Analysing Student Performance using Sparse Data of Core Bachelor Courses

Mirka Saarela
University of Jyväskylä
mirka.saarela@jyu.fi

Tommi Kärkkäinen
University of Jyväskylä
tommi.karkkainen@jyu.fi

Curricula for Computer Science (CS) degrees are characterized by the strong occupational orientation of the discipline. In the BSc degree structure, with clearly separate CS core studies, the learning skills for these and other required courses may vary a lot, which is shown in students' overall performance. To analyze this situation, we apply nonstandard educational data mining techniques on a preprocessed log file of the passed courses. The joint variation in the course grades is studied through correlation analysis while intrinsic groups of students are created and analyzed using a robust clustering technique. Since not all students attended all courses, there is a nonstructured sparsity pattern to cope with. Finally, multilayer perceptron neural network with cross-validation based generalization assurance is trained and analyzed using analytic mean sensitivity to explain the nonlinear regression model constructed. Local (within-methods) and global (between-methods) triangulation of different analysis methods is argued to improve the technical soundness of the presented approaches, giving more confidence to our final conclusion that general learning capabilities predict the students' success better than specific IT skills learned as part of the core studies.

Keywords: Sparse Educational Data, Triangulation, Curricula Refinement, Correlation Analysis, Robust Clustering, Multilayer Perceptron
[This version was updated August 19, 2015, for format revision]

1. INTRODUCTION

The development of a curriculum for Computer Science (CS) can be challenging in an academic environment, given the discipline's strong occupational orientation. Especially at multidisciplinary universities (i.e., with many subject areas), the CS curriculum differs from the curricula of many other disciplines, as the core courses reflect to a large extent the vocational side of the program. In the case of the Department of Mathematical Information Technology (DMIT) at the University of Jyväskylä in Finland (reflecting both Finnish and European degree structures), the core bachelor courses compose only about 50 out of the minimum 180 ECTS (i.e., credits measured using the European Credit Transfer and Accumulation System) for the 3-year BSc degree (see Table 2). The degree contains other major courses in addition to separate introductory topics (e.g., general science, language and communication skills, statistics) and minor subject studies (especially mathematics). Students should acquire knowledge of very specific technical (e.g., programming) skills; however, computing interacts with many different domains, and in order to prepare students as the workforce of the future, domain knowledge as well as soft skills

and personal attributes are important (Sahami et al., 2013a). For more than 40 years, roughly every 10 years, the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE) have promoted the creation of international curricular guidelines for bachelor programs in computing (Sahami et al., 2013b). Thus far, however, there has been little discussion about the relation between specific CS courses and other courses, in terms of the overall study performance.

Some researchers (see, e.g., Kinnunen et al. 2013 and references therein) indicate that primarily difficulties in mastering programming lead to high dropout rates in CS, therefore, one should pay special attention to them. Furthermore, a popular belief is that mathematical talent is the key skill for CS students to be successful (Jerkins et al., 2013). Although these topics are important, they do not cover the whole degree. The CS core of the DMIT curriculum for undergraduate students at the University of Jyväskylä, one of the largest and most popular multidisciplinary universities in Finland, has been more or less the same in recent years. Since the curriculum is typically updated every three years, the aim of this research is to focus on a set of mandatory courses related to the data collection period August 2009 through July 2013.

In addition, DMIT undergraduate students require more time to finish their studies compared to students of other disciplines at the University of Jyväskylä (Halonen, 2012). This happens even if the student's view on the quality of teaching and the study atmosphere at DMIT Jyväskylä is very positive and, in fact, better than in the whole Faculty of Information Technology (of which the DMIT is a part) or in the other departments at the university (Halonen, 2012). Actually, only a very few students (on average 12.8%) of DMIT complete the national target of at least 55 ECTS per academic year (Harden and Tervo, 2012). These study efficiency shortcomings apply to the absolute and relative number of credits and are especially important compared with students of other departments at the University of Jyväskylä, who amass many more credits in an academic year (29% acquire at least 55 ECTS).

To assess the current curriculum, we apply the educational data mining (EDM) approach. EDM consists of developing or utilizing data mining methods that are especially feasible for discovering novel knowledge originating in educational settings (Baker and Yacef, 2009) and supporting decision-making in educational institutions (Calders and Pechenizkiy, 2012). Most of the current case studies in EDM (see Table 1) analyze the steadily growing amount of log data from different computer-based learning environments, such as *Learning Management Systems* (e.g., Valsamidis et al. 2012), *Intelligent Tutoring Systems* (e.g., Hawkins et al. 2013; Bouchet et al. 2012; Carlson et al. 2013; Springer et al. 2013), or even *Educational Games* (e.g., Kerr and Chung 2012; Harpstead et al. 2013). Mining those data supports the understanding of how students learn and interact in such systems.

In our study, however, we are interested in understanding the effects of core CS courses and providing novel information for refining repetitive curricula. More specifically, we want to understand the effect of the current profile of the core courses on students' study success. These courses are taught in an ordinary fashion, meaning that in order to successfully complete a course, the student has to attend lectures, complete related exercises, and pass a final exam or assignment at the end of the course. The data analyzed in this paper are the historical log file from the study database at DMIT about all courses students passed for the period August 2009[1] until the end of July 2013. Patterns in our data provide improved profiling of the core courses and an indication of which study skills support timely and successful graduation.

[1]This is because since 8/2009 only ECTS-credits and separated Bachelor and Master degrees can be done.

The remainder of this paper is structured as follows. In Section 2, the overall methodology is explained. Section 3 is devoted to the correlation analysis, while in Section 4 we discuss our clustering analysis with robust prototypes. In Section 5, prediction analysis is realized with the multilayer perceptron (MLP) neural network. Conclusions from the domain as well as from the methodological level are presented in Section 6.

2. THE OVERALL METHODOLOGY: ADVOCATING MULTIPHASE TRIANGULATION

Baker et al. (2010) classify EDM methods into five categories: prediction, clustering, relationship mining, discovery with models, and distillation of data for human judgment. In Table 1, we summarize a representative set of EDM studies according to a) their data and the environment, b) goal of the study, c) EDM category and methods, and d) the knowledge discovered. This work was selected from forums, such as the Journal of Educational Data Mining, related annual conferences, and Google Scholar during autumn 2013. According to the table, which is organized by the different tasks and publication dates, scholars usually apply methods belonging to one of the classes of Baker et al.'s taxonomy to address a particular EDM problem. Moreover, predictive studies may apply many classifiers to assess the stability and reliability of the results. We, however, aim at multiphase triangulation: Different phases of the overall treatment within-methods and between-methods are varied and assessed (using rankings) to increase the technical soundness of the procedures and the overall reliability of the concluded results.

Generally, triangulation means that the same research objective is investigated by different data, theories, analysis methods, or researchers and then combined to arrive at convergent findings (Denzin, 1970). Probably the most popular way to apply triangulation is to use qualitative and quantitative methods and merge their results (Jick, 1979). We employ *between-method* triangulation (e.g., Denzin 1970; Bryman 2003), using techniques from distinct classes of the EDM taxonomy, to study the success patterns of the students who take the core courses of the computer science program in our department. First, we apply correlation analysis (Section 3), a key technique in *relationship mining*. Second, we utilize a special *clustering* approach (see Section 4) to find groups of students with similar course success. Third, we apply *prediction* (see Section 5) with model sensitivity analysis. In all between-methods, we discuss different *within-methods* that tighten the soundness of the respective between-method result. Moreover, we support our decision making a) in clustering with the *distillation of data for human judgement* (see our explorative and visual analysis in Section 4.2.1) and b) in prediction with *discovery with models* (model sensitivity is used as a component to calculate the mean variable sensitivity of the prediction model; see Section 5.1. To combine and interpret our results from the individual EDM techniques, we introduce a ranking system to which all the between and within analysis methods contribute.

In practice, the whole knowledge discovery process in our study is conducted by following the five classical stages (select the target data from the application domain, preprocess, transform, mine the transformed data, and interpret the results) introduced by Fayyad et al. (1996). Data preprocessing and transformation were performed in Java, while the data mining / machine learning techniques were either used as is (correlation analysis in Matlab's Mathematics package) or completely self-implemented (clustering and prediction as a whole) on the Mathworks Matlab R2013b platform.

Table 1: Overview of related work.

Environment and Data	Goal	**Category**: Methods	Obtained Knowledge
(San Pedro et al., 2013), United States (New York):			
Interaction data of a web-based tutoring system for mathematics from 3747 middle school students in New England plus college enrollment information for the students	Predict whether a student will (5 years later) attend college	**Prediction**: Logistic Regression Classifier	Students who are successful in middle school mathematics as measured by the tutoring system are more likely to enroll 5 years later in college, while students who are bored, confused, or careless in the system have a lower probability of enrolling.
(Vihavainen et al., 2013), Finland:			
Helsinki University, snapshot data from Computer Science student programming course	Predict whether a student will fail the introductory mathematics course	**Prediction**: Non-parametric Bayesian network tool (B-Course)	Students who cram at deadlines in their programming course are at high risk of failing their introductory mathematics course.
(Bayer et al., 2012), Czech Republic:			
Masaryk University, data of Applied Informatics bachelor students, their studies, and their activities in the university's information system (e.g., communication with other students via email/discussion board)	Predict whether a bachelor student will drop out of the university	**Prediction**: J48 decision tree learner, IB1 lazy learner, PART rule learner, SMO support vector machines, NB	Students who communicate with students who have good grades can successfully graduate with a higher probability than students with similar performance but who do not communicate with successful students.
(Kotsiantis, 2012), Greece:			
Hellenic Open University, data from distance learning course on Informatics	Predict students' final marks	**Prediction** M5', BP, LR, LWR, SMOreg, M5rules	Two written assignments predict the students' final grade the best.

Continued on next page

Table 1 – continued from previous page

Environment and Data	Goal	**Category**: Methods	Obtained Knowledge
(Bhardwaj and Pal, 2011), India:			
Purvanchal University, Department of Computer Applications, student data	Predict students' performance	**Prediction**: Bayesian Classifier	Living location has high influence on students' final grade.
(Mendez et al., 2008), United States (Arizona):			
Arizona State University, Science and Engineering student data	Prediction of student's persistence	**Prediction**: Decision Tree, Regression, Random Forest	High school and freshmen GPAs influence persistence the most.
(Erdogan and Tymor, 2005), Turkey:			
Maltepe University, data from student database	Find relations between performance on the entrance exam and later success	**Clustering**: K-means	The results of a student's university entrance exam determine the student's major in many cases.
(Campagni et al., 2012), Italy:			
University of Florence, Department of Computer Science, data of how and when exams were taken	Determine whether students who take exams in the recommended order are more successful	**Clustering**: K-means	Students who follow the *ideal path* perform better in terms of graduation time and final grade.
(Chandra and Nandhini, 2010), Nigeria:			
University in Nigeria, Department of Computer Science, course result data	Identify students' failure patterns	**Relationship Mining**: Apriori Association Rule Mining	Relationship between failed courses which can be used in order to restructure the curriculum (e.g., 2 introductory courses should be passed before the *Mathematical Modeling* course).

