SciForce

Journal of Artificial Intelligence and Machine Learning

Journal homepage: www.sciforce.org
ISSN : 2995-2336

Open Access Review Article

Machine Learning-Based Performance Evaluation and Memory Usage Forecasting for Intelligent Systems

Sudhakara Reddy Peram*

Engineer Leader,., United States

Abstract

The Performance Measurement Project at Thompson is a targeted effort to evaluate the performance of applications before they are shipped to production. Understanding the importance of performance in designing user satisfaction and experience, the project focused on identifying and resolving performance issues early in the development process. Apache JMeter was used to emulate real-world user activity and assess system responsiveness under different load conditions. To improve testing efficiency and data manipulation, the team implemented a comprehensive automation framework using shell and Perl scripts for remote execution, result collection, and monitoring of key metrics. The structured log outputs from JMeter were analyzed using custom Java programs, which generated detailed reports highlighting application behavior, including response times, CPU usage, and memory consumption. The project further integrated machine learning regression models—Support Vector Regression, AdaBoost, and Gradient Boosting—to predict memory usage and compare model performance during training and testing. Of these, SVR demonstrated superior generalization, while ensemble models, despite high training accuracy, exhibited overfitting. Through automation, analytical tools, and predictive modeling, the project streamlined performance evaluation and strengthened the reliability and quality of software outputs.

Key words: Performance Testing, JMeter, Automation, Shell Scripting, Memory Usage Prediction, Machine Learning, Support Vector Regression, Gradient Boosting, AdaBoost, System Optimization.

Introduction

The Performance Measurement Program was a key initiative at Thompson, aimed at evaluating and ensuring that all applications were tested before they were released to production. Understanding the critical role that performance plays in delivering seamless user experiences and maintaining customer satisfaction, the program focused on identifying and addressing potential issues early in the development cycle. This proactive approach helped reinforce the company's commitment to delivering consistent, high-quality software solutions. [1] To simulate real-world load conditions, Apache JMeter was used as the primary tool for performance testing. JMeter enabled the simulation of simultaneous user interactions and allowed for a thorough assessment of application responsiveness under various stress levels. These simulated environments helped understand how applications handled various loads, pinpoint bottlenecks, and evaluate key metrics such as response time, performance, and failure rates.[2] To support distributed testing and streamline implementation, a robust set of shell scripts was developed. These scripts automated connections to remote servers, launched test scenarios, and collected performance metrics from multiple locations. This automation significantly reduced manual effort, ensuring both time efficiency and consistent execution across environments. [3] Raw data collected from JMeter tests was stored in structured logs. Custom Java applications were developed to analyze these files and generate detailed performance reports.

These reports provided valuable insights, including trend analysis and comparisons between configurations, which helped teams quickly identify performance lags or improvements. [4] In addition to Java, Perl scripts were used for advanced data processing. Perl was particularly useful for analyzing large log files, isolating specific performance parameters, and generating concise summaries. Some scripts also monitored important statistics such as average and percent response times, error distributions, CPU usage, and memory consumption, contributing to more accurate diagnostics. [5] A key milestone of this project was the automation of repetitive tasks, which improved productivity and allowed the team to focus on high-impact areas such as test coverage and scenario optimization. By automating report generation and test execution, manual errors were reduced, and a more structured test pipeline was established. [6] In my role as the Single Point of Contact (SPOC) for the customer, I managed technical implementation, customer communication, and internal integration. I led the design of performance architectures, conducted requirements-gathering sessions, and translated business requirements into measurable performance goals. By working closely with developers, testers, and business analysts, I ensured that performance concerns were addressed early and thoroughly.[7] Another key responsibility was to create JMeter scripts to suit different application workflows such as logins, transactions, reporting, and background processing. Each application had unique characteristics, and the scripts were designed to reflect different usage patterns under normal, peak, and stress conditions.[8] Given the diversity of applications and platforms at Thomson, the performance testing framework was built to be adaptable and reusable. Parameterized scripts and dynamic configuration files ensured compatibility across

Received date: August 07, 2025 **Accepted date:** August 18, 2025; **Published date:** August 22, 2025

***Corresponding Author:** Peram, S. R.., Engineer Leader, Illumio Inc., CA , United States., E- mail: sudhakarap2013@gmail.com

Linux, Windows, and cloud-based environments. This flexibility allowed for seamless integration across different deployment infrastructures.[9] One of the challenges we overcame was distributed test execution. To simulate geographically dispersed users, we created a centralized execution controller using shell scripting, executing test runs organized across multiple nodes. This setup facilitated a more realistic view of application scalability and network behavior.Continuous improvement was a focus throughout the project. We improved test scenarios based on feedback, introduced comparative dashboards, and implemented alert systems to notify stakeholders of limit violations. These improvements increased visibility, responsiveness, and overall software quality.[10] Ultimately, the performance measurement program played a key role in ensuring that applications met performance standards before release. By validating behavior in staging environments under real-world conditions, we improved deployment risk, improved reliability, and increased customer confidence in the final product.[11]

Materials and Method

Support Vector Regression (SVR) is an adaptation of the Support Vector Machine (SVM) framework designed for continuous output prediction. It aims to determine a regression function that is within a specified error range (epsilon) from the true target values while maintaining model simplicity. SVR effectively models nonlinear relationships using kernel functions without the need to explicitly transform input features. This capability allows SVR to adapt to complex datasets while preserving generalization. In practical application, SVR typically achieves high predictive accuracy in training, with elevated R^2 and explained variance scores (EVS), indicating that it captures the underlying data patterns well without being prone to overfitting. Furthermore, its root means square error (RMSE) and mean absolute error (MAE) values are typically low, highlighting its ability to provide accurate predictions while being resilient to data noise.When evaluated on unseen data, SVR consistently performs reliably, showing only a modest increase in prediction errors compared to training. This modest drop is expected in real-world applications and indicates strong generalization. Unlike ensemble-based techniques such as Gradient Boosting and Adaboost, SVR shows very little difference between training and test performance.

