International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 / https://doi.org/10.14445/22315381/IJETT-V70I4P209

Volume 70 Issue 4, 107-121, April 2022
© 2022 Seventh Sense Research Group®

Original Article

Semantic Feature Driven Consensus-Based Model for Software Reliability Assessment: A Reusability Sensitive Verification Paradigm

Prakash V. Parande[1], M K. Banga[2]

[1]*School of Computing and Information Technology, Reva University, Karnataka, India.*
[2]*Dayanand Sagar University, Karnataka, India.*

[1]prakashvp2010@gmail.com

Received: 03 February 2022 Revised: 04 April 2022 Accepted: 04 April 2022 Published: 25 April 2022

Abstract - *The exponential rise in global competitiveness and quality concern has forced software enterprises to ensure cost-efficiency with uncompromising product reliability. Software developers often intend to use the free-open-source software or class-reuse paradigm to reduce development costs. However, excessive reuse of software components often leads to pre-mature ageing, smells, and malfunction. To alleviate such issues, assessing each class for its reusability can be of great significance. Despite the numerous efforts, the existing approaches have failed to address the problems like class imbalance, shallow feature learning, and, more importantly, low accuracy. In this paper, a robust semantic-feature-driven consensus-based software reusability prediction model is developed for software reliability assessment. To achieve it, at first, it exploits a set of 17 Chidamber and Kamerer OOP matrices obtained by means of WSImport and the CKJM tool. To further enrich intrinsic feature information for future learning, s-Skip Gram-based semantic feature extraction over each metric for every class and SMOTE-ENNresampling algorithm has been employed with variance threshold algorithm and Mann-Whitney significant predictor tests. Min-Max normalization was done on the results to handle issues that rose from convergence and the over-fitting behaviour of classifiers. The simulation results confirmed that the proposed semantic-feature-driven consensus model achieves an accuracy of 98.27%, F-score of 0.983, and AUC of 0.996, which is the highest performance across the existing state-of-art methods.*

Keywords - *Software Reliability, Reusability Prediction, Consensus Learning, Semantic Features, Software Metrics.*

1. Introduction

In the past few decades, software technology has become an inevitable need of contemporary human society, though the thrust to innovate and improve at-hand solutions is still the key driving force behind the software industry [1][2]. On the other hand, increasing global competitiveness and decentralized and global operating culture too have forced the industry to produce software solutions with minimum development cycle delay and cost [1]. In this reference, developers or enterprises have been trying to reuse free-open-source software components (FOSS) or even function-reuse concepts to reduce time as well as cost [2][3]. Despite the cost-efficiency of the software reuse paradigm, the events of ageing [4], smelling [4][5], fault [6][7], etc., remain the key challenge, particularly due to excessive reuse of the FOSS or software components [8][9]. The exceedingly high reuse of software components often leads to software faults and malfunction and hence impacts software reliability [2-9]. Consequently, it raises questions on the efficacy and reliability of such (software) solutions for critical applications. In the past, numerous events can be found which were caused due to inappropriate software design and allied failures [4][6][7][9]. As a matter of fact is that there are numerous strategically important and sensitive software computing environments such as financial software, healthcare computer-aided diagnosis as well as electronic healthcare records, science and technologies, defence, and industrial monitoring and control where the reliability of the software remains as an uncompromisable need [9][10]. Thus, ensuring software reliability is a must while preserving cost-factor or allied expectations.

To address software reliability demands, software reusability and fault detection have always been the two key practices [9]. However, unlike software fault detection [8], reusability prediction has always remained a challenge due to higher complexity, inter-class dependency, and coupling demands [9]. The software reusability prediction methods demand assessing the need as well as tolerance of each class or function, or component in the program while ensuring that it doesn’t cause aforesaid adversaries like software smells, refactoring, ageing, and faults [4-7][9]. So far, the industry has been applying classical manual testing methods, including regression and or black-box approaches; however, it is resource exhaustive and time-consuming that eventually hindering the goal of enterprise [6]. Considering it as motivation, automatic reusability prediction approaches have been proposed, which exploit the different software metrics characterizing the software design features and machine learning methods [11-18]. In this reference, though a few efforts are made to exploit software metrics information to analyze the reusability of every single class (or function) in software; however, such approaches merely applied the feature association to perform classification and failed in employing semantic associations amongst the multiple classes. On the other hand, a few methods applied standalone machine learning methods [11-13][15-18] to perform two-class classification. On the contrary, with the same data, the various classifiers have shown different performances that raise questions over their generalization.

The existing reusability prediction methods, especially employing either code structure such as the line of code (LOC), or depth of inheritance tree (DIT), often ignore the intrinsic association amongst the classes or components, such as cohesion and coupling. Similarly, those approaches mainly focus on the complexity aspect to perform reusability prediction with standalone classifiers. On the contrary, merely employing complexity as the code feature can't be suitable as the coupling and cohesion are equally important to be considered when assessing the reuse-proneness of a class [8][9][11-18]. This, as a result, gives rise to the class-imbalance issue, and hence training a machine learning model with such class-imbalanced data (with non-reusable classes as the minority samples and reusable classes as the majority sample) can give skewed performance [19]. This problem has not been solved in any at-hand solution. Considering these facts, in this paper, more stress has been put on inculcating superior feature engineering, class-imbalance problems as well as classification methods to achieve highly accurate and reliable software reusability prediction [9][14][17][19][20].

Considering the above-stated problems, in this research paper, a standardized, stable semantic feature-driven consensus-based model is developed for reusability prediction towards software reliability assessment. In other words, this research proposes a reusability prediction-based model for software reliability assessment, especially for software that is developed using the objective-oriented programming (OOP) concept for development. To achieve it, the proposed model at first exploits a total of 22 software metrics obtained by means of the Chidamber and Kamerer (CK) Java Machine tool, often called CKJM-tool. More specifically, CKJM-tool was employed to extract OOP software metrics from a standard software solution. Some of the key metrics are Lines of Code, Inheritance tree depth, The weight of the methods per class, Number of child classes, Object Coupling, Cohesion or lack thereof between methods, Response for a Class (RFC), Public methods, Efferent Couplings (Ce), Afferent Couplings (Ca), Data Access Metric (DAM), Aggregation Measure (MOA), Cohesion Among Methods of Class (CAM), Functional Abstraction Measure (MFA), Cyclomatic Complexity (CC), Lines of Code IC- Inheritance Coupling (LOC), Coupling Between Methods (CBM), and Average Method Complexity (AMC). To perform reliability assessment, we perform a two-class classification for each software function or class where the above stated OOP-metrics are considered as the antecedent variable while the corresponding reuse-proneness is defined as the class variable. In this manner, the software reusability and hence reliability are defined as a function of OOP metrics. Once obtaining the aforesaid OOP-metrics, it is semantic feature extraction, which is achieved by means of the word2vec method named n-Skip Gram (SKG). Now, the extracted features are processed for SMOTE resampling to remove any possibility of data skewness or skewed classification results. Subsequently, to reduce redundant computation, the variance threshold method and Mann-Whitney significant prediction test are applied distinctly over the SKG features. Here, the key motive is to identify the feature selection method which could yield superior performance. Once selecting the features, the Min-Max normalization algorithm was applied over the specific features that were selected, which helped in alleviating any probable convergence and over-fitting problems. Now, the normalized features are projected for two-class classification by means of a consensus-based model encompassing the following binary classifiers. Naïve Bayes, Decision Tree, k-NN, Logistic Regression, ANN-LM, AdaBoost, Gradient Boosting, Random Forest, and Extra Tree classifier. Here, each base classifier predicts each class or function as Reusable or Non-Reusable and labels it as 1 and 0, respectively. Thus, with the obtained labels for each class, the proposed ensemble method estimated Consensus, also called maximum voting score, which was later used to perform reusability prediction. Unlike standalone classifier-based methods, our proposed consensus-based method provides better reliability and superior generalizable performance.

2. Literature Survey

In the initial days, the problem of reusability prediction was considered an analytical issue that explored the use of hierarchical analytical process (AHP) [21] to examine the different factors of testability of software. However, such approaches failed in addressing the inter-relation amongst the different code aspects like coupling, cohesion, etc., towards reuse-proneness estimation. To alleviate such limitations, in later years, authors explored the different software metrics, including LOC, WMC, DIT, and NLOC, for reusability prediction [22-28]. Despite the fact that these OOP metrics have a high impact on reuse-proneness [29], it could use merely structural artefacts to check the reusability of each class. Authors suggested the different OOP metrics representing cohesion, coupling, structure, and inter-dependency features to perform reusability prediction [29]. Most of the above-stated methods failed in defining a specific threshold with respect to which a class of function is called as non-Reusable [30]. In this reference, authors [30][31] proposed a predefined threshold-based model where for each OOP metric, authors applied a threshold called tolerability. A class with tolerability lower than the threshold was considered reusable else was classified as non-reusable. Yet, the use of a predefined threshold over a non-linear or unpredictable input source puts a question mark on its generalizability.

