UDC 004.4
DOI https://doi.org/10.32782/2663-5941/2025.4.2/07

Burchak P.V.
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"

Oleshchenko L.M.
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"

MACHINE LEARNING SOFTWARE FRAMEWORK FOR PREDICTING AND OPTIMIZING WEB APPLICATION PERFORMANCE

The article introduces a software framework for analyzing and predicting the performance of web applications using machine learning (ML) techniques. Instead of relying solely on traditional performance testing, an automated pipeline has been introduced that has collected, processed, and modeled key system metrics – such as server response time, memory usage, CPU load, and UI responsiveness – to proactively identify and address performance issues. The dataset used in the study has been gathered from real-world web applications and has included over 200,000 performance records, encompassing web page load time, layout recalculations, JavaScript memory usage, and network throughput.

Multiple ML algorithms have been integrated into the modeling pipeline, including linear regression (LR), polynomial regression, decision trees, and neural networks (NN), to compare their effectiveness in forecasting application performance. Experimental results have shown that LR has achieved the best overall performance (MAE = 187.25, R^2 = 0.9357), indicating a strong linear correlation in the data. Neural network also demonstrated high accuracy and advantages in modeling more dynamic, nonlinear behaviors. Proposed developed software framework has been implemented using Python technologies, such as Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn, and TensorFlow.

The research has resulted in a scalable, automated, and interpretable software solution for predictive diagnostics and performance optimization of web applications. Proposed software system has also generated categorized web applications performance reports with actionable recommendations. The research results have confirmed the feasibility and effectiveness of incorporating machine learning into modern web applications performance monitoring workflows.

Key words: *software framework, software performance monitoring, machine learning, predictive analytics, web systems optimization, regression models, neural network, Python, TensorFlow, web application performance forecasting.*

Formulation of the problem. In the modern digital landscape, the performance of web applications plays a crucial role in determining the success of software products. Fast load times, responsiveness, and system stability directly impact user satisfaction and a company's competitiveness. With the increasing number of users, the variety of devices, and the complexity of server-client interactions, evaluating and maintaining optimal performance levels has become a significant challenge.

Traditional methods, such as manual testing or standard automated scripts, often fall short when it comes to handling large volumes of data or adapting to dynamic conditions like network delays, traffic spikes, or shifts in user behavior. As web applications grow more sophisticated, performance becomes a vital component of both user experience and business outcomes.

Research indicates that even a one-second delay in page loading can lead to a 7% drop in conversions, while more than half of mobile users abandon websites that take over three seconds to load. Search engines like Google factor in site speed when ranking content. Industry experience demonstrates that implementing optimizations – such as reducing HTTP requests, compressing resources, and applying caching techniques – can cut load times by 20–50%, leading to better retention and overall performance. Recently, machine learning (ML) has emerged as a promising solution, enabling predictive load analysis and adaptive resource management, thus enhancing performance without requiring major infrastructure upgrades.

Evaluating the performance of web applications remains a critical challenge in contemporary software engineering, as system efficiency directly impacts both user satisfaction and business outcomes. A wide range of techniques is employed in this domain, including analytical frameworks, empirical testing, and ML approaches. Conventional performance evaluation typically relies on metrics such as server response time, requests per second (RPS), and resource utilization (CPU, memory, and network bandwidth). Tools like Apache JMeter, New Relic, and Dynatrace provide valuable outcomes into system behavior, though they often lack capabilities for forecasting future loads or uncovering subtle performance trends. In response to these limitations, ML-based methods have gained popularity for their ability to analyze and predict application behavior more accurately.

Techniques such as linear and polynomial regression, decision trees, neural networks, and ensemble models have proven effective in detecting anomalies and forecasting performance issues. The research indicates that deep neural networks can outperform traditional models by 15–30% in prediction accuracy.

Another emerging area is the application of reinforcement learning for dynamic, real-time adjustment of system parameters, with some studies reporting up to 40% reduction in average response times.

Modern strategies for performance analysis are moving toward a hybrid approach that combines classical monitoring tools with intelligent ML models. Ongoing challenges include model interpretability, the need for extensive training datasets, and ensuring adaptability across various web application architectures.

This research focuses on creating a software-based solution for testing and evaluating the performance of web applications by integrating ML techniques, particularly regression models and neural networks.

