International Journal for Quality Research x(x) x–x
ISSN 1800-6450

PhD student, Ognjen Tomić
Lola institute, Belgrade, Serbia

Article info:
Received xx.xx.xxxx
Accepted xx.xx.xxxx

UDC – xxxxxxxxxx
DOI – 10.18421/IJQRxx.xx-xx

APPLYING MLOPS TO OPTIMIZE SOFTWARE QUALITY ASSURANCE IN THE ANALYTICS APPLICATION MARKET -LITERATURE REVIEW-

Abstract: *In this research, we explore the application of MLOps (Machine Learning Operations) as a strategy to optimize software quality assurance in the analytics application market. Using the MLOps approach, we integrate tools such as Databricks, Apache Zeppelin, KNIME and RapidMiner to automate the testing, analysis and optimization processes of software applications.*

This case study illustrates specific examples of the application of MLOps in the context of analytics applications, but also provides a basis for understanding the broader range of opportunities that MLOps provides in improving software quality assurance in the marketplace. Through this integration of tools and practices, we explore how MLOps can be a key factor in achieving competitive advantage and long-term success in the analytics software market.

Keywords: *Machine learning, analystics app., software, quality assurance*

1. Introduction

1.1. *The basics of MLOps*

MLOps application context refers to the situations and circumstances in which MLOps, or machine learning operations management practices, are applied to ensure effective implementation, maintenance, and improvement of machine learning (ML) systems [1]. MLOps is particularly important in environments where a diverse range of machine models are used. This context includes situations where image recognition, natural language processing or regression models are present, with the need to manage their diversity and complexity. In software development, MLOps is applied to integrate machine learning development with standard software processes. This includes version control, code management, testing and implementation. Situations where it is necessary to quickly adjust models or introduce new iterations require MLOps practices. This is especially true in areas where data changes frequently or where it is necessary to respond quickly to new requirements. MLOps is crucial in sectors where high standards are set for data security and regulatory compliance [2]. Applications with high data volume or variable requirements require an MLOps approach to ensure scalability and efficient resource management. In areas where data is dynamic and requires continuous model education,

[1] Corresponding author: *PhD student, Ognjen Tomić,*
Lola institute Kneza Viseslava 70a
Belgrade, Serbia
Email: *ognjen.tomic@li.rs*

MLOps is critical to keeping models up-to-date and accurate. Introducing MLOps into these contexts allows organizations to effectively manage the complexity of machine learning systems, ensure high levels of performance, and adapt more quickly to changes in the environment.

2.2. *MLOps cycle*

The MLOps cycle, or MLOps process, is a continuous sequence of activities implemented to develop, test, implement, and maintain machine models over time. Identification of the specific problem that the machine model should solve (Figure 1). Collection of relevant data necessary for model training and evaluation. This phase involves cleaning, transforming and normalizing the data to make it suitable for machine learning. Creating and training a machine model using prepared data [3].
Here, different algorithms and parameters are experimented with in order to achieve the best possible accuracy. Evaluation of model performance on separate data sets that were not used during training. This includes parameter validation and testing to ensure the generalizability of the model. Integration of the model into the real environment. Here, MLOps practices are used to deploy the model into production, including automated processes for updating and rolling back old versions when needed. Setting up a real-time model performance monitoring system [4]. This enables the identification of potential problems and performance degradation in anticipation of a negative impact on users. Continuous optimization of the model based on new data and experiences. Updating the model to maintain high accuracy and adequately respond to changes in the environment. The MLOps cycle is a dynamic process that implies continuous attention, adaptability and cooperation between different teams in order to achieve and maintain a high level of performance of machine models in production [5].

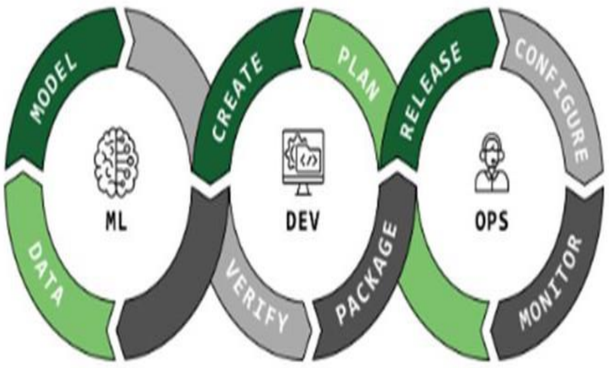

Figure 1. MLOps cycle [9]

2. Elements of MLOps

Identification of key elements in MLOps, including automation, model performance monitoring:

Automation is a fundamental element of MLOps, and its goal is to reduce human intervention and accelerate all stages of development, [6] implementation and maintenance of machine learning models (Figure 2).

