Predicting Assignment Submissions in a Multi-class Classification Problem

Bogdan Drăgulescu [1], Marian Bucos[1], Radu Vasiu[1]

[1]*Politehnica University Timisoara, Piata Victoriei No 2, Timișoara, Romania*

***Abstract* – Predicting student failure is an important task that can empower educators to counteract the factors that affect student performance. In this paper, a part of the bigger problem of predicting student failure is addressed: predicting the students that do not complete their assignment tasks. For solving this problem, real data collected by our university's educational platform was used. Because the problem consisted of predicting one of three possible classes (multi-class classification), the appropriate algorithms and methods were selected. Several experiments were carried out to find the best approach for this prediction problem and the used data set. An approach of time segmentation is proposed in order to facilitate the prediction from early on. Methods that address the problems of high dimensionality and imbalanced data were also evaluated. The outcome of each approach is shown and compared in order to select the best performing classification algorithm for the problem at hand.**

***Keywords* – Multi-class classification, machine learning, e-learning, learning analytics, Moodle.**

1. Introduction

Learning Management Systems (LMS) empower educators to distribute information to students, produce course materials, evaluate learners' progress through assignments and tests, enable collaborative learning with communication tools, etc. [1]. These systems accumulate high volumes of data and could create a gold mine of educational data [2]. Educational Data Mining (EDM) emerges as a paradigm to design models, tasks, methods, and algorithms for exploring this gold mine and enhance the quality and efficiency of the educational process [3].

One problem that becomes very important for educational professional to better understand and counteract is detecting student failure. This is a difficult problem to resolve due to the large amount of factors that can influence student failure [4].

In our educational platform, used in distance and blended learning programs, we observed a high number of assignments with a low submissions percentage. By failing to submit a graded assignment, the student diminishes his chances to graduate the course. Predicting students that do not complete their assignments on time will permit the teacher to intervene and possibly improve the course success rate. Therefore, predicting assignment submissions can be used as a component of detecting student failure.

This study proposes to predict if students submit their assignments on time, by using Data Mining (DM) methods. In fact, we want to predict if a student will complete the assignment on time, with some delay, or not at all. This is a multi-class classification problem in DM and Machine Learning (ML).

The remainder of this paper is organized as follows. Section 2 shows some relevant concepts and findings from the field of EDM. Section 3 highlights the aim and approach employed in the study. Section 4 describes the employed data set, the feature extraction process, the classification algorithms used, and the results of tested prediction models. In section 5 there is a discussion of this work, and finally section 6 summarizes the main conclusions.

2. Background

According to Witten and Frank [5], Data Mining is the process of discovering patterns in data. By analyzing data already present in databases, this process extracts useful and comprehensible knowledge in an automatic or semiautomatic way. DM methods and techniques were borrowed to offer insight into the educational process by two research communities: Educational Data Mining and Learning Analytics (LA). One key distinction between the two research communities is that EDM has a considerably greater focus on automated discovery, as oppose to LA which focus on human judgment for the final decision [6]. Thereby, EDM models are more often used in automated adaptation tasks conducted by a computer system, versus LA models that are designed to inform educators and learners. The techniques used from DM are the same for both EDM and LA models; the only distinction is the integration of such methods in the education support software.

One popular application of DM in education is the prediction of student's performance [7]. Peña-Ayala

argued that many indicators of performance are worthy to be modeled, for example [3]: evaluation, achievement, efficiency, competence, elapsed time, correctness, deficiencies, etc. According to the author, the goal of student performance modeling is to estimate the capability of the learner to complete a given task. The value of the indicators for task completion can be numerical (grades) or categorical (a label). In the estimation of numerical variables, regression analysis is used. It finds the relationship between a dependent variable and one or more independent variables [8]. For categorical variables, classification procedure is used. In this task, the items are placed into groups based on information regarding characteristics inherited in the items and a previously labeled training set [9].

An empirical study that compares five classification methods for predicting course failure or success is presented in a paper from 2006 [10]. The authors used a very small data set consisted of only exercise points for a specific course. All tested methods achieved about the same accuracy, 80% in the middle of the course. Another comparison of data mining algorithms used to classify students is based on usage data collected in Moodle [11]. In this approach, the data set consists of data collected from multiple courses. The accuracy is not as high as the previous paper; the evaluated algorithm that performs best does not exceed 70% of correctly classified results.

Lopez used classification via clustering to predict students' final mark on the basis of their participation in forums [12]. The authors argued that the student's participation in the course forum is a good predictor of the final marks for the course, and the proposed classification via clustering approach obtained similar accuracy to traditional classification algorithms.

Another paper describes a study that shows that success or failure of a learner can be predicted using information about learner interactions with course materials [13]. The authors argued that an LMS that could use its own log files to identify learners who appeared to be struggling with a course would be extremely valuable to educators.

Typically, in a course a learner needs to complete multiple tasks. The completion of these intermediary tasks can affect the learner final mark. Therefore, it is also important to predict the completion of educational tasks. One approach is to predict completion of a reading task by formulating tutorial dialog classification as a sequence classification problem [14]. A different approach is to predict how much time a student needs to solve a problem, as opposed to predict the completion of that given problem, and recommend students problems of suitable difficulty [15].