2.1. Data and nonstructured sparsity pattern

The original data, the historical log files of the four years, $8/2009 - 7/2013$, of all courses completed by all DMIT students, are challenging: Students are in different stages of their programs, their mandatory courses depend on their starting semester, they come with varying backgrounds,

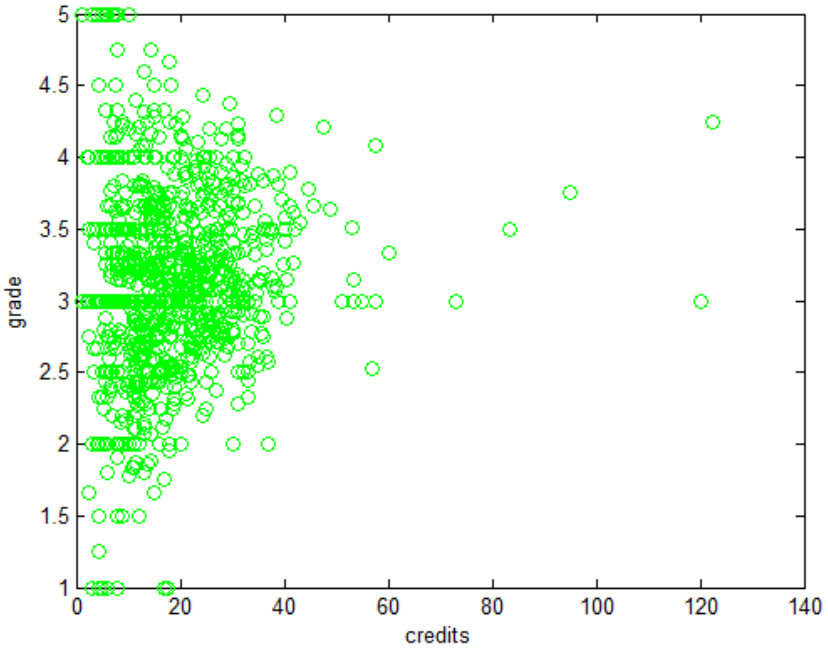

Figure 1: Relationship between the average credits per semester and grades.

have diverse interests, choose their optional courses accordingly, and, as a consequence, realize very different study profiles. This is a typical situation in multidisciplinary universities where students have the opportunity to choose from a large pool of courses. Altogether, our dataset consists of 13640 study records with 21 attributes, related to the passed course and the student's affiliation, and of 1040 students who attended a total of 1271 different courses, completing a total of 64905 credits. Only 64% of these credits the CS students obtained from courses in their own faculty.

When measuring the performance of individual students, in addition to quality, i.e., the grades, the quantity of studies, i.e., the number of all earned credits, is important. However, since our dataset consists of many students at different stages of their education, we cannot compare their individual sums of credits as is. Therefore, we assigned each passed course/record in our dataset to a semester, so that the *mean credits* (i.e., the average number of credits per student per semester) over the active semesters could be computed for all students. An active semester, in turn, is computed as the sum of all semesters between the first semester and the last semester that a student successfully completed a course. For example, a student who passed his or her first course in April 2010 and his or her last course in June 2013 has 7 active semesters. This may include semesters in which the student did not earn any credits. The *mean grade* is simply the sum of all grades divided by the number of courses a particular student has passed.

In general, quality and quantity of the studies of DMIT students do not correlate. The correlation coefficient between the average number of credits per student and the average grade is close to zero (0.0848). The per-student plot of the relationship between the number of credits per semester and the average grade is shown in Figure 1. Also the figure, which looks like a turned bell curve, which means that the grading of the courses resembles the normal distribution, shows visually that earned credits per student do not correlate with the average grade.

Table 2: Core bachelor courses.

course name	course code	course type	completion mode[2]	credits
Computer and Datanetworks as Tools	PCtools	introductory	assignment	2-4
Datanetworks	Datanet	introductory	exercises & final exam	3-5
Object Oriented Analysis and Design[3]	OOA&D	professional	exercises & final exam	3-6
Algorithms 1	Alg1	professional	final exam	4
Introduction to Software Engineering	IntroSE	professional	final exam	3
Operating Systems	OpSys	professional	final exam	4
Basics of Databases and Data Management	DB&DMgm	professional	final exam	4
Programming 1	Prog1	programming	assigment & final exam	6
Programming 2	Prog2	programming	assigment & final exam	8
Computer Structure and Architecture	CompArc	introductory	exercises & final exam	3
Programming of Graphical User Interfaces	GUIprog	programming	exercises & final exam	5
Research Methods in Computing	CompRes	methodological	essay	2
All core courses				47-54

Our goal is to better understand the students' success patterns, given the core courses, in relation to the rest of their studies. Therefore, we want to analyze the students who have completed a certain percentage of the courses of interest. The core courses, a specific set of 12 courses that, for that period of time we study, have been a mandatory part of the curriculum for all DMIT bachelor students, are listed and characterized in Table 2.

If we transform our data in such a way that the 12 core courses become the variables and the attribute value of each observation, corresponding to one student, is the grade of the core course or *missing* if the student did not attend or pass the course, the assembled matrix is very sparse. Only for 13 students are the rows full; the students have passed all the core courses. In Table 3, the high percentage of missing values and the sparsity of the matrix are summarized. The table shows how many students have completed exactly, and respectively at least, q of the 12 courses. Moreover, in each case the percentage of missing values of the cumulative data matrix is provided. The missing data values in the matrix are *missing at random* (Rubin, 1976; Rubin and Little, 2002). This means that the missing values are related to particular variables (some courses that are usually taken later in the program are completed by fewer students; see Figure 2) but not missing because of the values (grades) that could be observed if a particular course is passed.

To analyze such data, one cannot accept too many missing values. In this respect, the *breakdown point* related to statistical estimates (see, e.g., Hettmansperger and McKean 1998) on how much contamination (errors, missing values) in data can be tolerated is informative. An upper bound is easy to establish: If more than 50% of data is missing, then "missing" is the most typical value (mode) of the data. Furthermore, tests conducted with synthetic data show that,

[2]The difference between *assignment* and *exercises* in our system is important: While *assignment* denotes a mandatory work that the student has to fulfill in order to pass the course and affects the final grade the student will receive, *exercises* are smaller (usually weekly) optional tasks that correspond to the current lecture material.

[3]In spring 2012, the *Object Oriented Analysis and Design* ($OOA\&D$) course was split into two separate courses, *Object Oriented Analysis* and *Object Oriented Design*. Therefore, in further analysis the following strategy was applied: If a student completed the original *Object Oriented Analysis and Design* course, the grade from this course was taken for the analysis. However, in case the student did not attend the original course, we used the mean grade of the *Object Oriented Analysis* and the *Object Oriented Design* course as the grade for $OOA\&D$ if the student had completed both newly created courses, or just the grade of the one course if the student had completed only one of these two courses.

Table 3: Number of students who have completed exactly q (n_q) or at least q ($\sum_{q=12}^{Q} n_q,\ Q = 12, \ldots, 0$) of the core courses during the analyzed period.

q	n_q	$\sum n_q$	missing values
12	13	13	0.0%
11	16	29	4.56%
10	22	51	9.81%
9	26	77	14.93%
8	49	100	19.17%
7	28	128	24.10%
6	35	163	29.65%
5	44	207	35.75%
4	46	253	41.37%
3	40	293	45.96%
2	82	375	54.13%
1	126	501	63.57%
0	539	1040	82.45%

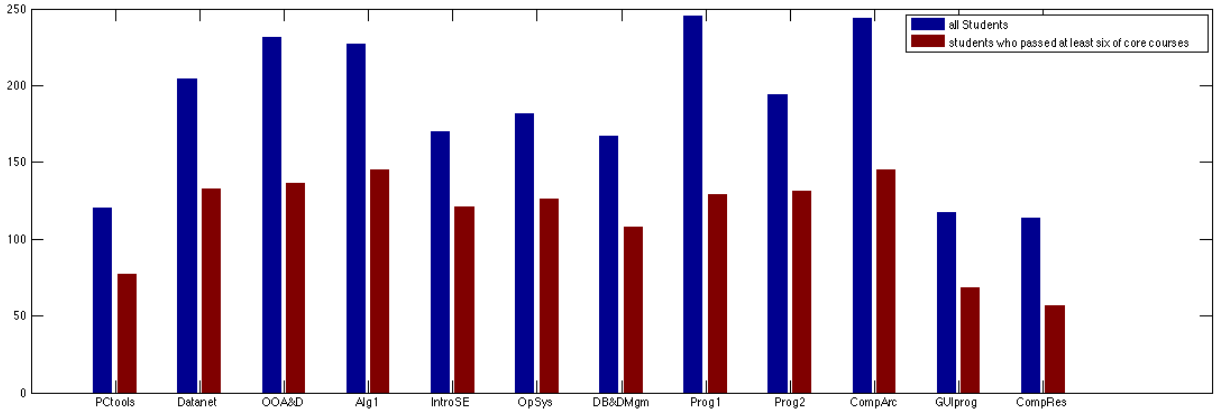

Figure 2: Number of students who passed coursewise.

for example, in clustering with robust methods, reliable results, i.e., almost zero error, can be obtained even if around 30% of the data is missing (Äyrämö 2006; see in particular Figure 22 at page 131). Therefore, our *data selection strategy* is to use that part of the whole, sparse data matrix, which contains the students who have completed at least half of the core courses. This dataset has about 30% missing values (see Table 3) for the multivariate techniques. In the correlation analysis (see Section 3), where the courses are analyzed individually, we similarly use the subsets of the students who have passed the particular course and at least five other courses additionally. In addition, different subsets of the sparse study matrix are utilized to realize some parts of cluster analysis and predictive analysis procedures.

A further challenge, particularly for predicting the study success (see Section 5), is that, for our primary target group, the number of credits related to the core courses is typically less than half of the total number of the earned credits. In Table 4, the percentages of credits originating from the core courses in relation to the total number of credits for the 163 students of interest (see Table 3) are shown. As can be seen in the table, for more than 70% of the students, the core courses account for fewer than half of their studies.

Table 4: Binning of students (nr = number) according to means of the number of core courses in relation to whole studies.

% core courses	nr	cumulative (%)
0-10%	16	16 (10%)
10-20%	24	40 (25%)
20-30%	24	64 (39%)
30-40%	29	93 (57%)
40-50%	25	118 (72%)
50-60%	17	135 (83%)
60-70%	15	150 (92%)
70-80%	7	157 (96%)
80-90%	4	161 (99%)
90-100%	2	163 (100%)

Summing up, for our analysis we have the entire base of completed courses (1040x21) that is processed and transformed to further subsets and the sparse 163x12 data matrix of the students who have completed at least half of the core courses and the grades they received in these courses.

3. CORRELATION ANALYSIS WITH BONFERRONI CORRECTION

As our first EDM technique, we apply relationship mining using correlation analysis. In general, we know from Figure 1 that in terms of grades well-scoring students are not necessarily more likely to study actively. But how about the correlation for our target group, those students who have already completed at least half of the core courses? In the correlation analysis, we do not need special methods for the sparse data. However, the number of students who have passed an individual course differs considerably (see Figure 2) so that the correlation coefficients are computed for different student subsets. The mean number of credits and the mean grade are computed in the same way as explained in Section 2.1.

In Table 5, the correlation of each core course to (i) the mean grade of a student (denoted as *corr.grades*) and (ii) the mean number of credits per semester (denoted as *corr.credits*) is summarized. In each case, *r* identifies the calculated correlation, and *p* corresponds to the *p-value* for testing the hypothesis of no correlation, respectively. The number of stars indicates the strength of the evidence for no correlation. As usual, $\star$ symbolizes the *borderline to be significant* ($p <= 0.05$), $\star\star$ symbolizes *statistically significant* ($p <= 0.01$), and $\star\star\star$ symbolizes *highly statistically significant* ($p <= 0.005$). *rank* denotes the ordering of courses by means of the computed correlations.

