

Predictive Process Monitoring: An Implementation and Comparison of Student Performance Prediction

Lisa Arnold^{1,*}, Marius Breitmayer¹ and Manfred Reichert¹

¹Institute of Databases and Information Systems, Ulm University, Germany

Abstract

Predictive monitoring can support students during the semester by motivating them if they are not performing well enough in a lecture or exercise. Furthermore, supervisor can create additional exercise sheets or can adapt their lectures and exercises to the needs (i.e. knowledge gaps) of the students. To realise this, three different regression algorithms (i.e. Neuronal Networks, Decision Trees, and Random Forest) are implemented to continuously predict further exercise points and the final grade during a semester. These algorithms are trained and tested based on student points and grades from previous semesters. A total of 17,136 predictions were determined, analysed, and compared. In several exercise sheets, all algorithms achieve between 91% and 96% correct predictions with a variance of 10% (i.e. up to 2.5 points at the exercise sheet). With 15% variance, up to 98.5% of corrected predictions are possible. The prediction of grades with a variance of 0.3 (i.e. one grade level) with the Decision Tree and the Random Forest only achieves 32% to 35% correctness.

Keywords

predictive process monitoring, predictions, machine learning algorithms, business processes

1. Introduction

Some students are overwhelmed by the number of lectures and exercises at the beginning of their time at a university. Others underestimate the exams and do not invest enough effort and performance in the exercises and learning during the semester. *Predictive Process Monitoring* aims to predicting the results (e.g. grades) of a running and uncompleted business process (e.g. lecture during a semester) [1]. To motivate students to prepare for the upcoming exams during the semester, perform the exercises, and study the lecture material continuously we will predict their grades and points of upcoming exercises based on the already gained exercise points. Moreover, a supervisor of a lecture may provide additional exercises to support the students, when the algorithm predicts worse grades. For this, the regression algorithms Neuronal Networks, Decision Trees, and Random Forest are implemented and tested. To train and test the algorithm a data set of 334 students (i.e. 266 for training and 68 for testing) including their exercise points and grades is used. In total, 17,136 predictions are determined, evaluated, and compared with each other.

16th Central European Workshop on Services and their Composition, February 29 – March 1, 89081, Ulm, Germany

*Corresponding author.

✉ lisa.arnold@uni-ulm.de (L. Arnold); marius.breitmayer@uni-ulm.de (M. Breitmayer);
manfred.reichert@uni-ulm.de (M. Reichert)

ORCID 0000-0002-2358-2571 (L. Arnold); 0000-0003-1572-4573 (M. Breitmayer); 0000-0003-2536-4153 (M. Reichert)

 © 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

S. Böhm and D. Lübke (Eds.): 16th ZEUS Workshop, ZEUS 2024, Ulm, Germany, 29 February–1 March 2024, published at <http://ceur-ws.org>

The remainder of this paper is structured as follows. Section 2 explains the basics of the three machine learning algorithms. In Section 3 the initialisation and implementation of the algorithms is presented. Section 4 evaluates the results of the prediction and compares them with each other. Related work is discussed in Section 5 and Section 6 concludes the paper.

2. Background

In the context of this paper, the following three machine learning algorithms for the continuous prediction of grades and exercise points for students during a semester are implemented. The fundamentals of the three algorithms are explained in this section.

The **Neuronal Network** [2, 3] comprises a large number of artificial neurons and their connections to each other. These are designed to mimic the function of the human brain. The Neuronal Network exists of one input layer, at least one hidden layer, and one output layer. The input layer provides the Neuronal Network with the necessary information (i.e. input data). The input neurons process these input data and pass them to the next layer in a weighted manner. The hidden layer receives the information from the input layer and forwards the information from neuron to neuron to the output layer. The transport of information between two neurons of different layers is always weighted individually. The hidden layer is essentially a black box and not visible for users. The output layer is directly connected to the last hidden layer. The output neurons contain the resulting decision.

The **Decision Tree** [3, 4, 5] is a binary, hierarchical tree, in which each node has at most two children with one root node at the top. The Decision Tree is built on the data training set. The algorithm starts at the root node (i.e. at the top of the tree) and pass through the tree until a leaf (i.e. node without own children) is reached. In each node a decision is made which children node will be activated next. Decision trees can be categorised into classification trees, where the target variable can assume a discrete set of values, or regression trees, which are based on continuous values (i.e. real numbers). A node of a Decision Tree may consist of decision rules, several parameters (e.g. maximum depth of the tree), and/or eventually result values. The Decision Tree is individual for each data set. After executing the Decision Tree, the activated leaf node contains the resulting decision.