In this context, we address in this paper the problem of predicting student completion of an assignment in a given course. This approach has the advantage of predicting an academic failure earlier and allowing the educator to take the necessary corrections. With this approach, we aim to provide the best performing classification algorithm that can be deployed in our educational platform that is based on Moodle.

3. Method

In this section, the steps followed to predict students' assignment submissions are described. According to Romero, the e-learning data mining process consists of the same four steps in the general data mining process: data collection, preprocessing, apply data mining, and interpretation of the results [1].

For the first step, data collection, we need to gather all available information on students and prior completed assignments. Our educational platform is based on Moodle. On each start of an academic year, the platform is updated to the new stable version of the LMS. Therefore, the data is collected in different versions of the database. Therefore, we need to identify the set of factors that can be used in the data mining process, collect the data from the different database versions, and integrate it into a unified data set.

Next, the data is preprocessed to transform it into an appropriate format to be mined. The general data preprocessing tasks are data cleaning, user identification, session identification, path completion, transaction identification, data transformation and enrichment, data integration, and data reduction [16]. Other techniques such as re-balancing the data in the case of an imbalanced data set, or attributes selection in the case of a high dimensionality data set, have to be tested if they can help to produce better results.

To apply data mining on the constructed data set we first define the problem and constraints. The goal is to predict if a student will submit the assignment on time, will exceed the deadline, or will not complete the task. This can be solved as a multi-class classification problem in data mining, with three class to predict. Another constraint is that we want to identify the best performing algorithm for this particular problem that can be used in our educational problem. Prior developed learning analytic scenarios were integrated using Python for processing power. Therefore, we plan to evaluate the best performing DM algorithm for our multi-class classification problem available in this specific programming language.

Finally, in the interpretation step, the obtained models are analyzed to detect student failure to complete an assignment on time. We discuss the means of deployment of the results to the instructor, who may use the information discovered to make decisions to improve the assignment completion rate.

4. Prediction model

This section is organized in four subsections. In 4.1, the data sources and the process of constructing the data set used in the experiments are described. In the next subsection, the preprocessing tasks used to better refine the data set are presented. Subsection 4.3 presents the selected algorithm, methods and metrics used in the experiment. Finally, in 4.4 the results of the experiments are presented.

Data Set

As the data source for our experiment, we used our university's educational platform based on Moodle. This LMS keeps detailed logs of all activities a learner performs [17]. It logs the clicks that a student makes for navigational purposes, the interaction with the learning resources a teacher provides in a course, or the time when a student completes an assignment. Moodle stores the logs in the same relational database used for the platform. This facilitates the data collection and identification. By identification, we refer to the process of correlating the collected data with the corresponding user, course, or assignment.

The current platform is used from 2010, at first for the distance education program, and then extended for master and some bachelor programs. We have used data from four full academic years (2010-2011, 2011-2012, 2012-2013, 2013-2014) and first semester of the current one (2014-2015). At the current moment the process of preparing the platform for a new academic year involves creating an archive version for administrative reasons, upgrading to the latest Moodle version and in-house developed modules, resetting the courses, and enrolling the students. This implies that the data storage structure modifies. To extract the needed data we needed to customize the process for each instance of the educational platform. In addition, in order to merge the resulted data sets, all selected attributes needed to be collected in the same manner in all instances of the platform.

In our problem the output attribute or class to be predicted is whether a student will submit a particular assignment on time, which has three possible values: *On time* (students that will complete the assignment on time), *Over time* (students that will submit their assignment with a time delay), and *No* (students that will not complete that assignment). The submission status is stored in the database tables designated for the assignment module in Moodle. This module suffered a major redesign in the version used from 2013 forward. The tables' structure and module functionality changed, but the information used for this prediction problem remained the same.

To build the attributes used for predicting the class we consider particularities of the course in which the assignment was instantiated, assignment particularities, and the interaction of the specified user with the course. Based on his preferences, the teacher can employ a different module to implement the course. This hinders the possibility of using entries from multiple courses in the same prediction problem. Therefore, the possible modules a teacher can use in a course were grouped based on their role in the educational process: resources for all available course materials, activities for all organized interactions in a course, and assignment for all tasks. From the logs were extracted the number of interactions of the given student with aforementioned groups and the assignment's course in general. For the assignment, the following attributes were selected: number of days students have to complete the assignment, if overtime is permitted, and the number of students that completed the assignment prior to the current student. As like the assignment module, the log table suffered major changes from 2014 forward. This will not affect the data set because the changes were targeted to collect the information in a more precise manner, and the data that we used in our experiment was still collected.

In order to predict the class more efficiently, we devised a time segmentation approach. This means that the time interval from the date the assignment is available to the date the assignment is due is divided into segments. With this approach, we can predict in a more reliable means the outcome at the beginning of an assignment or in a specific time segment. For this to work all the time sensitive aforementioned attributes were collected for each segment. To predict a class in a particular time segment, the current and the prior segments attributes will be used in the prediction. In our particular classification problem, we chose to use five segments.

