Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI)
Vol. 11, No. 2, June 2025, pp. 276-289
ISSN: 2338-3070, DOI: 10.26555/jiteki.v11i2.30839

Enhancing Refactoring Prediction at the Method-Level Using Stacking and Boosting Models

Shahbaa I. Khaleel, Rasha Ahmed Mahmood
Department of Software, College of Computer Science and Mathematics, University of Mosul, Mosul, Iraq

ARTICLE INFO

Article history:

Received April 09, 2025
Revised May 09, 2025
Accepted May 17, 2025

Keywords:

Software Refactoring;
Method-Level Refactoring;
Meta-Learning;
Code Smell Detection;
Software Maintainability;
Swarm Optimization Algorithm;
Lion Optimization Algorithm;
Stacking Algorithm;
Boosting Algorithm

ABSTRACT

Refactoring software code is crucial for developers since it enhances code maintainability and decreases technical complexity. The existing manual approach to refactoring demonstrates restricted scalability because of its requirement for substantial human intervention and big training information. A method-level refactoring prediction technique based on meta-learning uses classifier stacking and boosting and Lion Optimization Algorithm (LOA) for feature selection. The evaluation of the proposed model used four Java open source projects namely JUnit, McMMO, MapDB, and ANTLR4 showing exceptional predictive results. The technique successfully decreased training data necessities by 30% yet generated better prediction results by 10–15% above typical models to deliver 100% accuracy and F1 scores on DTS3 and DTS4 datasets. The system decreased incorrect refactoring alert counts by 40% which lowered the amount of needed developer examination.

Corresponding Author:

Shahbaa I. Khaleel, Department of Software, College of Computer Science and Mathematics, Mosul University
Mosul, Iraq
Email: shahbaaibrkh@uomosul.edu.iq

1. INTRODUCTION

A systematic process of software development comprises user requirements analysis, design, development and testing of software to achieve efficient requirements fulfillment [1]. Through refactoring which serves as the essential practice of this process programmers improve program code quality by making internal changes that leave external functionalities intact [2].

Refactoring technologies serve a crucial part in enhancing maintenance and efficiency of code while enabling developers to eliminate duplicated code and simplify program complexity[3]. Staff undertake manual refactoring operations by exploiting their expertise leading to extended durations and elevated probability of mistakes [4]. The examination and strategic functioning of software improvement tasks has attracted research into artificial intelligence (AI) solutions for automated code analysis and decision-based procedures [5].

Machine learning which is part of artificial intelligence demonstrates successful capabilities in identifying refactoring possibilities [6]. through software metric pattern and historical data assessment. system learning capabilities through ML enable the processing of available data with automated output generation without human involvement [7].

The prediction uses different types of ML models according to numerous research studies. Real-world projects face difficulties when implementing these models because their dependent requirement for big datasets with high quality for optimal accuracy performance .Such models demonstrate reduced functionality since they lack capabilities to choose optimal aspects for subset selection and effective generalization across software programs [8].

The field of meta-learning emerged to tackle the existing issues. Through meta-learning models access knowledge gained from comparable tasks enabling them to handle fresh data circumstances with limited sample sizes [9]. This research introduces a hybrid meta-learning framework which unites stacking and

boosting classifiers with feature selection through the Lion Optimization Algorithm (LOA). The research aims to boost method-level refactoring prediction by improving feature relevance and reducing false positives and enhancing generalizability.

This research contributes to two main points: a prediction system using meta-learning techniques that combine reinforcement with stacking classifiers, and the use of a lion optimization algorithm to optimize models and efficiently select features. The research team conducted experiments on four open-source Java projects, achieving quantitative improvements in F1 scores and prediction accuracy.

2. RELATED WORKS

In 2017 Surika and Kumar applied their method to discover the most suitable areas for refactoring at the class level. A machine learning model uses Least Squares Support Vector Machine (LSSVM) as its base algorithm for operation. The model received its initial parameters along with training through seven open source software systems database. The researchers trained LS-SVM through the implementation of linear along with radial basis function (RBF) and polynomial functions. The model achieved 0.97 of accuracy through RBF while performing its evaluations. The model displayed competitive performance by both predicting refactoring and determining correct refactoring type applications to improve software code quality and efficiency levels [10].

The study team set out to create a method which detects the necessity of source code refactoring at the method level. The proposed model was built through implementation with five different open-source frameworks. Research-based analysts developed a model using ten machine learning classifiers which included logistic regression (LR), Naive Bayes (NB), random decision forest (RF), adaptive boosting (AdaBoost), artificial neural network using gradient (ANN+GD), Bayesian network (BN), radial basis function (RBF), multilayer neural network (MLP), boosting using logistic classification (LoG), and artificial neural network using Levenberg-Marquardt algorithm (ANN+LM). Each classifier underwent training through the provided dataset while evaluation happened through an accuracy performance assessment. Among the seven classifiers the AdaBoost classifier along with the ANN+GD classifier demonstrated the highest evaluation scores by reaching 98.16% accuracy and 98.17% accuracy respectively. The proposed model succeeded in effectively predicting code refactoring requirements at the method level according to study results [11].

Research studies have proven that refactoring methods deliver beneficial results for maintaining accurate programs. The introduction of machine learning methods for picking proper refactoring candidates achieved successful identification of optimal changes through algorithms established for class, method, and variable refinement. The proposed model uses logistic regression (LR) as well as naive Bayes (NB) and support regression machine (SVR) and decision tree (DT) in combination with random forest (RF) and neural network (NNA) algorithms. The training process used a substantial dataset derived from Apache, F-Droid as well as from GitHub. The evaluation of the models reached an accuracy metric which exhibited a ratio above 0.90. According to the obtained results the random forest algorithm proved most effective at predicting refactoring. The proposed model proves that machine learning algorithms achieve effective prediction of refactoring tasks and assist developers with refactoring decision-making [12].

The precision of static type information prediction in probability analysis during restructuring heavily depends on the research-developed algorithm described in Akour et al., 2022. The supports mechanism regression algorithm obtained dataset information through four open-source system sources: ANTLR4, JUnit, MapDB and McMMO. Performance assessment of the algorithm together with training occurred using these specific data points. The implementation of SVM algorithm began independently before researchers integrated it with optimization algorithms such as GA and WA to enhance output. The evaluation depended on accuracy as the main performance metric. The SVR algorithm achieved 0.90 accuracy on MapDB and McMMO dataset while reaching 0.93 accuracy through the SVM and GA and WA algorithm combination. The proposed model was subjected to performance testing alongside Naive Bayes while a comparison took place with the state-based nearest neighbor and random decision tree and random decision forest methods. The research findings suggest that using SVM in combination with GA and WA brings significant improvements to prediction accuracy in restructuring operations above traditional predictive models [13].

3. SOFTWARE RESTRUCTURING

The structural changes conducted in a program through refactoring result in increased efficiency and maintainability without modifying how the program operates externally.The reason for refactoring is to address technical debt resulting from the accumulation of modifications and shortcuts during the development process. The refactoring process involves several steps that are followed in the code to implement the required modifications and improve the code [14].

