frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 06 October 2023
DOI 10.3389/fcomp.2023.1214722

OPEN ACCESS

EDITED BY
Niusha Shafiabady,
Charles Darwin University, Australia

REVIEWED BY
Herman Wandabwa,
Auckland University of Technology,
New Zealand
Mirna Muñoz,
Centro de Investigación en
Matemáticas, Mexico

*CORRESPONDENCE
Gopalakrishnan Sriraman
✉ gopalakrishnan.sriraman2019@vitbhopal.ac.in

RECEIVED 30 April 2023
ACCEPTED 13 September 2023
PUBLISHED 06 October 2023

CITATION
Sriraman G and R. S (2023) A machine learning approach to predict DevOps readiness and adaptation in a heterogeneous IT environment. *Front. Comput. Sci.* 5:1214722. doi: 10.3389/fcomp.2023.1214722

A machine learning approach to predict DevOps readiness and adaptation in a heterogeneous IT environment

Gopalakrishnan Sriraman* and Shriram R.

Department of Computing Science and Engineering, VIT Bhopal University, Sehore, MP, India

Software and information systems have become a core competency for every business in this connected world. Any enhancement in software delivery and operations will tremendously impact businesses and society. Sustainable software development is one of the key focus areas for software organizations. The application of intelligent automation leveraging artificial intelligence and cloud computing to deliver continuous value from software is in its nascent stage across the industry and is evolving rapidly. The advent of agile methodologies with DevOps has increased software quality and accelerated its delivery. Numerous software organizations have adopted DevOps to develop and operate their software systems and improve efficiency. Software organizations try to implement DevOps activities by taking advantage of various expert services. The adoption of DevOps by software organizations is beset with multiple challenges. These issues can be overcome by understanding and structurally addressing the pain points. This paper presents the preliminary analysis of the interviews with the relevant stakeholders. Ground truths were established and applied to evaluate various machine learning algorithms to compare their accuracy and test our hypothesis. This study aims to help researchers and practitioners understand the adoption of DevOps and the contexts in which the DevOps practices are viable. The experimental results will show that machine learning can predict an organization's readiness to adopt DevOps.

KEYWORDS

DevOps, machine learning, survey, adaption, accelerated software delivery, continuous delivery pipeline, technical agility

1. Introduction

DevOps helps businesses accelerate software delivery and experience to deliver optimal value. More broadly, DevOps is a philosophy that promotes better communication and collaboration between these teams. In this context, other observed trends include software increasingly being delivered over the Internet, either server-side (for example, Software-as-a-Service) or as a channel with direct delivery to the customer. Mobile platforms and their technology becoming increasingly pervasive and technologies upon which this software runs (Varia and Mathew, 2014).

Agile and DevOps process models require the delivery and execution of software engineering activities to add real value to the business. A fundamental success factor for the Agile and DevOps process models is the continuous delivery of incremental value to the organization. This requires an industrial assembly

line mindset for delivering software with a solid foundation of underlying toolsets, infrastructure, automation, and lean methodologies. Cloud computing and artificial intelligence are key technological advancements that are accelerating this development. These emerging trends support quick and short software delivery cycles within the Internet's fast-paced, dynamic world. DevOps has addressed the following drawback of the Agile software development methodology, i.e., iterative and rapid code development does not essentially result in iterative and rapid code deployment; annual "State of DevOps" reports have shown that there has been a 46% increase in the number of DevOps teams in organizations (DevOps Trends Survey, 2020).

The definition of DevOps is a combination of cultural and engineering practices, patterns, and tools that can increase an organization's capacity to deliver services and applications at high speed and with better quality. Continuous Integration, Continuous Delivery, Infrastructure as Code, and Monitoring and Logging are the various essential practices that have emerged over time in the adoption of DevOps. With DevOps, organizations can fully optimize the accomplishment of their goals of increasing revenue, minimizing costs, and retaining good employees. Increasing income will require the faster release of an organization's services or products on the market, which in turn necessitates much quicker and more reliable methods for building and deploying. To reduce costs by reducing rework, quality must be built into the process from the initial stages. DevOps teams tend to build and deploy early and often, away from the intensity of release night. Due to this, groups follow a systems-thinking approach that considers the time needed for fine-tuning scripts, discovering and resolving any problems, and cross-training team members. A DevOps team will employ a systems-thinking approach to facilitate the continuous improvement of critical skills and, thus, lead to increased expertise. DevOps will construct continuous integration and delivery capabilities without managing any servers or build nodes, and leverage Infrastructure as Code to iteratively and consistently provision and manage one's cloud resources (Hamunen, 2016).

Challenges in Adopting DevOps: DevOps adoption has certain aspects in its implementation that will slow it down either by inhibiting the DevOps enablers or by increasing the risk of not accomplishing the DevOps goals. These challenges include having staff with the right technical skills, resistance to change and uncertainty, changing the technology stack and tools, and uncertainty in responsibilities (Senapathi et al., 2018). The implementation of DevOps is easiest for small organizations. Since these organizations have few IT staff, the engineers will serve various roles to successfully complete all IT tasks. It is the developers' responsibility to help support and test the apps. This will tend to occur in a more organic way than in a deliberate manner. Even large companies may find themselves involved in a more organic construction of their strategies. The primary reason for this is that in large enterprises, the IT organization tends to be divided into various small-sized teams, each working on different apps and processes. Due to the small size of these individual teams, their members also tend to embrace DevOps practices by carrying out various IT roles, either unintentionally or deliberately. In general, large companies find it beneficial to plan and implement an enterprise-wide strategy. At a company level, promoting DevOps best practices will enhance collaboration between the various teams within an organization. Medium-sized organizations tend to have the most challenging time with the adequate performance of DevOps because they are big enough to have a sizeable IT organization but too small to divide it into multiple project teams. Another chief challenge is deciding whether an organization is DevOps-ready. Assessing an organization's characteristics and current state is essential to determining whether it can undergo a DevOps transformation.