Following the above constrains, we designed an extraction script written in Python for each academic year. By using an automated process to extract the selected attributes ensures that further corrections can easily be made. In addition, human error in extracting the attributes can be avoided. The resulted data sets were merged in a unified file after the preprocessing steps described in the next subsection.

Preprocessing

A very important task in this experiment was data preprocessing, which can affect the results due to reliability and quality of available information. The prevision obtained data files were integrated into a single consistent data set. Some specific data preprocessing tasks were applied to prepare the previously described data for the classification task of the experiment.

First, all the entries in which the student had no activity in the course were eliminated from the data set. A student with no activity in the educational platform will not complete the assignment, and probably the student is dropping-out. No prediction is needed in this case the teacher can infer this information from the general student activity report in Moodle. Other discarded entries were those produce for assignments that had a zero number of days to complete the task. Those assignments were used by teachers to collect the materials from the students in a face-to-face activity and no prediction is needed. Because the data was extracted from the Moodle database there were no missing attributes for any entry, and thus no extra cleaning task was needed.

Based on the design of the course, this can contain a variable number of resources, activities or assignments. This influence the number of interaction a student has on each of the above groups. In order to mix entries from different courses in the classification process, we needed to ensure that the attributes are compatible. This was achieved by normalizing the attributes that contain the interaction a student has with a particular group of modules. The normalized attributes were obtained by dividing the interactions on a specific group to the number of instances in that group for that particular course.

After the above steps, all the information was integrated into a single data set and it was saved in coma separated variables format. Therefore, after preprocessing we have a data set of 39 attributes and 33772 entries. From the list of attributes, seven are repeated for each of the five segments with values specific for that particular time interval (the last 7 from table 1). To predict a class in a particular time segment, the current and the prior segments attributes will be used in the prediction. Therefore, the number of attributes used in the classification will vary on the time segment in which the prediction is made: 11 for segment 1, 18 for segment 2, 25 for segment 3, 32 for segment 4, and 39 for segment 5.

One problem that needed to be addressed is high dimensionality of the data set; meaning the number of attributes becomes large. A high dimensional data set can diminish the performance of the prediction model and require more processing power to compute [18]. This can be the case only for the last two time segments where the number of attributes tends to be greater. A simple feature reduction is to use only the attributes specific for that particular time segment.

In this case, a fixed number of 11 attributes will be used for predicting the class for each segment. On the other hand, in our experiment due to the high number of entries in the data set compared to the number of attributes, the performance of the prediction was not diminished by using all attributes.

The class distribution in the data set is imbalanced as follows: 38% assignment submission on time, 7% assignment submission with a time delay, and 55% assignment not submitted. The problem with imbalanced data sets is that classification algorithms tend to overlook minority classes and focus the attention on the most frequent ones [19]. This will result in low performance on predicting the students that will submit their assignments with a time delay.

Table 1. Attributes used for each assignment status

Name	Description
n_assign_sub_no	Number of submissions for that assignment divided by the number of students in that course
course_students	Number of students that need to complete the assignment
assign_days_to_complete	Number of days the students had to complete the assignments
assignment_allow_overtime	If assignment allows overtime submission
course_resources_s1	Number of resources available in that time segment
course_activities_s1	Number of activities available in that time segment
course_assignments_s1	Number of assignments available in that time segment
course_hits_s1	Number of that particular student interactions with the assignment course in that time segment
n_resource_hits_s1	Number of that particular student interactions with course resources available in that time segment
n_activity_hit_s1	Number of that particular student interactions with course activities available in that time segment
n_assign_hits_s1	Number of that particular student interactions with course assignments available in that time segment

In the experiment, we tested the performance of the classifier when counteracting the imbalances in the data set by assigning weights to classes. This is

an algorithm-level approach, and the correction is done when running the classification process.

Multi-class classification algorithms

The classification problem is tackled in educational data mining with a wide range of algorithms: decision trees, naïve Bayes, instance-based learning, neural networks, logistic regression, random forest, and support vector machines [7]. In this paper, the chosen classification algorithms were selected for evaluation based on two conditions. First, the algorithm had to handle our multi-class classification problem. Second, the final prediction model had to be easily integrated into our educational platform. We already used the Python programing language for some learning analytics scenarios, and it was only logical also to use it for this particular problem, considering the staff programing skills and that the deployment infrastructure was built. For machine learning algorithms, the Scikit-learn Python module was used. It integrates a wide range of state-of-the-art machine learning algorithms for classification problems using a consistent interface, thus enabling easy comparison of methods [20].

Regarding the first condition, Scikit-learn can handle multi-class problems by using one of the three following strategies: using an algorithm that handles multi-class inherently; one versus all, where each class is fitted against all the other classes; one versus one, where one classifier is constructed per pair of classes, and for prediction the class which received the most votes is selected. In this paper, all the 6 inherently multi-class algorithms available in Scikit-learn were evaluated, and one algorithm that can handle the one versus all strategy.

