Journal of Information Systems Engineering and Management
2025, 10(14s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Intrinsic Feature Sensitive Software Reusability Prediction Using Parallelized Feature Selection and Heterogeneous Ensemble Learning Method

Dr. Prakash V. Parande[1], Dr. M. K. Banga[2]
[1]*Assistant Professor, Department of MCA, Visvesvaraya Technological University, Belagavi, Karnataka, India, 590018*
prakashvp2010@gmail.com
[2] *Professor, Department of CSE, Dayanand Sagar University, Bangalore, Karnataka, India, 560076*
banga.mkrishna@gmail.com

ARTICLE INFO

Received: 24 Nov 2024
Revised: 15 Jan 2025
Accepted: 31 Jan 2025

ABSTRACT

The rising global competitiveness in information technology industry has triggered industry to achieve and contribute efficient software solution even at the lower cost. With such a demand, the use of free open source software (FOSS) and component-reusability have become common in software industry. Though, software component reusability can be advantageous; improper reuse might even cause fault, smell, failure and eventual massive losses. Identifying a class with no-reusability, a developer can avoid it to inculcate in program that can improve software reliability. Unlike manual testing methods, which are often criticized for high cost and time-consumption, we propose a novel machine learning based automatic software reusability prediction that predicts each class as REUSABLE or NON-REUSABLE.

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

Introduction

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 source of program or allied cost; however, there has been the evidences where the exceedingly reuse of software component caused system failure and losses 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, a software turns into faulty, smelling and eventually fails in delivering the expected performance [4]. 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. A few existing methods [1-4] made effort to exploit machine learning algorithms for reusability prediction; however, failed in addressing intrinsic relationship amongst the different classes of functions to make optimal classification. In this paper, we exploited different Chidamber and Kamerer (CK) metrics, as intrinsic feature to perform reusability prediction. Noticeably, the modular and architectural design of an OOP-software often yields 22 different kinds of CK-Metrics, including coupling information, cohesion, correlation, structural feature etc. However, amongst the different CK metrics, the one characterizing coupling, cohesion, complexity and correlation (4C) are vital towards software reusability prediction task [1].

Methods and Materials

To begin with a set of literatures was studied to obtain a better grasp of the way that code re-usability has been achieved and how other research works have utilized various machine learning methods to assess the same.

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 classical machine learning over such imbalanced data might force it to exhibit false-positive, and hence 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 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. 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 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 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. 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 unparalleled reliability for any classification problem [36]. In [32] [35] [44], authors found that though 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 base classifiers can yield better accuracy. It can be 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. 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 neuro-computing 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. [55] explored the rainfall pattern and groundwater level and predicted a rise in the groundwater level using SARIMA, multi-variable regression, ridge regression, and KNN regression.

Noticeably, unlike classical methods where authors often use different base classifiers one-after another, in the proposed model, we designed heterogeneous ensemble learning model in such manner that inputting 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 great diversity of result. 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.

Though, the use of CKJM tool enables extracting a total of 22 software metrics; however, considering intrinsic feature sensitive reusability estimation goal, we considered 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).

In the proposed model, a total of 11 machine learning algorithms are applied in parallel to perform software reusability prediction. These key algorithms are given as follows:

1. Decision Tree Algorithm
2. k-NN Algorithm
3. Naïve Bayes
4. Support Vector Machine (RBF)
5. Logistic Regression
6. Multivariate -Adaptive Regression Spline (MARS)
7. ANN-GD
8. ANN-LM
9. ANN-RBF
10. Extreme Learning Machine
11. PNN.

1. Decision Tree (DT) Algorithm

Decision tree is one of the most used and conventional method used for data mining, pattern and text classification. Its efficiency or robustness witnessed numerous evolutions to cope up with contemporary demands. For example, different variants such as IDE, CART, DT C4.5 and DT C5.0 were developed to perform data mining and classification over more complex data or features. Starting at the root node and applying association rule between the split-condition the input feature was divided into multiple branches at each node of the tree. While, in the subsequent phase applying the Information Gain Ratio (IGR) value for each branch it performed two class classification. Thus, the proposed C5.0 decision tree algorithm labeled each class as REUSABLE and NON-REUSABLE.

