IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 13, Issue 5, May 2024

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.423

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305043 |

Predicting Software Faults with Machine Learning Tool

Miss. Mehnaz Sheikh, Prof. Hirendra Hajare

M.Tech Scholar, Department of Computer Engineering, Ballarpur Institute of Technology, Bamni, Maharashtra, India

Assistant Professor, Department of Computer Engineering, Ballarpur Institute of Technology, Bamni, Maharashtra, India

ABSTRACT: Software quality is important in many business sectors such as tool, aviation, healthcare, insurance, where software metrics measured accuracy, performance, variability, accuracy, and defect-freeness will assess quality, although it's different for every organization**.** Machine learning (ML) helps predict faults, recover reliability and performance, and thus reduce maintenance costs. Application of various ML algorithms such as ANN, RF, DT, LR, GP, SMOREG, M5P, where SMOREG shows good performance. ML methods are optimized by K-means clustering and Particle Swarm Optimization for accuracy and precision. Evaluation criteria include precision, accuracy, recall, f-measure, and confusion matrix. The optimized SVM and SVM models achieve maximum accuracies of 99% and 99.80%, respectively, which exceed preceding research efforts. Testing seeks to improve software quality, increase customer satisfaction, and make testing hard work economically viable by obtaining bug-free software Bug testing is essential to software quality, especially as complex software and subsequently advanced error rates make manual searches more time-consuming. The purpose of this paper is to evaluate the performance of state-of-the-art machine learning algorithms for software error classification using seven datasets obtained from the NASA Promise Dataset Repository Among the proposed algorithms, neural networks and gradient boosting classifiers appear as the top performers.

KEYWORDS: Fault Prediction, Bug Prediction, Defect Prediction, Software Quality Assurance, Software Maintenance, Software Metrics, Code Analysis, Feature Selection, Classification Algorithms, Regression Analysis, Neural Networks Support Vector Machines (SVM), Decision Trees, Random Forests, Gradient Boosting Machines (GBM).

I. INTRODUCTION

Repeated software failures over time usually indicate the presence of software bugs, which are bugs caused by software developers and stakeholders. The main goal of software bug prediction is to increase quality, reduce costs, and reduce the time associated with software products. Software bugs, also known as defects, refer to deficiencies in the functionality of software products that programmers and customers want. Machine learning represents an important and exciting research area, focused on extracting meaningful insights from large datasets. Its main goal is to identify useful patterns in data, whether structured (e.g., multiple databases) or unstructured (e.g., natural language documentation). This study examines in detail the root causes of software failures that contribute to software bugs, consumes resources for testing and maintenance after allotted stakeholders Furthermore, it explores recommended solutions shows to address software failures, including machine learning principles and applications in software engineering.

Software Defect Prediction (SDP) aims to improve software quality and reduce testing costs through different classification models based on different machine learning techniques It is a key part of the software development lifecycle, using historical data to predict defects predict and prioritize resource allocations. Several machine learning techniques are used in SDP, including decision trees, Naive Bayes, support vector machine (SVM), and random forest. Recent developments include ensemble Ing techniques and feature selection methods such as principal component analysis (PCA). Attention is paid to the software parameters that are required to successfully predict software bugs. The data set used in this study comes from the NASA Promise Repository, which provides insight into the various applications explored by NASA since 2005. After data preprocessing and feature selection, K-means clustering output is classified, and ML algorithms such as SVM, NB, and invalid RF. A group approach combines the results, and evaluates the performance of the model using accuracy, precision, recall, F-measure, performance error coefficient, and confusion matrix analysis.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305043 |

In recent years, software quality has emerged as an important aspect of software development. Software error prediction plays an important role in improving software quality, increasing productivity, and reducing maintenance costs. Defective software directly incurs costs, project schedules, and maintenance costs. Implementing defect prediction models prior to software implementation improves performance, reliability, and user satisfaction. This study investigates the performance of eight machine learning (ML) algorithms: ANN, RF, RT, DT, LR, GP, SMOreg, and M5P. The WEKA 3.8.3 tool is used with k-fold cross-validation and percentage split techniques on a dataset obtained from the NASA Promise Repository. The measures include the correlation coefficient (R2), mean standard error (MAE), root mean square error (RMSE), root mean square error (RAE), and root mean square error (RRSE).