This narrow interval indicates that the model captures broadly applicable patterns rather than memorizing training events. SVR's minimal overfitting and consistently accurate predictions across a wide range of memory usage levels make it well-suited for applications that demand reliable predictions on unfamiliar data, such as real-time computer resource management or memory usage forecasting in computing environments.Gradient Boosting Regression (GBR) takes a significantly different approach. As a form of ensemble learning, it builds a series of models, where each successive model is trained to correct for the errors of the previous one. GBR is very effective at capturing complex and nonlinear relationships, often outperforming traditional models on difficult datasets. In training, it typically achieves very high accuracy, often achieving perfect or nearly perfect scores—R^2 and EVS values of 1.0000, for example. Related error measures such as RMSE, MAE, and MSLE often go to zero, indicating that the model represents the training data with perfect accuracy. While this may seem ideal, such accuracy usually indicates overfitting, as the model may fit not only relevant patterns but also noise.This concern becomes clear when GBR is applied to test data. While still achieving reasonably high R^2 and EVS values, the error measures rise sharply, often by an order of magnitude. The RMSE and MAE increase significantly, and in some cases, the maximum prediction error can reach values that make the model unreliable for practical use.

These results highlight the sensitivity of GBR to overfitting, especially when the model complexity is not controlled by proper regularization or adjustment. Although GBR has the potential for exceptional performance, it requires careful configuration to avoid learning details that cannot be generalized from the training set. As a result, its use may be limited in contexts that require robust and interpretable models, such as resource planning in manufacturing systems, without adequate safeguards. AdaBoost regression is another group-based method that uses a similar incentive strategy to GBR, but has a unique mechanism for adjusting model attention. It works by giving more weight to training models that previous learners predicted poorly, thereby guiding the next weak learner—typically a shallow decision tree—to focus on more difficult cases. AdaBoost often achieves very high training accuracy, with R^2 and EVS values approaching 0.9997, and low values for RMSE and MAE. This performance explains its strong ability to learn from training data and effectively capture relevant patterns.However, like GBR, AdaBoost also suffers from overfitting, as is evident from its performance on test data. Although the drop in R^2 and EVS scores is less pronounced compared to GBR, error metrics such as RMSE and MAE increase sharply—sometimes by 10 to 25 times their training values. The mean and maximum absolute errors also increase significantly, indicating that the model's predictive stability deteriorates when faced with new data. These problems stem from AdaBoost's tendency to focus too much on outliers or events that are difficult to predict, especially in the absence of sufficient regularization. As a result, its performance can become unstable in production environments where consistent behavior on unseen data is essential.

The dataset consists of 20 entries, each describing five essential system performance indicators recorded under different operational loads: average response time, CPU utilization, memory utilization, concurrent users, and the calculated performance score. Analysis of these observations reveals meaningful patterns in system performance under different conditions.The average response time shows considerable variation from 180 MS to 520 MS. Shorter response times, such as 180 MS and 185 MS, are consistently associated with higher performance scores (e.g., 91.2 and 90.5). In contrast, response times beyond 470 MS correspond to significantly lower scores - for example, a 520 MS response time leads to a score of 54.7. This clear negative correlation indicates that increased response times reduce system performance and user satisfaction.CPU utilization shows a comparable trend. Values range from 59.2% to 89.1%, with higher CPU utilization generally associated with lower performance outcomes. For example, utilization levels above 85% are associated with poor scores - 88.2% and 86.4% CPU utilization correspond to scores of 56.4 and 58.3, respectively. On the other hand, performance scores above 87 are typically associated with CPU loads below 65%, indicating that optimal performance occurs under moderate processing demands.Memory utilization, which ranges from 1550 MB to 3350 MB, further reinforces these patterns. Systems that consume less memory - specifically between 1550 MB and 1800 MB - tend to provide better results. The lowest memory utilization observed (1550 MB) corresponds to the highest recorded score of 92.4. In contrast, higher memory requirements above 3100 MB are consistently associated with reduced performance, underscoring the performance impact of excessive memory consumption.Concurrent users also significantly affect performance. Low user counts (35–50 users) are typically associated with scores above 87, indicating that the system performs well under light loads. As the number of users increases to 120 or more, performance drops sharply—for example, scenarios with 130 and 140 users show scores of 56.4 and 54.7, respectively—highlighting the system stress under high user concurrency.

