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

Novel Fault Prediction Model in Component based Software System for KC1 Dataset

Anjali Banga[1], Pradeep Kumar Bhatia[2]

Submitted:10/03/2024 **Revised**: 25/04/2024 **Accepted**: 02/05/2024

Abstract: Present research work is focusing on fault prediction in component-based software system. In order to achieve this objective dataset of KC1 has been considered. This research work outlines a method for establishing reusable software component evaluation criteria. Optimization mechanism named hybrid PSO-MVO mechanism has been applied on component dataset that impact selection. The major objective is to provide smart and optimized solution for component selection. Dataset that is filtered after detecting of optimized value is trained using deep learning model. The accuracy parameters such as recall value, precision, F1 -score would be considered to evaluate the accuracy of optimized component selection model. Such research is supposed to play significant role in area of CBSE by providing high performance and accurate solution. Optimized value has been calculated for Line count of code, Cyclomatic Complexity, Design Complexity, Estimate Time, Difficulty, Intelligence, Efforts. Considering optimized value, the dataset has been filtered to build LSTM based model to detect the faults. Selection of significant attributes and elimination of non-optimized dataset has increased reliability of model.

Keywords: *PSO-MVO, Deep Learning, LSTM, Fault Prediction, Feature Selection*.

1. Introduction

Software engineering may be considered as a method to produce numerous software projects with many categories. For this reason, applications of computer engineering are applied, although it is becoming a tough job to anticipate the software system dependability. Component-based software engineering is method that is competent to solve the problem of dependability. Component-based software engineering is a comprehensive technique that facilitates the construction of numerous components based recent software investigations. It is not simple for a newbie to produce new software but Component-Based Software Engineering (CBSE) helps him to reduce his efforts in development of new software. In CBSE, numerous elements like reusability, reliance of component, and interaction between components are significant. These characteristics aid in designing new software and reducing the system complexity. There is several soft computing methods that has been applied for forecasting dependability of system. Selection of Component in Software is substantial stage in component dependent software engineering patterns that is essential to retrieve the components for adaption and to construct them. Previous investigations on component based selection conducted little discussion about qualities of a component. Revealing procedure of influencing elements that are impacting industry practitioners at the time of component selection is continuing. Optimized component dependent development of software is designed to make selection

of component easier in reusable software. Fault is the real error in the code that causes the program not to operate as it is intended. A defect stays undiscovered throughout testing and debugging procedure and is found after installation at customer's operating system. Faults leads to software breakdowns. Hence affects the software quality. Software failures lead to substantial expenses and sometimes in safety essential systems it costs even the human life. After all software's were built by people who makes blunders by accident or due to demands like dead line. Predicting software defects improves the quality of the product. Using software failure prediction information helps one to concentrate on the bad modules in subsequent versions and in future projects of the same sort. Software For complex software, archiving reliability is a difficult task. Fault proneness, dependability, and reusability are all aspects of software quality assurance that must be taken into consideration. The accuracy and complexity of a fault prediction model are the two most important factors to take into account. Building a defect prediction model with better accuracy and less complexity may be the goal. An important part of feature selection is the elimination of unnecessary, duplicated, missing or incorrect data. As a result of the complexity of modern software, the selection of features is important to their ability to accurately anticipate software failure. Features may be selected using filters and wrappers. We use statistical and machine-learning methodologies like parametric models and hybrid

[1] *Research scholar, Computer Science Eng. Dept., Guru Jambheshwar University of Science and Technology, Hisar-125001, Haryana, India*
ORCID ID: 0009-0005-7703-9143
[2] *Professor, Computer Science Eng, Dept, Guru Jambheshwar University of Science and Technology, Hisar-125001, Haryana, India*
** Corresponding Author Email: banga.anjali88@gmail.com*

algorithms in order to predict the errors. It is possible to educate a computer to better foresee the future by using a supervised machine learning technique that takes input and output into consideration. We used the promise repository's KC1 metrics dataset to run the tests. It is essential to know your accuracy, MAE, and RMSE in order to compare various feature selection methods.

1.1. Component-based software systems

In the process of creating huge, intricate systems, CBS systems emerge as the final result. Conventional software development methodologies are no longer applicable in favour of CBSSs. Such approaches are accused of low productivity, large development costs, unpredictability in software quality, and the high risk of migrating to new machines, among other problems. In order to keep development costs and time to market to minimum, component-based software systems rely on high-quality components. Maintenance, reliability, and overall quality may all be improved using these components. Even if composition techniques are used in a given system, the CBSS class is exceedingly difficult to regulate. It's not easy to maintain a system that changes its components and structure on a regular basis. According to current research, component-based software engineering may be explained by two evolutionary processes (CBSE). The creation of reusable components is the initial step in the development process. At this phase, Component-based software systems are developed and reused (CBSSs). All of the optional extras that may be purchased individually are included into these systems as well.

1.2. Convolutional Network Layer

To categories the data after a convolutional operation, a non-linear activation function is applied to the outputs, and then a full connection layer is used. The kernel function, which is another name for filter, is at the core of all convolutional procedures. To complete feature extraction, the original matrix moves from top to bottom along with left to right. In natural language processing, a kernel function as large as the original matrix is used to maintain the integrity of a word at the finest granularity possible. Depending on whether the original matrix is emptied out, you may utilize either zero padding or valid padding when sliding a kernel function. In this scenario, we use proper padding. A separate layer, the LSTM, analyses the data that comes from our convolutional layer. Because LSTM needs the input of a sequential relationship, the pooling process breaks this link, thus we remove it. Instead of applying an activation function on the muddled results like in a traditional CNN, we skip that step entirely in our implementation.

1.3. LSTM

At the heart of LSTM is a cell state mechanism that may selectively allow information to flow through the door mechanism and add or remove information from cells. Forget, input, and output gates are all part of LSTM. Prior to updating the cell state, input gate determines what information should be removed from it through forget gate along with what information should be inserted by input gate. The cell's state may be modified when these two points have been established. The ultimate output of the network is determined by the output gate. A RNN with capacity to develop long-term reliance is known as a LSTM. Figure 1 depicts its internal workings.

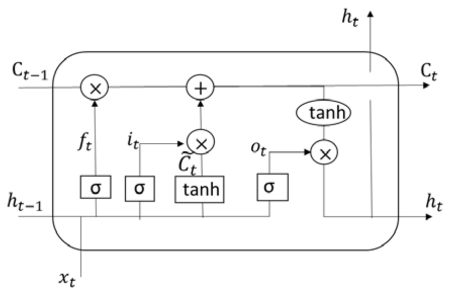

Fig.1. LSTM: standard

Nodes' states are calculated using Equation (1)-. (6).

