REPORT ON INDUSTRY INTERACTION ACTIVITIES CONDUCTED UNDER IEEE-CIS (Bangalore Section)

IEEE Computational Intelligence Chapter Bangalore Section (R-10), Bangalore

INDUSTRY INTERACTION ACTIVITIES 2024-25

Activity 1 : (Partial Delivery of Courses)
End-to-End Machine Learning Workflow using MLOps: From Experimentation to Deployment

1. Introduction

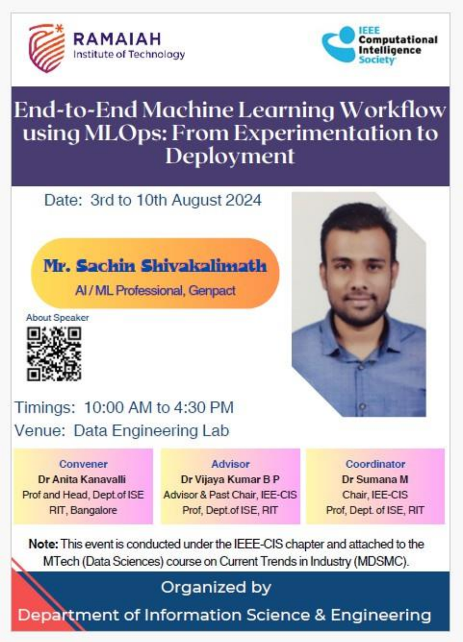
1.1 Objective

The two-day workshop aimed to provide participants with practical skills in machine learning (ML) experimentation, artifact management, and deployment strategies using various tools and platforms. The focus was on hands-on experience with MLflow for tracking experiments, deploying models locally and as APIs, and implementing CI/CD automation using GitHub Actions, Docker, and related technologies.

1.2 Overview

The workshop was conducted over two days with the following focus:

- **Day 1 (August 3rd):** Machine Learning Lifecycle, including classification problems, data processing, model building, and local deployment.
- **Day 2 (August 10th):** CI/CD pipeline setup, version control, and running ML applications in Docker, with a focus on CI/CD automation and deployment strategies.



RAMAIAH
Institute of Technology

IEEE
Computational
Intelligence
Society

End-to-End Machine Learning Workflow using MLOps: From Experimentation to Deployment

Date: 3rd to 10th August 2024

Mr. Sachin Shivakalimath
AI / ML Professional, Genpact

About Speaker

Timings: 10:00 AM to 4:30 PM
Venue: Data Engineering Lab

Convener	Advisor	Coordinator
Dr Anita Kanavalli Prof and Head, Dept.of ISE RIT, Bangalore	Dr Vijaya Kumar B P Advisor & Past Chair, IEE-CIS Prof, Dept.of ISE, RIT	Dr Sumana M Chair, IEE-CIS Prof, Dept. of ISE, RIT

Note: This event is conducted under the IEEE-CIS chapter and attached to the MTech (Data Sciences) course on Current Trends in Industry (MDSMC).

Organized by
Department of Information Science & Engineering

2. Day 1: Machine Learning Lifecycle and Local Deployment

2.1 Classification Problems

The workshop began with an introduction to classification problems:

- **Binary Classification:** Problems with binary outputs, such as spam detection.
- **Multi-class Classification:** Scenarios involving multiple classes, such as fruit classification.

2.2 Data Processing Pipeline

Participants learned about the data processing pipeline essential for preparing data:

- **Data Collection:** Gathering data from various sources.
- **Feature Extraction:** Selecting and engineering relevant features.
- **Data Normalization:** Scaling features for consistency.

2.3 Model Building

The model-building process was covered in detail:

- **Data Splitting:** Dividing data into training and testing sets.
- **Model Training:** Using algorithms to learn patterns from training data.
- **Model Validation:** Evaluating performance on unseen data.
- **Model Deployment:** Integrating models into production.

2.4 ML Experimentation with MLflow

Participants were introduced to MLflow:

- **Tracking:** Logging parameters, metrics, and results.

- **Artifacts:** Managing models, datasets, and files.

2.5 Local Deployment

Deployment methods were covered:

- **Streamlit:** Building and sharing data apps.
- **Flask:** Deploying models as web services.
- **.EXE Deployment:** Converting Python scripts into executable files for Windows systems.

2.6 Exposing the ML Model as an API

The process of exposing ML models as APIs was discussed:

- **API Development:** Creating RESTful APIs using Flask.
- **Integration:** Integrating APIs into software systems.

2.7 Logging and Exception Handling

Participants learned about logging and handling exceptions:

- **Logging:** Implementing logging for monitoring and debugging.
- **Exception Handling:** Strategies for error management in production.

2.8 Unit Testing and Precommit

Instructions were given on:

- **Unit Testing:** Writing tests to validate model correctness.
- **Precommit Hooks:** Setting up hooks to ensure code quality before commits.

3. Day 2: CI/CD Pipeline, Version Control, and Docker

3.1 CI/CD Pipeline Overview

Day 2 focused on CI/CD pipelines:

- **Source Code Management:** Using Git for version control.
- **Continuous Integration:** Automating code integration and testing.
- **Continuous Deployment:** Automating code deployment to production.

3.2 Setting Up Version Control with Git

Participants learned essential Git operations:

- **Initializing Repositories:** Creating and linking repositories.
- **Basic Commands:** Adding, committing, and pushing changes.
- **Branching and Merging:** Best practices for managing branches.

3.3 Configuring the CI/CD Pipeline

The CI/CD pipeline configuration was covered:

- **Build and Test Automation:** Writing scripts for builds and tests.
- **Deployment Automation:** Setting up deployment scripts and managing secrets.
- **Monitoring and Logging:** Implementing mechanisms for performance and issue tracking.

