



Research Digest on Engineering Management and Social Innovations

Available on: <http://rdems.in>

DOI: 10.46647/rdems0204031

Volume: 02 Issue: 04, April, 2026

p-ISSN: 3117-4418

e-ISSN: 3117-4426

Machine Learning-Based Prediction Of Software Behavior

¹Dr.K.Pavan Kumar, ²Boya Naveen

¹Associate Professor, Dr.K.V.Subba Reddy Institute of Technology

²MCA Student, Master of Computer Applications, Dr.K.V.Subba Reddy Institute of Technology

ABSTRACT

The increasing complexity of modern software systems has made it challenging to anticipate system behavior under varying operational conditions. Predictive modeling using machine learning offers a powerful approach to analyze historical execution data and forecast software behavior, enabling proactive decision-making and improved system reliability. This study presents a comprehensive framework for predicting software behavior by leveraging supervised and unsupervised machine learning techniques on software metrics such as execution logs, resource utilization, and user interaction patterns. The proposed approach involves data preprocessing, feature extraction, and the application of models including regression algorithms, decision trees, and ensemble methods to identify patterns and predict outcomes such as performance degradation, fault occurrence, and system anomalies. The model is trained and validated using real-world datasets to ensure accuracy and generalizability. Additionally, techniques for handling imbalanced data and improving model interpretability are incorporated to enhance practical applicability. Experimental results demonstrate that machine learning-based predictive models can significantly improve the accuracy of software behavior prediction compared to traditional rule-based approaches. The findings highlight the potential of predictive analytics in optimizing software maintenance, reducing downtime, and enhancing overall system performance. This research contributes to the development of intelligent, self-adaptive software systems capable of anticipating and responding to dynamic environments.

Keywords: Machine Learning, Software Behavior Prediction, Predictive Modeling, Anomaly Detection, Software Analytics, Log Analysis, Feature Engineering, Classification Algorithms, System Monitoring, Fault Detection.

I. INTRODUCTION

In recent years, the rapid evolution of software systems has led to increased complexity, scalability demands, and dynamic user interactions. Modern applications, ranging from cloud-based platforms to embedded systems, operate in highly variable environments where predicting system behavior is increasingly difficult. Traditional software engineering approaches rely heavily on static analysis and predefined rules, which often fail to capture the intricate and evolving patterns of software execution. As a result, there is a growing need for intelligent techniques that can analyze past behavior and accurately predict future system performance and anomalies.

Predictive modeling using machine learning has emerged as a promising solution to address these challenges. By leveraging historical data such as execution logs, system metrics, and user activity, machine learning models can identify hidden patterns and relationships that are not easily detectable through conventional methods. These models enable the prediction of key software behaviors, including performance bottlenecks, failure occurrences, resource utilization trends, and anomalous activities. This predictive capability allows developers and system administrators to take proactive measures, thereby improving system reliability and efficiency.

The integration of machine learning into software behavior analysis involves several stages, including data collection, preprocessing, feature engineering, model training, and evaluation. Various algorithms such as linear regression, decision trees, support vector machines, and ensemble



Research Digest on Engineering Management and Social Innovations

Available on: <http://rdems.in>

DOI: 10.46647/rdems0204031

Volume: 02 Issue: 04, April, 2026

p-ISSN: 3117-4418

e-ISSN: 3117-4426

techniques are commonly employed depending on the nature of the problem. Additionally, advancements in deep learning have further enhanced the ability to model complex, non-linear relationships in software systems. These approaches not only improve prediction accuracy but also support adaptive and self-learning systems.

Furthermore, predictive modeling plays a crucial role in areas such as fault prediction, performance optimization, and automated maintenance. By anticipating potential issues before they occur, organizations can reduce system downtime, lower maintenance costs, and enhance user experience. As software systems continue to grow in complexity, the adoption of machine learning-based predictive techniques becomes essential for building robust, scalable, and intelligent software solutions. This research focuses on exploring such methodologies and demonstrating their effectiveness in predicting software behavior in real-world scenarios.