1. **Continuous integration** (CI): Implementation of CI/CD (Continuous Integration/Continuous Deployment) practices enables automatic testing and integration of new code into a shared repository. This ensures a faster flow of the development cycle.
2. **Automating model training**: Creating processes that automatically train models when new data is delivered, while monitoring performance and adjusting models as needed.
3. **Automation of deployment**: Automation of the process of deploying the model to the production environment, including dependency management, environment configuration and resource monitoring.

4. **Automation of performance testing**: Regular automated testing of model performance to identify problems and ensure constant optimal functioning.

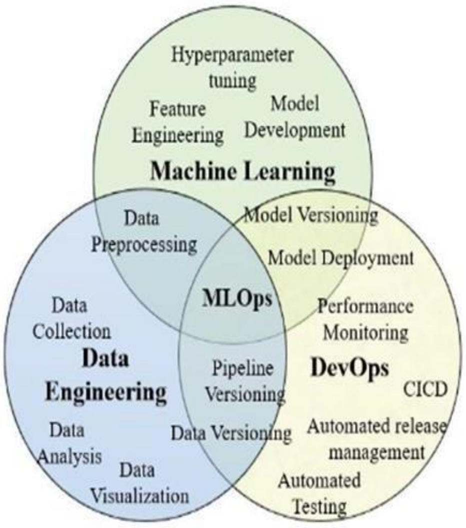

Figure 2. MLOps combination [9]

Monitoring model performance is critical to ensuring sustainable and effective model deployment in a real-world environment.

1. **Monitor model quality metrics**: Continuously monitor key metrics, such as accuracy, precision, and responsiveness, to identify changes in model performance.
2. **Model health monitoring**: Continuous monitoring of model health, including proactive recognition of performance degradation and alerts in case of irregularities.
3. **Analysis of the impact of data changes**: Monitoring how new data affects the performance of the model, thus ensuring the adaptation of the model to the changed conditions.
4. **Resource Tracking**: Track resources consumed during model training and implementation to optimize efficiency and control costs.

3. Integrating MLOps into software development

3.1. Contemporary challenges of software development

Software development faces a number of challenges, and the integration of MLOps provides the opportunity to solve some of the key problems in this area (Table 1):

Table 1. Contemporary challenges of software development

Complexity of current development processes	Challenge: Traditional development processes can be complex and inefficient, especially when it comes to implementing machine learning models. MLOps solution: MLOps enables the automation and standardization of the entire model life cycle, simplifying the processes from model training to its implementation.[7] [9]
Lack of coordination between development and operations teams	Challenge: Lack of collaboration between development and operations teams can lead to problems in implementing and maintaining the model. MLOps solution: MLOps integration creates a bridge between development and operations teams, fostering collaboration through shared tools and processes. [9]
Lack of transparency in	Challenge: Lack of clear visibility into the lifecycle of a model can lead to

the life cycle of the model	problems in monitoring performance and identifying the need for updates. MLOps Solution: MLOps provides tools to track all stages of a model's life, from training to implementation, ensuring transparency and change tracking.[7] [9]
Difficulties in managing different versions of the model	Challenge: Managing multiple versions of a model can be a challenge, especially when a large number of models need to be updated and maintained. MLOps Solution: MLOps enables simple model versioning, supporting easy update, pull, and change tracking.[8]
Lack of reproducibility in model training	Challenge: Lack of reproducibility in the model training process can lead to problems in validation and retraining. MLOps Solution: MLOps provides an environment that enables accurate reproducible training of models, thus ensuring consistency and validity of results.

Integrating MLOps into software development can significantly improve efficiency, transparency and governance in the development and implementation of machine learning models, responding to current challenges and improving the overall process.

