on the previous presentation (PrevPres). Also, these were compared with the theoretically best scenario when training on the testing data. The method was able to achieve accurate results close to the deadline but the performance decreased significantly as moving back in time.

4. Improvements of the Self-Learner

Based on the analysis of the preliminary results, we identified several issues of Self-Learning methods. In order to create the classification model, the labelling window technique with extending size results in three issues: the first two represent information loss while the third one noise in the data.

1. **Ignoring objects' behaviour in the labelling window** – the labelling window enables creating a proxy for distinguishing which objects will or will not achieve the goal within the deadline. To simulate the problem, the size of the window was set to the same length as the number of days remaining up to the deadline. However, the features of the data in the labelling window are not used for training the model but only for labelling objects as *Achieve* or *NotAchieve*, leaving some of the features not utilised.
2. **Ignoring early goal achievers** – some objects are not part of the training data because they achieved the goal before the start of the labelling window (see the method description, point 2 in Sec 3.1). More objects are excluded since we are closer to the deadline and the window is getting narrower. On the other hand, more objects achieve the goal closer to the deadline, potentially mitigating the impact.
3. **Imbalanced Data and Noise** – the problem is inherently imbalanced, the earlier the predictions are made, the higher the imbalance ratio. This is due to the majority of objects not yet achieving the goal. Some of the data are labelled as *NotAchieve* for the training purpose, though they will achieve the goal in the end. Consequently, in the prediction time, the data of these objects contribute to the construction of the *NotAchieve* class though their patterns already indicate that they will eventually belong to the *Achieve* class. The behaviour of the *NotAchieve* students can be perceived as a kind of noise in the data, which is one of the problems that accompanies imbalanced data and it is hindering the performance of the classifiers [24]. This domain knowledge may be useful in contributing to an under-sampling method.

As a result, we designed three modifications: (1) Modifying the labelling window size, (2) Including the early goal achievers and (3) Domain-driven sampling methods.

4.1. Modifying Labelling Window size

Originally, the size of the labelling window is the same as time to the deadline. It will be denoted as *wSame*. Let's relax this condition and introduce an additional parameter specifying the size of the window. This parameter will be denoted as *SizeOfLabellingWindow*. Therefore, in the original Self-Learning strategy, $SizeOfLabellingWindow = |t_d - t_p|$. Shrinking the labelling window allows the algorithms to use more information about each object, as the behaviour of the objects previously used only for labelling is now available and used only for training. As a result, the number of objects in the labelling window decreases.

4.2. Including Early Goal Achievers

The window shrinking will increase the imbalance ratio as fewer objects are used for training, however with more information about them. Instead of ignoring the objects that achieved the goal before the start of the labelling window, these objects will extend the training dataset. The time of their goal achievement will be set as a *virtual deadline* and the behavioural features will be aligned with respect to this time. This modification will become even more important with changing the size of the labelling window. We expect that for small window size further from the deadline, the performance will drop unless these early achievers are included because there is a low number of achievers.

Including early achievers raises a question whether the characteristics of such objects differ from the later achievers, which may negatively influence the performance. In the educational context, one can argue for students who were very active and submitted very early being outliers because they are likely to have behaved differently. Thus, we will examine if there is any performance decrease for the very early achievers. To evaluate this, we define the parameter $IncludeBackWindow$, which specifies the maximum number of days from the start of the labelling window that can be used to add the students back to the training data. The days are counted backwards in time. The students that submitted in the interval

$$[virt\_\tau_d + IncludeBackWindow, \tau_p]$$

will be included in the training data. Recall, that $virt\_\tau_d$ denotes the virtual deadline or the start of the labelling window. The minimum value of the parameter is $IncludeBackWindow = 0$, when no additional achievers outside of the labelling window will be added. The original Self-Learning approach counts with the size $IncludeBackWindow = virt\_\tau_d$ parameter.

4.3. Domain Driven Sampling Methods

To decrease the imbalance ratio and eliminate the possible noise in the data, we designed an informed under-sampling method with three different strategies. On the input, we expect a machine learning algorithm able to produce a scoring predictive model. First, the model is trained making use of all data and applied to obtain a probability of achieving the goal for all object. Achievers are in the training data usually minority, i.e. it is the confidence of a classifier of being a member of the minority class. We denote the minority class as c^{min} and the majority class c^{maj}. Finally, a function $remMajData$ is used to obtain a sample without the objects from the majority class, which are on the borderline with the minority class. The schema of the approach is described in the Algorithm 1.

We propose three methods of the function $remMajData$ for removing the bottom majority class data:

Algorithm 1: Algorithm for informed under-sampling on higher level.

Input : Training data D, vector of true labels y_true, $|D| = |y|$, Classifier C
Output: Sampled data $D_{sampled}$, $D_{sampled} \subseteq D$

Train classifier C on D and y_true
y_pred = obtain probability score for all $x \in D$ using C
Sort D in an ascending way by the probabilities in y_pred
$D_{sampled} = remMajData(D, y\_true, y\_pred, \dots)$
return $D_{sampled}$

4.3.1. Method 1: EqualClassNumber

The usual goal of the sampling algorithms for imbalanced data achieves an equal number of objects in the minority and majority classes. This method accomplishes this by using the function $remTopMajority$ in Algorithm 2. The function creates a sample with removed n objects from the majority class with the highest probability of being in the minority class. Let us denote the number of majority class objects n^{maj} and the number of minority class objects n^{min}. The sampling is performed using the function $remTopMajority(D, y\_true, y\_pred, n^{maj} - n^{min})$. $n^{maj} - n^{min}$ denotes the number of objects being removed.

Algorithm 2: Function remTopMajority(D,y_true,y_pred,n)

Input : Training data D sorted by the predicted probabilities, vector of true labels y_true, vector of predicted probabilities y_pred, number of objects to remove n
Output: Union of Minority and sampled majority class objects

$D^{min} = \{x_i | y\_true_i = c^{min}\}$
$D^{maj} = \{x_i | y\_true_i = c^{maj}\}$
$n_{keep} = |D^{maj}| - n$
$D^{maj}_{sampled}$ = n objects from D^{maj} with the lowest predicted probability score of being c^{min}
return $D^{min} \cup D^{maj}_{sampled}$

4.3.2. Method 2: ClassOverlapRemoval

Instead of removing the fixed number of majority class objects, this method focuses on removing the majority objects that are overlapping with the minority class. First, the lowest prediction probability of the minority class is taken, and then it is used with the procedure $remMajorityByThr$ in Algorithm 3. The function removes all data from majority class having the predicted probability to the class c^{min} higher than the specified threshold.

For example, selecting a threshold with as the minimal value of the minority class would remove all the overlap. Another possibility is to select it as the percentile of the minority class allowing for some overlap.

