International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 10 (2015) pp. 26823-26836

http://www.ripublication.com

Generating Classification Rules by Applying Rough Set Theory on Pair Programming Data

R.K. Kavitha[1], M.S. Irfan Ahmed[2]
[1]Department of Computer Applications
Kumaraguru College of Technology, Coimbatore, India
kavitha.rk.mca@kct.ac.in
[2]Department of Computer Applications
Sri Krishna College of Engineering and Technology, Coimbatore, India
msirfan@gmail.com

Abstract

Agile methodologies are frequently used by many self organizing and cross functional teams across various organizations. Complemented by its unique, iterative and incremental methodology, it has significantly provided effective business and software solutions for various computing domains of recent times. Rough set theory is an intelligent technique used for the discovering data dependencies, data reduction, approximate set classification, and rule induction from databases. This paper reports the results of application of rough set method on pair programming data generated by pair programming exercises carried out with a set of fifty seven post graduate students, who developed small applications as a part of their software development laboratory course at Kumaraguru College of Technology (KCT) during the academic year 2013-2014. The objective of this research is to relate the students' opinion on various aspects on pair programming collected during pair programming sessions and their exam scores. Rough set analysis was carried out to deal with inconsistent data and with the intention of identifying student groups which requires special attention.

Keywords: Knowledge discovery, Rough sets, Agile Software Development, Pair Programming, Rule generation.

Introduction

The usage of agile software is on the rise among the software developers for its unique and proven methodology. Known for its success agile software has delivered requisite productivity and quality in software development. Holding out an immense potential to synergize tacit and explicit knowledge, it culminates conventional

practices and fosters proactive planning in a collaborative environment and provides rapid and flexible responses for the programmers. As one of renowned agile software development methods, Extreme Programming [17] focuses on disseminating knowledge through collaborative practices viz., pair programming, planning game and retrospectives. Pair Programming [PP] is a subset of such extreme programming practice, where two programmers mutually collaborate at the same workstation to acquire added knowledge and experience on day to day basis. The programmers collaborate as pairs by sharing a single computer working with the same design, algorithm, code, or test etc. While one member of the pair, namely the driver types at the computer or writes down a design, the other who assumes the role of the navigator, observes the work of the driver to ensure objectivity, logic and process flexibility.

Pair programming as a pedagogy offers quick and consistent learning in higher academia. Through its collaborative nature it engendered greater participation and better interaction among learners when compared to conventional programming methods. Studies also revealed that pair programming complemented the learning process commendably within a short period of time [15]. Further, it also helped students to gain real time practical experience of software development through knowledge sharing and collaboration.

The theory of rough sets is a mathematical tool for extracting knowledge from uncertain and incomplete data based information. The theory assumes that with necessary information or knowledge of objects in the universe, the objects can be divided into different groups. With exactly same information of two objects, it can be said that they are indiscernible. The theory of rough set can be used to find dependence relationship among data, evaluate the importance of attributes, discover the patterns of data, learn common decision-making rules, reduce all redundant objects and attributes and seek the minimum subset of attributes so as to attain satisfying classification. This paper discusses how rough set theory can be used to analysis pair programming data, and for generating classification rules from the collected data set. The analysis was based on data sources gathered from a post graduate computer applications course. The rough set reduction technique is applied to find all reducts of the data which contains the minimal subset of attributes that are associated with a class label for classification. This paper is organized as follows. Existing studies are discussed in Section 2. Details regarding the proposed work are introduced in Section 3. Experimental results and discussion are reported in Section 4. Finally, conclusion is discussed in Section 5.

Existing Studies

A comprehensive review of literature was done in order to understand the impact of pair programming exercise on teaching learning process. Literature reveals plenty of studies related to the effects of pair vs. solo programming. The ability to work as part of a cross-disciplinary team in industry has been highlighted by Scott, et al. [15]. Subbaraya Kuppusami, Kalimuthu Vivekanandan [13] experimented with computer science course students comparing the learning efficiency of students who adopted pair programming with those using traditional method for laboratory exercises for a

short duration. The metrics used for the study were design documents, completion time, and marks obtained in a written test. They also concluded that the adoption of pair programming improves design ability, reduces the time spent on a laboratory exercise and also increases both knowledge and programming skills of the pairs involved.

