Accuracy enhancement of Component based selection model using Hybrid Soft computing

Anjali Banga
Department of Computer Science Engineering
Guru Jambheshwar University of Science & Technology
Hisar, Haryana,India
banga.anjali88@gmail.com

Pradeep Kumar Bhatia
Department of Computer Science Engineering
Guru Jambheshwar University of Science & Technology
Hisar,Haryana, India
pkbhatia.gju@gmail.com

***Abstract*— Accuracy enhancement of Component based selection model using Hybrid Soft computing Abstract: Need of component based selection is growing day by day. There have been several researches in existence that analyzed existing component based selection model using soft computing techniques. Several research works focused on optimizing software component selection model in CBSE using soft computing techniques. Thus there is need to improve the performance of component based selection model using hybrid soft computing techniques. Several performance parameters are considered during data analytics operation. Organization tends to utilize mapping tools in order build sustainable designs for their processes as well as capabilities. During data analytics, acquisition, security, governance and standards, Insights and analysis, storage, visualization, optimization operation are performed. Proposed research has integrated PSO optimization technique with ANN based neural network to enhance the accuracy of component based selection model.**

Keywords*— Component based selection, soft computing techniques, Hybrid soft computing techniques

I. INTRODUCTION

The popularity of component-based frameworks continues to rise as their reuse practises reduce software development costs, development times, and developer labour. A CBSS may be constructed using pre-existing, pre-tested components that the author can immediately put to use. Each of the several parts is unique in terms of its design, user interface, nomenclature, and physical composition. The function of a component may be inferred from its name; the interface is the point at which data enters the component; the component's body is the site of input and output. CBSD authors pull together various parts from a variety of design papers to create a whole.

A. Component Based Software Engineering

When designing and building computer-based systems, CBSE prioritizes the usage of reusable software components. It does more than just find potential pieces; it verifies the interfaces of those pieces, fixes any design flaws they may have, assembles them according to a predetermined architectural style, and keeps them current as system needs evolve. Component-based software development happens simultaneously with the process model for component-based software engineering.

*1) Component-based development:*CBSE includes the practise of CBD, which happens along with domain engineering. The software team develops an acceptable architectural style for the analytical model of the application they are building via the use of analysis and architectural design techniques.

2) CBSE Framework Activities: Component-Based Software Engineering's framework entails the following actions:

a) Component Qualification: By doing this task, you can be guaranteed that the system architecture will establish the specifications needed for the components to become reusable. Components that may be reused are often recognised by the characteristics of their interfaces.

b) Component Adaptation: To do this, the architecture must establish the design criteria for each component and specify how those components will link to one another. Existing reusable components may not always be permitted to be employed owing to the architecture's design rules and restrictions.

c) Component Composition: This step verifies that the software parts have been properly integrated according to the system's Architectural style. The architecture explains the final product's structure by pointing out its connecting and coordinating systems.

d) Component Update: This procedure guarantees that all reusable parts are current. As a result of integrating new features and functionality from third-party sources, upgrades may (the organisation that developed the reusable component may be outside the immediate control of the software engineering organisation accessing the component currently).

3) Component-Based Models Life Cycle Proces: Component-based software engineering employs almost identical approaches, instruments, and ideas to those used in software engineering generally. Nonetheless, there are distinguishing features.

a) Building systems from components: The principle underlying this is its potential for reuse. This implies that systems are assembled from parts that already exist. There are, however, certain drawbacks to using this strategy. This section lists a few of the repercussions.

- Component-based systems have their own unique set of procedures for creating new features and fixing bugs, which are distinct from the procedures used to create individual components.
- Component discovery and evaluation therefore become a whole new procedure.

979-8-3503-7462-9/24/$31.00 ©2024 IEEE
DOI 10.1109/CCICT62777.2024.00035

- Differences exist between process-based and non-component-based activities

b) Requirements analysis and specification: This process involves examining how the proposed fix measures up to the specifications. It is determined whether the available parts are enough by comparing them to the specifications.

c) System and software design: Similarly to the preceding stage, this one is dependent on the accessibility of the necessary parts. The prospective components should be compatible with a certain component model.

d) Implementation and unit testing: The system's components should be integrated during construction. Connectivity is specified through "glue code," a term used by the authors.

e) System Integration: The component framework's basic infrastructure components are combined with the application's own components. We sometimes refer to this as "component deployment.

f) System verification and validation: It's recommended to stick with tried and true methods. An error's precise location is an example of a challenge unique to the component-based strategy. Black-box components from several manufacturers are used here. As a result of another component's failure, the first one may display an error.

g) Operation support and maintenance: Essentially, it's not dissimilar to the integrating process. A system updates a part of itself. Most of the time, a change is made to an already existing part. An alternative is to include a new variant of the same component.

h) Component Model Implementations and Services: The components' operation necessitates a run-time environment. Commonality in the runtime environment is desirable. This encompasses the broad range of run-time services as well as those that are specialised to certain domains. Object creation, life cycle management, object persistence support, etc. are all examples of general services.

