ICT Express xxx (xxxx) xxx

Contents lists available at ScienceDirect

ICT Express

journal homepage: www.elsevier.com/locate/icte

Machine learning models in web applications: A comprehensive review

Kshiteesh Mani, Ajitha K.B. Shenoy *

Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, India

ARTICLE INFO

Keywords:
Artificial intelligence
Machine learning
Node.js
Python
Web applications
Web application firewalls

ABSTRACT

The rapid growth of web applications has increased the need for advanced features and strong security. Artificial intelligence (AI) and machine learning (ML) models play a crucial role in meeting these needs by improving efficiency and enhancing security. However, integrating these models into web applications can be challenging due to complex implementation and potential security risks. This paper compares Python and Node.js, two popular technology stacks, to determine their effectiveness in integrating ML models into web applications. It also explores the role of web application firewalls (WAF) and the ML algorithms that support them, analyzing current trends in their use and adoption. The overarching objective is to discern the technology stack that provides superior support for back-end ML integration and to identify the ML algorithms that are most effective in enhancing WAF capabilities against sophisticated security threats. By offering a synthesis of technical and security insights, this research seeks to empower developers and cybersecurity practitioners with the knowledge required to make well-informed decisions regarding technology stack selection and the implementation of ML-driven security mechanisms in web application development.

1. Introduction

As web applications continue to evolve into complex, user-centric platforms, the integration of artificial intelligence (AI) and machine learning (ML) has become critical for enhancing functionality and fortifying security. According to Tiwari et al. [1], incorporating AI and ML into web applications not only streamlines development and operational workflows but also significantly strengthens defense mechanisms against sophisticated attacks through adaptive and proactive security measures.

Beyond security, AI and ML have transformed user experience by enabling intelligent personalization and seamless automation. WebsitePulse [2] underscores how these technologies facilitate dynamic content delivery, optimize performance, and automate routine tasks, ultimately leading to more engaging and responsive applications. Similarly, Tranxndweb [3] highlights that enterprise-level web solutions increasingly rely on ML-driven analytics and intelligent components to achieve high scalability and deliver customized services at scale.

Despite these advantages, integrating ML models into web application backends introduces engineering complexities, including scalability, low-latency serving, and continuous model maintenance. Additionally, ensuring robust security against evolving cyber threats requires sophisticated approaches such as machine learning-augmented web application firewalls (WAFs).

This paper provides a comparative analysis of Python and Node.js technology stacks for ML integration, evaluating their strengths in supporting scalable, secure, and maintainable web architectures. It also reviews state-of-the-art ML algorithms used in enhancing WAF capabilities. By synthesizing these perspectives, the study aims to assist developers and cybersecurity professionals in making informed decisions on technology stack selection and security-focused ML deployment.

2. Literature review

This section is devoted to systematic review of the machine learning models used in web applications. The literature survey is divided in to two parts:

1. A thorough analysis of Machine Learning and Deep Learning techniques for detecting web attacks and assessing vulnerabilities, coupled with an extensive review of ML-driven enhancements to Web Application Firewalls (WAF) and Firewall Systems.
2. A detailed study of ML Model Integration into Web Applications using Backend Frameworks.

* Corresponding author.
E-mail address: ajith.shenoy@manipal.edu (Ajitha K.B. Shenoy).

https://doi.org/10.1016/j.icte.2025.09.001
Received 16 April 2025; Received in revised form 14 July 2025; Accepted 3 September 2025
Available online 12 September 2025

Please cite this article as: Kshiteesh Mani, Ajitha K.B. Shenoy, *ICT Express,* https://doi.org/10.1016/j.icte.2025.09.001

2.1. Machine learning and deep learning techniques for web attack detection and vulnerability analysis

Nowadays Machine learning and Deep learning models are used extensively to monitor and web applications and to detect anamolies. Deep learning techniques are well studied in the literature for web attack detection and vulnerability analysis. This subsection is devoted for literature review of the articles published in this area. The recent articles published in this area (from 2019 to 2024) are listed in Table 1 and Table 2 respectively. Table 1 summarizes ML/DL techniques for web attack detection and vulnerability and analysis, and Table 2 summarizes ML-based enhancements to WAF/Firewall Systems (works that specifically improve or integrate machine learning within Web Application Firewalls or related firewall technologies).

In the realm of security considerations for web applications, Web Application Firewalls (WAFs) have garnered significant attention, as evidenced by the scholarly works of Tekerek and Bay [21] and Kumar and Ponsam [22], both dedicated to fortifying the security posture of web applications.

Tekerek and Bay, present a pioneering approach with their hybrid learning-based WAF model, seamlessly integrating Signature-Based Detection (SBD) [23] and Anomaly-Based Detection (ABD) methodologies. SBD efficiently pinpoints known attack types like SQL Injection and Cross-Site Scripting through pre-established signatures [24]. Meanwhile, ABD, implemented via Artificial Neural Networks (ANN), focuses on discerning anomalous HTTP requests by scrutinizing features such as Alphanumerical Character Analysis, Letter Frequency Analysis, and Request Length Analysis. This synergistic methodology proves instrumental in preempting web-based attacks, thereby bolstering the overall security framework of web applications.

In a parallel vein, Kumar and Ponsam direct their efforts towards elevating web application security by integrating diverse machine learning (ML) models with a WAF. Serving as the primary defense layer, the WAF diligently monitors and filters incoming traffic to neutralize common attack patterns. The research advocates the amalgamation of ML algorithms such as Convolutional Neural Network (CNN) [25,26], Support Vector Machine (SVM), K-nearest neighbour (KNN), Logistic Regression, Random Forest (RF) Classifier, and Decision Tree Classifiers [27] to enhance the WAF's capabilities. These ML models, leveraging extensive datasets, enable the detection of intricate attack patterns and facilitate adaptation to emerging threats.

Further contributing to this discourse, the paper titled "Machine Learning Based Web Application Firewall" [28] introduces an ML-centric strategy to enhance web application security, specifically through an anomaly-based firewall. Acknowledging the susceptibility of web applications, the study employs character n-gram and tf-idf methods for feature extraction. The proposed model, employing a linear Support Vector Machine (SVM), attains a commendable detection rate of 99.53% on the CSIC 2010 dataset. In essence, the research pivots towards harnessing ML techniques, particularly character n-gram and tf-idf, to craft a resilient web application firewall, thereby fortifying defenses against potential cyber threats. The detailed analysis of ML-based enhancements to WAF/Firewall systems is summarized in Table 2.

The next subsection will provide detailed study of ML model integration into web applications using backedn frameworks.

2.2. ML model integration into web applications using backend frameworks

The integration of AI approaches in web development to automate and improve different stages of the development process is explored in the paper "How Artificial Intelligence Can Improve Web Development and Testing" [38]. Hand-drawn sketches are converted into HTML wireframes using AI technologies like machine learning (ML) and computer vision, greatly cutting down on design time. Artificial intelligence (AI)-driven testing technologies such as Sikuli and WebSee use computer vision to automate regression testing, cross-browser compatibility checks, and issue discovery. Key findings show that AI is capable of producing precise code from graphical inputs, finding and fixing defects, and maintaining test suites with little assistance from humans. The benefits include shorter development cycles, improved problem-solving accuracy, less manual labor, and an increased capacity to manage challenging development tasks, all of which eventually result in more reliable and user-friendly web applications.