The cost-effectiveness of PP and the potential contained in the same for developing codes with a few errors have been demonstrated by Muller [12]. According to Lui & Chan [11], pair programming promotes not only quality programming skills, but also enhances responsibility, mentoring, teamwork in addition to providing an increased sense of enjoyment. Jari Vanhanen and Harri Korpi [10] demonstrated their experiences of using PP extensively in an industrial project. The results seem to show that test-driven development and design in pairs not only minimized defects but also improved both quality and knowledge transfer, thus proving their suitability for complex tasks.

An extensive and substantial case study on pair programming was carried out in software development courses at the University of Dortmund, Germany by Tanja Bipp and Andreas Lepper [8]. Thirteen software development teams with a total of 100 students took part in the experiments. The groups were as follows: In one set, the group members worked on their projects in pairs. Not only did these teams produce nearly the same number of codes as the teams of individual workers in the same period, but their codes were easier to read and understand thus facilitating easy detection and correction of errors. Research conducted by Begel [9] also brought to fore the fact that freshly inducted software developers often struggled to adequately communicate, when they were in need of assistance or while they were struggling with a problem. M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson and J. Still [4] studied the communication aspect of agile software development and concluded that agile practices improve both formal and informal communication among team members.

On reviewing 66 studies, Norsaremah Salleh [7] identified certain psychosocial factors such as compatibility, personality and gender issues, which affect the effectiveness of pair programming among students. The effects of pair programming on knowledge transfer and the resulting sense of fulfillment experienced by students were reported by V. Venkatesan and A. Sankar [5]. Jo E. Hannay, Erik Arisholm, Harald Engvik, and Dag I.K. Sjøberg [3] observed that the personality of the pairs engaged in pair programming could be a valid predictor for long-term team performance. However, no conclusive evidence regarding the effect of personality on pair programming was observed. Norsaremah Salleh, Emilia Mendes, and John C. Grundy [4] presented evidence related to the effectiveness of pair programming (PP) as a pedagogical tool in higher education CS/SE courses.

Ella Hassanien, Jafar M.H. Ali has applied rough set classification on breast cancer data and the study showed that the theory of rough sets seems to be a useful tool for inductive learning and a valuable aid for building expert systems [19].

Studies reported in literature mostly involved experiments conducted for a limited duration ranging from a few laboratory sessions to a few months. Further, only a few studies in the Indian educational context have been reported so far. The current study

aims to plug the gap by undertaking a controlled experiment and extending it to a longer duration (i.e.) a period of six-months. Not many studies in literature report work related to applying rough set theory on academic data. In this work rough set theory has been applied to collected data to deal with inconsistencies. This approach will help the academic practioners to identify student groups to whom they have to be given special attention and training so as to make them perform well in final examination.

Proposed Work

The significance of knowledge sharing through pair programming was felt when the students of the Master of Computer Applications (MCA) program struggled a lot initially while developing applications, owing to their heterogeneous academic backgrounds [1]. Based on the researcher's experiences and insights drawn from existing literature, the researcher proposed to study the effects and experiences of the pair programming concept. The objectives of the study are:

1. To relate the students opinion on various aspects on pair programming collected during pair programming sessions and their exam scores.
2. To study the interesting rules generated and to analyze the reasons for poor scores secured by students in model exam, in spite of the knowledge gained through pair programming. These rules would help the instructors predict the student's performance in their end semester exams of the laboratory course. The instructor can give extra coaching and care to these students to improve their results in their end semester examinations.

Experimental Methodology and Context

In order to facilitate learning process of students in the Computer Applications course, the study investigated the use of pair programming as a teaching methodology and investigated its effectiveness on students overall learning process.

Research Instrument

Formal lists of questions were prepared and the responses were analyzed using standard statistical techniques. Two questionnaires with close-ended questions containing a 5-point rating scale were designed. The students were made to fill an entry questionnaire consisting of ten questions to assess their level of exposure to programming tasks, partner preferences etc. The worksheet also contained twelve open-ended questions, which allow the students to provide their own answers in an unprompted manner, thus yielding qualitative data. After the completion of the project, an exit questionnaire containing twenty questions on knowledge sharing, tool learning, pair programming effectiveness during various phases of software development and general experiences on pair programming [Table 1] was administered to each student practicing PP. Also, unstructured interviews were conducted to understand their pair programming experiences and clarify their responses to questionnaires.

