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

Reconnoitering Static Analysis Metrics for Predicting Software Component Reusability Using Ensemble Model

Srishti Bhugra[1], Puneet Goswami*[2]

Submitted: 28/04/2023 **Revised**: 27/06/2023 **Accepted**: 06/07/2023

Abstract: Data mining and machine learning have created new avenues for creating software and investigation. The software development life cycle (SDLC) has also benefited from incorporating machine learning, opening up prospects for efficient and well-planned growth. The SDLC includes the reusability of software as a key component. Software reuse management thus assumes an active part in the SDLC. It reduces the expense and time needed to produce a software application. Evaluating a software component's reusability, or how appropriate it is for reuse, is a key difficulty in this scenario. The most effective methods for determining whether a particular software part is reusable or not come from machine learning to evaluate reusability; this study aims to create an ensemble machine learning model that integrates Support Vector Machine, K-Nearest Neighbour, Decision Tree, Artificial Neural Network, and Naive Bayes. After pre-processing, the publicly accessible benchmark dataset is used for experimentation. Compared with base classifiers, the suggested model delivered the most favourable results, with accuracy, precision, recall, and f1-score values of 89.48%, 0.9406, 0.9484, and 0.9445, respectively. According to the assessment of our technique, our approach can accurately evaluate reusability as experienced by engineers.

Keywords: *Ensemble Model, Machine Learning, Prediction, Reusability, Software.*

1. Introduction

The life cycle of creating software is an exhausting endeavour that requires funds and staff. However, efficiency is increased with reusable software. It additionally boosts the software product's reliability and maintenance. It lowers the total expense for a piece of software. The discipline of software intelligence is created by integrating knowledge discovery and data mining methods into software development procedures [1]. AI creates smart software and may be reused using computerised reusability approaches [2]. Code pieces and non-code elements are the two main categories for reusability [3]. Reusability may also be evaluated using a variety of measures. To achieve "growth by reuse" and "growth for reuse," software indicators that assess the reusability of programs are essential [2].

The reuse criteria might assist in creating reusability forecasting techniques that software engineers may employ. By understanding the overall amount of script that may be reused, it is possible to gather data on the collaborative costs associated with creating a fresh release of a previously developed piece of software or upgrading an old one. Numerous reusability-focused methods are presented in the literature. However, the literature does not specifically identify the measures needed for reuse. Evaluating a source code component's reusability before incorporating it as part of the developer's source code is essential because low-quality parts are frequently challenging to incorporate and fix. In certain situations, they can produce errors. Because the software's functionality is extremely context-specific and may entail various things depending on the individuals [4, 5], assessing reusability can

be difficult. Reusability is truly a quality notion. Software reusability, or the degree to whereby a piece of software is reusable, is connected to reliability, following the ISO/IEC 25010:2011 quality norm [6]. An element may be viewed as extremely reusable through a somewhat smoother viewpoint if it is flexible, shows weak coupling and strong cohesion, and offers concealment of data and division of responsibilities [7].

Statistical analysis and classification algorithms are crucial in software reusability by categorising, identifying, evaluating, and forecasting the static metric [8, 9]. Software reusability is organised, analysed, determined, and predicted using a variety of classification algorithms, including artificial neural networks (ANN), K-nearest neighbours (KNN), naive bays, decision trees (DT), support vector machines (SVM), many more. A machine learning approach, an ensemble algorithm, integrates predictions from many machine learning models. One of the most essential frameworks for developing excellent, precise models for forecasting is the ensemble model.

1.1. Key Contributions

This paper proposes a novel automated reusability

[1] *Department of Computer Science &Engineering, SRM University, Delhi NCR, Sonepat, India*
ORCID ID : 0000-0002-1765-6837
[2] *Department of Computer Science &Engineering, SRM University, Delhi NCR, Sonepat, India*
ORCID ID : 0000-0002-9545-7163
** Corresponding Author Email: goswamipuneet@gmail.com*

prediction model that considers the dynamic aspects of a software project along with its static metrics to conquer the problem effectively.

- Employing the reuse rate as a measure of developer-perceived reusability.
- An ensemble model is introduced for reusability evaluation based on the values of various static analysis metrics.
- Reusability estimation is done based on different source code properties.
- The accuracy, precision, recall, and f1-score of the proposed solution are compared to those of each of the base classifiers, and it is shown that the suggested model performs better, demonstrating the originality of our approach.

1.2. Section Division

The paper is divided into the sections that follow. In Section 2, the literature review is examined. Section 3 presents several machine learning techniques considered in recent research, and the ensemble model is introduced. Section 4 provides a description of the experiments conducted and the results attained. The investigation concluded in Section 5.

2. Literature Survey

Investigators have been interested in software reuse for decades due to its generally acknowledged ability to save production expenses and time [10, 11]. Douglas McIlroy first advocated the widespread creation of software employing reusable elements in 1968, when the reuse notion first emerged [12]. As a result, the obstacle now is more than merely identifying operationally sufficient sections and ensuring that these elements are appropriate for reuse. The expansion of the open-source development effort and the emergence of internet-based code repositories have presented novel chances for reuse. Software reusability considered an aspect of quality associated with reliability and relates to the extent to which a software object may be utilised in multiple systems or for the construction of additional resources, is defined by the ISO/IEC 25010:2011 quality norm [6]. Because of this, several approaches have been put forth to evaluate software parts' reusability using static analysis indicators [10] and realistically constructing reusability measurements using recognised quality features [13, 14].

