Implementing MLOps in the Restaurant Industry: A Case Study on Deploying Prediction Services Using Flask/ FastAPI and Cloud Platforms

Soon Bang, Suicheng Lai  
AI 510 Artificial Intelligence in Cloud Computing,

School of Technology & Computing,

City University of Seattle

[bangsoon@cityuniversity.edu,](mailto:bangsoon@cityuniversity.edu) [laisuicheng@cityuniversity.edu](mailto:laisuicheng@cityuniversity.edu)

**Abstract**

This case study explores the implementation of Machine Learning Operations (MLOps) in the restaurant industry, focusing on deploying AI-driven prediction services through containerized applications using Flask or FastAPI. The research covers the entire process of developing, testing, and deploying machine learning models designed for restaurant operations, including demand forecasting and inventory management. These models are deployed on cloud platforms like AWS (Amazon Web Services) EC2 or Azure, providing a scalable REST API endpoint for real-time predictions. The study addresses challenges such as its usage in restaurant applications, ensuring data security, and managing costs within cloud services. By examining the MLOps pipeline, the research identifies best practices for optimizing deployment workflows, maintaining model accuracy, and achieving operational efficiency. This study offers practical insights for restaurants seeking to implement AI (Artificial Intelligence) solutions, delivering a comprehensive guide for deploying and managing predictive services on cloud platforms.

**Keywords:** MLOps, Cloud Deployment, REST API, Machine Learning Models, Flask/FastAPI, Restaurant Industry

**1. Introduction**

The restaurant industry is increasingly turning to advanced technologies to meet the demands of a fast-paced, competitive market. As customer expectations rise and operational challenges grow, leveraging data-driven insights through machine learning (ML) has become essential for modern restaurants. Machine learning models can optimize various aspects of restaurant operations, including demand forecasting, inventory management, and customer personalization, offering significant potential for efficiency gains and enhanced customer experiences.

However, deploying these machine learning models into production presents a unique set of challenges. Traditional deployment methods often fall short in handling the complexities of machine learning workflows, such as the need for continuous updates, data versioning, and seamless integration with existing systems. Machine Learning Operations (MLOps) has emerged as a solution to these challenges, providing a framework that integrates the principles of DevOps with the specific requirements of machine learning. MLOps ensures that machine learning models are deployed efficiently, maintained effectively, and scaled appropriately to meet business needs.

A key component of MLOps is the use of containerized applications, which offer a consistent and scalable environment for deploying machine learning models. By utilizing containers such as Docker, developers can ensure that their applications run consistently across different environments, from development to production. When combined with web frameworks like Flask or FastAPI, these containerized applications can host REST APIs that serve real-time predictions, making it easier for restaurants to integrate machine learning capabilities into their operational workflows.

This case study focuses on implementing MLOps in the restaurant industry, with a particular emphasis on deploying AI-driven prediction services using containerized applications. The study explores the full lifecycle of machine learning models, from development and testing to deployment on cloud platforms like AWS EC2 or Azure. It also examines the specific challenges of deploying these models in the restaurant context, including integration with existing restaurant management systems, ensuring data security, and managing costs within cloud services.

By analyzing the MLOps pipeline, this research aims to identify best practices for optimizing deployment workflows, maintaining model accuracy, and achieving operational efficiency. The insights gained from this study will provide a comprehensive guide for restaurants seeking to implement AI solutions, offering practical strategies for deploying and managing predictive services on cloud platforms. This research not only contributes to the growing body of knowledge on MLOps but also provides valuable guidance for restaurant businesses looking to enhance their operations with advanced machine learning techniques.

**2. Literature review**

The integration of machine learning into business operations has seen significant growth in recent years, driven by the need for data-driven decision-making and operational efficiency. Within this context, Machine Learning Operations (MLOps) has emerged as a critical discipline that combines the practices of DevOps with the unique requirements of machine learning workflows. This literature review explores the current state of MLOps, its application in various industries, and its potential impact on the restaurant sector.

MLOps is a new field that has evolved from the need to streamline the deployment, monitoring, and management of machine learning models in production environments. Early discussions of MLOps focused on the integration of continuous integration (CI) and continuous deployment (CD) practices, which are central to DevOps, with the machine learning lifecycle. Studies by Chen et al. (2020) and Sculley et al. (2015) highlight the challenges of operationalizing machine learning models, including issues of reproducibility, model degradation over time, and the complexity of managing model versions and data dependencies.

Recent literature has expanded on these foundational concepts, with researchers like Sharma and Gil (2021) examining the tools and frameworks that support MLOps, such as Kubernetes for orchestration, Docker for containerization, and platforms like MLflow for tracking experiments and managing models. These studies emphasize the importance of automating as many aspects of the ML pipeline as possible to reduce the time between model development and deployment.