Student Respondents

The pair programming experiment was carried out for fifty seven students in software development laboratory course in the fifth semester of the MCA Program. The study was carried out in a controlled experimental setup. Twenty eight pairs were formed by pair programming information system (PPIS) based on student responses to the following factors such as willingness to participate in the experiment, partner preferences, level of knowledge, cumulative grade point average secured till the previous semester and their level of expertise in developing software applications and tool usage. One student was made to work individually. Most students preferred to work with the same gender and had no problems working with partner of any knowledge level. The major intent of the study is to enable the average and slow learners to learn and display improved performance in the laboratory course. Hence, the students were categorized into 4 levels based on the cumulative grade point average secured. The students were grouped as follows: Level 1 consisting of top performers, level 2 the above average performers, level 3 the average category and level 4 the slow learners. Students who were in level 4 and 3 were either paired with students in level 1 or level 2 in order to facilitate effective knowledge sharing. This data was used to perform rough set analysis.

The Controlled Experiment

A process framework was designed in order to carry out the pair programming exercise systematically [Figure 1]. Appropriate user interfaces available in the framework enabled student respondents and the assessors to record data easily. Once the students were found to acquire the requisite understanding about pair programming, they were allowed to access online software Pair Programming Information System [PPIS], which forms a part of the framework. PPIS enabled the students to fill the entry and exit questionnaire online. The questionnaire entries were stored in appropriate databases [1].

Twenty eight software application development projects [Table 2] with equal levels of difficulty, were chosen for the experiment by the faculty. These projects were randomly allotted to the students and the scope and requirements were clearly explained to contextualize the results. These tasks were executed during separate lab sessions of five hours duration per week. The pairs were asked to interchange driver-navigator roles once in the middle of each laboratory session to ensure equal contribution to the project.

During the lab experiment, the students were asked to record their experiences individually for each lab session. In order to extract and record the software development and learning experiences of those students working in pairs, they were made to fill in a worksheet as detailed in Table 1. Subsequently, details related to knowledge sharing and transfers were also collected. After the tasks were completed, the students were asked to fill in an online exit questionnaire, specifically designed to collect their views on pair programming, knowledge sharing, tool learning and collaborative skill development.

Table 1: Sample Questions Asked In The Questionnaires

Entry Questionnaire	**Exit Questionnaire**	**Worksheet**
Rate your level of understanding on the subjects Software engineering and object oriented analysis and design.	Do you think working in pairs was useful?	Lines of code developed
Mention the number of software applications developed so far	Do you see yourself getting better in developing collaborative skills?	Types of errors and time spent for debugging
Rate your level of familiarity of the concept 'Pair Programming'.	Has your productivity increased?	Contribution of partner in correcting errors
Mention the preferred level of your partner while doing pair programming.	Did pair programming improve your work quality and skills?	Pair programming experience in the session
How far you will be comfortable working with a different gender?	Effectiveness of pair programming in inception, elaboration, construction and transition phases.	Difficulties faced in the session if any
Cumulative Grade Point Average (till current semester).	How far do you get the support and coordination of the pair?	Additional features/enhancement included apart from the basic requirement for the project.

The previous studies reported in literature have not used a complete process framework that is fully automated. When the entry questionnaire is filled by the respondent online, PPIS would automatically suggest pairs based on student preferences. It would also suggest pairs randomly on demand. Once the data entry is complete for the questionnaires, worksheets and assessment sheets, the data will be stored in a database that can be exported in Microsoft excel format, which in turn can be fed into the analysis tools.

Table 2: Sample projects

SNO	**Project Title**
1	Resource planner for a college
2	Student feedback system
3	Exam result analysis system
4	Library management system
5	Student attendance management system