Authors in [32][33] intended to use multiple software metrics and their association to perform reusability prediction; however, it failed in addressing the class imbalance problem. Moreover, the use of standalone classifiers limited its generalizability. In [34], the authors applied a regression method to check the reuse-proneness of each class. Similarly, [35][36] applied OOP-CK metrics for reusability prediction. In [37], the authors applied OOP-CK metrics characterizing complexity, customizability, and reusability to perform software reliability. The authors applied component reuse level (CRL) to assess reusability. Interestingly, the authors found that the LOC metric can be vital as a reuse proneness indicator. Though the use of machine learning has shown superior performance toward reusability prediction; however, no significant efforts were made to address the class imbalance, inferior performance. Authors [14][38-44] indicated that the use of ensemble learning could be superior to standalone methods [39].

To further improve performance, authors [45] suggest re-sampling [46], which could help in achieving better classification. To gain higher accuracy, the authors [46] recommended using the AdaBoost classifier. Though, numerous algorithms, including neuro-computing [14][15], have exhibited better accuracy. In [14], the authors suggested decision level fusion to improve classification accuracy; however, these approaches could not address the reusability prediction problem. Summarily, these approaches indicated that the use of the consensus approach could give more reliable and generalizable performance [14]. Authors [38] suggested that the use of heterogeneous ensembles can yield better accuracy than the classical machine learning algorithm or homogeneous ensemble [47]. In addition to the improvement scope in classification, authors [39] suggested using principal component analysis (PCA) for feature selection. Authors can achieve higher accuracy even with reduced redundant computation. However, the authors indicated that the use of feature selection is more effective with superior classifiers such as random forest [39] and AdaBoost [40]. In [48], the authors suggested a heterogeneous ensemble using k-NN, Rocchio, and SVM algorithms with Dempster's rule of decision level fusion for reusability prediction. As discussed in the previous sections, this research primarily focuses on improving both computing environments as well as feature engineering to attain higher accuracy and reliability towards software reusability prediction. Functionally, a combined system of multiple algorithms was utilized.

3. Methods

This method primarily focuses on alleviating any possibility of data imbalance or skewed learning. Once resampling the feature, two different feature extraction methods named variance threshold and Mann-Whitney significant predictor test were applied to select the most important features while dropping the insignificant feature elements. Here, the key goal is to improve features for learning while reducing redundant computation. Thus, based on the maximum voting or the Consensus, the proposed model performs per-class reusability prediction. The proposed model is represented as shown in Fig.1. in the next section. The details of the design and implementation are as follows.

3.1 JM-assisted Software Metrics Retrieval

Considering benchmark software, based on availability, random software available at SourceForge was considered. Additionally, since the overall proposed model is designed to perform reusability assessment in reference to the OOP metrics, only the software designed with the OOP paradigm was taken into consideration. The considered software program was developed in the Java programming language. Now, to retrieve software matrices, a well-known web application tool named Chidamber and Kamrer JavaVirtual Machine (CKJM) [26] was applied. Here, the CKJM tool helped extract a total of 22 OOO-metrices; however, considering missing elements and discontinuity, only 17 features were retained. To improve feature efficiency and intrinsic characteristics,

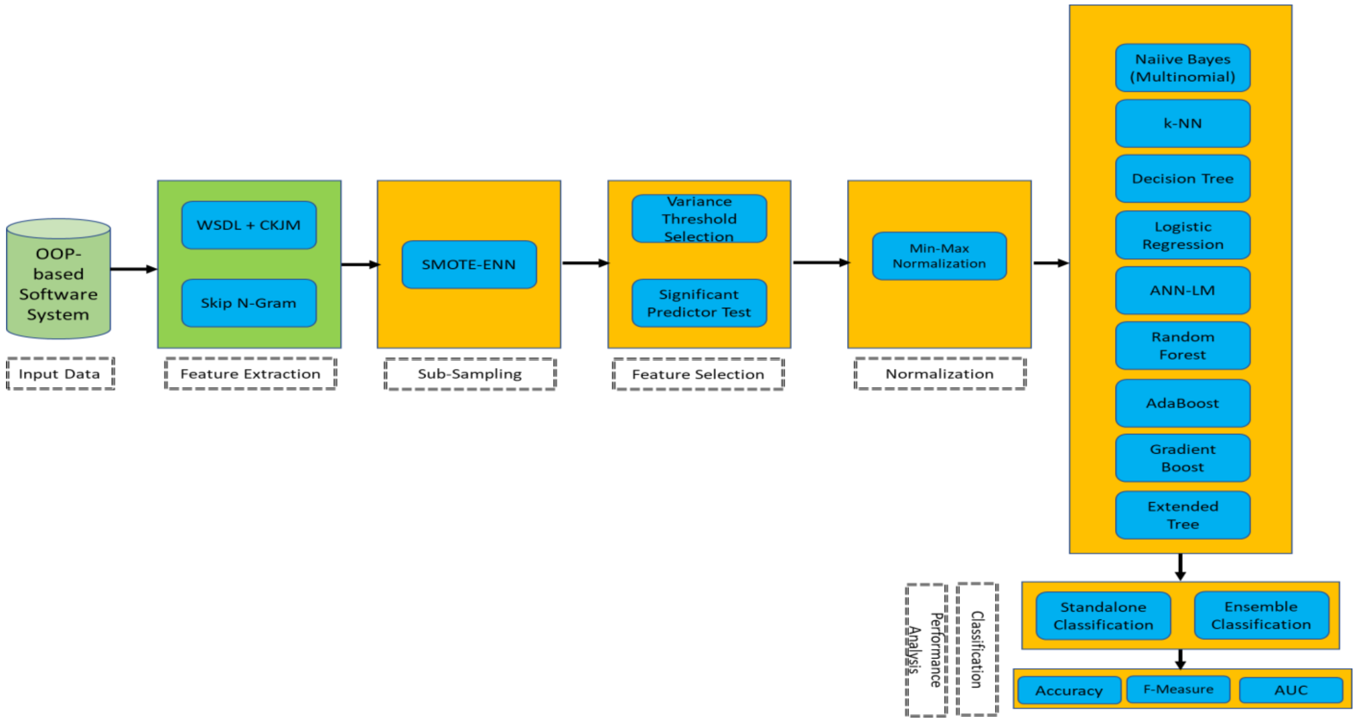

Fig. 1 Proposed reusability prediction model

In the proposed work, before executing CKJM, Web-service description language (WSDL) was applied concept that represents an XML-based interface definition language, identifying the distinct components or the software functions. Here, each function was considered as a component, also called a port type. These ports, in fact, perform varied tasks M_i by transferring input $I_i$into the corresponding output O_i. During this process, the functional component, along with its port type, is defined in the form of a unique nomenclature. These components contain the instructions to perform a certain task as defined to perform in the target software. Moreover, each data element represents specific categorical definitions, which are defined in terms of XML, while XML is stated in the form of XML Schema Definition (XSD) language representing the data type definition. In this method, the activities like encapsulation, restrictions, extension, strings, and integers are used to represent the complex software structure. Next, the XSD code was stored in a separate file which was connected to the WSDL document to estimate the type reuse. Here, initially, the services were coded, which was followed by the conversion of codes into the corresponding WSDL document. As depicted in Fig. 2, the use of the WSImport tool helped towards the conversion of the WSDL document into a Java file, which was subsequently used by CKJM to extract the OOP-metrices.

3.2 n-Skip Gram (Word2Vec Method) Based Feature Extraction

Unlike classical approaches toward software reusability prediction, where univariate logistic regression (ULR) [14][15] is applied to preserve the appropriate set of features for further analysis, in this paper Word2Vec method was applied for feature extraction. This approach intends to exploit the semantic features pertaining to each class of function and hence can yield a more efficient feature set for further learning and classification. Though, Word2Vec is a well-known semantic feature extraction method often used for text analysis, its variant called N-skip Gram (SKG) can be used to obtain the latent feature. In other words, in the proposed model, the feature set pertaining to each class is transformed into the corresponding word presentation, also called the feature vector.

