International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Neurocomputing assisted Consensus Based Web-of-Service Software Design Optimization: A Fault-Resilient Reusability Prediction Approach

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

Submitted: 04/02/2024 **Revised**: 12/03/2024 **Accepted**: 18/03/2024

Abstract: In this paper, a state-of-art new neuro-computing assisted consensus-based ensemble model was developed for Web-of-Service (WoS) software reusability prediction. In order to achieve higher accuracy with reliability of prediction, the proposed model made enhancement in both data-model as well as classifier-model. More specifically, it applied WSDL-CKJM tool to extract object-oriented-programming (OOP) metrics, which were subsequently processed using univariate logistic regression-based feature extraction followed by sub-sampling method. In the proposed reusability prediction model, to alleviate data or class-imbalance and skewness problem, three different sub-sampling methods were applied including up-sampling, down-sampling and SMOTE sampling. Once obtaining the differently sampled data with the confidence interval of 95%, it was amalgamated together to give rise a composite feature vector pertaining to WMC, CBO, DIT, LCOM, NOC, and RFC OOP-CK metrics, characterizing structural features of the software program. Subsequently, to alleviate computational overhead Wilcoxon Rank Sum Test (WRST) was applied, which retained the most suitable feature set towards reusability prediction. To alleviate the problem of convergence and over-fitting, Min-Max normalization was performed over the selected feature set. Thus, the normalized input features were processed for two-class classification using the proposed neuro-computing assisted homogenous ensemble model. Noticeably, being homogenous ensemble structure, we used ANN variants with gradient descent (GD), radial basis function (RBF), Levenberg Marquardt (LM) and probabilistic neural network (PNN) as base classifiers. The aforesaid base-classifiers helped in estimating the consensus to make each-class classification, where the proposed consensus-based classification model achieved superior accuracy (96.57%), precision (0.94) and recall (0.99), signifying its robustness over the classical standalone classifiers.

Keywords*: *Web-of-Service Software, Reusability Prediction, Fault-resilience, Ensemble learning, Machine learning, Neurocomputing.

1. Introduction

The high pace emergence in internet and allied software computing technologies have enabled human-life more effective, especially towards real-world decision making. To achieve it, software technology has been playing decisive role and is acknowledged as one of the most vital innovation made across human history to serve varied purposes including science and technology, business, healthcare sector, defense, industries, and varied civic personalized supports. Undeniably, software has become inevitable need of humanity. This at the one hand has broadened the horizon for business communities and engineers to achieve and introduce products or services with better efficacy; however, maintaining their reliability has remained a challenge for all. Moreover, maintaining low-cost solution too has become must for business communities, and therefore firms focus more on reducing costs such as development cost, maintenance cost etc. Considering industrial perspective, software development companies and allied developers often intend to reuse software components of even free-open-source software (FOSS) components

[1]. As a matter of fact, the reuse of existing or pre-employed functions enables a developer or firm to reduce hours of program or allied cost; however, there has been the evidences where the exceedingly reuse of software component caused system failure and loss(es) in terms of finance, time as well as human-life [2][3]. On the contrary, to cope up with competitive cost of the software solution, maintaining lower development is equally significant. It indicates the need of software design with optimal reusability as well as uncompromising reliability [2]. However, in practical world, due to ineffective and inappropriate software design with exceedingly high reuse of the software components, a software turns into faulty, smelling and eventually fails in delivering the expected performance [4]. To alleviate such problem, assessing "software reusability" is of great significance. Software reusability assessment or prediction can enable assessing whether a software program or allied class can be reused anymore or not. Moreover, identifying a class of highly reused it can be rectified to avoid any fault or smell [4]. This as a result can improve reliability of the system. However, manual reusability assessment can be a highly tedious task and even fault-prone, and therefore there is the inevitable need of automated machine learning based software reusability prediction system. Such reusability

[1]*Assistant Professor, Department of Master of Computer Applications, Visvesvaraya Technological University, Belagavi -590018 prakashvp2010@gmail.com*
[2]*Professor, Department of Computer Science & Engg, Dayananda Sagar University, Bangalore - 560076 banga.mkrishna@gmail.com*

assessment models can be vital for small as well as large software programs and allied developers, and would be catalytic for WoS applications. WoS applications often found in different scale with small size computation as well as large real-time computation to serve data mining or knowledge driven services or transactions. While, designing such programs, developer often use FOSS or small chunk of program iteratively to reduce cost. Therefore, the use of machine learning assisted automated reusability prediction model can be vital to ensure highly accurate each-class reusability assessment.

In sync with above discussions, in this paper a state-of-art new and robust machine learning based software reusability prediction model. Realizing the fact that the majority of the software programmes these days are developed using objective oriented programming (OOP) concept, and therefore, the proposed model intends to exploit inter-element (architectural) association-features such as coupling, cohesion, length of program, etc., to perform reusability prediction [1-4]. With this motive, the proposed work intends to use the different OOP metrics including Line of Code (LOC), Depth of Inheritance Tree (DIT), Weighted Method per Class (WMC), Number of Children (NOC), Coupling between Object (CBO), and Lack of Cohesion in Methods (LCOM) [5-8] to perform software reusability prediction. Exploiting the aforesaid OOP features, the proposed machine learning model intends to perform each class classification as REUSABLE and NON-REUSABLE. The overall proposed model encompasses data collection, pre-processing, feature extraction, selection and two-class classification. Being an OOP based model at first, we perform at first Chidamber and Kamrer (CK) metrics estimation, followed by feature extraction and feature selection. As feature extraction model, Univariate Logistic Regression model (ULR) which obtained a total of 22 distinct features. Once obtaining 22 different CK-Metrics, we applied multi-phased feature selection method applying Wilcoxon Rank Sum Test, Information Gain and Gini value, in sequence to perform feature selection. The proposed multi-phased feature selection model obtained a total of six OOP-metrics encompassing WMC, CBO, DIT, LCOM, NOC, and RFC for further processing. Noticeably, since the proposed model intended to perform each-class classification as REUSABLE or NON-REUSABLE, we obtained above stated six features for each class of the considered software program. Considering the probability of class-imbalance, which is highly probable in at-hand case where the presence on Non-Reusable classes can be significantly lower in comparison to the reusable classes, and therefore we performed min-max normalization followed by data sub-sampling. The proposed normalization model intended to alleviate the problem of under-fitting or over-fitting as well as convergence, while sub-sampling method helped alleviating the class-imbalance problem. Thus, performing above stated pre-processing and data sub sampling method (using up-sampling, down-sampling and SMOTE sampling), we obtained a composite feature-set, which was applied for further two-class classification. Unlike classical approaches where single machine learning algorithm is applied to perform classification, in this research paper to improve reliability and accuracy we designed a state-of-art new "Neuro-Computing assisted Consensus based classifier". The proposed consensus classifier mimics maximum voting ensemble [9] with different neural networks such as ANN-GD, ANN-RBF, ANN-LM, and PNN as base classifiers. In the proposed work, the aforesaid base classifiers performed each-class classification as REUSBALE or NON-REUSABLE, which were labeled as 1 or 0. Thus, obtaining labels by each base classifier, we applied MVE concept to achieve consensus for each class, and with the higher label value (either 1 or 0), each class of the program was classified as REUSABLE or NON-REUSABLE. Being a consensus-based approach, the reliability of the proposed reusability prediction model is higher in comparison to the existing state-of-art methods [9][10]. To assess performance, we obtained confusion metrics for each base classifier as well as the proposed Neuro-Computing assisted Consensus based classifier, where the relative analysis was done in terms of accuracy, precision, recall and F-score. The proposed neuro-computing assisted consensus model exhibited better than other-state-of-art reusability prediction techniques. The proposed reusability prediction model can be vital for major WoS oriented software design optimization.

