MLOps Report

IMDb Sentiment Analysis

UPC MLOps Team

October 13, 2025

Machine Learning Operations – Academic Year 2024/2025

Contents

1 Introduction

1.1 Goal of the Project

The project delivers an end-to-end analytics platform that ingests comment sections from major movie-review websites, filters noise such as spam or off-topic content, and extracts sentiment signals that can support downstream dashboards. Starting from the `cookiecutter-data-science` template, we built a reproducible pipeline that downloads the IMDb benchmark dataset, cleans and vectorises text, trains a logistic regression model, and evaluates it against held-out examples. The success criteria agreed during inception were:

- Produce sentiment metrics (accuracy, F1) above 0.85 while maintaining resilience to noisy inputs (spam-filtering and HTML stripping).
- Reproduce the full data and model workflow via DVC with a single `dvc repro`.
- Track experiments (hyperparameters, metrics, artefacts, emissions) in MLflow to support iterative analysis of comment streams.

The latest evaluation run (see Section 2.2) satisfies these targets, with the logistic regression classifier reaching 0.894 accuracy and 0.895 F1.

2 Methodology

2.1 Milestone 1: Inception

2.1.1 Problem Selection and Requirements

We selected binary sentiment classification for IMDb reviews because it combines natural language processing with clear business value (gauging audience sentiment) and publicly available benchmark data. Functional requirements include batch scoring of text reviews and traceable metrics. Non-functional requirements emphasise reproducibility, energy-awareness, and compatibility with container deployment.

2.1.2 Dataset Card Summary

The dataset card (`data/README.md`) follows the Hugging Face template. It documents:

- **Provenance and Licence:** IMDb reviews under the IMDb Terms of Use, crowdsourced annotations, English-only content.
- **Composition:** 50,000 labelled reviews (25k train / 25k test), binary sentiment.
- **Motivation and Uses:** Benchmarking sentiment analysis models, educational demonstrations, not suitable for nuanced or multilingual sentiment.
- **Collection Process:** Neutral reviews excluded, balance enforced between polarities.
- **Limitations & Biases:** Domain specificity, possible demographic bias, binary simplification of sentiment.
- **Citation:** Maas et al. (2011) reference for reproducible acknowledgement.

2.1.3 Model Card Summary

The model card (`models/README.md`) captures:

- **Model Description:** TF–IDF features with a scikit-learn logistic regression classifier.
- **Intended Uses:** Baseline benchmarking and teaching reproducible MLOps practices; not for high-stakes production.
- **Training Procedure:** 25k train / 25k test split, TF–IDF with 1–2 grams, logistic regression (liblinear solver).
- **Evaluation:** Accuracy and F1 typically above 0.82, with current run at 0.894/0.895.
- **Limitations and Ethics:** English-only, binary polarity, sensitivity to sarcasm, potential demographic bias.

2.1.4 Project Coordination and Communication

We operate with a lightweight Scrum cadence: weekly planning and review ceremonies guide backlog priorities, while daily asynchronous stand-ups surface blockers. Slack hosts coordination, and its GitHub and Jira integrations post activity notifications and ticket updates into a shared channel, keeping sprint status visible despite distributed schedules.

2.1.5 Selection of Cloud Provider

For Milestone 4 we plan to deploy the inference service on Google Cloud Run. Its fully managed container runtime, integration with Artifact Registry, and generous free tier make it a pragmatic choice for lightweight academic workloads while providing a path to autoscaling if needed.

2.2 Milestone 2: Model Building – Reproducibility

2.2.1 Project Structure

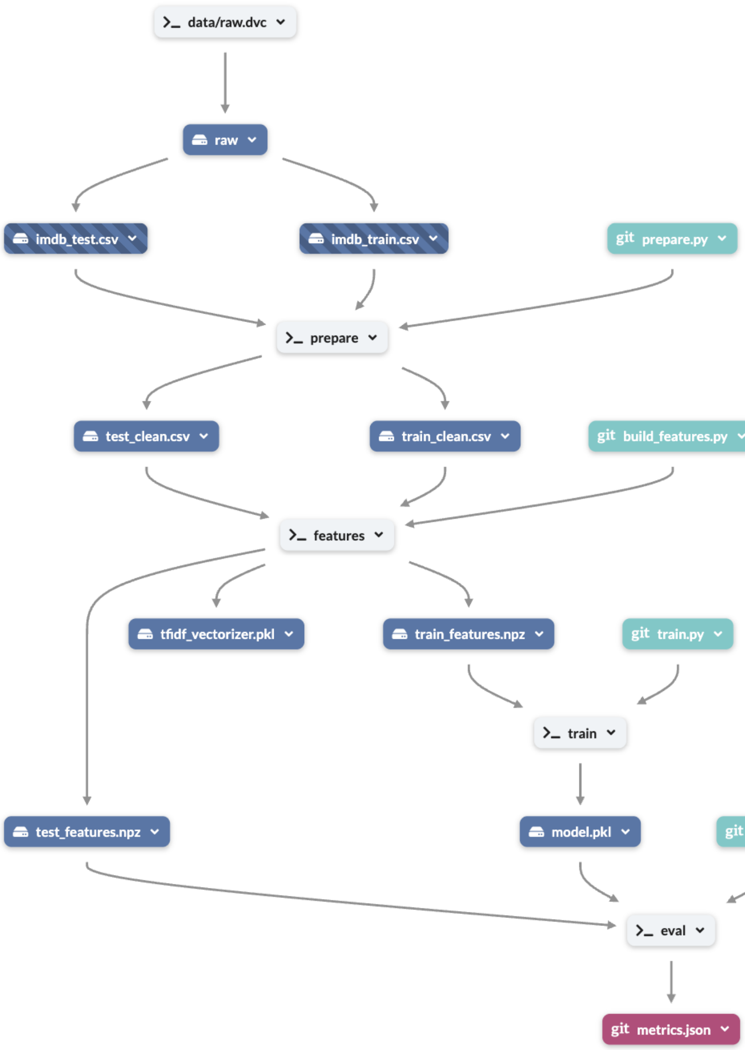
The repository inherits the `cookiecutter-data-science` scaffold, separating code under `mlops_imdb/`, configuration in `params.yaml`, data artefacts under `data/`, and generated reports beneath `reports/`. This consistent layout eases onboarding and tooling integration.