The **Random Forest** [3] consists of many Decision Trees (there is no limit to the number of Decision Trees). In general, the Random Forest defines an arbitrary number of Decision Trees and connect them with an independent root node. For the final result each Decision Tree is activated, and the individual results are aggregated (e.g. majority voting or averaging). Each of the Decision Trees result in different outputs. In addition to the parameters of the individual Decision Trees, the Random Forest requires further input variables (e.g. maximum number of individual trees into the Random Forest).

3. Implementation

Data set: For the prediction, the exercise points, and grades of 334 students from Ulm University, which are stored in the lecture 'databases' from the years 2018 until 2021 are used. The data set has been anonymised and does not contain any personal information. The maximum points

for each exercise sheet are different over the semester and compared to the previous semesters. To merge the data sheets a relative number is used in the form of percentage (i.e. 0-100%). Furthermore, the official grades (x.0, x.3, x.7 for $x = \{1, 2, 3\}$ and 4.0) are possible. Students that failed the exam receive a 5.0 automatically. Some students have a grade bonus that may falsify the results as students who receive this are automatically one grade level better (except for a 1.0 or a 5.0 for failing). To evaluate the results, the grade bonus will be subtracted from the grades.

Data Adjustments: The original data set is adjusted to uniform data types. When students failed or did not submit an exercise sheet, different input variables (e.g. '0', NULL-Value '-', or a blank field) are given for the same context at the data set. Regarding the regression algorithms used, all non-numerical values are converted into equivalent numbers (i.e. real numbers), in this case into '0'. Additionally, students who had not submitted any exercise sheets, but had nevertheless taken part in the exam were deleted from the data set.

Data Distribution: The data set is split into training and test data set. With the training data set the models (e.g. Neuronal Network or Decision Tree) are generated and with the test data set the algorithms and its models are tested. In general, 80% of the data set is used for the training and the remaining 20% for testing. For the distribution of the data set three different methods are implemented and compared to avoid irregular distribution and avoid potential discrepancies.

First method takes the first 80% of the data set for testing and the remaining for training. With this method, the data set for training only consist of the students who took part in the year 2021. To avoid differences between the years the second method randomly assigned 80% for training and 20% for testing. The third method uses every 5th line for testing and the remaining for training. With the latter, a repeatable and reproducible method in comparison to the randomised method and a mixed distribution over the different years can be achieved.

Array Preparation: To develop the algorithms the outcome needs to be extracted from the data arrays (i.e. training and test data). For the example of this paper, exercise points from upcoming exercise sheets and the final grade will be predicted. The two latter define the outcomes of the given context. For the training phase, the training array as well as the desired outcome handed over to algorithm. For the subsequent testing, only the test array without the outcomes is passed to the respective algorithms. The resulting predictions can then be compared to the actual values (i.e. outcome).

3.1. Implementation of the Algorithm

The **Neuronal Network** uses the described input and output arrays to predict their defined results (e.g. exercise points or grades). The number of neurons in the input layer is manifested in the variable *input_shape* and depends on the prediction (i.e. for the prediction of Exercise Sheet 5, the variable *input_shape* is set to 4 based on four previous sheets). In the hidden layer, the number of neurons is set to 200 and the activation function *ELU* is selected, as this function achieved the best results. Due to the small data set, the number of epochs (i.e. the number of passes through the training data set to train the network) is set to 900, which corresponds to the best results in manual tests. In the output layer, the number of neurons is set to 1 and either the exercise points or the grade may be predicted. To improve the manually tested Neuronal

Network, the function `GridSearch` (i.e. results in best fit values over all combinations of given parameters) and `callback` (i.e. stops the training when the best results have been achieved) is used. Through this improvement, the number of epochs is increased to 5000. In total, 1512 combinations are tested.

A **Decision Tree** requires several parameters to be initialised. In this example, the four parameters `criterion`, which measures the quality of a split within the branches, `max_depth`, which specifies the maximum depth of the Decision Tree, `max_features`, which specifies the maximum number of outcomes, and `random_state` for recovery a Decision Tree are selected in order to test the resulting predictions (i.e. exercise points and grades). For the parameter `criterion`, all four possible values (i.e. `squared_error`, `friedman_mse`, `absolute_error`, `poisson`) are tested. The `max_depth` and `max_features` accept fixed values, and a selection of values is tested. `Random_state` is used to achieve the same results when the algorithm is executed multiple times with the same parameters. The input array (e.g. `training_dataset`) depends on the outcome (e.g. points of Exercise Sheet 4). To improve the manual tested combination of parameters, additionally a `GridSearch` is implemented. For this, the described parameters with its values are extended with the parameters `splitter`, which defines the rule to split at a node, `min_weight_fraction_leaf`, which defines the total size of weights within the sample to reach a leaf node, `min_samples_leaf`, which defines the minimum sample size (i.e. no further splitting is allowed), and `max_leaf_nodes`, which defines the maximum number of leaf nodes. In total, 1440 combinations are tested.