Regression models help uncover key patterns and measure the influence of performance factors, while neural networks offer deeper outcomes into complex, nonlinear relationships in large datasets. This combined approach enables both real-time evaluation and performance forecasting under dynamic conditions.

The proposed method is designed to be scalable and efficient, capable of handling increased data volumes without compromising speed or accuracy.

Analysis of recent research and publications. The study in the article [1] demonstrated that among the existing approaches to regression testing of web applications, no single method consistently outperforms the others, as their effectiveness largely depends on various factors and deployment environments. The research emphasizes the importance of evaluating these methods not only by technical parameters but also by their cost-effectiveness. In the article [2], a performance evaluation methodology for web pages is proposed, leveraging ML algorithms to enhance the accuracy and efficiency of assessing web resource quality. A systematic literature review covering 2010–2024 identified 59 key factors affecting website performance. The proposed model integrates a wide range of metrics – such as usability, accessibility, content relevance, visual appeal, and technical characteristics – enabling it to overcome the limitations of traditional approaches. The study showed that SVM algorithms achieved the highest prediction accuracy (89%) after feature selection, compared to 87% without it. Random Forest models also saw a slight improvement, from 80% to 81%. The use of feature selection significantly improved model performance, highlighting the value of selecting influential predictors. This approach enables the automation of web performance evaluation and prediction of potential issues at the development stage, supporting more informed decision-making and setting new research benchmarks.

A systematic review in the article [3] presents an overview of current methods and strategies for regression testing of web applications, covering areas such as test prioritization, selection, and adaptation to dynamic environments. The review includes both classical and modern techniques, such as automated testing and change analysis tools. The results of research [4] indicate that optimizing non-functional performance attributes like speed, startup time, memory usage, and energy consumption is essential to improving user satisfaction with mobile applications. The authors also identify existing optimization techniques and research gaps for future exploration. According to the article [5], key design aspects – such as interface simplicity, navigability, content readability, feedback mechanisms, personalization, and aesthetic design – strongly influence users' perceptions of usability and utility, which in turn affects their engagement and satisfaction.

In the article [6], the authors examine performance optimization methods including caching, compression, CDNs, and tools like WebPageTest and YSlow, aimed at assisting developers in improving website speed and search engine visibility.

The study in the article [7] focuses on enhancing trace visualization for analyzing microservices performance by addressing the shortcomings of existing tools and introducing new techniques for better diagnostic efficiency.

The work [8] explores how Dynatrace monitoring data can be used to develop performance models for

Java EE applications, aiming to improve performance understanding and modeling precision.

The research [9] presents methods for optimizing the performance of web application state management, with a particular focus on improving responsiveness and reducing resource consumption. Through experimental evaluation of various state management strategies – such as client-side caching, minimizing state mutations, and asynchronous state updates – the authors demonstrate measurable gains in load time, memory usage, and CPU efficiency. The results confirm that careful architectural decisions in managing application state can significantly enhance overall performance, particularly in complex and dynamic web environments.

Across the reviewed literature, the main challenges in regression analysis of web application performance include the difficulty of adapting traditional testing approaches to dynamic, component-based architectures; minimizing test suite size while maintaining adequate coverage; prioritizing tests to detect critical defects more effectively; and automating the testing process – particularly for interactive elements like GUIs and JavaScript components.

Researchers also point to the high time and resource costs of large-scale regression testing and emphasize the need for tools that address both functional and non-functional aspects of performance. As application load or user numbers grow, predicting system behavior under future conditions becomes crucial. The nonlinear relationships between parameters such as requests per second, hardware resource usage, and average server response time complicate traditional analysis approaches, highlighting the need for more adaptive and flexible evaluation techniques that can account for these complex interdependencies.

Task statement. The goal of the research is to develop a performance evaluation software tool that automates analysis, detects system bottlenecks, and anticipates potential issues before they impact functionality. The key tasks include reviewing existing methods to identify their strengths and limitations; selecting a core set of performance metrics – such as response time, server latency, and resource usage – that provide meaningful outcomes; building and training ML models to generate predictive analytics; and developing a software framework that brings all these elements together into an integrated, usable solution.

The resulting software system must encompasses every stage of performance analysis – from data gathering to classification and reporting – delivering a unified and scalable method. It also must provides actionable recommendations to improve performance, including API optimization, asynchronous request handling, and reduction of data transfer volume and code complexity.

