S1-24_AIMLCZG523 – MLOps - Assignment 1

Group No. 120

Name	*StudentID*	*Contribution*
JOSHI NIRANJAN SURYAKANT	**2023AC05011**	**100%**
PRATEEK RALHAN	**2023AC05673**	**100%**
KESHARKAR SURAJ SANJAY	**2023AD05004**	**100%**
SAURABH SUNIT JOTSHI	**2023AC05565**	**100%**

MLOps Architecture Summary - California Housing Price Prediction Pipeline

Project Overview

End-to-end MLOps pipeline for California Housing Price Prediction using **Linear Regression and Decision Tree models, implementing industry best practices for model development, deployment, and monitoring**.

Architecture Components

1. Data Management & Versioning
    a. *Dataset:* California Housing dataset (housing.csv) with 8 features (MedInc, HouseAge, AveRooms, AveBedrms, Population, AveOccup, Latitude, Longitude)
    b. *Data Versioning:* DVC (Data Version Control) for tracking dataset changes
    c. *Data Pipeline:* Automated data loading via `src/utils.py` using scikit-learn's fetch_california_housing
    d. *Storage:* Local data directory with DVC tracking files

2. Model Development & Training
    a. *Algorithms*: Multiple model comparison (Linear Regression, Decision Tree, Ridge Regression)
    b. *Configuration*: YAML-based config (`src/config.yaml`) with hyperparameters (alpha, test_size, random_state, max_depth)
    c. *Training Pipeline*: `src/train.py` with automated model selection based on MSE
    d. *Model Storage*: Joblib serialization to `models/ridge_model.pkl`
    e. *Code Quality*: Pre-commit hooks (Black, Flake8, Prettier) for code formatting and linting

3. Experiment Tracking & Model Registry
    a. *Platform*: MLflow for experiment tracking and model versioning
    b. *Tracking*: Automatic logging of parameters, metrics (MSE), and model artifacts
    c. *Model Registry*: Registered models with versioning (`BestHousingModel`)
    d. *Storage*: SQLite backend (`mlflow_data/mlflow.db`) and local artifact storage (`mlruns/`)
    e. *UI*: MLflow tracking server on port 5555

4. API development & Deployment
    a. *Framework*: FastAPI with automatic OpenAPI documentation
    b. *Endpoints*:
        i. `/predict` - Housing price predictions
        ii. `/metrics` - Prometheus metrics exposure

iii. `/retrain` - Model retraining trigger

c. *Input Validation*: Pydantic models for request validation
d. *Containerization*: Docker with multi-stage build process
e. *Docker Hub*: Published image (`niranjanjoshi14/housing-api:latest`)

5. Monitoring & Observability
   a. *Metrics Collection*: Prometheus client integration with custom metrics
      - `prediction_requests_total` - Request counter
      - `prediction_latency_seconds` - Response time histogram
   b. *Database Logging:* SQLite database (`prediction_logs.db`) for prediction history
   c. *Monitoring Stack:*
      - Prometheus (port 9090) for metrics scraping
      - Grafana (port 3000) for visualization dashboards
   d. *Logging:* Structured logging with timestamps and input/output data

6. CI/CD Pipelines
   a. *Platform*: GitHub Actions with automated workflows
   b. *Triggers*: Push to main branch, pull requests, data changes
   c. *Pipeline Stages*:
      1. Code checkout and Python setup
      2. Dependency installation and caching
      3. Code quality checks (Flake8, Black)
      4. Docker image building and publishing
      5. Model training validation
      6. Automated testing
   d. *Security*: Docker Hub authentication via secrets

7. Infrastructure & Orchestration
   a. *Container Orchestration*: Docker Compose with 4 services
      - API service (FastAPI)
      - Prometheus (metrics collection)
      - Grafana (visualization)
      - MLflow (experiment tracking)
   b. *Networking*: Internal service communication with external port mapping
   c. *Volumes*: Persistent storage for Grafana and MLflow data
   d. *Environment*: Python 3.10 with all dependencies in requirements.txt

8. Data Pipeline Automation
   a. *DVC Pipeline*: Automated training pipeline (`dvc.yaml`)
   b. *Dependencies*: Clear dependency tracking between data, code, and models
   c. *Reproducibility*: Deterministic training with fixed random seeds
   d. *Artifact Management*: Automatic model artifact generation and tracking

Key Metrics & KPIs

- *Model Performance:* MSE (Mean Squared Error) tracking
- *API Performance:* Request latency, throughput, error rates
- *System Health:* Container status, resource utilization
- *Data Quality:* Input validation success rates
- *Deployment Success:* CI/CD pipeline success rates

This architecture demonstrates a **production-ready MLOps pipeline with comprehensive monitoring, automated deployment, and scalable infrastructure following industry best practices.**

Appendix:

- Github repository : link
- Video demo: link
- Published docker image: link
- System Architecture (mermaid diagram): link