Comparing the **Random Forest** with the Decision Tree they have the common parameters `max_depth`, `max_features`, `min_samples_leaf`, and `random_state`, but with different values for each tree. In addition to them, the parameter `n_estimators`, which defines the maximum number of individual trees within the forest is required for the implementation. To improve the Random Forest again a `GridSearch` is implemented. For this, the described parameters are extended with the parameters `bootstrap` and `n_jobs`. If the parameter of `bootstrap` is set to `False`, the whole data set is used to train each tree, otherwise, the size for samples may be defined. The parameter `n_jobs` defines the number of jobs (i.e. functions) that can run in parallel. In total, 1620 combinations are tested.

4. Evaluation and Results

For the comparison of the predicted results, **Actual Value** describes the points or grade a student received in the exercise sheet or exam. The parameter **Prediction** presents the results determined from the algorithms. The parameter **Variance** defines different values of possible deviations. First, no variance between the actual value and the prediction is acceptable (i.e. the variance is 0% for the exercise points and 0.0 for the grades). Second, a variance of 10% is acceptable for the exercise points and 0.4 for the grades and, third, 15% for the exercise points or 0.7 for grades are acceptable. Moreover, on average, an exercise sheet has between 20 and 25 points. An acceptable variance of 10% results in a total variance between 2 and 2.5 points. In addition, the variance of 4% for exercise points is also compared in the evaluation, which corresponds to a variance of one point per exercise sheet. The parameter **Match**, checks whether the defined variance is within the scope (i.e. `true`) or not (i.e. `false`).

4.1. Results

In total, the grades, and points of all 68 students are predicted for each sheet (i.e. Exercise Sheet 2 to 12) and their grade. In addition, these predictions are determined for each defined variance and each algorithm. In total, 17,136 predictions are determined, evaluated, and compared. In Tab. 2, the averages of correct answer (i.e. parameter *match* results in *true*) over each specified variance and algorithm for all Exercise Sheets 2 to 12 are shown. The percentage results (RES for short) of Tab. 2 are coloured in **red** for $RES < 30\%$, **orange** for $30\% \leq RES < 50\%$, **yellow** for $50\% \leq RES < 75.0\%$, **light green** for $75\% \leq RES < 90\%$ and **green** $90\% \leq RES$.

Table 1

The percentages of correct prediction for each exercise sheet and algorithm over the specified variances.

Exercise Sheet	Neuronal Network				Decision Tree				Random Forest			
	0%	4%	10%	15%	0%	4%	10%	15%	0%	4%	10%	15%
2	0.0	57.4	86.8	92.6	1.5	77.9	94.1	95.6	0.0	76.5	94.1	95.6
3	0.0	47.1	82.4	95.6	0.0	48.5	82.4	91.2	0.0	47.1	80.9	91.2
4	0.0	60.3	92.6	97.1	0.0	58.8	95.6	95.6	0.0	79.4	94.1	97.1
5	0.0	60.3	94.1	97.1	1.5	57.4	88.2	98.5	0.0	64.7	91.2	97.1
6	0.0	36.8	77.9	89.7	0.0	33.8	77.9	88.2	0.0	42.6	76.5	88.2
7	1.5	25.0	52.9	82.4	0.0	25.0	52.9	77.9	0.0	25.0	60.3	73.5
8	0.0	35.3	70.6	92.6	0.0	29.9	77.9	89.7	0.0	36.8	72.1	89.7
9	0.0	41.2	80.9	91.2	0.0	42.6	67.6	89.7	0.0	42.6	82.4	86.8
10	0.0	27.9	67.6	80.9	0.0	32.4	67.6	88.2	0.0	36.8	79.4	86.8
11	0.0	17.6	36.8	57.4	0.0	13.2	41.2	64.7	0.0	23.5	54.4	72.1
12	0.0	4.4	11.8	19.1	2.9	8.8	11.8	27.9	0.0	8.8	25.0	36.8

Predicting 0% variance is close to impossible in any algorithm. This is because the points of the exercise sheets only allow whole or half points (i.e. $x.0$ or $x.5$). When transforming these points (between 0 and 25) into percentages (0 and 100) more than half of the percentage values are not addressed. In addition, the predictions allow decimal numbers with one decimal place. To predict the exact value when the points are converted is almost impossible. Exercise Sheet 11 and 12 also predict the points in most of the cases wrong. Considering the distribution of points across all sheets, it is remarkable that many students did not receive any points in the last two sheet (i.e. 0 points). The reason for this could be that the last exercise sheets deal with content that is no longer relevant to the exam or the students have completed their admission to the exam (i.e. students have achieved a minimum number of points across all exercise sheets). The Random Forest results the best predictions at 4% variance (i.e. variance of 1 point at the exercise sheet), however, these results are between 25.0% and 79.4%. In general, Exercise 7 achieves the worst results, while Exercise 4 achieves the best results. In Exercise Sheets 2 to 5, very positive results are achieved across all algorithms for 10% and 15% variance. The algorithms differ only minimally (i.e. up to 4.4%). In total, the best results are achieved by the Decision Tree with 98.5 correct predictions for 15% variance and 95.6% at 10%.