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big) \quad (1)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (3)$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad (4)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t \cdot \tanh(C_t) \quad (6)$$

h_{t-1} denotes the previous layer's hidden state, x_t is current input, W along with b is weights & biases, σ is sigmoid function, f_t is forget gate's output, iit is input gate's output, $\tilde{C}_t$ is intermediate temporary state, C_{t-1} is previous layer's cell state, C_t is next layer's cell state, and o_t is output gate's output along with h_t is hidden state of next layer. LSTM & one of its versions, named COIF-LSTM, are compared in this paper's first section. Figure 2 depicts its internal workings.

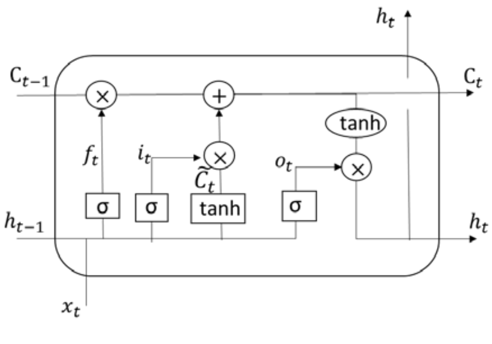

Fig. 2. LSTM: coupled of input and forget gates

Sole variation between them is that input gates are calculated differently. To represent the updating manner of cell state in the following layer in Equation, output of input gates is determined by 1-fft, rather than output of forget gate along with the input gate being calculated independently.

1.4. Deep Learning

Deep learning is an AI approach that replicates the way people learn about certain types of information. Deep learning is a key component of data science, which includes statistical analysis and predictive modelling. A subset of deep learning, artificial neural networks are algorithms inspired by structure along with function of the brain. If you are in this circumstance, machine learning (ML) provides a solution that offers the benefits of quick prediction and simpler parameter structures that match early design stages. In a DSE process, this enables designers and engineers to swiftly alter designs and evaluate the repercussions for performance. To be useful in new contexts, present ML techniques must be revised since they were built expressly for design situations.

1.5. Fault Prediction

Predictive maintenance relies on failure prediction to keep downtime and repair costs to a minimum. Mathematical and statistical models are now the most often utilized methods for predicting failure. System failures may be avoided or mitigated by using methods such as predictive failure analysis (PFA), which is used to describe a group of techniques aimed at predicting when a system or component (whether software or hardware) would fail and, in some cases, providing ways to do so. Because software dependability is becoming more and more essential in the software business, failure prediction is a tough, intriguing, and significant Endeavour. It is necessary to use a variety of methods to find errors in the software's development. To assess a program's reliability, failures must be accounted for, and this is difficult to do until the product has been fully created. Among all quality criteria, software dependability is the most researched [1]. An important prerequisite for understanding the software's global behavior and ensuring its appropriate functioning in the future is the prediction of defects for the estimation of its dependability.

1.6. Software Fault Prediction

The average execution time or the numbers of defects that occur in an execution time period are important determinants of software dependability. Faults may emerge at any point in the software design process, from the requirements all the way through to the final product. As the size and complexity of the program rises, so does the amount of errors. The rate of software failure grows as several defects in the program increases, & software's dependability falls. In addition to design flaws, implementation flaws are also regarded as design flaws. It is unusual, if ever, that complicated software can be developed without errors, and even when it does, it is almost never possible to ensure that program is error-free. Correctness of software may be proven using formal techniques, which implies that it satisfies a specification document. In contrast, today's formal verification methodologies are not meant for huge software systems, such as operating systems or word processors, to be tested with. As a result, accuracy alone does not guarantee dependability since the specification document itself may be flawed. The reliability of software must be analysed in order to meet high dependability criteria since it is impossible to design sophisticated software systems devoid of flaws. If the identical settings are used to run two copies of the same (normal) programme, they will both fail. This disproves the independence hypothesis. Furthermore, the failure rates of software copies are entirely interdependent. This means that many hardware fault tolerance ideas are worthless for software. Software dependability may be increased by leveraging design diversity rather than redundant copies. The so-called N-version programming, developed by Chen and Avizienis, is a typical method to this problem. N-modular redundancy is more successful in hardware reliability engineering than design diversity, according to Knight and Leveson's study. The majority of failures in complex systems are caused by software errors, according to certain research. In reliable systems, software flaws behave like transitory hardware flaws despite the fact that they are design flaws. This is because their activation conditions are stochastic.

1.7. Feature Selection

Reliability of software is often determined by the failure rate in mean execution time or the number of errors that occur in execution time gap in program. Most errors arise in the course of creating software, from gathering requirements through putting that program into production form. As the size and complexity of the program rises, so does the frequency of errors. When there are a greater number of software defects, the likelihood of software failure rises and the program's dependability drops. Faults created during implementation are also regarded as design flaws by the engineering community. Complex fault-free software is seldom possible to design, and even when it can, it is rarely guaranteed to be fault-free. The correctness of software may be shown using formal techniques, which indicates that it satisfies the requirements of a document. However, today's formal verification methodologies are not meant for huge software systems like operating systems or word processors. Because the specification document might be incorrect, accuracy does not guarantee dependability. The reliability of software must be analyzed in order to meet high dependability criteria since it is not possible to design sophisticated software systems devoid of defects. All copies of typical software will fail if they are performed with the same settings. This disproves the notion of independence. To be more specific, the failure probabilities of software copies are fully reliant. In software, this means that many hardware fault-tolerance techniques are useless. Software dependability may be increased by leveraging design diversity rather than duplicate copies of the same program. The so-called N-version programming, developed by Chen and Avizienis, is a popular strategy for accomplishing this objective. As Knight and Leveson's research has shown, N-modular redundancy in hardware reliability engineering is likely to be more successful than design variety for software. Studies have indicated that the majority of failures in complex systems are caused by software errors. In reliable systems, software flaws behave like transitory hardware flaws, despite the fact that they are design errors. Because their activation circumstances are stochastic, this happens.

1.8. PSO-MVO Optimization

PSO is transformed into an evaluation tool. The shape it takes is a procedure that is both easy to apply and put into practice often. It was previously determined that this kind of evaluation efficiently finds the optimal answer. One way to describe this approach in the context of IT is as a strategy for optimizing any given issue. It has been noted that in a PSO-based model, each candidate solution's performance is improved one by one. It addresses any population-related problem with potential answers. The particles that have been labeled move about in search-space. The method relies on the mathematical rule that is independent of the particle's location and velocity. A major factor influencing its movement is its domestically famous position. There have been improvements to this place in the shape of higher positions. Other particles will have little trouble locating these spots. It is anticipated that this will guide the swarm to the most optimal options. PSO is considered a good heuristic since it optimizes the issue without making many assumptions. Nevertheless, there is no assurance that an ideal solution will ever be discovered when using Meta Heuristics like PSO. Given its track record of effectiveness in solving a wide range of optimization problems, it has emerged as one of the most essential and practical met heuristics in use today. This model can arrange itself. It revealed how dynamic these intricate systems are. In a cooperative and intelligent framework, it employs a very simplified model of social behavior to handle optimization challenges.