II. RELATED WORK

Recently, several studies investigated the effectiveness of machine learning (ML) algorithms in predicting software bugs. This study evaluated Naive Base, Multi-Layer Perceptron, Radial Basis Function, Support Vector Machine, K-Nearest Neighbors and other algorithms on 12 NASA data sets to evaluate accuracy, recall, F-measure, Metrics like ROC area and Matthew's correlation coefficient was used. Furthermore, the implementation of the Levenberg-Marquart algorithm showed higher accuracy in error prediction compared to polynomial neural networks. Another study compared artificial neural networks and SVM for error prediction and found that SVM outperformed ANN, especially in memory-related tasks. Furthermore, the comparison between SVM, Decision Trees, and Random Forests on NASA data sets revealed that Random Forests is the best performing classifier.

Addressing the challenge of interdisciplinary fault forecasting, many studies proposed various approaches to address class imbalance problems, including strategies using data modelling and over-transfer learning techniques The empirical study demonstrated the effectiveness of these techniques to improve the performance of industry-specific error prediction models. However, traditional software error prediction models that rely solely on program features may fail to adequately analyse program semantics. To address this, recent studies have proposed alternative approaches, such as the production of control flow graphs that are excluded from source code compilation and the use of Multiview convolutional neural networks These approaches have demonstrated error prediction performance a higher proved than traditional methods. In addition, researchers have explored other approaches such as genetic simulation, mixed modelling and genetic simulation to combine backward and backpropagation neural networks, to improve the accuracy of fault prediction.

The document discusses a study on software defect prediction using machine learning (ML) algorithms. The main objective is to improve software reliability and performance while reducing maintenance costs by predicting defects in software. The paper evaluates the performance of eight ML algorithms, including artificial neural networks (ANNs), random forest (RF), random tree (RT), decision table (DT), linear regression (LR), gaussian processes (GP), SMOreg, and M5P, in predicting software defects. The study uses a dataset from the NASA Promise repository and employs various statistical metrics, such as correlation coefficient (R^2), mean absolute error (MAE), root mean squared error (RMSE), relative absolute error (RAE), and root-relative squared error (RRSE) to evaluate the accuracy of the predictions. The investigation focuses on the importance of software defect prediction in the software development cycle to enhance software quality, performance, and reduce maintenance costs. The study evaluates the performance of ML algorithms in predicting software defects using historical data. It assesses the accuracy of the predictions through various statistical metrics and testing modes, such as 10-fold cross-validation and percentage split techniques.

The results show that the SMOreg classifier performed the best, while the ANN classifier had the worst performance. Additionally, the study highlights the future work, including the investigation of other ML classifiers, the use of more datasets, and the consideration of practical software defect prediction scenarios. In short, the study emphasizes the significance of software defect prediction using ML algorithms to enhance software performance and reduce maintenance costs. The results demonstrate the effectiveness of ML algorithms in predicting software defects, with the SMOreg classifier showing the best performance. The study suggests future work to explore and compare additional ML classifiers, incorporate more datasets, and consider practical software defect prediction scenarios to further improve prediction accuracy.

III. METHODOLOGY

3.1 EXISTING METHODOLOGY:

As this paper is about another family of metrics in SDP ie. test metrics, it was considered that more algorithms are needed for many classes because the prediction performance of the test metrics was not known to be successful in previous studies where therefore it was considered important Keeping this in mind, it was developed literature review

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305043 |

to gain more insights in these areas. The results of the literature review were then used in the experiment. Figure 3.1 shows the relationship between these two factors. This figure also shows how the answers to the survey questions related to the literature review served as input to the experiment.As this paper is about another family of metrics in SDP ie. test metrics, many algorithms were considered important for many classes because the predictive performance of test metrics was not known to be successful in previous studies where therefore considered important Keeping this in mind, literature was developed were analyzed to gain more insight in these areas. The results of the literature review were then used in the experiment. Figure 1 shows the relationship between these two factors. This figure also shows how the answers to the survey questions related to the literature review served as input to the experiment.