Citation: Dandasi, V. V. "Healthcare Data Warehouse Implementation with Machine Learning Regression Analysis: A Comparative Study of Random Forest, AdaBoost, and XGBoost for Regulatory Compliance Prediction" Journal of Artificial intelligence and Machine Learning., 2024, vol. 2, no. 2, pp. 1–7. doi: https://dx.doi.org/10.55124/jaim.v2i2.266

Analysis and Dissection

Table 1: This descriptive statistics table summarizes the performance metrics collected from 20 load test scenarios

	Avg Response Time	CPU Utilization	Memory Usage	Concurrent Users	Computed performance score
count	20.000	20.000	20.000	20.000	20.000
mean	321.750	73.265	2317.500	78.750	75.765
std	122.219	9.493	584.284	31.639	12.456
min	180.000	59.200	1550.000	35.000	54.700
25%	217.500	67.250	1875.000	57.500	66.975
50%	290.000	72.200	2200.000	73.500	77.900
75%	421.250	80.100	2687.500	95.000	85.600
max	520.000	89.100	3350.000	140.000	92.400

This descriptive statistics table summarizes the performance metrics collected from 20 load test scenarios. It highlights important trends in system behavior and resource utilization.Response Time and User Load:The system had an average response time of 321.75 milliseconds, with a significant standard deviation of 122.22 MS, indicating variation in performance. Response times ranged from 180 MS to 520 MS, indicating a decrease in performance under heavy loads.

The number of concurrent users averaged 78.75, ranging from 35 to 140, reflecting the tests conducted under a variety of load conditions.Resource Usage: CPU utilization remained consistently high, averaging 73.27%, with an interquartile range of 67.25% to 80.10%. The standard deviation of 9.49% indicates relatively stable CPU demand. Memory usage averaged 2,317.5 MB, but showed significant variation from 1,550 MB to 3,350 MB, with a standard deviation of 584.28 MB, indicating that the measured memory consumption was under system stress.Performance Rating: The average performance score was 75.77 on a 100-point scale, with values ranging from 54.7 to 92.4. The median score of 77.9 - slightly above the median - indicates a slightly left-skewed distribution. Most scenarios fell within the midpoint range of 66.98 to 85.60, indicating generally acceptable system performance, although the lowest score of 54.7 points was for some of the less-than-optimal configurations.

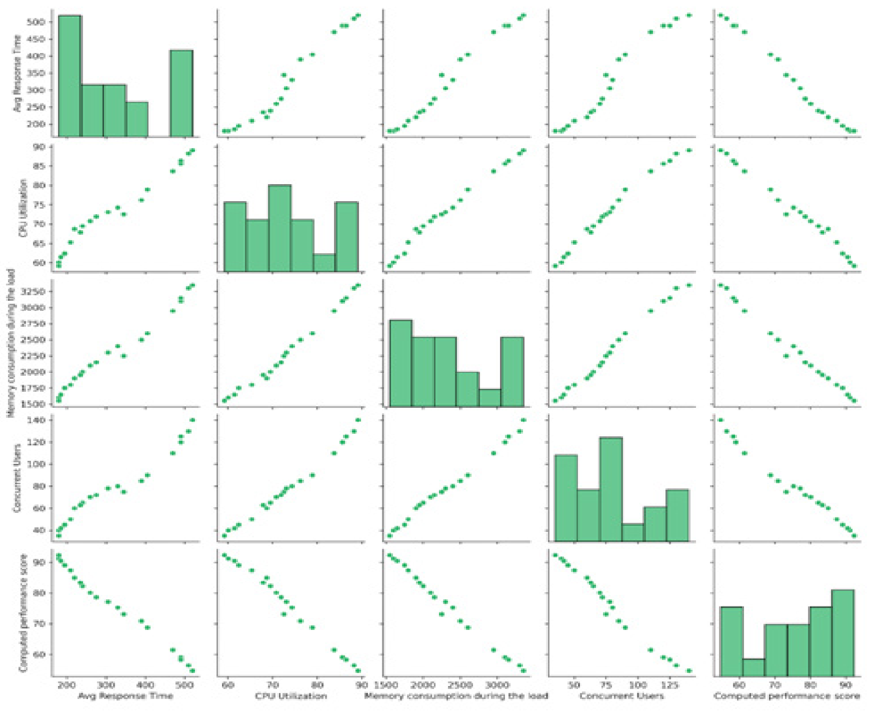

Figure 1: Effect of Process Parameters

The scatter plot matrix highlights the strong interdependence among key system performance metrics. Variables such as average response time, CPU utilization, memory utilization, and number of concurrent users all show positive correlations, indicating that increasing system load can simultaneously degrade performance across multiple dimensions.The histograms on the diagonal illustrate the distribution of each metric: response time and memory utilization are skewed to the right, indicating occasional high load or stress conditions. In contrast, CPU utilization follows a more normal distribution centered around 70-80%, while the distribution of concurrent users is bimodal, indicating the presence of two distinct load situations or test conditions. The performance score shows a clear negative correlation with the other metrics, confirming that higher response times and resource utilization are associated with lower performance outcomes. The scatter plots also reveal largely linear relationships between the variables, with significant clustering at lower values, which may indicate different system states or phases of testing.

