Pak. j. life soc. Sci. (2024), 22(2): 5192-5209 E-ISSN: 2221-7630;P-ISSN: 1727-4915

Pakistan Journal of Life and Social Sciences
www.pjlss.edu.pk

https://doi.org/10.57239/PJLSS-2024-22.1.00382

RESEARCH ARTICLE

Enhancing Supply Chain Management: Leveraging Machine Learning and Software Engineering for Reliable and Efficient Applications

Torky Althaqafi

College of Business, University of Jeddah, Jeddah, Saudi Arabia

ARTICLE INFO

Received: Apr 24, 2024

Accepted: Jul 9, 2024

Keywords

Software engineering
Empirical study
Supply chain management
Machine learning
Software development
Software application

***Corresponding Author:**

tmalthaqafi@uj.edu.sa

ABSTRACT

This research paper delves into the integration of machine learning (ML) and software engineering (SE) practices, aiming to enhance the reliability and efficiency of supply chain management applications. With the significant advancements in ML and its widespread adoption across various industries, there exists a growing interest in exploring the synergies between ML and SE. This study focuses on investigating the impact of ML integration on software development techniques and identifying areas for improvement. Grounded-theory oriented coding techniques were employed to conduct an empirical investigation. Through qualitative analysis, various issue topics and excellent activities topics were identified, shedding light on the challenges and best practices in integrating ML and SE. This research aims to provide valuable insights and recommendations for organizations looking to adopt machine learning and software engineering solutions in their supply chain operations. Through the analysis of data obtained from the questionnaire responses, this study seeks to identify both best practices and potential challenges related to the implementation of machine learning and software engineering in supply chain management. By doing so, the research aims to provide decision-makers with valuable guidance on how to effectively leverage these technologies, fostering innovation and gaining a competitive advantage in the field of supply chain management. The practical insights gained from this study serve as valuable guidelines for both practitioners and researchers seeking to integrate SE practices into ML projects within the realm of supply chain management. By adhering to these guidelines, developers can augment the reliability and efficiency of ML applications, ultimately benefiting supply chain management processes. This research serves as a crucial step towards bridging the gap between ML and SE, facilitating the development of more robust and effective applications in the field of supply chain management.

INTRODUCTION

Machine learning (ML) is one of the contemporary inventions which has improved numerous business and professional procedures as well as daily human activities. It is anticipated that machine learning techniques and applications will advance, pushing the boundaries of technology. Voice recognition software, picture identification, clinical issue, predictive modeling, as well as statistical arbitrage are a few examples of real applications of ML [1-4]. The use of machine learning carries a significant amount of responsibility because it has permeated so many businesses today. Developing a statement of the problem is the initial step in developing a machine learning program. It is

important to take considerable time on the issue and consider the ultimate aim because data engineers and researchers often neglect and deprioritize this phase. For example, an issue may arise in a specific industry and damage the profitability of the company. The main task is to identify the target, from which individuals should determine the measure to optimize.

Supply chain management plays a pivotal role in the success of businesses operating in today's highly competitive and interconnected global marketplace. As companies strive to optimize their supply chain processes, emerging technologies such as ML and software engineering (SE) practices offer tremendous potential to enhance the reliability and efficiency of supply chain management applications. This paper aims to explore the integration of ML and SE in the context of supply chain management, providing practical guidelines and empirical insights to foster the development of robust and effective applications. The rapid advancement of ML has transformed it from a theoretical concept into a commercially viable technology with broad applications. ML techniques empower systems to analyze large volumes of data, recognize intricate patterns, and make accurate predictions and decisions. In the realm of supply chain management, where efficient logistics management, demand forecasting, and inventory optimization are paramount, leveraging ML holds significant promise. However, integrating ML into supply chain management applications introduces unique challenges that require careful consideration of SE practices [5-8].

Machine learning is usually accompanied by some degree of ambiguity. The machine learning model ought to not be dependent on a whole infrastructure. Simply stated, developer should create an end-to-end approach where each component of the system can support itself. When necessary, they can move and modify the rest of the system [9, 10]. They can take the following actions to maintain the architecture:

- Utilize a test that includes feeding the algorithm with data. Additionally, contrast the statistics for the route with those for similar data processed outside.

- Evaluate the components of the system by separating each one, including data preprocessing, model construction, model testing, as well as model service. One might effectively duplicate and modify the system's components in this manner.

- Infrastructure components that are judged not valuable can be removed because they will simply add errors and defects.

Testing is crucial since they act as a buffer between the engineer and the system's problems. One must be sure to utilize multiple tests as well as sanity checking before releasing the model if developers want to provide the greatest customer experience for the machine learning application. Verify that the model's metrics can produce useful results. Standard measures like recall, F1 score, as well as accuracy can be employed for this purpose [11]. Figure 1 illustrates the integration of software engineering with machine learning techniques through a flow diagram. The diagram presents a step-by-step process of combining software engineering principles with machine learning to develop effective applications. This flow diagram demonstrates a holistic approach to integrate software engineering and machine learning, emphasizing the iterative nature of the process and the importance of collaboration between the two domains [12].

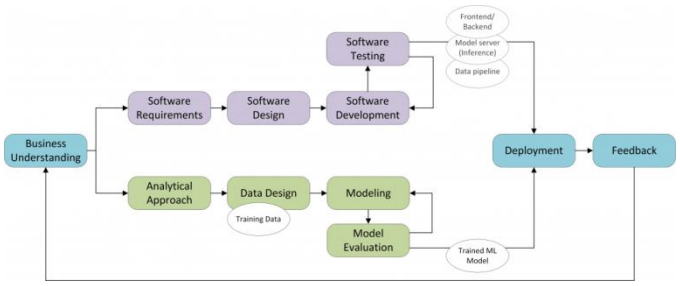

Figure 1 Integrating software engineering with machine learning techniques

The two process models for software engineering and machine learning must be combined in some way when creating a software application which employs these techniques. Integrating these two approaches demonstrates the necessity of developing two distinct streams (software as well as ML model) concurrently, beginning with a shared understanding and concluding with a combined software approach. If the streams are not run by the same individual, then this obviously calls for regular synchronisation between them. The implementation of agile machine learning practices can help with this problem by ensuring that both sources following an identical sprint rhythm, ideally in the same sprint group. In this method, a combined MVP (Minimum Viable Product) is delivered and evaluated during each sprint.