2. k-NN Algorithm

k-NN, 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 regression predictive purposes; however, it has been found robust for numerous classification scenarios. By default, KNN classifier applies Euclidean distance to estimate inter-attribute distance using (1).

$$D(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_n - q_n)^2} \quad (1)$$

In (1), p and q are subjected to be compared with n features. In addition to it, even Manhattan distance can also be used.

In k-NN classifier, the efficacy significantly depends on the value of K that decides how many neighbors can be selected for classification. Selection of an optimal K value helps achieving better performance. The large value of K minimizes the effect of variance imposed by random error and forces it to be used with low number of sample data. Summarily, the driving force maintaining optimal balance between performance and computation (by selecting optimal K value) depends on maintaining better balance between over-fitting and under-fitting. In 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 a large-scale data with varying pattern can't be guaranteed. In majority of the existing approaches, K value are applied based on sample size by applying the cross-validation scheme; however, at the cost of increased time-exhaustion. Unlike classical k-NN algorithm, in this paper a tree learning model has been developed that enables learning varied optimal k values for the different training samples, by encompassing training stage when performing k-NN based reusability prediction. During training phase tree model at first performs learning the optimal value of k for all data samples under study by applying sparse reconstruction mechanism. It is then followed by constructing a decision tree using training samples and the learned optimal k values. During testing the proposed K-Tree model outputs the value of K swiftly for each data (testing data) samples, which is then followed by k-NN classification using learned optimal k value and data training. The proposed model enables similar running cost but with higher accuracy that makes it a potential candidate towards WoS software reusability prediction.

3. Naïve Bayes Classifier Algorithm

Naive Bayes classification is one of the most effective probabilistic classifiers that exploit the concept of Bayes' rules with certain independence assumptions. NB as a probabilistic approach is defined as "independent feature model" where it assumes that all considered features are independent and don't impact result significantly. NB algorithm allocates the specified object x to class $e^* = \text{argmax}_d P(d|x)$ using Bayes′ rule. In mathematical form it can be depicted as (2).

$$P(d|x) = \frac{P(x|d)P(d)}{P(x)} \tag{2}$$

In above expression, $P(d)$ signifies the probability of the parameter c before observing the data. The factor $P(d|x)$ refers the likelihood of data x. Mathematically (3),

$$P(x|d) = \prod_{l=1}^{m} P(x_l|d) \tag{3}$$

4. Support Vector Machine (SVM-RBF)

SVM has been one of the most used machine learning methods for pattern classification. The computational efficiency and robustness of SVM make it suitable for numerous classification purposes including text classification, image processing etc. Being a supervised learning concept, SVM learns over the input patterns and behaves as non-probabilistic binary classifier. To predict a solution or classify the inputs, it reduces the generalization error over the unobserved instances by means of a structural risk reduction concept. Here, the support vector represents a subset of the training set which retrieves the boundary values called hyper-place in between two classes having distinct features or the patterns.

$$Y' = w * \phi(x) + b \tag{4}$$

In (4), the parameter $\phi(x)$ states the non-linear transform where the emphasis is made on retrieving the suitable weight w and bias valueb. In (4), $Y'$is estimated by reducing the regression-risk parameter (14), iteratively.

$$R_{reg}(Y') = C * \sum_{i=0}^{l} \gamma(Y_i' - Y_i) + \frac{1}{2} * \|w\|^2 \tag{5}$$

In (5), C states a penalty factor, while γ presents cost function. The weight values are obtained using (6).

$$w = \sum_{j=1}^{l} (\alpha_j - \alpha_j^*)\phi(x_j) \tag{6}$$

To be noted, in above equation the components α and α^* states a relaxation factor, often called Lagrange multipliers, which are always selected as non-zero. The final output of SVM is (7).

$$Y' = \sum_{j=1}^{l} (\alpha_j - \alpha_j^*)\phi(x_j) * \phi(x) + b \tag{7}$$

$$= \sum_{j=1}^{l} (\alpha_j - \alpha_j^*) * K(x_j, x) + b$$

In (7), $K(x_j, x)$ presents the kernel function. We used RBF kernel to perform two-class classification that labels each class as REUSBALE ('1') and NON-REUSABLE ('0').

