International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 13, No.1 (Jan-2023)

http://dx.doi.org/10.12785/ijcds/130107

Assuring Software Reuse Success Using Ensemble Machine Learning Algorithms

Mustafa Hammad[1] **and Mariam Amin**[2]

[1]*Department of Software Engineering, Mutah University, Al-karak, Jordan*
[2]*Department of Computer Science, University of Bahrain, Sakheer, Kingdom of Bahrain*

Received 6 Dec. 2021, Revised 29 Sep. 2022, Accepted 13 Dec. 2022, Published 31 Jan. 2023

Abstract: Software reuse is a critical practice that helps software developers to increase their productivity. Also, it reduces the developing effort and project budget. Howsoever, some factors may lead to a software reuse failure. Software development companies have to consider these factors to prevent project failure due to software reuse. These factors are not only related to the technical aspects of the project, but they cover the companies' managerial decisions too. This work incorporates ensemble machine learning to predict successful software reuse experience. To the best of our knowledge, this the first work that used ensemble learning to predict successful software reuse. Also, a feature selection technique was used to extract the essential attributes from the dataset. The empirical study showed remarkable results that scored an accuracy of 100%.

Keywords: Software Reuse, Ensemble Leaning, Stacking, Bagging, Voting, Wrapper Subset Evaluation

1. Introduction

In 1968, the NATO science committee sponsored a conference, which discussed all concerns related to software development and software engineering discipline. Some of these concerns tackled the massive gap between the users' expectations and the developed software. Also, the developed software lack meeting the requirements specifications and suffered from budget overrun. Furthermore, it took more time than scheduled to complete the software projects [1]. It has been 53 years since the NATO conference, but still, these concerns persist. Moreover, they become more complicated nowadays due to the growth of software project complexity. Typically, developing software takes a lot of time and effort. Recruiting more workforce may significantly increase the project cost. Large-scale software projects or critical software such as space navigation systems require even more time and budget since any software defect could be catastrophic.

According to Lehman's laws of software evolution [2], software changes are inevitable. The software's complexity will continue to grow until a software replacement is more cost-effective than maintaining the existing one. Furthermore, it is trivial to implement complex software from scratch. Therefore, reusing well-tested software assets would increase software development productivity and reduce the project cost. Furthermore, the surging growth of Open Source Software (OSS) has offered a wide selection of reusable software components [3]. Although software reuse increases the software dependability and reduces the error margins, selecting reusable software components can be critical. In 1996, during flight 501 of Ariane 5, the rocket veered the flight path at an altitude of 3700 miles and exploded. The investigation concluded that the cause of failure was a bug in a reused source code in the flight control system [4]. Therefore, assessing software assets is for reuse is very crucial and must be systematic.

Software reuse is a strategy that is adopted widely among software developers. This strategy improves software quality and reduces project time, which relieves some pressure of the delivery deadlines. Also, it dramatically increases the software developers' productivity. There is few successful commercial OSS provided a good source for reusable software assets. In addition to that, some organizations have developed their own software reuse repository. However, assessing and selecting the appropriate component could be a complicated, lengthy, and expensive process that contradicts the purpose of reusing software. One of the solutions that could improve the software artifacts assessment is by utilizing machine learning.

Many researches focused on assessing the reusability of software components by evaluating the source code. However, other factors could play a role in the success of the software reuse, such as the project settings, software developers' experience, motivation, and the top management commitment. There are few pieces of researches that

E-mail address:[1]*hammad@mutah.edu.jo,* [2]*19980608@stu.uob.edu.bh,*

utilize a single classifier to predict successful software reuse experience. Therefore, there is a demand for prediction models that use multiple classifiers to predict software reuse experience. This work proposed two ensemble prediction models using three types of ensemble machine learning algorithms. These algorithms are stacking, bagging, and voting.

The remaining of the study is structured as follows. Section 2 briefly reviews some works of literature about software reuse, machine learning, and ensemble learning. Section 3 defines the proposed ensemble learning models. Section 4, 5, 6 presents the proposed models' components in detail. Section 7 depicts the used evaluation criteria. The results of the experiments are presented in Section 8. Section 9 presents the best and worst ensemble predation models. The threats to validity are discussed in Section 10. Finally, Section 11 concludes this work.

2. Related Work

Software reuse reduces software development costs. Al-Badareen et al. [5] studied the software reuse decision-making based on the cost of specific scenarios for seven software components. The study showed that developing a new reusable software component from scratch might cost more than the development of a standard software component. However, reusing an acquired reusable software asset might reduce the project cost by about 23%. The work in [6] compares the cost of two software reuse strategies. These strategies are platform-oriented and clown and own by conducting 26 interviews and reviewing 57 pieces of literature. The authors concluded to a similar result as [5], which is developing reusable components is more expensive than developing a single-use software component. Patrick [7] hypothesized that software reuse cost is directly affected by the software understandability and indirectly by software complexity. The author proposed a machine learning classifier (LANLAN). The classifier's training data was extracted from a Question and Answer (Q&S) forum. The classifier had to distinguish whether the software reuse cost was influenced by software understandability or software complexity. The proposed classifier performed very well with a ROC curve about 0.9.

Software reuse is considered a goal in modern development methodologies. For instance, in Software Product Lines (SPLs), Abbas et al. [8] proposed an approach that recommended reusing software assets based on customer requirements. The proposed approach links the new requirements with existed assets in the requirements repository using natural language processing and machine learning. The evaluation showed that the proposed approach could match new requirements with existing assets with about 74% accuracy. In [9], The authors proposed a reuse system. This system was integrated into the DevOps process to manage and maintain the reuse repository. Also, the proposed system would ease the process of reviewing, identifying, and retrieving the reusable assets. A controlled experiment was conducted to evaluate the effectiveness of the proposed system. The results showed that the proposed system performed very well but need further evaluation in more complex situations. Lahouij et al. [10] proposed an approach that helps preserve the reliability of the composite cloud services. The proposed approach focuses on finding and reusing the appropriate substitution if any composite service component is unavailable. The authors' evaluation concluded that the proposed approach had reconfigured the composite cloud services successfully.

