JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 8, No. 1, Maret 2023, Pp. 077-083

PREDICTION OF A SPRINT DELIVERY'S CAPABILITIES IN ITERATIVE-BASED SOFTWARE DEVELOPMENT

Clements Enrico Bramantyo Hady*[1)], Dana Sulistyo Kusumo[2)]
1. Telkom University, Indonesia
2. Telkom University, Indonesia

Article Info
Keywords: prediction model, software development, sprint

Article history:
Received 26 October 2022
Revised 9 November 2022
Accepted 16 November 2022
Available online March 2023

DOI :
https://doi.org/10.29100/jipi.v8i1.3292

* Corresponding author.
Corresponding Author
E-mail address:
clementsceb@student.telkomuniversity.ac.id

ABSTRACT
Iterative-based software development has been frequently implemented in working environment. A modern era software project demands that the product is delivered on every sprint development. Hence, the execution of a sprint requires ample supervision and capabilities to deliver a high quality product at the end of the software project development. This research's purpose is to give support for a software project's supervisor or owner in predicting the end product's capability by knowing the performance level of each sprint. The method proposed for this purpose is to build a prediction model utilizing a number of features in a form of characteristics from a dataset containing software project iterations. The proposed model is built using Random Forest Regressor as a main method, with KNN (K-Nearest Neighbors) and Decision Tree Regressor being the comparison methods. Testing results show that compared to KNN and Decision Tree, Random Forest Regressor yields the best performance through its steady results on every stage progression of all tested software projects.

I. INTRODUCTION

MODERN software development of today is often based on an iterative-based approach where the software is developed and tested in an iterative cycle known as sprints [1]. This approach benefits the developer because the end product will always get inputs or feedback from previous study results. This is one of the implementations [2] of sprints. Sprints are the end results of Scrum [3] [4] [5] which is one of the most widely used software development methods.

However, there is always uncertainty [6] in software development projects. Due to the dynamic nature of the work, through constant changes and inputs to sprints, changes to product requirements can occur. Some other examples of its uncertainties are the possibilities of having late sprint deliveries or excessive use of budgets [7] in the development of said software project. To deal with this, careful planning, progress monitoring, and constant interactions regarding project risk identification [7] [8] with the Scrum team [9] and customers are essential in iterative-based software development.

This study focuses on the ability of the prediction model to predict the level of sprint delivery in the form of performance levels of each software projects at certain intervals. Predictions are given based on available features or issues in a sprint which includes several matters such as work duration, change logs [7], scale of work and also the status of a sprint [7]. By studying the characteristics of features that may affect the performance of each sprint, a prediction model is built to predict whether the execution of a software project is able to achieve the specified work target.

Our study is loosely based on similar studies regarding software development projects. Our references are studies in software projects regarding effort and duration estimation of a project [10], story points estimation [11], and a prediction of bug fixing time [12]. There is also a prediction of delivery capabilities [7] that utilized a directed acyclic graph formed from the dependencies of one or more issues that blocked the resolving of another issue in a sprint.

However, unlike the previously mentioned study regarding delivery capability prediction [7], our study did not utilize the dependencies of issues. This obstacle is due to constraints met on both research time and the absence of issue dependencies on the obtained dataset [7]. Instead, we are building a simpler prediction model using parameters in the form of available features with the highest correlation value. The utilization of a simpler model is done with the thought of giving an alternative option by giving out another method on how to predict the performance level of a software project.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 8, No. 1, Maret 2023, Pp. 000-000

Hence, this study focuses on predicting whether the given work target can be achieved at the end of each stage progression of the software project. Through the prediction results, the project owner or supervisor can find out how much the project development performs at each stage of its working progress. By knowing the level of performance at each interval, the owner or supervisor can estimate the quality of work that can be achieved in the overall result of the software project.

II. RESEARCH METHODOLOGY

A. System Workflow

The workflow of the built prediction system begins with data preprocessing. Preprocessing is initiated by deleting unnecessary columns. Then some columns containing categorical data type are encoded into numerical data type. After that, the iteration dataset is aggregated with the issue dataset. The preprocessed data will then be used to build a prediction model. Figure 1 displays our system's workflow.

