Journal of Education and Practice
ISSN 2222-1735 (Paper) ISSN 2222-288X (Online)
Vol.11, No.31, 2020

The Impact of Software Team Project Measurements on Students' Performance in Software Engineering Education

Bilal Al-Ahmad[1*] Shadi Banitaan[2*] Rami Alkhawaldeh[3]
1.Faculty of Information Technology and Systems, The University of Jordan, Aqaba 77110, Jordan
2.College of Engineering & Science, University of Detroit Mercy, 4001McNicholsRd, Detroit, USA
3.Faculty of Information Technology and Systems, The University of Jordan, Aqaba 77110, Jordan

Abstract
It is essential to the software engineering instructors to monitor the students' performance in their course projects. Detecting key measures of software engineering project helps to get a better assessment for students' performance, resolve difficulties of low expectation-team's, and consequently improves the overall learning outcomes. Several studies attempted to present the important measures of software project but they only captured the early phases of the whole project time period. This paper introduces a hybrid approach of classification and feature selection techniques, which aims to comprehensively cover all phases of software development through investigating all product and process measures of software project. Experiments were conducted using five classifiers and two feature selection techniques. The results show the significant process and product measures for the software engineering team projects, which primarily improves the students' performance assessment. The performance prediction of our proposed assessment model outperforms prediction of the previous models.
Keywords: Assessment, Classification, Feature selection, Software engineering education, Software team
DOI: 10.7176/JEP/11-31-02
Publication date: November 30th 2020

1. Introduction

Software engineering (SE) is an important course to study the overall software development life cycle and increase the poor-quality of software (Guéhéneuc & Khomh 2019; Möller 2016; Standish Group 2009). Software engineering teams shall have specific skills to practice and learn the software process development. Several causes lead to software project failure linked to failures in teamwork aspects of software engineering, such as lack of experience, schedule surpasses, and globally distributed student teams, as in the studies by (Cuthbertson & Sauer 2003; Sauer *et al.* 2007; Daughtrey 2014; Reel 2009; Charette 2005; Duhigg 2016). Many factors primarily affect software engineering teams' learning process. So, measuring software engineering students' team activities during the classes is very important to assess their performance and eventually achieve better learning outcomes. The software engineering process measures involve certain team activities during the adaptation of good practices of software development processes like quality and time completeness of non-software deliverables such as website design, meetings participation, and documentation. While the software engineering product captures software deliverables (i.e., team outputs) issues such as user interface, performance, architecture, database design, code quality, and presentation for the project's final delivery to the instructors as described in SETAP project by (Lichman 2013). The previous studies only captured early design and implementation phases of software development. This study focuses on exploring the most critical measures for software engineering projects by assessing the team activities' learning capabilities to the students who participate in such projects.

Our approach lets the students expect their software engineering course performance and focus more on their weaknesses activities during the software project. The proposed method effectively improves the prediction of the grades for the software engineering students' projects. The paper makes the following main contributions:

- Investigating the effectiveness of applying a hybrid assessment model on the all the time intervals of software project to improve the prediction performance of software engineering teams.
- Finding essential product and process measures for each phase for the identification of low-expectation teams in software engineering education.

2. Related Work

Several approaches used several techniques in the educational environment to assess individual student performance by exploring the teaching classes' personal and quantitative issues such as grades, dropping frequency, teaching effectiveness, and e-learning techniques proposed by (Kotsiantis 2012; Lykourentzou, *et al.* 2009; Castro, *et al.* 2007; Baker & Yacef 2009; Baker 2002; Macfadyen & Dawson 2010; Delen 2010; Jovanovic *et al.* 2012; Hu *et al.*, 2014; Guo *et al.* 2015). The research studies were conducted by (Petkovic *et al.* 2014; Petkovic *et al.* 2012; Petkovic *et al.* 2018) used Random Forest classifier as recommended by (Gomes, et al., 2017) to predict and assess software engineering teamwork rather than individual students by collecting the objective and quantitative data about team activity measures through a joint project among San Francisco State University (SFSU), Fulda University (Fulda) and Florida Atlantic University (FAU). They created a machine learning database that contains

team activity measurements, instructor observation, and their grades. Their approach used four primary means: time cards, software tools, class data, and instructor observations to collect data about the participating software engineering teams. The previous studies (Petkovic *et al.* 2014; Petkovic *et al.* 2012; Petkovic *et al.* 2018) used only Random Forest classifiers and considered only a few time intervals such as T2 and T3 to report their results. Also, they lack to detect the essential measures across all the time intervals extensively. Also, a study (Naseer *et al.* 2020) introduced a combined approach that employed feature selection and then classification on the same dataset. This study considers only five time intervals from T1 to T5 and captures only the product software measures.

Another study was conducted by (Abidin *et al.* 2019) to explore students' performance in software engineering courses using adaboost-multilayer perceptron. This study uses only one classifier and lacks to focus on team projects of the students. The study conducted by (Le *et al.* 2017) used text similarity and machine learning to construct formative evaluation for software engineering team projects. Some approaches also used traditional methodologies such as genetic programming implemented by (Zafra & Ventura 2019), and task recommendation systems by the study conducted by (Thai-Nghe *et al.* 2011). Nevertheless, these approaches used specific attributes such as quizzes, assignments, attendance, and GPA to evaluate the students' performance. They used individual assessment and did not take into account the teamwork projects. Unlike the previous studies which only considers early design and implementation phases, this study captures all the phases of Software Development Life Cycle (SDLC) and detect the essential measures for each phase. The previous studies make an effort to present the important measures of software project but they only capture few time intervals of the whole project period and they lack to broadly present the all-important measures for each particular interval. To overcome the aforementioned limitations, this paper aims to comprehensively cover all the time intervals of software project period by investigating all product and process software measures.