The novelty of this research is to take a data-driven decision-making approach in DevOps readiness prediction; within this approach, validate various machine learning methods, and combine the A* search algorithm with Adaboost to improve the accuracy of the prediction. By combining the power of intelligent search, the A* algorithm with the traditional Adaboost framework improves optimal weak classifier selection and generalization, leading to more accurate and resource-efficient classification prediction.

By combining the A* search algorithm with AdaBoost to create A* AdaBoost, we are presenting a novel and innovative method. A* AdaBoost combines two distinct algorithms, AdaBoost and A* search, which are traditionally used in different domains. AdaBoost is a well-known ensemble learning method, while A* search is primarily utilized in pathfinding and optimization problems. Incorporating the A* search algorithm into AdaBoost, A* AdaBoost can efficiently navigate the feature space and focus on informative regions while building an ensemble of weak learners. This intelligent search strategy can lead to faster convergence and reduced computation time, setting it apart from traditional boosting algorithms.

This work is organized into the following four sections: Sections 2 and 3 will detail DevOps practices, which describe practices in software engineering and various machine learning techniques, respectively. Section 4 will review some of the available related literature. Section 5 will present the different methods employed in this work. Section 6 will show the experimental results and include the discussion, while Section 7 will conclude the work.

2. Devops practices

The software engineering process can be envisioned as a value stream with various activities from requirement intake to design, build, test, deploy, maintain, and monitor. Automating a part of the value stream provides local efficiencies. However, more is needed to make the system flow optimally. This requires an orchestration approach rather than a simple automation approach. Orchestration involves a workflow approach to automating multiple tasks and aligning an efficient flow that smooths and helps reduce the overall cycle time and accuracy of the value stream. Ensuring a high Percent Complete and Accurate is another critical enabler to benefit from automation; otherwise, it will be garbage in and garbage out. This requires a shift left/built-in quality approach across the entire life cycle. The problem is that the value stream is broken down into multiple layers/blocks, and each layer is analyzed for opportunities for automation and orchestration to get the best business value.

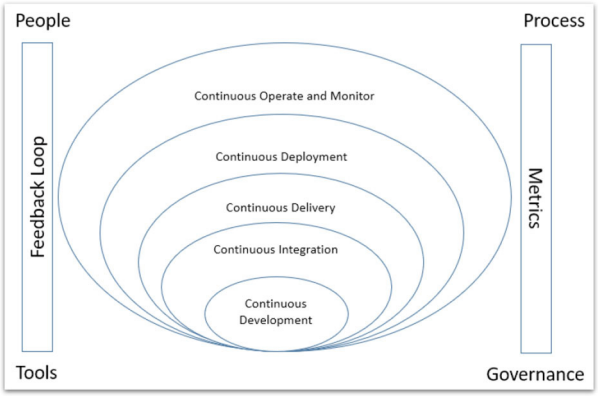

FIGURE 1
Software value stream as a set of layers.

As shown in Figure 1, the software value stream can be represented as a set of layers supported by feedback loops and metrics.

Continuous Development is the foundation for improving the software delivery process; however, it is also the least focused on in many organizations. This is the first building block where requirements are understood, coded/configured, and committed to the code base by the developer. Agile design practices, the definition of ready/done, standard Integrated Development Environment (IDE), coding standards, SCM, integration, and automation are the critical practices required at this level.

Continuous Integration extends continuous build, where the committed code of the developers is compiled as a package and assessed for code quality, automated unit testing/behavior-driven testing is run, and feedback is fed back to the developers. Static code analysis (Haas et al., 2020) leveraging various tools helps automate the identification of code quality issues—critical practices to adhere to commit frequency, fully automated CI, and behavior-driven tests.

Continuous Delivery extends further the Continuous Integration workflow and allows the deployment of any version of the code to any platform. It focuses on developing deliverable software that meets the functional and non-functional requirements of the Minimum Viable Product (MVP). Critical practices identified and needed for teams are MVP feature prioritization and Feature-driven development, ensuring that technical debt is tracked, measured, and regularly addressed so that software meets all critical quality needs, and implementing quality gates, elastic infrastructure, infrastructure management, integration, and automation essential for quality testing. Robotic Process Automation can be leveraged for end-user testing.

Continuous Deployment automates software delivery to production as part of the build pipeline and requires extreme maturity and agile deployment practices to benefit the organization. Critical practices identified and proposed include an agile deployment architecture, which gives the team the flexibility to switch on and off the capabilities released to Production. Since we have automated the deployment of code to Production, it is necessary to ensure that the functionality is not visible to end-users unless needed and that it can be switched on dynamically when required by the business. An automated rollback mechanism is required before considering continuous deployment. Disaster—technical or human error—can happen, and a system should be built to recover quickly from any such failure. Agile architecture—modular and based on microservices/Application Programming Interfaces (APIs) that reduce the size and impact of each deployment. Deployable architecture needs products and services/functions composed of loosely coupled, well-encapsulated components.

Continuous Operation and Monitoring provide the ability to continuously ensure system health, performance, and reliability at the infrastructure layer and business process level. Critical practices studied and proposed are visual monitoring of infrastructure, predictive monitoring and forecasting of events, regular vulnerability scans, internal and external audits, and time-bound PDCA for outcomes, observability, and architecture. System design and architecture should support monitoring with proper log tracking, audit trail and easy debugging, and automatic alerting mechanisms for all critical events.

The cloud provides an abstraction of the underlying infrastructure that is managed by the cloud vendors, who are committed to high SLAs. For example, cloud vendors provide highly resilient, multi-site federated services to organizations, which helps them not worry about the underlying infrastructure and focus on the application layer for operation and monitoring. This resilience allows businesses to scale up and down their infrastructure and dynamically manage costs. Also, the use of AI-powered monitoring tools helps ensure that there is no human dependency/error in the 24 × 7 monitoring of their business applications. Predictive and auto-healing algorithms can detect any critical hardware/software failures and automatically take actions to address them and alert relevant stakeholders.