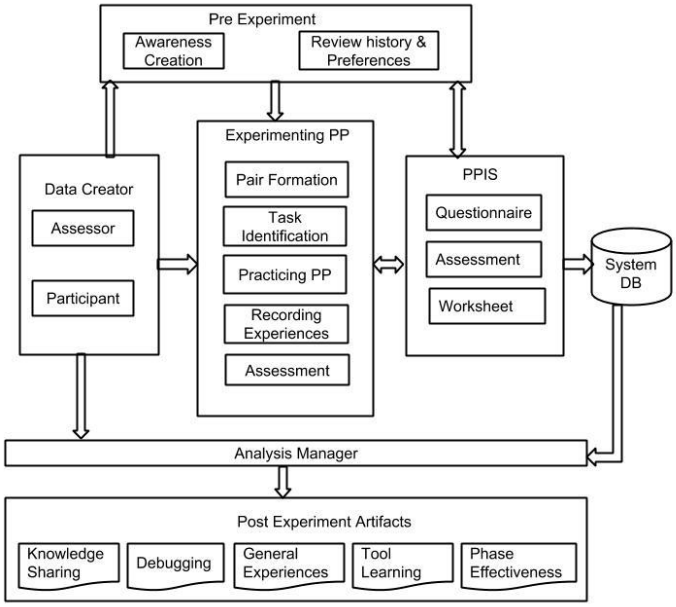


Figure 1: Pair Programming Process Framework

Validity and Reliability of the experiment

Generally, the attitudes and behavior of student respondents might not be consistent. At times, it is possible that the questions may not be interpreted by them as they are intended to be. In all likelihood, the student respondents may rate a factor without understanding the question carefully, thus yielding imprecise data and creating a threat to the validity of the data. This problem was addressed by designing questions that can be both clearly and easily understood by student respondents. The significance of the study and the need for recording accurate data were also explained and they were motivated and guided by the faculty, as and when needed.

Experimental Results and Discussion

Rough Sets Theory

The idea of rough set as a new mathematical tool to deal with vague concepts was proposed by Pawlak [18]. Rough set theory proposes a new mathematical approach to imperfect knowledge or vagueness. It offers mathematical tools to discover patterns in hidden data. By applying rough sets theory to knowledge discovery systems, it is possible both to identify and remove redundant variable, and also to classify imprecise and incomplete information. Thus, rough set theory is useful for reasoning about the knowledge of objects represented by attributes. The fundamental assumptions here are as follows: (i) the objects are represented by values of attributes and (ii) objects with same information are indiscernible. The main advantages of rough set tools are as follows: (i) it allows generating set of decision rules from the

given data automatically (ii) provides an easy interpretation of the results. The most important areas which rough set data analysis addresses are as follows: describing object sets by attribute values, finding dependencies between attributes, reducing attribute descriptions, analyzing attribute significance and generating decision rules.

Each rough set contains objects which cannot be classified with certainty as members of the set or its complement by employing the available knowledge [16]. Obviously, rough sets in contrast to precise sets cannot be characterized in terms of information about their elements. With any rough set is associated a pair of precise sets, called the lower and the upper approximation of the rough set, is associated. The lower approximation consists of all objects which belong to the set, while the upper approximation contains all objects which possibly belong to the set. The difference between the upper and the lower approximation constitutes the boundary region of the rough set. The rough set based data analysis starts from a data table known as a decision table, the columns of which are labeled by attributes, the rows by objects of interest and the entries of the table by attribute values. Attributes of the decision table are divided into two disjoint groups, known as the condition and decision attributes respectively. Each row of a decision table induces a decision rule, which specifies a decision, if some conditions are satisfied. If a decision rule uniquely determines a decision in terms of conditions, then the decision rule is considered to be certain. Otherwise, the decision rule remains uncertain. Decision rules are closely connected with approximations. While certain decision rules describe lower approximation of decisions in terms of conditions, uncertain decision rules refer to the boundary region of decisions. Each decision rule is related to two conditional probabilities, namely the *certainty* and the *coverage coefficient*. The certainty coefficient expresses the conditional probability that an object belongs to the decision class specified by the decision rule, provided it satisfies the conditions specified by the rule. The coverage coefficient provides the conditional probability of reasons for a given decision [6].

Pattern recognition and machine learning knowledge reduction are the two most important problems in data mining. The rough set theory has been applied to the development of learning and data reduction algorithms for data mining tasks. Rough set theory based classification of the pair programming data handles minimal set of attributes and vagueness which reduces the complexity of the data set.

Table 3: Sample PP Data Set

Student	Usefulness	Productivity	K. improvement	K. sharing	Proactive learning	Satisfaction	Work quality	Tool Learning	Grade
S1	4	3	4	4	4	4	4	4	s
S2	4	4	3	5	4	4	4	4	c
S3	4	4	5	5	4	4	3	4	a
S4	4	4	3	4	3	4	3	3	a
S5	4	5	4	4	4	5	5	5	s
S6	4	4	5	5	4	4	3	4	s
S7	3	4	4	5	3	3	3	4	s

S8	3	4	4	5	3	3	3	4	b
S9	5	5	5	5	4	5	4	5	b
S10	5	5	5	5	4	5	4	3	s

Table 3 shows a sample data set about the student's opinion on pair programming. The students have rated the various attributes of pair programming that helped them to improve their knowledge. The grades obtained by students in the model examination of the software development laboratory course are provided in the last column of the table and they have been calculated as shown in Table 4.