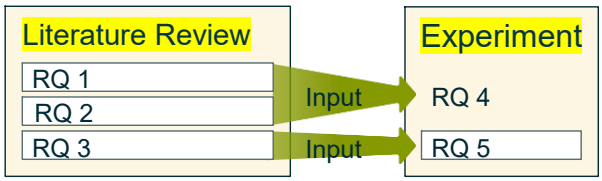

Figure 1: Relationship between methods and research questions

- **Problem definition**

Different ML algorithms were used to test the discriminatory power of different metrics The way they are spun to store these metrics, and the way predictive models are built and tested Select the repository to use the metrics on in before error accumulation. The main factors in choosing the repository for this experiment were size in terms of file count and tests. Larger than larger than smaller sizes were preferred to ensure accuracy and generalizability of the results. Since the main theme of the program is to incorporate testing metrics, the shop can incur high testing costs. Choosing a repository with high test coverage assures that more files have been tested and therefore reduces testing bias. About 200 KLOCs from the selected archives were spread over about 2000 files and had about 70% test coverage.

- **Problem statement**

Therefore, the general consensus is that meta-analysis is a powerful tool but, not surprisingly, it can be misused when:

• Used without any underlying model or theory. One example of a meta-analysis from an epidemiological study was published by Egger et al. [26] who argue that the causal relationship between smoking and suicide rates is not biologically obvious, rather it is the social and psychological conditions that predispose to suicide and in turn smoking communicate and yet subsequently can easily convince one of the 'validities' of a set of results.

• Meta-analysis cannot turn poor quality studies into 'gold'. If the primary study is not minimal, the conclusions are at risk no matter how they integrate with other studies

• Discrepancies are ignored such as differences between studies, blinding, etc. Although this can be addressed by sensitivity analysis.

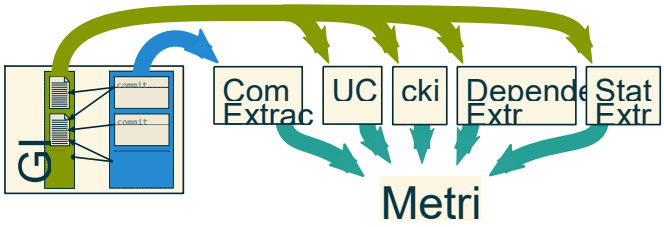

Figure 2: Visualization of the extraction process

3.1.1. Identify faulty files

In order to train a learner, the source code files need to be classified as either defective or non-defective. For this analysis, all commits to the SCM were investigated. If a TR identification number is included in the commit, it is an indication that a fault has been corrected and hence, the file that is affected by the commit has been corrected. That file is thereafter classified as defective, while files that are not affected by any TRs are classified as non-defective.

3.1.2. Process Design Collection

Process metrics are derived from analyzing commitments made to the repository, extracting information on files suffering from each devote. SCM enables the extraction of usage metrics like authors and line changes, such as additions, deletions, and adjustments. Author metrics are generated via assigning a binary fee to every creator throughout all documents. Line trade metrics embody adjustments, deletions, and churn, with statistical parameters which includes minimum, most, mean, median, trendy deviation, and sum calculated for every metric. These metrics, targeted in Table 4.1 below "Process" and "Author" headings, offer insights into the improvement procedure.

3.1.3. Collection of static signal meters

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305043 |

There are a number of independent tools for collecting static code metrics, but each of them has somewhat limited functionality. Therefore, several different tools are used to collect more static code metrics. Unified code computation (UCC) was used to collect LOC and other relevant metrics. UCC stores metrics such as SLOC-P, BLOC, CLOC and SLOC-L along with compiler instruction counts and data declaration counts. UCC and its properties have been documented by Nguyen et al. in. CCN was extracted using an instrument developed for this master thesis project. CKJM6 is a tool used to calculate CKOO metrices suite.The metrics calculated by ckjm consist of eight different metrics, six from the original CK metric set, and two additional metrics.