MVO represents a novel innovation. It is an environmentally friendly approach to maximize that works. "This" was created by Mirjalili et al. When putting this into action, they considered two specific considerations. Three cosmological ideologies were used to develop this approach. Not only does it gain notoriety in this manner, but it also does so in a novel meta-heuristic optimization technique. When it comes to issues connected to OPF, it finds them quickly and easily. From a biological and societal scientific perspective, it is a methodology that receives ongoing inspiration. This approach incorporates several cosmological ideologies into its operation. This approach makes use of the wormhole notion in addition to the white hole and black hole ideas. The method's ability to quickly determine the intersection rate is one of its most significant strengths. It does this by selecting a roulette wheel. Furthermore, this technique can handle both discrete and regular optimization problems.

Deriving hybrid PSO-MVO equation

The Hybrid PSO-MVO set combines PSO with MVO. By combining the strengths of PSO and MVO, a hybrid system may achieve optimal performance in a specific application. The MVO Universe value is taking the place of the PSO Pbest value.

$$v_{ij}^{t+1} = wv_{ij}^{t} + C_1 R_1 (\text{Universes}^{t} - X^{t}) + C_2 R_2 (\text{Gbest}^{t} - X^{t})$$

2. Literature Review

LSTM recurrent neural networks were developed by Liyuan Liu, et al. for influenza trend prediction. Acute respiratory infection known as ILI has a significant death and morbidity rate. Predicting influenza patterns and responding quickly to a health crisis is essential to reducing the number of deaths. LSTM RNNs are used in this research to predict influenza trends. The only one to integrate virologic surveillance, influenza geographic dispersion, Google trends, climate along with air pollution, and other unique data sources to predict influenza trends. Several environmental and climatic factors were also discovered to have a strong correlation with the incidence of ILI [1].

Different security vulnerabilities were plaguing the Internet and computer networks at the time of Mingyi Zhu,et al. Deep learning solution based on AMF-LSTM for net-work anomaly identification. AMF-LSTM, this model for anomalous traffic detection, is presented in this research. An attention mechanism is added to the original LSTM in order to assist model understand which traffic flow has a greater impact on its final output by using statistical features of multi-flows rather than a single-flow or features retrieved from log data. AMF-LSTM approach provides a high level of accuracy and recall when it comes to identifying anomaly types, according to experiments. [2]

For sentiment analysis in Arabic, Abdulaziz M. Alayba,et al. coupled the CNN and LSTM models. The ability of deep neural networks to represent complex and massive datasets from a wide range of applications has been demonstrated. CNNs have bene-fits in picking appropriate features, but LSTM networks have proved to be excellent at learning sequential data. Image processing, speech recognition, language transla-tion, and other NLP applications benefit from both techniques, according to research. For emotion categorization of brief text messages from Twitter, the complexity in-creases since Arabic is a rich language in morphology. The lack of proper pre-processing technologies for Arabic and study into the topic are both limiting factors at the moment. For Arabic sentiment analysis, we found that merging CNNs with LSTMs resulted in a significant increase in accuracy. Additionally, we'd want to take into account the particu's morphological variety [3].

Sentic LSTM using a hybrid network was studied by Yukun Ma and colleagues for aspect-based sentiment analysis. In recent years, sentiment analysis has become one of the most common NLP jobs. Emotional classification of inputs is a common task in the workplace. Sentence form presumes harmony along with coherence of the emotion it expresses, however this isn't always true. As a result of its more realistic assumption that sentiment is dependent on particular traits and entities, sentiment analysis has gotten a lot of attention from the community. The use of deep neural networks for sentiment analysis has made significant progress in the last several years. To gain implicit information from data, (LSTMs), a functional reproduction of human brain activity, are one of the most successful deep neural models for sequential data. Even while the training data may teach LSTM certain implicit information, such as common sense facts, it is difficult for them to gain explicit knowledge. Consequently, new knowledge sources have evolved, and it has been agreed that providing background information is an essential addition to many NLP positions. Information-rich sentiment analysis is suggested in this work with a specific emphasis on leveraging commonsense information in the deep neural sequence model for sentiment analysis, as shown in this paper Stacked attention models for the target and phrase levels are used to formally depict the inference of the dependent sentiment in LSTM. As an extension of long-term memory, Sentic LSTM directly incorporates explicit information into implicit knowledge. Token-level memory and concept-level inputs are interpolated by a separate output gate in an enlarged LSTM cell an extension of Sentic LSTM, which replicates sentic patterns, is an LSTM-recurrent additive hybrid network. In particular, we're looking for a task that combines the detection of target-dependent features with the categorization of target-dependent aspects. For this combined task, two benchmark datasets are employed to measure performance. [4].

The attention-based two-phase lstm model developed by Kuntal Dey and colleagues was used to identify topical stances on Twitter. It is possible to determine the con-tent's general attitude (positive, negative, or neutral) toward a certain issue by analys-ing its topical position. The topical stance detection issue aims to address this issue. Based on the principle of attention, we came up with a two-phase strategy. In first phase, we determine if a tweet is neutral or subjective with respect to a certain issue. Secondly, they classify tweets depending on whether they are in favour or against the topic (excluding neutral tweets) in the second phase. An attention-based neural net-work (LSTM) is proposed for each stage of the process, and attention is included in each phase. On SemEval 2016 stance detection Twitter task dataset [7], we outper-form DL-based methods in terms of best case macro F-score (68.84 percent) and best case accuracy (60.2%). A new topical stance detection system called T-PAN [5] uses DL in a two-phase approach to improve accuracy.

Tae-Young Kim,et al. was an essential aspect of the energy supply company's opera-tions and planning. To ensure a steady supply of power, it is necessary to have a back-up plan in place. However, because power is difficult to store, it is vital to forecast demand. A CNN-LSTM hybrid network is proposed in this research to extract spatio-temporal information to successfully estimate consumption of household electricity. Using experiments, it has been demonstrated CNN-LSTM hybrid networks, which linearly mix CNN, LSTM, and DNN, can detect irregularities in electric power consumption patterns. Time series may be forecasted using DNNs and LSTMs in conjunction with the CNNs and LSTMs in combination. The CNN-LSTM hybrid technique is nearly perfect in predicting power use. For the UCI repository's individual home power consumption data sets, CNN-LSTM hybrid technique yields a greater RMSE than standard prediction algorithms. [6].

Jeff Heaton,et al. was considered Deep learning. Deep Learning presents an in-depth look at the current status of deep learning research, as well as a glimpse into the future of the field. With the help of his doctoral adviser Yoshua Bengio, Ian Goodfellow authored the paper. Aaron Courville also contributed. All three are well-known artificial intelligence experts (AI). Individual chapter PDFs of the book are also accessible for free online.1 The book, which is available in hardcover and Kindle versions, is geared for academic researchers who already have a working knowledge of calculus, linear algebra, probability, along with basic programming skills. [7].