Software reusability estimation by static analysis metrics is a challenging endeavour that frequently requires the assistance of qualified specialists who review the source code. Manually checking the source code might take time and effort, particularly for large and complicated applications and applications that evolve often. Considering the consistently rising needs for functional and non-functional criteria, this holds for most applications worldwide.

A thorough taxonomical breakdown of several AI approaches used in software engineering is given in [1]. The classification system is known by the term AI-SEAL. It helps researchers and specialists discover and comprehend the benefits and drawbacks of using AI techniques for software development. The taxonomy provides instructions on how to use a particular method for reusability. The type of AI (TAI) utilised, Point of Application (PA), and Level of Automation (LA) supplied are the three aspects that have been suggested as crucial. The study goes into more depth on how PA is implemented in the SWEBOK cognitive domains. Wangoo, D.P. [5] has identified many AI strategies that result in software intelligence. The article illustrates how software intelligence may be produced by doing intelligent knowledge discovery utilising AI approaches to data from software engineering. Software intelligence eventually results in smart automation through intelligent code reuse. Software processes are translated to different AI approaches. According to a case study on Microsoft [15], the company uses AI strategies in its ongoing software creation procedures. When incorporating AI into processes for developing software, programmers encountered three difficulties: finding, organising, managing, and versioning the data required for machine learning systems is far more challenging than other software development areas. The software development team must have a specific skill set to personalise, customise, and reuse a model. Compared to standard software modules, AI components are more challenging to handle as independent objects and sections because simulations may become "entangled" in intermittent failure conduct that has a complex character. The contributor has done surveys by interviewing people and examining how software is used. Each problem and the related techniques taken to fix it are identified in the findings, which are provided in quantitative form.

Ammar et al. [16] apply cutting-edge AI methods to software operations. For instance, it connects knowledge-based systems to demand engineering by pointing out that recycling design data from experts may significantly improve the efficiency and standard of the software creation procedure. Similar connections exist between different systems and their efficient AI techniques.

Component-based software engineering (CBSE) is an emergent phenomenon in [2]. Numerous reusable non-code sections, including the project plan, design, layout, comprehensive structure & scheme, code snippets, and validation instances, have been affected by adopting algorithms like Kmean and cosine analogy. Kaur et al. classified the information in the article [17] using algorithms for classification, sometimes called meta-