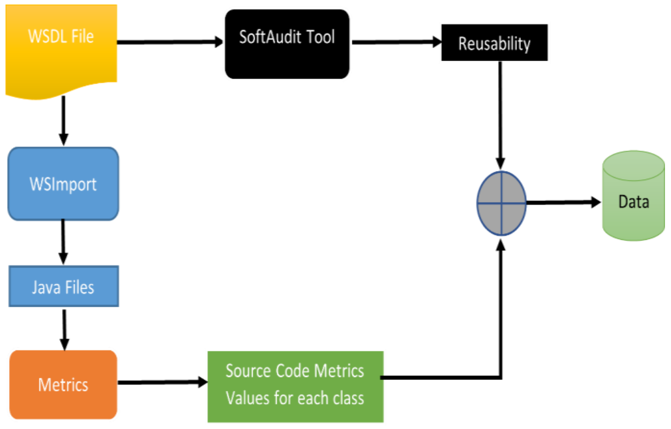

Fig. 2 OOP-Metrics estimation

In this work, the Gensim Word2Vec method was applied; it had a two-layered neuro-computing model encompassing two hidden layers. It helped extract sparser and more information-rich (semantic) features for each class and allied metrices values. Unlike the continuous bag of words (CBOW) method, in which the current token (here., metrics) is predicted based on the window of the neighboring context window, in SKG with a token W_i, the set of possible tokens W_{i-1} W_{i-2} W_{i+1}, W_{i+2} is predicted, which are highly related to W_i. This kind of semantic feature can be highly effective toward reusability prediction under uncertain design conditions or environments [49]. To assess the efficacy of this method, SKG was utilized to generate corresponding features, where a hierarchical Softmax layer model was applied for training and feature vector generation. A snippet of the SKG method used for the semantic feature extraction per class information is given as follows:

3.3 N-Skip Gram (SKG)

A typical SKG feature extraction model learns the vector-formation of each token w specific to the provided set of words $\{1, \dots, W\}$ obtained from the 17 OOP-software metrices per class. These vector representations were trained to predict the different metrics values with a higher likelihood to exist within the range of a central context of a specific keyword. Thus, with the provided sequence of tokens (often called training corpus or $w_1, \dots, w_T$), it enhances the log-likelihood of the context word in a testimonial to the specific centre word. In the proposed model, the key intention was defined as per (1).

$$\sum_{t=1}^{T} \sum_{c \in C_t} log\, log\, p(w_c|w_t) \tag{1}$$

In (1) C_t states, the context window is present around the centre token w_t. The likelihood of the context word around w_t is the Softmax over the scoring function (2).

$$p(w_c|w_t) = \frac{e^{s(w_c|w_t)}}{\sum_{j=1}^{W} e^{s(w_t.j)}} \tag{2}$$

In (2) s represents the scoring function signifying the multiplication of w_t and the context word. Thus, the objective function J_θ, which is expected to be minimized with extended training, was obtained as the estimation of the feature metric in the presence of noise. Mathematically, it is defined as (3).

$$J_\theta = -\sum_{w_i \in V} log\, log\, \frac{p(w_i|c)}{p(w_i|c)+kQ(w_i)} + \sum_{j=1}^{k} log\, log\, \frac{p(\underline{w}_i|c)}{p(\underline{w}_{ij}|c)+kQ(\underline{w}_{ij})} \tag{3}$$

The objective function (3) Q signifies the noise distributions that were used to generate k-noise samples. The proposed SKG model employed the embedding matrices belonging to the n-gram (n=1), a number of elements/tokens in n-gram vocabulary, to generate context words. Moreover, it employed a token-only embedding matrix with n=1, skip-gram size 1, and the window size of 5. Functionally, with each mini-batch of input corpus and allied feature labels exchanged during each run, it passed allied mapping to the n-gram. This, as a result, estimated the embedding vectors, which were then added using word-only embedding to constitute the needed word-vector representations. Finally, these word vectors were passed to the loss estimation function, where a stochastic and holistic back-propagation method was used to evaluate and reassign the embedding metrics. After extracting the SKG feature vector for each metric per class (of the software), it was processed for data resampling.

3.4 Edited Nearest Neighbor SMOTE (SMOTE-ENN)

In a software solution, the likelihood of non-reusable component(s) would be relatively significantly smaller than the usable classes. In such cases, the probability of class imbalance can't be denied. In such conditions, where the samples pertaining to the minority class are significantly lower than the majority class, the learning and eventual prediction results can be skewed. Such data skewness might force the learning model to often show the results inclined towards the majority class and hence false-positive. Considering this unavoidable problem, in this paper, resampling was performed over the extracted SKG feature vectors. Unlike conventional random sampling methods of the up/down sampling methods, where the fraction of resampled data might vary and can eventually impact the final classification results, an extended SMOTE (SMOTE-ENN) resampling method was applied[50]. In fact, the use of random sampling, especially in those cases where the fraction of the minority class is significantly small, the use of random sampling might even increase the majority class examples and hence more skewed performance. Similarly, up-sampling approaches might increase non-reusable (say, minority class) samples or allied features uncontrollably and hence can impact the overall performance [50]. Considering these at hand issues, in this paper, an extended SMOTE resampling method named SMOTE-ENN was applied to estimate the suitable set of (resampled) feature vectors for further computing.

The classical SMOTE (resampling) algorithm model generated synthetic positive samples using a k-NN algorithm. In this method, n-nearest neighborhood elements are selected for the minority "non-Reusable" class, which was followed by equalization of the samples in such a manner that it yields the number of minority classes the same as the number of the majority class. Despite the superior performance over the classical random or up-sampling methods, the classical SMOTE, as stated above, undergoes limitations like over-generalization and variance [51]. In a real-time problem, there can be many conditions where there would not be clear class boundaries as the synthetic minority class instances, in an uncontrollable fashion, can cross over or come close to the majority class [52]. Such problems can be more frequent in the case of non-linear data with broadened feature space [52]. It can later cause mislabeling of the synthetic samples in the minority class and can give higher false-positive results [53]. To alleviate this problem, in the proposed work, SMOTE-ENN was applied, specially designed in sync with the SMOTE and Edited ENN algorithms to pre-process the samples in the synthetic set. In our proposed SMOTE-ENN method, the label of each synthetic instance was compared with the Consensus-based value of the first k-NN. In case of any inconsistency with reference to its k-nearest neighbors, the instance was removed, while the consistent instances were retained, and accordingly, the feature vectors were updated.

3.5 Variance Threshold and Significant Predictor Test-based Hybrid Feature Selection

Undeniably, the use of SMOTE-ENN helped our proposed model to address and possibly solve the key problem of class imbalance or skewness; however, at the cost of increased samples and hence computation. In sync with this issue, in our proposed model, feature selection has been performed where the focus is to retain the more diverse and significant features while dropping the low-significant feature elements. In this work, two different feature selection algorithms were applied - the first algorithm acts as a filter element while the subsequent method identifies the most suitable or significant set of features for further computation. As the first layer of filtering, variance threshold-based feature selection (VTFS) was applied, while in the subsequent step Mann-Whitney Predictor Test (MSPT) was applied.

3.5.1 Variance Threshold Feature Selection (VTFS) Methods

Typically, the VT method is considered the baseline concept for feature selection; however, its ability to eliminate those all samples or feature elements whose variance doesn't fulfill some predefined or expected threshold makes it a goal-oriented method that results in satisfactory performance. Functionally, the VT method eliminates all zero-variance feature elements, signifying those all features have the same value in all samples. We hypothesize that the feature elements with high variance can have more significant information to make predictions. Noticeably, unlike correlation test-based approaches, VT doesn't consider any correlation or relationship between features. In this work, the VT threshold as 0 was applied, signifying features with zero-variance. And thus, those elements having low variance or zero-variance were dropped, while the remaining were retained for further computation.

3.5.2 Mann-Whitney Significant Predictor Test (MSPT)

Over the retained features by the VTFS algorithm, Mann-Whitney Rank Sum Test or the significant predictor test was executed to further refine the feature set. The proposed MSPT model estimated correlation amidst the different elements to evaluate the extent of significance towards the reusability. Here, the Mann-Whitney algorithm helped in estimating the correlation amidst the resampled features. The resampled feature for every class was interpreted as an independent variable, while its corresponding reuse-proneness was defined as the class label or consequent variable. Hence, once recovering the significance level of every feature, the feature with a higher probability of affecting the reusability was preserved, while the samples with a lower level of significance were removed from further consideration. The significance level $p = 0.05$ was assigned. In this manner, the feature elements of higher significance than 0.05 were selected, while the others were taken off. Now, after performance features are selected, it is hypothesized that the chosen features maintain lower feasible significant elements than that of performance prediction.

Now, once the input feature vectors were obtained, data normalization was executed to alleviate any possible convergence or over-fitting problem. Normalization used for this purpose is iterated in the next section.