Micah J. Sheller, et al. have shown that deep learning models require a lot of data to correctly segment images meaningfully. In the medical imaging field, obtaining enough data is a major difficulty. Expertise and training are required for medical imaging data labelling. Data sharing in the medical field raises several challenges for international organisations dealing with privacy, technology and data ownership. This is especially true in the United States. As a result of our work, we are the first to use federated learning in multi-institutional cooperation to allow for deep learning model-ling without disclosing patient data. Quantitatively, we found federated semantic segmentation models perform on par with models trained by sharing data on multi-modal brain scans. According to our study, there are two other collaborative methods that don't perform as well as federated learning. [8].

Interpretable/disentangled middle-layer representations were the subject of this study by Quan-shi ZHANG and colleagues. The interpretability of deep neural networks is usually their Achilles' heel, despite the fact that they are better in many tasks. Al-though deep neural networks can currently make excellent distinctions, their black-box representations are tough to comprehend. Learning from a few annotations, human–computer cooperation at the semantic level, and semantically debugging network models are all examples of deep learning constraints that can be solved with high model interpretability. There are many ways for diagnosing and detangling pre-trained CNN representations, as well as methods for executing middle-to-end learning on the basis of model interpretability and training CNNs with disentangled representa-tion. Last but not least, let's take a look at what the future of AI looks like from a layperson's perspective. [9].

Deepak Sharma,et al presented software quality, maintainability, and reusability, machine-learning techniques are employed to discover defects, faults, ambiguities, and foul smells. To forecast software defects, statistical approaches are employed to analyse data. Machine-learning approaches, on the other hand, are useful for spotting software errors. Using ML approaches, the authors of this study provide an overview of software failure prediction. Conventional methods are also covered in this study. Fault-proneness is described in detail here. [10].

Anum Kalsoom,et al. expressed the quality of a company's software is critical to its success. It is important to note that traditional software quality assurance methods have significant time and money constraints. For this reason, the use of ML to fore-cast software errors is increasing. In early phases of program life cycle, software fault prediction can assist developers in discovering software issues. Identifying discrimin-ative software metrics and generalizing these approaches to other program sizes, as well as the problem of class imbalance, are the most difficult difficulties to overcome. For the first time in this research, performance of nine commonly used ML classifiers — Bayes Net, NB, ANN, SVM, K closest neighbours, AdaBoost, Bagging, Zero R, and Random Forest in case of software defect prediction — has been evaluated in this study. The problem of class imbalance is addressed using SMOTE and Resample with replacement, two basic sampling approaches. In order to pick the most discriminative metrics, we employed a mix of the FLDA-based feature selection technique, SMOTE, and Resample. This research makes use of 15 openly accessible datasets from PROMISE repository. In terms of accuracy, recall, f-measure, along with area under curve, suggested Resample-FLDA technique outperforms previous methods. [11].

Chang Mook Kang,et al. considered kinematics-based fault-tolerant techniques. As a workaround to the camera vision sensor failing, they suggest using fault-tolerant ap-proaches for a LKS. Prior to the driver assuming control, lateral control system must maintain its stability without vision sensor's output owing to failure or environmental circumstances. Lateral kinematic vehicle motion model is used to offer a fault-tolerant control approach. Kinematic motion model-based lane estimate technique addresses the possibility of camera vision sensor failure in face of complicated shadowing, missing lane markers, and illumination variations. The suggested lane estimating approach allows the LKS to work even when sensors fail. Computational simulation data from CarSim and MATLAB/Simulink were used to verify the new approach. AutoBox from dSPACE was also tested on a test car for comparison [12].

3. Problem Statement

Research paper explains various Soft computing mechanisms that have been utilized for Software Component Selection. It has been observed that existing Software Selection Modules have their own limitations. ACO based Software Component Selection modules do not support to Application Complexity whereas Neuro Fuzzy based module is enable to provide reliability in Software Component Selection. ACO based module can be used only to compute the length of shortest path. But PSO with MVO provides best solution in a small time. So, it has become essential to propose an optimized software selection module which can resolve the existing issues in this field. Moreover it has been observed that MVO based PSO is performing fast as compare to other optimization techniques. Time taken and accuracy of solution, need to be considered in order to perform comparative analysis of these optimization techniques.

4. Proposed Work

Research is considering KC1 dataset that have 22 attribute to build reliable LSTM based network model. Research has used PSO-MVO mechanism to get the optimized solution for significant fields. For Line count of code, Cyclomatic Complexity, De-sign Complexity, Estimated Time, Difficulty, Intelligence, Efforts, the optimal value has been estimated. The dataset has been filtered to develop an LSTM-based model to identify errors, taking into account the optimal value. The model's dependability has been improved by the selection of important features and the deletion of non-optimized datasets.

4.1. Process Flow Proposed Model

Researchers use a training dataset for 70% of their work and a testing dataset for 30%. The training step would generate two LSTM models, one using the filtered dataset and the other using the unfiltered dataset. A filtered model is referred to as a suggested model, while an unfiltered model is called a conventional model. In order to cut down on time spent training and testing, the suggested work takes into account an optimization strategy based on PSO to remove the KC1 dataset. He planned to train the LSTM model using the dataset that had been filtered. To simulate a more precise answer, an LSTM with hidden layers would be ideal. Hidden layers, batch size, and epoch size are among of the accuracy-influencing elements that are now under investigation. The quantity of the training and testing datasets also affects the accuracy and time consumption. Simply put, the research uses a hybrid approach that integrates an optimization mechanism with LSTM to train a sentiment analysis model quickly, taking the KC1 dataset into account. Our study's overarching goal is to take into account previous work in the areas of optimization mechanisms, machine learning, and sentiment analysis.