The following are the stages of the refactoring process [15]:

Selecting the code that must be refactored: The evaluation process searches for code sections needing modification in order to enhance code maintainability.
Determining appropriate refactoring techniques: Restructuring involves many types, and the appropriate type is determined based on the type of problem identified.
The refactoring techniques applied must maintain the program's behavior: The main purpose of using refactoring techniques involves improving code quality levels with no negative effects on original functionality.
Applying refactoring techniques: The modifications are implemented using appropriate restructuring methods.
Enhancing the assessment process rallies to determine both internal and external software quality effects of refactoring Methods: The impact of changes on code quality is measured using software metrics.
Keeping the integrity between the software that was refactored and other software parts: After modifying the code, it must not conflict with other modules in the system. Fig. 1 The following figure shows a flowchart of the stages of the software refactoring process.

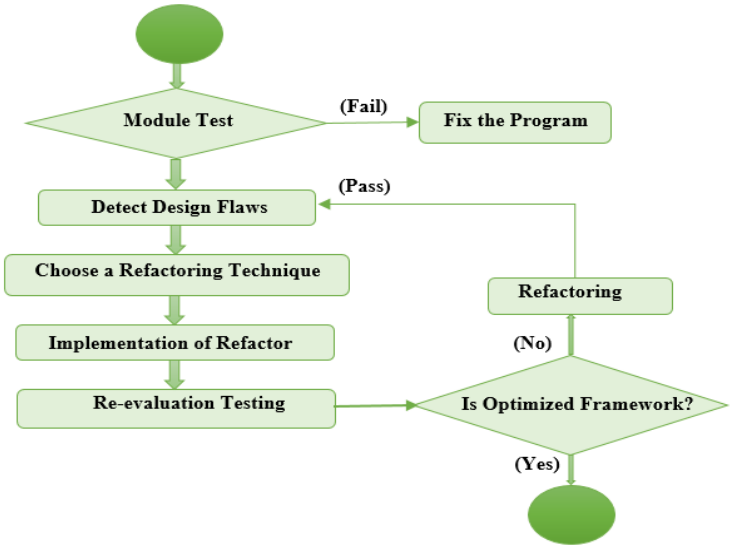

Fig. 1. The flowchart of Refactoring Process

4. METHODOLOGY OF THE PROPOSED MODEL

The proposed methodology introduces a machine learning framework designed to enhance prediction of refactoring opportunities at the curriculum level. This framework combines two ensemble learning strategies—stacking and boosting—with a feature selection mechanism based on the Lion optimization algorithm (LOA) applied to a dataset to extract features that significantly impact model accuracy. This approach selection basis stems from the unique beneficial features that each component brings to the model. Sequential prediction error correction along with enhanced accuracy stands as a known strength of AdaBoost and CatBoost and LightGBM while stacking uses a meta-learner to integrate multiple base classifiers including logistic regression support vector machines and random forests to enhance generalization. The model becomes more efficient at detecting linear and nonlinear data patterns because both model types are included in its design. The Lion optimization algorithm selects influential features for use by conducting feature selection. Analysis professionals selected this approach because it showed excellence in the management of intricate feature domains which standard methods often struggle to handle. The data processing starts with cleansing of primary data obtained from four open source Java projects JUnit, McMMO, MapDB and ANTLR4. The data cleaning process follows standardization before a LOA algorithm selects features with the greatest relevance. Feature selection concludes followed by training-test set separation of the data. The selected features get utilized for training both stacking and boosting models before their prediction outputs combine to generate end results. Each model uses a grid search optimization process to determine the best values for tree numbers combined with learning rate along with regularization parameter adjustment. The main aim is to enhance performance while minimizing overfitting occurrences. A set of standard performance criteria that includes precision, accuracy, recall, and F1 score serves in determining the effectiveness of the proposed framework through final evaluation. Fig. 2 illustrates flowchart of the stages of the proposed approach.

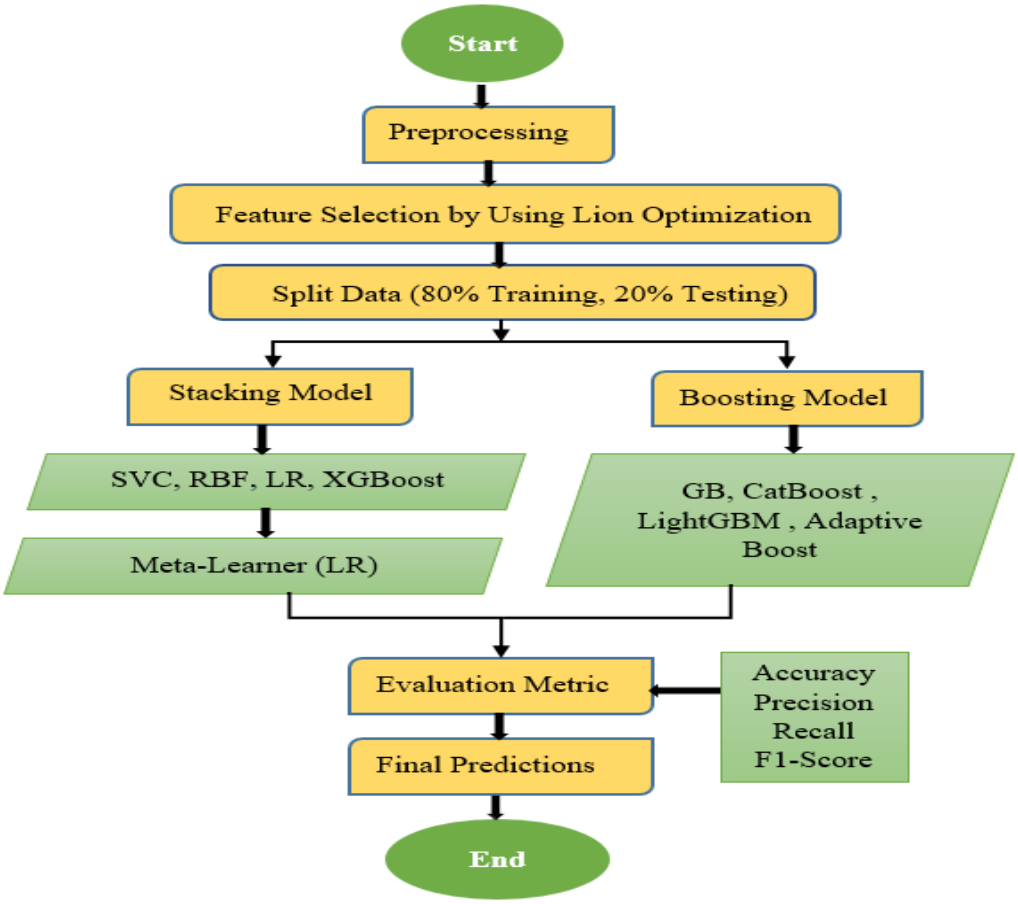

Fig. 2. The flowchart of Proposed Approach

5. DATASET AND PREPROCESSING

The data collection stage is a crucial stage. dataset used in this research includes data related to software projects. This dataset includes experimental refactorings of four open source software projects: JUnit, McMMO, MapDB, and ANTLR4. Named DTS1, DTS2, DTS3, and DTS4, respectively, it includes features representing software characteristics and statistics that determine the likelihood of system components requiring refactoring.