Verma et al. [39] introduced a college community web application that utilizes React.js for frontend development and Django for backend functionalities. This system seamlessly integrates machine learning capabilities through Flask, serving as an API endpoint within the Django backend architecture. The application showcases the deployment of a sophisticated NLP model, enabling real-time text analysis, sentiment analysis, question categorization, and detection of inappropriate language. This integration plays a pivotal role in facilitating user-generated text analysis and content moderation within the platform.

Rahutomo et al. [40] conducted research in Indonesia's horticulture sector, implementing Agricultural 4.0 technology with a focus on AI for enhanced efficiency. The tech stack involves Python and Flask for web-based application development, employing Keras-RetinaNet for AI training models. The application utilizes a Web Server Gateway Interface (WSGI) [41] for AI model deployment and incorporates crowd-sourcing [42] for gathering extensive training data. The AI model, centered on Keras-RetinaNet, specializes in object detection and employs Python with the Keras framework. Deployment is achieved through WSGI, middleware capabilities, and user authentication, with the application hosted on an Apache 2.0 web server using mod_wsgi.

Sujatha et al. [43] project centers on developing a loan prediction system within a web application using Python's Flask framework for the backend. HTML, CSS, and JavaScript contribute to the front end. The ML model, based on logistic regression from Scikit-Learn, enhances decision-making in banking and insurance by accurately predicting loan outcomes. Data preprocessing, cleaning, visualization, and statistical analysis are performed using Pandas, NumPy, Seaborn, and Matplotlib libraries, showcasing the integration of ML techniques in this context.

Chauhan et al. [44] describe a web application using ML to classify music genres. The tech stack includes Python, NumPy, and Scikit-learn for dataset normalization and feature extraction. Librosa is employed to extract audio features from the GTZAN dataset, and three classification techniques–Support Vector Machine (SVM) [45,46], K Nearest Neighbours (KNN) [47], and a Deep Neural Network (DNN) [48] built with Keras–are investigated. Django forms the core of the web application's backend, providing an integrated solution for music genre classification by combining web application development with machine learning algorithms.

The research paper titled "Web App for Business Needs" [49], introduces a web application named "START KARO", designed to be a comprehensive solution for aspiring entrepreneurs seeking success in their ventures. Built using the MERN (MongoDB, Express.js, React, Node.js) stack, the application seamlessly integrates a Python-based Machine Learning (ML) model. This ML model, employing a Classification Algorithm under Supervised Learning, plays a pivotal role in automating the tasks of a business analyst, assisting users in selecting optimal business locations based on various criteria. The Python-Flask server serves as the backend for the ML model, handling requests from the React front end. The Express.js server, part of the MERN stack, acts as a bridge facilitating communication between the React front end and the Python-Flask server.

The study by Singh et al. [50] investigates how to use the IRIS database to integrate a machine-learning model into an online application for classifying iris flowers. The Flask web framework, which provides an intuitive user interface for seamless flower recognition, is

Table 1
ML/DL techniques for web attack detection and vulnerability analysis.

Title of the paper	Year	Methodology	ML Algorithm/Backend Tech	Results/key findings
Detecting web attacks with end-to-end deep learning [4]	2019	Unsupervised/semi-supervised approach using stacked denoising autoencoders on call graphs	Stacked Denoising Autoencoders, PCA, One-Class SVM	Detects SQLi, XSS, and deserialization attacks effectively; reduces false positives, maintaining high accuracy.
Predicting web vulnerabilities in web applications based on machine learning [5]	2019	NMPREDICTOR: Two-tier ML approach using software metrics and text features	J48, Naive Bayes, Random Forest (with meta-classifier)	Achieves higher F-measure; meta-classifier improves accuracy over individual classifiers.
Cybersecurity data science: An overview from machine learning perspective [6]	2020	Literature review; multi-layered ML-based cybersecurity framework	Naive Bayes, Decision Trees, SVM, Logistic Regression, K-Means, CNN, RNN, LSTM, RL, GA	ML automates threat detection, supports incremental learning, enhances web app security with adaptive models.
Web application attacks detection using deep learning [7]	2021	Two-step learning: Pre-train RoBERTa on HTTP requests, then OCSVM for anomaly detection	RoBERTa (DL for feature extraction), One-Class SVM	Outperforms ModSecurity; achieves higher TPR and lower FPR; reduces need for expert features.
Brain inspired spiking neural networks for WiFi based human activity recognition [8]	2021	WiFi-based HAR using Spiking Neural Networks and computer vision	Spiking Neural Networks, Bi-LSTM, LSTM, GRU	Good accuracy; potential future use of more complex ML/DL models for HAR.
PHIBOOST - A novel phishing detection model using adaptive boosting approach [9]	2021	Feature selection and ML models for phishing detection	Adaptive Boosting	Good accuracy; future work on more complex ML/DL models for cyberattacks.
SQL Injection, Cross-site scripting and buffer overflow attacks detection using machine learning [10]	2022	ML classification of malicious vs benign API calls	KNN, Naive Bayes, Decision tree, SVM, random forest	Four algorithms near 100% accuracy; Naïve Bayes lags with 59% accuracy.
An attack detection framework based on BERT and deep learning [11]	2022	Identifying anomalous HTTP requests	BERT and NLP	Proposed approach has success rate of 99.98% and F1 score of 0.987. Also, the detection is lower compared to other methods in the literature i.e.,0.4 ms.
Detection of cross-site scripting (XSS) attacks using machine learning techniques: a review [12]	2023	Literature review of ML-based XSS detection techniques	Decision Trees, Random Forest, SVM, Naive Bayes, K-means, CNN, LSTM, DNN	ML improves detection accuracy, reduces false positives; highlights shift from traditional to ML methods.
Front-end deep learning web apps development and deployment: a review [13]	2023	Review of front-end deployment of DL models in browsers	CNN, RNN, LSTM, VAEs (via TensorFlow.js)	Front-end deployment improves accessibility, reduces server costs, enhances privacy, real-time interaction.
Detection of network attacks using machine learning and deep learning models [14]	2023	identifying network intrusion and attack types using machine learning and deep learning models	decision tree, AdaBoost, XG Boost, Deep learning model	able to achieve 99% accuracy UNSW-NB15 dataset of 49 features for nine different attack samples.
Vulnerability detection through machine learning-based fuzzing: A systematic review [15]	2024	enhancing traditional fuzzing approaches using Deep learning, reinforcement learning and deep reinforcement learning techniques	LSTM, GAN, Seq2Seq, Generative Randomized Unit (GRU)	DNN can enhance the performance of fuzzing techniques significantly.
A comprehensive review on detection of cyber-attacks: Data sets, methods, challenges, and future research directions [16]	2024	tools and techniques for automating the detection of cyber attacks and prediction of attack types using ML models	feature selection, dimensionality reduction, classification	Listed the challenges involved in using machine learning models for classifying and predicting network attacks.
A comprehensive evaluation of machine learning algorithms for web application attack detection with knowledge graph integration [17]	2024	ML algorithms with Knowledge Graphs for threat detection	19 traditional ML algorithms, Neural Networks	High performance on known datasets; discusses complexity of KG integration and future enhancements, addresses overfitting and generalization issues.
A Comparative Analysis of Deep Learning Approaches for Enhancing Security in Web Application [18]	2024	detecting web attacks and code vulnerability using deep learning models	LSTM, Autoencoders and other deep learning models	importance of improving JavaScript's security
QUIC website fingerprinting based on automated machine learning [19]	2024	QUIC WF technique based on Automated Machine Learning (AutoML)	AutoML	Proposed technique outperforms state-of-the-art WF techniques. F1-score : 0.9979 and precision: 0.926.
A study of the relationship of malware detection mechanisms using Artificial Intelligence [20]	2024	Shallow and Deep learning models to detect malware.	Few-shot, One-shot, Zero-shot, and learning methods, GAN, XAI, Local Interpretable Model-Agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP) are used to explain the predictions of AI models.	To understand the research flow in the field of AI-based malware detection. Also, to understand various future research directions.