Algorithm 3: Function remMajorityByThr(D, y_true, y_pred, threshold)

Input : Training data D sorted by the predicted probabilities, vector of true labels *y_true*, vector of predicted probabilities, *y_pred*, *threshold* defining maximum allowed value of prediction for the majority class

Output: Union of Minority and sampled majority class objects

$D^{min} = \{x_i | y\_true_i = c^{min}\}$

$D^{maj} = \{x_i | y\_true_i = c^{maj}\}$

$D^{maj}_{sampled} = \{x_i | x_i \in D^{maj} \wedge y\_pred_i \leq threshold\}$

return $D^{min} \cup D^{maj}_{sampled}$

4.3.3. Method 3: EstimateGoalAchievmentNumber

This method also utilises the $remTopMajority(D, y\_true, y\_pred, n)$ function in Algorithm 2. Instead of balancing the classes equally, it estimates this number based on the domain information. In our case, we have the anonymised OULAD dataset coming from the educational area. If we plot the relative number of students that had this assessment submitted on different days before the deadline, we obtain the graph in Figure 5. This suggests that in this case, the number of goal-achievers follows the exponential function, that is, the number of submitted assessments grows exponentially as the deadline approaches.

Using this data it is possible to estimate the parameters of the exponential function, that would be created by the average of all the functions. The function $\lambda(\tau)$ is defined for relatively specified time $\tau \in [0, \tau_{first}]$, where 0 denotes the deadline and τ_{first} the first goal achievement. Following this,

$$\lambda(\tau) = \lambda_0 e^{\beta\tau} \tag{21}$$

, where λ_0 is the estimated number (ratio) of objects achieved the goal in $\tau = 0$, i.e. in the deadline and β is the coefficient for time τ. We took the daily submission data from all the courses in the 2013 presentations and applied the nonlinear regression using least-squares to approximate the parameters of the exponential function. Taking the average of the courses, we get the function with the following parameters:

$$\lambda(\tau) = 0.7818e^{-0.4167\tau} \tag{22}$$

To perform the under-sampling, we need only an estimate of the function for $\tau = 0$, i.e. $\lambda(0) = 0.7818e^{-0.4167 \cdot 0} = 0.7818$. If n denotes the number of all the predicted students, the estimated number for removal is $n - 0.7818 \cdot n^{min}$). Consequently, the sampled training data are obtained using the function:

$$remTopMajority(D, y\_true, y\_pred, n - 0.7818 \cdot n^{min})).$$

On one hand, the domain informs us about the presence of a noise and the need for under-sampling. Nevertheless, the domain is fully utilised only in the Method 3, where the algorithm using the information about the underlying process and expected distribution of goal achievement in time.

5. Evaluation and Results

The evaluation data have been taken from the educational domain, in particular from a distance based higher educational institution. The Self-Learning approach with the proposed modifications has been applied to identify students at risk of failing the course by focusing on those that are unlikely to submit the first assessment.

5.1. Experimental setup

We utilised The Open University Learning Analytics Dataset (OULAD) [23] for the evaluation. This anonymised dataset contains 7 courses denoted as AAA to GGG with 4 presentations of the courses in years 2013 and 2014. Presentations starting in February are denoted as B and presentations in October as J. The dataset contains the presentations 2013B, 2013J, 2014B and 2014J. The courses cover a broad range of fields such as science, technology, engineering, maths (STEM) and social sciences. AAA is a level three course, GGG is a preparatory course, and the rest are level one courses.

We excluded from the experiments the level-3 course AAA. At this level, students are already advanced and identification of at-risk students is replaced by focusing on improving the knowledge gain of such students. To compare the Self-Learning approach with training on the previous presentation, we selected only those courses from the 2014J and 2014B presentations, for which 2013J or 2014J presentation exists. For this reason, the course CCC is missing in the experiments.

The courses have between 750 and 2500 students with the pass-rate ranging from 37 to 60%. The goal was to predict the submission of the first assessment by students registered in the course within the deadline. The number of students, pass rate, submission rate and the deadline day for all the courses under analysis is described in Table 1.

The earliest deadline is the 12th day (BBB-2014B) but the evaluation was performed for days 1-19. This was selected as the common minimal day for all the courses when the models were able to be trained, i.e. at least one student submitted the assessment. The courses have a start day (day 0) but the VLE opens even before so students are able to study in advance. Some students submit even before the official start of the course, which states also for BBB-2014B and that's why these models were able to be trained even before the course start.

5.2. Difference in Setup with Published Results

Here, the experimental setup slightly differs from the published results in the paper in [9]. In that article, only one presentation (the most recent one) was used, while here the focus was extended to both of 2014 presentations. Further, we decided to include a previously discarded preparatory course GGG, since there is interest at The Open University to widen analysis of at-risk students at an early level.

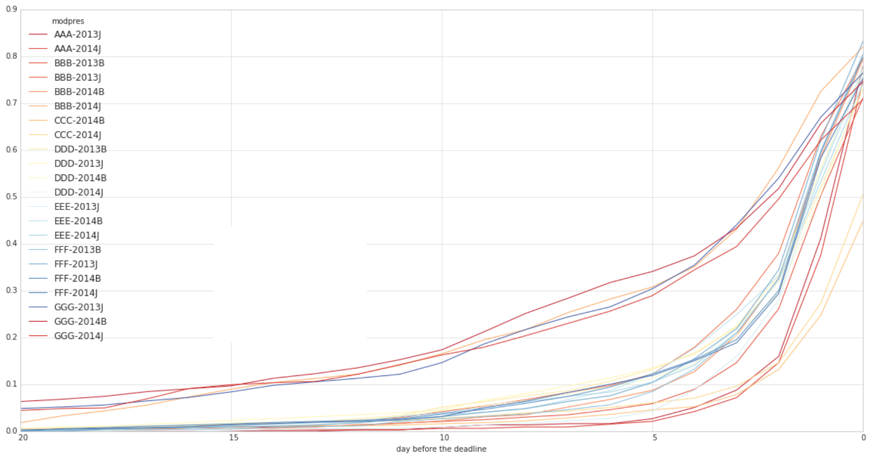

Figure 5: Ratio of submitted students in the data in all the courses of OULAD.

Table 1: Information about the courses under analysis - 2014 presentation

Course	Pres	No. of students	Pass Rate [%]	A1 S/NS [%]	Deadline [Day]
BBB	2014B	1613	54.93	73.65	12
BBB	2014J	2292	49.74	77.31	19
DDD	2014B	1228	60.99	75.65	25
DDD	2014J	1803	56.07	78.48	20
EEE	2014J	1188	42.42	78.20	33
FFF	2014B	1500	56.40	79.40	24
FFF	2014J	2365	52.77	77.12	24
GGG	2014J	749	40.72	77.97	61

The ROC AUC was added as a supplemental metric for the evaluation, as it shows a different view on the performance, counting also the correctly identified submitted students. Most importantly, the values for the PR AUC slightly differ from the article. We discovered that the area under PR curve in the used Sci-Kit library [25] was computed by linear interpolation. In this case, it might give overly optimistic results for poorly performing models, particularly the baseline models.[5]

5.3. Evaluation strategy

The evaluation of the models from the case study in [9] was based on comparing the performance measures in each day separately.