Implementation, while the simulated results are allied inferences are given in Section V. Section VI presents the conclusion, and the references used in this research are given at the last of the manuscript.

2. Related Work

This research focuses on dual objectives; first to exploit state-of-art advanced data model concept to use OOP-CK metrics for class-level reusability prediction, while second it intends to design a state-of-art new ensemble learning concept for higher accuracy and reliability. With this motive, this section discusses some of the key literatures pertaining to reusability prediction and ensemble classifier design to achieve a novel and robust fault-resilient software reusability prediction system.

Authors [11] applied Analytical Hierarchical Process (AHP) to assess varied factors influencing testability of OOP software; however, failed to address the key concerns including class-imbalance, local minima and convergence, which can have decisive impact on the accuracy of the proposed model. Realizing the up-surge of OOP based software design and corresponding CK metrics

characterizing architectural artifacts of the program, authors [12-18] found that LOC, WMC, DIT, NLOC etc. are highly associated with design robustness and fault-probability [19]. Authors [19] too observed that OOP extracted CK metrics can provide better insight to characterize association in between class-reusability and its fault-resilience. In other words, authors [19] suggested that learning OOP-CK metrics can help identifying reusability of a specific class in a software module and its corresponding reliability over unknown-operating period. To ensure fault-resilient software design, authors in [20][21] focused on pre-defining a threshold level for each CK metrics. However, its accuracy and suitability with a large software program could not be examined. Additionally, this research failed addressing above stated class-imbalance problem, which is common in major at-hand scenario where a program can have the major classes as reusable or sometime non-reusable. Thus, training a classical machine learning over such imbalanced data might force it to exhibit false-positive, and hance can impact overall design. Though, the efforts made in [22][23] focused to use the different CK metrics per class to achieve its reusability likelihood for optimal OOP software design; however, could not address aforesaid class-imbalance and convergence problem. Authors [24] applied regression concept to assess reusability of each class in a software where CK metrics were considered as the independent variable while their corresponding reuse-proneness was considered as the dependent variable. Still, it failed to ensure generalization of its solution, as other approaches such as [25][26] performed better with the same OOP-CK metrics-based reusability prediction. Later, authors in [27] stated that the key OOP-CK metrics characterizing complexity, customizability, and reusability can represent quality of software and its fault-resilience. In this reference, they applied interface model with component reuse level (CRL) to measure the extent of reuse-proneness of a class in a software. Authors found that LOC metric can be vital as reuse proneness indicator.

Undeniably, above stated approaches made significant effort to use machine learning methods for reusability prediction; however, failed in addressing accuracy problem, which is must to generalize robustness of a single solution. Considering this fact, enhancement is must not only in data model, but also for classifier design. Unlike standalone classifiers, the concept of consensus based learning, often called Ensemble Learning are found more reliable [9][10][28-45]. Being consensus-based approach (say, decision level fusion), the eventual prediction output is hypothesized to be more reliable than any comprising base classifier or standalone classifier [29]. Amongst the major decision level fusion concepts, maximum voting ensemble (say, consensus or weighed ensemble) yields more reliable classification outputs. Though, later authors [30] suggested performing sub-sampling [31] to achieve better accuracy using AdaBoost; however, its efficacy with highly correlated features and due to large tree construction, it undergoes convergence easily. Though, the concept of data sub-sampling opens up the horizon for data-model optimization to assist better training and class-imbalance alleviation. Neuro-computing ensemble learning was suggested in [32][34][35] as well, where the different neuro-computing algorithms were used as base classifier to perform multi-class classification. Decision level fusion was suggested in [31] as well; however, it was designed towards other classification problem, and had nothing to deal with OOP-CK metrics-based reusability assessment. Though, it indicated that the consensus or MVE can yield higher accuracy with unparallel reliability for any classification problem [36]. In [32][35][44], authors found that though an ideal ensemble learning can be constructed with highly correlated base-classifiers; however, the scope of heterogeneous ensemble can't be ruled out [37]. Still, authors stated that an ensemble learning structure with similar (highly correlated) base classifiers can yield better accuracy. It can eb considered as one of the key driving forces behind this study. Authors [38], designed a support vector machine (SVM) ensemble [40] to perform classification, where a standard SVM algorithm with boosted decision tree were applied as the base classifiers. However, applying MVE ensemble concept, it could yield accuracy lower than 90%, signifying undeniable inferiority towards at hand reusability prediction problem. Though, authors suggested to enhance data model (with better feature sets) as well as classifier model (with better performing base classifiers with MVE) to achieve higher accuracy. In this sync, authors [38] applied principal component analysis (PCA) based feature selection and classified selected features using random forest classifier. To further enhance accuracy, approaches like rotation forest and AdaBoost were applied together to constitute consensus-based classification [39], still its suitability remained unexplored for reusability prediction. To explore efficacy of neural-network based ensemble, authors [42] applied different variants; however, failed to address local minima and convergence problem, which are the key limitations of the neurocomputing based classifiers. Though, authors in [32][34][45] found that the performance of a NN can be improved by using an ensemble of similarly configured NN. It can be considered as one of the key motivations behind this research. Unlike MVE based ensemble, authors [45] applied SVM, k-NN and Rocchio machine learning algorithms with Dempster's rule of decision level fusion to perform classification. However, the maximum accuracy could be confined and hence seems limited towards class-level reusability prediction in large OOP based software solution.

Inference-The above discussion indicates that the depth exploitation of the different OOP-CK metrics can help

assessing relationship in between the different classes, consequently can examine reuse proneness of a class in a software. However, identifying a set of optimal OOP-CK metrics is a problem, which can be solved by certain level-of-significance centric feature extraction and selection. Additionally, the use of sub-sampling concepts and normalization can help alleviating over-fitting, under-fitting, convergence and data imbalance problem. It can help to reduce false-positive and hence can improve the classification accuracy. Moreover, unlike classical standalone classifier-based prediction, the use of homogenous ensemble method or consensus-based prediction can give higher accuracy as well as reliability, which can be vital towards at-hand (class-level) reusability prediction task. These key inferences can be considered as the key driving force behind this study.

3. Research Questions

Considering overall research intends and allied scopes, in this research paper a few questions were identified before implementation. Noticeably, these research questions signify the methodological paradigm to be implemented and its respective possible outcomes. In other words, it states whether the proposed methodology can yield the eventual goal of "fault-resilient reusability estimation and per-class reusability prediction". The key research questions identified are given as follows:

RQ1*: Can the use of CK metrics be effective towards reusability prediction in OOP based software systems?*

RQ2*: Can the use of a multi-phased feature selection method by using Wilcoxon Rank Sum Test, Information Gain and Gini-Index be able to retain most effective and highly accurate OOP-CK metrics for reusability prediction?*

RQ3*: Can the use of sub-sampling methods alleviate the problem of data imbalance or class imbalance to support highly accurate and low false positive rate performance?*

RQ4*: Can the use of Min-Max Normalization help in alleviating the problem of over-fitting or under fitting along with convergence?*

RQ5*: Can the use of a state-of-art new and robust homogenous neurocomputing ensemble method be effective to achieve highly accurate software reusability prediction?*

The overall research intends to achieve answers for the above stated research questions. The affirmative answers for these questions would lead achievement of an optimal model towards highly accurate software reusability prediction.