- **Limitations of the existing system**

Since I have no specific information about the existing framework you describe, I will provide some general limitations that can be attached to software bug prediction systems These limitations can vary depending on the method and techniques the method of production. Here are a few general restrictions:

1. Limited Feature Set: Existing systems are using limited features or metrics to predict faults. This may overlook important sources of errors. The inclusion of additional relevant parameters can improve the accuracy and efficiency of forecasts.
2. Lack of data: The quality and completeness of the training system data can significantly affect its performance. Inconsistencies in the historical data, missing standards, or incorrect error records can lead to biased or unreliable forecasts.
3. Challenges in Feature Selection: It is important to select more informative and relevant features for accurate error prediction. If the existing system lacks effective selection mechanisms, it may add unnecessary or unnecessary features, resulting in decreased forecast accuracy or overfitting
4. Insufficient training data: The amount and type of training data can affect the system's ability to generalize and make accurate predictions. If the existing system has limited or imbalanced training data, it may struggle to capture patterns and underlying processes related to error occurrence.

3.2 Proposed System

Based on the limitations of the existing system, here's a proposed system for software defect prediction:

1. Enhanced feature set: Expand the feature set used for defect prediction by incorporating additional relevant metrics and attributes. This can include code complexity, code churn, code coverage, developer experience, historical defect data, and any other domain-specific factors that can influence defect occurrence.
2. Robust data preprocessing: Implement thorough data preprocessing techniques to handle missing values, outliers, and data inconsistencies. This includes imputation methods for missing data, outlier detection and treatment, and normalization or scaling of the data to ensure consistency and improve model performance.
3. Advanced feature selection: Employ advanced feature selection techniques to identify the most informative and relevant features for defect prediction. This can include correlation analysis, information gain, feature importance analysis using machine learning algorithms, or domain expertise-driven feature selection methods.
4. Ensemble modeling: Utilize ensemble learning techniques to combine multiple machine learning models for defect prediction. Ensemble methods such as random forests, gradient boosting, or stacking can help improve prediction accuracy and reduce the impact of individual model biases.
5. Incorporate cross-validation: Apply cross-validation techniques during model training and evaluation to obtain more reliable and robust performance estimates. This involves splitting the data into multiple subsets, training and testing the model on different subsets, and aggregating the results to assess the model's generalization ability.

- **Advantages of Proposed System :**

1. Improved prediction accuracy: By expanding the feature set, implementing advanced feature selection techniques, and using ensemble modeling, the proposed system can enhance the accuracy of defect predictions. The inclusion of relevant metrics and the combination of multiple models help capture more complex patterns and dependencies, leading to more reliable predictions.
2. Better adaptability to evolving software: The proposed system incorporates mechanisms for incremental learning, allowing it to handle evolving software systems. By incorporating new data and adapting the models over time, the system can maintain its predictive performance as the software evolves, ensuring its relevance and effectiveness.
3. Enhanced resource allocation: The system optimizes resource allocation by considering the predictions and severity of defects. By efficiently allocating testing and debugging resources, the proposed system helps prioritize efforts, leading to more effective defect detection and resolution. This can result in improved overall software quality and reduced development and maintenance costs.
4. Interpretable predictions: By selecting machine learning algorithms that offer interpretability, the proposed system provides explanations for the factors contributing to defect predictions. This enhances the system's transparency

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305043 |

and trustworthiness, allowing developers and stakeholders to understand and validate the predictions, leading to better decision-making.

3.3 UML DIAGRAMS:

3.3.1 Sequence Diagram:

Sequence diagrams depict the dynamic behaviour of the system, particularly the interaction between objects or components over time. They illustrate the sequence of messages exchanged between objects and the order of their execution. Sequence diagrams help visualize the flow of control and data during a specific scenario or use case.