There is a lack of adherence to CBSE concepts among the development teams. As a result, the development processes can't benefit from a component-based strategy. The method relies on recycling preexisting materials. The amount of work required to put this into effect is drastically lowered. However, this complicates system verification even more. There has to be a readjustment here as the development process progresses. By analysing real-world examples from a wide range of sectors, we learn that total separation of the development process is very difficult to attain. It also places a heavy emphasis on system and component verification, as well as architectural concerns.

B. *Soft computing techniques*

In order to address the complicated computational challenge, we turn to soft computing approaches, which provide us with the computing methods and helpful algorithms needed to handle the issue. In other words, it tells us how much room for error we have in a specific system or situation. This method is sometimes referred to as computational intelligence since it enables us to present and solve issues that are either very resource-intensive for hardware or are intractable. In the next portion of the tutorial, we will examine its operation and the many contexts in which it may be used to improve our system's functionality and performance and to provide a comprehensive solution to the involved issue.

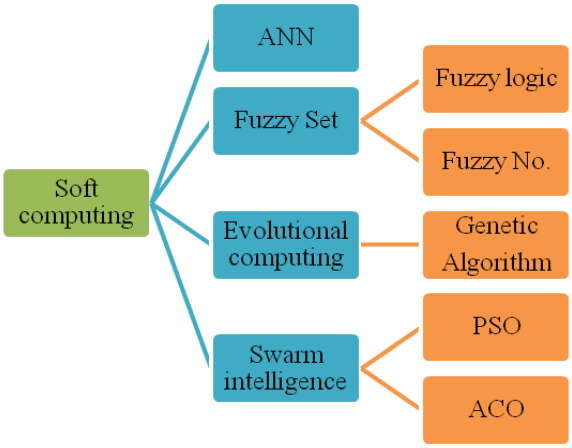

Fig. 1. Soft computing Techniques

The following fall under one of three categories of soft computing methods.

1) Artificial Neural Network: It uses a parallel distributed network and a connectionist approach. The two main varieties of neural networks are ANN and BNN. A unit is a neural network that operates on a single element. Input, mass, the processing element, and output are the unit's constituent parts. It works similarly to the way the human brain does. Using electrical impulses for communication, artificial neural networks are able to solve problems in parallel, which is their fundamental benefit. The primary drawback is that artificial neurons are not resilient to failure; if even one is destroyed, the system fails.

2) Fuzzy Logic: Models based on logical reasoning, but subject to imprecision and fuzziness, are solved using the fuzzy logic method. Invented by Latzi A. Zadeh, it first appeared in 1965. The truth value in fuzzy logic is always between 0 and 1, hence the range is closed. In where 0 indicates a fake value and 1 indicates a correct one.

3) Swarm Intelligence: In the field of artificial intelligence known as swarm intelligence (SI), researchers examine the emergent collective behaviour in self-organizing communities of agents. The study of social behaviour in natural communities, such as the migration of birds and fish, had a major influence on the development of SI.

4) Genetic Algorithm in Soft Computing: John Holland, a professor at Cambridge University, first described the genetic algorithm in 1965. Evolutionary algorithms are used to find solutions to issues using the same concepts as natural selection. Ant colonies and swarm particles are two common optimization techniques used for issues involving the maximising and minimization of objective functions. It's in line with genetics and evolution and other biological processes.

II. LITERATURE REVIEW

Kumar Singh et al. (2016) focused on a novel approach for improving testing technique of component based software. This study presents a strategy for better testing component-based software, analyses its efficacy, and highlights problems specific to testing the completed program's individual components. Some suggestions were also provided to enhance the testing method for component-based applications [1].

Vale et al. (2016) introduced twenty-eight years of component-based software engineering. This study discusses five aspects of CBSE based on the data that is currently available: primary goals, research subjects, fields of application, intensity of research, and applied research methodologies. There was a dramatic uptick in study effort within the previous fourteen years. This report does more than just analyze the data; it also synthesizes the existing evidence, identifies ongoing concerns, and highlights research gaps [2].

Oliveira et al. (2016) presented rigorous development of component-based systems using component metadata and patterns. In this research, they focus on analysis speed and provide a much faster replacement for checking the rule side conditions by doing partial verification on component information throughout all stages of a component's creation and by making use of behavioural patterns [3].

Qi et al. (2016) reviewed weighted principal component analysis-based service selection method for multimedia services in cloud. Due to these difficulties, they propose a W PCA MSSM for choosing among available services. Compared to other options, ours offers two distinct benefits [4].

Parsaie et al. (2018) provided applications of soft computing techniques for prediction of energy dissipation on stepped spillways. The results of the constructed models were reviewed, and it was determined that all of them had enough performance to estimate the energy dissipation. When compared to the rest of the methods, however, MARS and SVM provide the most reliable results. Attention to structures of GMDH and MARS models indicated that Froude number, drop number, and ratio of critical depth to height of step are the most essential factors for modelling energy dissipation [5].