4. System Model

In this section, the overall proposed fault-resilient software reusability prediction model using neuro-computing assisted consensus model is discussed. The overall proposed reusability prediction model has been accomplished in six subsequent (say, phase-wise) processes. These are:

4.1 Data Preparation

As already stated in the previous section, this research primarily focuses on designing a state-of-art new and robust software reusability prediction model for OOP-based WoS software solution. Therefore, we selected an arbitrary software module from www.sourceforge,com , which is developed using OOP programming concept in Java Language. In order to convert an OOP based Java program and retrieve targeted Chidamber and Kamrer software metrics, we applied Web-of-Service Data language (WSDL) tool. Typically, WSDL represents the XML-based interface definition language characterizing the different functions or components of the web services. Here, each function or allied functionalities used to be the component considered as a port type i.e., $P = \{M_0(I_0, O_0), M_1(I_1, O_1), \ldots\ldots, M_n(I_n, O_n)\}$. Practically, these functions or ports perform different tasks M_i by transferring input I_i into output O_i. In this mechanism, the functional component along with its port-type can be characterized in the form of a unique nomenclatures. These functional components often encompass instructions or the specific elements to perform data-exchange between the service provider and user or consumers. Moreover, individual data element states specific categorical definitions, which is defined in terms of XML, while XML is stated in the form of XML Schema Definition (XSD) language representing the data type definition. Thus, the associated activities including string, integer, restrictions, encapsulation and extension are employed to represent the complex software structure. Thus, the use of WSDL data enabled retrieval of the XSD code by using type's element. Moreover, XSD code was stored into a separate file which was connected to the WSDL document so as to obtain the type reuse. In the proposed data collection method, at first the services were coded, which was followed by conversion of codes into the corresponding WSDL document. To obtain SDL document or values from OOP based java programs other tools such as Java2 WSDL, and SOAP can be applied. Moreover, Apache CXF, eclipse Spring-Tool-Suite, Soap UI, and WSImport too can be applied to convert WSDL document into Java file for further software metrics estimation. In this research, we applied "WSImport" tool to convert WSDL documents into OOP-java file, which was later processed using CKJM (Chidamber and Kamrer Java Machine) tool [16]. CKJM tool helped extracting the different software metrics characterizing software structure and corresponding information including coupling, cohesion, complexity etc. The overall implementation schematic for the proposed data collection and CK metrics extraction is given in Fig. 1.

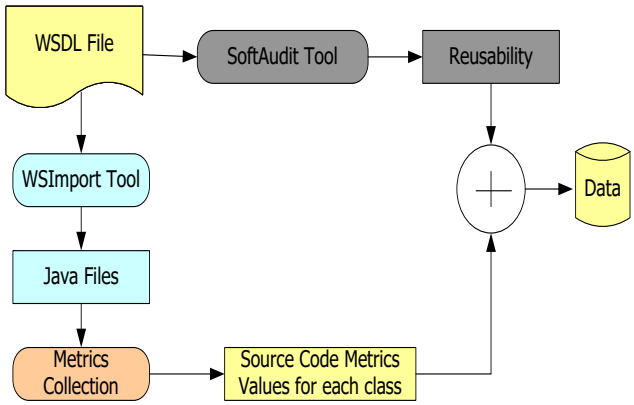


Fig. 1 Data Preparation for the proposed use case-study

Though, the use of CKJM tool enables extracting a total of 22 software (OOP) metrics; however, in the proposed work we extracted a total of 17 software metrics given as follows:

- *Weighted Methods per Class (WMC),*
- *Depth of Inheritance Tree (DIT),*
- *Number of Children (NOC),*
- *Coupling between Object Classes (CBO),*
- *Response for a Class (RFC),*
- *Lack of Cohesion in Methods (LCOM),*
- *Afferent Couplings (Ca),*
- *Efferent Couplings (Ce),*
- *Number of Public Methods (NPM),*
- *Data Access Metric (DAM),*
- *Measure of Aggregation (MOA),*
- *Measure of Functional Abstraction (MFA),*
- *Cohesion Among Methods of Class (CAM),*
- *Cyclomatic Complexity (CC),*
- *Lines of Code IC- Inheritance Coupling (LOC),*
- *Coupling Between Methods (CBM), and*
- *Average Method Complexity (AMC).*

Thus, once obtaining the above stated software metrics, we further processed for feature extraction using Univariate Logistic Regression. Though, the CKJM metrics can directly be processed for further computation such as pre-processing and classification; however, realizing the fact that in a software program not each class or component can have the equal significance towards reusability prediction, and therefore in the proposed work, we applied ULR based feature extraction followed by multi-phased feature selection. The details for the same is given in the sub sequential sections.

4.2 ULR based Feature Extraction

Logistic regression analysis typically performs statistical assessment by using dependent (here, per-class software reusability) as well as independent (software metrics) variables. Being a two-class problem; REUSABLE or NON-REUSABLE, the dependent variable serves two labels or values 1 or zero, signifies REUSABLE CLASS and NON-REUSABLE CLASS, respectively. In our proposed method, we obtained the level of significance of each metrics towards software reusability prediction. Mathematically, we use (1) to estimate logistic regression value.

$$logit[\pi(x)] = \alpha_0 + \alpha_1 X \qquad (1)$$

In (1), the function logit[π(x)] states the dependent variable while X presents the independent variable. The parameter π signifies the likelihood factor signifying the importance of each metrics. Mathematically we estimate $\pi(x)$ as per (2).

$$\pi(x) = \frac{e^{\alpha_0+\alpha_1 X}}{1 + e^{\alpha_0+\alpha_1 X}} \qquad (2)$$

In synch with our proposed software reusability prediction purpose, let the data be X that possesses N rows and $M + 1$ columns, where M signifies the number of independent variables for each raw signifying software metrics. Let the parameter vector, β be a column vector of length $K + 1$. Additionally, there is single parameter pertaining to each M columns of the independent variable. Thus, applying logistic regression function also called $Logit$ function we obtained the log-odds of the likelihood of success to the linear component. Mathematically,

$$Logit\left(\frac{\theta_i}{1-\theta_i}\right) = \sum_{m=0}^{M} x_{im}\beta_m \qquad i = 1,2,\dots,N \qquad (3)$$

In (3), $\left(\frac{\theta_i}{1-\theta_i}\right)$ states the component called odds-of-an-event. Now, let y takes a value 1 for REUSABLE and 0 for NON-REUSABLE, y can be stated to have a Bernoulli distribution with a probability parameter p. Thus, obtaining the probability parameter, also called p-value for each instance, we select the one with $p \geq 0.05$. Thus, implementing ULR, out of 17 software metrics, we obtained top-6 software metrics having higher significance towards reusability prediction. The selected finally software metrics were WMC, CBO, DIT, LCOM, NOC, and RFC, which were used for further processing.

4.3 Data Sub-Sampling

This is the matter of fact that in a software the probability of non-reusable or even reusable classes can be non-evenly distributed. In other words, a software can have very less non-reusable class as well or vice versa. Such probability can have the class-imbalance problem characterizing either majority of reusable class or non-reusable class. Thus, training a machine learning model with such class-imbalanced dataset often leads false-positive (prediction) result. This as a result can affect the accuracy of the prediction system and its reliability. Considering such class-imbalance or data-imbalance problem, in this paper we applied data-sub-sampling concept using UP-Sampling, Down-Sampling and Synthetic Minority Oversampling Technique (SMOTE) [47][48]. Noticeably, to alleviate the

key problem of data-skewness [46], in this paper three different sub-sampling methods have been applied whose respective outputs are concatenated together to generate a final feature vector for further computation.