5. Logistic Regression

It is one of the most applied regression techniques often used for text classification and mining purposes. In present software reusability prediction problem, Logistic regression applies regression over the selected OOP-CK metrics (8), where CK metrics and finally obtained fused feature set, as obtained in (8), was considered as the independent variable while reuse-proneness or allied probability was considered as the dependent variable. Thus, the regression outputs two results, signifying REUSABLE and NON-REUSABLE. Mathematically, we apply (17) to perform linear regression over the input features.

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

In (8), $\text{logit}[\pi(x)]$ represents the dependent variable while x_i signifies the independent variable. This approach converts the dichotomous outputs by logit function and therefore makes $\pi(x)$varying in the range of 0 to 1 to $-\infty$ to$+\infty$. Observing (18), the parameter m signifies the total number of the independent variables, while the likelihood of a reuse-proneness of each class is given byπ. In this manner, the probability outcome or the dependent variable as output is obtained using (9).

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

6. Multivariate Adaptive Regression Spline

MARS is a recently developed non-parametric regression paradigm which has been found highly robust for performing regression with complex multi-Variate associations. In general, MARS functions on the basis of "Divide-and-conquer" policy where initially the retrieved OO-CK metrics data is divided into the different regions, each defined with respective regression equation. The mathematical expression for MARS based reusability prediction is defined in (10).

$$Y = \sum_{i=1}^{m} C_i BF_i(X) \quad (10)$$

In (10), Y states the dependent and X states the independent variable. The parameter C_i states the fixed coefficients and $BF_i(X)$ states the basis functions. Here, each basis function is defined in different forms.

Amongst the major machine learning algorithms, neural network often called ANN has been applied extensively towards 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. 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 and Extreme Learning Machine (ELM) algorithms etc. 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. In addition to the ANN-GD, ANN-LM, ANN-RBF the neuro-computing models like PNN, ELM etc. too have gained widespread attention, due to ease of computation and fast learning ability. Interestingly, the different algorithms have shown different performance for the same problem, and hence exhibit diversity of the performance. Considering this fact, we applied all ANN variants including ANN-GD, ANN-LM, ANN-RBF, PNN and ELM algorithms.

Functionally, ANN mimics human brain functionalities to learn over certain input data or patterns. Thus, learning over such input patterns it classifies unknown input into target categories. An illustration of ANN model is given in Fig. 1. As depicted in the figure, ANN comprises three layers; input layer, hidden layer and output layer.

Considering architecture of ANN, it embodies multiple neurons representing the input data to be further processed at the distinct intermediate layers for classification. 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. In this research paper, the ANN algorithms perform two-class classification, where it classifies each class as REUSABLE or NON-REUSABLE and label them as “1” and “0”, respectively. Being a two-class classification problem, the ANN designs have the one output layer, as given in Fig. 2. In the proposed neuro-computing or allied learning model, the final features selected pertaining to each class of the WOS 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, 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 (20) to generate O_h.