The people aspect of Touchless automation is a critical element for successful adoption. This futuristic framework requires investment in building the culture and skills needed for the software team. To leverage emerging technologies, the team members should be familiar with them. Also, they should continuously seek and learn about new developments in their domain with respect to emerging processes, toolsets, and capabilities. Team members are typically specialists in one or more technical/functional domains and generalists in other relevant domains. Organizations should build a culture with practices such as open houses, hackathons, best practice sharing sessions, and well-defined recognition programs. A continuous improvement mindset should be cultivated by leveraging various lean tools like Kaizen, Value Stream Mapping, Problem Solving, 6S, and so on.

The Process aspect is another critical element. Any automation and standardization will only work when there is discipline. If we attempt to automate a process that is not standardized, the result will be suboptimal and non-value-adding. Hence, before we embark on the journey of touchless automation, it is essential that the organization first defines a standard process for software engineering practices and drives adherence to the same. This will also simplify the complexity and variants that we need to

accommodate while building the continuous delivery pipeline workflow for the organization.

3. Machine learning algorithms

Machine learning algorithms enable us to analyze existing systems and process attributes across various industry datasets, identifying intelligent patterns and classifications that can help in faster decision-making. In this study, machine learning techniques were used to validate their application for predicting DevOps readiness based on various software engineering attributes mentioned in the previous section.

What follows is an overview of the algorithms considered for this study.

Logistic Regression:

Logistic regression is used to estimate the likelihood of an event occurring based on a given dataset (Oates et al., 2022). It is a simple and efficient algorithm when the data is linear, but the results may need to be more accurate for nonlinear and smaller data sets.

Support vector machines (SVM):

SVM is a robust algorithm (Shafiabady et al., 2023; Wu Robert et al., 2023) with vast applicability in testing and data mining with large data sets. It aims to create the best line or decision boundary that can segregate the n-dimensional space into classes to quickly place the new data points in the correct category in the future. This best-decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help create the hyperplane. These extreme cases are called support vectors and, hence, Support Vector Machines.

K-nearest neighbor (KNN):

KNN is a non-parametric, supervised learning classifier that uses proximity to make classifications or predictions about the grouping of an individual data point. It assumes a similarity between the new case/data and the available cases and places the new case in a category that resembles the available ones.

Naive Bayes:

The Naive Bayes classifier (Jadhav and Channe, 2016) is one of the simpler forms of the Bayesian network, featuring independent attributes and a particular class variable value. This is also called conditional independence and has a numerical approach besides being accurate, fast, and straightforward. Naive Bayes is used for binar problems and multi-class classification problems; it assumes the mutual independence of the variables, contributing to their classification. The Naive Bayesian classifier is formed based on the theorem of total probabilities.

Decision Tree–Entropy:

A decision tree constructs classification or regression models in a tree structure. It will divide a data set into smaller subsets, incrementally developing an associated decision tree. The outcome will be a tree composed of decision nodes and leaf nodes. While a decision node will have two or more branches, a leaf node will denote a classification or decision. The top decision node in a tree will correspond to the best predictor, the root node. Decision trees can handle numerical and categorical data (Li et al., 2019).

Decision Tree–Gini Index:

The Gini index, also known as Gini impurity, evaluates the probability that a specific feature will be incorrectly classified in a random selection. If all elements are linked to a single class, they will be called pure. There is a variance in the Gini index between the values of 0 and 1. When the Gini index = 0, it will indicate the purity of the classification; that is, either all elements are members of a specific class or there is the presence of only a single class. When the Gini index = 1, it will indicate the random distribution of elements across multiple classes. In contrast, when the Gini index = 0.5, it will indicate the uniform distribution of the elements across certain classes (Raileanu and Stoffel, 2004).

Random Forest:

The Random Forest classifier will have a combination of tree classifiers in which the generation of each classifier is done using an unexpected vector that has been independently sampled from the input vector, and each tree will cast a unit vote for the most popular class for classifying an input vector (Breiman, 1999; Pal, 2005).

AdaBoost:

The AdaBoost algorithm will aid in integrating weak classifiers into a robust classifier. A weak classifier is defined as a simple classifier that continues to suffer from poor performance despite being capable of performing better than random extraction. It is possible to train multiple weak classifiers and combine their outcomes to yield a robust classifier. The AdaBoost algorithm will generate strong classifiers from weak classifiers (Wyner et al., 2017). AdaBoost will use the adaptive modification of the weights at each cycle to produce a group of weak classifiers from the members of the ensemble classifier. With a reduction in the correctly classified training sample weights of a current weak classifier, there will be an increase in the incorrectly classified training sample weights of the current weak classifier. Even though AdaBoost is an accurate algorithm for building ensemble classifiers, it may be unable to form ensemble classifiers that minimize the average generalization error.

Adaboost assigns equal weights to all data points. A decision stump is created for every feature, and the Gini index is calculated. The stump with the lowest Gini index is considered the first stump. In the next step, the classifier's "influence" (Importance) in classifying the data points is computed. The total error is the sum of all the sample weights of misclassified data points. The weights are increased for incorrect predictions and decreased for correct predictions. The new weights are normalized. This process is iterated until a low training error is attained (Wyner et al., 2017).

4. Related works

Some works in the literature on DevOps address the acquisition of adequate DevOps skills to meet the demand for DevOps practitioners in the industry (Suescún-Monsalve et al., 2021) and the adoption prospects of DevOps practices (Airaj, 2017; Hasselbring et al., 2019). Sen et al. (2021) attempted to demonstrate how the DevOps principles can effectively manage and deploy business problems in the classroom, more specifically, the execution of the DevOps methodology for managing the development and deployment of a small web application. Rafi et al. (2020) identified the critical factors that adversely impacted the data quality assessment procedure of DevOps. A systematic literature review (SLR) approach was employed to identify 13 challenging critical factors. Furthermore, these SLR findings were

validated by industry experts through a questionnaire. Finally, the Fuzzy TOPSIS approach was applied to prioritize the examined challenging factors in terms of their significance to the DevOps data quality assessment procedure.

