Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2023.028153
Article

Tech Science Press

Artificial Intelligence Model for Software Reusability Prediction System

R. Subha[1,*], Anandakumar Haldorai[1] and Arulmurugan Ramu[2]

[1]Sri Eshwar College of Engineering, Coimbatore, 641 202, India
[2]Presidency University, Bengaluru, 560064, India
*Corresponding Author: R. Subha. Email: kris.subha@gmail.com
Received: 03 February 2022; Accepted: 07 April 2022

Abstract: The most significant invention made in recent years to serve various applications is software. Developing a faultless software system requires the software system design to be resilient. To make the software design more efficient, it is essential to assess the reusability of the components used. This paper proposes a software reusability prediction model named Flexible Random Fit (FRF) based on aging resilience for a Service Net (SN) software system. The reusability prediction model is developed based on a multilevel optimization technique based on software characteristics such as cohesion, coupling, and complexity. Metrics are obtained from the SN software system, which is then subjected to min-max normalization to avoid any saturation during the learning process. The feature extraction process is made more feasible by enriching the data quality *via* outlier detection. The reusability of the classes is estimated based on a tool called Soft Audit. Software reusability can be predicted more effectively based on the proposed FRF-ANN (Flexible Random Fit - Artificial Neural Network) algorithm. Performance evaluation shows that the proposed algorithm outperforms all the other techniques, thus ensuring the optimization of software reusability based on aging resilient. The model is then tested using constraint-based testing techniques to make sure that it is perfect at optimizing and making predictions.

Keywords: Service net; aging resilient; software reusability; evolutionary computing; intelligent computing

1 Introduction

With the increase in the number of functional and non-functional requirements, the software systems that have been developed over recent years are becoming more complex. If the system being developed is of lower quality, it has a drastic effect on the application using it. It is to be noted that the software development cost is higher than the cost of the system using it. In the last few decades, projects have started adopting Artificial Intelligence techniques in developing software systems. As software is considered one of the most significant inventions in recent years, it has led to the development of various applications in almost all fields, including medical, defense, science, engineering, and many more. This indicates the need for reliable software in the lives of human beings. The software system being developed consists of several modules and test cases, commonly known as software components. It

represents a software package that has been independently developed and can be used as a functional unit. In order to maintain the semantic relationship between the data and the associated functions, the system processes need to be placed in a distinct class [1,2].

To obtain a reliable and error-free design, it is necessary for the software components to be reused. This process of reusing the software components to develop a software system is called reusability. It refers to the maximum use of the software component [3]. The following are some of the advantages of software reusability:

- Reduced development cost
- Increased reliability
- Better QoS
- Designing takes lesser time
- Low time consumption for market acquisition
- Maintenance cost is low

Due to the reusability of components and because of its cost-effectiveness and reliability, Component-Based Software Design (CBSD) has been the centre of attention recently. It leads to high-quality software design in a cost-effective manner. The software design has been enhanced over the past few years to make it highly efficient, effective, and reliable. It is highly essential to ensure reliability, stability, scalability, and cost-effectiveness while designing software, which has become one of the hottest areas in the research domain [4,5]. The cost-effective nature of the software can be ensured by the concept called reusability. Ensure not to use excessive reusability as it may lead to more complex systems and result in the problem of software aging. Some of the problems that arise as a result of it are data loss, unscheduled shut downs, and the process being interrupted more frequently than ever. Service Net-based software development has gained significant attention over recent years, which led to the software industry starting to adopt this [6].

As software reusability is prone to design complexity, it may result in faults and aging problems despite its cost-effective nature. Hence, it is essential to maintain a balance between software reusability and aging proneness in order to avoid software failures. In order to avoid such failures as a result of reusability, it is better to define the extent to which a component can be reused. Thus, optimal reusable components and the extent to which the components can be reused can be identified by means of assessing the software metrics [7,8]. Software metrics are defined as the indicators of the software project that help in designing, developing, managing access and controlling the entire Software Development Life Cycle (SDLC). Based on this, the quality of the software can be evaluated.

The use of software metrics plays a prominent role in estimating the reusability in the case of Service Net software systems as well as OO software systems [8,9]. But, on the other hand, it remains a story untold, which is about assessing the fault proneness. Hence, we proposed an optimization scheme for ageing resilient software design. The proposed software design is thought of as a multi-level optimization scheme that does things like normalize, look for outliers, and more based on classification and prediction algorithms [9].

The research being conducted makes use of 150 Service Net (SN) software systems and ensures that the proposed Flexible Random Fit (FRF) algorithm based on ANN works along with the evolutionary computing technique. The system takes the CK metrics as input in predicting the reusability of the software components. The system also makes sure that the software component isn't likely to break or get old when it predicts how long it can be used.

The rest of this paper is organized as follows. Section 2 we are presented the detailed review of the paper, Section 3 discussed about the research measures and proposed approach discussed in Section 4. Reusability case studies and prediction model discussed in Sections 5 and 6. Result and conclusion discussed in Sections 7 and 8.