Machine learning was leveraged in various ways in software reuse. Negi and Taweri [11] assessed the reusability of open source Object Oriented (OO) source code using machine learning. The authors generated a dataset using Chidamber and Kemerer (CK) metrics from selected source codes. The prediction model scored about 98.4% accuracy. Similar work in [12] predicted software component reusability by proposing an algorithm that integrates gradient boosting and random forest. The results conclude that the proposed algorithm outperformed eight supervised machine learning algorithms by 23%. In [13], data mining was used to evaluate OO software components for reuse. The experiments showed that data mining could be useful in assessing software reusability. A novel approach (Deepclone) was proposed by Hammad et al. [14]. This approach uses deep learning to facilitate the reuse of code clones from a source code repository. Deepclone predicts the next required tokens based on what has been written. The evaluation of the proposed approach showed promising results.

Ensemble machine learning improves the prediction model outcomes. Zhang et al. [15] used the Internet of Things (IoT) and ensemble learning to improve the accuracy of electricity transformers fault detection. The ensemble model evaluation showed a prediction improvement reached about 99%. The work in [16] assesses the performance of five ensemble machine learning algorithms in classifying financial data. The experiment results showed that ensemble learning is effective and reliable in predicting corporate bankruptcy. In [17], an ensemble learning model was proposed. This model, which utilizes voting, detects brain cancer. The results showed that the model was able to diagnose brain tumors with an accuracy of 99%. Gao et al. [18] proposed an adaptive voting algorithm to improve the ensemble prediction of four machine learning classifiers. The authors compared the performance of the proposed algorithm with the prediction performance in previous works. The results concluded that the proposed voting algorithm improved the prediction outcomes. The work in [19] studied the effect of proposed bagging-based ensemble learning on classifying imbalanced datasets. The evaluation showed that there was a significant improvement in the performance of the proposed algorithms.

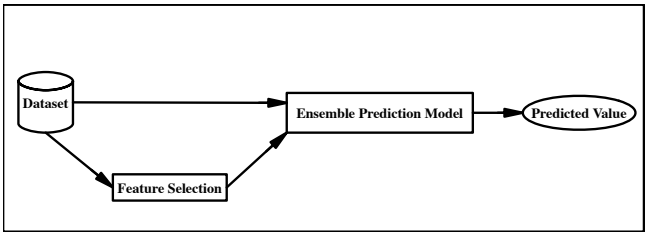

Figure 1. Proposed Ensemble Prediction Models

TABLE I. Dataset Statistical Details

Statistical Component	Statistical Detail
Number of the instances	23
Number of the attributes	24
Number of successful software reuse instances	15
Number of failure reuse instances	8
Percentage of successful software reuse	65.22%
Percentage of failure software reuse	34.78%

3. Proposed Ensemble Prediction Models

This work proposed two ensemble prediction models. Figure 1 depicts the proposed models. According to Figure 1, the first proposed ensemble prediction model utilizes the full dataset to predict software reuse experience. In the second model, features selection technique was introduced to the ensemble prediction model, as shown in Figure 1. The purpose of introducing feature selection to the proposed ensemble prediction model is to extract the most relevant features from the dataset. Since, the redundant and irrelevant features increase the data dimensionality without adding new information to the dataset. This could negatively affect the performance of the prediction model.

4. Used Dataset

The used dataset was constructed by interviewing 19 software development companies that introduced software reuse in their industrial projects. The interviews were guided by a questionnaire that was given to the interviewees. Furthermore, the interview questions focus on collecting information about the companies such as business domain and the number of staff. Also, information related to the reuse at the organization level, such as management commitment to software reuse and the employees' training and awareness about software reuse. In addition to that, reuse processes and assets [20].

The software reuse dataset consists of 23 instances. Each instance depicts a software project. Table I presents statistical information about the used dataset. The dataset is classified into two classes, successful software reuse, and failure software reuse. The number of instances with successful software reuses is 18 instances, which is about 65% of the dataset instances. Meanwhile, the total number of failure instances is eight that is around 35% of the dataset instances.

Furthermore, the dataset attributes were divided into groups based on the company's control over these attributes. These groups are state variables and control variables. Any attribute that requires weeks to change was considered as a state variable. Table II presents the state variable of the used dataset, the value of the attributes, and the attributes' description.

The type of software production is one of the state variables. The dataset authors defined a product-family company that develops software that may evolve with time. Meanwhile, the product company develops software that shares anything in common. According to [20], fifteen projects from the 20 projects were developed by product-family companies have succeeded in reusing software assets. Meanwhile, none of the product companies have succeeded in reusing software. Furthermore, Table II shows that the dataset covered 13 application domains. Telecommunication (TLC) and manufacturing application projects are more likely to succeed in reusing software [20].

The second attribute group is the control variables, which are influenced the company's process of decisions. These variables are divided into high-level control and low-level control. Table III presents the control variable, possible attribute values, and the attributes' description. According to [20], most of the projects that introduced the high control variables had a successful software reuse experience. Also, projects that have a loose reuse approach were successful. In addition to that, companies with higher-grained assets are more likely to succeed in their projects. Projects 51-100 or 100+ assets have a higher number of successful software reuse experiences.

5. Wrapper Feature Selection

There are many benefits of using feature selection techniques. For instance, it reduces the training time, helps visualize the data, and optimizes the storage requirements. In this paper, the primary purpose of using feature selection techniques is to improve the prediction model's performance by removing the irrelevant attributes [21]. The selected feature selection algorithm is Wrapper Subset Evaluation. The Wrapper Subset Evaluation uses a machine learning classifier as an inducer. The inducer's role is to define a subset of dataset attributes where the accuracy of the inducer algorithms is the maximum, and the selected features are relevant [22]. The performance of the chosen feature selection algorithm was tested in previous work [23]. Figure 2 depicts the variance of error rates between the between the tested feature selection algorithms. In most case the Wrapper Subset Evaluation scored the minimal error rate. The low error rates indicate that the Wrapper Subset Evaluation is the most accurate feature selection algorithms among the tested algorithms. Also, this feature selection algorithm improves the prediction of successful software experience. Therefore, we choose the Wrapper Subset Algorithm as a feature selection technique in the proposed model. The wrapper subset evaluation algorithm