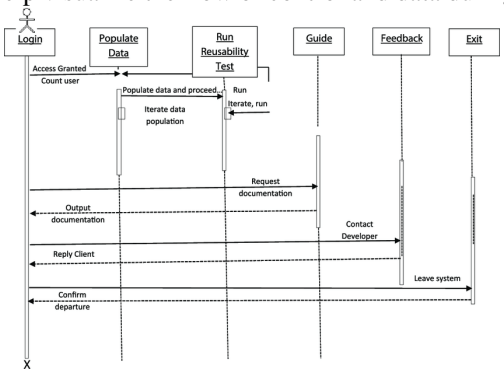

Figure 3: Sequence a software defect prediction system

3.3.2 Activity Diagram:

Activity diagrams capture the workflow or business processes within the system. They represent the activities, actions, and decision points, and the flow of control between them. Activity diagrams provide a visual representation of the system's logic and the order of operations.

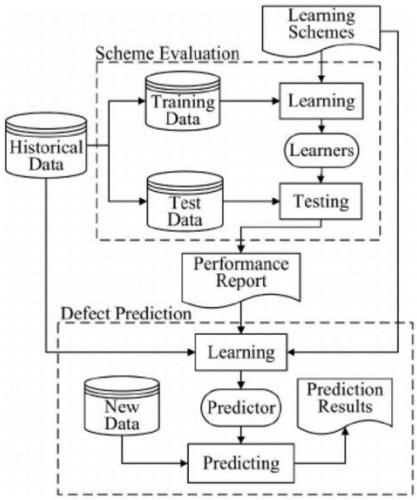

Figure 4: Activity Diagram a software defect prediction system.

IV. EXPERIMENTAL RESULTS

4.1 Pre-processing:

This study utilized seven datasets sourced from the NASA promise dataset repository [16] for software defect detection. The datasets employed are KC1, JM1, CM1, KC2, PC1, AT, and KC1 CL. Further details about each dataset are outlined in Table 1.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305043 |

Dataset	No. of attributes	No. of instances	% of faulty instances
CM1	22	498	9.83
JM1	22	10,885	19.35
KC1	22	2,109	15.45
KC2	22	522	20.50
PC1	22	1,109	6.94
AT	9	130	8.46
KC1 CL	95	145	44.82

Table 1: Characteristics of Dataset

To gain insight into how each factor affects outcome prediction, we examined each data set more closely. It turned out that many items showed strong correlations with others, creating an overlap. Consequently, these items were deleted in the final training phase. Principal component analysis (PCA) [17] was used as a method to trim the dimensionality of extensive data sets to reduce loss of information PCA for additional uncorrelated variables with high variance. Applying this approach to the KC1 and JM1 data sets resulted in significant performance improvements.

4.2 Models:

In our search, we tested various machine learning models to determine the most effective for our purposes. These models include logistic regression, random forest [18], naive Bayes, gradient boosting classifier, support vector machine, artificial neural network [19].

- Logistic regression: This model uses the logistic function to model a binary dependent variable [20].
- Random Forest: A cluster learning method, a random forest consists of many decision trees and outputs a class of mode or mean with predictions of the trees incorporating feature randomness during construction to ensure that the trees are not correlated.
- Naïve Bayes: This probabilistic classifier is based on Bayes theorem, which works on the assumption that features are conditionally independent [21].
- Gradient Boosting Classifier: This ensemble algorithm, widely used in machine learning competitions, adds samples sequentially, and subsequently improves the performance of its predecessors [22
- Support Vector Machine: This algorithm attempts to parse data in order to separate observations of a particular class by as wide an explicit path as possible. In addition to linear classification, SVMs can effectively perform nonlinear classification [23] .
- Artificial Neural Networks (ANN): ANNs are computational systems that consist of a collection of artificial neurons driven by the natural brain.