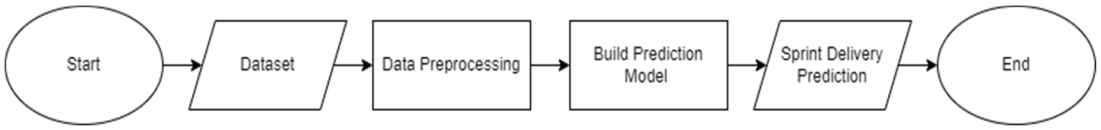

Fig 1. System Workflow

B. Dataset

The dataset used in this study is the dataset developed by [7] [8] [13] [11] in their respective research topics relating to software projects. The dataset consists of 3 groups of data, namely datasets regarding agile sprints, delayed issues, and story points estimation. In this study, the dataset that will be used is the agile sprints dataset. The sprint data used are data of iterations and issues from Apache, JBoss, and Spring software projects. The dataset consists of sprints and issues on three stage progressions, specifically 30%, 50%, and 80% of each project's progression.

Each project's sprint numbers are listed as follows, Apache project consists of 347 sprints with 5826 issues in its life cycle. JBoss project has 352 sprints with 4984 issues, and the Spring project's life cycle consists of 476 sprints with 17497 issues. Table 1 displays the used features from the iterations dataset. Table 2 displays the used features from the issues dataset.

TABLE I
USED FEATURES FROM THE ITERATIONS DATASET

Features	Description
planday	The number of days from the sprint's start date to planned completion date
vel_diff	The difference of a sprint's committed story points to the sprint's targeted story points
vel_starttime	The sum of story points of issues assigned at the beginning of a sprint
vel_todo	The sum of story points of a sprint's to-do issues
vel_added	The sum of story points of issues added during a sprint's duration
vel_inprogress	The sum of story points of in-progress issues of a sprint
vel_done	The sum of story points of resolved issues
vel_removed	The sum of story points of issues removed during a sprint's duration
no_issue_starttime	The number of issues assigned at the beginning of a sprint
no_issue_todo	The number of to-do issues in a sprint
no_issue_added	The number of issues added during a sprint's duration
no_issue_inprogress	The number of in-progress issues in a sprint
no_issuedone	The number of resolved issues in a sprint
no_issue_removed	The number of removed issues during a sprint's duration
no_teammember	The number of team members working on a sprint

TABLE II
USED FEATURES FROM THE ISSUES DATASET

Features	Description
type	A sprint's issue type
priority	The priority level of an issue
no_comment	The number of comments
no_priority_change	The number of times an issues' priority level is changed
no_des_change	The number of times an issues' description is changed

C. Data Preprocessing

Data preprocessing is done with the purpose of cleaning and preparing the data so that the data can be used to build the required prediction model. In this section, the dimensions of said data are reduced so that there is no data redundancy in the dataset used. Here are the steps of the data preprocessing:

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 8, No. 1, Maret 2023, Pp. 000-000

Step 1. Remove any unnecessary columns.

Step 2. Encode the categorical data type contained in *"type"* and *"priority"* columns into a numerical data type. The data in *"type"* column is encoded into binary values corresponding to a sprint's respective issue type. As for the *"priority"* column's encoding, it is represented in Table 3 shown below.

TABLE III
ENCODED RESULTS OF "PRIORITY" COLUMN

Pre-encoded	Post-encoded
Trivial	1
Minor	2
Major	3
Critical	4
Blocker	5

Step 3. Aggregate the features from both iterations and issues dataset.

Step 4. Determine the importance levels of features to the prediction model's targeted feature, namely the *"vel_diff"* column.

Step 5. Determine the correlation of each feature. Features with a low correlation index value will not be used in the building of the prediction model. Used features' correlation levels are shown in Figure 2 to Figure 4.