3.4 Working with Virtual Environments

Participants learned to manage virtual environments:

- **Creating Environments:** Using Python's venv module.
- **Managing Dependencies:** Installing and managing dependencies.

3.5 GitHub Actions for CI/CD

GitHub Actions for CI/CD automation were introduced:

- **YAML Configuration:** Writing YAML files for defining workflows.
- **Pipeline Setup:** Automating tests, builds, and deployments.

3.6 Docker for ML Deployment

Docker was covered for containerization:

- **Docker Desktop:** Installation and local development.
- **Docker Hub:** Pushing Docker images for distribution.

3.7 Running the ML App in Docker

Participants learned to containerize and deploy ML applications:

- **Containerization:** Creating Docker containers for ML apps.
- **Deployment:** Running containers in different environments.

4. Best Practices and Conclusion

4.1 Best Practices

Key best practices emphasized included:

- **Commit Messages:** Writing clear, descriptive commit messages.
- **Repository Management:** Regularly merging changes to avoid conflicts.
- **CI/CD Pipeline Maintenance:** Keeping pipelines updated and monitored.
- **Experiment Tracking:** Logging experiments for reproducibility.
- **Dockerization:** Containerizing applications for consistency.
- **Version Control:** Maintaining clean version control practices.

4.2 Conclusion

The workshop successfully equipped participants with comprehensive skills in managing the ML lifecycle, from experimentation to deployment. Attendees gained practical experience in deploying models locally and using containers, automating CI/CD pipelines, and ensuring robust development practices. The importance of continuous learning and staying updated with evolving tools and technologies in ML and DevOps was emphasized, concluding with discussions on the value of these skills in professional roles.

Links to photos and videos :

https://drive.google.com/drive/folders/1yxJ0kKK0Z3G1UFNhXuz7IySl2aKR_g6_?usp=drive_link
https://drive.google.com/drive/folders/1y3GMwHzOsHgtDBmERswsRnsihp9IuGRe?usp=drive_link

Link to the materials: https://drive.google.com/drive/folders/1fY1-VwZIWV -vCPvcYWrv3xi48wah2q?usp=drive_link

Session 3 : 27/08/2024

MLOps Workshop - Class 3 Summary Report

Overview

This report summarizes the key points and steps covered in the MLOps workshop, focusing on building and managing machine learning pipelines using Docker, Streamlit, FastAPI, and Evidently. The workshop emphasizes best practices in model deployment, data drift detection, and continuous integration/continuous deployment (CI/CD).

1. Docker Image Creation and Continuous Integration

- **Docker Image**:
  - A Docker image is built as part of the workflow.
  - This image encapsulates the model and its dependencies, ensuring consistency across environments.
- **CI/CD Process**:
  - A Continuous Integration (CI) pipeline is configured to automate the building and deployment of the Docker image.
  - The process includes steps to validate and score the model, followed by manual approval before deployment.
- **Approval Workflow**:
  - After generating predictions (y_hat values), the results are subject to an approval process.
  - Monitoring for data drift is a key part of this process to ensure model accuracy over time.

2. Data Preprocessing and Machine Learning Workflow

- **Data Handling**:
  - Preprocessing steps include standardization for numeric data and dummy variable creation for categorical data.
  - Response variables are label-encoded, and the best model from experiments is saved as a .pkl file.
- **Model Training**:

    - Models are trained with validation splits, and the best performing model is identified and serialized for use in production.
- **Drift Monitoring**:
    - An Evidently report is generated to monitor data drift, comparing new data against reference data.
    - The appearance of new categories in categorical columns is discussed, with clarification that this should not always be classified as drift.

3. Evidently Report Generation

- **Generating Evidently Report**:
    - Steps to generate the report include installing the evidently package and running a Python script (generate_evidently_report.py) from the appropriate directory.
    - The report highlights how much the data has drifted from the reference set.
- **Decision on Drift**:
    - The report creation is followed by a manual review to determine if any drift detected requires approval or further action.
- **Software Dependencies**:
    - Evidently version 0.4.36 is used, and it should be listed in the requirements.txt.
    - Integration with FastAPI and Docker is set up, where Streamlit also depends on FastAPI for handling user requests.

4. Local and Batch Execution

- **Local Execution**:
    - For running the app locally, the command streamlit run app.py is used.
    - Batch predictions can be run using uvicorn batch_api_app --host.
- **Version Control and CI/CD**:
    - git status and git commit commands are used for version control.
    - Changes are made in the CI/CD pipeline configuration (ci.yml) to ensure smooth deployment.
- **Docker Hub Integration**:
    - A personal access token (PAT) is generated in Docker Hub for secure access, which is added to repository secrets for automation.
- **Drift Testing**:
    - The workshop encourages intentionally creating data drift scenarios to test the system's ability to detect and respond to such changes.
- **Container Management**:
    - Instructions are provided for managing Docker images and containers, including stopping, deleting, and reviewing logs.

5. Key Takeaways

- **Comprehensive MLOps Pipeline**: The workshop guides through the setup of a full MLOps pipeline, emphasizing automation, monitoring, and security.
- **Data Drift Management**: Continuous monitoring for data drift is critical to maintain model accuracy over time.

- **Integration with Docker and CI/CD**: The use of Docker, FastAPI, and Streamlit, combined with CI/CD practices, ensures that models are deployed consistently and efficiently.

This report encapsulates the workshop's key lessons and steps, providing a practical guide for setting up and managing MLOps pipelines in real-world scenarios.

Link to videos and photos:
https://drive.google.com/drive/folders/16HAfTA8BU75j6yGkZugkHlYurjZOwwpj?usp=drive_link