Article

A Novel Approach for Evaluating Web Page Performance Based on Machine Learning Algorithms and Optimization Algorithms

Mohammad Ghattas [1], Antonio M. Mora [1,*] and Suhail Odeh [2]

1 Department of Signal Theory, Telematics and Communications, School of Computer Sciences and Telecommunications (ETSIIT) and Research Center on Information and Communication Technologies (CITIC-UGR), University of Granada, 18071 Granada, Spain; mohamadghattas@correo.ugr.es

2 Department of Software Engineering, Faculty of Science, Bethlehem University, Bethlehem P1520468, Palestine; sodeh@bethlehem.edu

* Correspondence: amorag@ugr.es

Abstract: This study introduces a novel evaluation framework for predicting web page performance, utilizing state-of-the-art machine learning algorithms to enhance the accuracy and efficiency of web quality assessment. We systematically identify and analyze 59 key attributes that influence website performance, derived from an extensive literature review spanning from 2010 to 2024. By integrating a comprehensive set of performance metrics—encompassing usability, accessibility, content relevance, visual appeal, and technical performance—our framework transcends traditional methods that often rely on limited indicators. Employing various classification algorithms, including Support Vector Machines (SVMs), Logistic Regression, and Random Forest, we compare their effectiveness on both original and feature-selected datasets. Our findings reveal that SVMs achieved the highest predictive accuracy of 89% with feature selection, compared to 87% without feature selection. Similarly, Random Forest models showed a slight improvement, reaching 81% with feature selection versus 80% without. The application of feature selection techniques significantly enhances model performance, demonstrating the importance of focusing on impactful predictors. This research addresses critical gaps in the existing literature by proposing a methodology that utilizes newly extracted features, making it adaptable for evaluating the performance of various website types. The integration of automated tools for evaluation and predictive capabilities allows for proactive identification of potential performance issues, facilitating informed decision-making during the design and development phases. By bridging the gap between predictive modeling and optimization, this study contributes valuable insights to practitioners and researchers alike, establishing new benchmarks for future investigations in web page performance evaluation.

Keywords: machine learning; web page; statistic model; page load time; performance; prediction models; early stage; classification

check for updates

Academic Editor: Luis Javier Garcia Villalba

Received: 2 November 2024
Revised: 15 December 2024
Accepted: 18 December 2024
Published: 21 January 2025

Citation: 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**, *6*, 19. https://doi.org/10.3390/ai6020019

1. Introduction

The digital revolution has spurred the adoption of web-based service delivery across various sectors, including government, retail, travel, finance, and banking. These web interfaces, accessible on a multitude of devices from desktops to mobiles, serve as the primary point of interaction for users availing of digital services. Consequently, web page performance emerges as a critical research area due to its well-established influence on user experience (UX). Prior studies have consistently demonstrated a positive correlation between UX and factors such as productivity, profitability, and brand equity within organizations [1–4].

However, despite extensive research, a gap remains in the holistic evaluation of web page performance, particularly in accounting for dynamic user interactions, scalability, and real-world usage scenarios. In this study, 'quality' is defined as a set of measurable characteristics that determine the overall performance and user experience of a webpage. These characteristics include factors such as page load time, responsiveness, accessibility, and user interaction. Since page load time has been shown to strongly correlate with user satisfaction and bounce rates [5], it is considered a key component of webpage quality. Other factors, such as how responsive the webpage is to user actions and its overall stability, are also critical for ensuring a seamless user experience. Given that 'quality' can be a subjective term, our definition is grounded in both literature and practical industry insights and is operationalized through a set of specific, measurable metrics that are relevant to web developers and users alike. These findings underscore the importance of minimizing response times as a key driver of web page performance.

Studies investigating the impact of web page response time on user behavior reveal a significant negative correlation. For example, an investigation by Aberdeen [6] involving 160 organizations indicated a 16% decrease in customer satisfaction and a 7% decline in conversion rates when response time increased by one second. Similarly, Gomez et al. [7] analyzed over 150 million page views across 150 websites and observed a 33% surge in page abandonment rate as response time rose from 2 to 8 s.

To ensure optimal user experience (UX), organizations have adopted a multifaceted approach to monitoring the performance of critical user journeys. This approach often integrates synthetic and real user monitoring (RUM) techniques [8–12]. Synthetic monitoring leverages pre-recorded scripts to emulate user journeys, while Real User Monitoring (RUM) captures actual user interaction data through JavaScript code embedded in web pages. Together, these methods provide a comprehensive performance analysis. However, their focus on runtime evaluation limits their ability to address performance issues during the early design phases.

A key performance metric, page load time, predicts user experience during development by encompassing events like downloading and rendering HTML, JavaScript, CSS, and images [6,13]. This metric is influenced by extrinsic factors (e.g., network latency, bandwidth, server capacity) and intrinsic factors (e.g., page size, resource usage, third-party content). Industry practices and a Google study highlight the strong correlation between page load time and user bounce rates, emphasizing its importance as a Service Level Objective (SLO) [4].

While traditional runtime metrics such as SLO are valuable for operational monitoring, they fall short in addressing the early prediction of performance bottlenecks. Integrating performance prediction models during the Software Development Life Cycle (SDLC, such as Agile) has shown promise in mitigating such issues. Early architectural decisions greatly influence software performance [14,15]. Integrating performance considerations and prediction models into the early stages of the SDLC is essential [15–17]. Traditional approaches often rely on intuition and limited data, while agile methodologies address these limitations by promoting iterative development, automation, and CI/CD practices [18].

Although performance prediction has been widely studied in system architectures, its application to web page performance remains underexplored. For instance, agile testing environments often fail to replicate real-world conditions, as scaled-down production-like environments introduce inaccuracies in page load time measurements [15,18–20]. This creates a need for predictive models capable of estimating performance during the early development phases, enabling proactive risk mitigation and reducing rework costs.

In recent years, traditional methods for evaluating web page performance, such as analyzing load time, first contentful paint, and other surface-level metrics, have proven insufficient in capturing the complex factors influencing modern web applications. These conventional techniques do not fully account for the intricacies of user engagement, multi-platform performance, and the growing diversity in web technologies. Recent approaches have sought to incorporate machine learning (ML) to predict web performance, but these models often suffer from limitations in accuracy or fail to optimize critical performance variables effectively [21,22].

The novelty of our work lies in its ability to bridge the gap between machine learning's predictive power and optimization algorithms' fine-tuning capabilities. Unlike previous studies that primarily focus on load time or singular metrics, our methodology evaluates a broader range of performance indicators, including user experience, page responsiveness, and scalability across various devices and network conditions, by systematically applying existing machine learning techniques and feature selection methods. This novel contribution ensures that web applications are not only faster but also optimized for real-world usage scenarios, providing measurable improvements over existing state-of-the-art models. Specifically, our contributions include the proposal of a methodology that integrates machine learning with optimization techniques to enhance web performance evaluation, the identification and application of an expanded set of performance metrics tailored for a comprehensive evaluation, and extensive benchmarking to validate the effectiveness of the methodology using existing machine learning models. Moreover, our extensive benchmarking and empirical testing have demonstrated the effectiveness of this methodology in achieving high prediction accuracy and significant performance improvement, highlighting the value of integrating machine learning with optimization techniques in this domain.

1.1. Motivating Example

Consider a real-world scenario involving a critical government web application. The home page size is 471,931 bytes, comprising six JavaScript files (59,139 bytes), two CSS files (4634 bytes), and 43 images (382,989 bytes). This single-domain page lacks a content delivery network (CDN). The service-level objective (SLO) mandates 90% of page loads to occur within 5 s using a web browser. A synthetic monitoring tool will simulate user behavior by replaying a script that opens the homepage four times per hour on a 5 Mbps connection with a cleared browser cache.

To assess performance headroom for real-world network and backend variations, the architect requires a model capable of predicting page load time with the available data (page size, file types, network speed, etc.). The conventional approach necessitates waiting until the testing phase to evaluate load time. However, a readily available, efficient model would enable prediction without waiting for testing. Ideally, such a model should require minimal effort and time investment for both data collection and usage.

This scenario exemplifies the limitations of traditional methods and underscores the need for a more efficient approach. We propose a model-based prediction technique that addresses these challenges, as detailed in the following sections.

Our motivating example highlights the critical need for early-stage performance prediction models. Machine learning (ML) has become increasingly prominent within the software industry, driven by two key trends: the growing capability to collect vast amounts of data and the declining cost of processing it. Unlike human experts, ML can uncover hidden relationships within data, presenting an opportunity to develop accurate prediction models for web page load time.

Existing research explores various ML approaches for this purpose. Butkiewicz et al. leverage regression techniques, while Zhou et al. employ classification models to categorize web page performance into different tiers (excellent, good, fair, unacceptable) [13,23]. Additionally, Calvano investigates the correlation between performance metrics and page characteristics [24]. However, a gap exists in comprehensively evaluating the suitability of ML techniques for load time prediction using website-specific metrics like page size, number of requests, content types, and server distribution [24].

This research aims to address this gap by proposing a systematic methodology that utilizes extracted features to predict web page performance during the design phase, leveraging readily available website attributes.

1.2. Research Objectives

The research objectives of our work are as follows.

- RO1: What are the features that affect web page performance?
- RO2: Which ML technique shows the highest accuracy in predicting web page performance?
- RQ3: Which ML techniques show a statistically significant difference in predictive accuracy?

1.3. Organization of This Paper

The remainder of the paper is organized as follows. We describe the necessary background for this study in Section 2. Section 3 elaborates on the research on the existing literature. In Sections 4 and 5, we outline our overall methodology and report the results. Section 6 lists the possible threats to the validity of our findings. Finally, Section 7 gives the conclusions and directions for future work.

2. Research on Existing Literature

Throughout the software development process, evaluating website performance plays a crucial role. Traditionally, this has involved either building models or employing measurement tools. Models, often based on machine learning techniques, create an approximate representation of the system to predict future behavior. Measurement, on the other hand, involves directly observing the actual website.

This paper focuses on leveraging machine learning models to predict website performance early in the development lifecycle. By doing so, potential performance issues can be identified and addressed proactively, leading to a more optimized final product.

2.1. Models for Web Page Performance Prediction

Web page performance modeling can be broadly categorized into reference models, analytical models, and simulation models.

Reference Models: These provide a conceptual framework for understanding web page load times by breaking down different components and suggesting improvements. For example, Loosely et al. [25] created a model to optimize various factors such as reducing images and using closer servers. PeterSevcik et al. [26] developed a formula considering data transfer amount, internet speed, and processing times. Chiew [27] studied how web page elements like code size and number of images affect load times.

Analytical Models: These use mathematical equations to predict web page download times. Early models by Menasce et al. [16] and Zhi [28] considered factors like page size and bandwidth. More complex models included server and client processing times and payload size (Peter Sevcik et al. [26], Nagarajan et al. [29], Butkiewicz et al. [13], Krzysztof et al. [30]). Machine learning techniques have also been explored for performance prediction (Zhou et al. [23]). These models are valuable in the early development stages for quick performance predictions.

Simulation Models: These replicate the actual loading process to predict web page load times. For instance, WebProphet simulates how different elements on a webpage interact to predict load time changes due to factors like server processing time or network delays, offering more nuanced predictions compared to analytical models [31].

In evaluating website quality, various methodologies and frameworks have been employed (presented in Table 1). For example, Elsater et al. (2022) [32] evaluated hotel websites using user surveys, while Adepoju et al. [33] (2019) introduced a usability framework for university websites. Allison et al. [34] (2019) developed a comprehensive framework through a literature review. Comparatively, our study distinguishes itself by focusing on the identification of critical attributes that influence website performance and proposing a systematic methodology that can be adapted for evaluating diverse website types. Unlike prior research, which often lacks flexibility and broad applicability, our approach integrates automated tools for real-time evaluation and predictive analytics, equipping it to address future performance improvements and potential issues proactively. This framework not only builds upon but also surpasses previous studies by incorporating a wider range of performance metrics, capturing user experience, load distribution, and scalability factors. By addressing these aspects, our research offers a holistic solution that effectively combines prediction accuracy, flexibility, and comprehensive evaluation, making a significant contribution to advancing methodologies in website quality assessment.

Table 1. This table provides an overview of different methodologies and frameworks used for web performance evaluation. It lists the advantages and disadvantages of each approach, along with the specific methods employed in each case, offering a clear comparison to assist in understanding the strengths and limitations of the various approaches.