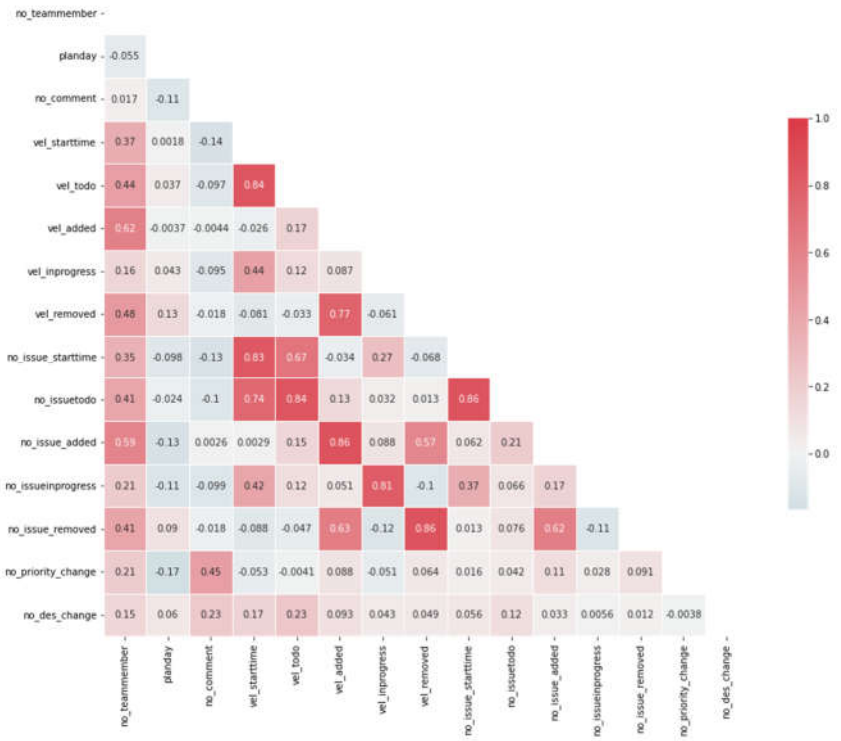

Fig 2. An example of correlation levels of used features in Apache software project

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 8, No. 1, Maret 2023, Pp. 000-000

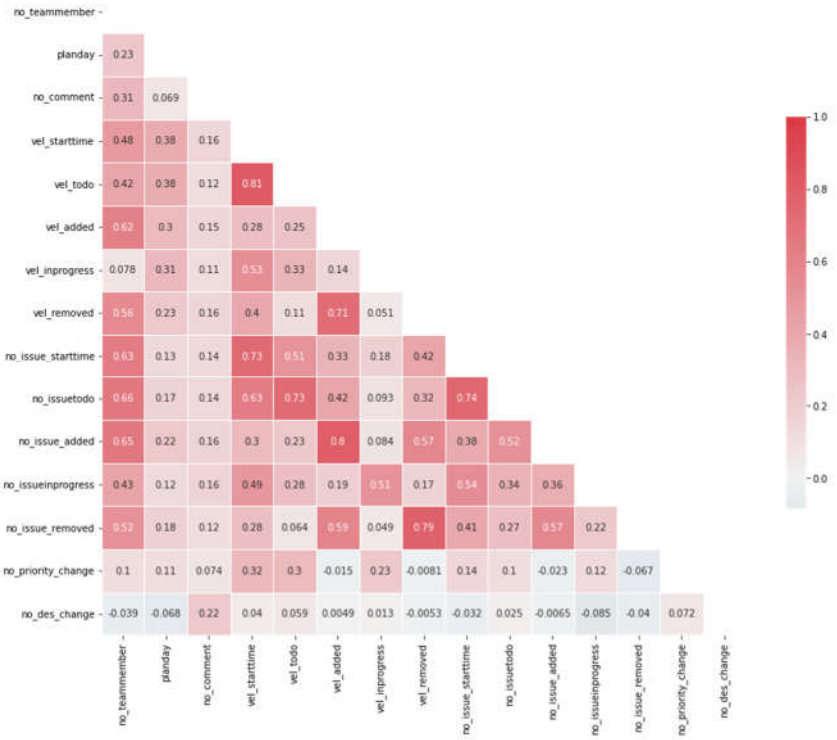

Fig 3. An example of correlation levels of used features in JBoss software project