This paper uses five classifiers, namely Bayesian Network, Decision Tree, Random Forest, Bagging, and Boosting, to overcome the limitations above. It considers all the time intervals for both process and product measures to assess students' learning process effectiveness. Also, it focuses on the team assessment, not into the individual evaluation. This paper mainly uses the classification and feature selection phases. In the classification phase, all the software team measures have been studied to observe different machine learning classifiers' performance. On the other hand, the feature selection phase uses evolutionary and PSO search techniques to capture the essential team activity measures for each time interval in both the software process and product aspects.

3. Methodology

The proposed assessment model for students' performance contains two phases: classification, and feature selection as illustrated in Figure 1. Our approach acts as a hybrid approach of classification and feature selection techniques, which aims to comprehensively cover all phases of software development through investigating all product and process measures of software project. It investigates all the process and product measures which have been collected in SETAP project. The classification phase is used to early predict the performance of low expectation-team's projects. The feature selection phase is employed to detect the most important product and process measures for low expectation team. Detecting the key product and process measures is very important task since it affects the learning aspects of educational Software Engineering projects.

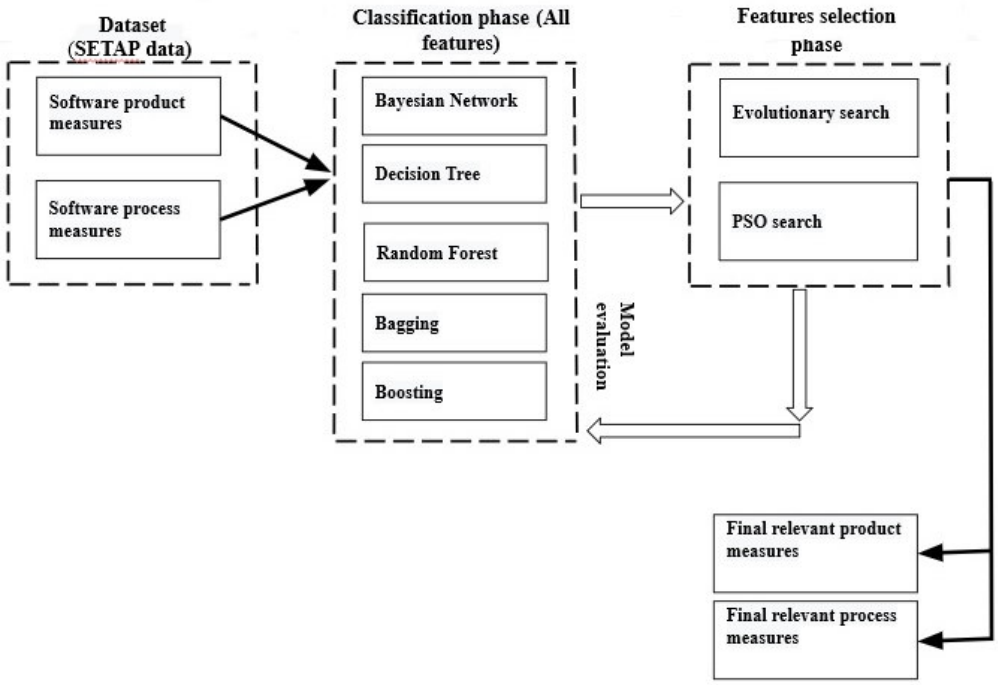

Figure 1. The proposed assessment model

3.1 Classification Phase

This phase includes the use of five machine learning classifiers: Bayesian Network, Decision Tree, Random Forest, Bagging, and Boosting to explore the effectiveness of all software process and project measures on students' learning process for each particular time interval. We briefly describe the five classifiers below:

- Bayesian networks are directed acyclic graphs that are used to represent the joint probability distribution over a set of random variables. Each vertex in the graph represents a random variable, and edges represent the correlations between variables. The Bayesian network classifier returns the class label that maximizes the posterior probability as proposed by (Han *et al.* 2011).
- Decision Trees construct a flowchart-like structure where each internal node represents an attribute, and each external node holds a class label. For a testing instance, a path is traced from the root to a leaf node to get the class prediction, as explained by (Thai-Nghe *et al.* 2011).
- Random Forest classifier is an ensemble classification technique that works by generating many decision trees from bootstrap samples of the training data. Each tree predicts a class label for each input vector. The output of the classifier is selected by taking the majority voted class from all the decision trees in the forest as proposed in the study (Guo *et al.* 2015).
- Bagging is an ensemble method that uses several training sets generated by a random draw with replacement. Each data set is used to train a model. The outputs of all models are combined to produce a single class label such used by (Friedman & Popescu 2003).
- Boosting is the sequential learning of the predictors. The first predictive model learns from the whole data set, while the next learns from training sets based on the performance of the previous one. The weights of the misclassified examples increase, so they will have a higher probability of being included in the next predictor's training set, as concluded by (Friedman & Popescu 2003).

3.2 Feature Selection Phase

This phase employs two meta-heuristic nature-inspired techniques; evolutionary and PSO (Particle Swarm Optimization) search as features selection techniques to detect the most important measures. These techniques search in the space of solutions (or population) to find the optimal solution that represents the optimal features for student's performance assessment that help in getting better students' performance through the learning in software engineering classes. This paper uses evolutionary and PSO search techniques. The two techniques described as below:

- Evolutionary Search is an iterative process in the form of generation to refine the population with fittest solutions using three main steps as in (De La Iglesia 2003; Namous *et al.* 2020). These steps include: (1) initializing a set of random solutions on a form of individual chromosomes, each representing the set of parameters of a problem. Each chromosome represents an array of features for evaluation in the

form of binary digits where each position identifies if a feature is selected (one) or not (zero); The strength of a chromosome is derived from applying an ML classifier as an evaluator on the ones' features in that chromosome where an ML evaluation metric is used as fitness value for that chromosome. (2) Two fittest chromosomes with the highest fitness values are selected as the elitism of the population (or fittest parent) to create new chromosomes (or children). (3) The process of creating the children is maintained in two main steps derived from nature with a random probability of using either; called crossover and mutation. The crossover process exchanges parts in the two chromosomes between each other, while the mutation process flips bits in a random position in the chromosome to generate a new child. The evolutionary search process then estimates the fitness values for the children and replaces them with non-relevant solutions.