Reference	Advantages	Disadvantages	Methods Used
Elsater et al. (2022) [32]	Detailed quality evaluation specific to hotel websites; comprehensive criteria	Limited to five and four-star hotels in Egypt	Evaluation of website quality, user surveys
Adepoju et al. (2019) [33]	Integrated usability framework; covers multiple aspects	Specific to university websites; may not be generalizable	Usability evaluation framework, user testing
Allison et al. (2019) [34]	Comprehensive framework; extensive literature review	Complex implementation; requires significant resources	Literature review, framework development
Alsulami et al. (2021) [35]	Effective for measuring website performance	Limited to sustainability perspective	Performance measurement, sustainability analysis
Amjad et al. (2021) [36]	Empirical study of e-commerce sites; performance-focused	Specific to e-commerce in Bangladesh	Performance analysis, empirical study
Armaini et al. (2022) [37]	Focused on government websites; performance variables	Specific to Labuhanbatu Regency	Performance evaluation, government websites
Aziz et al. (2019) [38]	Quality measurement using AHP; structured approach	Requires expertise in AHP	Analytical hierarchy process (AHP)
Barus et al. (2022) [39]	Performance testing and optimization; actionable insights	Limited to DiTenun website	Performance testing, optimization

Table 1. *Cont.*

Reference	Advantages	Disadvantages	Methods Used
Kulkarni & Dixit (2012) [40]	Empirical and automated analysis; practical approach	May require technical skills for automation	Empirical analysis, automated tools
Cai et al. (2020) [41]	Automatic assessment system; objective evaluation	Specific to government websites	TFN-AHP methodology, automated assessment
Devi & Sharma (2016) [42]	Academic website evaluation framework	Limited to academic websites	Framework development, academic websites
Dhiman & Anjali (2014) [43]	Empirical validation; combines statistical and machine learning	Requires statistical and ML knowledge	Statistical methods, machine learning
Dominic & Jati (2011) [44]	Comparison of Asian airline websites; non-parametric test	Specific to airline industry	Non-parametric test, website comparison
Dominic et al. (2011) [45]	E-government websites quality comparison; structured approach	Limited to Asian e-government	Non-parametric test, quality ranking
Erokhin et al. (2019) [46]	Mathematical simulation for website effectiveness	Requires mathematical modeling skills	Mathematical simulation, effectiveness evaluation
Faustina & Balaji (2016) [47]	Performance evaluation using AHP; structured approach	Specific to university websites in Chennai	Analytical hierarchy process (AHP)
Gangurde & Kumar (2020) [48]	Web page prediction using GA and LR; innovative methods	Requires knowledge of GA and LR	Genetic algorithm, logistic regression
Gharibe Niazi et al. (2020) [49]	Proposed framework for university websites	Specific to university websites	Framework development, university websites
Harshan et al. (2016) [50]	AHP-based model for library websites	Specific to library websites	Analytical hierarchy process (AHP)
Hidayah et al. (2019) [51]	Combines Webqual and IPA; government websites	Limited to government websites	Webqual, Importance Performance Analysis (IPA)
Jati (2011) [52]	Quality ranking using PROMETHEE II; structured approach	Limited to e-government websites	PROMETHEE II, quality ranking
Kabassi (2018) [53]	AHP for website evaluation; structured approach	Requires AHP expertise	Analytical hierarchy process (AHP)
Kinnunen (2020) [54]	Web performance improvement using free tools	Limited to free tools	Performance improvement, free tools
Kumar & Arora (2019) [5]	Feature selection for website quality prediction	Requires feature selection knowledge	Filter-wrapper, website quality prediction
Kumar et al. (2021) [55]	Website performance analysis using automated tools	Requires technical skills for automation	Automated tools, performance analysis
Kwangsawad et al. (2019) [56]	Automated evaluation tools; website performance	Limited to automated tools	Automated tools, performance evaluation

Table 1. *Cont.*

Reference	Advantages	Disadvantages	Methods Used
Najadat et al. (2021) [57]	Web diagnostic tools and DEA; website evaluation	Requires DEA knowledge	Web diagnostic tools, Data Envelopment Analysis (DEA)
Olaleye et al. (2018) [58]	Comparative analysis of Nigerian universities	Limited to Nigeria	Comparative analysis, university websites
Saleh et al. (2022) [59]	Systematic literature review on university websites	Limited to university websites	Systematic literature review, university websites
Shayganmehr & Montazer (2019) [60]	Quality of e-government websites; identifying key indexes	Specific to e-government	Index identification, quality assessment
Shayganmehr & Montazer (2021) [61]	Hybrid fuzzy decision-making for e-government	Requires fuzzy decision-making knowledge	Hybrid fuzzy decision-making, quality assessment
Deloitte (2024) [62]	Future web technologies; business applications	Predictive; not focused on current evaluation	Web 3.0, business applications
Erokhin et al. (2019) [46]	Mathematical simulation for website effectiveness	Requires mathematical modeling skills	Mathematical simulation, effectiveness evaluation

Building upon the limitations identified in Table 1, our research proposes a comprehensive methodology that integrates predictive and analytical capabilities to address these challenges in a holistic manner. Unlike prior studies conducted between 2010 and 2024, which primarily focused on specific domains such as academic, government, or commercial websites and often relied on standalone methodologies like AHP or Webqual, our work applies existing machine learning algorithms alongside optimization techniques. This approach enables a broader evaluation of performance indicators, extending beyond traditional metrics to address complex datasets and diverse real-world scenarios. By incorporating contemporary technologies and methodologies, our research not only addresses the shortcomings of earlier studies but also ensures higher accuracy, adaptability, and relevance to current and emerging web application needs, demonstrating a practical advancement in the field.

2.2. Techniques for Measuring Web Page Performance

Web page performance measurement comes into play after a website is up and running, focusing on real-world user experience. There are several techniques used by these tools:

- Page Instrumentation: Scripts embedded directly in the web page track user-experienced response times. Modern browsers offer built-in support for this through Navigation Timing and Paint Timing APIs, providing detailed performance data [63,64].
- Traffic Analysis: Techniques like those developed by Olshefski et al. [65] analyze server-side traffic data to estimate how long it takes for users to see content.
- Synthetic Transactions: Tools like sMonitor (v4.2) [66] simulate user actions to identify performance issues that might arise under heavy traffic loads.
- In-Browser Profiling: Wang et al.'s Wprof (v1.0) [62] is an example of this approach. It gathers timing and task dependency information directly within the browser during a real page load, helping pinpoint bottlenecks and areas for improvement.

Many popular tools exist to assess web page performance, including web page testing services (WebPageTest (v22.01) [12], GTmetrix (v1.1.0) [67]), web analytics platforms (Google Analytics [68]), and browser developer tools (like Chrome DevTools with its network emulator [69]). There are also other options available such as Fiddler (v5.0.20238.1) [70], YSlow (v3.1.2) [71], PageSpeed Insights [72], and Lighthouse (v10.0.0) [73]. By leveraging these tools, developers can gain valuable insights into user experience and make data-driven decisions to optimize website performance.

Existing web page performance measurement tools, while useful, face significant limitations that hinder their effectiveness in today's dynamic web environment. These tools typically rely on synthetic transactions or server-side monitoring, which may not accurately reflect real-world user experiences, particularly as network conditions and device configurations vary. Additionally, as websites grow more complex, with numerous dependencies and interactions among various components, traditional tools often fall short in capturing and analyzing these intricate relationships. Most importantly, traditional tools tend to be retrospective, focusing on past performance metrics rather than proactively identifying future issues. This reliance on historical data without predictive insights limits their ability to foresee performance bottlenecks or optimization needs. Moreover, traditional tools generally require substantial manual intervention for setup and configuration, making them less scalable and adaptable as site requirements evolve.

Our research addresses these limitations by applying the predictive and analytical strengths of existing machine learning and optimization algorithms to develop a comprehensive methodology for web performance evaluation. Unlike previous studies that focus on isolated metrics, such as load time, our methodology integrates a broader range of performance indicators, including user experience, page responsiveness, and scalability across diverse network conditions and device types. The novelty of our methodology lies in its capacity to utilize real-world performance data, capturing the complexities of modern websites and anticipating potential issues before they impact users. Moreover, by incorporating continuous learning from new performance data, our methodology remains adaptable to emerging trends in web technologies and user behaviors, offering flexible solutions for web performance management. By combining machine learning's predictive power with optimization techniques, we provide a fine-tuned, adaptable methodology that not only improves prediction accuracy but also offers actionable insights for enhancing performance. This integration of predictive and optimization models enables the system to recommend specific actions for performance improvement based on the identified bottlenecks, making it an invaluable tool for developers and decision-makers. This approach has demonstrated strong results in benchmarking tests, highlighting its potential to complement existing tools and contribute to data-driven, proactive web page performance optimization. As a result, our work pushes the boundaries of traditional performance evaluation by moving from reactive analysis to proactive, predictive insights, setting a new benchmark for future research in the field.

3. Background

In this section, we describe the main steps in the loading of most web pages, the metrics that affect the loading performance, and the metrics used to measure the performance. We also provide a summary of the various predictive modeling techniques used in this work.

3.1. How Browsers Build Web Pages: A Metric-Driven Analysis

Modern web pages are composed of HTML, CSS, JavaScript, and images, requiring several behind-the-scenes processes to render content on a user's screen. Initially, the browser establishes a TCP connection with the server, potentially performing a DNS lookup if the IP address is not cached. The server processes the request, generates the content, and assembles the final HTML document, often involving database queries and API interactions. The browser then parses the HTML to construct a Document Object Model (DOM), initiating the Critical Rendering Path (CRP), which is essential for optimizing webpage rendering performance [3].

During this process, the browser requests and parses the CSS file to build the CSS Object Model (CSSOM). If it encounters JavaScript, the parser waits for the CSSOM to be available and for the script to download and execute unless the script is marked with the async attribute, allowing DOM construction to continue concurrently [4]. Rendering only starts once both the DOM and CSSOM are fully built (presented in Figure 1).

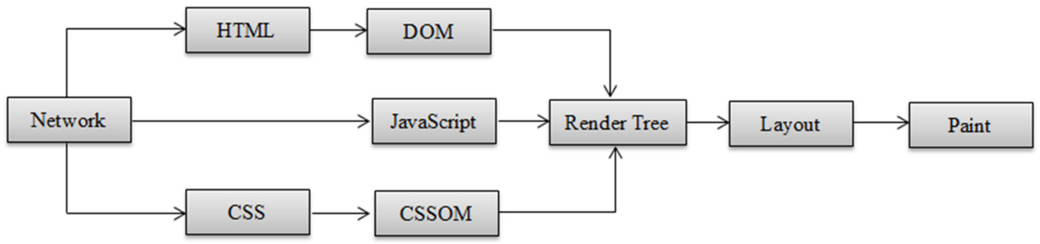


Figure 1. These models are combined into the render tree, which the browser uses to lay out visible elements and initiate rendering.

Key Metrics in Browser Performance

- User Initiates Request: The user clicks a link or enters a URL.
- DNS Lookup: The browser translates the URL into an IP address using the Domain Name System (DNS).
- Connection Establishment: The browser connects to the web server using the IP address.
- Sending Request and Receiving Response (Response Time): The browser sends an HTTP request to the server, which processes it and sends back an HTML file and other resources. Response Time measures this round trip (lower is better).
- Parsing and Rendering (Load Time, First Byte Time, Start Render Time, Largest Contentful Paint): The browser parses the HTML to build the DOM, fetches additional resources, and begins rendering the page. First Byte Time measures the time to receive the first byte of data (lower is better). Start Render Time indicates when the basic content starts displaying (lower is better). Largest Contentful Paint marks when the main content is displayed (lower is better).
- Downloading Resources (Page Size, Byte In): The browser downloads all requested resources. Page Size indicates the total size of downloaded resources (smaller is better). Byte In represents the total amount of downloaded data.
- Processing Resources: The browser processes downloaded resources like images, and executes JavaScript.
- Interactive Page (Time to Interactive, Document Complete Time): Once the page is fully rendered and scripts executed, it becomes interactive. Time to Interactive measures this duration (lower is better). Document Complete Time represents the total time to load the entire page, including all resources (lower is better).

3.2. Predictive Modeling Techniques

Prediction models are essentially functions that correlate a set of input variables (also referred to as predictable, explanatory, or independent variables) with a variable response (also known as the outcome or dependent variable). To construct a predictive model, a foundational dataset must be prepared from historical data, which can be collected through experimentation or during system operations. This involves organizing a set of fields representing the object of interest into a structured record, where these fields represent both the input and response variables. This foundational dataset typically consists of numerous such records, which are then divided into two partitions: the training dataset and the test dataset.

Initially, the training dataset is utilized to train the prediction model. This training process continues until the model effectively learns the relationship or mapping function between the input variables and the response variable. Subsequently, the trained model is employed to forecast the values of the response variable for the records within the test dataset. The accuracy of the model's predictions on the test dataset is then assessed using various accuracy metrics. If these metrics indicate satisfactory performance, the predictive model is deemed to have successfully generalized the knowledge extracted from historical data.

In such instances, the model can be applied to predict the value of the response variable for given input values. This learning process is commonly referred to as supervised learning. Presently, there exists a plethora of machine learning techniques tailored for constructing predictive models. In this section, we provide a succinct overview of nine cutting-edge classification techniques. The selection of these methods was driven by their ability to handle various complexities in the dataset, such as imbalanced data and high-dimensional features, which are common in web performance prediction tasks.

3.2.1. Naive Bayes

NB [74] means that features (i.e., web page attributes) are conditionally independent of a specified label (in this instance, a performance label). Based on this concept, for a web page $WP_i = \{t_1, t_2, \ldots, t \mid WP_i \mid \}$, where t_i is a metric of web page, and a label C_i, we have

$$P(WP_i \mid C_i) = \prod_{i=1}^{|WP_i|} p(t_i \mid C_i) \tag{1}$$

Applying Bayes' theorem to Equation (1) yields

$$\begin{aligned} P(C_i = c|WP_i) &= \frac{p(C_i{=}c) \times p(WP_i|C_i{=}c)}{\sum_{c' \in \{c,\bar{c}\}} p(C_i{=}c) \times p\left(WP_i|C_i{=}c'\right)} \\ &= \frac{p(C_i{=}c) \times \prod_{i=1}^{|WP|} p(t_i|C_i)}{\sum_{c' \in \{c,\bar{c}\}} p(C_i{=}c) \times \prod_{i=1}^{|WP|} p(t_i|C_i)} \end{aligned} \tag{2}$$

We apply Equation (2) to predict the label for a web page WP_i. For example, if max $P(C_i = c \mid WP_i)$ is determined for a web page, we classify the web page as a performance level c; else is similar. The key advantage of NB is the short training time for its computational complexity since it assumes the conditional independence between features.

3.2.2. Naive Bayes Multinomial

Naive Bayes Multinomial (NBM) builds upon the original Naive Bayes (NB) [74]. NBM recognizes that the impact of a webpage characteristic (feature) on performance is not solely dependent on its presence or absence. Instead, the time at which that feature appears can also be significant. This consideration can be particularly advantageous when dealing with datasets containing a large number of unique values for webpage characteristics. In essence, NBM often outperforms NB in such scenarios.

3.2.3. K-Nearest Neighbours

k-Nearest Neighbors (kNN) [75] is a machine learning technique that relies on similar instances. In the context of website performance prediction, kNN predicts the performance label of a new webpage by considering its k-Nearest Neighbors (kNN) in the training data. The underlying assumption is that websites with similar characteristics will likely have similar performance. The kNN algorithm works in two steps:

1. Find Nearest Neighbors: Given a new, unlabeled webpage, kNN searches the training data to identify its k closest neighbors based on a chosen distance metric. Common distance metrics include Euclidean distance, Minkowski distance, and Manhattan distance. In this study, we utilize Euclidean distance.
2. Predict Performance Label: kNN assigns the most frequent performance label from the k nearest neighbors to the new webpage. For example, if the majority of its neighbors are labeled as "high performance", the new webpage is also predicted to have "high performance".

3.2.4. Support Vector Machine

Support Vector Machine (SVM) [75] is a powerful machine learning technique rooted in statistical learning theory. SVMs excel at classification tasks by creating hyperplanes (decision boundaries) in a high-dimensional space. Each website in the training data is transformed into a point within this space, with each feature acting as a dimension.