One requires a sizable amount of data in order to make accurate pattern detections or forecasts. Make sure the system developers are designing can collect enough information for business. If the data is inadequate, one could, as previously indicated, engage in the existing dataset and afterwards build the model's advancements on that. When transfer learning is used, some designers short-circuit information if necessary. Due to actual evidence can occasionally be inaccurate, sparse, or both, businesses could engage in feature extraction and data pre-processing to improve the quality. If done correctly, the data you obtain from the data collection element may be subjected to changes in the transformation element, including scaling, causal inferences, and many others [13, 14]. When applying the same modifications to the freshly obtained data from the system, the conversion component produces the training data. As a result, it can produce features that were derived utilizing only raw inputs.

Software engineering practices are vital for ensuring the reliability, scalability, and maintainability of applications. The seamless integration of ML algorithms with SE principles allows for the effective utilization of ML's predictive capabilities while addressing the specific requirements and constraints of supply chain management. By harnessing the synergy between ML and SE, practitioners can create applications that streamline operations, reduce costs, and improve customer satisfaction. The primary objective of this research paper is to provide comprehensive guidelines for integrating ML and SE practices within supply chain management. By delving into the impact of ML integration on software development techniques, we aim to identify the challenges and opportunities for improvement in this domain. Through an empirical investigation using grounded-theory oriented coding techniques, we extract valuable insights and practical recommendations for practitioners and researchers seeking to integrate SE practices into ML projects within the context of supply chain management. Ultimately, the insights gained from this study serve as a bridge between the domains of ML and SE, facilitating the development of reliable and efficient supply chain management applications. By following the guidelines outlined in this paper, stakeholders can unlock the full potential of ML while ensuring the robustness and effectiveness of their applications, thereby gaining a competitive edge in the dynamic landscape of supply chain management.

The research paper aims to achieve several objectives. Firstly, it seeks to investigate the potential

benefits and challenges associated with the integration of machine learning and software engineering in supply chain management. By examining real-world applications and case studies, the research aims to identify the ways in which these technologies can enhance the reliability and efficiency of supply chain operations. Secondly, the research aims to explore the impact of machine learning and software engineering on key supply chain metrics such as inventory management, demand forecasting, order fulfillment, and transportation optimization. By analyzing data from the questionnaire responses, the study intends to uncover the extent to which these technologies contribute to improved performance and cost savings within supply chain processes. To construct and distribute the questionnaire globally for this research work, a meticulous process was followed. The questionnaire was designed to gather insights on the integration of machine learning and software engineering in supply chain management. The distribution of the questionnaire was carried out through email, reaching a wide range of potential participants worldwide.

The remaining sections of the research paper are arranged as follows. In Section 2, we first go over the background and associated work. Following, in Section 3, we provide a description of the tools and techniques used in mining operations. Section 4 presents the findings of the study. In Section 5, we examine the interpretation and constraints of our observations. Section 6 concludes with reasonable conclusions and suggestions for further research.

RELATED WORKS

The integration of ML and SE in supply chain management has garnered significant attention in both academia and industry. Researchers have recognized the potential of ML techniques to revolutionize supply chain processes, offering advanced capabilities for demand forecasting, inventory optimization, risk management, and logistics optimization. Simultaneously, the application of SE practices ensures the reliability, scalability, and maintainability of software systems. In this section, we review existing literature and studies that have explored the integration of ML and SE in the context of supply chain management, shedding light on the current state of research, identifying gaps, and highlighting the benefits and challenges associated with this integration. By building upon the existing body of knowledge, this review lays the foundation for our research and provides a comprehensive understanding of the landscape in which we aim to enhance supply chain management through the synergistic use of ML and SE.

A research was conducted on monitoring Microsoft software developers as they create AI-based products and published by Amershi et al. [15]. They explored a nine-stage processing procedure that is based on the development of data science and AI tools (such as search as well as NLP) in the past (e.g. application diagnostics and bug reporting). They discovered that different Microsoft teams had combined this approach with preexisting, highly developed, Agile-like software development processes, offering insights into numerous crucial engineering difficulties that businesses may encounter when developing expansive AI products for the market. To deal with such issues, they gathered several best practises from Microsoft teams. Additionally, they have highlighted three characteristics of the AI domain that set it apart from earlier software application areas profoundly: 1) Finding, managing, and configuration management the data required for machine learning applications was significantly more complicated and challenging than other kinds of engineering; 2) model modification and model reusability necessitate entirely different abilities than are commonly found in software development teams; as well as 3) AI aspects were also more difficult to manage as different modules than conventional software components — designs may be "entangled" in complicated ways and encounter non-monotonic error behaviour. They believed that other companies would benefit from the insights that Microsoft teams have experienced.

The state of the art in how organizations create, release, and manage software incorporating ML

components was experimentally evaluated by Serban et al. [16]. They discovered 29 engineering best practises for ML applications by mining scholarly and grey publications. To ascertain the level of acceptance for such methods and to validate their alleged effects, they carried out a questionnaire among 313 professionals. They evaluated practise uptake using the survey results, differentiating it based on demographic factors like area or team size. Employing various predictive methods, they also assessed correlations and looked into linear and non-linear links between practises and their reported effects. Their research showed that larger teams tended to adopt more practises and that ML-specific practises were more widely adopted than typical software engineering practises. Additionally, the statistical models may effectively forecast observed consequences from the level of adoption for particular sets of practises, such as agility, system functionality, and accountability.

