Ensuring Reproducibility in Machine Learning Experiments: Tools, Best Practices & Real-World Implementation

Timeline: Day 91–105

Focus: Experimentation & Reproducibility

Why Reproducibility Matters in Machine Learning

In machine learning, reproducibility means being able to run the same experiment again and get the same results. It's not just a good habit—it's necessary for others to trust your work, use it, and build on it. If someone can't repeat your results, they might question how reliable your model really is, no matter how impressive it looks.

Over the last two weeks, we worked on making our machine learning experiments reproducible. We used tools like Docker to create consistent environments and DVC (Data Version Control) to track datasets and models. This blog is a reflection of that experience—what tools we used, the problems we ran into, and how we solved them to build a more reliable and sharable workflow.

Topics Covered

Reproducible Research: What It Means
A reproducible ML workflow ensures that:

• Experiments can be re-run with the same results.
• Environments are controlled, so dependency versions don't affect outcomes.
• Data and model versions are tracked, so results can be traced back to exact input settings.

Tools we Used

Docker: Containerization for Environment Management
Docker helped us encapsulate the development environment so that anyone can run the code without worrying about OS or library mismatches.
- Defined all dependencies in a Dockerfile.
- Used docker-compose to manage multiple services (like Jupyter and DVC).

DVC: Data Version Control
DVC made it easier to:

- Manage and version large datasets efficiently without storing them directly in the Git repository.
- Save and track different versions of model outputs or checkpoints to easily reproduce or compare results.
- Create automated workflows where every data processing and training step can be rerun exactly the same way.

Hands-On Tasks

Step 1: Writing Reproducible Experiment Scripts
- Modularized the code using config files (.yaml) for parameters.
- Logged results, metrics, and configurations using MLflow.

Step 2: Integrating with Version Control
- Pushed codebase and DVC config to GitHub.
- Tracked .dvc files for data and model checkpoints.
- Used .gitignore to manage large file tracking separately via DVC.

Step 3: Documenting the Process
We wrote this blog post to:
- Share the overall architecture.
- Detail the setup challenges.
- Provide code snippets for beginners.

Challenges Faced

- DVC Setup with Remote Storage
  Took time to configure AWS S3 as remote storage. Had to troubleshoot permission issues.
- Docker Image Size
  Our first Docker build was over 5GB. We optimized it using slim base images and multi-stage builds.

- Version Drift in Dependencies
  Installing dependencies without pinning versions led to inconsistent behavior. Resolved by using requirements.txt with fixed versions and Docker.

Final Deliverables

GitHub Repository
Public repository containing:

- **All experiment code:** Complete set of scripts and configurations to run and reproduce all experiments.
- **DVC-tracked data:** Version-controlled datasets and model outputs managed through DVC for easy tracking and retrieval.
- **Docker setup and usage guide:** Docker files and instructions to create a consistent, reproducible development environment.

Research Report & Blog Tutorial

- Best practices guide on reproducibility
- Docker and DVC setup walkthrough
- Sample pipeline with train.py, evaluate.py, and config.yaml

Best Practices for Reproducibility

- Use Docker to run experiments in the same environment on any system.
- Track all data and model files using DVC for easy version control.
- Fix dependency versions to avoid errors from software updates.
- Log parameters, metrics, and results to keep a clear experiment record.
- Version control code, data, and models to make the entire project reproducible.

Conclusion

This project was a significant milestone for our team, helping us realize the true importance of reproducibility in machine learning research. By adopting tools like Docker and DVC, we learned how to properly manage environments, track data and model versions, and keep our experiments well-documented. These practices improved the overall structure and reliability of our workflow, making it much easier to collaborate and avoid confusion. As a team, we now feel more confident in setting up experiments that anyone can reproduce and build upon. Reproducibility has helped us work more efficiently together and ensures that our results remain transparent, consistent, and impactful for future projects.