🏠 Local Deployment Guide - Step by Step

📋 Complete Guide to Run the Project Locally

This guide will walk you through **every step** to run the Heart Disease Prediction MLOps project on your local machine.

✅ Prerequisites Check

Before starting, ensure you have:

1. **Python Installation**

```
# Check Python version (should be 3.9 or higher)
python3 --version
```

Expected Output: `Python 3.9.x` or higher

If not installed:

- **macOS:** `brew install python@3.9`
- **Windows:** Download from python.org
- **Linux:** `sudo apt-get install python3.9`

---

2. **Git Installation** (Optional, for version control)

```
# Check Git
git --version
```

---

3. **Docker Installation** (For containerized deployment)

```
# Check Docker
docker --version
```

If not installed: Download from docker.com

🚀 Step 1: Navigate to Project Directory

```
```

```
# Open Terminal and navigate to the project
cd heart-disease-mlops

# Verify you're in the correct directory
pwd
# Should show: .../heart-disease-mlops

# List files to confirm
ls
# Should see: src/, data/, models/, requirements.txt, etc.
```

🔧 Step 2: Set Up Python Virtual Environment

A virtual environment keeps your project dependencies isolated from your system Python.

```
# Create virtual environment
python3 -m venv venv

# Activate virtual environment
# For macOS/Linux:
source venv/bin/activate

# For Windows:
# venv\Scripts\activate

# Your terminal should now show (venv) at the beginning
```

Verify activation:

```
which python
# Should show: .../heart-disease-mlops/venv/bin/python
```

📦 Step 3: Install Dependencies

```
# Upgrade pip first
pip install --upgrade pip

# Install all required packages
pip install -r requirements.txt

# This will install:
# - FastAPI (API framework)
# - scikit-learn (ML models)
# - pandas, numpy (data processing)
# - mlflow (experiment tracking)
# - uvicorn (ASGI server)
# - and more...
```

Verify installation:

```
pip list | grep -E "fastapi|scikit-learn|mlflow"
```

📊 Step 4: Download and Prepare Data

Option A: If dataset is already in raw_dataSet/

```
# Run the download script
python src/download_data.py
```

Expected Output:

```
================================================================================
======
Heart Disease Dataset Download Script
================================================================================
======
✓ Data downloaded successfully!
✓ Dataset saved to data/processed/heart_disease.csv
✓ Raw data copied to data/raw/
Dataset shape: (303, 14)
```

Option B: If dataset is NOT available

1. Download from: UCI Heart Disease Dataset
2. Extract to `raw_dataSet/heart+disease/` within the project directory
3. Run `python src/download_data.py`

🤖 Step 5: Train the Models

```
# Train all models with MLflow tracking
python src/train.py
```

What happens:

- Trains 3 models: Logistic Regression, Random Forest, Gradient Boosting

- Logs experiments to MLflow
- Saves best model to `models/best_model.pkl`
- Saves preprocessor to `models/preprocessor.pkl`

Expected Output:

```
INFO:__main__:Training Logistic Regression...
INFO:__main__:✓ Logistic Regression - Accuracy: 0.85, ROC-AUC: 0.90
INFO:__main__:Training Random Forest...
INFO:__main__:✓ Random Forest - Accuracy: 0.88, ROC-AUC: 0.92
INFO:__main__:Training Gradient Boosting...
INFO:__main__:✓ Gradient Boosting - Accuracy: 0.87, ROC-AUC: 0.91
INFO:__main__:✓ Best model saved to models/best_model.pkl
```

🎯 Step 6: Start the API Server

Method A: Using Uvicorn (Recommended for Development)

```
# Start the FastAPI server
uvicorn src.app:app --reload --host 0.0.0.0 --port 8000
```

Expected Output:

```
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     Started reloader process
INFO:     Started server process
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

Leave this terminal running!

Method B: Using Python directly

```
# Alternative way to start
python -m uvicorn src.app:app --reload
```

🧪 Step 7: Test the API

Open a **NEW Terminal** (keep the server running in the first terminal)

Test 1: Health Check

```
curl http://localhost:8000/health
```

Expected Response:

```
{
  "status": "healthy",
  "model_loaded": true,
  "preprocessor_loaded": true
}
```

✅ **If you see this, your API is working!**

Test 2: Make a Prediction

```
curl -X POST "http://localhost:8000/predict" \
  -H "Content-Type: application/json" \
  -d '{
    "age": 63,
    "sex": 1,
    "cp": 3,
    "trestbps": 145,
    "chol": 233,
    "fbs": 1,
    "restecg": 0,
    "thalach": 150,
    "exang": 0,
    "oldpeak": 2.3,
    "slope": 0,
    "ca": 0,
    "thal": 1
  }'
```

Expected Response:

```
{
  "prediction": 1,
  "probability": 0.85,
  "risk_level": "high",
  "confidence": 0.85
}
```

✅ **If you see this, your predictions are working!**

Test 3: View API Documentation (Browser)

1. Open your web browser
2. Go to: `http://localhost:8000/docs`
3. You should see **Swagger UI** with interactive API documentation
4. Try clicking on `/predict` → "Try it out" → Enter data → "Execute"

📈 Step 8: View MLflow Experiments (Optional but Recommended)

Open a **THIRD Terminal**

```
# Navigate to project
cd heart-disease-mlops

# Activate virtual environment
source venv/bin/activate

# Start MLflow UI
mlflow ui --port 5000
```

Expected Output:

```
[INFO] Starting gunicorn 20.1.0
[INFO] Listening at: http://127.0.0.1:5000
```

View in Browser

1. Open: `http://localhost:5000`
2. You'll see all your experiments, metrics, and model artifacts

---

🐳 Step 9: Docker Deployment (Alternative Method)

If you prefer using Docker:

Build Docker Image

```
# Build the image
docker build -t heart-disease-api .

# Wait for build to complete (2-5 minutes)
```

Run Docker Container

```
# Run the container
docker run -d \
  -p 8000:8000 \
  --name heart-api \
  heart-disease-api

# Check if running
docker ps
```

Test Docker Container

```
# Health check
curl http://localhost:8000/health

# Make prediction
curl -X POST "http://localhost:8000/predict" \
  -H "Content-Type: application/json" \
  -d @sample_input.json
```

View Logs

```
docker logs -f heart-api
```

Stop Container

```
docker stop heart-api
docker rm heart-api
```

🎭 Step 10: Full Stack with Monitoring (Docker Compose)

For the complete setup with Prometheus and Grafana:

```
# Start all services
docker-compose up -d

# Check all services are running
docker-compose ps
```

Services Available:

- **API:** http://localhost:8000

- **Prometheus:** http://localhost:9090
- **Grafana:** http://localhost:3000 (admin/admin)

View Logs

```
# All services
docker-compose logs -f

# Specific service
docker-compose logs -f api
```

Stop All Services

```
docker-compose down
```

🧪 Step 11: Run Tests

```
# Make sure virtual environment is activated
source venv/bin/activate

# Install test dependencies (if not already installed)
pip install pytest pytest-cov

# Run all tests
pytest tests/ -v

# Run with coverage
pytest tests/ -v --cov=src --cov-report=html
```

Expected Output:

```
tests/test_api.py::test_health_endpoint PASSED
tests/test_api.py::test_predict_endpoint PASSED
tests/test_model.py::test_model_training PASSED
tests/test_preprocessing.py::test_data_loading PASSED
...
======================================
```