AI-DRIVEN SOFTWARE DEFECT PREDICTION USING CLOUD-BASED MULTI-LAYER PERCEPTRON MODELS

[1]Basava Ramanjaneyulu Gudivaka
Genpact India Private Limited, Telangana, India
basava.gudivaka537@gmail.com

[2]R. Pushpakumar
Department of Information Technology,
Vel Tech Rangarajan Dr. Sagunthala R&D Institute of
Science and Technology, Tamil Nadu, Chennai, India.
pushpakumar@veltech.edu.in

ABSTRACT

Software defects bring a big problem for software development because they may cause functional failures and may degrade performance. Conventional prediction techniques are often not sufficient in handling the increasing complexity of modern software systems. This study, however, presents a cloud-based artificial intelligence-powered defect prediction approach, applying the Multi-Layer Perceptron models. The method harnesses the potential of modifying hyperparameter using Bayesian Optimization methodology for accuracy and subsequently cloud provisioning for scalability, large dataset storing, and as well as defect prediction models. The model is trained using Lasso regression and Z-score normalization for effective feature selection. The performance evaluation of the model shows remarkable improvement with an F1-score of 0.87, accuracy of 0.92, precision of 0.89, and recall of 0.85. Such results demonstrate how the cloud-based AI model can handle large datasets and predict defects in real-time. This method minimizes the time and resources for manual testing with the incorporation of AI with cloud technologies for fast, scalable, and effective fault prediction.

Keywords: Software Defect Prediction, Multi-Layer Perceptron (MLP), Cloud Computing, Artificial Intelligence, Z-score Normalization, Lasso Regression, Bayesian Optimization, Continuous Integration, AI-driven testing.

1. INTRODUCTION

In modern software engineering, defect detection and management have become cornerstone activities essential for ensuring software quality and reliability [1]. Software defects, commonly referred to as bugs or faults, can originate from various stages of development—ranging from requirements gathering to code implementation and testing [2]. When left unchecked, these defects can have a cascading impact on functionality, security, and user satisfaction [3]. Real-world incidents, such as system crashes or data breaches, frequently trace back to overlooked software defects [4]. Thus, addressing these issues at the earliest possible stage is paramount for minimizing operational risks and avoiding costly post-deployment fixes [5]. The complexity of contemporary software applications necessitates advanced methods for identifying and mitigating such defects systematically and proactively [6]. Historically, software testing has relied on manual methods, including code reviews, test case execution, and heuristic-based analysis [7]. While effective to some degree, these methods are increasingly proving to be insufficient in the face of growing codebase sizes and rapid deployment demands [8]. Manual testing is not only time-consuming but also subject to human error, making it difficult to ensure consistent defect detection across diverse environments [9]. Moreover, with the adoption of Agile and DevOps methodologies, where software updates are rolled out frequently, traditional approaches fail to scale effectively [10]. Rule-based static analysis tools, though automated, often produce false positives or miss context-sensitive bugs, further highlighting the need for more intelligent testing techniques.

The shift towards intelligent defect prediction is fueled by the desire to improve precision, efficiency, and adaptability in testing practices [11]. Defect prediction involves identifying parts of the software that are likely to contain faults, allowing developers to focus testing efforts where they are most needed [12]. This proactive approach aids in preempting issues before they manifest in production environments [13]. By utilizing historical

data and software metrics, machine learning models can uncover patterns that correlate with defect-prone code [14]. This method enables data-driven decision-making and can significantly enhance software quality assurance strategies [15]. The integration of AI into defect prediction represents a major leap forward in modernizing software testing workflows. Cloud computing plays a transformative role in supporting large-scale software testing initiatives [16]. It offers on-demand access to computational resources, enabling organizations to perform complex analyses without investing in extensive physical infrastructure [17]. Cloud platforms support scalability, elasticity, and high availability, which are crucial for handling diverse testing requirements [18]. These capabilities are particularly advantageous for defect prediction tasks that involve processing vast amounts of data and running compute-intensive models [19]. With cloud infrastructure, development teams can parallelize testing processes, optimize resource usage, and reduce the turnaround time for defect analysis [20].

The combination of cloud computing and artificial intelligence opens new possibilities in predictive defect management [21]. AI algorithms can be trained and deployed on cloud platforms, allowing for efficient model training, testing, and real-time inference at scale [22]. This synergy facilitates the continuous monitoring of codebases for anomalies, regression issues, and quality concerns [23]. The elasticity of the cloud ensures that resource allocation adjusts dynamically based on workload intensity [24]. Moreover, the cloud's support for distributed computing enhances the capability to manage multiple software projects and modules concurrently, thereby aligning with modern development pipelines that rely on continuous integration and delivery [25]. Machine learning (ML) has emerged as a powerful tool for automating various aspects of software quality assurance [26]. In defect prediction, ML models learn from historical software repositories, including source code, change logs, bug reports, and code metrics [27]. These models identify statistical relationships between software characteristics and the likelihood of defects [28]. Algorithms such as decision trees, support vector machines, and neural networks have been applied with varying degrees of success [29]. The choice of algorithm depends on the complexity of the dataset and the desired prediction accuracy. Among these, neural networks—particularly Multi-Layer Perceptrons—have shown remarkable ability in modeling non-linear relationships inherent in software defect data [30].

Multi-Layer Perceptrons (MLPs) are a class of feedforward artificial neural networks that consist of input, hidden, and output layers [31]. Each neuron in an MLP applies a weighted sum of its inputs followed by a non-linear activation function [32]. MLPs are capable of learning complex mappings from input features to target labels, making them suitable for defect prediction tasks [33]. When trained on labeled software datasets, MLPs can classify code segments as defective or clean with high accuracy [34]. The model's capacity to handle multidimensional input makes it ideal for processing various software metrics and indicators that contribute to defect occurrence [35]. The effectiveness of MLPs in defect prediction hinges on the selection and engineering of relevant input features [36]. Common features include code complexity metrics (e.g., cyclomatic complexity, lines of code), change metrics (e.g., number of revisions, developer count), and process metrics (e.g., commit frequency, test coverage) [37]. These features serve as proxies for software quality and development activity [38]. Proper normalization and feature selection techniques ensure that the MLP model generalizes well to unseen data [39]. Feature engineering also involves removing redundancy and noise to avoid overfitting during training, thereby improving the robustness of the predictive model [40].

Training deep learning models like MLPs requires substantial computational resources, especially when working with large datasets [41]. Cloud platforms such as AWS, Microsoft Azure, and Google Cloud provide virtual machines and GPUs tailored for machine learning workloads [42]. These services allow rapid provisioning of training environments with configurable compute power and memory [43]. The cloud also facilitates collaborative model development through shared notebooks, version control, and containerization [44]. Model training can be distributed across multiple nodes to accelerate convergence and experimentation [45]. As a result, development teams can iterate faster and deploy optimized models in a cost-effective and scalable manner [46]. To ensure the reliability of defect prediction systems, MLP models must be rigorously evaluated using appropriate performance metrics. Common metrics include accuracy, precision, recall, F1-score, and Area Under the ROC Curve (AUC). Precision and recall are particularly important in defect prediction, as they measure the model's ability to correctly identify defective modules while minimizing false positives. Cross-validation techniques are used to assess model generalizability, and confusion matrices provide insights into classification errors. Performance benchmarking helps in selecting optimal model architectures and hyperparameters for specific datasets and use cases.