Continuous delivery of software applications using automation (Humble and Farley, 2010) is a cornerstone development that we extend further in the proposed approach. Leveraging automation, tooling, and lean thinking in software is revolutionizing the software value stream and driving continuous value to the business. Continuous delivery and deployment are still in their nascent stages and face numerous challenges (Shahin et al., 2017) that software organizations need to overcome to reap the benefits. Various factors, such as compliance and regulatory requirements, tool limitations, and human trust, continue to challenge the level of automation in the build pipeline. Standardization challenges are another key barrier to adoption, as it is evident that automating poor and non-standard processes leads to poor outcomes. The impact of continuous delivery practices from a software developer's point of view and the prerequisites for a good continuous delivery pipeline were explored by Kärpänoja et al. (2016), and this is still evolving rapidly. Addressing this cultural and human aspect of the continuous delivery pipeline is key to the success and adoption of our proposed framework. The work of Olszewska and Waldén (2015) focused on how formal modeling can function within DevOps and thus promote various dimensions of quality and continuous delivery. The authors also noted that DevOps is still not a well-structured and defined concept, but utilizing existing developments and tools through a well-structured merge is important to support the end-to-end spectrum of software development and its related people, processes, and artifacts, which also include operations and quality management tasks. A survey (Bezemer et al., 2019) on how performance is addressed in the DevOps lifecycle has provided a detailed view of the practices, processes, and toolsets that can be leveraged in a continuous lifecycle; this also shows the challenges and the need to have simplified and lean processes and tools that can learn and adapt quickly for success.

Another critical success factor in implementing DevOps and automation in mainstream software organizations is architecture. The work done by Shahin (2015) provides an understanding and addresses new challenges for designing architectures to support DevOps in the context of Continuous deployment. Lianping Chen's six strategies (Chen, 2017) for overcoming the challenges of Continuous deployment adoption broadly address the threats and opportunities in CD adoption. "The Impact of Artificial Intelligence on Innovation" (Iain, 2018) details how the latest technologies can revolutionize innovation. Research can shift from labor-intensive research to one that takes advantage of the interplay between passively generated large data sets and enhanced predictive algorithms. The work by Bhandari et al. leverages various machine learning and deep learning algorithms (Bhandari et al., 2023) to make predictions with high accuracy. The validity and assurance of the use of machine learning algorithms in predicting the accuracy and classification of ground truths are explained by various research works (Ambagtsheer et al., 2020).

Despite the plethora of research on DevOps implementation, it is critical to address the numerous benefits and challenges of determining an organization's readiness in a heterogeneous IT environment. Most of our work is based on a survey questionnaire that collected responses from industry practitioners working in the DevOps environment. Unlike other observational methods, this approach offers the best opportunity to gather opinions on the related research problem from a diverse and well-distributed population. Inferences from survey responses can be made with either frequency analysis or a case study approach to determine the observations. The research was then used to establish the ground truths. This work investigates the use of a machine learning approach for the prediction of DevOps readiness and its adoption within a heterogeneous IT environment. To accomplish this work's goals, a survey has been conducted with frequency analysis to infer ground truths. Subsequently, the same data has been employed to train the machine learning algorithms to predict effectiveness work by Chang and Rong (2019) provided insight into how various ensemble algorithms can be used for data-driven prediction analysis, and that by Nevendra and Singh (2019) provided insight into the weaknesses of ensemble algorithms and the potential to combine various other techniques to improve accuracy and performance.

5. Methodology

A survey was designed as a combination of objective and subjective questions to allow the audience to express their opinions. The questionnaire design comprised five sections with 19 high-level questions and 35 sub-questions. Three of the 19 high-level questions were open-ended, while the others were in single-choice, multiple-choice, or Likert format. Furthermore, the survey was limited to software and management professionals in the field of software delivery. Sample questions from the questionnaire are as follows:

- What is the main challenge in your company's IT application landscape?
- Are your software development practices truly agile and able to meet business and end-user needs seamlessly and quickly?
- Do business and IT work well together as a team?
- Do you work in sprints and release working solutions at each sprint?
- Do you plan and prioritize work for each sprint, considering emerging business needs?
- Does your IT management believe in agility and support the team without much interference?
- Do your Software Engineering practices enable agility and not create too much overhead?
- What is the frequency of releases to customers?
- Do you version control all software deliverables to include code, documents, test cases, and all artifacts?
- Are developers performing well-documented unit testing, and are most defects being caught during this process?

We analyzed the data and performed a frequency analysis based on various identified variables, frequencies, and patterns to understand the following ground truths: the variables were based on application legacy, various DevOps practices, industry size and attributes, and perceived success and outcomes of DevOps transformation. We evaluated the statistical insights from the survey results and combined them with our qualitative study of DevOps practices.

The key insights that helped us build the mentioned ground truths are that larger organizations with a larger legacy and complex system landscape find it very challenging to adopt DevOps, as it increases the cost and risk of the transformation. A systematic approach and risk-based approach seems to be helping the organizations to adopt DevOps more smoothly. Also, organizations with an agile mindset and lean startup culture could implement DevOps faster with less resistance and start reaping its benefits.

Based on these findings, the following fundamental truths were established:

Truth 1 —The organization has many legacy applications; it is effort-intensive and complex to migrate to the DevOps model. Application rationalization is required before DevOps can be adapted to deliver business value. High-risk and effort-intensive.

Truth 2—The organization is already equipped with basic DevOps building blocks and has an extensive and diverse IT landscape; by introducing a structured framework, it can quickly be onboarded to the Touchless Automation Framework for a group of applications, while the rest follow the traditional approach, more suitable for a bi-model setup.

Truth 3 —The nascent organization and IT landscape, processes, and tools/frameworks are highly evolving, have a start-up mindset, and can quickly leverage DevOps/touchless automation practices to deliver business value.