- PSO Search is used for optimizing constrained and unconstrained tasks described by (Eberhart & Kennedy 1995). It depends on a set of particles that iteratively change their positions based on some stochastic process. This identifies the best fittest particle among all particles. Therefore, all particles aim to reach the location of the fittest particle. Also, each particle considers its best position during the search found by itself.

4. Results and Discussions

4.1 Dataset

The dataset includes over 115 measures of both process and product metrics which obtained from measuring the core activities of student teams through their final project in joint software engineering classes which applied at San Francisco State University, Fulda University and Florida Atlantic University. The software process measures related to non-software issues such: team participation, students' feedback, delivery of documentation, instructor intervention, cooperation concerns, and following proper software engineering practices. Also, the software product measures captured the software issues such: functionality, architectural design, code, database, and final product demo. The dataset includes 74 student teams with about 380 students. These data are combined into 11 different time intervals, the first five intervals measure requirement gathering (T1), design (T2), development (T3), testing (T4), and delivery (T5) phases correspondingly of Software Development Life Cycle (SDLC), while the remaining time intervals (T6-T11) are generated by aggregating various software phases from the main five time intervals (T1-T5). The semester is divided into five formally managed milestones, Ml through M5. The product and process measures are individually assigned to final grade as A (high expectation) or F (low expectation). Also, each software team project is assessed at 11-time intervals. This study focuses on predicting the class labeled F rather than the class labeled A.

4.2 Experiment Results

To evaluate the predictive models, we use the Recall, where it represents the percentage of positive tuples the classifier label as positive. There is a diversity in assessing students' performance during the time interval of SE activities. These activities come in individual and grouped tasks that result in a sense by merging the values of evaluation metrics. Figure 2 shows the results of using five machine learning techniques conducted on the product measures. The x-axis represents the time interval of eleven tasks; individual tasks (or T1-T5) and grouped tasks (or T6-T11), while the y-axis represents the Recall value for class F (under-estimation). For the product component and in terms of a block of SDLC, including the first five tasks, the Bayesian Network model (BN) outperforms the other ML techniques with effect Recall value at task five of 0.844 and average recall values over all other tasks. The conditional probability in BN effectively ties the relationship between the measures (or features) and the target, due to the low divergence in the data distribution. The entropy-based techniques manifest plausible results affected by the random distribution of the data with feasible Recall value to bagging technique using a decision tree as a base classifier. In contrast to the other time intervals (or grouped-based tasks), the entropy-based techniques leverage high performance to average Recall. This is because of the random skewness of the data distribution with a large variance that increases the entropy values and helps the entropy-based techniques to discretize the target label.

Figure 3 also shows the Recall for the F class label for the different classifiers in all intervals (T1 to T11) for all the process measures. The results show that the Decision Tree classifier gives the best Recall in seven-time intervals while Bagging gives the best Recall in the remaining four intervals. The highest Recall is 0.68 that is obtained using Bagging for the T2 time interval. The expected reason to achieve the best Recall in the T2 time interval is that this time interval represents the detailed specifications phase where teams are expected to have a lot of communication and collaboration to complete this phase. Bayesian Networks gives the worst results in six-time intervals while the Random Forest gives the worst results in four-time intervals. We also note that none of the classifiers obtain good results in the prediction of the F class labeled for time intervals T4 and T5 where T4 represents the beta launch milestone while T5 represents the final delivery milestone. The previous studies (Petkovic *et al.* 2014; Petkovic *et al.* 2012; Petkovic *et al.* 2018) lacked to present the classification on the

remaining time intervals, focused only on T2 and T3, used only Random Forest classifier, and implemented stratified sampling in cross validation, since class labeled F is minority while the class labeled A is the majority as assigned in the dataset. Also, the study (Naseer *et al.* 2020) investigated only the software product measures for the first five intervals T1, T2, T3, T4, and T5 by implementing several different classifiers. On the other side, our proposed assessment model acts as comprehensive approach that covers all the time intervals from T1 until T11 for both of software product and process, and it demonstrates classification and feature selection techniques.