2. LITERATURE SURVEY

The convergence of artificial intelligence (AI), cloud computing, and the Internet of Things (IoT) has brought transformative changes across multiple sectors [47]. Each of these technologies plays a distinct role: IoT devices collect data from various sources in real-time, cloud platforms offer scalable and elastic infrastructure for data storage and computation, and AI algorithms provide intelligent analysis and decision-making capabilities [48]. When integrated, these technologies enable highly responsive, adaptive, and predictive systems capable of supporting diverse applications—from healthcare and finance to cybersecurity and customer relationship management (CRM). The literature reveals a growing trend of hybrid frameworks that leverage these synergies for enhanced precision, reliability, and scalability [49]. In healthcare, integrated AI-Cloud-IoT frameworks have proven particularly impactful. Hybrid neural-fuzzy models, which combine rule-based fuzzy systems with the adaptive learning capabilities of neural networks, have been deployed to manage and analyze uncertain medical data [50]. These systems are typically hosted on cloud infrastructure, enabling real-time patient monitoring and decision support. Studies have demonstrated accuracy rates exceeding 97%, highlighting their effectiveness in clinical settings [51]. The responsiveness and scalability of cloud platforms ensure that large volumes of data from wearable sensors and medical devices can be processed efficiently.

Medical decision support systems benefit from cloud-enabled real-time data processing. Continuous data strems from IoT devices, such as ECG monitors, glucose meters, and blood pressure cuffs, are transmitted to the cloud, where AI models analyze them instantly [52]. This capability allows for the early detection of anomalies and the prompt dispatch of alerts to medical professionals [53]. Hybrid fog-cloud architectures help reduce latency by preprocessing data at the network edge (fog layer) before sending it to the cloud. This architecture has proven valuable in applications like cardiovascular diagnostics and emergency response systems. Deep learning models, such as Deep Belief Networks (DBNs) and Long Short-Term Memory (LSTM) networks, have been successfully applied in chronic disease monitoring [54]. These models analyze longitudinal health data to detect patterns and predict the onset of conditions like diabetes, hypertension, and chronic obstructive pulmonary disease. Heuristic optimization techniques such as Ant Colony Optimization (ACO) are often employed to enhance model performance by fine-tuning hyperparameters [55]. Hybrid models like GWO-DBN (Gray Wolf Optimizer with DBN) and ACO-LSTM outperform traditional models in both accuracy and specificity.

Interpreting biomedical signals such as ECG and EEG requires high-precision tools capable of distinguishing between normal and pathological patterns [56]. Hybrid AI models deployed on fog-cloud infrastructures have demonstrated efficacy in this regard. These systems use AI to extract features from signals and classify them based on trained models [57]. The fog layer ensures quick feedback for urgent anomalies, while the cloud performs in-depth historical analysis for long-term treatment planning. Such architectures reduce energy consumption and network load, making them suitable for resource-constrained environments [58]. Clinical Decision Support Systems (CDSS) have evolved to become more intelligent and context-aware through AI and cloud integration. These systems aggregate patient records, lab results, imaging data, and treatment histories to generate evidence-based recommendations. Cloud infrastructure supports seamless integration of disparate data sources, while AI models ensure timely, personalized medical advice. Studies report that cloud-based CDSS improve diagnostic accuracy, reduce treatment errors, and assist clinicians in handling complex decision-making scenarios [59].

Elder care has seen the deployment of AI models to predict risks such as falls, delirium, and dysphagia. These models leverage both sensor-generated behavioral data and clinical records to assess patient vulnerability. Ensemble learning methods, which combine multiple weak learners to form a strong predictor, have significantly boosted accuracy rates. Real-time alerts generated by cloud-hosted models enable timely caregiver interventions, improving outcomes and reducing hospitalizations. Privacy-preserving architectures ensure compliance with regulations like HIPAA and GDPR. Cybersecurity is another domain where AI-Cloud-IoT integration has yielded promising results [60]. Traditional security mechanisms often fail to detect sophisticated, multi-vector cyber threats. AI-driven solutions, however, can adaptively learn threat patterns from network behavior and generate predictive threat intelligence [61]. Systems integrating graphical password schemes, AI CAPTCHAs, and AES encryption with neural networks have demonstrated improved resilience to brute-force and phishing attacks. These innovations are particularly crucial for high-security sectors such as banking and healthcare.

Federated learning frameworks allow multiple parties to collaboratively train AI models without sharing raw data. Techniques like split learning, graph neural networks (GNNs), and hash graph-based consensus mechanisms have been incorporated to enhance privacy and scalability [62]. These decentralized architectures

enable secure and efficient data exchange in sensitive environments, such as healthcare and finance. Studies show substantial improvements in anomaly detection, latency reduction, and throughput using these techniques. The financial sector has embraced AI and cloud technologies to extend services to underbanked and rural populations. IoT devices collect data on agricultural outputs, transaction behaviors, and socio-economic conditions. Cloud platforms process this data to assess creditworthiness, thereby facilitating microfinancing and insurance services. AI models personalize financial recommendations and detect fraudulent activities [63]. These systems promote economic inclusivity, support small businesses, and reduce the urban-rural income divide.

AI-powered CRM systems, when integrated with cloud infrastructures, enable real-time analysis of customer behavior and feedback. These models use sentiment analysis, natural language processing, and predictive modeling to anticipate customer needs, improve response times, and personalize engagement [64]. Cloud support allows CRM systems to scale across global operations while maintaining low latency and high availability. These systems have led to measurable improvements in customer satisfaction, loyalty, and operational efficiency. Predictive models built on machine learning techniques like Random Forests, Gradient Boosting Machines, and Support Vector Machines have shown success in identifying high-risk individuals in elder care [65]. Combining IoT sensor data with clinical records, these systems predict conditions such as pneumonia, falls, and cardiovascular anomalies. Cloud-based analytics platforms offer scalable and secure environments to manage large datasets and deploy real-time alerts to caregivers and medical professionals.

To secure the vast volumes of IoT data, researchers have developed hybrid cryptographic key generation mechanisms using swarm intelligence and evolutionary algorithms. These techniques provide high entropy, resilience against brute-force attacks, and compatibility with quantum-resistant frameworks [66]. AI is used to adaptively select encryption schemes based on usage context, further enhancing the robustness of data protection across heterogeneous devices and networks [67]. Vehicular Cloud Computing (VCC) systems benefit from trust estimation frameworks that integrate AI-driven anomaly detection, blockchain-based validation, and reputation scoring. These architectures assess trustworthiness in real-time, helping vehicles make informed decisions about data sharing and route coordination [68]. Formal threat modeling ensures resilience against spoofing, data injection, and denial-of-service attacks. These trust-based mechanisms significantly improve traffic coordination and reduce accident risks.

