MLOps Course Project

NLP Pipeline with Text Classification

1 Project Overview

In this project, students will build an end-to-end MLOps pipeline for text classification using a publicly available NLP dataset. They will implement dataset versioning, model experimentation, and deployment of a text classification service that can be run without specialized hardware requirements.

2 Learning Objectives

- Implement NLP preprocessing and feature engineering techniques
- Master data pipeline orchestration using Airflow
- Track NLP experiments with MLflow
- Deploy an NLP model via REST API
- Monitor model performance in production

3 Dataset

News Category Dataset - A collection of news headlines categorized by topic (business, technology, entertainment, etc.)
`https://www.kaggle.com/datasets/rmisra/news-category-dataset`

4 Project Requirements

4.1 Data Pipeline Orchestration with Airflow

- Set up Apache Airflow for orchestrating data processing pipelines
- Create DAGs (Directed Acyclic Graphs) for:
  - Data ingestion from the news dataset source
  - Data preprocessing with multiple stages:
    - Basic cleaning (lowercase, remove punctuation)
    - Advanced processing (stopword removal, lemmatization)
    - Feature engineering (text length, sentiment scores)
  - Data splitting (train/validation/test) with consistent random seeds
- Implement proper error handling and retry mechanisms
- Create sensors to monitor the completion of upstream tasks
- Schedule periodic pipeline runs
- Document all Airflow operators and data transformations

4.2 NLP Model Development with MLflow

- Experiment with at least 4 different approaches:
  - Traditional ML with TF-IDF features (e.g., Naive Bayes, SVM)
  - Word embeddings with simple neural networks
  - Pre-trained embeddings (Word2Vec, GloVe)
  - Simple transformer-based approaches (e.g., DistilBERT with limited layers)
- Track all experiments in MLflow with:
  - Model hyperparameters
  - Performance metrics (accuracy, F1-score, confusion matrix)
  - Training and inference time
  - Dataset version used
- Register the best performing models to MLflow Model Registry
- Create visualizations comparing model performance

4.3 Model Serving API (FastAPI)

- Develop a REST API that:
  - Loads the best model from MLflow
  - Provides endpoints for:
    - Single text classification
    - Batch prediction
    - Model information (version, metrics)
  - Includes proper input validation
  - Returns prediction probabilities for top categories
- Document API with Swagger/OpenAPI
- Implement request logging and error handling
- Create a simple web demo for testing the API

4.4 Monitoring System

- Instrument the API with Prometheus metrics:
  - Request count and latency
  - Prediction distribution across categories
  - Input text characteristics (length, language detection)
  - Error rates
- Create Grafana dashboards to visualize:
  - API performance
  - Prediction distributions
  - Data drift indicators
- Implement basic alerting for unusual patterns

5 Deliverables

1. GitHub repository with complete code and documentation
2. Airflow DAGs for data processing pipelines
3. MLflow tracking logs and registered models
4. Working FastAPI application for serving predictions
5. Prometheus + Grafana monitoring setup
6. Project report explaining approach, challenges, and results

6 Evaluation Criteria

- Implementation of NLP techniques (20%)
- Airflow implementation for pipeline orchestration (20%)
- MLflow usage for experiment tracking (20%)
- API functionality and documentation (20%)
- Monitoring implementation (10%)
- Code quality and project documentation (10%)

6.1 Bonus Marks

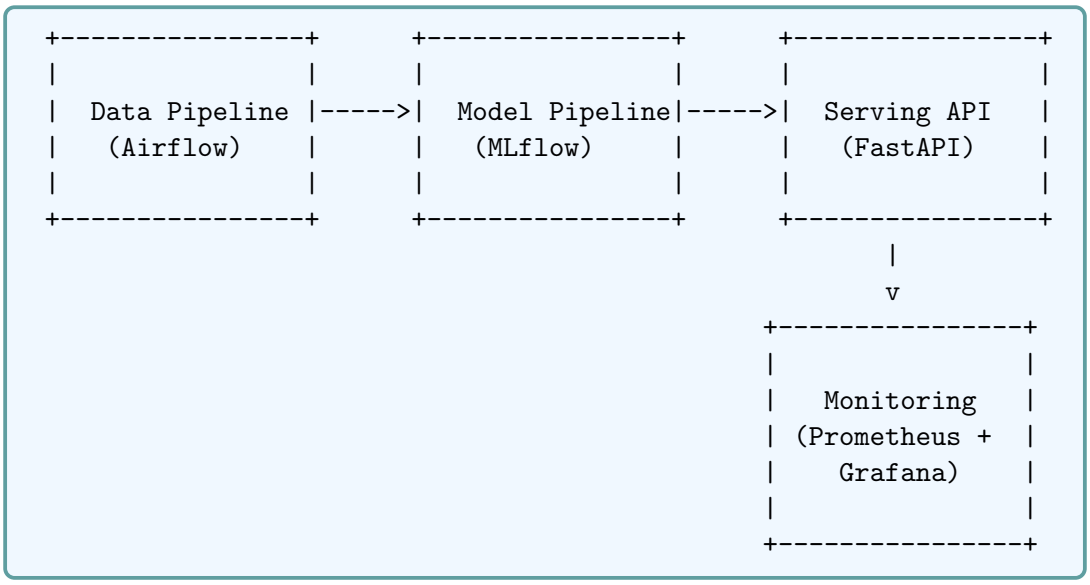
Bonus Opportunities (Up to 15% Extra)

Students can earn bonus marks for implementing the following advanced concepts:

- **GitHub Actions (7.5%):** Implement CI/CD pipelines using GitHub Actions to:
  - Automate testing of code changes
  - Validate data preprocessing steps
  - Run model performance checks
  - Automate deployment process
- **Containerization with Docker (7.5%):** Create Docker containers for:
  - Training environment
  - Model serving API
  - Monitoring stack
  - Complete solution with Docker Compose

7 Project Architecture

The project should follow the architecture depicted below:



8 Conclusion

This NLP-focused project provides students with practical experience building an MLOps pipeline for text classification. The dataset is readily available, the computational requirements are manageable without specialized hardware, and the project incorporates all the core MLOps components: data versioning, experiment tracking, model serving, and monitoring.

Best of Luck!

This project will challenge you to apply MLOps concepts in a real-world scenario. Remember that the journey of building this pipeline is as valuable as the final result. Embrace the challenges, learn from errors, and don't hesitate to collaborate and seek help when needed. The skills you develop during this project will be directly applicable to industry positions in ML engineering. We're excited to see your innovative solutions and creative approaches!

9 Resources and References

- **Airflow Documentation:** `https://airflow.apache.org/docs/`
- **MLflow Documentation:** `https://mlflow.org/docs/latest/index.html`
- **FastAPI Documentation:** `https://fastapi.tiangolo.com/`
- **Prometheus:** `https://prometheus.io/docs/introduction/overview/`
- **Grafana:** `https://grafana.com/docs/`
- **NLTK:** `https://www.nltk.org/`
- **spaCy:** `https://spacy.io/`
- **Hugging Face Transformers:** `https://huggingface.co/docs/transformers/index`