International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2025: 7.089

Machine Learning-Driven PdMin Software Engineering: Models, Methods, and Applications

Dr. Ashish Jolly

Associate Professor, Department of Computer Science, Govt. PG College, Ambala Cantt
Email: *ashishjolly76[at]gmail.com*

Abstract: ***Machine Learning (ML) has been increasingly applied in the domain of Software Engineering (SE) for improving Predictive Maintenance (PdM) approaches. In this paper, we discuss ML-based PdMin software systems, with an aim to develop models, methodologies, and applied practices. PdM approach, which was initially used in manufacturing and industrial system applications, has great potential in the software domain to predict software failures, performance, and system vulnerabilities before potential failures happen. This paper provides an overview of various ML algorithms, that is, supervised, unsupervised, and reinforcement learning, used for software fault diagnosis, anomaly detection, and system performance optimization. Moreover, we address the problem and discuss the emerging problems, including the lack of data, features, and interpretation of the models. Applying the ML-based PdM methods to several real-world scenarios, this paper illustrates the potential of the ML-driven PdM for the downtime and reliability of software systems, and the operational costs of the software industry. Finally, the paper outlines future work on how to extend the models and optimize their scalability and applicability in a real-world software system.***

Keywords: ML, predictive maintenance, SE, performance, optimization, software failure.

1. Introduction

Predictive Maintenance (PdM) has been broadly used in various industries, especially to hardware systems, which mitigates the risk of unexpected failures and enhances utilization of resources.[1] In hardware, it uses sensor data and predictive analytics to predict and prevent machines, automobiles, and industrial equipment from breaking, thus saving those companies time, resources, and dollars [2][3]. For all its potential to change the way that software systems are maintained PdM is still a relatively under-researched area within SE. With ever more complex systems of software and the relevance of such with best possible availability proactive maintenance is nowadays even more crucial [4][5].

Software failure is more elusive than hardware failure, because software failure has nothing to do with wear and tear [6][7]. Code changes, system design, network dynamics are some of their complex interactions [8]. Today, mission-critical applications are no longer simply audited and intervened post-failure to keep software systems growing, diversified, and running. Computer systems must be taken care of to be dependable, efficient, and sustainable for a long time [9].

As a subset of Artificial Intelligence (AI), ML is a promising approach to predictive maintenance in SE. Artificial intelligence models could study system performance and predict software failures with huge amounts of historical data, real-time monitoring, and pattern detection techniques [10]. And with this skill, software developers can be proactive in detecting potential problems before they occur, minimizing risk, and ensuring systems are performing optimally. ML models can predict system crashes and identify root causes such as resource usage, buggy code, and system bottlenecks to enhance software quality and security [11].

PdM using ML brings many software-system improvements. ML models can predict the failures of software based on system log, error report, performance metric, user interaction, etc. Software developers can be notified early when their code may have introduced an error so they can address it before it disrupts the system or user. PdM models can optimise resource allocation, change system configurations on the fly, and enhance security by detecting strange activity or vulnerabilities before they have been used [12].

Despite the potential, the actually implementation of ML-based PdM at scale is challenging. Improved-quality labeled data is key for training effective ML models, yet the quality and accessibility of data is a significant challenge [13]. Many software systems have difficulty getting the high-quality training data they need. Like many sophisticated ML models, including Deep Learning (DL) algorithms, they are "black-boxes" in which engineers often have difficulty in trusting predictions and understanding the rationale. Generalizing these techniques to large, distributed systems presents challenges related to computational cost, system requirements, and real-time performance [14].

This work advances the domain of ML software engineering applications by showing how to use PdM to enhance the stability, performance, and efficiency of software systems. We present this paper to discuss PdM techniques and their applications and challenges with respect to SE, to encourage engineers and researchers to bring proactive and data-driven maintenance strategies in practice that could enable software systems to last longer and decrease the rate of their system failures while being used in a more complex and available manner.