In this paper, we decided to use the strategy, which provides a performance measure for the Multi-Goal Problem, i.e. for all the times and also for more datasets. This also enables easier comparison of the proposed modifications. We used the performance of the gold standard defined as the model that is trained using all the data that are available during the testing, i.e. with the correct labels. As such we utilised the performance loss for the Problem from the Eq. 20, and for Multi-Goal Problem 18.

We used the same machine learning algorithms as in our previous results in [9] and listed previously in 3.2. Also we used the following sampling methods from the *imbalanced-learn library* [26]. Namely we used Random under-sampling (Rand-Under), Tomek-Links, Extended Nearest Neighbours (ENN), Neighbour Cleaning Rule (NCR), Random over-sampling (Rand-Over), SMOTE [27], SMOTE-ENN [28], SMOTE-Tomek [29]. They include both uninformed and informed methods based on under-sampling and over-sampling. They are the algorithms used in many papers for tackling the imbalanced data.

5.4. Results

The evaluation is split into two parts, first replicating the results from [9] using the new evaluation setup. Only one measure was applied to describe the performance of the whole system. Selected machine learning algorithms were used together with several sampling methods to improve the performance in the imbalanced data. Both ROC AUC and PR AUC were used to evaluate the results. The second part presents the results and analysis of the improvements.

5.4.1. Replicated original Self-Learning results

The results are depicted in Table 2 for PR AUC and in Table 3 for ROC AUC. The tables indicate that the lowest overall loss was achieved by Random Forest. For PR AUC, SMOTE-ENN performed best, followed by random over-sampling. The solution without any sampling was worse by 0.0074. For ROC

[5]This issue has been fixed in the new release in August 2017 in https://github.com/scikit-learn/scikit-learn/issues/5379.

AUC the lowest loss was achieved by NCR, but again, with only small improvement 0.0012 over Random Forest without any sampling method.

For PR AUC, SMOTE-ENN was the technique that improved the performance best for four of the models and random undersampling for the other three. The results are the same for ROC AUC, with the only exception being Random Forest with NCR. The highest impact of sampling methods was achieved for LR decreasing the loss of PR AUC by 0.1958 and for SVM by 0.1351. A similar result had been achieved for ROC AUC, but the gap between LR and SVM has widened.

5.5. Modification 1 and 2: Window size and Including Early Goal Achievers

Changing the size of the labelling window enables us to compare whether it is more important to have additional information about an individual student or more students who submitted (i.e. the minority class) in the training data. For each prediction day, the size of the labelling window has been changed from 1 to 19. The performance losses have been averaged across all prediction days and across all the courses. The results for various sizes of the window were compared to the original solution *wSame* when the size changed with respect to the number of days remaining up to the deadline.

The models were built both with and without including the submitted students before the beginning of the labelling window (Modification 2). We present both modifications together to highlight their relationship. As the results will show, it is more important to include the objects when the window gets smaller.

Figure 6 and Table 4 demonstrate the results for both ROC and PR AUC losses. INC denotes the solution with including students, NOTINC is the original solution, i.e. without including these students.

The predictions have been computed for days 1 to 19, and the performance losses have been averaged across all prediction days and across all the courses. With the *wSame* strategy, the performance measure was also computed across days and courses. This value is independent of the parameter for the fixed window size $SizeOfLabellingWindow$; the value is constant, and it is represented as a horizontal line. Therefore, two different window sizes, both with INC and NOTINC strategies, will result in four possible strategies and models. Afterwards, we computed their performance loss according to the Eq. 20. Two window sizes 1 and 2 result in strategies created by $(1, INC)$, $(1, NOTINC)$,$(2, INC)$,$(2, NOTINC)$.

Figure 6 shows that the loss of PR AUC is higher than that for ROC AUC. For both measures, the loss slowly decreases for INC and NOTINC from size 19 to size 7. For the sizes 19 to 10, the difference between the INC and NOTINC are only 0.001. Moving from the window 9 to 1, the differences start increasing, mainly because the loss of NOTINC starts increasing exponentially until the window size 1. This is caused by increasing the imbalance in the training data due to narrowing the labelling window and consequently the number of students submitting in this interval. However, close to the deadline, this problem is not observed, because enough students submit during the interval. It confirms the results from the original solution [9], where the highest performance is achieved in the last days despite a small window size. However, the performance decreases for days further away from the deadline t_d.

Adding the early submitting students also helps to mitigate the impact of the narrow window. For PR AUC the loss is the lowest for size 4. For ROC AUC, the loss is decreasing until the end of window size 1.

For comparison, the dotted lines in Figure 6 denote the *wSame* solutions. For both metrics, the values for NOTINC and NOTINC of *wSame* are almost equal, note the last row in Table 4. This means, that including the students for *wSame* itself does not significantly improve the performance. For NOTINC, there is an interval with loss lower than for the *wSame*. For PR AUC it is $[5, 9]$ and for ROC AUC $[5, 11]$. Due to narrowing window, the performance degrades from the window size 7 to 1.

The results showed the best performance improvement for $SizeOfLabellingWindow \leq 7$ and the case when the early submitting students were included. Window sizes $[1, 7]$ were selected for further evaluation and analysis for improvement of the model. Window size 1 is the global minimum of the ROC AUC loss and 4 and 6 are the global minimums for the PR AUC loss. ROC AUC has a local minimum for the size 7.

5.5.1. Impact of Very Early Achievers

As shown, including students who submitted before the start of the labelling window improves the performance. The question is, whether the students who submitted a long time before the start of the labelling interval do not hinder the performance. Especially, as approaching the deadline, one might expect that students who submit among the first behave differently than those who submit at the last moment.

Having $SizeOfLabellingWindow = 1$, we varied the maximum number of days (IncludeBackWindow) before the labelling window that is allowed for a student to be added to the training data. The value of the parameter was set from 0 to 40, which is the same as considering 'infinity', given that the maximum deadline in the dataset is 61, seen in course GGG. If the data pattern of the early achievers was different, we would notice the decrease of performance measures for increasing value of the parameter *IncludeBackWindow*. The results in Figure 7 show this neither for PR AUC loss nor for the ROC AUC loss. The only visible trend is the exponential increase of loss when lowering the maximum window size. The further analysis showed that the main loss does not come from the days close to the deadline, but those that are far away.

In conclusion, including all the students back into the analysis, even with the size of the labelling window 1, does not negatively influence performance.

5.6. Modification 3: Domain Driven Sampling Methods

Taking the best results from the previous experiments, labelling window of sizes $1-7$ were taken for evaluation together with the original window, (i.e. *wSame*). The three proposed sampling methods were compared with each other and to the

Table 2: PR AUC loss on the selected courses using all the machine learning models and the sampling methods.