During training, the ANN was optimized with a learning rate of 0.0001, which decreased 10-fold with each validation loss plateau [24] . Binary cross-entropy loss functions were used:

$$L(y, \hat{y}) = -(y * \log(\hat{y}) + (1 - \hat{y}) * \log(1 - \hat{y})) \quad (1)$$

Throughout training, muscle weights are adjusted with the goal of minimizing functional loss. However, the choice of adapter significantly affects the magnitude and direction of these changes. For this experiment, we used the widely used adaptive moment estimation (ADAM) optimizer, which is known to perform well in similar cases [25] .

4.3 Result:

These results are obtained by 10-fold cross validation

	CM1	JM1	KC1	KC2	PC1	AT	KC1 CL
Naïve Byes	**83**	**77**	**75**	**82**	**81**	**72.7**	**73.7**
Logistic Regression	**86**	**70**	**80.1**	**83.7**	**79.8**	**81**	**80.5**
Gradient Boosting Classifier	**88**	**79**	**87.5**	**84**	**86**	**87**	**89.5**
ANN	**80**	**83.4**	**83**	**88.9**	**93**	**90**	**79**

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305043 |

Support Vector Machine	**85**	**75.4**	**83**	**86**	**85**	**88**	**78**
Random Forest	**83**	**77**	**85**	**79**	**89**	**91.2**	**81**

Table 2. COMPARITIVE RESULTS OF ALL ALGORITHMS

4.4 Future scope:

1. CI/CD Pipeline Integration: Enables proactive defect handling, by linking defect prediction systems to CI/CD pipelines for real-time defect potential response during development and in the functional area.
2. Integrating NLP techniques: Using NLP techniques to extract insights from text such as error reports and code statements, to improve error prediction through context.
3. Context Analysis: Incorporating factors such as developer experience and software complexity into error prediction models to increase accuracy.
4. Inclusion of various software artifacts: Extending the system to examine requirements documents, design drawings and test cases along with historical defect information for detailed insights.
5. Active Bug Assignment: Automating bug assignment based on developer expertise and hard work to get it fixed faster.
6. Feedback Loop Implementation: Establish a feedback loop to adjust predictive models based on defect resolution activities, ensuring they can be adapted to the ongoing progress of the project.

V. CONCLUSION

In summary, a software bug prediction system is an important asset for software development teams, enabling them to anticipate and correct potential bugs in their projects by using historical bug data and applying machine learning techniques will be implemented, system bug probability in different software modules or components This proposed algorithm, which can distinguish between models, correlations, and dependencies for forecasting, offers several advantages over traditional methods on, including increased accuracy, efficiency, and resource efficiency. By identifying error-prone areas, development teams can focus their testing and debugging efforts where they are most needed, ultimately leading to increased software quality and reduced defect rates. System design objectives, such as accuracy, efficiency, flexibility, scalability, and user-friendliness, ensure that it can be used in a variety of software development environments to integrate UML diagrams, including use cases, classes, and sequence diagrams including, helping to visualize and understand system design With modules for data collection, preprocessing, feature extraction, model training, prediction generation, and resource allocation, the system provides solutions for fault prediction and and overall implementation for Graphics and reporting to facilitate stakeholder interpretation and benefit from error prediction.

Overall, a well-designed software defect prediction system goes a long way toward increasing software quality, optimizing resource utilization, and streamlining the software development process, enabling teams to deploy software a reliable and more convenient. The proliferation of malicious software poses significant challenges to software quality, affecting both end-users and developers. As software programs and technologies become more complex, manual error detection becomes more difficult and time-consuming. Consequently, automatic fault detection has emerged as a major focus of engineering research in recent years. In our study, we use machine learning and deep learning techniques to address this issue, using seven datasets obtained from the NASA Promise Dataset Repository and comparing them with state-of-the-art machine learning algorithms. Despite impressive progress, there is much room for improvement in this area. New approaches involving complex deep learning algorithms hold promise, and researchers should prioritize expanding data collection efforts to further improve the effectiveness of error prediction algorithms