TABLE II. Description of Dataset State Variables

Attribute	Attribute Values	Attribute Description
Software staff	S: (Small)	Software development team is small which about 1- 50 team members
	M: (Medium)	Software development team is medium which can between 51- 200 team members
	L: (Large)	Software development team is large and ranges from 201 and more
Overall staff	S: (Small)	Company's employee is small which ranges from 1 to 50 employees
	M: (Medium)	Hired employees is considered medium which can be between 51 and 200
	L: (Large)	Overall staff is large and ranges from 201and 500
	X: (Extra-large)	Company's staff is extra-large which is more the 501 employees
Type of software production	Isolated	There is no commonality between the company's project
	Product Family	There is commonality between the company's project
Software and product	Product	The developed software is embedded inside a product
	Alone	The developed software is a standalone product
	Process	The developed software embedded inside a process
SP maturity	Low	The Capability Maturity Model (CMM) level of the software process is 1 or it is not ISO9001 certified
	Middle	The CMM level of the software process is 2 it is ISO9001 certified
	High	The CMM level of the software process is 3 or more
Application domain	TLC	Telecommunications
	SE-Tools	Software tools
	Bank	Bank applications
	Engine Controller	Engine controller application
	FMS	Fire management applications
	TS	Train simulation applications
	ATC	Air traffic control application
	Manufacturing7	Manufacturing applications
	Measurement	Measurement environment control and management applications
	Finance	Finance applications
	TTC	Train Traffic Applications
	Space	Aerospace applications
	Book-keeping	Book keeping applications
Type of software	Technical	The develop software neither embedded nor real-time with limit or no database
	Business	The develop software neither embedded nor real-time but database intensive
	Embedded-RT	The developed software is embedded and real-time
	Non-Embedded-RT	The develop software is not embedded but real-time
Size of baseline	S: (Small)	The of reused assets in a projects is less than 10KLOC and take 10 person-months of effort
	M: (Medium)	The of reused assets in a projects is between 10KLOC and 100KLOC and take effort of 10 to 100 person-months
	L: (Large)	The of reused assets in a projects ranges between 100 to 500KLOC and take more than 100 person-months of effort
Development approach	OO	Object oriented development approach
	Proc	Procedural development approach
Staff experience	Low	The experience of project members is one year or less
	Middle	The experience of project members between 2 and 4 years
	High	The experience of project members is more than five years

TABLE III. Description of Dataset Control Variables

Control Level	Attribute	Attribute Values	Attribute Description
High	Top management commitment	yes	Whether the top management are committed to adopt software reuse as a prominent practice
		no	The reuse initiative came from the middle management or the individuals
	Key reuse roles introduced	yes	Indicator whether any reuse role was given to a project team member
		no	
	Reuse processes introduced	yes	At least one software reuse process was introduced
		no	
	Non-reuse processes modified	yes	The development process which are not related to reuse were modified
		no	
	Repository	yes	Whether the software assets is established are assets are stored in repository
		no	
	Human factors	yes	Whether the staff were aware and trained about software reuse
		no	
Low	Reuse approach	Tight	Reusable product the tightly coupled cannot be isolated
		Loose	Reusable product the loosely coupled can be isolated
	Work products	D+C	Type of the reusable assets Design D, Requirement R, Code C
		C	
		R+D+C	
	Domain analysis	yes	Whether is the baseline of the product was established by domain analysis or not
		no	
	Origin	ex-novo	Software assets were developed from scratch
		as-is	Software assets were reused as it is
		reeng	Software assets were reengineered from existing product
	Independent team	yes	Indicator that there is a dedicated reuse team with the project to develop the reusable assets
		no	
	When assets developed	before	The reusable assets were developed before the project's need
		Just in time	The reusable assets were developed during the project
	Qualification	yes	There is a quality process that defined the reusable assets
		no	
	Configuration management	yes	The reusable assets are maintained by a configuration managements
		no	
	Reward s policy	yes	The existence of a reward policy that promotes software reuse
		no	
	# assets	1 - 20	
		21 - 50	Total number of the reusable assets which are stored in the repository
		51 - 100	
		100+	

extracted a features subset that contains three attributes from the used dataset. These attributes are the type of software production, application domain, and human factors.

6. Ensemble Algorithms

Ensemble learning is a machine learning methodology that combines multiple learning classifiers. The primary purpose of combining machine learning classifiers is to eliminate the errors and risks when using a single classifier [24]. Moreover, ensemble learning improves the performance of the prediction by counterbalance the weakness of a single classifier. Furthermore, the volume of the dataset may cause problems while training the classifier. For instance, it is impractical to introduce a single classifier on a large dataset. Therefore, the optimal solution is to divide

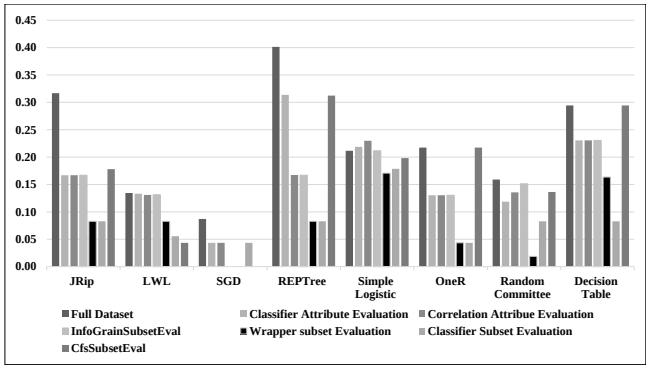

Figure 2. Summary of Testing Feature Selection Algorithms on The Selected Dataset

the dataset into subsets where each subset is used to train multiple classifiers. Also, ensemble learning may resolve problems that are caused when using a small dataset [25]. In this work, three ensemble algorithms were used to improve the prediction models' performance. These algorithms are Stacking, Bagging, and Voting. The selected ensemble algorithms are described as following:

- Stacking: is a type of ensemble learning that combines multiple classification algorithms. Basically, the learning process of stacking learning consists of two stages. Basically, the learning process of stacking learning consists of two stages. The first stage combines different machine learning classifiers P={P_i (s,f),i=1,...,I}, which are called base classifiers. Each base classifier utilizes the training set f. The second stage is the meta-learning stage, where a single machine learning classifier M uses the outputs on the base classifiers as an input to generate the ensemble learning final outcome [24], as shown in Figure 3a. Therefore:

$$j_{stack}(s) = M(s, P) \tag{1}$$

  where j_{stack} (s) is the final stacking prediction of the input s. In this paper, six supervised machine learning classifiers were chosen for the first stage of the stacking. These classifiers are Hoeffding Tree, Random Forest, Naïve Bayes, Stochastic Gradient Descent (SGD), J48, and Multilayer Perceptron (MLP). Also, three machine learning algorithms were used as meta-learners in the second stage of the stacking. These algorithms are SGD, Sequential minimal optimization (SMO), MLP.