To perform up-sampling and down-sampling we considered the confidence level of 95%. In up-sampling we performed random duplication of the observations or patterns from the minority classes so as to reinforce its value. On the other hand, down-sampling was performed in such manner that it removes observations randomly from the majority class so as to avoid its presence from dominating the learning model. Undeniably, performing over-sampling of the minority class or under-sampling of the majority class can't alleviate data-skewness or class-imbalance problem completely, as it eventually turns into bias of the model towards the majority class [49][50]. In such conditions, when new sample comes into the learning model, it is finally predicted as majority class due to bias towards the majority class [51][52]. Such limitations can force most of the machine learning to yield false prediction output(s) and can reduce accurate of the prediction model. Considering this fact, in addition to the up-sampling and down-sampling we applied SMOTE technique. In our proposed SMOTE model, we generated synthetic positive samples using K-nearest neighbor (k-NN) algorithm. Here, we employed 5-Nearest Neighborhood to the minority "NON-REUSABLE" class, which was followed by equalization of the samples in such manner that it yields the number of majority class same as the number of minority class. Thus, performing above stated three sampling techniques we obtained Up-Sampled Data, Down-Sampled Data, SMOTE Data, which were combined or concatenated together to result a final feature set for further computation.

4.4 Wilcoxon Rank Sum Test based Feature Selection

This is the matter of fact that the use of data sub-sampling can alleviate the problem of data skewness or data-imbalance; however, the use of Up-Sampling, Down-Sampling and SMOTE altogether generates a significantly large dataset, which can force a machine learning model to undergo pre-mature convergence and hence can affect overall prediction or classification accuracy [53]. Considering this problem, in this paper Wilcoxon Rank Sum Test (WRST) based feature selection method, which is also called significant predictor test has been applied. WRST is a type of non-parametric test with independent samples. Functionally this method examines the correlation between the variables (software metrics or CK-metrics WMC, CBO, DIT, LCOM, NOC, and RFC) and their significance towards reusability prediction accuracy. Here. WRST algorithm estimates correlation between or amongst the software metrics per class and corresponding feature values towards reusability likelihood. Here, we considered different software metrics (i.e., WMC, CBO, DIT, LCOM, NOC, and RFC) as independent variable while the reusability likelihood was considered as dependent variable. This method obtained p-value for each feature variable in reference to its significance towards reusability prediction. Thus, based on the p-value each feature element was labeled as significant or insignificant. Since, we considered $p = 0.05$ as the level of significance, the data instance having p-value higher than the level of significance were retained, while rest were dropped for further computation.

4.5 Min-Max Data Normalization

This is the matter of fact that in major classification or prediction systems, especially in large features-based models data imbalance and convergence are the key problems, which hinder the overall performance of the system. Post feature extraction and selection the retrieved data elements are of the different size and range and hence computing over such unstructured data can force learning model to undergo pre-mature convergence and even over-fitting. It can affect overall computational efficiency (i.e., accuracy and reliability) and therefore to alleviate it, we performed Min-Max normalization over the retained significant features. The proposed Min-Max normalization algorithm, as indicated in (4) mapped or normalized feature values in the range of 0 to 1. This method linearly transformed and mapped the input features in the range of [0, 1]. Functionally, each data element x_i of the selected features X was mapped to the corresponding normalized value x_i' in the range of [0, 1]. We used (4) to estimate normalized value(s) of the input data x_i .

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

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

4.6 Neuro-Computing assisted Consensus based Reusability Prediction

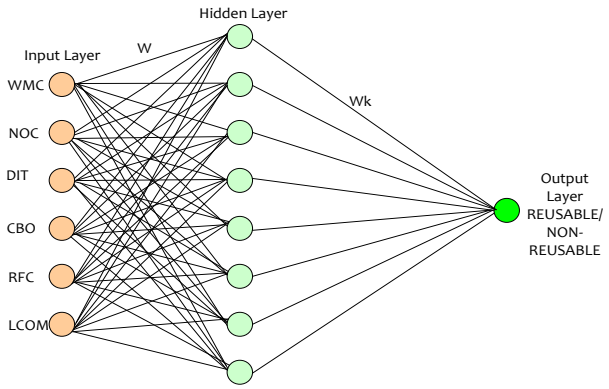
Amongst the major machine learning algorithms, neural network has been applied extensively towards data learning and classification purposes [32][34]. The robustness of ANN makes it efficient to be used in diverse classification problems, though based on computational complexities and adaptive computation ANN has evolved through different phases. Exploring in depth it can be found that the performance of ANN is directly related to the corresponding learning method. Thus, based on learning method, ANN has been evolved as ANN with steepest gradient (SD), ANN with gradient descent (GD), ANN with RBF (ANN-RBF), ANN with Levenberg Marquardt (ANN-LM), Probabilistic Neural Network (PNN), etc. However, in sync with non-linear heterogeneous data classification ANN-GD, ANN-RBF, ANN-LM and PNN have performed

well. Unlike ANN-SD, ANN-GD avoids local minima and convergence issue, even with large non-linear feature set. Similarly, ANN-LM possesses higher robustness than ANN-SD and ANN-GD, individually. Moreover, ANN-LM can be configured to possess feature of ANN-SD as well as ANN-GD and therefore has better performance stability even with large, non-linear and heterogeneous data. However, in literatures ANN variants are found performing distinctly with the different accuracy towards any classification problem. Consequently, it makes ambiguity in selecting the best neurocomputing algorithm. On the other hand, the recent literatures [32][34] revealed that unlike standalone classifier, ANN based ensemble can give more accurate and reliable prediction result [54]. Considering this fact, in this paper we applied different ANN variants or algorithms as base-classifier to design an MVE ensemble learning model, also called consensus learning model to perform reusability prediction. More specifically, in the proposed ensemble learning model, we applied the following key classifiers as base classifiers.

1. *ANN-GD,*
2. *ANN-RBF,*
3. *ANN-LM (with One and Two Hidden Layers), and*
4. *PNN.*

Thus, the use of above stated neurocomputing model with four ANN variants constitutes a homogenous ensemble structure to perform consensus-based prediction. Noticeably, being base classifiers, these ANN variants functions independently and classifies each class as REUSABLE or NON-REUSABLE, and subsequently labels each class as 1 and 0, respectively. The voting per class by each base classifier has been used to estimate consensus or the maximum voting score. Thus, the higher score label (0 or 1) has been applied to predict reusability probability of that specific class.

The functional architecture of the applied ANN model with the input layer, the hidden layer and the output layer is given in Fig. 2. Functionally, it embodies multiple neurons representing the input data (or the CK metrics) that are processed at the distinct intermediate layers (say, hidden layers) for two-class classification (at the output layer). To learn over the input data, ANN applies error-reduction method, where during learning it estimates the difference between the expected output and the observed output (signifying error). The learning process continues till the error output becomes zero or near zero. Thus, achieving zero-error the outputs at the output layer is predicted as the final output. Considering the at-hand problem of link-prediction, ANN is expected to perform two-class classification at the output layer. At the input layer of the ANN, it applied linear activation function, which generates output same as the input (i.e.,$O_o = I_i$), while the output of the hidden layer is fed to the input of the output layer. Noticeably, output layer of the ANN applies Sigmoid function (5) to generate O_h.



5. Neuro-computing Based Reusability prediction

$$O_h = \frac{1}{1 + e^{-I_h}} \tag{5}$$

In (5), I_h represents the input at the hidden layer. ANN is often defined as $Y' = f(W, X)$ where Y′ states the output vector, while X and W presents the allied input and the weight values, respectively. Functionally, ANN applies certain error function such as mean square error (MSE) to achieve the higher accuracy, which is estimated using (6).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i' - y_i)^2 \tag{6}$$

In (6), y presents the observed output value, while the expected value is y_i'. As stated above, the key difference between the different ANN variants is the way it schedules or updates its weight values over training. A snippet of the different ANN variants (i.e., ANN-GD and ANN-LM) is given as follows.

a. ANN-GD

Let the regression for the learning method, while reducing error value be (7).

$$w^* = \underset{w}{argmin}\, L(w) \tag{7}$$

$$L(w) = \sum_{t=1}^{N} L\big(y_t, f_w(x_t)\big) + \lambda R(w) \tag{8}$$

In ANN-GD setup, $f_w(x)$ factor states the non-linear weight w, and thus it intends to achieve a local optimum for (8) using GD method, which updates w iteratively by updating w_t by w_{t+1}.

$$w_{t+1} = w_t - \eta_t \nabla L \tag{9}$$

$$w_{j,t+1} = w_{j,t} - \eta_t \frac{\partial L}{\partial w_j} \tag{10}$$

In (9), the parameter ∇Lsignifies the error value, which is mathematically given as (11).