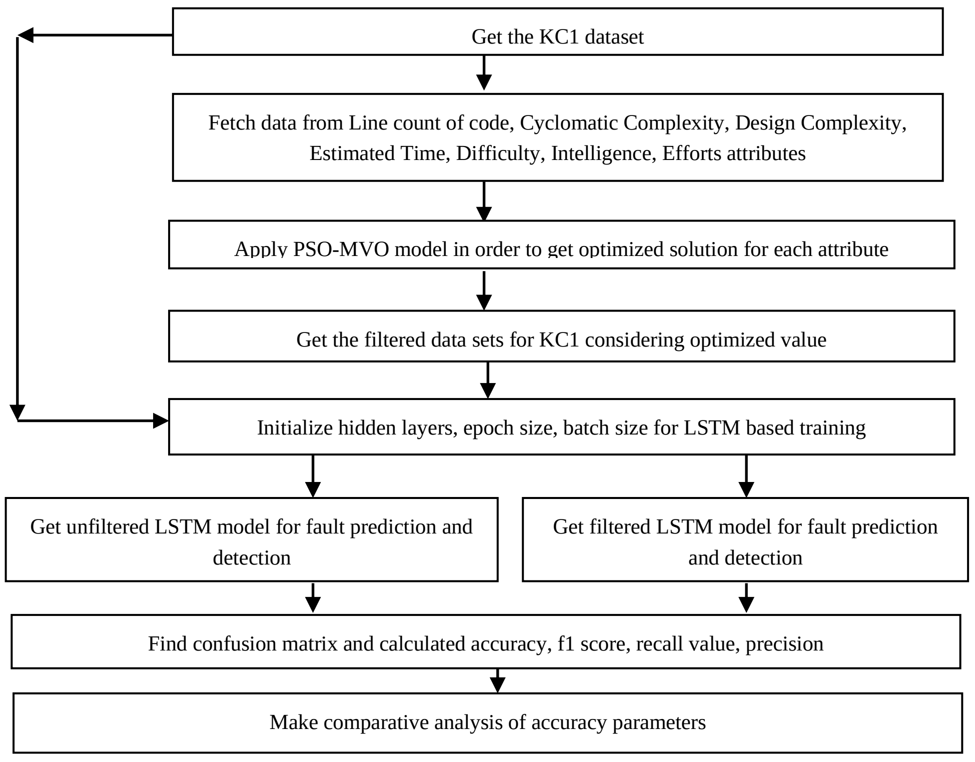
Get the KC1 dataset
Fetch data from Line count of code, Cyclomatic Complexity, Design Complexity, Estimated Time, Difficulty, Intelligence, Efforts attributes
Apply PSO-MVO model in order to get optimized solution for each attribute
Get the filtered data sets for KC1 considering optimized value
Initialize hidden layers, epoch size, batch size for LSTM based training
Get unfiltered LSTM model for fault prediction and detection
Get filtered LSTM model for fault prediction and detection
Find confusion matrix and calculated accuracy, f1 score, recall value, precision
Make comparative analysis of accuracy parameters

Fig. 3. Proposed work

Issues and limitations with prior research approaches, such as inaccuracy and time consumption, are being investigated in the present study. It is intended to construct a hybrid model that combines optimization mechanisms with learning approaches to provide a system that is both accurate and highly performing. Last but not least, the suggested model compares and contrasts conventional fault detection in software systems that rely on components with the proposed method. In order to improve dependability, optimization approaches based on PSO-MVO were used to filter out the KC1 dataset before training and testing. He planned to train the LSTM model using the dataset that had been filtered. To simulate a more precise answer, an LSTM with hidden layers would be ideal. Hidden layers, batch size, and epoch size are among of the accuracy-influencing elements that are now under investigation. The quantity of the training and testing datasets also affects the accuracy and time consumption.

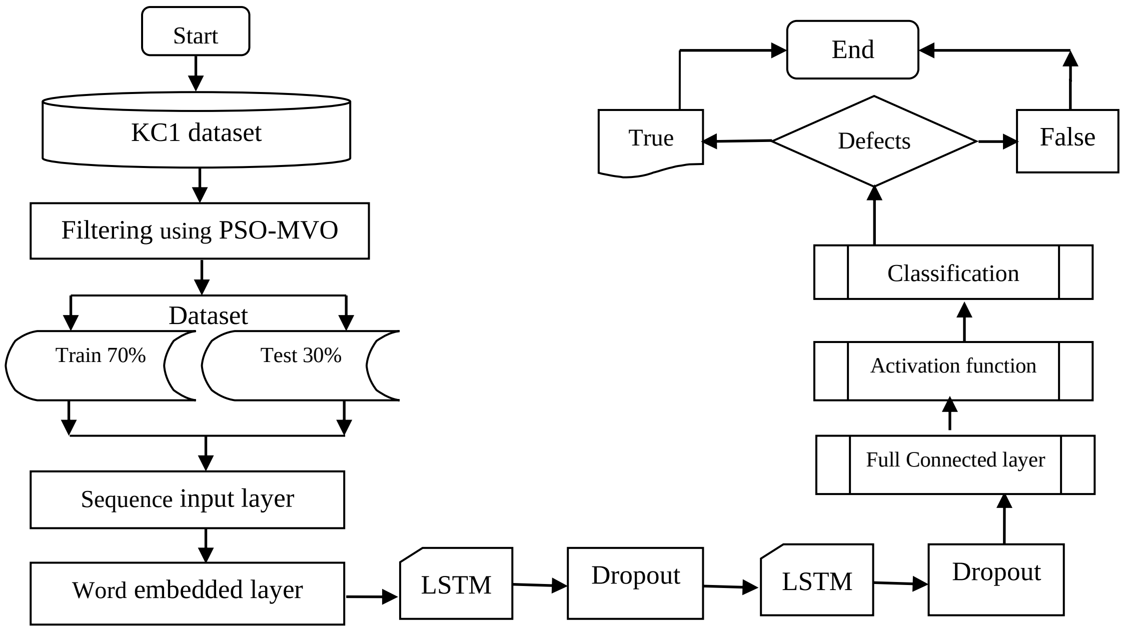

Fig. 4. Working of LSTM model in proposed work

4.2. Dataset

In KC1 Dataset there are 22 attributes and 2109 records where 326 are true for defect attribute and 1783 . These attributes are loc, v(g), ev(g), iv(g), n, v, L, T, d, i, e, B, lOCode, lOComment, lOBlank, locCodeAndComment, uniq_Op, uniq_Opnd, total_Op, total_Opnd, branchCount, defects. A module's loc specifies how many lines it contains in total. This is total number of lines of code, including executables, com-ments, & blanks. It is counting from the first bracket to the last, regardless of the number of characters in each line. Measures of "linearly independent pathways" (v(g)) The term "linearly independent" refers to the fact that in a program's "flow graph," no route is linear combination of any other path in set. In flowchart, each node represents program statement, & each arc depicts direction in which control moves between those statements. v(g) is computed using

$$v(g)-e-n+2 \qquad (7)$$

g is flow graph, "e" is number of arcs in flow graph, & "n" is number of nodes in flow graph. McCabes rules ("v(g)">10) are often used to pinpoint a module's problem po-tential. It is possible to decrease the essential complexity of a flow graph by decom-posing all of 'g' into "D-structured primes" (ev(g)). These "D-structured primes" are referred to proper one-entry one-exit sub flow graphs [1].

$$ev(G)-v(G)-m \qquad (8)$$

Where "m" is number of D-structured primes in "g" flow graph. It is cyclomatic com-plexity of reduced flow graph of a module that is measured by Design Complexity (iv(g)). A module's flow graph, "g," is simplified to remove any unnecessary com-plexity that has no bearing on how different design modules interact with one another. A module's call patterns to its immediate subordinate modules indicate its complexity, according to McCabe. N is total operands and operators for Halstead. V To code a program, a minimum amount of bits (V) must be included in a module to achieve this measure. A module's level (L) statistic describes the understanding level (L) of the program. A module's programming time is denoted by the letter T. A time estimate for implementing the method is given here. The greater distinct operators in program, more complicated or error-prone it is. Intelligence (i) is a metric that measures algorithm complexity irrespective of the language in which it is expressed. When it comes to a television show, the intelligence Content controls how much is spoken. Effort (e) is defined as total time taken by each module. Implementing the program requires mental discriminations, as does understanding and learning the program. The error estimate measure for a module is given by the letter B. It's a rough estimate of how many mistakes were made throughout the implementation process. All code that isn't properly commented is included in IOCode. In modules, the amount of lines devoted to comments is known as the IOComment property. There is a count of blank lines in IQBlank. The geographic location of a module The number of lines that include both code and comment is specified by the Code And Comment property. The term "uniq Op" refers to the number of distinct operators in a module that are either unique to that module or that are not. A module's number of unique operands is referred to as its uniq Opnd. The account module is what you're looking for. Total number of variables and constants the term "op" stands for "all operators." (Total of twenty)

Opnd is a measure of how many operands have been used. A module's branch Count tells you how many decision points it has. For example, flaws describe whether or not a given module is flawed, and this affects decisions. Predictions are made using this property.