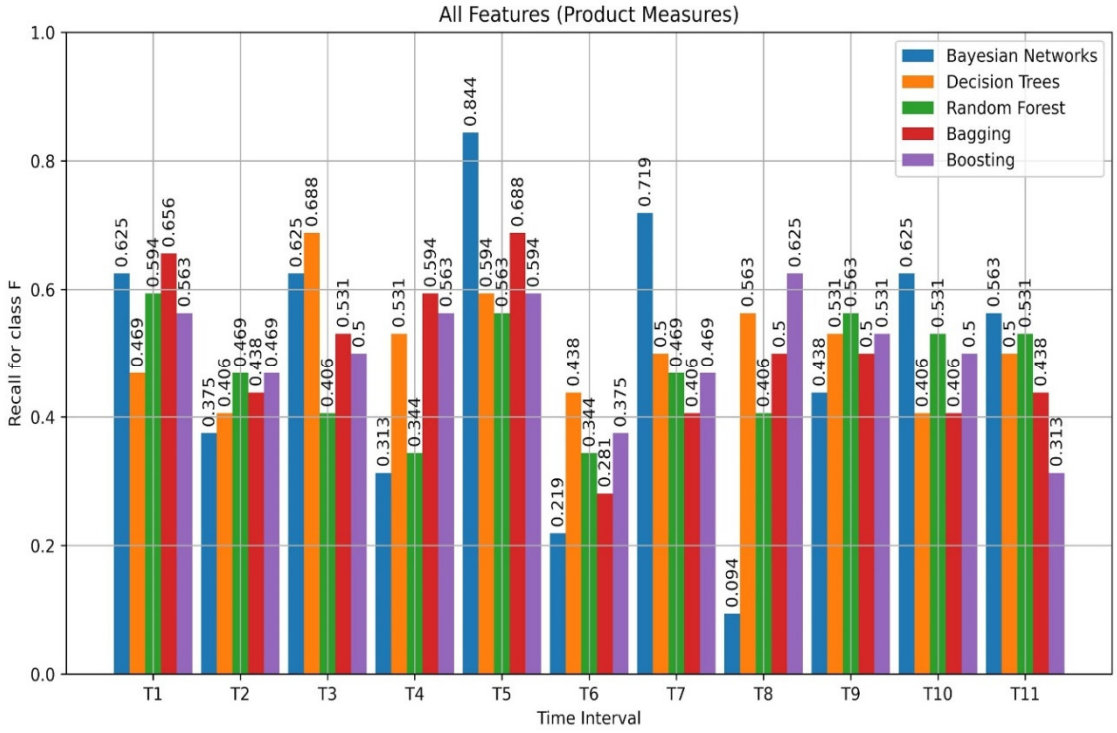

Figure 2. Classification results of software product measures

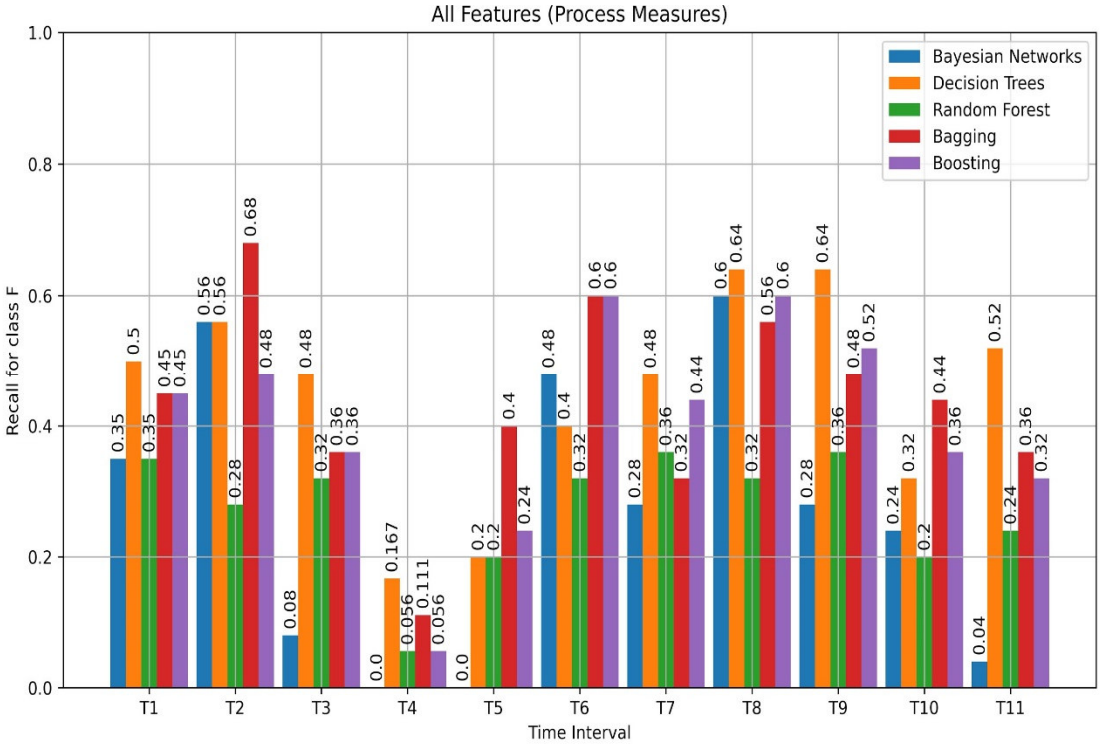

Figure 3. Classification results of software product measures

Table 1 summarizes the comparison between previous models and the proposed assessment model. The results were reported in the prior studies (Petkovic *et al.* 2014; Petkovic *et al.* 2012; Petkovic *et al.* 2018) presented the important measures for only the time intervals T2 and T3 of the whole project time period and they lack to cover the key measures for all 11 time intervals of the software project period. For software process measures at T2 as in the study (Petkovic *et al.* 2014), the best Recall is 66.7 % in their approach whereas it is 68% in our approach. Moreover, for software product measures at T3, the best Recall is 60% by using Random Forest classifier although it is 68.8 %. These significant findings indicate that our assessment model improves the prediction of the final grades for the low expectation SE teams. Furthermore, our results comprehensively report all process and product measure's for all the 11 time intervals of software project period. On the other words, the previous studies only captured early design and implementation phases of SDLC, while our approach captures all the phases of software development as well as the combination of these phases. This lead to get a better perceiving for the overall software development.

