

# Using the Random Forest Classifier to Assess and Predict Student Learning of Software Engineering Teamwork

Dragutin Petkovic<sup>1</sup>, Marc Sosnick-Pérez<sup>1</sup>, Kazunori Okada<sup>1</sup>, Rainer Todtenhoefer<sup>3</sup>, Shihong Huang<sup>2</sup>,  
Nidhi Miglani<sup>1</sup>, Arthur Vigil<sup>1</sup>

<sup>1</sup>Department of Computer Science  
San Francisco State University  
San Francisco, U.S.A.  
{petkovic, msosnick, kazokada,  
nidhimig, ahvigil}@sfsu.edu

<sup>2</sup>Department of Computer Science and  
Engineering  
Florida Atlantic University  
Boca Raton, U.S.A.  
shihong@fau.edu

<sup>3</sup>Department of Applied Computer Science  
University of Applied Science, Fulda  
Fulda, Germany  
rainer.todtenhoefer@informatik.hs-fulda.de

**Abstract**—The overall goal of our Software Engineering Teamwork Assessment and Prediction (SETAP) project is to develop effective machine-learning-based methods for *assessment* and early *prediction* of student learning effectiveness in software engineering teamwork. Specifically, we use the Random Forest (RF) machine learning (ML) method to predict the effectiveness of SE teamwork learning based on data collected during student team project development. These data include over 100 objective and quantitative Team Activity Measures (TAM) obtained from monitoring and measuring activities of student teams during the creation of their final class project in our joint SE classes which ran concurrently at San Francisco State University (SFSU), Fulda University (Fulda) and Florida Atlantic University (FAU). Although we have previously published RF analysis on a very limited data set, in this paper we provide the first RF analysis results done at SFSU on our full data set covering four years of our joint SE classes. These data include 74 student teams with over 350 students, totalling over 30000 discrete data points. These data are grouped into 11 time intervals, each measuring important phase of project development during the class (e.g. early requirement gathering and design, development, testing and delivery). In this paper we briefly elaborate on the methods of data collection and description of the data itself. We then show prediction results of the RF analysis applied to this full data set. Results show that we are able to detect student teams who are bound to fail or need attention in early class time with good (about 70%) accuracy. Moreover, the variable importance analysis shows that the features (TAM measures) with high predictive power (highly ranked by RF) make intuitive sense and even pointed us to measurements we did not originally expect to have high predictive importance, such as statistics on comments to source code postings. These measures can be used to guide educators and software engineering managers to ensure early intervention for teams bound to fail. This research is funded in part by NSF TUES Grant # 1140172.

**Keywords**—Assessment; software engineering teamwork; machine learning; education

## I. INTRODUCTION

There is now a consensus across industry and academia that

---

This research was funded in part by NSF TUES Grant #1140172

to be successful in today's workplace, computer science students and software engineers must learn and practice effective *software engineering teamwork skills*. This need is evidenced by the unacceptably high incidence of failures of software projects in industry: about 9% are abandoned, about one third fail, and over half experience cost and schedule overruns. These project failures apparently stem from failures in communication, organization and teamwork aspects of software engineering [1-6, 25]. The emergence of global software development projects utilizing geographically distributed teams adds significant difficulty to overcoming these failure points. For the education community, though it is clear where the problem lies, little is known about the factors that influence actual student learning of software engineering teamwork skills or about how to objectively and quantitatively *assess*, *monitor* and *predict* student progress in the acquisition of these skills. This knowledge, especially the knowledge of the factors that most influence or best predict learning effectiveness, will enable educators to better and more efficiently assess and improve software engineering education and classroom practice and apply early classroom intervention when necessary. For industry, this knowledge will benefit project managers to improve software engineering project management.