Facial recognition systems have achieved new heights with the integration of cloud analytics. AI models deployed on scalable cloud platforms perform face detection, recognition, and tracking with high accuracy and low latency. Real-time performance is enabled through distributed cloud infrastructure, supporting applications in surveillance, smart cities, and social networking [69]. Privacy-preserving techniques such as differential privacy and homomorphic encryption ensure data security and compliance.

3. PROBLEM STATEMENT

As software systems grow in complexity [70], traditional approaches to defect prediction and testing are becoming less and less viable [71], thus creating inefficiencies in defect detection and resolution [72]. Manual inspection, heuristic-based methods, and traditional defect prediction models offer scalability challenges in the sense that they would not accommodate large software codebases under rapid changes of constant updates. The aforementioned increase in complexity compels the need for an adaptable, efficient new approach that can predict defects on the fly, prioritize testing resources, and scale with continuous integration practices. The ultimate aim of the research is the design of a cloud-based AI solution that employs Multi-Layer Perceptron (MLP) models for the defect prediction of software, thus enhancing efficacy and accuracy in defect detection, while addressing the pitfalls of the traditional testing and defect prediction approaches.

3.1 OBJECTIVES

- Define the constraints of traditional defect prediction for complicated and evolving software.
- Create an MLP framework for Defect Detection that is AI-cloud based to accomplish precise and scalable defect detection from software projects.
- Implement the Z-score normalization method and a Lasso data preprocessing step.
- Configure the MLP using sigmoid activation to classify the software defects.
- Utilize Bayesian Optimization to identify the optimal hyperparameters of the model.

- Evaluate the end result of the model using metrics of: accuracy, precision, recall and F1 score.
- Show the benefits of scaling and near real-time integration with the cloud.

4. PROPOSED METHODOLOGY

The Figure starts with the collection of the software defect dataset, which works as the first pre-processing of the input data. The Z-score normalization technique is then applied to pre-process the data and standardize the features. The Lasso Regression method is used for feature selection to find the most relevant attributes. Meanwhile, the Multi-Layered Perceptron (MLP) model configured with a Sigmoid activation function is prepared for the defect prediction. The selected features and pre-processed data are simultaneously uploaded into cloud storage for future retrieval and use. The defect prediction harnesses the fine-tuned input to render the defect prediction results highlighting the likelihood of defects in the software modules. Such a structured approach leads to better accuracy, scalability, and defect detection efficiency.

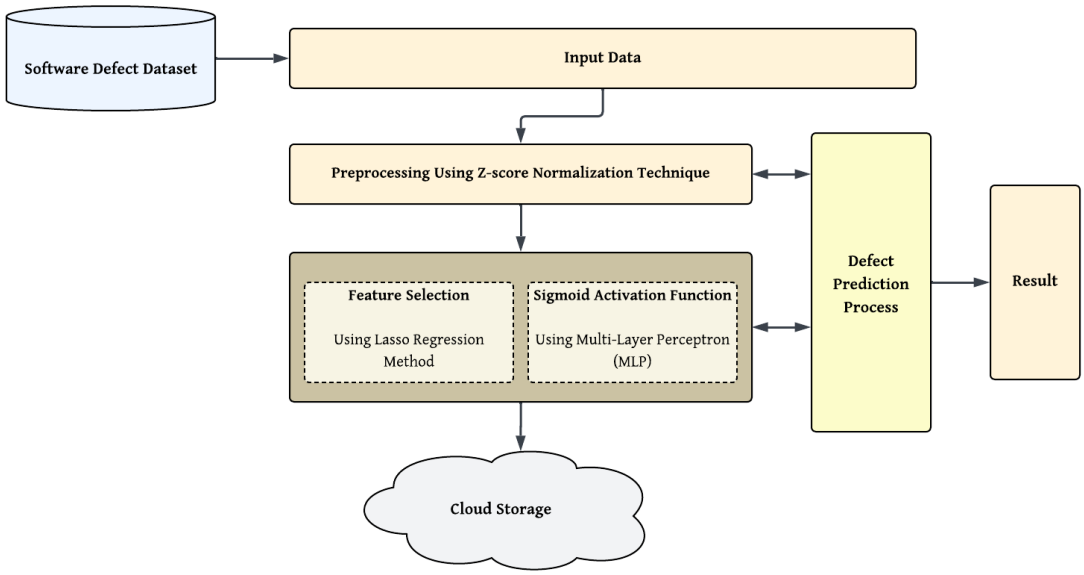


Figure 1: AI-Driven Software Defect Prediction Using Cloud Infrastructure

4.1 DATA COLLECTION

The data gathering for this study involves using a software defect dataset, with multiple features that are crucial to defect prediction. Such features are AvgCyclomatic (Cyclomatic complexity), CountClassBase (Base class count), CountClassCoupled, coupling between classes), CountDeclInstanceMethod (Declared instance methods), CountDeclMethodPublic (Declared public methods), and CountLineCode (Lines of code). A good metric reflects the structural complexity of some software modules that tend to be related to the defect-obliviousness probability correlate. A label indicating whether a module is defective or non-defective is also included in order to apply predictive models to classify the defect status from these characteristics. This structured dataset forms the base for training the AI-driven defect prediction model.

4.2 DATA PREPROCESSING

A. Handling Missing Values

The dataset might have some missing values due to incompleteness and unavailability of data. Handling of missing values assumes primary importance as this can cause inaccurate predictions of the model or can lead to improper training of the model. Three common techniques for your reference are as follows,

Mean Imputation: We can fill up missing values by using the mean of those value available instead of dropping it. It is a straightforward method and could possibly harm the distribution of data.

KNN Imputation: You conduct prediction of missingness with the help of the K-nearest neighbours, which is applied on the data points nearest to the one with missing attribute values. It is a much more sophisticated and superior approach since it preserves the correlation between associated features.

Multiple Imputation: One generates differently imputed datasets on which analyses are performed, and the results are then combined. This approach is robust because it considers the variability of imputation.

B. Normalization

Normalization and standardization are a must to assure all features are on the same scale before being inserted into differing AI models. When varying units and scales of features skew the working of the model, it becomes more evident in algorithms such as neural networks.

Z-score Normalization: Z-score normalization (standardization) adjusts all features to a distribution with a mean of 0 and a variance of 1. Therefore, the features are centred around zero with unit variance.

$$Z = \frac{X-\mu}{\sigma} \quad (1)$$

Where, μ is the mean and σ is the standard deviation of the feature.

4.3 CLOUD STORAGE

Pre-processed data, FEATURES Selected upon Z-score normalization, and Lasso regression are to be stored on a cloud platform for secure storage. This processed data is made easy to access and is also scalable and preserved for a long time for important information to be used in defect prediction. Cloud storage can be seamlessly integrated with Artificial Intelligence, allowing for easy retrieval of datasets to be updated when necessary. The advantage of this method also lies in carrying out collaborative research and remote accessibility, making future model upgrades and retraining flexible. This further enables version control of data at different stages of preprocessing and feature selection.