In close cooperation with businesses of all sizes and types, Arpteg et al. [17] used an explanatory research technique to identify the key difficulties. To illustrate the potential of this revolutionary technology and to pinpoint its primary obstacles, a group of seven projects have been chosen. The three categories of development, production, and organisational issues have been used to group together a total of 12 major challenges. Along with chosen inspiring explanations of how and why the issues apply to particular initiatives, a mapping among the difficulties and the projects was also established.

A thorough methodology for dataset production openness that promotes accountability as well as decision-making was introduced by Hutchinson et al. [18]. The methodology draws on best practises from the software development lifecycle by taking use of the cyclical, infrastructure-based, and engineering nature of dataset production. Every phase of the data production lifecycle produces documents that aid in better communication and decision-making while also highlighting the importance and value of diligent data work. The suggested approach highlighted the frequently hidden labour and choices that go into creating datasets, which is an important step in bridging the accountability barrier in AI and a crucial/essential resource in line with current work on auditing procedures.

For ML approaches, Washizaki et al. [19] compiled a list of excellent and problematic SE design patterns in order to give developers a thorough and organised classification of these patterns. Furthermore, they provided preliminary findings from an SLR of effective and ineffective design patterns for machine learning.

In order to identify substantial differences among the advancement of machine learning systems as well as the advancement of non-machine learning systems, Wan et al. [20] conducted a combination of qualitative and quantitative research findings with 14 interview participants and 342 questionnaire respondents from 26 countries throughout four continents. Their research revealed considerable disparities in job characteristics and many software engineering areas, such as requirements, development, testing, and procedure. They highlighted potential areas for further research basis of the findings and offered suggestions for professionals.

By concentrating on how software engineers could profit from implementing or adapting the conventional software development procedure to the Machine Learning workflow, Lorenzoni et al. [21] examined the difficulties and practises that originate during the development of ML models from the software engineering standpoint.

For the creation, management, and assembly of complex software components, Kriens & Verbelen [22] created tools and procedures. They provided an overview of the methods used today to manage complicated software and discuss how they relate to machine learning (ML) models.

Using historical data, Srinivasan & Fisher [23] developed estimators of software development effort using two machine learning techniques. Their tests revealed that such methods are superior to traditional estimators on a single dataset, but they also demonstrated that such strategies depend on the training data. This warning remark applied to any model-building approach that makes use of past data. Assessing model sensitivity on various historical data sets should be employed to assess all such models for development effort estimation.

Machine learning (ML) was used by Rahman et al. [24] in an enterprise case study to automatically identify transaction issues and suggest fixes. From three different perspectives—Software Engineering, Machine Learning, as well as industry-academia collaboration—they outlined and discussed the difficulties they encountered during this cooperative research and development project. They provided advice for the highlighted difficulties along with a report on our experience and project-related observations. They felt that their conclusions and suggestions could be useful to researchers and professionals starting similar projects.

Reimann & Kniesel-Wünsche [25] examined the lack of direction that currently employed development environments and machine learning (ML) APIs provide to programmers of ML applications, compared them to software engineering standard practises, and identified deficiencies in the state of the art.They demonstrated that some fundamental software engineering best practises are not met by current machine learning (ML) tools, as well as highlighted the ways in which software development notions, techniques, and strategies must be expanded and modified to meet the unique requirements of ML application development. Their results emphasised numerous areas where ML-specific software engineering research is necessary.

The integration of machine learning and software engineering practices in supply chain management applications is a rapidly evolving field with immense potential. The reviewed literature highlights the growing interest and significant advancements in leveraging ML techniques to enhance various aspects of supply chain management. Moreover, studies emphasize the importance of incorporating software engineering principles to ensure the reliability and efficiency of ML-powered solutions. However, while existing research provides valuable insights, there are still gaps to be addressed, such as specific guidelines and empirical evidence for integrating ML and SE in the context of supply chain management. The findings from this literature review lay the groundwork for our research, aiming to contribute practical guidelines and empirical insights that bridge the gap between ML and SE, ultimately leading to the development of reliable and efficient supply chain management applications.

MATERIALS AND METHODS

Initially, we compile a list of best practises from scholarly and grey literature [26-32]. This collection of knowledge can be utilized by professionals to enhance their growth process and acts as a starting point for related reading. The incorporation of the practises is measured in order to establish the state of the art. Such findings are employed to rate the practises according to their level of acceptance and can be utilized to gauge how well-liked a given practise is. Thirdly, we look into the interaction among sets of practises and their envisioned impacts using only a wide range of viewfinders. The implementation of practises according to the type of information being processed as well as depending on the practise categories mentioned above is then investigated.

Our findings imply that the techniques are broadly applicable to all ML applications and unaffected by the kind of data being used. We also discovered a significant relationship between practise groups and the outcome they were trying to achieve. We describe a strategy for prioritising practise improvements targeted for obtaining particular effects, such as better traceability or program

quality, utilising relevance of each practise to the intended impact and their widespread adoption rate. Several of our observations may have extensive applicability, even if our research is limited to machine learning (ML) instead of the more expansive and poorly defined subject of artificial intelligence (AI). The following Figure 2 shows the mining activities from existing literature.

Figure 2 Procedure of mining activities from literature

LITERATURE SEARCH

We looked into scientific and grey research on the optimum Software engineering activities for ML applications in addition to the papers covered in the literature review section. Both researchgate and Google Scholar were utilized, and we created a collection of common search terms for both. The terms used for searching, such as development, implementation, maintenance, etc., represent various phases of the development process. In addition, for every search, we created two variations by (1) changing instances of the term "machine learning" to "deep learning," as well as (2) eliminating stop words and creating a Boolean AND search from the essential words that remained. Take the search "software engineering" AND "machine learning," which is derived from the search string "software engineering in machine learning implementations," as an illustration of the second variation. The first five pages of each search outcome were carefully reviewed after being sent to Researchgate and Google Scholar. There were 62 searches used in all, including variations, and 47 of the articles returned were chosen for initial review. All searches were made using a wireless internet and a chrome browser that deletes cookies to prevent search engine personalization.