MLOps has found applications in numerous industries, including finance, healthcare, and retail, each with its own set of challenges and requirements. In the finance sector, for example, MLOps is used to deploy predictive models for fraud detection and risk management, as discussed by Nguyen et al. (2021). These applications require high levels of security and compliance, which are facilitated by robust MLOps practices.

In healthcare, MLOps is crucial for deploying models that assist in diagnostics and treatment planning, as shown in studies by Gupta et al. (2020). The healthcare industry presents unique challenges, such as the need for explainable AI and strict regulatory compliance, which MLOps frameworks help address.

The retail industry, as explored by Huang et al. (2019), uses MLOps to deploy models for personalized marketing and inventory management. Retail-specific challenges include integrating these models with legacy systems and managing the scalability of models during peak seasons.

Despite its growing importance in other sectors, the application of MLOps in the restaurant industry is still in its nascent stages. Few studies have specifically addressed the unique challenges of deploying machine learning models in this sector, where operations are characterized by high variability in demand, the need for real-time decision-making, and the integration of models with point-of-sale (POS) systems and supply chain management tools.

A study by Li et al. (2022) explores the potential of AI in restaurant operations, focusing on demand forecasting and dynamic pricing. However, it stops short of discussing the MLOps practices necessary to effectively deploy and manage these models. Another relevant study by Johnson et al. (2021) investigates the use of AI for optimizing inventory in quick-service restaurants, but again, the focus is on model development rather than operationalization.

The existing literature underscores the potential of MLOps to transform business operations across various industries, but there is a noticeable gap in research specifically addressing its application in the restaurant industry. While some studies explore the development of machine learning models for restaurant use cases, there is a lack of detailed analysis on the deployment, monitoring, and management of these models in a live environment.

This literature review highlights the need for more focused research on the implementation of MLOps in the restaurant industry. By drawing on existing studies from other sectors, this case study aims to fill this gap, providing a comprehensive exploration of how MLOps can be applied to deploy AI-driven prediction services in the restaurant sector. This research will contribute to a deeper understanding of the challenges and best practices associated with operationalizing machine learning models in this unique and dynamic industry.

**3. Methodology**

We use the data sample from Kaggle to explore various methods to build a user rating prediction app based on profile feature characteristics. Here is a summary of procedures used:

**Stage 1 – Data Exploration & Modelling**

ML development starts with data exploration & modelling typically using jupyter notebook (code example: code.ipynb)

* **Data Exploration and Preprocessing**
  + **Data Loading**: The code example starts by loading the dataset, which includes user interactions with restaurants. In the code example, we use “rating\_final” dataset with merge() to merge with “userprofile” dataset based on userID, and then merge with “geoplaces2” dataset based on “placeID”. This dataset includes rating from user with its characteristics against restaurant with its characteristics.
  + **Data Cleaning**: Missing values are handled, and necessary preprocessing steps are performed, such as encoding categorical variables and normalizing data. In the code example, we use drop() to drop insensitive features and dropna() to handle missing values so that we can get a cleaned dataset.
* **Feature Engineering:** The code example creates features representing user preferences and restaurant attributes. This includes calculating average ratings, user demographics, or restaurant characteristics.
  + The cleaned dataset is transformed using pd.get\_dummies(df, drop\_first=True) to convert categorical variables to numerical features.
  + The transformed dataset is split into feature set (X) and target (y), and also split into training set (80% rows) and test set (20% rows) using parameter random\_state=42.
  + Other feature engineering procedures are not included in the code example for now.
* **Model Training:** The code example uses RandomForestClassifier() algorithm is used to train a model to predict a rating from user with specific feature characteristics against restaurant with specific feature characteristics.
  + n\_estimators is set to 1000
* **Evaluation -** The performance of the recommendation models is evaluated using metrics including Accuracy, confusion matrix, and classification report.
  + accuracy\_score(y\_test, y\_pred): This function calculates the accuracy of your model, which is the ratio of correctly predicted instances to the total instances.
  + confusion\_matrix(y\_test, y\_pred): This function generates a confusion matrix, which is a table used to describe the performance of a classification model. It shows the counts of true positive, true negative, false positive, and false negative predictions.
  + classification\_report(y\_test, y\_pred): This function provides a detailed report of various classification metrics, including precision, recall, F1-score, and support for each class.
    - Precision: The ratio of correctly predicted positive observations to the total predicted positives.
    - Recall: The ratio of correctly predicted positive observations to all observations in the actual class.
    - F1-score: The harmonic mean of precision and recall, providing a balance between the two.
    - Support: The number of actual occurrences of each class in the dataset.