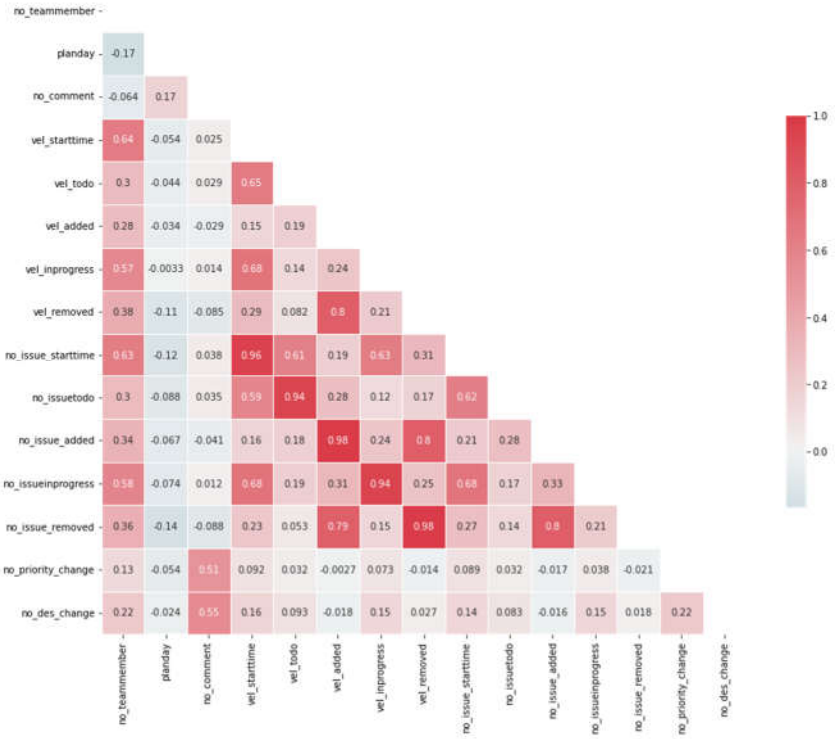

Fig 4. An example of correlation levels of used features in Spring software project

Step 6. Split the aggregated data into three parts, namely training data, validation data, and testing data

The steps mentioned above yields an aggregated dataset containing features from both the iterations and issues dataset. The data will then be used for training and testing the suggested prediction model.

D. *Building the Prediction Model*

The suggested model is built to predict the level of sprint delivery from a software project. Three prediction models are built using three different methods. The main method used for building the model is Random Forest [7] [14] [15] Regressor. The other two methods used are KNN (K-Nearest Neighbors) [16] [17] and Decision Tree [18] [19] Regressor. Three models are built with the purpose of testing the influence of data dimensions as well as testing the overall effectiveness of each algorithm to determine which model performs the best.

Aggregated features from the data are fed into a regressor in which said features are used as parameters of the prediction model. The earlier preprocessing is done so that the parameters used in the prediction model built with each of the three methods will have reduced chances of overfitting when the data is evaluated in the validation set. In turn, the prediction model built with the testing set will be ensured to have similar or better results compared to the evaluated results on the validation set. The prediction is done in three stages in which said stages are each software projects' stage of progression, the said stage progressions being each project's progress at 30%, 50%, and 80% of work.

There is also a categorization regarding each projects' level of sprint delivery. A sprint is considered successful if the value contained in *"vel_diff"* column is greater than or equal to zero. The levels of sprint deliveries are categorized as such:

Underachieved : value of *"vel_diff"* is less than zero
On Target : value of *"vel_diff"* is equal to zero
Overachieved : value of *"vel_diff"* is greater than zero

The expected output of the prediction model is each model's level of accuracy as well as the performance levels of sprints done by the three tested software projects. Results from each prediction model and each performance level are then compared to one another to determine the best performing algorithm and the best performing software project.

III. Results and Discussions

Here are the tables representing the test results displaying each model's accuracy level as well as each software project's sprint deliveries. Table 4 to Table 6 displays the accuracy level of each prediction model across the three tested software projects. The sprint deliveries of each software project are shown on Table 7 to Table 9.

Table IV
Accuracy Levels On 30% Of Project Progression

	Random Forest Regressor	K-Nearest Neighbors	Decision Tree Regressor
Apache	0.778	0.693	0.791
JBoss	0.780	0.688	0.716
Spring	0.845	0.621	0.686

Table V
Accuracy Levels On 50% Of Project Progression

	Random Forest Regressor	K-Nearest Neighbors	Decision Tree Regressor
Apache	0.797	0.718	0.797
JBoss	0.841	0.563	0.501
Spring	0.837	0.650	0.451

Table VI
Accuracy Levels On 80% Of Project Progression

	Random Forest Regressor	K-Nearest Neighbors	Decision Tree Regressor
Apache	0.779	0.794	0.715
JBoss	0.780	0.717	0.555
Spring	0.819	0.582	0.350

The results on Table 4 to Table 6 showed that the model built using Random Forest Regressor manages to give out consistent test results. With values ranging from 0.778 up to 0.841, that means the model yields a commendable performance with an accuracy level of about 77% up to 84% across each stage progression of all software projects. As for the KNN method, it showed fluctuating results across both JBoss and Spring projects with an accuracy level going as low as 0.56 and 0.58. However, the method performs quite well on Apache project, shown by its steady increase of accuracy levels spanning across the stage progressions.