	ENN	NCR	None	Rand Over	Rand Under	SMOTE Tomek	SMOTE	SMOTE ENN	Tomek Links
B[NS]	0.4366	0.4366	0.4366	0.4366	0.4366	0.4366	0.4366	0.4366	0.4366
B[NA]	0.3285	0.3285	0.3285	0.3285	0.3285	0.3285	0.3285	0.3285	0.3285
LR-W	0.2570	0.2605	0.2694	0.2911	0.2048	0.3337	0.3340	0.2290	0.2688
LR	0.3788	0.3849	0.4006	0.2911	0.2048	0.3370	0.3373	0.2266	0.4005
NB	0.4055	0.4081	0.4143	0.4154	0.3594	0.4405	0.4401	0.3780	0.4139
RF	0.1800	0.1786	0.1788	0.1879	0.1741	0.1826	0.1839	**0.1714**	0.1783
SVM	0.3430	0.3506	0.3693	0.2666	0.2654	0.3111	0.3106	0.2342	0.3686
SVM-W	0.2257	0.2263	0.2322	0.2666	0.2654	0.3107	0.3095	0.2253	0.2306
XGB	0.3239	0.3238	0.3262	0.2481	0.2441	0.2549	0.2554	0.2360	0.3263

Table 3: ROC AUC loss on the selected courses using all the machine learning models and the sampling methods.

	ENN	NCR	None	Rand Over	Rand Under	SMOTE Tomek	SMOTE	SMOTE ENN	Tomek Links
B[NS]	0.3746	0.3746	0.3746	0.3746	0.3746	0.3746	0.3746	0.3746	0.3746
B[NA]	0.2715	0.2715	0.2715	0.2715	0.2715	0.2715	0.2715	0.2715	0.2715
LR-W	0.1817	0.1837	0.1888	0.2055	0.1490	0.2408	0.2408	0.1698	0.1885
LR	0.3557	0.3629	0.3811	0.2055	0.1490	0.2522	0.2521	0.1784	0.3812
NB	0.3201	0.3232	0.3319	0.3334	0.2552	0.3833	0.3826	0.2881	0.3314
RF	0.1434	**0.1413**	0.1425	0.1543	0.1432	0.1606	0.1616	0.1436	0.1428
SVM	0.2461	0.2508	0.2662	0.1902	0.2780	0.2249	0.2246	0.1762	0.2656
SVM-W	0.1679	0.1676	0.1710	0.1902	0.2780	0.2243	0.2226	0.1673	0.1702
XGB	0.2330	0.2342	0.2377	0.1849	0.1796	0.1972	0.1971	0.1725	0.2379

Table 4: Loss of PR AUC and ROC AUC for different sizes of the labelling window and the influence of including the early achievers in the training data. INC denotes including students that submitted the assessment before the start of the labelling window.

winSize	PR AUC LOSS NOTINC	ROC AUC LOSS NOTINC	PR AUC LOSS INC	ROC AUC LOSS INC
1	0.2553	0.1956	0.1707	**0.1265**
2	0.2133	0.1584	0.1716	0.1297
3	0.1958	0.1495	0.1716	0.1304
4	0.1844	0.1431	**0.1672**	0.1279
5	0.1794	0.1379	0.1680	0.1288
6	**0.1730**	0.1332	**0.1674**	0.1307
7	**0.1732**	**0.1320**	0.1690	0.1289
8	0.1757	0.1347	0.1703	0.1319
9	0.1775	0.1366	0.1717	0.1332
10	0.1791	0.1386	0.1780	0.1378
11	0.1771	0.1382	0.1782	0.1382
12	0.1794	0.1408	0.1784	0.1404
13	0.1797	0.1430	0.1805	0.1438
14	0.1860	0.1472	0.1841	0.1477
15	0.1887	0.1526	0.1881	0.1536
16	0.1867	0.1519	0.1866	0.1527
17	0.1891	0.1565	0.1900	0.1560
18	0.1908	0.1567	0.1883	0.1558
19	0.1920	0.1595	0.1918	0.1594
same	0.1788	0.1425	0.1765	0.1408

results without any sampling. Again, the loss of PR AUC and ROC AUC were the measures of interest.

Table 5 shows the loss of PR AUC. EQ_CLS denotes the sampling with the equal number of data in both classes, EST_RAT is the estimation of the submission ratio, and RM_OVLAP stands for the removal of the overlap between the classes. For RM_OVLAP, 100 denotes the removal of all the majority data that overlap with the minority class and 25 al-

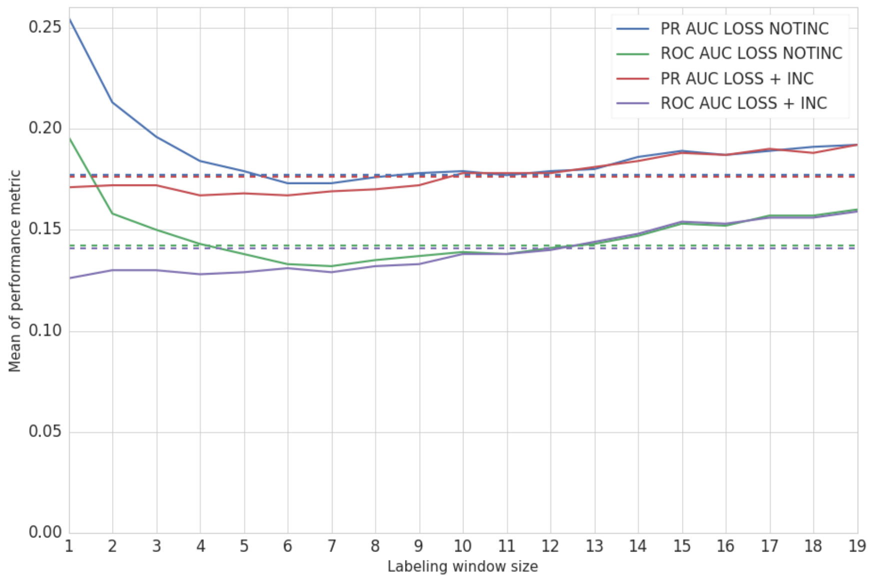

Figure 6: Loss of PR AUC and ROC AUC for different sizes of the labelling window and the influence of including the early achievers in the training data. INC denotes including students that submitted the assessment before the start of the labelling window. The dotted lines correspond to the *wSame* approach, being comparable to the strategy denoted with the same colour with full line.

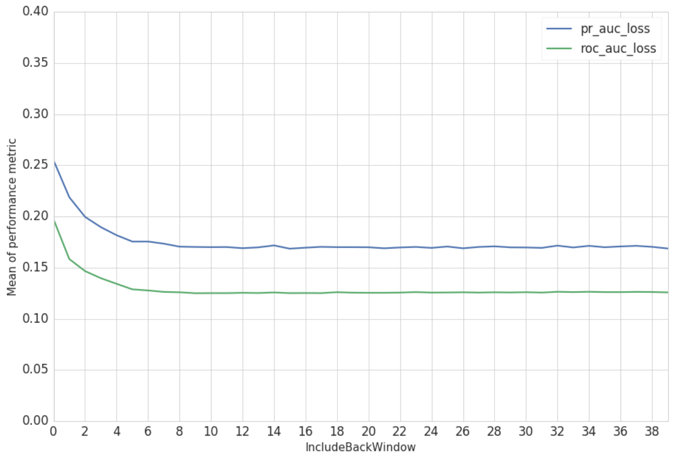

Figure 7: Loss of PR AUC and ROC AUC for varying maximum days from the start of the labelling window for students to be included back in the training data, for $SizeOfLabellingWindow = 1$

lowing 25% of minority data to overlap with the majority data. Table 6 shows the same analysis using ROC AUC loss as the measure.