REFERENCES

1. Han, X.; Heussen, K.; Gehrke, O.; Bindner, H.W.; Kroposki, B. Taxonomy for Evaluation of Distributed Control Strategies for Distributed Energy Resources. *IEEE Trans. Smart Grid* **2018**, *9*, 5185–5195. [CrossRef]
2. Afzal, M.; Huang, Q.; Amin, W.; Umer, K.; Raza, A.; Naeem, M. Blockchain Enabled Distributed Demand Side Management in Community Energy System with Smart Homes. *IEEE Access* **2020**, *8*, 37428–37439. [CrossRef]
3. Jeya Mala, D.; Eswaran, M.; Deepika Malar, N. Intelligent vulnerability analyzer—A novel dynamic vulnerability analysis framework for mobile based online applications. *Commun. Comput. Inf. Sci.* **2018**, 805–823. [CrossRef]
4. Miraoui, M.; El-Etriby, S.; Abid, A.Z.; Tadj, C. Agent-Based Context-Aware Architecture for a Smart Living Room. *Int. J. Smart Home* **2016**, *10*, 39–54. [CrossRef]

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305043 |

5. Wang, Y.; Xu, Y.; Tang, Y. Distributed aggregation control of grid-interactive smart buildings for power system frequency support. *Appl. Energy* **2019**, *251*, 113371. [CrossRef]
6. Cormane, J.; Nascimento, F.A. Spectral Shape Estimation in Data Compression for Smart Grid Monitoring. *IEEE Trans. Smart Grid* **2016**, *7*, 1214–1221. [CrossRef]
7. Maitra, S. Smart Energy meter using Power Factor Meter and Instrument Transformer. *Commun. Appl. Electron.* **2016**, *4*, 31–37. [CrossRef]
8. Rocha, H.R.O.; Honorato, I.H.; Fiorotti, R.; Celeste, W.C.; Silvestre, L.J.; Silva, J.A.L. An artificial intelligence based scheduling algorithm for demand-side energy management in Smart Homes. *Appl. Energy* **2021**, *282*, 116145.
9. Dai, R.; Liu, G.; Wang, Z.; Kan, B.; Yuan, C. A Novel Graph-Based Energy Management System. *IEEE Trans. Smart Grid* **2019**, *11*, 1845–1853. [CrossRef]
10. Chhaya, L.; Sharma, P.; Kumar, A.; Bhagwatikar, G. IoT-Based Implementation of Field Area Network Using Smart Grid Communication Infrastructure. *Smart Cities* **2018**, *1*, 176–189. [CrossRef]
11. Aleksic, S. A Survey on Optical Technologies for IoT, Smart Industry, and Smart Infrastructures. *J. Sens. Actuator Netw.* **2019**, *8*, 47. [CrossRef]
12. Yousif, M. Convergence of IoT, Edge and Cloud Computing for Smart Cities. *IEEE Cloud Comput.* **2018**, *5*, 4–5. [CrossRef]
13. Yaghmaee, M.H.; Leon-Garcia, A.; Moghaddassian, M.; Moghaddam, M.H.Y. On the Performance of Distributed and Cloud-Based Demand Response in Smart Grid. *IEEE Trans. Smart Grid* **2018**, *9*, 5403–5417. [CrossRef]
14. Rahmani, R.; Li, Y. A Scalable Digital Infrastructure for Sustainable Energy Grid Enabled by Distributed Ledger Technology.
J. Ubiquitous Syst. Pervasive Networks **2020**, *12*, 17–24. [CrossRef]
15. Almehizia, A.A.; Al-Masri, H.M.K.; Ehsani, M. Integration of Renewable Energy Sources by Load Shifting and Utilizing Value Storage. *IEEE Trans. Smart Grid* **2019**, *10*, 4974–4984. [CrossRef]
16. Donaldson, D.L.; Jayaweera, D. Effective solar prosumer identification using net smart meter data. *Int. J. Electr. Power Energy Syst.* **2020**, *118*, 105823. [CrossRef]
17. Schultis, D.-L.; Ilo, A.; Schirmer, C. Overall performance evaluation of reactive power control strategies in low voltage grids with high prosumer share. *Electr. Power Syst. Res.* **2019**, *168*, 336–349. [CrossRef]