The SVM algorithm strategically identifies a small number of critical training instances, called support vectors. These support vectors represent the boundaries between different performance levels. The SVM then constructs a function, either linear or non-linear depending on the data, to separate these performance levels while maximizing the margin (distance) between them.

3.2.5. Bayesian Network

Bayesian Network (BN) [76] is a powerful tool that leverages probability theory to understand the connections between website characteristics (features) and performance levels (labels). Unlike traditional models, BN utilizes a directed acyclic graph (DAG) where each node represents a feature or a label. Importantly, directed edges between nodes indicate a causal relationship.

In our framework, during the model-building phase, BN would automatically construct such a graph based on the training data containing web page characteristics. Once built, this BN can be used in the prediction phase to estimate the unknown performance level for a new webpage.

3.2.6. Decision Tree

Decision Tree algorithms [75] are popular machine learning techniques that use a tree-like structure to make predictions. Each internal node in the tree asks a question about a specific feature of the data (e.g., "Is the webpage performance good?"). The branches represent the possible answers to that question, and the leaf nodes represent the final prediction (e.g., "high performance" or "low performance"). The topmost node is called the root node.

There are many different decision tree algorithms, but one of the most well-known is ID3, developed by Ross Quinlan. ID3 focuses on creating a shallow tree (with minimal depth) without necessarily considering the number of leaf nodes. In this study, we utilize C4.5, an improved version of ID3. C4.5 builds upon ID3 by addressing areas like handling default values, pruning unnecessary branches, and other refinements.

3.2.7. Logistic Regression

Logistic Regression [77] is a statistical method commonly used for classification tasks. It analyzes the relationship between independent variables (like webpage features) and a dependent variable (performance label) to predict the probability of an outcome. In this study, we leverage multinomial logistic regression, an extension of binary logistic regression, to handle multiple performance levels (more than two categories).

Here, we assign one specific performance level (e.g., "unacceptable") as the baseline class, denoted as $Y = h_0$. This allows us to model the probabilities of all other performance levels relative to the baseline.

$$P(Y = h_0 \mid X_1, X_2, \ldots, X_k) = \frac{1}{1 + \sum_{h=1}^{M-1} e^{(a_h + b_{h1}X_1 + b_{h2}X_2 + \ldots + B_{hk}X_k)}} \tag{3}$$

For other classes (except for h_0), we have

$$P(Y = h \mid X_1, X_2, \ldots, X_k) = \frac{e^{(a_k + b_{k1}X_1 + b_{k2}X_2 + \ldots + B_{kk}X_k)}}{1 + \sum_{h=1}^{M-1} e^{(a_h + b_{k1}X_1 + b_{k2}X_2 + \ldots + B_{kk}X_k)}} \tag{4}$$

The link between the independent variables and the dependent variable must be described by $M - 1$ equations if there are M classes. We can assign a new web page to a certain performance level by comparing probabilities across various labels.

3.2.8. AdaBoost

The AdaBoost algorithm, introduced by Freund and Schapire in 1997 [78], operates by iteratively calling a specified weak or base learner across multiple rounds, using a training set (x_i, y_i). Here, each x_i belongs to a specific domain or instance space X (such as web page characteristics), and each label y_i exists within a designated label set Y (representing web page performance levels). The fundamental concept underlying AdaBoost involves establishing a distribution or set of weights over the training set. These weights, denoted as $D_t(i)$ for training sample i at round t, are initially assigned equally. During each subsequent round, the weights of misclassified samples are increased, encouraging the base learner to focus on challenging examples within the training set.

3.2.9. Random Forests

The Random Forest algorithm [79] stands as another ensemble classification technique. Random Forest combines tree predictors in such a way that each tree's structure relies on the values of a randomly sampled vector, drawn independently and from the same distribution across all trees in the forest. The typical procedure of Random Forest entails generating a random vector V_k for the k-th tree, which is independent of previous random vectors but adheres to the same distribution. Using this vector and the training set, a tree is constructed, resulting in a distinct classifier $C(x, V_k)$, where x represents an input vector. The composition and dimensions of vector V vary depending on its use in tree construction. Upon generating a significant number of trees, each tree contributes a unit vote towards determining the most prevalent class for a given input x.

4. Methodology

The research work (see Figure 2) consists of the following tasks: (i) dataset acquisition, (ii) preprocessing, (iii) feature selection, and (iv) deploying a machine learning model. Each module is described as follows:

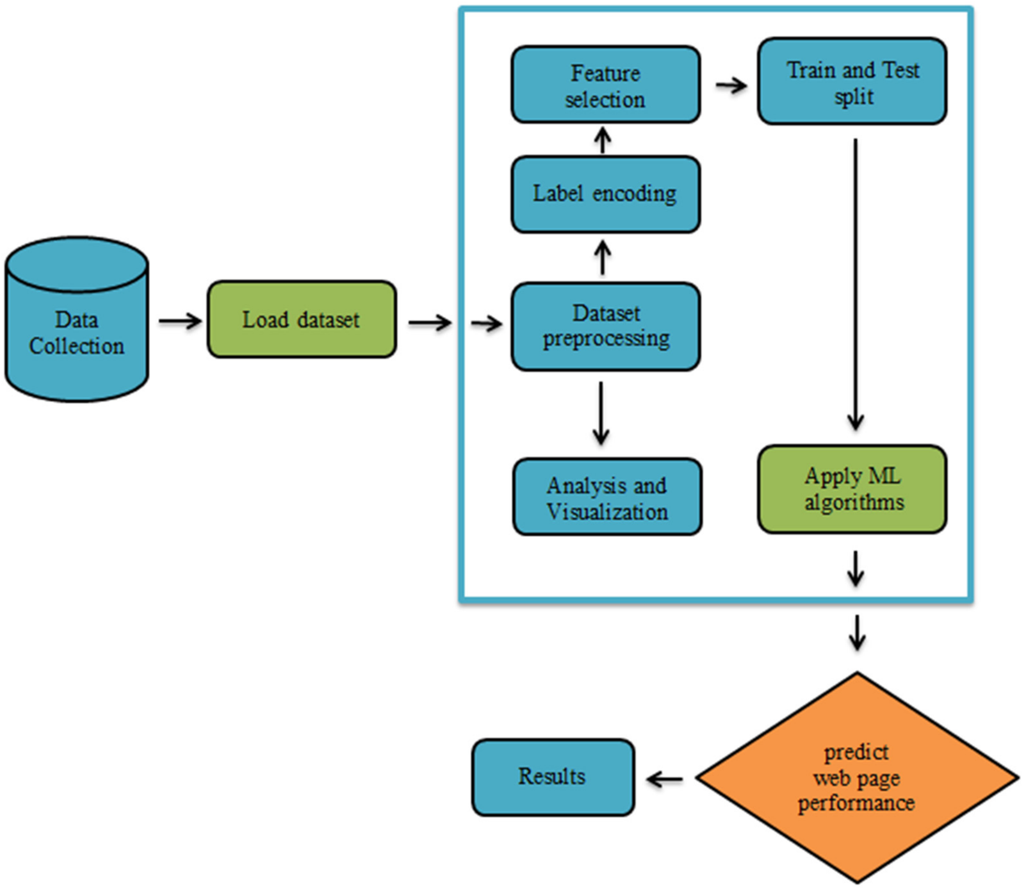


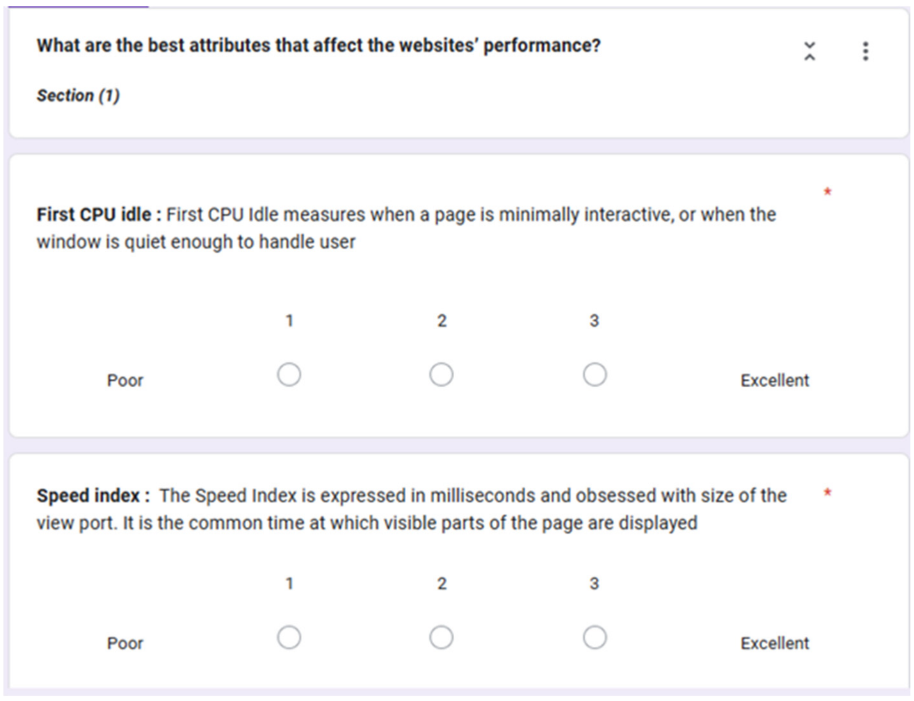
Figure 2. Block diagram of the proposed approach.

4.1. Dataset

Initially, 223 quality factors were extracted based on a comprehensive review of literature, standards, and practical considerations. These factors were subjected to a process of eliminating duplicates and employing filtering techniques (memoing) the pool to 59 relevant metrics. After refining the 59 quality factors, we conducted an online survey (sample questionnaire in Figure 3) to gather feedback from a sample of 35 web developers, performance experts, and industry professionals. The survey included the identified metrics and asked respondents to rate them on a scale from 1 (poor) to 3 (excellent). The selection of the final 16 metrics was driven by both statistical analysis and expert consensus.

A critical aspect of the feature selection process was the application of a threshold approach, where metrics rated as "excellent" (score of 3) by more than 50% of the participants were immediately included in the final set. In cases where metrics received a rating of 40–50%, they were further evaluated using a weighted approach, considering their practical importance and relevance to specific performance characteristics. For example, metrics that showed high relevance to real-world performance issues, such as load times and responsiveness, were prioritized.

Additionally, to ensure the robustness of the feature selection, metrics that showed high correlation or overlap in terms of the information they provided were carefully evaluated. Redundant metrics were excluded from the final set to ensure that each of the 16 metrics contributed unique and complementary information to the overall assessment of website performance. The selected features were then used for machine learning modeling, ensuring that the metrics chosen were not only statistically significant but also practically impactful for developers and end-users, forming the basis of our dataset for analysis.


What are the best attributes that affect the websites' performance?

Section (1)

First CPU idle : First CPU Idle measures when a page is minimally interactive, or when the window is quiet enough to handle user *

	1	2	3	
Poor	○	○	○	Excellent

Speed index : The Speed Index is expressed in milliseconds and obsessed with size of the view port. It is the common time at which visible parts of the page are displayed *

	1	2	3	
Poor	○	○	○	Excellent

Figure 3. Sample of questionnaire online.

To ensure diversity and minimize dataset biases, data were gathered from a wide range of sources and meticulously prepared for analysis. A total of 11,200 values were collected from 800 websites, selected from the Alexa Top Sites, which ranks websites globally based on popularity. This dataset includes domains such as .com, .net, .org, .edu, .gov, .info, and .int, representing a broad spectrum of industries and levels of complexity, including educational, governmental, informational, and commercial sectors.

While the selection of popular websites ensures robust and diverse representation, we acknowledge the potential limitation of over-representing highly ranked websites, which may not fully capture the characteristics of less popular or niche websites. Future research could expand this dataset to include websites across a wider range of popularity rankings to further enhance the generalizability of the findings.

However, during the data collection process, it was noted that one of the classes was underrepresented compared to the others. To address this imbalance, several techniques were applied to balance the dataset before further processing. These techniques included oversampling the minority class by generating synthetic samples using methods such as SMOTE (Synthetic Minority Over-sampling Technique) [80] and undersampling the majority class by randomly removing samples from it [80]. These adjustments ensured a more balanced and representative dataset for training and analysis, thereby improving model performance and addressing the bias toward the majority class.

This dataset was based on the 16 selected attributes chosen from the initial pool of 59. The acquisition of data for all metrics involved the utilization of website diagnostic tools. The specifics of these tools are presented in Table 2, providing insight into the instruments employed in the study. The dataset consists of the following 16 metrics:

- Response time: Period of time for sending a request and receiving a response (seconds).
- Load time: used to calculate the time required to load a page and its graphics (seconds).
- Page size: the size of the web pages in the website (MB).
- Broken link: broken links always reduce the quality of the website. Websites have internal or external links. A visitor expects the links to be valid, loads successfully to the clicked page (number).

- No of Request: the number of requests/responses between a client and a host (number).
- First byte: measures the time elapsed between the moment an internet user makes an HTTP request, like loading a webpage, to the time the first byte is received by the client's browser (seconds).
- Start render time: the moment when a web page begins to display content in a user's browser (seconds).
- Largest contentful paint: measuring perceived load speed as it marks the page load timeline when the page's main content has been loaded (seconds).
- Total link: total links on page (number).
- Markup validation: calculate the number of HTML errors that exist on the website, such as orphan codes, coding errors, missing tags and etc. (number).
- Time to interactive: the duration it takes for a web page to become fully interactive for users (seconds).
- Compression: JavaScript and CSS ensure proper compression; this makes the website run much faster (KB)
- Document complete time: the duration it takes for a web page to fully load, including all its resources such as images, stylesheets, and scripts (seconds).
- Byte in: the amount of information that the browser had to download to load the page (MB).
- Design optimization: The scripts, HTML or CSS codes optimized for faster loading. The optimization also reduces the number of website elements such as images, scripts, html, css codes or video (%).
- Speed Index (Performance): A metric that measures how quickly content is visually displayed during the loading of a web page. It quantifies the perceived load time of a webpage, taking into account both the time to load and the visual completeness of the page as it loads (%).

Table 2. Online web-diagnostic tools for data collection.