2 Related Works

This section discusses previous works on software reusability and reusability estimation. The author [10] discusses factors affecting the testability of OOP software by means of a hierarchical process called the Analytical Hierarchical Process (AHP). The author [8] also found that the Chidamber and Kemerer (CK) metrics have a greater impact on the reusability of the components. Based on these CK metrics, the software quality has been defined by the author [11] to maintain a balance between the software and its reusability extent. It is also to be noted that these CK metrics are used for estimating the reusability extent based on their threshold value [12]. The authors [13] and [14] proposed components-based metrics, especially in the case of OOP software design. To estimate the reusability of the OOP software design, the author [15] has proposed four metrics, namely-Method Template Inheritance Factor (MTIF) and Attribute Template Inheritance Factor (ATIF), Number of Template Children (NTC), and Depth of Template Tree (DTT), respectively.

By examining the components of both black-box and white-box and testing them using the existing software metrics, the author [16] found that to measure the reusability extent successfully, object-oriented metrics are considered the most effective. The relationship between the reusability components and the software design properties has been explored to identify the OOP software metrics to achieve design optimization [17]. The CK metrics are also used for identifying design faults that may affect the reusability and testability of the system [18]. Based on this, several other software metrics such as regularity metrics, coupling metrics, and the like were retrieved [19].

As the reusability is assessed based on the software metrics, the relationship among the components can be established with the help of a technique called Artificial Neural Networks (ANN) [17]. Based on this relationship, it is easy to estimate the reusability of the component identified in the relationship. A knowledge-based tool and natural language processing are also used for identifying the component reusability in OOP software systems. In order to obtain a constructive assessment for the reusability components, machine learning regression techniques were used. It also helps in assessing the software component reusability. Apart from machine learning and artificial intelligence, fuzzy logic is also used to construct an estimation model for detecting software reusability extent.

Factors such as modularity, flexibility, and adaptability also help in determining the software component's reusability. The main CK metrics such as Depth of Inheritance Tree (DIT), Coupling between Objects (CBO), Weighted Methods for Class (WMC), Number of Children (NOC) and Lack of Cohesion of Methods (LCOM) are used in assessing the reusability extent of the software component. The authors of [18] used statistical approaches to determine the relationship between cohesion and coupling, and thus estimated the reusability of the software component using fan-in and fan-out metrics.

In the following section, the OO-based software components and how they are used in achieving the reusability of the components are discussed.

2.1 OO-Based Software Metrics

Software engineering makes use of an important tool called software metrics. These metrics helps in measuring the software quality. These metrics play a major role in performing activities such as quality control, project estimation, assessment, and control. The OO-software metrics are mainly used for the purposes of reusability assessment, analyzing and predicting faults, assessing the maintenance cost, and

the like. In this paper, we make use of two metrics, namely the Kamrrer metric and the object-oriented Chidamber, for predicting the reusability of the software component. In this paper, the metrics are defined based on cohesion and complexity for assessing the software component reusability.

2.2 OO Software Metrics Set

The Object-Oriented software metrics along with their functionality in estimating reusability are discussed in the following Tab. 1. The metrics chosen are DIT, WMC, NOC, CBO, RFC, and LCOM.

Table 1: OO Software metrics for reusability estimation

OO software metrics	Functionality
DIT	This metric is used to indicate the length between the root and the node in a tree that is the highest.
WMC	In the given software project, the complexities of the classes are added cumulatively and are represented by WMC.
NOC	In a class hierarchy, the number of sub-classes that are subsidiary to the given class is represented by NOC.
CBO	It is used to indicate the connection between the classes.
RFC	Whenever the object of a class leaves a message, the methods to be executed are specified by RFC.
LCOM	The dissimilarity between the methods defined above is specified by LCOM

2.3 Reusability Assessment and Effectiveness of Software Metrics

The relationship between the OO software metrics and its reusability is identified in order to identify the reusability data. For this, two variables are considered. One is the dependent variable and the other is the independent variable. The dependent variable is represented by reusability, whereas the independent variable is represented by the software metrics. Thus, reusability can be defined as the function of the above-defined software metrics as shown in Eq (1).

$$\text{Reusability} = \text{f}(\text{WMC}, \text{DIT}, \text{CBO}, \text{LCOM}, \text{NOC}, \text{RFC}) \tag{1}$$

In order to avoid the problems concerning aging proneness and complexity issues, a Binary Regression (BR) algorithm based on feature extraction has been proposed in this paper. The features thus extracted are used in reusability estimation. Reusability can also be defined as the function of the features extracted from the BR algorithm, as follows in Eq. (2).

$$\text{Reusability} = \text{f}(\text{features extracted from BR}) \tag{2}$$

A feature reduction mechanism has been employed in this paper to avoid the computational overhead caused by the extraction of features. In this paper, we perform feature reduction by means of the Reduced Feature Selection (RFS) method. The following Eq. (3) shows the mathematical representation of it.

$$\text{Reusability} = \text{f}(\text{RFS based feature reduction}) \tag{3}$$

3 Research Measures

The following research questions, when imposed, help the developers in designing software of better quality. The research questions are as follows in Tab. 2.