2.2.2 Code Versioning

We apply GitHub Flow on top of the enforced branch naming convention (`feature/`, `fix/`, `chore/`, `docs/`, `experiment/`). Conventional Commit messages feed changelogs and are validated via PR automation (Section **??**).

2.2.3 Data Versioning

DVC stages (`prepare` → `features` → `train` → `eval`) encode dependencies and outputs. Regenerating the pipeline uses `dvc repro`, guaranteeing deterministic artefact creation (`data/processed`,



`models/`, `reports/metrics.json`, figures).

2.2.4 Experiment Tracking

MLflow is configured through `mlops_imdb/config.py`. Both training and evaluation stages wrap their execution in `mlflow.start_run`, logging hyperparameters, dataset dimensions, metrics, confusion-matrix artefacts, and CodeCarbon emissions under the shared `emissions_kg` metric. Local runs default to `./mlruns`, while remote tracking URIs can be supplied via environment variables.

Evaluation Results

Table 1 summarises the most recent evaluation metrics sourced from `reports/metrics.json`. The confusion matrix in Figure 1 reveals a balanced error profile across classes.

Table 1: IMDb test performance (logistic regression baseline).

Metric	Accuracy	Precision	Recall	F1-score
Value	0.8943	0.8907	0.8990	0.8948

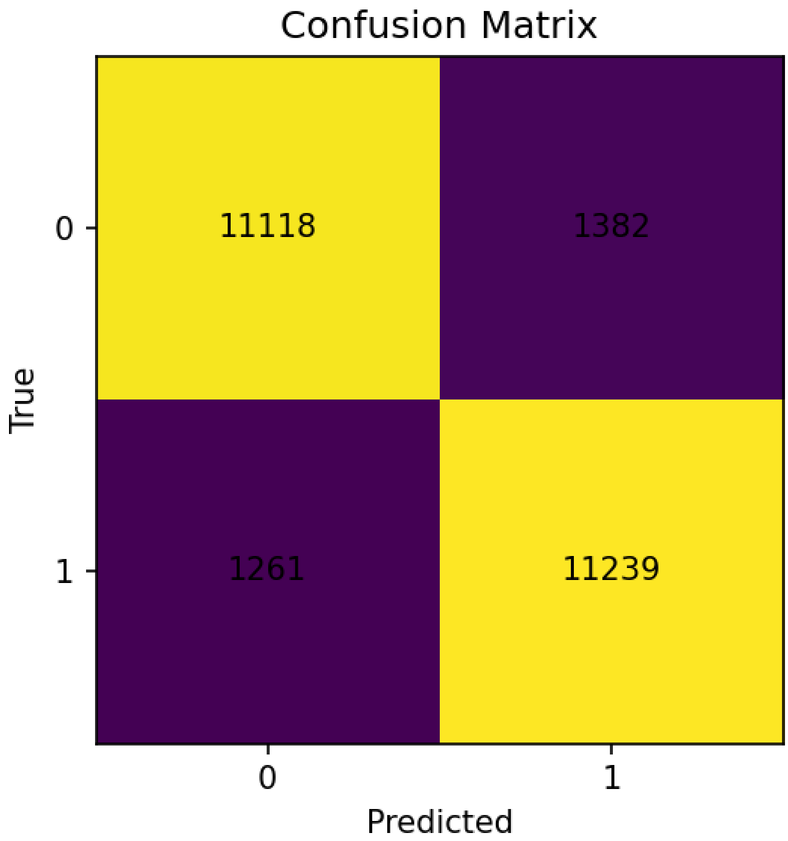


Figure 1: Confusion matrix on the IMDb test split.

2.3 Milestone 3: Model Building – Quality Assurance

2.3.1 Energy Efficiency Awareness

CodeCarbon contexts wrap every DVC stage. The run captured in `reports/codecarbon_emissions.csv` emitted 7.15 mg CO_2eq over 27.1 seconds (Table 2), with feature generation dominating the footprint. Emission figures are logged back to MLflow, enabling longitudinal monitoring.

Table 2: CodeCarbon emissions per stage (mg CO_2eq).

Stage	Duration (s)	Emissions (mg)
Prepare data	5.51	1.21
Build features	14.46	5.52
Train model	3.47	0.21
Evaluate model	3.63	0.22
Total	27.07	**7.15**

2.3.2 Static Code Analysis

Ruff enforces formatting and linting (line length 99, import sorting, code-style checks). The CI pipeline (CI/CD Automation in Milestone 5) runs `uv run ruff format --check` and `uv run ruff check` on every push and pull request.

2.3.3 Model Testing

Pytest suites validate the training and evaluation flows using synthetic sparse matrices (see `tests/modeling/`). Assertions confirm persisted models expose coefficients, metrics JSON contains expected fields, and confusion matrices are rendered as PNG files.

2.3.4 Data Testing

Data preparation tests (`tests/data/test_prepare.py`) exercise schema validation and text-cleaning behaviours, while `tests/data/test_download_imdb.py` mocks the dataset download to assure reproducible CSV outputs. Although we do not yet employ Great Expectations, these lightweight checks catch regressions in preprocessing logic.