The data was preprocessed to ensure that the data is ready, of good quality, and achieves a balanced distribution of classes. This dataset is then used to train models, which contributes to improving the performance of models used to predict refactoring. Unnecessary columns, such as Long-Name, Parent, and Path, were removed to reduce complexity and noise in the models, as they do not contribute to the refactoring prediction process.

The focus was solely on features that support the prediction process. The target values are converted to binary values of zero or one in order to help the model in binary classification, This value of 1 signifies class restructuring necessity whereas a value of zero indicates no need for restructuring making data processing easier for the model. Table 1 shows description of the data set.

Table 1. Description of the Data Set

Data Set	No. oF Attributes	Instances	No. oF Refactoring
DTS1	84	3,298	40
DTS2	84	2,280	12
DTS3	84	3,501	3
DTS4	84	2,531	5

6. FEATURES SELECTION

Features represent software metrics, and since they influence each other, incorrectly selecting metrics can lead to poor predictive models [16]. Feature selection increases the efficiency of the model and helps enhance its ability to classify effectively. Four traditional approaches exist for feature selection consisting of filter

methods and wrapper methods and embedded methods and hybrid methods. Although these methods are effective, when dealing with large and complex datasets, they can be inefficient and produce unsatisfactory results. Therefore, swarm intelligence (SWI) has been used as an effective approach and an advanced alternative to traditional methods [17]. The foundation of swarm intelligence uses limited capability and simple behaviors among individuals like the ants to complete complicated tasks via their coordinated social methods even though their actions are basic [18] Swarm intelligence algorithms identify optimal problems solutions through simulations of collective natural organism behaviors such as ant colonies and bird flocks for complex task improvement [19].

Swarm intelligence functions through fundamental principles including the proximity principle for local interactions with environments and the information quality principle for efficient solutions together with the diverse response principle for change adaptiveness and the stability principle for maintaining equilibrium and the adaptive principle for different environment responses [20]. Swarm members communicate to each other by direct methods like visual signals or auditory messages such as the bee dance in addition to indirect methods such as environmental changes through pheromone use by ants. The behaviour of different agents is affected by environmental changes through communication across the environment [21].

6.1. Lion Optimization Algorithm

The Lion's algorithm was introduced as an optimization algorithm by Rajakumar, after which improvements were made to the algorithm and many algorithms were developed based on the basic principles of the Lion's algorithm [22]. It represents an improved model derived from nature, mimicking the behavior of lions. This algorithm is mainly used to address optimization tasks by using two focused steps: territorial capture with territorial defense. Lions are characterized by a wonderful social behavior called pride, where 1-3 pairs of lions are included within the pride, meaning it will include adult males, a number of females, and cubs. Each pride has a territory called the area of influence, and the strongest lion is the one who controls the territory and is called the territorial lion. It must be defended against nomadic or attacking lions, called the nomadic lions. This defense is called territorial defense. When the territorial lion wins, the territory remains as it is. However, when the nomadic lion wins, it will chase or kill the territorial lion. The cubs of the defeated lion are also killed, and the dominant lions associate with the females, and new offspring are formed. These processes are repeated until the cubs reach adulthood. If the cubs are stronger, they will take over the territorial lion and associate with the females. New cubs are born and territorial conquest occurs [23]. The algorithm makes use of complex cluster behaviors which interact with one another. The approach effectively balances discovery with optimization because it works with many different types of optimization problems [24].
The following is an explanation of the steps of the LOA algorithm [25]:
Step 1: Initialization: Initially, a random population is created in the solution space, where each lion represents a solution to an $N_v$optimization problem, where the lion is L= x_1 , x_2 $\ldots x_{N_v}$

$$\mathrm{L} = x_1 \tag{1}$$

The cost (fitness value per lion) is calculated by evaluating the cost function shown as follows:

$$Fitness\ value\ of\ lion = f\ (L) = f(x_1\ , x_2\ , x_3\ \ldots\ldots x_{N_v}\ ) \tag{2}$$

M_{pop}solutions are generated randomly in the Exploration range. M% of the solutions (lions), Select it randomly and are called M. The remaining solutions are divided into groups p. For each solution in the LOA, a specific sex remains constant While optimizing. In each pride, approximately 75-90% of the output from the previous stage is identified as female lions, At the time when you are remaining lions are males. During the search process, Lion detects the best place based on the selected places. For each group, a region p is created by the locations selected by its members.
Step 2: Hunting: for each p, a certain number of female lions search for victim for feeding the individuals in the p. Complex methods are used to monitor and capture their victim. Stander divided lions into several classes based on the role they play. During a hunt, each lioness adjusts her position based on her own or other individuals' position. As a result, Some hunting lions surround the victim and launch an attack from the opposite direction. The law of attraction is based on conflict-based learning. H are divided classes. The middle class includes the highest degree of fitness, and the left side and right side are randomly assigned to each class without prior determination. During a hunt, as the hunter's fitness increases, the victim flees to a new place, and represented as follows:

$$PY' = PYI + rand(0,1) \times PY \times (PY - H) \tag{3}$$

where PY refers to the current place of the victim, H refers to the new place of the hunter attacking the victim, PYI refers to percentage improvement In the ability hunter.
The location belonging to the left side and the right side is calculated as shown in the following equation:

$$H' = \begin{cases} \text{rand}((2 \times \text{PY} - \text{H}), \text{PY}), (2 \times \text{PY} - \text{H}) < \text{PY} \\ \text{rand}(\text{P}, (2 \times \text{PY} - \text{H})), ((2 \times \text{PY} - \text{H}) > \text{PY} \end{cases} \quad (4)$$

New sites for concentrated hunters are created according to the following equation:

$$H' = \begin{cases} \text{rand}(\text{H}, \text{PY}), \text{H} < \text{PY} \\ \text{rand}(\text{PY}, \text{H}), \text{H} > \text{PY} \end{cases} \quad (5)$$

The function rand() generates numbers by random within a specific range.
Step 3: Moving to a Safe Place: Each group's p-zone contains the best personal positions acquired by each group member. This helps the Law of Attraction store the best solutions previously acquired over a period of time, which can be useful for improving solutions according to the Law of Attraction. Therefore, the new position of the female lion is determined according to the following equation:

$$\text{FL}' = \text{FL} + 2\text{C} \times \text{rand}(0,1)\{\text{K1}\} + \text{U}(-1,1) \times \tan(\theta) \times \text{rand}(0,1) \times \{\text{K}\} \quad (6)$$

$$\{\text{K1}\}.\{\text{K2}\}=0, \| \text{K2} \|=1 \quad (7)$$

where FL is current location represents the lioness, C indicates the area between the location of the FL and the point determined by championship in the pride area, K1 represents a direction whose starting point is the female lion's oldest location and directed toward the select place. K2 is vertical to K1.
Step 4: Roaming: Every lioness roams the pride space. resembles the natural model, a lion selects a random portion of the pride's territory, K. When a male identifies a previously unexplored site that is better than the one he currently occupies, he updates the solution he previously visited to a better one. Machine learning (ML) uses the law of attraction (LOA) to randomly distribute exploration areas and avoid the trap of local optima. New sites are selected for ML according to the following equation:

$$L'_{ij} = \begin{cases} pr_i \text{ if } > \ Lion_{ij} \\ RAND_i \quad \text{other wise} \end{cases} \quad (8)$$

where $Lion_i$represents the current position of the lion, j is a dimension, $rand_j$ represents Irregular number within [1,0], RAND represents a randomly generated direction in the Exploration area, pr_i represents a possibility calculated independently for each lion ML, as shown in the following equation:

$$pr_i = 0.1 + \min(0.5, \frac{ML_i - Best_{ML}}{Best_{ML}}) \quad (9)$$

where ML_i, $Best_{ML}$represent the temporal location costs lion in LN.
Step Five: Bonding: This process ensures the survival of lions and provides an opportunity for them to bond with other members. In each pride, a percentage of the female lionesses bond with a male lion or several randomly selected resident males from the same pride to produce offspring. The law of attraction transfers information between the sexes during bonding, and new cubs then inherit the personality of both sexes.
Step 6: Defense: In each p, when a lioness reaches completion, he becomes aggressive and battles with other lioness in the p. The lion that loses the fight give up the p and becomes ML. Similarly, ML male is has great physical strengthWhen a DL male fights in p, the ML becomes DL, and vice versa. Defense plays a role in maintaining the strength of male lions.
Step 7: Migration: Depending on the movement and migration that from one area to another, this will aim to enhance pride security. Female lions from each area P are counted, and then a small number of lionesses are randomly selected to form ML. Female Lions are ranked based on their fitness value.
Step Eight: Termination: The stopping criterion represents a condition for determining when the algorithm stops searching for solutions. The stopping criterion may be a specific period of time for the algorithm to stop, or the algorithm stops after a specific number of iterations, or it stops when no progress is made in a specific number of iterations. Fig. 3 shows the flowchart for implementing the steps of the LOA algorithm.

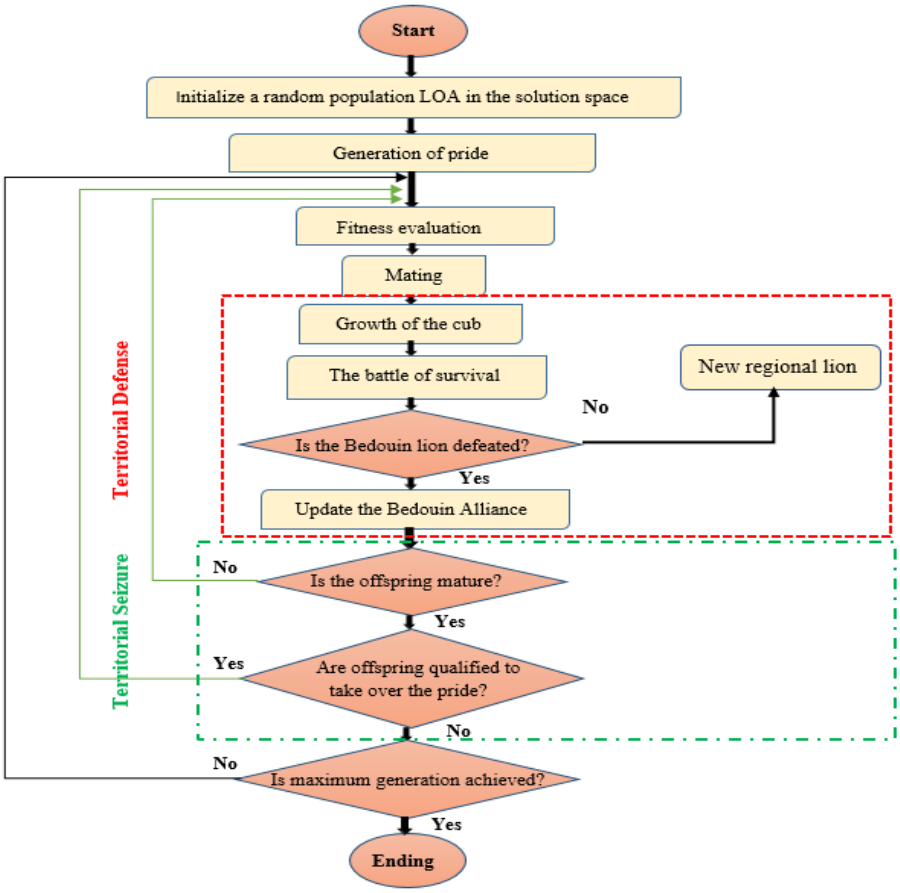

Fig. 3. Research Methodology

7. DATA SET SPILT

At this stage, the data will be split into two class: the training class and the test class. The training class is used to train the proposed model, the test class is used to evaluate the proposed model. According to previous machine learning research, the split ratio was 80% for training data set and 20% for test data set. The appropriate ratio is determined based on the problem or available dataset. Here, the split was made so that the training data ratio was 80% and the test data ratio was 20%. This split ratio was chosen because it yielded better results and was appropriate for the dataset.

8. MODEL TRAINING

At this stage, machine learning algorithms are trained to predict refactoring probabilities. A stacking classifier and a boosting classifier are used.

First: The stacking model: This is a commonly used technique in meta-learning, introduced by Wolpert . Machine learning problems benefit from this method because it decreases the occurrence of errors during generalization. This technique represents an ensemble framework, and is considered a heterogeneous method because it relies on training base models and then combining their predictions. A meta-model is then trained on the base models' prediction outputs. Stacking models achieve better performance because they exploit the variance and differences between base models [26]. The stacking approach requires base classifiers or zero-level models that work together with a higher-level learning algorithm named meta-learner. The first-level model receives predictions from base models to develop a combination strategy which enhances the predictive power of the outcomes. The training data serves to prepare the base models before the system tests against new data that remained out of initial training. A new data dataset is created by using the output predictions and actual values from the data which gets used to train the higher-level model also known as the first-level classifier [27].The model structure consists of two steps [28]:

The first phase included four basic models, which were implemented and are known as basal learners:

Logistic Regression (LR): It is one of the supervised learning techniques, It is primarily used in the field of classification, where the goal is to predict the probability of a value or instance belonging to a specific class or not [29] Logistic regression operates as an effective mode to forecast chronic disease risk but shows

performance levels similar to those of contemporary machine learning systems. Scientific research panels confirm that logistic regression stands together with neural networks as leading approaches for making such predictions [30].The following is an explanation of the Logistic Regression equation [31]:

$$H(x) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p \quad (10)$$

where $\beta_0$represents the constant coefficient, $\beta_1$represents the weights that reflect the effect of the independent variables on the model's prediction.

Extreme Gradient Boosting (XGBoost): is considered one of the clustering algorithms that relies on decision trees and uses the gradient boosting method. XGBoost relies on the principle of decision trees to represent potential decisions. Performance is improved by reducing errors using gradients. It also works to build sequential models to reduce errors. The system is improved by applying parallel processing and removing unnecessary branches from the trees to enhance efficiency, so it is considered one of the best options for classification and regression problems [32]. The objective function, which includes obtaining better predictions, works by minimizing the loss function. The objective function includes two parts: an error function, which captures the difference between the actual values and the previously predicted values, and a regularization function, which controls the difficulty of the trees to avoid excessive complexity. The following equation provides this [33]:

$$\mathrm{Oj(k)=M(k)+U(k)} \quad (11)$$

where k represents the parameter that is modified during the training phase. M represents the loss function, and U represents the regularization function. XGBoost demonstrates excellent performance across financial services and healthcare together with supply chain management because of its successful implementation in different industries [34].

