Assignment 5

Shandon Herft

This project aims to classify images of buildings into one of 25 architectural styles and, in parallel, predict era years for historical context. After exploratory analysis, feature engineering, and model comparison (including Random Forest, CNN, and KNN), the Random Forest classifier emerged as the most viable option for immediate deployment. This report outlines a strategy for model deployment, ongoing MLOps practices, and potential security concerns along with mitigation strategies.

1. Model Deployment Strategy
   • Containerization & API:
   – Package the trained Random Forest model within a Docker container.
   – Expose a RESTful API (using frameworks like Flask or FastAPI) to serve predictions.

• Integration into Production Environment:
– Deploy the containerized model on a cloud service (e.g., AWS, GCP, or Azure) using orchestration tools like Kubernetes to ensure scalability and high availability.
– Utilize load balancers to distribute incoming requests efficiently.

• Data Pipeline Integration:
– Implement an ETL (Extract, Transform, Load) process to ensure that incoming images are preprocessed (resized, normalized, and encoded) consistently with the training data.
– Maintain versioning of both the model and preprocessing code to ensure reproducibility.

2. MLOps Considerations
   • Continuous Monitoring and Logging:
   – Monitor key metrics such as prediction accuracy, latency, and error rates in real time using tools like Prometheus or CloudWatch.
   – Track input data distribution and detect drift, ensuring that the model's performance remains stable over time.

• Model Maintenance and Retraining:
– Establish automated retraining pipelines that trigger when performance metrics fall below a predefined threshold.
– Use A/B testing to compare updated models against the current production model.
– Version control models and maintain detailed logs for auditability and compliance.

• Automated Alerts and Incident Response:
– Configure alerting systems to notify the development team of anomalies or performance degradation.
– Schedule periodic reviews of the deployed model to incorporate improvements from ongoing research or additional feature engineering.

3. Security Considerations and Mitigation
   • Input Validation and Sanitization:
   – Validate incoming image data to ensure that only properly formatted images are processed, preventing potential injection attacks.
   – Implement rate limiting and CAPTCHA mechanisms on public endpoints to mitigate denial-of-service (DoS) attacks.

• Secure Communication and Access Control:
– Use HTTPS to encrypt data in transit between clients and the model API.
– Restrict API access with authentication tokens and implement role-based access control (RBAC) to ensure that only authorized users can access sensitive endpoints.

• Data Privacy and Model Vulnerabilities:
– Regularly audit logs for anomalous activities that might suggest adversarial attacks on the model.
– Employ techniques such as adversarial training and input sanitization to protect against malicious inputs designed to trick the model.