The predictions of the grades are capable of expansion. The Neuronal Network achieves only 29.4% with a variance of 0.3, and 48.5% with a variance of 0.7. The Decision Tree is slightly better with 32.4% with a variance of 0.3, and 57.4% with a variance of 0.7. The Random Forest

Table 2

The percentages of correct grade prediction for algorithm over the specified variances.

Variance	Neuronal Network			Decision Tree			Random Forest		
	0	0.3	0.7	0	0.3	0.7	0	0.3	0.7
Accuracy	14.7%	29.4%	48.5%	10.3%	32.4%	57.4%	8.8%	35.3%	61.8%

has a slightly better result than the Decision Tree with 35.3% with a variance of 0.3, and 61.8% with a variance of 0.7. For a variance of 0.0 the Neuronal Network predicts the best results with only 14.7% correctness (e.g. Decision Tree results in 10.3% and Random Forest results in 8.8%). There is no strict individual examination in the exercise sheets, i.e. students can help each other, read the script again and have no time pressure during the editing the exercise sheet. Additionally, the effort of learning directly before the exam is individual. All these factors are not considered at the current implementation and may be the reason for the inaccurate results.

5. Related Work

In [6], the performance (i.e. excellent, good, poorly, or fail) for around 400 students from the university of Bangladesh is predicted comparing eight different supervised algorithms (i.e. Support Vector Machines, Decision Tree, Multilayer Perceptron, Extra Tree, K-Nearest-Neighbour, AdaBoost, Logistic Regression, and Weighted Voting Classifier). In total, the Weighted Voting Classifier achieves the best results in predicting the performance category (i.e. excellent, good, poorly, or fail) of each student with 81.73% correctness. In [7], Neuronal Networks are used to predict the drop-out rates of students. The results are based on 2670 students at a Public University in Ecuador. Thereby, the input layer of the Neuronal Network consists of 11 neurons that are based on university factors (i.e. type of learning) and personal life decisions (e.g. pregnancy, born children and financial commitment). Two different kinds of Neuronal Networks were tried out: Multi-layer perception with 98.6% correctness and radial basis function with 98.1% correctness.

6. Conclusions

This paper implements three different regression algorithms (i.e. Neuronal Networks, Decision Trees, and Random Forest) to support and motivate students continuously during the semester by predicting their grades depending on their current performance (i.e. exercise points). To train and test the algorithms, a data set of 334 students with their exercise points and grades from previous semesters are used. Different variances (i.e. 0%, 4%, 10%, and 15%) for exercise points are tested and compared. With a variance of only 10% each algorithm may predict (for some but not all exercise sheets) a correctness between 91% and 96%. Predicting the exact points of upcoming exercise sheets (variance of 0%) none of the implemented algorithms works due to the transformation of points into percentages (i.e. finer range of points). On top, only a prediction of 35.3% with a variance of 0.3 and 61.8% with a variance of 0.7 using a Random Forest is possible. The results of the other two algorithms are slightly worse. In future work,

the data set will be extended by collecting more data over the coming semesters. In addition, further algorithm will be implemented and tested to predict the grades more accurate. Moreover, other factors (i.e. learning type or financial commitment as student jobs) will be considered.

Acknowledgments. *This work is part of the ProcMape project, funded by the KMU Innovativ Program of the Federal Ministry of Education and Research, Germany (F.No. 01IS23045B).*

References

- [1] W. M. van der Aalst, J. Carmona, Process mining handbook, Springer Nature, 2022.
- [2] J. Moolayil, J. Moolayil, S. John, Learn Keras for deep neural networks, Springer, 2019.
- [3] A. Géron, Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow, " O'Reilly Media, Inc.", 2022.
- [4] B. Gupta, A. Rawat, A. Jain, A. Arora, N. Dhimi, Analysis of various decision tree algorithms for classification in data mining, International Journal of Computer Applications 163 (2017) 15–19.
- [5] Y. Zhao, Y. Zhang, Comparison of decision tree methods for finding active objects, Advances in Space Research 41 (2008) 1955–1959.
- [6] M. S. Zulfiker, N. Kabir, A. A. Biswas, P. Chakraborty, M. M. Rahman, Predicting students' performance of the private universities of bangladesh using machine learning approaches, International Journal of Advanced Computer Science and Applications 11 (2020).
- [7] M. Alban, D. Mauricio, Neural networks to predict dropout at the universities, International Journal of Machine Learning and Computing 9 (2019) 149–153.