Results indicate that the best performance for both measures is achieved for the EST_RAT. With the best PR AUC achieved for window size 2, the loss was decreased from 0.1716 to 0.1417, i.e. by 0.0299. Window 4 has loss of only 0.0010 higher, i.e. 0.1427. For window 1 the loss is 0.1432. The results for ROC AUC are similar, EST_RAT for window size 1 achieving the best results with the loss 0.1133 followed by EST_RAT with window size 2 with a loss of 1162.

From the other sampling methods, EQ_CLS improved performance but only for PR AUC. The problem with EQ_CLS is probably the removal of too many students, retaining mainly the most obvious submitters. For the PR AUC, it still performs well.

5.6.1. Sampling With the Original Solution

The sampling methods were used to improve the performance of the original version of Self-Learning, adjusting the labelling window and not including students submitting before the start of the window. Table 7 shows the results for both PR AUC and ROC AUC loss. Two important findings are that (1) the EST_RAT performs again best for both measures and (2) the results for sampling confirm the previous finding that using a smaller labelling window leads to a better performance.

5.7. Comparison With Existing Sampling Methods

Existing sampling methods were applied to data with window sizes $1-7$ and compared with the domain-driven methods. The results for existing methods for PR AUC loss in Table 8 and for ROC AUC in Table 9 indicate that out of them the best-performing method, in general, is random under-sampling. For PR AUC it reaches the minimum loss for the window 2. For ROC AUC the global minimum is achieved by the SMOTE-ENN for window size 1, but in the other windows, random under-sampling performs better. Decreasing the window size helps the performance, but not as much as for the domain-driven sampling. When compared with the domain driven techniques, for PR AUC the best existing method reaches the loss 0.1554 while the EST_RAT 0.1417, see Table 5. Similarly, for ROC AUC, the best solution from the existing methods with the loss 0.1235 is outperformed by EST_RAT with the loss 0.1130, see Table 6.

5.7.1. Daily Performance Analysis

For closer examination, Random Under-Sampling and SMOTE-ENN have been selected together with

Table 5: PR AUC loss for domain-driven sampling techniques in various window sizes, winSize denotes the SizeOfLabellingWindow parameter

winSize	EQ_CLS	EST_RAT	None	RM_OVLAP_100	RM_OVLAP_75
1	0.1559	0.1432	0.1707	0.1772	0.1728
2	0.1554	**0.1417**	0.1716	0.1781	0.1730
3	0.1594	0.1447	0.1716	0.1750	0.1728
4	0.1596	0.1427	0.1672	0.1746	0.1684
5	0.1616	0.1450	0.1680	0.1737	0.1691
6	0.1625	0.1458	0.1674	0.1723	0.1692
7	0.1644	0.1484	0.1690	0.1733	0.1682
wSame	0.1833	0.1603	0.1765	0.1814	0.1783

Table 6: ROC AUC loss for domain-driven sampling techniques in various window sizes, winSize denotes the SizeOfLabellingWindow parameter

winSize	EQ_CLS	EST_RAT	None	RM_OVLAP_100	RM_OVLAP_75
1	0.1317	**0.1130**	0.1265	0.1321	0.1279
2	0.1330	0.1160	0.1297	0.1352	0.1304
3	0.1377	0.1174	0.1304	0.1347	0.1308
4	0.1368	0.1170	0.1279	0.1361	0.1289
5	0.1374	0.1199	0.1288	0.1360	0.1299
6	0.1395	0.1220	0.1307	0.1352	0.1327
7	0.1395	0.1233	0.1289	0.1364	0.1300
wSame	0.1597	0.1345	0.1408	0.1470	0.1430

Table 7: PR AUC and ROC AUC loss for domain-driven sampling techniques for the original version of Self-Learning

sampler	PR AUC LOSS	ROC AUC LOSS
EQ_CLS	0.1824	0.1570
EST_RAT	**0.1603**	**0.1349**
None	0.1788	0.1425
RM_OVLAP_100	0.1781	0.1443
RM_OVLAP_75	0.1779	0.1420

the two best performing domain-driven methods, i.e. EST_RAT and EQ_CLS, all of them with the parameter $SizeOfLabellingWindow = 1$. Their results in terms of the average absolute performance are plotted in Figure 8 and 9. For both measures, the best performer is EST_RAT. For PR AUC, however, in days $14 - 11$ the EQ_CLS performs slightly better. Furthermore, the main increase in performance of the sampling methods occurs between days $19 - 10$. From day 10 to the deadline, the differences are negligible, apart from day 1. On day 1, the EQ_CLS method performs worse than the other methods. The plotted results for window size 2, which achieves slightly higher results for ROC AUC, were consistent with Figure 8 and 9, thus they were omitted in the figures for brevity.

5.8. Impact of the Improvements

To make the impact of the single improvements and their combination clear, we selected the best results for the window analysis (i.e. window sizes 1 and 2) and the best sampling method: EST_RAT. We compute their losses both separately and in combination.

Table 10 shows the losses of PR AUC and ROC AUC and the difference between the loss of the original solution (the first row in italics) and the loss of the improvement. The differences are denoted as d_prauc_loss and d_rocauc_loss. This reveals that the highest individual contribution is achieved by EST_RAT sampling for both metrics. The combination of improvements substantially contributes to the results. Especially, the small window size is only useful when combined with including the early achiever. But the best results are achieved when all improvements are applied together. For example, while using only window size 2 with INC leads to a difference of 0.0072 and using EST_RAT to 0.0184 for PR AUC, their combination makes the difference $d\_prauc\_loss = 0.0371$.

Finally, Figures 10 and 11 show the impact of improvements per day in the context of the baseline model, the model trained on the previous presentation (*PrevPres*) and the model trained on the testing data (*Self-Test*). EST_RAT with both window sizes 1 and 2 are presented, denoted in the figure as *Self-LearningSW1* and *Self-LearningSW2*. Both of them improved both PR ROC and ROC AUC especially in the early phases of predictions. They narrowed the performance gap, especially to the *PrevPres* strategy. For example, the difference for ROC AUC instead of being visible from day 10 back to the past, is now visible around days 16-17.