From Table 5, we can conclude that, except the *Research Methods in Computing*, all courses have a moderate positive linear relationship to the students' general study success. The course-wise correlations to mean credits per semester are all positive as it should be (passing a course increases credits). In addition, all *corr.grades* illustrate that students who score high in those courses tend to score high in their other courses as well. In particular, this applies to four courses: *Algorithms 1*, *Computer Structure and Architecture*, *Datanetworks*, and *Programming 2*. The correlation between the grades for these four courses and the average grade of the student is in all cases highly statistically significant as the p-values for testing the hypothesis of no correlation are all smaller than 0.005. Similarly as with the classical p-test, we obtained with the conserva-

Table 5: Correlation of each core course to the students' general performance.

	corr.grades				corr.credits			
Course Code	*r*	*p*	*Bonferroni*	*rank*	*r*	*p*	*Bonferroni*	*rank*
PCtools	0.4164	⋆⋆⋆	⋆⋆⋆	11	0.1058	–	–	12
Datanet	0.6244	⋆⋆⋆	⋆⋆⋆	**3**	0.3887	⋆⋆⋆	⋆⋆⋆	**1**
OOA&D	0.5346	⋆⋆⋆	⋆⋆⋆	6	0.1327	–	–	11
Alg1	0.6593	⋆⋆⋆	⋆⋆⋆	**1**	0.3082	⋆⋆⋆	⋆⋆⋆	**4**
IntroSE	0.4197	⋆⋆⋆	⋆⋆⋆	10	0.1717	–	–	9
OpSys	0.5113	⋆⋆⋆	⋆⋆⋆	8	0.1905	⋆	–	8
DB&DMgm	0.5572	⋆⋆⋆	⋆⋆⋆	5	0.3312	⋆⋆⋆	⋆⋆	5
Prog1	0.4314	⋆⋆⋆	⋆⋆⋆	9	0.2438	⋆⋆	–	7
Prog2	0.5731	⋆⋆⋆	⋆⋆⋆	**4**	0.3549	⋆⋆⋆	⋆⋆⋆	**2**
CompArc	0.6511	⋆⋆⋆	⋆⋆⋆	**2**	0.3216	⋆⋆⋆	⋆⋆⋆	**3**
GUIprog	0.5343	⋆⋆⋆	⋆⋆⋆	7	0.3054	⋆	–	6
CompRes	0.2543	–	–	12	0.1609	–	–	10

tive *Bonferroni correction* (Rice, 1989) that the correlation of all core courses to the student's overall grade (except the *Research Methods in Computing*) are highly statistical relevant.

Another conclusion that can be made from Table 5 is that the same four courses that have the highest correlations to the general success of the student also have the highest correlation to the average number of credits. This means that if a student gets a high grade in these courses he or she will probably earn, on average, a high number of credits in the semester as well. Again, all of these findings are, according to the classical p-test as well as the Bonferroni correction, highly statistically significant. Although the ranking is different (e.g., while *Algorithm 1* correlates the most with the mean grade for the student, *Datanetworks* correlates the most with the mean number of credits per semester), we can conclude that those four courses correlate with the students' general performance the best.

To sum up, it can be inferred that a student who achieves a high grade in *Algorithms 1*, *Computer Structure and Architecture*, *Datanetworks*, or *Programming 2* is likely to be successful in the remaining part of his or her studies not only with the grade level but also in terms of speed of completing courses. Albeit overall semesterwise credits and average grade do not correlate at all (see Figure 1), a linear dependency between the grades a student received in the core courses and the general performance exists.

4. CLUSTER ANALYSIS USING ROBUST PROTOTYPES

Our second EDM method is clustering. Generally, clustering can be divided into *partitional* and *hierarchical* clustering (Jain, 2010; Steinbach et al., 2004). However, hierarchical clustering is appropriate only in very small datasets since most of the hierarchical algorithms have quadratic or higher computational complexity (Emre Celebi et al., 2012). Partitional clustering, however, is very efficient and scalable. It partitions the data, such that similar observations are assigned to the same subset of data (referred as a cluster), each observation is attributed to exactly one subset, and each subset contains at least one observation. Since we want to obtain a directly interpretable result, prototype-based partitional clustering is an appropriate approach here. If we can find a partition of data, where each cluster is represented by exactly one prototype, we can use this prototype to analyze the corresponding cluster. Prototype-based partitional clustering

Algorithm 1: Iterative relocation clustering algorithm

Input: Dataset and the number of clusters K.
Output: K partitions of the given dataset.
Select K points as the initial prototypes;
repeat
1. Assign individual observation to the closest prototype;
2. Recompute the prototypes with the assigned observations;

until *The partition does not change*;

can be realized using the iterative relocation algorithm skeleton presented in Algorithm 1 with different score functions (Han et al., 2001) according to which the two steps inside the loop of Algorithm 1 are optimized.

However, in order to realize a prototype-based partitive clustering algorithm, two main issues should be addressed. First, a well-known problem of all iterative relocation algorithms is their initialization. They minimize the given score function locally by iteratively relocating data points between clusters until an optimal partition is attained. Therefore, basic iterative algorithms, such as K-means, always converge to a local, and not necessarily to the global, optimum. Although much work has focused this problem, no efficient and universal method for identifying the initial partitions and the number of clusters exists. This problem is discussed more thoroughly in Section 4.2. The second problem is the sparse student data with around 30% missing values (see Section 2.1). In Section 4.1, a solution is presented for adjusting the score function of the basic algorithm skeleton in order to deal with the random sparsity pattern. A similar approach was also applied in Saarela and Kärkkäinen (2014) to other educational data.

4.1. Score function for K-spatialmedians

Our (available) data consist of course grades of fixed values $\{1, 2, 3, 4, 5\}$. Therefore, there is evidently a significant quantization error from uniform distribution in the probability distribution for a grade g_i:

$$g_i(x) = \begin{cases} 1, \text{if } g_i - \frac{1}{2} \leq x < g_i + \frac{1}{2}, \\ 0, \text{elsewhere.} \end{cases} \tag{1}$$

Thus, second-order statistics that rely on the normally distributed error are not suitable here, and we need to use the so-called nonparametric (i.e., robust) statistical techniques (Huber, 1981; Rousseeuw and Leroy, 1987; Hettmansperger and McKean, 1998). The simplest of robust location estimates are the median and the spatial median. The median, i.e., the middle value of the ordered univariate sample, is inherently one-dimensional, and thus with missing data uses only the available values of an individual variable. The spatial median, however, is truly a multidimensional location estimate and can take advantage of the available data pattern as a whole. This is illustrated and more thoroughly explained in Kärkkäinen and Heikkola (2004); especially, formulae (2.8) and (2.9) and Figures 1 and 2. As stated, e.g., in Croux et al. (2010), the spatial median is not affine but only orthogonally equivariant. However, because we have the fixed grade scale, this property of a statistical estimate is not necessary here. Moreover, for elliptical distributions, this behavior creates more scatter than location estimation (Croux et al., 2010). As a whole, the spatial median has many attractive statistical properties. In particular, its

breakdown point is 0.5; it can handle up to 50% of contaminated data, which makes the spatial median very appealing for high-dimensional data with severe degradations and outliers. A missing value can be thought of as an infinite outlier because it can have any value (from the value range).

Äyrämö (2006) introduced a robust approach utilizing the spatial median to cluster very sparse and apparently noisy data: The *K-spatialmedians* clustering algorithm is based on the same algorithm skeleton as presented in Algorithm 1 but uses the projected spatial median as a score function:

$$\mathcal{J} = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \| \operatorname{diag} \{\boldsymbol{p}_i\} (\boldsymbol{x}_i - \boldsymbol{c}_j) \|_2, \tag{2}$$

Here, diag transforms a vector into a diagonal matrix. The latter sum in (2) is computed over the subset of data attached to cluster j and the projection vectors $\boldsymbol{p}_i, i = 1, \ldots, N$, capture the existing variable values:

$$(\boldsymbol{p}_i)_j = \begin{cases} 1, \text{if } (\boldsymbol{x}_i)_j \text{ exists,} \\ 0, \text{otherwise.} \end{cases}$$

In Algorithm 1, the projected distance as defined in (2) is used in the first step, and recomputation of the prototypes, as the spatial median with the available data, is realized using the sequential overrelaxation (SOR) algorithm (Äyrämö, 2006) with the overrelaxation parameter $\omega = 1.5$. In what follows, we refer to Algorithm 1 with the score function (2) as *K-spatialmedians* clustering.

4.2. Initialization

It is a well-known problem that all iterative clustering algorithms are highly sensitive to the initial placement of the cluster prototypes, and thus, such algorithms do not guarantee unique clustering (Meilă and Heckerman, 1998; Emre Celebi et al., 2012; Bai et al., 2012; Jain, 2010). One might even argue that the results are not reliable if the initial prototypes are randomly chosen since the algorithms do not converge to a global optimum. Numerous methods have been introduced to address this problem. Random initialization is still often chosen as the general strategy (Xu and Wunsch, 2005). However, several researchers (e.g., Aldahdooh and Ashour 2013; Bai et al. 2011) report that having some other than random strategy for the initialization often improves final clustering results significantly.

An important issue when clustering data and finding an appropriate initialization method is the definition of (dis-)similarity of objects. Bai et al. (2011) and Bai et al. (2012) proposed initialization methods for categorical data. The attribute values of our dataset (grades from 1-5, or missing) are also categorical. However, the ordering of our attribute values has meaning (ordinal data). For example, a student who received grade 5 in all his or her courses is more dissimilar to a student who got mostly grade 2 than to a student who received on average grade 4. Therefore, an initialization method for data where only enough information is given to distinguish one object from another (nominal data) might not be suitable for our case.

Chen et al. (2009) proposed a novel approach to find good initial prototypes. Chen et al. argue that in the high-dimensional space data are inherently sparse. Therefore, the distance between each pair of observations becomes almost the same for a wide variety of data distributions. However, this approach seems more suitable for very high-dimensional data than for our 12-dimensional case. Emre Celebi et al. (2012) compared different initialization methods. They conclude that for small datasets (fewer than 10000 observations) Bradley and Fayyad's method

Algorithm 2: Constructive initialization approach for robust clustering

Input: Datasets $\mathbf{D_0}$ to $\mathbf{D_6}$.
Output: The set of prototypes for every value of K
for $K = size(D_0)$ **to** 2 **do**
  $KBestPrototypes$ = *globalBestSolution*(D_0,K);
  for $p = 1$ **to** 6 **do**
    $KBestPrototypes$ = *K-spatialmedians*(D_p,K,$KBestPrototypes$);
  end
end

leads to best results. In Bradley and Fayyad's method (1998), the original dataset is first split into smaller subsets that themselves are clustered. Then the temporary prototypes obtained from clustering the subsets are combined and clustered as many times as there are different subsets. Thus, each time one different set of temporary prototypes is tried as initialization and the best, i.e., that set of temporary prototypes which resulted in the smallest clustering error, is finally used as initialization for clustering the original dataset.

To sum up, the ideal approach for computing initial prototypes depends on the data, and is therefore context dependent. However, some general criteria apply: First, initial prototypes should be as far from each other as possible (Khan and Ahmad, 2013; Jain, 2010). Second, outliers or noisy observations are not good candidates as initial prototypes. Moreover, for relatively small datasets it seems to be a good idea to further divide the set into subsets and utilize the best prototypes of the smaller sets for further computations. Furthermore, as pointed out by Bai et al. (2012), it is advantageous if at least one initial prototype is close to a real solution. Bearing these issues in mind, we developed a new deterministic and context-sensitive approach to find good initial prototypes.