Table 2
ML-based enhancements to WAF/Firewall systems.

Title of the paper	Year	Methodology	ML Algorithm/Backend Tech	Results/key findings
Improving efficiency of Web Application Firewall to detect code injection attacks with Random Forest method [29]	2020	WAF enhancement using tf-idf and feature analysis	Random Forest	Improves detection accuracy, high TPR/low FPR on CSIC 2010 dataset.
WAF A MoLE: An adversarial tool for assessing ML-based WAFs [30]	2020	WAF breaching tool which uses guided mutational based fuzzing to generate adversarial examples. Machine learning based WAFs	WAF Brain, Token based models, SQLiGoT	Generate adversarial examples for ML based WAFs, Security assessment of the WAFs. Future Direction: Apply WAF A MoLE to commercial WAFs.
Improving Web Application Firewalls with automatic language detection [31]	2020	Rule based WAF and CNN for language classification	CNN, tokenization, word2vec	Near perfect TPR, reduces false positives vs ModSecurity alone.
Machine learning based Web Application Firewall [28]	2021	Anomaly-based WAF with NLP	SVM, KNN, Decision Trees, character n grams	Character n grams with linear SVM yield high accuracy for web attack detection.
Machine learning based model to identify Firewall decisions to improve cyber-defense [32]	2021	Analyzing firewall logs with shallow neural networks	SNN, Optimized Decision Trees, Bi LSTM	Bi LSTM hybrid achieves 97.38% accuracy; improves firewall data classification.
Web Application Firewall Using Machine Learning and Features Engineering [33]	2022	ML based WAF using feature engineering	Naive Bayes, Logistic Regression, Decision Tree, SVM	Achieves up to 99.6% accuracy; effective feature engineering improves classification.
Detection of payload injection in Firewall Using Machine Learning [34]	2023	Collecting XSS payload data, analyzing firewall logs	KNN, Decision Trees, Naive Bayes	Decision Tree achieves high accuracy in detecting malicious payloads.
Securing Web Application using Web Application Firewall (WAF) and Machine Learning [22]	2023	ML based anomaly detection in WAF	CVM, KNN, SVM, Decision Trees	Decision Tree classifier: 99.86% accuracy; dynamic and intelligent anomaly based WAF.
An Efficient Machine Learning-Based Web Application Firewall with Deep Automated Pattern Categorization [35]	2023	ML based WAF with automated pattern categorization	Logistic Regression, CNN	Demonstrates need and benefits of ML-assisted WAFs in organizations and industries.
Optimizing real time performance in ML based application Layer Firewalls [36]	2024	Performance and scalability analysis of ML-based firewalls	SVM	Addresses latency, scalability in large-scale scenarios; ensures real-time protection.
Securing web applications against XSS and SQLi attacks using a novel deep learning approach [37]	2024	DL based approach for WAF against XSS, SQLi	CNN, LSTM	High accuracy, minimal false positives; applicable to WAFs and IDS.

used to accomplish the connection. The technology stack comprises the Python programming language, the Anaconda distribution, the Scikit-Learn package, and the VS Code editor. A NumPy array stores the dimensional data for 150 samples, and the architecture includes a data flow diagram that uses Fisher's discriminant analysis.

The detailed study of integration of ML Models into Web Applications and Backend Frameworks (studies focusing on how ML models are integrated and deployed in web applications using frameworks like Flask, Django, etc.) are summarized in Table 4.

Case study: Google cloud armor adaptive protection. Google Cloud Armor offers a powerful example of integrating ML into web application security at scale. Its Adaptive Protection module employs advanced anomaly detection models trained on large volumes of global HTTP(S) traffic patterns. By learning baseline behaviors and deviations per application or service, it can automatically detect Layer-7 DDoS attacks, credential stuffing, and bot traffic anomalies [51]. When a threat is detected, Cloud Armor automatically generates custom security policies and WAF rules, providing recommendations with explainability metrics (e.g., confidence scores, anomaly distribution graphs) to security teams. In real deployments, this system has shown the ability to reduce mean time to mitigation (MTTM) from hours to under five minutes, and mitigate multi-vector attacks across Google Cloud's global edge network without affecting legitimate user traffic. The adaptive approach combines rule-based signatures and behavior-based detection, significantly reducing false positives and enabling continuous security posture improvement.

Case study: Netflix's foundation recommendation model. Netflix's transition to a unified foundation model for personalization illustrates the impact of centralized ML architectures on scalability and maintainability. Traditionally, separate models served features such as "Top Picks" or "Because You Watched", leading to duplicated infrastructure and inconsistent latency. The new foundation model integrates multiple objectives into a single shared neural network, trained on billions of viewing and interaction events [52]. This shared architecture has reduced system maintenance overhead, simplified feature engineering pipelines, and decreased median inference latency by approximately 20% during A/B testing. Additionally, it allowed Netflix to deploy new personalization features faster, as modifications to the shared encoder can simultaneously improve several user-facing modules. The approach also supports fine-grained audience segmentation and localized personalization without duplicating serving resources—illustrating the trade-off benefits of centralization over micro-model deployments in large-scale online systems.

Enhancing privacy with federated learning. Federated learning (FL) offers a privacy-preserving alternative to traditional centralized ML by allowing models to be trained across distributed user devices or edge servers without aggregating raw data centrally. In a recent study, Neumann et al. [53] applied FL to movie recommendation systems, achieving comparable accuracy to centralized training while maintaining user data locally. Their system used gradient compression and secure aggregation, reducing communication overhead by 38% and preserving differential privacy guarantees. Chronis et al. [54] further demonstrated that FL-enabled recommender systems can support personalization in sensitive domains (e.g., healthcare, financial services) without exposing individual behavior logs. In addition, FL architectures enable continuous model updating on user devices, ensuring models remain up-to-date without requiring large-scale data migrations. For security-centric use cases such as WAF anomaly detection, FL allows distributed edge servers to collaboratively train threat models without sharing raw traffic logs, reducing data leakage risks and improving real-time adaptability to region-specific attack patterns. This makes FL a promising approach for privacy-preserving personalization and adaptive security in modern online applications.

3. Common ML model deployment challenges and solutions

Machine learning (ML) model deployment is not simply about moving a trained model into production; it involves multiple technical layers, each with potential pitfalls that can compromise reliability and scalability. Among these, versioning, reproducibility, and operational scalability are the most critical and frequently overlooked challenges.