Web Metric	Web-Diagnostic Tools
Response Time	www.websitepulse.com (accessed on 9 January 2025)
Load Time	www.gtmetrix.com (accessed on 9 January 2025)
Broken Links	www.duplichecker.com/broken-link-checker.php (accessed on 9 January 2025)
No. of Requests	www.gtmetrix.com (accessed on 9 January 2025)
page size	www.gtmetrix.com (accessed on 9 January 2025)
page speed	www.gtmetrix.com (accessed on 9 January 2025)
mark-up validation	validator.w3.org/#validate_by_url (accessed on 9 January 2025)
design optimization	www.1and1.com/website-checker (accessed on 9 January 2025)
First byte	www.websitepulse.com/ (accessed on 9 January 2025)
Start Time Render	www.webpagetest.org/ (accessed on 9 January 2025)
Largest contentful paint	www.websitepulse.com/ (accessed on 9 January 2025)
Total link	www.duplichecker.com/link-count-checker.php (accessed on 9 January 2025)
Time to interactive	gtmetrix.com/ (accessed on 9 January 2025)
Compression	www.giftofspeed.com/gzip-test/ (accessed on 9 January 2025)
Document complete time	wpt.fasterize.com/ (accessed on 9 January 2025)
Byte in	wpt.fasterize.com/%20(accessed on 9 January 2025)

4.2. Data Preprocessing

Preprocessing helps transform data so that a better machine learning model can be built, providing higher accuracy. The preprocessing performs various functions: outlier rejection, filling missing values, and feature selection to improve the quality of data.

4.2.1. Missing Value Identification

Using the Python libraries pandas, numpy, seaborn and matplotlib.pyplot, we obtained the missing values in the datasets, shown in Figure 4. To handle these missing values, we replaced the missing value with the corresponding mean value. This imputation technique is commonly used for numerical features when the proportion of missing data is relatively small, and when the data are assumed to be missing at random.

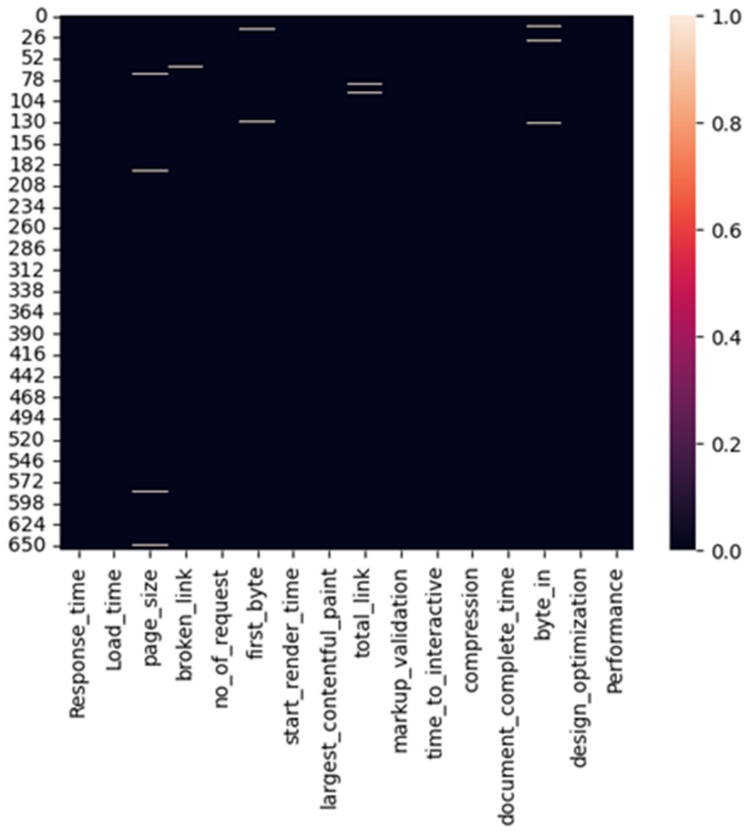


Figure 4. This figure is a missing value heatmap showing the distribution of missing data across the dataset. The x-axis represents variables, the y-axis represents data points, and the color bar indicates missing values (white for missing), with white lines highlighting rows where data is missing for specific variables.

4.2.2. Outlier Identification and Handling

Using the Python libraries pandas and plotly.express, we filtered the dataset to detect and handle outliers based on the interquartile range (IQR). We generated box plots for numerical features to visualize their distribution and identify any data points significantly outside the IQR, which could be potential outliers, as shown in Figure 5.

After that, to address the issue of outliers, we employed the interquartile range (IQR) technique, also called the midspread or middle 50%, which is a measure of statistical dispersion equal to the difference between the 75th and 25th percentiles (Q3 − Q1). This method is used for handling outliers, as shown in Figure 6.

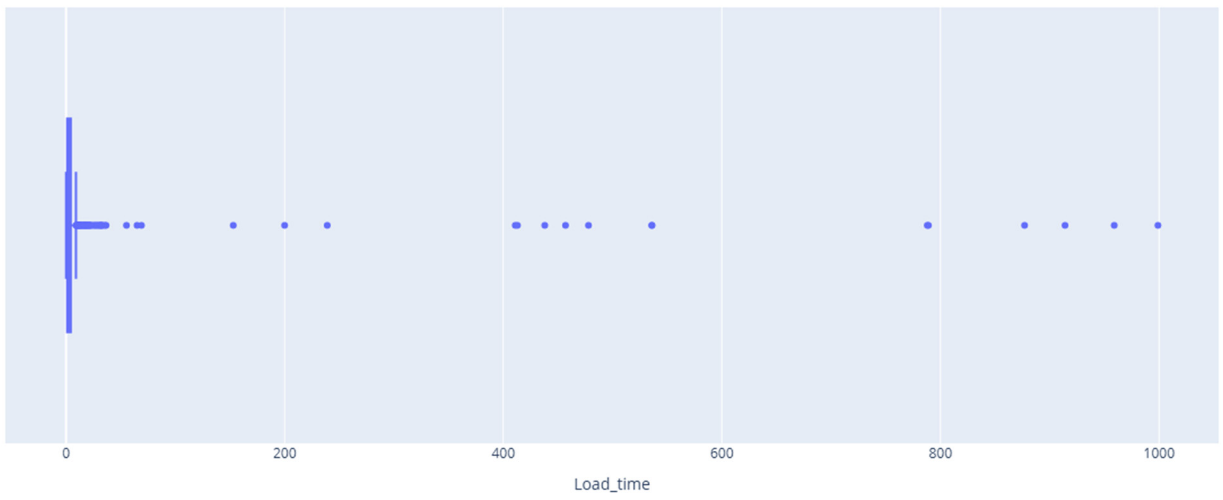


Figure 5. This figure illustrates a boxplot of the Load time variable, displaying its median, interquartile range, and outliers. The dots beyond the whiskers represent outliers, indicating unusually high or low load time values in the dataset.

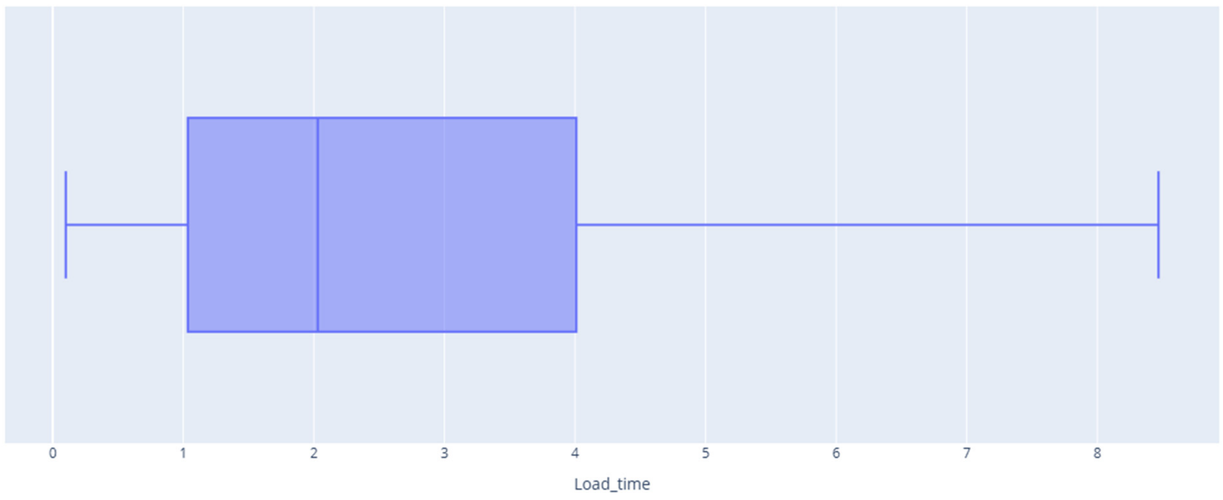


Figure 6. This boxplot illustrates the distribution of load time, with the median around 3 s and the majority of values falling between 2 and 4 s. Outliers are present, indicating some Load time values exceeding 8 s.

4.3. *Dataset Analysis*

The heatmap presented in Figure 7 shows the correlation matrix for various web performance metrics. Here is an analysis based on the heatmap:

4.3.1. High Positive Correlations

- Document Complete Time and Load Time: These metrics show a strong positive correlation, indicating that as the load time increases, the document complete time also increases. This is expected since longer load times usually result in longer overall document completion times.
- Speed and Response Time: A high correlation exists between these two metrics, suggesting that better performance is associated with faster response times.

4.3.2. High Negative Correlations

- Load Time and Compression: There is a noticeable negative correlation between load time and compression. This indicates that better compression (lower values) is associated with faster load times.

- Speed and Load Time: There is a strong negative correlation between performance and load time, implying that higher performance is linked to lower load times.

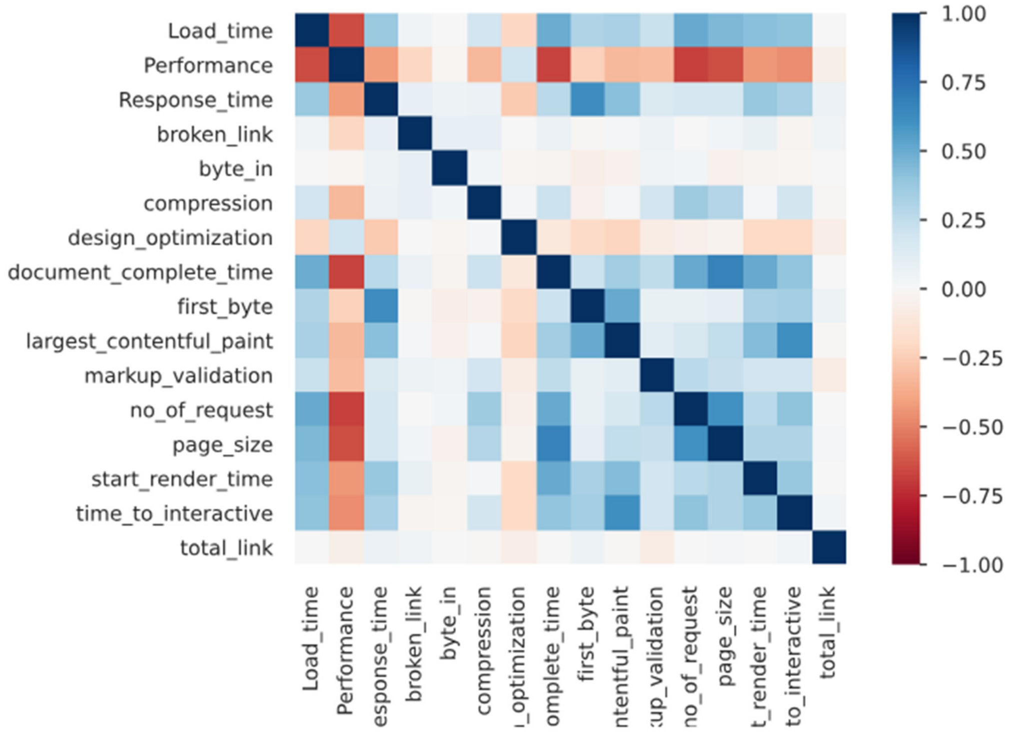


Figure 7. This matrix shows the Pearson correlation coefficients between pairs of variables, with values ranging from −1 to 1.

4.3.3. Moderate Positive Correlations

- Start Render Time and Load Time: There is a moderate positive correlation, which indicates that pages that take longer to load also take longer to start rendering.
- Total Link and No of Request: The correlation here suggests that pages with more links tend to have more requests, which is logical as each link often involves additional HTTP requests.

4.3.4. Moderate Negative Correlations

- Time to Interactive and Compression: This indicates that better compression is linked to a faster time to interact, enhancing the user experience.
- Markup Validation and Speed: A moderate negative correlation indicates that better HTML markup validation (fewer errors) is associated with better speed.

4.3.5. Low or No Correlations

- Byte In and Total Link: The correlation is relatively low, indicating that the amount of data downloaded is not strongly related to the number of links on a page.

4.3.6. Implications

- Performance Optimization: Focus on improving load time, compression, and response time, as these metrics strongly influence overall performance.
- Feature Selection: Metrics with low correlations with performance, such as broken links and bytes in, might be less critical for prediction models focusing solely on performance.
- User Experience: Ensuring good compression and minimizing load times can significantly enhance the user's interactive experience.

This correlation matrix provides valuable insights into the relationships between different web performance metrics. Understanding these correlations can help in identifying key areas for optimization, improving the overall user experience, and building more accurate predictive models for web page performance.

4.4. Feature Selection

In this study, we calculated Pearson's R coefficient for each pair of features to determine the strength and direction of the linear relationship between them. Pearson's R, which ranges from −1 to 1, measures the linear correlation between two variables, where values closer to 1 indicate a strong positive correlation, values closer to −1 indicate a strong negative correlation, and values around 0 indicate no linear correlation [81]. We visualized the results using a heatmap, which allowed us to identify the most significant features. By examining these correlations, we were able to select the features that have the strongest relationships with each other, ensuring a more robust analysis and better optimization of web performance metrics are shown in Figure 8.