Redial Basis Function (RBF): The artificial neural network architecture is execute to regression and classification Tasks. It is based on transforming input data into a multidimensional space using complex or nonlinear functions (radial basis functions), followed by a linear transformation to obtain the expected final output. The network architecture consists: the input layer, the intermediate layer, and the output layer. The inputs pass from the input layer to the intermediate layer, which contains radial functions, where the data is processed. The output layer is then applied to predict the final output [35]. RBFNNs operate as neural networks based on function approximation principles to predict nonlinear models between inputs and outputs within modeling systems [36].

Support Vector Classifier (SVC): It is considered one of the support vector machine methods. This method is distinguished by its reliance on computational learning theory and its invariance in multiple domains, so classification can be performed on different features and with different sizes of training data [37]. The work of this algorithm revolves around identifying the best hyperplane, which is defined as the dividing line between data classes in a high-dimensional space. The hyperplane includes support vectors that represent data points that are adjacent to the hyperplane, while the margin represents the distance between the hyperplane and the support vectors. Increasing the margin ratio leads to reducing classification errors [38]. The algorithm applies to non-linear complex data (using a kernel) during classification problems with two or multiple outcome categories. Multilayer perceptron becomes a suitable choice for creating accurate classification models which show resistance towards over-learning [39].

In the second stage, the Logistic Regression (LR) algorithm was applied, which is known as the super-learner, which is trained on the results of the base learners in order to obtain the final classification.

Second: Boosting model: It is a sequential iterative ensemble method that works to reduce bias and variance by improving the performance of weak models. A series of weak models are created in such a way that each new model learns from the previous model's mistakes. The goal of this technique is to train several weak models in order to obtain a strong model. This technique helps improve accuracy and reduce errors. This model focuses on selecting misclassified data to train models on. That is, the first model will be trained on training data, the second model will be trained to correct the errors resulting from the first model. Therefore, it is considered an iterative algorithm, as the weights are adjusted based on the previous model [40]. The successive operation of the boosting algorithm starts with selecting an initial classifier to predict data from the entire dataset before moving on to error evaluation and weight adjustment. The algorithm determines errors of the first classifier after which misclassified items receive weighted increases for enhanced prioritization in future stages. A new classifier receives training which targets the mistakes from the preceding model for the next stage. A sequential pattern applies multiple times until performance reaches its maximum level. The building of the final model takes place by developing an average weight-based combination of all previous classifiers. Model performance accuracy in the final result becomes enhanced by giving more weight to the models that

demonstrate the best results [41]. The boosting model includes prominent algorithms that were applied in this research:

Gradient Boosting (GB): It is considered one of the most effective machine learning techniques frameworks based on clustering algorithms and works on classification and regression tasks [42]. based on combining gradient descent with boosting techniques, and a strong model is formed by gradually adding weak learners [43]. Techniques that use a gradient algorithm build their models incrementally and make continual improvements by fixing the mistakes of existing models. The creation of Gradient Boosting emerged from adaptive boosting theory generalization and today shows capabilities of processing various loss functions effectively [44].

Adaptive Boosting (AdaBoosting): is one of the most widely used boosting techniques, resulting in more effective predictive models. This algorithm uses the entire dataset to train the classifier rather than randomly selecting samples, which makes it more efficient when data is limited. This algorithm also gives greater weight to misclassified samples, so the next model will focus on correcting these errors. In contrast, samples that were correctly classified will receive less weight [45]. The training process of AdaBoost algorithm involves continuous weight changes across samples but needs no understanding of the weak classifier learning process. AdaBoost mainly operates as a solution for classification tasks yet maintains a reputation for being simple to implement as well as quick. The main advantage of this algorithm consists of small configuration needs except for setting the iteration number. Any weak classification algorithm can integrate with the AdaBoost method since it needs no specific understanding of the algorithm's operational principles [46].

Light Gradient Boosting Machine (LGBM): A new approach to supporting gradients based on decision tree methods. This framework supports a set of algorithms such as GBM, Gradient Boosted Regression Tree (GBRT), and Multiple Regression Tree (MART). It is therefore scalable, robust, and efficient [47]. The execution time together with prediction accuracy of GBDT deteriorates significantly when processing vast amounts of data. LightGBM solves the speed and accuracy issues by providing enhanced training performance that neither compromises prediction accuracy nor slows down the process. Continuous data values in LightGBM are transformed into histogram-based bins which decrease runtime complexity. The leaf-wise structure enables the most error-minimizing leaf to grow the most as opposed to level-wise expansion across all tree branches. The Exclusive Feature Bundling method combines sparse features to decrease their total number while averting the need for PCA techniques and minimizing data redundance [48].

Categorical Boosting (CatBoost): is an advanced high-performance automated algorithm belonging to the Gradient Boosting family. It creates an ensemble of weak decision trees to gradually improve classification performance. These trees are then combined iteratively, with residual values from the previous step used to improve the model at each step [49]. Artificial scheduling functions as the core principle of this method through data partitioning during training to develop successive tree models without fresh data input. Data leakage avoidance becomes possible through this approach because it integrates available information through an innovative phased procedure. This algorithm uses symmetric trees to produce identical splitting operations which improve the speedy training process and avoids traditional regression algorithms that need entire data sets to train trees for target value prediction [50].

9. MODEL EVALUATION

There are several metrics used to evaluate the proposed model, depending on the problem. Some are appropriate for classification problems, while others are appropriate for regression. Here, we used metrics commonly used to predict refactoring: accuracy, recall, precision, and F1-score. Results were evaluated for the dataset used at the method level.

10. DISCUSSION AND RESULTS

The proposed approach was evaluated using four datasets. The proposed model classifiers were implemented using Python. The Lion's Optimization algorithm was used for feature selection and applied to the four datasets. Table 2 and Table 3 the prediction performance results for the dataset are summarized using evaluation metrics: accuracy, positive precision, F-measures, and recall.

In this research, the efficiency of applying Lion's Optimization algorithm with stacking and boosting classifier is tested in terms of the performance efficiency of predicting optimal restructuring opportunities at the method level.

In this research, the prediction efficiency is estimated through evaluation metrics. Table (2) presents the results of the stacking and boosting classifier on the four datasets. The stacking and boosting classifier achieved an accuracy of 0.99 on the DTS3 dataset, achieving the highest prediction accuracy. However, there were discrepancies between the recall, positive precision, and F1 metrics, and the results were unbalanced for some stacking and boosting methods.

