A New Weighted Ensemble Model to Improve the Performance of Software Project Failure Prediction

Mohammad A. Ibraigheeth[1], Aws I. Abu Eid[2*], Yazan A. Alsariera[3], Waleed F. Awwad[4], Majid Nawaz[5]
Department of Software Engineering, Bethlehem University, Bethlehem, Palestine[1]
Faculty of Computing Studies, Arab Open University, Amman, Jordan[2]
Department of Computer Science-College of Information and Communications Technology,
Tafila Technical University, Tafila, Jordan[3]
Department of Computer Science-Collage of Science, Northern Border University, Arar, Saudi Arabia[4, 5]

Abstract*—The development of a software project is frequently influenced by various risk factors that can lead to project failure. Predicting potential software project failures early can aid organizations in making decisions regarding possible solutions and improvements. This paper proposes a software project failure prediction model based on a weighted ensemble learning approach. The proposed model aims to determine the failure probability as well as the expected project outcome (Success/Failure). Various ensemble approaches, such as simple majority voting, can be employed in predicting software project failure. However, in majority voting algorithms, all base models have the same weights, resulting in an equal effect on the final prediction result, regardless of their predictive abilities. Our proposed algorithm assigns higher weights to base models that demonstrate a greater ability to correctly predict more challenging data instances. The proposed model is developed based on a dataset gathered from real previous software project reports, comprising both successful and failed projects, to provide evidence supporting the predictive model's capabilities and to obtain high-confidence results. The performance of the developed model is comprehensively assessed through various measures, revealing its superiority in predicting software project failures compared to both simple majority voting and individual models. This research suggests that the proposed model can be integrated into the software system development process, spanning requirement analysis, planning, design, and implementation phases, to evaluate the project's status and identify potential risks.

***Keywords*—Ensemble learning; failure prediction; base models; project outcome**

I. Introduction

Assessing the probable software project failure early during development process can mitigate the effect of the undesirable events that could lead to project failure [10]. The paper aims to develop new weighted ensemble predictive model which use historical failure data gathered from several past software projects to accurately predicting possible failures in future software projects. The developed model can be used early in the system software engineering process at inception and planning phase when decisions are being made to specify the projects to be embarked upon in the project portfolio. Furthermore, this model can be used during any phase of software development process to avoid project failure and improve reliability.

Ensemble learning is selected because it has been observed that this method achieves better results in terms of diversity and accuracy [1]. Using ensemble methods improve prediction results by combining abilities of different single predictors into one prediction model [7]. As these single predictors differ in the approach used, parameters, and dealing with training data, combining prediction abilities of these predictors enable the ensemble algorithms to capture different characteristics of the training data and produce more reliable and accurate prediction [7].

Ensemble learning is a machine learning technique where multiple base models are combined to produce one optimal model ([3]; [4]). The ensemble model constructs a set of base models on training data and then combines them or selects the best one to use [11]. The objective of this technique is to improve the model predictive accuracy over traditional single component models [18]. In many cases, the ensemble predictors show higher performance than other individual prediction models [1], [7]. According to [7], there are three reasons why this technique can improve the prediction accuracy:

1) The single component models learn from training data to perform prediction of the new examples. However, it can be hard to perform accurate prediction when the amount of training data is small. This problem can be solved by constructing a set of base models (combined to one ensemble model) and find the optimal prediction result.

2) Several prediction techniques use local search approaches such as gradient decent to find the optimal class. Even if the available training data is enough, these searches can stick to a local optimum. Since finding the global optimum can be computationally expensive, ensemble classifiers perform multiple local searches started at different data points to find the optimal class.

3) In such situations, it can be hard to find the optimal solution in the search space of the single classifiers. A combination of multiple classifiers could approximate the optimal solutions than the separated single classifiers. An example of a two-class classification problem is shown in Fig.1. In this example, none of the three classifiers A, B, and C can separate the two classes (+ and −) perfectly. The ensemble classifier as illustrated by a bold line in Fig. 1, that

*Corresponding Author.

combines the three single classifiers is capable of classifying the two classes accurately.

