Final Report: MLOps Starter — Bank Marketing (Term Deposit Prediction)

Date: August 10, 2025  
Repository: <https://github.com/Mitch1789/mlops_starter>

# 1. Overview

This project delivers a complete, reproducible MLOps workflow for predicting whether a client will subscribe to a term deposit, using the UCI Bank Marketing dataset (Portugal). The solution includes: Git + DVC for code/data/artifact versioning, a modular training pipeline (data\_ingest, data\_validation, train\_and\_tune, evaluate), a Dockerized FastAPI inference service, deployment to Amazon SageMaker via a custom ECR image, and basic monitoring using Amazon CloudWatch. CI/CD is automated with GitHub Actions.

# 2. Architecture Diagram (Mermaid)

flowchart LR  
 A[GitHub Repo] -->|CI| B[Build & Test]  
 B -->|Docker push| C[ECR]  
 C -->|CD| D[SageMaker Endpoint]  
 subgraph DVC Pipeline  
 I[data\_ingest] --> V[data\_validation] --> T[train\_and\_tune] --> E[evaluate]  
 end  
 S3[(S3 DVC Remote)] <--> DVC Pipeline  
 User -->|/predict| D

# 3. Key Design Decisions

* Dataset — UCI Bank Marketing (tabular, PII-free) enables fast iteration and clear baselines.
* Versioning — Git for code; DVC for data and model artifacts (S3 remote).
* Pipeline — Four explicit stages wired in dvc.yaml improve cacheability and failure isolation.
* Baseline Model — scikit-learn RandomForest + small GridSearchCV. Persist the full Pipeline (preprocessor + estimator) to prevent train/serve skew.
* Inference — FastAPI accepts raw features; preprocessing occurs inside the saved Pipeline (OneHotEncoder(handle\_unknown='ignore')).
* Containerization — Single Dockerfile ensures identical environments across local, CI, and SageMaker.
* CI/CD — PRs: lint, tests, SMALL\_RUN sanity training. Main: full DVC, Docker build/push to ECR, deploy/update SageMaker endpoint.
* Monitoring — Optional CloudWatch metrics (latency, request counts) and alarms.

# 4. Baseline Performance Results (latest)

* Train/Test sizes (metrics.json): 1600 / 400
* Accuracy (metrics.json): 0.905
* F1 (positive, metrics.json): 0.367
* Accuracy (evaluate): 0.910
* ROC AUC (evaluate): 0.937
* Class 1 — precision: 0.731, recall: 0.319, f1: 0.444, support: 928
* Macro/Weighted F1 (evaluate): 0.698 / 0.894
* Best hyperparameters: {'clf\_\_max\_depth': 12, 'clf\_\_min\_samples\_split': 2, 'clf\_\_n\_estimators': 100}
* Note: The dataset is imbalanced; consider threshold tuning, class weights, or resampling to lift recall while monitoring precision.

# 5. Deployment Summary

The Docker image is built locally or in CI and pushed to Amazon ECR. The deployment script creates/updates SageMaker Model, EndpointConfig, and Endpoint. Inference uses sagemaker-runtime with JSON inputs matching training-time raw schema. Environment flags (PUBLISH\_CW, METRICS\_NAMESPACE) enable CloudWatch metrics.

# 6. Monitoring & Drift Strategy

* System metrics — track CPU, memory, request latency; set CloudWatch alarms on p90 latency and 5xx error rate.
* Data/Concept drift — compare categorical frequencies and numeric summary stats vs. training baselines; watch prediction rate/confidence drift. Auto-retrain via CI/CD on threshold breaches.
* Logging — persist inference logs (request ID, latency, outcome) for auditability and incident response.

# 7. Lessons Learned

* Keep preprocessing inside the saved Pipeline to avoid train/serve skew.
* Initialize Git before DVC; configure DVC remote after dvc init.
* Container parity matters — single Dockerfile across local, CI, SageMaker.
* Windows hygiene — separate PowerShell commands (no &&); keep repos outside OneDrive to avoid venv locks.
* Cloud consistency — align regions across S3/ECR/SageMaker; allow iam:PassRole for deploy automation.

# Appendix: Reproduction Commands

dvc repro  
dvc push  
uvicorn inference.predict:app --host 0.0.0.0 --port 8080  
python scripts/deploy\_sagemaker.py --image-uri <ECR\_IMAGE\_URI> --role-arn <ROLE\_ARN> --endpoint mlops-starter-endpoint