Table 2: Research measures

Research question 1	Do the OO software metrics help in identifying the reusability of all the software products?
Research question 2	Does the process of feature reduction have a role to play in reusability estimation and optimization?
Research question 3	Is it possible, with the given CK metrics, to design a robust reusability prediction model that is web-based?
Research question 4	How far do evolutionary algorithms increase the accuracy of reusability prediction?

The research questions mentioned in the above table are explored throughout the paper.

3.1 SDLC using Artificial Intelligence (AI) Techniques

It is well known that software architecture plays a major role in the maintenance of software systems. Hence, it is essential to perform a Software Requirement Analysis. It comprises of the following: The first one is the Requirement engineering where the requirements are gathered, analyzed, and then transformed into a more suitable representation. During this process, it is essential to ensure that the requirements are complete, unambiguous, and easily manageable. The biggest challenge faced by most people is to design architecture efficiently from the requirements specified. In the case of AI, developing software architecture involves identifying the components based on the requirements provided. In this paper, the component chosen for developing the architecture is reusability.

4 Proposed Work

In order to create a feasible software design, it is essential to maintain the relationship between reuse proneness and aging probability. The OO design can be optimized by ensuring that the reusability level is always less than the aging index. A software reusability model based on Service Net (SN) software systems has been developed. For the reusability estimation model that has been developed, the CK metrics are taken as the input. In order to provide quality software, it is essential to maintain a balance between software reusability and aging resilience. The schematic diagram of the proposed reusability prediction model is shown in the following Fig. 1.

5 Reusability Case Studies

The functional procedures involved in the reusability prediction method are discussed here.

5.1 Data Preparation

We have taken 150 software projects for consideration and converted these software projects into Java files, where WSDL is applied. The main use of WSDL is to document the service interfaces used. In order to perform reusability estimation of the software components, WSDL files attached with “source forge” have

been chosen. WSDL helps in describing the functionalities of the web services, which in turn are defined as the components of the port type as defined as follows in Eq. (4).

$$PT = \{F_0(I_0, O_0), F_1(I_1, O_1), \ldots, F_x(I_x, O_x)\} \tag{4}$$

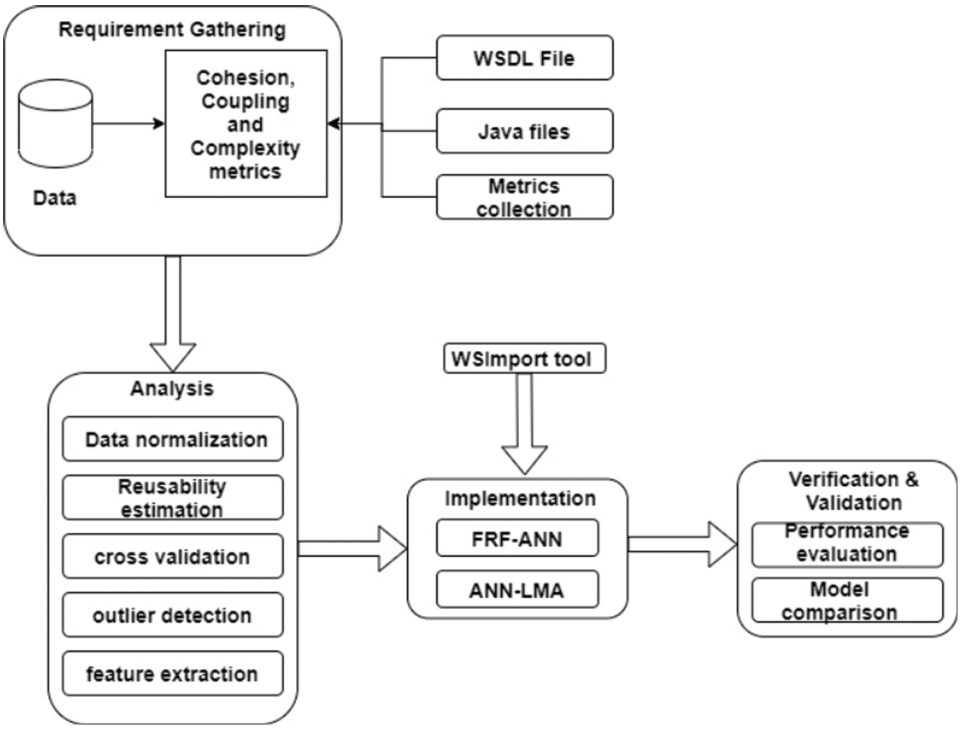

Figure 1: Architecture of reusability prediction model

Thus, the port type is defined as a function F of the input I and the output message O. Here, XML helps in providing the categorical definition for each data point [20]. The XML Schema Definition helps in defining the structure of XML. The mapping between the Java class components is shown in the following Tab. 3.

Table 3: Mapping the java class components to WSDL

WSDL doc	Java doc
Wsdl:service	Service class
Wsdl:portType	SEI class
Wsdl:portType/wsdl:operation	SEI class approaches
Wsdl:service/wsdlport	getABCPort() methods corresponding to the service class

Once the mapping is completed, the software metrics are used to estimate the quality of the Java files that are mapped to the various classes.