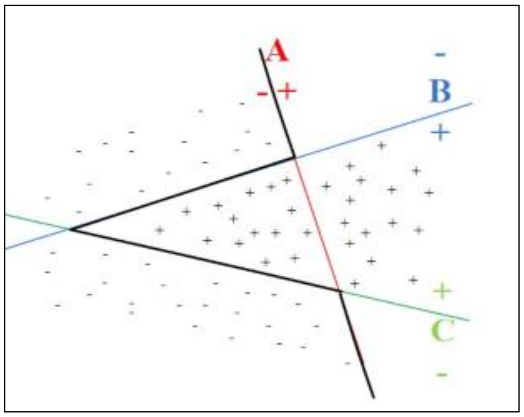

Fig. 1. Example of the three classifiers.

Although these reasons show that the ensemble classifiers could perform better than single classifiers, ensemble learning needs enough diversity to obtain accurate results [13]. This means that the classifiers should produce different errors in order to be able to learn from each other. When the classifiers make nearly the same errors, they will behave like a single classifier.

In this paper, a new weighted ensemble prediction model is proposed to predict the software project failure. This model combines ensemble-learning prediction with the predictor selection approach. The proposed algorithm incorporates six base models, namely, neural networks (NNs), logistic regression (LR), support vector machine (SVM), naïve Bayes (NB), adaptive neurofuzzy inference system (ANFIS), and decision trees (DT). These methods are selected because they show adequate prediction performance according to the study conducted by Ibraigheeth and Eid [9]. We suggest that the different prediction abilities of these six methods enable the proposed algorithm to capture different characteristics of the training data and produce more reliable and accurate prediction.

The proposed algorithm assigns a unique "ranking number" to each base model according to its ability to predict the most difficult-to-predict data. A higher performance model on the difficult-to-predict data will be assigned higher ranking numbers. A normalize weighted vector is constructed based on these ranking numbers, and the final probability of failure result is obtained based on the weight assigned for each base model.

In the proposed method, when the base models are constructed, a unique "ranking number" is assigned to each one according to its performance result over the data subset with lower average performance result, which was the most difficult subset to predict. The algorithm constructs a performance vector for each base model over all data subsets. In addition, the approach constructs a vector that represents the average performance results over each data subset for all base models. Furthermore, a ranking vector for base models is constructed as well as a normalized weight vector, which represents the weights of the base models.

II. Related Works

Over the past years, many ensemble approaches have been proposed. According to [5], ensemble methods can be categorized into two types: homogeneous and heterogeneous ensembles. In the homogeneous ensemble, the same learning algorithm using different training subsets trains a set of individual models. The final decision is taken by combining the outputs of these models. Examples of such ensemble methods in the literature include bagging [15], AdaBoost [37], and Random Forest [16]. In the heterogeneous ensemble, different learning algorithms using the same training set generate different models. The heterogeneous ensemble learning emphasizes more on meta-data combination techniques ([17]; [26-30]) to achieve a higher performance than an individual model. Wang and Zhong [33] applied the information granularity approach to develop an ensemble system combining multiple classifiers. First, the weighted distances between granularity prototypes and the base classifiers outputs are observed. Then, the shortest distance prototype is selected to predict the class label. Wu [35] proposed a new weighted ensemble method that considers the performance information for the base models in previous literature to obtain their optimal weights. Blaser and Fryzlewicz [22] developed a new ensemble system that generates the base models after generating matrices to rotate the features space. Moreover, different learning methods were applied for many ensemble systems, such as supervised learning [39], incremental learning ([2], [14], [36]) and multilabel classifiers ([12], [21]). Several researches focused on enhancing the performance of the existing ensemble approaches. Several methods were applied for this purpose, e.g., clustering approach [24], dynamic classifiers selection [23], and hybrid methods used in a random subspace to assign weights for the base classifiers [40]. Hybrid ensemble, which combines sample and feature space-based learning was proposed in [38]. Several techniques have been proposed to enhance AdaBoost performance, for example, by applying linear programming to maximize the margin between different classes and training instances [20].

Even though there are many researches concentrate on addressing software project failures [8], [31], [32], [41], most of these researches don't perform project failure prediction. Ewusi-Mensah [8] was aimed to identify the impact of different failure factors on the SDLC stages. The empirically based study defines the reasons behind these factors and how they can prevented. Takagi et al. [31] performed a questionnaire based approach in order to determine core risk factors. A logistic regression model is used to characterize the confused projects, and to predict if the software project is risky or not risky. However, the developed model does not predict failures. Verner et al. [32] investigated number of failed projects to determine the factors behind project failing. This research aimed only to identify failure factors, and it did not predict the project failures. Rayes et al. [25] also propose an effective project resources allocation to maximize the probability of the project success. The authors suggest a strategy for effective resources allocation to get high success rate with minimum cost. The developed model was to identify and control the risks that affect the project success. However, the failure prediction is not observed also in this research.