2. Literature Review

PdM is increasingly popular in SE, where companies attempt to avoid software failures, increase system performance, or simply ensure durability of the software. Historically, software maintenance has been reactive to remedy the faults. Software engineers can predict failure, performance, or security risks through the help of ML, because of the PdM solutions. Research by Menzies et al. (2017) demonstrates that ML models such as decision trees and ensemble methods such as random forests can predict software failures from historical failure data and logs of the systems [15]. These models are able to detect abnormal system behavior which may indicate problems. Zhang et al. (2019) demonstrated that autoencoders can be trained to detect system anomalies in software which could lead to failure [16]. In the study by Zimmermann, T., et al. (2009), techniques have the relationship between the degradation and failure, whereas reinforcement learning (RL) has achieved a new maintenance schedule and resource allocation [17]. editorial much of the relationship between failure and degradation could learning (RL). RL techniques allow maintenance procedures to be optimised in accordance with the system performance with continuous feedback, reducing downtime and increasing uptime. Software PdM is an evolving alternative for this approach, but it holds the potential to adapt to variation in system condition.

Notwithstanding all the progress in PdM in ML, here are some of the challenges that are facing in real life. In paper Hosseini, S., et al. (2018), quality and accessibility of the data are real concerns. Large and complex software systems do not yield enough high-quality, labelled data for the creation of accurate ML models in many organisations [18]. The interpretability of ML model is also very important. It's harder to get a sense of why predictions are made as models get more complex – particularly DL models, which can erode engineers' trust in systems. Recent developments of Explainable AI (XAI) can explain the model outputs, help show its features and (in some sense) justify directly or indirectly the output of the applied model, which ultimately increase transparency and user confidence. Live monitoring and prediction are necessary for IoT applications, so this link is useful. Work for the future are under way on hybrid models that would use ML methods to enhance forecasting. Such composite methods may provide better predictions in complex and dynamic software environments, by using a combination of supervised, unsupervised, and reinforcement models. And Auto ML (Automated ML) holds: the promise to simplify model selection, training, and deployment so that PdM is within reach for engineers without ML expertise. SE-style PdM can be used in any industry to have the same.

3. ML Models for PdM in Software

In the domain of ML-Enabled PdM in SE, different ML models are used to predict possible failures, to increase system performance, and to ensure the reliability of software systems [19]. These methods rely on past and up-to-date data (e.g., system logs, code changes, performance counters, error reports) in order to anticipate issues and to act proactively [20]. Table 1 presents some ML algorithms which are used in software for predictive maintenance.

Table 1: ML algorithms with applications in SE

Model Type	Description	Applications in Software
Supervised Learning Models	Supervised learning algorithms are applied when training data with the desired outcome are known. These predictive models are trained on past data to forecast future events [21].	Failure predictability determination when a failure of a software component will occur based upon past failures and system behaviour. Also, recognize performance degradation [22].
Decision Trees (DT)	A tree model that divides the data into other nodes, or branches, using decision points. Decision is represented by individual branch [23].	Distinguishing system behavior (failure or non-failure behavior) such as memory usage, response times, and CPU usage based on attributes. Also, simple to understand and applicable to PdM [24].
Random Forest (RF)	An ensemble approach in which many decision trees to be created to be more accurate and less over-fit [25].	Utilized for classification problems, such as for predicting risk of software failure. It achieves excellent results for a high number of input features and data complexity [26].
Support Vector Machines (SVM)	SVM makes data classification by forming the best separated hyperplane between classes. It works well in high dimensions [27].	Applicable in binary classification scenarios such as failure prediction (failure vs. no failure). Work well on small to medium datasets [27].
Neural Networks (NN)	Neural networks, such as DL models, have power to model complex non-linear relationships among features [28].	Anomaly detection based on complex log, code, and system metric patterns. For finding subtle non-linear patterns in data [28].
Unsupervised Learning Models	Unsupervised learning is applied when data are not labeled. These models then find patterns or anomalies that could indicate a problem or failure [29].	Anomaly Detection: detection of abnormal system behavior that occur as precursor to failures. Performance Tuning: Identifying what could be better optimized or altered from expected system behavior [29].
K-Means Clustering	An algorithm for unsupervised learning from data grouped into K similar clusters. It can learn patterns without labels [30].	Clustering together similar system states or behaviors, which might mean normal operation or malfunction. Helpful in classification of various kinds of failures [30].
Autoencoders	A class of neural network that learns to encode data in a lower-dimensional space and reconstruct the input. Anomalies are	Anomaly Detection: Recognition of deviations in system activities, e.g., odd log lines or out-of-place resource