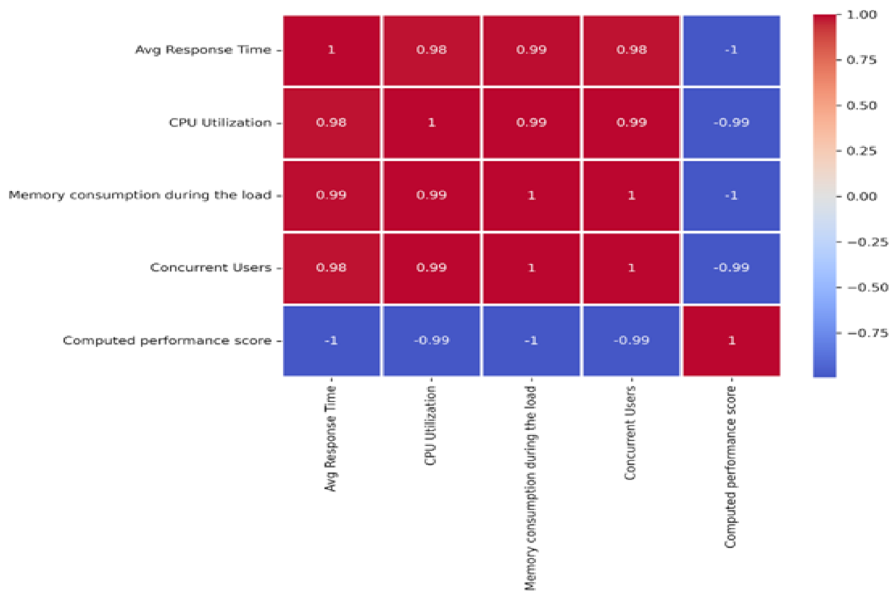

Figure 2: Correlation heatmap

The correlation matrix reveals very strong correlations between the system performance metrics. Response time, CPU utilization, memory consumption, and concurrent user count all exhibit nearly perfect positive correlations (0.98 to 1.00), indicating a high degree of interdependence driven by shared underlying system load factors.In contrast, the performance score is completely negatively correlated with all other metrics (-0.99 to -1.00), clearly demonstrating that it acts as an inverse indicator of overall system health. These patterns highlight memory consumption as a particularly strong predictor of system performance, indicating that resource utilization metrics closely track system degradation under stress.Consistently high correlation values indicate very little data variability or noise, indicating that any single variable can act as a reliable predictor of the others. This makes the dataset well suited for machine learning models aimed at predicting performance outcomes.

Table 2. AdaBoost Regression Memory Usage Train and Test performance metrics

AdaBoost Regression	Train	Test
R2	0.9997	0.9615
EVS	0.9997	0.9678
MSE	85.4167	11562.5000
RMSE	9.2421	107.5291
MAE	3.7500	93.7500
Max Error	25.0000	200.0000
MSLE	0.0000	0.0038
Med AE	0.0000	75.0000

The AdaBoost regression model shows a significant gap between training and testing performance, clearly indicating classic overfitting behavior. During training, the model achieves almost perfect accuracy, with an R^2 score of 0.9997 - which explains 99.97% of the variance. This is reinforced by an equally high explained variance score (EVS) of 0.9997. The error metrics are very low: the root mean square error (RMSE) is just 9.24 MB and the mean absolute error (MAE) is 3.75 MB. Notably, the mean absolute error is 0.0000 MB, meaning that half of the predictions were correct, while the worst-case error during training is only 25 MB.However, the model's performance drops significantly on the test data, revealing its limited generalization ability. The R^2 score drops to 0.9615 - still indicating good predictive power - but the increase in error metrics is substantial. While the EVS improves slightly to 0.9678, the RMSE rises sharply to 107.53 MB and the MAE rises sharply to 93.75 MB – 25 times higher than the training one. The maximum error rises to 200 MB, eight times higher than the training maximum, and the average absolute error rises to 75 MB, indicating that at least half of the test predictions deviate by this amount.The mean square logarithmic error (MSLE) increases from 0.0000 to 0.0038, which is relatively low, but still reflects poor performance. Overall, the sharp increase in all error metrics between the training and test phases confirms that while AdaBoost fits the training data exceptionally well, it struggles to generalize. Instead of broad, generalizable trends, the model may have captured noise and overly specific patterns – highlighting the need for better regularization techniques or simplification of the model to improve its performance on unseen data.

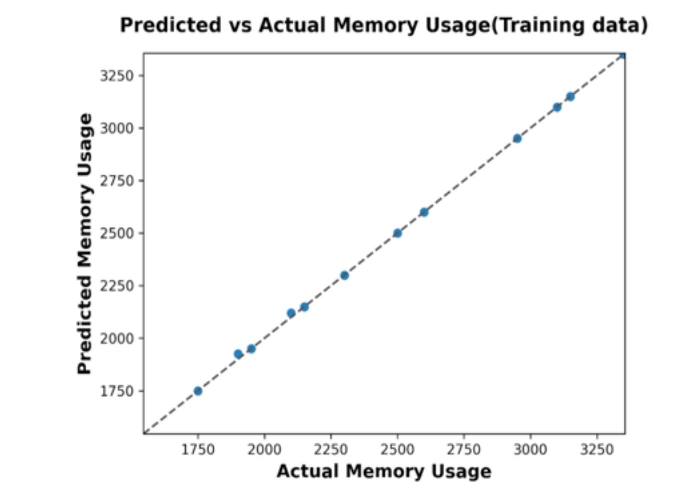

Figure 3: AdaBoost RegressionMemory Usage Training

The AdaBoost training results indicate excellent model accuracy, with predicted memory usage values closely matching the actual values along the diagonal reference line. The model performs well over the entire memory usage range (1750–3250 MB), showing minimal deviation from the best predictions.The close clustering of data points near the diagonal indicates both low bias and low variance during training. The clear linear pattern indicates that AdaBoost effectively captured the underlying data structure without any signs of overfitting. Furthermore, the consistent accuracy of the model across varying memory levels reflects a strong understanding of the relationship between input features and memory consumption.