Literature categorization

We eliminated low-quality papers using the criteria, for example the authority of the source and authorship as well as fairness of the style and substance, and categorise the required documentation as either scholarly papers or research articles [33-40]. Also, we searched for duplication since grey literature occasionally duplicated sections of information. Following the findings' classification and screening, we found 19 pertinent documents, comprising published studies, white papers, blog posts, and powerpoint presentations. These documents—along with the articles mentioned in literature review section—were utilised to analyze Software Engineering's effective ML-related operations. Using a snowball approach, other pertinent sources were chosen by exploring references and arrows from the first articles.

The search phrases that were successful (without variations) and out of which minimum one article made the cut. Just the first query was taken into consideration when the results of the inquiries were similar. The articles chosen from the basic query and their variations are displayed as mentioned below.

Data classifying best practices, software development for machine learning, machine learning designing and implementing best activities, machine learning processing best activities, machine learning implementation best practices, machine learning infrastructure best practices, machine learning activities, machine learning regression testing best activities and machine learning practices.

Creating a practice categorization that is commonly applicable

Many of the chosen publications offer or imply a categorization of tasks based on ML-specific developing operations. We were able to reconstruct a wide taxonomy that is compatible with all partitionings identified in the literature, despite the fact that no specific partitioning of ML operations emerged as most authoritative. This classification will be used to classify ML development initiatives, to organise our investigation, and to arrange the explanation of our observations that follows:

Records - Tasks done prior to training, such as gathering and getting ready data for ML model development.

• Preparation - Actions involving the design, creation, and execution of training procedures.

• Implementation - Tasks involved in setting up a prototype for deployment, establishing it, and then keeping it in use.

• Coding - Tasks including the creation, testing, and implementation of code.

• Group - Tasks pertaining to coordination and communication within a software development group.

• Control - Actions that are related to assuring the ethical use of ML, such as responsibility for privacy, transparency, as well as the use of resources like time, money, or power.

Generating a list of Software Engineering activities

Generating a comprehensive and well-structured list of software engineering activities is crucial for effective project planning, development, and management. The field of software engineering encompasses a multitude of tasks and processes that are essential for the successful design, implementation, testing, and maintenance of software systems. This section of the research aims to provide an extensive and categorized list of software engineering activities, encompassing both fundamental and specialized tasks. By systematically identifying and organizing these activities, their relationships, and dependencies, this research serves as a valuable reference for software developers, project managers, and stakeholders involved in software development projects. The generated list of software engineering activities will facilitate better understanding, communication, and coordination among team members, enabling more efficient and streamlined software development processes.

Utilizing the following process, we created a preliminary set of tasks from the chosen papers. First, we determined which exams, activities, or suggestions shared the same aims. In other publications, the suggestions just outline the desired outcome, such as ensuring that trained ML systems can be linked to the input data and training procedures, without going into specifics about how to get there. Employ versioning for information, models, settings, as well as training scripts are a few examples of advice made in previous publications that specified specific procedures to take in order to accomplish the objectives. In this illustration, traceability is a result of accurately versioning all instructional artefacts. Every time we came into a situation that was comparable, we chose or simplified the executable tasks and assigned the high-level objectives to a unique category that we term activities

group and detail in Table 2. After that we evaluated the outcomes and chose those that were explicitly connected to technology or the structuring of systems engineering. Twenty eight tasks were provided by this initial pick, which naturally fit into 6 classifications mentioned above. This collection of tasks reflected the ML design process, although standard SE tasks were not included. In a third phase, we supplemented the preliminary set of assignments with 6 traditional SE activities—three of a purely technological nature, starting to fall into the "Coding" class, as well as three pertaining to social implications, falling into the "Team" class—because professionals with an extensive understanding in ML might not be cognizant of the advancements in SE. We chose these tasks because they are difficult yet crucial in software engineering, in our opinion. Table 1 lists the 30 activities that resulted, and Table 2 lists the different groups of software engineering activities as impacts. Practitioners can access the activities in a more complex style in an online catalogue1 that includes thorough descriptions and succinct declarations of intent, inspiration, supporting processes, references, as well as a level of complexity.

Table 1 Top activities in software engineering for implementing in machine learning

Activity No.	**Software engineering activities**	**Group**	**Rating**
1	Agile development methodology	Development methodologies	1
2	Waterfall development methodology		3
3	Using lean methodology, the production line is optimised to reduce waste and enhance customer satisfaction.		4
4	Prototype Model		5
5	Rapid Application Development (RAD) produces software in a lot less time while maintaining high quality		7
6	Scrum for an adaptable software development methodology		2
7	Allow for concurrent training activities	Rapid an	6
8	Deploy consistent Integration		3

		d ad ap tiv e	0
9	Turn Shadow deployment active		25
10	Automated Rollbacks for Production Units should be enabled		26
11	Coordinate, Integrate, and work together with Iindividuals of interdisciplinary teams		11
12	Concentrates on the calling structure's control flow	Co de an aly sis	12
13	Ensures that defined data is utilized appropriately and that data objects are functioning as intended		13
14	Examines the flaws and shortcomings of model parts		15
15	Checks simulations to ensure that the code is correct and that the interface is compatible with the model as well as simulation		17
16	Establishing the goals of software testing	Te sti ng	18
17	Changing the test duration and assets		22
18	Deploying and carrying out tests		23
19	Assessment of potential software bugs and problem identification		21
20	Provide reproducible scripts for combining and retrieving information.	Tr ac ea bil ity	8
21	Render private or public data sets accessible on shared network.		14
22	Employ versioning for training scripts, models, settings, and information.		10
2	Log Productivity Forecasts Using		1

3	Input Data as well as the Current Version of the model		9
24	Attempt to Reduce a Shared Backlog		9
25	Give every feature an ownership and record the justification for it	Software quality	29
26	Actively Delete or Archive Inactive Capabilities		28
27	Programs for peer assessment training		20
28	Implement automatic regression testing		27
29	Use Configuration Management		16
30	Assess the quality of the code using static code analysis		24

RESULTS

The data collection process yielded a total of 365 legitimate responses to our questionnaire. However, after excluding participants with insufficient comments or those who provided rushed responses, we obtained a final sample of 323 full replies for analysis. Additionally, 17 responses were eliminated as they came from individuals who were not part of a group utilizing machine learning.