The following inherently multi-class algorithms were used: GaussianNB, a naïve Bayes method that implements the gaussian version for classification [21]; LDA, linear discriminant analysis, a classical classifier, using a linear decision boundary generated by fitting class conditional densities to the data [22]; CART, which is a decision tree type algorithm similar to C4.5 (Scikit-learn implements an optimized version of CART) [23]; RandomForest, an ensemble method that uses the perturb-and-combine technique for decision trees [24]; KNeighborsClassifier, which is a nearest-neighbor classifier that implements learning based on the k nearest neighbors [22]; logit, or logistic regression, is a linear model for classification using a logistic function [25]. For one versus all strategy, the SVC method was used. SVC is an implementation of support vector machine for classification based on libsvm [26].

In order to evaluate the classification algorithms, appropriate performance measures need to be obtained. The confusion matrix is used in classification problems to display the difference between the predicted and actual class. It contains the numbers of true positive (TP), true negative (TN), false negative (FN), and false positive (FP). The values from the confusion matrix are used to define a wide range of measures. In this paper, the following measures were employed for evaluating the performance of the classifiers:

- Accuracy is the overall rate of correctly predicted classes by the classification algorithm, and is calculated as

$$ACC = TP + TN/TP + TN + FP + FN. \quad (1)$$

- Precision, also known as positive predictive value (PPV), is the proportion of actual positives from the predicted positives, and is calculated as

$$Precission = TP/(TP + FP) \ . \quad (2)$$

- Recall, also known as true positive rate (TPR) or sensitivity, is the proportion of actual positives that are also predicted as positive, and is calculated as

$$Recall = TP/(TP + FN). \quad (3)$$

- F-score is a measure of the test accuracy and is defined as a weighted average of the corresponding precision and recall

$$F1 = 2 \times \frac{precission \times recall}{precission + recall}. \quad (4)$$

- ROC AUC is the area under the Receiver Operating Characteristic (ROC) curve. ROC curves are created by plotting true positive rate against the false positive rate at different decision thresholds. AUC is equivalent to the probability that a classifier will rank a randomly chosen positive instance, higher than a randomly chosen negative instance [27].
- ACUPR is the area under the precision recall (PR) curve. PR curves are created in a similar way to ROC curves, by plotting precision versus recall. ACUPR is used as a general measure of performance, particularly useful in imbalanced or skewed data sets [28].
- Kappa, also known as Cohen's kappa coefficient, is thought to be a more robust measure since it takes into account the chance [29]. Kappa is a metric that compares the observed accuracy (OA) with the expected accuracy (EA) (if the prediction model follows random chance). It is calculated as

$$K = (OA - EA)/(1 - EA). \quad (5)$$

Algorithms Comparison

We carried out several experiments to test and compare the selected algorithm available in Scikit-learn for our multi-class classification problem. To improve the prediction of the classifier DM approaches were tested for high-dimensional and imbalanced data.

In the first experiment, all the classification algorithms from Scikit-learn that have native support for multi-class were used. This are: DT (CART), RandomForest, KNeighborsClassifier (NN), GaussianNB, logit, and LDA. The 33772 entries were randomly split into the training set and evaluation set (two thirds for training and one third for evaluation). The division into training and evaluation sets remained constant in all experiments. The training set was used in tenfold cross-validation from which the first accuracy metric was obtained. The rest of the metrics were obtained from the evaluation set. The results from this experiment are shown in Table 2.

Table 2. Classification results for each time segment

Segment 1								
Algorithm	CV(accuracy)	Kappa	F1	Acc	Pre	Rec	AUC	AUCPR
DT (CART)	0.70 (+/- 0.02)	0.42	0.68	0.7	0.69	0.7	0.87	0.78
RandomForest	0.75 (+/- 0.01)	0.52	0.73	0.75	0.74	0.75	0.9	0.83
NN	0.73 (+/- 0.01)	0.44	0.7	0.72	0.7	0.72	0.89	0.81
GaussianNB	0.43 (+/- 0.05)	0.12	0.46	0.43	0.58	0.43	0.65	0.49
logit	0.59 (+/- 0.01)	0.13	0.52	0.59	0.56	0.59	0.79	0.61
LDA	0.59 (+/- 0.01)	0.12	0.51	0.59	0.56	0.59	0.79	0.61
SVC	-	-	0.52	-	0.64	0.55	0.8	0.68
Segment 2								
Algorithm	CV(accuracy)	Kappa	F1	Acc	Pre	Rec	AUC	AUCPR
DT (CART)	0.70 (+/- 0.01)	0.43	0.69	0.71	0.69	0.71	0.87	0.77
RandomForest	0.76 (+/- 0.01)	0.54	0.76	0.76	0.75	0.76	0.91	0.85
NN	0.75 (+/- 0.02)	0.5	0.73	0.74	0.73	0.74	0.89	0.82
GaussianNB	0.38 (+/- 0.02)	0.15	0.45	0.39	0.64	0.39	0.63	0.49
logit	0.63 (+/- 0.01)	0.24	0.59	0.64	0.6	0.64	0.82	0.68
LDA	0.62 (+/- 0.01)	0.21	0.57	0.62	0.6	0.62	0.82	0.68
SVC	-	-	0.57	-	0.69	0.59	0.82	0.71
Segment 3								
Algorithm	CV(accuracy)	Kappa	F1	Acc	Pre	Rec	AUC	AUCPR
DT (CART)	0.71 (+/- 0.01)	0.44	0.69	0.71	0.69	0.71	0.87	0.77
RandomForest	0.77 (+/- 0.01)	0.55	0.75	0.77	0.76	0.76	0.91	0.85
NN	0.76 (+/- 0.02)	0.52	0.74	0.76	0.75	0.74	0.9	0.84
GaussianNB	0.39 (+/- 0.03)	0.15	0.45	0.38	0.66	0.39	0.63	0.5
logit	0.65 (+/- 0.01)	0.29	0.61	0.66	0.62	0.64	0.83	0.7
LDA	0.64 (+/- 0.01)	0.25	0.59	0.64	0.62	0.62	0.83	0.69
SVC	-	-	0.58	-	0.69	0.59	0.83	0.73
Segment 4								
Algorithm	CV(accuracy)	Kappa	F1	Acc	Pre	Rec	AUC	AUCPR
DT (CART)	0.72 (+/- 0.01)	0.47	0.7	0.73	0.69	0.73	0.88	0.8
RandomForest	0.78 (+/- 0.02)	0.58	0.77	0.78	0.77	0.78	0.92	0.86
NN	0.76 (+/- 0.01)	0.54	0.74	0.76	0.75	0.76	0.91	0.84
GaussianNB	0.37 (+/- 0.02)	0.16	0.45	0.37	0.59	0.38	0.64	0.51
logit	0.68 (+/- 0.02)	0.35	0.64	0.68	0.66	0.68	0.85	0.74
LDA	0.66 (+/- 0.02)	0.31	0.63	0.66	0.66	0.66	0.84	0.72
SVC	-	-	0.6	-	0.7	0.63	0.84	0.74