3.2. MLOps integration steps

Integrating MLOps into existing software development processes requires a carefully planned implementation to ensure a smooth transition and realize the desired benefits. Here are the steps that are key to this integration [9]

Detailed analysis of existing software development processes to identify the correct steps, resources and roles in the current environment. Identification of key stages in the software development life cycle where MLOps could bring added value, e.g. in training, testing, implementation and maintenance of the model. Conduct training for team members on MLOps concepts, tools to be used and changes to be introduced in the work process. Selecting appropriate tools and platforms to support MLOps, including tools for version control, automation, monitoring and resource management. Setting standards and guidelines for managing models, data, and code to ensure consistency and ease of maintenance. Implement continuous integration (CI) and continuous delivery (CD) into the model training process to ensure automatic model verification, testing, and deployment. Introducing a model performance monitoring system to automatically detect model degradation and respond to changes in a timely manner. Implementation of a configuration management system to enable efficient management of model settings and environment configuration. Conduct testing and validation on the integrated MLOps system to ensure functionality and quality. Defining the model maintenance process, including scheduled updates, tracking changes and responding to user needs. Introducing mechanisms for continuous improvement of the integrated MLOps system, taking into account feedback from the team and end users. Monitoring the performance of the integrated MLOps system to assess effectiveness and take corrective action as needed.

Integrating MLOps into existing software development processes requires a systematic and careful approach. These steps are the foundation for successful integration, ensuring consistency, transparency and efficiency in the management of machine learning models.

3.3. Benefits of integration

Integrating MLOps into the software development process can bring a number of significant benefits related to various aspects, including speeding up development time, improving model quality, and facilitating model lifecycle management.

1. **Acceleration of development time**

Benefit: The implementation of MLOps enables the automation of key phases of model development, including training, testing and implementation. This leads to an acceleration of the entire development cycle, reducing the time it takes from conceptualization to product on the market.

2. **Improvement of model quality**

Benefit: MLOps provides frameworks for continuous testing and real-time monitoring of model performance. This allows problems to be identified and resolved before they affect users, resulting in improved model quality.

3. **Easier model lifecycle management**

Benefit: MLOps integration simplifies the management of all phases of the model life cycle, including training, testing, implementation and maintenance. This makes it easier for teams to effectively manage changes, versions and configuration of models. [13]

4. **Increasing reproducibility**

Benefit: MLOps enables accurate reproduction of the environment during model training, ensuring consistency of results and enabling easier validation and reproduction of experiments.

5. **More efficient use of resources**

Benefit: Automation enables better management of resources during model training and implementation, leading to more efficient use of infrastructure and cost reduction.

6. **Improvement of team cooperation**

Benefit: MLOps supports integration between different teams, including development, operations and data exploration. This improves collaboration, reduces gaps between teams and contributes to better understanding and management of models.

7. **Faster response to changes**

Benefit: Integrating MLOps enables teams to react more quickly to changes in data, environment or customer requirements, ensuring agility in adapting models.

8. **Reducing the risk of problems in the production environment**

Benefit: Continuous monitoring and testing of model performance helps identify potential problems before they affect the production environment, reducing the risk of adverse incidents.[14]

9. **Improving data security**

Benefit: MLOps includes mechanisms for managing data security during the entire lifecycle of the model, providing additional security when working with sensitive information.

10. **Transparency and monitoring of changes**

Benefit: MLOps enables tracking of all changes in models and code, ensuring transparency in the development process and enabling better change management.

Overall, the integration of MLOps brings a number of key benefits that improve the efficiency, quality and management of machine learning models throughout their lifecycle.

4. Case study: Integrating MLOps into analytics application development

4.1. Databricks

One example of a data analytics application that can serve as a foundation for MLOps integration is the Databricks Platform (Figure 3).

Figure 3. Databricks platform

Databricks provides a comprehensive ecosystem for data analytics and machine learning, offering tools for working with large data sets and support for integration with MLOps practices.

Machine learning support: Databricks provides support for popular machine learning libraries like Apache Spark MLlib and scikit-learn. Also, it enables easy training, evaluation and implementation of the model directly in the platform.

Model training automation: Databricks makes it easy to automate the model training process through the use of scripts and notebooks. This allows easy integration with CI/CD tools and setting up automatic training processes.

Model performance monitoring: Databricks provides the ability to monitor model performance through built-in libraries and integration with monitoring tools such as MLflow. This allows the team to monitor metrics and identify the need to optimize the model.

