

9. Documentation & Reporting

A comprehensive and professional project report has been prepared to document the complete MLOps lifecycle, covering setup, experimentation, deployment, and monitoring. The report ensures clarity, reproducibility, and alignment with industry-standard MLOps practices.

9.1 Setup & Installation Instructions

The project is designed to be easily reproducible on any machine using Python and Docker.

Prerequisites

- Python \geq 3.9
- Docker & Docker Compose
- Git

Local Setup Steps

```
git clone https://github.com/<username>/heart-disease-mlops.git
```

```
cd heart-disease-mlops
```

```
pip install -r requirements.txt
```

Run Training Pipeline

```
python src/train.py
```

Run Inference API

```
uvicorn app.main:app --reload
```

Docker-Based Execution

```
docker build -t heart-mlops .
```

```
docker run -p 8000:8000 heart-mlops
```

These steps ensure consistent environment setup and eliminate dependency conflicts.

9.2 EDA and Modelling Choices

Exploratory Data Analysis (EDA)

EDA was performed using a Jupyter Notebook to understand:

- Feature distributions
- Class imbalance
- Correlation between features
- Missing values and outliers

Key observations:

- Dataset contains mixed numerical and categorical features.
- Target variable is binary (presence/absence of heart disease).
- Certain features show strong correlation with the target variable.

Modelling Decisions

- **Logistic Regression** selected as a baseline model due to interpretability.
- **Random Forest Classifier** used for capturing non-linear feature interactions.
- **ROC-AUC** chosen as the primary evaluation metric due to class imbalance sensitivity.

Feature scaling was applied using StandardScaler for models sensitive to feature magnitudes.

9.3 Experiment Tracking Summary

Experiment tracking was implemented using **MLflow** to ensure reproducibility and comparison.

Tracked artifacts include:

- Model hyperparameters
- Evaluation metrics (Accuracy, Precision, Recall, F1-score, ROC-AUC)
- Trained model files
- Scaler objects

Each experiment run is uniquely logged, enabling systematic comparison and informed model selection. The best-performing model is automatically selected based on ROC-AUC score.

9.4 Architecture Diagram

The project follows a modular, end-to-end MLOps architecture covering:

- Data ingestion
- Preprocessing and feature engineering
- Model training and selection
- Model serving and deployment
- Monitoring and testing

(Refer to Section 8 for the detailed architecture diagram and explanation.)

9.5 CI/CD and Deployment Workflow

The CI/CD pipeline automates model validation and deployment readiness.

CI Pipeline

- Code linting and formatting checks
- Unit testing using Pytest
- Model training validation

CD Pipeline

- Docker image creation
- Containerized model serving using FastAPI
- Kubernetes-ready deployment configuration

Screenshots of:

- GitHub Actions workflow
- Successful pipeline execution
- Running API service

are included in the report to demonstrate deployment readiness.

9.6 Code Repository Link

The complete source code, including notebooks, training scripts, Docker files, and CI/CD configuration, is hosted on GitHub:

 **Repository:**

<https://github.com/<username>/heart-disease-mlops>

The repository follows best practices for:

- Modular folder structure
 - Version control
 - Clear README documentation
-

9.7 Summary

This documentation provides a clear, professional, and reproducible account of the full MLOps pipeline. The report demonstrates not only correct implementation but also adherence to industry standards, making the solution production-ready and academically rigorous.