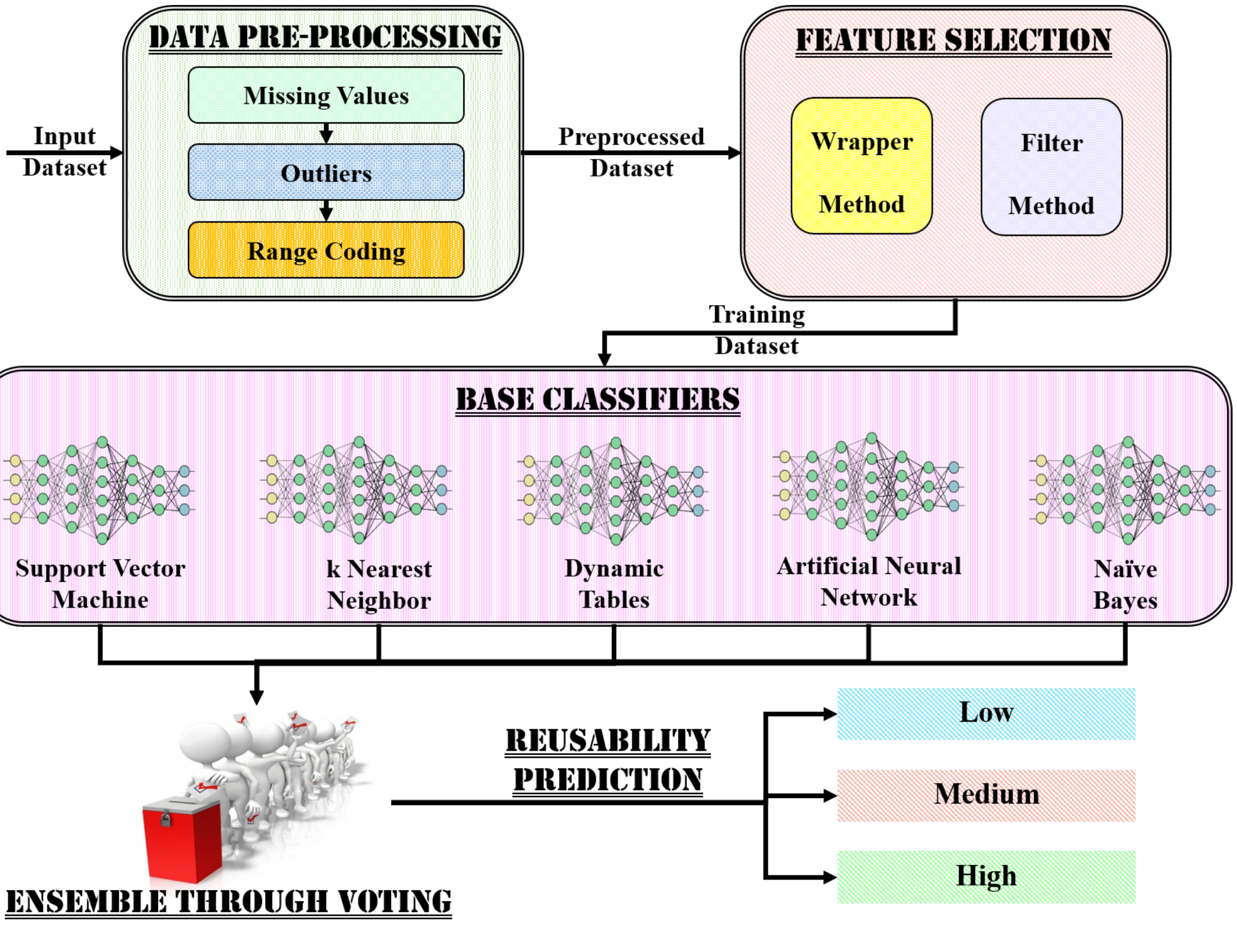

Fig. 1. Proposed Model

classifiers, that utilised seven reusable criteria on four distinct versions of the same application. Components with comparable features are grouped using the component clustering approach in the article [2]. The data mining techniques employed in this work include hierarchical clustering, K-mean, K-mode, and K-medoid. After the cluster has been established, it may be searched for predicted components. Therefore, this method makes it possible to reuse parts.

According to Chythanya et al. [18], CBSE classifies generation into two categories: "The Creation for reuse" and "The Creation by reuse." As a result, enormous component databases are built for future reuse. Effectively scanning the databases to find and get a certain component is done using neural network techniques. The study demonstrates two ways to determine reusability: empirically and qualitatively.

The objective of this study is to construct an ensemble machine learning model that incorporates Support Vector Machine (SVM), K-Nearest Neighbour (KNN), Decision Tree (DT), Artificial Neural Network (ANN), and Naive Bayes (NB) on static analysis metrics to assess reusability.

3. Proposed Model

The architecture of the proposed model is presented in **Error! Reference source not found.**.

3.1. Data Pre-processing

The benchmark dataset recommended by Papamichail et al. has been employed in the current investigation [8, 19]. The dataset was generated after examining the majority of prevalent endeavours listed in the Maven registry and using the SourceMeter tool [20] to calculate an extensive set of static analysis metrics across class and package categories that measure each of the six main source code characteristics: complexity, cohesion, coupling, inheritance, documentation, and size. The obtained dataset may be an empirical foundation for creating and implementing data-driven reusability assessment methods because it comprises statistical data for over 24,000 classes and 2000 packages. 28 of the 32 attributes defined by the dataset specify the properties of the fundamental software element. 27494 instances altogether are used in the current investigation. The dataset is initially pre-processed to eliminate any cases with missing values and outliers [21, 22]. After that, the target attribute (ReuseRate) is converted from a numerical value to an enumeration type with three categories: Low, Medium, and High.

3.2. Feature Selection

A full classification model may be produced by eliminating pointless characteristics from the data set and reducing the overall dimension of the part. The authors combined the ranker search approach with the Info Gain Attribute Evaluator [23] feature selection technique to choose the most pertinent attributes. The Info Gain Attribute Evaluator determines a feature's worth by assessing the information gained about the categories. Rather than appropriately separating the characteristics, the Info Gain Attribute Evaluator may consider binary numerical factors. The missing value can either be handled as a distinct value or distributed over additional discounts in proportion to their mean value for a category property or the typical occurrence for a numeric characteristic. The Info Gain Attribute Evaluator can recognise Nominal characteristics, Date features, Integer characteristics, Unary properties, Binary properties, Null nominal characteristics, and missing values.

In addition to feature evaluators like gain ratio and entropy, the ranker search technique used to produce rankings attributes rates each attribute by its particular evaluator. It is equipped to make attribute ranking. The best initial search technique has been implemented with the additional feature selection authors employed in this investigation wrapper subset evaluator. When evaluating attribute sets, the wrapper subset evaluator uses a learning pattern. Cross-validation has been conducted to determine the correctness of the learning pattern for a group of qualities. It can identify the types of characteristics, including string format characteristics, null attributes, Absent values, Date features, Relationship characteristics, Numeral features, Unary characteristics, Binary characteristics, and Nominal features. It can also determine the absence of values, the Nominal category, the Binary category, the Date category, and the Numeric group. The best first search by greedy hill climbing involving more backtracking capability is used for exploring a pool of feature groups. The number of permitted successive non-improving nodes controls the amount of backtracking. Best first may begin with an unpopulated set of characteristics and search forward, begin with the complete set of parts and search backwards, begin at any stage and explore in either direction (taking into account all potential individual feature alterations and additions at a particular moment), or any combination of these.

3.3. Base Classifiers

The objective of this study is to construct an ensemble machine learning model that incorporates Support Vector Machine (SVM), K-Nearest Neighbour (KNN), Decision Tree (DT), Artificial Neural Network (ANN), and Naive Bayes (NB) on static analysis metrics to assess reusability.