	detected by comparing the reconstructed data to the original [31].	usages. \item Fault diagnosis before a complete failure happens [31].
Reinforcement Learning (RL)	Reinforcement learning learns optimal actions by trying and failing, being rewarded for successful attempts and penalized for unsuccessful ones [32].	Scheduling of maintenance: Scheduling maintenance in time and method as per system health. How to understand the best practices for software optimization and preventing failures [32].
DL Models (e.g., CNN, LSTM)	DL models, like CNN and LSTM, are good at dealing with high-dimensional, continuous data [33].	Time Series Data Analysis - analysis logs or performance metrics which are changing over time to predict eventual problems. For analysis of system log sequences to identify possible failure trends LSTM can be employed [33].

4. Methods for Implementing PdMin SE

Applying ml-based PdM for SE requires multiple steps shown in Figure 1 from collecting and preprocessing data to feature engineering and choosing a model followed by training/evaluation/realtime monitoring [34].

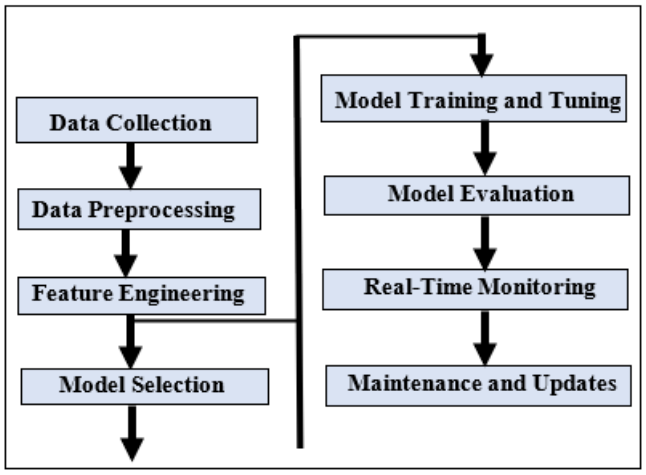

Figure 1: Implementation of PdMin SE

There are a variety of important steps from collecting of data to ongoing model updates in place to embed ML-based PdM in SE. First-step is gathering relevant information from logs, performance monitoring, error reports and code changes. These data can be used to identify issues before they breach the systems and for model training[35]. Secondly, feature engineering is necessary for selecting and transforming the data into actionable features that can improve model's performance. That might mean adding a feature based on time series (or system) measurements e.g. trends/ moving averages. The next step is model selection, i.e., choosing appropriate ML models according to the data qualities and the nature of issues to resolve. A wide variety of unsupervised and supervised learning models are commonly utilized in PdM [36].

After being selected, the model gets either trained or tuned; it is being trained by historical data, and the hyperparameter is the one setting to make it perform better. After training, the model's prediction accuracy is assessed based on various scores such as accuracy, precision, recall, F1-score etc [37]. The next step is real-time monitoring, during which the learned model is used to constantly assess the data coming from the system in real-time, predicting eventual failures and finding performance bottlenecks before they affect operations. And the soul of the model must be updated on an ongoing basis to keep pace with ongoing changes to ensure its accuracy and continued relevance. This involves periodically retraining the model with new data, improving the accuracy of its predictions, and updating it to accommodate changes in the infrastructure or behaviour of the system [38]. With these methods, companies can effectively leverage ML powered PdM to reduce system downtime, improve the economy of scale, and make software systems more reliable in general.