Gupta et al. (2018) focused on hybrid leakage management for water network using psf algorithm and soft computing techniques. This research introduces a pressure management strategy that combines maximizing the tank's water storage level with optimizing the control and localization of the PRV in a water distribution system to provide a hybrid model for decreasing leakage. [6]

Kaur et al. (2018) introduced an analytic review on image enhancement techniques based on soft computing approach. In this article, they will go through many methods for improving images by making use of soft computing methods. Genetic algorithm, fuzzy-based enhancement, neural networks, and optimization methods are some of the methods used [7].

Delafrouz et al. (2018) reviewed a novel hybrid neural network based on phase space reconstruction technique for daily river flow prediction. The primary goal of this research is to improve the precision of daily river flow forecasts by developing a novel hybrid model (PSR-ANN) that utilizes both PSR and ANN methods. The data utilized in this study came from three different measuring sites in the United States and tracked the flow of several rivers [8].

Diwaker et al. (2018) presented prediction of software reliability using bio inspired soft computing techniques. Consequently, it is important to pinpoint the variables that have the most impact on the system's dependability. Nowadays, reusability is widely applied in many fields of study. CBS are built on the principle of reusability [9].

Gupta et al. (2018) explained advances in applying soft computing techniques for big data and cloud computing. Soft computing refers to a group of techniques that are unique in their approach and provide consistent results [10].

Bhardwaj et al. (2018) provided quality assurance through soft computing techniques in component based software. In this study, the author suggests a method of testing called "monkey testing" to ensure the system's continued high quality and safety [11].

Shreyas et al. (2019) focused on application of soft computing techniques in tunnelling and underground excavations: state of the art and future prospects. This article delves into the use of ANNs, RBFs, DTs, the RF method, SVMs, nonlinear regression techniques like MARS, and hybrid intelligent models for predicting the engineering response of tunnels and underground excavations [12].

Banga, A., et. Al (2021) focused on training and evaluating a dataset after optimizing a software component using a deep neural network technique. The chosen components must be adjusted and adaptable according to the specified requirements. The suggested technique utilizes an LSTM layer to enhance the accuracy of the output [13].

Banga, A., et al. (2021) focused on the creation of an optimization model for selecting software components in the construction of CBSS. The proposed model aims to enhance the functional performance of CBSS. ACO has been used in prior studies to tackle optimization problems [14].

Bousmaha, R., et al. (2021) presented a novel training approach called hybrid PSO with PLMVO. The purpose of this approach is to maximize both the number of hidden neurons and connection weights in FFNN concurrently. The hybrid method is used to enhance search performance in the solution space, hence demonstrating its effectiveness in mitigating the issue of being stuck in local minima [15].

Gupta, Dr. V. et al. (2021) proposed Software engineering is a field that provides standardized approaches for developing, operating, and maintaining software. The major objective of the study is to identify a mechanism capable of generating the required level of force consistently throughout the testing phase [16].

Wang, R., et al. (2023) focused PS) is a straight forward and effective technique that uses a population-based approach to solve a wide range of optimization problems. The enhanced DMS technique incorporates the Quasi-Newton method to generate the exploitation subpopulation [17].

III. Problem Statement

There have been several soft computing techniques that have been used for component selection in conventional researches. These techniques could be optimization or machine learning approach. But it has been observed that hybrid soft computing techniques are capable to enhance the scalability, reliability and efficiency of system. Thus proposed work is making use of hybrid soft computing mechanism in order to resolve the issues found in conventional soft computing mechanisms.

IV. Proposed Work

Present research has considered research in area of component based software selection along with soft computing techniques. Fuzzy computing, neural network, swarm intelligence are soft computing techniques. Research is considering ANN from neural network and PSO from swarm intelligence to propose hybrid approach. Process flow of Software component selection is shown below:

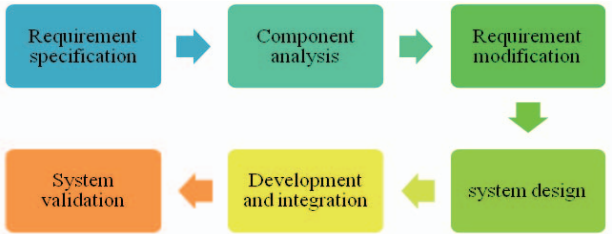

Fig. 2. Process of Software component selection

After considering the research related to component based software selection and soft computing, issues of conventional researches are identified and hybrid model is proposed that is considering ANN and PSO optimization mechanism.

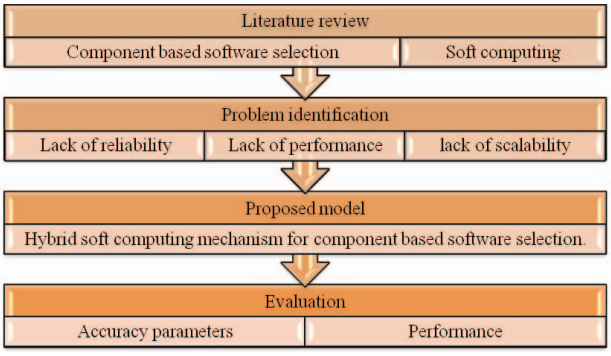

Fig. 3. Process flow of proposed work

In proposed research component selection and classification has been made by integration of optimizer and machine learning approach. Weighted method class and depth of inheritance are calculated for software component. Then PSO optimizer is applied to get best solution. Component whose corresponding WMC and DIT is less than optimized value are filtered and this filtered dataset is transferred to ANN model for training in order to perform classification of software components.