Table 4: Grade calculation

Range of Marks	Grade
90-100	s
80-89	a
70-79	b
60-69	c
55-59	d
50-54	e

From the table, it can be observed that two students (S7and S8) had given the same rating for all the above mentioned aspects of knowledge improvement. However, their model examination scores are different. Rough set theory can be used in such situations to handle imprecise data.

Rough Set Based Rule Evaluation

Empirical data was collected during pair programming sessions using the pair programming process framework. The data collected through the questionnaires were classified and transformed into a decision table for further analysis. Initially, the data was loaded into the data mining tool and made to undergo the preprocessing phases [17], namely the completion phase and discretisation phase. During the completion phase, the missing values in the table were attended to. Those objects with missing values in the table were either removed or the missing values were filled up with the mean value of the entries that are present. Discretization involves deciding the cuts that determine the intervals and the values fell within this interval were then mapped to the same value. This was carried out to ensure that the rules induced by the tool were not specific. Therefore, equal frequency scaler was used. For each attribute, the algorithm discretized the given attribute into a number of intervals such that each interval contained approximately the same number of objects. For example, rating 3 was discretized as (*, 4) and rating 4 as (4, 5). After discretization, the data set was randomly split into two disjoint sets. The first set, known as the training set, is used to extract knowledge used for creating general rules, relations and descriptions in the data set. The goal is to gain knowledge which is valid not only in the case of the specific data being considered, but also for other similar data sets. The knowledge thus extracted may be tested against the second set known as the test set. If the

knowledge gained from the training set is general, it is likely to be correct for most parts of the test set as well. Different sample sizes were randomly selected from the data set and a split factor of 0.6 was used for training and testing purposes.

Computing reducts is the task of finding minimal attribute subsets. There are many algorithms for reduct computation. This study uses the genetic RSES algorithm to generate both reducts and rules based on data collected in the exit questionnaire, as shown in Tables 5 and 6. Rules are generated on the basis of the computed reducts that constitute one of the most important results of the rough set data analysis. Two objects are considered to be conflicting when they are characterized by the same values for all attributes, but belonging to different classes. In such cases, rough sets compute both the lower and upper approximations. The tool used in this work generates rules for every object and its related reducts, considering the inconsistent data. The rules thus generated were used to discover knowledge by comparing two factors, namely students' performance in the model examination and the students' perceptions about various aspects about pair programming. The rules thus generated take into account the inconsistencies between two or more sets of information, describing the same variable which occurs in the data collected. Rules induced from the lower approximation of the classes certainly describe the class, hence such rules are called 'certain'. On the other hand, rules induced from the upper approximation of the class describe the class possibly and so, these rules are called the possible rules. Using rough sets, both the certain and possible rules were generated.

Table 5: Sample Reducts

SNO	Reduct
1	{productivity, satisfaction level}
2	{productivity, sharing knowledge, work quality}
3	{productivity, knowledge improvement, work quality}
4	{sharing knowledge, work quality, proactive learning}
5	{sharing knowledge, proactive learning, proactive learning }
6	{productivity, sharing knowledge, proactive learning }
7	{usefulness, knowledge improvement, work quality, tool learning }
8	{usefulness, knowledge improvement, proactive learning, tool learning }
9	{knowledge improvement, sharing knowledge, tool learning }
10	{productivity, knowledge improvement, tool learning }

Table 6: Sample Generated Rules

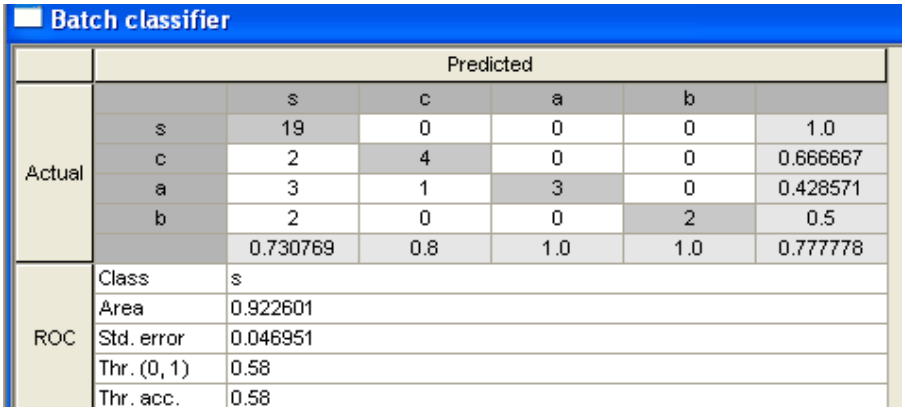
S. NO	Rules
1	productivity([*, 4)) AND satisfaction level([4, 5)) => Grade(s)
2	productivity([*, 4)) AND satisfaction level([*, 4)) => Grade(c)
3	productivity([*, 4)) AND sharing knowledge([*, 5)) AND work quality([4, 5)) => Grade(s)

