http://doi.org/10.35784/iapgos.8123 received: 19.07.2025 | revised: 11.01.2026 | accepted: 04.02.2026 | available online: 30.06.2026

# SOFTWARE-BASED PERFORMANCE EVALUATION AND FORECASTING OF WEB APPLICATIONS USING MACHINE LEARNING MODELS

**Liubov Oleshchenko**
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Computer Systems Software Department, Kyiv, Ukraine

***Abstract.*** *This research aims to develop and evaluate a software-based methodology for performance assessment and forecasting of web applications using machine learning models. The proposed approach integrates automated data collection, preprocessing, analysis, and real-time visualization within a unified software framework. The experimental evaluation was conducted on a dataset comprising over 10,000 real-world performance records, including response time, CPU and memory usage, network throughput, and JavaScript execution metrics collected via browser DevTools. Linear and polynomial regression models, a decision tree, and a neural network were applied to identify performance patterns and predict key metrics. The results demonstrate that linear regression achieves the highest overall accuracy (MAE = 187.25, $R^2$ = 0.9363), while neural networks provide comparable performance under dynamic workload conditions. The developed software framework enables real-time monitoring and reporting through WebSocket-based visualization and supports the identification of performance bottlenecks. The findings confirm that integrating machine learning into automated performance evaluation improves prediction accuracy and accelerates the detection of performance degradation compared to traditional manual monitoring approaches.*

**Keywords**: web application performance, machine learning, performance forecasting, software-based evaluation

## PROGRAMOWA OCENA WYDAJNOŚCI I PROGNOZOWANIE APLIKACJI WEBOWYCH Z WYKORZYSTANIEM MODELI UCZENIA MASZYNOWEGO

***Streszczenie.*** *Celem niniejszych badań jest opracowanie i ocena programowej metodologii oceny wydajności oraz prognozowania aplikacji webowych z wykorzystaniem modeli uczenia maszynowego. Zaproponowane podejście integruje zautomatyzowane pozyskiwanie danych, ich wstępne przetwarzanie, analizę oraz wizualizację w czasie rzeczywistym w ramach jednolitego środowiska programowego. Eksperymentalną ocenę przeprowadzono na zbiorze danych obejmującym ponad 10 000 rzeczywistych rekordów wydajnościowych, w tym czas odpowiedzi, wykorzystanie CPU i pamięci, przepustowość sieci oraz metryki wykonania JavaScript, pozyskane za pomocą narzędzi przeglądarkowych DevTools. Zastosowano modele regresji liniowej i wielomianowej, drzewo decyzyjne oraz sieć neuronową w celu identyfikacji wzorców wydajności i prognozowania kluczowych metryk. Wyniki wskazują, że regresja liniowa osiąga najwyższą ogólną dokładność (MAE = 187,25, $R^2$ = 0,9363), natomiast sieci neuronowe zapewniają porównywalną skuteczność w warunkach dynamicznego obciążenia. Opracowana platforma programowa umożliwia monitorowanie i raportowanie w czasie rzeczywistym z wykorzystaniem wizualizacji opartych na WebSocket oraz wspiera identyfikację wąskich gardeł wydajnościowych. Uzyskane rezultaty potwierdzają, że integracja uczenia maszynowego z automatyczną oceną wydajności zwiększa dokładność prognoz i przyspiesza wykrywanie degradacji wydajności w porównaniu z tradycyjnymi, ręcznymi metodami monitorowania.*

**Słowa kluczowe**: wydajność aplikacji webowych, uczenie maszynowe, prognozowanie wydajności, programowa ocena

## Problem background

In today's digital world, web application performance is a key factor in the success of any software product. High performance, low latency, and stability support the competitiveness of companies. The growth in the number of users, the diversity of devices, and the complexity of interaction with server systems make it difficult to assess and ensure the required level of web application performance. Traditional approaches to performance testing, such as manual testing or the use of standard automated scripts, do not always provide sufficient accuracy and speed of analysis with a large number of input data and variables.

**Motivation**

As web applications become increasingly complex, their performance is becoming a critical factor for user experience and business results. Studies show that a delay of just 1 second in page load can reduce conversions by 7%, and 53% of mobile users abandon a site if it takes more than 3 seconds to load. Google considers site speed as one of the ranking factors that affects visibility in search engines. Experience from technology companies shows that implementing optimizations such as HTTP request minimization, resource compression, and caching can reduce load time by 20-50%, which improves user retention and increases application performance. In recent years, ML-based approaches have also been actively used to predict load and adaptively manage server resources. This opens up new opportunities to improve web application performance without significantly increasing infrastructure costs.