Wang et al. [34] developed a predictive model based on Bayesian Network in order to predict the software projects outcome through prediction and controlling the variance in estimated project schedule. This research aimed to maximize the opportunity to complete the project on time through project re-planning, resource re-allocation, and schedule variance factors identification. Therefore, no software failure prediction is performed in this research. Lehtinen et al. [19] performed analysis in corporation with four software organizations to recognize the reasons behind failures and the relations among them. Their research developed diagrams that describe causal relationships among failure causes, and they recommended performing specific analysis for each cause, and managing these causes outside the development area to prevent the software project failure. However, a limitation of this research is the limited number of failure analyzed cases.

Most of previously developed methods were applied on certain case studies. Consequently, those methods might not be applicable for other software projects. Furthermore, many of these methods were implemented to assess the failure through specific phase of software development process. In this context, one of main contributions of our research is developing a new prediction model that can be applied on any software project during any phase of software development process.

III. Methodology

In the initial phase of this study, careful consideration is given to the selection of model inputs, also known as predictors, and the acquisition of a suitable dataset. This process involves identifying key factors that may influence the outcome of software projects, such as project size, complexity, development methodologies, and team composition. Subsequently, the dataset is divided into two distinct sets: the training set and the testing set. The training data serves as the foundation for model development, where algorithms are trained and fine-tuned to learn patterns and relationships within the data. Meanwhile, the testing data is reserved for evaluating the performance of the trained models, providing an independent measure of their predictive accuracy and generalization capabilities. Upon successful development and validation, the deployed model becomes a valuable asset in the software project development lifecycle. By leveraging historical project data and learned patterns, the model can effectively forecast the future outcomes of ongoing or upcoming projects. This predictive capability empowers project stakeholders with valuable insights, enabling informed decision-making and proactive risk management throughout the software development process.

For building the model, the list of failure factors identified by Ibraigheeth and Fadzli [10] is selected to be the model input. This list of identified factors is presented in Table I. The dataset constructed in their research is also selected to fit and verify the developed models. This dataset was gathered from 236 (n = 236) failed and successful software projects and used in our research to fit and verify the developed models.

TABLE I. List of Failure Factors

ID	Failure Factor
X1	Unrealistic project objectives
X2	Team technical problems
X3	Lack of users involvement
X4	Requirements instability
X5	Problematic technology
X6	Problems in project management
X1	Unrealistic project objectives

Table II presents the sample of collected data, which identifies the six failure factors results in 10 software projects. This table illustrates the actual projects outcome (0: Success and 1: Failed).

TABLE II. Data Sample of Actual Outcome for 10 Software Projects

Project ID	Failure factors						Actual outcome
	X1	X2	X3	X4	X5	X6	
P121	0	1	0	0	1	1	0
P130	0	0	0	1	1	1	1
P131	0	1	1	0	1	1	1
P132	0	0	1	0	0	1	0
P133	0	0	1	1	1	1	1
P134	0	1	1	1	0	1	1
P143	0	0	0	1	0	0	0
P153	0	0	1	1	0	0	0
P161	0	0	0	0	1	1	0
P175	0	1	0	1	0	1	1

IV. Developing the Model

In this section, the ensemble-weighted algorithm, which combines six base prediction models, is developed.

The algorithm begins with randomly splitting the training dataset into n subsets, and then each base model is trained over all subsets. The average prediction performance for all base models over each subset is measured. Then, we identify the subset on which the base models achieved the worst average performance. The lowest performance rate indicates that this subset was the most difficult to predict. A unique “ranking number” is assigned to each base model according to its performance result over this subset.

The approach constructs four types of vectors:

1) A performance vector for each base model over all data subsets,

2) A vector represents the average performance results over each data subset for all base models,

3) A ranking vector represents the ranks of base models, and

4) A normalized weight vector represents the base models weights.