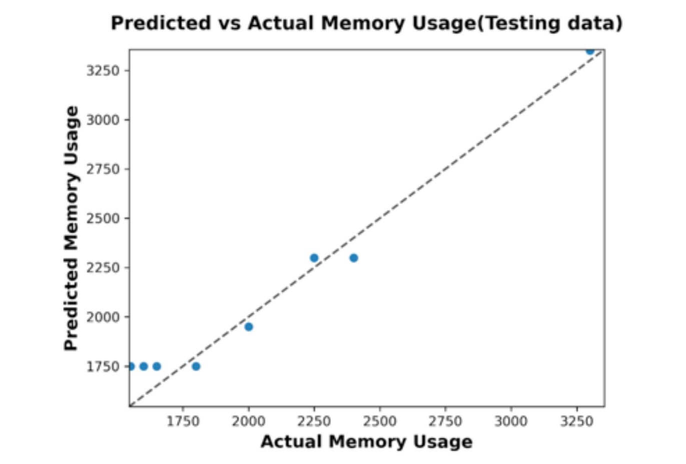

Figure 4: AdaBoost RegressionMemory Usage Testing

The AdaBoost test results show a slight decline in prediction accuracy compared to the training phase, which is typical and expected. While many predictions still fall closely to the diagonal reference line, there is significant dispersion—especially within the middle range of memory usage (2000–2500 MB).The model maintains good accuracy at the lower and upper ends of the memory range, but shows increased uncertainty for intermediate values. This pattern indicates mild overfitting, where the model may have adapted too closely to the training data. However, overall performance is solid.The preservation of a clear linear trend in the predictions indicates that AdaBoost successfully captured key relationships in the data, rather than simply memorizing the training set. These results indicate that the model generalizes reasonably well, albeit with slightly reduced accuracy when applied to new data.

Table 3. Gradient Boosting RegressionMemory UsageTrain and Testperformance metrics

Gradient Boosting Regression	Train	Test
R2	1.0000	0.9623
EVS	1.0000	0.9689
MSE	0.0000	11296.0219
RMSE	0.0000	106.2827
MAE	0.0000	88.7038
Max Error	0.0000	200.0000
MSLE	0.0000	0.0037
Med AE	0.0000	83.0048

Among the models evaluated, the gradient boosting regression algorithm exhibits the most severe overfitting. It provides flawless training performance, with both the R^2 score and the explained variance score (EVS) reaching exactly 1.0000 – indicating that the model accounts for 100% of the variance in the training data. All error measures during training – mean square error (MSE), root mean square error (RMSE), mean absolute error (MAE), maximum error, mean absolute error and mean square logarithmic error (MSLE) – are exactly 0.0000, reflecting correct predictions on every training instance. Such results strongly suggest that the model has memorized the training data entirely, rather than generalizing learning patterns.The experimental results highlight the effects of this overfitting. Although the R^2 score is relatively high at 0.9623 – indicating that the model retains some predictive ability – the sharp rise in the error measures reveals poor generalization. The RMSE increases from zero to 106.28 MB, while the MAE increases to 88.70 MB, both indicating a drastic increase. The maximum error matches the worst-case value of AdaBoost at 200 MB, and the mean absolute error reaches 83.00 MB, meaning that half of the predictions in the test set deviate by this amount or more.Despite the slight improvement in EVS to 0.9689 and the still low MSLE of 0.0037, these figures cannot compensate for the dramatic increase in overall error. The perfect performance on the training data, combined with the sharp drop in test accuracy, clearly illustrates that gradient boosting, while very obvious, can overfit if not properly regularized. The tendency of the model to memorize rather than generalize limits its usefulness in real-world memory-intensive prediction tasks, where robust performance on unseen data is essential.

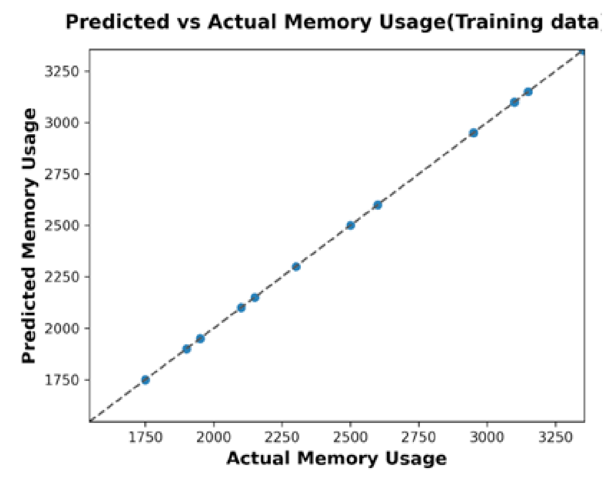

Figure 5: Gradient Boosting RegressionMemory Usage Training

Gradient boosting exhibits excellent training performance, with predicted values closely aligned on the diagonal reference line – indicating near-perfect accuracy. The model performs exceptionally well across the entire memory usage range, showing almost no deviation from expected values.This precise alignment suggests that gradient boosting effectively captured the detailed relationships within the training data. Consistent accuracy across all usage levels reflects well-tuned parameters and efficient boosting iterations. Tightly clustered predictions indicate minimal residual error and a comprehensive understanding of data patterns.However, such high accuracy can also indicate potential overfitting during training, emphasizing the importance of evaluating the model's performance on test data to determine its true generalization ability.