5.2 Data Normalization

The values of the CK metrics thus retrieved from various software projects are thus normalized to ensure that the metrics have no convergence and high accuracy. A Min-Max normalization is used to normalize these metric values. The values are normalized over the range [0–1]. Here, Min-Max normalization is performed by performing a linear transformation to the original data, to which mapping is performed in the range [0–1]. The normalized value of the attribute A is given by a_i and is defined as follows in Eq. (5).

$$\text{Normalized}(\text{a}_\text{i}) = a_i' = \frac{a_i - \min(A)}{\max(A) - \min(A)} \tag{5}$$

Here,

$\min(A)$ represents the minimum value of the attribute A

$\max(A)$ represents the maximum value of the attribute A

5.3 Cross-Validation

It is one of the statistical learning approaches that are used for comparing and estimating the model. It does so by dividing the data into two different parts. The first part is used for training the model, while the second one is used for validating the model. Here, in this paper, we make use of the N-fold cross validation technique. The technique works as follows:

- Partition the data into N-folds of equal size.
- Perform training based on N-1 folds. Use the remaining one-fold for testing

Since better performance can be obtained with 5-fold and 10-fold cross validation techniques, we, in this paper, considered using the 5-fold cross validation technique. Outlier detection is used to remove extreme data values, if any, in order to prevent the performance from degrading. The following mathematical expression helps in detecting the outliers in Eq. (6).

$$d_i = \left\{ \begin{array}{c} if \left| m_{ji} > \widehat{m}_j \right| > 3 * \sigma; Outliers \\ if \left| m_{ji} > \widehat{m}_j \right| \leq 3 * \sigma; no\ Outliers \end{array} \right\} \tag{6}$$

The i^{th} value of the attribute j is represented by m_{ji}. σ represents the standard deviation.

5.4 Feature Selection Based on Logistic Regression Analysis

Logistic regression has been applied to the software metrics in order to identify its significance level. It is a form of statistical analysis where the dependent variable is presented with two distinct values. The logistical Regression is represented in the mathematical form as follows in Eq. (7).

$$\text{logit}(\mu(\text{n})) = \rho_0 + \rho_1 N \tag{7}$$

Here,

μ represents the values that vary during the reusability of software. $\mu(n)$ in turn can be represented as follows in Eq. (8).

$$\mu(\text{n}) = \frac{e^{\rho_0 + \rho_1 N}}{1 + e^{\rho_0 + \rho_1 N}} \tag{8}$$

The regression coefficient and σ value help in determining the significance level. Here in our paper, the regression coefficient is chosen to be positive and the value of $\sigma = 0.5$. The six OO-CK metrics defined earlier were chosen to predict the reusability of the software designed.

5.5 Reduced Feature Selection (RFS) Method

It is a generic approximation method that helps in assessing the effectiveness of various kinds of data, such as arbitrary data, indefinite data, and the like, by determining the upper and lower bound of the data. The Reduced Feature Selection (RFS) method works on the concept that the "Precision Extent" of the data is reduced, making the pattern more perceptible. The end result is the reduced data set or the classification of the data. Here in this paper, a reduced dataset is obtained as a result of the RFS method. Implementation of this technique consists of the following

Step 1: In the first phase, the features of the CK metrics are extracted.

Step 2: Here, data discretization by means of K-means clustering is performed.

Step 3: In this step, the lower and upper bound values of the data are obtained.

In Eq. (9), the lower bound value UB(n) of the variable n is given mathematically.

$$\mathrm{LB(n)} = \{n_i \in \cup | n_i \subset N\} \tag{9}$$

Similarly, the upper bound UB(n)value is given by Eq. (10).

$$\mathrm{UB(n)} = \{n_i \in \cup | n_i \cap N \neq 0\} \tag{10}$$

Step 4: Estimate the accuracy of the set of data.

Accuracy (ACC) is given by the ratio of the cardinality of the lower bound data to the cardinality of the upper bound data in Eq. (11).

$$\mathrm{ACC} = \frac{Card(LB(n))}{Card(UB(n))} \tag{11}$$

Step 5: Based on how accurate each piece is, choose the best data from a set of data. This way, the accuracy of each piece of data is equal to how accurate the whole set of data is.

Step 6: Finally, the data set having the minimum cardinality value is selected as the reduced set, which is then used for the classification of data.

6 Proposed Model for Software Reusability Prediction

The proposed model for estimating software reusability works as follows:

- The software component's reusability level is calculated initially.
- Estimate the threshold value for each component identified.
- Perform classification on the extracted features.

In most of the literature, classification methods like the support vector machine and the like were used, which can get stuck in a small area and have problems with how quickly it can get to the right answer. Clustering techniques, on the other hand, are faced with coherence related issues. In this paper, we proposed an AGA-based ANN algorithm and compared the performance of the same with other existing prediction and classification algorithms such as the Levenberg-Marquardt Algorithm (LMA) and Artificial Neural Networks (ANN). A performance comparison is done in terms of accuracy and its error profile.