The proposed algorithm is described as follows:

a) Select the optimal predictor subset.

b) Randomly split the training dataset into n subsets.

c) Fit base models using all training data subsets.

d) Set the vector for each base model that represents the average performance for this model over each data subset. For i = 1 to k, calculate Pi,

$$P_i = \frac{AC_i + F_i + K_i + AUC_i}{4} \tag{1}$$

where, AC is the accuracy, F is the F-measure, k is the kappa coefficient, and AUC is the area under the receiver operating characteristic curve.

e) A vector that represents the average of performance results (of all base models) is set over each data subset.

$$AvP_j = \frac{\sum_{i=1}^{n} P_j}{n} \tag{2}$$

where, AvP_j is the average performance value for all base models over subset j. The subset with lowest AvP is the most difficult subset to predict.

f) A ranking vector that represents the ranks of base models according to their performance over the most difficult-to-predict data subset is set. Each base model gets a ranking number from 1 to k (k is the number of base models). The higher performance model (over the most difficult data subset) gets a higher ranking number.

g) Set a vector that represents the base model weights.

The weight of model m can be estimated by:

$$w_m = \frac{R_m P_m}{\sum_{i=1}^{k} R_i P_i} \tag{3}$$

where, k is the number of base models, and R is the base model rank (from 1 to k) over the most difficult dataset.

In the proposed algorithm, the weight assigned to each base model mm is determined by Eq. (3). This equation calculates the weight wmwm based on the rank and performance of the model mm relative to other base models in the ensemble. Here's a detailed explanation:

i) Base Model Rank R_m: The rank of a base model mm represents its performance relative to other models in the ensemble on the most challenging dataset instances. Models that exhibit better predictive capabilities or accuracy are assigned lower ranks, indicating higher effectiveness in handling difficult data instances.

ii) Base Model Performance P_m: The performance of each base model *m* is measured by its predictive ability or accuracy. Higher performance models, which accurately predict the outcomes of software project instances, are assigned higher values for P_m.

iii) Normalization Factor $\sum_{i=1}^{k} R_i P_i$: This term represents the sum of the ranks multiplied by the corresponding performance measures for all base models in the ensemble. It acts as a normalization factor to ensure that the weights w_m sum up to 1, thereby maintaining the integrity of the weighted ensemble.

h) The final predicted value Pr (probability of failure) is obtained according to each base model weight:

$$\Pr = \sum_{i=1}^{k} w_i D_i, \tag{4}$$

where, Di is the predicted value of base model i.

Eq. (4) calculates the final predicted value of the probability of failure (Pr) based on the weighted contributions of each base model in the ensemble. Here's a detailed explanation:

i) Base model weight w_i: Each base model ii in the ensemble is assigned a weight wiwi determined by its effectiveness in predicting software project failures. The weight reflects the relative importance or influence of the corresponding model in the ensemble. Models with higher weights contribute more significantly to the final prediction.

ii) Predicted value D_i: D_i represents the predicted value of failure probability by the base model ii. Each base model generates its own prediction based on its internal algorithms and training data. These predicted values represent the likelihood of failure for individual software project instances.

iii) Weighted summation: The final predicted value of failure probability (Pr) is obtained by summing the weighted contributions of all base models in the ensemble. Each predicted value D_i is multiplied by its corresponding weight wiwi, and these weighted values are summed up for all k base models in the ensemble.

By aggregating the predictions from multiple base models according to their respective weights, Eq. (4) produces a composite prediction of failure probability that leverages the strengths of individual models while mitigating the impact of potential weaknesses. This weighted ensemble approach enhances the overall accuracy and reliability of the prediction, providing a more robust assessment of the likelihood of failure for software project instances.

To illustrate the algorithm, a simple example of ensemble with three base models and three data subsets is considered. We define P = (P1| P2 |P3) and let P1 = (0.77, 0.66, 0.84), P2 = (0.78, 0.90, 0.81) and P3 = (0.97, 0.60, 0.78), where Pi represents the performance vector of base model i over the three data subset. We obtain AvP = (0.84, 0.72, 0.81), which indicates the average performance of the three base models over the three data subset. According to AvP values, the second data subset was the most difficult subset to predict as it gets the lowest average performance score (0.72). Therefore, we rank the base models according to their performance results over the most difficult subset to predict. The second base model will get the higher rank number = 3 as it achieved a higher performance (0.90) result over the most difficult subset. The rank number = 2 is given to first base model, and rank number = 1 is given to the third base model. The normalized weight vector W=(0.286,0.584,0.13) is obtained by:

$$W = \left(\frac{\mathbf{0.66 \times 2}}{(\mathbf{0.66 * 2})+(\mathbf{0.90 \times 3})+(\mathbf{0.60 \times 1})}, \frac{\mathbf{0.90 \times 3}}{(\mathbf{0.66 \times 2})+(\mathbf{0.90 \times 3})+(\mathbf{0.60 \times 1})}, \frac{\mathbf{0.60 \times 1}}{(\mathbf{0.66 * 2})+(\mathbf{0.90 \times 3})+(\mathbf{0.60 \times 1})}\right)$$

$$W = (0.286, 0.584, 0.13)$$

The highest weight is given for the second base model as it obtained a higher rank according to its performance in predicting the most difficult data subset to predict.

Finally, the prediction value for each data instant is obtained based on the above base model weights. Suppose that the three base models generate probabilities of failure: 0.66, 0.35, and 0.74; therefore, the final failure probability Pr generated by the model based on the estimated weight is 0.49.

$$Pr=(0.286 \times 0.66)+(0.584 \times 0.35)+(0.13 \times 0.74)= 0.49$$

The failure probability result is used to classify the expected project outcome (failed/success). The default probability value 0.5 is selected to be the classification threshold, with failure expected for any result higher than 0.5. Future research can be conducted to determine the optimal threshold for determining project failure.

V. Experimental Results

For comparison purpose, in addition to building the proposed model, the experiments included running the model using four methods. Initially, the model is implemented using three of the existing individual prediction techniques: LR, SVM, and ANFIS. These methods were chosen because they showed a high efficiency in failure prediction compared with other methods in a study conducted by Ibraigheeth and Eid [6]. These three models were combined to create a simple majority voting model. To run the simple majority voting model, the training dataset is used to build the three base models (LR, SVM, and ANFIS), and then the final prediction decision for any test instance is generated by majority voting. The new weighted ensemble model is implemented and tested in terms of its ability to predict software project failures. The experiments included calculating eight performance measures: sensitivity or recall, specificity, precision, negative predictive value, accuracy, F-measure, kappa coefficient, and AUC value.

Several statistical tests were applied on the proposed model to evaluate its performance. Confusion matrix that includes information about actual and predicted model outputs is shown in Fig. 2. For the project failure prediction problem, the confusion matrix is used to evaluate the model performance. The column of the confusion matrix represents the actual result (class), while the row represents the predicted result. TP (True Positive) and TN (True Negative) denote how many instances are classified correctly, while FP (False Positive) and FN (False Negative) denote how many instants are classified incorrectly.

		Predicted output	
		+	−
Actual Output	+	T P = 1 2	F N = 1
	−	F P = 1	T N = 1 0

Fig. 2. Confusion matrix.

TABLE III. Performance Measures

Measure	Value
Sensitivity (Recall)	0.923
Specificity	0.909
Precision	0.923
Negative predictive value	0.909
Accuracy	0.916
F-measure	0.923
Kappa	0.914
AUC	0.96
Average	0.92

Several performance evaluation metrics can be generated from the confusion matrix. Table III shows performance measures of the proposed model over testing dataset.

A comparative evaluation for the proposed weighted ensemble prediction model is performed. In this paper, the ensemble model was run to predict the software project failure. Table IV shows a summary of the performance measure for the proposed ensemble model versus the simple majority voting model as well as the other three individual models: LR, SVM, and ANFIS.

Table IV shows that the proposed weighted ensemble model has the highest values for most of performance measures; it has an average performance of 92% compared with 89% for the proposed majority voting model, 81% for LR model, 82% for SVM model, and 83% for ANFIS. Experiments also prove that the simple majority-voting model performs better than individual models.