II. LITERATURE SURVEY

1. Title: Predictive Modeling of Software Behavior Using Machine Learning

Authors: C. Saksupawattanakul, W. Vatanawood

Abstract:

This study focuses on predicting software behavior to improve system reliability and verification processes. The authors propose a Software Behavior Prediction (SBP) model that utilizes Multi-Label Classification (MLC) and Long Short-Term Memory (LSTM) networks to capture complex execution patterns. The model addresses challenges such as concurrency and resource constraints in software systems. Experimental results show that the proposed approach improves prediction accuracy and efficiency compared to traditional methods, enabling better identification of software states and behaviors.

2. Title: Predictive Models in Software Engineering: Challenges and Opportunities

Authors: Yanming Yang, Xin Xia, David Lo, et al.

Abstract:

This paper provides a comprehensive survey of predictive models in software engineering, analyzing numerous studies across domains such as software testing, debugging, and maintenance. The authors classify different predictive modeling approaches and highlight their applications in improving software quality. The study also identifies key challenges, including data quality issues, model interpretability, and scalability, while proposing future research directions for enhancing predictive modeling techniques in software systems.

3. Title: Predictive Performance Modeling for Distributed Computing using Machine Learning

Authors: Carl Witt, Marc Bux, Wladislaw Gusew, Ulf Leser

Abstract:

This research explores predictive performance modeling techniques for distributed computing environments. It emphasizes the use of machine learning to estimate performance metrics such as execution time, memory usage, and resource allocation. The study highlights the importance of historical data and black-box monitoring for building accurate predictive models. It also identifies challenges related to variability in workloads and proposes methods to improve prediction accuracy in large-scale systems.

4. Title: BehavDT: A Behavioral Decision Tree Learning Model

Authors: Iqbal H. Sarker, Alan Colman, Jun Han, et al.

Abstract:

This paper introduces a behavioral decision tree (BehavDT) model designed to predict user behavior in context-aware systems. Unlike traditional decision trees, the proposed model incorporates generalized and context-specific decision-making to improve prediction accuracy. The



Research Digest on Engineering Management and Social Innovations

Available on: <http://rdems.in>

DOI: 10.46647/rdems0204031

Volume: 02 Issue: 04, April, 2026

p-ISSN: 3117-4418

e-ISSN: 3117-4426

study demonstrates that the BehavDT model effectively captures user behavior patterns and outperforms conventional machine learning techniques in real-world scenarios involving multi-dimensional data.

5. Title: An Empirical Analysis of Feature Engineering for Predictive Modeling

Authors: Jeff Heaton

Abstract:

This research investigates the role of feature engineering in improving the performance of machine learning models. The study evaluates how different types of engineered features impact various algorithms such as neural networks, decision trees, and ensemble models. Results indicate that proper feature selection and transformation significantly enhance predictive accuracy, emphasizing the importance of preprocessing in predictive modeling tasks.

6. Title: Predictive Modeling and Decision Support using Machine Learning

Authors: A. Bojanowska

Abstract:

This paper presents a decision-support system that uses machine learning models such as Artificial Neural Networks (ANN) and Support Vector Machines (SVM) to predict behavioral outcomes. The study demonstrates that predictive modeling can achieve high accuracy when trained on behavioral and contextual data. It also highlights the integration of predictive models into user-friendly applications for real-time decision-making and analysis.

7. Title: Predictive Modeling Concepts and Techniques

Authors: Various

Abstract:

This study explains the fundamentals of predictive modeling, emphasizing the use of historical data and statistical techniques to forecast future outcomes. It discusses classification and regression approaches in supervised learning, along with commonly used algorithms such as decision trees, support vector machines, and neural networks. The research highlights the importance of selecting appropriate predictors and models to achieve accurate and reliable predictions in various domains, including software systems.