3.1. Deployment mistakes

A major challenge is the lack of proper versioning for both models and the data they rely on. According to Sculley et al. model versioning often gets neglected in favor of rapid iteration, leading to situations where it becomes impossible to reproduce previous model results or trace back failures [55]. Dependency conflicts further exacerbate reproducibility issues. A model trained in one environment (e.g., a specific combination of Python, TensorFlow, CUDA) might fail when deployed elsewhere due to mismatched package versions or system libraries.

Moreover, scalability issues arise when the training environment does not reflect production-scale data and request loads. Zhang et al. highlight that models evaluated on static batch data often underperform when faced with real-time, concurrent requests, leading to high latency and dropped requests [56]. Hidden dependencies on local resources (e.g., hardcoded file paths, local GPUs) and hard-to-detect stateful code can cause erratic failures when scaled horizontally.

3.2. Solutions with Docker and Kubernetes

To mitigate these issues, containerization has become a standard practice. Docker provides isolated and consistent environments by packaging the application, model files, and all dependencies into a single image [57]. This guarantees that the environment used during training is identical to that used during deployment, drastically improving reproducibility. Additionally, Docker's layered image architecture allows for efficient updates; for example, updating only the model layer without rebuilding the entire environment saves both time and computational resources.

Kubernetes offers advanced orchestration features that complement Docker's packaging capabilities. According to Burns et al. Kubernetes provides declarative deployment strategies, allowing operators to define desired system states and rely on automated controllers to reconcile discrepancies [58]. Features like horizontal pod autoscaling dynamically adjust the number of replicas based on metrics such as CPU usage or request latency, ensuring the system can handle peak traffic without manual intervention.

Kubernetes also supports rolling updates and rollbacks, which are essential for safely deploying new model versions. The system gradually replaces old pods with new ones, monitoring health checks to prevent downtime. Moreover, resource quotas and limits protect against noisy neighbor effects, preventing a single model container from monopolizing cluster resources and affecting overall system stability.

3.3. CI/CD and monitoring

Integrating Continuous Integration and Continuous Deployment (CI/CD) pipelines further enhances deployment robustness. Zaharia et al. emphasize that automated workflows, such as MLflow, can enforce systematic testing of models before deployment, including checks on performance metrics, drift detection, and dependency integrity [59]. CI/CD pipelines can automate container image building, vulnerability scanning, and push to Kubernetes clusters, reducing manual errors and accelerating iteration cycles.

Effective monitoring is critical after deployment. Runtime monitoring should capture both infrastructure-level metrics (CPU, memory, I/O) and model-level signals (accuracy drift, distribution shift, latency anomalies). Incorporating automated alerts and rollbacks based on these metrics ensures that faulty model behaviors can be caught early, minimizing impact on users and business operations.

3.4. Best practices

Some established best practices include:

- Maintaining a centralized model registry to log model artifacts, metadata, performance metrics, and lineage, thereby simplifying audits and reproducibility [55].
- Using immutable Docker images to prevent configuration drift across environments.
- Implementing canary or shadow deployments to test new models on a small subset of live traffic before full roll out.
- Applying Kubernetes resource requests and limits to guarantee performance isolation among services.
- Automating periodic retraining and redeployment to adapt to new data distributions without manual intervention.

Real-world production trade-offs. Though tools like Docker, Kubernetes, and CI/CD pipelines help address scalability, latency, and maintenance, deploying ML in production introduces complex trade-offs. For example:

Scalability vs. Resource Utilization: Horizontal scaling (e.g., Kubernetes HPA) improves capacity but increases orchestration overhead and potential cost overruns without careful tuning [60].

Low Latency vs. Precision and Cost: Using optimized runtimes (TensorRT, ONNX Runtime) reduces inference time (10–30 ms) but may require more GPU resources and incur higher runtime costs [61].

Frequent Retraining vs. Operational Complexity: Automating retraining mitigates model drift but adds pipeline complexity, requiring robust versioning and testing to avoid degradation in production [62].

In a 2022 IEEE-authored survey, Talagala et al. [63] emphasized that achieving high performance in live systems demands balancing throughput, latency, model accuracy, and operational agility. Future work should thus focus on **adaptive orchestration methods**–such as load-aware scaling and cost-aware retraining–to dynamically optimize these trade-offs based on real-time traffic and performance metrics.

4. Node.js deployment challenges and best practices

Node.js has emerged as a popular choice for building scalable and performant web backend due to its event-driven, non-blocking I/O model. However, deploying Node.js applications in production environments introduces unique challenges that differ from traditional back-end frameworks.

4.1. Single-threaded limitations

By design, Node.js operates on a single-threaded event loop. Although this model efficiently handles I/O-bound tasks, it struggles with CPU-bound workloads and can become a bottleneck when processing large computations or high concurrency levels. To utilize multi-core systems effectively, clustering approaches using tools like `pm2` or Node.js's built-in `cluster` module are employed. These tools spawn multiple worker processes, each running an instance of the application, enabling horizontal scaling on a single machine [64].

4.2. Dependency management and environment consistency

Node.js heavily relies on the npm ecosystem, which can lead to complex dependency trees and potential conflicts. Mismatched package versions or transitive dependency vulnerabilities can cause run-time errors in production. To mitigate these risks, strict use of lock files (`package-lock.json` or `yarn.lock`) ensures consistent dependency versions across environments. Additionally, containerizing applications with Docker standardizes runtime environments, minimizing configuration drift between development and production [57].

4.3. Scalability and load balancing

Scaling Node.js applications across multiple machines or containers introduces additional considerations, such as session state sharing and load balancing. Stateless design patterns and centralized session stores (e.g., Redis) enable seamless horizontal scaling. Reverse proxies like NGINX or HAProxy are commonly used to distribute incoming requests across multiple Node.js instances, manage SSL termination, and improve overall security and performance [65].

4.4. Container orchestration and deployment automation

Modern deployment workflows often leverage container orchestration platforms such as Kubernetes. Kubernetes manages container life cycle, automates horizontal scaling, and supports rolling updates and rollbacks, significantly reducing operational overhead. Node.js applications can be deployed as containerized microservices, allowing independent scaling and updates without affecting other components of the system [58].

4.5. Observability and resilience

Monitoring and observability are essential for ensuring the resilience of Node.js applications in production. Tools such as Prometheus and Grafana collect and visualize metrics like event loop latency, memory consumption, and error rates. Distributed tracing tools like Jaeger and Zipkin provide insights into request flows across microservices, helping diagnose performance bottlenecks and failures. Properly configured alerts enable proactive remediation and minimize downtime.

4.6. Best practices

Best practices for Node.js deployments include:

- Designing stateless applications to simplify scaling and failure recovery.
- Using process managers (e.g., `pm2`) to handle automatic restarts and clustering.
- Implementing containerization and orchestration for predictable deployments.
- Incorporating robust monitoring, logging, and alerting systems.
- Regularly scanning dependencies for vulnerabilities and applying security patches.

Table 3
Key trade-offs between flask and django frameworks for ML integration.

Aspect	Flask	Django
Flexibility	Highly flexible; developers have full control over components and architecture.	Less flexible; enforces a standard structure and strict design patterns, reducing customization freedom.
Scalability	Better suited for small to medium projects; requires extra design considerations for large-scale scalability.	Designed for high scalability and complex, enterprise-level applications out-of-the-box.
Security	Manual implementation of security features needed (e.g., authentication, CSRF protection).	Comprehensive built-in security features; easier to achieve secure deployments with less manual effort.
Development speed	Excellent for rapid prototyping and small proof-of-concept projects due to minimal setup and boilerplate.	Slower initial development but provides strong foundation for large, long-term production systems.
Project suitability	Ideal for microservices, MVPs, or simple web apps needing quick iteration.	Ideal for large, feature-rich, monolithic applications requiring robustness and scalability.