5. Applications of ML-Driven PdM in Software Engineering

In SE there are a number of uses for ML-based PdM(PdM) as it provides proactive solutions to potential issues and mitigates problems that might impact system performance down the line [39]. The key areas ML models are being applied to enhance the performance, security, fault tolerance of software systems include:

Failure prediction: The most famous use of PdMin SE is probably the prediction of likely software failures. Based on error logs, system performance metrics and historical failure data, ML models can predict when specific parts or components are likely to fail. This helps software developers to act before the issue occurs, such as changing the code, more testing, or patches. It provides for predicting failure and, therefore, improves reliability of the system and should minimize downtime.

Performance Improvement: Software systems can also be improved with ML models, which can optimize their performance. Patterns of resource usage: ML-based algorithms are able to identify bottlenecks or weak areas where resources have become underutilized over time, for example CPU, memory, disc and network usages. An ML model, for example, could predict when certain application modules might come under heavy use and engineers could proactively reallocate resources, change configurations, or even refactor code to get more performance. This contributes to better resource utilisation and thus improved performance of the software.

Anomaly Detection: To discover strange system behavior that can signify underlying problems you need anomaly detection. ML models, in particular unsupervised methods such as clustering and autoencoders, can process large volumes of data to uncover abnormal patterns that do not match regular system activity. Abnormal read/write behaviour in your source code

and sudden surges in memory usage, for example, could indicate a potential issue such as a memory leak or security vulnerability. Prompt risk reduction and response can be implemented by early recognition of these anomalies [40].

Security and Risk/Potential Detection: Where the software system becomes increasingly vulnerable to cyber-attacks, PdM with ML can be able to help security to decrease risks. Through the analysis of code changes, network traffic attributes, and system access logs, ML models could be leveraged for security vulnerability detection. ML can be employed to predict and prevent potential security breaches by identifying unusual or unauthorized behavior, for example suspicious login attempts or questionable API calls. Especially in highly sensitive environments, this is a necessity for the secrecy and integrity of software systems.

Software Refactoring and Code Quality: PdM with ML is also critical to reduce technical debt and improving code quality. ML models could analyze the complexity of code modules and identify places- such as strongly connected modules or high cyclomatic complexity functions- that are likely to cause issues in the future. ML can let developers know which areas of their code might need refactorings or more optimisation, by identifying potential danger zones. In the long run, this leads to better program quality and maintainability.

Incident Response and Troubleshooting: With issue resolution time-reduced, ML models can assist by identifying the source of problems which they are doing so. ML algorithms could quickly analyze logs, error messages or records of system state to find out why a system failed or why its performance waned. This can reduce the time spent troubleshooting by orders of magnitude, meaning less time to resolve those issues for engineers and, ideally, fewer problems resulting in a negative impact or customer experience for end users [41].

Scheduling maintenance: Predicative maintenance will helps also planning patches and upgrades for software systems. The condition and likelihood of the system failure can be predicted by ML models that can tell you when the system is most vulnerable, and in turn propose best times for updates or downtime. This raises the overall stability of the system by ensuring that essential maintenance tasks, such as patching vulnerabilities or applying software updates, are performed prior to problems occurring.

Enhance End-User Experience: PdM also serves to enhance the end-user experience by predicting any outages or errors that affect customers while using equipment through ML-powered PdM. An ML model, for example, may predict when a server is expected to become full or when latency will increase, both of which might affect user interactions [41]. Also, the system can preventatively perform load balancing, resource scaling, traffic rerouting, or other such operations that would ensure an end user does not suffer any adverse effects as a result of such a scenario happening.