Outline of the main material of the research. To evaluate the performance of web applications, data containing over 200,000 real user records was selected for in-depth analysis of various performance aspects. The Chrome User Experience Report (CrUX) offers a valuable source of real-world performance data, gathered directly from users of the Chrome browser.

Unlike synthetic testing methods or controlled lab environments, CrUX reflects how websites actually perform under diverse real-life conditions, including different devices, network types, and geographic locations. This makes it an essential resource for studying web performance, enhancing user experience, and improving frontend efficiency.

The dataset features critical performance metrics such as First Contentful Paint (FCP), Largest Contentful Paint (LCP), First Input Delay (FID), Cumulative Layout Shift (CLS), and Interaction to Next Paint (INP). These metrics align closely with Google's Core Web Vitals, which serve as key indicators for assessing responsiveness and visual stability of web applications.

Since CrUX data is updated monthly, it enables continuous tracking of performance trends, identification of regressions, and assessment of the effects of UI or backend changes on user experience.

One of CrUX's major strengths lies in its ability to filter data by various dimensions, including device type (mobile, desktop, tablet), geographic region, and network quality (e.g., 3G, 4G). This flexibility allows for targeted analysis of specific user segments and helps identify performance issues affecting particular demographics. As the dataset is hosted on Google BigQuery, researchers can execute scalable SQL queries without the need for manual file handling, making it ideal for large-scale performance evaluations.

To analyze and model web application performance using Python, a variety of libraries and tools were utilized. Pandas was employed for convenient data manipulation in tabular format, while Numpy handled efficient numerical computations. For visualizing data and exploring inter-variable relationships, Matplotlib and Seaborn were applied. Regression models were developed using Scikit-learn, including *LinearRegression* for linear modeling, *PolynomialFeatures* for polynomial expansion, and *StandardScaler* for data normalization.

Accuracy evaluation was carried out using metrics such as mean absolute error (MAE), mean squared error (MSE), and the coefficient of determination

(R^2). To construct NN, TensorFlow was used – specifically, the Sequential API and Dense layers – to capture non-linear dependencies in the data and compare these findings to regression results.

TensorFlow proved beneficial due to its flexible architecture, which supports deployment on CPUs, GPUs, and TPUs, as well as mobile and embedded systems. This scalability makes it suitable for both small-scale experiments and complex production environments.

The research focused on a set of key performance indicators most relevant to evaluating web application performance. These included: *onloadtime* (total page load time including all resources), *dominteractive* (time at which the page becomes interactive), and *domcontentloadedeventend* (time after the HTML is fully parsed, before all media loads). Additional metrics involved *layoutcount* (number of layout recalculations), *jsheaptotalsize* (JavaScript memory usage), *img_initiated_transfer_size* and *css_initiated_transfer_size* (network sizes for images and CSS), and *numberofrequests* (total server requests triggered during load). To streamline processing and improve analysis efficiency, the dataset was reduced to include only the most relevant parameters. This helped focus the evaluation specifically on metrics that most significantly impact web application performance.

The selected data, stored in float64 format across 10,000 records, allowed for detailed dependency analysis and accurate regression modeling.

For model training and testing, the dataset was divided using an 80/20 split via the *train_test_split function* (with *random_state=42* for reproducibility). Training was conducted using a GPU-based setup (NVIDIA T4/P100), with 12 GB of RAM and 2 vCPUs, which significantly accelerated the learning process.

This research proposes a software-driven methodology for assessing web application performance through the use of ML techniques. It addresses both the structural components and predictive modeling of performance, aiming to detect inefficiencies and forecast issues before they negatively affect user experience.

The approach starts with selecting critical performance indicators such as response time, memory and CPU usage, network speed, full load time, time to interactivity, layout recalculations, JavaScript memory consumption, and the number of server requests. These metrics were chosen for their practical importance in evaluating real-world application behavior. To facilitate data acquisition, custom software agents and backend services were implemented, enabling continuous monitoring of application activity. After collection, the data was cleaned and prepared using Python libraries to ensure quality and consistency for analysis. The modeling process incorporated ML algorithms – including linear and polynomial regression, decision tree, and neural network – to identify anomalies and predict potential performance drops.

This methodology offers a replicable and scalable framework that integrates automated data gathering, feature extraction, and predictive analysis. It provides a robust foundation for combining machine learning with advanced web performance monitoring systems.