The model built using Decision Tree gives out declining results through each stage progressions of all tested software projects. While the decline of accuracy levels in Apache project is moderate at best, the results in both JBoss and Spring projects show a more significant decline. The model yields an accuracy level ranging from as high as 0.716 to as low as 0.350. Performances from both KNN and Decision Tree Regressor may be considered average at best.

Results shown above can be attributed to several reasons. Namely the influence of data dimensions, especially the dimensions of the issues dataset may have been the main factor that had affected the performance of the prediction model built using KNN. Then, for the Decision Tree's results, those may be attributed to its inadequacy for applying regression as well as the complexity of data features that may have hindered its overall performance across all three stage progressions of both JBoss and Spring software projects.

TABLE VII
SPRINT DELIVERIES OF APACHE SOFTWARE PROJECT

Progress	*Underachieved*	*On Target*	*Overachieved*
30%	243	46	58
50%	251	45	51
80%	263	42	42

TABLE VIII
SPRINT DELIVERIES OF JBOSS SOFTWARE PROJECT

Progress	*Underachieved*	*On Target*	*Overachieved*
30%	210	96	46
50%	217	94	41
80%	225	101	26

TABLE IX
SPRINT DELIVERIES OF SPRING SOFTWARE PROJECT

Progress	*Underachieved*	*On Target*	*Overachieved*
30%	118	34	324
50%	131	48	297
80%	155	65	256

Results on Table 7 to Table 9 have shown that across all three tested software projects, the best software project delivery is achieved by Spring project. The numbers on both Apache and JBoss projects showed that both of these projects have mostly underachieving sprint deliveries. Across 347 and 352 sprints, each of these two projects have over 200 underachieving sprints in every stage progression. However, the Spring project performed quite admirably. Across 476 sprints and three stages of progression, at least over 300 sprints managed to either fulfill or even surpass the minimum working target.

It has to be noted that the results obtained on this research were gained through a simpler approach based on a previous study [7]. Although the dataset used is indeed from a similar [7] study, shown by the data of software projects we used in this research, the previously said constraints regarding the lack of issue dependencies are still influential to the results. Therefore, a different method must be implemented to build our suggested prediction model. Thus, it is not feasible to have a direct comparison of test results due to the difference of conditions occured between this study and the previous one.

IV. CONCLUSION

In this study, we were able to implement our suggested prediction models in an environment of iterative-based software development. The models were created using three methods, namely Random Forest Regressor as our main method, with KNN and Decision Tree Regressor being the comparison methods. The results of the test have shown that the Random Forest Regressor is the best performing algorithm. In addition, the performance of each sprint in the three software projects tested, namely Apache, JBoss, and Spring were also obtained. Through the results of performance levels from each sprint of the three software projects, it was found that most of the sprints in the Spring project were able to fulfill and/or surpass the minimum target. This shows that the Spring project is the software project with the best sprint delivery rate.

It should be noted that in the test results, there were several occasions where results from both KNN and Decision Tree Regressor that were able to exceed the performance levels of the model built with Random Forest Regressor. However, the steady performance of Random Forest Regressor at every stage progression of all tested software projects indicates that this algorithm is better for use in the building of prediction models. Therefore, although this model does not yield the best results, the results obtained by our prediction system still allowed it to be applied in an environment of iterative-based software development. For future work suggestions, this research can be further developed to achieve a more optimum result. For example, by testing the performance of the prediction model from this research on other software project data, or by optimizing the data preprocessing done in this study, or even try to build another prediction model using a combination of ensemble learning methods such as Random Forest and Deep Neural Networks.