Besides, the study (Naseer *et al.* 2020) investigated only the software product measures for the first five intervals T1, T2, T3, T4, and T5 by implementing several different classifiers. The best Recall in their assessment model reported as 3.1 %, 15.6 %, 53.1 %, 78.1 %, and 75% for the time intervals T1, T2, T3, T4, and T5 respectively. On the other hand, our proposed assessment model outperforms the performance of their Recall values as specified in Table 1. The best Recall values in our findings are 65.6 %, 46.9 %, 68.8 %, 59.4 %, and 84.4 % for the time intervals T1, T2, T3, T4, and T5 correspondingly. In comparing with the previous studies These noteworthy findings indicate that our assessment model improves the prediction of the final grades for the low expectation SE teams. In addition, our study aims to propose an effective assessment model for predicting low expectation-teams (F-grade), it uses several machine learning classifiers on all process and product measures for each software team. Based on the best resulting Recall, evolutionary and PSO search are implemented to identify the most relevant measures for process and product individually. Knowing such software processes and product measures allow software engineering instructors to put more effort and attention to overcome the ambiguity and difficulties of low-expectation teams. Also, our approach lets the students to early expect their performance in software engineering course as well as they can keep attention to focus more on their weakness activities during the software project. Our study aims to improve the software projects learning which leads to a better software engineering education process.

Table 1. Comparing of performance for predicting F-class

Approaches	Time interval used	Software measures type	Employed classifiers	Feature selection	Best recall	Sampling
Petkovic *et al.* 2014	Only T2	Process	Only Random Forest (RF)	GINI	66.7 % (RF at T2)	Stratified sampling
Proposed approach	T1 until T11	Process	Bayesian Network, Decision Tree, Random Forest, Bagging, and Boosting	Evolutionary and PSO search	68 % (Bagging at T2)	No sampling
Petkovic *et al.* 2014	Only T3	Product	Only Random Forest (RF)	GINI	60 % (RF at T3)	Stratified sampling
Proposed approach	T1 until T11	Product	Bayesian Network, Decision Tree, Random Forest, Bagging, and Boosting	Evolutionary and PSO search	68.8 % (Decision Trees at T3)	No sampling
Naseer *et al.* 2020	T1 until T5	Product	Naïve Bayes, Artificial neural network, logistic regression, RIPPER, and sequential minimal optimization	J48 decision tree	3.1 % (at T1) 15.6 % (at T2) 53.1 % (at T3) 78.1 % (at T4) 75% (at T5)	No sampling
Proposed approach	T1 until T11	Product	Bayesian Network, Decision Tree, Random Forest, Bagging, and Boosting	Evolutionary and PSO search	65.6 % (at T1) 46.9 % (at T2) 68.8 % (at T3) 59.4 % (at T4) 84.4 % (at T5)	No sampling

In respect to the feature selection phase, it is very essential to select the most important process and product measures to help instructors for assessing the final grades of low-expectation teams. Figure 4 and Figure 5 show the experimental results on selected measures extracted using evolutionary and PSO search compared to entire measures for the software product and process, respectively. The two techniques derived relevant measures with remarkable results compared with the overall measures in the product component, especially in tasks (T6 to T11)

having a measure reduction of approximately 95.7%. This indicates that there are some measures with high impacts on assessing the performance as shown in Figure 4. These measures are mainly linked to software product attributes, such as collecting all important activities that are handled by students in the code repository for each software team.

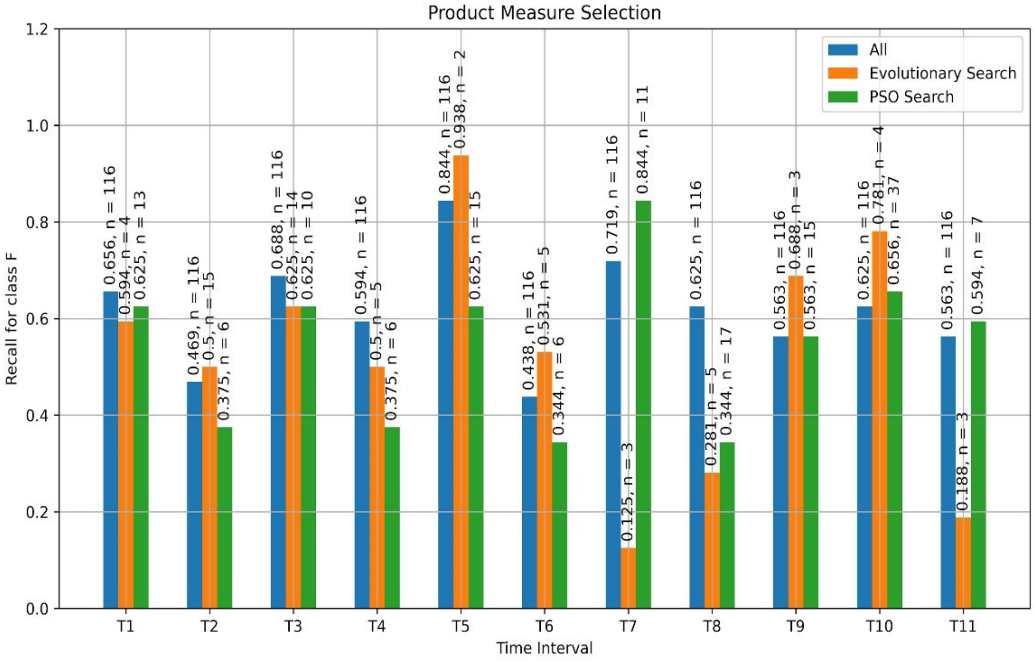

Figure 4. Recall for software product measures selection

Figure 5 also shows the accuracy for the software process measures where the x-axis represents the time interval and the y-axis represents the Recall for the F class label (a grade below expectations) for the process measures. Remarkable, the best results obtained from Figure 1 are used for comparison with both evolutionary search and PSO search in Figure 3. Similarly, the top results obtained from Figure 2 are used for comparison with both evolutionary search and PSO search in Figure 4. For example, the Recall for time interval T1 is 50 %, 52.4 %, and 54.2 % using all measures, measure selected using evolutionary search, and features selected using PSO search respectively. Building predictive models using features selected by evolutionary search and PSO search improve the Recall for time interval T1 only while using all measures give the best results for the remaining time intervals.