5.9. Summary Results

We discussed issues pertaining to the original Self-Learning approach. Based on these issues, three types of modifications

Table 8: PR AUC loss for sampling techniques in various window sizes

	ENN	NCR	None	Rand Over	Rand Under	SMOTE Tomek	SMOTE	SMOTE ENN	Tomek Links
1	0.1712	0.1714	0.1707	0.1714	0.1573	0.1720	0.1699	0.1629	0.1709
2	0.1729	0.1710	0.1716	0.1723	**0.1554**	0.1714	0.1704	0.1632	0.1719
3	0.1712	0.1709	0.1716	0.1720	0.1594	0.1725	0.1717	0.1629	0.1691
4	0.1692	0.1692	0.1672	0.1713	0.1595	0.1683	0.1672	0.1609	0.1666
5	0.1693	0.1670	0.1680	0.1690	0.1578	0.1662	0.1673	0.1593	0.1677
6	0.1716	0.1687	0.1674	0.1733	0.1571	0.1692	0.1689	0.1593	0.1668
7	0.1710	0.1706	0.1690	0.1749	0.1589	0.1690	0.1686	0.1591	0.1675
wSame	0.1777	0.1774	0.1765	0.1867	0.1694	0.1782	0.1771	0.1693	0.1773

Table 9: ROC AUC loss for sampling techniques in various window sizes

	ENN	NCR	None	Rand Over	Rand Under	SMOTE Tomek	SMOTE	SMOTE ENN	Tomek Links
1	0.1260	0.1256	0.1265	0.1310	0.1260	0.1359	0.1349	**0.1235**	0.1256
2	0.1308	0.1302	0.1297	0.1338	0.1256	0.1383	0.1382	0.1277	0.1294
3	0.1306	0.1303	0.1304	0.1355	0.1272	0.1414	0.1409	0.1277	0.1297
4	0.1298	0.1288	0.1279	0.1347	0.1275	0.1382	0.1385	0.1282	0.1272
5	0.1294	0.1298	0.1288	0.1341	0.1278	0.1378	0.1388	0.1286	0.1285
6	0.1325	0.1311	0.1307	0.1374	0.1278	0.1412	0.1409	0.1297	0.1305
7	0.1311	0.1313	0.1289	0.1361	0.1307	0.1404	0.1409	0.1298	0.1301
wSame	0.1415	0.1414	0.1408	0.1534	0.1409	0.1549	0.1539	0.1421	0.1417

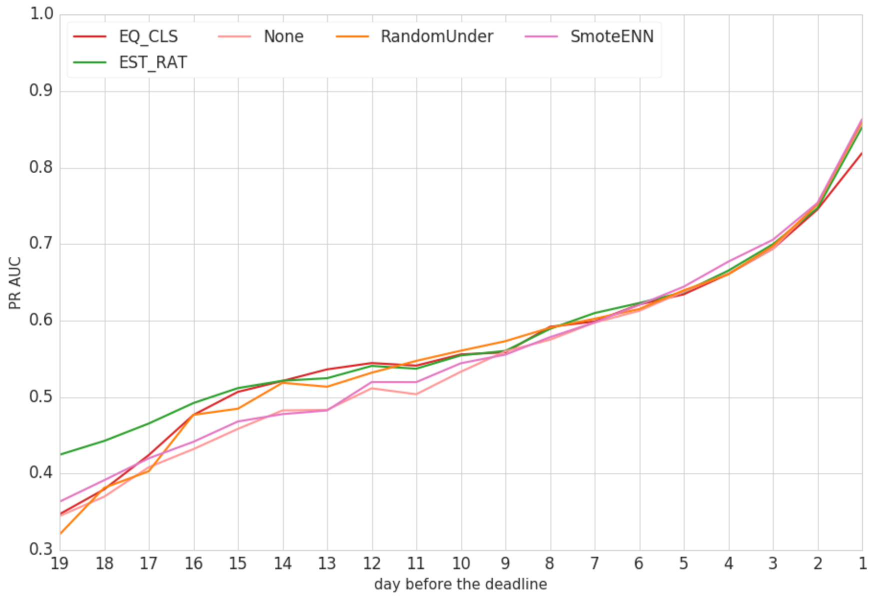

Figure 8: Daily comparison of PR AUC for sampling methods

were designed: (1) modifying labelling window size, (2) including students submitting before the start of the window, and (3) using domain-driven sampling methods.

To better evaluate the impact of the modifications, a new evaluation strategy was defined. This strategy is based on computing the loss of performance against the model that was trained on the same data as tested (*Self-Test*) representing the limits of what the model can explain based on the given features.

Using the modification 1 and 2 lead to the improvement only when combined together. Window size = 1 and 2 produced the best results. If such window sizes are used without including early achievers, the performance even drops. On the other hand, including only early achievers without narrowing the labelling window doesn't lead to any improvement. The best results were achieved when such improvements were combined with the best performing sampling-method (EST_RAT). This method was able to increase the performance even when used

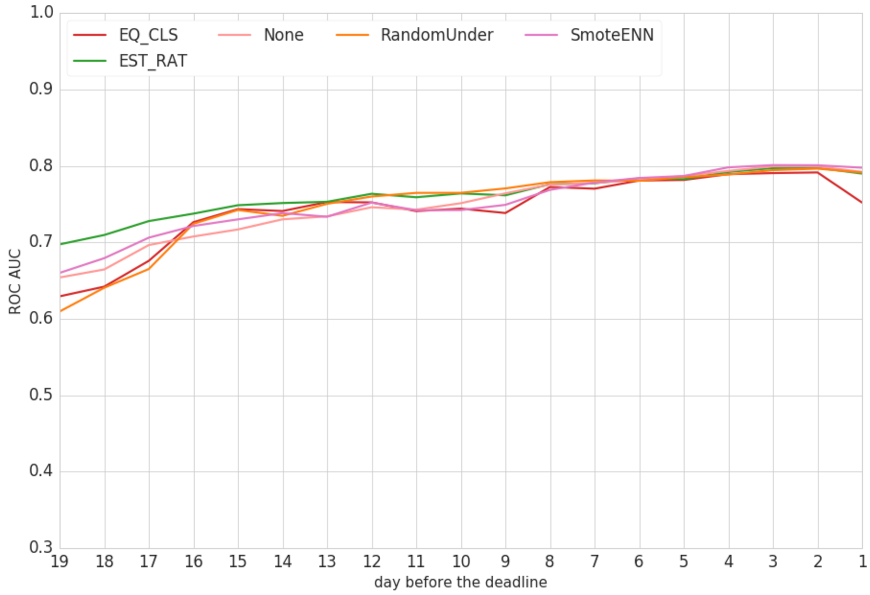

Figure 9: Daily comparison of ROC AUC for sampling methods

Table 10: Performance loss of individual best improvements and their combination and their difference from the original solution (first row - italic). Positive difference means a performance gain, negative difference performance loss.

(1)winSize	(2)Include	(3)EST_RAT	PR AUC	ROC AUC	d_prauc_loss	d_rocauc_loss
wSame	-	-	*0.1788*	*0.1425*	*0*	*0*
wSame	INC	-	0.1765	0.1408	0.0023	0.0005
wSame	-	YES	0.1603	0.1349	0.0185	0.0064
wSame	INC	YES	0.1603	0.1345	0.0185	0.0068
1	-	-	0.2553	0.1956	-0.0765	-0.0543
1	INC	-	0.1707	0.1265	0.0081	0.0148
1	-	YES	0.1957	0.1601	-0.0169	-0.0188
1	INC	YES	0.1432	**0.1130**	0.0356	**0.0283**
2	-	-	0.2133	0.1584	-0.0345	-0.0171
2	INC	-	0.1716	0.1297	0.0072	0.0116
2	-	YES	0.1637	0.1327	0.0151	0.0086
2	INC	YES	**0.1417**	0.1160	**0.0371**	0.0253

separately but not as much as when in combination.