3.6 Min-Max Normalization

In almost all major classification or prediction systems that utilize large dataset-based models, data imbalance and convergence are the key problems which hinder the gross performance of the system. Post feature extraction and selection, the heterogeneity of the retrieved data elements causes a vast amount of hindrance to the learning model and makes it undergo premature convergence and even over-fitting. It can affect overall computational efficiency, and therefore to alleviate it, Min-Max normalization was performed over the retained salient features. The proposed Min-Max normalization algorithm, as indicated in (4), normalized features or mapped values in the range of zero to one. The Min-Max normalization method linearly transforms all data using binary normalization. Functionally it maps all values to a fixed range between [0,1] inclusive of both limits. An equation in (4) was applied to approximate the normalized value(s) of the input datax_i .

$$Norm(x_i) = x_i' = \frac{x_i - min\ (X)}{(X)\ -\ min\ (X)} \quad (4)$$

In (4), the data elements max(X) and $min(X)$ state the maximum and minimum values of X, respectively.

3.7 Heterogenous Ensemble Driven Consensus-based Reusability Prediction

Though several machine learning methods have been applied to perform software aging detection, smell detection, reusability prediction, etc.; however, an interesting inference can be derived that the different machine learning methods perform differently on the same data or feature. In other words, the different machine learning algorithm or classifier results in different performance for the same problem, putting the question on their generalization. In such cases, considering or proposing a specific machine learning method as the best or optimal solution doesn't guarantee its optimality. Taking into consideration the above-cited, in this research work, unlike the existing classical standalone classifier-based software prediction model, the heterogeneous ensemble learning model was designed to encompass multiple base learners. Noticeably, our key purpose was to use the machine learning classifiers from the different categories to provide a diversity of performance. Noticeably, unlike classical methods where authors often use different base classifiers one-after-another, in the proposed model, the heterogeneous ensemble learning model was designed in a manner that inputting the same data as input to all classifiers they function in parallel. Thus, such parallel computation ability enables the proposed model to achieve time efficiency as well as a great diversity of results. Additionally, reusability prediction as a two-class classification problem classifies each class as Reusable or Non-Reusable, and labels each class as 1 and 0, respectively. In the current work, a total of 9 machine learning algorithms were applied as the base classifier to perform ensemble or consensus-based classification. These key algorithms are given as follows:

3.7.1 Naïve Bayes

It is typically used as a probabilistic classification approach that applies Bayes' rules with autonomous hypotheses to classify input patterns. Being a probabilistic approach, it is also expressed as an "independent feature model", which presumes that all allied features are independent of one another and do not affect the classification result decisively. It assumes that the presence of a particular feature in a class is not related to the existence of another feature. Functionally, the NB algorithm allocates an object x to the class $e^* = argmax_d P(d|x)$ as per the Bayes′ rule. Mathematically, it is derived as (5).

$$P(d|x) = \frac{P(d)P(d)}{P(x)} \tag{5}$$

In (5) $P(d)$ refers to the class-prior probability of c. The other parameter $P(d|x)$ states the likelihood of x the data element while $P(x)$ states the predictor prior probability and is defined as (6).

$$P(x|d) = \prod_{l=1}^{m} P(x_l|d) \tag{6}$$

In this work, the multinomial Naïve Bayes algorithm was applied to classify each class as reusable or non-reusable. Unlike Gaussian NB, Multinomial NB (MNB) learns on the basis of the count's frequency, signifying the number of times x_i it occurs over n trails. Here, feature vectors state the frequency with which a specific event is caused by a multinomial function. The applied multinomial NB classifier applies the occurrence(s) of the binary terms to classify the input. Functionally, the NB algorithms classify each query as reusable and non-reusable and label it as "0" and "1", respectively.

3.7.2 Decision Tree (DT)

Decision trees and their many variants have been some of the most applied machine learning tools for the classification of data. There are limitations to the classic DT, but they have been overcome using many improvements to the algorithm in its application and the pre-processing data stages. Some advanced variants are ID3, CART, DT C 4.5, and DT 5.0. These methods are specifically developed for data mining on complex data sets. The tree traditionally starts at the root node and maps the consequences to their respective antecedents using an association rule. Wherever a split in the rule is observed, the branches are formed at each node of the tree. Then, using the information gain ratio (IGR) in the next phase, each node gives rise to two branches. Thus, the proposed C5.0 decision tree algorithm labelled each class as Reusable (label- '0') and non-Reusable (label- '1').

3.7.3 k-NN

k-NN, a commonly known classifier, is one of the most popular models that classify unlabeled observations or patterns by assigning it the class of the most similar labeled examples. The simple implementation of k-NN enables it to be used for major data mining and predictive regression purposes; however, it has been found robust for numerous classification scenarios. By default, Euclidean distance is applied to estimate inter-attribute distance using (7) fork-NN classifiers.

$$D(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_n - q_n)^2} \tag{7}$$

In (7), p and q are compared with n features. In addition to it, even Manhattan distance can also be used.

Summarily, the driving force in maintaining an optimal balance between performance and computation depends on maintaining a better balance between overfitting and under-fitting. In the majority of the classical approaches, authors have assigned the value of K as the square root of the number of instances or observations in the training data; however, its efficacy for large-scale data with varying patterns can't be guaranteed. In the majority of the existing approaches, K values are applied based on sample size by applying the cross-validation scheme; however, at the cost of increased time exhaustion. Unlike classical k-NN algorithms, in this paper, a kTree learning model has been developed that enables learning varied optimal k values for the different training samples by encompassing the training stage when performing k-NN based reusability prediction. During the training phase, three models first perform learning the optimal value of k for all data samples under study by applying a sparse reconstruction mechanism.

3.7.4 Logistic Regression

Logistic regression is the most favored regression mechanism when dealing with multivariate systems. In the software of the present reusability prediction problem, Logistic regression applies regression over the selected OOP-CK metrics, where CK metrics and finally obtained fused feature set, was the causal or antecedent whereas the reuse-proneness or the probability of reuse has been taken as the consequent or effect variable. Hence, the regression provides two results, signifying non-reusable and reusable. Mathematically, (7) was applied to perform linear regression over the input features.

$$logit[\pi(x)] = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots \ldots + \beta_m X_m \quad (8)$$

In (8), $logit[\pi(x)]$is the intermediate output variable while $x_i$represents the input variable. This method transforms the variable output into a limited variance output$\pi(x)$varying in the range of 0 to 1 $-\infty\ +\infty$. Observing (9), the value $m$implies the total available input variables, while the prospect of a reuse-proneness of each class is given by π.

$$\pi(x) = \frac{e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots\ldots+\beta_m X_m}}{1+e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots\ldots+\beta_m X_m}} \quad (9)$$

3.7.5 Artificial Neural Network Variants

Amongst the major machine learning algorithms, the neural network, often called ANN, has been applied extensively for data learning and classification purposes. The robustness of ANN makes it efficient to be used in diverse classification problems, though based on computational complexities and adaptive computation ANN has evolved through different phases. Investigating in-depth, it can be found that the performance of ANN is directly related to the corresponding learning method. Thus, based on the learning method, ANN has been evolved as ANN with the steepest gradient (SD), ANN with RBF (ANN-RBF), ANN with Levenberg Marquardt (ANN-LM), ANN with gradient descent (GD), Probabilistic Neural Network and Extreme Learning Machine (ELM) algorithms, etc. ANN-LM possesses higher robustness than ANN-SD and ANN-GD, individually. Functionally, it can be configured to possess features of ANN-GD as well as ANN-SD and therefore provides higher stability and exhibits better performance characteristics even with large, non-linear, and heterogeneous data.

Artificial Neural Network (ANN), in essence, attempts to copy the way a human brain learns patterns from a series of input data or update knowledge using those patterns. Thus, learning over such input patterns classifies unrevealed input into target categories. An illustration of the ANN model is given in Fig. 3. As depicted in the figure, ANN typically has three layers, output layer, hidden layer, and input layer. Considering the architecture of ANN, it generally involves multiple inputs being brought into it via input neurons titled perceptrons and pushed into the next layers for trigger-based classification. To display any learning, ANN performs two phases of functions. The first is an error function using difference identification, and the second is back-propagation, where the error is fed into the network repeatedly till the error function reaches a near 0 value. In this research paper, the ANN algorithms perform two-class classification, where it classifies each class as reusable or non-reusable and labels them as “1” and “0”, respectively. Being a two-class classification problem, the ANN designs have one output layer, as given in Fig. 2. In our proposed neuro-computing or allied learning model, the final features selected pertaining to each class of the software are fed as input to the ANN (Fig. 2), while the number of hidden layers is varied. At the input layer of the ANN, a rather simple linear activation is performed, which generates an output equal in magnitude to the input (i.e.,$O_o = I_i$); the output layer, however, takes the input from the final hidden layer where a summing function is present to get the highest contributor. Noticeably, the output layer applies the Sigmoid function (10) to generate O_h.

$$O_h = \frac{1}{1+e^{-I_h}} \quad (10)$$

ANN applies certain error functions such as mean square error (MSE) as a measure of accuracy, which is estimated using (11).