4	productivity([4, 5)) AND sharing knowledge([*, 5)) AND work quality([*, 4)) => Grade(a)
5	productivity([*, 4)) AND knowledge improvement([4, 5)) AND work quality([4, 5)) => Grade(s)
6	productivity([4, 5)) AND knowledge improvement([4, 5)) AND work quality([*, 4)) => Grade(s) OR Grade(b)
7	productivity([5, *)) AND knowledge improvement([*, 4)) AND work quality([4, 5)) => Grade(s)
8	productivity([*, 4)) AND knowledge improvement([*, 4)) AND work quality([4, 5)) => Grade(a)
9	sharing knowledge([*, 5)) AND work quality([4, 5)) AND tool learning([4, 5)) => Grade(s)
10	sharing knowledge([*, 5)) AND work quality([5, *)) AND tool learning ([5, *)) => Grade(s) OR Grade(c)

The rules shown in Table 6 are based on the scores obtained by students in the model examination conducted a few weeks ahead of end semester examination. The predicted rules thus help the course instructors to identify those students who require extra care and coaching. From the rule productivity ([*, 4)) AND satisfaction level ([4, 5)) => Grade(s), it can be inferred that a student respondent who has rated 3 for the attribute productivity and 4 for the attribute satisfaction level shall score an s grade in the final examination. As per rule productivity ([*, 4)) AND satisfaction level ([*, 4)) => Grade(c), a student respondent who has rated 3 for the attribute productivity and 3 for the attribute satisfaction level shall score a grade c in the final examination. The rule productivity ([4, 5)) AND knowledge improvement ([4, 5)) AND work quality ([*, 4)) => Grade(s) OR Grade (b) can be interpreted as any student who rates 4 for the attributes productivity, knowledge improvement and 3 for the attribute work quality will score either s grade or b grade in the final examination. Thus it can be observed that while rules 1 and 2 are certain rules, rule 6 is uncertain. As per rule 6, the student who feels that productivity, knowledge improvement and work quality has improved significantly has scored in the range of either 90-100 or 70-79 marks, leading to uncertainty. As per rule 10, the student who feels that work quality and tool learning has improved significantly has scored in the range of either 90-100 or 60-69 marks, leading to uncertainty. Such rules help the instructors to identify those students who have given such ratings and to tightly monitor them. The results seem to indicate that rough sets work better in cases of inconsistent data.

The set of rules induced on the basis of the computed reducts are often used to classify new and unseen objects. Batch classifier has been used to classify the data in this work. The classifier performance can be analyzed and compared by those measures generated by the confusion matrix. The confusion matrix is a specific table layout that allows visualization of the performance of an algorithm and is typically a supervised learning one. Each column of the matrix represents the instances in a predicted class and each row represents the instances in an actual class. The matrix displays the number of correct and incorrect predictions made by the rules on the test

data. From Figure 2, it can be observed that the percentage of accuracy for classifying the pair programming data using the genetic algorithm is 77.



Batch classifier

		Predicted				
		s	c	a	b	
Actual	s	19	0	0	0	1.0
	c	2	4	0	0	0.666667
	a	3	1	3	0	0.428571
	b	2	0	0	2	0.5
		0.730769	0.8	1.0	1.0	0.777778

ROC		
	Class	s
	Area	0.922601
	Std. error	0.046951
	Thr. (0, 1)	0.58
	Thr. acc.	0.58

Figure 2: Confusion Matrix

Conclusion

Collaborative work is now being looked upon more seriously than ever in teaching-learning process. This was the impetus for carrying out the above–reported research. The study reports the results of preliminary work carried out in implementing pair programming as a teaching methodology. The results of the study conducted in the context of a programming laboratory course appear to be positive and also reveal the potential of PP in improving both programming practice and collaborative skills. The researcher developed a process framework for pair programming and experimented the effects of the same for a longer duration. Most student respondents have acknowledged in the questionnaire that practicing in pairs did help them experience a sense of reward and accomplishment. To deal with the inconsistent data, rough set analysis was carried out and the rules were generated. These rules would help in predicting students' performance in final examination and also in identifying those student groups which require personal attention by the faculty. The accuracy of the classifier was found to be at 0.77, which is a significant value.