Hybrid ML classifier-based web page classification. Mahmoud et al. [66] proposed an automatic web page classification system that leverages a hybrid machine learning approach combining Naive Bayes (NB), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). Their method involves initial feature extraction from page titles and body text, followed by sequential classification: NB for rapid initial categorization, SVM for robust margin-based refinement, and KNN for correcting misclassified samples. Using the Reuters-21578 dataset comprising 10,000 web pages, this hybrid system achieved approximately 99.98% accuracy and an F-measure close to 1, outperforming individual classifiers significantly. While their focus was on content categorization, similar hybrid classification approaches can be adapted for intelligent web application firewalls (WAFs) or backend moderation systems to dynamically categorize and filter web content with high precision.

5. Discussion and summary

In the sphere of web application integration, the Python frameworks Flask and Django stand out, particularly when enhanced by Python's extensive machine learning libraries like PyTorch and TensorFlow. This synergy is evident from research papers and practical applications showcasing their compatibility with Python's ML capabilities. Flask, known for its lightweight and user-friendly design, is ideal for smaller-scale projects, offering ease of development and secure cookie support. However, its scalability is somewhat limited for larger applications due to a lack of built-in management tools. Conversely, Django's "battery-included" approach provides a robust architecture suitable for larger, more complex projects, with enhanced security features, pre-defined templates, and multi-language support. Yet, its monolithic structure and extensive codebase can be daunting for beginners and may be excessive for simpler projects. The contrast between Flask's streamlined processing and Django's comprehensive framework illustrates Python's versatility in web development. Both frameworks, leveraging Python's strengths in machine learning, cater to different project needs, enabling developers to create sophisticated, AI-driven web applications tailored to the scale and complexity of the task at hand [67–69] (see Table 3).

Node.js ML ecosystem: Tensorflow.js and Brain.js. TensorFlow.js is a comprehensive JavaScript library that enables seamless integration of machine learning into web applications by leveraging Node.js's event-driven architecture [75–77]. It supports both training and inference directly within JavaScript environments, allowing models to be exported in JSON or binary formats for easy deployment in Node.js backends. This design ensures compatibility and reduces context switching

Table 4
ML model integration into web applications using backend frameworks.

Title of the paper	Year	Methodology	ML Algorithm/Backend Tech	Results/key findings
Artificial intelligence model implementation in web-based application for pineapple object counting [40]	2019	Crowdsourcing for dataset annotation, web app integration	Flask, WSGI	Ensures scalability and smooth integration of AI model for real-time object counting.
Multi disease prediction model by using machine learning and flask API [70]	2020	Web application for disease prediction	Flask, TensorFlow	High accuracy across multiple diseases; seamless integration into web service.
Identifying machine learning techniques for classification of target advertising [71]	2020	Twenty three machine learning based online advertising stratergies, user centric and content centric approaches.	Behavioral targeting, Keyword browsing, Click through rate, User profiling, Sponsored search, Click frauds, Contextual advertising, Web documents, blogs, Real time bidding etc.	Proposed method identifies an underexamined area, algorithm based detection of click frauds. Illustrate how machine learning techniques can be integrated to preserve the viability of online advertising.
Web application implementation with machine learning [39]	2021	ML integration in web apps	Flask, Django	NLP for sentiment analysis (SVM: high precision/recall); profanity detection ensures safer platform.
Loan Prediction Using Machine Learning and Its Deployment On Web [43]	2021	Web-based loan eligibility prediction	Flask	Real-time predictions at scale; instant feedback on loan eligibility.
Web Application for Machine Learning based Music Genre Classification [44]	2021	ML-based music genre classification web app	Django; SVM chosen over KNN, DNN	SVM achieves highest accuracy (73.2%), integrated into web application.
Web App for Business Needs [49]	2021	Integrating ML into business-oriented web app	Flask, Node.js	Utilizes Justdial data for improved business recommendations.
Flower classifier web app using Ml [50]	2022	ML model for flower classification in web app	Flask	Successful IRIS classification; seamless Flask integration.
Leveraging machine learning methods for multiple disease prediction using Python ML Libraries and Flask API [72]	2022	Research project integrating ML into web app	Flask, TensorFlow	Random Forest performs well; Flask API facilitates integration and scalability.
Evaluation of machine learning methods for impostor detection in web applications [73]	2023	Machine learning based systems for impostor detection/recognition using biometric traits	k-nearest neighbors algorithm, Gaussian Naive Bayes, multilayer perceptron, support vector machine, logistic regression, random forest, and gradient boosting, SVM	False Acceptance Rate (FAR) and False Rejection Rate (FRR) smaller than 0.05. Achieved FAR 0.05 on financial service data.
SmartHealth: An intelligent framework to secure IoMT service applications using machine learning [74]	2024	ML-based framework: SmartHealth	Multilayer Perception Algorithm (MLP), Artificial Neural Network (ANN), Decision Tree, Random Forest, and KNN algorithms	SmartHealth a ML based framework proposed in this study is capable of analyzing the overall condition of an Smart Healthcare Systems (SHS). It also detect any dangerous act that has been attempted in SHS.

between front-end and back-end development workflows. The use of JavaScript across the entire stack enhances consistency and simplifies maintenance, while GPU acceleration through WebGL or Node.js bindings supports real-time predictions and low-latency responses [78]. Empirical studies have shown that although TensorFlow.js offers strong usability, it can experience higher computational overhead and slightly lower numerical precision compared to Python-based TensorFlow implementations [79]. Furthermore, system-level analyses highlight fault patterns and debugging challenges inherent in JavaScript-based ML deployments [80].

Brain.js provides a simplified, lightweight alternative focused on basic neural network architectures such as feedforward and recurrent (including LSTM) networks. It is optimized for smaller-scale tasks, enabling real-time analytics and immediate predictions in applications that demand quick feedback, such as live user interactions or streaming data processing. Brain.js uses GPU acceleration via headless-gl, further enhancing its suitability for fast, event-driven Node.js environments [81]. However, compared to Python's rich ecosystem, these frameworks have fewer advanced APIs and pre-trained models, making them better suited for lightweight or microservice-based solutions rather than large-scale deep learning deployments. Despite these limitations, TensorFlow.js and Brain.js provide valuable tools for integrating ML into modern web applications where responsiveness and architectural simplicity are priorities.

Python and its frameworks have long been central to machine learning integration, but the potential of Node.js in this field remains largely underexplored. This paper highlights the importance of future research efforts directed at unlocking Node.js's capabilities for machine learning, addressing its limitations, and leveraging its strengths alongside Python. Such an approach could pave the way for developing more efficient, versatile, and scalable web applications by harnessing the best aspects of both ecosystems.

In parallel, the study of Web Application Firewalls (WAFs) underscores a notable shift towards Anomaly-Based Detection (ABD) techniques that utilize machine learning to combat increasingly sophisticated cyber threats. The research emphasizes the effectiveness of algorithms such as Support Vector Machines (SVM), Decision Trees, Random Forest, Neural Networks, Naive Bayes, and K-Nearest Neighbors (K-NN) in enhancing WAF functionality.

