


Article

Prediction of Coding Intricacy in a Software Engineering Team through Machine Learning to Ensure Cooperative Learning and Sustainable Education

Mehwish Naseer , Wu Zhang * and Wenhao Zhu

School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China; whzhu@shu.edu.cn

* Correspondence: mehwishnaseer@gmail.com (M.N.); wzhang@shu.edu.cn (W.Z.)

Received: 19 September 2020; Accepted: 26 October 2020; Published: 29 October 2020



Abstract: Coding deliverables are vital part of the software project. Teams are formed to develop a software project in a term. The performance of the team for each milestone results in the success or failure of the project. Coding intricacy is a major issue faced by students as coding is believed to be a complex field demanding skill and practice. Future education demands a smart environment for understanding students. Prediction of the coding intricacy level in teams can assist in cultivating a cooperative educational environment for sustainable education. This study proposed a boosting-based approach of a random forest (RF) algorithm of machine learning (ML) for predicting the coding intricacy level among software engineering teams. The performance of the proposed approach is compared with viable ML algorithms to evaluate its excellence. Results revealed promising results for the prediction of coding intricacy by boosting the RF algorithm as compared to bagging, J48, sequential minimal optimization (SMO), multilayer perceptron (MLP), and Naïve Bayes (NB). Logistic regression-based boosting (LogitBoost) and adaptive boosting (AdaBoost) are outperforming with 85.14% accuracy of prediction. The concerns leading towards high coding intricacy level can be resolved by discussing with peers and instructors. The proposed approach can ensure a responsible attitude among software engineering teams and drive towards fulfilling the goals of education for sustainable development by optimizing the learning environment.

Keywords: sustainable education; educational data mining; software engineering; machine learning; predictive modeling; boosting; ensembles

1. Introduction

Education for sustainable development (ESD) is allied with the qualified performance of all the pupils in the academic group and awareness of the responsibility of their learning [1]. Understanding and solving the snags and complications in the academic context are important aspects of achieving ESD. Therefore, ESD is fundamental in providing quality education. Learning outcomes are specially addressed concerns in ESD. Stimulating learning, endorsing critical thinking, and decision making are the fundamental extents of ESD [2]. Predictive modeling through machine learning (ML) and artificial intelligence (AI) is proved to be an effective resource in previous studies for accomplishing the aims of ESD [3]. Technological involvement in academia is taking educational practices to a whole new and improved level [4]. The concept of technology-enhanced learning (TEL) is about enhancing the upshots of learning and teaching by incorporating technological facilities. It has rehabilitated the traditional educational system and provided innovation in the educational domain [5].

Software engineering education is among the competitive educational fields. Software engineering involves attaining knowledge of the software process, theoretical concepts of software engineering,

and coding in the programming language. Coding is a complex process for students requiring lots of skills and practice. Difficulties in coding can impact a student's achievements and interests [6]. Coding deliverables are important components of the software project of the term. Student teams spend a considerable amount of time on their project for producing the coding deliverables along with achieving other milestones. Coding deliverables, therefore, are considered a significant standard for determining the overall performance of the team in the software project. Strict deadlines are associated with the accomplishment of a successful project. Therefore, time is an important resource of the software team [7]. It is necessary to utilize time effectively to achieve goals. A large number of projects face time overrun in the software industry [8]. Time management should be part of the training of software engineers. Software projects are designed in a software engineering course to train the students for the professional environment. However, coding intricacy is among the major problems faced by software engineering students that cause delayed projects. Predicting teams with a high level of coding intricacy during software project development can help in resolving the difficulties of the teams. Predictions can result in achieving co-operative learning among teams and instructors to make the academic environment lead to better learning. Support by peers can result in cooperative learning rather than pressurized learning. This research proposed a cooperative learning-based environment among software teams by predicting the coding intricacy level. On identification of teams having higher intricacy levels, instructors should also guide them about the concerns leading towards delays in producing coding deliverables.

ML has been anchored in the educational field in the form of educational data mining (EDM) [9]. This domain is further flourishing by adding in the ESD [10,11]. Algorithms of ML have been used to solve problems of education and data analysis. Educational data from e-learning systems are providing foundations for novel strategies of EDM. Data are readily available from the logs of e-learning systems. An e-learning system provides the capability to inevitably detect the learner's style of learning. This helps in the personalization of the learning system. Classification practices are mostly used to classify the learners according to their learning habits on the e-learning system. This has resulted in the formulation of several classification-based approaches for different styles of learning behaviors [12].

Several studies offer the ML-based methodologies for predicting difficulties in massive open online courses (MOOCs) [13–15], drop out of students from the e-learning system [16–18], and the engagement level of the students [19–22]. However, incorporating ML in project-based educational data is still an attention-demanding field. Predictive modeling on software engineering project data can lead to the attainment of ESD for software engineering.

This study focuses on ensuring cooperative learning and achieving sustainable education by predicting the high coding intricacy level among teams. The study leads to the development of an efficacious algorithm for the prediction of coding intricacy levels among software engineering teams through ML. The data set used for the research purpose was software engineering teamwork assessment and prediction (SETAP) product data [23]. It encompasses data of 74 teams residing locally or globally and working on the same software project. Assessment criteria for each team are defined in the form of team activity measures. These measures assess each team at different milestones and generate the performance grade accordingly. The data set distinguishes between software process data and software product data. Software product data were used for this study. Coding deliverables are one of the measures that determine the time in which the team produces the respective code. The time varies for each team. Prediction of teams facing higher coding intricacy levels can help in achieving better results of the software project. Exploring the reasons for the higher level of coding intricacy and resolving the concerns can aid in achieving ESD and co-operative learning. Predictive modeling can help in better decision-making strategies for the future. Waikato Environment for Knowledge Analysis (Weka) was developed at the University of Waikato, New Zealand. Weka v3.8. was used to apply ML algorithms for the prediction of teams facing high coding intricacy level.

1.1. Research Questions

This study intends to answer the following research questions:

- (1) Can we predict the coding intricacy level efficiently among software teams working on the same software project?
- (2) Can cooperative learning and ESD be achieved by predicting the high coding intricacy level among teams?

The structure of the paper is as follows. Section 2 presents an outline of related works and studies on the explicit topic. Section 3 defines and describes the proposed model for predicting coding intricacy. Section 4 delivers the results and evaluates the performance of the proposed model. Section 5 presents an exhaustive discussion, and Section 6 states the concluding thoughts and future recommendations.

2. Related Work

2.1. Difficulties in Learning Programming

Student behavior analysis is the foremost concern of EDM and learning analytics. It helps in understanding learners and making decisions accordingly for their betterment. It is directly impacting the goals of ESD as well. Coding difficulties for learners and their solution formulation carries importance in academia. Students face difficulties in coding and automated tools can help in teaching programming to students. Scratch and Alice are programming aiding tools. A positive impact on the reflective and computational thinking of students by programming teaching through Scratch can be achieved [24].