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i' - y_i)^2 \quad (11)$$

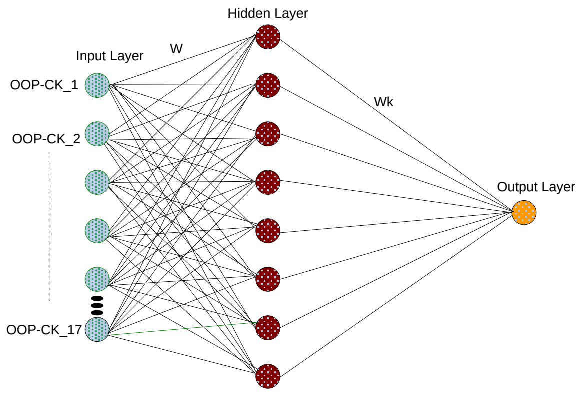

Fig. 3 ANN-LM architecture with a single hidden layer with one output node

In (11), y is the observed value, while y_i' is the expected value. Typically, the ANN model is applied (12-13) to perform learning while minimizing the error values as given in (11).

$$w^* = L(w) \quad (12)$$

$$L(w) = \sum_{t=1}^{N} L\big(y_t, f_w(x_t)\big) + \lambda R(w) \tag{13}$$

Unlike classical ANN-GD and ANN-SD, ANN-LM possesses more robustness in learning over large non-linear data input. It confines the multivariate function to its base values, called Sum of Squares (SoS), thus converting the non-linear real-valued functions to linear integral valued functions. This feature enables ANN-LM to perform fast and more efficient weight updates. Also, this feature aids ANN-LM to avoid issues arising from convergence and local minima. Hence, this method is better suited for large data sets. As already stated, ANN-LM has the ability of both ANN-GD and ANN-SD ANN-GD, which provides retrieving swift error minimization. ANN-LM utilizes (14) to perform weight updates during learning.

$$W_{j+1} = W_j - \left(J_j^T J_j + \mu I\right)^{-1} J_{j^{e}j} \tag{14}$$

In (14), the parameter W_j signifies the at-hand weight while W_{j+1} providing the updated weight. Similarly, I represent the identity matrix, while the J indicate the jacobian matrix (15). Equation(15) μ contains the combination coefficient and the minimum value of μ changes in the behavior of ANN-LM to ANN-GD. On the other hand, the maximum value pushes it towards ANN-SD.

$$J = \begin{bmatrix} \frac{d}{dW_1}(E_{1,1}) \frac{d}{dW_2}(E_{1,1}) \cdots \frac{d}{dW_N}(E_{1,1}) & \frac{d}{dW_1}(E_{1,2}) \frac{d}{dW_2}(E_{1,2}) \\ \cdots & \\ \frac{d}{dW_N}(E_{1,2}) \ \vdots\vdots\vdots\vdots \ \frac{d}{dW_1}(E_{P,M}) & \frac{d}{dW_2}(E_{P,M}) \cdots \frac{d}{dW_N}(E_{P,M}) \end{bmatrix} \tag{15}$$

In (15), P indicate input features. The output is given by $M$the total weight counts are represented by N.

3.7.6 Random Forest (RF)

Random Forest is one of the most successful ensemble-learning algorithms that structurally encompasses multiple tree-based classifiers, behaving as an ensemble learning model. In the proposed tree model (or tree structure), each tree provides its corresponding choice for the feature with the highest probability for each class. Let the total training samples be N, and then a sample encompassing Ncases is randomly selected from the initial data. These selected samples are further utilized as the training set to generate a new tree. Now, if M input variables, then the optimal split on these m is initiated to split the node. Here, the value m was maintained as constant during forest development, also called the growing phase. In this manner, each tree is developed to the maximum possible limit. Unlike classical machine learning methods, the random forest algorithm needs a smaller number of parameters to be estimated during classification. It makes overall computation more efficient and suitable for real-time uses. A complete random forest algorithm can be eventually defined as the combination of the different tree structures, as presented in (16).

$$\{h(x, \theta_k), k = 1,2, \ldots i \ldots\} \tag{16}$$

In (16), the parameter h signifies the classifier function while $\{\theta_k\}$ presenting the random vector. An individual tree contains a vote for the maximum probable class as input x. Tree formation governs the dimensionality θ. In fact, the key reason behind random forest success is its ability the formation of each decision tree that forms the forest. In the proposed method, the random forest was developed to accommodate about 70% of the samples using a bootstrapped data set and achieve the training, whereas the remaining samples were treated as out-of-the-bag samples, and the model was validated using the same. This validation provided the data for inter-class validation as well.

3.7.7 AdaBoost (ADAB)

AdaBoost is a type of adaptive boosting method that possesses the potential to enhance the characterization ability recursively. The prerequisite tests that are used to initialize the boosting are initially weak learners with data pre-processing given the higher emphasis. Functionally, post each cycle of computation, the applied ADAB model [54] calculates the error rate for the weak classifier. Then the weights of the correctly classified samples are expanded to counter the weights for the incorrectly classified samples. Eventually, the weak learner turns out to be a strong learner that finally classifies each class of the software as the reusable or non-reusable class and labels them as "0" and :1", respectively.

Like the above stated AdaBoost method, the gradient boost algorithm as well has been applied as a base classifier.

3.7.8 Extra Tree Classifier (EXT)

The EXT classifier establishes a bunch of unpruned or unshaped choice trees according to the traditional hierarchical methodology. Dissimilar to RF calculation, it includes randomization of both properties just as a cut-point choice while parting a hub of a tree. However, it can likewise make a total arrangement of randomized trees that are totally autonomous of one another in their group results, structures, and the info for preparing tests. Basically, it is separating itself from other tree-based outfit techniques because of two key variables. These are, it divides hubs by choosing cut-foci totally blind and utilizes the total preparing test to empower tree development. Along these lines, the ordered results, or the forecasts of the relative multitude of trees, are joined to give the last expectation yield by applying the MVE strategy. Immediately, the critical idea driving EXT is that the total randomization of the cut-point and characteristic out and out with troupe averaging diminishes the difference better in contrast with the more fragile randomization approaches utilized in different strategies. In addition, the utilization of the first preparing tests rather than the bootstrap imitations also diminishes the probability of inclination and thus accomplishes more exact and effective grouping yields.

The above-discussed machine learning algorithms were applied as base classifiers to perform two-class classification. Each of the classifiers labeled each class of the software as reusable ("0") or non-reusable ("1"). Thus, a consensus value using the concept of the maximum voting ensemble (MVE) was applied to estimate the highest label value per class, and the highest prediction output (1 or 0) for a class was predicted as a result. Thus, applying this consensus approach, each class is predicted as reusable or non-reusable.

4. Results and Discussion

Recalling the fact that the excessive reuse of the software component might lead to aging, refactoring, premature shutdown, computational errors, etc. This research work hypothesizes that the use of an automated reusability prediction can help identify the components with reuse-proneness. This, as a result, can help design a robust and highly efficient software solution to meet up-surging demands. Though a few efforts have been made towards reusability prediction; however, most of the existing methods could not address an inevitable (and unavoidable) problem of class imbalance, feature-sensitiveness, and diversity of performance with the different machine learning methods. On the contrary, these computational issues do have a direct impact on the performance of the overall reusability prediction systems(s), and hence most of the existing methods can't be generalized, especially under an unknown environment (i.e., the software under test (SUT) with different size, component size, and design paradigm).

Unlike classical approaches, this research leaned on latent or developmental features from each OOP-metrics for the comprising classes or functions. To achieve it, this research first obtained an OOP-based software solution from www.sourceforge.com. The considered software program had 1000s of classes incorporated to perform different correlated as well as independent tasks. Once obtaining the software solution, it was processed for WSImport, followed by the CKJM tool [14], which obtained a total of 22 OOP metrics. However, processing the extracted features for missing elements and outliers, a total of 17 features characterizing coupling, cohesion, complexity, and structural details were taken into consideration. The retained OOP metrics were WMC, DIT, NOC, Ce, NPM, DAM, MOA, MFA, CBO, RFC, LCOM, Ca CAM, CC, LOC, CBM, and AMC. Once obtaining these OOP metrics using CKJM, unlike classical approaches where these features are directly passed to the classifiers, semantic feature extraction was processed using n-Skip Gram (SKG), a Word2Vec method. The applied SKG method obtained the set of semantic features for each metric pertaining to each class. Subsequently, to address the problem of class imbalance or skewed learning, SMOTE-ENN resampling was applied over the extracted SKG features, which was followed by a hybrid feature selection process. As a matter of fact that the use of SKG followed by SMOTE-ENN increased the sample size, and hence a cascaded feature selection method was applied for avoiding repetitive computation and improving the learning curve. In this approach, two distinct feature selection algorithms named the Variance Threshold method, and MSPT were applied in a cascade manner. Here, the prime motive is to retain only significant features while removing the relatively insignificant or redundant feature values. Once selecting the feature, Min-max normalization was applied to alleviate the problem of convergence and over-fitting. Realizing the fact that not only the feature engineering can help achieve superior performance but also demands a better learning environment.