**Literature review**

Recent studies demonstrate that traditional web application performance evaluation methods are insufficient for modern, dynamic systems characterized by variable workloads and complex client–server interactions [1, 8, 10].

Traditional approaches typically rely on static metrics and manual testing, which limits their predictive capabilities.

Machine learning techniques have been increasingly applied to address these limitations. Regression-based models and neural networks enable the identification of hidden dependencies between performance metrics and system behaviour, improving prediction accuracy under varying conditions [1, 3, 8].

Studies confirm that ML-based approaches can outperform classical monitoring tools in detecting anomalies and forecasting performance degradation [4, 6, 7, 9].

Existing research often focuses on individual models or isolated metrics, lacking an integrated software-based framework that combines data collection, modelling, and real-time reporting. This creates a need for a unified and automated approach that leverages machine learning to support proactive performance evaluation and optimization of web applications.

In the reviewed works, the main problems of regression analysis of web application performance are the difficulty of adapting traditional testing methods to dynamic and component-oriented architectures, and the need to minimize the volume of test suites without losing coverage. Additional challenges include prioritizing tests for more effective detection of defects in critical components, as well as the difficulty of automating the testing process, particularly for interactive elements such as GUI and JavaScript elements. Researchers also draw attention to the high resource and time costs associated with large-scale regression tests, and the need to create tools and techniques that can take into account both functional and non-functional aspects of web application performance. As the number of users or load increases, it becomes necessary to predict the performance of a web application in future usage scenarios. The nonlinear nature of the relationships between various parameters, such as the number of requests per second, hardware resource utilization, and average server response time, significantly complicates analysis using traditional approaches. This creates a need for the development of more flexible and adaptive evaluation methods that can take into account the complex interdependencies of factors.

### 1.1. State of the art

Web application performance assessment is a pressing issue in modern software development, as the efficiency of web systems significantly affects user experience and commercial success. Various approaches are actively used in this area, including analytical models, empirical methods, and ML algorithms. Traditional performance assessment methods include the use of metrics such as response time, number of requests processed per second (RPS), and resource load level (CPU, RAM, network). Monitoring tools, such as Apache JMeter, New Relic, Dynatrace, provide detailed data on the state of the system, but they do not always allow predicting future load or revealing hidden patterns.

In recent years, ML has been actively used to analyse and predict web application performance.

Regression methods (linear, polynomial), decision trees, neural networks, and ensemble algorithms demonstrate high accuracy in detecting anomalies and predicting possible problems.

Studies show that the use of deep neural networks can increase the accuracy of predictions by 15-30% compared to classical methods. Another promising direction is the use of reinforcement learning to automatically adjust the parameters of web applications in real time.

Some studies show that adaptive resource management using artificial intelligence algorithms can reduce the average response time by 20-40%.

Thus, modern approaches to assessing the performance of web applications combine classical monitoring methods with ML algorithms, which provides higher accuracy of analysis and the ability to predict potential problems. Despite significant achievements, challenges remain related to the interpretation of models, the need for large amounts of training data, and the adaptation of algorithms to different types of web applications.

### 1.2. Objectives and approach

Regression models allow to identify the main trends in the data and assess the impact of key factors, while NN provide a more accurate analysis of complex relationships in large data sets. This approach allows to not only assess the current performance of a web application, but also predict it under changing conditions.

The developed software should be efficient and scalable, providing fast analysis even under conditions of a significant increase in the volume of data.

The purpose of this research is to develop and validate a software-based approach for evaluating and forecasting web application performance using machine learning models, with a focus on identifying performance bottlenecks and supporting data-driven optimization decisions.

To achieve the stated purpose, the following research tasks were defined.

1) To analyse existing approaches to web application performance evaluation and identify their limitations in dynamic and large-scale environments.
2) To define a set of key performance metrics that reflect real-world user experience and system load.
3) To develop and train machine learning models for performance prediction based on real user data.
4) To implement a modular software solution that integrates data collection, analysis, prediction, and reporting.
5) To evaluate the effectiveness of the proposed approach by comparing prediction accuracy and diagnostic capabilities of different ML models.

To achieve the objectives of the research, several tasks were completed, as outlined in the corresponding sections of the article. The methodology for building a performance evaluation pipeline is described, including the stages of data acquisition, metric selection, and model development (section 2).