Team formulation and project-based learning for software engineering are quite effective. Teamwork and leadership skills are important for a software engineer to accomplish intricate projects. Authors of reference [25] made teams among software engineering students and evaluated their abilities to cope with project-based learning. Role rotation is offered as a significant parameter for project-based learning. This results in all the team members going through different types of challenges. Algorithm education is a solution to coding intricacy. Algorithm education can pointedly upsurge the student's aptitude to perform simple and complex programming tasks. However, it is not shown to raise their efficacy of computational thinking [26].

Students face substantial difficulties in learning programming. A questionnaire-based survey explored the motives for the fragility behind new programming learners. Tool support is a great way to overcome the hindrances in learning programming [27]. Programming-learning behavior of the students can be evaluated using different means. Knowing the learning pattern can help the instructor in the formulation of teaching style. Static analysis tools can be used to analyze the pattern of coding of the students. Error frequency in code is directly related to the overall course grade [28]. Incorporating ML for the prediction of students' concerns is a leading research domain in academia. Many studies offer novel methods and advantages of making student-related predictions. A learning management system (LMS) can complement the face to face learning in the classroom. Data collected from an LMS can predict a student's performance through their login activity in the LMS [29].

An online judge system for programming students can help them to compile, execute, and evaluate their code. This approach can aid in improving programming ability. The classification method is used to identify students "at-risk", by analyzing the log data of the online judge system. A deep neural network is trained on the data to classify the students as risky, intermediate, and advanced. This system proved its performance by correlating with the examination result [30].

2.2. Performance Prediction of Students in an E-Learning System

The performance of students in an e-learning system is of great interest for the researchers. In this regard, different approaches are contributing by considering multiple parameters of student behavior. Procrastination behavior can predict the performance of the students in an e-learning system.

These behavioral statistics link the assignment submission behavior with the success factor in the e-learning system. Clustering can predict the student's performance based on procrastination behavior with 96% accuracy [31].

Data analysis of students reveals imperative statistics of students that can help in perceiving their relation with the academic responsibilities. The survey presented in reference [32] showed how effectually different domains of AI, including ML, recommender systems, and collaborating filtering, are contributing to the development of innovative procedures and practices for enhancing student-academic relationships.

Students dropping out can be considered in terms of deteriorating the resources. It is central to appraise the factors contributing to increased drop out of students. Deployment of ML algorithms is helpful to explore the factors and potential students that are at risk of dropping out of the university. Decision tree, logistic regression, random forest, K nearest neighbor, and neural network can be trained for making predictions. Logistic regression is proved to be the superlative model for the case in reference [33].

2.3. ML for Predicting Programming Learning

Authors of reference [34] defined the potential components for learning computer programming as knowledge and proficiency. Student study performance is determined by applying the clustering technique. An optimal set of features is identified for students using the best subset selection and least absolute shrinkage and selection operator (LASSO) algorithm. The model is validated using a linear regression model with an accuracy of 76.52% for knowledge and 70.44% for skills in computer programming.

Reference [35] offered a deep learning recommendation system for programming. Students are divided into two groups. A group of students used the deep learning recommendation system for programming and the other one participated in the control group. Results declared the positive impact of a deep-learning-based recommendation system on the programming structure of the students. The two groups are evaluated according to computational thinking, creativity, logical computing, critical thinking, and problem-solving skills. The students using a deep-learning-based recommendation system performed better in the respective evaluations.

2.4. Potential Limitations in the Literature

The review of previous studies explains how ML has its influence on academics generally and for resolving computer programming issues specifically. Different methodologies suggest procedures that make coding easy for students. However, no technique is offered to determine the coding intricacy level among software engineering teams. Approaches of EDM and ESD collectively offer an optimized learning environment with quality education.

3. Materials and Methods

This section explains the materials and methods used to accomplish the anticipated objects of this study. The input of the proposed system was SETAP software product data of phase T11. The output of the approach is the best prediction of the coding intricacy level among software engineering teams. Figure 1 illustrates the phases of the process involved in the prediction of coding intricacy level among software engineering teams.

The proposed approach is composed of four phases as depicted in Figure 1:

- (1) Data Preparation
- (2) Prediction Phase
- (3) Evaluation Phase
- (4) Boosting Phase

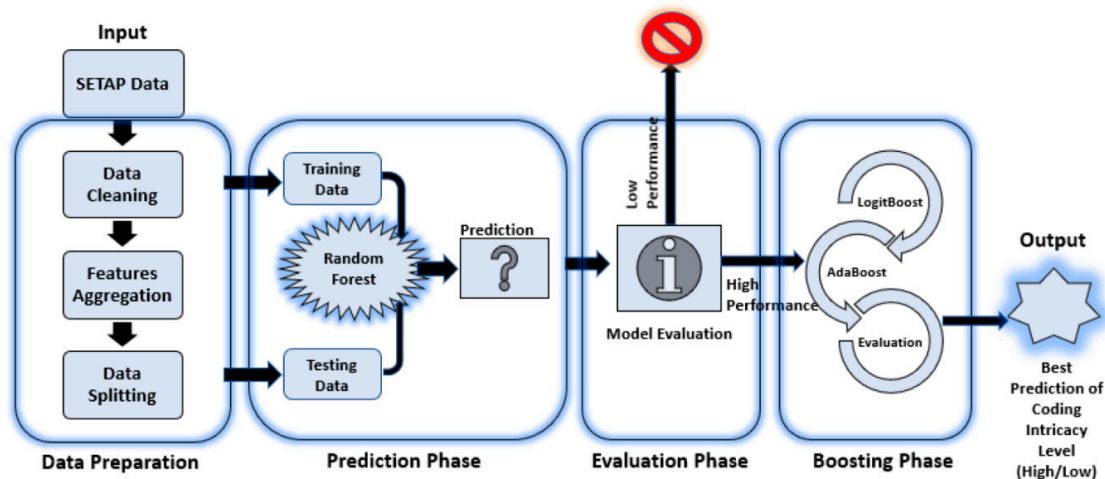


Figure 1. Proposed Approach for Prediction of Coding Intricacy Level among Software Engineering Team. SETAP = software engineering teamwork assessment and prediction, LogitBoost = logistic regression-based boosting, AdaBoost = adaptive boosting.

3.1. Data Preparation

SETAP data enter into the first phase called data preparation. SETAP is the data of software engineering students for assessment of progress in the software project. Data preparation involves the process of data cleaning, features aggregation, and data splitting.

3.1.1. Data Cleaning

NULL values and missing values are fixed in the original data. NULL values in SETAP data are accommodated by filling the average values of the fields having NULL values. Some teams are residing locally and others are global. If the values of local teams are stated as NULL for their global variables, they are substituted by 0.

