PREDICTING MALARIA OUTBREAKS IN RURAL LIBERIA USING MACHINE LEARNING

FTL Liberia Group Six (6)

ABSTRACT

This deployment documentation describes the comprehensive strategy for operationalizing our machine learning-based malaria outbreak prediction system in rural Liberia. Building upon the refined ensemble model that achieved 89.8% test accuracy with 86.4% outbreak detection sensitivity, this phase encompasses model serialization, serving architecture, API integration, security implementation, and operational monitoring protocols. The deployment strategy prioritizes reliability, scalability, and seamless integration with Liberia's existing health information systems (DHIS2 and LMIS) while addressing the unique challenges of resource-constrained rural healthcare settings. This system directly supports SDG 3 (Good Health and Well-being) by providing health authorities with actionable 3-month advance warning of district-level malaria outbreaks, enabling proactive resource allocation and targeted interventions.

Group Members:

1. Nehemiah Kemayah
2. Thomas Adonis Marwolo
3. Robert Bright Yekeh
4. Joel J. Barclay

Machine Learning Deployment Documentation

1. OVERVIEW

The deployment phase represents the critical transition from validated machine learning models to an operational early warning system serving Liberia's rural health infrastructure. This phase transforms our refined ensemble model—comprising XGBoost (weight: 0.65) and Random Forest (weight: 0.35)—into a production-ready system capable of generating monthly county-level outbreak predictions with 89.8% accuracy and 86.4% sensitivity.

Deployment Objectives

- **Operational Readiness:** Establish production infrastructure capable of generating predictions for all 15 counties monthly with 99.5% uptime
- **System Integration:** Seamlessly integrate with existing DHIS2 surveillance and LMIS commodity management platforms
- **Scalability:** Design architecture supporting future expansion to county-level granularity and extended prediction horizons
- **Reliability:** Implement comprehensive monitoring, automated alerting, and failover mechanisms, ensuring continuous availability
- **Security:** Protect sensitive health data through encryption, access control, and audit logging compliant with health data privacy standards

Deployment Context and Constraints

Rural Liberian healthcare infrastructure presents unique deployment challenges, including intermittent internet connectivity requiring offline-capable solutions, limited computational resources at county health offices necessitating cloud-based architecture, fragmented data reporting systems requiring robust data validation and imputation, and existing DHIS2 and LMIS platforms requiring careful integration planning. Our deployment strategy explicitly addresses these constraints through lightweight API endpoints, offline prediction caching, automated data quality checks, and phased rollout validation.

Phased Deployment Strategy

Following best practices for health system technology deployment, we implement a three-phase rollout strategy. Phase 1 (Months 1-3) establishes pilot deployment in three counties with the strongest data quality— Montserrado, Bong, and Grand Bassa—enabling validation, stakeholder feedback collection, and iterative refinement. Phase 2 (Months 4-6) expands to eight additional counties with DHIS2 integration, automated prediction scheduling, and county health officer training. Phase 3 (Months 7-12) achieves national rollout across all 15 counties with full operational monitoring, continuous model retraining protocols, and established feedback mechanisms for system improvement.

2. MODEL SERIALIZATION

Model serialization transforms our trained ensemble model into persistent storage format enabling consistent predictions across deployment infrastructure. This process ensures reproducibility, version control, and efficient model loading for production inference.

Serialization Format and Rationale

We employ joblib for model serialization due to its superior performance with large NumPy arrays and scikit-learn objects. Joblib provides efficient compression, fast loading times critical for operational deployment, and broad compatibility with our technology stack (Python 3.9+, scikit-learn 1.0+, XGBoost 1.6+).