3.3.1. Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning technique built on statistical instruction principles. SVM has a good efficiency level in classification, regression, and prediction [24, 25, 26]. It creates a unique hyperplane in the interplanetary description of the training information, and blends are categorised according to where the hyperplane is placed across. Regression and classification algorithms are employed to forecast and analyse the collected data [27, 28]. SVM stands for supervised learning methods frequently used for classification in data mining. SVM uses multiple algorithms for sorting to provide the right outcome. It can reduce the likelihood of error by maximising the sum of the occurrences of the two classes. The advantage of the SVM is that it eliminates the need to perform an explicit conversion of the primary characteristics by using a "kernel trick" to determine the proximity of an element and the hyperplane in a modified (nonlinear) characteristic space.

3.3.2. K-nearest Neighbor

The KNN method is a supervised learning technique that is nonparametric by definition [27]. Separating linear from nonlinear is not necessary. KNN works well for many records and trains models quickly. KNN identifies the items from the provided k number of items that are closest to the majority decision or precise point query. It operates based on the nearest class item closest to the training instance and has the least distance between them. KNN is the model-building technique with the quickest execution period, as defined by [29, 30]. Simple majority voting is applied to the forecast question by KNN, which gathers all of the neighbouring items. The k instances that must be classified for each query Xn are x1, x2,... xk, Various distance metrics, including the Euclidean (E), Manhattan (Ma), and Minkowski (Mi) as presented in Equation (1), Equation (2) and Equation (3), respectively, will be used to identify the closest class.

$$E = \left(\sum_{\forall i \in 1,2,..n} (X_i - Y_i)^2\right)^{1/2} \tag{1}$$

$$Ma = \sum_{\forall i \in 1,2,..n} abs(X_i - Y_i) \tag{2}$$

$$Mi = \sqrt[n]{\sum_{\forall i \in 1,2,..n} abs((X_i - Y_i)^n)} \tag{3}$$

3.3.3. Decision Tree (DT): J48

A freely available Java implementation of the C4.5 DT technique is a J48 DT [31]. This application expands Ross Quinlan's earlier ID3 technique [31]. J48 classification algorithm builds trees using top-down greedy search techniques [32]. The J48 DT generates a sorting tree wherein the leaf indicates the finishing category, and intrinsic features provide a range of potential fork feature outcomes [32]. The distinction is between the splitting characteristics of information acquisition. The indicator of the disorder information is called entropy. Entropy is a gauge of the degree of uncertainty in any random element. Entropy (Γ) is easily computed as presented in Equation (4) for every likelihood l and sample δ.

$$\Gamma(\delta) = \sum_{\forall i \in 1,2,..n} [-l_i \ln(pi)] \tag{4}$$

Information gain, or choosing the ideal characteristic for selecting the particular vertex in a tree. When computing the quantity of a feature concerning collection, refer to the attribute's value as the numerical value of (Ω), the value of sample δ from δi, then Equation (5) is adopted to compute information gain, G.

$$G(\delta,\Omega) = \Gamma(\delta) - \sum_{i\in value(\Omega)}^{n}\left(\Gamma(\delta)\frac{abs(\delta_i)}{abs(\delta)}\right) \qquad (5)$$

3.3.4. Artificial neural network

Artificial neural networks (ANN), usually called "neural networks," are frequently used in practical applications and are based on actual neurons. The interconnected components of artificial neurons make up an ANN, which links every node with movable weights that alter its prearrangement as messages are sent [33]. Since ANN is a system that learns, it may adapt to changing its composition due to learning and the data it receives through its inner and outer environments [34]. Many levels are included in ANN for transmitting different data. Input, hidden, and output layers make up the layer structure. One or more layers with a certain amount of nodes are hidden layers. Each of these layers is arranged about one another, with weight connected to every single node. A supervised learning method known as an ANN uses input from network participants to create output. The perception of an ANN is its fundamental operational unit. Data collection may be divided into two classes using a perceptron algorithm. Individual nodes, along with weights, make up perceptron. The three basic components of a perceptron are connection, adder, and activation mechanisms.

3.3.5. Naïve Bayes

A probabilistic method of classification using distinct premise characteristics is known as the naive Bayes classifier. In various practical supervised classification situations, the Naïve-Bayes classification method works effectively and picks up new information rapidly [27]. A worldwide issue is diagnosed and predicted using naive Bayes. To forecast and assess the parameter, the Naïve Bayes method needs fewer training data from classification [35]. A member of each class is predicted using the Naive Bayes classification algorithm. For instance, the target class's supplied record will likely exist. The type with the greatest odds is the one that is most likely to occur.

3.4. Ensemble Classifier

Ensemble learning is a machine learning algorithm that builds an assortment of ensemble forecasting models and integrates the results to improve the outcomes of each distinct method. Ensembles with the greatest heterogeneous classifier typically have an excellent prediction assessment [36]. The most effective technique to fix mistakes caused by the base classifier is utilising an ensemble classifier [36]. Using many classifiers rather than one classifier is becoming quite common in machine learning combined classification. The benefit is that one may employ two or more categorisation algorithms instead of just one with greater strength. To categorise examples from the training set, cross-fold validation set, or testing set, the model authors develop will thus be more potent and advanced. The ensemble classification model aims to integrate many classifiers, each uniquely impacting the outcome [37]. The strategies have altered the training procedure to create classifier models that produce results in various classification conclusions [37]. The major benefit of ensemble methods is that they integrate distinct classifier criteria into stronger predictions than those rules alone. The ensemble model concept blends diverse individual classifiers to provide better prediction ability. For this objective, the authors of the current work use a voting classifier.