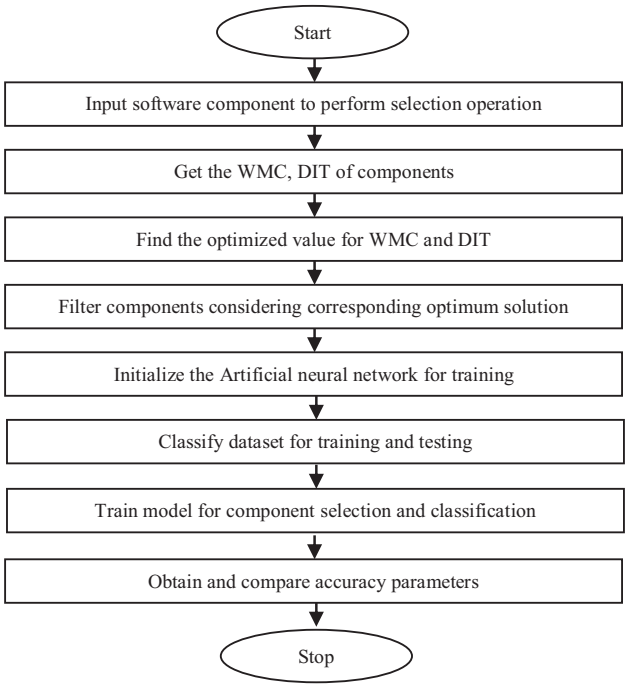

Fig. 4. Flow chart for Hybrid soft computing for CBSE model

A. *Hybridization of PSO and ANN*

The integration of ANN and PSO is a popular approach in optimization problems, leveraging the strengths of both techniques to enhance performance. The selection criteria for integrating these methods depend on the characteristics of the problem at hand and the desired optimization goals. ANNs are powerful machine learning models inspired by the structure and function of the human brain. They excel in learning complex patterns and relationships from data, making them suitable for various tasks such as classification, regression, and function approximation. PSO is a population-based stochastic optimization technique inspired by the social behavior of bird flocking or fish schooling. The hybridization of ANN and PSO involves using PSO to optimize the parameters of the neural network model. This integration offers several advantages:

a) Global Search Capability: PSO's ability to explore the search space efficiently enables the ANN to avoid local optima and find better solutions.

b) Parameter Tuning: ANNs often require careful tuning of parameters such as learning rates, network architecture, and activation functions. PSO can automate this process by searching for optimal parameter values.

c) Convergence Speed: PSO can accelerate the convergence of ANNs by guiding the search towards promising regions of the solution space.

d) Robustness: The hybrid approach enhances the robustness of the optimization process, making it less sensitive to the choice of initial conditions and hyperparameters.

Overall, the hybridization of ANN and PSO offers a robust and efficient approach to optimization problems, leveraging the complementary strengths of both techniques to achieve superior performance and scalability.

B. Dataset Details

In component-based selection simulations, the choice of dataset is crucial as it directly impacts the evaluation and validation of the selection process. Here's detailed information about a hypothetical dataset used for such simulations:

1) Dataset Source: The dataset used for simulations in component-based selection could be sourced from various sources, such as publicly available repositories, real-world industry data, or generated synthetically for research purposes. In present research source of data is kaggle.com

2) Size: The size of the dataset is below 10 MB that is sufficient to represent a diverse range of software components while being manageable for computational resources. For example, the dataset could contain thousands to tens of thousands of software components, each characterized by various attributes.

3) Characteristics: The dataset should include relevant characteristics or features of software components that are crucial for the selection process. These characteristics could include:

a) Functional Attributes: Such as the type of functionality provided by the component .

b) Technical Attributes: Such as programming language, compatibility with platforms/frameworks, dependencies, and performance metrics.

c) Quality Attributes: Including reliability, scalability, security, and maintainability metrics.

d) Metadata: Such as version history, documentation availability, license type, and user ratings.

4) Simulation Setup: The simulation setup involves defining the experimental framework and parameters for evaluating the component-based selection process. This includes:

a) Selection Criteria: Clearly defining the criteria or objectives for component selection, such as maximizing functionality coverage, minimizing development effort, or optimizing for performance.

b) Evaluation Metrics: Determining the metrics to evaluate the effectiveness of the selection process, such as precision, recall, F1-score, or utility measures tailored to specific objectives.

c) Simulation Scenarios: Designing different simulation scenarios to represent diverse selection scenarios and constraints, such as resource limitations, compatibility requirements, or trade-offs between conflicting objectives.

d) Cross-Validation: Implementing cross-validation techniques to ensure the robustness and generalization of the selection process by splitting the dataset into training and testing sets.

5) Parameters: Various parameters influence the simulation process, including:

a) Algorithm Parameters: If using machine learning or optimization algorithms for selection, parameters such as learning rates, convergence criteria, population size, and mutation rates need to be specified.

b) Simulation Environment: Details about the computational environment, such as hardware specifications, software dependencies, and runtime configurations.

c) Simulation Duration: Duration of the simulation experiment, including the number of iterations or epochs for iterative algorithms.