3.1.2. Features Aggregation

Time is among the major contributors to assess the performance in the software project. codingDeliverablesHoursTotal, meetingHoursTotal, and helpHoursTotal are taken as predictors to assess coding intricacy level. codingDeliverablesHoursTotal is the time taken to produce the coding deliverables, meetingHoursTotal represents total time spent on meetings of the teams, and helpHoursTotal is time utilized for getting help. These three variables are combined according to the logic to form a new attribute: coding intricacy level. The value of coding intricacy level resides as “High” or “Low” conferring to the fulfillment of the below-mentioned condition.

If (codingDeliverablesHoursTotal) > Total mean (codingDeliverablesHoursTotal)
 and (meetingHoursTotal) > Total mean (meetingHoursTotal)
 and (helpHoursTotal) > Total mean (helpHoursTotal) then (High) otherwise (Low)

Features selection eliminates the least contributing variable. Features aggregation in contrast to features selection combines the available variables to form a new attribute. This method helps in dimensionality reduction [36].

3.1.3. Data Splitting

Input data are split into training and test data. K-folds cross-validation is used for training and validation of the model [37]. The model is trained on K-1 of individual folds and one fold is used to test the model. K rounds of the execution result in generating the predictions from the model. Ten folds cross-validation is used in the proposed approach for attaining reliable outcomes.

3.2. Prediction Phase

Training and testing data are fed into the prediction phase. Random forest (RF) is trained on the training data for making predictions of testing data through K-folds cross-validation. RF is a supervised learning algorithm of machine learning made up of the ensemble of decision trees [38]. Being a merger of multiple decision trees, it is expected to generate reliable predictions for coding intricacy levels. The RF takes a subdivision of random features from the data set for the building individual trees. Hence, the output of RF deals with assortment and leads to the best predictions.

3.3. Evaluation Phase

The efficacy of RF for predicting the coding intricacy level is assessed against other ML algorithms in the evaluation phase. Bagging, J48, sequential minimal optimization (SMO), multilayer perceptron (MLP), and Naïve Bayes (NB) are well-known algorithms of ML. They have been used in multiple studies for making predictions. Bagging is an ensemble classifier like RF that combines the output of weak classifiers to enhance the output [39]. J48 belongs to the class of decision trees. It can extract patterns from large and complex data sets. J48 is considered one of the efficient algorithms of data mining [40]. SMO is used to train the support vector machine. It solves the quadratic programming problems that arise during the training of the support vector machine [41]. MLP is the implementation of artificial neural networks. MLP is trained by adjusting the number of nodes, hidden layers, and weights for the classification of data. MLP has a huge contribution to the research [42]. NB is a probabilistic classifier. It evaluates the features to make assumptions about independence between them by applying Bayes theorem. The algorithm possesses a viable contribution to the field of data mining [43].

Accuracy, receiver operating curve (ROC), Kappa, and root mean squared error (RMSE) values of each model are computed to appraise against the proposed approach to demonstrate the performance of the proposed RF-based model for prediction of coding intricacy level of the software engineering team. ROC relates the true positive rate (sensitivity) with a false-positive rate (1-specificity) [44].

An ROC can be interpreted to depict the efficiency of the model as shown in Table 1.

Table 1. Interpretation of the Receiver Operating Curve (ROC).

ROC Area	Interpretation
More than 0.9	Exceptional
From 0.9 to 0.8	Good
From 0.8 to 0.7	Acceptable
From 0.7 to 0.6	Reasonable
Less than 0.6	Poor

A Kappa value indicates how closely the machine learning algorithm predicts the labels to the actual labels [45]. Kappa values can be interpreted using Table 2.

Table 2. Interpretation of Kappa Values for Machine Learning (ML) Model.

Kappa Value	Interpretation
0.61–0.80	Considerable agreement between predicted and actual labels
0.41–0.60	Moderate agreement between predicted and actual labels
0.21–0.40	Fair agreement between predicted and actual labels
0.01–0.20	No to a slight agreement between predicted and actual labels
values ≤ 0	No agreement between predicted and actual labels

The study is focused on determining the software teams with a high level of coding intricacy for the resolution of their concerns. Therefore, the performance of each model is determined by keeping an

eye towards the identification of a “High” level class. F-measure, recall, and precision are determined and evaluated for the high predicted class through each model.

3.4. Boosting Phase

Boosting is an ensemble-based approach used in ML to enhance the performance of the base classifier [46]. RF is an ensemble of multiple decision trees, that aggregates the performance of each random tree to generate the vindicated predictions. However, this study proposed a boosting-based RF approach for predicting the coding intricacy level of software engineering teams. Adaptive boosting (AdaBoost) and logistic regression-based boosting (LogitBoost) are recognized techniques to boost the prediction capability of the base classifier. AdaBoost combines the output of weak learners to generate boosted output. AdaBoost minimizes the exponential loss [47]. LogitBoost is similarly a boosting classification algorithm. LogitBoost minimizes the logistic loss [48].

Both boosting algorithms are applied by taking RF as a base classifier to check whether boosting increases the efficiency of the proposed RF model or not for predicting coding intricacy level.

4. Results

Prepared SETAP data were fed into the RF-based proposed model for predicting coding intricacy level. Services of bagging, J48, SMO, MLP, and NB were utilized to observe the predictions compared to the proposed approach. Table 3 depicts the performance metrics of each model from the perspective of accuracy, RMSE, and Kappa statistics for engendering predictions after the evaluation phase.

Table 3. Performance Metrics of Models.

Model	Accuracy	RMSE	Kappa
RF	82.43%	0.3430	0.5850
Bagging	82.43%	0.3541	0.5623
J48	78.38%	0.4584	0.5083
SMO	78.38%	0.4650	0.4686
MLP	72.97%	0.5023	0.3357
NB	72.97%	0.5059	0.4003

RMSE = root mean squared error, RF = random forest, sequential minimal optimization (SMO), multilayer perceptron (MLP), Naïve Bayes (NB).

Results from Table 3 illustrate that the RF-based proposed model and bagging are performing with the best accuracy of 82.43%. RF is an ensemble-based concept based on bagging the outputs of multiple decision trees; therefore, results are quite similar. Kappa values are illustrating the best predictions through the RF model. The boosting phase results in the application of LogitBoost and AdaBoost by taking RF as a base classifier. Improved accuracy, reduction in RMSE, and improved value of Kappa by boosting the RF can be observed in Table 4.

Table 4. Performance Evaluation of Model by Boosting.

Model	Accuracy	RMSE	Kappa
LogitBoost	85.14%	0.3380	0.6395
AdaBoost	85.14%	0.3431	0.6297
RF	82.43%	0.3430	0.5850

Values for accuracy of LogitBoost–RF and AdaBoost–RF are the same. Figure 2 illustrates the overall comparative evaluation of the accuracy of each model and boosted RF.