These truths are classified as Not Recommended, Recommended, and Highly Recommended, respectively. Based on this, machine learning algorithms such as Logistic Regression, support vector machines (SVM), K-nearest neighbor (KNN), Naïve Bayes, Decision Tree—Entropy, Decision Tree—Gini Index, Random Forest, and AdaBoost were implemented and tested to validate the accuracy of whether they can be used to test for DevOps readiness.

The framework for applying machine learning to predict DevOps readiness and adaptation in IT environments is shown in Figure 2. In summary, the feedback from our survey was analyzed for ground truths. The obtained feedback data set was then applied to various classification machine learning algorithms, and the results were compared to test the hypothesis of whether the algorithms provided accurate results.

As part of this research, we also experimented with the algorithms to provide better accuracy. With the latest developments in software engineering and the complexity of the multiple factors involved in successful DevOps implementation, we need to build a prediction model. In this study, we validated the hypothesis that we can predict DevOps readiness by evaluating it with multiple available machine learning and statistical approaches.

Machine language algorithms were used for this research to handle large data sets and their potential to scale as we deploy them to a larger number of organizations in the future. In addition, one of our other objectives is to test whether automated, data-driven decision-making can be applied to various software engineering problems and drive sustainability.

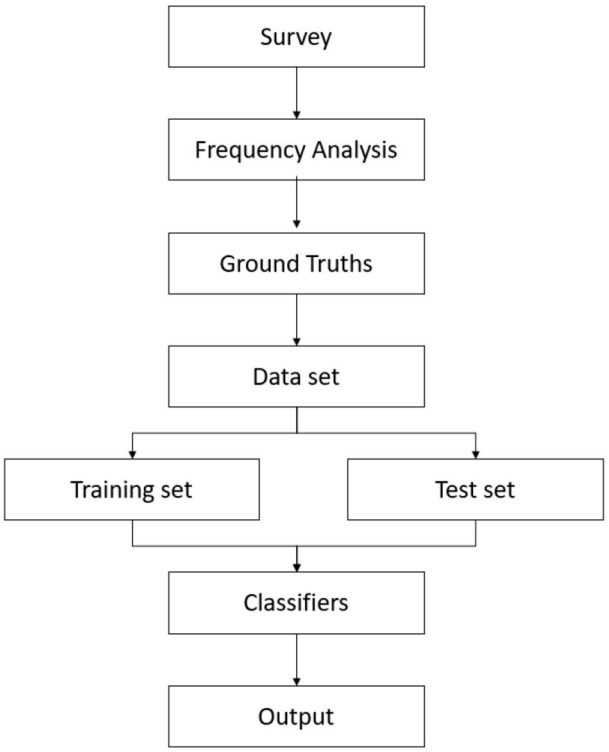

FIGURE 2
Framework for predicting DevOps readiness with machine learning.

5.1. Proposed A* adaboost algorithm

Adaboost is a powerful classification algorithm. However, it has limitations such as a lack of explicit search, sensitivity to outliers in the data sets, and insufficient handling of dynamic environments. An important parameter for Adaboost is the number of weak learners used, which is generally adjusted to the accuracy of the model. A higher number of trees is required for the model to be efficient. Prediction also depends on the depth of the tree. The learning rate is generally fixed at 0.5, but a smaller learning rate is feasible, as higher rates lead to overfitting. Thus, optimizing the values of several weak learners, the depth of the trees, and the learning rate parameters of Adaboost is essential to improving the performance of this ensemble algorithm.

In this work, it is proposed to use the A* algorithm to optimize the Adaboost parameters. A* is a best-first search path search algorithm that aims to find the path with the minimum cost. The A* search algorithm guarantees to find an optimal path by combining heuristics; it prunes unnecessary branches of the search space and optimizes the solution; it takes advantage of domain-specific knowledge data sets and searches efficiently. The A* AdaBoost steps are explained in Figure 3 below:

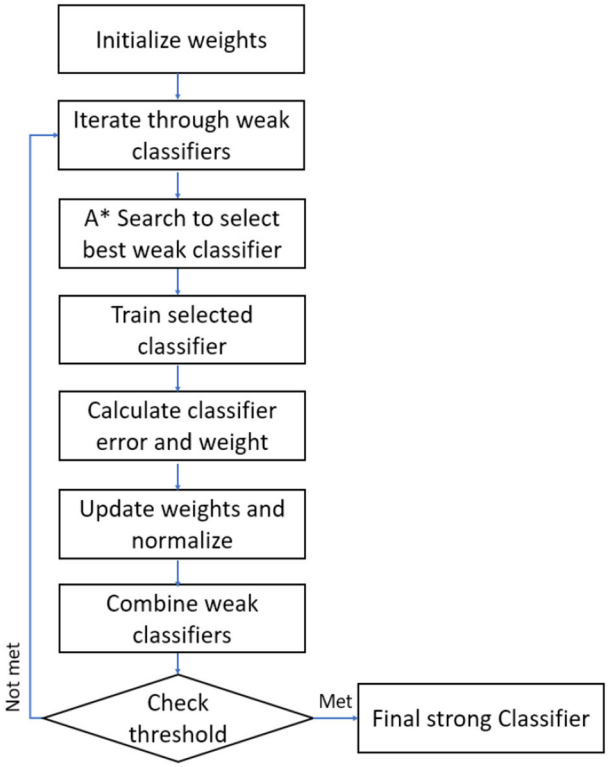

FIGURE 3
A* AdaBoost workflow.

```
def adaboost_a_star(D, T):
    n = len(D) # Number of training data sets
    weights = initialize_weights(n) # Initialize weights
    weak_classifiers = [] # List to store weak classifiers
    weak_classifier_weights = [] # List to store weak classifier weights
    for t in range(T):
    weak_classifier = train_weak_classifier(D, weights)
    weak_classifiers.append(weak_classifier)
    errors = np.array([weak_classifier_error(weak_classifier, dataset, weight) for dataset, weight in zip(D, weights)])
    weighted_error = np.dot(weights, errors)
    beta_t = weighted_error / (1 - weighted_error)
    alpha_t = np.log(1 / beta_t)
    weights *= np.power(beta_t, 1 - errors)
    weights /= np.sum(weights)
    weak_classifier_weights.append(alpha_t)
```

```
# Combine weak classifiers to form strong classifier
def strong_classifier(x):
    total_score = 0
    for classifier, weight in zip(weak_classifiers, weak_classifier_weights):
        total_score += weight * classifier(x)
    return np.sign(total_score)
return strong_classifier
```