Overall, the simulation setup and parameters should be carefully defined to accurately reflect the complexities and nuances of real-world component-based selection scenarios, ensuring reliable and meaningful evaluations of selection strategies and algorithms.

V. SIMULATION RESULT

The provided metrics present a comparison between the performance of a classification model on a non-optimized dataset and an optimized dataset across five different classes. In the non-optimized dataset, the model achieved accuracy rates ranging from 93.44% to 98.31% across the five classes, with consistent precision of 0.92 for all classes. However, there were variations in recall and F1-score, indicating differences in the model's ability to correctly identify instances and balance precision and recall. Specifically, while Classes 1, 2, 3, and 4 demonstrated high recall and F1-scores (ranging from 0.94 to 0.97), Class 5 exhibited lower recall (0.81) and F1-score (0.86), suggesting potential challenges in accurately classifying instances in this class.

On the other hand, the optimized dataset showed improvements in accuracy for all classes, with rates ranging from 95.97% to 98.53%. Notably, precision remained consistently high at 0.94 across all classes, indicating a robust ability to correctly classify instances. Moreover, the recall rates also improved across all classes compared to the non-optimized dataset, ranging from 0.95 to 0.96, demonstrating enhanced sensitivity in identifying instances. This improvement in recall contributed to higher F1-scores across all classes, with values ranging from 0.91 to 0.95.

Overall, the optimization of the dataset led to enhancements in model performance, particularly in terms of recall and F1-score, indicating improved sensitivity and balance between precision and recall. These improvements suggest that the optimized dataset provided better training data or feature representation, enabling the model to make more accurate predictions across all classes, including the previously challenging Class 5.

In present dataset there are 5 classes of components. Training and testing of dataset is made using ANN model without optimization has provided following confusion matrix:

TABLE I. CONFUSION MATRIX FOR WITHOUT OPTIMIZATION

	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 5**
Class 1	920	0	4	9	67
Class 2	6	460	5	1	28
Class 3	6	4	552	9	29
Class 4	0	8	0	644	48
Class 5	46	9	9	0	736

To extract accuracy, precision, recall, and F1 score from a confusion matrix, you first need to understand what each of these metrics represents:

1) Accuracy: It measures the proportion of correctly classified instances among the total number of instances. Mathematically, it's calculated as (TP + TN) / (TP + TN + FP + FN), where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

2) Precision: It measures the proportion of correctly predicted positive cases among all cases predicted as positive. Mathematically, it's calculated as TP / (TP + FP).

3) Recall (Sensitivity): It measures the proportion of correctly predicted positive cases among all actual positive cases. Mathematically, it's calculated as TP / (TP + FN).

4) F1 Score: It is the harmonic mean of precision and recall. It provides a single score that balances both precision and recall. Mathematically, it's calculated as 2 * (precision * recall) / (precision + recall).

Step-by-step process:

a) Calculate the true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) for each class using the confusion matrix.

b) Use the formulas mentioned above to calculate accuracy, precision, recall, and F1 score for each class.

c) Optionally, you can compute micro-averaged and macro-averaged values for precision, recall, and F1 score across all classes.

This process enables you to extract evaluation metrics from the confusion matrix, providing insights into the performance of a classification model across different classes.

Results:

TP: 3312

Overall Accuracy: 92%

TABLE II. ACCURACY CHART FOR NON OPTIMIZED DATASET

Class	n (truth)	n (classified)	Accuracy	Precision	Recall	F1 Score
1	978	1000	96.17%	0.92	0.94	0.93
2	481	500	98.31%	0.92	0.96	0.94
3	570	600	98.17%	0.92	0.97	0.94
4	663	700	97.92%	0.92	0.97	0.94
5	908	800	93.44%	0.92	0.81	0.86

Training and testing of dataset using ANN model after filter dataset using PSO has provided following confusion matrix:

TABLE III. CONFUSION MATRIX FOR PROPOSED WORK

	Class 1	Class 2	Class 3	Class 4	Class 5
Class 1	940	7	4	9	40
Class 2	6	470	5	3	16
Class 3	8	2	564	9	17
Class 4	3	7	8	658	24
Class 5	32	7	4	5	752

Result:

TP: 3384

Overall Accuracy: 94%

TABLE IV. ACCURACY CHART FOR PROPOSED WORK

Class	n (truth)	n (classified)	Accuracy	Precision	Recall	F1 Score
1	989	1000	96.97%	0.94	0.95	0.95
2	493	500	98.53%	0.94	0.95	0.95
3	585	600	98.42%	0.94	0.96	0.95
4	684	700	98.11%	0.94	0.96	0.95
5	849	800	95.97%	0.94	0.89	0.91

In the comparative analysis section, we assess the performance of our proposed hybrid approach against several benchmarks, including standalone ANN and PSO models, as well as other established techniques commonly employed in component-based software selection. Our evaluation aims to provide a comprehensive understanding of the effectiveness and practical utility of our approach in comparison to existing methodologies. Firstly, we compare our hybrid approach to standalone ANN and PSO models. We rigorously evaluate the performance of each individual technique across a range of metrics, including accuracy, convergence speed, and robustness.