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

In (10) η_t states the learning rate, which reduces over time t. Thus, performing GD based weight estimation and learning it classifies each software component or class as REUSABLE or NON-REUSABLE, and labels each class as 1 and 0, respectively.

b. ANN-RBF

Similar to the standard ANN algorithm, ANN-RBF too encompasses input layer, hidden layer and output layer; however, unlike classical methods the neurons in the hidden layer contains Gaussian transfer functions whose outputs are reverse proportional to the distance from the center of the neuron. Functionally, ANN-RBF is equivalent to K-Means clustering concepts and Probabilistic Neural Network (PNN); however, the prime disparity is that the other methods such as PNN has single neuron for each data-point (in training), while ANN-RBF comprises multiple neurons (but lower as compared to the number of training points). Though, for a low or medium size dataset, PNN can be appropriate; however, its efficiency remains confined over large, non-linear input pattern. Considering this motive, we applied ANN-RBF as one of the base classifiers to perform (each-class) software reusability prediction. In the proposed ANN-RBF model, the hidden units enable a set of functions comprising random basis for the input patterns. In this approach the hidden units are called as the radial centers which refer a vector $c_1, c_2, \dots, c_h$.

In ANN-RBF, the input features, often called input-space is transferred into hidden space by means of non-linear transformation method. The transformation from the hidden unit space to the output space remains the linear dimension of each centre for $n-$point input network (i.e., $n \times 1$ dimension). In the proposed ANN-RBF, the RBF in the hidden layer generates non-zero response and each hidden unit and contains its own receptive field in the input space. Here, x_i be the input vector present in the receptive field for center c_j, c_j, is activated by selecting proper weights to achieve expected target output (12).

$$y' = \sum_{j=1}^{h} \emptyset_j w_j, \quad \emptyset_j = \emptyset\left(\|x - c_j\|\right) \tag{12}$$

In (12), w_j states the weight of the $j-$th center, while $\emptyset$ represents the radial function. We applied GD weight update method for ANN-RBF to learn over the input features or the OOP-CK metrics to classify each class as REUSABLE or NON-REUSABLE.

c. ANN-LM

ANN-LM is stated to be the most efficient neurocomputing algorithm because of its ability to exploit efficacy of both ANN-SD as well as ANN-GD. Additionally, better learning and allied weight update mechanism enables ANN-LM to exhibit higher accuracy amongst the other variants. Functionally, ANN-LM performs localization of the minimum value of the multivariate function, called Sum of Squares (SoS) of the non-linear real-valued functions. It strengthens ANN-LM algorithm to perform swift weight estimation (13) and tuning, and thus helps in achieving higher accuracy even without undergoing local-minima and convergence, easily. It makes ANN-LM robust to be used for large feature learning for classification. ANN-LM applies equation (13) to perform weight update for efficient learning.

$$W_{j+1} = W_j - \left(J_j^T J_j + \mu I\right)^{-1} J_j e_j \tag{13}$$

In (13), W_j states the present weight while W_{j+1} signifies the updated weight. Ipresents the identity matrix, while the Jacobian matrix is calculated as per (14). In (13), the learning parameter μ signifies the combination coefficient. Typically, the low value of μ triggers ANN-LM to behave as ANN-GD, while higher value forces to act as ANN-SD.

$$\mathrm{J} = \begin{bmatrix} \frac{d}{dW_1}(E_{1,1}) & \frac{d}{dW_2}(E_{1,1}) & \cdots & \frac{d}{dW_N}(E_{1,1}) \\ \frac{d}{dW_1}(E_{1,2}) & \frac{d}{dW_2}(E_{1,2}) & \cdots & \frac{d}{dW_N}(E_{1,2}) \\ \vdots & \vdots & & \vdots \ \vdots \\ \frac{d}{dW_1}(E_{P,M}) & \frac{d}{dW_2}(E_{P,M}) & \cdots & \frac{d}{dW_N}(E_{P,M}) \end{bmatrix} \tag{14}$$

In (14), N states the total weight counts and P presents the input features. The output is given by M. In our proposed work, ANN-LM was designed with two distinct structures, one comparison single hidden layer, while another model was designed with two hidden layers. Thus, a total of two ANN-LM models with 1 and 2 hidden layers were used as the base classifiers.

Undeniably, above discussed neurocomputing models or ANN variants (ANN-GD, ANN-RBF and ANN-LM) have played vital role towards data science purposes, such as classification or identification. However, iterative learning and weight calculation confine their robustness and imposes significantly large computation. To alleviate such problems, recently a new variant named Probabilistic Neural Network (PNN) has been proposed. In our proposed HEL model, we applied PNN as one of the base classifiers. The details of PNN are given as follow.

d. Probabilistic Neural Network (PNN)

Probabilistic Neural Network (PNN) represents a variant of feed forward neural network, often used for classification problems. In our proposed PNN model, the parent probability distribution function (PDF) of each subject-class or patient is approximated by means of a Parzan window and a non-parametric function. Subsequently, employing PDF of each subject class label, the likelihood of a new input is obtained as per Bayes rule. Here, the use of Bayes rule helps in assigning class the highest posterior probability to the new input. This method reduces probability of mis-detection or mis-classification significantly. Structurally, PNN model was derived from Bayesian network in conjunction with Kernel Fisher discriminant analysis. It was designed as a multi-layered feed forward network with four layers; input later, pattern layer, summation layer and output layer. Here, the first layer estimates the distance from the input vector to the training input vector. Consequently, it generates a vector where each element signifies how close the input is to the training input. Similarly, the second layer performs summation of the contribution of each class-label of the input and eventually generates its output as a probability vector. Eventually, a transfer function is applied on the output of the second layer and thus selects the maximum of the probability vector and generates 1 for positive class (i.e., REUSABLE) and 0 for negative (i.e., NON-REUSABLE). The detailed discussion of the PNN architecture and its function is given as follows.

We designed three-layered PNN architecture with input layer, hidden layer and output layers. The overall model was designed to perform two-class classification (i.e., K=2). The input layer comprised six nodes each carrying WMC, CBO, DIT, LCOM, NOC, and RFC metrics, distinctly. These are the fan-out nodes that split or branch each subject-feature to nodes in all hidden layers, so as to ensure that each hidden layer received the complete subject-features (say, feature vector x). Here, the hidden layer's nodes are transformed into groups, one group for each class of target category. In the proposed design each hidden node in a group for class K belongs to a Gaussian function, centered on corresponding feature vector in the kth class. The comprising Gaussians in a class group feed their respective functional values to the same output layer node for that class. This as a result generates K output nodes. At the output layer or output node for class K (say, REUSABLE "1" and NON-REUSABLE as "0"), all the Gaussian values for that class K are aggregated. Subsequently, the sum is further scaled in such manner that the probability value under the sum function remains unity (it constitutes PDF). Let P be the feature vector such that $\{x^{(p)}: p = 1, \dots, P\}$, which is labelled as Class 1 (i.e., REUSABLE). Similarly, let Q be the feature vector $\{y^{(r)}: r = 1, \dots, R\}$ to be labeled as class 2 (i.e., NON-REUSABLE). Thus, in the hidden layer of the PNN there would be P nodes in the group for Class 1 and R-nodes in the group for class 2. The mathematical model for each Gaussian centered on the corresponding class 1 and class 2 point be $x^{(p)}$ and $y^{(q)}$ (it signifies feature vector for N-dimensional vector) for any input vector x be (15) and (16).