The pseudocode and explanation of the proposed are as follows:

Initialize weights:

```
import numpy as np
def initialize_weights(n):
    weights = np.ones(n) / n
    return weights
```

By initializing the weights to equal values, the A* AdaBoost algorithm starts with the assumption that all data sets are equally important. As the algorithm progresses, the weights are updated based on the performance of weak classifiers, assigning higher weights to misclassified data sets and lower weights to correctly classified data. This way, subsequent weak classifiers focus more on the previously misclassified data, improving overall performance.

Iterating through weak classifiers to build robust ones:

In each iteration, the weak classifier is trained using the current data set weights, and the weights are updated based on the performance of the weak classifier. This iterative process enables the algorithm to focus on the misclassified data set, improving the performance of the ensemble classifier over time. Using the A* search algorithm, the modified AdaBoost algorithm efficiently explores the space of weak classifiers and identifies the best performance on the training data set. This helps construct a strong classifier that generalizes well to an unseen new data set and improves the overall classification performance.

6. Results and discussion

As every industry domain leverages software and systems to drive efficiency and quality, it becomes critical for industry professionals to understand and apply software engineering best practices, what works, and what attributes lead to failure. DevOps practices are rapidly evolving with advancements in cloud computing, data technologies, and virtualization. To capture the latest trends in DevOps and use them to predict a model, a survey was carried out to understand the industry's technical agility/automation and engineering practices. The survey format consisted of five sections with 19 high-level questions (of which three were open-ended while the rest were either single-choice, multiple-choice, or Likert format) and 35 sub-questions. This survey was delivered to a target audience of about 200+ software workers. A total of 105 responses were received, meaning that 52.5% of the target audience responded to the study. Since there was a healthy distribution of company sizes in the audience, the range of survey submissions across different company sizes was good. Participation in this survey was both anonymous and voluntary. Based on the frequency analysis, we can infer the following from the responses to the IT application landscape: the majority of organizations have fragmented applications with an overlap of

abilities, and while there is a fair awareness of agility among the participants, there is, however, a significant number of respondents who feel that agile practices have high overheads, which in turn would indicate that the participants have a limited understanding of Lean-Agile practices, and also that there continue to be gaps in basic hygiene practices such as Version Control and Unit Testing. The fragmented IT landscape leads to multiple challenges, such as a lack of integration possibilities, data inconsistency, and inefficient and broken workflows with high manual effort that result in increased maintenance and operations costs. This also exposes organizations to higher security and compliance risks. A summary of the survey profile and data set is presented in Appendix I.

We also observed that awareness of Lean-Agile practices could be more balanced even in established organizations, leading to inefficient processes, poor collaboration across teams, and high IT costs and time for businesses. The lack of efficient Lean-Agile DevOps practices is seen by organizations as the primary root cause of not getting the best value from IT investments.

By applying frequency analysis to the acquired data, the following ground truths were inferred and used to apply the algorithms to the data.

Truth 1 —Not Recommended; The organization has many legacy applications; it is effort-intensive and complex to migrate to the DevOps model.

Truth 2—Recommended; The organization is already equipped with basic DevOps building blocks and has an extensive and diverse IT landscape; by introducing a structured framework, it can quickly be onboarded.

Truth 3 —Highly Recommended; The nascent organization and IT landscape, processes, and tools/frameworks are highly evolving, have a start-up mindset, and can quickly leverage DevOps/touchless automation practices to deliver business value.

The resulting data set ended up containing 340 Not Recommended, 410 Highly Recommended, and 120 Recommended instances. This data set was created based on the survey submission data, feature selection, and processing. The evaluations were carried out using Python software.

The answers to the questions were used as the features for the classifiers. The features were both numerical and categorical in nature. In this work, supervised learning algorithms were used for classification. The data set was split into a test set (20%) and a training set (80%). The classifiers were trained using the training data. The classifiers used in this investigation are Logistic Regression, Support Vector Machines (SVM), K-nearest neighbor (KNN), Naive Bayes, Decision Tree- Entropy, Decision Tree—Gini Index, Random Forest, AdaBoost, and the proposed A* Adaboost. The obtained output is Not Recommended (Class A), Highly Recommended (Class B), and Recommended (Class C).

Appendix II (Tables 1, 2) provides the results on the Accuracy, Precision, Recall (Sensitivity), Specificity, and F measure values across the algorithms used by the classification.

- Accuracy is a measure of the correctness of the model's prediction.
- Precision is the ratio of true positive predictions to the total number of predictions.
- Recall is a measure of the ability to capture all positive instances.
- F-measure is the balance between Precision and Recall and provides insight to evaluate the performance of the model considering the trade-offs.

It was observed that the accuracy of the proposed A* Adaboost is effective in classifying the correct class as compared to the other classifiers. A* Adaboost achieves better accuracy by 16.12% for Naive Bayes, 12.42% for Decision tree—Entropy, 10.32% for Decision Tree- Gini Index, 8.12% for Random Forest, and 5.24% for Adaboost. Combining the weak learners' performance of the proposed A* Adaboost with optimized parameters to formulate a robust classifier that has better performance than any other weak classifier helped achieve higher precision. A*Adaboost outperformed Naïve Bayes by 21.43%, Decision Tree—Entropy by 15.51%, Decision Tree—Gini Index by 12.27%, Random Forest by 11.01%, and Adaboost by 7.95%.

The proposed A* Adaboost achieved a better recall of 22.9% for Naïve Bayes, 15.94% for a Decision tree—Entropy, 11.9% for Decision Tree- Gini Index, 11.63% for Random Forest and 9.03% for Adaboost. It was observed that the optimization of the A* Adaboost parameters significantly improved the recall performance.