To alleviate any possibility of time exhaustion, these nine base classifiers were applied in parallel, which classified each class as "Reusable" and "Non-Reusable", and labeled them as "0" and "1", respectively. Now, once obtaining the labeled output for each class of the considered SUT, the maximum voting ensemble (MVE) was executed, driven consensus model, which obtained the highest score or majority voting for each class. A class with a minimum of five '0s' was classified as "Reusable", while a class with a minimum of five '1s' was predicted as "non-Reusable". Thus, applying this methodology, the proposed model exhibited the reusability prediction for each class of the SUT. Eventually, with the predicted output, the software can be designed or redesigned to retain higher reliability.

Intra-model assessment, the performance is assessed in terms of the different base classifiers, while in inter-model assessment, the comparison is done with the existing contemporary methods. A detailed discussion of the overall results is given as follows:

4.1 Intra-Model Characterization

As an intra-model comparison, the performance with the different machine learning classifiers as well as the proposed consensus-based classifier is analyzed. Here, our key objective is to assess the efficacy of the different standalone classifiers in comparison to the proposed MVE-driven consensus learning model. The relative performance outcomes in terms of accuracy (Fig. 4), F-Measure or F-score (Fig. 4), and AUC (Fig. 5) are given as follows. To be noted, the results obtained in Fig. 4, Fig. 5, and Fig. 6 are in reference to the SKG (i.e., n-Skip Gram) features followed by the SMOTE-ENN resampling cascaded hybrid feature selection method.

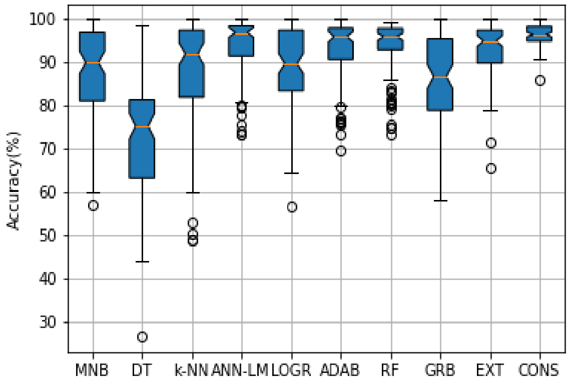

Fig. 4 Accuracy performance by the different standalone classifiers as well as the proposed Consensus driven model

From Fig. 4, .the proposed CONS (Consensus) learning method, which is derived as a heterogeneous ensemble with nine distinct base classifiers (Fig. 4), outperforms other approaches. It is observable that the proposed model exhibits an accuracy of 98.27%, which is significantly higher than other base classifiers. Though, the other classifiers like ANN-LM (98.14%), AdaBoost (ADAB, 97.8%), and Random Forest (RF, 98.16%), too, have performed significantly well. The other ensemble method, named Extra Tree Classifier (EXT) too, has exhibited a prediction accuracy of 96.8%. The decision tree (DT) model may show an accuracy of 76.21%, which is the least amongst the nine-base classifiers. The k-NN based prediction too resulted in 92.3% accuracy, which is higher than the multinomial naïve Bayes method and logistic regression (LOGR, 89.92%). The proposed consensus-driven model (CONS) achieves the highest accuracy of 98.27% of other methods. The proposed CONS model encompasses the voting from the multiple base classifiers to perform eventual classification. Its reliability is higher than other standalone methods. Table 1 depicts the accuracy values for each developed method.

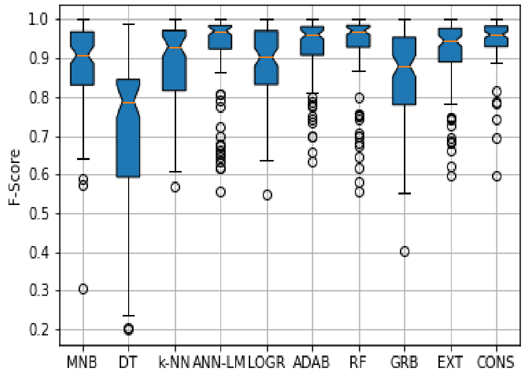

Fig. 5 F-Score performance by the different standalone classifiers as well as the proposed Consensus driven model

In software computing, especially in classification problems under data imbalance conditions or non-linear pattern scenarios, it is always expected to have higher F-score or F-measure values signifying superior precision and recall under different test conditions. In sync with this statement, the results obtained in Fig. 5 affirm that the proposed CONS (Consensus)-based classification model achieves a superior F-score (0.983) than other standalone classifiers. Observing the result (Fig. 5), it can also be observed that though the key standalone methods like ANN-LM (0.971), ADAB (0.968), RF (0.979), and EXT (0.968) have exhibited satisfactory F-score signifying its suitability towards learning under imbalanced data with significantly large features. However, the higher efficacy of the CONS method outperforms these standalone classifiers. The minimum F-Score obtained was with the DT (decision tree 4.5) algorithm, which could achieve the F-score of 84%, though the highest value observed was 0.920. Thus, similar to the accuracy performance (Fig. 4), the result in terms of F-score (Fig. 5) confirms the superiority of the proposed CONS model towards software reusability prediction.

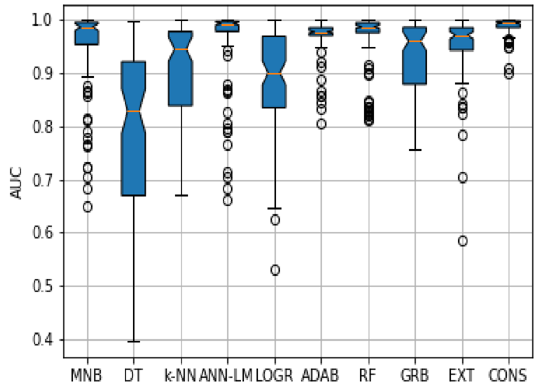

Fig. 6 AUC performance by the different standalone classifiers as well as the proposed Consensus driven model

The AUC parameter not only signifies the robustness of a machine learning model under an imbalanced, non-linear test environment but also represents the sensitivity of a model to yield higher accuracy. To assess the robustness of the proposed reusability prediction model, the AUC performance was obtained for the different machine learning algorithms. The results obtained (Fig. 6) reveal that the proposed CONS-driven learning model exhibits an AUC of 0.996, which is higher than the other standalone classifier. An interesting fact can be observed that though the proposed CONS model exhibits the highest AUC value; however, the efficacy of ANN-LM (0.983) and ensemble variants like ADAB (0.972), RF (0.985), and EXT (0.980) can't be ignored. This result (Fig. 6) affirms the robustness of the proposed model toward reusability prediction. Despite the fact that the other standalone classifiers as stated above (i.e., ADAB, RF, EXT, and ANN-LM) too have shown satisfactory performance, being a Consensus driven approach, our proposed model can be more efficient, reliable towards hand reusability prediction.

Table 1. Performance values for each method

	Accuracy	F-score	AUC
MNB	97%	0.98	1.0
DT	76.21%	0.83	0.93
KNN	92.3%	0.97	0.96
ANN-LB	98.14%	0.971	0.983
LOGR	89.93%	0.98	0.97
ADAB	97.8%	0.968	0.972
RF	98.16%	0.979	0.985
GRB	95%	0.95	0.97
EXT	96.8%	0.968	0.980
CONS	98.27%	0.99	0.99

From Fig 4 through Fig 6, the proposed CONS-driven model with SKG features followed by SMOTE-ENN resampling and cascaded hybrid feature selection (VTFS and MSPT) can yield optimal and the best results towards reusability prediction. The higher accuracy (98.27%), F-score (0.983), and AUC (0.996) affirm the robustness of the proposed model toward reusability prediction.