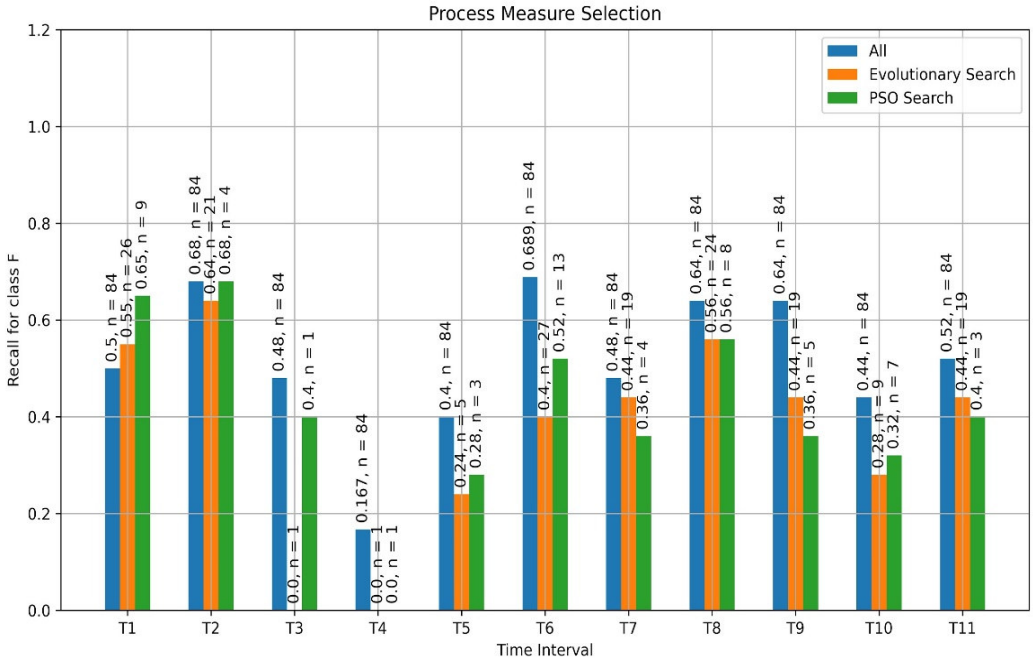

Figure 5. Recall for software process measures selection

The results show that using only a small number of measures (1 measure to 27 measures) can give results almost similar to the results obtained by using all 85 measures. It is noteworthy that the two feature selection techniques capture the relevant features with good results compared with all measures with an average feature reduction of 87.6%. Since our goal is to predict how well the team applied the best software engineering practices, we would like to look deeply into the important measures identified by the feature selection techniques in the first time interval. The selected measures are related to software process attributes such as the total number of meeting hours and the total of in-person meeting hours. The results of this phase are described as in Table 2 and Table 3. Regarding the software product and process measures, the most important measures are related to tools logs, class data, and weekly time card surveys. The time intervals T4, T6, and T8 have the same product measures *semesterId* and *helpHoursAverage*. This refers to the mutual milestones among these time intervals during the software engineering team activities measurements.

Table 2. Mutual selected product measures between evolutionary and PSO search

Time Interval	Product Measures
T1	Standard Deviation Commit Count By Student
T2	semesterId, in Person Meeting Hours Average, average Coding Deliverables Hours Total By Week
T3	Standard Deviation In Person Meeting Hours Average By Week, unique Commit Message Count, Commit Message Length Total, standard Deviation Unique Commit Message Count ByStudent
T4	Semester Id, help Hours Average
T5	no mutual measures
T6	semesterId, help Hours Average
T7	no mutual measures
T8	semesterId, help Hours Average
T9	Standard Deviation Meeting Hours Total By Week, standard Deviation Help Hours Total By Week, standard Deviation Non Coding Deliverables Hours Average By Student
T10	Average Responses By Week,standard Deviation Coding Deliverables Hours Total By Student
T11	Standard Deviation Unique Commit Message Percent By Student,

Table 3. Mutual selected process measures between evolutionary and PSO search

Time Interval	Product Measures
T1	Meeting Hours Total, In Person Meeting Hours Total, average Help Hours Total By Week, standard Deviation Unique Commit Message Count By Week, standard Deviation Unique Commit Message Percent By Week
T2	Late Issue Count, issue Count
T3	No mutual measures
T4	No mutual measures
T5	Average Responses By Week
T6	Late Issue Count, coding Deliverables Hours Total, average Non Coding Deliverables Hours Average By Student, non Coding Deliverables Hours Average, issue Count, average Non Coding Deliverables Hours Average By Week, team Member Response Count, help Hours Total
T7	Issue Count, on Time Issue Count, unique Commit Message Count, team Member Response Count
T8	Team Member Response Count, issue Count, help Hours Total, late Issue Count, on Time Issue Count, standard Deviation In Person Meeting Hours Total By Student
T9	Issue Count, average Responses By Student, team Member Response Count, on Time Issue Count
T10	In Person Meeting Hours Total, standard Deviation Meeting Hours Total By Week, average Responses By Week, in Person Meeting Hours Total, standard Deviation Meeting Hours Total By Week, average Responses By Week, in Person Meeting Hours Total, standard Deviation Meeting Hours Total By Week, average Responses By Week, in Person Meeting Hours Total, standard Deviation Meeting Hours Total By Week, average Responses By Week
T11	Team Member Response Count, issue Count, average Meeting Hours Average By Student

The study (Petkovic *et al.* 2018) used GINI feature ranking to show the top 10 ranked process and product measures for time interval T2 and T3 respectively. The process measures of T2 were: lateissue Count, issue Count, standard Deviation Help Hours Total By Week average Help Hourstotal By Week, standard Deviation Help Hours Average By Week, coding Deliverables Hours Average, help Hours Standard Deviation, and standard Deviation MeetingHoursAverageByWeek. Furthermore, the study (Naseer *et al.* 2020) used J48 decision tree in sequential phases of assessments from T1 to T5 to select the software product features for the classification, they indicated few product measures as listed in their study. The proposed model detects the mutual measures after implementing

evolutionary and PSO search as summarized in Table 2 and Table 3.