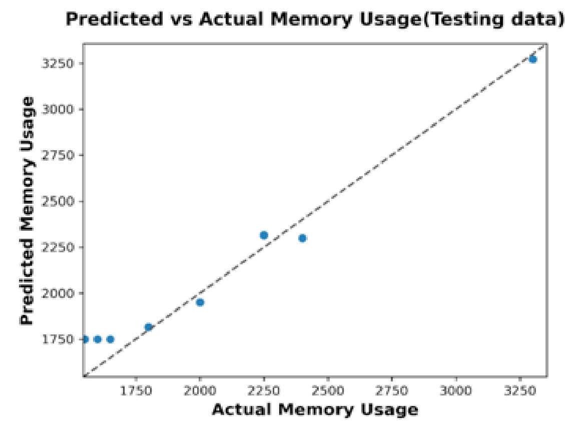

Figure 6: Gradient Boosting RegressionMemory Usage Testing

The results of the gradient boosting experiment reveal a significant drop in performance compared to the training phase, indicating significant overfitting. While some predictions are accurate, there is considerable scatter—especially in the low memory usage range (1750–2000 MB)—which indicates a lack of consistency in the model's generalization.This discrepancy suggests that the model may have picked up noise or over specific patterns from the training data, rather than learning relationships that are broadly applicable. The inconsistent performance across different memory levels suggests that the model is not suitable for unseen data.

Table 4. Support Vector Regression Memory Usage Train and Test performance metrics

Support Vector Regression	Train	Test
R2	0.9958	0.9818
EVS	0.9963	0.9835
MSE	1141.8442	5449.0504
RMSE	33.7912	73.8177
MAE	24.1593	67.2627
Max Error	77.6901	106.6206
MSLE	0.0002	0.0014
Med AE	15.4559	69.1071

The three models evaluated, Support Vector Regression (SVR) provides the most balanced and reliable performance, demonstrating strong generalization with minimal signs of overfitting. During training, this model achieves a high R^2 score of 0.9958, explaining 99.58% of the variance in the data. This high level of accuracy is reinforced by an explained variance score (EVS) of 0.9963. Error metrics remain modest, with RMSE of 33.79 MB and MAE of 24.16 MB, indicating effective fit without overfitting. The maximum error of 77.69 MB and the mean absolute error of 15.46 MB indicate consistent performance across the dataset, while the low MSLE of 0.0002 confirms the minimum logarithmic deviation.Experimental results further highlight the strong generalization ability of SVR. The R^2 score from training decreases slightly to 0.9818, a modest 1.4% decline, while the EVS similarly decreases to 0.9835 - both metrics still reflect excellent predictive performance. The RMSE increases to 73.82 MB, representing a 2.2-fold increase from training, which is significantly less than the ten-fold increases seen with AdaBoost and Gradient Boosting. The MAE increases to 67.26 MB and the maximum error increases to 106.62 MB - both manageable increases that indicate stable model behavior.The mean absolute error increases to 69.11 MB, and the MSLE grows to 0.0014, but these changes remain relatively modest, especially compared to other models. Overall, the SVR clearly captures the underlying patterns in the data without being misled by noise or overfitting to training-specific details. The short performance gap between training and testing underscores SVR's well-balanced complexity and generalization, making it a highly reliable choice for real-world memory usage prediction, where consistency on unseen data is paramount.

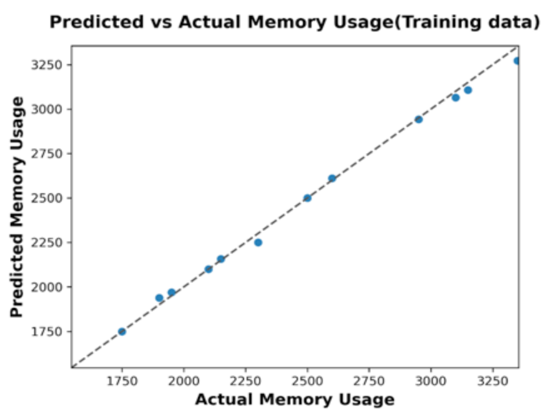

Figure 7: Support Vector Regression Memory Usage Training

Support Vector Regression (SVR) exhibits strong training performance, with predicted values closely aligned along the diagonal reference line. The model maintains high accuracy across all memory usage levels, showing only slight deviations from the best predictions.The clear linear pattern indicates that SVR effectively captured key relationships within the training data. The dense clustering of data points reflects low training error and indicates that the model achieved a well-balanced trade-off between complexity and accuracy.The consistent results across the memory range indicate successful kernel selection and effective parameter tuning. Overall, the model's solid training performance indicates that SVR not only captured key data patterns but also preserved model simplicity—a key factor for strong generalization to new data