The Software Engineering Teamwork Assessment and Prediction (SETAP) project, led by San Francisco State University (SFSU) with collaborators at Fulda University, Germany (Fulda) and Florida Atlantic University (FAU), addressing this need by using the Random Forest (RF) [18, 20] machine learning (ML) classification method for *assessment*, *prediction*, and most importantly *discovery of factors determining the prediction* of learning effectiveness of software engineering teamwork in the educational setting [13-17, 30]. In this research the effectiveness of learning software engineering teamwork is defined as an ability of a *student team*: (i) to learn and effectively apply *software engineering processes* in a teamwork setting, and (*process* component) (ii) to work well in developing satisfactory software product (*product* component).

ML has been used in education for other similar applications such as predicting student dropout rate, teaching effectiveness, grades etc. Machine learning can be used in education to discover models that can help in understanding or predicting some aspects of educational situations, to provide some characterization of the teaching or learning process, or to assist in generating tools for education [10, 30]. ML techniques are applied on subjective (e.g. surveys) or objective (e.g. student age) data extracted from an educational environment. For example, a class in a semester may yield data of: the demographics of students, survey responses of the students, registration and academic data, student activity and grades. Paired with independently obtained outcome assessments, these data constitute so-called “training databases”. ML systems are then trained on those training databases, and tested in terms of their ability to correctly predict or mimic independently obtained outcomes on variables under investigation such as grading, dropout rate, learning achievement etc. [7-12, 26, 30]. Though ML methods (often RF) have been applied to software engineering [27-28], we are not aware of any major work using ML to predict teamwork learning outcomes in software engineering.

In this paper we provide first RF teamwork prediction and factor analysis results on our full data set which covers over 4 years of our joint software engineering classes, conducted from Fall 2012 through Fall 2015. These classes constitute 74 student teams of over 350 students. We obtained over 30000 discrete data items used to create our ML training database. We briefly elaborate on the methods of data collection and description of the data, and then show prediction accuracy results of RF analysis applied to this full data set together with ranking of team activity measures (TAMs) offering the most predictive power.

## II. SETAP PROJECT DATA COLLECTION AND THE CREATION OF THE MACHINE LEARNING TRAINING DATABASE

SETAP data are obtained from a joint software engineering class taught concurrently at SFSU, Fulda and FAU, where student teams at all three schools are “embedded and observed” in as realistic project and teamwork development environment as possible, thus providing a rich learning environment for students and more realistic data for researchers. The class now involves about 140 students each year, working in 25-30 teams of 5-6 students each. *Local* student teams are composed of students from the same university, and *global student teams* are composed of volunteer students from multiple—usually two—universities. Each student team develops the same software application. The semester is divided into five formally managed milestones, M1 through M5, which are synchronized across all three schools (Table I)

All student teams use the same software development and communication tools (source code management, development and deployment software and servers), which are hosted on an Amazon Web Service cloud instance, and managed by the SFSU team. Details about organization and data collection in our joint SE class have been reported earlier [13-17]. Data collection and analysis is done at SFSU.

TABLE I. STUDENT PROJECT MILESTONE DESCRIPTIONS

Milestone	Description
M1	high level requirements and specs
M2	detailed requirements and specs
M3	prototype development;
M4	beta launch
M5	final delivery and demo

The SETAP *ML training database* used to train and develop the RF predictive model is a critical component of the project. The most time consuming tasks in this project were creating, curating and maintaining this database. This forced us to pay the utmost attention and significant resources to ensuring data accuracy and validity. The final outcome of this work is a reliable training database that will be available to researchers at <http://setaproject.org>.

The data are organized by student teams and milestones, and comprise TAM data for each student team paired with separate evaluations of software engineering teamwork learning outcomes, one for software engineering *process* and one for software engineering *product*. These outcomes, for the purposes of this research, are categorized in two ML classes: "A", represents student work at or above expectations, and "F", represents student work below expectation or needing attention. The grades A and F are therefore considered ML class labels whose prediction we are aiming for. Note that these grades/labels are a different from student class grades and are derived by applying cutoffs to student team percentage grades for process and product. More details of grading, including rubrics for our software engineering class are available in [13]. To protect student privacy, the ML training database contains no individually identifiable student information.