$$g_1(x) = \left[\frac{1}{\sqrt{2\pi\sigma^{2N}}}\right] exp\left\{-\frac{-\|x - x^{(p)}\|^2}{(2\sigma^2)}\right\} \quad (15)$$

$$g_2(y) = \left[\frac{1}{\sqrt{2\pi\sigma^{2N}}}\right] exp\left\{-\frac{-\|y - y^{(q)}\|^2}{(2\sigma^2)}\right\} \quad (16)$$

The value of σ is taken to be 50% of the average distance between the feature vectors in the same group. The kth output node summarizes the values received from the hidden nodes in the k-th group, which is also called Parzen Window or the mixed Gaussian. We defined sum as (17) and (18).

$$f_1(x) = \left[\frac{1}{(2\pi\sigma^2)^N}\right]\left(\frac{1}{P}\right)\sum_{(p-1,P)} exp\left\{-\frac{-\|x - x^{(p)}\|^2}{(2\sigma^2)}\right\} \quad (17)$$

$$f_2(y) = \left[\frac{1}{(2\pi\sigma^2)^N}\right]\left(\frac{1}{Q}\right)\sum_{(q-1,Q)} exp\left\{-\frac{-\|y - y^{(q)}\|^2}{(2\sigma^2)}\right\} \quad (18)$$

In (17-18), x represents an input feature vector, σ represents the standard deviation for Gaussians (in Class 1 and Class 2), N =6 presents the input vector dimension, P be the number of center vectors in Class 1, and R be the number of centers in Class 2. $x^{(p)}$ and $y^{(r)}$ be the centers in the corresponding classes 1 and 2. The nominator component $\|x - x^{(p)}\|^2$ presents the Euclidean distance between $x$and $x^{(p)}$. Here, any input vector x is put-through both sum functions $f_1(x)$ and $f_2(y)$ and therefore the highest value of $f_1(x)$ and $f_2(y)$ decides the class. Thus, implementing PNN we performed two-class classification which labelled each subject as "REUSABLE" or "NON-REUSABLE".

- *Consensus based Ensemble Learning*

The above discussed machine learning algorithms were applied as base classifier to perform two-class classification. Each of the classifier labeled each class as REUSABLE ("1") or NON_REUSABLE ("0"). Thus, obtaining the classified label for each class by all five base classifiers, we applied MVE ensemble concept also called consensus-based ensemble model. In this approach, the higher prediction output (1 or 0) by each of the base classifier was considered as the final prediction result. In other words, if for a specific class in the software, in case out of five base classifiers (i.e., ANN-GD, ANN-RBF, ANN-LM (with one hidden layer), ANN-LM (with two hidden layers) and PNN), if three classifiers predict a class as "1", then the

proposed consensus-based ensemble model predicts that specific class as REUSABLE. On the contrary, in case three base classifiers predicts a class as "NON-REUSABLE". Thus, unlike classical standalone classifier-based prediction, the proposed consensus-based approach employs maximum votes and hence has higher reliability towards software component reusability prediction.

6. Results and Discussion

In this paper, a novel and robust neurocomputing assisted homogenous ensemble learning model was developed for per-class software reusability prediction in OOP based WoS software solutions. Unlike classical approaches, in the proposed method, the key emphasis was made on improving both data model as well as classification model so as to achieve higher accuracy. Realizing the fact that the suitability and optimality of data often impacts eventual classification output, at first, we focused on improving dataset while alleviating classical problems of data imbalance, local minima and convergence etc. Considering OOP based software design demands, we considered open to access software available at www.sourceforge.com. Noticeably, the considered data was a WoS software developed using OOP concept. Thus, obtaining the raw software (for the present case study), at first, we applied WSDL tool in conjunction with CKJM that helped extracting a total of 17 OOP-CK metrics. However, realizing the fact that not all metrics can have the decisive significance towards reusability prediction, and therefore we applied ULR based feature extraction followed by WRST based significant predictor test. For WRST, we assigned level of significance P=0.05. Thus, performing feature selection a total of six OOP-CK metrics were obtained. These were WMC, CBO, DIT, LCOM, NOC, and RFC. Subsequently, realizing the likelihood of data-imbalance, we performed data sub-sampling using up-sampling, down-sampling and SMOTE sampling. Noticeably, we performed sub-sampling with the confidence interval of 95%. Once obtaining different data samples, we concatenated together to constitute All Matrix, though for comparison we considered up-sample data, down-sampled data, and SMOTE sampled data as well for which we performed each class classification. In other words, we performed two-class classification over each sample (i.e., up-sampled, down-sampled, SMOTE sampled and All Matrix sample). Here, our prime motive was to assess whether performing different sampling method can help achieving higher accuracy or not. The accuracy precision, recall and AUC performance over the different data samples are given in Table 1 to Table 4. To further improve the performance we performance, we applied Min-Max normalization over the sampled data which mapped each data element or metrics values in the range or [0-1]. Here, the key intend was to alleviate any probability of premature convergence and over-fitting problem. Thus, once performing data-normalization the OOP-CK metrics (i.e., selected features of WMC, CBO, DIT, LCOM, NOC, and RFC) were fed as input to the different neurocomputing algorithms. Noticeably, being a homogenous ensemble model, we designed ANN-GD, ANN-RBF, ANN-LM (with one hidden layer), ANN-LM (with two hidden layers) and PNN as base classifiers. Here, each base classifier performed two-class classification and perform per-class or each-class classification as REUSABLE or NON-REUSABLE, and labeled each class as 1 and 0, respectively. Thus, performing each-class classification by all five base classifiers we applied MVE ensemble concept to obtain consensus (for each class) were a class with three "1's" was predicted as REUSABLE, while a class with three "0's" was classified or predicted as NON-REUSABLE. Thus, applying this method, we performed eventual software reusability prediction. Being a consensus-based approach, the accuracy and reliability of the proposed model can be hypothesized to be higher.

Being a classification problem, to assess performance by the proposed model, we obtained confusion matrix by each classifier, over each sample data (up-sampled, down-sampled, SMOTE and All Matrix sampled data). The confusion matrix was obtained in the form of true positive (TP), true negative (TN), false positive (FP) and false negative (FN). Retrieving these matrix values for each base learner as well as ensemble classifier, the performance has been obtained in terms of accuracy, precision and recall, also called sensitivity. The definitions of these performance variables are given in Table 2.

Table 1. Performance Parameters

Parameter	Mathematical Expression	Definition
Accuracy	$\frac{(TN+TP)}{(TN+FN+FP+TP)}$	Signifies the proportion of predicted fault prone modules that are inspected out of all modules.

Precision	$\frac{TP}{(TP+FP)}$	States the degree to which the repeated measurements under unchanged conditions show the same results.
Recall	$\frac{TP}{(TP+FN)}$	It indicates how many of the relevant items are to be identified.

Table 2 presents the accuracy performance by the different base classifiers as well as proposed neurocomputing assisted consensus ensemble learning model. Considering the results obtained, it can easily be visualized that amongst the different neuro-computing models and the derived consensus-based ensemble learning concept, the proposed maximum voting-based ensemble exhibits the highest classification accuracy (93.57%). Noticeably, the proposed MVE ensemble or consensus-based ensemble achieves the highest accuracy of 96.57% with All-Matrix. It justifies that the sub-sampling of the input features can help not only to alleviate data imbalance, but can also enable higher accuracy. The relative performance shows that after the proposed MVE ensemble of consensus-based ensemble model, ANN-LM (with two hidden layers) performs better, and therefore as standalone solution it can be considered as the potential (say, sub-optimal) classifier. However, the superiority of the proposed MVE based consensus ensemble classifier confirms its suitability towards at hand software reusability prediction. The other base-classifiers are found relatively low accurate. Thus, the results obtained (Table 2) affirms that the use of data sub-sampling and composite feature learning can be optimal towards reusability prediction where MVE based neurocomputing (homogenous) ensemble classifier can be most effective approach to perform reusability prediction. It affirms acceptance of RQ3. Additionally, the other research questions such as RQ1 and RQ2 too can be justified affirmatively as the proposed software metrics have yield high accuracy with manual verification similarity.