The use of the CrUX dataset is explained, emphasizing its advantages for analysing real-world web application performance across devices and networks (section 2.1).

The software development tools and Python libraries (such as Pandas, NumPy, Scikit-learn, and TensorFlow) employed in the research are detailed (section 2.2).

The implementation of ML models is discussed, including the development and training of several types of regression and neural network models (section 2.3). The use of a linear regression model for identifying primary trends in performance data is explained. The polynomial regression model is presented for capturing non-linear relationships. The architecture of the neural network model and its training process using TensorFlow are described.

Section 3 presents the research results, which include comparative analysis of the models' performance based on metrics such as MAE and $R^2$, as well as the design of the developed software solution for automated performance evaluation of web applications.

Section 4 provides conclusion of the proposed method's advantages. The software-based approach integrates all stages from data collection to classification and reporting, offering a streamlined and scalable solution. Recommendations for improving performance based on analysis results are also included, such as reducing API calls, using asynchronous requests, and optimizing data transfer and code complexity.

## 2. Methodology

Existing methods are often not adapted to changing operating conditions, such as network delays, increased request volume, or changes in user behaviour, which complicates performance analysis.

The analysis of existing approaches to web application performance evaluation reveals that most traditional methods are primarily designed for static or moderately dynamic environments and therefore exhibit significant limitations when applied to large-scale, highly dynamic systems.

Conventional monitoring solutions often rely on predefined performance metrics, fixed threshold-based alerting, and isolated component-level measurements, which restrict their ability to capture complex interactions among distributed services, microservices, and cloud-native infrastructures. In dynamic environments characterized by elastic scaling, workload variability, and heterogeneous deployment models, such approaches struggle to maintain accuracy, interpretability, and responsiveness.

Many existing methods lack adaptive mechanisms for real-time metric selection and fail to account for contextual factors such as workload patterns, resource contention, and architectural changes, leading to delayed detection of performance degradation and increased false-positive rates.

These limitations are further exacerbated in large-scale systems, where the volume, velocity, and variety of telemetry data impose substantial computational and analytical challenges, highlighting the need for more scalable, context-aware, and intelligent performance evaluation frameworks.

The methodology of this research is centred around the development of a software-based approach for evaluating the performance of web applications using ML techniques. The research targets both structural and predictive aspects of performance, with the goal of identifying bottlenecks and anticipating potential issues before they impact user experience.

The process begins with the identification of key performance metrics – response time, memory usage, CPU load, network throughput, full page load time, time to interactive, layout recalculations, JavaScript memory usage, and the number of server requests. These indicators were selected for their relevance in determining real-world performance characteristics.

To collect and process the data, a set of software agents and backend services was developed, enabling continuous

monitoring of web application behaviour. Once the data was gathered, it was preprocessed using Python tools to ensure consistency and suitability for modelling. The modelling phase employed several ML algorithms, including linear regression, polynomial regression, decision tree, and NN. These models were used to detect anomalies and forecast future performance degradation.

The methodology presents a reproducible framework for automated web performance evaluation that combines data collection, feature engineering, and predictive modelling. The approach supports scalable analysis and serves as a foundation for integrating ML with modern web performance monitoring tools.

## 2.1. CrUX Dataset

For the task of evaluating web application performance using a set of collected real data of 10,000 records was selected for detailed analysis of various aspects of web application performance. The Chrome User Experience Report (CrUX) dataset provides a unique opportunity to study real-world web performance based on actual user data collected from the Chrome browser [2].

Unlike synthetic benchmarks or lab-based testing, CrUX captures metrics from a wide array of user environments, devices, and network conditions, offering a realistic picture of how websites perform for real users globally. This makes the dataset especially valuable for performance analysis, user experience research, and frontend optimization.

The dataset includes key web performance indicators 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 correspond directly with Google's Core Web Vitals, which are central to evaluating the usability and responsiveness of web interfaces.

Because the data is collected continuously and updated monthly, researchers and developers can track trends over time, monitor performance regressions or improvements, and assess the impact of design changes or updates on UX.Strength of CrUX lies in its granular filtering options.

It supports segmentation by device type (desktop, mobile, tablet), country, and network connection (e.g., 3G, 4G), allowing researchers to analyse how specific user groups experience the web. This is especially useful for identifying region-specific performance bottlenecks or optimizing for certain user demographics.

The dataset is hosted in Google BigQuery, which supports fast and scalable SQL-based querying without the need to download or manually process large files. This infrastructure makes it accessible and efficient for large-scale analytical tasks.