6. Challenges and Future Scope in ML-Driven PdM in Software Engineering

Despite the promising results of the PdM (or ML-based) technology, in enhancing the security, performance, and dependability of software systems, the adoption of PdM in SE is not free from challenges. And as it progresses the area provides a plethora or opportunities for further research and discoveries [42]. Main limitations, barriers and future work directions for the application of ML in the predictive maintenance of SE are presented.

6.1. Data Quality and Availability: The ML models are dependent on the high-quality, the most comprehensive volumes of data for training. Yet many software systems lack abundant labeled data. Data are often sparse, incomplete, or noisy, which might not be sufficient to form accurate predictive models. For a lot of systems, error log and performance data are mostly noisy, unstructured or fragmented and it is hard to exploit the useful features from them. Predictions that are based on low-quality data may be inaccurate and unsuccessful to calculate PdMprobabilities, thereby lessening the accuracy of PdManalysis [43].

6.2. Model Interpretability and Transparency: For a good number of state-of-the-art ML models, deep neural networks included, the models are “black boxes”-- and as such, their decision-making process is not easily understandable. Software engineers might be temperamental to trust and deploy a model if its working cannot be completely understood or explained which will limit the model’s acceptance for practical maintenance tasks [41].

6.3. Scalability: Real-time software systems can produce huge data (big data) that is expensive to manipulate at scale. Conventional ML models may not be sufficient to process the large amounts and fast-paced information found in today's large software systems. Furthermore, managing and delivering models at scale and at the edge can be difficult and resource-intensive. In practice – as much as small resource underfinanced systems may, can't scale PdM solutions effectively with ease: you can of course doubt the kids health, and say there are no kids, but assuming otherwise, it will grow [43].

6.4. Data Privacy and Security Concerns: PdMmay need to access sensitive information, like system logs, user behavior, and performance, etc. For a sector sensitive to security and privacy concerns (e.g., finance, healthcare, or government), the potential risk of data leakages or unauthorized access is very strong when dealing with sensitive information. Privacy concerns around data and security concerns may preclude the adoption of ML-based PdM or inhibit the data sharing required for training models [44].

6.5. Dynamic and Evolving Software Environments: Software architecture is dynamic, it continuously changes due to software updates, patches and new releases. These changes can have a drastic impact on the system that could make existing PdMmodels useless. Facility managers

must be sure to monitor and retrain models as the software itself changes. Lose of effectiveness of PdMmodels with a possible decrease in prediction accuracy and reliability if not up-to-date [44].

6.6. High Computational Costs: Some ML models, especially DL models have high computational requirement for training and inference. This may be challenging for companies with constrained resources or when integrating PdM capabilities into realtime systems. The large computational cost could make it difficult to implement advanced prediction maintenance on-site, especially for those applications running in resource-constrained condition. The future of ML-based PdM in SE There are a number of promising directions for the future of ML-based PdM in SE that can help whether the limitations we observe and extend the usefulness of predictive models [43].

Learn supersedes model explainability and explainable AI (XAI) in importance. Software engineers and developers can gain confidence in their findings by learning what's going on in the "brain" behind ML results using such techniques as feature importance, decision trees, and model-agnostic explanations. Engineers will simply make better decisions because they will better understand how to use ML in new and different ways in PdM [45].

Another good track is the track of Federated Learning for Data Privacy. Federated learning offers a solution to the increasing data privacy issues in this respect since it would allow for local training of ML models within decentralized devices or systems without letting sensitive data leave the device. It is possible to build predictive models with strong data privacy and confidentiality guarantees as only the model updates are sent to a central server. This approach could help make it easier to deploy PdM solutions in data privacy sensitive sectors such as healthcare and finance [45].

One of the indispensable future ways for improving the performance of PdM systems is the utilization of edge computing, real-time data processing and integration of both in the system. ML models are able to produce predictions and make decisions directly at the data source by running on edge devices without the need to transmit large amounts of data to centralised servers [46]. This is ideal for real-time PdM systems, which are distributed, particularly in Internet of Things (IoT) devices as well in mobile applications, since it can reduce reaction time and computational costs substantially.