4.2.1. Initialization for sparse student data

Our intention is to interpret and characterize each cluster by its prototype. Therefore, we should prefer full prototypes, those that have no missing values. For this approach, we first note that the rows of Table 3 represent cascadic (see Kärkkäinen and Toivanen 2001) sets of data. Let us denote the datasets as D_p with $p = 0 \ldots 12$, where $p = q - 12$. Thus, D_0 represents the very small but full dataset with the 13 students who have completed all 12 core courses and D_1 the 29 students who have completed at least 11 of them (containing D_0). Therefore, in general D_p consists of students who have passed exactly $12 - p$ of the core courses, and we always have $D_{p-1} \subset D_p$. This creates the basis for the proposed initialization approach, which is depicted as a whole in Algorithm 2.

Our initial, the complete dataset D_0 is so small that we can easily determine the globally best solution by minimizing the error of the spatial median by testing all possible initializations for the values of K[4] In Algorithm 2, *globalBestSolution* refers a function that tests all possible K combinations of the observations in the small complete dataset and returns the prototypes of the combination that resulted in the smallest clustering error. In that way, we obtain for every K for our small dataset the K global best prototypes. We then use the K best prototypes (denoted as $KBestPrototypes$ in the algorithm) on D_p as the initial prototypes for the next larger dataset

[4]Even if K is unknown, we can assume that K is at least 2 and smaller than the total number of observations.

Table 6: Comparison of context-sensitive and random initialization for robust clustering.

	context-sensitive		random	
K	error	missing values	error	missing values
13	373.54	15.38%	425.46	51.54%
12	374.04	8.33%	429.26	46.67%
11	372.31	0.00%	432.89	38.18%
10	376.57	0.00%	434.51	44.00%
9	391.42	0.00%	436.98	38.89%
8	396.06	0.00%	434.77	33.75%
7	409.70	0.00%	444.50	31.43%
6	425.93	0.00%	443.27	18.33%
5	437.32	0.00%	452.74	16.00%
4	454.27	0.00%	461.71	10.00%
3	471.79	0.00%	480.99	6.66%
2	506.84	0.00%	515.06	5.00%

D_{p+1}. Thus, throughout the constructive approach full prototypes and small clustering error are favored. The dataset D_6, the students who have completed at least half of the core courses, is our actual target data for clustering.

In Table 6, it is shown how the score function changes and the number of missing values with the proposed initialization strategy for different values of K for D_6. For comparison, the table also shows the average results of 10 test runs of the *K-spatialmedians* algorithm with random initialization. We obtain better results with our approach for the clustering error and, especially, with respect to the missing values. For example, already for $K = 3$, 6.66% of the prototypes' values are missing with random initialization and, thus, uninterpretable. Moreover, we also studied the stability of the results by checking whether the students in D_{p-1}, $p \geq 1$, still belong to the same cluster when new students are added and the reclustering of D_p is performed in Algorithm 2. Confusion matrices between the two consecutive clustering levels were computed. It turned out that the confusion matrices are almost perfect, so that the formation of clusters is very stable and the clusters themselves are reliably structured. We conclude that the proposed context-sensitive initialization provides a clustering result with low error and high interpretability.

The best value for *K* is next determined using visual inspection. To avoid overfitting, our goal is to have a small number of clusters. However, the observations should not be too far away from the prototype to which they belong. From Figure 3, the plot of the second column in Table 6 (change in the score function when D_6 is clustered using the proposed strategy), we conclude that $K = 3, K = 5$, and $K = 8$ are potential values for the number of clusters. Namely, after precisely these points, the speed of the decrease (improvement) of the clustering error, the discrete derivative, slows down slightly (see, e.g., Zhong et al. 2008 for a similar approach). Of the potential values, we choose the first one that provides the smallest number of clusters for further analysis and, in such a way, generalizes data the most. The prototypes for $K = 3$ are visualized in Figure 4.

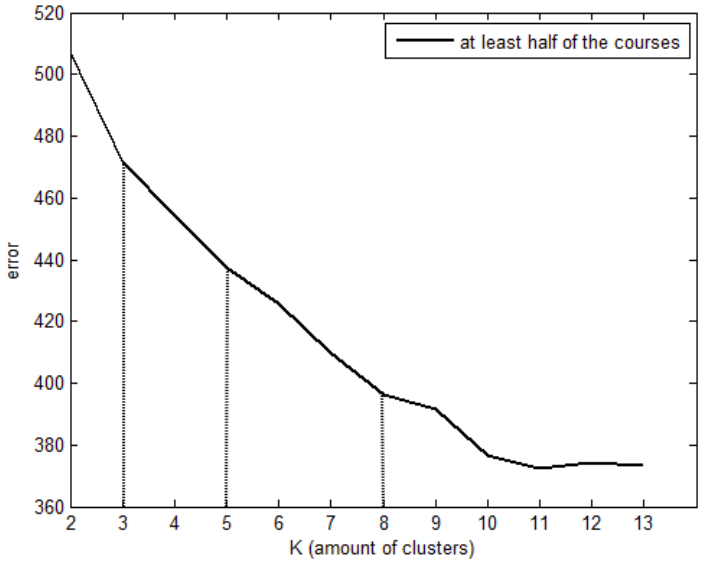

Figure 3: Decrease in errors for target data when more clusters are introduced successively.

4.3. Analyzing the clustering results

In the first two columns of Table 7, the ranking of the core courses based on their prototype separation is provided. Since the general profile of the three clusters is "medium" (*cluster 1*), "high" (*cluster 2*), and "low" (*cluster 3*), we compute, for each variable, two distances: $d_1 = |C_2 - C_1|$ and $d_2 = |C_3 - C_1|$. The two measures are computed (i) as the mean of $\{(d_1)_i, (d_2)_i\}$ (denoted as *measure 1*) and (ii) the minimum of $\{(d_1)_i, (d_2)_i\}$ (denoted as *measure 2*). As can be seen from the table, *measures 1* and *2* provide practically the same ranking. However, we think that of these two indicators, the second measure provides clearer variable separation. For example, with *measure 1* we could have a high distance value for a course even if only one prototype value C_i is very dissimilar from the other two. Moreover, in order to assess even further the explanative power of variables related to the clustering result with $K = 3$, we also applied the nonparametric Kruskal-Wallis test (Hollander et al., 2013) to compare the subsets of data in the three clusters. Since the actual clusterwise datasets contain missing values, we used one iteration of the *hot deck imputation* (Äyrämö, 2006; Batista and Monard, 2003) to complete them: We imputed the missing values using the cluster prototype values of the *K-spatialmedians* algorithm (see Section 4.1) with eight clusters (see Figure 3). As concluded in Section 4.2.1, eight was another good value for the number of clusters *K*. Because of this imputation and because of the form of the quantization error as explained in connection with formula (1), a nonparametric test should be used. According to the Kruskal-Wallis test, the difference between the different clusters is highly statistically significant for all courses. Again, the same four courses provide the highest differentiation between the clusters (see third column of Table 7) with only the *Operating Systems* very different from the distance-based separators. In the fourth column of Table 7, the sum of ranks from the second distance measure and the Kruskal-Wallis test are given, and the overall ranking of the courses based on the sum is provided. This rank-of-rankings approach is an example of within-method triangulation, where the final order of importance combines assessments of the prototypes and the clusterwise data subsets.

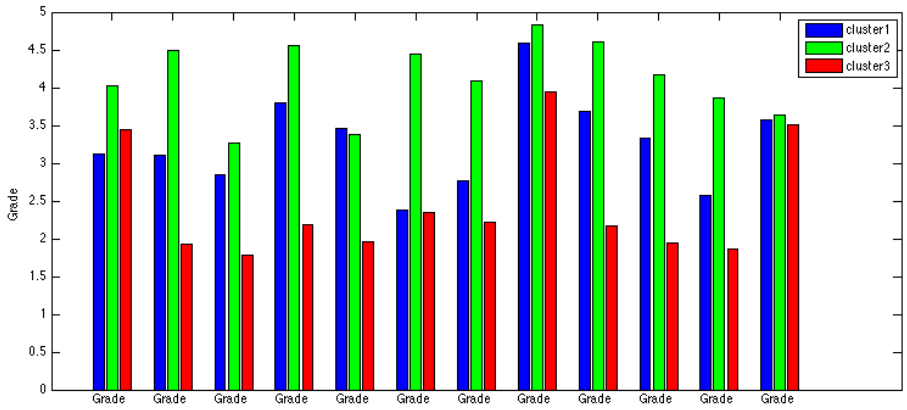

Figure 4: Prototypes of the three student clusters.

Table 7: Distances between the clusters.

	measure 1		measure 2		Kruskal-Wallis			
course code	distance	*rank*	distance	*rank*	χ^2	*p*	*rank*	sum (*rank*)
PCtools	0.1472	8	0.3220	8	33.10	$\star\star\star$	9	17 (9)
Datanet	0.8183	1	1.1754	1	84.69	$\star\star\star$	1	2 (**1**)
OOA&D	0.2249	7	0.4247	7	47.97	$\star\star\star$	7	14 (6)
Alg1	0.6090	3	0.7501	4	82.39	$\star\star\star$	2	6 (**2**)
IntroSE	0.0588	10	0.0781	10	34 .94	$\star\star\star$	8	18 (10)
OpSys	0.0413	11	0.0402	12	67.06	$\star\star\star$	4	16 (7)
DB&DMgm	0.3666	6	0.5490	6	53.96	$\star\star\star$	6	12 (5)
Prog1	0.0796	9	0.2417	9	32.21	$\star\star\star$	10	19 (11)
Prog2	0.7064	2	0.9309	2	54.35	$\star\star\star$	5	7 (**4**)
CompArc	0.5872	4	0.8439	3	78.49	$\star\star\star$	3	6 (**3**)
GUIprog	0.4602	5	0.7118	5	31.04	$\star\star\star$	11	16 (8)
CompRes	0.0021	12	0.0633	11	17.23	$\star\star\star$	12	23 (12)

The first observation that can be made by comparing the overall cluster ranking with the correlation analysis (see Table 5) is that the correlations are reflected in the different clusters. The four courses with the highest correlations clearly separate the three clusters. This can be seen as well from the visualization of the cluster prototypes (Figure 4). The students in the lowest-performing *cluster 3* also have the lowest performance in the *Datanetworks* and *Algorithms 1* course. The prototype of the best *cluster 2* is represented by a remarkable higher grade for those courses. The same applies for the other two courses with a high correlation to the average grade and the average number of credits of the students, *Computer Structure and Architecture* and *Programming 2*. A second interesting observation is that one of the smallest deviations in the grade is obtained for the *Research Methods in Computing*. This was also the only course that did not show a significant correlation to the students' overall grade (see Section 3). However, also the content of this course differs from the other core courses by being not directly related to IT knowledge. Moreover, in contrast to all other courses, this course is evaluated solely by an essay that the student has to write (see Table 2). Somewhat exceptional behavior for this course

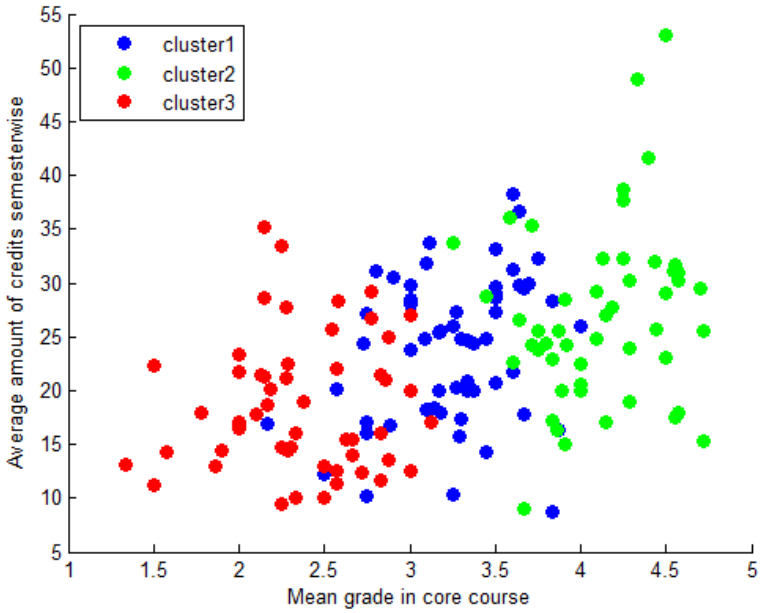

Figure 5: Semesterwise credits versus the mean grade in core courses for the students in each cluster.

was expected.