To simplify the task and improve the efficiency of analysis, the dataset was reduced by limiting the number of rows and selecting only the most essential parameters. This approach minimized the volume of processed data while maintaining focus on the key metrics necessary for assessing the performance of web applications. The selected data is stored in the int64 numerical format, which allows for detailed examination of relationships between variables and enables the construction of regression models to predict web application performance:

< class ‘ pandas.core.frame.DataFrame ‘>
RangeIndex: 10000 entries, 0 to 9999
dtypes : float64(8)
memory usage: 625.1 KB
onloadtime dominteractive domcontentload-edeventend layoutcount \
count 10000.000000 10000.000000 10000.000000 10000.00000
mean 4849.606800 2165.309672 2345.924978 39.31970
standard 4577.348315 1961.989792 2084.977438 76.60846
min 0.000000 0.000000 0.000000 0.00000
25% 1820.750000 950.393750 1044.675000 7.00000
50% 3501.000000 1634.337500 1779.035000 20.00000
75% 6152.500000 2761.778750 3023.257500 44.00000
max 29919.000000 29423.770000 29423.805000 2566.00000
jsheaptotalsize img_initiated_transfer_size \
count 1.000000e+04 1.000000e+04
mean 1.422299e+07 7.054487e+05
std 1.344788e+07 2.328241e+06
min 1.843200e+06 0.000000e+00
25% 5.251072e+06 5.130000e+02
50% 1.041613e+07 7.473500e+04
75% 2.032845e+07 5.419240e+05
max 1.884570e+08 8.057756e+07
css_initiated_transfer_size numberofrequests
count 1.000000e+04 10000.000000 …

The dataset was standardized and then underwent an initial correlation analysis to identify relationships between the variables (Fig. 1).

The data underwent normalization and preliminary correlation analysis. Clear correlations were found among key time-based metrics, such as *onloadtime*, *dominteractive*, and *domcontentloadedeventend*. A strong correlation was also observed between *jsheaptotalsize* and *onloadtime*, suggesting a link between JavaScript memory usage and loading time. Conversely, transfer sizes for images and CSS files showed weak correlations with performance metrics.

In this research, a combination of linear regression (LR), polynomial regression, decision tree, and neural network (NN) models was applied to evaluate web application performance. These models were chosen to capture a variety of complexities in the data.

Linear and polynomial regression serve as interpretable baselines, suitable for identifying straightforward relationships in performance metrics.

Decision trees offer a flexible, non-linear alternative capable of capturing feature interactions, while NN are ideal for modeling intricate, non-linear dependencies in high-dimensional data, offering robust performance in dynamic environments. This diverse set of models ensures a balance between interpretability and predictive power. For the linear regression model, the LR class was used to fit the training data and generate predictions. The model achieved a high R^2 score of 0.936, explaining about 93.6% of the variance.

The resulting regression equation included key predictors with both positive and negative impacts on the target variable:

$$y=2164.4+110.6\cdot x_0+1885.6\cdot x_1+(-13.4)\cdot x_2+(-9.6)\cdot x_3 +(-0.2)\cdot x_4+(-80)\cdot x_5+(-15.1)\cdot x_6, \quad (1)$$

where y represents the predicted outcome, while $x_0 \ldots x_6$ are the predictor variables contributing to the model.

The variable x_1 had the strongest positive effect, while x_3 and x_6 contributed most significantly to decreases in the predicted value, providing valuable outcomes into performance influencers.

Polynomial regression was used to capture more complex, non-linear relationships by introducing second-degree polynomial features. The model achieved a MAE of 190.91, MSE of 222,434.98, and an R^2 of 0.9336, indicating strong performance.

A decision tree model was also tested, which, while capable of modeling non-linear patterns, underperformed with a lower R^2 of 0.8628 and higher error rates, possibly due to data noise or lack of tuning.