Integration with MLOps tools: Databricks can be easily integrated with MLOps tools like Apache Airflow, Jenkins, or Azure DevOps, enabling automation, monitoring, and management throughout the model lifecycle.

Scalability and resilience: The Databricks Platform is scalable and adaptable, which makes it suitable for working with large data sets and complex analyses. This scalability is essential for model management in a production environment.

Openness to integration: Databricks supports working with different data sources and formats, and also allows easy integration with various data warehouses, whether they are in the cloud or on-premises.

This example illustrates how Databricks can serve as a solid foundation for MLOps integration, providing tools for data analysis and machine learning, with support for automation, performance monitoring, and integration with MLOps practices.

4.2. Apache Zeppelin

Apache Zeppelin is another example of a data analytics application that can serve as a foundation for MLOps integration.

Figure 4. Apache Zeppelin [10]

Zeppelin is an open-source project designed for interactive data analysis, visualization and machine learning execution.

Interactivity and visualization: Zeppelin provides an interactive environment that supports multiple programming languages, including Scala, Python, R, and SQL. This enables analysts and data scientists to effectively explore and visualize data.

Built-in machine learning tools: Zeppelin has built-in machine learning libraries and tools, including support for Apache Spark, TensorFlow, and scikit-learn. This allows for easy training and testing of models directly within the Zeppelin environment. [10]

Support for continuous integration delivery (CI/CD): Zeppelin can be integrated with CI/CD tools like Jenkins, which enables automation of model training, evaluation and deployment processes.

Easy sharing and reproducibility: Zeppelin allows easy sharing of analyzes and experiments, and also saves session state. This provides the ability to reproduce results and share analytical results within the team.

Integration with MLOps platforms: Zeppelin can be integrated with popular MLOps platforms like MLflow, providing support for tracking metrics, managing experiments and implementing models.

Dynamic configuration: Zeppelin allows dynamic configuration of the interpreter and the environment, which is useful for adapting the environment to the specific needs of MLOps and the team.

Security and access management: Zeppelin provides access management capabilities and security mechanisms, which are important for managing sensitive data and analytics.

Openness and flexibility: Zeppelin is an open-source project with a large community of users, which allows customization and expansion of functionality according to the needs of the team.

Zeppelin can serve as the foundation for MLOps integration by providing analysts and data scientists with an interactive environment for working with data and machine learning models, with support for automation and performance monitoring.

4.3. KNIME

KNIME (Konstanz information miner) Analytics platform is another powerful data analysis application that can be used as the basis for MLOps integration.

Figure 5. KNIME Konstanz Information Miner [11]

KNIME provides visual programming, allowing users to construct analytics workflows without having to write code. [11]

Visual programming: KNIME enables users to create analytical workflows by connecting visual nodes, facilitating the construction of complex data analysis and machine learning models without the need for coding.

Extensive library of data analysis tools: KNIME includes a rich library of nodes for data analysis, statistics, and machine learning. This provides flexibility in experimenting and applying different methods and techniques.

Integration with various data sources: KNIME supports integration with various data sources, including SQL databases, CSV files, Hadoop and others. This allows working with different types of data and sources of information.
Automation of workflows: KNIME facilitates the automation of workflows, which is crucial for the implementation of MLOps principles. Users can set scheduled tasks, optimize workflows and easily repeat analyses.

Integration with MLOps tools: KNIME is integrative and can be easily connected to MLOps platforms like MLflow, Kubeflow and others. This allows tracking experiments, managing models and applying MLOps principles within the KNIME environment.

Model creation and sharing: KNIME enables the creation, training and sharing of models within workflows. This supports team collaboration and allows tracking changes in models over time.

Security and access control: KNIME provides access management capabilities and the implementation of security rules, providing control over data and analytical workflows.

Openness and active community: KNIME is an open-source platform with an active community of users. This gives users the opportunity to customize and extend functionality according to specific needs.[11]

The KNIME analytics platform combines powerful visual programming with a wide range of data analysis tools, making it the ideal foundation for integrating MLOps and effectively managing machine learning models.

4.4. RapidMiner

RapidMiner is an advanced data analysis platform that allows users to easily build analytical models and perform complex data analysis.