To focus our analysis only on factors influencing team success exhibited during the class and minimize the influence of an individual student’s experience and skills developed prior to the class, student teams were formed with approximately the same overall combination of skills and experience. We form student teams by using a team placement survey with about 20 questions about student experience, and a simple programming test, which we then analyze. The analysis provides the skill criteria that are then used to create teams such that the skill profile in each team is approximately equal. Individual student skills and experiences are not included in TAMs.

Team leads are chosen from a volunteer pool of students in the class. Potential team leads are briefly interviewed and chosen by the instructor. Teams must approve of the instructors' choice of team lead before final appointment. More details about our software engineering class management is in [13].

There are several issues of possible bias that we had to address. In order to reduce inherent bias, where instructors grade and at the same time try to use ML to predict grades, two techniques were used. For software engineering product grading we involve reviewers external to the class, usually two. Also, grading rubrics have been devised that in general have more and different items from what we measure in TAMs.

TAM data consists of aggregated individual student activity measures (SAM) from each team. SAM and TAM data are collected use several methods such as:

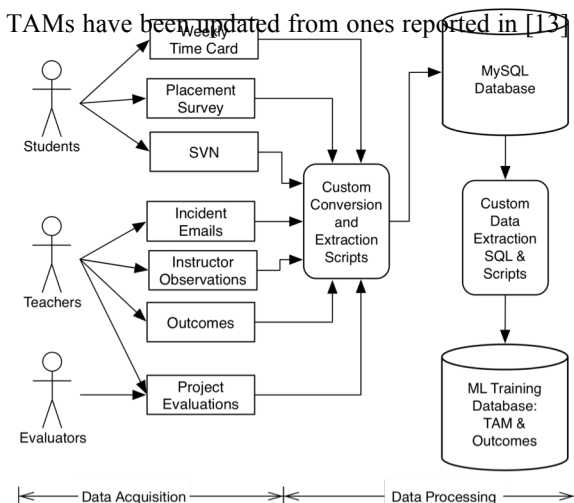
- a) *Weekly Timecard Surveys (WTS)*: these mandatory surveys collect information from each student about the time spent during the week on coding, meetings, teamwork, etc.;
- b) *Tool Logs (TL)*: comprise the collected statistics of individual student usage of software engineering communication and development tools such as code repository; and
- c) *Instructor Observation (IO)*: logs of team activity such as team member participation, the number of issues requiring instructor intervention, number and percent of issues closed late, etc. [13].

TABLE II. TIME INTERVAL TO MILESTONE CORRESPONDENCE

Time Interval	Corresponding Milestone
T1	M1
T2	M2
T3	M3
T4	M4
T5	M5
T6	M1 – M2
T7	M1 – M3
T8	M1 – M4
T9	M1 – M5
T10	M3 – M4
T11	M3 – M5

The ML analysis is performed on different time intervals, numbered T1-T11 (Table II), which correspond to the five predefined milestones M1-M5 times and groupings of them. Grouped milestones are intended to find different trends and dynamics during the lifecycle of the student projects. For example, T6 corresponds to M1 and M2 – covering early high-level requirements through detailed specs, or T11 which covers M3–M5 covering implementation, testing and delivery. ML analysis is applied separately to each time interval. Special focus was placed on interpretation of early time intervals (T1, T2, T3, T6) due to our goal of early prediction.

Fig 1. Setap Data Collection and Processing Flow



SETAP data collection and processing is described in Fig 1.

For each team, 115 TAMS were calculated from SAMs for every time interval. Many of these TAMs are averages and standard deviations derived from core values, such as hours spent in person in meetings, number of commits, etc, over intervals of weeks, student, or time interval. We list here only the core values. Full TAM information is available at the project website <http://setaproject.org>.