- Bagging: this type of ensemble algorithms uses a single machine learning classifier. However, it generates different replicates from the classifier. These replicates use different subsets of instances from the dataset [26]. Bagging algorithms improve the prediction outcome by reducing the classifier variance and the mean square error, which will create a more stable classifier [27]. The predictor P(s,f) that utilizes the training set of f{(j_1,...,j_m,s_n), m= 1,...,M, n= 1,...,N} and input j to predict the value j can predict a better results if replicates of the predictor were trained on different subsets of the training set f_k {$f_k \subset f$, k=1,...,K }, Therefore:

$$j_{bag}(s) = \underset{k=1}{\overset{K}{\operatorname{argmax}}}(P(s, f_k)) \tag{2}$$

  where j_{bag}(s) is the maximum aggregated prediction, K is the number of training subsets. The bagging algorithm's final result is generated by plurality voting [28], shown in Figure 3b. In the bagging experiments, three machine learning algorithms were used as base classifiers. These algorithms are SGD, SMO, and MLP.

- Voting: the vote combiner utilizes different robust machine learning classifiers on the same dataset f. The predictions' results are combined using a voting rule to obtain the final outcome, as shown in Figure 3c. Voting can provide a better output than using a single machine learning classifier. Therefore, it improves the prediction model reliability [29]. In this work, six machine learning classifiers were combined using voting ensemble learning to predict successful software reuse experience. These classifiers are Hoeffding Tree, Random Forest, Naive Bayes, SGD, J48, and MLP. Moreover, the voting learning experiment was repeated five times for each combination rules. The selected combination rules are as following [30]:
  - Average of probabilities (AP): this voting rule calculates the mean value of each predicted class. The final predict would the class that has the maximum mean value such that

$$j_{AP}(s) = \max_{k=1}^{K} \sum_{m=1}^{M} P_m(s|f)/m \tag{3}$$

  - Product of probabilities (PP): the Final class is class that has the maximum product value of among the products values of predicted classes. Therefore final classification is computed by the following formula:

$$j_{PP}(s) = \max_{k=1}^{K} \prod_{m=1}^{M} P_m(s|f) \tag{4}$$

  - Majority voting (MV): when assigning the final class using the MV the class the has the highest votes is considered the final results. Therefore,

$$j_{MV}(s) = \max_{k=1}^{K} P_k(s|f) \tag{5}$$

  - Minimum probability (MinP): in this combination rule, the minimum posterior probabilities of each class computed. Then, the final classification is the maximum value among these minimum values

$$j_{MinP}(s) = \max_{k=1}^{K} \min_{m=1}^{M} P_m(s|f) \tag{6}$$

  - Maximum probability (MaxP): this combination rule assign the final classification to the maximum value among the maximum posterior probabilities for each individual. Therefore the final prediction is calculated by the following equation:

$$j_{MaxP}(s) = \max_{k=1}^{K} \max_{m=1}^{M} P_m(s|f) \tag{7}$$

  Where k is the number of voting classifiers, m number of predictions of each classifier, P is the posterior probabilities of the classifier, and s is the true class of the features f.

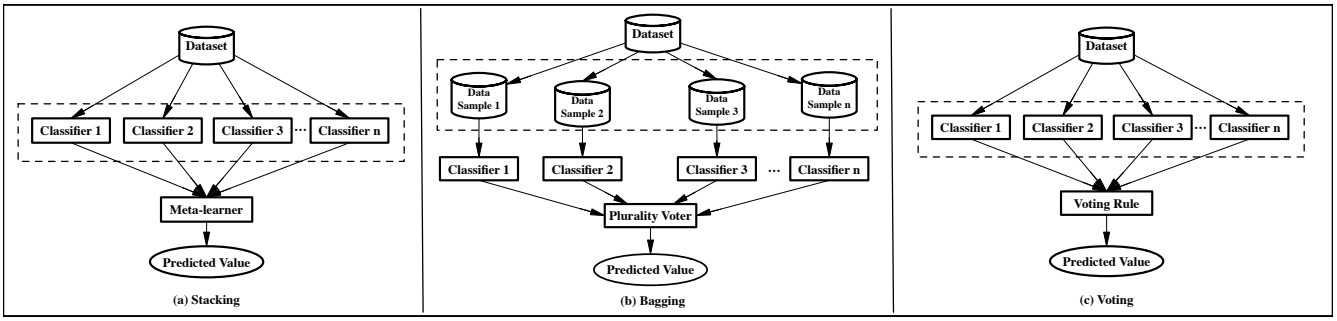

Figure 3. Ensemble Algorithms

7. Used Evaluation Criteria

There are various statistical metrics to assess the performance of machine learning models. Seven evaluation metrics were selected to evaluate the chosen ensemble learning algorithms. These metrics are the following:

- True Positive Rate (TPR): TPR also referred to as recall or sensitivity, is one of the evaluation criteria derived from the confusion matrix. TPR is the ratio of the number of correctly classified instances and the instance of actual positive instances [31]. TPR can be calculated using the following formula

$$TPR = TP/(TP + FN) \tag{8}$$

  where TP is the number of true positive instances, FN the number of false negative instances.

- False Positive Rate (FPR): FPR is the rate at which the classifier has incorrectly classified the instances as false positives. The FPR value is the fraction of the number of false positive instances by the number of actual negative instances [31]. The FPR is evaluated as the following

$$FPR = FP/(FP + TN) \tag{9}$$

  where FP is the number of false positive instances, TN the number of true negative instances.

- Precision: This criterion presents the ratio between the true positive instances and positive classified instances [31]. The following formula can derive the precision value

$$Precision = TP/(TP + FP) \tag{10}$$

  where TP is the number of true positive instances, and FP is the total number of false positives instances.

- Mean Absolute Error (MAE): MAE is an evaluation metric that measures the error of predicting the overall instances in the dataset [31]. This error rate can be derived from the following formula

$$MAE = \sum_{i=1}^{n} |y_i - x_i|/n \tag{11}$$

  where y_i is the real target value of an instance, x_i is the predicted value of an instance, n is the total number of dataset instances.

- Receiver Operating Characteristic (ROC) area: The ROC area value shows the tradeoff between the sensitivity and FPR. It represents the threshold value, which the classifier distinguishes, the positive from negative classification. ROC area is useful to identify how to minimize the error rates. Also, it is helpful to compare the performance between multiple classifiers [31].