REFERENCES

[1] L. Gonçalves, "Scrum," *Control. Manag. Rev.*, vol. 62, no. 4, pp. 40–42, May 2018, doi: 10.1007/s12176-018-0020-3.
[2] F. Hayat, A. U. Rehman, K. S. Arif, K. Wahab, and M. Abbas, "The Influence of Agile Methodology (Scrum) on Software Project Management," *Proc. - 20th IEEE/ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput. SNPD 2019*, pp. 145–149, Jul. 2019, doi: 10.1109/SNPD.2019.8935813.
[3] J. Wright, "Scrum: the complete guide to the agile project management framework that helps the software development lean team to efficiently structure and simplify the work & solve problems in half the time." p. 95, 2020.
[4] M. Hron and N. Obwegeser, "Why and how is Scrum being adapted in practice: A systematic review," *J. Syst. Softw.*, vol. 183, p. 111110, Jan. 2022, doi: 10.1016/J.JSS.2021.111110.
[5] M. Hron and N. Obwegeser, "Scrum in practice: An overview of Scrum adaptations," *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, vol. 2018-Janua, pp. 5445–5454, 2018, doi: 10.24251/hicss.2018.679.
[6] M. Marinho, J. Noll, and S. Beecham, "Uncertainty management for global software development teams," *Proc. - 2018 Int. Conf. Qual. Inf. Commun. Technol. QUATIC 2018*, pp. 238–246, Dec. 2018, doi: 10.1109/QUATIC.2018.00042.
[7] M. Choetkiertikul, H. K. Dam, T. Tran, A. Ghose, and J. Grundy, "Predicting Delivery Capability in Iterative Software Development," *IEEE Trans. Softw. Eng.*, vol. 44, no. 6, pp. 551–573, 2018, doi: 10.1109/TSE.2017.2693989.
[8] M. Choetkiertikul, H. K. Dam, T. Tran, and A. Ghose, "Predicting delays in software projects using networked classification," *Proc. - 2015 30th IEEE/ACM Int. Conf. Autom. Softw. Eng. ASE 2015*, pp. 353–364, 2016, doi: 10.1109/ASE.2015.55.
[9] C. Verwijs and D. Russo, "A Theory of Scrum Team Effectiveness." 2021. [Online]. Available: http://arxiv.org/abs/2105.12439
[10] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, "An effective approach for software project effort and duration estimation with machine learning algorithms," *J. Syst. Softw.*, vol. 137, pp. 184–196, Mar. 2018, doi: 10.1016/J.JSS.2017.11.066.
[11] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, "A Deep Learning Model for Estimating Story Points," *IEEE Trans. Softw. Eng.*, vol. 45, no. 7, pp. 637–656, 2019, doi: 10.1109/TSE.2018.2792473.
[12] P. Ardimento and C. Mele, "Using BERT to Predict Bug-Fixing Time," *IEEE Conf. Evol. Adapt. Intell. Syst.*, vol. 2020-May, May 2020, doi: 10.1109/EAIS48028.2020.9122781.
[13] M. Choetkiertikul, H. K. Dam, T. Tran, and A. Ghose, "Characterization and prediction of issue-related risks in software projects," *IEEE Int. Work. Conf. Min. Softw. Repos.*, vol. 2015-Augus, pp. 280–291, 2015, doi: 10.1109/MSR.2015.33.
[14] D. Denisko and M. M. Hoffman, "Classification and interaction in random forests," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 115, no. 8, pp. 1690–1692, Feb. 2018, doi: 10.1073/PNAS.1800256115.
[15] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 8, no. 4, p. e1249, Jul. 2018, doi: 10.1002/WIDM.1249.
[16] S. Zhang, M. Zong, X. Zhu, D. Cheng, and X. Li, "Learning k for kNN classifi-cation," *ACM Trans. Intell. Syst. Technol*, vol. 8, no. 43, 2017, doi: 10.1145/2990508.
[17] H. Al-Shehri *et al.*, "Student performance prediction using Support Vector Machine and K-Nearest Neighbor," *Can. Conf. Electr. Comput. Eng.*, Jun. 2017, doi: 10.1109/CCECE.2017.7946847.
[18] R. Hasan, S. Palaniappan, A. R. A. Raziff, S. Mahmood, and K. U. Sarker, "Student Academic Performance Prediction by using Decision Tree Algorithm," *2018 4th Int. Conf. Comput. Inf. Sci. Revolutionising Digit. Landsc. Sustain. Smart Soc. ICCOINS 2018 - Proc.*, Oct. 2018, doi: 10.1109/ICCOINS.2018.8510600.
[19] A. K. Hamoud, A. S. Hashim, and W. A. Awadh, "Predicting Student Performance in Higher Education Institutions Using Decision Tree Analysis," *Int. J. Interact. Multimed. Artif. Intell.*, vol. 5, no. 2, p. 26, 2018, doi: 10.9781/ijimai.2018.02.004.