Acknowledgements

The authors would like to thank the Management of Kumaraguru College of Technology, Coimbatore, India for providing the necessary support to conduct the experiment and collect the necessary data for research. We would like to thank all the students who participated in the study and faculty friends who supported the study. This work was funded by the All India Council for Technical Education (AICTE) Ref. No. 8023/RID/RPS-61/2011-12 (Pvt).

References

[1] R.K. Kavitha, M.S. Irfan Ahmed (2013), "Knowledge sharing through pair programming in learning environments: An empirical study", Education and Information Technologies, Springer, Published online:09 October 2013.

[2] Norsaremah Salleh, Emilia Mendes, John Grundy (2011), "Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review", IEEE Transactions on Software Engineering, Vol. 37, No. 4, 509-525.

[3] Hannay J.E, Erik Arisholm, Harald Engvik, Dag I.K. Sjøberg (2010), "Effects of Personality on Pair Programming" IEEE Transactions on Software Engineering, VOL. 36, NO. 1.

[4] M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson, J. Still (2008), "The impact of agile practices on communication in software development", Published online: 23 May 2008, Springer Science + Business Media

[5] V. Venkatesan, A. Sankar (2010), "Adoption of Pair Programming in the Academic Environment with different Degree of Complexity in Students Perspective– An Empirical Study", International Journal of Engineering Science and Technology Vol. 2(9), 4791-4800.

[6] Rajendra Akerkar, Priti Sajja (2010), "Knowledge-Based Systems", Jones and Bartlett Publishers, Canada

[7] Norsaremah Salleh (2008), "A Systematic Review of Pair Programming Research –Initial Results", Proceedings of *NZCSRSC 2008*, Christchurch, New Zealand, 151-158.

[8] Tanja Bipp, Andreas Lepper, Doris Schmedding (2008), "Pair Programming in Software Development Teams-An empirical study of its benefits", Science Direct Information and Software Technology, 231–240.

[9] Andrew Begel, Beth Simon (2008), "Novice software developers, all over again", Proceedings of the Fourth international Workshop on Computing Education Research, ICER '08, 3-14, New York, ACM.

[10] Jari Vanhanen, Harri Korpi (2007), "Experiences of Using Pair Programming in an Agile Project", Proceedings of the 40th Hawaii International Conference on System Sciences IEEE Computer Society, 1530-1605.

[11] Kim Lui, Keith C.C. Chan (2006), "Pair Programming Productivity: Novice-Novice vs. Expert-Expert", International Journal of Human-Computer Studies, 64 (9), 915-925.

[12] Matthias M.Muller (2005), "Two controlled experiments concerning the comparison of pair programming to peer review", The Journal of Systems and Software,78(2), 166-179.

[13] Subbaraya Kuppuswami, Kalimuthu Vivekanandan (2004), "The Effects of Pair Programming on Learning Efficiency in Short Programming Assignments", Informatics in Education, Vol. 3, No. 2, 251–266, Institute of Mathematics and Informatics, Vilnius.

[14] Laurie Williams, Robert Kessler (2003), "Pair Programming Illuminated", Addison Wesley.
[15] Geoff Scott and David Wilson (2002), "Tracking and profiling successful IT graduates: An exploratory study", Proceedings of the 13th Australasian Conference on Information Systems, ACIS '02, 1185-1195.
[16] Zdzisław Pawlak (2002), "Rough set theory and its applications**",** Journal of Telecommunications and information technology.
[17] Kent Beck (2000), "Extreme Programming Explained", Addison-Wesley.
[18] Zdzisław Pawlak (1982), "Rough sets", International Journal of Computer and Information sciences", Volume 11, Issue 5, 341-356, Springer.
[19] Ella HASSANIEN, Jafar M.H. ALI (2004), "Rough Set Approach for Generation of Classification Rules of Breast Cancer Data" INFORMATICA, Vol. 15, No. 1, 23–38.