Table 1. List of KC1 Data Set

1	loc	Line count of code
2	v(g)	Complexity of Cyclomatic
3	ev(g)	Complexity of Essential
4	iv(g)	Complexity of Design
5	n	Number of Operands
6	v	Presenting Volume
7	L	Length of Program
8	T	Estimate Time
9	d	Difficulty
10	i	Intelligence
11	e	Efforts
12	B	Estimate Error
13	lOCode	Count of Line
14	lOComment	Line Count of Comment
15	lOBlank	Count of Blank line
16	locCodeAndComment	Code and Comment Line
17	uniq_Op	Unique Operators
18	uniq_Opnd	Unique Operands
19	total_Op	Total number of Operators
20	total_Opnd	Total number of Operands
21	branchCount	Flow Graph
22	defects	{True, False}

5. Result and Discussion

Simulation work has considered conventional and proposed model for machine learning in case of KC1 data set training along with testing. Confusion matrix has been obtained in order to get accuracy parameters to evaluate reliability of proposed model with respect to conventional model. The study analyzed classification performance using confusion matrices for both conventional and proposed models. Accuracy, precision, recall, along with F1 score were calculated for each class. Per-class performance was evaluated using graphs and tables. Visual representation was used to compare metrics between the models. Accuracy, precision, recall, along with F1 score were calculated using the formula (TP+TN)/(TP+TN+FP+FN).

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{9}$$

$$\text{Precicion} = \frac{\text{TP}}{\text{FP}} \tag{10}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{11}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision+Recall}} \tag{12}$$

5.1. Simulation for Conventional research

Conventional research provided accuracy of 84.54% in case of reduced feature selection method:

Table 2. Confusion matrix in case of reduced feature selection method

	True	***False***
True	152	36
False	41	271

TP: 423 and Overall Accuracy: 84.54%

Table 3. Accuracy parameters in case of reduced feature selection method

Class	***n (truth)***	***n (classified)***	***Accuracy***	***Precision***	***Recall***	***F1 Score***
1	193	188	84.6%	0.81	0.79	0.80
2	307	312	84.6%	0.87	0.88	0.88

Conventional approach has provided 85.63% prediction value in case of Naive bayes model:

Table 4. Confusion matrix in case of Naive bayes model

	True	***False***
True	155	36
False	35	286

TP: 423 and Overall Accuracy: 85.63%

Table 5. Accuracy parameters in case of reduced feature selection method

Class	***n (truth)***	***n (classified)***	***Accuracy***	***Precision***	***Recall***	***F1 Score***
1	190	191	85.63 %	0.81	0.82	0.81
2	304	303	85.63 %	0.88	0.88	0.88

5.2. Simulation for proposed work

Simulation work considers filtered and unfiltered dataset for training and testing using LSTM based model. The confusion matrix that has been obtained by unfiltered data set of KC1 is as follow when 500 records have been passed for testing:

Table 6. Confusion matrix in case of unfiltered LSTM model

	True	***False***
True	152	36
False	41	271

TP: 423 and Overall Accuracy: 87.2%

Table 7. Accuracy parameters in case of unfiltered LSTM model

Class	***n (truth)***	***n (classified)***	***Accuracy***	***Precision***	***Recall***	***F1 Score***
1	196	178	87.2%	0.87	0.79	0.83
2	304	322	87.2%	0.87	0.92	0.90

The confusion matrix that has been obtained by filtered data set of KC1 is as follow when 500 records have been passed for testing:

Table 8. Confusion matrix in case of filtered LSTM model

	True	***False***
True	160	15
False	38	287

TP: 447 and Overall Accuracy: 89.4%

Table 9. Accuracy parameters in case of filtered LSTM model

Class	***n (truth)***	***n (classified)***	***Accuracy***	***Precision***	***Recall***	***F1 Score***
1	198	175	89.4%	0.91	0.81	0.86
2	302	325	89.4%	0.88	0.95	0.92

5.3. Bert based Implementation

Simulation work considers filtered and unfiltered dataset for training and testing using Bert based model. The confusion matrix that has been obtained by filtered data set of KC1 using Bert is as follow when 500 records have been passed for testing:

Table 10. Confusion matrix in case of Bert based Implementation

	True	***False***
True	176	11
False	22	291

TP: 467 and Overall Accuracy: 93.40%

Table 11. Accuracy parameters in case of Bert based Implementation

Class	***n (truth)***	***n (classified)***	***Accuracy***	***Precision***	***Recall***	***F1 Score***
1	198	187	94.12 %	0.94	0.88	0.91
2	302	313	92.97 %	0.92	0.96	0.94

5.4. RoBert based Implementation

Simulation work considers filtered and unfiltered dataset for training and testing using Roberta based model. The confusion matrix that has been obtained by filtered data set of KC1 using Roberta is as follow when 500 records have been passed for testing:

Table 12. Confusion matrix in case of RoBert based Implementation

	True	***False***
True	187	8
False	11	294

TP: 447 and Overall Accuracy: 89.4%

Table 13. Accuracy parameters in case of RoBert based Implementation

Class	***n (truth)***	***n (classified)***	***Accuracy***	***Precision***	***Recall***	***F1 Score***
1	198	195	95.9%	0.95	0.94	0.95
2	302	294	96.39 %	0.96	0.97	0.96

5.5. Comparative analysis

Research has made comparison of accuracy, precision, f1-score and recall value in case of Conventional reduced feature selection, Conventional approach using Naïve Bayes, LSTM approach without optimization, LSTM approach after optimization.