The proposed A* Adaboost achieved a better recall by 22.16% for Naïve Bayes, 15.84% for a Decision tree—Entropy, 12.34% for Decision Tree- Gini Index, 11.34% for Random Forest and 8.4% for Adaboost.

The superiority of A* Adaboost lies in its ability to use machine learning to refine the A* search algorithm. By incorporating Adaboost, A* Adaboost can learn from the characteristics of the problem domain and adaptively adjust its search strategy. The optimized parameters of A* Adaboost play a crucial role in its enhanced performance. Tuning the classifiers and finding the right balance between multiple training iterations and classifier weights improves the performance of our algorithm.

Based on the above results, we observed that the modified A* Adaboost machine learning algorithm yielded better prediction results and that it could be adapted to predicting readiness as we scaled the data sets and increased the volume of data to build an industry prediction model. The proposed A* Adaboost is more effective than Logistic Regression, SVM, KNN, Naïve Bayes, Decision Tree—Entropy and Gini indexes, Random Forest, and Adaboost. In A* Adaboost, the enhanced performance of the resulting combined classifier is due to optimized added weights. This unique combination of algorithms provides efficient feature space navigation, unbalanced data handling, dynamic adaptability, application flexibility, and the opportunity it presents to explore new directions in ensemble learning. This novel approach promises to advance state-of-the-art machine learning and may offer significant advantages over traditional boosting techniques in various real-world applications.

In this study, we conducted an in-depth analysis of the consistency of parameter adjustments and hyperparameters for the A* AdaBoost algorithm and its performance. The objective was to understand how variations in these settings influence

model behavior and predictive capabilities. We systematically manipulated specific hyperparameters, such as the number of weak learners, the learning rate, and the depth of the decision trees, while keeping other settings fixed. We performed multiple experimental runs, each with a different combination of parameter values, and recorded the corresponding model performance metrics.

Our results demonstrated a clear relationship between parameter adjustments and the model's performance. We observed that increasing the number of weak learners improved model accuracy and generalization ability up to a certain point. Beyond that point, increasing the number of weak learners yielded little performance gain, suggesting diminishing returns. Similarly, the learning rate was crucial in balancing the tradeoff between convergence speed and stability. A higher learning rate enabled faster convergence but risked overshooting the optimal solution, resulting in suboptimal performance.

Conversely, a lower learning rate ensured more stable updates but could prolong the convergence process. We also investigated the effect of decision tree depth on model performance. We found that shallow decision trees improved the model's interpretability but could not capture complex patterns in the data, leading to reduced predictive power. Deeper decision trees could capture intricate relationships in the data but there is a risk of overfitting, resulting in poor generalization to unseen data. Our sensitivity analysis further revealed that the optimal parameter configurations were specific to the data set and problem. However, we identified trends across different data sets, suggesting some degree of transferability in parameter settings.

Overall, our findings indicate that tuning the hyperparameters of the A* AdaBoost algorithm is crucial to achieving optimal performance. Our results highlighted the need to consider the interplay between different parameters and their effects on model behavior. Several consistent trends and patterns were observed that had a significant impact on the model's predictive capabilities. The number of weak learners, learning rate, search heuristic, weight, depth of decision trees, and space pruning are several parameters that we leveraged in our experiment. These insights provide avenues to enhance its performance in predicting DevOps readiness and potentially in other machine learning applications.

7. Conclusion and future work

This work has employed machine learning approaches to classify an organization's DevOps adoption into the following categories: Highly Recommended, Recommended, and Not Recommended. Various machine learning algorithms have been deployed and tested with the available data, validating the approach for predicting DevOps readiness and the different parameters enabling DevOps in organizations. The effectiveness of the proposed A* AdaBoost in predicting an organization'sx willingness to adopt DevOps has been noted. It is possible to explore the optimization of AdaBoost to improve the accuracy levels in future research studies. Metaheuristic algorithms to optimize classifiers can be further investigated. Applying various data science techniques to predict DevOps readiness and different DevOps practices leads to more sustainable and efficient computing. In our future work, we aim to work further on how emerging data science technologies can be leveraged to improve sustainable computing and DevOps practices in detail.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary material, further inquiries can be directed to the corresponding author.

Ethics statement

Ethical review and approval was not required for the study on human participants in accordance with the local legislation and institutional requirements. Written informed consent from the participants was not required to participate in this study in accordance with the national legislation and the institutional requirements.

Author contributions

Conceptualization, methodology, validation, formal analysis, project administration, investigation, resources, writing, original draft preparation, and writing—review and editing: GS. Supervision: SR. All authors contributed to the article and approved the submitted version.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fcomp.2023.1214722/full#supplementary-material

References

Airaj, M. (2017). Enable cloud DevOps approach for industry and higher education. *Concurr. Comput.* 29, e3937. doi: 10.1002/cpe.3937

Ambagtsheer, R., Shafiabady, N., Dent, E., Seiboth, C., and Beilby, J. (2020). The application of artificial intelligence (AI) techniques to identify frailty within a residential aged care administrative data set. *Int. J. Med. Informat.* 136, 104094. doi: 10.1016/j.ijmedinf.2020.104094

Bezemer, C.-P., Walter, J., Willnecker, F., Eismann, S., Ferme, V., Grohmann, J., et al. (2019). "How is Performance Addressed in DevOps?," in *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering - ICPE'19*. doi: 10.48550/arXiv.1808.06915

Bhandari, B., Park, G., and Shafiabady, N. (2023). Implementation of transformer-based deep learning architecture for the development of surface roughness classifier using sound and cutting force signals. *Neural Comp. Appl.* 35, 1–18. doi: 10.1007/s00521-023-08425-z

Breiman, L. (1999). *Random forests - Random Features. Technical Report 567, Statistics Department*. Berkeley: University of California. Available online at: ftp://ftp.stat.berkeley.edu/pub/users/breiman.