General TAMs:

- Year,
- semester,
- timeInterval,
- teamNumber,
- semesterId,
- teamMemberCount,
- femaleTeamMembersPercent,
- teamLeadGender,
- teamDistribution

Weekly Time Cards (WTS) TAMs:

- teamMemberResponseCount,
- meetingHours,
- inPersonMeetingHours,
- nonCodingDeliverablesHours,
- codingDeliverablesHours,
- helpHours,
- globalLeadAdminHours,
- LeadAdminHoursResponseCount,
- GlobalLeadAdminHoursResponseCount

Tool Logs (TL) TAMs:

- commitCount,
- uniqueCommitMessageCount,
- uniqueCommitMessagePercent,
- CommitMessageLength

Instructors' Observations (IO) TAMs:

- issueCount,
- onTimeIssueCount,
- lateIssueCount

For each of the TAMs “core” variables e.g. CommitCount, where applicable we compute several TAMs. This is done, for example, by computing average for the week separately by the team and then by the student, then by computing of standard deviation over weekly and student averages in each team (the latter serving to show dynamics of intergroup participation). The core variable name then gets labels pre-and-post-pended consistently, by use of formal naming grammar, to reflect specific aggregation method and statistical measures. For example variable *CommitCount* is aggregated by week for each student to become *CommitCountByWeek* then its average and standard deviation are computed to yield final TAM variables *standardDeviationCommitCountByWeek*. These names are made to be easily read by humans and are consistently used in all data fields, documentations and in final training data files.

To complete the ML training DB, TAMs for each team are paired with two ML class labels, one for software engineering

Process (A or F) and one for software engineering Product (A or F). Our goal is to try to predict occurrences of F for software engineering process and software engineering Product especially in early stages of the class (time periods T1, T2, T3, T6).

To implement creation of ML training database we developed a complex data collection infrastructure. At its heart a master SETAP MySQL relational DB that first collects all raw (SAM) data from various sources (WTS, TL, IO). Dedicated scripts extract data from weekly on-line time card surveys (WTS) and tool logs (TL) databases. Some values such as instructors' observations (IO) are entered manually from paper forms used in the class. To aggregate SAM into TAMs for each team and in each desired time interval, we created dedicated SQL code using its statistical functions. Extracted TAMs are paired with class labels A and F and stored back in the master SETAP DB as a final training database table. A custom Python script then exports training database data for chosen time interval into CSV files ready to be used by ML analysis software. Each of these CSV files has extensive human-readable header information automatically generated for data provenance and management. Finally, ML analysis uses the randomForest machine learning package for the R statistical mathematics program to perform RF analysis ([19]).

Guided by our experience and the complexity of data collection, as well as criticality of good ML training DB, we decided to pay utmost attention to data accuracy and validity. This was achieved by several software engineering "best practices" including: a) testing of all data gathering, aggregation and extraction software with real and synthetic data; b) manual spot checking of data by two independent researchers; c) dealing with NULL or missing data in appropriate ways (some records are dropped, some are handled by appropriate statistics and some are imputed based on specific ways variables were extracted). In addition, given that we also intend to disseminate our training database for others to use, we designed extensive human-readable documentation integrated with the files themselves as file headers for documentation, ease of management and data provenance.

The current training data is collected from 74 student teams from Fall 2012 through Fall 2015 from our joint software engineering classes. This data involves 383 Students and 18 class sections. For each team 115 TAM measures have been aggregated from related SAM measures. Total number of grades for software engineering Process were 49 As and 25 Fs, and for software engineering Product 42 As and 32 Fs. For each team we collected about 400 data items (student team selection survey, time cards, deliverable tracking, grading of outcomes etc.), hence our training DB involves about 30000 data points. In two semesters, for T1 and T4 intervals we had to drop some teams do to missing time card surveys.

### III. USE OF RANDOM FOREST TO DETERMINE FACTORS THAT PREDICT STUDENT LEARNING EFFECTIVENESS OF SOFTWARE ENGINEERING TEAMWORK

We use RF [18,20] as our ML technology, which we have also tested successfully on other applications [24], and we designed experiments to be consistent with our other experiences in using ML for bioinformatics [23]. RF is an ensemble classifier, consisting of a set of CART (decision tree)

classifiers, each of which is generated by the Bagging algorithm [18]. To train a RF, two parameters, the number of CARTs (*n<sub>tree</sub>*) and the number of randomly selected features used to evaluate at each CART node (*m<sub>try</sub>*), must be supplied, as well as a training database with ground-truth class labels. RF also allows adjustment of the voting threshold or *cutoff* (fraction of trees needed to vote for a given class), which we have exploited in this study.