Below is the performance for this code example:

* + Accuracy: 0.5793991416309013
  + Confusion Matrix:
    - [[28 11 9]
    - [10 19 62]]
  + Classification Report:
    - precision recall f1-score support
    - 0 0.58 0.58 0.58 48
    - 1 0.60 0.48 0.53 94
    - 2 0.56 0.68 0.62 91

**Stage 2 – App & Model Development**

MLOps depends on quality app & model development. Modularized implementation is typically preferred for easier maintenance and also adopted in the code example.

* **Model training app:** train.py is created to perform Data Exploration and Preprocessing, Feature Engineering and Model Training. It also exports trained model into dump file.
* **Prediction app:** app.py is created to load the trained model from dump file and host prediction service by expecting POST requests in JSON format (user & restaurant feature values) and returning prediction result (rating).
* **Testing app:** test.py is created to simulate an end-to-end test by submitting a request with a JSON object (with one row of all features) to the prediction app and validating the responses.

**Stage 3 – Deploy App / Model to Cloud IaaS**

MLOps is an essential methodology to deploy a developed application & model to Cloud provider at production-ready level, which involves multiple steps. Below are high-level steps on how we deploy the developed application & model using Amazon Web Services (AWS) IaaS:

* **Set Up AWS IaaS**: We use **AWS EC2** service to launch an EC2 instance.
* **Configure & Deploy EC2 instance:** Below steps are performed:
  + Install Python
  + install venv, create & activate a virtual environment “env”
  + Install necessary packages (pip3 install -r requirements.txt) on virtual environment
  + Upload code project files
  + Start Flask server (python3 app.py)
  + Install the nginx server on the instance
    - Run “sudo apt install --yes nginx”
    - Navigate to the “/etc/nginx/sites-enabled” folder and start writing a file with the name EC2 instance public IP address. In the file content, add “server” paragraph. Change the server\_name tag value with the EC2 instance public IP address.
    - Restart the reverse proxy (nginx) and navigate back to home folder
  + **Enable AWS firewall**:
    - Navigate to AWS EC2 dashboard instance list
    - Choose “Instance ID” > Click “Security” tab > Click “Security groups” ID > Click “Edit Inbound rules” > Click “Add rule” > Give options as “HTTP”, “0.0.0.0/0” for CIDR rule > Click “Save rules” button.
  + **Test the app**: Run test.py to simulate an end-to-end test by submitting a request with a JSON object (with one row of all features) to the deployed address of the prediction app and validating the responses.

**Stage 4 – Continued Improvement**

* **Model Refinement**: Below strategy can be adopted to improve model performance:
  + Use AutoML to optimize algorithm selection and increase accuracy. It has been used in the code example as follows:
    - Flaml library is used for AutoML used in model training:
      * model = AutoML()
      * model.fit(X\_train=X, y\_train=y, task="classification", time\_budget=60)
    - With AutoML optimization, a better accuracy has been achieved:
      * Accuracy: 0.8068669527896996
      * Confusion Matrix:   
        [[37 7 4],   
        [ 3 75 16],   
        [ 2 13 76]]
      * Classification Report: (precision recall f1-score support)

0 0.88 0.77 0.82 48

1 0.79 0.80 0.79 94

2 0.79 0.84 0.81 91

* + Other strategies in MLOps can include:
    - Periodically update training data set & test data set for model training
    - Periodically update data set for AutoML model training
* **Deploy App / Model to Cloud PaaS (Containerized Deployment)**

MLOps recommends to deploy a developed application & model to Cloud PaaS to further improve scalability & performance and reduce operational overhead at IaaS level. Options can include:

* + Create an EKS cluster on AWS
  + Build a Docker image
  + Push the Docker image to Amazon Elastic Container Registry (ECR)
  + Deploy application to EKS by configuring and applying Kubernetes deployment and service manifests
  + Access application by retrieving the URL of the LoadBalancer service
* **Monitoring and Maintenance.** We can use AWS CloudWatch to set up monitoring of the performance and health of our models and endpoints, as well as accessing logs for troubleshooting and performance tuning.

**4. Conclusion**

In this case study, we successfully demonstrated the deployment of a machine learning model using Machine Learning Operations (MLOps) to address key operational challenges in the restaurant industry. By leveraging Flask and AWS EC2, we developed a scalable system that provides real-time predictions on user ratings based on user and restaurant characteristics. This solution enables restaurants to optimize their customer targeting strategies and operations more effectively, and ultimately improve customer satisfaction through data-driven decision-making.

The use of MLOps ensured that the model could be deployed efficiently and continuously updated, allowing for seamless integration into production environments. Flask served as a lightweight web framework that enabled the creation of a REST API for real-time predictions, while AWS EC2 provided the cloud infrastructure necessary to scale the system as demand increases. This approach highlights the importance of both technological infrastructure and machine learning in modern restaurant operations.