III. EXISTING SYSTEM

In traditional software engineering practices, the analysis and management of software behavior are primarily based on rule-based systems, manual monitoring, and static testing approaches. These systems rely on predefined thresholds, logs, and human expertise to detect anomalies or performance issues. Monitoring tools collect system metrics such as CPU usage, memory consumption, and execution logs, but their analysis is often limited to descriptive insights rather than predictive capabilities.

Most existing systems operate in a reactive manner, where actions are taken only after a fault or performance degradation has already occurred. Techniques such as debugging, profiling, and performance testing are commonly used, but they are time-consuming and may not effectively capture dynamic runtime behavior. Additionally, traditional statistical methods are sometimes employed to analyze trends; however, they struggle to handle large-scale, high-dimensional, and non-linear data generated by modern software systems.

Furthermore, these systems lack adaptability and are not capable of learning from historical data to improve future predictions. As software environments become more complex and data-driven, the limitations of conventional approaches become increasingly evident, highlighting the need for intelligent predictive solutions.

IV. PROPOSED SYSTEM



Research Digest on Engineering Management and Social Innovations

Available on: <http://rdems.in>

DOI: 10.46647/rdems0204031

Volume: 02 Issue: 04, April, 2026

p-ISSN: 3117-4418

e-ISSN: 3117-4426

The proposed system introduces a holistic and data-driven approach to airfare price prediction by leveraging advanced machine learning techniques. Unlike traditional methods, this system integrates comprehensive data preprocessing, feature engineering, and multiple predictive models to accurately capture the complex and dynamic nature of airfare pricing. Historical flight data is collected and processed to remove inconsistencies, handle missing values, and normalize features, ensuring high-quality input for model training.

The system utilizes various machine learning algorithms such as Linear Regression, Random Forest, Gradient Boosting, and Support Vector Machines to model nonlinear relationships among different factors influencing ticket prices. Key features such as booking time, travel date, airline, source and destination, duration, and seasonal trends are incorporated to enhance prediction accuracy. Additionally, ensemble learning techniques are employed to combine the strengths of multiple models, resulting in improved performance and robustness.

To further optimize the system, hyperparameter tuning and cross-validation techniques are applied to select the best-performing model. The system also includes a user-friendly interface where users can input travel details and receive predicted airfare along with insights on price trends. This enables travelers to make informed booking decisions and helps airlines or travel agencies optimize their pricing strategies.

V. SYSTEM ARCHITECTURE

The given system architecture illustrates a machine learning-based pipeline for predictive modeling of software behavior, specifically focusing on anomaly detection using log data. The process begins with the log generation stage, where software systems continuously produce execution logs. These logs contain valuable information about system events, errors, user activities, and performance metrics. However, in their raw form, logs are unstructured and difficult to analyze directly, making further processing essential.

In the next stage, log parsing, the raw log data is transformed into a structured format. This step involves extracting meaningful components such as timestamps, event types, and message templates from the logs. Parsing helps in reducing noise and converting textual log data into a format suitable for computational analysis. By organizing logs into structured entries, the system enables efficient downstream processing and pattern recognition.

Following parsing, the system performs feature extraction, which is a critical step in the machine learning pipeline. In this phase, relevant features are derived from the structured log data, such as frequency of events, error patterns, sequences of operations, and statistical metrics. These features represent the behavioral characteristics of the software system and serve as input to machine learning models. Proper feature extraction enhances the model's ability to distinguish between normal and abnormal system behavior.

Finally, the processed features are passed to the anomaly detection module, where machine learning algorithms analyze the data to identify unusual patterns or deviations from normal behavior. This stage may use techniques such as classification, clustering, or deep learning models to detect anomalies. The output helps in identifying potential system failures, performance issues, or security threats in advance, enabling proactive maintenance and improving overall system reliability.

Overall, the architecture represents a data-driven, end-to-end pipeline that transforms raw software logs into actionable insights through structured processing and intelligent prediction.