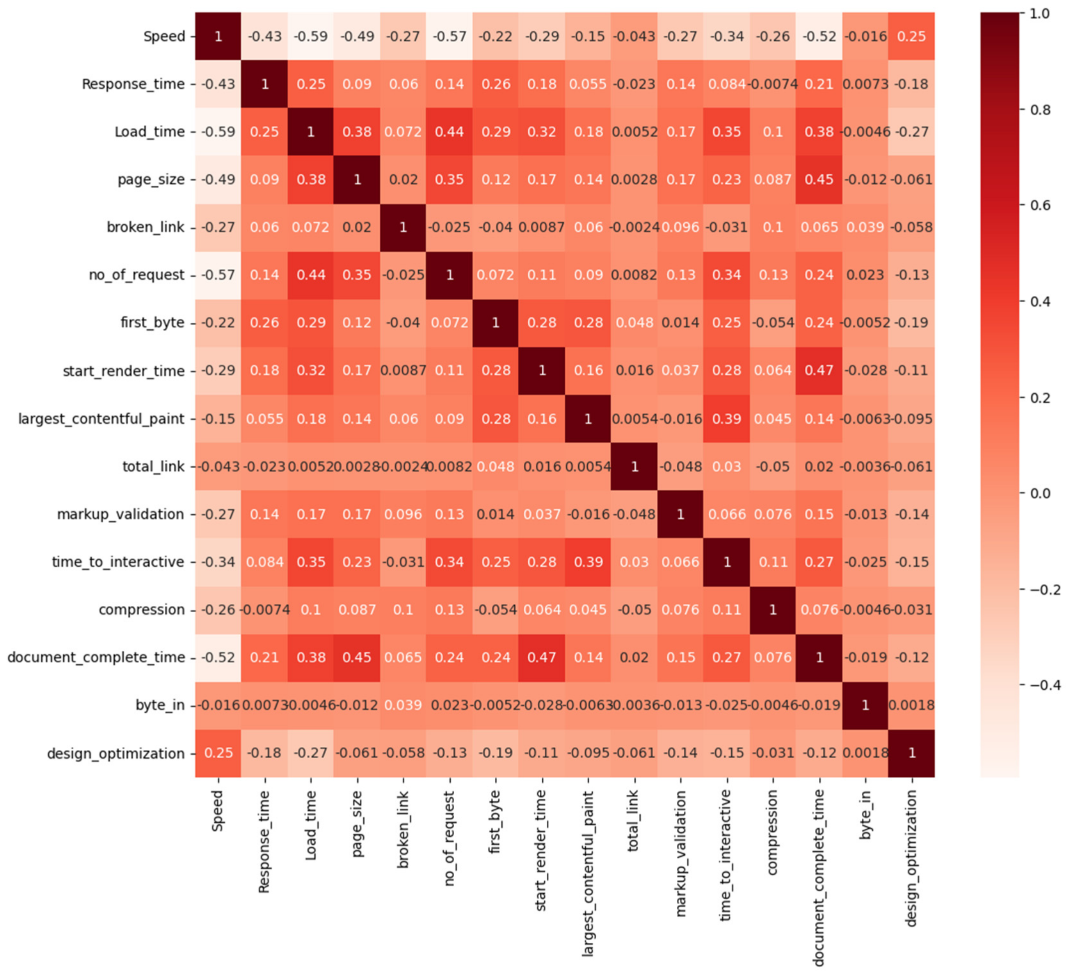


Figure 8. The heatmap shows the correlation between variables: dark red (close to +1) indicates strong positive correlation, light red/white (close to 0) indicates weak or no correlation, and dark negative red (close to −1) indicates strong negative correlation. Strongly correlated features (positive or negative) may indicate redundancy or inverse relationships. This helps in feature selection, understanding dependencies, and optimizing performance.

4.4.1. Correlation Analysis

In our study, we employed correlation analysis to understand the relationship between various input attributes and the target output, which in this context is web page performance as it is shown in Table 3.

Table 3. The correlation coefficients between input attributes such as load time, number of requests, document complete time, page size, response time, time to interactive, start render time, markup validation, broken links, and compression, and the performance output.

Attributes	Correlation Coefficient
Load_time	−0.593609
no_of_request	−0.571076
document_complete_time	−0.519318
page_size	−0.490623
Response_time	−0.428263
time_to_interactive	−0.342505
start_render_time	−0.291571
markup_validation	−0.266116
broken_link	−0.265718
compression	−0.255235

- Load Time (Correlation: −0.593609): Load time has a strong negative correlation with performance, indicating that longer load times are associated with poorer performance.
- Number of Requests (Correlation: −0.571076): Similarly, a higher number of requests negatively impacts performance, likely due to increased server load and latency.
- Document Complete Time (Correlation: −0.519318): This metric also shows a significant negative correlation, suggesting that pages taking longer to reach completion negatively affect user experience.
- Page Size (Correlation: −0.490623): Larger page sizes are correlated with lower performance, likely due to increased download times.
- Response Time (Correlation: −0.428263): Faster response times improve performance, as expected.
- Time to Interactive (Correlation: −0.342505): This metric measures the time taken for the page to become fully interactive, and shorter times are associated with better performance.
- Start Render Time (Correlation: −0.291571): Pages that start rendering more quickly tend to perform better.
- Markup Validation (Correlation: −0.266116): Better HTML markup validation (fewer errors) slightly improves performance.
- Broken Links (Correlation: −0.265718): The presence of broken links slightly negatively impacts performance, though less significantly than other factors.
- Compression (Correlation: −0.255235): Effective compression techniques correlate with improved performance by reducing load times and data transfer.

4.4.2. Feature Selection Strategy

Based on the correlation analysis, we prioritized features with the highest absolute correlation coefficients for inclusion in our predictive models. Features like Load Time, Number of Requests, Document Complete Time, and Page Size were identified as key predictors due to their strong negative correlations with performance (see Figure 9).

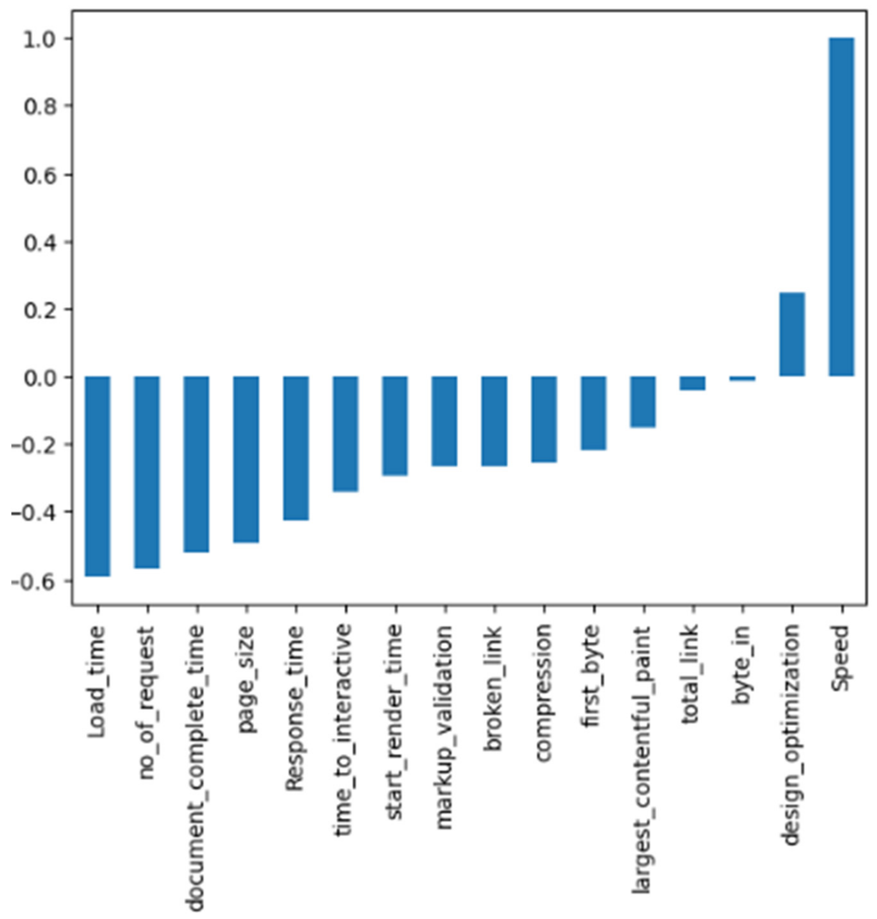


Figure 9. Feature importance for predicting website speed based on correlation analysis. Positive values indicate features positively correlated with speed, while negative values indicate negative correlations.

- High-Correlation Features: Features such as load time, number of requests, and document complete time were selected for their significant impact on performance. These features are critical as they directly influence the user's experience and are highly predictive of performance issues.
- Moderate-Correlation Features: Attributes like page size and response time, while slightly less correlated, still provide valuable information and were included in the model. These features often capture different aspects of the page-loading process that are not fully represented by high-correlation features.
- Low-Correlation Features: Features with lower correlation coefficients, such as markup validation, broken links, and compression, were considered less critical but still potentially useful. These features were included selectively based on their contribution to the overall model accuracy during preliminary testing.

4.4.3. Impact of Feature Selection

The application of feature selection significantly enhanced the performance of our classification models. By focusing on the most relevant features, we were able to

- Improve Model Accuracy: Models trained on the selected features demonstrated higher accuracy and better generalization to new data.
- Reduce Overfitting: By eliminating irrelevant or redundant features, we reduced the risk of overfitting, ensuring that the model performs well on unseen data.

- Enhance Interpretability: A reduced feature set simplifies the model, making it easier to interpret and understand the relationships between input variables and performance outcomes.

In conclusion, feature selection based on correlation analysis was pivotal in refining our predictive framework. It allowed us to build more efficient and effective models, ultimately will aid web developers and software engineers in predicting and optimizing web page performance.

5. Experiments and Results

5.1. Experiment Setup

All experiments were conducted using Python 3.10 within the Google Colab environment, ensuring an up-to-date and stable setup. The implementation relied on widely used libraries, including Scikit-learn (1.3.0) for machine learning models and preprocessing, NumPy (1.24) for numerical computations, Pandas (2.1.0) for data handling, XGBoost (1.7) for gradient boosting, and SciPy (1.11) for statistical analysis and optimization tasks. Data visualization was performed using Matplotlib (3.8) and Seaborn (0.13.0). These tools were chosen for their robustness, extensive community support, and proven application in similar research domains, ensuring the reliability, adaptability, and reproducibility of the experiments.

This study explores the effectiveness of various ML classification algorithms for web page performance evaluation. Two experiments are conducted, comparing model performance on both the original dataset and a feature-selected version.

Prior to employing ML techniques, the performance level of each web page within the dataset was manually assigned. This assignment was based solely on the web page speed feature.

The following criteria were used for performance-level labeling:

performance level = if (Speed $\geq$ 80 and Speed $\leq$ 100) then “Excellent”
  else if (Speed $\geq$ 70 and Speed $\leq$ 79) then “Good”
else “Unacceptable”

We utilize stratified ten-fold cross-validation [82] to assess the performance of our proposed tool. The dataset is randomly partitioned into ten folds, with nine of these folds used for training the classification model, and the remaining fold used for evaluating the model’s performance. This process is repeated ten times, and the average performance across all iterations is recorded. Stratified cross-validation is a widely adopted evaluation method in software engineering research [83,84], ensuring a balanced and comprehensive assessment of model performance.

This study explores the effectiveness of various machine learning (ML) classification algorithms for web page performance evaluation. Two experiments are conducted, comparing model performance on both the original dataset and a feature-selected version. The implementation includes nine models, such as Support Vector Machines (SVMs), Random Forest (RF), k-Nearest Neighbors (KNN), Naive Bayes, Multinomial Naive Bayes, Bayesian Network, Decision Tree, Logistic Regression, and AdaBoost Classifier, all of which are implemented using the Sklearn library package available in Python. To optimize the performance of these machine learning algorithms, several hyperparameters were carefully selected after an exhaustive experimentation phase and fine-tuned for each model. For the Random Forest classifier, 100 estimators were chosen with max_features = ‘sqrt’, a commonly used configuration for improved performance. The SVM model utilized a linear kernel with a regularization parameter C = 1.0, while the K-Nearest Neighbor (KNN) algorithm was set with 5 neighbors, a setting shown to perform well for a variety of tasks.

For Naive Bayes, both Gaussian and Multinomial variants were employed with default parameters, as they are effective for the given data type. The Decision Tree model was trained using default settings, but further tuning could enhance its depth. Logistic Regression and AdaBoost were applied with default configurations, though adjustments such as the regularization strength C for Logistic Regression and the learning rate for AdaBoost could optimize performance further. Table 4 provides a detailed summary of the hyperparameters used for each algorithm.

Table 4. This table summarizes the hyperparameter configurations used for each machine learning model applied in this study. It includes fine-tuned values (e.g., Random Forest with n_estimators = 100 and max_features = 'sqrt') as well as default settings (e.g., Logistic Regression with C = 1.0). The table also provides descriptions of each hyperparameter, helping to clarify their roles in model optimization.

Algorithm	**Hyperparameter**	**Value**	**Description**
Random Forest (RF)	Number of Estimators (n_estimators)	100	Number of trees in the forest.
	Maximum Features (max_features)	'sqrt'	The number of features considered when looking for the best split.
Support Vector Machine (SVM)	Kernel (kernel)	Linear	Specifies the kernel type to be used in the algorithm.
	Regularization (C)	1.0	Regularization parameter; the strength of the penalty is inversely proportional.
K-Nearest Neighbor (KNN)	Number of Neighbors (n_neighbors)	5	Number of neighbors to use for voting.
	Weight Function (weights)	Uniform	All points in each neighborhood are weighted equally.
Naive Bayes (Gaussian)	Smoothing Parameter (var_smoothing)	Default (1×10^{-9})	Portion of the largest variance of all features added to variances for stability.
Naive Bayes (Multinomial)	Alpha (alpha)	Default (1.0)	Additive smoothing parameter.
Bayesian Network	Configuration	Default	Utilized with default parameter settings.
Decision Tree	Criterion (criterion)	Default ('gini')	The function to measure the quality of a split.
	Maximum Depth (max_depth)	Default (None)	The maximum depth of the tree.
Logistic Regression	Regularization (C)	Default (1.0)	Inverse of regularization strength.
	Solver (solver)	Default ('lbfgs')	Algorithm to use in the optimization problem.
AdaBoost Classifier	Base Estimator (base_estimator)	Default (None)	If None, a Decision Tree with max_depth = 1 is used.
	Number of Estimators (n_estimators)	Default (50)	The maximum number of estimators.
	Learning Rate (learning_rate)	Default (1.0)	Weight applied to each classifier at each boosting iteration.