For feature extraction and classification, SVMs and Convolutional Neural Networks (CNNs) demonstrate superior accuracy, making them the preferred methods. Character n-gram approaches, combined with linear SVM algorithms, outperform word n-grams in web attack detection.

Convolutional Neural Networks (CNNs) are highly proficient at analyzing intricate data structures, thereby enhancing the accuracy of threat identification. Additionally, Bi-Directional Long Short-Term Memory (Bi-LSTM) networks have shown remarkable effectiveness in real-time threat detection, achieving accuracy rates as high as 97.38% in sequential data classification.

Ensemble methods, particularly those that integrate multiple decision trees, achieve high True Positive Rates (TPR) while minimizing False Positive Rates (FPR). These advancements significantly improve the reliability and efficiency of Web Application Firewall (WAF) systems in detecting and preventing threats such as code injection attacks.

By incorporating these approaches, future developments can establish a more secure and adaptive framework for web application security, effectively addressing the dynamic landscape of cybersecurity challenges.

6. Conclusion

In conclusion, this comparative analysis of Python and Node.js highlights their strengths and weaknesses in the integration of machine learning models into web applications, offering key insights for developers. Furthermore, examination of Web Application Firewalls (WAFs) and their reliance on machine learning algorithms underscores their critical role in improving security.

The findings suggest that the future of WAF development lies in refining anomaly-based detection algorithms, marking a shift from reactive to proactive security measures. This evolution addresses the growing complexity of digital threats and positions WAFs as essential tools for securing web applications. By adopting these advances, developers and security professionals can better navigate modern digital challenges, ensuring robust protection for web applications. Moreover, a transformative direction for future research involves embedding Automated Machine Learning (AutoML) pipelines into WAF architectures to enable real-time adaptability against sophisticated threats. For example, the GenXSS framework leverages GPT4o to generate 264 new XSS payloads, 83% of which were syntactically valid and 80% successfully bypassed the ModSecurity OWASP rule set. Using just 15 autogenerated rules, GenXSS blocked 86% previously undetected attacks, outperforming the blocking capability of Gemini Pro 63% [82]. Meanwhile, AdvSQLi demonstrated a success rate 100% against ML-based SQL injection detectors and up to 79% bypass rate against commercial WAF-as-a service platforms like F5, exposing alarming vulnerabilities in current defenses [83]. These results highlight the pressing need for WAF systems that can automatically test, generate, and validate new rules in the face of evolving attacks. Integrating AutoML can operate in a closed loop: collecting payloads, retraining detection models, and applying new rule sets, thus enabling WAFs to self-optimize and defend proactively. Future work should carefully evaluate trade-offs between detection accuracy, latency, and resource overhead of such AutoML-enabled WAFs, ensuring their feasibility for deployment in high-traffic environments.

CRediT authorship contribution statement

Kshiteesh Mani: Writing – original draft, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Ajitha K.B. Shenoy:** Writing – review & editing, Supervision, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] C. Tiwari, S. Pillai, A.J. Obaid, A.R. Saear, A.K. Sabri, Integration of artificial intelligence/machine learning in developing and defending web applications, AIP Conf. Proc. 2736 (1) (2023) 060038, http://dx.doi.org/10.1063/5.0171097.

[2] WebsitePulse, How artificial intelligence and machine learning are revolutionizing website development, 2023, https://www.websitepulse.com/blog. (Accessed 25 March 2024).

[3] Tranxndweb, The role of artificial intelligence and machine learning in enterprise web applications, 2023, https://medium.com/@tranxndweb. (Accessed 25 March 2024).

[4] Y. Pan, F. Sun, Z. Teng, J. White, D.C. Schmidt, J. Staples, L. Krause, Detecting web attacks with end-to-end deep learning, J. Internet Serv. Appl. 10 (1) (2019) 1–22, http://dx.doi.org/10.1186/s13174-019-0115-x.

[5] M.N. Khalid, H. Farooq, M. Iqbal, M.T. Alam, K. Rasheed, Predicting web vulnerabilities in web applications based on machine learning, in: Proceedings of the International Conference on Intelligent Technologies and Applications, Springer, 2019, pp. 473–484, http://dx.doi.org/10.1007/978-981-13-6052-7-41.

[6] I.H. Sarker, A. Kayes, S. Badsha, H. Alqahtani, P. Watters, A. Ng, Cybersecurity data science: an overview from machine learning perspective, J. Big Data 7 (41) (2020) http://dx.doi.org/10.1186/s40537-020-00318-5.

[7] N. Montes, G. Betarte, R. Martınez, A. Pardo, Web application attacks detection using deep learning, in: Iberoamerican Congress on Pattern Recognition, Springer, 2021, pp. 227–236, http://dx.doi.org/10.1007/978-3-030-93420-0-22.

[8] S.A.R. Yee Leong Tan, Brain-inspired spiking neural networks for wi-fi based human activity recognition, Jordanian J. Comput. Inf. Technol. (JJCIT) 07 (04) (2021) 363–372, http://dx.doi.org/10.5455/jjcit.71-1629096728.

[9] E.A. Ammar Odeh, Phiboost- a novel phishing detection model using adaptive boosting approach, Jordanian J. Comput. Inf. Technol. (JJCIT) 07 (01) (2021) 64–73, http://dx.doi.org/10.5455/jjcit.71-1600061738.

[10] N.B. Ghannam, N. Salhab, M.A. Rahman, SQL injection, cross-site scripting and buffer overflow attacks detection using machine learning, in: 2022 International Conference on Data Analytics for Business and Industry, ICDABI, 2022, pp. 292–296, http://dx.doi.org/10.1109/ICDABI56818.2022.10041495.

[11] Y.E. Seyyar, A.G. Yavuz, H.M. Ünver, An attack detection framework based on BERT and deep learning, IEEE Access 10 (2022) 68633–68644, http://dx.doi.org/10.1109/ACCESS.2022.3185748.

[12] J. Kaur, U. Garg, G. Bathla, Detection of cross-site scripting (XSS) attacks using machine learning techniques: a review, Artif. Intell. Rev. 56 (2023) 12725–12769, http://dx.doi.org/10.1007/s10462-023-10433-3.

[13] H.-A. Goh, C.-K. Ho, F.S. Abas, Front-end deep learning web apps development and deployment: a review, Appl. Intell. 53 (2023) 15923–15945, http://dx.doi.org/10.1007/s10489-022-04278-6.

[14] K. Dhanya, S. Vajipayajula, K. Srinivasan, A. Tibrewal, T.S. Kumar, T.G. Kumar, Detection of network attacks using machine learning and deep learning models, Procedia Comput. Sci. 218 (2023) 57–66, http://dx.doi.org/10.1016/j.procs.2022.12.401, International Conference on Machine Learning and Data Engineering.

[15] S. Bamohabbat Chafjiri, P. Legg, J. Hong, M.-A. Tsompanas, Vulnerability detection through machine learning-based fuzzing: A systematic review, Comput. Secur. 143 (2024) 103903, http://dx.doi.org/10.1016/j.cose.2024.103903.

[16] H. Ahmetoglu, R. Das, A comprehensive review on detection of cyber-attacks: Data sets, methods, challenges, and future research directions, Internet Things 20 (2022) 100615, http://dx.doi.org/10.1016/j.iot.2022.100615.