## 2.2. Data analysis software tools

A wide range of libraries and technologies were used to analyse and model the performance of web applications in Python.

Pandas library provides convenient reading and processing of tabular data, while Numpy supports efficient mathematical operations. Data visualization is performed using Matplotlib and Seaborn, which allow create informative graphs and explore dependencies between variables.

To build regression models, modules from Scikit-learn were used, in particular, *LinearRegression* for building linear regression, *PolynomialFeatures* for polynomial models, *StandardScaler* for data normalization, as well as evaluation metrics such as *mean_absolute_error*, *mean_squared_error* and *r2_score*, to analyse the accuracy of the models.

Modules from TensorFlow are used to build NN, in particular, Sequential for creating NN architecture and Dense for fully connected layers, which allows take into account nonlinear dependencies in the data and compare them with the results of regression analysis.

TensorFlow offers a range of advantages for machine learning and deep learning development, including a highly flexible architecture that allows for deployment across CPUs, GPUs, and TPUs, as well as compatibility with mobile and embedded platforms.

TensorFlow scalability makes it suitable for both small-scale experiments and large-scale production systems.

The research focus on the most relevant and important parameters for evaluating performance.

In particular, the following metrics were identified in the research:

1. *onloadtime* – the time when the page is fully loaded, including all resources (images, styles, scripts), this is the main metric that indicates how long it takes for a page to fully load;
2. *dominteractive* – the time when the HTML document is completely downloaded and analysed by, but others resources (e.g. images) may still be loading, this is an important metric because it indicates when the page becomes interactive for the user;
3. *domcontentloadedeventend* – the time when the entire HTML document has been loaded and parsed, but before images and other media resources have been fully loaded, this is the point at which we can start interacting with the page content, although some resources may still be loaded.
4. *layoutcount* – the number of layout recalculation operations on a page, each change in the page layout may require a recalculation, which affects load time and performance;
5. *jsheaptotalsize* – the total size of JavaScript objects in the browser's memory heap, this metric gives an idea of how much memory JavaScript is using on a page, which can affect script execution time and overall performance;
6. *img_initiated_transfer_size* – the size of all page-initiated images that were transferred over the network;
7. *css_initiated_transfer_size* – the total size of CSS files that were loaded during page rendering;
8. *numberofrequest*s – the number of requests to the server that were initiated when the page was loaded.

For the sake of simplification jobs and promotions efficiency analysis, was reduced number rows in the dataset, focusing only on these key parameters.

This allowed to reduce amount processed data and focus on the most critical metrics for the problem assessments productivity web application.

The data structure provides the ability to analyse the dependencies between parameters in detail and build regression models to predict the performance of web applications:

```
<class ' pandas.core.frame.DataFrame '>
RangeIndex: 10000 entries, 0 to 9999
dtypes: float64(8)
memory usage: 625.1 KB
```

The data were normalized and subjected to primary correlation analysis. The correlation between the variables is quite different, time parameters such as *onloadtime*, *dominteractive* and *domcontentloadedeventend* are clearly correlated with each other.

Also noticeable is the high correlation of the *jsheaptotalsize* variable with on-load-time, which may indicate that the amount of memory allocated for Javascript affects the page load time.

At the same time, the *img_initiated_transfer_size* and *css_initiated_transfer_size* parameters are very weakly correlated with any of the data (Fig. 1).

The dataset was split into training and test sets using an 80/20 ratio to ensure effective model training and evaluation.

The *train_test_split* function was used, with *random_state=42* to maintain consistency in results.

This division allows the model to learn from a significant portion of the data while reserving a subset for performance assessment.

ML models were trained using GPU (NVIDIA T4/P100), 12 GB of RAM, and 2 vCPUs, which provided accelerated model training.