In terms of quality, the students in D_6 are clearly separated into the three clusters. To check whether the clusters also differentiate the students according to their quantity of studies, we looked also (see Section 2.1 and 3) at the students' average number of credits. In Figure 5, the semester-wise relation of grades in the core courses and the overall credits of the individual students in the different clusters is visualized. From this figure, we deduce that the students who belong to *cluster 2* not only are the best when it comes to the average grades in the core courses but also are the most efficient as they earn on average the most number of credits per semester. Likewise, the students in the gradewise low-performing *cluster 3* also earn the fewest credits per semester (on average eight credits less than the students in *cluster 2*). The correlation coefficient of the mean grade in the core courses and the average number of credits semesterwise per student is 0.4415 with a p-value that is highly statistically significant. We know that this relation does not exist in the whole student level and when the average of all studies is used (see Figure 1). Thus, we conclude that for the core CS courses, the students who perform well in terms of grades also perform well in terms of the number of courses.

5. PREDICTIVE ANALYSIS USING MULTILAYER PERCEPTRON

The goal, when addressing the third EDM category in this study, is to predict the mean grades and credits of the students given only the grades of the core courses they have passed. Similarly as in Section 4, we are interested in interpretable results, which here correspond to detecting the inputs (courses) that contribute to the prediction model the most. Concerning the model, multilayer perceptron (MLP) neural networks are universal nonlinear regression approximators (see, e.g., Pinkus 1999 and articles therein), which can be used in supervised learning. The feedforward MLP transformation starts directly from the input variables, different from other

popular techniques such as radial basis function networks or support vector machines, which construct their basis in the space of observations. This is an appropriate starting point because our purpose is to assess the importance of the model inputs, which correspond to the core courses being analyzed. In this way, we close our between-method triangulation by contrasting the previous results and conclusions based on unsupervised analysis with the corresponding results from a supervised, predictive technique.

There are many inherent difficulties when a flexible model is used in prediction and trained using a given set of input-output samples. First, because of the universality, such a model could actually represent the discrete dataset precisely (e.g., Tamura and Tateishi 1997; Huang 2003), which would mean that all the noise in the samples would be reproduced. Thus, one needs to restrict the flexibility of such models. This can be done in two ways: by restricting the size of the network's configuration (number and size of layers; structural simplicity) or restricting the nonlinearity of the encoded function (size of weights, see Bartlett 1998; functional simplicity). Here we will assess the network's simplicity along both dimensions, in order to favor and restore the simplest model (cf. Occam's razor). Second, we look for a prediction model that provides the best generalization of the sample data, and, for this purpose, apply the well-known stratified cross-validation (see Kohavi 1995) to compute an estimate of the generalization error. Stratification means that, given a certain labeling to encode classes in a discrete dataset, the number of samples in the created folds (subsets) coincides with the sizes of the different classes as closely as possible. Clearly, the number of classes and number of folds do not need to be the same. Third, as in clustering, use of a local optimizer to solve the nonlinear optimization problem to determine the network weights provides only local search (exploitation), and for exploration, we use multiple restarts with random initialization (see Kärkkäinen 2002). The whole training approach as just summarized has been more thoroughly introduced and tested in Kärkkäinen (2014) and successfully applied in time-series analysis in Kärkkäinen et al. (2014).

Next we will derive and detail the whole predictive approach. First, the MLP neural network and its determination are formalized, and then the overall training algorithm and the input-sensitivity analysis are developed and described.

5.1. Prediction with input sensitivity analysis

5.1.1. MLP training approach

The action of the multilayer perceptron in a layered, compact form can be given by (e.g., Hagan and Menhaj 1994)

$$\mathbf{o}^0 = \mathbf{x}, \quad \mathbf{o}^l = \mathcal{F}^l(\mathbf{W}^l \tilde{\mathbf{o}}^{(l-1)}) \text{ for } l = 1, \ldots, L. \tag{3}$$

Here the layer number (starting from zero for the input) has been placed as an upper index. By $\tilde{}$ we indicate the addition of bias terms to the transformation, which is realized by enlarging a vector $\mathbf{v}$ with constant: $\tilde{\mathbf{v}}^T = \begin{bmatrix} 1 & \mathbf{v}^T \end{bmatrix}$. In practice, this places the bias weights as the first columns of the layer matrices that then have the factorization $\mathbf{W}^l = \begin{bmatrix} \mathbf{W}_0^l & \mathbf{W}_1^l \end{bmatrix}$. $\mathcal{F}^l(\cdot)$ denotes the application of activation functions on the lth level. Formally, this corresponds to matrix-vector multiplication in which the matrix components are functions, and component multiplication is replaced with application of the corresponding component function (Kärkkäinen, 2002). The dimensions of the weight-matrices are given by $\dim(\mathbf{W}^l) = n_l \times (n_{l-1} + 1)$, $l = 1, \ldots, L$, where n_0 is the length of an input-vector $\mathbf{x}$, n_L the length of the output-vector $\mathbf{o}^L$, and $n_l, 0 < l < L$, determine the sizes (number of neurons) of the hidden layers.

Using the given training data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$, with $\mathbf{x}_i \in \mathbb{R}^{n_0}$ denoting the input-vectors and $\mathbf{y}_i \in \mathbb{R}^{n_L}$ the output vectors, respectively, the unknown weight matrices $\{\mathbf{W}^l\}_{l=1}^L$ in (3) are determined as a solution of an optimization problem

$$\min_{\{\mathbf{W}^l\}_{l=1}^L} \mathcal{J}(\{\mathbf{W}^l\}). \tag{4}$$

We restrict ourselves to MLP with one hidden layer, and the actual cost function reads as follows:

$$\mathcal{J}(\mathbf{W}^1, \mathbf{W}^2) = \frac{1}{2N} \sum_{i=1}^{N} \left\| \mathcal{N}(\mathbf{W}^1, \mathbf{W}^2)(\mathbf{x}_i) - \mathbf{y}_i \right\|^2 + \frac{\beta}{2n_1} \sum_{(i,j)} \left(|\mathbf{W}_{i,j}^1|^2 + |(\mathbf{W}_1^2)_{i,j}|^2 \right) \tag{5}$$

for $\beta \geq 0$ and $\mathcal{N}(\mathbf{W}^1, \mathbf{W}^2)(\mathbf{x}_i) = \mathbf{W}^2 \tilde{\mathcal{F}}^1(\mathbf{W}^1 \tilde{\mathbf{x}}_i)$. The special form of regularization omitting the bias column $\mathbf{W}_0^2$ is due to Corollary 1 by Kärkkäinen (2002): *Every locally optimal solution to* (4) *with the cost functional* (5) *provides an unbiased regression estimate having zero mean error over the training data*.

The universal approximation property guarantees the potential accuracy of an MLP network for given data and the unbiasedness as just described provides statistical support for its use, but as explained above, we also address the network's *simplicity* and *generalization*. Thus, in our actual training method we grid-search the size of the hidden layer n_1 and the size of the regularization coefficient β: The smaller n_1, the simpler the structure of the network; and the larger β, the smaller the weight values and the closer the MLP to a (simpler) linear, single-layered network. Moreover, cross-validation is used as the technique to ensure that generalization ability of the network is taken as the main accuracy criterion. Finally, the usual gradient-based optimization methods for minimizing (5) act locally, so that we repeat the optimization with random initialization twice when we search for the values of metaparameters n_1 and β. When they have been fixed, the final network is optimized using five local restarts to further improve the exploration of the search landscape.

The whole training approach for the MLP network is given in Algorithm 3. We use the following set of possible regularization parameter values, which were determined according to prior computational tests:

$$\vec{\beta} = \begin{bmatrix} 10^{-2} & 7.5 \cdot 10^{-3} & 5 \cdot 10^{-3} & 2.5 \cdot 10^{-3} & 10^{-3} & 7.5 \cdot 10^{-4} & 5 \cdot 10^{-4} & 2.5 \cdot 10^{-4} & 10^{-4} \end{bmatrix}.$$

The prediction error with a training or test set is computed as the mean Euclidian error

$$\frac{1}{N} \sum_{i=1}^{N} \left\| \mathcal{N}(\mathbf{W}^1, \mathbf{W}^2)(\mathbf{x}_i) - \mathbf{y}_i \right\|. \tag{6}$$

We use the most common sigmoidal activation functions $s(x) = \frac{1}{1+\exp(-x)}$ for $\mathcal{F}^1$. All input variables are preprocessed into the range $[0, 1]$ of $s(x)$ to balance their scaling with each other and with the range of the overall MLP transformation (see Kärkkäinen 2002 for a more thorough argument).

5.1.2. Derivation of input sensitivity of MLP

To assess the relevancy of the input (see John et al. 1994; Kohavi and John 1997) of an MLP model, one basic technique is to estimate the sensitivity of the network's output compared to

Algorithm 3: Reliable determination of MLP neural network.

Input: Training data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$.
Output: MLP neural network $\mathcal{N}(\mathbf{W}^1, \mathbf{W}^2)$.
Define a vector $\vec{\beta}$ of regularization coefficients, maximum size of the hidden layer $n1max$, and $nfolds$, the number of folds for cross-validation, created using stratified random sampling;
for $n_1 \leftarrow 1$ **to** $n1max$ **do**
  for $regs \leftarrow 1$ **to** $|\vec{\beta}|$ *(*$|\cdot|$ *denotes the size of a vector)* **do**
    for $k \leftarrow 1$ **to** $nfolds$ **do**
      for $i \leftarrow 1$ **to** *2* **do**
        Initialize $(\mathbf{W}^1, \mathbf{W}^2)$ from the uniform distribution $\mathcal{U}([-1, 1])$;
        Minimize (5) with current n_1 and $\vec{\beta}(regs)$, and the CV Training set;
        Store Network for smallest Training_Set_Prediction_Error;
      end
      Compute Test_Set_Prediction_Error for the stored Network;
    end
    Store $n_1^* = n_1$ and $\beta^* = \beta$ for the smallest mean Test_Set_Prediction_Error;
  end
end
for $i \leftarrow 1$ **to** *5* **do**
  Initialize $(\mathbf{W}^1, \mathbf{W}^2)$ from $\mathcal{U}([-1, 1])$;
  Minimize (5) using n_1^*, β^* and the whole training data;
end

its input. Seven possible definitions of sensitivity were compared in Gevrey et al. (2003) in an ecological context and four of them, further, in relation to chemical engineering in Shojaeefard et al. (2013). Both comparisons concluded that in order to assess the relevancy and rank the features, the partial derivatives (PaD) method proposed by Dimopoulos et al. (1995) provides appropriate information and computational coherency in the form of stability. Thus, we also use the analytic partial derivative as the core of the sensitivity measure, but in a more general and more robust fashion than Dimopoulos et al. (1995).

An analytical formula for the MLP input sensitivity can be directly calculated from the layer-wise formula (3). The precise result is stated in the next proposition.