- Time Taken to Build the Model: This evaluation criterion presents the execution time required to build the model and classify the dataset instances [32]. The time taken to build the model is useful when assessing the model's effectiveness [33].

- F-measure: The f-measure helps in assessing the accuracy of the prediction model. F-measure creates a harmonic mean that combines the values of the recall and precision. This combined value is used to judge whether the prediction model has a good performance or not [31]. The following formula calculates the f-measure value

$$f - measure = \frac{2(Precision * Recall)}{(Precision + Recall)} \tag{12}$$

8. Experimental Results

The experiments were conducted using WEKA 3-9-3 with the default settings. The selected training technique was ten folds cross-validation. This technique is widely used to test the machine learning classifier due to its reliability. Training and testing the machine learning classifier is repeated in iterations. Also, the dataset is divided equally into ten subsets called folds. In each iteration, the classifier used nine folds for training and one fold for testing. This process ended when the classifier tested all the data [34]. The following subsections present the experiments' results in detail.

A. Stacking

This subsection presents the results of using stacking algorithm to predict successful software reuse experience.

Also, the subsection shows the effect of incorporating feature selection with stacking algorithm.

1) Prediction Using Full Dataset

This subsection analyzes the results obtained from applying stacking algorithms on the full dataset. Table IV shows the performance results of the stacking prediction models. Overall, all prediction models took about eight seconds to build the model. The stacking experiments' true positive rates vary between 0.826 and 0.913, while the false positive rates were between 0.105 and 0.209. The MAE rates were relatively low. The error rate values vary between 0.0969 and 0.1739. Also, the ROC area results indicate the prediction models' good performance. In general, all prediction models have a high capability of distinguishing between success and the failure of the software reuse experience. However, applying stacking algorithms with MLP as a meta-classifier has outperformed the other models since its ROC area scored 0.973, which is the closest to 1.

2) Prediction After Feature Selection

This subsection analyzes the results of using stacking algorithm after applying the feature selection technique. Table V stated the performance criteria results, which shows that evaluation criteria scored the best results. All prediction models have successfully classified all the dataset instances shown since TPR and FPR are 1 and 0, respectively. Furthermore, the MAE of the Stacking with MLP was 0.0154, which is extremely low to negligible. Meanwhile, the MAE of stacking with SMO and SGD was 0. Also, the precision results of the prediction models scored were the best value that is 1. In addition to that, all models took about 0.5 seconds to be built. The ROC area results indicate that the stacking prediction models have an excellent performance. The models scored the best results, which is 1. A Roc area equals one indicates that the models have a good measure of separating the successful software reuse experience from failed experience.

3) Comparison Performance Results Before and After Feature Selection

This subsection presents the effect of applying feature selection techniques on stacking algorithm. According to Table IV and Table V , there is a significant improvement in instances classification. The prediction models failed to classify some of the dataset instances before applying the feature selection technique; meanwhile, the prediction models have all the instances successfully. Figure 4 shows the improvement of MAE rates after applying feature selection. The error rates of stacking using the SGD and SMO as meta-classifiers have sharply plunged to zero. Meanwhile, the error rate stacking with MLP has significantly declined from about 0.1 to 0.01.

B. Bagging

The subsection presents experimental results of applying bagging as an ensemble prediction model on the full dataset. Also, it discusses the effect of feature selection on bagging algorithms.

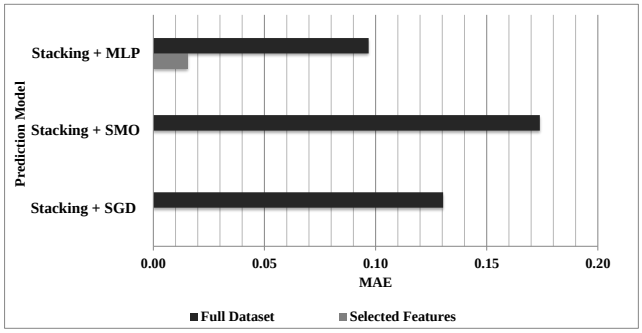

Figure 4. Progressions of The Error Rates Before and After Feature Selection

1) Prediction Using Full Dataset

Table VI presents the evaluation criteria of applying bagging ensemble learning of the full dataset. The TPR and FPR of bagging with SMO have the best results that are 1 and 0, respectively. This points out that bagging with SMO had classified all the instances correctly. Meanwhile, bagging with SGD and MLP has failed in classifying all the instances correctly. The effectiveness of bagging with SGD and SMO is more than bagging with MLP since the time taken to build the model was 8.86 seconds. Moreover, the error rates were minimal. The bagging with MLP scored the highest MAE rate that is 0.0618, which indicates that this prediction model is less efficient than bagging with SMO and SGD. However, the ROC area of bagging MLP has the best value that is 1. The ROC area measures the performance of the prediction model. The closer the ROC area value is to 1, the better. Therefore, bagging with MLP and SMO outperformed bagging with SGD. Furthermore, the precision of bagging with SMO scored the best value; meanwhile, the precision of bagging with MLP and bagging with SGD was slightly lower by 0.087 and 0.0041, respectively.

2) Prediction After Feature Selection

To improve the bagging algorithms' performance feature selection technique was applied. Table VII presents the results of the bagging prediction models after selecting the relevant attributes. According to Table VII, all bagging models have correctly classified the dataset instances. The true positive rate, false positive rate, and Roc area scored 1,0, and 1 respectively. The results are considered the best outcome that any classifier can achieve. The ROC area, which evaluates the prediction model performance, showed that all bagging models have good performance since the ROC area has reached 1. The MAE rates range between 0.0087 and 0.0711, which are very superficial. Also, bagging with SGD and SMO were more effective than bagging with MLP since the time taken to build the models is better by 0.44 and 0.32 seconds. Also, all prediction models have the best precision value.