The selection of hyperparameters was based on a combination of recommendations from the literature and a systematic experimentation process. For example, the number of estimators in Random Forest (n_estimators = 100) is commonly suggested in the literature as a balance between performance and computational efficiency [80]. The max_features parameter ('sqrt') is widely used as it prevents overfitting by limiting the number of features considered at each split [85]. Similarly, the parameters for other algorithms (such as the kernel and regularization in SVM) were set after extensive trial and error, aiming to achieve optimal performance across different classifiers. These values were chosen after evaluating their impact on the model's accuracy and generalization through cross-validation, ensuring that the selected parameters provided the best trade-off between bias and variance for the task at hand.

5.2. Evaluation Metrics

To assess the predictive performance of our proposed approach and framework, we employ multi-label classification. A web page can either be correctly assigned to a specific performance label (true positive, TP_i) or misclassified as that label when it does not actually belong to it (false positive, FP_i). Alternatively, it may be incorrectly classified as not having a specific label (false negative, FN_i) or correctly classified as not having the label (true negative, TN_j). For each performance label, we calculate key metrics such as accuracy, precision, recall, and F-measure scores to evaluate the performance of our framework. Each label c_i is treated as a binary classification problem, allowing us to calculate these metrics to assess the model's overall effectiveness in predicting performance labels.

Accuracy: The ratio of correctly classified web pages (for all three labels) to total web pages is known as accuracy. We have one label, c_i:

$$Accuracy = B(TP_i, FP_i, FN_i, TN_i) = \frac{TP_i + TN_i}{TP_i + FP_i + TN + FN} \tag{5}$$

The average accuracy of all the labels can be represented as:

$$AvgAcc = \frac{1}{q}\sum_{i=1}^{q} B(TP_i, FP_i, FN_i, TN_i) \tag{6}$$

Precision: the percentage of web pages with the proper label (c_i) out of those with the exact label (c_i), i.e.,

$$P(c_i) = \frac{TP_i}{TP_i + FP_i} \tag{7}$$

Recall: the percentage of pages with the label "c$_i$" that are appropriately labeled, that is,

$$P(c_i) = \frac{TP_i}{TP_i + FN_i} \tag{8}$$

F-measure: is a composite metric that balances both precision and recall. It assesses whether an improvement in one of these metrics (e.g., precision) compensates for a decline in the other (e.g., recall). In the context of web page performance, the F-measure provides a unified evaluation of a model's ability to accurately classify performance labels, considering both the precision and recall trade-offs:

$$F(C) = \frac{2 * P(C) * R(C)}{P(C) + R(C)} \tag{9}$$

The metrics P(C) and R(C) represent the average precision and recall for all the labels, with equal weight given to each. Both precision and recall are crucial for evaluating web page performance prediction, as they provide insights into the effectiveness of our tool from two different perspectives. Low precision implies that developers may avoid using the tool due to a high rate of false positives [82]. On the other hand, low recall would also discourage use, as the tool would fail to predict the performance of many web pages accurately. There is an inherent tradeoff between precision and recall, where improving one often leads to a decline in the other. To address this, the F-measure, which is the harmonic mean of precision and recall, is used to determine whether enhancing one metric compensates for the decrease in the other. F-measure is widely adopted in software engineering literature as a comprehensive evaluation metric.

5.3. Experimental Results

To evaluate the practical applicability of our framework, we tested nine state-of-the-art classification algorithms on both the original dataset and a version with feature selection. The primary objectives of our experiments are twofold:

- Evaluate the effectiveness of the proposed framework on both the original dataset and a feature-selected version.
- Validate which classification algorithm has the best performance for realistic usage.

To the best of our knowledge, we are pioneers in applying classification algorithms for predicting web page performance in the accessible literature, and as such, we did not have any existing methods with which to compare our prediction framework.

5.3.1. Scenario 1

In this experiment, the original collected dataset is used. Figure 10 shows a comparison between state-of-the-art methods.

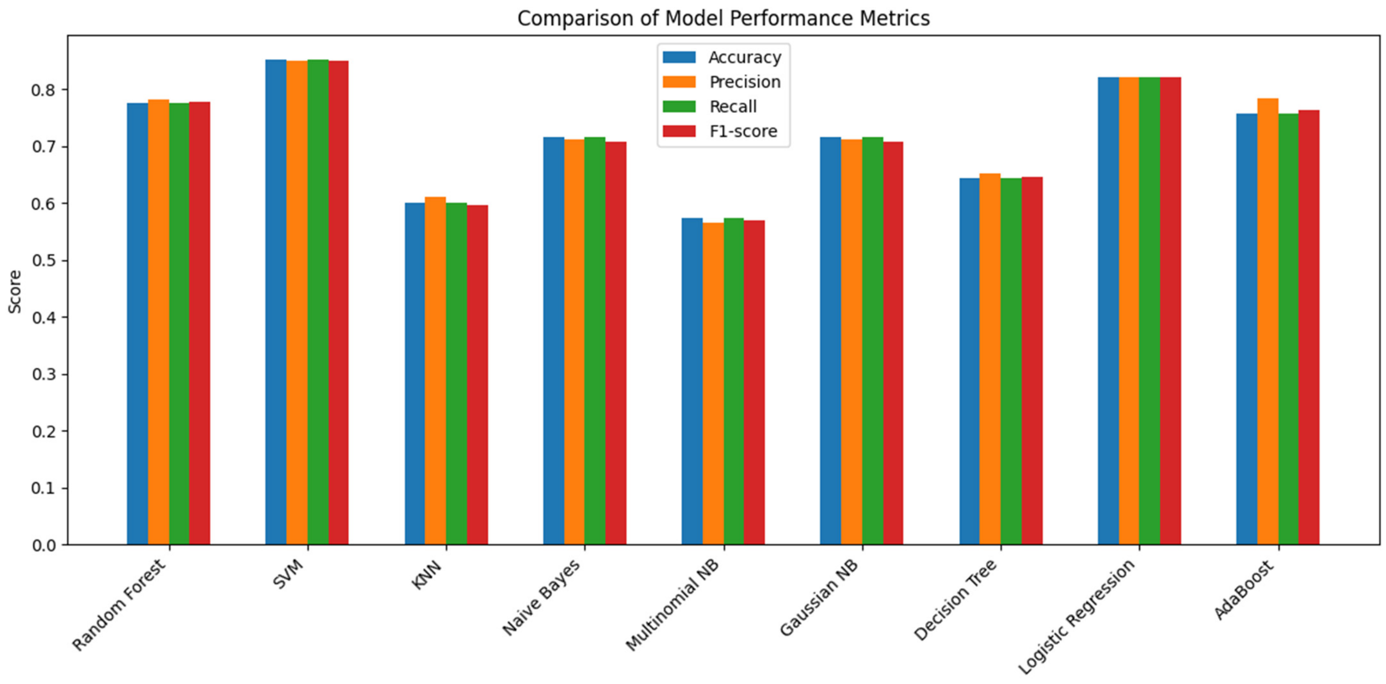


Figure 10. Experimental results on dataset without feature selection and a comparison between state-of-the-art methods.

Table 5 shows the accuracy, precision, recall and F-measure scores for nine classification algorithms, respectively. We can find that some classification algorithms present obviously better performance than others. In terms of accuracy, SVMs (0.87) presented the best performance, followed by logistic regression (0.84).

Table 5. The prediction performance of selected classifier on dataset without feature selection, measured using metrics such as accuracy, precision, recall, and F1-score.

Method	Acc. (%)	P (%)	R (%)	F1-Score
RF	80	81	80	80
SVMs	87	87	87	87
KNN	63	64	63	63
Naive Bayes	73	73	72	72
Naive Bayes multinomial	62	62	62	62
Bayesian network	73	73	73	72
DecisionTree	67	67	67	67
Logistic regression	84	84	84	84
AdaBoost	80	83	80	80

5.3.2. Scenario 2

In this experiment, a feature-selected dataset is used. Figure 11 shows a comparison between state-of-the-art methods.

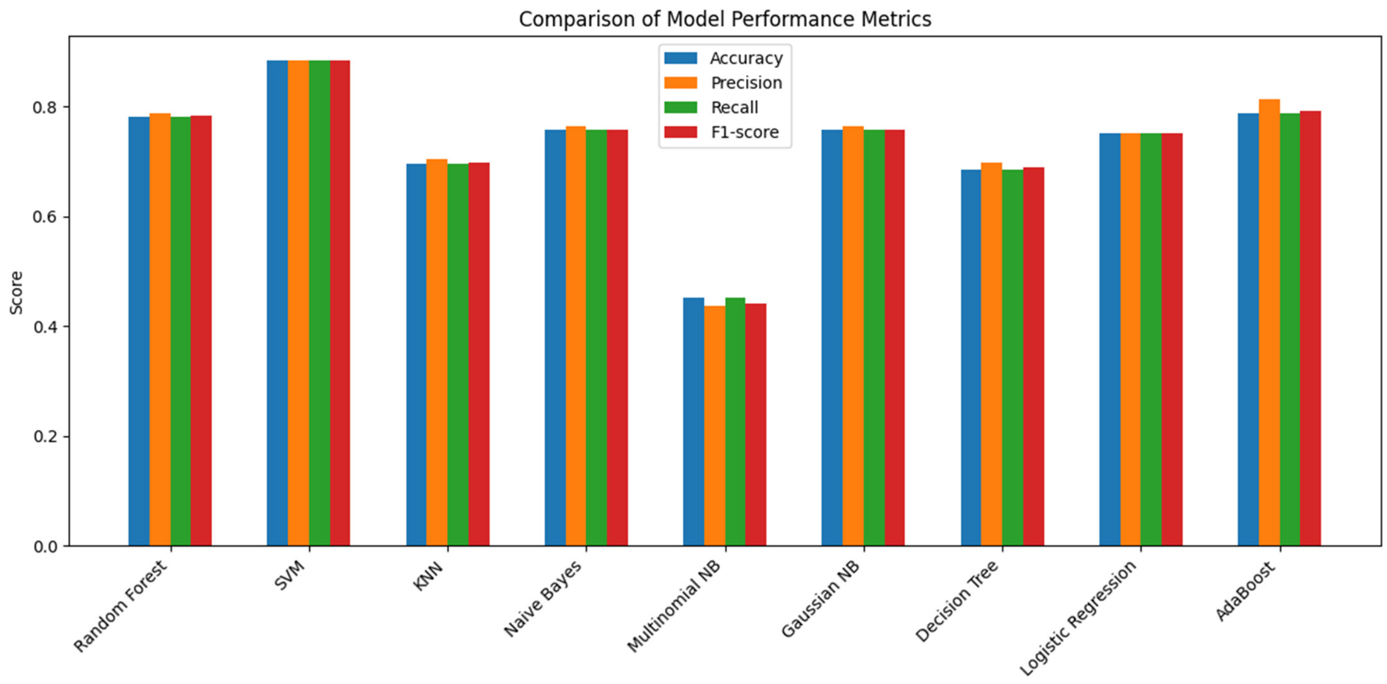


Figure 11. Experimental results on dataset with feature selection and a comparison between state-of-the-art methods.

Table 6 shows the accuracy, precision, recall and F-measure scores for nine classification algorithms on a feature-selected dataset, respectively. We can find that some classification algorithms present obviously better performance than others. In terms of accuracy, SVMs (0.89) presented the best performance, followed by Random Forest Classifier (0.81).

Table 6. The prediction performance of selected classifier on a dataset with feature selection, measured using metrics such as accuracy, precision, recall, and F1-score.

Method	**Acc. (%)**	**P (%)**	**R (%)**	**F1-Score**
RF	81	81	81	81
SVMs	89	89	89	89
KNN	68	69	68	68
Naive Bayes	74	75	74	74
Naive Bayes multinomial	49	48	49	48
Bayesian network	74	75	74	75
DecisionTree	69	69	69	69
Logistic regression	77	79	77	77
AdaBoost	77	80	77	77

5.3.3. Using Statistical Test

To assess if there are significant differences in predictive accuracy among the employed Machine Learning (ML) techniques for web page load time prediction, we utilized the Friedman test [86]. The Friedman test is a non-parametric test suitable when the assumptions for one-way ANOVA with repeated measures, like normality of residuals and constant error variance across treatments, are not guaranteed. In our case, the treatments represent different ML techniques, and individual web pages act as the subjects. The observations are the absolute residual errors in web page load time predicted by each ML algorithm. The test statistic of the Friedman test approximates a chi-square distribution with k-1 degrees of freedom, where k represents the number of ML techniques employed. A statistically significant result (p-value < α), where α is the chosen significance level (typically 0.05), indicates a difference in performance exists across the models. If a significant difference is found, post hoc pairwise comparisons can be conducted to pinpoint which specific pairs of ML techniques differ significantly.

The Friedman test yielded a statistic of 18.2295 (χ^2 (7)) with a p-value of 0.0196, which is statistically significant (p-value < 0.05). This suggests that at least one ML technique exhibits a statistically different predictive accuracy compared to others for web page load time prediction.

Following the significant Friedman test result, Wilcoxon signed-rank tests were conducted for pairwise comparisons between the ML techniques. Due to space limitations, only significant results (p-value < 0.05) are reported here.

- SVM outperformed Naive Bayes (p-value = 0.014).
- Random Forest outperformed Naive Bayes (p-value = 0.013) and Gaussian NB (p-value = 0.013).

Multinomial NB exhibited lower accuracy compared to both KNN (p-value = 0.026) and Naive Bayes (p-value = 0.026)

The results of the Friedman test revealed a statistically significant difference in the predictive performance of the employed ML techniques for web page load time prediction. Subsequent post hoc analysis using Wilcoxon signed-rank tests identified SVM as the model with the highest accuracy, statistically outperforming Naive Bayes. Additionally, Random Forest demonstrated a significant advantage over Naive Bayes and Gaussian NB. Conversely, Multinomial NB displayed the lowest accuracy among the techniques, performing significantly worse than both KNN and Naive Bayes (Table 7).

Table 7. Pairwise statistical comparisons of machine learning techniques for web page load time prediction.