Serialization Components
Our serialization strategy persists multiple components essential for complete prediction pipeline reconstruction. The trained ensemble model bundle includes XGBoost classifier with optimized hyperparameters (learning_rate=0.03, max_depth=6, n_estimators=500), Random Forest classifier with tuned parameters (n_estimators=300, max_features='sqrt'), and ensemble weights (0.65 for XGBoost, 0.35 for Random Forest). Feature preprocessing artifacts comprise StandardScaler fitted on training data for climate variables, imputation parameters for missing value handling, and engineered feature computation functions (climate suitability index, vector breeding potential). Metadata and configuration include feature names and ordering, model version and training date, performance metrics from validation and test phases, and prediction threshold (0.42) optimized for outbreak detection.

Implementation Code
Model Serialization Script:

```
import joblib import json from datetime import datetime  # Serialize trained models model_bundle = {     'xgboost_model': xgb_model,     'random_forest_model': rf_model,     'ensemble_weights': {'xgboost': 0.65, 'random_forest': 0.35},     'scaler': scaler,     'imputer': imputer,     'feature_names': feature_names,     'prediction_threshold': 0.42,     'version': '1.0.0',     'training_date': datetime.now().isoformat(),     'performance_metrics': {         'test_accuracy': 0.898,         'test_sensitivity': 0.864,         'test_specificity': 0.917,         'roc_auc': 0.947     } }  # Save with compression joblib.dump(model_bundle, 'models/malaria_ensemble_v1.0.0.joblib', compress=3)  # Save metadata separately for quick access metadata = {     'version': model_bundle['version'],     'training_date': model_bundle['training_date'],     'performance_metrics': model_bundle['performance_metrics'],     'feature_count': len(feature_names) } with open('models/metadata_v1.0.0.json', 'w') as f:     json.dump(metadata, f, indent=2)
```

Version Control Strategy
We implement semantic versioning (MAJOR.MINOR.PATCH) for model versions. Major version increments indicate architectural changes requiring API modifications, minor version increments reflect model retraining with improved performance or new features, and patch version increments address bug fixes without model retraining. Each serialized model is stored with a version identifier, training timestamp, and performance metrics, enabling rollback if deployment issues arise. Model registry maintains a complete versioning history supporting reproducibility and compliance requirements.

Storage and Backup
Serialized models are stored in cloud object storage (Amazon S3 or equivalent) with versioning enabled, ensuring immutability and disaster recovery capability. Primary storage maintains the current production model (v1.0.0) plus two previous versions for rapid rollback. Backup storage replicates all model versions to a geographically separate region, preventing data loss from regional failures. Local caching at application servers reduces latency and enables continued operation during temporary cloud connectivity issues. Model size optimization through compression (joblib compress=3) reduces storage costs and network transfer times while maintaining full model fidelity.

3. MODEL SERVING

Model serving architecture transforms serialized models into a scalable prediction infrastructure capable of handling production workloads. Our serving strategy balances performance, reliability, and cost considerations while accommodating rural Liberia's infrastructure constraints.

Deployment Platform Selection
We plan to deploy on cloud infrastructure (Amazon Web Services or Google Cloud Platform) selected for reliability, global reach, and comprehensive health data compliance certifications. Cloud deployment provides elastic scalability supporting variable prediction workloads, managed services reducing operational complexity for limited IT staff, a global content delivery network enabling low-latency access from rural areas, and automated backup and disaster recovery meeting health system reliability requirements.

Serving Architecture

Our three-tier architecture comprises a presentation layer (web dashboard and mobile interface for county health officers), application layer (Flask REST API serving predictions with authentication and rate limiting), and data layer (PostgreSQL for prediction history and user management, Redis cache for frequently accessed predictions). Load balancing across multiple application servers ensures high availability and fault tolerance. Autoscaling dynamically adjusts compute resources based on request volume maintaining consistent performance during peak usage periods.