Table 2 Accuracy (%) Performance

Sampled Data	**ANN-GD**	**ANN-RBF**	**ANN_LM (1-Hidden Layer)**	**ANN-LM (2 Hidden layer)**	**PNN**	**MVE**
Original Data	92.01	93.17	95.15	95.46	**94.21**	89.31
Up-Sampled Data	95.31	94.61	94.80	95.64	96.1	91.94
Down-Sampled Data	94.95	95.46	95.5	95.64	95.7	91.00
SMOTE Sampled Data	89.17	89.10	92.3	91.21	91.4	91.04
ALL Matrix	95.09	95.13	95.5	95.47	93.6	**96.57**

Table 3 Precision Performance

Samples	ANN-GD	ANN-RBF	ANN_LM (1-Hidden Layer)	ANN-LM (2 Hidden layer)	PNN	MVE
Original Data	0.91	0.91	0.93	0.91	0.93	0.89
Up-Sampled Data	0.92	0.93	0.93	0.91	0.92	0.88
Down-Sampled Data	0.91	0.92	0.92	0.92	0.93	0.88
SMOTE Sampled Data	0.83	0.83	0.87	0.87	0.88	0.89
ALL Matrix	0.91	0.92	0.92	0.92	0.91	**0.94**

Table 3 presents the precision performance by the proposed model. Noticeably, the highest precision obtained by the proposed model was 0.94, amongst all classifiers, especially with the All-Matrix features. It indicates that similar to the accuracy performance the proposed MVE ensemble (neurocomputing assisted consensus (homogenous) ensemble) model shows higher precision and unchanged performance over changed samples or environment. It confirms reliability and robustness of the proposed consensus-based ensemble learning model towards software reusability prediction. The results obtained (Table III) too confirms acceptability of RQ2, RQ3, RQ4 and RQ5.

Table 4 Recall Performance

Samples	ANN-GD	ANN-RBF	ANN_LM (1-Hidden Layer)	ANN-LM (2 Hidden layer)	PNN	MVE
Original Data	0.93	0.93	0.94	0.98	0.96	0.98
Up-Sampled Data	0.93	0.92	0.94	0.98	0.95	0.97
Down-Sampled Data	0.92	0.94	0.95	0.99	0/96	0.97
SMOTE Sampled Data	0.94	0.94	0.97	0.96	0.97	0.81
ALL Matrix	0.96	0.97	0.97	0.99	0.98	**0.99**

In Table 4, the recall performance by the proposed model is presented. Noticeably, recall is also called as sensitivity. Observing the results (Table 4), it can be found that the proposed neurocomputing assisted homogenous ensemble model achieves the highest recall or sensitivity of 0.99. It indicates higher sensitivity of the proposed model, which could be contributed due to better feature learning ability. Noticeably, the higher values of precision and recall indicates higher F-score and thus, signifies robustness of the proposed model to achieve higher superior performance even over large non-linear features.Thus, observing overall performance and allied inferences, it can be stated that the proposed neuro-computing assisted consensus (or ensemble) learning model achieves higher accuracy, precision and recall (or sensitivity), signifying robustness of the proposed model towards WoS software reusability prediction. The research questions as defined in Section 3 are found to be affirmatively accepted, and hence affirms suitability of the proposed (homogenous) ensemble learning model towards software reusability prediction. Though, the

role of the strategic model with WSDL-CKJM based OOP metrics estimation, followed by ULR based feature extraction and WRST based feature selection and sub-sampling can't be ignored. The improved data model can have the decisive impact in achieving above discussed superior performance. Additionally, the robustness of the proposed neurocomputing assisted homogenous ensemble model with MVE or consensus ensemble helped achieving the most efficient performance towards software reusability prediction. Thus, the research questions as defined in Section 3 (RQ1-RQ5) are answered affirmatively.

7. Conclusion

Considering the fact that the reuse of a software component can at one hand reduce development cost; however, the excessive reusability of the same component or class can force a software solution to undergo aging, smells and sometime failure, in this paper a software metrics learning based reusability prediction model was developed. Contemporary, due to ease of implementation and agile in nature OOP based software development is prevalent and has been applied in major WoS software development that often undergoes exceedingly high transaction and dynamic computing. In such application environment excessive use of software components or improper reusability might limit the reliability of the software solution. Therefore, for an optimal OOP WoS software design, assessing component reusability can be of great significance to alleviate any future faults or failure. With this motivation, in this research paper at first OOP-CK metrics were obtained which characterizes the structural features such as cohesion, coupling and complexity of a software program to assess reuse-proneness of a component. Unlike manual testing methods, in this paper a state-of-art new neurocomputing assisted consensus-based ensemble learning model was developed which exploits OOP-CK metrics, specially WMC, CBO, DIT, LCOM, NOC, and RFC to perform each-class reusability prediction. To achieve an accurate and reliable prediction solution, in this paper the optimization efforts were made for both data model as well as classifier model. Once obtaining aforesaid OOP-CK metrics using WSDL-CKJM tool, the extracted features were further processed using ULR algorithm which retained a total of six OOP-CK metrics to be used for further computation. Additionally, to alleviate the problem of data-imbalance and skewness, the proposed model applied data-subsampling concept using Up-sampling, Down-sampling, SMOTE sampling techniques that provided a composite feature set with sufficiently data-features for further computation. However, realizing large data size of the composite features, the proposed model applied WRST algorithm, a well-known significant predictor test, which retained only those feature elements having significant impact (likelihood) on reusability prediction. Thus, retaining the optimal feature set, to further reduce the probability of pre-mature convergence and over-fitting Min-Max normalization was applied. Finally, the proposed model applied neurocomputing algorithms assisted maximum voting ensemble learning structure for two-class classification. Here, the proposed homogenous ensemble model applied ANN-GD, ANN-RBF, ANN-LM (with 1 and 2 hidden layers) and PNN neural network algorithms as the base classifier to perform two-class classification, where it employed MVE (maximum voting or consensus) to classify each class as REUSABLE or NON-REUSABLE. Unlike standalone base classifiers, the proposed consensus or MVE ensemble model exhibited higher accuracy (96.57%), precision (0.94) and recall (0.99), signifying its robustness and reliability towards WoS software component reusability prediction system. Thus, the superior performance by the proposed neuro-computing assisted homogenous (consensus-based) ensemble model affirms its suitability towards accurate reusability prediction, which can enable a firm or developer to assess fault-resilience in terms of optimal reusability of a software component before its release. It can not only ensure fault-resilience of the software solution but can also avoid the maintenance cost and failure probability of WoS solutions.

References

[1] Caldiera G., and Basili V.R. (1991). Identifying and qualifying reusable software components, *IEEE Software*, vol. 24, pp. 61-70.

[2] Sommerville I. (2011) Software Engineering. 9th Edition, *Addison-Wesley*, New York.

[3] Singh G. (2013). Metrics for measuring the quality of object-oriented software, *ACM SIGSOFT Software Engineering Notes*, vol. 38, pp. 1-5.

[4] Stapic M. M. (2015). Reusability metrics of software components: *Survey, conference paper September.*

[5] Srivastava S. and Kumar R. (2013). Indirect method to measure software quality using CK-OO suite, Intelligent Systems and Signal Processing (ISSP), International Conference on, Gujarat, pp. 47-51.