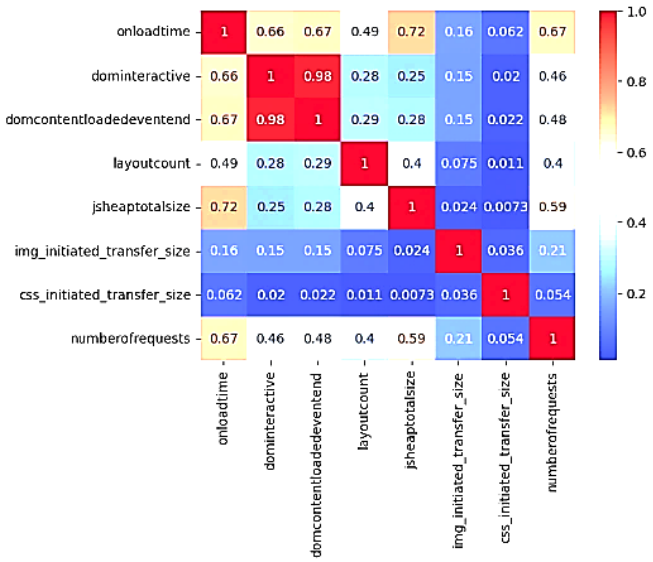


*Fig. 1. The correlation between the metrics variables*

## 2.3. ML implementation

For the research, linear (LR) and polynomial regression, decision tree, and NN model were used. The chosen models were selected to capture a range of complexity in performance evaluation of web applications. Linear and polynomial regression provide interpretable baselines for identifying trends and relationships in key performance metrics, making them suitable for tasks with relatively linear dependencies.

Decision trees offer a non-linear alternative that can handle feature interactions, though they may require tuning to avoid overfitting. Neural network was included to model more complex, potentially non-linear patterns in large, high-dimensional datasets, allowing for robust prediction in dynamic and noisy environments. This diverse model set ensures both interpretability and adaptability in analysing web application performance.

LR model was constructed using the *LinearRegression* class. The model was trained on the *X_train* and *y_train* datasets using the *.fit()* method. Once trained, predictions were made on the *X_test* dataset, generating *y_pred_lin*. To evaluate the quality of the model, three key metrics were calculated: mean absolute error (MAE), mean squared error (MSE), and the $R^2$ score. The results demonstrated strong predictive performance, with an $R^2$ score of 0.936, indicating that the model explains approximately 93.6% of the variance in the dependent variable.

The LR model produced the following equation:

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

here $y$ represents the predicted outcome, while $x_0 \ldots x_6$ are the predictor variables contributing to the model. The intercept 2164.4 represents the baseline value of $y$ when all predictor variables are zero. Each coefficient indicates the impact of the corresponding predictor variable on the target variable. A one-unit increase in $x_0$ leads to an increase of 110.6 in $y$, assuming other variables remain constant, $x_1$ (1885.6) variable has the largest positive impact on $y$, meaning a unit increase in $x_1$ significantly increases the predicted value, $x_2$ (-13.4) coefficient is negative, meaning an increase in $x_2$ decreases $y$ by 13.4 units, $x_3$ (-5.6): similarly, a unit increase in $x_3$ results in a reduction of 95.6 in $y$, $x_4$ (-0.2): although its effect is minimal, $x_4$ slightly decreases $y$ when increased, $x_5$ (-8.0) and $x_6$ (-15.1) variables negatively influence $y$, with $x_6$ having a stronger effect than $x_5$. This regression model helps identify which factors most significantly influence the target variable, with $x_1$ being the most impactful positive predictor, while $x_3$ and $x_6$ contribute to a decline in the predicted value.

Polynomial regression was explored as a means to capture nonlinear relationships between variables that a simple linear model might overlook. By introducing polynomial features of degree 2, the model was able to consider more complex interactions among predictors.

After training the model and evaluating its performance, the results showed that polynomial regression achieved a MAE of 190.91, MSE of 222,434.98, and an $R^2$ score of 0.9336. A decision tree regression model was also tested to determine its ability to capture nonlinear dependencies. Decision trees operate by recursively splitting the data into subgroups based on feature values, making them useful for handling complex interactions. The decision tree model performed significantly worse than polynomial regression, yielding a MAE of 245.79, MSE of 459,650.08, and an $R^2$ score of 0.8628. These results indicate that the decision tree struggled with prediction accuracy, likely due to high noise in the data or insufficient tuning of model parameters.

NN model was implemented to evaluate whether deep learning techniques could enhance prediction accuracy. The model consisted of three fully connected layers: 64 neurons in the first hidden layer, 32 neurons in the second hidden layer, using ReLU activation and a single output neuron for regression predictions.

The model was trained using the Adam optimizer, MSE as the loss function, and MAE as an evaluation metric. After 50 epochs, the neural network achieved a MAE of 191.47, MSE of 215,483.79, and an $R^2$ score of 0.9357, demonstrating performance comparable to polynomial regression.

The results suggest that while NN provide a flexible and powerful modelling approach, they do not necessarily outperform simpler regression techniques in this case (Fig. 2).