4.4 FEATURE SELECTION USING LASSO REGRESSION METHOD

Feature selection with the use of Lasso regression refers to a technique in which only the important features are selected by penalizing the coefficients of the less-valued features. The basic idea behind Lasso regression is that it shrinks coefficients of less productive variables toward zero, finally resulting in all these coefficients being removed from the analysis. Thereby, there arise only those features of interest that give the best fit to the model. Lasso guarantees that using the model, the most vital features would turn out to be some of the factors for predicting software defects like Cyclomatic Complexity or Lines of code, whereas other less-redundant minor features are dropped for efficient model improvement without model overfitting. Thus, by this approach, the model can both accurately and computationally efficiently predict defects.

4.5 MODEL DEVELOPMENT IN AI-DRIVEN DEFECT PREDICTION

A. AI Model Selection (MLP with Sigmoid Activation)

It consists of many hidden layers of neurons in a Multilayer Perceptron (MLP) model. Each neuron in the network performs the weighted sum of the inputs and takes the result into an activation function. The equation of a single neuron in layer l can be written as,

$$z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}) \quad (2)$$

Where, $z^{(l)}$ is the weighted sum (input to the activation function) for layer l. $W^{(l)}$ is the weight matrix for layer l. $a^{(l-1)}$ is the output from the previous layer (or the input features for the first layer). $b^{(l)}$ is the bias term for layer l.

Subsequently, after this step, the output values of neurons should be calculated by employing an activation function. For the MLP, normally the output of a neuron going into the output layer in a case of binary classification is processed by a sigmoid activation function as follows,

$$a^{(l)} = \sigma\left(z^{(l)}\right) = \frac{1}{1+e^{-z^{(l)}}} \quad (3)$$

Where, $\sigma(z^{(l)})$ represents the sigmoid activation function, which outputs values between 0 and 1, making it suitable for binary classification tasks like predicting defect status (defective or non-defective). $e^{-z^{(l)}}$ is the exponentiation of the negative input to the neuron.

The final result of the software module is the probability of it being defective, where output is mostly Output ≈ 1. means it is highly likely to be defective Output ≈ 0 shows the software module against a defect.

B. Training the Model (Backpropagation and Loss Function)

During the training of the model, it learns the best weights that reduce the error between the predicted output and the actual label given. The error will be measured using a loss function, most commonly the binary cross-entropy loss regarding binary classification, as follows,

$$L(y, \hat{y}) = -\frac{1}{m}\sum_{i=1}^{m} \left[y^{(i)} log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right) log\left(1 - \hat{y}^{(i)}\right)\right] \quad (4)$$

Where, $L(y, \hat{y})$ is the binary cross-entropy loss. m is the number of samples. $y^{(i)}$ is the true label (1 if the module is defective, 0 if not). $\hat{y}^{(i)}$ is the predicted probability for the i-th sample. General Relativistic information theory does indeed exhibit better burstiness and lower perplexity because data compression algorithm.

The training process, called gradient descent, uses backpropagation to modify the weights and minimize the loss. Backpropagation finds the gradient of the loss with respect to each weight. These gradients can then be used to modify the weights with gradient descent. The rule for updating the weights is,

$$W^{(l)} := W^{(l)} - \alpha \frac{\partial L}{\partial W^{(l)}} \quad (5)$$

Where, α is the learning rate, which controls the step size for each update. $\frac{\partial L}{\partial W^{(1)}}$ is the gradient of the loss with respect to the weight matrix $W^{(l)}$. As an operation of backpropagation that continues to run until its weights are adjusted and defects in prediction are deemed minimized.

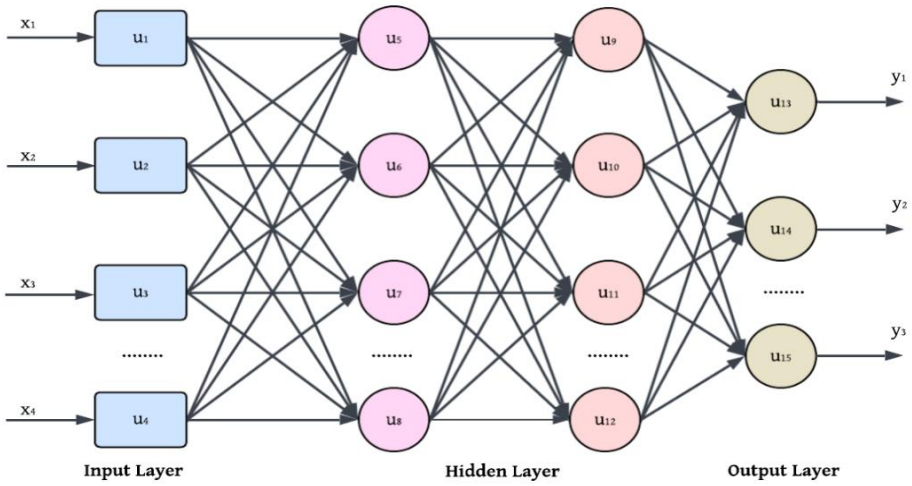


Figure 2: Multilayer Perceptron (MLP) Architecture for Software Defect Prediction

4.6 HYPERPARAMETER TUNING USING BAYESIAN OPTIMIZATION TECHNIQUE

Bayesian optimization is a powerful technique for searching the optimal hyperparameter values of an AI model. In making these searches, it iteratively improves over the hyperparameters while using a probabilistic model to estimate the performance of the hyperparameter settings. An explanation of each step of the Bayesian Optimization process is given below with accompanying equations.

a) Search Space

Identify the ranges for value of each hyperparameter you want to optimize. With Bayesian optimization, you first identify the search space, which comprises the range of values for the hyperparameters you want to optimize. For instance, in tuning a Multilayer Perceptron (MLP) model, you might want to optimize for learning rate, hidden-layers, and batch size. In general, a search space can be graphically illustrated using hyperparameter.

$$X = \{x_1 \in [a_1, b_1], x_2 \in [a_2, b_2], \dots, x_n \in [a_n, b_n]\} \quad (6)$$

Where, X is the search space. $x_1, x_2, \dots, x_n$ are the hyperparameters you are tuning (e.g., learning rate, number of hidden layers, batch size). $[a_i, b_i]$ is the range for each hyperparameter x_i, where a_i is the minimum value and b_i is the maximum value for that hyperparameter. Each of these parameters has a range associated with it, and Bayesian Optimization will explore this space to locate the optimal set of hyperparameters.

b) Initial Exploration

The optimization algorithm commences with the random selection of a few hyperparameter combinations from the defined search space for the model's performance evaluation. So, the Bayesian Optimization is said to start with the exploration of the search space. The algorithm essentially chooses some random hyperparameter combinations from the defined ranges and evaluates these combinations against a performance metric. The evaluation results then dictate the further action in the optimization process. The initial phase of exploring random hyperparameter combinations has been described since the algorithm has little prior information about the hyperparameter space; hence it employs a random sampling strategy to explore different combinations. The evaluation results assist in forming an understanding of which hyperparameters provide for a good model.