6. Conclusions

The Learning Analytics domain, and identification of at-risk students without legacy data, in particular, motivated the need to articulate the general problem of achieving a goal within a deadline. As such, the problem faces a large imbalance especially in the beginning as only a few objects satisfy the goal very early. We proposed the Self-Learning method [9] and evaluated it in a case study of predicting at-risk students. In this domain, the lack of legacy data means that the course is presented for the first time and there is no other course that is structurally similar in order to provide data for building the predictive model by machine learning algorithms. The proposed approach showed the predictive power, but there was also a performance gap with respect to training the model using legacy data from the previous presentation of the same course. Based on knowledge about the problem, we designed three modifications that tackle the loss of information and the imbalance in the data caused by the noise. This is crucial especially at the beginning of the predictions. Modification (1) and (2) improved the performance of the original solution only when used in combination and for small window sizes. The best results were achieved when these were used together with modification (3).

To evaluate the quality of the suggested solutions and modifications, we designed new evaluation strategies that measure the performance summarised both across all prediction times and for all datasets (here the available courses) with the classification measure by a single value. Instead of counting the absolute value, it calculates the loss against the best achievable model, which is the one trained on the testing dataset. Relating performance measure of the method by comparing with the theoretical baseline makes it possible to evaluate the methods while eliminating the impact of other factors, such as using different data and features.

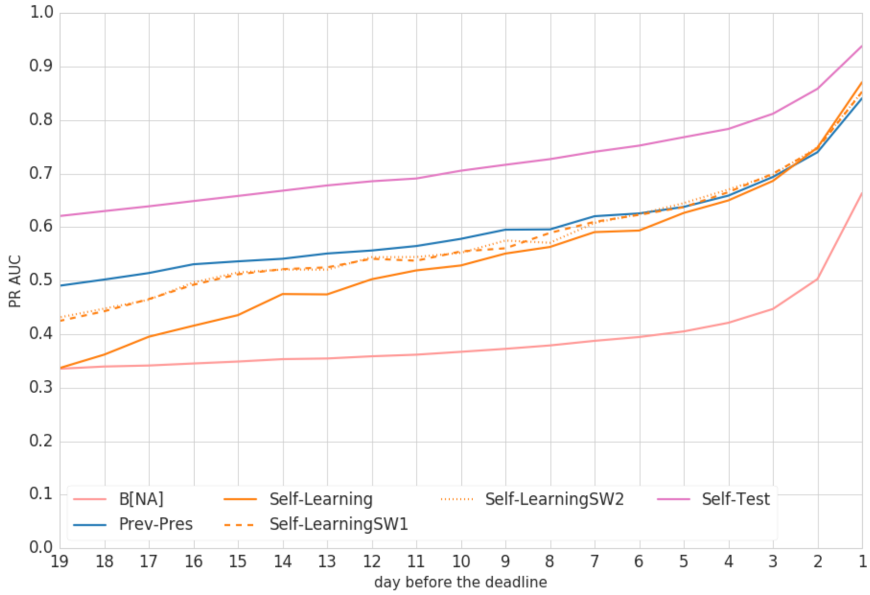

Figure 10: Comparison of PR AUC of the best Self-Learning improvement with the best solutions from the original approach

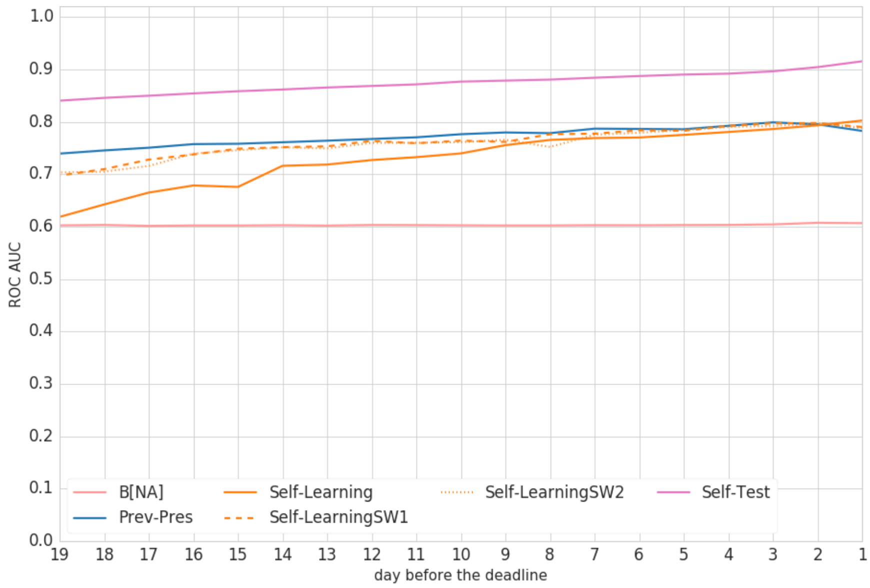

Figure 11: Comparison of ROC AUC of the best improvement with the best solutions from the original approach

The domain helped to define the problem of achieving a goal within the deadline, revealing that the problem naturally generates imbalanced data. Moreover, the information about the process helped to realise the loss of information in the original solution and guide the design of the sampling method. The underlying process of student submission generates high activity and more submissions close to the deadline and motivated us to use the exponential function for estimating the number of sampled instances. The presence of a deadline is something, what makes this problem unique and influences the behaviour of the participating subjects, i.e. students. It is likely a problem specific to a human behaviour. One of the possible explanation is procrastination, a phenomenon of preferring short-term goal over the long-term goals and then postponing the activity until the very end [20, 30].

The contribution of our work can be summarised according to the posed research questions as:

1. We provide a generalised problem for prediction of goal achievement by objects within a specified deadline, with a natural presence of imbalanced data especially in the beginning of the training. (RQ1).
2. Using the information about the problem, we extended the framework and improved the performance by (1) parametrised labelling windows size, (2) including objects that were not included in the labelling window; and (3) designing a domain-driven under-sampling strategy with estim-

ating the number of expected objects that will achieve the goal. Strategy (1) and (2) lead to improvement when used in combination together and the best improvement was reached best when these were combined with strategy (3). In this way, the performance narrows the gap between the Self-Learning and the theoretical possibilities of the machine learning defined by training on the testing data, i.e. Self-Test model (RQ2).

6.1. Future work

Several avenues for further research are possible. First, the suitability of the method across different domains can be investigated and possibly discover whether the domain-specific improvements are generalisable in different contexts. These can include other tasks, such as completing individual or team-based goals within a company [31], or paying the tax returns in time [21]. Indeed, the data collection is necessary to confirm or refute this hypothesis.