Hybrid models offer potential as an option to improve the performance of PdM system in order to get more reliable predictions. Hybrid models can benefit from the strengths of several algorithms by blending different ML approaches – including supervised vs unsupervised learning or DL vs classical statistical methods. These models can indeed provide more reliable and more accurate prediction, especially when dealing with complex or heterogeneous datasets, which could include both labelled and unlabelled data.

Finally, maintaining the accuracy of PdMmodels long-term is a process that includes monitoring a model for deviation between predictions and actual performance over time, using new performance data to refine the model. The PdM models need to have adaptive learning and real-time performance monitoring because software systems are continuously evolving. This will ensure predictions remain accurate and relevant to changing conditions by allowing models to automatically retrain as new data is introduced, or when system behaviour changes. Continued Adaptation: By constantly updating predictive maintenance models to reflect the evolving needs of the software system, continued adaptation offers long-term gains.

By addressing the current limitations and expanding the horizons for ML-based PdMin SE, these future works offer a potential for more usable, effective, and adaptable PdM solutions in various software systems.

7. Conclusion

ML-based PdMis regarded as a revolutionary solution for software development, with the potential of handling software failure adversities, ensuring performance gain, and system reliability improvement. This paper has demonstrated that by experimenting with various ML models like supervised/unsupervised/ reinforcement learning, it is definitely possible to utilize these for predicting software failures in order to avoid issues before reaching the end users. However, despite the many benefits, there are several roadblocks, including data quality, model interpretability, and how difficult it can be to incorporate ML models into existing software, both internally and with third parties. Nevertheless, the best practices and use cases surveyed give strong evidence of the benefits of ML-based PdMin terms of downtime reduction, enhanced software quality, reduction of maintenance costs, and so on. Prospects for the future development of ML-solutions must be built on more sophisticated ML-algorithms as well as improved data quality at data collection and enhanced completed models for model transparency to improve the effectiveness and scalability of PdMin SE. While such trends will increasingly push software maintenance towards automation, they will also provide a foundation for more resilient, context-aware, and effective systems in the future.

References

[1] Lee, J., Ni, J., Singh, J., Jiang, B., Azamfar, M., & Feng, J. (2020). Intelligent maintenance systems and predictive manufacturing. *Journal of Manufacturing Science and Engineering*, *142*(11), 110805.

[2] Oye, E., & Owen, A. (2025). Leveraging Cloud Computing for PdMin IoT Devices.

[3] Zhu, T., Ran, Y., Zhou, X., & Wen, Y. (2019). A survey of predictive maintenance: Systems, purposes and approaches. *arXiv preprint arXiv:1912.07383*.

[4] Maktoubian, J., Taskhiri, M. S., & Turner, P. (2021). Intelligent PdM(Ipdm) in forestry: A review of challenges and opportunities. *Forests*, *12*(11), 1495.

[5] Abd Wahab, N. H., Hasikin, K., Lai, K. W., Xia, K., Bei, L., Huang, K., & Wu, X. (2024). Systematic review of PdMand digital twin technologies challenges, opportunities, and best practices. *PeerJ Computer Science*, *10*, e1943.

[6] Hafsi, M., & Agoun, J. (2024, December). Strategic PdMfor Internet System Security and Risk Management: A Roadmap. In *8th International Conference on Future Networks and Distributed Systems*.

[7] Zhang, W., Yang, D., & Wang, H. (2019). Data-driven methods for PdMof industrial equipment: A survey. *IEEE systems journal*, *13*(3), 2213-2227.

[8] Zhu, T., Ran, Y., Zhou, X., & Wen, Y. (2025). A Survey on Intelligent PdM(IPdM) in the Era of Fully Connected Intelligence. *IEEE Communications Surveys & Tutorials*.