In this phase, the algorithm selects random hyperparameters and evaluates model performance, which can be expressed as,

$$f(x) = PerformanceMetric(x) \quad (7)$$

Where, $f(x)$ represents the performance of the model for a given hyperparameter set x. x is a vector of randomly selected hyperparameters from the defined search space.

c) Surrogate Model

Estimate hyperparameter performances using a probabilistic model, usually in this case a Gaussian Process. Bayesian Optimization, therefore, uses a surrogate model to predict how well the actual model will work for vantage points of hyperparameters, combinations that may not even have been evaluated yet after the initial evaluations. The data gathered from the initial evaluations are then used by the surrogate model, a probabilistic model, to estimate hyperparameter combination performance. Surrogate model equation (Gaussian process) is,

$$f(x) = GP\big(m(x), k(x, x')\big) \quad (8)$$

Where, $f(x)$ represents the function we are trying to optimize (e.g., the performance metric). $GP\big(m(x), k(x, x')\big)$ is the Gaussian Process. $m(x)$ is the mean function. $k(x, x')$ is the covariance (kernel) function between hyperparameters. The Gaussian Process helps predict the value of the objective function $(f(x))$ at unseen points in the search space. The model can catch all those portions on the search space where the undoubtedly best hyperparameters lie.

d) Acquisition Function

Use an acquisition function to decide where to explore next in the search space. The acquisition function is a key component of Bayesian Optimization. It decides where the next sampling will take place in the search space based on the surrogate model. The function tries to balance exploration (searching areas that have not been tested yet) and exploitation (focusing on areas where the model has performed well). Some acquisition functions include: Expected Improvement (EI): Measures the amount of improvement expected over the best-known value. Probability of Improvement (PI): Measures the probability that a new sample will improve over the best-known value. Expected Improvement (EI) equation is,

$$EI(x) = E\left[max\left(0, f(x) - f(x_{best})\right)\right] \quad (9)$$

Where, $f(x)$ is the predicted performance at hyperparameter x. $f(x_{best})$ is the best performance observed so far. The term $max\left(0, f(x) - f(x_{best})\right)$ represents the improvement over the best observed result. Acquisition functions are aimed at helping an optimization algorithm with selecting next hyperparameter sets for evaluation. The main idea is to balance between exploring regions that have not yet been adequately explored and offering a refined search in the vicinity of hyperparameters that look promising.

e) Iteration

In iterative fashion using the surrogate model and acquisition function to almost find the optimal hyperparameters. In Bayesian optimization, the selection of new hyperparameter values proceeds in an iterative manner, based on the surrogate model and acquisition function. After each iteration, the algorithm selects a fresh batch of hyperparameters for testing based on the current model and acquisition function. Once a new hyperparameter set is found, the model is trained and its performance evaluated. The results of this iteration are then appended to the data, which in turn acts to improve the surrogate model and instruct on the next search. Thus, we have the equation to Update Surrogate Model.

$$GP_{new} = Update(GP_{old}, f(x), x) \quad (10)$$

Where, GP_{new} is the updated Gaussian Process after each iteration. $f(x)$ is the observed performance from the model evaluation. This reading iterative process keeps refining the hyperparameters by consistently and multiple times improving the search direction toward the best-performance areas of the hyperparameter search space.

f) Convergence

The model will keep on iterating until it attains satisfactory performance or it reaches the maximum number of iterations. The Bayesian Optimization continues this way until it either finds an optimal set of hyperparameters that produces the best performance or reaches a predefined number of iterations or a convergence criterion. Convergence is expressed in the equation,

$$\text{Convergence} = |\text{ New Best Performance - Previous Best Performance }| < \epsilon \quad (11)$$

Where, ϵ is the tolerance value that determines when the optimization has sufficiently converged (e.g., when the performance improvement is smaller than a given threshold). The process comes to a close when the model has satisfied the necessity for an adequate performance or when it has gone through enough combinations of the hyperparameters.

According to the algorithm of Bayesian Optimization, the first step is to define a search space wherein each hyperparameter to be optimized is assigned the possible range. Initially, a few random combinations of hyperparameters are selected and evaluated based on their performance metrics like accuracy or F1-Score. The organization of this data is for building an approximative model, commonly a Gaussian Process for predicting the performance of unseen ones. The acquisition function then chooses combinations that will be tested, weighing the balance between exploration and exploitation. The iteration is then repeated until the desired model performance is reached, or until some convergence condition is satisfied.

5. RESULT AND DISCUSSION

Defect prediction models based on AI have indicated promising results in starkly detecting and accurately classifying defects. This model showed a remarkable reliability for defective modules as it delivered an accuracy of 0.92. The model attained a precision of 0.89 and a recall of 0.85 that is an indication of its indoors capacity for true positive identification and to avoid as much occurrence of false negatives as possible. The average of the combined result of an F1-score of 0.87 proves compromise in its precision and recall in affirmation of the efficiency of the model for defect prediction. Thus, this suggests that the model can carry out such tasks at very high efficiency, executing rather complex software defect prediction tasks. The capability of scaling and

processing big datasets quickly, therefore, enables real-time applications since it provides rapid detection of defects due to its cloud-based infrastructure. By hyperparameter tuning using Bayesian optimization, the model performance enhanced since it is now able to work very efficiently with continuous integration systems.