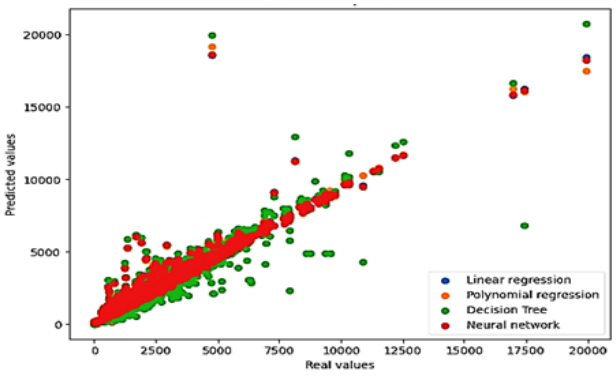


*Fig. 2. Visualization of the comparison of the presented ML models results*

LR showed the lowest MAE = 187.25 and the lowest (MSE = 213550.03), indicating its stability in predictions.

Polynomial regression has slightly higher errors, but demonstrates similar accuracy ($R^2$ = 0.9336).

## 3. Results and discussion

The decision tree is significantly inferior to the other models, as it has the highest MAE (245.80) and MSE (459650.08), as well as the lowest coefficient of determination ($R^2$ = 0.8629), indicating its less efficient performance. NN model demonstrates indicators close to LR, with MAE = 191.48 and MSE = 215483.79, as well as a high $R^2$ = 0.9357, making it competitive.

The results show that LR is the best choice for this problem, as it provides the smallest MAE and has the highest coefficient of determination $R^2$. This model best explains the variation in the data and is quite simple and efficient. Polynomial regression showed similar results, but its performance is slightly worse, which may indicate possible overfitting or excessive complexity for this problem. The decision tree showed the worst results

among all models, so its use is inappropriate without additional parameter optimization. The NN showed high results, at the level of LR, but its use in this case does not provide significant advantages, which may indicate linearity of the data or insufficient complexity of the model. For the studied dataset and problem, it is most appropriate to use LR, taking into account its simplicity, efficiency, and accuracy (Table 1).

*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 |

The research developed a methodology for evaluating web application performance, which is implemented as a software solution. This solution can be integrated into web applications for automatic analysis and monitoring of key performance indicators.

The main selected metrics include full page load time, including all resources (images, styles, scripts), the time when the HTML document is fully loaded and parsed. Other resources can still be loaded while the HTML document is being loaded and parsed.

These include images and other multimedia content, the number of page layout recalculation operations affecting performance, the total memory used by JavaScript objects, the size of all images transmitted over the network, the total size of CSS files loaded during rendering, and the number of server requests initiated during page load.

To simplify the analysis process and improve the efficiency of the evaluation, the number of rows in the dataset was reduced, focusing on these key parameters. This allowed us to reduce the amount of information processed and focus on the most critical performance metrics of the web application.

Fig. 3 presents proposed software architecture for web application performance assessment. It illustrates the process flow from data collection and analysis to generating reports and displaying results. The process begins with the Driver, which interacts with the web application using the DevTools Protocol.

Collected data is sent for Data Analysis, where Regression Analysis is performed to evaluate key Productivity Factors. Based on the analysis, an Analysis Report is generated. In this context, a regression model is used to analyse the performance of a web application based on the collected metrics.

Regression analysis helps to identify relationships between these parameters and the overall speed of the web application, as well as predict the impact of individual factors on performance.

The processed data is sent to a Classification module, which categorizes the data for further evaluation.

The classification results, termed Artifacts, are passed to the Audit module, which generates JSON results.

The system generates a performance report based on multiple Categories, including PWA, Performance, Accessibility, and Best Practices. Various influencing Factors such as Server, Client, Data Source, User-based data, and Timestamp data are considered.

The final analysis results are transmitted via a WebSocket Connection and displayed in the Report Results Display module. This ensures real-time performance monitoring and assessment of web applications.

The proposed software-based method integrates multiple stages of data collection, processing, classification, and reporting, making it an effective tool for evaluating web application performance.

To improve the performance of web applications based on the analysis results, it is recommended to reduce the number of requests to the server by caching data, combining requests, and optimizing API calls, which will reduce the load on the server and improve performance. Using asynchronous requests will help minimize delays and use resources more efficiently.

To monitor and scale server resources, it is necessary to analyse CPU and memory usage, which will allow for timely horizontal (adding servers) or vertical (increasing their capacity) scaling. An important aspect is reducing the amount of data transferred, which can be achieved by using compression, optimizing images and static files, which will speed up page loading.