[9] Khan, U., Cheng, D., Setti, F., Fummi, F., Cristani, M., & Capogrosso, L. (2025). A Comprehensive Survey on DL-based Predictive Maintenance. *ACM Transactions on Embedded Computing Systems*.

[10] Carvalho, T. P., Soares, F. A., Vita, R., Francisco, R. D. P., Basto, J. P., & Alcalá, S. G. (2019). A systematic literature review of ML methods applied to predictive maintenance. *Computers & Industrial Engineering*, *137*, 106024.

[11] Cakir, M., Guvenc, M. A., & Mistikoglu, S. (2021). The experimental application of popular ML algorithms on PdMand the design of IIoT based condition monitoring system. *Computers & Industrial Engineering*, *151*, 106948.

[12] Alsolai, H., & Roper, M. (2020). A systematic literature review of ML techniques for software maintainability prediction. *Information and Software Technology*, *119*, 106214.

[13] Paolanti, M., Romeo, L., Felicetti, A., Mancini, A., Frontoni, E., & Loncarski, J. (2018, July). ML approach for PdMin industry 4.0. In *2018 14th IEEE/ASME international conference on mechatronic and embedded systems and applications (MESA)* (pp. 1-6). IEEE.

[14] Dalzochio, J., Kunst, R., Pignaton, E., Binotto, A., Sanyal, S., Favilla, J., & Barbosa, J. (2020). ML and reasoning for PdMin Industry 4.0: Current status and challenges. *Computers in Industry*, *123*, 103298.

[15] Menzies, T., et al. (2017). Predicting software maintenance problems using ML: A systematic review. *Journal of Software Maintenance and Evolution*, 29(2), 134-155.

[16] Zhang, J., et al. (2019). Anomaly detection in software logs using ML. *Journal of SE Research and Development*, 7(1), 1-16.

[17] Zimmermann, T., et al. (2009). Mining software repositories for predictive models of software maintenance. *IEEE Transactions on SE*, 35(3), 289-299.

[18] Hosseini, S., et al. (2018). ML for software defect prediction: A review and future directions. *ACM Computing Surveys*, 51(3), 1-34.

[19] Paolanti, M., Romeo, L., Felicetti, A., Mancini, A., Frontoni, E., & Loncarski, J. (2018, July). ML approach for PdMin industry 4.0. In *2018 14th IEEE/ASME international conference on mechatronic and embedded systems and applications (MESA)* (pp. 1-6). IEEE.

[20] Liulys, K. (2019, April). ML application in predictive maintenance. In *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)* (pp. 1-4). IEEE.

[21] Zhang, D., & Tsai, J. J. (2003). ML and SE. *Software Quality Journal*, *11*, 87-119.

[22] Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., ... & Zimmermann, T. (2019, May). SE for ML: A case study. In *2019 IEEE/ACM 41st International Conference on SE: SE in Practice (ICSE-SEIP)* (pp. 291-300). IEEE.

[23] Naidu, M. S., & Geethanjali, N. (2013). Classification of defects in software using decision tree algorithm. *International Journal of Engineering Science and Technology*, *5*(6), 1332.

[24] Singh, J., Singh, K., & Singh, J. (2019). Reengineering framework for open source software using decision tree approach. *International Journal of electrical and computer engineering (IJECE)*, *9*(3), 2041-2048.

[25] Petkovic, D., Sosnick-Pérez, M., Okada, K., Todtenhoefer, R., Huang, S., Miglani, N., & Vigil, A. (2016, October). Using the random forest classifier to assess and predict student learning of SE teamwork. In *2016 IEEE frontiers in education conference (FIE)* (pp. 1-7). IEEE.

[26] Mustapha, H., & Abdelwahed, N. (2019). Investigating the use of random forest in software effort estimation. *Procedia computer science*, *148*, 343-352.

[27] Salcedo-Sanz, S., Rojo-Álvarez, J. L., Martínez-Ramón, M., & Camps-Valls, G. (2014). Support vector machines in engineering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *4*(3), 234-267.