The common measures selected by our employed feature selection techniques and the feature selection technique employed by (Naseer *et al.* 2020) as: Female Team Members Percent, average In Person Meeting Hours Total by Week, average Meeting Hours Total by Week, average Responses By Week, on Time Issue Count, team Lead Gender, team Distribution, average Unique Commit Message Count By Student, unique Commit Message Count, average Responses By student, and.Team Member Response Count.

Our feature selection technique detected many of the essential measures selected by previous work. Also, the proposed model detects many other important features that undetected by the previous studies (Petkovic, *et al.* 2018; Naseer *et al.* 2020). The literature proved that evolutionary and PSO search could select significant features. Therefore, the chosen measures can guide instructors to monitor low-expectation teams who are at risk and enable students to perceive the important activities in their projects that will affect their final grade.

4.3 Discussions

Performance prediction conveys major importance in educational software engineering and will lead to better academic learning outcomes for students and instructors. Software engineering education concentrates on monitoring the low-expectation teams to improve their skills and activities during the learning process. Many studies produced outcomes using different machine learning techniques to obtain better academic practices in educational environment by collecting educational dataset through sequential semesters. Software engineering education involves project-based learning task. The SE students work in group teams in several time intervals during the project period to eventually pass the course. Then, different phases of assessments help to monitor the performance of each team in different time intervals. Nonetheless, the prediction of low-expectation teams includes the using of machine learning techniques. The SETAP dataset is a rich educational dataset for software engineering education which enable the SE instructors to observe the performance of SE teams over 11 phases of assessments. A hybrid assessment model of classification and feature selection techniques was employed to comprehensively covered all the assessment phases to improve the performance prediction. The results discovered that it is possible to predict the performance of teams at all stages of evaluation to achieve better software engineering education with significant accuracy in the prediction of low-expectation teams which enable the SE instructors to identify their problems and learning difficulties.

Hence, to evaluate the best prediction, the proposed model was applied at the first level of assessment and sequentially until the eleventh level of assessment for both of product and process software measures. By considering the similar explored time intervals in the previous studies (Petkovic *et al.* 2018; Naseer *et al.* 2020), the results reveal significant predictions in comparing with the previous assessment models.

The performance prediction of low-expectation teams for software product measures is much better than for software process measures. This difference might refer to the large number of measures in both types, there are 115 product measures and 84 process measures. The software process measures related to non-software issues such: team participation, students' feedback, delivery of documentation, instructor intervention, cooperation concerns, and following proper software engineering practices.

Also, the software product measures captured the software issues such: functionality, architectural design, code, database, and final product demo. The assessment models (Petkovic *et al.* 2018; Naseer *et al.* 2020) are used to compare the proposed model to check its efficiency for the prediction of low-expectation teams. The proposed model outperformed other methods at assessment levels two, three, fourth, and five. At level five, the proposed model achieved the best Recall in predicting the low-expectation teams. The identification of learning difficulties of low-expectation teams considered as an important concern for software engineering education. The proposed assessment model helps to achieve the learning objectives by improving the activities for software engineering teams. The prediction for software product and process development can help instructors detect low-expectation teams, and allow instructors and students to perceive the final grades expectation. This proposed model improves the software project assessment and enables for moving toward the success of software engineering courses in software engineering education.

4.4 Threats to validity

The main challenges for this study are: 1) the dataset has small size since it only contains 74 student teams, 2) the dataset is unbalanced since the class labeled F occurrences are few comparing with the large number records of class labeled A since the focus of our study to predict the class labeled F (low-expectation) rather than the class labeled (high-expectation), and 3) to estimate the Recall in the presence of unbalanced training data, this study lacks to use any sampling type in cross validation. We believe that our performance prediction results get improved if we use sampling to increase the occurrences of class labeled F in the dataset.

Conclusion

This paper proposes a hybrid assessment model for evaluating the students' performance in software engineering

projects. Unlike the previous approaches which studies only design and implementation phases, this study captures all the phases of software development and detect the essential measures for each phase. It effectively employs several machine learning classifiers and feature selection techniques to cover all the time intervals of software project period. Experiments are conducted on a rich academic dataset which has been collected through joint software engineering classes among three universities. The Recall measure is applied to show the accuracy of low expectation-teams across over 11 sequential time intervals and based on the predefined criterion, which has been determined by assigned instructors. The results have shown a significant process and product measures for the software engineering team projects, which mainly affects the students' performance assessment. The proposed model helps software engineering instructors to assess the students by highlighting the essential software measures for each specific time interval of software project duration. It also improves the students' learning capabilities during software projects to concentrate on their final grades' vital software measurements. Future directions include applying deep learning techniques to get a new direction in the software engineering education environment.

References

Abidin, A. F. Z., Darmawan, M. F., Osman, M. Z., Anwar, S., Kasim, S., Yunianta, A., & Sutikno, T. (2019), "Adaboost-multilayer perceptron to predict the student's performance in software engineering", *Bulletin of Electrical Engineering and Informatics* **8**(4), 1556-1562.

Baker, M. J. (2000), "The roles of models in Artificial Intelligence and Education research: a prospective view".

Baker, R. S., & Yacef, K. (2009), "The state of educational data mining in 2009: A review and future visions", *JEDM| Journal of Educational Data Mining* **1**(1), 3-17.