One of the RF algorithm's strengths, and reasons we chose it, is its ability to calculate the variable importance (VI) measure, namely Mean Decrease Gini (MDG), to determine rank of each RF input variable (in our case TAM) based on its contribution to the RF prediction [18, 20]. MDG represents variable-wise information gain averaged over all decision trees included in a RF classifier. During the RF's training, each CART is built by iteratively expanding a tree node by selecting the best single variable thresholding function to split training data to gain most information. This information gain is quantified by decrease of Gini impurities between before and after the data split. This Gini decrease is then associated with the variable chosen for that node. When a tree building is completed, The Gini decreases from all tree nodes are aggregated for each variable. These variable-wise aggregated Gini decreases are then averaged over the trees, yielding the MDG measures. Finally variables are ranked according to MDG values indicating their importance to RF prediction/classification.

The accuracy estimate built into the RF algorithm and all its software implementations, and recommended by inventors of RF [18] is called Out of Bag Error (OOB), which measures the average misclassification ratio. We augment our report by also computing recall and precision for our target class F, accuracy (1 – OOB), and confusion matrices.

In order to implement RF for our study, after evaluation, we chose statistical package R [19,21,22] namely its *randomForest* package.

## IV. EXPERIMENTS AND RESULTS

Our experiments and specific questions we seek to answer are twofold:

a) *Determine Prediction Accuracy*: How accurate are we in predicting software engineering process and software engineering product class labels, specifically in the target class F? In which time intervals is the best accuracy achieved? This prediction accuracy (OOB, accuracy, recall and precision for F) are estimated by performing RF training and accuracy estimation using R package by varying the RF parameters as follows: *n<sub>tree</sub>* = 1000; *m<sub>try</sub>* = {5,10,20,30} and by varying voting cutoff threshold as {10%, 20%, 25%, 30%, 35%, 40%, 50%} to adjust optimal RF sensitivity for our needs e.g. favoring F class detection. This is repeated for software engineering Product and software engineering Product and for each time interval T1, T2, T3, T6, T9 and T11. We also repeated each experiment several times with different random seed, observing only very minimal changes. Results are shown in Tables III – V.

b) *Discover factors that contribute to prediction:* For the above optimal RF predictive models (e.g. operating points with maximal accuracy) we compute best ranked TAM variables using Gini measures as provided by R package, and investigate if they have an intuitive explanation based on instructors' or any other experience. These factors (e.g. top ranked TAMs) can serve as a guidance to practitioners. Results are shown in Table VI and Table VII.

TABLE III ACCURACY RESULTS FOR SOFTWARE ENGINEERING PROCESS AND PRODUCT.

Teamwork Component	Time Interval with best prediction	OOB (ntree, mtry, cutoff)	Overall Accuracy	Recall for F	Precision for F
SE Process	T2	0.30 (1000,20, 35%)	0.7	0.76	0.54
SE Product	T3	0.29 (1000, 30, 40%)	0.71	0.81	0.61

TABLE IV. CONFUSION MATRIX FOR SE PROCESS T2.

SE process for T2	Predicted A	Predicted F
True A	33	16
True F	6	19

TABLE V. CONFUSION MATRIX FOR SE PRODUCT T3.

SE product for T3	Predicted A	Predicted F
True A	26	16
True F	6	26

TABLE VI. TOP RANKED TAM MEASURES BY GINI, FOR SOFTWARE ENGINEERING PROCESS FOR RF BEST PREDICTION PARAMETERS: TIME INTERVAL = T2

TAM Name	GINI
lateIssueCount	3.86
issueCount	1.05
standardDeviationHelpHoursTotalByWeek	1.05
averageHelpHoursTotalByWeek	1.04
standardDeviationHelpHoursAverageByWeek	.08
codingDeliverablesHoursAverage	0.79
standardDeviationMeetingHoursAverageByWeek	0.75
helpHoursStandardDeviation	0.71