TABLE IV. Results of Applying Stacking Algorithm on The Full Dataset

Prediction Model (Full Dataset)	TP Rate	FP Rate	Precision	ROC Area	MAE	Time taken to build model (seconds)
Stacking + SGD	0.870	0.186	0.869	0.842	0.1304	8.75
Stacking + SMO	0.826	0.209	0.826	0.808	0.1739	8.76
Stacking + MLP	0.913	0.105	0.913	0.973	0.0969	8.88

TABLE V. Results of Applying Stacking Algorithm on Selected Features

Prediction Model (Selected Features)	TP Rate	FP Rate	Precision	ROC Area	MAE	Time taken to build model (seconds)
Stacking + SGD	1.000	0.000	1.000	1.000	0.0000	0.53
Stacking + SMO	1.000	0.000	1.000	1.000	0.0000	0.54
Stacking + MLP	1.000	0.000	1.000	1.000	0.0154	0.54

TABLE VI. Results of Predicting Software Reuse Using Bagging

Prediction Model (Full Dataset)	TP Rate	FP Rate	Precision	ROC Area	MAE	Time taken to build model (seconds)
Bagging + SGD	0.913	0.105	0.913	0.992	0.0478	0.04
Bagging + SMO	1.000	1.000	1.000	1.000	0.0304	0.49
Bagging + MLP	0.957	0.082	0.959	1.000	0.0618	8.86

TABLE VII. Evaluation Criteria Results of Using Bagging with Feature Selection

Prediction Model (Selected Features)	TP Rate	FP Rate	Precision	ROC Area	MAE	Time taken to build model (seconds)
Bagging + SGD	1.000	0.000	1.000	1.000	0.0087	0.01
Bagging + SMO	1.000	0.000	1.000	1.000	0.0087	0.13
Bagging + MLP	1.000	0.000	1.000	1.000	0.0711	0.45

3) Comparison Performance Results Before and After Feature Selection

Features selection technique helps to improve the prediction model's performance. The bagging models have benefited from feature selection. For instance, the time taken to build the model has significantly improved, especially for bagging with MLP, where the time taken has decreased from 8.86 to 0.54 seconds, as shown in Table VI and Table VII. However, Figure 5 shows the error rate of bagging with MLP has slightly increased by 0.00093 points. On the other hand, the MAE of bagging with SGD and SMO have efficiently improved by 0.0391 and 0.0217 points, respectively.

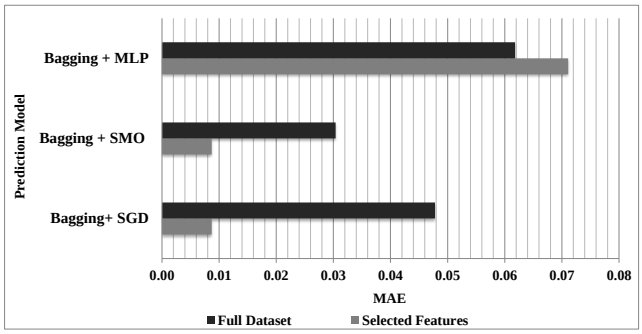

Figure 5. Effect of Feature Selection on MAE rates

C. Voting

This subsection of presents the voting results of the voting ensemble prediction models before and after feature selection.

1) Prediction Using Full Dataset

Table VIII shows that the best combination rule to classify the dataset was the majority voting. The majority voting got the best TPR and FPR results that is, 1 and 0 consecutively. Therefore, the classified majority voting has classified all instances correctly. Meanwhile, the worst combination rule was the maximum probability since it scored the lowest TPR and the highest FPR. Also, the error rates are considered low. The MAE results were between 0.0952 and 0.2126. Furthermore, the products of probabilities and minimum probability voting rules failed to classify two instances.

Although voting with majority voting has the best results of the evaluation criteria, the model is the least effective since it took 0.93 seconds to build it. Two voting models that use the product of probabilities and minimum probability could not classify two instances, while the rest of the voting models classified all the instances.

The ROC area results measure the prediction model's ability to distinguish between the dataset's classes. The closer the ROC area value to one, the better. According to Table VIII, voting with majority voting has the best performance, followed by voting with average probabilities. Meanwhile, the least performance model was the voting using minimum possibility and product of probabilities. Also, voting with maximum probability has the least precision results, which 0.869 meanwhile, the most precise voting model use the majority voting.

2) Prediction After Feature Selection

Table IX illustrates the results of the voting prediction models after applying the feature selection technique. The results showed that two of the voting models have correctly classified all the dataset instances. These voting models used the average of probabilities, majority voting. The true positive rates and false positive rates for the voting with products of possibilities and minimum probability were 1 and 0, respectively. However, These products of possibilities and minimum probability could not classify one instance. Moreover, the TPR and FPR of voting with maximum probability indicate that the model had misclassified some instances.

Furthermore, the voting models' error rates were minimal and varied between 0.140 and 0. In addition to that, the ROC area results show that all voting models have a good performance in predicting software reuse experience. Furthermore, voting using the maximum probability, as a voting rule, was the least precise model among the voting model, shown in Table IX.

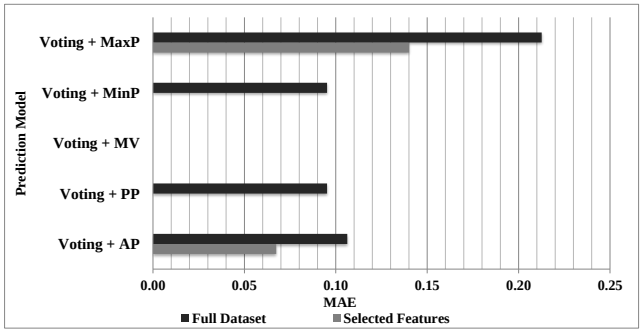

Figure 6. Progression of MAE Before and After Using Feature Selection

3) Comparison Performance Results Before and After Feature Selection

This subsection highlights the features selection technique's effect on the prediction models' performance. In general, there is a significant improvement. The time taken to build the model was reduced for all voting models in an average of 0.838 seconds. Also, the true positive rates and false positive rate of voting with average of probabilities, products of possibilities, and minimum probability improved to the best results, which is 1 and 0, respectively. Moreover, the number of unclassified instances for voting with products of probabilities and minimum probability has reduced to one instance, as shown in Table VIII and Table IX.

Figure 6 illustrates the error rate improvement before and after applying feature selection techniques. MAE result for voting with majority voting remains the same, which is 0. Voting with average of probabilities has improved by 0.0389 points. In addition to that, voting with maximum probability has also been enhanced by 0.0725 points. Meanwhile, error rates of voting minimum probability and products of probabilities experienced a reduction to the best error rate results that is 0.