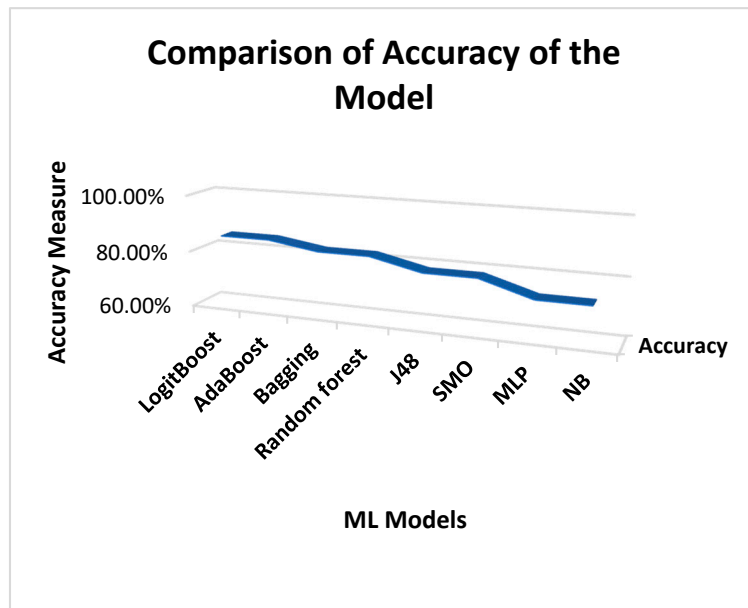


Figure 2. Comparison of Accuracy of Models.

The value of accuracy is portraying that boosting significantly aided the results. Further refinement of the results is depicting greater Kappa value and reduced RMSE through the LogitBoost approach. The higher the value of Kappa, the better the agreement between predicted and actual labels as demonstrated in Table 2 (Section 3).

The goals of the study revolved specifically around the determination of teams having high intricacy levels. Those teams need attention from the instructor and cooperation of peers to resolve the concerns leading towards troubles causing a delay in producing coding deliverables. Results generated through comparative measures are depicted through Pareto charts.

Pareto charts are composed of bar and line graphs [49]. The left vertical axis is a measure of the parameter value precision, recall, or f-measure. The right vertical axis depicts the cumulative percentage of the total value of the parameter. The horizontal axis represents the respective model. The Pareto chart highlights the prominent algorithm for the corresponding evaluation. Results are promising by boosting the RF. Both practices, LogitBoost and AdaBoost, enhanced the prediction capability of RF. Pareto charts in Figures 3–5 enlighten the perception of precision, recall, and f-measure, respectively.

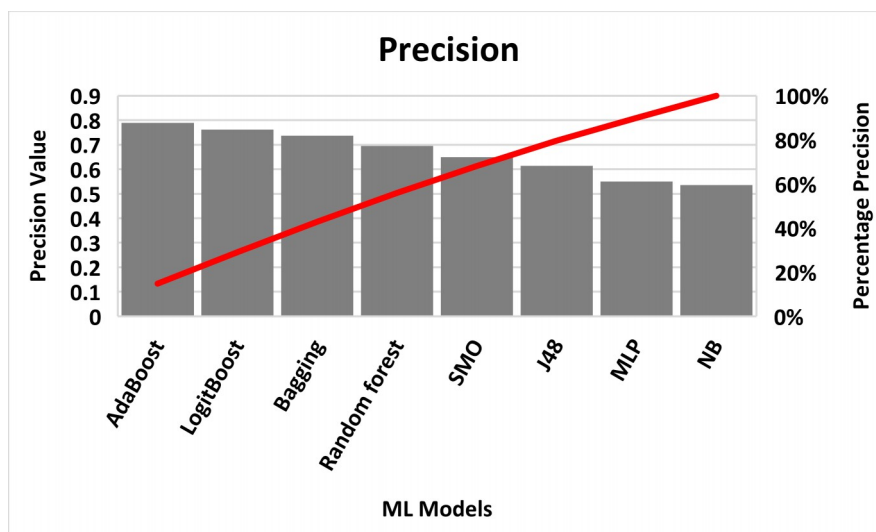


Figure 3. Pareto Chart of Precision.

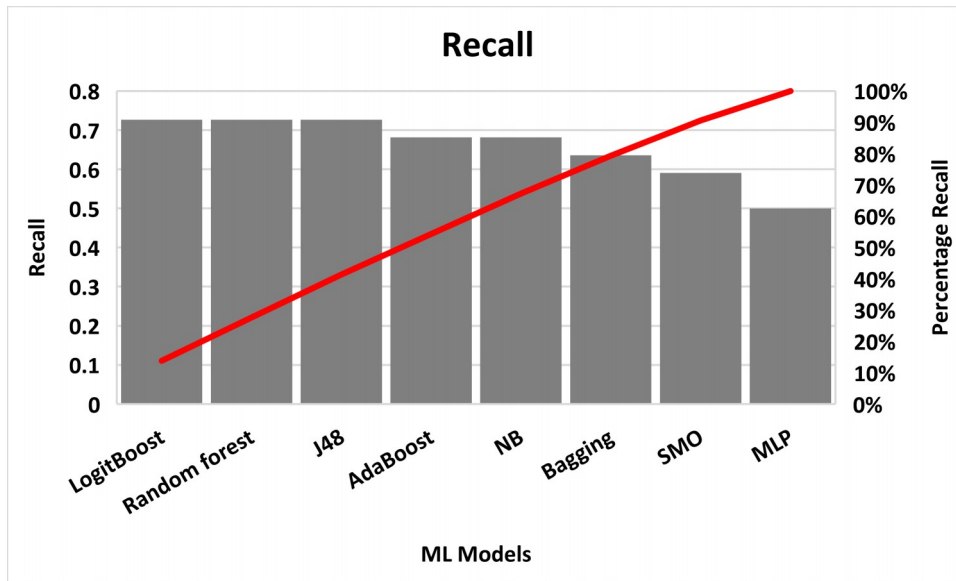


Figure 4. Pareto Chart of Recall.