The ensemble approach known as the voting classifier offers a means to aggregate forecasts from several forecasting models. A majority vote determines the ultimate prognosis or the anticipated possibilities are averaged. The authors utilised DT, ANN, and NB as the basis classifiers for this research. Authors additionally employed the 'soft voting' technique, which averages the likelihoods from every classifier to determine the ultimate forecast for every category.

4. Results and Discussions

The efficacy of the ensemble model has been assessed by several critical tests described in this section. To run the ensemble model, a Jupyter Notebook version 6.4.6 was used. Python code is simple to run and write using Jupyter Notebook, a well-liked open-source tool for creating and executing neural network algorithms for classification. To evaluate the effectiveness of the ensemble model, five parameters have been used to assess the final forecast produced by the ensemble model. The efficiency of the ensemble model described in the current work has been assessed using accuracy, precision, recall, and F1 score [38, 39]. The proportion of accurate forecasts to total forecasts is known as accuracy. It refers to a classification approach's capacity to forecast the classes in a dataset. It demonstrates how the algorithms are accurately classified—equation (6), which calculates. Accuracy is defined as the ratio of correct forecasts to all forecasts.

$$Accuracy = \frac{True\_Positive + True\_Negative}{All\ Instances} \qquad (6)$$

The recall is also known as sensitivity which recovered pertinent occurrences. The recall is the proportion of true positives to all other positives. The recall is calculated using Equation (7).

$$\text{Recall} = \frac{\text{True\_Positive}}{\text{True\_Positive}+\text{False\_Negative}} \qquad (7)$$

The ratio of true positive outcomes to all anticipated positive results is known as precision. The precision is calculated using Equation (8).

$$Precision = \frac{True\_Positive}{True\_Positive + False\_Positive} \tag{8}$$

F1-score serves as a gauge for test accuracy. In terms of recall and precision, it is a skewed mean. It is calculated using criteria for recall and precision. The greatest F1 score is achieved when memory and accuracy are equal. The F1 score is determined using Equation (9).

$$F1 - Score = \frac{2*Recall*Precision}{Recall+Precision} \tag{9}$$

Because the dataset contains statistical information for more than 24,000 classes and 2000 packages, the produced dataset may be used as an empirical basis for developing and using data-driven reusability evaluation techniques. The qualities of the essential software element are described by 28 of the 32 attributes included in the dataset. In all, 27494 examples are employed in this research. The dataset is initially pre-processed to remove any occurrences with missing values and outliers. The target characteristic (ReuseRate), originally a numerical number, is then changed to an enumeration type with three categories: Low, Medium, and High. The dataset is split into 20% for testing and 80% for training.

The SVM, KNN, DT, ANN, and NB base classifiers have been implemented in this study to compare the effectiveness of the ensemble model against these methods. Accuracy, precision, recall, and F1 score have all been used as evaluation criteria to determine the extent to which these classification algorithms worked. Each prediction algorithm has been evaluated using 10-fold cross-validation, with a 95% testing error, to avoid overfitting. Experimental outcomes for accuracy, precision, recall, and F1 score for various foundational models are presented in Table 1. Fig. 2. illustrates how the ensemble model, which had an accuracy rating of 89.48%, accomplished higher. The accuracy of the NB model ranked the lowest, reaching 81.93%. Focusing on the F1 score, the ensemble model produced superior results (0.9445), while the NB model produced the least satisfactory results (0.9007). Furthermore, the ANN had the lowest recall scores (0.9049), while the ensemble model had the best precision assessments (0.9406). However, the ensemble model fared second best, with a recall of 0.9484, and the ANN model provided the most astounding results (0.9759), while the NB model gave the least impressive results (0.886 only). The confusion matrix for the different fundamental models, including SVM, KNN, DT, ANN, NB, and the suggested ensemble model, is shown in Fig. 3.

Table 1. Comparative performance of the various models

Model	Accuracy	Precision	Recall	F1 Score
SVM	86.84%	0.9154	0.9442	0.9296
KNN	84.96%	0.9166	0.9207	0.9186
DT	88.33%	0.9269	0.9494	0.938
ANN	88.51%	0.9049	0.9759	0.9391
NB	81.93%	0.9159	0.886	0.9007
Proposed Ensemble	89.48%	0.9406	0.9484	0.9445