Proposition 1

$$\nabla_{\mathbf{x}} \mathcal{N}(\{\mathbf{W}^l\})(\mathbf{x}) = \frac{\partial \mathbf{o}^L}{\partial \mathbf{x}} = \prod_{l=L}^{1} \operatorname{diag}\left\{(\mathcal{F}^l)'\right\} \mathbf{W}_1^l. \tag{7}$$

Here $\mathbf{W}_1^l$ denotes, as before, the lth weight matrix without the first bias column. In particular, for an MLP with one hidden layer and linear output ($\mathbf{o}^2 = \mathbf{W}^2\, \tilde{\mathcal{F}}^1(\mathbf{W}^1\, \tilde{\mathbf{x}})$), (7) states that

$$\frac{\partial \mathbf{o}^2}{\partial \mathbf{x}} = \mathbf{W}_1^2\, \operatorname{diag}\left\{(\mathcal{F}^1)'\right\} \mathbf{W}_1^1. \tag{8}$$

Algorithm 4: Input sensitivity ranking.

Input: Data $(\mathbf{X}, \mathbf{Y}) = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ of inputs and desired outputs.
Output: Ranked list of MLP input variables.
1: Fix $\vec{\beta}$ and $n1max$, and apply Algorithm 3 to obtain $\mathcal{N}(\mathbf{W}^1, \mathbf{W}^2)$;
2: Compute MAS of $\mathcal{N}(\mathbf{W}^1, \mathbf{W}^2)$ according to formula (9);
3: Order input variables in descending order with respect to MAS to establish ranking;

With the discrete data $\{\mathbf{x}_i\}_{i=1}^N$, input sensitivity must be assessed and computed over the dataset. Thus, we apply (7) to compute the *mean absolute sensitivity, MAS* (see Ruck et al. 1990):

$$\frac{1}{N}\sum_{i=1}^{N}\left|\frac{\partial \mathbf{o}^L}{\partial \mathbf{x}_i}\right| \tag{9}$$

of the trained network for all input variables. After this formula is applied, the approach for input ranking is based on the following concept: The higher the MAS, the more salient the feature is for the network. This is due to the well-known Taylor theorem in calculus related to local approximation of smooth functions (see Apostol 1969). Namely, if a function is locally constant, its gradient vector (i.e., the vector of partial derivatives) is zero, and such a function could be (locally) represented and absorbed to the MLP bias. Thus, the larger the mean sum of the absolute values of the local partial derivatives for an input variable, the more important that input variable is for representing the variability of an unknown function approximated by the MLP. Thus, the descending order of MAS values defines the ranking of input variables over one run of Algorithm 3. The method described by Dimopoulos et al. (1995) starts with the similar analytic formula (formula (3)) as in (7), but (7) is a generalization because our MLP model contains the bias nodes in order to always guarantee unbiased regression estimate for the training data in Algorithm 3. Moreover, as with clustering, we compute the overall input-output sensitivity formulae using the robust *mean absolute error* instead of the sum-of-squares proposed in Dimopoulos et al. (1995), which nonuniformly concentrates on large deviations from zero (see Kärkkäinen and Heikkola 2004).

The whole algorithm for deriving the MLP input sensitivity is given in Algorithm 4. To this end, many points in this algorithm may produce variability in the final result, the ranking. With different runs, different foldings appear in cross-validation and different local initializations are tested when seeking the values of the metaparameters n_1 and β. Thus, it typically happens that a different final network is encountered from repetitions of Algorithm 3 whose ranking (1–12, where 1 represents the most significant) is then determined using Algorithm 4. To assess the stability and soundness of this result, we repeat Algorithm 4 five times, store the rankings obtained with different runs, and, then, compute the classical Fleiss kappa κ (Fleiss, 1971), which precisely quantifies the reliability of agreement between a fixed number of MLP network raters. The actual variable rating is then based on the ascending order of the sum of rankings from these five repetitions (between 5–60, where 5 means that such a variable was declared as the most significant for all the repetitions).

5.2. PREDICTIVE RESULTS AND THEIR ANALYSIS

As input data for MLP, we use the same set as in the cluster analysis, i.e. the grades of the students who have completed at least half of the core courses; see Table 3. Moreover, the missing values (29.65% altogether) are again completed by using the hot-deck imputation with 8 prototypes (see Section 4.3 for more thorough description). As output data, for each student considered, we use (i) the mean grade and (ii) the mean number of credits per semester, individually.

Results of the predictive analysis process, as described above, are provided in Table 8. There, for each course, the "RSum" provides the sum of rankings (1–12) of five individual runs of Algorithm 4. Moreover, in order to assess the stability of the final ranking, we have tested 3-fold, 7-fold, and 10-fold stratified cross-validation. As labels for the 3-fold stratification, we used the three cluster indices that were obtained in the previous section for $K = 3$ (the analyzed result). For the 7-fold CV, the labels corresponded to the number of completed courses in Table 3, i.e., to the separate groups of students for $q = 6, \ldots, 12$, whose sizes are given by n_q. In the third stratified cross-validation strategy with 10 folds, we used the labels that were obtained when clustering the students into 8 clusters (same as in imputation).

Thus, the strategy to create the different number of stratified folds was completely different, but the final rankings of the 7-fold and 10-fold CV were exactly the same, and there was only one very small difference compared to the 3-fold CV: For the mean grade, rankings of the *PCtools* and *Datanet* courses were swapped. We conclude that there is high reliability concerning the final rankings, because the Fleiss κ shows *moderate agreement* for grades with 7 and 10 folds and the rest of the cases witness *substantial agreement* between the ratings of the individual runs of Algorithm 4. From "MeanError" (see Table 8), which represents the mean of the prediction error (6) over the five runs, we conclude that mean grades can be predicted (in the generalization sense as explained above) about twice as accurately as the mean number of credits semesterwise. Again, this illustrates the higher and more random individual variability of the number of credits obtained per semester compared to the level of grades (see also Figures 6 and 7).

Based on the results presented in Table 8, we draw the following main conclusions: Compared to the correlation and clustering analysis results, also based on the predictive MLP input sensitivity analysis, the courses *Datanetworks* and *Computer Structure and Architecture* seem to be most influential to the overall performance in the studies. For the performance in grades, also the course *Object Oriented Analysis and Design* pops up, and, for the overall credits, the largest course *Programming 2* shows (as in the previous analyses) high significance.

For some course, like *Computer and Datanetworks as Tools* and *Programming of Graphical User Interfaces*, there is a large difference in the ranks between the mean grades and the mean credits, which was not addressed as strongly by the other two EDM techniques. One reason for this might be the varying number of students passing a course, which is reflected in the predictive analysis as the higher need of imputation. As can be seen from Figure 2, many fewer students have passed these two courses compared to the other courses[5].

The predictions and the prediction errors for grades and credits, studentwise, are illustrated in Figures 6 and 7. In the figures, the x-axis corresponds to a student index, where the students are taken in the ascending order for missing courses; the larger the index, the more core course grades are missing, and were imputed in the MLP training data. With this respect, the accuracy

[5] Actually, also fewer students passed *Research Methods in Computing*, but this course has already been found to be less influential and the most different to the courses (see especially the discussion in Section 4.3).

Table 8: Input rankings for the three foldings.

	3-fold CV				7-fold CV				10-fold CV			
	grades		credits		grades		credits		grades		credits	
MeanError	6.44e-3		1.22e-2		6.37e-3		1.22e-2		6.36e-3		1.22e-2	
Fleiss κ	0.76		0.72		0.49		0.62		0.52		0.78	
Course	RSum	*rank*	RSum	*rank*	RSum	*rank*	RSum	*rank*	RSum	*rank*	RSum	*rank*
PCtools	19	4	49	10	16	3	47	10	15	3	50	10
Datanet	17	3	10	2	19	4	10	2	18	4	10	2
OOA&D	10	2	39	7	12	2	40	7	13	2	39	7
Alg1	30	6	20	4	31	6	20	4	31	6	20	4
IntroSE	36	7	42	9	36	7	42	9	36	7	41	9
OpSys	39	8	57	11	41	8	57	11	41	8	57	11
DB&DMgm	45	9	15	3	42	9	15	3	42	9	15	3
Prog1	60	12	30	6	59	12	32	6	59	12	30	6
Prog2	50	10	5	1	50	10	5	1	50	10	5	1
CompArc	5	1	25	5	6	1	25	5	6	1	25	5
GUIprog	24	5	58	12	22	5	58	12	23	5	58	12
CompRes	55	11	40	8	56	11	41	8	56	11	40	8

of the mean number of credits per semester shows large increase at the end. As can be seen from Figure 6, the grades of the core courses predict, with reasonable accuracy, the overall mean grade level of a student. This result is promising, especially when the number of credits related to the analyzed core courses is typically less than half of the total number of credits; see Table 4.

In contrast, the generalization accuracy of the average number of credits per semester is very bad (see Figure 7), and the last students, i.e., those with the most missing values, are the most erroneous. Thus, we do not recommend the final network for actual prediction, but the network is considered suitable for the sensitivity analysis. The difference between accurate prediction and stable detection of input relevance is also clearly captured in Table 8 as explained above: The rankings in the repeated attempts in Table 8 are very stable, as shown by the Fleiss κ's, even if the prediction accuracy can be very poor as shown in Figures 6 and 7.

Hornik et al. (1989) summarize the essence of MLP training: "We have thus established that such 'mapping' networks are universal approximators. This implies that any lack of success in applications must arise from inadequate learning, insufficient numbers of hidden units or the lack of a deterministic relationship between input and target." The proposed training approach here tries to manage all these issues in order to end up with *the most reliably generalizing MLP network*. Thus, we try to capture the deterministic behavior within the data and use this to compute the input relevance. Stability of the results as witnessed in Table 8, with substantial within-method triangulation, supports the conclusion that this was obtained here.

6. CONCLUSIONS

This paper presents methods for detecting the main courses that determine the general success in CS-oriented studies. We employed techniques from the three main categories of educational data mining, partly working in relation to the remaining two categories as assistance in individual analyses. Moreover, we showed how to cope with the nonstructured sparsity pattern in data,