Chang, L., and Rong, H. (2019). "Application of an improved BP AdaBoost model in semiconductor quality prediction," in *2019 IEEE International Symposium on Predictive Control of Electrical Drives and Power Electronics (PRECEDE)*. Quanzhou, China: IEEE, 1–4.

Chen, L. (2017). Continuous Delivery: Overcoming adoption challenges. *J. Syst. Softw.* 128, 72–86. doi: 10.1016/j.jss.2017.02.013

DevOps Trends Survey (2020). *Atlassian & CITE Research*. Available online at: https://www.atlassian.com/whitepapers/devops-survey-2020

Haas, R., Niedermayr, R., Roehm, T., and Apel, S. (2020). Is static analysis able to identify unnecessary source code? *ACM Trans. Softw. Eng. Methodol.* 29, 1. doi: 10.1145/3368267

Hamunen, J. (2016). *Challenges in Adopting a DevOps Approach to Software Development and Operations*. Available online at: http://urn.fi/URN:NBN:fi:aalto-201609083476

Hasselbring, W., Henning, S., Latte, B., Möbius, A., Richter, T., Schalk, S., et al. (2019,. March). "Industrial DevOps," in *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*. Hamburg: IEEE, 123–126.

Humble, J., and Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Boston: Addison-Wesley Professional.

Iain, M. (2018). "Cockburn Rebecca Henderson Scott Stern," in *The Impact Of Artificial Intelligence On Innovation Working Paper 24449*. Available online at: http://www.nber.org/papers/w24449 (accessed February 04, 2023).

Jadhav, S. D., and Channe, H. P. (2016). Comparative study of K-NN, naive Bayes and decision tree classification techniques. *Int. J. Sci. Res.* 5, 1. doi: 10.21275/v5i1.NOV153131

Kärpänoja, P., Virtanen, A., Lehtonen, T., and Mikkonen, T. (2016). "Exploring peopleware in continuous delivery," in *Proceedings of the Scientific Workshop Proceedings of XP2016 on - XP '16 Workshops*. doi: 10.1145/2962695.2962708

Li, M., Xu, H., and Deng, Y. (2019). Evidential decision tree based on belief entropy. *Entropy* 21, 897. doi: 10.3390/e21090897

Nevendra, M., and Singh, P. (2019). "Software bug count prediction via AdaBoost R.-E. T.," in *2019 IEEE 9th International Conference on Advanced Computing (IACC)*. Tiruchirappalli, India: IEEE, 7–12. doi: 10.1109/IACC48062.2019.8971588

Oates, J., Shafiabady, N., Ambagtsheer, R., Beilby, J., Seiboth, C., and Dent, E. (2022). Evolving hybrid partial genetic algorithm classification model for cost-effective frailty screening: Investigative study. *JMIR Aging* 5, e38464. doi: 10.2196/38464

Olszewska, M., and Waldén, M. (2015). "DevOps meets formal modelling in high-criticality complex systems," in *Proceedings of the 1st International Workshop on Quality-Aware DevOps - QUDOS 2015*. doi: 10.1145/2804371.2804373

Pal, M. (2005). Random forest classifier for remote sensing classification. *Int. J. Rem. Sens.* 26, 217–222. doi: 10.1080/01431160412331269698

Rafi, S., Yu, W., Akbar, M. A., Alsanad, A., and Gumaei, A. (2020). Multicriteria based decision making of DevOps data quality assessment challenges using fuzzy TOPSIS. *IEEE Access* 8, 46958–46980. doi: 10.1109/ACCESS.2020.2976803

Raileanu, L. E., and Stoffel, K. (2004). Theoretical comparison between the gini index and information gain criteria. *Ann. Math. Artif. Intell.* 41, 77–93. doi: 10.1023/B:AMAI.0000018580.96245.c6

Sen, A., Baumgartner, L., Hei,ß, K., and Wagner, C. (2021). DevOps paradigm-a pedagogical approach to manage and implement IT projects. *Issues Inform. Syst.* 22, 4.

Senapathi, M., Buchan, J., and Osman, H. (2018). "DevOps capabilities, practices, and challenges: Insights from a case study," in *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering*, 57–67). doi: 10.1145/3210459.3210465

Shafiabady, N., Hadjinicolaou, N., Din, F. U., Bhandari, B., Wu, R. M. X., and Vakilian, J. (2023) Using artificial intelligence (AI) to predict organizational agility. *PLOS ONE* 18, e0283066. doi: 10.1371/journal.pone.0283066

Shahin, M. (2015). "Architecting for DevOps and continuous deployment," in *Proceedings of the ASWEC 2015 24th Australasian Software Engineering Conference on – ASWEC*. doi: 10.1145/2811681.2824996

Shahin, M., Babar, M. A., Zahedi, M., and Zhu, L. (2017). "Beyond continuous delivery: an empirical investigation of continuous deployment challenges," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. doi: 10.1109/ESEM.2017.18

Suescún-Monsalve, E., Pardo-Calvache, C. J., Rojas-Muñoz, S. A., and Velásquez-Uribe, A. (2021). DevOps in industry 4.0: a systematic mapping. *Revista Facultad de Ingeniería* 30, e13314–e13314. doi: 10.19053/01211129.v30.n57.2021.13314

Varia, J., and Mathew, S. (2014). "Overview of amazon web services," in *Amazon Web Services 105*. Available online at: https://media.amazonwebservices.com/AWS_Overview.pdf (accessed February 03, 2023).

Wu Robert, M. X., Wang, Y. W., Shafiabady, N., Zhang, H., Yan, W. J., Gou, J. W., et al. (2023). Using multi-focus group method in systems analysis and design: a case study. *PloS ONE* 18, e0281603. doi: 10.1371/journal.pone.0281603

Wyner, A. J., Olson, M., Bleich, J., and Mease, D. (2017). Explaining the success of adaboost and random forests as interpolating classifiers. *J. Mach. Learn. Res.* 18, 1558–1590. Available online at: https://jmlr.org/papers/v18/15-240.html (accessed March 10, 2023).