# Machine Learning Project Documentation

## Deployment

### 1. Overview

The deployment phase focuses on operationalizing the AI-powered multimodal diagnostic model for endemic infectious diseases in Liberia—covering malaria, TB, HIV/AIDS, and typhoid. This phase ensures that the refined model (achieving up to 93% multimodal accuracy) is made accessible through a scalable, secure, and efficient mobile and cloud-based health screening platform. Deployment activities involved exporting trained models to optimized formats for mobile inference, integrating model APIs into Android/iOS applications, establishing real-time prediction endpoints and analytics dashboards, and implementing continuous monitoring for performance and data drift.

### 2. Model Serialization

The multimodal models were serialized using appropriate frameworks to ensure cross-platform deployment efficiency. CNN models for malaria and TB detection were saved in TensorFlow Lite (.tflite) format to support on-device inference. XGBoost models for clinical and environmental data were serialized using joblib, while ensemble weights were stored as pickle (.pkl) files for runtime model fusion. These formats were selected for compactness, compatibility, and optimized inference speed on both mobile and cloud infrastructures (Chollet, 2017; Pedregosa et al., 2011).

### 3. Model Serving

The deployment architecture integrates on-device and cloud-hosted components to enable both offline and real-time functionality. On-device inference allows the model to operate in low-connectivity regions using TensorFlow Lite embedded within the mobile health application, while a cloud-hosted Flask/FastAPI backend deployed on AWS EC2 manages structured data predictions and model fusion. PostgreSQL databases store patient-level records, and AWS S3 provides encrypted cloud storage for models and artifacts (Paszke et al., 2019). AWS Elastic Beanstalk and Docker ensure horizontal scalability and fault tolerance.

### 4. API Integration

The models are served through a RESTful API architecture enabling seamless access to prediction services. The API is designed using FastAPI with endpoints for both image-based and structured data predictions. Endpoints include /predict/image for image uploads (malaria smears, chest X-rays), /predict/clinical for structured data inputs, /predict/multimodal for integrated disease outcome predictions, and /monitor/status for service health monitoring. Requests are accepted in JSON and multipart formats, while responses are returned as JSON objects with probability distributions and confidence intervals.

### 5. Security Considerations

Security protocols were prioritized to ensure compliance with WHO data governance standards and Liberia’s Ministry of Health data protection guidelines (WHO, 2023; Liberia Ministry of Health, 2022). All API communications are secured using HTTPS and SSL encryption, and stored data is encrypted with AES-256. OAuth2 authentication manages access control, and role-based permissions define usage tiers for administrators, clinicians, and researchers. Comprehensive logs capture API requests, responses, and prediction events for traceability.

### 6. Monitoring and Logging

Model performance is continuously monitored using AWS CloudWatch and the ELK (Elasticsearch, Logstash, Kibana) stack. Tracked metrics include accuracy, latency, drift, request volumes, and error rates. Automated alerts are configured to trigger when accuracy drops below thresholds. Version tagging ensures traceability across model iterations, and user feedback is integrated into retraining pipelines to improve generalization (Chen & Guestrin, 2016).

### 7. Conclusion

The deployment framework ensures that the multimodal diagnostic system operates effectively in both urban and rural Liberian settings. By leveraging TensorFlow Lite, FastAPI, and AWS, the system achieves real-time disease diagnostics and scalable cloud integration. This deployment supports Liberia’s digital health agenda by enhancing disease surveillance, diagnostic access, and healthcare decision-making.

## References

* Chollet, F. (2017). Deep Learning with Python. Manning Publications.
* Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.
* Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems (NeurIPS).
* Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.
* World Health Organization. (2023). Global Health Observatory Data Repository. Geneva: WHO.
* Liberia Ministry of Health. (2022). Annual Health Statistics Report (HMIS). Monrovia: Ministry of Health.