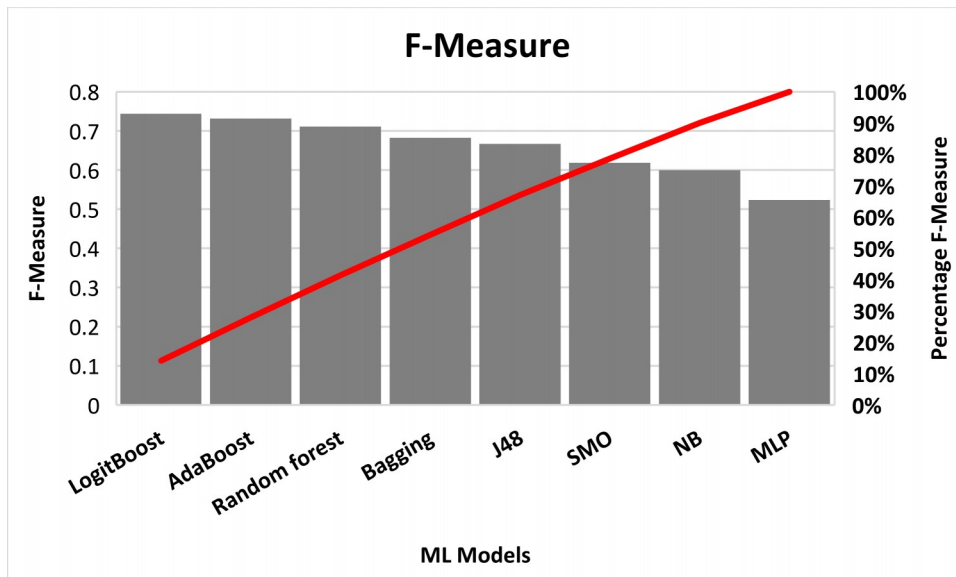


Figure 5. Pareto Chart of F-Measure.

Precision represents the ratio of true positive to the total identified instances predicted as high. Figure 3 is illustrating that boosting approaches are leading with an increased value precision for identifying the high coding intricacy level.

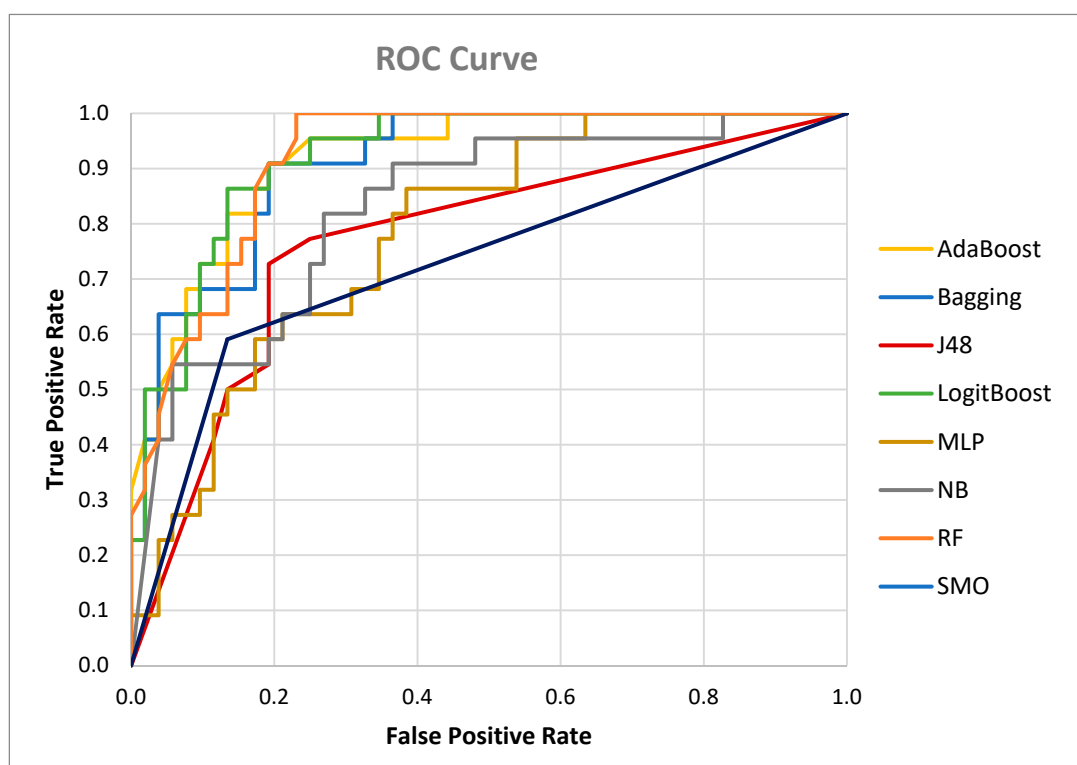
The recall represents the ratio between true positive and total actual high labels available in the data set. Figure 4 depicts LogitBoost is leading with the highest recall value for the prediction of high coding intricacy among software engineering teams.

F-measure evaluates the model’s accuracy on the test data. F-measure is the harmonic mean of precision and recall for prediction of high coding intricacy. Figure 5 is declaring boosting approaches as victors with the highest f-measure value. Overall applying boosting on RF proved to be an outperforming strategy for prediction of coding intricacy levels of software engineering teams. Table 5 represents the ROC-based analysis measures for prediction of the high level of coding intricacy in software engineering teams.

Table 5. ROC-Based Analysis for Prediction of High level of Coding Intricacy.

Model	True Positive Rate	False Positive Rate	ROC Area
LogitBoost	0.727	0.096	0.922
AdaBoost	0.682	0.077	0.920
Bagging	0.636	0.096	0.911
Random forest	0.727	0.135	0.920
J48	0.727	0.192	0.771
SMO	0.591	0.135	0.728
MLP	0.500	0.173	0.783
NB	0.682	0.250	0.828

The ROC area represents the area under the ROC curve. Interpretation of the ROC area according to the proportions given in Table 1 of Section 3 demonstrates the exceptional performance of LogitBoost among other competing algorithms. Therefore, boosting the RF provided a substantial boost to the accuracy of predicted outcomes. Figure 6 illustrates an ROC curve to visualize the ROC area and performance of all models.

**Figure 6.** ROC Curve of Models for Predicting High Level of Coding Intricacy.

The ROC curve is formed by relating the false positive rate to the true positive rate. Figure 6 represents the ROC curve for prediction of high coding intricacy level of software engineering teams. The ROC area of LogitBoost is 0.922 and AdaBoost is 0.920.

Results revealed that the proposed approach of boosting the RF significantly increased the capability of the model for predicting the coding intricacy level of the student. It is taking software engineering from a project-oriented learning platform to a smart prediction-based environment for the betterment of learners and instructors.

5. Discussion

Predicting student coding behavior is a challenging and exciting problem. Educational predictions and perceiving student capabilities are focused concerns of ESD for attaining a smart teaching and learning environment. Online education on a learning management system gives the advantage of perceiving a student's caliber through clicks and streaming information [50]. A smart academic environment is specifically concerned with the identification of alarming situations about students such as dropping out issues, low-performance issues, or low engagement in the system [51]. Cooperative learning is a mode of learning which supports learning from peers. Its basic idea revolves around student groups in which students learn from each other [52]. The project-oriented study involves teams having many students working together and learning from each other. Future education demands a smart environment for project-based courses, as well, where teams are located globally or locally for the project and ensuring cooperative learning. Software engineering teams are the best case of observing such a course. A software project plays a key role in the successful completion of the course. However, the collection of student data regarding the project is a foremost hurdle in such type of courses, as data are not available from any log. Time to time assessment is required to observe students constantly. The team activity measures express their participation in the project. SETAP is data of software engineering students composed of detailed teams activity measures. The data set is quite comprehensive and carries much-required information for performing research.