Comparison	***p*-Value**
Random Forest vs. SVM	0.639
Random Forest vs. KNN	0.06
Random Forest vs. Naive Bayes	0.013
Random Forest vs. Multinomial NB	0.453
Random Forest vs. Gaussian NB	0.013
Random Forest vs. Decision Tree	0.986
Random Forest vs. Logistic Regression	0.793
Random Forest vs. AdaBoost	0.17
SVM vs. KNN	0.087
SVM vs. Naive Bayes	0.014
SVM vs. Multinomial NB	0.296
SVM vs. Gaussian NB	0.014
SVM vs. Decision Tree	0.773
SVM vs. Logistic Regression	0.879
SVM vs. AdaBoost	0.366
KNN vs. Naive Bayes	0.819
KNN vs. Multinomial NB	0.026
KNN vs. Gaussian NB	0.819
KNN vs. Decision Tree	0.117
KNN vs. Logistic Regression	0.123
KNN vs. AdaBoost	0.335
Naive Bayes vs. Multinomial NB	0.026
Naive Bayes vs. Decision Tree	0.049
Naive Bayes vs. Logistic Regression	0.027
Naive Bayes vs. AdaBoost	0.166
Multinomial NB vs. Gaussian NB	0.026
Multinomial NB vs. Decision Tree	0.44
Multinomial NB vs. Logistic Regression	0.368
Multinomial NB vs. AdaBoost	0.156
Gaussian NB vs. Decision Tree	0.049
Gaussian NB vs. Logistic Regression	0.027
Gaussian NB vs. AdaBoost	0.166
Decision Tree vs. Logistic Regression	0.859
Decision Tree vs. AdaBoost	0.268
Logistic Regression vs. AdaBoost	0.349

This study investigated the effectiveness of various Machine Learning techniques in predicting web page load time. The findings indicate that the chosen ML techniques exhibit statistically significant differences in their predictive accuracy. SVM emerged as the best performing model, followed by Random Forest. These results provide valuable insights for selecting appropriate ML techniques for web page load time prediction tasks.

6. Discussion

This section analyzes the performance of the classification algorithms used in this study, highlights the implications for practical usage, emphasizes the benefits of feature selection, and outlines the contributions of this research compared to previously published studies.

The comparative analysis of the algorithms revealed that Support Vector Machines (SVMs) consistently achieved superior performance across all metrics in both the original and feature-selected datasets. For the original dataset, SVM achieved an accuracy of 0.87, with equivalent precision, recall, and F1-score, demonstrating its robustness in handling the complexity of the data. Logistic Regression also performed well, with slightly lower but competitive metrics of 0.84 across all evaluation criteria. Random Forest and the AdaBoost classifier followed closely, each achieving an accuracy of 0.80. In contrast, k-Nearest Neighbors and Naive Bayes Multinomial were the least effective, with accuracies of 0.63 and 0.62, respectively.

The application of feature selection techniques further enhanced the performance of most algorithms, particularly SVMs and Random Forest. SVM achieved an improved accuracy of 0.89 with feature selection, alongside similar gains in precision, recall, and F1-score, underscoring the importance of preprocessing and feature engineering. Random Forest also demonstrated a slight improvement, achieving an accuracy of 0.81, while Bayesian Network and Naive Bayes exhibited competitive precision and recall, suggesting their utility in specific contexts. Although the k-Nearest Neighbor classifier showed some improvement with feature selection, it remained one of the least effective models.

These results underscore the superior capability of SVMs for predicting web page performance, especially when combined with an effective feature selection process. SVMs' ability to handle complex datasets and focus on significant predictors makes them the most reliable choice for this task. Logistic Regression and Random Forest also emerged as viable alternatives, offering balanced accuracy and interpretability, which can be advantageous for general applications. The Bayesian Network demonstrated high recall, making it suitable for identifying web pages with potential performance issues.

Feature selection played a critical role in improving the performance of the models. By reducing the dimensionality of the dataset and eliminating irrelevant or redundant features, feature selection allowed the algorithms to focus on the most informative predictors. This process not only enhanced the accuracy of the models but also demonstrated the value of preprocessing in improving the overall effectiveness of machine learning applications.

To evaluate the computational efficiency of the algorithms, both training and prediction times were measured using Python's timeit module. Random Forest showed a reasonable balance with a training time of 220 ms and a prediction time of 8.99 ms. While SVM required 489 ms for training, it achieved the highest accuracy of 89%, making it the most suitable choice despite being computationally intensive. KNN and Gaussian Naive Bayes exhibited minimal training times (2.14 ms and 2.2 ms, respectively), but with lower accuracy. Decision Tree and Multinomial Naive Bayes offered moderate efficiencies, with training times ranging from 3.17 ms to 5.86 ms. Logistic Regression faced convergence issues during training. Despite its higher computational cost, SVM's superior accuracy aligns with the study's focus on predictive performance.

The findings of this study have several practical implications. For general web page performance prediction, SVMs are recommended due to their consistent top performance across both datasets. Logistic Regression and Random Forest provide robust alternatives for achieving balanced performance across multiple metrics, making them suitable for a variety of real-world scenarios. For tasks requiring high recall, the Bayesian Network is particularly effective in identifying web pages with potential issues.

Compared to previous studies, this research offers several unique contributions. Unlike traditional methods that often rely on heuristic or rule-based approaches, this study demonstrates the efficacy of state-of-the-art classification algorithms in web page performance prediction, achieving significantly higher accuracy and reliability. The use of an effective feature selection process further distinguishes this work, as it highlights the impact of feature engineering in enhancing model performance—a factor often overlooked in prior research. Additionally, the comprehensive dataset and rigorous methodology employed in this study ensure the generalizability of the results, addressing limitations in earlier studies that relied on smaller or less diverse datasets. Finally, this paper introduces a practical, real-world application framework capable of handling diverse data and providing actionable insights for web developers.

7. Conclusions and Future Work

7.1. Recommendations and Future Work

This study highlights several key recommendations to guide the application and further development of web page performance prediction frameworks. First, the findings strongly advocate for prioritizing Support Vector Machines (SVMs) as the primary algorithm, given its superior performance across all evaluated datasets. Organizations and developers are encouraged to implement SVM as a reliable tool for predicting web page load times.

Additionally, the study underscores the importance of robust feature selection. Thorough analysis and selection of relevant features can significantly enhance model accuracy, efficiency, and interpretability. Expanding the range of performance metrics beyond load time, such as metrics that measure user interaction and engagement, would provide a more holistic understanding of web page performance.

To ensure the framework remains effective over time, continuous monitoring and adaptation are recommended. Regular updates reflecting advancements in web technologies and user behavior patterns will maintain the framework's relevance and utility. By following these recommendations, businesses and developers can achieve significant improvements in web page performance, leading to enhanced user satisfaction and measurable business outcomes.

Future advancements in web page performance prediction could benefit from expanding the diversity of datasets to include a broader range of web pages from different domains, industries, and regions. This would enhance the generalizability of models, making them applicable across various contexts. Additionally, exploring advanced algorithms such as ensemble methods, deep learning architectures, and hybrid models may uncover new pathways to achieving greater predictive accuracy and robustness.

Addressing cross-browser compatibility represents another essential avenue, ensuring that the proposed framework remains effective when applied to data from different web browsers. Real-world validation through deployment in live settings would provide valuable insights into the framework's impact on user experience and its operational performance. Furthermore, feature engineering processes must be refined and updated continuously to adapt to the evolving landscape of web technologies and ensure that the most relevant predictors are included.

Finally, incorporating user-centric metrics, such as perceived load time and other aspects of user experience, could further expand the framework's scope, bridging the gap between technical performance metrics and end-user satisfaction. By pursuing these directions, future research can contribute to a more comprehensive, adaptable, and impactful tool for optimizing web page performance.

7.2. Conclusions

This study proposed a comprehensive evaluation methodology for predicting web page performance through the application of state-of-the-art classification algorithms. Our comparative analysis identified Support Vector Machines (SVMs) as the most reliable model, achieving the highest performance metrics in both original and feature-selected datasets. This finding highlights the robustness and suitability of SVMs for web page performance prediction, alongside Logistic Regression and Random Forest, which also demonstrated significant potential as practical alternatives.

Moreover, our findings emphasize the critical importance of feature selection in enhancing model performance. By applying feature selection techniques, we were able to significantly improve the accuracy of most algorithms, particularly SVMs and Random Forest, by eliminating irrelevant or redundant features. This pre-processing step proved essential in directing the models toward the most impactful predictors, thereby enhancing their overall effectiveness.

In contrast to existing studies, our research provides a more nuanced understanding of machine learning techniques in the context of web performance evaluation. While prior works have primarily focused on individual performance metrics, our holistic approach integrates a broader range of performance indicators, including user experience, page responsiveness, and scalability across various devices and network conditions. This comprehensive framework not only bridges the gap between predictive modeling and optimization algorithms but also offers actionable insights for web developers and performance engineers.

The statistical analysis using the Friedman test and subsequent Wilcoxon signed-rank tests further corroborated our findings, revealing significant differences in predictive accuracy among various machine learning techniques. The superior performance of SVM and Random Forest underscores the necessity of integrating advanced machine learning approaches for effective web page performance assessment, setting a new benchmark in the field and paving the way for future research.

In summary, this study contributes to the literature by demonstrating the efficacy of machine learning models for web page performance prediction and highlighting the significance of feature selection, thereby providing a valuable resource for practitioners aiming to enhance user experience through informed design and development strategies.

Author Contributions: Conceptualization, M.G. and S.O.; methodology, M.G. and A.M.M.; software, M.G.; validation, A.M.M., M.G. and S.O.; formal analysis, A.M.M. and S.O.; investigation, M.G.; resources, M.G.; data curation, M.G.; writing—original draft preparation, M.G.; writ-ing—review and editing, A.M.M. and S.O.; visualization, M.G.; supervision, A.M.M. and S.O.; project administration, S.O. and A.M.M.; funding acquisition, A.M.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been developed under the grant PID2023-147409NB-C21, funded by the Spanish Ministerio de Ciencia Innovación y Universidades (Agencia Estatal de Investigación) MICIU/AEI/10.13039/501100011033, as well as by ERDF (European Union). The research has also been funded by projects TED2021-131699B-I00 and TED2021-129938B-I00 (MICU and AEI), as well as projects PID2020-113462RB-I00 and PID2020-115570GB-C22 of the Spanish Ministry of Economy and Competitiveness; project C-ING-179-UGR23 financed by the "Consejería de Universidades, Investigación e Inno-vación" (Andalusian Government, FEDER Program 2021-2027); and project PPJIA2023-031 (Plan Propio de Investigación y Transferencia UGR).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: https://github.com/ghattas1984/Approach-for-Evaluating-Web-Page-Performance-by-ML (accessed 9 January 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Retail Web Site Performance: Consumer Reaction to a Poor Online Shopping Experience. JupiterResearch and Akamai Report. 2006. Available online: https://blogs.constantcontact.com/wp-content/uploads/2011/12/site_abandonment_final_report.pdf (accessed on 2 May 2024).
2. Dixon, P. Shopzilla's site redo-you get what you measure. In Proceedings of the 2009 Web Performance and Operations Conference (Velocity), San Jose, CA, USA, 22–24 June 2009.
3. Forrest, B. Bing and google agree: Slow pages lose users. *Retrieved* **2009**, *5*, 2014.
4. Why Marketers Should Care About Mobile Page Speed. Available online: https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/mobile-page-speed-load-time/ (accessed on 2 May 2024).
5. Kumar, A.; Arora, A. A Filter-Wrapper based Feature Selection for Optimized Website Quality Prediction. In Proceedings of the 2019 Amity International Conference on Artificial Intelligence (AICAI), Dubai, United Arab Emirates, 4–6 February 2019; pp. 284–291. [CrossRef]
6. Grigorik, I. *High Performance Browser Networking: What Every Web Developer Should Know About Networking and Web Performance*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2013.
7. Why Web Performance Matters: Is Your Site Driving Customers Away?—PDF Free Download. Available online: https://docplayer.net/1736871-Why-web-performance-matters-is-your-site-driving-customers-away.html (accessed on 2 May 2024).
8. Willnecker, F.; Brunnert, A.; Gottesheim, W.; Krcmar, H. Using Dynatrace Monitoring Data for Generating Performance Models of Java EE Applications. In Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering, Austin, TX, USA, 31 January–4 February 2015; Association for Computing Machinery: New York, NY, USA, 2015; pp. 103–104. [CrossRef]
9. Chrome User Experience Report. Available online: https://developer.chrome.com/docs/crux?hl=ar (accessed on 2 May 2024).
10. End User Monitoring (EUM) | End-to-End User Insights. Available online: https://www.appdynamics.com/product/end-user-monitoring/index.html (accessed on 2 May 2024).
11. Application Monitoring | New Relic. Available online: https://newrelic.com/platform/application-monitoring (accessed on 2 May 2024).
12. WebPageTest. Available online: https://www.webpagetest.org/ (accessed on 2 May 2024).
13. Butkiewicz, M.; Madhyastha, H.V.; Sekar, V. Understanding website complexity: Measurements, metrics, and implications. In Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, Berlin, Germany, 2–4 November 2011; Association for Computing Machinery: New York, NY, USA, 2011; pp. 313–328. [CrossRef]
14. Denaro, G.; Polini, A.; Emmerich, W. Early performance testing of distributed software applications. In Proceedings of the 4th International Workshop on Software and Performance, Redwood Shores, CA, USA, 14–16 January 2004; Association for Computing Machinery: New York, NY, USA, 2004; pp. 94–103. [CrossRef]
15. Bondi, A.B. Incorporating Software Performance Engineering Methods and Practices into the Software Development Life Cycle. In Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering, Delft, The Netherlands, 12–16 March 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 327–330. [CrossRef]
16. Menascé, D.A.; Almeida, V.A.F.; Dowdy, L.W.; Dowdy, L. *Performance by Design: Computer Capacity Planning by Example*; Prentice Hall Professional: Saddle River, NJ, USA, 2004.
17. Smith, C.U.; Woodside, M. Performance validation at early stages of software development. In *System Performance Evaluation: Methodologies and Applications*; Taylor & Francis: Santa Fe, NM, USA, 1999; pp. 383–396.
18. Smith, C.U.; Williams, L.G. *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*; Addison-Wesley Reading: Redwood City, CA, USA, 2002.
19. Aggarwal, K.K. *Software Engineering*; New Age International: Delhi, India, 2005.
20. Balsamo, S.; Di Marco, A.; Inverardi, P.; Simeoni, M. Model-based performance prediction in software development: A survey. *IEEE Trans. Softw. Eng.* **2004**, *30*, 295–310. [CrossRef]
21. Nunnagoppula, H.; Katragadda, K.; Ramesh, M. Website Traffic Forecasting Using Deep Learning Techniques. In Proceedings of the 2023 International Conference on Artificial Intelligence and Smart Communication (AISC), Greater Noida, India, 27–29 January 2023. Available online: https://ieeexplore.ieee.org/document/10085005 (accessed on 23 October 2024).
22. Pasieka, N.; Sheketa, V.; Romanyshyn, Y.; Pasieka, M.; Domska, U.; Struk, A. Models, Methods and Algorithms of Web System Architecture Optimization. In Proceedings of the 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8–11 October 2019. Available online: https://ieeexplore.ieee.org/abstract/document/9061539 (accessed on 24 October 2024).
23. Zhou, J.; Zhang, Y.; Zhou, B.; Li, S. Predicting web page performance level based on web page characteristics. *IJWET* **2015**, *10*, 152. [CrossRef]
24. Correlating Performance Metrics to Page Characteristics—Analysis. Available online: https://discuss.httparchive.org/t/correlating-performance-metrics-to-page-characteristics/1548 (accessed on 2 May 2024).