9. Discussion

The f-measure was utilized to evaluate, compare the performance for experimented prediction models. Moreover, two evaluation criteria were used to identify the best and worst ensemble prediction models. These evaluation criteria are the f-measure and MAE. Figure 7 illustrates the stacking models' performance also depicts the effect of using the Wrapper Subset Evaluation as a feature selection technique. Overall, the performance of all stacking ensemble models had improved after applying feature selection. Stacking with SMO as a meta-learner witnessed the best progression. The f-measure has jumped from about 0.8 to 1. This shows the wrapper subset evaluation has a positive impact on the stacking ensemble learning performance. The best stacking ensemble model was the stacking with SMO since the MAE and f-measure have achieved the best results, which are 0 and 1, sequentially. Meanwhile, stacking with MLP is

TABLE VIII. Performance Results of Applying Voting Learning on The Full Dataset

Prediction Model (Full Dataset)	TP Rate	FP Rate	Precision	ROC Area	MAE	Time taken to build model (seconds)	Unclassified Instances
Voting + AP	0.913	0.105	0.913	0.992	0.1063	0.92	0
Voting + PP	0.905	0.119	0.905	0.861	0.0952	0.91	2
Voting + MV	1.000	0.000	1.000	1.000	0.0000	0.93	0
Voting + MinP	0.905	0.119	0.905	0.861	0.0952	0.88	2
Voting + MaxP	0.870	0.186	0.869	0.963	0.2126	0.89	0

TABLE IX. Evaluation Criteria of Voting Learning on Selected Features

Prediction Model (Selected Features)	TP Rate	FP Rate	Precision	ROC Area	MAE	Time taken to build model (seconds)	Unclassified Instances
Voting + AP	1.000	0.000	1.000	1.000	0.067	0.08	0
Voting + PP	1.000	0.000	1.000	0.980	0.000	0.06	1
Voting + MV	1.000	0.000	1.000	1.000	0.000	0.07	0
Voting + MinP	1.000	0.000	1.000	0.980	0.000	0.06	1
Voting + MaxP	0.957	0.082	0.959	1.000	0.140	0.07	0

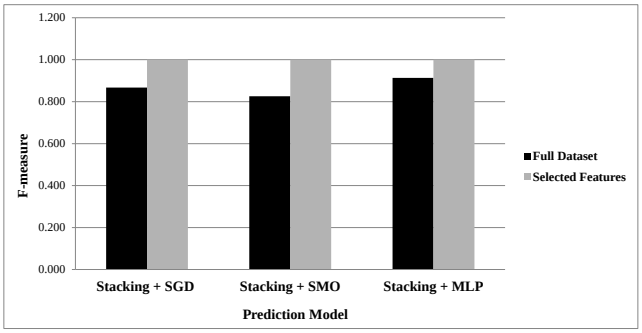

Figure 7. Performances of Stacking Prediction Models Before and After Feature Selection

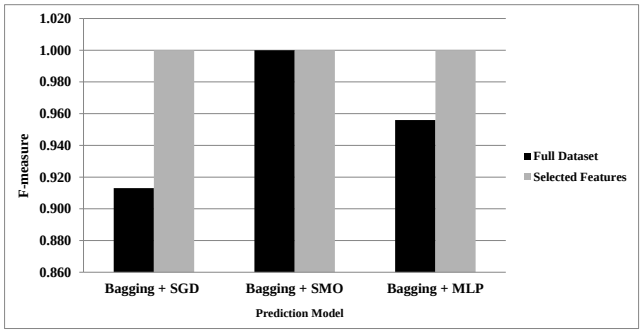

Figure 8. Performances of Bagging Prediction Models Before and After Feature Selection

considered the worst since it scored the highest MAE even after applying feature selection, as shown in Figure 4.

Figure 8 depicts the bagging ensemble learning performance with and without feature selection. In general, all bagging prediction models have scored the best results for the f-measure after utilizing feature selection performance. The performance of bagging using SGD on the full dataset was the worst. However, bagging with SGD performance had sharply improved when feature selection was introduced to the model. Meanwhile, the feature selection had neither a positive nor negative impact on bagging with SMO. The f-measure of bagging with SMO was one, which is the highest result a prediction model could achieve. The performance of bagging with MLP has significantly improved; however, the error rate has slightly increased after applying feature selection, as shown in Figure 5 . When comparing the error rates in Figure 5 and performance in Figure 8, it can be educed that introducing Wrapper Subset Evaluation to bagging with MLP is best in terms of performance and efficiency. Meanwhile, bagging with MLP is the worst among the bagging models.

The impact of the feature selection on the voting ensemble prediction models was also positive. However, voting ensemble prediction with majority voting did not improve when applying feature selection since the majority voting outperformed all the other voting rules with or without feature selection. The model had scored the best performance results, which is one, as shown in Figure 9. Also, Figure 6 shows that majority voting has the lowest error rates. Meanwhile, the maximum probability has the most deficient performance and error rates among the voting prediction models.

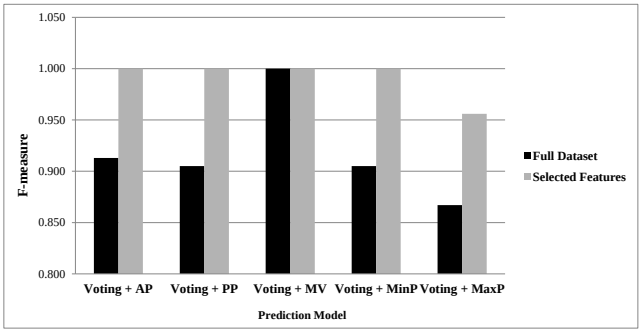

Figure 9. Performances of Voting Prediction Models Before and After Feature Selection

10. Threats to Validity

There is no machine learning algorithm that superiorly classifies all the instances. Therefore, another classifier may obtain different outcomes while classifying the dataset. Moreover, the size of the dataset is extremely small, which may affect the classifier's performance. Also, the prediction models may react differently if the dataset was the larger than the used dataset.