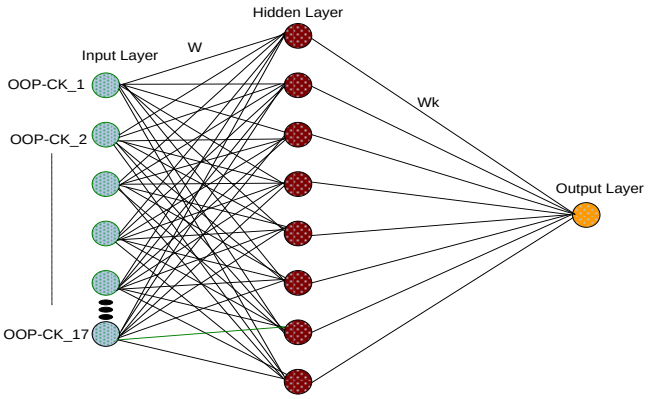

Fig. 2 An illustration of ANN architecture with single hidden layer with one output node.

$$O_h = \frac{1}{1 + e^{-I_h}} \tag{11}$$

In (11), 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 (12).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i' - y_i)^2 \tag{12}$$

In (12), yis the observed value, while y_i' is the expected value.

As already indicated, the key difference between the different ANN variants is its way of learning that eventually impacts eventual output at the output layer. In this reference, we applied the different ANN variants including ANN-GD, ANN-LM, ANN-RBF, PNN and ELM as neuro-computing base classifiers to perform two-class software component reusability prediction. The detailed discussion about these algorithms is given as follows:

7. ANN-GD

Let the regression for the learning method, while reducing error value be (22).

$$w^* = \underset{w}{\operatorname{argmin}}\, L(w) \tag{13}$$

$$L(w) = \sum_{t=1}^{N} L\big(y_t, f_w(x_t)\big) + \lambda R(w) \tag{14}$$

In ANN-GD setup, $f_w(x)$ factor states the non-linear weightw, and thus it intends to achieve a local optimum for (14) using GD method, which updates w iteratively by updating $w_t$byw_{t+1}.

$$w_{t+1} = w_t - \eta_t \nabla L \tag{15}$$

$$w_{j,t+1} = w_{j,t} - \eta_t \frac{\partial L}{\partial w_j} \tag{16}$$

In (15), the parameter ∇Lsignifies the error value, which is mathematically given as (17).

$$\nabla = \frac{1}{n}\sum_{i=1}^{n}(y_i' - y_i)^2 \tag{17}$$

In (16) η_t states the learning rate, which reduces over time t.

8. ANN-LM

Unlike classical ANN-GD and ANN-SD, ANN-LM possesses more robustness in learning over large non-linear data input. It performs localization of the minimum value of the multivariate function, called Sum of Squares (SoS) of the non-linear real-valued functions. This feature enables ANN-LM to perform fast and more efficient weight update. It also avoids local minima and convergence type issues and hence makes it suitable for large datasets. As already stated, ANN-LM has ability of both ANN-SD as well as ANN-GD, which helps retrieving swift error minimization. ANN-LM applies (27) to perform weight update during learning.

$$W_{j+1} = W_j - \left(J_j^T J_j + \mu I\right)^{-1} J_j e_j \tag{18}$$

In (18), the parameter W_j signifies the at-hand weight while W_{j+1} presents the updated weight. Similarly, I indicates the identity matrix, while the Jacobian matrix is given by J (19). Here, μ states the combination coefficient and the lower value of μ triggers ANN-LM to behave as ANN-GD, while higher value forces to act as ANN-SD.

$$J = \begin{bmatrix} \frac{d}{dW_1}(E_{1,1}) & \frac{d}{dW_2}(E_{1,1}) & \cdots & \frac{d}{dW_N}(E_{1,1}) \\ \frac{d}{dW_1}(E_{1,2}) & \frac{d}{dW_2}(E_{1,2}) & \cdots & \frac{d}{dW_N}(E_{1,2}) \\ \vdots & \vdots & & \vdots \ \vdots \\ \frac{d}{dW_1}(E_{P,M}) & \frac{d}{dW_2}(E_{P,M}) & \cdots & \frac{d}{dW_N}(E_{P,M}) \end{bmatrix} \tag{19}$$

In (19), N represents the total weight counts and P presents the input features. The output is given byM.
Similar to the above stated ANN-LM model, we implemented ANN-RBF as one of the base classifiers to perform reusability prediction. Due to space constraints, its details are not given in this manuscript.