TABLE IV. Proposed Weighted Ensemble Predictive Model Performance Measures

Measure	Proposed weighted ensemble model	Simple majority voting model	LR	SVM	ANFIS
Sensitivity (Recall)	0.9 2	0. 9 2	0. 8 5	0.7 7	0. 8 1
Specificity	0.9 0	0. 8 4	0. 7 7	0. 9 1	0. 9 6
Negative predictive value	0.9 2	0. 8 7	0. 8 9	0. 9 0	0. 9 6
Accuracy	0.9 0	0. 9 0	0. 7 1	0. 8 3	0. 8 2
Precision	0.9 1	0. 8 9	0. 8 3	0. 8 4	0. 8 8
F-measure	0.9 2	0. 9 0	0. 8 7	0. 8 3	0. 8 8
Kappa	0.9 1	0. 8 6	0. 7 9	0. 6 9	0. 7 7
AUC	0.9 6	0. 9 4	0. 9 4	0. 8 4	0. 6 2
Average	0.9 2	0. 8 9	0. 8 1	0. 8 2	0. 8 3

Table V illustrates a sample of the data of 10 software projects and their corresponding actual outcome, the proposed weighted ensemble model predicted outcome. The table illustrates that all projects except P16 were labeled correctly. Projects P11, P13, P17, P19, and P20 were correctly labeled as failed projects with actual outcome = 1, and Projects P12, P14, P15, and P18 were correctly labeled as success projects with actual outcome =0. Project P16 was inaccurately labeled as success (Predicted outcome = 0) when the projects were failed (Actual outcome = 1).

In the comparative evaluation of the proposed weighted ensemble prediction model, the analysis delves into its performance in relation to alternative methodologies. By subjecting the ensemble model to prediction tasks for software project failure, a comprehensive understanding of its efficacy is garnered. Table IV elucidates the performance metrics, showcasing the superiority of the proposed weighted ensemble model over the simple majority voting model and individual models such as LR, SVM, and ANFIS. Notably, the ensemble model consistently achieves higher performance across various metrics, with an average accuracy of 92%, outperforming its counterparts. Moreover, insights gleaned from experimentation highlight the advantageous nature of employing a simple majority-voting model over relying solely on individual models. As the software industry navigates complex project landscapes, such evaluations play a pivotal role in informing decision-making processes and shaping future research directions

TABLE V. SAMPLE OF ACTUAL AND PREDICTED OUTCOMES

Project ID	Actual outcome	Predicted outcome
P 1 1	1	1
P 1 2	0	0
P 1 3	1	1
P 1 4	0	0
P 1 5	0	0
P 1 6	0	1
P 1 7	1	1
P 1 8	0	0
P 1 9	1	1
P 20	1	1

VI. CONCLUSION

In this paper, a new ensemble weighted model is proposed for predicting software project failures. The proposed algorithm incorporates six base models to provide the final decision of the software project outcome. These models are: NNs, LR, SVM, NB, ANFIS, and DT. We suggest that the different prediction abilities of these six methods enable the proposed algorithm to capture different characteristics of the training data and produce more reliable and accurate prediction. The proposed algorithm assigns a unique "ranking number" to each base model according to its ability to predict the most difficult data. Higher performance base models over the most difficult-to-predict data will be assigned higher ranking numbers. A normalized weighted vector is constructed based on these ranking numbers, and the final predicted value is obtained based on the weight assigned for each base model.

In the empirical analysis, eight performance measures are used to evaluate the proposed model performance. The research proves that the weighted ensemble model outperforms the simple majority voting and the individual prediction models. The experiments have also shown that the simple majority-voting model outperforms the other three individual models.

As this paper introduces a novel ensemble weighted model for predicting software project failures, future work could explore several avenues to enhance and extend the proposed approach. Firstly, further investigation could be conducted into the selection and incorporation of additional base models beyond the six currently utilized (NNs, LR, SVM, NB, ANFIS, and DT). This exploration may involve considering emerging machine learning techniques or domain-specific models tailored to software project prediction tasks. Additionally, research efforts could focus on refining the methodology for assigning ranking numbers to base models based on their predictive capabilities across different data instances. Fine-tuning this ranking system could potentially lead to more accurate and nuanced weighting of base models, thereby improving the overall performance of the ensemble model. Furthermore, the evaluation framework utilized in this study could be expanded to include additional performance metrics or consider alternative evaluation methodologies to provide a more comprehensive assessment of the proposed model's efficacy. Lastly, real-world deployment and validation of the model within software development environments could offer valuable insights into its practical utility and effectiveness in mitigating project failure risks. By addressing these future research directions, advancements can be made towards developing more robust and reliable predictive models for software project management.

REFERENCES

[1] A. BEHERA, Rabi Narayan; ROY, Manan; DASH, Sujata. Ensemble based hybrid machine learning approach for sentiment classification-a review. International Journal of Computer Applications, 2016, 146.6: 31-36.