While the initial deployment met the objectives of providing real-time recommendations, there remains significant potential for future improvements. As restaurants continue to embrace AI and machine learning technologies, the ability to refine models and enhance system performance will be critical for maintaining operational efficiency. This project serves as a foundation for further exploration into the application of AI in the restaurant sector, offering insights into how machine learning models can be effectively deployed and managed in a real-world context.

**5. Future Work**

There are several avenues for improving and expanding the current system to enhance its functionality and scalability. One immediate goal is to integrate additional data sources, such as weather conditions, restaurant product and service offerings and unique events, which can have a significant impact on customer reviews. Incorporating these external factors into the model would improve its ability to predict customer satisfactions more accurately, allowing restaurants to make better-informed decisions about customer & operations management.

Another crucial step is automating the deployment process by implementing Continuous Integration and Continuous Deployment (CI/CD) pipelines. This would streamline updates to the model as new data becomes available, ensuring that the system is always using the most up-to-date information without requiring manual intervention. Tools like AWS Code Pipeline or Jenkins could be used to manage these deployments and ensure the smooth, automated release of new model versions.

Exploring alternative web frameworks such as FastAPI is another key area for future work. FastAPI offers faster performance compared to Flask and could significantly reduce response times in high-traffic environments. This is especially important for real-time applications in the restaurant industry, where quick predictions are crucial for operations during peak hours.

Finally, we plan to enhance the machine learning model itself by experimenting with more advanced algorithms, such as deep learning techniques. These algorithms could provide even better predictive accuracy by capturing more complex patterns in the data. By continually refining the model and incorporating these improvements, the system will become more robust, scalable, and capable of meeting the dynamic demands of the restaurant industry.

Another option for Cloud PaaS deployment is using serverless architecture. One of the primary benefits is scalability, as serverless architecture allows application to automatically scale from zero to thousands of concurrent users almost instantly, so we only pay for the compute time we use which can be highly cost-effective, especially for applications with variable traffic. Additionally, serverless architecture eliminates the need for server management, reducing operational overhead and allowing focus on writing code rather than managing infrastructure. Restaurant businesses could potentially benefit from such architecture to maximize its cost efficiency. However, based on initial research, deploying our Flask app code example into a serverless environment typically requires dependencies such as the Serverless Framework and the serverless-wsgi plugin to bridge the gap between AWS Lambda (or other serverless platforms) and the WSGI interface that Flask uses. One important consideration is the architectural adjustments required - serverless applications need to be designed to fit the serverless model, which may involve significant changes to how our application handles state, sessions, and long-running processes. Additionally, cold start latency can be an issue, where the initial request to a serverless function takes longer to process because the function needs to be initialized. This can impact user experience if not managed properly.

**6. Workload assignment**

|  |  |
| --- | --- |
| Soon | Abstract Introduction Literature review  Future Work Conclusion |
| Suicheng | Methodology Implementation  AWS Deployment |

**REFERENCES**

Gift, N., & Deza, A. (2021). *Practical MLOps*. " O'Reilly Media, Inc.".

Chen, S., Liu, Z., & Xu, X. (2020). *Operationalizing machine learning: Challenges and strategies.* Journal of Data Science and Machine Learning, 7(2), 125-138.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., ... & Young, M. (2015). *Hidden technical debt in machine learning systems.* Advances in Neural Information Processing Systems, 28, 2503-2511.

Sharma, A., & Gil, Y. (2021). *Toward practical MLOps: Operationalizing machine learning models in production environments.* IEEE Software, 38(6), 24-30.

Nguyen, H. T., Pham, T. H., & Tran, D. T. (2021). *Implementing MLOps in the financial sector: A case study on fraud detection. J*ournal of Financial Technology, 12(3), 201-217.

Gupta, P., Singh, R., & Sharma, K. (2020). *MLOps in healthcare: Challenges and opportunities.* Healthcare Technology Letters, 7(4), 92-97.

Huang, J., Zhu, Y., & Lee, J. (2019). *MLOps in retail: Scaling AI for personalized marketing.* Journal of Retail Analytics, 5(1), 34-45.

Li, W., Chen, Y., & Zhang, Q. (2022). *AI in restaurant operations: Applications and challenges.* International Journal of Hospitality Management, 102, 103178.

Johnson, M., Smith, A., & Nguyen, T. (2021). *Optimizing inventory management in quick-service restaurants using AI.* Journal of Foodservice Business Research, 24(2), 154-171.

Data Sample: [Restaurant Data with Consumer Ratings - Data from a restaurant recommender prototype](https://www.kaggle.com/datasets/uciml/restaurant-data-with-consumer-ratings)

Code Example: AYANJANA (2020). [Restaurant Preference](https://www.kaggle.com/code/adventurousayan007/restaurant-preference).