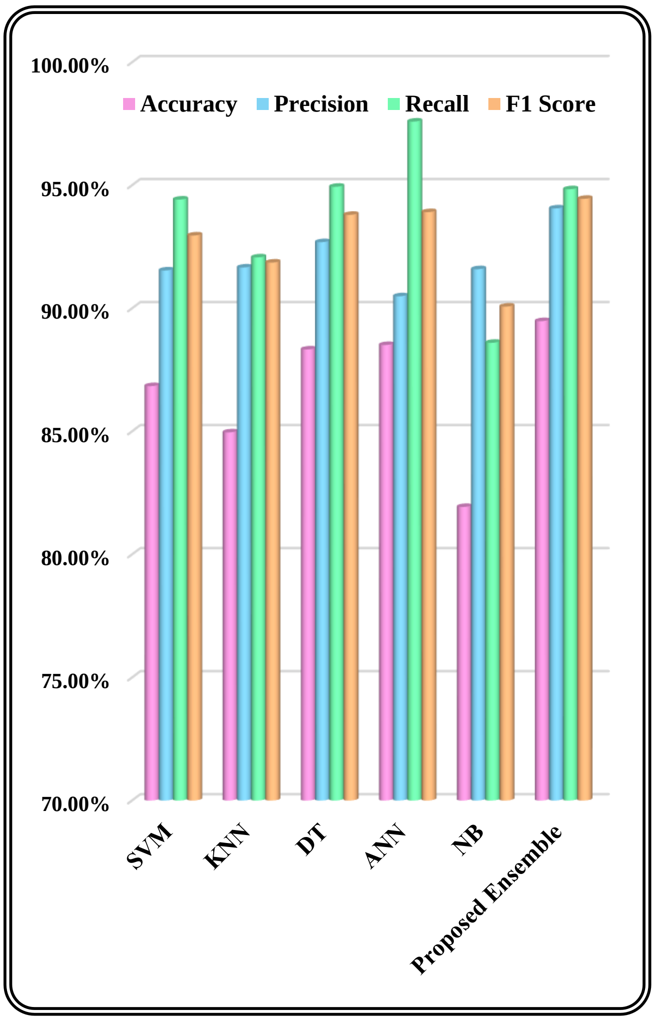

Fig. 1. Comparative performance of the various models

5. Conclusions

Software reusability improves the software development process since it spares the programmer the time, money, and effort required to implement the identical basic functionality repeatedly. Reusability of software is a cost-effective strategy that benefits the programmer and produces software of excellent quality. A software component's reusability is evaluated using a variety of measures. These metrics