4.2 Inter-Model Characterization

To assess relative performance, its performance has been compared with some other existing approaches. The detailed discussion of the inter-model comparison and analysis is given as follows:

Authors in [11] made an effort to exploit the efficacy of the OOP metrics towards reusability prediction using the Leven Marquardt ANN (LM-ANN) model. Interestingly, the highest accuracy observed was below 90%, which is less than the proposed consensus-based model, which achieves an accuracy of almost 98.27%. In [12], the authors applied a total of six software metrics representing DIT, WMC, LOC, CBO, LCOM, and NOC features to perform reusability prediction. The authors applied K-Means clustering over the extracted features, which were subsequently processed for classification using a decision tree classifier with 10-fold cross-validation. Interestingly, the highest accuracy observed in [12] was 67.22%, which lags significantly from the proposed model of 98.27%. A similar effort was made in [13] as well, where authors applied the aforesaid six software metrics to perform reusability prediction. As classifiers, authors applied regression and decision tree; however, they underwent reduced performance (accuracy <90%). Though the authors tried to improve feature engineering using the rough set method; however, they could not achieve an accuracy of more than 90%. This alone is sufficient proof of the robustness of the proposed model over the existing methods [11][12]. Authors [3] too applied CK metrics to perform reusability prediction, where two metrics, DIT and WMC, were taken into consideration. To improve the performance, the authors applied a self-organizing map (SOP) to cluster CK metrics, followed by a threshold definition with reference to which it performed a reusability severity assessment. Noticeably, the authors [13] primarily focused on assessing the role and impact of the different software metrics on reusability. In our previous research as well, six different CK metrics with the ensemble learning method were applied for reusability prediction. The highest accuracy obtained was 97.02%, which is almost 1.25% lower than the proposed method in this paper. Recalling the previous work [14], DIT, WMC, LOC, CBO, LCOM, and NOC features were applied, which were further trained using machine learning classifiers named NB, LOGR, DT, Linear regression (LR), SVM, and MARS. As a feature selection method, univariate logistic regression and rough set algorithms have been used. Despite the heterogeneous ensemble learning, a maximum of 97% of accuracy was achieved and a 0.961 F-score, which is considerably lesser than the proposed method in this paper. It affirms that the proposed semantic feature set using the SKG method over a total of 17 OOP metrics provides sufficiently large feature information to perform reusability prediction. Moreover, the role of the proposed SMOTE-ENN cannot be ignored as it could have helped avoid class imbalance. Additionally, the proposed cascaded feature selection method with VTFS and MSPT too could have helped in achieving superior performance. Authors [15][16] too applied CK metrics followed by machine learning algorithms for software reusability prediction; however, the relative performance affirms that the proposed model in the paper exhibits superior and more reliable reusability prediction. In [17], the authors applied the aforesaid six CK metrics as features, which were learnt using the different machine learning algorithms, including regression techniques (LR, LOGR, MARS), NB, SVM, and ANN variants. Though to alleviate convergence issues, the authors applied genetic algorithms. However, the highest accuracy (with genetic algorithm-based ANN) could be 97.71%, while the F-score was 93.20%. In comparison to our proposed model in this paper, the proposed method exhibits superior to the existing work [17]. The same authors in [18] applied a total of 11 machine learning algorithms over six OOP-metrices for reusability prediction; however, the highest accuracy obtained was 97.71%, and F-score was 96.71% with adaptive genetic algorithm-based ANN (AGA-ANN). Noticeably, authors [15-18] had applied different machine learning algorithms as the standalone classifier and have obtained the performance independently.

In reference to the above inferences, it can be now confirmed that the use of semantic features extracted onto the OOP-metrics can provide a more significant feature vector for reusability prediction. Additionally, with the above-extracted features, the use of the SMOTE-ENN algorithm followed by the proposed cascaded hybrid feature selection method cannot only alleviate the class imbalance problem but also improve learning while removing redundant computation. It makes learning more efficient and hence helps achieve superior performance. Last but not the least, the use of consensus models with classical machine learning methods as well as ensemble classifiers has strengthened the proposed model to achieve superior performance. The set of techniques, as stated above, can be applied for real-time reusability prediction tasks.

5. Conclusion

In this paper, a highly robust and optimally calibrated software reusability prediction model was developed for reliable software engineering or design. This research mainly focused on feature engineering followed by classification or computing environment to achieve higher prediction accuracy. In this reference, recalling the fact that most contemporary software solutions are designed based on the OOP concept, large OOP metrics were extracted representing coupling, cohesion, connectivity, structural artifacts, and complexity to perform reusability prediction. Unlike conventional methods, this research processed semantic feature extraction onto the extracted OOP-metrics per class that provided an information-rich semantic feature vector characterizing the depth code features and their association with the reusability. More specifically, this work applied n-Skip Gram (SKG), a well-known Word2Vec embedding concept, to generate high-dimensional semantic features. Realizing the unavoidable presence of a class imbalance in at hand reusability prediction problem, the extracted semantic feature was processed for resampling using SMOTE-ENN. Noticeably, the use of SMOTE-ENN not only helped in alleviating the class-imbalance problem but also retained optimal intrinsic features and synthetic sample ratio to enable accurate learning. Subsequently, over the SMOTE-ENN resampled features, a novel cascaded hybrid feature selection method was applied using the Variance threshold method VTFS and Mann-Whitney significant predictor test (MSPT). The cascaded implementation of VTFS and MSPT helped ensure the retention of the most suitable feature set while dropping insignificant or redundant samples. It can be highly effective to improve the computation ecosystem. Normalization was carried out to handle the issues with over-fitting and convergence. Finally, the proposed model presents a highly robust heterogeneous ensemble method encompassing MNB, k-NN, LOGR, DT, RF, AdaBoost, Gradient Boost, and Extra Tree classifier to perform consensus-based prediction. The consensus-based classification helped achieve more accurate and reliable prediction results. In the depth performance analysis over a test software model, the proposed model achieved an accuracy of 98.27%, F-score of 0.983, and AUC of 0.996, which is higher than any known algorithm so far. Undeniably, the use of more OOP-metrices and allied semantic feature learning with class-imbalance resilient feature engineering and eventual consensus-based prediction could be the prime reason behind such superlative performance. The tools and technologies used in this work are relevant to major analytics tasks, and therefore its implementation for real-time computation could be easier and more scalable. Additionally, the proposed analytics concept address almost major at hand issues of Big data analytics, and therefore it can be applied to any analytics problem. In the future, researchers can use the proposed model for other Bigdata analytics tasks as well.

References

[1] Q. Li and H. Pham, A Generalized Software Reliability Growth Model with Consideration of the Uncertainty of Operating Environments, in IEEE Access,. 7(2019)84253-84267.

[2] Martínez-Fernández et al., Continuously Assessing and Improving Software Quality with Software Analytics Tools: A Case Study, in IEEE Access 7(2019). 68219-68239.

[3] Stapic M. M. Reusability metrics of Software Components: Survey, Conference Paper (2015).

[4] M. Lafi, J. W. Botros, H. Kafaween, A. B. Al-Dasoqi, and A. Al-Tamimi, Code Smells Analysis Mechanisms, Detection Issues, and Effect on Software Maintainability, IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (2019)663-666.

[5] H. Liu, Q. Liu, Z. Niu, and Y. Liu, Dynamic and Automatic Feedback-Based Threshold Adaptation for Code Smell Detection, in IEEE Transactions on Software Engineering, 42(6) (2016)544-558.

[6] Baaba, H. B. Zulzalil, S. Hassan and S. B. Baharom, Software Architecture Degradation in Open-Source Software: A Systematic Literature Review, in IEEE Access, 8(2020) 173681-173709.

[7] F. Palomba, M. Zanoni, F. A. Fontana, A. De Lucia, and R. Oliveto, Toward a Smell-Aware Bug Prediction Model, in IEEE Transactions on Software Engineering, 45(2) (2019)194-218.

[8] J. L. Barros Justo, N. Martinez Araujo, and A. Gonzalez Garcia, Software Reuse and Continuous Software Development: A Systematic Mapping Study, in IEEE Latin America Trans 16(5) (2018)1539-1546.

[9] Diwaker et al., A New Model for Predicting Component-Based Software Reliability Using Soft Computing, in IEEE Access, 7(2019) pp. 147191-203.

[10] M. -C. Chiang, C. -Y. Huang, C. -Y. Wu and C. -Y. Tsai, Analysis of a Fault-Tolerant Framework for Reliability Prediction of Service-Oriented Architecture Systems, in IEEE Trans. on Reliability, 70(1)(2021) 13-48.

[11] S. Maggo and C. Gupta, A Machine Learning-based Efficient Software Reusability Prediction Model for Java-Based Object-Oriented Software, Intl. Journal of Information Tech. And Comp. Sci. 6(2)(2014)1-13.

[12] Sanyam, Prediction of Reusability of Object-Oriented Software Systems using Clustering Approach, World Academy of Science, Engineering and Technology, 43(2010)853-56.

[13] Huda, A. Huneiti and I. Othman, Software Reusability Classification and Predication Using Self-Organizing Map (SOM), Communications and Network, 8(2016)179-192.

[14] P. V. Parande and M. K. Banga, Web-of-Service Software Reusability Prediction using Heterogeneous Ensemble Classifier, Intl. Journal of Innovative Tech. And Exploring Engg., 8(9S3)(2019)1276-82.