By conducting experiments under various scenarios and constraints, we gain insights into the strengths and limitations of standalone ANN and PSO models in component-based selection tasks. Additionally, we benchmark our proposed approach against other prominent techniques utilized in the field. This includes methodologies such as Decision Trees, SVM, GA, and Rule-Based Systems. Each technique is evaluated based on its ability to handle the complexities of component selection, considering factors such as interpretability, scalability, and computational efficiency.

Furthermore, we establish appropriate benchmarks and evaluation metrics to ensure fair and consistent comparisons across different techniques. Real-world datasets or synthetic datasets with known ground truths are utilized to validate the performance of each approach under controlled conditions. Through extensive experimentation and analysis, we uncover valuable insights into the comparative strengths and weaknesses of our proposed hybrid approach relative. Comparison of Non optimized and optimized dataset overall accuracy during CBSE

TABLE V. COMPARISON OF OVERALL ACCURACY

Overall Accuracy for Non optimized CBSE model	Overall Accuracy for Optimized CBSE model
92	94

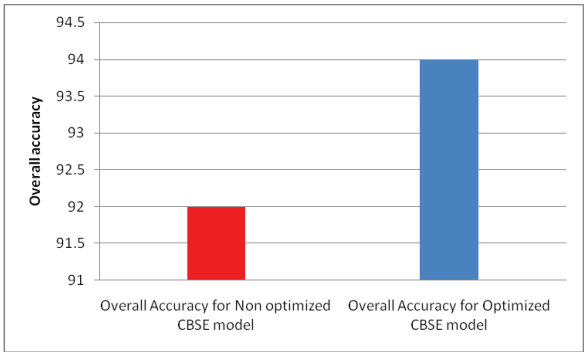

Fig. 5. Comparison of overall accuracy

TABLE VI. COMPARISON OF OVERALL LOSS

Overall Accuracy for Non optimized CBSE model	Overall Accuracy for Optimized CBSE model
8	6

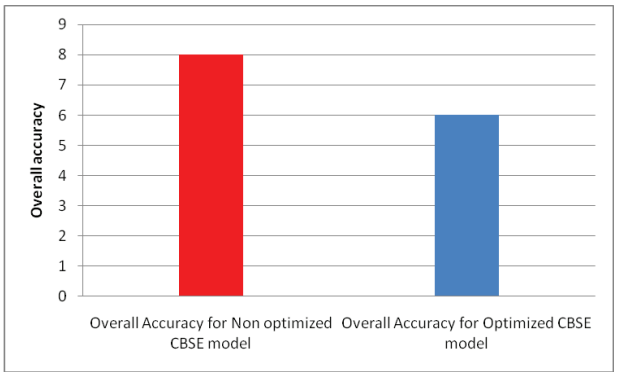

Fig. 6. Comparison of overall loss

When comparing the time consumption between an optimized and a non-optimized dataset, several factors could influence why the overall time consumption might be greater for the optimized dataset despite its optimizations.

1) Preprocessing and Feature Engineering: The optimized dataset may involve more complex preprocessing steps or feature engineering techniques to improve the quality of the data or to create more informative features.

2) Model Training: With an optimized dataset, models may take longer to train due to increased data volume or complexity resulting from feature engineering.

3) Hyperparameter Tuning: Optimizing a dataset often involves tuning model hyperparameters to achieve better performance. This process typically requires training multiple models with different hyperparameter configurations, which can significantly increase the overall time consumption compared to training a single model with default hyperparameters.

4) Cross-Validation: To ensure robustness and generalization of the model on the optimized dataset, cross-validation techniques might be employed.

5) Evaluation and Validation: After training, models on the optimized dataset may undergo more extensive evaluation and validation procedures to assess their performance thoroughly.

6) Iterative Optimization Process: Optimization is often an iterative process, where adjustments are made based on performance metrics, and models are retrained multiple times until satisfactory results are achieved.

7) Resource Constraints: Despite optimizations, the computational resources available for training and evaluation might be limited, leading to longer processing times due to queuing or resource contention.

Overall, while optimizations in the dataset can lead to better model performance, they often come at the cost of increased time consumption due to the additional steps involved in data preprocessing, model training, hyperparameter tuning, and evaluation. These trade-offs should be carefully considered based on the specific requirements and constraints of the project.