Table 2. Results of the proposed model for the four data sets

DataSet	Model	Algorithm	Accuracy	Precision	F1-Score
DTS1	Stack	SVC	0.97	0.60	0.24
		RBF	0.97	0.60	0.24
		LR	0.96	0.25	0.08
		XGBoost	0.98	0.85	0.70
		Stack	0.98	0.81	0.72
	Boosting	GB	0.98	0.80	0.80
		CatBoost	0.98	0.93	0.80
		LightGBM	0.98	0.92	0.72
		AdapBoost	0.97	0.50	0.16
DTS2	Stack	SVC	0.98	0.91	0.65
		RBF	0.97	0.72	0.68
		LR	0.95	1.00	0.09
		XGBoost	0.98	0.79	0.86
		Stack	0.98	0.88	0.80
	Boosting	GB	0.97	0.62	0.76
		CatBoost	0.98	0.71	0.83
		LightGBM	0.97	1.00	0.33
		AdapBoost	0.95	1.00	0.09
DTS3	Stack	SVC	0.99	1.00	0.85
		RBF	0.99	0.80	0.88
		LR	0.98	0.71	0.73
		XGBoost	0.99	0.95	0.97
		Stack	0.99	0.95	0.97
	Boosting	GB	0.99	0.94	0.86
		CatBoost	0.99	1.00	0.85
		LightGBM	0.99	0.95	0.97
		AdapBoost	0.99	0.90	0.95
DTS4	Stack	SVC	0.99	0.86	0.90
		RBF	0.99	0.83	0.90
		LR	0.98	0.88	0.81
		XGBoost	0.99	0.86	0.93
		Stack	0.99	1.00	0.97
	Boosting	GB	0.99	1.00	0.88
		CatBoost	0.97	1.00	0.46
		LightGBM	0.98	1.00	0.82
		AdapBoost	0.98	1.00	0.70

Table 3 shows that the highest accuracy was achieved, with the stacking and boosting classifier with the LOA algorithm reaching 1.00. Furthermore, there were improvements in the evaluation metrics for the stacking and boosting classification methods, with the positive precision metric achieving 1.00 for the DTS1-AdapBoost model and the DTS2-LR model, while the stacking and boosting classifier achieved 1.00 on the DTS4 and DTS3 datasets.While the recall measure achieved 0.90 for the XGBoost model and boosting classifier on DTS1, 0.95 for the XGBoost model and boosting classifier on DTS2, and 1.00 for the stacking and boosting classifier on DTS4 and DTS3 datasets. While the F1 criterion achieved the highest ratio of 0.90 for the XGBoost model and 0.94 for the AdapBoost model for the DTS1 dataset, it achieved 0.93 for the SVC model and 0.90 for the CatBoost+XGBoost model for the DTS2 dataset, and the highest ratio was 1.00 for the stacking classifier and boosting classifier for the DTS4 and DTS3 datasets. The results show that combining the stacking classifier and boosting classifier with the LOA optimization algorithm improved prediction and achieved the highest ratio of 1.00 for the DTS4 and DTS3 datasets.

The best prediction results emerged from DTS3 and DTS4 datasets because the model scored 100% accuracy and F1 score. The model exactly forecasted all actual refactoring requirements leading to high generalizability levels when the datasets maintain perfect balance and representativeness.

Nevertheless, the DTS1 and DTS2 data results generated noticeable mismatches between precision rates and recall measures. The AdaBoost model obtained 1.00 precision from DTS1 while achieving 0.90 recall while logistic regression exhibited matching behavior when evaluating DTS2. The differences between predicted and actual outcomes reveal that particular models function cautiously to decrease wrong positives yet face more wrong negatives. The identified pattern suggests that the methodology learned data distributions specifically from the dominating samples. Related studies based on feature selection methods including genetic algorithm (GA), whale algorithm (WA) and SVM algorithm which united GA and WA for performance enhancement

yielded maximum accuracy of 0.93. The proposed model outperformed baseline values by reaching 1.00 along with other metrics which exceeded previous results in some cases. This data enhancement becomes possible because of the combinational power between ensemble learning and the law of attraction. However, some limitations remain. The whale algorithm increases computational expenses particularly during its application to large datasets.

Table 3. Results of the proposed model for the four data sets using the lion optimization algorithm.

DataSet + LOA	Model	Algorithm	Accuracy	Precision	Recall	F1-Score
DTS1	Stack	SVC	0.99	0.94	0.75	0.83
		RBF	0.97	0.58	0.35	0.43
		LR	0.96	0.25	0.05	0.08
		XGBoost	0.99	0.90	0.90	0.90
		Stack	0.99	0.93	0.75	0.83
	Boosting	GB	0.99	0.94	0.90	0.92
		CatBoost	0.99	0.94	0.90	0.92
		LightGBM	0.99	0.94	0.90	0.92
		AdapBoost	0.99	1.00	0.90	0.94
DTS2	Stack	SVC	0.99	0.90	0.95	0.93
		RBF	0.98	0.75	0.90	0.81
		LR	0.95	1.00	0.05	0.09
		XGBoost	0.98	0.79	0.95	0.86
		Stack	0.99	0.95	0.90	0.92
	Boosting	GB	0.98	0.82	0.95	0.88
		CatBoost	0.99	0.86	0.95	0.90
		LightGBM	0.98	0.82	0.95	0.88
		AdapBoost	0.99	0.86	0.95	0.90
DTS3	Stack	SVC	1.00	0.95	1.00	0.98
		RBF	0.99	0.83	1.00	0.90
		LR	0.98	0.75	0.75	0.75
		XGBoost	1.00	1.00	1.00	1.00
		Stack	1.00	1.00	1.00	1.00
	Boosting	GB	1.00	1.00	1.00	1.00
		CatBoost	1.00	1.00	1.00	1.00
		LightGBM	1.00	1.00	1.00	1.00
		AdapBoost	1.00	1.00	1.00	1.00
DTS4	Stack	SVC	1.00	1.00	1.00	1.00
		RBF	0.98	0.93	0.70	0.80
		LR	0.98	0.93	0.75	0.83
		XGBoost	1.00	1.00	1.00	1.00
		Stack	1.00	1.00	1.00	1.00
	Boosting	GB	1.00	1.00	1.00	1.00
		CatBoost	1.00	1.00	1.00	1.00
		LightGBM	0.99	0.95	1.00	0.97
		AdapBoost	1.00	1.00	1.00	1.00

11. CONCLUSION

A combination of stacking and boosting approaches and Lion Optimization Algorithm selection methods forms a research design to enhance software prediction of refactoring. The model demonstrated 100% accuracy and F1-score in particular data tests (DTS3 and DTS4) through evaluations on four open-source Java projects.The research results demonstrated that the Meta learning model combined with intelligent feature selection will enhance the accuracy of restructuring prediction and reduce development time by aligning meta-learning with swarm intelligence within a flexible processing framework that enhances theoretical understanding.The widespread adoption of this system faces obstacles due to computational expenses as well as subpar result quality in datasets that have diverse distribution or contain ambiguous data.Future research will resolve these complexities and dilemmas by implementing transfer learning and graphical neural networks (GNNs) as next-generation methods to enhance generalization and better understand code structure.By evaluating this research the essential components of intelligent refactoring tools are determined which makes it possible to develop automated software maintenance procedures.

Acknowledgments

Authors would like to thank the University of Mosul in Iraq for providing Moral support.

REFERENCES

[1] N. Moaid Edan, S. Mahmood Hadeed, and S. A. Mahmood, "Using Software Engineering to Design and Implement Mixer of Multiple Cameras, Microphone, and Screens*." International Journal of Applied Sciences and Technology*, vol. 4, no. 2, 2022, https://www.minarjournal.com/dergi/using-software-engineering-to-design-and-implement-mixer-of-multiple-cameras-microphone-and-screens20220702024845.pdf.