25. Loosley: E-Commerce Response Time: A Reference Model—Google Scholar. Available online: https://scholar.google.com/scholar_lookup?title=E-commerce%20response%20time:%20a%20reference%20model&author=C.%20Loosley&publication_year=2000 (accessed on 2 May 2024).
26. Sevcik, P.; Wetzel, R. Pocket guide to application delivery systems. *Bus. Commun. Rev.* **2006**, *36*, 28.
27. Chiew, T.K. Web Page Performance Analysis. 2009. Available online: https://eleanor.lib.gla.ac.uk/record=b2660750 (accessed on 20 October 2024).
28. Zhi, J. Web page design and download time. In Proceedings of the Int. CMG Conference, Anaheim, CA, USA, 2–7 December 2001; Citeseer: San Mateo, CA, USA, 2001; pp. 689–704.
29. Nagarajan, S.N.; Ravikumar, S. Model for Predicting End User Web Page Response Time. *arXiv* **2012**, arXiv:1204.6304. [CrossRef]
30. Zatwarnicki, K.; Zatwarnicka, A. Estimation of Web Page Download Time. In *Computer Networks*; Kwiecień, A., Gaj, P., Stera, P., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 144–152. [CrossRef]
31. Li, Z.; Zhang, M.; Zhu, Z.; Chen, Y.; Greenberg, A.G.; Wang, Y.-M. WebProphet: Automating Performance Prediction for Web Services. In Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2010, San Jose, CA, USA, 28–30 April 2010; pp. 143–158.
32. A Elsater, S.A.E.; Dawood, A.E.A.A.; Mohamed Hussein, M.M.; Ali, M.A. Evaluating the Websites' Quality of Five and Four Star Hotels in Egypt. *Minia J. Tour. Hosp. Res. MJTHR* **2022**, *13*, 183–193. [CrossRef]
33. Adepoju, S.A.; Oyefolahan, I.O.; Abdullahi, M.B.; Mohammed, A.A. Integrated usability evaluation framework for university websites. *I-Manag. J. Inf. Technol.* **2019**, *8*, 40–48. [CrossRef]
34. Allison, R.; Hayes, C.; McNulty, C.A.M.; Young, V. A Comprehensive Framework to Evaluate Websites: Literature Review and Development of GoodWeb. *JMIR Form. Res.* **2019**, *3*, e14372. [CrossRef] [PubMed]
35. Alsulami, M.H.; Khayyat, M.M.; Aboulola, O.I.; Alsaqer, M.S. Development of an Approach to Evaluate Website Effectiveness. *Sustainability* **2021**, *13*, 13304. [CrossRef]
36. Amjad, M.; Tutul Hossain, M.; Hassan, R.; Rahman, M.A. Web Application Performance Analysis of ECommerce Sites in Bangladesh: An Empirical Study. *IJIEEB* **2021**, *13*, 47–54. [CrossRef]
37. Armaini, I.; Dar, M.H.; Bangun, B. Evaluation of Labuhanbatu Regency Government Website based on Performance Variables. *Sink. J. Dan Penelit. Tek. Inform.* **2022**, *7*, 760–766. [CrossRef]
38. Aziz, U.A.; Wibisono, A.; Nisafani, A.S. Measuring the quality of e-commerce websites using analytical hierarchy process. *TELKOMNIKA (Telecommun. Comput. Electron. Control)* **2019**, *17*, 1202–1208. [CrossRef]
39. Barus, A.C.; Sinambela, E.S.; Purba, I.; Simatupang, J.; Marpaung, M.; Pandjaitan, N. Performance Testing and Optimization of DiTenun Website. *J. Appl. Sci. Eng. Technol. Educ.* **2022**, *4*, 45–54. [CrossRef]
40. Kulkarni, R.B.; Dixit, S.K. Empirical and Automated Analysis of Web Applications. *IJCA* **2012**, *38*, 1–8. [CrossRef]
41. Cai, X.; Li, S.; Feng, G. Evaluating the performance of government websites: An automatic assessment system based on the TFN-AHP methodology. *J. Inf. Sci.* **2020**, *46*, 760–775. [CrossRef]
42. Devi, K.; Sharma, A.K. Framework for Evaluation of Academic Website. *Int. J. Comput. Tech.* **2016**, *3*, 234–239.
43. Dhiman, P.; Anjali. Empirical validation of website quality using statistical and machine learning methods. In Proceedings of the 2014 5th International Conference—Confluence The Next Generation Information Technology Summit (Confluence), Noida, India, 25–26 September 2014; pp. 286–291. [CrossRef]
44. Dominic, P.D.D.; Jati, H. A comparison of Asian airlines websites quality: Using a non-parametric test. *Int. J. Bus. Innov. Res.* **2011**, *5*, 599–623. [CrossRef]
45. Dominic, P.D.D.; Jati, H.; Kannabiran, G. Performance evaluation on quality of Asian e-government websites—an AHP approach. *Int. J. Bus. Inf. Syst.* **2010**, *6*, 219–239. [CrossRef]
46. Erokhin, A.G.; Vanina, M.F.; Frolova, E.A. Application of Mathematical Simulation Methods for Evaluating the Websites Effectiveness. In Proceedings of the 2019 Systems of Signals Generating and Processing in the Field of on Board Communications, Moscow, Russia, 20–21 March 2019; pp. 1–5. [CrossRef]
47. Faustina, F.; Balaji, T. Evaluation of universities websites in Chennai city, India using analytical hierarchy process. In Proceedings of the 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, India, 3–5 March 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 112–116. [CrossRef]
48. Gangurde, R.; Kumar, B. Web Page Prediction Using Genetic Algorithm and Logistic Regression based on Weblog and Web Content Features. In Proceedings of the 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2–4 July 2020; pp. 68–74. [CrossRef]
49. Gharibe Niazi, M.; Karbala Aghaei Kamran, M.; Ghaebi, A. Presenting a proposed framework for evaluating university websites. *Electron. Libr.* **2020**, *38*, 881–904. [CrossRef]
50. Harshan, R.K.; Chen, X.; Shi, B. Analytic Hierarchy Process (AHP) Based Model for Assessing Performance Quality of Library Websites. *Inf. Technol. J.* **2016**, *16*, 35–43. [CrossRef]

51. Hidayah, N.A.; Subiyakto, A.; Setyaningsih, F. Combining Webqual and Importance Performance Analysis for Assessing a Government Website. In Proceedings of the 2019 7th International Conference on Cyber and IT Service Management (CITSM), Jakarta, Indonesia, 6–8 November 2019; pp. 1–6. [CrossRef]
52. Jati, H. Quality Ranking of E-Government Websites—PROMETHEE II Approach. In *Proceedings of the International Conference on Informatics for Development 2011 (ICID 2011)*; Universitas Islam Indonesia: Yogyakarta, Indonesia, 2011; pp. 39–45.
53. Kabassi, K. Analytic Hierarchy Process for website evaluation. *Intell. Decis. Technol.* **2018**, *12*, 137–148. [CrossRef]
54. Kinnunen, M. Evaluating and Improving Web Performance Using Free-To-Use Tools. Available online: https://oulurepo.oulu.fi/handle/10024/15601 (accessed on 18 February 2024).
55. Kumar, N.; Kumar, S.; Rajak, R. Website Performance Analysis and Evaluation using Automated Tools. In Proceedings of the 2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT), Mysuru, India, 10–11 December 2021; pp. 210–214. [CrossRef]
56. Kwangsawad, A.; Jattamart, A.; Nusawat, P. The Performance Evaluation of a Website using Automated Evaluation Tools. In Proceedings of the 2019 4th Technology Innovation Management and Engineering Science International Conference (TIMES-iCON), Bangkok, Thailand, 11–13 December 2019; pp. 1–5. [CrossRef]
57. Najadat, H.; Al-Badarneh, A.; Alodibat, S. A review of website evaluation using web diagnostic tools and data envelopment analysis. *Bull. Electr. Eng. Inform.* **2021**, *10*, 258–265. [CrossRef]
58. Olaleye, S.A.; Sanusi, I.T.; Ukpabi, D.C.; Okunoye, A. Evaluation of Nigeria Universities Websites Quality: A Comparative Analysis. Available online: https://oulurepo.oulu.fi/handle/10024/23263 (accessed on 17 February 2024).
59. Saleh, A.H.; Yusoff, R.C.M.; Bakar, N.A.A.; Ibrahim, R. Systematic literature review on university website quality. *IJEECS* **2022**, *25*, 511–520. [CrossRef]
60. Shayganmehr, M.; Montazer, G.A. Identifying Indexes Affecting the Quality of E-Government Websites. In Proceedings of the 2019 5th International Conference on Web Research (ICWR), Tehran, Iran, 24–25 April 2019; pp. 167–171. [CrossRef]
61. Shayganmehr, M.; Montazer, G.A. A Novel Model for Assessing e-Government Websites Using Hybrid Fuzzy Decision-Making Methods. *Int. J. Comput. Intell. Syst.* **2021**, *14*, 1468–1488. [CrossRef]
62. Wang, X.S.; Balasubramanian, A.; Krishnamurthy, A.; Wetherall, D. Demystifying Page Load Performance with {WProf}. In Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), Lombard, IL, USA, 2–5 April 2013.
63. Navigation Timing. Available online: https://www.w3.org/TR/navigation-timing/ (accessed on 4 May 2024).
64. Paint Timing. Available online: https://www.w3.org/TR/paint-timing/ (accessed on 4 May 2024).
65. Olshefski, D.; Nieh, J.; Agrawal, D. Using certes to infer client response time at the web server. *ACM Trans. Comput. Syst.* **2004**, *22*, 49–93. [CrossRef]
66. Wei, J.; Xu, C.-Z. Measuring Client-Perceived Pageview Response Time of Internet Services. *IEEE Trans. Parallel Distrib. Syst.* **2011**, *22*, 773–785. [CrossRef]
67. GTmetrix | Website Performance Testing and Monitoring. Available online: https://gtmetrix.com/ (accessed on 4 May 2024).
68. Analytics | Home. Available online: https://analytics.google.com/analytics/web/?pli=1#/p327966253/reports/intelligenthome (accessed on 4 May 2024).
69. web.dev. Available online: https://web.dev/ (accessed on 4 May 2024).
70. Web Debugging Proxy and Troubleshooting Tools | Fiddler. Available online: https://www.telerik.com/fiddler (accessed on 4 May 2024).
71. YSlow—Official Open Source Project Website. Available online: https://yslow.org/ (accessed on 4 May 2024).
72. PageSpeed Insights. Available online: https://pagespeed.web.dev/?utm_source=psi&utm_medium=redirect (accessed on 4 May 2024).
73. Overview | Lighthouse. Available online: https://developer.chrome.com/docs/lighthouse/overview (accessed on 4 May 2024).
74. McCallum, A.; Nigam, K. A comparison of event models for naive bayes text classification. In Proceedings of the AAAI-98 Workshop on Learning for Text Categorization, Madison, WI, USA, 26–27 July 1998; pp. 41–48.
75. Wu, X.; Kumar, V.; Ross Quinlan, J.; Ghosh, J.; Yang, Q.; Motoda, H.; McLachlan, G.J.; Ng, A.; Liu, B.; Yu, P.S. Top 10 algorithms in data mining. *Knowl. Inf. Syst.* **2008**, *14*, 1–37. [CrossRef]
76. Koller, D.; Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*; MIT Press: Cambridge, MA, USA, 2009.
77. Bishop, C.M. Pattern recognition and machine learning. *Springer Google Sch.* **2006**, *2*, 645–678.
78. Freund, Y.; Schapire, R.E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.* **1997**, *55*, 119–139. [CrossRef]
79. Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *24*, 123–140. [CrossRef]
80. Wongvorachan, T.; He, S.; Bulut, O. A Comparison of Undersampling, Oversampling, and SMOTE Methods for Dealing with Imbalanced Classification in Educational Data Mining. *Information* **2023**, *14*, 54. [CrossRef]

81. Hamadouche, S.; Boudraa, O.; Gasmi, M. Combining Lexical, Host, and Content-based features for Phishing Websites detection using Machine Learning Models. *EAI Endorsed Trans. Scalable Inf. Syst.* **2024**, *11*, 1–15. [CrossRef]
82. Han, J.; Pei, J.; Tong, H. *Data Mining: Concepts and Techniques*; Morgan Kaufmann: Burlington, MA, USA, 2022.
83. Xia, X.; Feng, Y.; Lo, D.; Chen, Z.; Wang, X. Towards more accurate multi-label software behavior learning. In Proceedings of the 2014 Software Evolution Week—IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE), Antwerp, Belgium, 3–6 February 2014; pp. 134–143. [CrossRef]
84. Xia, X.; Lo, D.; Wang, X.; Zhou, B. Tag recommendation in software information sites. In Proceedings of the 2013 10th Working Conference on Mining Software Repositories (MSR), San Francisco, CA, USA, 18–19 May 2013; pp. 287–296. [CrossRef]
85. Joloudari, J.H.; Marefat, A.; Nematollahi, M.A.; Oyelere, S.S.; Hussain, S. Effective Class-Imbalance Learning Based on SMOTE and Convolutional Neural Networks. *Appl. Sci.* **2023**, *13*, 4006. [CrossRef]
86. Friedman, M. A Comparison of Alternative Tests of Significance for the Problem of m Rankings. *Ann. Math. Stat.* **1940**, *11*, 86–92. [CrossRef]