TABLE VII. COMPARISON OF OVERALL TIME CONSUMPTION

Overall Time consumption for Non optimized CBSE model (Seconds)	Overall Time consumption for Optimized CBSE model (Seconds)
345	245

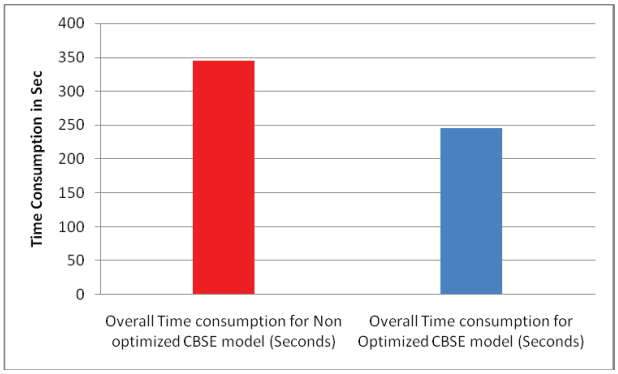

Fig. 7. Comparison of overall Time consumption for non optimized and optimized CBSE (Hybrid) model

VI. CONCLUSION

In the proposed research it has been observed that an CBSE model that did not consider optimized dataset is providing less accuracy. Whereas the proposed model is considering optimized dataset has provided more accuracy as compared to the conventional approach. Several factors that influenced the accuracy are the size of the dataset, batch size, number of epochs, and hidden layers used in the research. It is concluded that proposed model is yielding better accuracy, less error and better performance as compared to conventional models.

VII. FUTURE SCOPE OF RESEARCH

Component-based selection model is capable to provide more control and lower maintenance costs. Searching for

other applications where similar code might be used is reduced. It allows faster development to save time and increase revenue. Moreover it is also capable to take advantage of specialized skills. Thus hybrid approach proposed for CBSE model is capable to enhance its scalability, flexibility and efficiency.

REFERENCES

[1] R. K. Singh and S. K. Jha, "A novel approach for improving testing technique of component based software," 2016 5th Int. Conf. Reliab. Infocom Technol. Optim. ICRITO 2016 Trends Futur. Dir., pp. 135–138, 2016, doi: 10.1109/ICRITO.2016.7784940.

[2] T. Vale, I. Crnkovic, E. S. De Almeida, P. A. D. M. Silveira Neto, Y. C. Cavalcanti, and S. R. D. L. Meira, "Twenty-eight years of component-based software engineering," J. Syst. Softw., vol. 111, pp. 128–148, 2016, doi: 10.1016/j.jss.2015.09.019.

[3] M. V. M. Oliveira, P. Antonino, R. Ramos, A. Sampaio, A. Mota, and A. W. Roscoe, "Rigorous development of component-based systems using component metadata and patterns," Form. Asp. Comput., vol. 28, no. 6, pp. 937–1004, 2016, doi: 10.1007/s00165-016-0375-1.

[4] L. Qi, W. Dou, and J. Chen, "Weighted principal component analysis-based service selection method for multimedia services in cloud," Computing, vol. 98, no. 1–2, pp. 195–214, 2016, doi: 10.1007/s00607-014- 0413-x.

[5] A. Parsaie, A. H. Haghiabi, M. Saneie, and H. Torabi, "Applications of soft computing techniques for prediction of energy dissipation on stepped spillways," Neural Comput. Appl., vol. 29, no. 12, pp. 1393–1409, 2018, doi: 10.1007/s00521-016-2667-z.

[6] A. Gupta, N. Bokde, and K. D. Kulat, "Hybrid Leakage Management for Water Network Using PSF Algorithm and Soft Computing Techniques," Water Resour. Manag., vol. 32, no. 3, pp. 1133–1151, 2018, doi: 10.1007/s11269-017-1859-3.

[7] G. Kaur, N. Bhardwaj, and P. K. Singh, "An analytic review on image enhancement techniques based on soft computing approach," Adv. Intell. Syst. Comput., vol. 651, pp. 255–265, 2018, doi: 10.1007/978-981- 10-6614-6_26.

[8] H. Delafrouz, A. Ghaheri, and M. A. Ghorbani, "A novel hybrid neural network based on phase space reconstruction technique for daily river flow prediction," Soft Comput., vol. 22, no. 7, pp. 2205–2215, 2018, doi: 10.1007/s00500-016-2480-8.

[9] C. Diwaker, P. Tomar, R. C. Poonia, and V. Singh, "Prediction of Software Reliability using Bio Inspired Soft Computing Techniques," J. Med. Syst., vol. 42, no. 5, 2018, doi: 10.1007/s10916-018-0952-3.

[10] B. B. Gupta, D. P. Agrawal, S. Yamaguchi, and M. Sheng, "Advances in applying soft computing techniques for big data and cloud computing," Soft Comput., vol. 22, no. 23, pp. 7679–7683, 2018, doi: 10.1007/s00500-018-3575-1.

[11] O. Bhardwaj and S. Kumar Jha, "Quality assurance through soft computing techniques in component based software," Proc. 2017 Int. Conf. Smart Technol. Smart Nation, SmartTechCon 2017, pp. 277–282, 2018, doi: 10.1109/SmartTechCon.2017.8358382.

[12] S. K. Shreyas and A. Dey, "Application of soft computing techniques in tunnelling and underground excavations: state of the art and future prospects," Innov. Infrastruct. Solut., vol. 4, no. 1, 2019, doi: 10.1007/s41062-019-0234-z.