9. Probabilistic Neural Network (PNN)

Probabilistic Neural Network (PNN) represents a variant of feed forward neural network, often used for classification problems. In the 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 miss-detection or miss-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 and 0 for negative. The detailed discussion of the PNN architecture and its function is given as follows.

The overall model was designed to perform two-class classification (i.e., K=2). The input layer comprised 17 nodes each carrying LOC, DIT, WMC, NOC, CBO, LCOM, RFC, Ca, Ce, NPM, DAM, MOA, MFA, CAM, CC, CBM, and AMC 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 vectorx). 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, 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 Qbe 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 Pnodes 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 $y^{(q)}$ for any input vector x be (20) and (21).

$$g_1(x) = \left[\frac{1}{\sqrt{2\pi\sigma^{2N}}}\right] \exp\left\{-\frac{-\|x - x^{(p)}\|^2}{(2\sigma^2)}\right\} \quad (20)$$

$$g_2(y) = \left[\frac{1}{\sqrt{2\pi\sigma^{2N}}}\right] \exp\left\{-\frac{-\|y - y^{(q)}\|^2}{(2\sigma^2)}\right\} \quad (21)$$

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 (22) and (23).

$$f_1(x) = \left[\frac{1}{(2\pi\sigma^2)^N}\right]\left(\frac{1}{P}\right) \sum_{(p-1,P)} \exp\left\{-\frac{-\|x - x^{(p)}\|^2}{(2\sigma^2)}\right\} \quad (22)$$

$$f_2(y) = \left[\frac{1}{(2\pi\sigma^2)^N}\right]\left(\frac{1}{Q}\right) \sum_{(q-1,Q)} \exp\left\{-\frac{-\|y - y^{(q)}\|^2}{(2\sigma^2)}\right\} \quad (23)$$

In (22-23), 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. $x^{(p)}$Andy$^{(r)}$ be the centers in the corresponding classes 1 and 2. The nominator component $\|x - x^{(p)}\|^2$ presents the Euclidean distance between xand$x^{(p)}$. 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 labeled each subject as “REUSABLE” or “NON-REUSABLE”.

10. Extreme Learning Machines (ELM)

The higher input data and input layers the classical ANN models often suffer local minima and convergence issue. In addition, it results into increased number of hidden layers and hence affects overall learning and classification processes. The increased number of hidden layers too increases the need of total weights for learning, and hence overall learning process turns out to be time consuming. To alleviate such issues, a new machine learning method called ELM has been proposed. In a few literatures ELM has been found potential to perform ensemble learning to achieve higher accuracy of the classification. It can be characterized as a single-layered multi feed forward neural network (SL-MFNN) that enables more efficient generalization than the classical algorithms. One noticeable efficiency of ELM is its ability to enable swift learning by performing random hidden node selection and respective weight estimation. In this paper, we have used ELM as a base classifier to perform SOFTWARE REUSABILITY where it takes OO-CK metrics as input and performing multivariate regression, classifies each class as REUSABLE and NON-REUSABLE. A snippet of the overall ELM assisted reusability classification is given in Fig. 4. Now, performing selection of the retrieved OO-CK metrics characterizing each class of the software components, it has been given as input to ELM.