Prediction Service Implementation

Flask API Server Code:

```
from flask import Flask, request, jsonify import joblib import numpy as np import pandas as pd from datetime import datetime app = Flask(__name__)  # Load model on startup MODEL_PATH = 'models/malaria_ensemble_v1.0.0.joblib' model_bundle = joblib.load(MODEL_PATH)  @app.route('/predict', methods=['POST']) def predict_outbreak():     try:         # Extract input features         data = request.get_json()         county = data['county']         features = pd.DataFrame([data['features']])                 # Preprocessing pipeline         features_scaled = model_bundle['scaler'].transform(features)                 # Ensemble prediction         xgb_pred = model_bundle['xgboost_model'].predict_proba(features_scaled)[:, 1]         rf_pred = model_bundle['random_forest_model'].predict_proba(features_scaled)[:, 1]                 ensemble_prob = (0.65 * xgb_pred + 0.35 * rf_pred)[0]         prediction = int(ensemble_prob >= model_bundle['prediction_threshold'])                 # Confidence interval estimation         confidence = abs(ensemble_prob - 0.5) * 2                 response = {             'county': county,             'prediction': prediction,             'probability': float(ensemble_prob),             'confidence': float(confidence),             'risk_level': get_risk_level(ensemble_prob),             'timestamp': datetime.now().isoformat(),             'model_version': model_bundle['version']         }                 return jsonify(response), 200             except Exception as e:         return jsonify({'error': str(e)}), 500  def get_risk_level(probability):     if probability >= 0.7:         return 'HIGH'     elif probability >= 0.5:         return 'MODERATE'     elif probability >= 0.3:         return 'LOW'     else:         return 'VERY_LOW'  if __name__ == '__main__':     app.run(host='0.0.0.0', port=5000)
```

Performance Optimization

Several optimization strategies ensure a responsive prediction service. Model caching loads serialized models into memory at server startup, eliminating repeated disk I/O. Response caching with Redis stores frequently requested predictions (county-month combinations), reducing redundant computation. Batch prediction processing handles multiple counties simultaneously when monthly updates occur, improving throughput efficiency. Asynchronous processing for non-urgent predictions prevents blocking interactive requests, maintaining dashboard responsiveness.

Offline Capability

Addressing intermittent connectivity in rural areas requires offline operation capability. Prediction caching stores the last 6 months of county-level predictions locally at county health offices, enabling access during connectivity outages. Progressive web application (PWA) technology allows the dashboard to function offline with cached predictions. Automatic synchronization resumes when connectivity is restored, uploading locally generated data and downloading the latest predictions. Prediction validity indicators alert users when cached predictions become stale, requiring refresh.

4. API INTEGRATION

API integration provides standardized interfaces enabling seamless communication between the prediction system and Liberia's health information infrastructure. Our RESTful API design follows industry best practices, ensuring ease of integration, comprehensive documentation, and long-term maintainability.

API Architecture and Design Principles

We implement RESTful API architecture adhering to REST principles, including stateless communication, resource-based URL structure, standard HTTP methods (GET, POST, PUT, DELETE), and JSON data format. API versioning (v1, v2) in URL paths ensures backward compatibility as the system evolves. Comprehensive error handling provides informative status codes and error messages, facilitating client-side debugging. Pagination supports efficient retrieval of large datasets when historical predictions are requested.

Core API Endpoints

Endpoint	Method	Description
/api/v1/predict	POST	Generate single county-month outbreak prediction with probability and confidence scores
/api/v1/predict/batch	POST	Process multiple county predictions simultaneously for monthly updates
/api/v1/predictions/history	GET	Retrieve historical predictions filtered by county, date range, and risk level
/api/v1/model/info	GET	Return current model version, performance metrics, and last training date
/api/v1/health	GET	Health check endpoint for monitoring system availability and response time

Request and Response Formats

Sample Prediction Request:

POST /api/v1/predict Content-Type: application/json Authorization: Bearer <access_token> { "county": "Bong", "month": "2025-11", "features": { "rainfall_3month_lag": 245.3, "temperature_2month_lag": 26.8, "humidity_1month_lag": 78.5, "climate_suitability_index": 0.82, "vector_breeding_potential": 0.76, "population_density": 87.2, "seasonal_indicator": 1 } }