[13] Banga, A., & Bhatia, P. K. (2021). Optimized Component based Selection using LSTM Model by Integrating Hybrid MVO-PSO Soft Computing Technique. In Advances in Science, Technology and Engineering Systems Journal (Vol. 6, Issue 4, pp. 62–71). ASTES Journal. https://doi.org/10.25046/aj060408

[14] Banga, A., & Bhatia, P. K. (2021). Software Component Selection in CBSE Considering Cost, Reliability, and Delivery Delay Using PSO-integrated MVO and ALO. In Emerging Research in Computing, Information, Communication and Applications (pp. 455–479). Springer Singapore. https://doi.org/10.1007/978-981-16-1342-5_36

[15] Bousmaha, R., Hamou, R. M., & Amine, A. (2021). Automatic selection of hidden neurons and weights in neural networks for data classification using hybrid particle swarm optimization, multi-verse optimization based on Lévy flight. In Evolutionary Intelligence (Vol. 15, Issue 3, pp. 1695–1714). Springer Science and Business Media LLC. https://doi.org/10.1007/s12065-021-00579-w

[16] Gupta, Dr. V. (2021). STUDY OF PSO AND MVO OPTIMIZATION TECHNIQUES FOR TEST EFFORT ESTIMATION. In Innovative Research Thoughts (pp. 223–235). Shodh Sagar. https://doi.org/10.36676/irt.2021-v7i4-31

[17] Wang, R., Hao, K., Chen, L., Liu, X., Zhu, X., & Zhao, C. (2023). A modified hybrid particle swarm optimization based on comprehensive learning and dynamic multi-swarm strategy. In Soft Computing (Vol. 28, Issue 5, pp. 3879–3903). Springer Science and Business Media LLC. https://doi.org/10.1007/s00500-023-09332-0

[18] Y. Yu, C. Zhang, X. Gu, and Y. Cui, "Neural Comput. Appl., vol. 31, no. 12, pp. 8641–8660, 2019, doi: 10.1007/s00521-018-3679-7.

[19] Q. Zhou, P. Yan, H. Liu, and Y. Xin, "A hybrid fault diagnosis method for mechanical components based on ontology and signal analysis," J. Intell. Manuf., vol. 30, no. 4, pp. 1693–1715, 2019, doi: 10.1007/s10845- 017-1351-1.

[20] A. Tolba and E. Elashkar, "Soft computing approaches based bookmark selection and clustering techniques for social tagging systems," Cluster Comput., vol. 22, pp. 3183–3189, 2019, doi: 10.1007/s10586-018-2014- 5.

[21] N. Natarajan and C. Sudheer, "Groundwater level forecasting using soft computing techniques," Neural Comput. Appl., vol. 32, no. 12, pp. 7691–7708, 2020, doi: 10.1007/s00521-019-04234-5.

[22] K. Sheoran, P. Tomar, and R. Mishra, "A novel quality prediction model for component based software system using ACO–NM optimized extreme learning machine," Cogn. Neurodyn., vol. 14, no. 4, pp. 509– 522, 2020, doi: 10.1007/s11571-020-09585-7.

[23] N. Basurto and Á. Herrero, "Data Selection to Improve Anomaly Detection in a Component-Based Robot," Adv. Intell. Syst. Comput., vol. 950, pp. 241–250, 2020, doi: 10.1007/978-3-030-20055-8_23.

[24] A. Charitopoulos, M. Rangoussi, and D. Koulouriotis, "On the Use of Soft Computing Methods in Educational Data Mining and Learning Analytics Research: a Review of Years 2010–2018," Int. J. Artif. Intell. Educ., vol. 30, no. 3, pp. 371–430, 2020, doi: 10.1007/s40593-020-00200-8.

[25] N. Sharma et al., "Optimal deflection and stacking sequence prediction of curved composite structure using hybrid (FEM and soft computing) technique," Eng. Comput., vol. 37, no. 1, pp. 477–487, 2021, doi: 10.1007/s00366-019-00836-8.

[26] A. Mokni et al., "A formal approach for managing component-based architecture evolution To cite this version : HAL Id : hal-01380397," 2021.

[27] E. E. Başakın, Ö. Ekmekcioğlu, and M. Özger, "Drought prediction using hybrid soft-computing methods for semi-arid region," Model. Earth Syst. Environ., vol. 7, no. 4, pp. 2363–2371, 2021, doi: 10.1007/s40808- 020-01010-6.

[28] S. Ghani, S. Kumari, and A. Bardhan, "A novel liquefaction study for fine-grained soil using PCA-based hybrid soft computing models," Sadhana - Acad. Proc. Eng. Sci., vol. 46, no. 3, pp. 1–17, 2021, doi: 10.1007/s12046-021-01640-1.

[29] A. Radhika and M. S. Masood, "Effective dimensionality reduction by using soft computing method in data mining techniques," Soft Comput., vol. 25, no. 6, pp. 4643–4651, 2021, doi: 10.1007/s00500-020-05474-7.

[30] O. P. Singh, A. K. Singh, G. Srivastava, and N. Kumar, "Image watermarking using soft computing techniques: A comprehensive survey," Multimed. Tools Appl., vol. 80, no. 20, pp. 30367–30398, 2021, doi: 10.1007/s11042-020-09606-x.