Table 14. Comparative analysis of Accuracy

Clas s	*Conventional feature selection*	*Conventional using Naïve Bayes*	*LSTM without optimization*	*LSTM after optimization*	*Bert after optimization*	*RoBert after optimization*
1	84.60%	85.63%	87.20%	89.40%	94.12%	95.9%
2	84.60%	85.63%	87.20%	89.40%	92.97%	96.39%

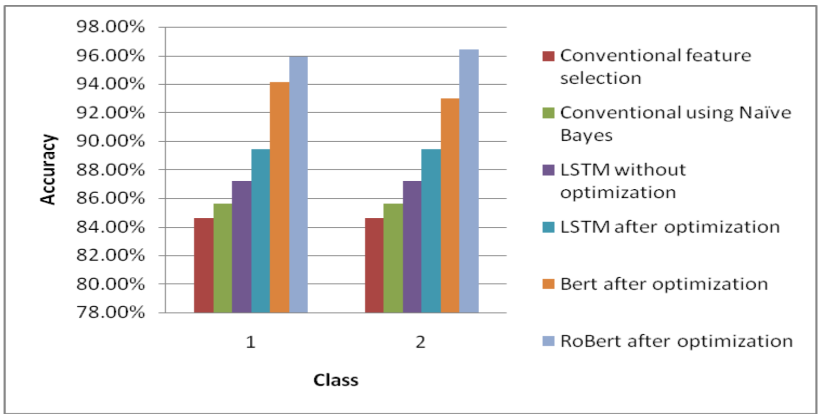

Fig 5. Comparative analysis of Accuracy

Table 15. Comparative analysis of Precision

Class	*Conventional feature selection*	*Conventional using Naïve Bayes*	*LSTM without optimization*	*LSTM after optimization*	*Bert after optimization*	*RoBert after optimization*
1	0.81	0.81	0.87	0.91	0.94	0.95
2	0.87	0.88	0.87	0.88	0.92	0.96

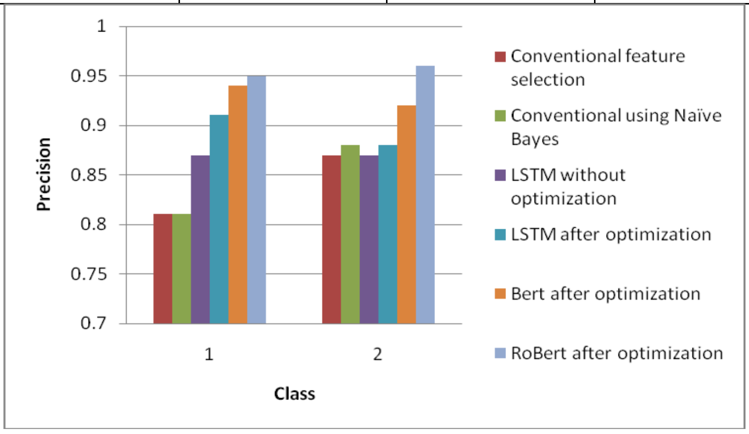

Fig 6. Comparative analysis of Precision

Table 16. Comparative analysis of Recall

Class	*Conventional feature selection*	*Conventional using Naïve Bayes*	*LSTM without optimization*	*LSTM after optimization*	*Bert after optimization*	*RoBert after optimization*
1	0.79	0.82	0.79	0.81	0.88	0.94
2	0.88	0.88	0.92	0.95	0.96	0.97

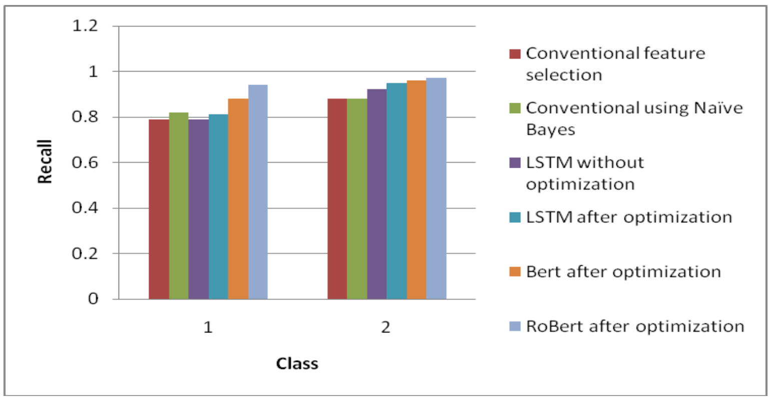

Fig 7. Comparative analysis of Recall

Table 17. Comparative analysis of F1-score

Class	***Conventional feature selection***	***Conventional using Naïve Bayes***	***LSTM without optimization***	***LSTM after optimization***	***Bert after optimization***	***RoBert after optimization***
1	0.80	0.81	0.83	0.86	0.91	0.95
2	0.88	0.88	0.90	0.92	0.94	0.96

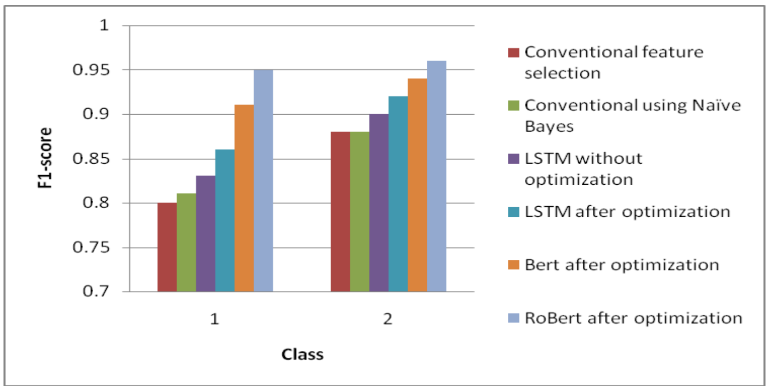

Fig 8. Comparative analysis of F1-score

6. Conclusion

This paper presents lot of researches on Software Component Selection in which different optimization mechanism in existing Software Selection Modules are suffering from many issues. Thus optimized Software Selection System is supposed to resolve the issue of the existing researches. Paper considered PSO-MVO as meta-heuristic optimization technique that is used to predict quality, reliability, applicability etc of software programs at high speed. This research work is beneficial to develop an efficient, fast and optimized Software Component Selection module. Conventional research provided accuracy of 84.54% in case of reduced feature selection method whereas it provided 85.63% prediction value in case of Naive bayes model. But proposed model has provided accuracy of 87.2 % if KC1 dataset is processed by LSTM model without attribute and record elimination and accuracy of 89.4% when dataset is filtered by PSO-MVO approach. The Bert-based and Roberta-based models were used for training along with testing KC1 on filtered and unfiltered datasets. Accuracy of 93.40% and 89.4%, respectively, for classification of KC1 based on the confusion matrix obtained from the filtered dataset. The Roberta-based model achieved an accuracy of 95.9% and a precision of 0.95%.

7. Future Scope

This research paper would be beneficial for coming researchers to know the present status of Software Selection in CBSE. It is a technical work in which PSO, MVO are integrated to propose an optimized Component Selection Module during LSTM based training. Therefore it would be easy for future researchers to implement an efficient and optimized Software Selection Module. This paper is supposed to guide future research work in field of CBSE. Moreover this work is supposed to play significant role in getting present status of Software Selection in CBSE.