[2] B. Krawczyk, Alberto Cano, Online ensemble learning with abstaining classifiers for drifting and noisy data streams, Applied Soft Computing, 2018, 68, 677-692.

[3] Basaran, K., Özçift, A., & Kılınç, D. A new approach for prediction of solar radiation with using ensemble learning algorithm. Arabian Journal for Science and Engineering, 2019, 44(8), 7159-7171.

[4] BreiGanaie, M. A., & Hu, M. (2021). Ensemble deep learning: A review. arXiv preprint arXiv:2104.02395.man, L. Bagging predictors. Machine learning,1996, 24(2), 123-140.

[5] C.-X. Zhang, R.P.W. Duin, An experimental study of one- and two-level classifier fusion for different sample sizes, Pattern Recogn. Lett, 2011, (32) 1756-1767.

[6] Damen, J. A., Pajouheshnia, R., Heus, P., Moons, K. G., Reitsma, J. B., Scholten, R. J., ... & Debray, T. P. Performance of the Framingham risk models and pooled cohort equations for predicting 10-year risk of cardiovascular disease: a systematic review and meta-analysis. BMC medicine, 2019, 17(1), 109.

[7] Dietterich, T. G. Ensemble methods in machine learning. In International workshop on multiple classifier systems, 2000 (pp. 1-15). Springer, Berlin, Heidelberg.

[8] Ewusi-Mensah, K. Software Development Failures: Anatomy of Abandoned Projects. Cambridge: MIT Press, 2003.

[9] Ibraigheeth, M. A., & Eid, S. A. Software project risk assessment using machine learning appro. In 2022 American Journal of Multidisciplinary Research & Development (AJMRD), 2022, 4,2 (pp. 35-41). IEEE.

[10] Ibraigheeth, M. A., & Fadzli, S. A. Software project failures prediction using logistic regression modeling. In 2020 2nd International

Conference on Computer and Information Sciences (ICCIS), 2020, (pp. 1-5). IEEE.

[11] Idrees, F., Rajarajan, M., Conti, M., Chen, T. M., & Rahulamathavan, Y. PIndroid: A novel Android malware detection system using ensemble learning methods. Computers & Security, 2017, 68, 36-46.

[12] J.M. Moyano, E.L. Gibaja, K.J. Cios, S. Ventura , Review of ensembles of multi-label classifiers: Models, experimental study and prospects, Information Fusion. 44 2018, 33-45.

[13] Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering, 2007, 160, 3-24.

[14] Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., & Woźniak, M. Ensemble learning for data stream analysis: A survey. Information Fusion, 2017, 37, 132-156.

[15] L. Breiman, Bagging Predictors, Machine Learning, 1996, 24, 123-140.

[16] L. Breiman, Random Forests, Machine Learning, 2001,45, 5-32.

[17] L.I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley, 2004.

[18] Laradji, I. H., Alshayeb, M., & Ghouti, L. Software defect prediction using ensemble learning on selected features. Information and Software Technology,2015, 58, 388-402.

[19] Lehtinen, T. O., Mäntylä, M. V., Vanhanen, J., Itkonen, J., &Lassenius, C. Perceived causes of software project failures–an analysis of their relationships. Information and Software Technology,2014 56(6), 623-643.

[20] M. Warmuth, J. Liao, G. Ratsch, Totally corrective boosting algorithms that maximize the margin, in Proc. 23rd Int. Conf. on Machine Learning, 2006, pp. 10011008.

[21] Q. Wu, M. Tan, H. Song, J. Chen, M.K. Ng, ML-FOREST: A Multi-Label Tree Ensemble Method for Multi-Label Classification, IEEE Transactions On Knowledge And Data Engineering. 2016,28(10).

[22] R. Blaser, P. Fryzlewicz, Random Rotation Ensemble, Journal of Machine Learning Research.2, 2015, 1-15.

[23] R.M.O.Cruza, R. Sabourin, G.D.C. Cavalcanti, Dynamic classifier selection: Recent advances and perspectives, Information Fusion, 2018, 41, 195-216.

[24] Rashid, M., Khan, M. A., Sharif, M., Raza, M., Sarfraz, M. M., & Afza, F, Object detection and classification: a joint selection and fusion strategy of deep convolutional neural network and SIFT point features. Multimedia Tools and Applications, 2019, 78(12), 15751-15777.