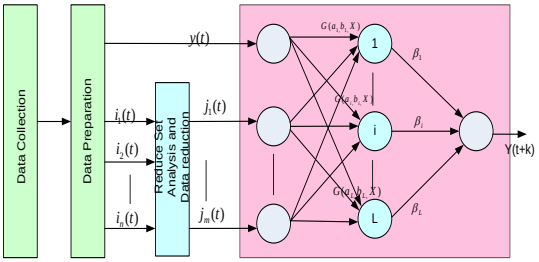

Fig. 3 ELM algorithm for software reusability prediction

In Fig. 3 the OO-CK metrics is feed as input (here,$X = (j_1, j_2, \ldots\ldots, j_m, y)$ where ystates the expected value. Here the data referred signifies a vector with $m + 1$ values, and $G(a_i, b_i, X)$presents the outcome of the iTh hidden neuron. In$G(a_i, b_i, X)$, b_i signifies the bias value of the iTh hidden neuron, while $a_i = (a_{i1}, a_{i2}, \ldots\ldots, a_{im}, a_{iy})$ presents the weight vector. Similarly, $a_{is}(s = 1,2, \ldots\ldots, m, y)$ represents the weights in between the sTh input layer and the iTh hidden layer. Considering above developed ELM structure, the output of the developed ELM base classifier is obtained as (24).

$$y(t + k) = f(X) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, X) \tag{24}$$

In (24) $\beta_i = (\beta_{i1}, \ldots\ldots, \beta_{in}, \beta_{iy})'$signifies the weight vector joining hidden layer and output layer. Furthermore, β_{ik} signifies the connecting weight in between iTh hidden layer (or neuron) and the kTh output neuron. For additive hidden neurons $G(a_i, b_i, X)$ can be defined as (25).

$$G(a_i, b_i, X) = g(a_i'X + b_i) \tag{25}$$

In (25), g: $R \rightarrow R$ refers the activation function for ELM.

The random weight factor a_i and biasing component b_i have been used here, and thus the ELM with L hidden neurons be sufficient for approximating N samples with zero error only when with certain value of β_i, retrieve Y_j as (35).

$$Y_j = \sum_{i=1}^{L} \beta_i G(a_i, b_i, X_j), j = 1,2, \ldots\ldots, N \tag{26}$$

Reframing above stated expression, we find (36).

$$Y = \beta H \tag{27}$$

Here,

$$Y = \begin{bmatrix} Y_1^T \\ \vdots \\ Y_N^T \end{bmatrix}_{N \times M}, \beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times M}$$

$$\text{And}H = \begin{bmatrix} G(a_i, b_i, X_1) & \ldots & G(a_L, b_L, X_1) \\ \vdots & \ldots & \vdots \\ G(a_i, b_i, X_N) & \ldots & G(a_L, b_L, X_N) \end{bmatrix}_{N \times L}$$

Estimating the learning components like weight parameters and the biasing component, ELM initiates learning. To perform learning an additional factor called minimum norm least-squares solution (β^*) has been applied where it is intended to reduceβ^*. Mathematically, β^* is referred as (28).

$$\beta^* = H^+ Y \tag{28}$$

In (28), H^+ states the Moore–Penrose generalized inverse of matrix H. In the proposed ELM the following key functions have been executed to perform 2 class classifications over the OO-CK software metrics.

In this paper, ELM as base classifier results y^* as output for each class. Mathematically, (29)

$$y^* = \sum_{i=1}^{L} \beta_i^* g(a_i x + b_i) \tag{29}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} (y(i) - \hat{y}(i))^2}{N}} \tag{30}$$

In (30), the resulting output y(i) signifies the desired output, while $\hat{y}(i)$ stands for the actual output. In the proposed prediction model RMSE which signifies the square root of the variance of the residuals which is considered as the standard deviation of the unexplained variance. When performing learning ELM updates its weight parameters while reducing RMSE to achieve higher accuracy. In this paper, ELM as base classifier has been applied to perform two class classifications to characterize each class of the software component as REUSABLE or NON-REUSABLE.