6.1 Levenberg Marquardt Algorithm (LMA)

The model is mainly used for the training of Artificial Neural Networks. Most software applications make use of this algorithm in solving minimization problems. Compared to the Gauss-Newton algorithm (GNA), the LMA algorithm is more robust and, hence, it is able to find a solution, though it starts off from a final minimum value. The input to this algorithm is the set of both dependent and non-dependent variables. The algorithm follows an iterative procedure in finding the solution. The algorithm also works well for both local minimum and global minimum values. The algorithm combines the work of both the steepest descendent and the Gauss Newton Algorithm (GNA). In the proposed work, the weight estimation is done as follows using the Levenberg-Marquardt algorithm in Eq. (12).

$$\mathrm{W}_{k+1} = W_k - (J_k + \omega I)^{-1} J_k \tag{12}$$

Here,

W_{k+1} represents the new updated weight

J represents the Jacobian matrix

ω represents the combination coefficient

I refer to the identity matrix

When the value of ω is zero, the LMA model behaves similarly to the GN algorithm. On the other hand, for maximum values of ω, the model functions as that of the GD algorithm. The Jacobian matrix is defined as follows in Eq. (13).

$$\mathrm{J} = \begin{matrix} \frac{\partial}{\partial w1}E(1,1) & \frac{\partial}{\partial w1}E(1,2)\ldots & \frac{\partial}{\partial w1}E(1,n) \\ \frac{\partial}{\partial w1}E(2,1) & \frac{\partial}{\partial w1}E(2,2)\ldots & \frac{\partial}{\partial w1}E(1,n) \\ \frac{\partial}{\partial w1}E(m,1) & \frac{\partial}{\partial w1}E(m,2)\ldots & \frac{\partial}{\partial w1}E(m,n) \end{matrix} \tag{13}$$

6.2 Artificial Neural Networks Based on Evolutionary Computing

In order to predict reusability proneness, an evolutionary computing technique called the Flexible Random Fit (FRF) algorithm based on Artificial Neural Network has been used. In this method, the relationship between reuse proneness and the associated software metrics is identified. The six CK metrics defined above are considered the independent variables in this research. The Artificial Neural Network to be designed takes as input the software metrics derived from the feature selection phase. The ANN model proposed here takes into account 4 hidden layers and a single output layer. The classification of the metrics is done based on two classes, namely:–REUSABLE and NON-REUSABLE. In the figure shown below, the ANN consists of three layers—namely the input layer, the hidden layer, and the output layer. The input layer takes as input, the six CK metrics. The output generated from the input layer is represented as I_o, which is taken as the input by the hidden layer. Similarly, the hidden layer makes use of the sigmoid function in calculating the output H_o, which is given as input to the output layer. The output generated by the output layer is represented as O_o. The value of H_o is calculated as follows in Eq. (14).

$$\mathrm{H_o} = \frac{1}{1 + e^{-I_o}} \tag{14}$$

The final output from the output node O_o is obtained as follows in Eq. (15).

$$O_o = \frac{1}{1 + e^{-O_i}} \tag{15}$$

The ANN can be represented in terms of a function as in Eq. (16).

$$V' = f(W, U) \tag{16}$$

Here,

V' represents the output vector

W represents the weight factor

U represents the input vector

It is essential to update the value of W iteratively during the learning process in order to minimize the Mean Square Error (MSE) value. The CK metrics identified earlier need to be processed for Min-Max normalization before being used for reusability classification by the ANN method. While processing, it is essential to consider the limitations of the ANN model, which are the convergence issues and the local minima, respectively. Another thing to keep in mind is that the weight estimation also makes the ANN method less efficient.

Due to these limitations, we propose in this paper a weight estimation mode based on evolutionary computing which enhances both the prediction and the classification accuracy of the ANN model. Hence, one of the AI algorithms called the Genetic Algorithm also suffers from local minima and convergence issues. Hence, by considering the crossover probability and mutation probability of the Genetic Algorithm, we propose in this paper a new algorithm based on the evolutionary computing paradigm called the Flexible Random Fit (FRF) algorithm. The FRF algorithm based on ANN is used for classification and reusability prediction, which is discussed further in the following sections.

6.3 Flexible Random Fit (FRF) Algorithm

By applying the evolutionary concept, it is possible to obtain both optimal and near-optimal solutions. Here, activities needed for evolution are simulated by the proposed FRF algorithm. Initially, growth is generated at random where the solution is identified by the growth generated. This in turn represents the binary strings used to encode the parameters. The fitness value of each entity present is calculated during this process.

If the fitness value of the entity is higher, it means that the probability of obtaining a better solution is higher and vice versa. This fitness value, thus obtained, helps in generating the crossover probability and the mutation probability. The process is continued until the best solution is found or the stopping criteria are met.

From the above Fig. 2, it is made clear that the ANN model is a three-layered architecture consisting of an input layer, a hidden layer, and an output layer. Six CK metrics are given as input to the input layer of the ANN network. Each element corresponding to the given metric refers to a distinct class. To train the ANN model, the required weight is calculated as follows in Algorithm 1.