[2] G. A. Armijo and V. V De Camargo, "Refactoring Recommendations with Machine Learning," *Simpósio Brasileiro de Qualidade de Software (SBQS)*, pp. 15-22, 2022, https://doi.org/10.5753/sbqs_estendido.2022.227650.

[3] S. I. Khaleel and G. K. Al-Khatouni, "A literature review for measuring maintainability of code clone," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 31, no. 2, pp. 1118–1127, Aug. 2023, https://doi.org/10.11591/ijeecs.v31.i2.pp1118-1127.

[4] D. Dig *et al.*, "Refactoring tools," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 193–202, 2008, https://doi.org/10.1007/978-3-540-78195-0_19.

[5] S. I. Khaleel and R. Anan, "A review paper: optimal test cases for regression testing using artificial intelligent techniques," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 2, pp. 1803–1816, Apr. 2023, https://doi.org/10.11591/ijece.v13i2.pp1803-1816.

[6] M. mzeri and L. Ibrahim, "Detecting A Medical Mask During The COVID-19 Pandemic Using Machine Learning: A Review Study," *Journal of Education and Science*, vol. 31, no. 2, pp. 55–68, Jun. 2022, https://doi.org/10.33899/edusj.2022.133181.1221.

[7] A. Ali and N. Nimat Saleem, "Classification of Software Systems attributes based on quality factors using linguistic knowledge and machine learning: A review," *Journal of Education and Science*, vol. 31, no. 3, pp. 66–90, Sep. 2022, https://doi.org/10.33899/edusj.2022.134024.1245.

[8] S. I. Khaleel and L. F. Salih, "A survey of predicting software reliability using machine learning methods," *IAES International Journal of Artificial Intelligence*, vol. 13, no. 1, pp. 35–44, Mar. 2024, https://doi.org/10.11591/ijai.v13.i1.pp35-44.

[9] H. Khosravi and A. Rasoolzadegan, "A Meta-Learning Approach for Software Refactoring," *arXiv preprint arXiv:2301.08061*, 2023, https://doi.org/10.48550/arXiv.2301.08061.

[10] L. Kumar and A. Sureka, "Application of LSSVM and SMOTE on Seven OpenSource Projects for Predicting Refactoring at Class Level," in *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, pp. 90–99, 2017, https://doi.org/10.1109/APSEC.2017.15.

[11] L. Kumar, S. M. Satapathy, and L. B. Murthy, "Method level refactoring prediction on five open source Java projects using machine learning techniques," in *ACM International Conference Proceeding Series*, pp. 1-10, 2019. https://doi.org/10.1145/3299771.3299777.

[12] M. Aniche, E. Maziero, R. Durelli, and V. Durelli, "The Effectiveness of Supervised Machine Learning Algorithms in Predicting Software Refactoring," *IEEE Transactions on Software Engineering*, vol. 48, no. 4, pp. 1432-1450, Jan. 2020, [Online]. Available: http://arxiv.org/abs/2001.03338.

[13] M. Akour, M. Alenezi, and H. Alsghaier, "Software Refactoring Prediction Using SVM and Optimization Algorithms," *Processes*, vol. 10, no. 8, Aug. 2022, https://doi.org/10.3390/pr10081611.

[14] E. Zabardast, J. Gonzalez-Huerta, and D. Smite, "Refactoring, Bug Fixing, and New Development Effect on Technical Debt: An Industrial Case Study," in *Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA*., pp. 376–384, Aug. 2020, https://doi.org/10.1109/SEAA51224.2020.00068.

[15] S. I. Khaleel and R. A. Mahmood, "Automatic Software Refactoring to Enhance Quality: A Review," *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI)*, vol. 10, no. 4, pp. 734–746, 2024, https://doi.org/10.26555/jiteki.v10i4.30277.

[16] M. M. A. Dabdawb and B. Mahmood, "On the relations among object-oriented software metrics: A network-based approach," *International Journal of Computing and Digital Systems*, vol. 10, no. 1, pp. 901–915, 2021, https://doi.org/10.12785/ijcds/100182.

[17] M. Rostami, K. Berahmand, E. Nasiri, and S. Forouzande, "Review of swarm intelligence-based feature selection methods," *Engineering Applications of Artificial Intelligence*, vol. 100, p. 104210, 2021, https://doi.org/10.1016/j.engappai.2021.104210.

[18] X. Li, M. Clerc, "Swarm intelligence," In *Handbook of Metaheuristics*, *Springer International Publishing,* pp. 353-384, 2018, https://doi.org/10.1007/978-3-319-91086-4_11.

[19] B. Khaleel, "A Review Of Clustering Methods Based on Artificial Intelligent Techniques," *Journal Of Education And Science*, vol. 31, no. 2, pp. 69–82, Jun. 2022, https://doi.org/10.33899/edusj.2022.133092.1218.

[20] L. Brezočnik, I. Fister, and V. Podgorelec, "Swarm intelligence algorithms for feature selection: A review," *Applied Sciences*, vol. 8, no. 9, p.1521, 2018, https://doi.org/10.3390/app8091521.

[21] H. Ahmed and J. Glasgow, "Swarm Intelligence: Concepts, Models and Applications," 2012, https://research.cs.queensu.ca/TechReports/Reports/2012-585.pdf.

[22] S. M. Almufti, "Lion algorithm: Overview, modifications and applications E I N F O," *International Research Journal of Science*, vol. 2, no. 2, pp. 176–186, 2022, https://doi.org/10.5281/zenodo.6973555.

[23] M. Khosravy, N. Gupta, N. Patel, and T. Senjyu, "Frontier Applications of Nature Inspired Computation," *Springer,* 2020, [Online]. Available: http://www.springer.com/series/16134.