11. Heterogeneous Ensemble Learning based Reusability Prediction

The above discussed machine learning algorithms were applied as base classifier to perform two-class classification. Each of the classifier labeled each node pair as Linked ("1") or Not-Linked ("0"). Thus, obtaining the classified label for each node-pair, the proposed consensus-based ensemble learning (CEL) model applied the concept of maximum voting ensemble (MVE) to estimate consensus-based prediction. In this approach, the highest prediction output (1 or 0) by each base classifier was considered as the final prediction result. In other words, if for a specific node-pair if a total of 7 base classifiers predicts that node-pair as "Linked", while 5 other base classifiers predict the same as "Not-Linked", the proposed CEL model finally predicts that node pair as "Linked". Thus, obtaining the consensus of prediction by each base classifier, the proposed CEL model classifies each node pair as linked or not linked. Thus, the overall linked prediction problem was solved in this paper.

Results and Discussion

To assess the performance of the proposed model, we obtained confusion matrix 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.

Table 1 Performance outcomes

Performance Parameter	Machine Learning Accuracy (%) (Base Classifiers)											Ensemble
	DT	k-NN	NB	SVM-RBF	LOGR	MARS	ANN-GD	ANN-LM	ANN-RBF	PNN	ELM	Proposed MVE Ensemble
Accuracy (%)	89.91	91.37	92.00	92.70	88.03	90.07	89.94	95.41	94.15	95.01	93.97	96.91
Precision	0.871	0.891	0.891	0.913	0.903	0.914	0.891	0.922	0.924	0.934	0.917	0.948
Recall	0.903	0.910	0.893	0.921	0.912	0.928	0.926	0.931	0.937	0.947	0.923	0.957
F-Score	0.887	0.900	0.891	0.918	0.907	0.920	0.908	0.926	0.930	0.929	0.922	0.944

Table 1 represents the overall performance outcomes by the different standalone classifiers as well as the proposed heterogeneous ensemble (MVE) classifier. Observing the overall results, it can be found that the proposed MVE ensemble learning model exhibits higher accuracy of 96.91% than other standalone machine learning models. The graphical depiction for the accuracy performance by the different machine learning classifiers is given in Fig. 4. It affirms that the proposed model can yield higher accuracy even within expected time-limit and hence affirms its suitability towards real-world applications. Fig. 5 presents the precision performance, indicating how precise the algorithms can yield results even under different set of inputs or if the data is changed. The simulation result shows that the proposed heterogeneous MVE ensemble model outperforms other approaches (precision, 0.948).Considering other machine learning models, after the proposed ensemble learning method, PNN (0.934), ANN-LM (0.922) and SVM-RBF (0.913) has shown performance in decreasing order. Similar performance is achieved in terms of recall, signifying sensitivity of the classification method.

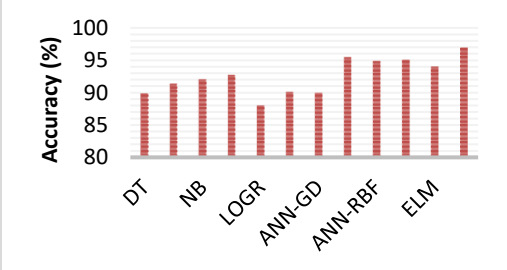

Fig. 4 Accuracy analysis for the different algorithms

Observing the result (Fig. 6), it can be found that the proposed heterogeneous ensemble model outperforms (recall, 0.957) other approaches; however, the performance by PNN (0.947), ANN-LM (0.931), ANN-RBF (0.937), MARS (0.928) was also satisfactory. Considering as an optimal solution, the proposed heterogeneous ensemble model exhibits superior sensitivity and hence indicates robustness towards any classification problem having very sensitive input (learning feature) patterns. It can be helpful to avoid false positive kind of issues.

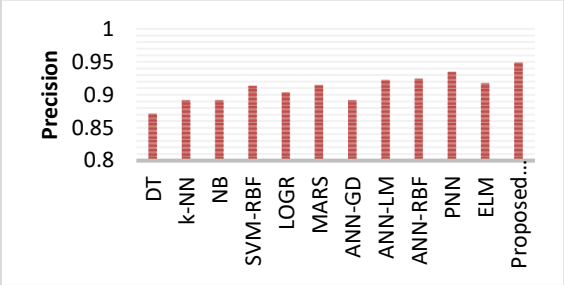

Fig. 5 Precision analysis for the different algorithms

For any classification problem, F-score, often called F-Measure is considered vital. It affirms robustness of the classifier model even under unbalanced class conditions. Considering the simulation results of the proposed model, we can find that the proposed heterogeneous ensemble method exhibits the highest F-measure or F-Score (0.944) than other standalone classifiers.