Moreover, the theoretical properties of the underlying process can be studied with more focus on parameters influencing the distribution of achievement times. Investigating which parameters affect the submission of the assessment, or achieving goals in general, can lead not only to an additional classification improvement but also to better understanding of this process. Parameters that would allow controlling the process may be discovered, and the process can be optimised so that more objects achieve the goal.

Acknowledgement

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science - LQ1602".

References

[1] H. Vossensteyn, A. Kottmann, B. Jongbloed, F. Kaiser, Drop-out and completion in higher education in europe.

[2] D. Koller, A. Ng, C. Do, Z. Chen, Retention and intention in massive open online courses: In depth, EDUCAUSE.

[3] K. E. Arnold, M. D. Pistilli, Course signals at purdue: Using learning analytics to increase student success, in: Proc. of the 2Nd Intl. Conf. on Learning Analytics and Knowledge, LAK '12, ACM, New York, NY, USA, 2012, pp. 267–270. doi:10.1145/2330601.2330666.

[4] S. M. Jayaprakash, E. W. Moody, E. J. M. Lauria, J. R. Regan, J. D. Baron, Early Alert of Academically At-Risk Students: An Open Source Analytics Initiative, Journal of Learning Analytics (1) (2014) 6–47.

[5] J. Kuzilek, M. Hlosta, D. Herrmannova, Z. Zdrahal, A. Wolff, Ou analyse: Analysing at-risk students at the open university, Learning Analytics Review (2015) 1–16.

[6] R. S. Baker, D. Lindrum, M. J. Lindrum, D. Perkowski, Analyzing early at-risk factors in higher education e-learning courses, Students at Risk: Detection and Remediation.

[7] A. Wolff, Z. Zdrahal, D. Herrmannova, J. Kuzilek, M. Hlosta, Developing predictive models for early detection of at-risk students on distance learning modules, in: Machine Learning and Learning Analytics workshop at LAK14, 24-28 March 2014, Indianapolis, Indiana, USA, 2014, p. 4.

[8] C. Taylor, K. Veeramachaneni, U. O'Reilly, Likely to stop? predicting stopout in massive open online courses, CoRR.

[9] M. Hlosta, Z. Zdrahal, J. Zendulka, Ouroboros: Early identification of at-risk students without models based on legacy data, in: Proc. of the 7th Intl. Learning Analytics & Knowledge Conference, LAK '17, ACM, New York, NY, USA, 2017, pp. 6–15. doi:10.1145/3027385.3027449.

[10] H. He, E. A. Garcia, Learning from imbalanced data, IEEE Trans. on Knowl. and Data Eng. (9) (2009) 1263–1284.

[11] N. Japkowicz, S. Stephen, The class imbalance problem: A systematic study, Intell. Data Anal. (5) (2002) 429–449.

[12] P. Branco, L. Torgo, R. P. Ribeiro, A survey of predictive modeling on imbalanced domains, ACM Comput. Surv. (2) (2016) 31:1–31:50.

[13] L. A. Jeni, J. F. Cohn, F. De La Torre, Facing imbalanced data–recommendations for the use of performance metrics, in: Proc. of the 2013 Humaine Association Conf. on Affective Computing and Intelligent Interaction, ACII '13, 2013, pp. 245–251. doi:10.1109/ACII.2013.47. URL `http://dx.doi.org/10.1109/ACII.2013.47`

[14] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, CoRR. URL `http://arxiv.org/abs/1603.02754`

[15] B. Krawczyk, Learning from imbalanced data: open challenges and future directions, Progress in Artificial Intelligence 5 (4) (2016) 221–232. doi:10.1007/s13748-016-0094-0.

[16] S. Wang, L. L. Minku, X. Yao, Resampling-based ensemble methods for online class imbalance learning, IEEE Transactions on Knowledge and Data Engineering 27 (5) (2015) 1356–1368.

[17] M. Tan, L. Tan, S. Dara, C. Mayeux, Online defect prediction for imbalanced data, in: Proc. of the 37th Intl. Conf. on Software Engineering - Vol. 2, ICSE '15, IEEE Press, Piscataway, NJ, USA, 2015, pp. 99–108.

[18] Y. Levy, M. Ramim, [chais] a study of online exams procrastination using data analytics techniques, Interdisciplinary Journal of E-Learning and Learning Objects 8 (1) (2012) 97–113.

[19] R. Gafni, N. Geri, Time management: Procrastination tendency in individual and collaborative tasks, Interdisciplinary Journal of Information, Knowledge, and Management 5 (1) (2010) 15–125.

[20] C. J. Konig, M. Kleinmann, Deadline rush: A time management phenomenon and its mathematical description relationships between critical thinking and attitudes toward women's roles in society, The Journal of psychology 139 (1) (2005) 33–45.

[21] B. A. Fernie, A.-M. McKenzie, A. V. Nikčević, G. Caselli, M. M. Spada, The contribution of metacognitions and attentional control to decisional procrastination, Journal of Rational-Emotive & Cognitive-Behavior Therapy 34 (1) (2016) 1–13.

[22] A. Stanescu, D. Caragea, Semi-supervised self-training approaches for imbalanced splice site datasets, in: Proc. of The Sixth Intl. Conf. on Bioinformatics and Computational Biology, BICoB 2014, 2014, pp. 131–136.

[23] J. Kuzilek, M. Hlosta, Z. Zdrahal, A. Wolff, Open university learning analytics dataset, Sci. Data (2017) 1–5[in print]. doi:10.1038/sdata.2017.171.

[24] V. López, A. Fernández, S. García, V. Palade, F. Herrera, An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics, Information Sciences 250 (2013) 113–141.

[25] F. Pedregosa, G. Varoquaux, e. Gramfort, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[26] G. Lemaître, F. Nogueira, C. K. Aridas, Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning, Journal of Machine Learning Research (17) (2017) 1–5. URL `http://jmlr.org/papers/v18/16-365`

[27] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, Smote: Synthetic minority over-sampling technique, J. Artif. Int. Res. (1) (2002) 321–357.

[28] G. Batista, R. C. Prati, M. C. Monard, A study of the behavior of several methods for balancing machine learning training data, SIGKDD Explor. Newsl. (1) (2004) 20–29. doi:10.1145/1007730.1007735.

[29] G. Batista, A. L. Bazzan, M. C. Monard, Balancing training data for automated annotation of keywords: a case study., in: WOB, 2003, pp. 10–18.

[30] A. M. Kazerouni, S. H. Edwards, C. A. Shaffer, Quantifying incremental development practices and their relationship to procrastination, in: Proc. of the 2017 ACM Conf. on Intl. Computing Education Research, ICER '17, ACM, New York, NY, USA, 2017, pp. 191–199.

[31] J.-E. Kim, D. A. Nembhard, J. H. Kim, The effects of group size and task complexity on deadline reactivity, Intl. Journal of Industrial Ergonomics 56 (2016) 106–114.