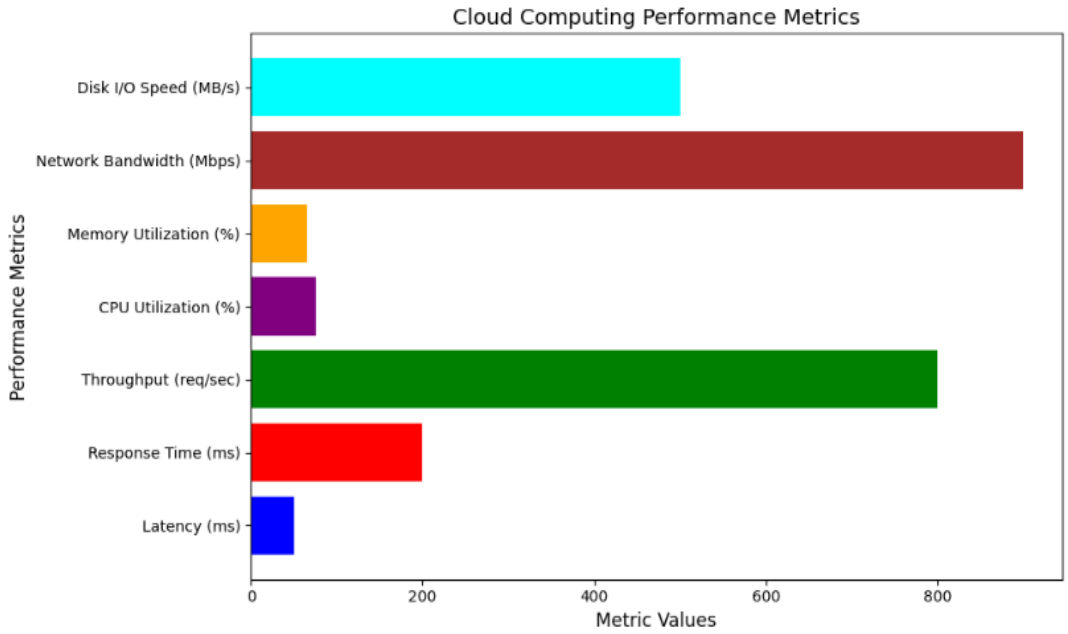
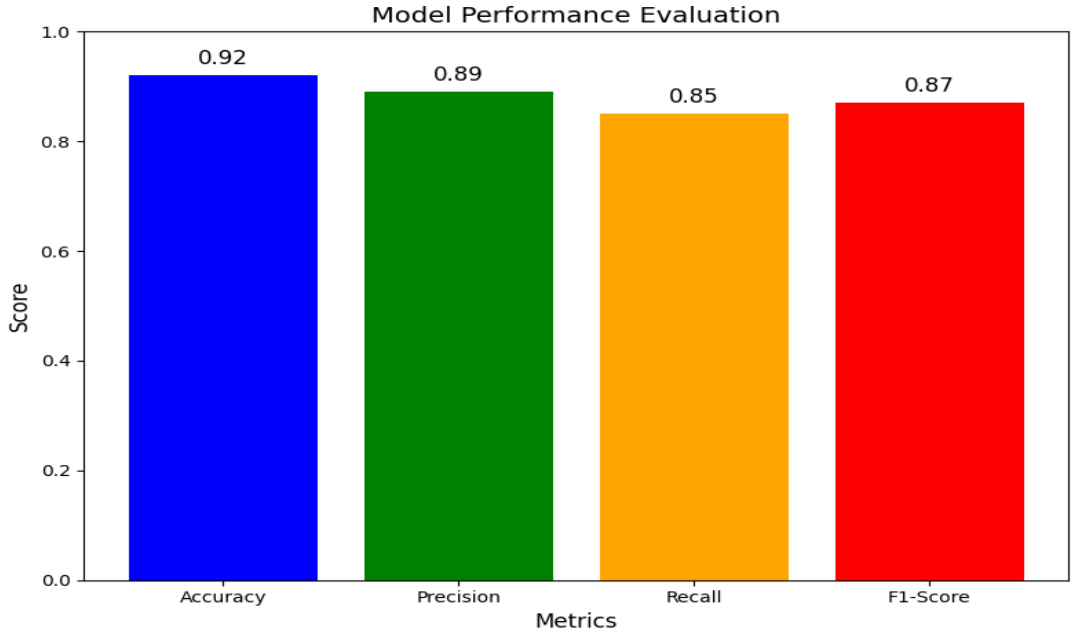


Figure 3: Cloud Computing Performance Metrics for AI-Driven Defect Prediction

The cloud performance metrics of the AI-driven predictive model for faults are illustrated in the next image. The parameters include Disk I/O Speed, Network Bandwidth, Memory Utilization, CPU Utilization, Throughput, Response Time, and Latency. The attributes concern the use of horizontal bars on each metric where Network Bandwidth and Throughput share the highest levels, demonstrating the ability of the system to handle large amounts of data and requests efficiently. Secondly, Disk I/O Speed and Response Time reflect data access and processing efficiency. Latency, a time delay in the system that could affect real-time predictions, indicates delayed instructions within the system. The graph contributes to a combined view of the global performance of the system in carrying out defect prediction-related tasks.



ISSN: 2456-4265
IJMEC 2021

Figure 4: Model Performance Evaluation for AI-Driven Defect Prediction

This shows the performance evaluation of the AI model used in your AI-driven defect prediction system. The bar chart is a depiction of four measures, namely Accuracy, Precision, Recall, and F1-Score, with the scores ranging from 0 to 1. Among them, the Accuracy score is the best, 0.92, indicating that most of the time, the model predicts the defect status correctly. Precision and Recall are also high at 0.89 and 0.85, respectively, confirming the model's ability to detect true defects and reject false positives. F1-Score, with a weight of 0.87, shows a good trade-off between precision and recall, thus supporting the overall performance rating of the model regarding defect detection.

6. CONCLUSION

This paper presents a highly effective, cloud-based, AI-enabled model for software defect prediction using Multi-Layer Perceptron (MLP) models. The model provides high performance-achieving an accuracy of 0.92 with a precision of 0.89, a recall of 0.85, and an F1 score of 0.87 by providing the cloud storage for scalability and Bayesian optimization for hyperparameter tuning. The restrictive yet relevant features are selected through Lasso regression, which helps in improving the efficiency of the model and counteracting overfitting. Cloud infrastructure is such that predictions can be made instantly and scaled easily for handling large amounts of data, thus promoting making the model very compatible with continuous integration systems. This approach significantly improves at predicting defects and enables more accurate, efficient, and automated testing in software development. It also lays a stronger foundation for further advancements in AI-driven software quality assurance in the future.

REFERENCE

[1] Hew, T. S., & Syed A. Kadir, S. L. (2017). Applying channel expansion and self-determination theory in predicting use behaviour of cloud-based VLE. Behaviour & Information Technology, 36(9), 875-896.
[2] Mohanarangan, V.D (2020). Improving Security Control in Cloud Computing for Healthcare Environments.Journal of Science and Technology, 5(6).
[3] Angelopoulos, A., Michailidis, E. T., Nomikos, N., Trakadas, P., Hatziefremidis, A., Voliotis, S., & Zahariadis, T. (2019). Tackling faults in the industry 4.0 era—a survey of machine-learning solutions and key aspects. Sensors, 20(1), 109.
[4] Ganesan, T. (2020). Machine learning-driven AI for financial fraud detection in IoT environments. International Journal of HRM and Organizational Behavior, 8(4).
[5] Saravanan, A., Jerald, J., & Delphin Carolina Rani, A. (2020). An intelligent constitutive and collaborative framework by integrating the design, inspection and testing activities using a cloud platform. International Journal of Computer Integrated Manufacturing, 33(5), 440-459.
[6] Deevi, D. P. (2020). Improving patient data security and privacy in mobile health care: A structure employing WBANs, multi-biometric key creation, and dynamic metadata rebuilding. International Journal of Engineering Research & Science & Technology, 16(4).
[7] Kusiak, A. (2020). Convolutional and generative adversarial neural networks in manufacturing. International Journal of Production Research, 58(5), 1594-1604.
[8] Mohanarangan, V.D. (2020). Assessing Long-Term Serum Sample Viability for Cardiovascular Risk Prediction in Rheumatoid Arthritis. International Journal of Information Technology & Computer Engineering, 8(2), 2347–3657.
[9] Farahani, B., Barzegari, M., Aliee, F. S., & Shaik, K. A. (2020). Towards collaborative intelligent IoT eHealth: From device to fog, and cloud. Microprocessors and Microsystems, 72, 102938.
[10] Koteswararao, D. (2020). Robust Software Testing for Distributed Systems Using Cloud Infrastructure, Automated Fault Injection, and XML Scenarios. International Journal of Information Technology & Computer Engineering, 8(2), ISSN 2347–3657.
[11] Kasturi, S. (2020, October). Post Implementation Evaluation of Coverage in Software Testing Using Monitoring Tools. In 2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON) (pp. 13-21). IEEE.