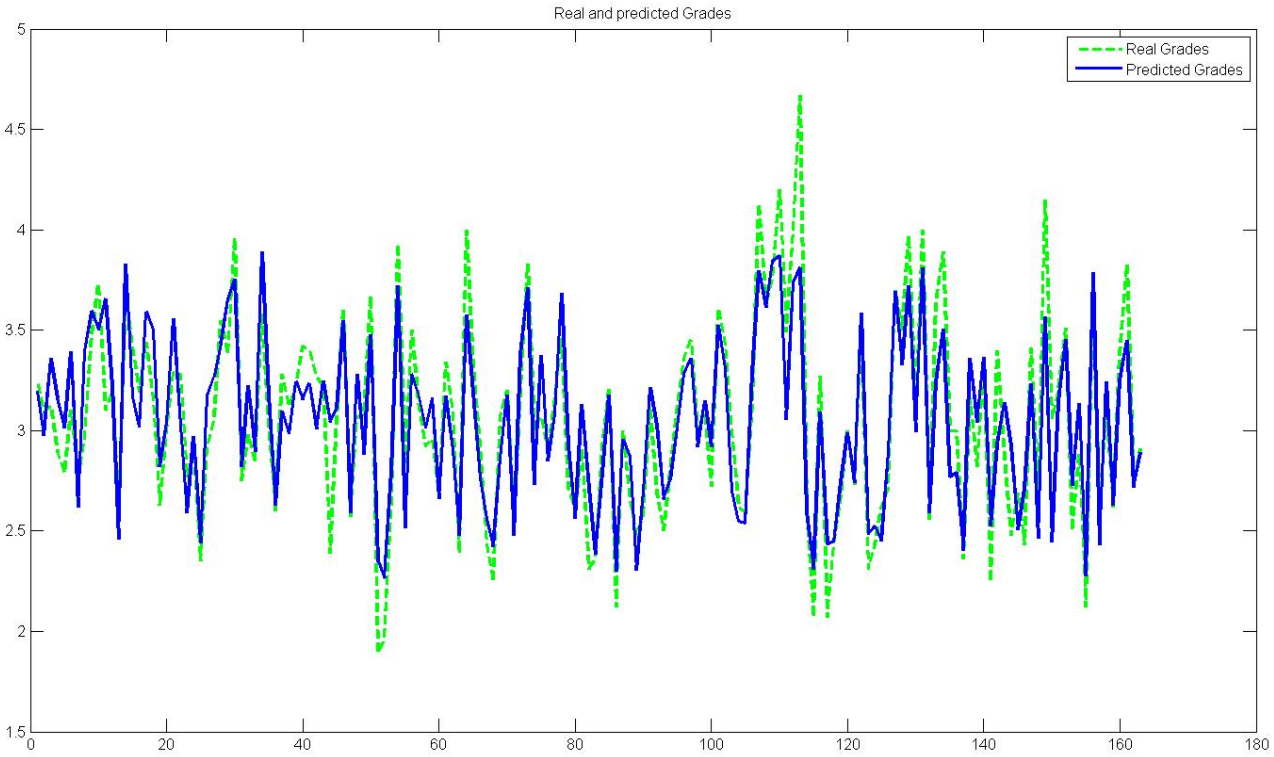

Figure 6: Prediction of mean grades: real (green) and predicted (blue) values.

using the available data strategy and prototype-based imputation. In Table 9, all analysis results are summarized. We can conclude the study from the educational domain level point of view and from the methodological point of view.

From the domain level point of view and based on Table 9, we conclude that the quality of studies is determined by the first introductory courses, *Datanetworks* and *Computer Structure and Architecture*, offered in the first year of the program. Both courses test more the general capability of a student to study than the actual knowledge of professional CS skills. Though they have technical topics, they are taught on a conceptual level, and especially compared with the third introductory course (see Table 2), they are completed by a final examination at the end of the course. Therefore, these courses test how well the student is able to learn, understand, and explain concepts instead of testing specific (IT) skills. When it comes to credits/timely graduation, a student's success is also determined by sedulousness and perseverance: The *Programming 2* which is also creditwise the largest course (see Table 2), is strongly related to the number of credits that a student can earn in general with hard work. Thus, for the overall performance, general study capabilities are more important than the occupational skills and students can succeed in CS studies with diligent and goal-oriented study behavior without being the most skilled programmers with mathematical talent. This is important knowledge that should be communicated to the students in the beginning of their studies.

Naturally, our conclusions from the organizational level as such are not generalizable to

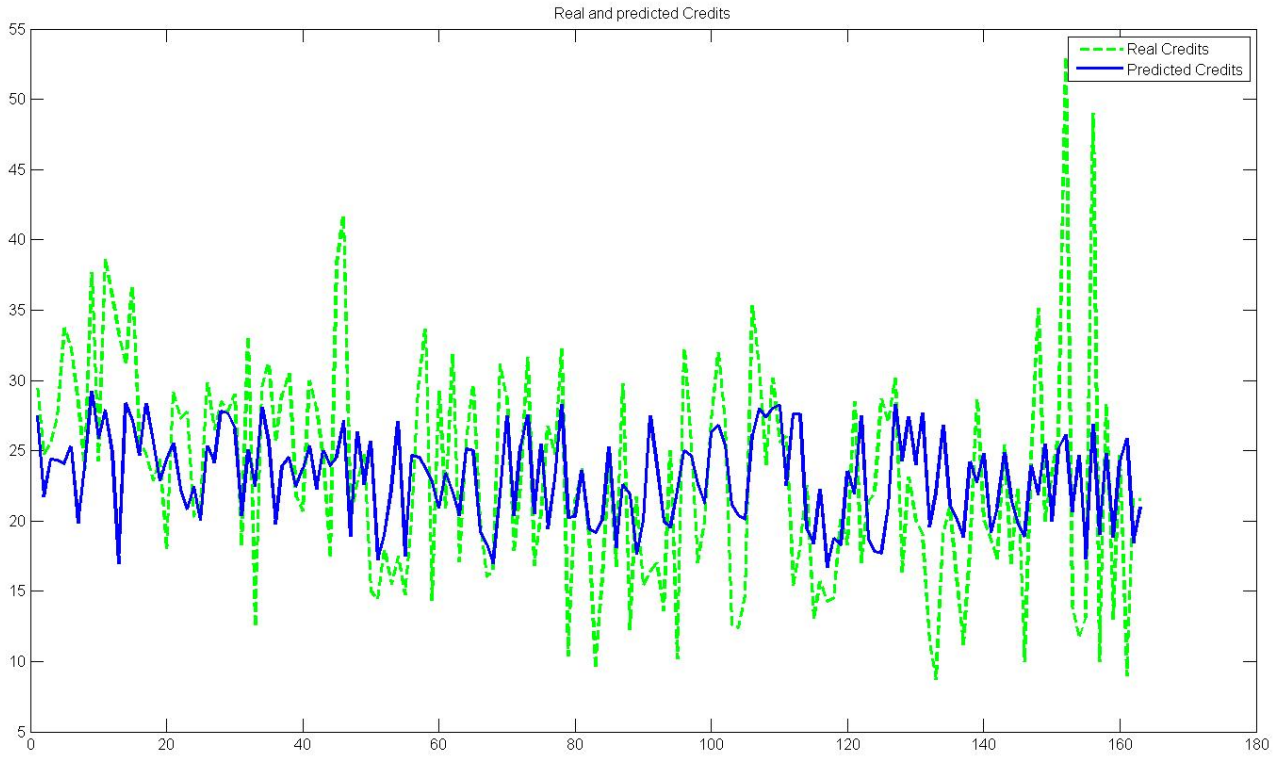

Figure 7: Prediction of mean credits: real (green) and predicted (blue) values.

other institutions since educational data and the subsequent knowledge of particular courses are different. From the methodological perspective, however, both overall approach and the individual methods with their varying but argumented details are general and can be applied to analyze the sparse data of student performance. If a snapshot of a study registry of an arbitrary educational institution were taken, there were missing values similarly to our case for the uncompleted courses. And, then, all methods and approaches could be applied. Furthermore, according to our current computational experience, we can conclude that for around a dozen variables (even using Matlab): i) correlation analysis scales up to one million observations, ii) clustering analysis scales up to hundreds of thousands of observations, and iii) predictive MLP analysis scales up to thousands of observations. This means that our methods can also be used for larger datasets.

In general, on the methodological level, the combination of within-method and between-method triangulation provided very solid results concerning the overall effects and impact of the analyzed courses. To deal with our student data, it was necessary to augment the existing methods and approaches to work with the sparse data. What about the soundness of the algorithms and the overall analysis presented here? There is lot of novelty in the procedures applied. The prototype-based clustering approach with available data spatial median as a statistical estimate is not a standard data mining technique. It was developed in the earlier work of the research group (Äyrämö, 2006; Kärkkäinen and Äyrämö, 2005), and its application is based on our own implementation throughout. Similarly, the way the clustering algorithm is construc-

Table 9: Summary of the results.

	grades				credits			
course	M1	M2	M3	sum (rank)	M1	M2	M3	sum (rank)
Computer and Datanetworks as Tools	11	9	3	23 (8)	12	9	10	31 (12)
Datanetworks	3	1	4	8 (2)	1	1	2	4 (1)
Object Oriented Analysis and Design	6	6	2	14 (4)	11	6	7	24 (7)
Algorithms 1	1	2	6	9 (3)	4	2	4	10 (3)
Introduction to Software Engineering	10	10	7	27 (10)	9	10	9	28 (10)
Operating Systems	8	7	8	23 (9)	8	7	11	26 (9)
Basics of Databases and Data Management	5	5	9	19 (6)	5	5	3	13 (5)
Programming 1	9	11	12	32 (11)	7	11	6	24 (6)
Programming 2	4	4	10	18 (5)	2	4	1	7 (2)
Computer Structure and Architecture	2	3	1	6 (1)	3	3	5	11 (4)
Programming of Graphical User Interfaces	7	8	5	20 (7)	6	8	12	26 (8)
Research Methods in Computing	12	12	11	35 (12)	10	12	8	30 (11)

tively initialized and how the variable ranking of prototypes is derived are not standard choices in cluster analysis. Moreover, the whole computational process for the predictive analysis — use of MLP with a) hot-deck imputation, b) complexity-aware training for best generalization, c) analytic formula-based robust input sensitivity derivation, d) sensitivity ranking, e) Fleiss κ as stability measure for rankings is completely novel. It is also based on our own implementation throughout. Training phase b) has been recently proposed and tested in Kärkkäinen (2014) and Kärkkäinen et al. (2014).

The underlying principle to study soundness in all the treatments here was based on local and global triangulation: In the correlation analysis, significancy was computed with and without Bonferroni correction. In cluster analysis, variable ranking was computed in two ways and assessed using the nonparametric Kruskal-Wallis test. Similarly, in the predictive analysis three different foldings (number of folds and how they are created) were used and Fleiss κ was then applied to the results of five iterations of the overall algorithm to study its stability. Thus, locally (for each method separately), we have made serious and versatile attempts to vary the meta-parametrization of the approaches and reported all the results. Globally, on the whole analysis level, we have again based our overall conclusions on the results and conclusions of the three methods of different orientations in EDM. We reason that such two-level treatment, where locally and globally the same results and their interpretation are supported by different approaches, improves the technical soundness of the study. Furthermore, the method for obtaining the final ranking, in clustering and in the MLP analysis, is novel and establishes a practical framework that can be used in similar applications.

ACKNOWLEDGEMENT

The unknown reviewers are gratefully acknowledged; their suggestions and comments led to significant improvement in the original manuscript. Finally, we thank Sami Äyrämö for enlightening discussions on the clustering.

REFERENCES

ALDAHDOOH, R. T. AND ASHOUR, W. 2013. Dimk-means distance-based initialization method for k-means clustering algorithm. *International Journal of Intelligent Systems and Applications (IJISA) 5,* 2, 41.

APOSTOL, T. M. 1969. *Calculus, Volume 2: Multi-variable Calculus and Linear Algebra with Applications to Differential Equations and Probability*. Wiley.

ÄYRÄMÖ, S. 2006. *Knowledge Mining Using Robust Clustering*. Jyväskylä Studies in Computing, vol. 63. University of Jyväskylä.

BAI, L., LIANG, J., AND DANG, C. 2011. An initialization method to simultaneously find initial cluster centers and the number of clusters for clustering categorical data. *Knowledge-Based Systems 24,* 6, 785–795.

BAI, L., LIANG, J., DANG, C., AND CAO, F. 2012. A cluster centers initialization method for clustering categorical data. *Expert Systems with Applications 39,* 9, 8022–8029.

BAKER, R. ET AL. 2010. Data mining for education. *International Encyclopedia of Education 7*, 112–118.

BAKER, R. S. AND YACEF, K. 2009. The state of educational data mining in 2009: A review and future visions. *Journal of Educational Data Mining 1,* 1, 3–17.

BARTLETT, P. L. 1998. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *Information Theory, IEEE Transactions on 44,* 2, 525–536.

BATISTA, G. AND MONARD, M. C. 2003. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence 17*, 519–533.