Segment 5								
Algorithm	CV(accuracy)	Kappa	F1	Acc	Pre	Rec	AUC	AUCPR
DT (CART)	0.83 (+/- 0.01)	0.68	0.81	0.84	0.8	0.84	0.94	0.89
RandomForest	0.86 (+/- 0.01)	0.73	0.85	0.86	0.85	0.86	0.96	0.93
NN	0.81 (+/- 0.01)	0.66	0.81	0.82	0.81	0.82	0.94	0.89
GaussianNB	0.42 (+/- 0.03)	0.22	0.5	0.42	0.74	0.42	0.69	0.56
logit	0.79 (+/- 0.01)	0.59	0.77	0.79	0.77	0.79	0.92	0.86
LDA	0.77 (+/- 0.01)	0.55	0.75	0.77	0.75	0.77	0.91	0.85
SVC	-	-	0.73	-	0.8	0.71	0.91	0.84

This table shows the metrics for each classification algorithm: CV (accuracy) represents the accuracy from tenfold cross-validation, Kappa is the Cohen's kappa coefficient, F1 is the f-score, Acc is the accuracy computed from the evaluation set, Pre represents the precision, Rec is the recall, AUC represents the area under the ROC curve, and AUCPR is the area under the precision-recall curve. The ROC AUC and AUCPR are computed by employing a micro-average technique over the corresponding curves for each class [30]. Furthermore, the classification was run for each of the five segments, and the results are presented accordingly. From the 39 attributes available in the data set, 35 are specific to a time segment (7 for each segment). For the first segment were used 11 attributes. With each segment increment by one, the corresponded set of 7 attributes was added to the list used in that classification.

Therefore, the number of attributes used in the classification will vary on the time segment in which the prediction is made: 11 for segment 1, 18 for segment 2, 25 for segment 3, 32 for segment 4, and all the 39 for segment 5.

In addition, Table 2 presents the results from the second experiment, where we used a one versus rest approach to use a classification algorithm that does not normally support multi-class problems. The results are the last entries for each segment. The tested algorithm is SVC, which is an implementation of support vector machine algorithm for classification problems. In order to run the experiment, first the classes were binarized. In this experiment, the same procedure in splitting the data in training and evaluation sets was used. We did not used tenfold cross-validation and therefore the first metric in the table is empty. In addition, we chose not to compute kappa and accuracy for this scenario. The main reasons are that some major modifications in the libraries were needed and from the computed metrics resulted that other algorithms outperforms SVC for this classification problem.

It can be seen in Table 2 that the accuracy computed in tenfold cross-validation and the accuracy computed for the evaluation set, are equal or in the variation range for cross-validation. This is due to the large data set used in the experiment. This also concludes that we can use the metrics computed with the evaluation set to be sufficient for algorithm comparison. It can be observed that the best performing algorithm for every metric and every segment is RandomForest. NN and CART were second; the first performed better on the lower time segments, versus the second that performed better on the last time segment. A pattern can be observed in the evolution of metrics from first to last segment. To better visualize this patter the data for kappa and AUC is summarized in Table 3. The performance of all classifiers slightly improves from one time segment to the next, except for the last segment where the improvement is significant.