The screening process involved assessing the proportion of queries answered in the preliminaries and the activity adoption questions. These filtering criteria ensured that only comprehensive and thoughtful responses were included in the analysis.

The subsequent analysis and findings presented in this paper are based on the 323 full responses obtained from the participants who met the criteria for inclusion. It is important to note that unless explicitly stated otherwise, the conclusions and interpretations are derived from this subset of responses. The robustness of the dataset and the careful selection of participants enable us to draw meaningful insights and provide valuable recommendations regarding the integration of machine learning and software engineering for enhancing supply chain management applications.

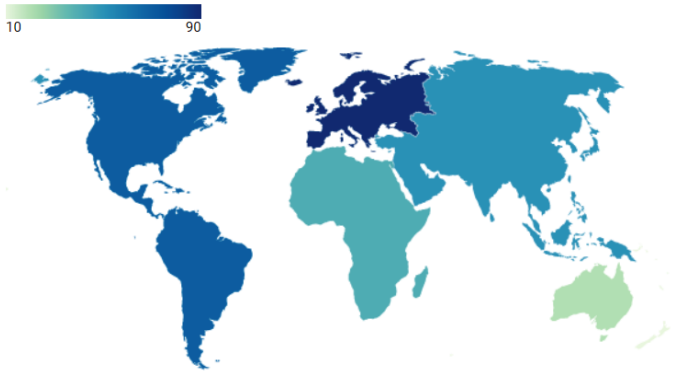

Figure 3 Geographic organization of survey participants

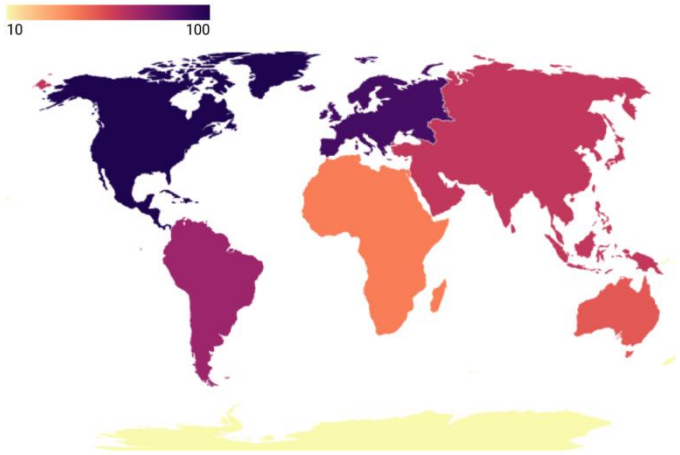

Figure 4 Acceptance of ML activities arranged by geographic location

The geographic organization of survey participants refers to how individuals or groups participating in a survey are organized based on their geographical location. This organization can provide valuable insights into the distribution of survey responses across different regions or countries. Figure 3 gives a geographic organization of the responders based on the early preliminaries. Initially, we organized the responses based on the location attributes, whereas Figure 4 shows the outcomes. While other countries are also well underrepresented, we note that Europe contributes more overall. This section's analysis of the responses for each region will include a discussion of this potential bias. The proportion of respondents broken down by type of company is shown in Figure 5. Groups employed by IT firms and research facilities have larger proportions. Such findings are not unexpected given that these two kinds of practitioners are responsible for both ML development and implementation. The governments and non-tech institutions are also highly represented, such as businesses and banks.

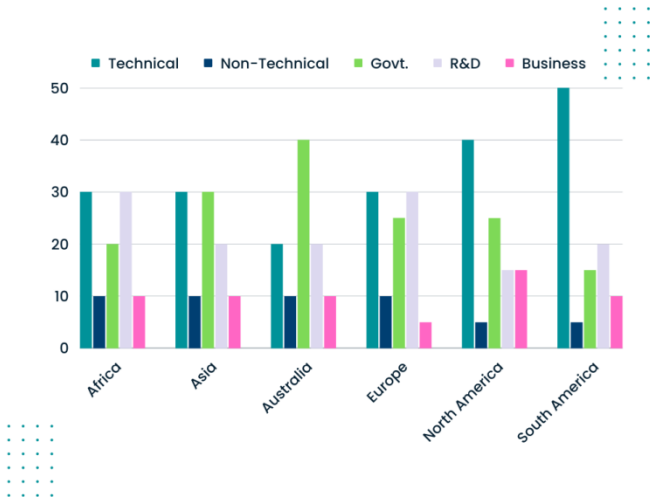

Figure 5 Implementation of practises arranged by organizational structure

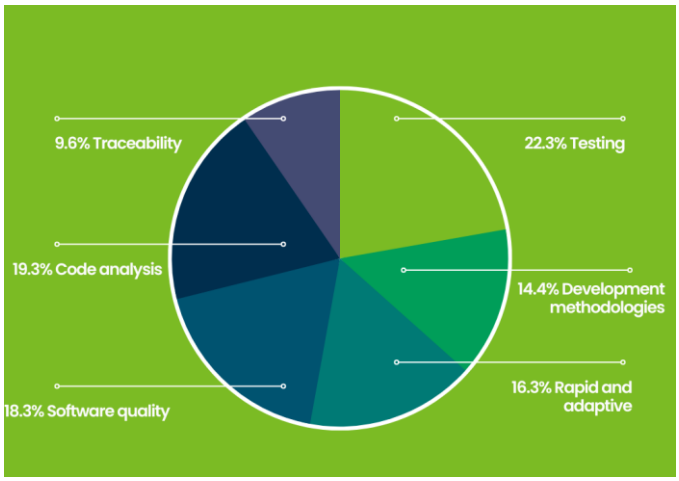

Figure 6 Acceptance of practises and their value, for every practise and consequence