18. Wesche, J.P.; Dütschke, E. Organisations as electricity agents: Identifying success factors to become a prosumer. *J. Clean. Prod.* **2021**, *315*, 127888. [CrossRef]
19. Shin, M.; Kim, H.; Kim, H.; Jang, H. Building an Interoperability Test System for Electric Vehicle Chargers Based on ISO/IEC 15118 and IEC 61850 Standards. *Appl. Sci.* **2016**, *6*, 165. [CrossRef]
20. Farooq, S.M.; Hussain, S.M.S.; Kiran, S.; Ustun, T.S. Certificate Based Authentication Mechanism for PMU Communication Networks Based on IEC 61850-90-5. *Electronics* **2018**, *7*, 370. [CrossRef]
21. Werthen-Brabants, L.; Dhaene, T.; Deschrijver, D. Uncertainty quantification for appliance recognition in non-intrusive load monitoring using Bayesian deep learning. *Energy Build.* **2022**, *270*, 112282. [CrossRef]
22. Tekler, Z.D.; Low, R.; Zhou, Y.; Yuen, C.; Blessing, L.; Spanos, C. Near-real-time plug load identification using low-frequency power data in office spaces: Experiments and applications. *Appl. Energy* **2020**, *275*, 115391. [CrossRef]
23. Bichiou, Y.; Krarti, M. Optimization of envelope and HVAC systems selection for residential buildings. *Energy Build.* **2011**, *43*, 3373–3382. [CrossRef]
24. Homod, R.Z.; Gaeid, K.S.; Dawood, S.M.; Hatami, A.; Sahari, K.S. Evaluation of energy-saving potential for optimal time response of HVAC control system in smart buildings. *Appl. Energy* **2020**, *271*, 115255. [CrossRef]
25. Balaji, B.; Xu, J.; Nwokafor, A.; Gupta, R.; Agarwal, Y. Using WiFi connection counts and camera-based occupancy counts to estimate and predict building occupancy. In Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems, Roma, Italy, 11–15 November 2022; Volume 257, p. 111759.
26. Tekler, Z.; Low, R.; Yuen, C.; Blessing, L. Plug-Mate: An IoT-based occupancy-driven plug load management system in smart buildings. *Build. Environ.* **2022**, *223*, 109472. [CrossRef]
27. Zou, H.; Zhou, Y.; Jiang, H.; Chien, S.-C.; Xie, L.; Spanos, C.J. WinLight: A WiFi-based occupancy-driven lighting control system for smart building. *Energy Build.* **2018**, *158*, 924–938. [CrossRef]
28. Alfita, R.; Joni, K.; Darmawan, F.D. Design of Monitoring Battery Solar Power Plant and Load Control System based Internet of Things. *TEKNIK* **2021**, *42*, 35–44. [CrossRef]
29. Dziadak, B. Structural Health Monitoring System for Snow and Wind Load Measurement. *Electronics* **2020**, *9*, 609. [CrossRef]
30. Wilhelm, S.; Kasbauer, J. Exploiting Smart Meter Power Consumption Measurements for Human Activity Recogni-tion (HAR) with a Motif-Detection-Based Non-Intrusive Load Monitoring (NILM) Approach. *Sensors* **2021**, *21*, 8036. [CrossRef] [PubMed]

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305043 |

31. Souza, W.; Marafão, F.; Liberado, E.; Simões, M.; Da Silva, L. A NILM Dataset for Cognitive Meters Based on Con-servative Power Theory and Pattern Recognition Techniques. *J. Control. Autom. Electr. Syst.* **2018**, *29*, 742–755. [CrossRef]
32. Jiang, W.; Tang, L.; Wei, X. Home energy efficiency evaluation based on NILM. *Procedia Comput. Sci.* **2021**, *183*, 53–60. [CrossRef]
33. Abubakar, I.; Khalid, S.; Mustafa, M.; Shareef, H.; Mustapha, M. Application of load monitoring in appliances' energy management—A review. *Renew. Sustain. Energy Rev.* **2017**, *67*, 235–245. [CrossRef]

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details