[25] Reyes, F., Cerpa, N., Candia-Véjar, A., & Bardeen, M. The optimization of success probability for software projects using genetic algorithms. Journal of Systems and Software, 2011, 84(5), 775-785.

[26] T.T. Nguyen, A.W.C. Liew, M.T. Tran, T.T.T. Nguyen, M.P. Nguyen, Classifier Fusion Based On A Novel 2-Stage Model, in: X. Wang, W. Pedrycz, P. Chan, Q. He (Eds.), Machine Learning and Cybernetics, Springer, 2014, pp. 60-68.

[27] T.T. Nguyen, A.W.C. Liew, M.T. Tran, X.C. Pham, M.P. Nguyen, A novel genetic algorithm approach for simultaneous feature and classifier selection in multi classifier system, in: IEEE Congress on Evolutionary Computation (CEC), 2014, pp.1698-1705.

[28] T.T. Nguyen, A.W.C. Liew, X.C. Pham, M.P. Nguyen, A Novel 2-Stage Combining Classifier Model with Stacking and Genetic Algorithm Based Feature Selection, in: D.-S. Huang, K.- H. Jo, L. Wang (Eds.), Intelligent Computing Methodologies, Springer International Publishing, 2014, pp. 33-43.

[29] T.T. Nguyen, A.W.C. Liew, X.C. Pham, M.P. Nguyen, Optimization of ensemble classifier system based on multiple objectives genetic algorithm, International Conference on Machine Learning and Cybernetics (ICMLC), 2014 (Vol.1), pp. 46 51.

[30] T.T. Nguyen, T.T.T. Nguyen, X.C. Pham, A.W.C. Liew, A Novel Combining Classifier Method based on Variational Inference, Pattern Recognition, 2016, 49, 198-212.

[31] Takagi, Y., Mizuno, O., &Kikuno. An empirical approach to characterizing risky software projects based on logistic regression analysis. Empirical Software Engineering, 2005, 10(4), 495-515.

[32] Verner, J., Sampson, J., &Cerpa, N. What factors lead to software project failure?.In Research Challenges in Information Science, 2008.RCIS 2008. Second International Conference (IEEE) , 2008, (pp. 71-80).

[33] Wang, X., & Zhong, R. A New Weighted Ensemble Classifier Based on Granular Model. In The International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, Springer, Cham, 2020, (pp. 866-873).

[34] Wang, X., Wu, C., & Ma, L. Software project schedule variance prediction using Bayesian Network.In Advanced Management Science (ICAMS), 2010 IEEE International Conference, 2010, Vol. 2, pp. 26-30.

[35] Wu, Classifier Ensemble by Exploring Supplementary Ordering Information, IEEE Transactions on Knowledge and Data Engineering, 2018, In Press, DOI: 10.1109/TKDE.2018.2818138.

[36] X.C. Pham, M.T. Dang, S.V. Dinh, S. Hoang, T.T. Nguyen, A.W.C. Liew, Learning from Data Stream Based on Random Projection and Hoeffding Tree Classifier, in Proceeding of Digital Image Computing: Techniques and Applications (DICTA), 2017.

[37] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: Proceedings of International Conference on Machine Learning (ICML), 1996, pp. 148-156.

[38] Z. Yu , D. Wang, Z. Zhao, C.L.P. Chen, J. You, H.-S. Wong, J. Zhang, Hybrid Incremental Ensemble Learning for Noisy Real-World Data Classification, IEEE Transactions on Cybernetics, 2018, In Press, DOI: 10.1109/TCYB.2017.2774266.

[39] Z. Yu , Y. Zhang, J. You, C.L. P. Chen, H.-S. Wong, G. Han, J. Zhang, Adaptive Semi-Supervised Classifier Ensemble for High Dimensional Data Classification, IEEE Transactions on Cybernetics, 2018, In Press, DOI: 10.1109/TCYB.2017.2761908.

[40] Z. Yu, L. Li, J. Liu, G. Han, Hybrid Adaptive Classifier Ensemble, IEEE Transactions on Cybernetics, 2015, 45(2) 177 – 19.

[41] Ibraigheeth, M. A., & Fadzli, S. A. (2019). Fuzzy Logic Driven Expert System for the Assessment of Software Projects Risk. International Journal of Advanced Computer Science and Applications, 10(2).