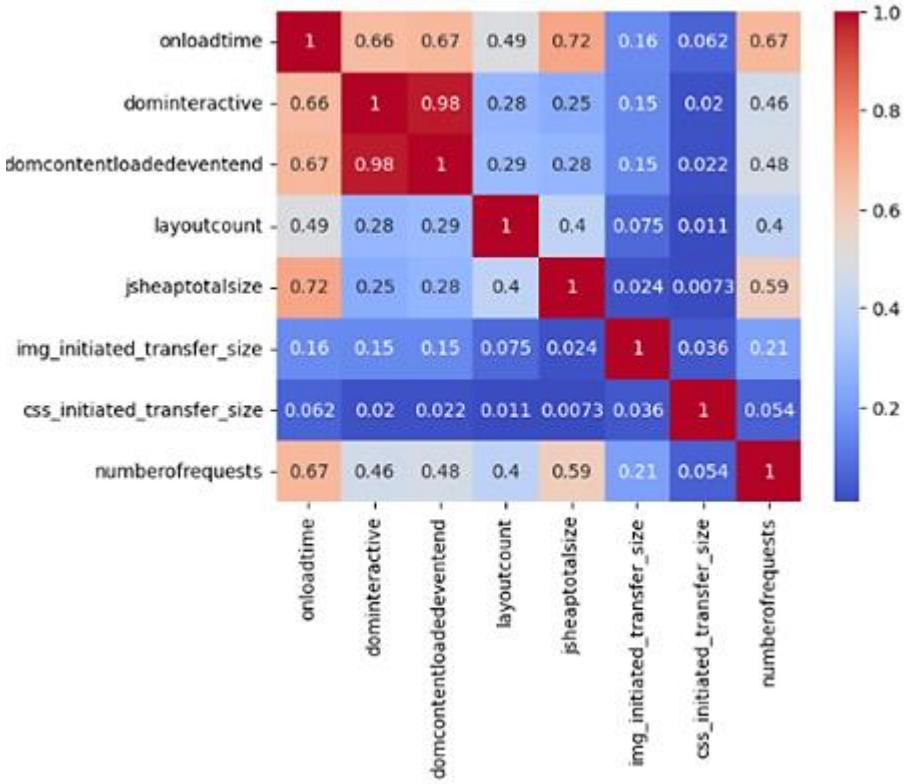

Fig. 1. The correlation between the variables

NN was designed with three dense layers (64, 32, and 1 neurons) using ReLU activation and trained for 50 epochs with the Adam optimizer. The model achieved a MAE of 191.47, MSE of 215,483.79, and an R^2 score of 0.9357, closely matching the performance of polynomial regression. These results suggest that while NN adds modeling power for complex data, polynomial regression offers similar accuracy with greater interpretability in this context.

The findings indicate that while NN offer a powerful and flexible modeling framework, they do not significantly outperform simpler regression models in this particular case. LR model demonstrated the best performance, achieving the lowest Mean Absolute Error (MAE = 187.25) and Mean Squared Error (MSE = 213,550.03), which confirms its reliability and consistency in predictions.

Proposed software architecture designed for evaluating web application performance. It outlines the full workflow – from initial data acquisition to analysis, reporting, and result visualization. The process starts with the Driver module, which communicates with the web application through the DevTools Protocol to gather performance-related data [10].

The gathered data is forwarded to the Data Analysis module, where regression techniques are applied to examine the key factors influencing web application performance. This step leads to the creation of an Analysis Report. Within this framework, regression models are utilized to assess how specific performance metrics affect the overall responsiveness of the application.

Through regression analysis, correlations between different parameters and page load efficiency are identified, and the influence of each metric on performance can be forecasted. Applying various regression approaches allows for comparison of prediction accuracy and the selection of the most suitable method for ongoing monitoring and optimization. After analysis, the refined data is directed to a Classification module, which organizes it for further interpretation. These categorized outputs – referred to as Artifacts – are sent to the Audit module, where they are transformed into JSON-formatted results.

Proposed software system compiles a comprehensive performance report covering several categories such as PWA, Performance, Accessibility, and Best Practices. It also considers multiple contextual factors including server and client-side data, user behavior, timestamps, and data source characteristics.

The analyzed results are delivered through a WebSocket connection and visualized in the Report Results Display module, enabling real-time tracking and evaluation of web application performance. For the dataset and problem under research, LR proved to be the most suitable approach due to its simplicity, high efficiency, and strong accuracy. The table 1 summarizes the performance of four ML models used for web application performance prediction.

Table 1

Experimental research results

ML model	MAE	MSE	R^2
Linear regression	187.25	213550.03	0.9363
Polynomial regression	190.91	222434.98	0.9336
Decision tree	245.80	459650.08	0.8629
Neural network	191.48	215483.79	0.9357

LR model showed the best results with the lowest MAE (187.25) and MSE (213550.03), and the highest R^2 (0.9363), indicating high accuracy and stability. Polynomial regression and the NN achieved comparable results, slightly less accurate. The Decision Tree performed the worst, with the highest errors and the lowest R^2, showing limited effectiveness for this task.