This study utilized a predictive modeling approach using ML to attain a smart environment for software engineering students. This system is helpful for the personalization of the project-based learning course and achieving ESD goals. The provision of quality education with a higher success rate is among the primary goals of ESD [53]. Prediction of coding intricacy is a prime concern of software engineering students. Cooperative learning can aid in dealing with the problem by learning from peers and improvement of work behavior [54]. The proposed approach of predicting coding intricacy level addresses an imperative problem of software engineering students regarding delayed delivery of coding deliverables due to high coding intricacy. Taking time as a primary concern, coding intricacy level was determined for the teams. Features aggregation was used to evaluate the coding intricacy level of the teams as high or low. Teams facing a high level of coding intricacy during software engineering projects were predicted using boosting of the RF model. The proposed model resulted in generating the highest accuracy and f-score for predicting the teams facing high coding intricacy.

The contributions of this study can be summarized by answering the research questions raised in Section 1.1 (Research Questions) of Section 1 (Introduction).

- (1) Can we predict the coding intricacy level efficiently among software teams working on the same software project in a term? Yes, through the proposed approach of boosting of an RF model, coding intricacy levels of different teams working on the same project in a term can be predicted. Results of the proposed approach verdict its efficiency by thoroughly comparing it with other effective approaches of ML. Through the proposed strategy, 85.14% accuracy was achieved.
- (2) Can cooperative learning and ESD goals be achieved by predicting the coding intricacy level among teams? Yes, predictions about students can help as a second eye for the instructors to perceive the apprehensions of the students. Predicting the coding intricacy level of software engineering teams is an aid for assisting the instructors in determining the students' concerns, which otherwise could be very difficult. Student-instructor healthy discussions and support sessions can help resolve the issues that are leading to high coding intricacy. Peer support and guidance can help address issues of high intricacy level. Thus, the prediction of coding intricacy levels is directly influencing the learning space. It would be a great source for producing a cooperative learning environment [55]. This will eventually lead towards the accomplishment of the goals of ESD by optimizing the educational environment.

To ensure cooperative learning in a software engineering team, there are dimensions required to be encountered by the team such as positive interdependence, individual accountability, promotive

interaction, appropriate use of social skills, and group processing [56]. The proposed approach of prediction of coding intricacy is a mechanism leading towards the accomplishment of attributes of cooperative learning. There exists a positive interdependence when the learning outcome of the team members is directly related to the activities done by all the affiliates of the team. Software engineering teams work with role assignments of team members. Therefore, trouble in the coding part can lead to the disaster of the project. Predicting the coding intricacy lays the foundation for individual accountability for ensuring cooperative learning. It is related to the communication between all the team members. Communication among the team can help in resolving the apprehensions in coding. Promotive interaction is part of software projects already. Teamwork is associated with the success of the project. So all the members have to be responsible for their part of work to attain promotive interactions for ensuring cooperative learning. Social skills play an important role in determining the complications associated with the coding intricacy of the project. Operative communication, supporting each other, and conflict resolution can be achieved by effective usage of social skills. This approach will eventually aid in making the project successful. Group processing can help monitor the reasons for coding intricacy in a team. This attribute determines what went well and what did not. It can be a source of optimizing the work environment in a software team. Therefore, cooperative learning can be achieved by predicting coding intricacy among the team.

Section 4 (Proposed Methodology) presented a thorough comparison of the results of the proposed boosted RF (LogitBoost, AdaBoost) approach with the previous state of the art methods using different metrics. The proposed approach was outperforming with the highest value of accuracy, precision, recall, F-measure, kappa, and ROC. The proposed approach has the lowest RMSE value as compared to other approaches.

Education is among the prime composites of sustainable development. ESD devised some goals for its fulfillment in general education practices. Software engineering follows a project-based pedagogical approach. Competences associated with ESD, such as systems thinking, interdisciplinary work, anticipatory thinking, critical thinking and analysis, communication and use of media, strategic actions, personal involvement, assessment and evaluation, and tolerance for ambiguity and uncertainty, provide good coverage to project-based learning [57]. ESD is a learner-centered approach that considers the learner's perspective as prime [58]. This is intended for challenging learners to partake enthusiastically, think critically, and imitate [59]. Critical thinking and active participation of team members for resolution of concerns of coding intricacy are fulfilling the goals of ESD. Predictions, therefore, directly encourage the achievement of the goals of ESD by looking into the future and making foundations for decision making in favor of learners. Coding challenges are among the prime concerns of software engineering students. Teamwork in the software engineering course project has a foundation for cooperative learning as teams are representing groups. Cooperative learning can be strengthened by the prediction of major complications in a project such as coding intricacy. Hence, the proposed approach helps to promote cooperation and interaction among peers to ensure cooperative learning alongside the achievement of ESD goals. The proposed approach is presenting a strategy for dealing with the coding problems by early identification of teams for resolution of their issues. A smart software engineering project environment to predict the high coding intricacy of the teams provides a forum of co-operative learning, responsible learning, and sustainable education.

6. Conclusions and Future Work

The paper proposed a boosting-based approach of an RF classifier for predicting the coding intricacy of software engineering teams. Features aggregation is used for developing the intricacy class with labels high and low. It is significant to predict the teams expected to face the high intricacy level. Coding intricacy is among the major issues faced by the students. Many tools for supporting coding are designed to assist the students in learning programming. However, sometimes only minor issues faced during coding can cause a delay in the delivery of the project. Early prediction of coding intricacy can be a great source of transforming the traditional software project environment. Cooperative

learning is ensured by fulfilling the prominent domains using the proposed approach. ESD goals can be accomplished by resolving the problems of teams facing high coding intricacy through cooperative learning and critical thinking. The cooperative learning environment with the aid of technology can be the best revolution of the project-oriented academic environment. The proposed approach has its direct way to the triumph of sustainable education. In the future, this work can be extended further by the utilization of deep-learning models. Approaches of recommender systems can assist in the evaluation of SETAP data for making useful recommendations that can aid in lowering the coding intricacy level of the teams.

Author Contributions: Conceptualization, M.N.; Formal analysis, M.N.; Funding acquisition, W.Z. (Wu Zhang); Methodology, M.N. and W.Z. (Wenhao Zhu); Resources, M.N. and W.Z. (Wenhao Zhu); Software, M.N.; Supervision, W.Z. (Wu Zhang); Validation, M.N. and W.Z. (Wenhao Zhu); Visualization, M.N.; Writing—original draft, M.N. All authors have read and agreed to the published version of the manuscript.