Algorithm 1: Fitness estimation in FRF algorithm

Input: $I = \{I_1, I_2, \ldots, I_i\}$

Output: $O = \{O_1, O_2, \ldots, O_i\}$

Here, I represent the input pair of the input layer of the ANN model

O represent the output pair of the output layer of the ANN model

Phase 1

The weight of each entity, E, can be calculated as follows:

$$W_{k+1} = W_k - (J_k + \omega I)^{-1} J_k$$

Phase 2

Assume the weight to be a constant value. Based on this assumption, train the input pairs and obtain the output.

Phase 3

For every instance j, its error value R_j is calculated

$$R_j = \left(I_j - O_j\right)$$

Phase 4

The Root Mean Square Error (RMSE) for the entity E is calculated as follows:

$$M_i = \sqrt{\frac{\sum_{j=1}^{j=N} M_j}{N}}$$

Here, N represents the training data set.

Phase 5

The fitness value of the entity E_i can be obtained as follows

$$F_i = \frac{1}{M_i} = \frac{1}{\sqrt{\frac{\sum_{j=1}^{j=N} M_j}{N}}}$$

7 Results and Discussion

This section deals with the implementation of the research model developed. Several techniques are used to investigate the predictability of reusability. The input to the model is taken from the software metrics, while the prediction output of individual classes' reusability is considered the output. Around 150 software products are being considered for testing. Six software metrics, namely DIT, LCOM, RFC, NOC, CBO, and WMC, are taken for software reusability estimation. The online open-source platform called sourceforge.com is used for obtaining the 150 web applications. WSImport is used to convert web applications to the Java language, which is in turn processed using Chidamber and Kamrer Java Machines, using which the software classes can be extracted. It is essential to prevent excess reusability of the software components to prevent any faults. It is also required to identify the threshold level after the reusability is completed. A linear regression technique is applied over the chosen six OO-CK software metrics to identify the threshold level. It helps in determining if the class possesses

REUSEFULNESS or NOT. Min-Max normalization is used to deal with any imbalance conditions. Here the software metrics are mapped in the range [0–1].

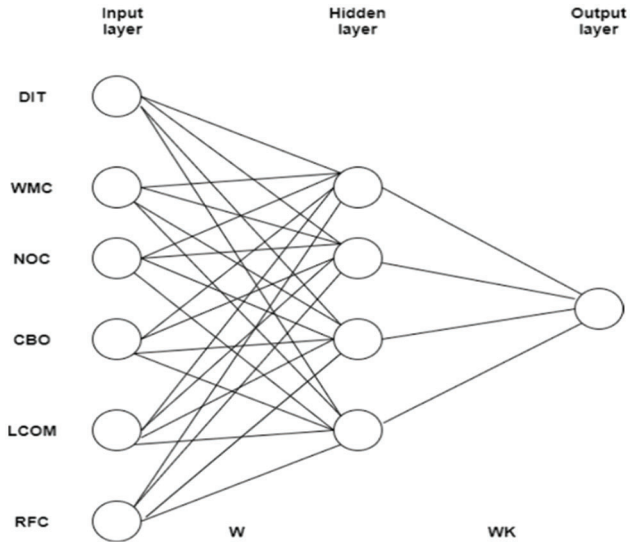

Figure 2: FRF algorithm based on ANN for software reusability prediction

Before the data is used for reusability prediction, it is subjected to 10-fold cross validation and then features selection and outlier detection. Our work also deals with ways to reduce computational overhead apart from reusability prediction. The computational overhead is reduced by means of the RFS-based feature selection method. In our proposed work, different types of classifiers are used for reusability prediction. MATLAB software is used for developing the prediction model.

7.1 Performance Comparison

Software reusability prediction is done based on parameters such as the Precision, Recall, Prediction Accuracy and F-Measure to identify the effectiveness of the classifiers chosen. These performance parameters are described as follows:

Accuracy–Accuracy is the measure of the fault-prone modules that are predicted out of the total number of modules.

Precision–It refers to the degree to which the observed measurements show the same results despite using the same conditions.

Recall–It represents the number of times the particular item needs to be identified.

F-Measure–It helps in obtaining a single score based on the mean value of Recall and Precision.

The performance comparisons of the various parameters corresponding to the different approaches are shown in the following Tab. 4.

7.2 Software Reusability Model Based on Evolutionary Computing

In this section, the performance of the proposed approach in performing software reusability is measured against various parameters such as RMSE, MMSE, r-value, MAE, p-value and many more. The features thus extracted are given as input to the FRF-ANN model, based on which the value of the weight is estimated to

identify the performance of the reusability prediction. Around 500 iterations were performed to obtain the values of the performance parameters. The performance parameters are as follows:

Table 4: Performance comparison of various approaches

Approaches	Precision	Recall	F-measure	Accuracy
ANN-LMA [1]	83.45	85.16	88.55	92.15
ANN-GD [2]	90.66	93.56	89.64	93.62
FRF-ANN [3]	95.18	92.99	89.32	93.56

Mean Absolute Error is shown in Eq. (17).

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|O_i - I_i|}{I_i} \tag{17}$$

Mean Magnitude of the Relative Error is shown in Eq. (18).