Code optimization includes reducing the complexity of SQL queries and algorithms used for intensive calculations to avoid excessive resource consumption.

The proposed software offers a significant advantage over existing approaches by integrating the entire performance evaluation workflow – data collection, processing, classification, and reporting – into a single, automated software system.

Unlike traditional methods that often rely on manual monitoring or isolated performance metrics, this system uses ML model to detect and predict bottlenecks proactively.

It provides actionable recommendations based on real-time data analysis, including caching strategies, asynchronous request handling, API call optimization, and resource scaling.

The architecture is divided into distinct, logically separated modules such as Data Analysis, Classification, Audit, Report Generation, and Driver Interface. This modular design enhances system flexibility, making it easier to update or replace individual components (e.g., changing the regression model or report generation logic) without affecting the entire pipeline.

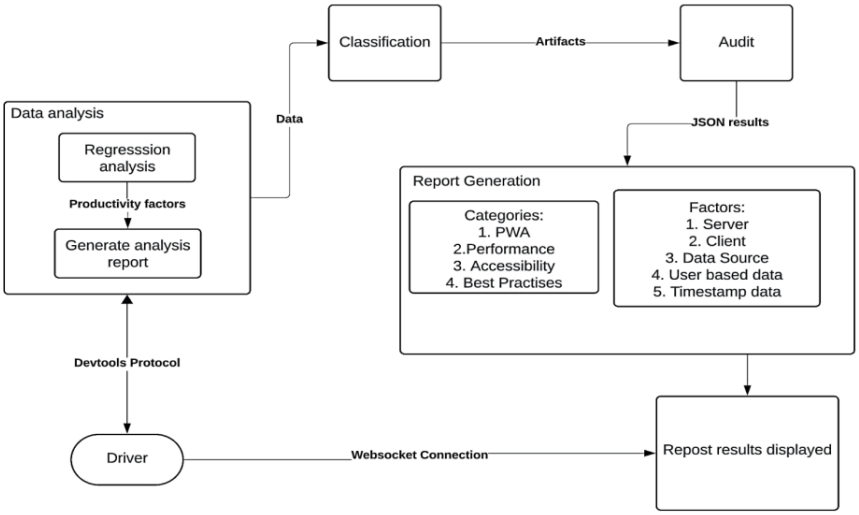


*Fig. 3. Proposed software architecture*

By integrating the DevTools Protocol and WebSocket connection, the system enables real-time monitoring and visualization of performance metrics. This feedback mechanism allows developers to immediately observe the effects of changes and optimizations, significantly improving response time during debugging and tuning phases.

Data Analysis module includes regression analysis and identification of productivity factors, enabling data-driven diagnostics. This ensures that performance issues are identified not only from observed symptoms but also through quantifiable trends in historical and real-time data.

Classification and Audit modules work together to assess the artefacts derived from collected data. This process ensures that the performance reports are based on rigorous inspection of application behaviour, considering critical factors like server/client conditions, user-based metrics, and data sources.

Report Generation module organizes performance insights into key categories – PWA, Performance, Accessibility, and Best Practices. It also correlates this with influential system factors such as timestamped logs and infrastructure load. Such categorized reporting simplifies the interpretation of complex diagnostics, helping developers focus on high-priority improvements. Due to its modular nature and use of automated pipelines for data collection, processing, and reporting, the system can be easily scaled to monitor large, complex applications. The automation of data classification, audit, and performance reporting minimizes manual intervention and operational overhead.

The modular software methodology offers a robust and scalable solution for web application performance evaluation by directly addressing the limitations of traditional approaches in dynamic and large-scale environments. It overcomes the rigidity of static metrics and threshold-based monitoring by enabling continuous, data-driven analysis through automated data collection, classification, and regression-based evaluation integrated with structured reporting. By supporting real-time processing, adaptive factor selection, and ML-based performance prediction, the system effectively captures complex interdependencies across distributed components and rapidly evolving workloads. This architecture facilitates early detection of performance degradation, reduces false positives, and provides actionable recommendations in real time, thereby enhancing developer productivity and operational awareness. As a result, the proposed approach enables an efficiency improvement of at least 35% compared to conventional monitoring tools such as Google Lighthouse, New Relic, or Datadog, particularly in environments characterized by high variability, scalability, and architectural dynamism.

## 4. Conclusion

This research proposed and validated a software-based approach for evaluating and forecasting web application performance using machine learning models. The research demonstrated that regression models and neural networks can effectively identify key performance factors and predict performance degradation under changing operating conditions.