TABLE VII. TOP RANKED TAM MEASURES BY GINI, FOR SOFTWARE ENGINEERING PRODUCT FOR RF BEST PREDICTION PARAMETERS: TIME INTERVAL = T3

TAM Name	GINI
averageUniqueCommitMessageCountByWeek	2.11
uniqueCommitMessageCount	1.35
commitMessageLengthStandardDeviation	1.17
standardDeviationInPersonMeetingHoursAverageByWeek	1.05
standardDeviationCodingDeliverablesHoursAverageByWeek	0.98
standardDeviationUniqueCommitMessagePercentByWeek	0.94
standardDeviationCodingDeliverablesHoursTotalByWeek	0.89
standardDeviationNonCodingDeliverablesHoursAverageByStudent	0.88
averageHelpHoursAverageByStudent	0.87
standardDeviationMeetingHoursAverageByWeek	0.83

## V. DISCUSSION

We have contributed toward demonstrating the success of ML (namely RF) to predict the teams that are likely to fail in software engineering educational context. We also show that this can be done based on easy to measure objective and quantitative variables, and can be done *early* in the class or project which offers great advantages. Moreover, we show that factors contributing to these predictions are intuitive and offer practical guidance to teachers and managers in software engineering.

More work remains in deeper understanding of why RF works, an issue we call *explainability*. This can be formulated in a number of questions a practitioner (non ML expert) might ask: What would be hit on accuracy if we use much less TAMs so I can reduce the cost of applying this approach? How exactly top ranked variables contribute: e.g. does more helping time or less helping time indicate good team? What variables most often mutually interact to produce correct decisions? We are actively engaged in this work.

We also cannot overemphasize the importance of proper data collection, data management and testing, which took a lot of our effort and required utmost focus.

Finally, in near future we will be disseminating our training database with full documentation so others can try their own ML approaches on it.

## VI. ACKNOWLEDGEMENTS

Special thanks to Dr. Byron Dom for his advice with machine learning.

## VII. REFERENCES

- [1] "Standish Group Report: CHAOS Summary 2009." Internet: [http://www1.standishgroup.com/newsroom/chaos\\_2009.php](http://www1.standishgroup.com/newsroom/chaos_2009.php) [May 18, 2011].
- [2] C. Jones, Software Engineering best Practices: Lessons from Successful Projects in the Top Companies. McGraw Hill, 2010, ISBN 978-0-07-162161-8.
- [3] J. Reel, "Critical success factors in software projects," IEEE Software 16(3). 1999. pp. 18-23.
- [4] R. N. Charette, "Why software fails," IEEE Spectrum 42(9). September 2005, p. 42.
- [5] R. Pressman, Software Engineering: A Practitioner's Approach, Sixth Edition, McGraw Hill, 2005.
- [6] B. Curtis, H. Krasner, N. Iscoe, "A field study of the software design process for large systems." Communications of the ACM 31(11), 1988, pp 1268-1287.
- [7] S. B. Kotsiantis: "Use of machine learning techniques for educational purposes: a decision support system for forecasting student grades" Artif Intell Rev (2012) 37:331-344
- [8] I. Lykourantzou, I. Giannoukos, V. Nikolopoulos, G. Mpardis, V. Loumos: "Dropout Prediction in e-learning courses through the combination of machine learning techniques", Computers & Education, 53 (2009) 950-965, Elsevier
- [9] F. Castro, A. Vellido, A. Nebot, F. Mugica: "Applying Data Mining Techniques to e-Learning Problems", Studies in Computational Intelligence (SCI) 62, 183-221 (2007)
- [10] R. Baker, K. Yacef: "The State of Educational Data Mining in 2009: A Review and Future Visions", Journal of Educational Data Mining, Article 1, Vol1, No. 1, Fall 2009