Table 3. Classification results, Kappa comparison by segment

	Kappa					AUC				
Algorithm	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
DT (CART)	0.42	0.42	0.44	0.47	0.68	0.87	0.87	0.87	0.88	0.94
RandomForest	0.52	0.52	0.55	0.58	0.73	0.9	0.91	0.91	0.92	0.96
NN	0.44	0.44	0.52	0.54	0.66	0.89	0.89	0.9	0.91	0.94
GaussianNB	0.12	0.12	0.15	0.16	0.22	0.65	0.63	0.63	0.64	0.69
logit	0.13	0.13	0.29	0.35	0.59	0.79	0.82	0.83	0.85	0.92
LDA	0.12	0.12	0.25	0.31	0.55	0.79	0.82	0.83	0.84	0.91
SVC	-	-	-	-	-	0.8	0.82	0.83	0.84	0.91

Table 4. Classification comparison per class on time segment 5

	Precision				Recall				AUC			
Algorithm	C1	C2	C3	AVG	C1	C2	C3	AVG	C1	C2	C3	AVG
DT (CART)	0.85	0.82	0.3	0.8	0.92	0.85	0.03	0.84	0.93	0.93	0.78	0.94
RandomForest	0.88	0.83	0.7	0.85	0.92	0.88	0.24	0.86	0.95	0.95	0.88	0.96
NN	0.84	0.81	0.58	0.81	0.9	0.79	0.25	0.82	0.92	0.92	0.88	0.94
GaussianNB	0.83	0.71	0.1	0.74	0.43	0.35	0.8	0.42	0.81	0.77	0.71	0.69
logit	0.8	0.78	0.44	0.77	0.91	0.75	0.03	0.79	0.91	0.9	0.82	0.92
LDA	0.79	0.76	0.32	0.75	0.9	0.7	0.06	0.77	0.89	0.88	0.81	0.91
SVC	0.81	0.88	0.33	0.8	0.91	0.54	0.01	0.71	0.9	0.9	0.77	0.91

Table 5. Classification class comparison on time segment 5 using data balancing

	Precision				Recall				AUC			
Algorithm	C1	C2	C3	AVG	C1	C2	C3	AVG	C1	C2	C3	AVG
DT (CART)	0.92	0.83	0.24	0.84	0.76	0.78	0.68	0.76	0.92	0.91	0.82	0.9
RandomForest	0.91	0.84	0.4	0.85	0.85	0.87	0.55	0.84	0.94	0.95	0.89	0.95

This is probably due to the study behavior of the students: the work is done very close to the deadline and hence more data is collected in that time frame. The prediction of submissions in the first segments remains a useful tool for the teacher if the prediction has a relatively high precision, and thus the predicted class is correct.

We also evaluated the algorithms by using only the attributes specific for each time segment. The results did not vary in case of classification performance. In our case, the computational performance did not matter, because after the selected classifier is trained the model is saved to disk and used for the whole semester. Computing the model once per semester will not hinder the overall performance of the system.

In another experiment, we aimed at comparing the algorithms for each of the possible classes. The data set was split using the same ratio in training and evaluation sets. The results are presented in Table 4 only for segment 5. The classes are represented as C1 for *No* (students that will not complete that assignment), C2 for *On time* (students that will complete the assignment on time), and C3 for *Over time* (students that will submit their assignment with a time delay). It can be observed that the C3 is very poorly predicted. Still the best performing algorithm is RandomForest. GaussianNB has high recall but with very low precision, this means that a high number of entries in that class are correctly predicted, but also a high number of entries from different classes are wrongly predicted as C3. The drop in performance for C3 is due to the imbalanced distribution of entries in the data set: C1 has 55%, C2 has 38%, and C3 has only 7% of the entries.

To counteract this problem, in the final experiment we test an algorithm-level approach to correct it in the classification process. We used the CART and RandomForest algorithms. The Scikit-learn implementations of these algorithms have a method for assigning weights to classes in an automatic manner, proportional to the class distribution in the data set. From the results presented in Table 5, it can be observed that the recall for C3 has greater improved but at the cost of the precision. In addition, the general precision for the classifiers had improved, but the recall and AUC decreased. The main concern is with the drop in recall for the C1 class. This is the main class we want to predict because it provides the teacher with alerts regarding the students that do not complete their assignments. The lower recall means that fewer students that do not submit their assignments on time are correctly predicted. Therefore, in our case the best solution was to use RandomForest with the imbalanced data set. This decision needs to be reevaluated if the balance in the distribution of classes changes.

5. Discussions

The problem of predicting student academic performance is of great interest to many researchers in the fields of educational data mining and learning analytics. Some previous approaches involve: predicting student final mark [31], [32], prediction of student failure [4], [33], [34], study the relationship between interactions and academic performance [35], or predicting the completion of a task [14].

In this paper, we have shown that classification algorithms can be used successfully to predict if a student submits his assignment on time. This can be viewed as a more specific problem in the general

context of academic performance. With the correct prediction of failure to submit the assignment, a teacher can intervene to counteract the problem.

In addition, we have shown that the data collected by our university platform, based on Moodle, is sufficient to build the features used in the prediction. We propose a time segmentation approach to enable the algorithm to predict the student submission status from early on. Because of this, the data set contains attributes that are used for every segment and attributes that are specific to a particular one. In our experiments, we had chosen to use five time segments. This is not a restriction; another number of segments can be used depending on the implementation constraints. A higher number will ensure a more fine-grained prediction because the time segments will be smaller and, therefore, closer to the time the prediction is made.