References

[1] L. Liu, M. Han, Y. Zhou, and Y. Wang, “LSTM recurrent neural networks for influenza trends prediction,” Lect. Notes Comput. Sci., vol. 10847 LNBI, pp. 259–264, 2018, doi: 10.1007/978-3-319-94968-0_25.

[2] M. Zhu, K. Ye, Y. Wang, and C. Z. Xu, A deep learning approach for network anomaly detection based on AMF-LSTM, vol. 11276 LNCS. Springer International Publishing, 2018.

[3] N. Ahuja, P. k. Bhatia, and L. Rani., A review on nature inspired algorithm for test suite optimization, In AIP Conference Proceedings, vol. 2782, no. 1. AIP Publishing, 2023.

[4] A. M. Alayba, V. Palade, M. England, and R. Iqbal, A combined CNN and LSTM model for Arabic sentiment analysis, vol. 11015 LNCS. Springer International Publishing, 2018.

[5] Y. Ma, H. Peng, T. Khan, E. Cambria, and A. Hussain, “Sentic LSTM: a Hybrid Network for Targeted Aspect-Based Sentiment Analysis,” Cognit. Comput., vol. 10, no. 4, pp. 639–650, 2018, doi: 10.1007/s12559-018-9549-x.

[6] K. Dey, R. Shrivastava, and S. Kaushik, “Topical stance detection for twitter: A two-phase LSTM model using attention,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 10772 LNCS, pp. 529–536, 2018, doi: 10.1007/978-3-319-76941-7_40.

[7] T. Y. Kim and S. B. Cho, Predicting the Household Power Consumption Using CNN-LSTM Hybrid Networks, vol. 11314 LNCS. Springer International Publishing, 2018.

[8] J. Heaton, “Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning,” Genet. Program. Evolvable Mach., vol. 19, no. 1–2, pp. 305–307, 2018, doi: 10.1007/s10710-017-9314-z.

[9] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, and S. Bakas, Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation, vol. 11383 LNCS. Springer International Publishing, 2019.

[10] Q. shi Zhang and S. chun Zhu, “Visual interpretability for deep learning: a survey,” Front. Inf. Technol. Electron. Eng., vol. 19, no. 1, pp. 27–39, 2018, doi: 10.1631/FITEE.1700808.

[11] D. Sharma and P. Chandra, “Software fault prediction using machine-learning techniques,” Smart Innov. Syst. Technol., vol. 78, pp. 541–549, 2018, doi: 10.1007/978-981-10-5547-8_56.

[12] A. Kalsoom, M. Maqsood, M. A. Ghazanfar, F. Aadil, and S. Rho, A dimensionality reduction-based efficient software fault prediction using Fisher linear discriminant analysis (FLDA), vol. 74, no. 9. Springer US, 2018.

[13] C. M. Kang, S. H. Lee, S. C. Kee, and C. C. Chung, “Kinematics-based Fault-tolerant Techniques: Lane Prediction for an Autonomous Lane Keeping System,” Int. J. Control. Autom. Syst., vol. 16, no. 3, pp. 1293–1302, 2018, doi: 10.1007/s12555-017-0449-8.

[14] R. Kaur and S. Sharma, An ANN based approach for software fault prediction using object oriented metrics, vol. 955. Springer Singapore, 2019.

[15] S. S. Rathore and S. Kumar, “Software fault prediction based on the dynamic selection of learning technique: findings from the eclipse project study,” Appl. Intell., vol. 51, no. 12, pp. 8945–8960, 2021, doi: 10.1007/s10489-021-02346-x.

[16] P. S. Saini, A. Bhatnagar, and L. Rani, "Loan Approval Prediction using Machine Learning: A Comparative Analysis of Classification Algorithms." In 2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering, pp. 1821-1826. IEEE, 2023.

[17] W. Rhmann, B. Pandey, G. Ansari, and D. K. Pandey, “Software fault prediction based on change metrics using hybrid algorithms: An empirical study,” J. King Saud Univ. - Comput. Inf. Sci., vol. 32, no. 4, pp. 419–424, 2020, doi: 10.1016/j.jksuci.2019.03.006.

[18] P. Chaudhari and H. Agarwal, Improving feature selection using elite breeding QPSO on gene data set for cancer classification, vol. 695. Springer Singapore, 2018.

[19] A. Banga, P. K. Bhatia, “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, Lecture Notes in Electrical Engineering, vol 790. pp 455–479. Springer, Singapore, doi: 10.1007/978-981-16-1342-5_36

[20] G. I. Sayed, G. Khoriba, and M. H. Haggag, “A novel chaotic salp swarm algorithm for global optimization and feature selection,” Appl. Intell., vol. 48, no. 10, pp. 3462–3481, 2018, doi: 10.1007/s10489-018-1158-6.

[21] H. P. B, D. C. Priambodo, and I. W. Mangku, for Analyzing Seismic Activity, vol. 1, no. 474. Springer International Publishing, 2018.

[22] H. P. B, D. C. Priambodo, and I. W. Mangku, for Analyzing Seismic Activity, vol. 1, no. 474. Springer International Publishing, 2018.

[23] D. Aksu, S. Üstebay, M. A. Aydin, and T. Atmaca, “Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm,” Commun. Comput. Inf. Sci., vol. 935, pp. 141–149, 2018, doi: 10.1007/978-3-030-00840-6_16.

[24] Q. Al-Tashi, H. Rais, and S. Jadid, Feature selection method based on grey wolf optimization for coronary artery disease classification, vol. 843. Springer International Publishing, 2019.

[25] A. Banga, P. K. Bhatia, “Accuracy enhancement of Component based selection model using Hybrid Soft computing”, in Computational Intelligence and Communication Technologies (CCICT)”, IEEE, 2024, doi: 10.1109/CCICT62777.2024.00035

[26] M. Hussain, J. J. Bird, and D. R. Faria, “A study on CNN transfer learning for image classification,” Adv. Intell. Syst. Comput., vol. 840, pp. 191–202, 2019, doi: 10.1007/978-3-319-97982-3_16.

[27] M. Nguyen and N. A. V. B, SVMs with Deep Learning and Random, vol. 2. Springer International Publishing, 2019.

[28] T. Pham, C. Luong, and M. Visani, “Deep CNN and Data Augmentation for Skin Lesion Classification,” vol. 5, pp. 573–582, doi: 10.1007/978-3-319-75420-8.

[29] A. Banga, P. K. Bhatia, “Optimized Component based Selection using LSTM Model by Integrating Hybrid MVO-PSO Soft Computing Technique,” Adv. Sci. Technol. Eng. Syst. J., 6(4), 62-71 (2021) Vol. 6(4), pp. 62-71, 2021, doi: 10.25046/aj060408

[30] G. Singh, P. K. Sarangi, L. Rani, K. Sharma, S. Sinha, A. K. Sahoo, and B. P. Rath. "CNN-RNN based Hybrid Machine Learning Model to Predict the Currency Exchange Rate: USD to INR." In 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), pp. 1668-1672. IEEE, 2022.