[6] Goel B. M., Pradeep Kumar Bhatia (2012). Analysis of reusability of object-oriented system using CK metrics, *International Journal of Computer Applications*, vol. 60, no.10.

[7] Bakar N.S. (2016). The analysis of object-oriented metrics in C++ programs, *Lecture Notes on Software Engineering*, vol. 4, no. 1.

[8] Mohammad A. T. (2014). Metric suite to evaluate reusability of software product line, *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 4, no. 2, pp. 285-294.

[9] Nanni L., Brahnam S., Ghidoni S., and Lumini A. (2015). Toward a General-PurposeHeterogeneous Ensemble for Pattern Classification, *Computational Intelligence and Neuroscience*.

[10] Barghash M.(2015). An effective and novel neural network ensemble for shift pattern detection in control charts, *Computational Intelligence and Neuroscience, Article ID939248.*

[11] Singhani H., Suri R. P. (2015). 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*, vol. 15, no.5.

[12] Mijac M., Stapic Z. (2015). Reusability metrics of software components: Survey, *Central European conference on Information and Intelligent system conference paper.*

[13] Srivastava S. and Kumar R. (2013). Indirect method to measure software quality using CK-OO suite, *Intelligent Systems and Signal Processing (ISSP), International Conference on, Gujarat*, pp. 47-51.

[14] Goel B.M. and Bhatia P.K. (2012). Analysis of reusability of object-oriented system using CK metrics, *International Journal of Computer Applications*, vol.60, no.10, pp.0975–8887.

[15] Rosenberg L.H. and Hyatt L.E. (1997). Software quality metrics for object-oriented environments, *Crosstalk Journal*, vol. 10, pp. 1-16.

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

[17] Antony P.J. (2013). Predicting Reliability of Software Using Thresholds of CK Metrics, *International Journal of Advanced Networking & Applications*, vol. 4, p. 6.

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

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

[20] Shatnawi R. (2010). A quantitative investigation of the acceptable risk levels of object-oriented metrics in open-source systems, *IEEE Transactions on Software Engineering*, vol. 36, pp. 216-225.

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

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

[23] Normi Sham Awang Abu Bakar. (2016). The analysis of object-oriented metrics in C++ programs, *Lecture Notes on Software Engineering, Springer, vol. 4, no. 1.*

[24] Zahara S. I., Ilyas M., and Zia T. (2013). 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, pp. 75-80.

[25] Torkamani M. A. (2014). Metric suite to evaluate reusability of software product line, International Journal of Electrical and Computer Engineering (IJECE), vol. 4, no. 2, pp. 285-294.

[26] Aloysius A., and Maheswar K. (2015). A review on component based software metrics, Intern. J. Fuzzy Mathematical Archive, vol. 7, no. 2, pp. 185-194. ISSN: 2320 –3242 (P), 2320 –3250 (online)

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

[28] Tsoumakas G., Angelis L., Vlahavas I. (2005). Selective fusion of heterogeneous classifiers, Intelligent Data Analysis vol. 9 (6), pp. 511–525.

[29] Benediktsson J. A., Chanussot J., and Fauvel M. (2007). Multiple classifier systems in remote sensing: from basics to recent developments. *In: M. Haindl, J. Heidelberg*, Germany: Springer, 501–512.

[30] Roli F., Giacinto G., Vernazza G. (2001). Methods for designing multiple classifier systems. *Proceedings of the second international workshop on multiple classifier systems.* Cambridge, UK, 78–87.

[31] Banfield R. (2007). A comparison of decision tree ensemble creation techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29, 173–180.

[32] Krogh A., Vedelsby J. (1995). Neural network ensembles, cross validation, and active learning. In: G. Tesauro, D. Touretzky and T. Leen, eds. *Advances*

in neural information processing systems, vol. 7, Cambridge, UK: MIT Press, 231–238.

[33] Kittler J. (1998). Combining classifiers: a theoretical framework. *Pattern Analysis and Applications*, 1, 18–27.

[34] Hansen L., Salamon P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12, 993–1001.

[35] Opitz D., Shavlik J. (1996). Actively searching for an effective neural-network ensemble. *Connection Science*, 8 (3/4), 337–353.

[36] Kuncheva L.I. (2001). Combining classifiers: soft computing solution, in: S.K.A. Pal (Ed.), *Pattern Recognition: From Classical to Modern Approaches, World Scientific*, Singapore, pp. 427–451.

[37] Canul-Reich J., Shoemaker L., Hall L.O. (2007). Ensembles of fuzzy classifiers, *in: IEEE International Fuzzy Systems Conference*, pp. 1–6.

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

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

[40] Nanni L., Lumini A. (2009). Ensemble generation and feature selection for the identification of students with learning disabilities, *Expert Systems with Applications* 36, pp.3896–3900.[41] Zhang X., Wang S., Shan T., Jiao L.C. (2005). Selective SVMs ensemble driven by immune clonal algorithm, in: Rothlauf, F. (Ed.) *Proceedings of the EvoWork- shops*, Springer, Berlin, pp. 325–333.

[41] Zhou Z.H., Wu J., Tang W. (2002). Ensembling neural networks: many could be better than all, *Artificial Intelligence* 137 (1–2), 239–263.

[42] Partalas I., Tsoumakas G., Vlahavas I. (2008). Focused ensemble selection: a diversity-based method for greedy ensemble selection, *in: Proceedings of the 18th International Conference on Artificial Intelligence*, pp. 117–121.

[43] Dong YS., Han KS. (2004). A comparison of several ensemble methods for text categorization," Services Computing, (SCC 2004). *Proceedings. 2004 IEEE International Conference* on, pp. 419-422.

[44] Bi Y., Bell D., Wang H., Guo G., Guan J. (2007). Combining Multiple Classifiers Using Dempster's Rule For Text Categorization. *Appl. Artif. Intell*. vol. 21, 211-239.

[45] P. N. Tan, M. Steinbach, and V. Kumar, Introduction to Data Mining. New York: Addison-Wesley, 2005.

[46] N. V. Chawla, K.W. Bowyer, L. O. Hall, andW. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," J. Artif. Intell.Res., vol. 16, pp. 321–357, 2002.

[47] B. J. Lee, B. Ku, J. Nam, D. D. Pham and J. Y. Kim, "Prediction of Fasting Plasma Glucose Status Using Anthropometric Measures for Diagnosing Type 2 Diabetes," in IEEE Journal of Biomedical and Health Informatics, vol. 18, no. 2, pp. 555-561, March 2014.

[48] Miller A.: 'Subset selection in regression' (Chapman & Hall/CRC, New York, 2002, 2nd edn.)

[49] R. Radivojac, N. V. Chawla, A. K. Dunker, and Z. Obradovic, "Classification and knowledge discovery in protein databases," J. Biomed.Informat., vol. 37, pp. 224–239, 2004.

[50] R. Batuwita and V. Palade, "microPred: Effective classification of premiRNAs for human miRNA gene prediction," Bioinformatics, vol. 25, no. 8, pp. 989–995, Apr. 2009.

[51] X. M. Zhao, X. Li, L. Chen, and K. Aihara, "Protein classification with imbalanced data," Proteins, vol. 70, no. 4, pp. 1125–1132, Mar. 2008.

[52] Y Saeys, I. Inza, and P Larra˜naga, "A review of feature selection techniques in bioinformatics," Bioinformatics, vol. 23, no. 19, pp. 2507–2517, 2007.

[53] Adnan O. M. A., Dezheng Z., Xiong L., Ahmad S. and Hazrat A., "Improving Classification Performance through an Advanced Ensemble Based Heterogeneous Extreme Learning Machines", Hindawi Computational Intelligence and Neuroscience Volume 2017, pp. 1-11.