Funding: The work of this paper is supported by the National Natural Science Foundation of China (Nos.61572434, 91630206 and 61303097) and the National Key R&D Program of China (No.2017YFB0701501).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. United Nations. Sustainable Development Goals. Available online: <http://www.undp.org/content/undp/en/home/sustainable-development-goals.html> (accessed on 15 September 2020).
2. UNESCO Roadmap for Implementing the Global Action Programme on Education for Sustainable Development. Available online: <https://unesdoc.unesco.org/ark:/48223/pf0000230514> (accessed on 15 September 2020).
3. Ricardo, V.; Azizpour, H.; Leite, I.; Balaam, M.; Dignum, V.; Domisch, S.; Felländer, A.; Max, L.; Tegmark, S.D.; Nerini, F.F. The role of artificial intelligence in achieving the Sustainable Development Goals. *Nat. Commun.* **2020**, *11*, 1–10.
4. Laurence, H.; Johannesen, M. The role of academic management in implementing technology-enhanced learning in higher education. *Technol. Pedagog. Educ.* **2020**, *29*, 129–146.
5. Chien-wen, S.; Ho, J.-T. Technology-enhanced learning in higher education: A bibliometric analysis with latent semantic approach. *Comput. Hum. Behav.* **2020**, *104*, 106177.
6. Denner, J.; Werner, L.; Ortiz, E. Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Comput. Educ.* **2012**, *58*, 240–249. [CrossRef]
7. Miikka, K.; Mäntylä, M.; Farooq, U.; Claes, M. Time pressure in software engineering: A systematic review. *Inf. Softw. Technol.* **2020**, *121*, 106257.
8. Lior, F.; Pinchovski, B. It is about time: Bias and its mitigation in time-saving decisions in software development projects. *Int. J. Proj. Manag.* **2020**, *38*, 99–111.
9. Cristobal, R.; Ventura, S. Educational data mining and learning analytics: An updated survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2020**, *10*, e1355.
10. Sung-Shun, W.; Liu, Y.; Dai, J.; Chuang, Y.-C. A Novel Improvement Strategy of Competency for Education for Sustainable Development (ESD) of University Teachers Based on Data Mining. *Sustainability* **2020**, *12*, 2679.
11. Mehwish, N.; Zhang, W.; Zhu, W. Early Prediction of a Team Performance in the Initial Assessment Phases of a Software Project for Sustainable Software Engineering Education. *Sustainability* **2020**, *12*, 4663.
12. Ibtissam, A.; Jeghal, A.; Radouane, A.; Yahyaouy, A.; Tairi, H. A robust classification to predict learning styles in adaptive e-learning systems. *Educ. Inf. Technol.* **2020**, *25*, 437–448.
13. Khe Foon, H.; Hu, X.; Qiao, C.; Tang, Y. What predicts student satisfaction with MOOCs: A gradient boosting trees supervised machine learning and sentiment analysis approach. *Comput. Educ.* **2020**, *145*, 103724.
14. Eyal, R.; Henderikx, M.; Yoram, K.; Kalz, M. What are the barriers to learners' satisfaction in MOOCs and what predicts them? The role of age, intention, self-regulation, self-efficacy and motivation. *Australas. J. Educ. Technol.* **2020**, *36*, 119–131.
15. Abdessamad, C.; Faddouli, N.-E.E. BERT and Prerequisite Based Ontology for Predicting Learner's Confusion in MOOCs Discussion Forums. In *International Conference on Artificial Intelligence in Education*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 54–58.

16. Mubarak, A.A.; Cao, H.; Ahmed, S.A. Predictive learning analytics using deep learning model in MOOCs' courses videos. *Educ. Inf. Technol.* **2020**. [[CrossRef](#)]
17. Moreno-Marcos, P.M.; Muñoz-Merino, P.J.; Maldonado-Mahauad, J.; Pérez-Sanagustín, M.; Alario-Hoyos, C.; Kloos, C.D. Temporal analysis for dropout prediction using self-regulated learning strategies in self-paced MOOCs. *Comput. Educ.* **2020**, *145*, 103728. [[CrossRef](#)]
18. Ya, Z.; Xu, Z. Multi-Model Stacking Ensemble Learning for Dropout Prediction in MOOCs. *J. Phys. Conf. Ser.* **2020**, *1607*, 012004.
19. Ruth, C.; Ruiz-García, J.C. Improving learner engagement in MOOCs using a learning intervention system: A research study in engineering education. *Comput. Appl. Eng. Educ.* **2020**. [[CrossRef](#)]
20. Min, L.; Hew, K.F. Examining learning engagement in MOOCs: A self-determination theoretical perspective using mixed method. *Int. J. Educ. Technol. High. Educ.* **2020**, *17*, 1–24.
21. Yongqiang, S.; Guo, Y.; Zhao, Y. Understanding the determinants of learner engagement in MOOCs: An adaptive structuration perspective. *Comput. Educ.* **2020**, *157*, 103963.
22. Ruiqi, D.; Benckendorff, P.; Gannaway, D. Learner engagement in MOOCs: Scale development and validation. *Br. J. Educ. Technol.* **2020**, *51*, 245–262.
23. UCISETAP Database at UC Irvine Machine Learning Archive. Available online: <https://archive.ics.uci.edu/ml/datasets/Data+for+Software+Engineering+Teamwork+Assessment+in+Education+Setting> (accessed on 18 July 2020).
24. Durak, H.Y. The effects of using different tools in programming teaching of secondary school students on engagement, computational thinking and reflective thinking skills for problem solving. *Technol. Knowl. Learn.* **2020**, *25*, 179–195. [[CrossRef](#)]
25. Pérez, B.; Rubio Á, L. A project-based learning approach for enhancing learning skills and motivation in software engineering. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education, Portland, OR, USA, 11–14 March 2020; pp. 309–315.
26. Türker, P.M.; Pala, F.K. The Effect of Algorithm Education on Students' Computer Programming Self-Efficacy Perceptions and Computational Thinking Skills. *Int. J. Comput. Sci. Educ. Sch.* **2020**, *3*, 19–32.
27. Aissa, M.; Al-Kalbani, M.; Al-Hatali, S.; BinTouq, A. Novice Learning Programming Languages in Omani Higher Education Institution (Nizwa University) Issues, Challenges and Solutions. In *Sustainable Development and Social Responsibility*; Springer: Berlin/Heidelberg, Germany, 2020; Volume 2, pp. 143–148.
28. Albluwi, I.; Salter, J. Using static analysis tools for analyzing student behavior in an introductory programming course. *Jordanian J. Comput. Inf. Technol.* **2020**. [[CrossRef](#)]
29. Mozahem, N.A. Using Learning Management System Activity Data to Predict Student Performance in Face-to-Face Courses. *Int. J. Mob. Blended Learn.* **2020**, *12*, 20–31. [[CrossRef](#)]
30. Xu, B.; Yan, S.; Jiang, X.; Feng, S. SCFH: A Student Analysis Model to Identify Students' Programming Levels in Online Judge Systems. *Symmetry* **2020**, *12*, 601. [[CrossRef](#)]
31. Hooshyar, D.; Pedaste, M.; Yang, Y. Mining Educational Data to Predict Students' Performance through Procrastination Behavior. *Entropy* **2020**, *22*, 12. [[CrossRef](#)]
32. Rastrollo-Guerrero, J.L.; Gómez-Pulido, J.A.; Durán-Domínguez, A. Analyzing and Predicting Students' Performance by Means of Machine Learning: A Review. *Appl. Sci.* **2020**, *10*, 1042. [[CrossRef](#)]
33. Yaacob, W.W.; Sobri, N.M.; Nasir, S.M.; Norshahidi, N.D.; Husin, W.W. Predicting Student Drop-Out in Higher Institution Using Data Mining Techniques. *J. Phys. Conf. Ser.* **2020**, *1496*, 012005. [[CrossRef](#)]
34. Ninrutsirikun, U.; Imai, H.; Watanapa, B.; Arpnikanondt, C. Principal Component Clustered Factors for Determining Study Performance in Computer Programming Class. *Wirel. Pers. Commun.* **2020**. [[CrossRef](#)]
35. Lin, P.H.; Chen, S.Y. Design and Evaluation of a Deep Learning Recommendation Based Augmented Reality System for Teaching Programming and Computational Thinking. *IEEE Access* **2020**, *8*, 45689–45699. [[CrossRef](#)]
36. Trevizan, B.; Chamby-Diaz, J.; Bazzan, A.L.; Recamonde-Mendoza, M. A comparative evaluation of aggregation methods for machine learning over vertically partitioned data. *Expert Syst. Appl.* **2020**. [[CrossRef](#)]
37. Tadayoshi, F. Estimation of prediction error by using K-fold cross-validation. *Stat. Comput.* **2011**, *21*, 137–146.
38. Oshiro, T.M.; Perez, P.S.; Baranauskas, J.A. How many trees in a random forest? In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 154–168.