[17] I. Muhusina, A. Saed, K.R.C. Kim, A. Luqman, H. Saad, A comprehensive evaluation of machine learning algorithms for web application attack detection with knowledge graph integration, Mob. Netw. Appl. Online published (2024) 1–30, http://dx.doi.org/10.1007/s11036-024-02367-z.

[18] H. Kadar, A. Zouhair, A comparative analysis of deep learning approaches for enhancing security in web applications, in: M. Ben Ahmed, A.A. Boudhir, R. El Meouche, I.R.p. Kara¸s (Eds.), Innovations in Smart Cities Applications Volume 7, Springer Nature Switzerland, Cham, 2024, pp. 561–570.

[19] J. Ha, H. Roh, QUIC website fingerprinting based on automated machine learning, ICT Express 10 (3) (2024) 594–599.

[20] J. Song, S. Choi, J. Kim, K. Park, C. Park, J. Kim, I. Kim, A study of the relationship of malware detection mechanisms using artificial intelligence, ICT Express 10 (3) (2024) 632–649.

[21] A. Tekerek, O.F. Bay, Design and implementation of an artificial intelligence-based web application firewall model, Neural Netw. World (2019).

[22] H.K. J, G.P. J, Securing web application using web application firewall (WAF) and machine learning, in: 2023 First International Conference on Advances in Electrical, Electronics and Computational Intelligence, ICAEECI, 2023, pp. 1–8, http://dx.doi.org/10.1109/ICAEECI58247.2023.10370872.

[23] J. Li, Q. Li, S. Zhou, Y. Yao, J. Ou, A review on signature-based detection for network threats, in: 2017 IEEE 9th International Conference on Communication Software and Networks, 2017, pp. 1117–1121, http://dx.doi.org/10.1109/ICCSN.2017.8230284.

[24] S. Cook, A web developer's guide to cross-site scripting, 2003, https://www.sans.org/white-papers/988/. (Accessed 25 March 2024).

[25] S. Albawi, T.A. Mohammed, S. Al-Zawi, Understanding of a convolutional neural network, in: 2017 International Conference on Engineering and Technology, ICET, 2017, pp. 1–6, http://dx.doi.org/10.1109/ICEngTechnol.2017.8308186.

[26] D.Y. Demirel, M.T. Sandikkaya, Web based anomaly detection using zero-shot learning with CNN, IEEE Access 11 (2023) 91511–91525, http://dx.doi.org/10.1109/ACCESS.2023.3303845.

[27] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.

[28] B. ISiker, I. SoGukpinar, Machine learning based web application firewall, in: 2021 2nd International Informatics and Software Engineering Conference, IISEC, 2021, pp. 1–6, http://dx.doi.org/10.1109/IISEC54230.2021.9672335.

[29] N.M. Thang, Improving efficiency of web application firewall to detect code injection attacks with random forest method, Program. Comput. Softw. 46 (5) (2020) 351–361, http://dx.doi.org/10.1134/S0361768820050072.

[30] A. Valenza, L. Demetrio, G. Costa, G. Lagorio, WAF-a-mole: An adversarial tool for assessing ML-based WAFs, SoftwareX 11 (2020) 100367.

[31] T.-C.-H. Nguyen, M.-K. Le-Nguyen, D.-T. Le, V.-H. Nguyen, L.-P. Ton, K. Nguyen-An, Improving web application firewalls with automatic language detection, SN Comput. Sci. 3 (446) (2022) http://dx.doi.org/10.1007/s42979-022-01327-2.

[32] Q.A. Al-Haija, A. Ishtaiwi, Machine learning based model to identify firewall decisions to improve cyber-defense, Int. J. Adv. Sci. Eng. Inf. Technol. 11 (2021) 1688.

[33] M.H.D.B.K. Aref Shaheed, D. Megias, Web application firewall using machine learning and features engineering, Secur. Commun. Netw. 2022 (2022) 1–14.

[34] K. Surendhar, B.K. Pandey, G. Geetha, H. Gohel, Detection of payload injection in firewall using machine learning, in: 2023 IEEE 12th International Conference on Communication Systems and Network Technologies, CSNT, 2023, pp. 186–190, http://dx.doi.org/10.1109/CSNT57126.2023.10134743.

[35] C.-V. Trinh, T.-T. Le, M.-K. Le-Nguyen, D.-T. Le, V.-H. Nguyen, K. Nguyen-An, An efficient machine learning-based web application firewall with deep automated pattern categorization, in: T.K. Dang, J. Küng, T.M. Chung (Eds.), Future Data and Security Engineering. Big Data, Security and Privacy, Smart City and Industry 4.0 Applications, Springer Nature Singapore, Singapore, 2023, pp. 212–225.

[36] V. Nayar, T. Malik, A.B. Khan, S. Srivastava, Optimizing real-time performance in ML-based application layer firewalls, in: S. Tanwar, P.K. Singh, M. Ganzha, G. Epiphaniou (Eds.), Proceedings of Fifth International Conference on Computing, Communications, and Cyber-Security, Springer Nature Singapore, Singapore, 2024, pp. 947–959.

[37] J. Tadhani, V. Vekariya, V. Sorathiya, A. Samah, E.S. Walid, Securing web applications against XSS and SQLi attacks using a novel deep learning approach, Sci. Rep. 14 (14) (2024) 1803, http://dx.doi.org/10.1038/s41598-023-48845-4.

[38] A. Stocco, How artificial intelligence can improve web development and testing, in: Companion of the 3rd International Conference on Art, Science, and Engineering of Programming (Programming 19), ACM, Genova, Italy, 2019, pp. 1–4, http://dx.doi.org/10.1145/3328433.3328447.

[39] A. Verma, C. Kapoor, A. Sharma, B. Mishra, Web application implementation with machine learning, in: 2021 2nd International Conference on Intelligent Engineering and Management, ICIEM, 2021, pp. 423–428.

[40] R. Rahutomo, A.S. Perbangsa, Y. Lie, T.W. Cenggoro, B. Pardamean, Artificial intelligence model implementation in web-based application for pineapple object counting, in: 2019 International Conference on Information Management and Technology, ICIMTech, 1, 2019, pp. 525–530,

[41] J. Gardner, The web server gateway interface (WSGI), 2009, pp. 369–388, A Press, Berkeley, CA.

[42] T.W. Cenggoro, F. Tanzil, A.H. Aslamiah, E.K. Karuppiah, B. Pardamean, Crowdsourcing annotation system of object counting dataset for deep learning algorithm, IOP Conf. Ser.: Earth Environ. Sci. 195 (2018).

[43] C.N. Sujatha, A. Gudipalli, B. Pushyami, N. Karthik, B. Sanjana, Loan prediction using machine learning and its deployement on web application, in: 2021 Innovations in Power and Ad- vanced Computing Technologies, I-PACT, 2021, pp. 1–7,

[44] J. Chauhan, J. Shah, E. Mundhe, I. Jain, Web application for machine learning based music genre classification, in: 2021 International Conference on Advances in Computing, Communication, and Control (ICAC3), 2021, pp. 1–6, http://dx.doi.org/10.1109/ICAC353642.2021.9697317.