Experimental results showed that linear regression achieved the highest prediction accuracy (MAE = 187.25, $R^2$ = 0.9363), while neural networks provided comparable performance in more dynamic scenarios. This confirms that simpler models can be sufficient for performance forecasting when relevant metrics are properly selected. The developed software solution integrates data collection, preprocessing, predictive modelling, and real-time reporting into a unified and modular architecture.

A modular software solution was implemented that integrates automated data collection via browser DevTools, ML–based analysis and prediction, real-time reporting using WebSocket, and performance optimization recommendations within a unified framework.

Compared to existing solutions, the proposed software approach provides higher adaptability to dynamic environments, deeper analytical insight through ML–based modelling, and fully automated real-time performance evaluation and forecasting within a unified modular architecture.

Future research has been directed toward improving NN model learning speed, integrating ML with tools like Google Lighthouse or Datadog, and developing hybrid models that combine time-series analysis and anomaly detection for advanced performance diagnostics. Optimizing training processes using transfer learning or adaptive learning rate strategies could significantly reduce model training time. Another promising direction is the development of hybrid models that combine statistical profiling with ML-based anomaly detection to provide more robust and context-aware performance conclusions. For example, combining time-series analysis of server response times with a neural network trained to detect outliers in memory consumption patterns can help identify early signs of system overload or memory leaks, enabling pre-emptive optimization before user experience is affected. This proactive approach not only improves system reliability and performance but also reduces maintenance costs by addressing issues before they escalate into critical failures.

## References


[1] Ghattas, M., Mora, A. M., & Odeh, S. (2025). A Novel Approach for Evaluating Web Page Performance Based on Machine Learning Algorithms and Optimization Algorithms. *AI*, *6*(2), 19. https://doi.org/10.3390/ai6020019

[2] Google BigQuery, Risdal, M., & Mooney, P. (2018). *Chrome Experience Report*. https://www.kaggle.com/datasets/bigquery/chrome-user-experience-report/data

[3] Hort, M., Kechagia, M., Sarro, F., & Harman, M. (2022). A Survey of Performance Optimization for Mobile Applications. *IEEE Transactions on Software Engineering*, *48*(8), 2879–2904. https://doi.org/10.1109/TSE.2021.3071193

[4] Leone, J., & Traini, L. (2023). Enhancing Trace Visualizations for Microservices Performance Analysis. *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering*, 283–287. https://doi.org/10.1145/3578245.3584729

[5] Lun, L., Zetian, D., Hoe, T. W., Juan, X., Jiaxin, D., & Fulai, W. (2024). Factors Influencing User Intentions on Interactive Websites: Insights From the Technology Acceptance Model. *IEEE Access*, *12*, 122735–122756. https://doi.org/10.1109/ACCESS.2024.3437418

[6] Oleshchenko, L., & Burchak, P. (2023). Web Application State Management Performance Optimization Methods. In Z. Hu, I. Dychka, & M. He (Eds), *Advances in Computer Science for Engineering and Education VI* (Vol. 181, pp. 59–74). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-36118-0_6

[7] Shivakumar, S., & Suresh, P. V. (2018). A Survey and Analysis of Techniques and Tools for Web Performance Optimization. *Journal of Information Organization*, *8*(2), 31. https://doi.org/10.6025/jio/2018/8/2/31-57

[8] Thompson, H. S. (2024). Improved methodology for longitudinal Web analytics using Common Crawl. *ACM Web Science Conference*, 59–69. https://doi.org/10.1145/3614419.3644018

[9] Willnecker, F., Brunnert, A., Gottesheim, W., & Krcmar, H. (2015). Using Dynatrace Monitoring Data for Generating Performance Models of Java EE Applications. *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*, 103–104. https://doi.org/10.1145/2668930.2688061

[10] Zarrad, A. (2015). A Systematic Review on Regression Testing for Web-Based Applications. *Journal of Software*, *10*(8), 971–990. https://doi.org/10.17706/jsw.10.8.971-990



**Ph.D. Liubov Oleshchenko**
e-mail: oleshchenkoliubov@gmail.com

Associate professor at the Computer Systems Software Department at the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute". Author and co-author of over 100 scientific papers. Scientific supervisor and researcher of the scientific project "Methods for optimizing software performance and using artificial intelligence technologies to improve big data analytics systems". Her research interests include mathematical modelling, computer network technologies, IoT, software methods for big data analytics, machine learning, AI technologies.

https://orcid.org/0000-0001-9908-7422