11. Conclusion

It has been more the half of century since the NATO science committee expressed its concerns about software development productivity and the quality of the developed software. The experts and researchers in the software engineering discipline are determined to increase software development productivity and reduce the development time and effort. Software reuse is one of the practices which software development professional adopt to solve NATO concerns. However, some factors may prevent the success of reusing software assets. The software engineering discipline has to consider the non-technical factors that contribute to the success of the software development project. In this work, ensemble learning was utilized to ensure the success of the software reuse experience. Three ensemble learning algorithms were selected. These algorithms are stacking, bagging, and voting. Also, Wrapper Subset Evaluation was used to improve the outcome of the ensemble learning. The results showed that ensemble leaner had performed very well in distinguishing the successful software reuse experience. Moreover, feature selection positively impacted the ensemble learner's prediction where the accuracy researched 100% in some experiments. The experimental results also showed that voting using the majority voting obtained the optimal results compared to other ensemble learning models. Meanwhile, bagging with SGD on the full dataset was the worst ensemble prediction model. However, the used dataset is relatively small. Also, the portion of the failure software reuse instance to successful software reuse instances is around 1:2. This portion indicated an imbalance in the used dataset. The future work will concentrate on finding the appropriate techniques to solve the dataset size and imbalance.

References

[1] P. Naur and B. Randell, "Report on the nato software engineering conference," *NATO Scientific Affairs Division*, 1968.

[2] M. M. Lehman, "Laws of software evolution revisited," in *European Workshop on Software Process Technology*. Springer, 1996, pp. 108–124.

[3] Lampropoulos *et al.*, "React-a process for improving open-source software reuse," in *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*. IEEE, 2018, pp. 251–254.

[4] J.-L. Lions *et al.*, "Flight 501 failure," *Report by the Inquiry Board*, vol. 190, 1996.

[5] A. B. AL-BADAREEN *et al.*, "In the process of software development: Available resources and applicable scenarios," *Journal of Theoretical and Applied Information Technology*, vol. 99, no. 1, 2021.

[6] J. Krüger and T. Berger, "An empirical analysis of the costs of clone- and platform-oriented software reuse," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 432–444.

[7] M. T. Patrick, "Exploring software reusability metrics with q&a forum data," *Journal of Systems and Software*, vol. 168, p. 110652, 2020.

[8] M. Abbas *et al.*, "Automated reuse recommendation of product line assets based on natural language requirements," in *International Conference on Software and Software Reuse*. Springer, 2020, pp. 173–189.

[9] N. Ali *et al.*, "A hybrid devops process supporting software reuse: A pilot project," *Journal of Software: Evolution and Process*, vol. 32, no. 7, p. e2248, 2020.

[10] A. Lahouij *et al.*, "Dynamic reconfiguration of cloud composite services using event-b," in *International Conference on Software and Software Reuse*. Springer, 2020, pp. 69–84.

[11] P. Negi *et al.*, "Machine learning algorithm for assessing reusability in component based software development," EasyChair, Tech. Rep., 2020.

[12] A. K. Sandhu *et al.*, "Software reuse analytics using integrated random forest and gradient boosting machine learning algorithm," *Software: Practice and Experience*, 2020.

[13] B. A. Prakash *et al.*, "Application of data mining techniques for software reuse process," *Procedia Technology*, vol. 4, pp. 384–389, 2012.

[14] M. Hammad *et al.*, "Deepclone: Modeling clones to generate code predictions," in *International Conference on Software and Software Reuse*. Springer, 2020, pp. 135–151.

[15] C. Zhang *et al.*, "Transformer fault diagnosis method using iot based monitoring system and ensemble machine learning," *Future Generation Computer Systems*, vol. 108, pp. 533–545, 2020.

[16] S. Lahmiri *et al.*, "Performance assessment of ensemble learning systems in financial data classification," *Intelligent Systems in Accounting, Finance and Management*, vol. 27, no. 1, pp. 3–9, 2020.

[17] L. Brunese *et al.*, "An ensemble learning approach for brain cancer detection exploiting radiomic features," *Computer methods and programs in biomedicine*, vol. 185, p. 105134, 2020.

[18] X. Gao *et al.*, "An adaptive ensemble machine learning model for intrusion detection," *IEEE Access*, vol. 7, pp. 82 512–82 521, 2019.

[19] W. Feng *et al.*, "Class imbalance ensemble learning based on the margin theory," *Applied Sciences*, vol. 8, no. 5, p. 815, 2018.

[20] M. Morisio *et al.*, "Success and failure factors in software reuse," *IEEE Transactions on software engineering*, vol. 28, no. 4, pp. 340–357, 2002.

[21] X. Deng *et al.*, "An improved method to construct basic probability assignment based on the confusion matrix for classification problem," *Information Sciences*, vol. 340, pp. 250–261, 2016.

[22] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

[23] M. Amin and M. Hammad, "Improving software reuse prediction using feature selection algorithms," in *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*. IEEE, 2020, pp. 1–6.

[24] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[25] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and systems magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[26] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[27] P. Bühlmann, B. Yu *et al.*, "Analyzing bagging," *The Annals of Statistics*, vol. 30, no. 4, pp. 927–961, 2002.

[28] L. I. Kuncheva, *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2014.

[29] B. Parhami, "Voting algorithms," *IEEE transactions on reliability*, vol. 43, no. 4, pp. 617–629, 1994.

[30] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, "On combining classifiers," *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 3, pp. 226–239, 1998.

[31] C. Sammut and G. I. Webb, *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

[32] S. Alabdulwahab and B. Moon, "Feature selection methods simultaneously improve the detection accuracy and model building time of machine learning classifiers," *Symmetry*, vol. 12, no. 9, p. 1424, 2020.

[33] A. Özdemir *et al.*, "Performance evaluation of different classification techniques using different datasets," *International Journal of Electrical and Computer Engineering*, vol. 9, no. 5, p. 3584, 2019.

[34] T. C. Smith and E. Frank, "Introducing machine learning concepts with weka," in *Statistical genomics*. Springer, 2016, pp. 353–378.

Mariam Amin received her M.Sc. degree in Software Engineering from the University of Bahrain in 2022. In 2005, she received her B.Sc. in Computer Science from the University of Bahrain. Also, she works as a Computer Specialist at Bahrain Defense Force, Royal Medical Services. Her current research focus is in software analysis and evolution and machine learning.

Mustafa Hammad is an Associate Professor in the Department of Computer Science at the University of Bahrain. He received his Ph.D. in Computer Science from New Mexico State University, the USA, in 2010. He received his Master's Degree in Computer Science from Al-Balqa Applied University, Jordan, in 2005 and his B.Sc. in Computer Science from The Hashemite University, Jordan, in 2002. His research interests include wireless sensor network, machine learning, software engineering with a focus on software analysis and evolution.