[45] Y. Zhang, Support vector machine classification algorithm and its application, in: C. Liu, L. Wang, A. Yang (Eds.), Information Computing and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 179–186.

[46] X. Li, X. Peng, C. Ma, T. Liu, Z. Li, Research on software development project implementation effect evaluation based on support vector machine, in: 2021 3rd International Conference on Industrial Artificial Intelligence, IAI, 2021, pp. 1–5.

[47] K. Taunk, S. De, S. Verma, A. Swetapadma, A brief review of nearest neighbor algorithm for learning and classification, in: 2019 International Conference on Intelligent Computing and Control Systems, ICCS, 2019, pp. 1255–1260, http://dx.doi.org/10.1109/ICCS45141.2019.9065747.

[48] H. Yi, S. Shiyu, D. Xiusheng, C. Zhigang, A study on deep neural networks framework, in: 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference, IMCEC, 2016, pp. 1519–1522, http://dx.doi.org/10.1109/IMCEC.2016.7867471.

[49] D. Neminathan, S.Y. Dinesh, L. Balasubramanian, Web app for business needs, in: 2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation, ICAECA, 2021, pp. 1–5, http://dx.doi.org/10.1109/ICAECA52838.2021.9675608.

[50] A. Singh, R. Akash, G.R. V, Flower classifier web app using ml & flask web framework, in: 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering, ICACITE, 2022, pp. 974–977, http://dx.doi.org/10.1109/ICACITE53722.2022.9823577.

[51] Google Cloud, Google cloud armor adaptive protection: ML-powered anomaly detection and rule generation, 2025, Google Cloud documentation, accessed June 2025.

[52] N.T. Blog, Foundation model for personalized recommendation, 2025, Netflix Tech Blog, March 2025.

[53] D. Neumann, A. Lutz, K. Müller, W. Samek, A privacy preserving system for movie recommendations using federated learning, 2023, arXiv preprint arXiv:2303.04689.

[54] C. Chronis, I. Varlamis, et al., A survey on the use of federated learning in privacy-preserving recommender systems, IEEE Open J. Comput. Soc. (2022).

[55] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.F. Crespo, D. Dennison, Hidden technical debt in machine learning systems, in: Advances in Neural Information Processing Systems, 2015.

[56] W. Zhang, M. Liu, H. Zhou, Scalability issues in large-scale machine learning deployments, J. Syst. Archit. 107 (2020) 101700.

[57] D. Merkel, Docker: Lightweight linux containers for consistent development and deployment, Linux J. 2014 (239) (2014) 2.

[58] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, J. Wilkes, Borg, omega, and kubernetes, ACM Queue 14 (1) (2016) 70–93.

[59] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, A. Hong, A. Konwinski, B. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, Accelerating the machine learning lifecycle with mlflow, IEEE Data Eng. Bull. 41 (4) (2018) 39–45.

[60] K. Shivashankar, G. Al Hajj, A. Martini, Scalability and maintainability challenges and solutions in machine learning: A systematic literature review, 2025, arXiv preprint arXiv:2504.11079.

[61] A. Khare, D. Garg, et al., Superserve: Fine-grained inference serving for unpredictable workloads, in: ASPLOS 2023, 2023.

[62] F. Kossmann, Z. Wu, et al., Cascadeserve: Unlocking model cascades for inference serving, 2024, arXiv preprint arXiv:2406.14424.

[63] D. Kreuzberger, N. Kühl, S. Hirschl, Machine learning operations (mlops): Overview, definition, and architecture, IEEE Access (2023).

[64] S. Tilkov, S. Vinoski, Node.js: Using JavaScript to build high-performance network programs, IEEE Internet Comput. 14 (6) (2010) 80–83.

[65] A. Mardan, Practical Node.js: Building Real-World Scalable Web Apps, A Press, 2018.

[66] A. Mahmoud, M. Shehab, M.F. Tolba, A design of an automatic web page classification system, Br. J. Appl. Sci. Technol. 18 (6) (2016) 1–17.

[67] B. Choudhary, Flask vs django, 2024, https://www.finoit.com/articles. (Accessed 25 March 2024).

[68] Harkiran, Why is python the best-suited programming language for machine learning? 2019, https://www.geeksforgeeks.org. (Accessed 25 March 2024).

[69] New Horizons, Why is python used for machine learning? 2023, https://www.newhorizons.com/resources/blog. (Accessed 25 March 2024).

[70] A. Yaganteeswarudu, Multi disease prediction model by using machine learning and flask API, in: 2020 5th International Conference on Communication and Electronics Systems, ICCES, 2020, pp. 1242–1246, http://dx.doi.org/10.1109/ICCES48766.2020.9137896.

[71] J.-A. Choi, K. Lim, Identifying machine learning techniques for classification of target advertising, ICT Express 6 (3) (2020) 175–180.

[72] S. Narayanan, N.M. Balamurugan, M. K, P.B. Palas, Leveraging machine learning methods for multiple disease prediction using python ML libraries and flask API, in: 2022 International Conference on Applied Artificial Intelligence and Computing, ICAAIC, 2022, pp. 694–701, http://dx.doi.org/10.1109/ICAAIC53929.2022.9792807.

[73] M. Grzenda, S. aw Kaźmierczak, M. Luckner, G. Borowik, J. Mańdziuk, Evaluation of machine learning methods for impostor detection in web applications, Expert Syst. Appl. 231 (2023) 120736.

[74] S. Rani, S. Kumar, A. Kataria, H. Min, Smarthealth: An intelligent framework to secure IoMT service applications using machine learning, ICT Express 10 (2) (2024) 425–430.
[75] Nodejs, Using node.js for machine learning and AI applications, 2023, https://vegibit.com. (Accessed 25 March 2024).
[76] IBM, An introduction to AI in node.js, 2020, https://developer.ibm.com/tutorials. (Accessed 25 March 2024).
[77] Tudip, Unlocking the power of brain.js: Revolutionizing machine learning in JavaScript, 2023, http://tinyurl.com/Unlocking-the-power-of-brainjs. (Accessed 25 March 2024).
[78] D. Smilkov, N. Thorat, Y. Assogba, N. Nicholson, N. Kreeger, P. Yu, S. Zhang, K. Kavukcuoglu, I. Sutskever, D. Mane, Tensorflow.js: Machine learning for the web and beyond, in: SysML (now MLSys) Workshop, 2019.
[79] Y. Ma, D. Xiang, S. Zheng, H. Li, X. Liu, Moving deep learning into web browser: How far can we go? 2019, arXiv preprint arXiv:1901.09388.
[80] L. Quan, Q. Guo, X. Xie, L. Zhang, Y. Li, Y. Liu, Z. Liu, Towards understanding the faults of JavaScript-based deep learning systems, 2022, arXiv preprint arXiv:2209.04791.
[81] Brain.js Contributors, Brain.js — GPU accelerated neural networks in JavaScript, 2023, GitHub repository, https://github.com/BrainJS/brain.js. (Accessed June 2025).
[82] V. Babaey, A. Ravindran, Genxss: an AI-driven framework for automated detection of XSS attacks in WAFs, 2025, arXiv preprint arXiv:2504.08176.
[83] Z. Qu, X. Ling, T. Wang, X. Chen, S. Ji, C. Wu, Advsqli: Generating adversarial SQL injections against real-world WAF-as-a-service, 2024, arXiv preprint arXiv:2401.02615.