This platform can serve as an excellent foundation for MLOps integration, providing support for automation, model performance monitoring, and team collaboration. Here's how RapidMiner meets the key features for MLOps integration:[12]

Visual programming and data analysis: RapidMiner uses visual programming that allows users to construct analytical models using a graphical interface without writing code. This simplifies the process of creating and understanding analytical workflows.

Support for machine learning and data analysis: RapidMiner provides an extensive library of tools and nodes for machine learning and data analysis. These tools allow users to apply various data analysis methods, including machine learning models.[12]

Automating workflows: The platform enables users to automate analytical workflows, thereby reducing the need for manual interventions. Automation contributes to efficient model life cycle management.

Integration with MLOps tools: RapidMiner can be integrated with various MLOps tools, including MLflow, Kubeflow and others. This integration enables model performance monitoring, experiment management and easier implementation of MLOps principles.

Ease of sharing models and workflows: RapidMiner allows users to easily share models and workflows with team members. This supports team collaboration and makes it easier to work together on projects.

Model performance metrics tracking: The platform allows users to track key model performance metrics over time. This functionality is essential for model evaluation and optimization.

Scalability and efficiency: RapidMiner is scalable and efficient in working with large data sets, thus providing the ability to manage models at different levels of complexity.

Openness and active community: RapidMiner is also an active open-source project with a vibrant community of users. This provides users with the opportunity to learn, share experiences and adapt the platform to specific needs.

RapidMiner, with its intuitive visual programming platform, is an excellent example of an analytics application that can serve as a foundation for MLOps integration and lifecycle management of machine learning models

5. Implementation of MLOps

Implementing MLOps in application development involves a series of steps that include planning, modeling, testing, implementation, and maintenance. Below is a detailed breakdown of the steps involved in implementing MLOps for the example applications we've previously discussed - Databricks, Apache Zeppelin, KNIME Analytics Platform, and RapidMiner. For each application, the tools and techniques that can be used are listed separately.

Databricks Platform

A. Defining MLOps objectives.

Tools: Databricks Workspace for defining goals and team communication. Techniques: Working in Databricks notebooks for collaborative planning.) Identification of key elements of the model development process.

B. Development of the model:

Using Databricks notebooks for experimentation and model development. Tools: Databricks notebooks, Apache Spark MLlib, MLflow for model experimentation, development and monitoring. Techniques: Using Databricks experimental libraries for model optimization and testing. Integration with Apache Spark MLlib for machine learning. Use MLflow to track experiments, manage models, and log metrics.

C. Testing:

Automated model testing and performance validation through MLflow. Tools: MLflow for automatic tracking of test results and performance evaluation. Techniques: Integrating automated tests into the Databricks environment.) Monitoring of test results and evaluations.

D. Implementation:

Automation of the model implementation process. Tools: Jenkins for deployment automation, Databricks API for integration with CI/CD tools. Techniques: Setting planned tasks for regular model implementation.) Integration with CI/CD tools (eg Jenkins) for continuous delivery.

E. Maintenance:

Monitoring model performance in production with MLflow. Tools: MLflow for monitoring model performance in production. Techniques: Automated model updating via MLflow tools. Automated model updating based on new data

Apache Zeppelin

A. Planning:

Setting goals for MLOps integration into the Zeppelin environment. Identification of key features for model monitoring and management. Tools: Apache Zeppelin for setting goals and defining features for MLOps integration. Techniques: Using Zeppelin notebooks for collaborative team planning.

B. Development of the Model:

Using Zeppelin notebooks to build and experiment with models. Integration with MLOps platforms like MLflow to track experiments. Tools: Apache Zeppelin notebooks, MLOps platforms like MLflow for tracking experiments. Techniques: Using the Zeppelin API to integrate with MLOps tools.

C. Testing:

Implementation of automated tests for model performance evaluation. Monitoring test results directly in the Zeppelin environment. Tools: Zeppelin notebooks for implementing automated tests.Techniques: Track test results directly in the Zeppelin environment.

D. Implementation:

Automated model deployment through integration with CI/CD tools. Using the Zeppelin API to integrate with MLOps tools. Tools: Jenkins for deployment automation, Zeppelin API for integration with CI/CD tools. Techniques: Setting planned tasks for regular implementation of the model.