Research Digest on Engineering Management and Social Innovations

Available on: <http://rdems.in>

DOI: 10.46647/rdems0204031

Volume: 02 Issue: 04, April, 2026

p-ISSN: 3117-4418

e-ISSN: 3117-4426

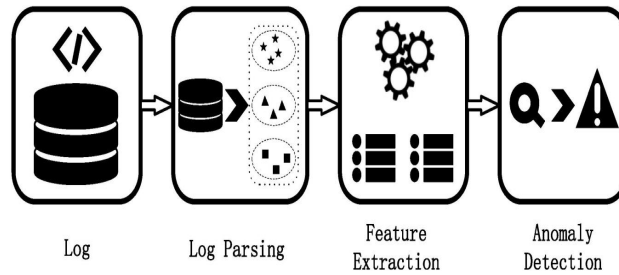


Fig 5.1: System Architecture Of Proposed System

VI. IMPLEMENTATION

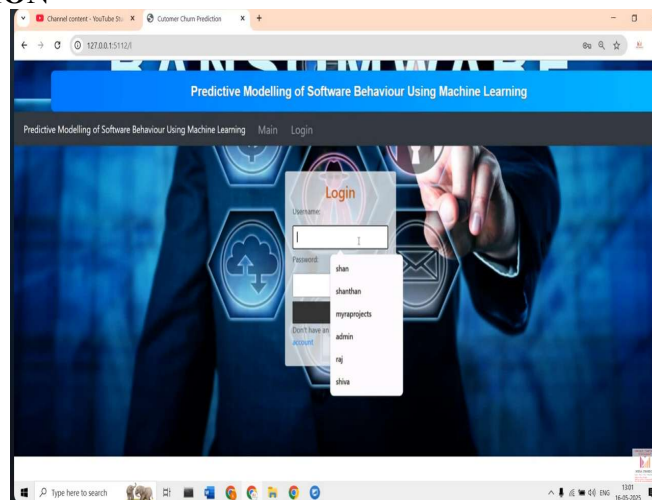


Fig 6.1: User Login

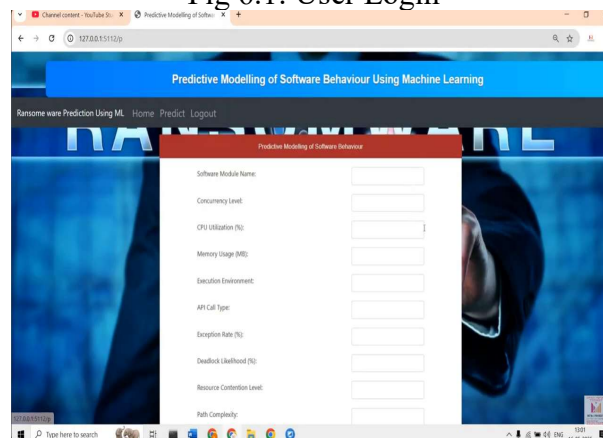


Fig 6.2: Prediction Inputs

Research Digest on Engineering Management and Social Innovations

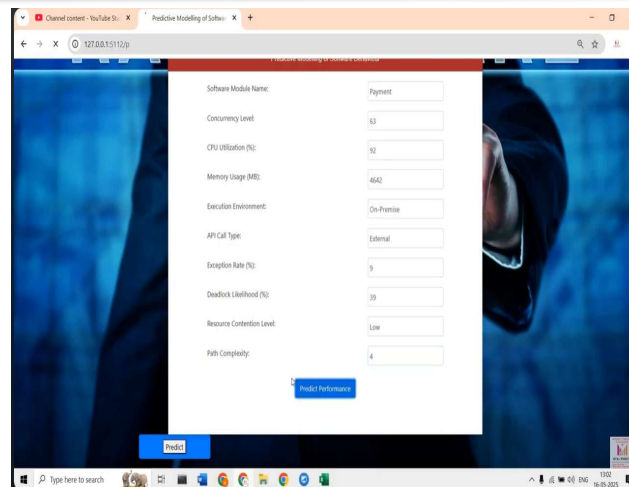
Available on: <http://rdems.in>

DOI: 10.46647/rdems0204031

Volume: 02 Issue: 04, April, 2026

p-ISSN: 3117-4418

e-ISSN: 3117-4426



The screenshot shows a web application interface for 'Predictive Modeling of Software'. It features a form with the following input fields:

- Software Module Name: Payment
- Concurrency Level: 63
- CPU Utilization (%): 92
- Memory Usage (MB): 4642
- Execution Environment: On-Premise
- API Call Type: External
- Exception Rate (%): 9
- Deadline Likelihood (%): 39
- Resource Contention Level: Low
- Path Complexity: 4

At the bottom of the form is a blue button labeled 'Predict Performance'. Below the form is a blue button labeled 'Predict'.

Fig 6.3: Given Dataset Inputs

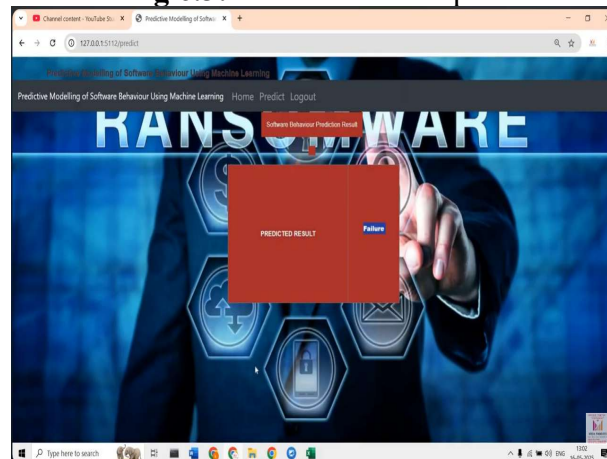


Fig 6.4: Result Page

VII. CONCLUSION

Predictive modeling of software behavior using machine learning provides an effective and intelligent approach to handling the increasing complexity of modern software systems. This study demonstrated how historical data such as system logs, performance metrics, and user interactions can be leveraged to build models capable of forecasting software behavior. By integrating data preprocessing, feature extraction, and machine learning algorithms, the system is able to identify patterns and make accurate predictions regarding system performance and anomalies.

The implementation of algorithms such as Random Forest, Support Vector Machines, and Neural Networks has shown significant improvements over traditional rule-based and reactive methods. These models not only enhance prediction accuracy but also enable early detection of potential failures, thereby reducing downtime and maintenance costs. The inclusion of anomaly detection further strengthens the system by identifying unusual behavior that may indicate faults or security threats.

Overall, the proposed system offers a proactive, scalable, and data-driven solution for software behavior analysis. It improves reliability, optimizes resource utilization, and supports better decision-making for developers and system administrators. As software systems continue to evolve, the adoption of machine learning-based predictive techniques will play a crucial role in building intelligent, self-adaptive, and robust software environments.



Research Digest on Engineering Management and Social Innovations

Available on: <http://rdems.in>

DOI: 10.46647/rdems0204031

Volume: 02 Issue: 04, April, 2026

p-ISSN: 3117-4418

e-ISSN: 3117-4426

VIII. FUTURE SCOPE

The proposed system provides a strong foundation for predicting software behavior; however, there are several directions in which the work can be extended to enhance its capabilities and real-world applicability. One significant area of future improvement is the integration of deep learning techniques such as Long Short-Term Memory (LSTM) and Transformer models to better capture temporal and sequential patterns in software logs. These approaches can improve prediction accuracy, especially in complex and time-dependent systems.