[12] Rajeswaran, A. (2020). Big Data Analytics and Demand-Information Sharing in ECommerce Supply Chains: Mitigating Manufacturer Encroachment and Channel Conflict. International Journal of Applied Science Engineering and Management, 14(2), ISSN2454-9940

[13] Arjoune, Y., & Faruque, S. (2020, January). Artificial intelligence for 5g wireless systems: Opportunities, challenges, and future research direction. In 2020 10th annual computing and communication workshop and conference (CCWC) (pp. 1023-1028). IEEE.

[14] Alagarsundaram, P. (2020). Analyzing the covariance matrix approach for DDoS HTTP attack detection in cloud environments. International Journal of Information Technology & Computer Engineering, 8(1).

[15] Kalusivalingam, A. K., Sharma, A., Patel, N., & Singh, V. (2020). Optimizing Resource Allocation with Reinforcement Learning and Genetic Algorithms: An AI-Driven Approach. International Journal of AI and ML, 1(2).

[16] Poovendran, A. (2020). Implementing AES Encryption Algorithm to Enhance Data Security in Cloud Computing. International Journal of Information technology & computer engineering, 8(2), I

[17] Tong, W., Hussain, A., Bo, W. X., & Maharjan, S. (2019). Artificial intelligence for vehicle-to-everything: A survey. IEEE Access, 7, 10823-10843.

[18] Sreekar, P. (2020). Cost-effective Cloud-Based Big Data Mining with K-means Clustering: An Analysis of Gaussian Data. International Journal of Engineering & Science Research,10(1), 229-249.

[19] Kalusivalingam, A. K., Sharma, A., Patel, N., & Singh, V. (2020). Enhancing Customer Segmentation through AI: Leveraging K-Means Clustering and Neural Network Classifiers. International Journal of AI and ML, 1(3).

[20] Karthikeyan, P. (2020). Real-Time Data Warehousing: Performance Insights of Semi-Stream Joins Using Mongodb. International Journal of Management Research & Review, 10(4), 38-49

[21] Rolan, G., Humphries, G., Jeffrey, L., Samaras, E., Antsoupova, T., & Stuart, K. (2019). More human than human? Artificial intelligence in the archive. Archives and Manuscripts, 47(2), 179-203.

[22] Mohan, R.S. (2020). Data-Driven Insights for Employee Retention: A Predictive Analytics Perspective. International Journal of Management Research & Review, 10(2), 44-59.

[23] Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., & Zhang, J. (2019). Edge intelligence: Paving the last mile of artificial intelligence with edge computing. Proceedings of the IEEE, 107(8), 1738-1762.

[24] Sitaraman, S. R. (2020). Optimizing Healthcare Data Streams Using Real-Time Big Data Analytics and AI Techniques. International Journal of Engineering Research and Science & Technology, 16(3), 9-22.

[25] Bora, A., & Sarma, K. K. (2018). Big data and deep learning for Stochastic Wireless channel. In Computational Intelligence in Sensor Networks (pp. 307-334). Berlin, Heidelberg: Springer Berlin Heidelberg.

[26] Panga, N. K. R. (2020). Leveraging heuristic sampling and ensemble learning for enhanced insurance big data classification. International Journal of Financial Management (IJFM), 9(1).

[27] Mushtaq, Z., Yaqub, A., Sani, S., & Khalid, A. (2020). Effective K-nearest neighbor classifications for Wisconsin breast cancer data sets. Journal of the Chinese Institute of Engineers, 43(1), 80-92.

[28] Gudivaka, R. L. (2020). Robotic Process Automation meets Cloud Computing: A Framework for Automated Scheduling in Social Robots. International Journal of Business and General Management (IJBGM), 8(4), 49-62.

[29] Wylie, B. K., Pastick, N. J., Picotte, J. J., & Deering, C. A. (2019). Geospatial data mining for digital raster mapping. GIScience & Remote Sensing, 56(3), 406-429.

[30] Gudivaka, R. K. (2020). Robotic Process Automation Optimization in Cloud Computing Via Two-Tier MAC and LYAPUNOV Techniques. International Journal of Business and General Management (IJBGM), 9(5), 75-92.

[31] Natarajan, D. R. (2020). AI-Generated Test Automation for Autonomous Software Verification: Enhancing Quality Assurance Through AI-Driven Testing. International Journal of HRM and Organizational Behavior, 8(4), 89-103.

[32] Deevi, D. P. (2020). Artificial neural network enhanced real-time simulation of electric traction systems incorporating electro-thermal inverter models and FEA. International Journal of Engineering and Science Research, 10(3), 36-48.

[33] Velaga, S. P. (2020). Ai-assisted Code Generation and Optimization: Leveraging Machine Learning to Enhance Software Development Processes. International Journal of Innovations in Engineering Research and Technology, 7(09), 177-186.

[34] Allur, N. S. (2020). Enhanced performance management in mobile networks: A big data framework incorporating DBSCAN speed anomaly detection and CCR efficiency assessment. Journal of Current Science, 8(4).