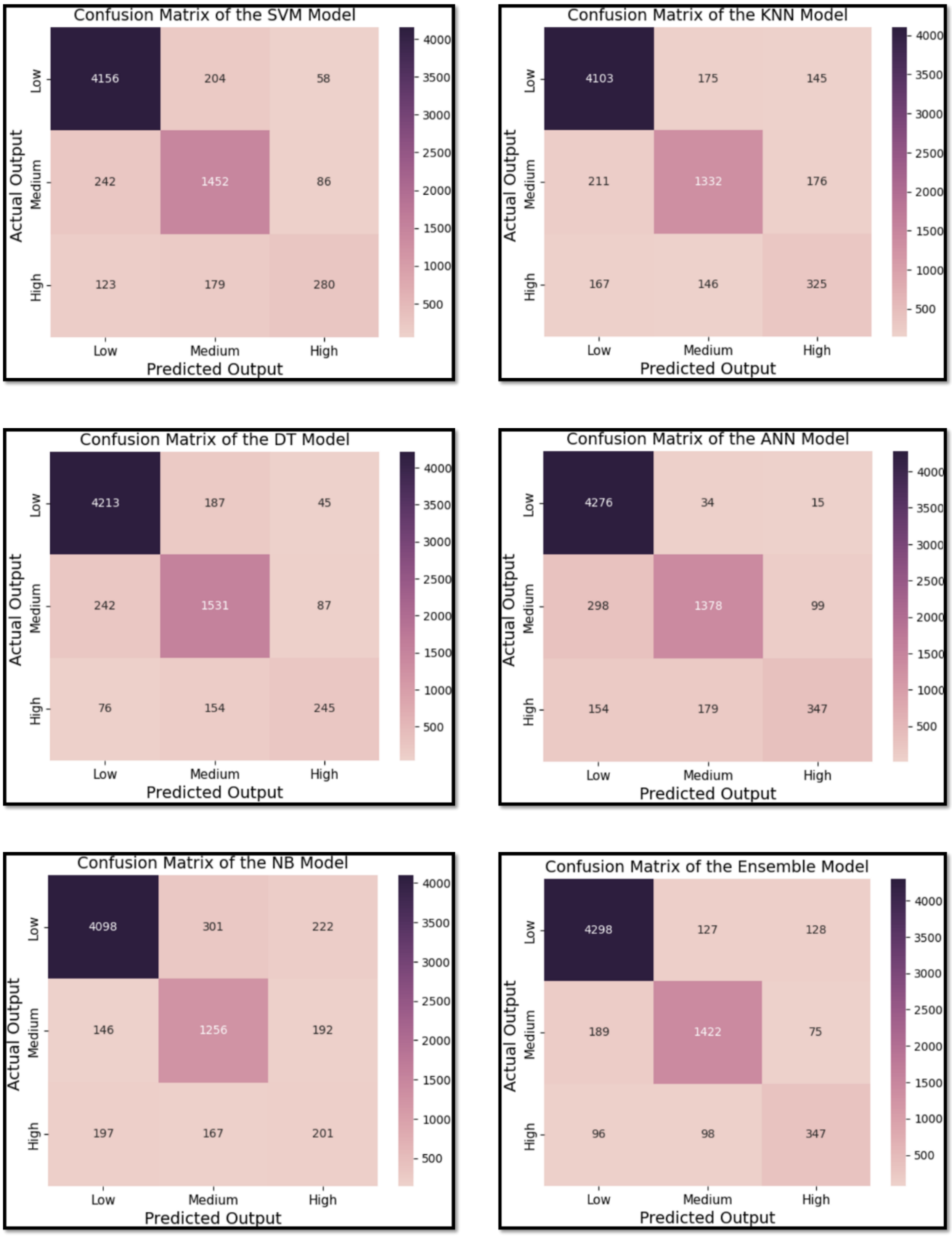

Fig. 3. Confusion Matrices of various models

support developing and using reusable software components through machine-learning approaches. The machine learning methodologies, as well as their related metrics that aid in determining reusability, are explained by the taxonomical cartography. The present investigation seeks to build an ensemble machine-learning model that combines SVM, KNN, DT, ANN, and NB to assess reusability. The publicly available benchmark dataset is utilised for experimentation after pre-processing. The recommended model outperformed basic classifiers in accuracy, precision, recall, and f1-score, with values of 89.48%, 0.9406, 0.9484, and 0.9445, respectively. The evaluation of our method indicates that it can effectively measure reusability as experienced by engineers.

Author contributions

Srishti Bhugra: Conceptualization, Methodology, Software, Visualization, Investigation, Writing-Reviewing and Editing **Puneet Goswami:** Data curation, Writing-Original draft preparation, Software, Validation.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] R. Feldt, F. de Oliveira Neto and R. Torkar, "Ways of applying artificial intelligence in software engineering," in *Proceedings of the 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering*, 2018.

[2] S. Martínez-Fernández, J. Bogner, X. Franch, M. Oriol, J. Siebert, A. Trendowicz, A. Vollmer and S. Wagner, "Software engineering for AI-based systems: a survey," *ACM Transactions on Software Engineering and Methodology (TOSEM),* vol. 31, no. 2, pp. 1-59, 2022.

[3] R. Ma, E. Sun and J. Zou, "A spectral method for assessing and combining multiple data visualizations," *Nature Communications,* vol. 14, no. 1, p. 780, 2023.

[4] P. Goswami, A. Noorwali, A. Kumar, M. Khan, P. Srivastava and S. Batra, "Appraising Early Reliability of a Software Component Using Fuzzy Inference," *Electronics,* vol. 12, p. 1137, 2023.

[5] D. Wangoo, "Artificial intelligence techniques in software engineering for automated software reuse and design," in *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, 2018.

[6] "ISO/IEC 25010:2011," [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en. [Accessed 23 March 2023].

[7] S. Pfleeger and J. Atlee, Software engineering: theory and practice, Pearson Education India, 2010.

[8] M. Papamichail, T. Diamantopoulos and A. Symeonidis, "Measuring the reusability of software components using static analysis metrics and reuse rate information," *Journal of Systems and Software,* vol. 158, p. 110423, 2019.

[9] R. Qayyum, J. Rubaab, U. Riaz and F. Arif, "Role of Data Mining and Machine Learning in Software Reusability," in *2021 International Conference on Innovative Computing (ICIC)*, 2021.

[10] A. Singh and P. Tomar, "Estimation of component reusability through reusability metrics," *International Journal of Computer and Information Engineering,* vol. 8, no. 11, pp. 2018-2025, 2014.

[11] L. Heinemann, F. Deissenboeck, M. Gleirscher, B. Hummel and M. Irlbeck, "On the extent and nature of software reuse in open source java projects," in *Top Productivity through Software Reuse: 12th International Conference on Software Reuse, ICSR 2011, Pohang, South Korea, June 13-17, 2011. Proceedings 12*, 2011.

[12] I. Mojica, B. Adams, M. Nagappan, S. Dienst, T. Berger and A. Hassan, "A large-scale empirical study on software reuse in mobile apps," *IEEE software,* vol. 31, no. 2, pp. 78-86, 2013.

[13] J. Bansiya and C. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Transactions on software engineering,* vol. 28, no. 1, pp. 4-17, 2002.

[14] O. Rotaru and M. Dobre, "Reusability metrics for software components," in The 3rd ACS/IEEE International Conference onComputer Systems and Applications, 2005, 2005.

[15] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi and T. Zimmermann, "Software engineering for machine learning: A case study," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2019.

[16] H. Ammar, W. Abdelmoez and M. Hamdi, "Software engineering using artificial intelligence techniques: Current state and open problems," in *Proceedings of the First Taibah University International Conference on Computing and Information Technology (ICCIT 2012)*, Al-Madinah Al-Munawwarah, Saudi Arabia, 2012.

[17] L. Kaur and A. Mishra, "An empirical analysis for predicting source code file reusability using meta-classification algorithms," in *Advanced Computational and Communication Paradigms: Proceedings of International Conference on ICACCP 2017*, Singapore, 2018.

[18] N. Chythanya and L. Rajamani, "Neural Network Approach for Reusable Component Handling," in *2017 IEEE 7th International Advance Computing Conference (IACC)*, 2017.

[19] M. Papamichail, T. Diamantopoulos and A. Symeonidis, "Software reusability dataset based on static analysis metrics and reuse rate information," *Data in brief,* vol. 27, p. 104687, 2019.