Researcher notice that most groups, which match to typical software development teams comprise around 5-6 persons. The majority of teams also have between one and five years of professional experience, which is expected given that these time frames coincide with the recent increase in acceptance of ML amongst professionals. Generally, the geographical data shows that the data is diversified and very well. The acceptance of activities categorized by the before mentioned demographic characteristics was then examined. Figure 3 shows the responses to the activity items aggregated and standardized according to the study's 5-point likert - type scale. The proportions of responses are classified by areas in Figure 4. European Union is a little bit overrepresented in the data collection, as was previously mentioned. In contrast to South America or Asia, moreover, the acceptance of activities for European Union does not show any glaring distinctions. In contrast, North American respondents reported significantly more accepted activities than participants from several other region. Since this location is extensively covered in our collection of replies, it is most likely that North American professionals embrace activities more widely. Furthermore, since European union does not differ noticeably from other locations, there is probably no bias induced by its overrepresentation. The deployments of activities are categorized by type of company in Figure 5. We notice that technology companies achieve adaptation at a higher pace than other industries. The rate of activity uptake is typically lower in research groups. This may indicate that they are conscious

of the activities however are simply creating prototypes, for which full acceptance is not required. In actuality, adoption rates are comparable for non-deployment operations only.

We compare the acceptance ranking of the software engineering activities to every practice's value for a consequence in order to highlight it. Figure 6 shows the normalised rankings of different software engineering activities. As team size is increased, we see a tendency towards higher activity adoption (and a decrease in the fraction of activities that were never accepted. This may be the outcome of more effectively distributed tasks among team members or larger teams with more diverse membership. Corresponding to the individual experience, shows a trend towards increasing adoption of activities as team experience rises. Such outcomes were expected since group members who have more experience or have a greater understanding of technology are exposed to the knowledge they need to implement best practises. With only groups with more than five years of combined experience can a divergent trend be shown, in which the proportion of operations that are only completely or not at all embraced grows modestly. This finding can indicate that early adopters of a practise are not necessarily aware of more recent activity. These findings attest to the clarity and effectiveness of our questions as well as the fact that no bias was introduced by the response scale.

DISCUSSION

The increased utilization of digital technologies in industrial development, such as the Internet of Things (IoT), Intelligent systems, ML, robotics, cloud, and many others, contributes to improving product longevity and quality. Organizations are also implementing computer-aided technological solutions, which are predicted to boost market expansion, in order to reduce the time and cost associated with product innovation. The movement towards computer-aided software development is anticipated to be influenced by growing engineering practises including building information modelling (BIM), 3D printing, as well as integrated design. Individuals can print any product as a three-dimensional picture using advanced manufacturing techniques known as 3D printing. This innovation helps to reduce production costs as well as the creation of new production techniques. Design approaches, product development, ML modelling, as well as automation design are among the market segments for software engineering depending on applications. [41-46].

In our research observation agile methodologies gets first rank in software engineering activities. In the modern IT industry, agile development has become one of the most widely used methodologies. In addition, many software development approaches are founded on agile concepts. We provided information on the participants' demographics as well as the extent to which each characteristic's associated set of operations had been adopted. For instance, we discovered that activities specialized to ML tend to be more widely adopted than typical SE activities, and also that bigger teams generally embrace more operations. Furthermore, we discovered that technology companies implement activities more frequently than non-tech companies, government entities, or research institutes. Further research found that particular groups of behaviors positively associated with outcomes including traceability, quality and software, responsiveness, and collective efficacy. We were capable of creating forecasting analytics that are highly accurate at predicting these perceived outcomes of activity uptake. In order to identify which tasks warrant more or less consideration from the machine learning community, we compared the significance of activities—that is, their influence on potential benefits as indicated by such prediction models—with activity adoption. For instance, our findings indicate that increased usage of first activity, which involves agile methodology based activities. These very same conclusions can be applied at the level of teams or organizations to evaluate current activity use critically and priorities activity adoption according to desired outcomes. For instance, a team that has a high demand for agility but limited adoption of related activities may decide to do so.

The research findings provide insights into the analysis conducted based on the responses received

from the survey participants. The geographic organization of the survey participants was examined to understand the distribution of responses across different regions or countries. Europe was found to have a higher contribution overall, although other countries were underrepresented. The analysis acknowledges the potential bias introduced by this geographic distribution and discusses it in detail. The proportion of respondents was also analyzed based on the type of company they belonged to. It was observed that IT firms and research facilities had larger proportions, which is expected as these practitioners are involved in both ML development and implementation. Additionally, governments, non-tech institutions, businesses, and banks were also well-represented in the survey. Regarding team characteristics, most groups that matched typical software development teams comprised around 5-6 members. The majority of teams had between one and five years of professional experience, which aligns with the recent increase in ML acceptance among professionals. The geographic data indicated that the responses were diversified and well-represented. The acceptance of activities was examined based on the aforementioned demographic characteristics. While the European Union was slightly overrepresented in the data collection, the acceptance of activities in this region did not show any significant distinctions compared to other regions. In contrast, North American respondents reported significantly more accepted activities than participants from other regions, which may be attributed to the extensive coverage of North American professionals in the survey. The deployment of activities was also categorized by type of company, with technology companies showing a higher pace of adaptation compared to other industries. Research groups had lower rates of activity uptake, possibly indicating their focus on creating prototypes rather than full-scale implementation. The acceptance ranking of software engineering activities was compared to the value of each practice. The findings revealed a tendency towards higher activity adoption as team size increased, suggesting that larger teams with more diverse membership are able to distribute tasks effectively. Additionally, as team experience increased, there was a trend towards higher adoption of activities, as members with greater experience and understanding of technology were more likely to implement best practices. However, for groups with more than five years of combined experience, a modest increase was observed in the proportion of operations that were either completely or not at all embraced, indicating that early adopters may not always be aware of more recent activities. Overall, the research findings demonstrate the clarity and effectiveness of the survey questions and indicate that no bias was introduced by the response scale used in the study. These insights contribute to the understanding of integrating machine learning and software engineering practices, providing guidelines for the development of reliable and efficient applications.