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|O_i - I_i|}{I_i + 0.01} \tag{18}$$

From the entries in Tab. 5, it is observed that the proposed FRF-ANN algorithm and the ANN-LMA perform better when compared to the other reusability techniques. The initial parameters (crossover probability and mutation probability) of the FRF are assigned the values of 0.8 and 0.2, respectively. As the parameter value remains constant throughout the execution process, the execution results in global optima, making the process continue further.

Table 5: Comparative analysis of various classification techniques

Techniques	MAE	MMRE	RMSE
ANN-GD	0.1246	0.7656	0.1365
ANN-LMA	0.1463	0.965	0.233
GA-ANN	0.746	0.3215	0.135
FRF-ANN	0.512	0.2130	0.0895

The next Tab. 5 shows how the performance metrics based on different classification techniques compare to each other.

The comparative analysis of the various classification techniques based on various metrics such as MAE, MMRE, and RMSE is shown in the above Fig. 3.

Fig. 4 represents the efficiency of the GA-ANN algorithm.

Here, Fig. 5 represents the efficiency of the FRF-ANN algorithm from the above two figures. It is made clear that the proposed FRF-ANN algorithm outperforms the GA-ANN algorithm. This is because the proposed FRF-ANN model avoids errors during the learning process itself. Also, as the number of entities increases, the MMRE value is significantly reduced. Thus, the proposed algorithm helps in reducing the errors at a faster rate, thereby helping in achieving convergence and ensuring effective reusability prediction.

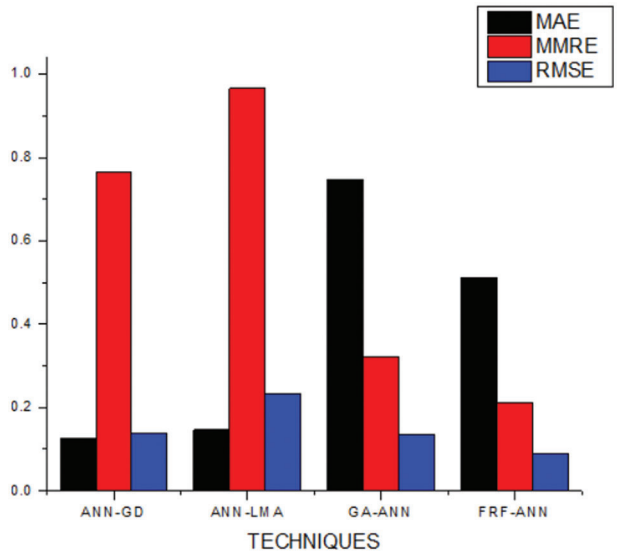

Figure 3: Comparison of reusability prediction based on various classification techniques

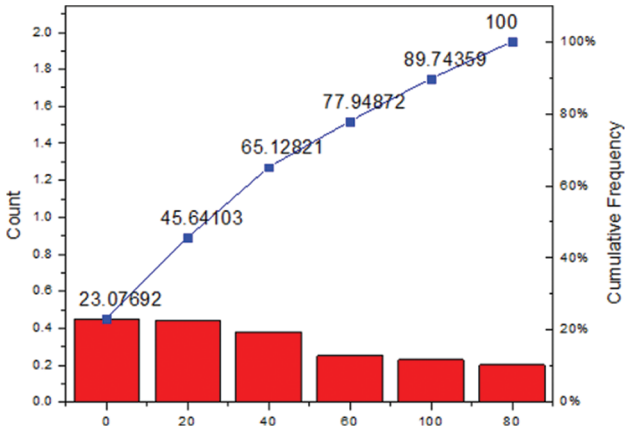

Figure 4: MMRE variation for GA-ANN method

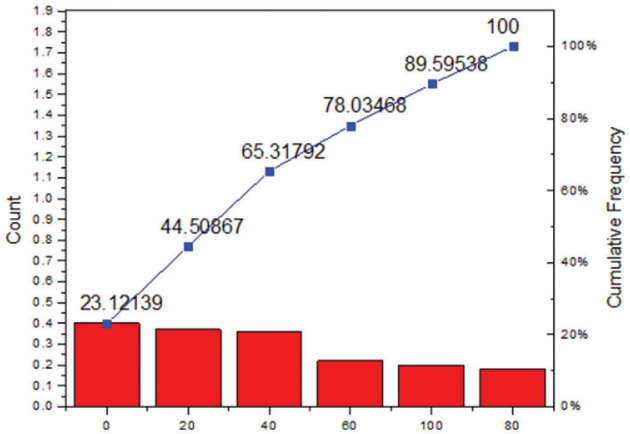

Figure 5: MMRE variation for the proposed FRF-ANN algorithm

7.3 Testing the Software Architecture

The main challenge associated with the development of the software is testing. Constraint solving is one of the AI techniques (Constraint-based testing) that helps in testing the software that has been developed. The testing involves creating test cases from the programs, which is taken care of by the traditional genetic algorithms. Testing is mainly done in order to increase the reliability of the software being developed.