[24] B. R. Rajakumar, "The Lion's Algorithm: A New Nature-Inspired Search Algorithm," *Procedia Technology*, vol. 6, pp. 126–135, 2012, https://doi.org/10.1016/j.protcy.2012.10.016.
[25] K. Geetha, V. Anitha, M. Elhoseny, S. Kathiresan, P. Shamsolmoali, and M. M. Selim, "An evolutionary lion optimization algorithm-based image compression technique for biomedical applications," in *Expert Systems*, vol. 38, no. 1, p. e12508, 2021, https://doi.org/10.1111/exsy.12508.
[26] Y. Zhang, J. Liu, and W. Shen, "A Review of Ensemble Learning Algorithms Used in Remote Sensing Applications," *Applied Sciences,* vol. 12, no. 17, p. 8654*,* 2022, https://doi.org/10.3390/app12178654.
[27] I. D. Mienye and Y. Sun, "A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects," *IEEE Access*, vol. 10, pp. 99129-99149, 2022, https://doi.org/10.1109/ACCESS.2022.3207287.
[28] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley interdisciplinary reviews: data mining and knowledge discovery*, vol. 8, no. 4, p. e1249, 2018, https://doi.org/10.1002/widm.1249.
[29] J. Daniel and J. H. Martin, "Speech and Language Processing," *Power Point Slides*, 2024, https://web.stanford.edu/~jurafsky/slp3/.
[30] S. Nusinovici *et al.*, "Logistic regression was as good as machine learning for predicting major chronic diseases," *J Clin Epidemiol*, vol. 122, pp. 56–69, Jun. 2020, https://doi.org/10.1016/j.jclinepi.2020.03.002.
[31] J. Sun, L. Xu, "Cloud-based adaptive quantum genetic algorithm for solving flexible job shop scheduling problem," IEEE *7th International Conference on Computer Science and Network Technology (ICCSNT)*, pp. 1-5, 2019, https://doi.org/10.1109/ICCSNT47585.2019.8962476.
[32] Z. A. Ali, Z. H. Abduljabbar, H. A. Tahir, A. Bibo Sallow, and S. M. Almufti, "eXtreme Gradient Boosting Algorithm with Machine Learning: a Review," *Academic Journal of Nawroz University*, vol. 12, no. 2, pp. 320–334, May 2023, https://doi.org/10.25007/ajnu.v12n2a1612.
[33] G. Martínez-Muñoz, C. Bentéjac, and A. Csörg″ O B Gonzalo Martínez-Muñoz, "A Comparative Analysis of XGBoost," *Artificial Intelligence Review*, vol. 54, pp. 1937-1967, 2021, https://doi.org/10.48550/arXiv.1911.01914.
[34] H. Sharma, H. Harsora, and B. Ogunleye, "An Optimal House Price Prediction Algorithm: XGBoost," *Analytics*, vol. 3, no. 1, pp. 30–45, Jan. 2024, https://doi.org/10.3390/analytics3010003.
[35] F. Sahin, J. S. Bay, C. A. Lynn, A. Hugh, and F. Vanlandingham, "A Radial Basis Function Approach to a Color Image Classification Problem in a Real Time Industrial Application," *Doctoral dissertation, Virginia Tech,* 1997, https://vtechworks.lib.vt.edu/items/03d6d0e9-8a9d-462f-a183-ee5953920c3f.
[36] G. A. Montazer, D. Giveki, M. Karami, and H. Rastegar, "Radial Basis Function Neural Networks: A Review," *Comput. Rev. J*, vol. 1, no. 1, pp. 52-74, 2018, http://purkh.com/index.php/tocomp.
[37] T. Oommen, D. Misra, N. K. C. Twarakavi, A. Prakash, B. Sahoo, and S. Bandopadhyay, "An objective analysis of support vector machine based classification for remote sensing," *Math Geosci*, vol. 40, no. 4, pp. 409–424, 2008, https://doi.org/10.1007/s11004-008-9156-6.
[38] A. Robles-Velasco, P. Cortés, J. Muñuzuri, and L. Onieva, "Prediction of pipe failures in water supply networks using logistic regression and support vector classification," *Reliab Eng Syst Saf*, vol. 196, Apr. 2020, https://doi.org/10.1016/j.ress.2019.106754.
[39] J. Yin and Q. Li, "A semismooth Newton method for support vector classification and regression," *Comput Optim Appl*, vol. 73, no. 2, pp. 477–508, Jun. 2019, https://doi.org/10.1007/s10589-019-00075-z.
[40] N. Mungoli, "Adaptive Ensemble Learning: Boosting Model Performance through Intelligent Feature Fusion in Deep Neural Networks," *arXiv preprint arXiv:2304.02653*, 2023, [Online]. Available: http://arxiv.org/abs/2304.02653
[41] P. Bühlmann, "Bagging, Boosting and Ensemble Methods," in *Handbook of Computational Statistics*, *Springer Berlin Heidelberg,* pp. 985–1022, 2012, https://doi.org/10.1007/978-3-642-21551-3_33.
[42] P. Florek and A. Zagdański, "Benchmarking state-of-the-art gradient boosting algorithms for classification," *arXiv preprint arXiv:2305.17094*, 2023, [Online]. Available: http://arxiv.org/abs/2305.17094.
[43] A. A. Ibrahim, R. L. Ridwan, M. M. Muhammed, R. O. Abdulaziz, and G. A. Saheed, "Comparison of the CatBoost Classifier with other Machine Learning Methods," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 11, pp. 738-748, 2020. [Online]. Available: www.ijacsa.thesai.org.
[44] H. Wang and L. Cheng, "CatBoost model with synthetic features in application to loan risk assessment of small businesses," *arXiv preprint arXiv:2106.07954*, 2021, [Online]. Available: http://arxiv.org/abs/2106.07954.
[45] K. W. Walker, "Exploring adaptive boosting (AdaBoost) as a platform for the predictive modeling of tangible collection usage," *Journal of Academic Librarianship*, vol. 47, no. 6, Dec. 2021, https://doi.org/10.1016/j.acalib.2021.102450.
[46] C. Tu, H. Liu, and B. Xu, "AdaBoost typical Algorithm and its application research," in *MATEC Web of Conferences*, EDP Sciences, Dec. 2017. https://doi.org/10.1051/matecconf/201713900222.
[47] M. Al-kasassbeh, M. A. Abbadi, and A. M. Al-Bustanji, "LightGBM Algorithm for Malware Detection," in *Advances in Intelligent Systems and Computing*, pp. 391–403, 2020, https://doi.org/10.1007/978-3-030-52243-8_28.
[48] Y. Ju, G. Sun, Q. Chen, M. Zhang, H. Zhu, and M. U. Rehman, "A model combining convolutional neural network and lightgbm algorithm for ultra-short-term wind power forecasting," *IEEE Access*, vol. 7, pp. 28309–28318, 2019, https://doi.org/10.1109/ACCESS.2019.2901920.
[49] L. Göcs and Z. Csaba Johanyák, "Catboost Algorithm Based Classifier Module For Brute Force Attack Detection," *Annals of the Faculty of Engineering Hunedoara-International Journal of Engineering*, vol. 21, no. 3, 2023, https://gocslaszlo.hu/phd/tezis_4.pdf.

[50] H. Wang *et al.*, "CatBoost-Based Framework for Intelligent Prediction and Reaction Condition Analysis of Coupling Reaction," *Match*, vol. 90, no. 1, pp. 53–71, 2023, https://doi.org/10.46793/match.90-1.053W.

BIOGRAPHY OF AUTHORS

Shahbaa I. Khaleel was born in Mosul, Nineveh, Iraq, She received the B.S., M.Sc. and Ph.D. degrees in computer science from Mosul University, in 1994, 2000 and 2006, respectively, assistant prof. since 2011, and finally, prof. degree on 2021. From 2000 to 2024, she was taught computer science, software engineering and techniques in the College of Computer Sciences and Mathematics, University of Mosul. She has research in a field computer science, software engineering, intelligent technologies. She can be contacted at shahbaaibrkh@uomosul.edu.iq.
https://www.researchgate.net/profile/Shahbaa-I-Khaleel/research.

Rasha Ahmed was born in Mosul, Nineveh, Iraq, She received the B.S., degree in software engineering from the Mosul University, in 2022, and she study now Master Degree in Software Department, College of Computer Science and Mathematics, Mosul University, Iraq. She can be contacted at email: rasha.23csp16@student.uomosul.edu.iq.
https://www.researchgate.net/profile/Rasha-Ahmed-46.