Greensteam MLOps

9 Steps in the MLOps Case Study:

This structured MLOps process enables GreenSteam to deliver reliable, scalable, and impactful machine learning solutions for the maritime industry.

1. Shared ML Codebase

GreenSteam maintains a single shared ML codebase that supports all development teams. This approach ensures consistency, reduces duplication, and simplifies collaboration across projects. Code reusability is maximized, enabling the rapid adaptation of models and pipelines to different use cases. This shared foundation also helps enforce coding standards and best practices.

2. Dealing with Dependencies and Reproducibility

Ensuring that ML models are reproducible requires careful management of dependencies and environments. GreenSteam uses tools like Docker to create consistent, portable containers that encapsulate all necessary libraries and dependencies. This ensures that models behave identically across development, testing, and production environments.

3. How to Deal with Tests That Don't Test

GreenSteam prioritizes meaningful testing by ensuring that tests validate real-world performance, not just edge cases or trivial conditions. Smoke tests and integration tests verify pipeline functionality, while domain-specific tests ensure that predictions align with operational expectations. This ensures that only robust models reach production.

4. Code Quality, Type Checking, and Continuous Integration

Code quality is maintained through automated type checking, linting, and peer reviews. Continuous integration pipelines, managed with tools like Jenkins, automatically run tests

for every code change, ensuring that only high-quality code is merged. This reduces technical debt and speeds up deployment cycles.

5. Adding Orchestration, Pipelines, and Moving from Monolith to Microservices

GreenSteam transitioned from a monolithic architecture to a microservices-based design. This enables modular pipelines where individual components (e.g., data preprocessing, model training, and serving) are independently managed and scaled. Kubernetes orchestrates these microservices, ensuring reliability and scalability.

6. Keeping Track of All Those Model Versions and Results

Managing model versions and results is critical for traceability and auditing. GreenSteam uses tools like Neptune.ai to log experiments, track model configurations, and store performance metrics. This centralized tracking ensures that each version can be traced back to its data, code, and results.

7. Custom Model Serving

GreenSteam developed a custom model-serving framework tailored to the maritime domain. This system integrates seamlessly with existing infrastructure, providing real-time predictions while optimizing for latency and resource efficiency. It also supports dynamic scaling based on operational demands.

8. Human-in-the-Loop Model Auditing

Post-training, domain experts manually review model outputs using detailed reports. This human-in-the-loop process ensures that models meet business goals and domain-specific standards. Visualizations and metrics provide transparency, enabling informed decisions about model deployment.

9. Feature Store and Microservices (In Progress)

GreenSteam is working on implementing a feature store to centralize feature management and reuse across models. Coupled with their microservices architecture, the

feature store will ensure consistency and scalability in feature engineering. This initiative aims to further streamline the development pipeline and enhance operational efficiency.

MLOps Tool Stack Summary

1. **Git for Version Control**: Centralized repository for code and configuration management.
2. **Jenkins for CI/CD**: Automates testing, training, and deployment pipelines to streamline updates.
3. **Kubernetes for Orchestration**: Manages containerized environments, ensuring scalability and reliability.
4. **Docker for Containers**: Encapsulates the environment for model training, evaluation, and deployment.
5. **Prometheus for Monitoring**: Tracks key metrics like resource utilization, model predictions, and system health.
6. **Grafana for Visualization**: Provides dashboards to visualize metrics, system performance, and model predictions.
7. **Neptune.ai for Experiment Tracking**: Logs experiments, tracks model parameters, and organizes metrics for easy comparison.
8. **Data Lake for Storage**: Centralized storage system for raw and processed data used in training and monitoring.
9. **Domain-Specific Tools**: Custom tools and libraries for feature engineering, validation, and domain-specific tasks.