It effectively balances interpretability with predictive performance, making it an optimal choice for this analysis.The research developed a methodology for evaluating the performance of web applications, implemented as a software-based solution. This solution can be integrated into web applications to automatically analyze and monitor key performance indicators. The primary metrics selected include the total page load time (covering all resources such as images, stylesheets, and scripts), and the time at which the HTML document is fully loaded and parsed. Other resources, like images and multimedia content, may continue loading while the HTML is being processed. Additional performance indicators include the number of layout recalculations (which affect rendering efficiency), the total memory usage by JavaScript objects, the total size of images transferred over the network, the cumulative size of CSS files loaded during rendering, and the number of server requests initiated during the page load. To streamline the analysis process and enhance evaluation efficiency, the dataset was reduced by focusing solely on these critical parameters. This approach minimized the volume of data processed while maintaining emphasis on the most impactful performance metrics of the web application. The developed software solution unifies all key stages of web application performance assessment – data gathering, processing, classification, and reporting – into a cohesive and automated system. This comprehensive approach enhances the efficiency and accuracy of performance evaluation.

To boost web application performance based on the analytical outcomes, it is advised to minimize server requests by implementing data caching, consolidating multiple requests, and refining API interactions. These measures help decrease server load and enhance responsiveness. Incorporating asynchronous requests can further reduce delays and optimize resource usage. Effective monitoring of CPU and memory consumption is essential for timely scaling – either horizontally by adding more servers or vertically by upgrading existing ones.

Reducing data transfer volume through compression techniques, image and static resource optimization, and the use of Content Delivery Networks (CDNs) can significantly accelerate page load times. Optimizing code, particularly by simplifying SQL queries and computational algorithms, can also help avoid unnecessary resource strain. Unlike conventional methods that often depend on manual checks or isolated performance indicators, this solution leverages ML to automatically identify and predict performance bottlenecks. It delivers real-time, data-driven outcomes and offers practical recommendations such as caching improvements, asynchronous communication, API optimization, and dynamic scaling – resulting in a smarter and more proactive performance management strategy. A software architecture is proposed for intelligent analysis of JavaScript web application performance. It collects numerical data, applies regression analysis to assess key factors. The software offers recommendations to developers for optimizing performance and reducing server load.

Conclusions. Applying ML techniques to evaluate the performance of web applications enhances the precision of assessments and enables efficient processing of extensive datasets. Regression models are instrumental in uncovering key performance trends, while deep NN allow for forecasting outcomes under dynamic and complex scenarios. The proposed software solution streamlines the entire evaluation process by automating data collection, analysis, reporting, and live monitoring. Data is collected through browser DevTools, analyzed using regression algorithms, and presented in real-time via WebSocket through a performance dashboard.

Optimization recommendations derived from this system include minimizing the number of server requests, utilizing asynchronous operations, implementing resource scaling, compressing transmitted data, and balancing loads across infrastructure. This comprehensive and predictive methodology not only increases the accuracy and speed of detecting performance bottlenecks but also supports timely backend and frontend optimization efforts, improving application responsiveness and scalability.

Future work will focus on enhancing neural network efficiency to speed up training processes and integrating ML with advanced performance monitoring tools for more insightful and automated diagnostics. Techniques such as transfer learning or dynamic learning rate adjustments can significantly shorten training durations. Coupling ML models with platforms like Google Lighthouse, New Relic, or Datadog could provide intelligent real-time feedback and adaptive system tuning. An additional direction involves developing hybrid models that merge statistical analysis with ML-based anomaly detection for deeper and more context-aware outcomes. For instance, integrating time-series analysis of server latency with NN capable of spotting unusual memory usage patterns could help flag early warning signs of overloads or memory leaks. Such preemptive detection facilitates proactive optimization, boosting both the reliability and performance of web systems while reducing operational costs by mitigating risks before they escalate.

Bibliography:

1. Anis Z. A systematic review on regression testing for web-based applications. *Journal of Software*. 2015. Vol. 10 (8). P. 971–990. DOI: 10.17706/jsw.10.8.971-990.

2. Ghattas M., Mora A. M., Odeh S. A novel approach for evaluating web page performance based on machine learning algorithms and optimization algorithms. *AI*. 2025. Vol. 6 (2). Article 19. DOI: 10.3390/ai6020019.