[35] Srinivas, N., Mandaloju, N., & Nadimpalli, S. V. (2020). Cross-platform application testing: AI-driven automation strategies. Artificial Intelligence and Machine Learning Review, 1(1), 8-17.
[36] Deevi, D. P. (2020). Real-time malware detection via adaptive gradient support vector regression combined with LSTM and hidden Markov models. Journal of Science and Technology, 5(4).
[37] Hourani, H., Hammad, A., & Lafi, M. (2019, April). The impact of artificial intelligence on software testing. In 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT) (pp. 565-570). IEEE.
[38] Dondapati, K. (2020). Integrating neural networks and heuristic methods in test case prioritization: A machine learning perspective. International Journal of Engineering & Science Research, 10(3), 49–56.
[39] Bosch, J., Olsson, H. H., & Crnkovic, I. (2018). It takes three to tango: Requirement, outcome/data, and AI driven development. In SiBW 2018, Software-intensive Business: Start-ups, Ecosystems and Platforms, Espoo, Finland, December 3, 2018 (pp. 177-192).
[40] Allur, N. S. (2020). Big data-driven agricultural supply chain management: Trustworthy scheduling optimization with DSS and MILP techniques. Current Science & Humanities, 8(4), 1–16.
[41] Benzaid, C., & Taleb, T. (2020). AI-driven zero touch network and service management in 5G and beyond: Challenges and research directions. Ieee Network, 34(2), 186-194.
[42] Gattupalli, K. (2020). Optimizing 3D printing materials for medical applications using AI, computational tools, and directed energy deposition. International Journal of Modern Electronics and Communication Engineering, 8(3).
[43] Dhieb, N., Ghazzai, H., Besbes, H., & Massoud, Y. (2020). A secure ai-driven architecture for automated insurance systems: Fraud detection and risk measurement. IEEE Access, 8, 58546-58558.
[44] Narla, S., Valivarthi, D. T., & Peddi, S. (2020). Cloud computing with artificial intelligence techniques: GWO-DBN hybrid algorithms for enhanced disease prediction in healthcare systems. Current Science & Humanities, 8(1), 14–30.
[45] David, L., Thakkar, A., Mercado, R., & Engkvist, O. (2020). Molecular representations in AI-driven drug discovery: a review and practical guide. Journal of cheminformatics, 12(1), 56.
[46] Kethu, S. S. (2020). AI and IoT-driven CRM with cloud computing: Intelligent frameworks and empirical models for banking industry applications. International Journal of Modern Electronics and Communication Engineering (IJMECE), 8(1), 54.
[47] Pentyala, D. K. (2018). AI-Driven Decision-Making for Ensuring Data Reliability in Distributed Cloud Systems. International Journal of Modern Computing, 1(1), 1-22.
[48] Vasamsetty, C. (2020). Clinical decision support systems and advanced data mining techniques for cardiovascular care: Unveiling patterns and trends. International Journal of Modern Electronics and Communication Engineering, 8(2).
[49] Chakriswaran, P., Vincent, D. R., Srinivasan, K., Sharma, V., Chang, C. Y., & Reina, D. G. (2019). Emotion AI-driven sentiment analysis: A survey, future research directions, and open issues. Applied Sciences, 9(24), 5462.
[50] Kadiyala, B. (2020). Multi-swarm adaptive differential evolution and Gaussian walk group search optimization for secured IoT data sharing using supersingular elliptic curve isogeny cryptography,International Journal of Modern Electronics and Communication Engineering,8(3).
[51] Savadjiev, P., Chong, J., Dohan, A., Vakalopoulou, M., Reinhold, C., Paragios, N., & Gallix, B. (2019). Demystification of AI-driven medical image interpretation: past, present and future. European radiology, 29, 1616-1624.
[52] Valivarthi, D. T. (2020). Blockchain-powered AI-based secure HRM data management: Machine learning-driven predictive control and sparse matrix decomposition techniques. International Journal of Modern Electronics and Communication Engineering.8(4)
[53] Firouzi, F., Farahani, B., Barzegari, M., & Daneshmand, M. (2020). AI-driven data monetization: The other face of data in IoT-based smart and connected health. IEEE Internet of Things Journal, 9(8), 5581-5599.
[54] Jadon, R. (2020). Improving AI-driven software solutions with memory-augmented neural networks, hierarchical multi-agent learning, and concept bottleneck models. International Journal of Information Technology and Computer Engineering, 8(2).
[55] Sodhro, A. H., Pirbhulal, S., & De Albuquerque, V. H. C. (2019). Artificial intelligence-driven mechanism for edge computing-based industrial applications. IEEE Transactions on Industrial Informatics, 15(7), 4235-4243.
[56] Boyapati, S. (2020). Assessing digital finance as a cloud path for income equality: Evidence from urban and rural economies. International Journal of Modern Electronics and Communication Engineering (IJMECE), 8(3).

[57] Pal, R., Sekh, A. A., Kar, S., & Prasad, D. K. (2020). Neural network based country wise risk prediction of COVID-19. Applied Sciences, 10(18), 6448.
[58] Gaius Yallamelli, A. R. (2020). A cloud-based financial data modeling system using GBDT, ALBERT, and Firefly algorithm optimization for high-dimensional generative topographic mapping. International Journal of Modern Electronics and Communication Engineering8(4).
[59] Pentyala, D. K. (2020). Enhancing the Reliability of Data Pipelines in Cloud Infrastructures Through AI-Driven Solutions. The Computertech, 30-49.
[60] Yalla, R. K. M. K., Yallamelli, A. R. G., & Mamidala, V. (2020). Comprehensive approach for mobile data security in cloud computing using RSA algorithm. Journal of Current Science & Humanities, 8(3).
[61] Wei, Y., Peng, M., & Liu, Y. (2020). Intent-based networks for 6G: Insights and challenges. Digital Communications and Networks, 6(3), 270-280.
[62] Samudrala, V. K. (2020). AI-powered anomaly detection for cross-cloud secure data sharing in multi-cloud healthcare networks. Journal of Current Science & Humanities, 8(2), 11–22.
[63] Angelopoulos, A., Michailidis, E. T., Nomikos, N., Trakadas, P., Hatziefremidis, A., Voliotis, S., & Zahariadis, T. (2019). Tackling faults in the industry 4.0 era—a survey of machine-learning solutions and key aspects. Sensors, 20(1), 109.
[64] Ayyadurai, R. (2020). Smart surveillance methodology: Utilizing machine learning and AI with blockchain for bitcoin transactions. World Journal of Advanced Engineering Technology and Sciences, 1(1), 110–120.
[65] Wan, J., Li, X., Dai, H. N., Kusiak, A., Martinez-Garcia, M., & Li, D. (2020). Artificial-intelligence-driven customized manufacturing factory: key technologies, applications, and challenges. Proceedings of the IEEE, 109(4), 377-398.
[66] Chauhan, G. S., & Jadon, R. (2020). AI and ML-powered CAPTCHA and advanced graphical passwords: Integrating the DROP methodology, AES encryption, and neural network-based authentication for enhanced security. World Journal of Advanced Engineering Technology and Sciences, 1(1), 121–132.
[67] Balasubramanian, A., Gurushankar, N., & Eng, M. E. (2019). AI-powered hardware fault detection and self-healing mechanisms. International Journal of Core Engineering & Management, 6(4), 23-30.
[68] Narla, S. (2020). Transforming smart environments with multi-tier cloud sensing, big data, and 5G technology. International Journal of Computer Science Engineering Techniques, 5(1), 1-10.
[69] Thomsen, K., Iversen, L., Titlestad, T. L., & Winther, O. (2020). Systematic review of machine learning for diagnosis and prognosis in dermatology. Journal of Dermatological Treatment, 31(5), 496-510.
[70] Alavilli, S. K. (2020). Predicting heart failure with explainable deep learning using advanced temporal convolutional networks. International Journal of Computer Science Engineering Techniques, 5(2).
[71] De Almeida, A. F., Moreira, R., & Rodrigues, T. (2019). Synthetic organic chemistry driven by artificial intelligence. Nature Reviews Chemistry, 3(10), 589-604.
[72] Dondapati, K. (2020). Leveraging backpropagation neural networks and generative adversarial networks to enhance channel state information synthesis in millimeter-wave networks. International Journal of Modern Electronics and Communication Engineering, 8(3), 81-90