[20] "SourceMeter," [Online]. Available: https://www.sourcemeter.com/. [Accessed 25 March 2023].

[21] S. Batra and S. Sachdeva, "Organizing standardized electronic healthcare records data for mining," *Health Policy and Technology,* vol. 5, no. 3, pp. 226-242, 2016.

[22] S. Batra and S. Sachdeva, "Pre-processing highly sparse and frequently evolving standardized electronic health records for mining," in *Handbook of Research on Disease Prediction Through Data Analytics and Machine*

Learning, IGI Global, 2021, pp. 8-21.

[23] S. Mishra, P. Mallick, H. Tripathy, A. Bhoi and A. González-Briones, "Performance evaluation of a proposed machine learning model for chronic disease datasets using an integrated attribute evaluator and an improved decision tree classifier," *Applied Sciences*, vol. 10, no. 22, p. 8137, 2020.

[24] H. Polat, H. Danaei Mehr and A. Cetin, "Diagnosis of chronic kidney disease based on support vector machine by feature selection methods," *Journal of medical systems*, vol. 41, pp. 1-11, 2017.

[25] A. Pathak, S. Batra and H. Chaudhary, "Imputing Missing Data in Electronic Health Records," in Proceedings of 3rd International Conference on Machine Learning, Advances in Computing, Renewable Energy and Communication: MARC 2021, Singapore, 2022.

[26] A. Pathak, S. Batra and V. Sharma, "An Assessment of the Missing Data Imputation Techniques for COVID-19 Data," in Proceedings of 3rd International Conference on Machine Learning, Advances in Computing, Renewable Energy and Communication: MARC 2021, Singapore, 2022.

[27] K. Chandel, V. Kunwar, S. Sabitha, T. Choudhury and S. Mukherjee, "A comparative study on thyroid disease detection using K-nearest neighbor and Naive Bayes classification techniques," *CSI transactions on ICT*, vol. 4, pp. 313-319, 2016.

[28] S. Sachdeva, D. Batra and S. Batra, "Storage Efficient Implementation of Standardized Electronic Health Records Data," in *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2020.

[29] B. Boukenze, A. Haqiq and H. Mousannif, "Predicting chronic kidney failure disease using data mining techniques," in *Advances in Ubiquitous Networking 2: Proceedings of the UNet'16 2*, 2017.

[30] S. Sachdeva, M. Singh, N. Kumar and P. Goswami, "Personalized e-learning based on ant colony optimization," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 30, no. 1, 2022.

[31] S. Perveen, M. Shahbaz, A. Guergachi and K. Keshavjee, "Performance analysis of data mining classification techniques to predict diabetes," *Procedia Computer Science*,, vol. 82, pp. 115-121, 2016.

[32] R. Dhruvi, P. Yavnika and R. Nutan, "Prediction of Probability of Chronic Diseases and Providing Relative Real-Time Statistical Report using data mining and machine learning techniques," *International Journal of Science, Engineering and Technology Research (IJSETR)*, vol. 5, no. 4, 2016.

[33] R. Ani, G. Sasi, U. Sankar and O. Deepa, "Decision support system for diagnosis and prediction of chronic renal failure using random subspace classification," in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2016.

[34] S. Ramya and N. Radha, "Diagnosis of chronic kidney disease using machine learning algorithms," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 4, no. 1, pp. 812-820, 2016.

[35] B. Deekshatulu and P. Chandra, "Classification of heart disease using k-nearest neighbor and genetic algorithm," *Procedia technology*, vol. 10, pp. 85-94, 2013.

[36] S. Bashir, U. Qamar, F. Khan and L. Naseem, "HMV: A medical decision support framework using multi-layer classifiers for disease prediction," *Journal of Computational Science*, vol. 13, pp. 10-25, 2016.

[37] S. Bashir, U. Qamar, F. Khan and M. Javed, "MV5: a clinical decision support framework for heart disease prediction using majority vote based classifier ensemble," *Arabian Journal for Science and Engineering*, vol. 39, pp. 7771-7783, 2014.

[38] S. Batra, H. Sharma, W. Boulila, V. Arya, P. Srivastava, M. Z. Khan and M. Krichen, "An Intelligent Sensor Based Decision Support System for Diagnosing Pulmonary Ailment through Standardized Chest X-ray Scans," *Sensors*, vol. 22, no. 19, p. Sensors, 2022.

[39] S. Batra, R. Khurana, M. Z. Khan, W. Boulila, A. Koubaa and P. Srivastava, "A Pragmatic Ensemble Strategy for Missing Values Imputation in Health Records," *Entropy*, vol. 24, no. 4, p. 533, 20

[40] Ch.Sarada, C., Lakshmi, K. V. ., & Padmavathamma, M. . (2023). MLO Mammogram Pectoral Masking with Ensemble of MSER and Slope Edge Detection and Extensive Pre-Processing. International Journal on Recent and Innovation Trends in Computing and Communication, 11(3), 135–144. https://doi.org/10.17762/ijritcc.v11i3.6330

[41] Chang Lee, Deep Learning for Speech Recognition in Intelligent Assistants , Machine Learning Applications Conference Proceedings, Vol 1 2021.