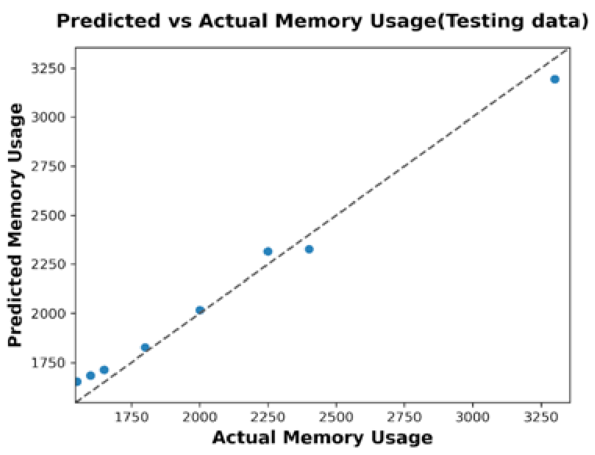

Figure 8: Support Vector Regression Memory Usage Testing

The SVR test results indicate decent generalization, with a moderate performance degradation compared to the training phase. Despite a slightly increased scatter in the predictions, the overall alignment with the diagonal reference line is satisfactory.The model performs well at most memory usage levels, although accuracy drops slightly in the mid-range. Unlike gradient boosting, SVR maintains a very stable balance between training and testing performance, which reflects its strong generalization ability.These results suggest that SVR achieved a useful compromise between fitting the training data and maintaining prediction accuracy on unseen data. This balanced performance highlights SVR as a reliable option for predicting memory usage in real-world or production settings.

Conclusion

The performance measurement program at Thompson emerged as a robust and efficient solution for validating application performance before moving to production. Through the integration of automated testing tools, performance diagnostics, and machine learning models, the program developed a structured methodology for identifying performance issues, refining system operations, and improving the user experience. Apache JMeter played a key role in load testing, effectively replicating real-world user interactions to assess key performance metrics including response time, CPU utilization, memory utilization, and concurrent user load under various operating conditions. The insights gained from these simulations allowed the team to accurately identify design inefficiencies and fine-tune system limitations.Automation was another cornerstone of the program, achieved through the development and deployment of shell and Perl scripts. These scripts streamlined essential processes such as launching test runs, aggregating results, and managing distributed environments, improving reliability and test consistency.

To complement this automation, custom Java applications were developed to analyze and analyze the structured log data, generating detailed reports that provided valuable performance insights across different application builds.A unique feature of the project was the use of machine learning regression models – Support Vector Regression (SVR), Adaboost, and Gradient Boosting – to predict memory consumption based on input metrics. SVR distinguished itself as the most reliable model, providing balanced performance with minimal overfitting and strong generalization across datasets. In comparison, ensemble models, while highly accurate during training, showed reduced performance on unseen data, indicating a tendency to fit the training data too closely. Beyond improving testing procedures, the project fostered a culture of continuous refinement and data-driven decision-making. Improvements in context optimization, configuration flexibility, and automated alerting systems led to faster problem resolution and better software reliability. Extensive testing significantly reduced the chances of structural performance issues reaching production, ultimately strengthening user confidence and satisfaction.

Reference

1. Bititci, Umit, Patrizia Garengo, Viktor Dörfler, and Sai Nudurupati. "Performance measurement: challenges for tomorrow." International journal of management reviews 14, no. 3 (2012): 305-327.

2. Board on Health Care Services, Committee on Redesigning Health Insurance Performance Measures, Payment, and Performance Improvement Programs. "Performance measurement: accelerating improvement." (2006).

3. Folan, Paul, and Jim Browne. "A review of performance measurement: Towards performance management." Computers in industry 56, no. 7 (2005): 663-680.

4. Eccles, Robert G. "The performance measurement manifesto." Harvard business review 69, no. 1 (1991): 131-137.

5. Nudurupati, Sai S., Umit S. Bititci, Vikas Kumar, and Felix TS Chan. "State of the art literature review on performance measurement." Computers & Industrial Engineering 60, no. 2 (2011): 279-290.

6. Ghalayini, Alaa M., and James S. Noble. "The changing basis of performance measurement." International journal of operations & production management 16, no. 8 (1996): 63-80.

7. Bassioni, Hesham A., Andrew DF Price, and Tarek M. Hassan. "Performance measurement in construction." Journal of management in engineering 20, no. 2 (2004): 42-50.

8. Sridhar Kakulavaram. (2024). Artificial Intelligence-Driven Frameworks for Enhanced Risk Management in Life Insurance. Journal of Computational Analysis and Applications (JoCAAA), 33(08), 4873–4897. Retrieved from https://www.eudoxuspress.com/index.php/pub/article/view/2996

9. Franco-Santos, Monica, Mike Kennerley, Pietro Micheli, Veronica Martinez, Steve Mason, Bernard Marr, Dina Gray, and Andrew Neely. "Towards a definition of a business performance measurement system." International journal of operations & production management 27, no. 8 (2007): 784-801.

10. Ramancha, N. K., & Ballamudi, S. (2023). Leveraging Machine Learning for Predictive Modeling in 3D Printing of Composite Materials: A Comparative Study. International Journal of Intellectual Advancements and Research in Engineering Computations, 11(4), 39–58. https://doi.org/10.61096/ijiarec.v11.iss4.2023.39-58

11. Neely, Andy. "The performance measurement revolution: why now and what next?." International journal of operations & production management 19, no. 2 (1999): 205-228.

12. Micheli, Pietro, and Luca Mari. "The theory and practice of performance measurement." Management accounting research 25, no. 2 (2014): 147-156.