[28] Kumar, S., Krishna, B. A., & Satsangi, P. S. (1994). Fuzzy systems and neural networks in SE project management. *Applied Intelligence*, *4*, 31-52.

[29] Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., ... & Zimmermann, T. (2019, May). SE for ML: A case study. In *2019 IEEE/ACM 41st International Conference on SE: SE in Practice (ICSE-SEIP)* (pp. 291-300). IEEE.

[30] Malviya, A. K., & Yadav, V. K. (2012, September). Maintenance activities in object oriented software systems using K-means clustering technique: A review. In *2012 CSI Sixth International Conference on SE (CONSEG)* (pp. 1-5). IEEE.

[31] Czibula, G., Lupea, M., & Briciu, A. (2022). Enhancing the performance of software authorship attribution using an ensemble of deep autoencoders. *Mathematics*, *10*(15), 2572.

[32] Abo-eleneen, A., Palliyali, A., & Catal, C. (2023). The role of Reinforcement Learning in software testing. *Information and Software Technology*, *164*, 107325.

[33] Akhtar, M. S., & Feng, T. (2022). Detection of malware by DL as CNN-LSTM ML techniques in real time. *Symmetry*, *14*(11), 2308.

[34] Goswami, M., & Bhatt, A. (2022). Study on Implementing AI for PdMin Software Releases. *International Journal of Research Radicals in Multidisciplinary Fields, ISSN*, 93-99.
[35] Selcuk, S. (2017). Predictive maintenance, its implementation and latest trends. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *231*(9), 1670-1679.
[36] Kumar, B. (2022). AI Implementation for PdMin Software Releases. *International Journal of Research and Review Techniques*, *1*(1), 37-42.
[37] Carvalho, T. P., Soares, F. A., Vita, R., Francisco, R. D. P., Basto, J. P., & Alcalá, S. G. (2019). A systematic literature review of ML methods applied to predictive maintenance. *Computers & Industrial Engineering*, *137*, 106024.
[38] Hashemian, H. M. (2010). State-of-the-art PdMtechniques. *IEEE Transactions on Instrumentation and measurement*, *60*(1), 226-236.
[39] Goswami, M., & Bhatt, A. (2022). Study on Implementing AI for PdMin Software Releases. *International Journal of Research Radicals in Multidisciplinary Fields, ISSN*, 93-99.
[40] Selcuk, S. (2017). Predictive maintenance, its implementation and latest trends. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *231*(9), 1670-1679.
[41] Carvalho, T. P., Soares, F. A., Vita, R., Francisco, R. D. P., Basto, J. P., & Alcalá, S. G. (2019). A systematic literature review of ML methods applied to predictive maintenance. *Computers & Industrial Engineering*, *137*, 106024.
[42] Strielkowski, W., Vlasov, A., Selivanov, K., Muraviev, K., & Shakhnov, V. (2023). Prospects and challenges of the ML and data-driven methods for the predictive analysis of power systems: A review. *Energies*, *16*(10), 4025.
[43] Li, Z., He, Q., & Li, J. (2024). A survey of DL-driven architecture for predictive maintenance. *Engineering applications of artificial intelligence*, *133*, 108285.
[44] Khan, U., Cheng, D., Setti, F., Fummi, F., Cristani, M., & Capogrosso, L. (2025). A Comprehensive Survey on DL-based Predictive Maintenance. *ACM Transactions on Embedded Computing Systems*.
[45] Manta-Costa, A., Araújo, S. O., Peres, R. S., & Barata, J. (2024). ML Applications in Manufacturing-Challenges, Trends, and Future Directions. *IEEE Open Journal of the Industrial Electronics Society*.
[46] Bai, J., Wu, D., Shelley, T., Schubel, P., Twine, D., Russell, J., ... & Zhang, J. (2024). A comprehensive survey on ML driven material defect detection: Challenges, solutions, and future prospects. *arXiv preprint arXiv:2406.07880*.