8 Conclusions

A software reusability model based on aging resilient has been proposed in this paper, especially for OO-Service Net software systems. As aging proneness occurs when the specified threshold value is not considered, it is essential to assess the software component based on the retrieved threshold value to achieve optimization. The min-max normalization technique helps to make the data suitable for processing. Once the data is processed, feature selection and outlier assessment help in obtaining the optimal outcome for reusability prediction. The proposed FRF algorithm is compared against the LMA learning model and the ANN model, and the results obtained show that the proposed FRF algorithm outperforms all the other techniques discussed before. Hence, it provides better optimization in the case of reusability prediction by avoiding the aging proneness and fault probability. The OO-CK metrics such as cohesion, coupling, and complexity help in obtaining a robust reusability prediction model. Thus, the results obtained reveal that an optimal reusability prediction can be obtained for OO-software systems with the help of the FRF algorithm, which is an evolutionary computing technique.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] G. Caldiera and V. R. Basili, "Identifying and qualifying reusable software components," *Computer*, vol. 24, no. 2, pp. 61–70, 1991.

[2] F. J. Peters, "On the implementation of an application oriented software system," *International Journal for Numerical Methods in Engineering*, vol. 14, no. 10, pp. 1477–1497, 1979.

[3] B. M. Goel and P. K. Bhatia, "Analysis of reusability of object-oriented systems using object-oriented metrics," *ACM SIGSOFT Software Engineering Notes*, vol. 38, no. 4, pp. 1–5, 2013.

[4] C. J. H. Mann, "Object-oriented metrics in practice: Using software metrics to characterize, evaluate, and improve the design of object-oriented systems," *Kybernetes*, vol. 36, no. 5/6, pp. 1–11, 2007.

[5] V. Kumar, R. Kumar and A. Sharma, "Applying neuro-fuzzy approach to build the reusability assessment framework across software component releases-an empirical evaluation," *International Journal of Computer Applications*, vol. 70, no. 15, pp. 41–47, 2013.

[6] J. Sametinger, "Software components," *Software Engineering with Reusable Components*, vol. 1, pp. 67–82, 1997.

[7] G. Singh, "Metrics for measuring the quality of object-oriented software," *ACM SIGSOFT Software Engineering Notes*, vol. 38, no. 5, pp. 1–5, 2013.

[8] S. Sarkar, A. C. Kak and G. M. Rama, "Metrics for measuring the quality of modularization of large-scale object-oriented software," *IEEE Transactions on Software Engineering*, vol. 34, no. 5, pp. 700–720, 2008.

[9] S. Maggo and C. Gupta, "A machine learning based efficient software reusability prediction model for java based object oriented software," *International Journal of Information Technology and Computer Science*, vol. 6, no. 2, pp. 1–13, 2014.

[10] M. Huda, Y. D. Sharma Aryaand and M. Hasan Khan, "Metric based testability estimation model for object oriented design: Quality perspective," *Journal of Software Engineering and Applications*, vol. 8, no. 4, pp. 234–243, 2015.

[11] S. Kaur, "Quality prediction of object oriented software using density based clustering approach," *International Journal of Engineering and Technology*, vol. 3, no. 4, pp. 440–445, 2011.

[12] C. Arumugam and C. Babu, "Test size estimation for object oriented software based on analysis model," *Journal of Software*, vol. 10, no. 6, pp. 713–729, 2015.

[13] R. Shatnawi, "A quantitative investigation of the acceptable risk levels of object-oriented metrics in open-source systems," *IEEE Transactions on Software Engineering*, vol. 36, no. 2, pp. 216–225, 2010.

[14] S. Singh and K. S. Kahlon, "Object oriented software metrics threshold values at quantitative acceptable risk level," *CSI Transactions on ICT*, vol. 2, no. 3, pp. 191–205, 2014.

[15] A. Aloysius and K. Maheswar, "A review on component based software metrics," *International Journal of Fuzzy Mathematical Archive*, vol. 7, no. 2, pp. 185–194, 2015.

[16] P. Gandhi and P. K. Bhatia, "Reusability metrics for object-oriented system: An alternative approach," *International Journal of Software Engineering*, vol. 1, no. 4, pp. 63–72, 2018.

[17] S. Singh, M. Thapa, S. Singh and G. Singh, "Software engineering—survey of reusability based on software component," *International Journal of Computers and Applications*, vol. 8, no. 12, pp. 39–42, 2016.

[18] M. Huda, Y. D. S. Arya and M. H. Khan, "Quantifying reusability of object oriented design: A testability perspective," *Journal of Software Engineering and Applications*, vol. 8, no. 4, pp. 175–183, 2015.

[19] N. Goyal and G. E. Deepali, "Reusability calculation of object oriented software model by analyzing CK metric," *International Journal of Advanced Research in Science, Engineering and Technology*, vol. 3, no. 7, pp. 2466–2470, 2014.

[20] X. R. Zhang, W. F. Zhang, W. Sun, X. M. Sun and S. K. Jha, "A robust 3-D medical watermarking based on wavelet transform for data protection," *Computer Systems Science & Engineering*, vol. 41, no. 3, pp. 1043–1056, 2022.