BAYER, J., BYDZOVSKÁ, H., GÉRYK, J., OBŠIVAC, T., AND POPELINSKỲ, L. 2012. Predicting drop-out from social behaviour of students. In *Educational Data Mining 2012*. 103–109.

BHARDWAJ, B. AND PAL, S. 2011. Mining educational data to analyze students' performance. *(IJCSIS) International Journal of Computer Science and Information Security, 9,* 4.

BOUCHET, F., KINNEBREW, J. S., BISWAS, G., AND AZEVEDO, R. 2012. Identifying students' characteristic learning behaviors in an intelligent tutoring system fostering self-regulated learning. In *Educational Data Mining 2012*. 65–72.

BRADLEY, P. AND FAYYAD, U. 1998. Refining initial points for k-means clustering. In *ICML*. Vol. 98. 91–99.

BRYMAN, A. 2003. Triangulation. *The Sage encyclopedia of social science research methods. Thousand Oaks, CA: Sage*.

CALDERS, T. AND PECHENIZKIY, M. 2012. Introduction to the special section on educational data mining. *ACM SIGKDD Explorations Newsletter 13,* 2, 3–6.

CAMPAGNI, R., MERLINI, D., AND SPRUGNOLI, R. 2012. Analyzing paths in a student database. In *Educational Data Mining 2012*. 208–209.

CARLSON, R., GENIN, K., RAU, M., AND SCHEINES, R. 2013. Student profiling from tutoring system log data: When do multiple graphical representations matter? In *Educational Data Mining 2013*. 12–20.

CHANDRA, E. AND NANDHINI, K. 2010. Knowledge mining from student data. *European Journal of Scientific Research 47,* 1, 156–163.

CHEN, L., CHEN, L., JIANG, Q., WANG, B., AND SHI, L. 2009. An initialization method for clustering high-dimensional data. In *Database Technology and Applications, 2009 First International Workshop on*. IEEE, 444–447.

CROUX, C., DEHON, C., AND YADINE, A. 2010. The k-step spatial sign covariance matrix. *Adv Data Anal Classif 4*, 137–150.

DENZIN, N. 1970. Strategies of multiple triangulation. *The research act in sociology: A theoretical introduction to sociological method*, 297–313.

DIMOPOULOS, Y., BOURRET, P., AND LEK, S. 1995. Use of some sensitivity criteria for choosing networks with good generalization ability. *Neural Processing Letters 2,* 6, 1–4.

EMRE CELEBI, M., KINGRAVI, H. A., AND VELA, P. A. 2012. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*.

ERDOGAN, S. AND TYMOR, M. 2005. A data mining application in a student database. *Journal Of Aeronautics and Space Technologies 2*, 53–57.

FAYYAD, U., PIATESKY-SHAPIRO, G., AND P., S. 1996. Extracting useful knowledge from volumes of data. *Communications of the ACM 39,* 11, pp. 27–34.

FLEISS, J. L. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin 76,* 5, 378–382.

GEVREY, M., DIMOPAULOS, I., AND LEK, S. 2003. Review and comparison of methods to study the contribution of variables in artificial neural network models. *Ecological Modelling 160*, 249–264.

HAGAN, M. T. AND MENHAJ, M. B. 1994. Training feedforward networks with the Marquardt algorithm. *IEEE Trans. Neural Networks 5*, 989–993.

HALONEN, P. 2012. Tietotekniikan laitos. 2. TIETOTEKNIIKKA 12a- valintasyyt- opetuksen laatumielipiteet.pdf.

HAN, J., KAMBER, M., AND TUNG, A. 2001. Spatial clustering methods in data mining: A survey. *Data Mining and Knowledge Discovery*.

HARDEN, T. AND TERVO, M. 2012. Informaatioteknologian tiedekunta. 1. ITK 4- opinnoista suoriutuminen.pdf.

HARPSTEAD, E., MACLELLAN, C. J., KOEDINGER, K. R., ALEVEN, V., DOW, S. P., AND MYERS, B. A. 2013. Investigating the solution space of an open-ended educational game using conceptual feature extraction. In *Educational Data Mining 2013*. 51–59.

HAWKINS, W., HEFFERNAN, N., WANG, Y., AND BAKER, R. S. 2013. Extending the assistance model: Analyzing the use of assistance over time. In *Educational Data Mining 2013*. 59–67.

HETTMANSPERGER, T. P. AND MCKEAN, J. W. 1998. *Robust nonparametric statistical methods*. Edward Arnold, London.

HOLLANDER, M., WOLFE, D. A., AND CHICKEN, E. 2013. *Nonparametric statistical methods*. Vol. 751. John Wiley & Sons.

HORNIK, K., STINCHCOMBE, M., AND WHITE, H. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks 2*, 359–366.

HUANG, G. B. 2003. Learning capability and storage capacity of two-hidden-layer feedforward networks. *Neural Networks, IEEE Transactions on 14,* 2, 274–281.

HUBER, P. J. 1981. *Robust Statistics*. John Wiley & Sons Inc., New York.

JAIN, A. K. 2010. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters 31,* 8, 651–666.

JERKINS, J. A., STENGER, C. L., STOVALL, J., AND JENKINS, J. T. 2013. Establishing the Impact of a Computer Science/Mathematics Anti-symbiotic Stereotype in CS Students. *Journal of Computing Sciences in Colleges 28,* 5 (May), 47–53.

JICK, T. D. 1979. Mixing qualitative and quantitative methods: Triangulation in action. *Administrative science quarterly 24,* 4, 602–611.

JOHN, G. H., KOHAVI, R., AND PFLEGER, K. 1994. Irrelevant features and the subset selection problem. In *Proceedings of the 11th International Conference on Machine Learning*. 121–129.

KÄRKKÄINEN, T. 2002. MLP in layer-wise form with applications in weight decay. *Neural Computation 14*, 1451–1480.

KÄRKKÄINEN, T. 2014. Feedforward Network - With or Without an Adaptive Hidden Layer. *IEEE Transactions on Neural Networks and Learning Systems*. In revision.

KÄRKKÄINEN, T. AND ÄYRÄMÖ, S. 2005. On computation of spatial median for robust data mining. *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems, EUROGEN, Munich*.

KÄRKKÄINEN, T. AND HEIKKOLA, E. 2004. Robust formulations for training multilayer perceptrons. *Neural Computation 16*, 837–862.

KÄRKKÄINEN, T., MASLOV, A., AND WARTIAINEN, P. 2014. Region of interest detection using MLP. In *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning - ESANN 2014*. 213–218.

KÄRKKÄINEN, T. AND TOIVANEN, J. 2001. Building blocks for odd–even multigrid with applications to reduced systems. *Journal of computational and applied mathematics 131,* 1, 15–33.

KERR, D. AND CHUNG, G. 2012. Identifying key features of student performance in educational video games and simulations through cluster analysis. *Journal of Educational Data Mining 4,* 1, 144–182.

KHAN, S. S. AND AHMAD, A. 2013. Cluster center initialization algorithm for k-modes clustering. *Expert Systems with Applications*.

KINNUNEN, P., MARTTILA-KONTIO, M., AND PESONEN, E. 2013. Getting to know computer science freshmen. In *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*. Koli Calling '13. ACM, New York, NY, USA, 59–66.

KOHAVI, R. 1995. Study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'95)*. 1137–1143.

KOHAVI, R. AND JOHN, G. H. 1997. Wrappers for feature subset selection. *Artificial Intelligence 97*, 273–324.

KOTSIANTIS, S. 2012. Use of machine learning techniques for educational proposes: a decision support system for forecasting students grades. *Artificial Intelligence Review 37,* 4, 331–344.

MEILĂ, M. AND HECKERMAN, D. 1998. An experimental comparison of several clustering and initialization methods. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 386–395.

MENDEZ, G., BUSKIRK, T., LOHR, S., AND HAAG, S. 2008. Factors associated with persistence in science and engineering majors: An exploratory study using classification trees and random forests. *Journal of Engineering Education 97,* 1.

PINKUS, A. 1999. Approximation theory of the MLP model in neural networks. *Acta Numerica*, 143–195.

RICE, W. R. 1989. Analyzing tables of statistical tests. *Evolution 43,* 1, 223–225.

ROUSSEEUW, P. J. AND LEROY, A. M. 1987. *Robust regression and outlier detection*. John Wiley & Sons Inc., New York.

RUBIN, D. B. 1976. Inference and missing data. *Biometrika 63,* 3, 581–592.

RUBIN, D. B. AND LITTLE, R. J. 2002. Statistical analysis with missing data. *Hoboken, NJ: J Wiley & Sons*.

RUCK, D. W., ROGERS, S. K., AND KABRISKY, M. 1990. Feature selection using a multilayer perceptron. *Neural Network Computing 2,* 2, 40–48.

SAARELA, M. AND KÄRKKÄINEN, T. 2014. Discovering Gender-Specific Knowledge from Finnish Basic Education using PISA Scale Indices. In *Educational Data Mining 2014*. 60–68.

SAHAMI, M., DANYLUK, A., FINCHER, S., FISHER, K., GROSSMAN, D., HAWTHRONE, E., KATZ, R., LEBLANC, R., REED, D., ROACH, S., CUADROS-VARGAS, E., DODGE, R., KUMAR, A., ROBINSON, B., SEKER, R., AND THOMPSON, A. 2013a. Computer science curricula 2013.

SAHAMI, M., ROACH, S., CUADROS-VARGAS, E., AND LEBLANC, R. 2013b. ACM/IEEE-CS Computer Science Curriculum 2013: Reviewing the Ironman Report. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. ACM, New York, USA, 13–14.

SAN PEDRO, M. O. Z., BAKER, R. S., BOWERS, A. J., AND HEFFERNAN, N. T. 2013. Predicting college enrollment from student interaction with an intelligent tutoring system in middle school. In *Educational Data Mining 2013*. 177–184.

SHOJAEEFARD, M. H., AKBARI, M., TAHANI, M., AND FARHANI, F. 2013. Sensitivity analysis of the artificial neural network outputs in friction stir lap joining of aluminum to brass. *Advances in Material Science and Engineering 2013*, 1–7.

SPRINGER, A., JOHNSON, M., EAGLE, M., AND BARNES, T. 2013. Using sequential pattern mining to increase graph comprehension in intelligent tutoring system student data. In *Proceeding of the 44th ACM technical symposium on Computer science education*. ACM, 732–732.

STEINBACH, M., ERTÖZ, L., AND KUMAR, V. 2004. The challenges of clustering high dimensional data. In *New Directions in Statistical Physics*. Springer, 273–309.

TAMURA, S. AND TATEISHI, M. 1997. Capabilities of a four-layered feedforward neural network: Four layers versus three. *IEEE Transactions on Neural Networks 8,* 2, 251–255.

VALSAMIDIS, S., KONTOGIANNIS, S., KAZANIDIS, I., THEODOSIOU, T., AND KARAKOS, A. 2012. A clustering methodology of web log data for learning management systems. *Educational Technology & Society 15,* 2, 154–167.

VIHAVAINEN, A., LUUKKAINEN, M., AND KURHILA, J. 2013. Using students' programming behavior to predict success in an introductory mathematics course. In *Educational Data Mining 2013*. 300–303.

XU, R. AND WUNSCH, D. C. 2005. Survey of clustering algorithms. *IEEE Transactions on Neural Networks 16,* 3, 645–678.

ZHONG, C., MIAO, D., WANG, R., AND ZHOU, X. 2008. Divfrp: An automatic divisive hierarchical clustering method based on the furthest reference points. *Pattern Recognition Letters 29,* 16, 2067–2077.