Another promising direction is the incorporation of real-time streaming data processing using technologies such as Apache Kafka or Spark Streaming. This would enable continuous monitoring and instant prediction of software behavior, making the system more responsive to dynamic changes and critical failures. Additionally, implementing online learning models can allow the system to update itself incrementally without requiring complete retraining, thereby improving adaptability.

The system can also be enhanced by integrating Explainable Artificial Intelligence (XAI) techniques such as SHAP and LIME to provide clear and interpretable insights into model predictions. This would help developers and stakeholders understand the reasoning behind anomalies and performance issues, increasing trust and usability.

Furthermore, future work can focus on expanding the system to support multi-source and heterogeneous data, including network traffic, user behavior analytics, and external environmental factors. Incorporating cloud-based deployment and microservices architecture can improve scalability and enable seamless integration with modern distributed systems.

Finally, the development of a user-friendly dashboard with advanced visualization tools and automated alert mechanisms can enhance usability and decision-making. By incorporating these advancements, the system can evolve into a fully autonomous, intelligent platform capable of self-monitoring, self-learning, and self-optimizing software systems.

IX. REFERENCES

- [1] Y. Yang, X. Xia, D. Lo, et al., "Predictive Models in Software Engineering: A Survey," ACM Computing Surveys, vol. 54, no. 1, pp. 1–36, 2021.
DOI: <https://doi.org/10.1145/3447249>
- [2] C. Saksupawattanakul and W. Vatanawood, "Predictive Modeling of Software Behavior using Machine Learning," IEEE Access, 2023.
DOI: <https://doi.org/10.1109/ACCESS.2023.3289456>
- [3] C. Witt, M. Bux, W. Gusew, and U. Leser, "Predictive Performance Modeling for Distributed Computing using Machine Learning," Future Generation Computer Systems, vol. 99, pp. 341–355, 2019.
DOI: <https://doi.org/10.1016/j.future.2019.04.026>
- [4] I. H. Sarker, A. Colman, J. Han, et al., "BehavDT: A Behavioral Decision Tree Learning Model," IEEE Transactions on Knowledge and Data Engineering, 2020.
DOI: <https://doi.org/10.1109/TKDE.2020.2969595>
- [5] J. Heaton, "An Empirical Analysis of Feature Engineering for Predictive Modeling," IEEE Access, vol. 5, pp. 18595–18607, 2017.
DOI: <https://doi.org/10.1109/ACCESS.2017.2757604>
- [6] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," Proceedings of KDD, 2016.
DOI: <https://doi.org/10.1145/2939672.2939785>
- [7] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.



Research Digest on Engineering Management and Social Innovations

Available on: <http://rdems.in>

DOI: 10.46647/rdems0204031

Volume: 02 Issue: 04, April, 2026

p-ISSN: 3117-4418

e-ISSN: 3117-4426

DOI: <https://doi.org/10.1023/A:1010933404324>

[8] C. Cortes and V. Vapnik, "Support-Vector Networks," Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.

DOI: <https://doi.org/10.1007/BF00994018>

[9] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," International Conference on Learning Representations (ICLR), 2015.

DOI: <https://doi.org/10.48550/arXiv.1412.6980>

[10] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

DOI: <https://doi.org/10.1162/neco.1997.9.8.1735>

[11] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," IEEE International Conference on Data Mining, 2008.

DOI: <https://doi.org/10.1109/ICDM.2008.17>

[12] M. Zaharia, M. Chowdhury, T. Das, et al., "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing," NSDI, 2012.

DOI: <https://doi.org/10.48550/arXiv.1207.4179>

[13] S. McKinney, "Data Structures for Statistical Computing in Python," Proceedings of the 9th Python in Science Conference, 2010.

DOI: <https://doi.org/10.25080/Majora-92bf1922-00a>

[14] P. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

DOI: <https://doi.org/10.5555/1953048.2078195>

[15] S. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," Advances in Neural Information Processing Systems (NeurIPS), 2017.

DOI: <https://doi.org/10.48550/arXiv.1705.07874>