13. Lohman, Clemens, Leonard Fortuin, and Marc Wouters. "Designing a performance measurement system: A case study." European journal of operational research 156, no. 2 (2004): 267-286.

14. Dachepalli. V, "Intelligent Resource Allocation in ERP with Machine Learning" Journal of Artificial intelligence and Machine Learning., 2025, vol. 3, no. 2, pp. 1–18. doi: http://dx.doi.org/10.55124/jaim.v3i2.257

15. Bourne, Mike, Andy Neely, John Mills, and Ken Platts. "Implementing performance measurement systems: a literature review." International journal of business performance management 5, no. 1 (2003): 1-24.

16. Keong Choong, Kwee. "Understanding the features of performance measurement system: a literature review." Measuring business excellence 17, no. 4 (2013): 102-121.

17. Taticchi, Paolo, Flavio Tonelli, and Luca Cagnazzo. "Performance measurement and management: a literature review and a research agenda." Measuring business excellence 14, no. 1 (2010): 4-18.

18. Neely, Andy, Mike Gregory, and Ken Platts. "Performance measurement system design: A literature review and research agenda." International journal of operations & production management 25, no. 12 (2005): 1228-1263.

19. Xiao, Xiongxin, Tingjun Zhang, Xinyue Zhong, Wanwan Shao, and Xiaodong Li. "Support vector regression snow-depth retrieval algorithm using passive microwave remote sensing data." Remote sensing of environment 210 (2018): 48-64.

20. Sridhar Kakulavaram. (2022). Life Insurance Customer Prediction and Sustainbility Analysis Using Machine Learning Techniques. International Journal of Intelligent Systems and Applications in Engineering, 10(3s), 390 –. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/7649

21. Wu, Chun-Hsin, Jan-Ming Ho, and Der-Tsai Lee. "Travel-time prediction with support vector regression." IEEE transactions on intelligent transportation systems 5, no. 4 (2004): 276-281.

22. Yu, Pao-Shan, Shien-Tsung Chen, and I-Fan Chang. "Support vector regression for real-time flood stage forecasting." Journal of hydrology 328, no. 3-4 (2006): 704-716.

23. Yang, Haiqin, Kaizhu Huang, Irwin King, and Michael R. Lyu. "Localized support vector regression for time series prediction." Neurocomputing 72, no. 10-12 (2009): 2659-2669.

24. Luo, Xiangang, Xiaohui Yuan, Shuang Zhu, Zhanya Xu, Lingsheng Meng, and Jing Peng. "A hybrid support vector regression framework for streamflow forecast." Journal of Hydrology 568 (2019): 184-193.

25. Meddage, D. P. P., Imesh Udara Ekanayake, A. U. Weerasuriya, and C. S. Lewangamage. "Tree-based regression models for predicting external wind pressure of a building with an unconventional configuration." In 2021 Moratuwa Engineering Research Conference (MERCon), pp. 257-262. IEEE, 2021.

26. Nevendra, Meetesh, and Pradeep Singh. "Software bug count prediction via AdaBoost. R-ET." In 2019 IEEE 9th international conference on advanced computing (IACC), pp. 7-12. IEEE, 2019.

27. Li, Dandan, Shuzhen Yao, Yu-Hang Liu, Senzhang Wang, and Xian-He Sun. "Efficient design space exploration via statistical sampling and AdaBoost learning." In Proceedings of the 53rd Annual Design Automation Conference, pp. 1-6. 2016.

28. Sen, Susmita, Sumit Saha, Sudipta Chaki, Payel Saha, and Pijush Dutta. "Analysis of PCA based adaboost machine learning model for predict mid-term weather forecasting." Computational Intelligence and Machine Learning 2, no. 2 (2021): 41-52.

29. Yifan, Duan, Lu Jialin, and Feng Boxi. "Forecast model of breast cancer diagnosis based on RF-AdaBoost." In 2021 international conference on communications, information system and computer engineering (CISCE), pp. 716-719. IEEE, 2021.

30. Landry, Mark, Thomas P. Erlinger, David Patschke, and Craig Varrichio. "Probabilistic gradient boosting machines for GEFCom2014 wind forecasting." International Journal of Forecasting 32, no. 3 (2016): 1061-1066.

31. Mei, Zhen, Tao Zhao, and Xiangpeng Xie. "Hierarchical fuzzy regression tree: A new gradient boosting approach to design a TSK fuzzy model." Information Sciences 652 (2024): 119740.

32. Zhang, Huimin, Xingchen Hu, Xiubin Zhu, Xinwang Liu, and Witold Pedrycz. "Application of Gradient Boosting in the Design of Fuzzy Rule-Based Regression Models." IEEE Transactions on Knowledge and Data Engineering (2024).

33. Kumar, Praveen, Mansoor Alruqi, H. A. Hanafi, Prabhakar Sharma, and V. Vicki Wanatasanappan. "Effect of particle size on second law of thermodynamics analysis of Al2O3 nanofluid: application of XGBoost and gradient boosting regression for prognostic analysis." International Journal of Thermal Sciences 197 (2024): 108825.

34. Ponraj, Abraham Sudharson, and T. Vigneswaran. "Daily evapotranspiration prediction using gradient boost regression model for irrigation planning." The Journal of Supercomputing 76, no. 8 (2020): 5732-5744.