Castro, F., Vellido, A., Nebot, A., & Mugica, F. (2007), "Applying data mining techniques to e-learning problems", In *Evolution of teaching and learning paradigms in intelligent environment,* Springer, 183-221.

Charette, R. N. (2005), "Why software fails [software failure]", *IEEE spectrum* ***42***(9), 42-49.

Daughtrey, T. (2014), "Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies", *Software Quality Professional* ***16***(3), 40.

De La Iglesia, B. (2013), "Evolutionary computation for feature selection in classification problems", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* ***3***(6), 381-407.

Delen, D. (2010), "A comparative analysis of machine learning techniques for student retention management", *Decision Support Systems*, ***49***(4), 498-506.

Duhigg, C. (2016), "What Google learned from its quest to build the perfect team", *The New York Times Magazine* ***26***.

Eberhart, R., & Kennedy, J. (1995), "A new optimizer using particle swarm theory", In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 39-43.

Friedman, J. H., & Popescu, B. E. (2003). "Importance sampled learning ensembles", *Journal of Machine Learning Research* ***94(305)***, 1-32.

Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., ... & Abdessalem, T. (2017), "Adaptive random forests for evolving data stream classification", *Machine Learning* ***106***(9-10), 1469-1495.

Guo, B., Zhang, R., Xu, G., Shi, C., & Yang, L. (2015), "Predicting students performance in educational data mining", In *2015 International Symposium on Educational Technology (ISET)*, 125-128.

Han, J., Kamber, M., & Pei, J. (2011), "Data mining concepts and techniques third edition", *The Morgan Kaufmann Series in Data Management Systems*, 83-124.

Hu, Y. H., Lo, C. L., & Shih, S. P. (2014), "Developing early warning systems to predict students' online learning performance", *Computers in Human Behavior* ***36***, 469-478.

Jovanovic, M., Vukicevic, M., Milovanovic, M., & Minovic, M. (2012), "Using data mining on student behavior and cognitive style data for improving e-learning systems: a case study", *International Journal of Computational Intelligence Systems* ***5***(3), 597-610.

Kotsiantis, S. B. (2012), "Use of machine learning techniques for educational proposes: a decision support system for forecasting students' grades", *Artificial Intelligence Review* ***37***(4), 331-344.

Le, K., Chua, C., & Wang, R. (2017), "Mining software engineering team project work logs to generate formative assessment", In *2017 24th Asia-Pacific Software Engineering Conference Workshops (APSECW)*,78-83.

Lichman, M. (2013), "UC Irvine Machine Learning Repository. *University of California, "http://archive. ics. uci. edu/ml", Irvine, School of Information and Computer Sciences*.

Lykourentzou, I., Giannoukos, I., Nikolopoulos, V., Mpardis, G., & Loumos, V. (2009), "Dropout prediction in e-learning courses through the combination of machine learning techniques", *Computers & Education* ***53***(3), 950-965.

Macfadyen, L. P., & Dawson, S. (2010), "Mining LMS data to develop an early warning system for educators: A proof of concept", *Computers & education*, ***54***(2), 588-599.

Namous, F., Faris, H., Heidari, A. A., Khalafat, M., Alkhawaldeh, R. S., & Ghatasheh, N. (2020), "Evolutionary

and Swarm-Based Feature Selection for Imbalanced Data Classification", In *Evolutionary Machine Learning Techniques*, 231-250.

Naseer, M., Zhang, W., & Zhu, W. (2020), "Early Prediction of a Team Performance in the Initial Assessment Phases of a Software Project for Sustainable Software Engineering Education", *Sustainability* ***12***(11), 4663.

Pernkopf, F. (2005), "Bayesian network classifiers versus selective k-NN classifier", *Pattern recognition* ***38***(1), 1-10.

Petkovic, D., Barlaskar, S. H., Yang, J., & Todtenhoefer, R. (2018), "From Explaining How Random Forest Classifier Predicts Learning of Software Engineering Teamwork to Guidance for Educators" In *2018 IEEE Frontiers in Education Conference (FIE)*, 1-7.

Petkovic, D., Sosnick-Pérez, M., Huang, S., Todtenhoefer, R., Okada, K., Arora, S., ... & Dubey, S. (2014), "Setap: Software engineering teamwork assessment and prediction using machine learning", In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, 1-8.

Petkovic, D., Sosnick-Pérez, M., Okada, K., Todtenhoefer, R., Huang, S., Miglani, N., & Vigil, A. (2016), "Using the random forest classifier to assess and predict student learning of software engineering teamwork", In *2016 IEEE Frontiers in Education Conference (FIE)*, 1-7.

Reel, J. S. (1999), "Critical success factors in software projects", *IEEE software* ***16***(3), 18-23.

Sauer, C., & Cuthbertson, C. (2003), "The state of IT project management in the UK 2002-2003", *Computer Weekly* ***15***, 1-82.

Sauer, C., Gemino, A., & Reich, B. H. (2007), "The impact of size and volatility on IT project performance", *Communications of the ACM*, ***50***(11), 79-84.

Standish Group. (2009). "Chaos summary 2009. *Online report", Accessed June*, 2020.

Thai-Nghe, N., Horváth, T., & Schmidt-Thieme, L. (2011), "Factorization models for forecasting student performance", In *Educational Data Mining 2011*.

Xue, B., Zhang, M., & Browne, W. N. (2012), "Particle swarm optimization for feature selection in classification: A multi-objective approach", *IEEE transactions on cybernetics* ***43***(6), 1656-1671.

Zafra, A., & Ventura, S. (2009), "Predicting Student Grades in Learning Management Systems with Multiple Instance Genetic Programming", *International working group on educational data mining*.