- [11] M. Baker: "The roles of models in Artificial Intelligence and Education Research: A Perspective View", *Int. Journal of Artificial Intelligence in Education* 11 (2000), 122-143
- [12] L. P. Macfayden, S. Dawson: "Mining LMS data to develop an "early warning system" for educators: a proof of concept", *Comput Educ* 54: 588-599 (2010)
- [13] D. Petkovic, M. Sosnick-Pérez, S. Huang, R. Todtenhoefer, K. Okada, S. Arora, et. al.: "SETAP: Software Engineering Teamwork Assessment and Prediction Using Machine Learning", *Proc. FIE2014, Madrid, Spain* 2014
- [14] D. Petkovic, R. Todtenhöfer, G. Thompson, "Teaching practical software engineering and global software engineering: case study and recommendations," *Proceedings of the 36th ASEE/IEEE Frontiers in Education Conference, San Diego, CA, 2006*, pp. 19–24.
- [15] D. Petkovic, G. Thompson, R. Todtenhöfer, "Assessment and comparison of local and global software engineering practices in a classroom setting," *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education, Madrid, Spain, June 2008*, pp. 78–82.
- [16] D. Petkovic, G. Thompson, R. Todtenhöfer, S. Huang, B. Levine, S. Parab, G. Singh, R. Soni, S. Shrestha : "Work in progress: e-TAT: online tool for teamwork and "soft skills" assessment in software engineering education," *Frontiers in Education (FIE) 2010, IEEE*. 27-30 Oct. 2010, pp. S1G-1-S1G-3,
- [17] D. Petkovic, K. Okada, M. Sosnick, A. Iyer, S. Zhu, R. Todtehoefer, S. Huang: "A Machine Learning Approach for Assessment and Prediction of Teamwork Effectiveness in Software Engineering Education", *Frontiers of Education FIE 2012, Seattle, WA, October 2012*
- [18] L. Breiman, "Random Forests," *Machine Learning* 45(1). 2001. pp. 5–32.
- [19] A. Liaw and M. Wiener (2002). *Classification and Regression by randomForest*. *R News* 2(3), 18--22. <http://cran.rproject.org/web/packages/randomForest/index.html> [October 22, 2013]
- [20] C. Chen, A. Liaw, L. Breiman, Leo, "Using Random Forest to Learn Imbalanced Data", Report ID: 666, UC Berkeley, July 2004
- [21] "The R Project for Statistical Computing." Internet: <http://www.rproject.org/> [June 24, 2012].
- [22] R Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. 2013. <http://www.R-project.org>
- [23] L. Buturović, M. Wong, G. W. Tang, R. B. Altman, D. Petković (2014) High Precision Prediction of Functional Sites in Protein Structures. *PLoS ONE* 2014 Feb 11; PONE-D-13-39539R1, EMID: b17311ed7a971c30
- [24] K. Okada, L. Flores, M. Wong, D. Petkovic, "Microenvironment-Based Protein Function Analysis by Random Forest", *Proc. ICPR - International Conference on Pattern Recognition, Stockholm, 2014*
- [25] *C.Duhig*: "What Google Learned From Its Quest to Build the Perfect Team", *NY Times*, Feb. 25, 2016
- [26] D. Delen: "A comparative analysis of machine learning techniques for student retention management", *Decision Support Systems, Volume 49, Issue 4*, November 2010, Pages 498–506
- [27] H. Malik at all: "Understanding the rationale for updating a function's comment", *IEEE Int. Conf. on Software Maintenance*, Oct 2008
- [28] E. Zhang, J. Tsai, ed., *Machine Learning Applications In Softwar Engineering*. Series on Software Engineering and Knowledge Engineering, Vol. 16. World Scientific, 2005.
- [29] Rajwinder Singh, Neeraj Mohan and Dr. Parvinder S. Sandhu: "Evaluation of Success of Software Reuse using Random Forest Algorithm", *International Conference on Artificial Intelligence and Embedded Systems (ICAIES'2012) July 15-16, 2012 Singapore*
- [30] Dragutin Petkovic: "Using Learning Analytics to Assess Capstone Project Teams", *IEEE Computer*, Issue No.01 - Jan. (2016 vol.49).