The algorithm needed to predict one of the following classes for a student regarding his submission status in a particular assignment: *On time* (the submission is on time), *Over time* (the student submits his assignment with a time delay), or *No* (if no submission is made). This is a multi-class classification problem. Another constraint is that we wished to use Python in implementing the prediction algorithms, because it was already used in other learning analytics scenarios and we wanted to integrate easily the model into the educational platform. Because of this, the Scikit-learn Python package was selected. To choose the best performing algorithm for this classification problem, we run experiments to compare all the native multi-class classification algorithms available in Scikit-learn and one algorithm in a one versus all approach.

In the first experiment, we had used all the attributes available in the data set: for the first segment 11 attributes and with each increment we added the time depended attributes for that segment. The best performing algorithm for our data set was RandomForest, with a better than chance performance of 0.73 (the Kappa metric). In addition, RandomForest outperformed the other evaluated algorithm on all metrics. The case in which the data set contained only 11 attributes for each segment (only the attributes for that specific time segment) was also evaluated. The prediction performance of the classifiers did not change. This is due to the large data set used to train the classifier, over 37000 entries. In our case, the computational performance did not matter because the model is built only once per semester and then saved to disk for reuse. If the training set is smaller or the model needs to be trained very often, feature selection techniques need to be used.

The prediction performance varies from segment to segment. The lowest values are obtained for the first segment and are gradually increased until the last but one. For the final one, the increase is significant. From this, we can conclude that students study patterns involve completing the task in the last time segment and, therefore, more data is collected by the platform for this segment. Even if the performance for the first segment is lower, the prediction still has value in the teaching process. By presenting to the educator predictions accompanied by the model confidence in that prediction, empowers the educator to take action only in the cases where the confidence is high.

By evaluating the performance of the algorithm on each class we observed a very low prediction performance of the *Over time* class. This is due to the imbalance in the data set, only 7% of the entries are in this class. To address this problem, we took advantage of the algorithm-level approach implemented in RandomForest and CART algorithms for imbalanced data sets. The prediction performance for *Over time* class had improved. Nevertheless, the performance for the No class had dropped. This class interested us more, in order to inform the teacher of the possible submission failures. Therefore, on our current data set the imbalance does not negatively influence the prediction model. If the class distribution change and the No class becomes under-sampled, then this approach will improve the desired prediction model. Therefore, the decision to counteract the imbalance in the data set depends on the distribution of classes and on the main goals of the prediction.

Regarding the limitations of this study, we have to mention that the data set is specific to the pedagogical methods employed in our university. With different methods, more data may be collected by the LMS. Therefore, the conclusion that RandomForest is the best algorithm for this particular classification problem cannot be generalized. Second, other LMSs can collect more or fewer data, and, therefore, different attributes need to be selected that can influence the performance of the prediction. The rest of the conclusions and the process of running the experiment remain valid.

For future work, we have to try to improve the performance of the algorithm for all segments. Some actions that we need to test their impact on performance are adding more features by considering the history of the users throughout different courses, grouping the courses based on the pedagogical methods used by the educator, or building a custom ensemble method with better performance that RandomForest. We also plan to integrate the results into a module for our university's educational platform that can run the analysis in an automatic way and display the predictions to the educator.

6. Conclusion

Predicting student failure is a difficult task that was addressed by many researchers. In this paper, we set off to solve a problem that is part of this larger task: predicting the accomplishment of an assignment by students. Without completing their graded assignments, students are destined to fail. In order to build a model that informs the educators regarding students that are at risk of not completing their assignments, several steps were taken. First, the data set used to train the prediction model was built from the data collected by our university's educational platform. We have shown different algorithms and methods suitable for multi-class classification problems, available in Sickie-learn. We carried out several experiments to find the best approach for our data set. Furthermore, we evaluated some approaches, such features reduction and data balancing, to improve the prediction performance. For our data set, these approaches do not improve the performance, but we discuss the cases in which they are binding. Finally, as the next step in our research, we aim at improving the performance of the prediction and implementing the model in a tool for the educational platform.

Acknowledgements

This work was partially supported by the strategic grant POSDRU/159/1.5/S/137070 (2014) of the Ministry of National Education, Romania, co-financed by the European Social Fund – Investing in People, within the Sectoral Operational Programme Human Resources Development 2007-2013.

References

[1]. Romero, C., Ventura, S., & García, E. (2008). *Data mining in course management systems: Moodle case study and tutorial.* Computers & Education, 51(1), 368–384.

[2]. Mostow, J., & Beck, J. (2006). *Some useful tactics to modify, map and mine data from intelligent tutors.* Natural Language Engineering, 12(02), 195–208.

[3]. Peña-Ayala, A. (2014). *Educational data mining: A survey and a data mining-based analysis of recent works.* Expert systems with applications, 41(4), 1432–1462.

[4]. Márquez-Vera, C., Cano, A., Romero, C., & Ventura, S. (2013). *Predicting student failure at school using genetic programming and different data mining approaches with high dimensional and imbalanced data.* Applied intelligence, 38(3), 315–330.

[5]. Witten, I. H., & Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann.

[6]. Siemens, G., & d Baker, R. S. (2012). *Learning analytics and educational data mining: towards communication and collaboration.* In Proceedings of the 2nd international conference on learning analytics and knowledge, ACM, pp. 252–254.