CONCLUSIONS

In this research paper, we have explored the integration of machine learning (ML) and software engineering (SE) practices to enhance supply chain management applications. Through a comprehensive investigation, we have provided practical guidelines and empirical insights for leveraging ML and SE in a synergistic manner, ultimately aiming to develop reliable and efficient applications. The literature review revealed a growing interest in harnessing ML techniques to optimize various aspects of supply chain management, including demand forecasting, inventory optimization, risk management, and logistics optimization. Concurrently, SE practices have been recognized as crucial for ensuring the reliability, scalability, and maintainability of software systems. However, there is a need to bridge the gap between ML and SE to effectively leverage their combined potential.

Our empirical investigation using grounded-theory oriented coding techniques has shed light on the challenges and opportunities associated with the integration of ML and SE in supply chain management applications. We identified key issue topics and excellent activities topics, providing valuable insights into the integration process. By incorporating effective practices, such as data preprocessing, model validation, and performance monitoring, practitioners can enhance the

reliability and efficiency of ML applications in the supply chain management domain. The practical guidelines derived from this study serve as a valuable resource for both practitioners and researchers embarking on ML projects within supply chain management. By following these guidelines, stakeholders can navigate the complexities of integrating ML and SE, ensuring the development of robust and effective applications that optimize supply chain processes.

Furthermore, this research contributes to bridging the gap between ML and SE in the supply chain management domain. By leveraging the insights gained from this study, practitioners can create more reliable and efficient applications, resulting in improved operational performance, reduced costs, and enhanced customer satisfaction. While this research provides practical guidelines and empirical insights, there are still avenues for future exploration. Further studies could delve into specific industry contexts, evaluate the long-term impact of ML integration on supply chain management, and investigate emerging technologies that enhance the collaboration between ML and SE. The integration of ML and SE holds tremendous potential for enhancing supply chain management applications. By effectively leveraging ML techniques and incorporating SE principles, stakeholders can achieve reliable and efficient operations, ultimately gaining a competitive edge in the dynamic marketplace. Through our research, we have provided a foundation for practitioners and researchers to embark on this integration journey, facilitating the development of innovative and impactful solutions that revolutionize supply chain management.

REFERENCES

[1] Javaid, M., Haleem, A., Singh, R. P., Suman, R., & Rab, S. (2022). Significance of machine learning in healthcare: Features, pillars and applications. *International Journal of Intelligent Networks*, *3*, 58-73.

[2] Bynagari, N. B. (2015). Machine Learning and Artificial Intelligence in Online Fake Transaction Alerting. *Engineering International*, *3*(2), 115-126.

[3] Kamalov, F., Cherukuri, A., Sulieman, H., Thabtah, F., & Hossain, A. (2021). Machine learning applications for COVID-19: a state-of-the-art review. *arXiv preprint arXiv:2101.07824*.

[4] Ansari, M. T. J., Pandey, D., & Alenezi, M. (2022). STORE: security threat oriented requirements engineering methodology. *Journal of King Saud University-Computer and Information Sciences*, *34*(2), 191-203.

[5] Ansari, M. T. J., Al-Zahrani, F. A., Pandey, D., & Agrawal, A. (2020). A fuzzy TOPSIS based analysis toward selection of effective security requirements engineering approach for trustworthy healthcare software development. *BMC Medical Informatics and Decision Making*, *20*(1), 1-13.

[6] Saiedian, H., & Dale, R. (2000). Requirements engineering: making the connection between the software developer and customer. *Information and software technology*, *42*(6), 419-428.

[7] Krishnan, M. S., Kriebel, C. H., Kekre, S., & Mukhopadhyay, T. (2000). An empirical analysis of productivity and quality in software products. *Management science*, *46*(6), 745-759.

[8] Abdel-Hamid, T. K., Sengupta, K., & Swett, C. (1999). The impact of goals on software project management: An experimental investigation. *MIS quarterly*, 531-555.

[9] Saltzer, J. H., Reed, D. P., & Clark, D. D. (1984). End-to-end arguments in system design. *ACM Transactions on Computer Systems (TOCS)*, *2*(4), 277-288.

[10] Beck, K. (2000). *Extreme programming explained: embrace change*. addison-wesley professional.

[11] Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics*, *21*, 1-13.

[12] Heck, P. (2019). Software engineering for machine learning applications. *Fontys Blogt*.

[13] Peters, J., Janzing, D., & Schölkopf, B. (2017). *Elements of causal inference: foundations and learning algorithms* (p. 288). The MIT Press.

[14] Ansari, M. T. J., Baz, A., Alhakami, H., Alhakami, W., Kumar, R., & Khan, R. A. (2021). P-STORE: Extension of STORE methodology to elicit privacy requirements. *Arabian Journal for Science and Engineering*, *46*, 8287-8310.

[15] Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., ... & Zimmermann, T. (2019, May). Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (pp. 291-300). IEEE.

[16] Serban, A., van der Blom, K., Hoos, H., & Visser, J. (2020, October). Adoption and effects of software engineering best practices in machine learning. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* (pp. 1-12).

[17] Arpteg, A., Brinne, B., Crnkovic-Friis, L., & Bosch, J. (2018, August). Software engineering challenges of deep learning. In *2018 44th euromicro conference on software engineering and advanced applications (SEAA)* (pp. 50-59). IEEE.

[18] Hutchinson, B., Smart, A., Hanna, A., Denton, E., Greer, C., Kjartansson, O., ... & Mitchell, M. (2021, March). Towards accountability for machine learning datasets: Practices from software engineering and infrastructure. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (pp. 560-575).

[19] Washizaki, H., Uchida, H., Khomh, F., & Guéhéneuc, Y. G. (2019, December). Studying software engineering patterns for designing machine learning systems. In *2019 10th International Workshop on Empirical Software Engineering in Practice (IWESEP)* (pp. 49-495). IEEE.

[20] Wan, Z., Xia, X., Lo, D., & Murphy, G. C. (2019). How does machine learning change software development practices?. *IEEE Transactions on Software Engineering*, *47*(9), 1857-1871.