E. Maintenance:

Model performance monitoring through Zeppelin notebooks. Periodic model updating and optimization. Tools: MLOps platforms like MLflow for monitoring model performance. Techniques: Periodic model updating and optimization through the Zeppelin environment.

KNIME Analytics Platform:888

A. Planning:

Defining MLOps objectives in the KNIME environment. Identification of key functionalities for monitoring and managing models.

B. Development of the Model:

Creating analytical workflows for development and experimentation. Integration with MLOps tools like MLflow to track metrics and experiments.

C. Testing:

Defining model performance tests within KNIME workflows. Automated testing using KNIME analytics nodes.

D. Implementation:

Automation of the implementation process through integration with CI/CD tools. Using KNIME Server to manage models in production.

E. Maintenance:

Model performance monitoring through KNIME Server and MLOps platforms. Periodic model updating and optimization.

RapidMiner

A. Planning:

Setting goals for integrating MLOps into RapidMiner. Identification of key functionalities for model lifecycle management.

B. Development of the model:

Using visual programming in RapidMiner Studio to build models. Integration with MLOps platforms like MLflow for tracking experiments and models.

C. Testing:

Creation and implementation of model performance tests within RapidMiner Studio. Test automation using RapidMiner Server.

D. Implementation:

Automation of model implementation through integration with CI/CD tools. Using RapidMiner Server to manage models in production.

E. Maintenance:

Model performance monitoring through RapidMiner Server and MLOps platforms. Periodic model updating and optimization.

6. **Conclusion**

Results of this research confirm that the application of MLOps is of critical importance for the optimization of software quality assurance in the analysis application market. We hope that this case study will serve as a basis for further research and application of MLOps in various industrial contexts, thus contributing to the innovation and advancement of software development worldwide. This case study highlights the key benefits of implementing MLOps, including faster iterations, real-time error detection, and continuous software optimization. Through this integration of tools and practices, we emphasize the importance of continuous evolution and adaptation in order to achieve a competitive advantage in the analytics applications market. [13], [14], [15].

References:

[1] Kreuzberger D., Kuhl N., Hirschl S., (2023). Machine Learning Operations (MLOps): Overview, Definition, and Architecture. Vol. 11, 31866-31879. doi: 0.1109/ACCESS.2023.3262138

[2] Subramanya R., Sierla S. Vyatkin V., (2022). From DevOps to MLOps: Overview and Application to Electricity Market Forecasting. Vol. 12 (9851), 1-31. doi: 10.3390/app12199851

[3] Zhengxin F., Zhang Y., Jingyu, Yue L., Yuechen M., Qinghua L., Xiwei X., Jeff W., Chen W., Shuai Z., Shiping C., (2020). MLOps Spanning Whole Machine Learning Life Cycle: A Survey 2-27.

[4] Nipuni Hewage, Dulani Meedeniya, (2022). Machine Learning Operations: A Survey On Mlops Tool Support. 1-12.

[5] Treveil M., (2021). Instroducing MLOps- How to scale ML in the enterprise. 3-159.

[6] Meenu J., Olsson H., Bosch J., (2021). Towards MLOps: A Framework and Maturity Model. 1-8.

[7] Geron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media.

[8] Satvik Garg, Pradyumn Pundir, Geetanjali Rathee, P.K. Gupta, Somya Garg, Saransh Ahlawat, (2022). On Continuous Integration / Continuous Delivery for Automated Deployment of Machine Learning Models using MLOps. 1-4.

[9] Georgios Symeonidis, Evangelos Nerantzis, Apostolos Kazakis, George A. Papakostas, (2022), MLOps - Definitions, Tools and Challenges, pp:1-8.

Links:

[10] Apache Zeppelin Documentation. (https://zeppelin.apache.org/)

[11] KNIME Documentation. (https://www.knime.com/knime)

[12] RapidMiner Documentation. (https://docs.rapidminer.com/)

[13] Microsoft Azure Documentation. (https://docs.microsoft.com/en-us/azure/)

[14] Google Cloud Documentation. (https://cloud.google.com/docs)

[15] Amazon Web Services Documentation. (https://aws.amazon.com/documentation/)

PhD student, Ognjen Tomić
Lola institute,
Belgrade,
Serbia
ognjen.tomic@li.rs