39. Skurichina, M.; Duin, R.P. Bagging, boosting and the random subspace method for linear classifiers. *Pattern Anal. Appl.* **2002**, *5*, 121–135. [[CrossRef](#)]
40. Bhargava, N.; Sharma, G.; Bhargava, R.; Mathuria, M. Decision tree analysis on j48 algorithm for data mining. *Proc. Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **2013**, *3*, 1114–1119.
41. Shevade, S.K.; Keerthi, S.S.; Bhattacharyya, C.; Murthy, K.R.K. Improvements to the SMO algorithm for SVM regression. *IEEE Trans. Neural Netw.* **2000**, *11*, 1188–1193. [[CrossRef](#)]
42. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Philip, S.Y. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**. [[CrossRef](#)]
43. Sen, P.C.; Hajra, M.; Ghosh, M. Supervised classification algorithms in machine learning: A survey and review. In *Emerging Technology in Modelling and Graphics*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 99–111.
44. Fawcett, T. An introduction to ROC analysis. *Pattern Recognit. Lett.* **2006**, *27*, 861–874. [[CrossRef](#)]
45. Viera, A.J.; Joanne, M.G. Understanding interobserver agreement: The kappa statistic. *Fam. Med.* **2005**, *37*, 360–363.
46. Sagi, O.; Rokach, L. Ensemble learning: A survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1249. [[CrossRef](#)]
47. Ying, C.; Qi-Guang, M.; Jia-Chen, L.; Lin, G. Advance and prospects of AdaBoost algorithm. *Acta Autom. Sin.* **2013**, *39*, 745–758.
48. Pham, B.T.; Prakash, I. Evaluation and comparison of LogitBoost Ensemble, Fisher’s Linear Discriminant Analysis, logistic regression and support vector machines methods for landslide susceptibility mapping. *Geocarto Int.* **2019**, *34*, 316–333. [[CrossRef](#)]
49. Stevenson, W.J. Supercharging your Pareto analysis. *Qual. Prog.* **2000**, *33*, 51.
50. Moreno-Marcos, P.M.; Pong, T.C.; Muñoz-Merino, P.J.; Kloos, C.D. Analysis of the factors influencing learners’ performance prediction with learning analytics. *IEEE Access* **2020**, *8*, 5264–5282. [[CrossRef](#)]
51. Rajabalee, B.Y.; Santally, M.I.; Rennie, F. A study of the relationship between students’ engagement and their academic performances in an eLearning environment. *E-Learn. Digit. Media* **2020**, *17*, 1–20. [[CrossRef](#)]
52. Jacobs, G.M.; Ivone, F.M. Infusing Cooperative Learning in Distance Education. *TESL-EJ* **2020**, *24*, 1.
53. Pigozzi, M.J. Quality in education defines ESD. *J. Educ. Sustain. Dev.* **2007**, *1*, 27–35. [[CrossRef](#)]
54. Van Ryzin, M.J.; Roseth, C.J. The Cascading Effects of Reducing Student Stress: Cooperative Learning as a Means to Reduce Emotional Problems and Promote Academic Engagement. *J. Early Adolesc.* **2020**. [[CrossRef](#)]
55. Topping, K.J. Peer Tutoring and Cooperative Learning. *Oxf. Res. Encycl. Educ.* **2020**. [[CrossRef](#)]
56. Cañabate, D.; Serra, T.; Bubnys, R.; Colomer, J. Pre-Service Teachers’ Reflections on Cooperative Learning: Instructional Approaches and Identity Construction. *Sustainability* **2019**, *11*, 5970. [[CrossRef](#)]
57. Lozano, R.; Merrill, M.Y.; Sammalisto, K.; Ceulemans, K.; Lozano, F.J. Connecting Competences and Pedagogical Approaches for Sustainable Development in Higher Education: A Literature Review and Framework Proposal. *Sustainability* **2017**, *9*, 1889. [[CrossRef](#)]
58. Berglund, T. *Student Views of Environmental, Social and Economic Dimensions of Sustainable Development and Their Interconnectedness: A Search for the Holistic Perspective in Education for Sustainable Development*; Karlstads Universitet: Karlstad, Sweden, 2020.
59. Hoogeveen, P.; Winkels, J. Het Didactische Werkvormenboek. Variatie en Differentiatie in de Praktijk. In *Teaching Methods Book. Variation and Differentiation in Practice*; Uitgeverij Van Gorcum: Assen, The Netherlands, 1996.

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).