[15] N. Padhy, S. C. Stapathy, J. R. Mohanty and R. Panigrahi, Software Reusability Metrics Prediction by Using Evolutionary Algorithms: The Interactive Mobile Learning Application RozGaar, International Journal of Knowledge-based and Intelligent Engg. Sys. 22(4)(2018) 261-276.

[16] R. Panigrahi, S. K. Kunar, L. Kumar, N. Padhy and S. C. Satapathy, Software reusability metrics prediction and cost estimation by using machine learning algorithms, International Journal of Knowledge-based and Intelligent Engineering Systems, 23(4)(2019)317-328.

[17] N. Padhy, R. P. Singh, S. C. Stapathy, Cost-effective and fault-resilient reusability prediction model by using adaptive genetic algorithm-based neural network for web-of-service applications, Cluster Computing, 22(2019)14559-81.

[18] N. Padhy, R. P. Singh, S. C. Stapathy, Enhanced Evolutionary Computing Based Artificial Intelligence Model for Web-Solutions Software Reusability Estimation, Cluster Computing, 22(2019)9787–9804.

[19] L. Gong, S. Jiang, L. Bo, L. Jiang, and J. Qian, A Novel Class-Imbalance Learning Approach for Both Within-Project and Cross-Project Defect Prediction, in IEEE Trans. on Reliability, 69(1)(2020)40-54.

[20] P. R. Bal and S. Kumar, WR-ELM: Weighted Regularization Extreme Learning Machine for Imbalance Learning in Software Fault Prediction, in IEEE Transactions on Reliability, 69(4)(2020)1355-1375.

[21] Singhani H., Suri R. P. Testability Assessment model for Object-Oriented Software Based on Internal and External Quality Factors, Global Journal of Computer Science and Technology: C Software & Data Engineering, 15(5)(2015).

[22] Mijac M., Stapic Z. Reusability Metrics of Software Components: Survey, Central European conf. On Information and Intelligent sys. (2015).

[23] Srivastava S. and Kumar R. An Indirect Method to Measure Software Quality using CK-OO suite, Intelligent Systems and Signal Processing (ISSP), International Conference on, Gujarat, (2013) 47-51.

[24] Goel B.M. and Bhatia P.K. Analysis of reusability of an object-oriented system using CK metrics, International Journal of Computer Applications, 60(10) (2012)0975–8887.

[25] Rosenberg L.H. and Hyatt L.E. Software Quality Metrics for Object-Oriented Environments, Crosstalk Journal, 10(1997)1-16.

[26] Chidamber S.R. and. Kemerer C. F. A metrics suite for object-oriented design, IEEE Transactions on Software Engineering, IEEE Press Piscataway, NJ, USA. 20(1994) 476-493.

[27] Antony P.J. Predicting Reliability of Software Using Thresholds of CK Metrics, Intl. Journal of Advanced Networking &Appl, 4(6)(2013).

[28] Hudiab A., Al-Zaghoul F., Saadeh M., and Saadeh H. ADTEM—Architecture Design Testability Evaluation Model to Assess Software Architecture Based on Testability Metrics, Journal of Software Engineering and Applications, 8 (2015)201-210.

[29] Berander P., Damm L-O-, Eriksson J., Gorschek T., Henningsson K., Jönsson P., Kågström S., Milicic D., Mårtensson F., Rönkkö K. Software quality attributes and trade-offs. Blekinge Instt. of Tech, Blekinge. (2005).

[30] Shatnawi R. A Quantitative Investigation of the Acceptable Risk levels of Object-oriented metrics in open-source systems, IEEE Transactions on Software Engineering, 36 (2010)216-225.

[31] Shatnawi R., Li W., Swain J., and Newman T. Finding software metrics threshold values using roc curves, Journal of Software Maintenance and Evolution: Research and Practice, John Wiley & Sons, Inc. New York, NY, USA. 22 (2010)1-16.

[32] Neelamdhab P., Satapathy S., Singh R. Utility of an Object-Oriented Reusability Metrics and Estimation Complexity. Indian Journal of Science and Technology, 10(3)(2017).

[33] Normi Sham Awang Abu Bakar. The analysis of object-oriented metrics in C++ programs, Lecture Notes on Software Engineering, Springer, 4(1)(2016).

[34] Zahara S. I., Ilyas M., and Zia T. A study of Comparative Analysis of Regression Algorithms for Reusability Evaluation of Object-oriented based Software Components, Open-Source Systems and Technologies (ICOSST), International Conference on, Lahore, (2013) 75-80.

[35] Torkamani M. A. Metric suite to evaluate reusability of software product line, International Journal of Electrical and Computer Engineering (ICE), 4(2) (2014)285-294.

[36] Aloysius A., and Maheswar K. A review on component-based software metrics, Intern. J. Fuzzy Mathematical Archive, vol. 7(2) (2015)185-194. ISSN: 2320 –3242 (P), (2015) 2320 –3250.

[37] Cho E.S., Kim M.S., and Kim S.D. Component metrics to measure component quality, Proceedings of the 8th Asia Pacific Software Engineering Conference (APSEC), Macau,. 4-7 (2001) 419-426.

[38] Canul-Reich J., Shoemaker L., Hall L.O. Ensembles of fuzzy Classifiers, in IEEE International Fuzzy Systems Conference, (2007)1–6.

[39] Rodriguez J.J., Kuncheva L.I. Rotation forest: a new classifier ensemble method, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (10) (2006)1619–1630.

[40] Z. Chun-Xia, Zhang Jiang-She. RotBoost: a technique for combining rotation forest and AdaBoost, Pattern Recognition Letters 29(2008)1524–1536.

[41] Nanni L., Lumini A. Ensemble generation and feature selection for the identification of students with learning disabilities, Expert Systems with Applications 36 (2009)3896–3900.

[42] Zhang X., Wang S., Shan T., Jiao L.C. Selective SVMs Ensemble-driven by Immune Clonal Algorithm, in Rothlauf, F. (Ed.) Proc. of the EvoWork- Shops, Springer, Berlin, (2005) 325–333.

[43] Zhou Z.H., Wu J., Tang W. Ensembling Neural Networks: many could be Better than all, Artificial Intelligence 137 (1–2) (2002) 239–263.
[44] Partalas I., Tsoumakas G., Vlahavas I. Focused Ensemble selection: a Diversity-based Method for Greedy Ensemble selection, in Proceedings of the 18th International Conference on Artificial Intelligence, (2008)117–121.
[45] Dong YS., Han KS. A Comparison of Several Ensemble methods for Text Categorization, Services Computing, (SCC 2004). Proceedings. IEEE International Conference (2004) 419-422.
[46] Roli F., Giacinto G., Vernazza G. Methods for Designing Multiple Classifier Systems. Proceedings of the Second International Workshop on Multiple Classifier Systems. Cambridge, UK, (2001)78–87.
[47] Banfield R. (2007). A Comparison of Decision Tree Ensemble Creation Techniques. IEEE Trans. on Pattern Analysis and Mach. Intell, 29 (2007) 173–180.
[48] Nanni L., Lumini A. Ensemble generation and Feature Selection for the Identification of Students with Learning disabilities, Expert Systems with Applications 36 (2009)3896–3900.
[49] Zhang X., Wang S., Shan T., Jiao L.C. Selective SVMs Ensemble-driven by Immune Clonal Algorithm, in Rothlauf, F. (Ed.) Proc. of the EvoWork- shops, Springer, Berlin, (2005)325–33.
[50] Bi Y., Bell D., Wang H., Guo G., Guan J. Combining Multiple Classifiers Using Dempster's Rule for Text Categorization. Appl. Artif. Intell. 21(2007) 211-239.
[51] Y. An, F. Qin, B. Chen, R. Simon, and H. Wu, OntoPLC: Semantic Model of PLC Programs for Code Exchange and Software Reuse, in IEEE Trans. on Industrial Informatics,17(3) (2021) 1702-1711.
[52] N. V. Chawla, K.W. Bowyer, L. O. Hall, and. P. Kegelmeyer, SMOTE: Synthetic Minority Over-Sampling Technique, J. Artif. Intell. Res., 16(2002)321–357.
[53] He H, Garcia EA Learning from Imbalanced Data. IEEE Trans Knowledge Data Eng 21(2009)1263–1284.
[54] García V, Sánchez JS, Mollineda RA On the Effectiveness of Pre-processing Methods when Dealing with Different Levels of Class Imbalance. Knowl Based Syst. (2012)
[55] Galar M, Fernández A, Barrenechea E, Herrera F EUSBoost: Enhancing Ensembles for Highly Imbalanced data-sets by Evolutionary Undersampling. Pattern Recognit 46 (2013) 3460–3471.
[56] Q. Li, W. Li, J. Wang, and M. Cheng, A SQL Injection Detection Method Based on Adaptive Deep Forest, IEEE Access, 7(2019) 145385-94.