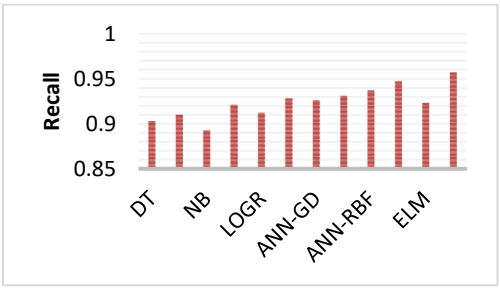

Fig. 6 Recall analysis for the different algorithms

Observing overall outputs, it can easily be inferred that the proposed ensemble model as a cumulative or composite solution with intrinsic software metrics learning ability, multi-phased features, SMOTE sampling and finally the proposed ensemble classifier can enable optimal and reliable software reusability prediction. The overall design, intend and eventual outcomes (Fig. 4 to Fig. 7) justifies that the proposed model can be more effective towards real-time software reusability prediction task, where its reliability and efficacy can be more justifiable than the manual testing or standalone machine learning based methods.

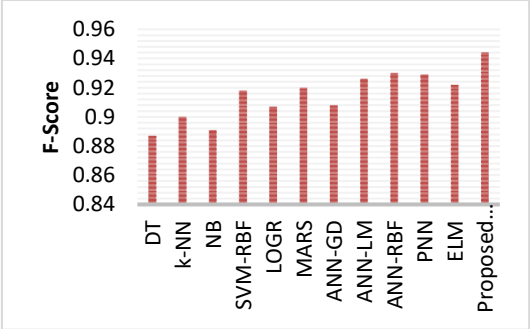

Fig. 7 F-score analysis for the different algorithms

Conclusions

In this research the key emphasis was on enhancing every possible step of a machine learning concept to improve accuracy and reliability of prediction system. Additionally, this research intended to examine whether and how the intrinsic software features or allied metrics could be employed to assess each-class reusability or reuse-

proneness. The proposed heterogeneous ensemble model employed decision tree algorithm, k-NN Algorithm, Naïve Bayes, SVM-RBF, LR, MARS, ANN-GD, ANN-LM, ANN-RBF, ELM, and PNN as base classifier. Noticeably, the aforesaid machine learning algorithms were applied in parallel and each algorithm predicted each class as REUSABLE or NON-REUSABLE and labelled as "1" and "0", respectively. Finally, applying weighted maximum voting ensemble the proposed heterogeneous ensemble model classifies each class as REUSABLE or NON-REUSABLE. The statistical performance analysis revealed that the proposed software prediction model exhibits superior performance in terms of accuracy (96.91%), precision (0.948), recall (0.957) and F-Score (0.944), signifying its robustness and reliability towards real-world software reusability prediction task. Though, the proposed method exhibited superior towards software reusability prediction; it was implemented over a single software unit, especially designed using OOP and modular software design paradigm. In future, research can exploit its efficacy towards other classification problems, demanding higher accuracy.

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, Intl. Journal of Advanced Networking & Appl, 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 Instt. of Tech, 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. Ttheetzky 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.

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

[43] 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.

[44] 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.

[45] 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.

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

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

[48] 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.

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

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] Maltare, N. N., Sharma, D. & Patel, S. (2023). An Exploration and Prediction of Rainfall and Groundwater Level for the District of Banaskantha, Gujrat, India. *International Journal of Environmental Sciences*, 9(1), 1-17. https://www.theaspd.com/resources/v9-1-1-Nilesh%20N.%20Maltare.pdf