3. Thompson H. S. Improved methodology for longitudinal web analytics using Common Crawl. *Communications of the ACM*. 2024. Vol. 27 (6). P. 929–948. DOI: 10.1145/3614419.3644018.

4. Hort M., Kechagia M., Sarro F., Harman M. A survey of performance optimization for mobile applications. *IEEE Transactions on Software Engineering*. 2022. Vol. 48 (8). P. 2879–2904. DOI: 10.1109/TSE.2021.3071193.

5. Lun L., Zetian D., Hoe T. W., Juan X., Jiaxin D., Fulai W. Factors influencing user intentions on interactive websites: Insights from the technology acceptance model. *IEEE Access*. 2024. Vol. 12. P. 122735–122756. DOI: 10.1109/ACCESS.2024.3437418.

6. Shailesh S., Suresh P. V. A survey and analysis of techniques and tools for web performance optimization. *Journal of Information Organization*. 2018. Vol. 8 (2). P. 31–57. DOI: 10.6025/jio/2018/8/2/31-57.

7. Leone J., Traini L. Enhancing trace visualizations for microservices performance analysis. *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering*. 2023. P. 283–287. DOI: 10.1145/3578245.3584729.

8. Willnecker F., Andreas B., Wolfgang G., Helmut K. Using Dynatrace monitoring data for generating performance models of Java EE applications. *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*. 2015. P. 103–104. DOI: 10.1145/2668930.2688061.

9. Oleshchenko L., Burchak P. Web application state management performance optimization methods. *Lecture Notes on Data Engineering and Communications Technologies*. 2023. Vol. 181. P. 59–74. Springer, Cham. DOI: 10.1007/978-3-031-36118-0_6.

10. Oleshchenko L.M., Burchak P.V. Software system architecture development for intelligent analysis of web application performance metrics. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки*. Том 35 (74). № 4. 2024. С. 141-150. https://doi.org/10.32782/2663-5941/2024.4/22.

Бурчак П.В., Олещенко Л.М. ПРОГРАМНИЙ ФРЕЙМВОРК МАШИННОГО НАВЧАННЯ ДЛЯ ПРОГНОЗУВАННЯ ТА ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ ВЕБЗАСТОСУНКІВ

У статті представлено програмний фреймворк для аналізу та прогнозування продуктивності вебзастосунків з використанням методів машинного навчання. Замість традиційного підходу до тестування продуктивності було запропоновано автоматизований конвеєр, який здійснює збір, обробку та моделювання ключових системних метрик – таких як час відповіді сервера, використання пам'яті, навантаження на процесор і чутливість інтерфейсу з метою проактивного виявлення та усунення проблем продуктивності. Для дослідження було використано набір даних із понад 200 000 реальних записів, що включають показники часу завантаження вебсторінки, перерахунку макета, споживання пам'яті JavaScript та пропускної здатності мережі.

У моделювальний конвеєр було інтегровано декілька алгоритмів машинного навчання, зокрема, лінійну регресію, поліноміальну регресію, дерева рішень та нейронну мережу для оцінювання їхньої ефективності у прогнозуванні продуктивності застосунків. За результатами експериментів найкращу точність показала лінійна регресія (MAE = 187,25, R^2 = 0,9357), що свідчить про сильний лінійний зв'язок у даних. Нейронна мережа також продемонструвала високу точність і переваги при моделюванні динамічних нелінійних залежностей. Розроблена програмна система реалізована з використанням технологій Python: Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn і TensorFlow.

У результаті проведеного дослідження було створено масштабоване, автоматизоване та інтерпретоване програмне рішення для прогнозної діагностики та оптимізації продуктивності вебзастосунків. Запропонована програмна система також дозволяє формувати категоризовані звіти з рекомендаціями щодо покращення продуктивності вебзастосунків. Отримані результати підтверджують ефективність і доцільність використання методів машинного навчання в сучасних системах моніторингу продуктивності вебзастосунків.

Ключові слова: *програмний фреймворк, моніторинг продуктивності програмного забезпечення, машинне навчання, прогнозна аналітика, оптимізація вебсистем, регресійні моделі, нейронні мережі, Python, TensorFlow, прогнозування продуктивності вебзастосунків.*

Дата надходження статті: 17.07.2025
Дата прийняття статті: 24.07.2025
Опубліковано: 27.10.2025