[7]. Romero, C., & Ventura, S. (2010). *Educational data mining: a review of the state of the art.* Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 40(6), 601–618.

[8]. Hand, D. J., Mannila, H., & Smyth, P. (2001). *Principles of data mining.* MIT press.

[9]. Espejo, P. G., Ventura, S., & Herrera, F. (2010). *A survey on the application of genetic programming to classification.* Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 40(2), 121–144.

[10]. Hämäläinen, W., & Vinni, M. (2006). *Comparison of machine learning methods for intelligent tutoring systems*. In Intelligent Tutoring Systems, Springer, pp. 525–534

[11]. Romero, C., Ventura, S., Espejo, P. G., & Hervás, C. (2008). *Data Mining Algorithms to Classify Students.* In EDM 2008, pp. 8–17.

[12]. Romero, C., López, M.-I., Luna, J.-M., & Ventura, S. (2013). *Predicting students' final performance from participation in on-line discussion forums*. Computers & Education, 68, 458–472.

[13]. McCuaig, J., & Baldwin, J. (2012). *Identifying Successful Learners from Interaction Behaviour.*In EDM 2012, pp. 160-163.

[14]. González-Brenes, J. P., Mostow, J., & Duan, W. (2011). *How to Classify Tutorial Dialogue? Comparing Feature Vectors vs. Sequences*. In EDM 2011 ,pp. 169–178.

[15]. Jarušek, P., & Pelánek, R. (2011). *Problem response theory and its application for tutoring.* Educational Data Mining, 374–375.

[16]. Romero, C., & Ventura, S. (2007). *Educational data mining: A survey from 1995 to 2005*. Expert systems with applications, 33(1), 135–146.

[17]. Rice, W. H. (2006). *Moodle: E-learning Course Development : a Complete Guide to Successful Learning Using Moodle.* Packt Publishing.

[18]. Ramaswami, M., & Bhaskaran, R. (2009). *A study on feature selection techniques in educational data mining*. Computing, 1(1), 7-11.

[19]. Gu, Q., Cai, Z., Zhu, L., & Huang, B. (2008). *Data Mining on Imbalanced Data Sets*. In International Conference on Advanced Computer Theory and Engineering, pp. 1020–1024.

[20]. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). *Scikit-learn: Machine learning in Python.* The Journal of Machine Learning Research, 12, 2825–2830.

[21]. Chan, T. F., Golub, G. H., & LeVeque, R. J. (1982). *Updating formulae and a pairwise algorithm for computing sample variances.* In COMPSTAT 1982, pp. 30–41.

[22]. Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J., & Tibshirani, R. (2009). *The elements of statistical learning (Vol. 2)*. Springer.

[23]. Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC press.
[24]. Breiman, L. (2001). *Random forests*. Machine learning, 45(1), 5–32.
[25]. Yu, H.-F., Huang, F.-L., & Lin, C.-J. (2011). *Dual coordinate descent methods for logistic regression and maximum entropy models*. Machine Learning, 85(1-2), 41–75.
[26]. Chang, C.-C., & Lin, C.-J. (2011). *LIBSVM: a library for support vector machines*. ACM Transactions on Intelligent Systems and Technology (TIST), 2(3), 27.
[27]. Fawcett, T. (2006). *An introduction to ROC analysis. Pattern recognition letters*, 27(8), 861–874.
[28]. Boyd, K., Eng, K. H., & Page, C. D. (2013). *Area under the precision-recall curve: Point estimates and confidence intervals*. In Machine Learning and Knowledge Discovery in Databases, pp. 451–466.
[29]. Carletta, J. (1996). *Assessing agreement on classification tasks: the kappa statistic*. Computational linguistics, 22(2), 249–254.
[30]. Yang, Y. (1999). *An evaluation of statistical approaches to text categorization*. Information retrieval, 1(1-2), 69–90.
[31]. Lopez, M. I., Luna, J. M., Romero, C., & Ventura, S. (2012). *Classification via Clustering for Predicting Final Marks Based on Student Participation in Forums*. In EDM 2012, pp. 148-152.
[32]. Romero, C., Espejo, P. G., Zafra, A., Romero, J. R., & Ventura, S. (2013). *Web usage mining for predicting final marks of students that use Moodle courses*. Computer Applications in Engineering Education, 21(1), 135–146.
[33]. Tan, M., & Shao, P. (2015). *Prediction of Student Dropout in E-Learning Program Through the Use of Machine Learning Method*. International Journal of Emerging Technologies in Learning (iJET), 10(1), pp. 11–17.
[34]. Thakur, G. S., Olama, M. M., McNair, A. W., Sukumar, S. R., & Studham, S. (2014). *Towards Adaptive Educational Assessments: Predicting Student Performance using Temporal Stability and Data Analytics in Learning Management Systems*. Oak Ridge National Laboratory (ORNL).
[35]. Agudo-Peregrina, A. F., Hernandez-Garcia, A., & Iglesias-Pradas, S. (2012). *Predicting academic performance with learning analytics in virtual learning environments: A comparative study of three interaction classifications*. In Computers in Education (SIIE), 2012 International Symposium on (pp. 1–6).