[21] Lorenzoni, G., Alencar, P., Nascimento, N., & Cowan, D. (2021). Machine learning model development from a software engineering perspective: A systematic literature review. *arXiv preprint arXiv:2102.07574*.

[22] Kriens, P., & Verbelen, T. (2019). Software engineering practices for machine learning. *arXiv preprint arXiv:1906.10366*.

[23] Srinivasan, K., & Fisher, D. (1995). Machine learning approaches to estimating software development effort. *IEEE Transactions on Software Engineering*, *21*(2), 126-137.

[24] Rahman, M. S., Rivera, E., Khomh, F., Guéhéneuc, Y. G., & Lehnert, B. (2019). Machine learning software engineering in practice: An industrial case study. *arXiv preprint arXiv:1906.07154*.

[25] Reimann, L., & Kniesel-Wünsche, G. (2020, March). Achieving guidance in applied machine learning through software engineering techniques. In *Companion Proceedings of the 4th International Conference on Art, Science, and Engineering of Programming* (pp. 7-12).

[26] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321-357.

[27] de Souza Nascimento, E., Ahmed, I., Oliveira, E., Palheta, M. P., Steinmacher, I., & Conte, T. (2019, September). Understanding development process of machine learning systems: Challenges and solutions. In *2019 acm/ieee international symposium on empirical software engineering and measurement (esem)* (pp. 1-6). IEEE.

[28] Alyami, H., Nadeem, M., Alharbi, A., Alosaimi, W., Ansari, M. T. J., Pandey, D., ... & Khan, R. A. (2021). The evaluation of software security through quantum computing techniques: A durability perspective. *Applied Sciences*, *11*(24), 11784.

[29] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. *Advances in neural information processing systems*, *28*.

[30] Alosaimi, W., Ansari, M. T. J., Alharbi, A., Alyami, H., Ali, S., Agrawal, A., & Khan, R. A. (2021). Toward a unified model approach for evaluating different electric vehicles. *Energies*, *14*(19), 6120.

[31] Garousi, V., Felderer, M., & Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and software technology*, *106*, 101-121.

[32] Herron, D. (2019). Principled Machine Learning: Practices and Tools for Efficient Collaboration.

[33] Alyami, H., Ansari, M. T. J., Alharbi, A., Alosaimi, W., Alshammari, M., Pandey, D., ... & Khan, R. A. (2022). Effectiveness evaluation of different IDSs using integrated fuzzy MCDM model. *Electronics*, *11*(6), 859.

[34] Hummer, W., Muthusamy, V., Rausch, T., Dube, P., El Maghraoui, K., Murthi, A., & Oum, P. (2019, June). Modelops: Cloud-based lifecycle management for reliable and trusted ai. In *2019 IEEE International Conference on Cloud Engineering (IC2E)* (pp. 113-120). IEEE.

[35] Alzahrani, F. A., Ahmad, M., & Ansari, M. T. J. (2022). Towards design and development of security assessment framework for internet of medical things. *Applied Sciences*, *12*(16), 8148.

[36] Ishikawa, F., & Yoshioka, N. (2019, May). How do engineers perceive difficulties in engineering of machine-learning systems?-questionnaire survey. In *2019 IEEE/ACM Joint 7th International Workshop on Conducting Empirical Studies in Industry (CESI) and 6th International Workshop on Software Engineering Research and Industrial Practice (SER&IP)* (pp. 2-9). IEEE.

[37] Khomh, F., Adams, B., Cheng, J., Fokaefs, M., & Antoniol, G. (2018). Software engineering for machine-learning applications: The road ahead. *IEEE Software*, *35*(5), 81-84.

[38] Alassery, F., Alzahrani, A., Khan, A. I., Khan, A., Nadeem, M., & Ansari, T. J. (2022). Quantitative Evaluation of Mental-Health in Type-2 Diabetes Patients Through Computational Model. *Intelligent Automation & Soft Computing*, *32*(3).

[39] Kitchenham, B. A., & Pfleeger, S. L. (2002). Principles of survey research part 2: designing a survey. *ACM SIGSOFT Software Engineering Notes*, *27*(1), 18-20.

[40] Alharbi, A., Ansari, M. T. J., Alosaimi, W., Alyami, H., Alshammari, M., Agrawal, A., ... & Khan, R. A. (2022). An Empirical Investigation to Understand the Issues of Distributed Software Testing amid COVID-19 Pandemic. *Processes*, *10*(5), 838.

[41] Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., ... & Lee, S. I. (2020). From local explanations to global understanding with explainable AI for trees. *Nature machine intelligence*, *2*(1), 56-67.

[42] Lwakatare, L. E., Raj, A., Bosch, J., Olsson, H. H., & Crnkovic, I. (2019). A taxonomy of software engineering challenges for machine learning systems: An empirical investigation. In *Agile Processes in Software Engineering and Extreme Programming: 20th International Conference, XP 2019, Montréal, QC, Canada, May 21–25, 2019, Proceedings 20* (pp. 227-243). Springer International Publishing.

[43] Agrawal, A., Khan, R. A., & Ansari, M. T. J. (2022). Empowering Indian citizens through the secure e-governance: the digital India initiative context. In *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2022, Volume 3* (pp. 3-11). Singapore: Springer Nature Singapore.

[44] Roh, Y., Heo, G., & Whang, S. E. (2019). A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering*, *33*(4), 1328-1347.

[45] Alshahrani, H. M., Alotaibi, S. S., Ansari, M. T. J., Asiri, M. M., Agrawal, A., Khan, R. A., ... & Hilal, A. M. (2022). Analysis and Ranking of IT Risk Factors Using Fuzzy TOPSIS-Based Approach. *Applied Sciences*, *12*(12), 5911.

[46] Serban, A., van der Blom, K., Hoos, H., & Visser, J. (2020). Adoption and Effects of Software Engineering Best Practices in Machine Learning-Supplementary Material. Zenodo.