Sample Prediction Response:

HTTP/1.1 200 OK Content-Type: application/json { "county": "Bong", "month": "2025-11", "prediction": 1, "probability": 0.73, "confidence": 0.46, "risk_level": "HIGH", "recommendation": "Increase ITN distribution and activate rapid response team", "timestamp": "2025-10-23T14:30:00Z", "model_version": "1.0.0" }

DHIS2 Integration

Integration with Liberia's DHIS2 platform enables automated data exchange and seamless workflow integration. Automated data extraction retrieves monthly climate data and malaria case reports from DHIS2 using its REST API with OAuth2 authentication. Prediction upload posts generated outbreak forecasts back to DHIS2 as custom indicators visible in existing dashboards. Event-based triggers automatically initiate prediction workflow when new surveillance data becomes available. Data validation ensures consistency between prediction system and DHIS2 preventing synchronization errors.

API Documentation and Client Libraries

Comprehensive API documentation generated with OpenAPI (Swagger) specification provides interactive testing interface, automatic client library generation for Python and JavaScript, and complete endpoint descriptions with request/response examples. Developer portal hosts documentation, code samples, and integration guides lowering technical barriers for third-party developers. Client libraries in Python and JavaScript simplify integration for common use cases reducing implementation time for county health information systems.

5. SECURITY CONSIDERATIONS

Security implementation protects sensitive health data, ensures system integrity, and maintains user privacy throughout the prediction workflow. Our security architecture addresses authentication, authorization, data encryption, audit logging, and compliance with health data protection regulations.

Authentication and Authorization

Multi-layered access control ensures appropriate system access. JWT (JSON Web Token) authentication provides stateless, secure API authentication with configurable token expiration (4 hours) and refresh mechanisms. Role-based access control (RBAC) defines three user roles: Administrator with full system access, including model management and user administration, County Health Officer with read access to predictions for assigned counties and the ability to submit validation feedback, and API Client with programmatic access for system integrations with rate limiting. OAuth2 integration with the existing Ministry of Health identity provider enables single sign-on, reducing password management burden while centralizing access control.

Data Encryption

Comprehensive encryption protects data throughout its lifecycle. Encryption in transit uses TLS 1.3 for all API communications preventing eavesdropping and man-in-the-middle attacks. Encryption at rest applies AES-256 encryption to database storage protecting sensitive health data from unauthorized access if storage media is compromised. Field-level encryption provides additional protection for personally identifiable information (PII) such as county-specific epidemiological details. Encryption key management uses cloud provider key management services with automatic rotation and secure backup ensuring both security and availability.

Data Privacy and Compliance

Our implementation adheres to health data privacy principles and emerging regulations. Data minimization ensures collection and retention only of data necessary for outbreak prediction avoiding unnecessary exposure of sensitive information. Anonymization applies to aggregated datasets used for model training removing identifiable information while preserving predictive utility. Access logging records all data access events enabling audit trails for compliance verification and security incident investigation. Data retention policies automatically purge prediction data older than 5 years consistent with public health data retention requirements while maintaining sufficient history for longitudinal analysis.

Network Security

Network-level protections defend against common attack vectors. Web application firewall (WAF) filters malicious requests protecting against SQL injection, cross-site scripting (XSS), and other OWASP Top 10 vulnerabilities. Rate limiting prevents denial-of-service attacks and API abuse by restricting request frequency per user/IP address. IP whitelisting for administrative functions restricts access to Ministry of Health network ranges reducing attack surface. DDoS protection through cloud provider services maintains availability during distributed denial-of-service attacks. Virtual private cloud (VPC) isolation segregates prediction infrastructure from public internet requiring authorized access through secure gateways.

Security Monitoring and Incident Response

Proactive security monitoring enables rapid threat detection and response. Security information and event management (SIEM) system aggregates logs from all infrastructure components identifying suspicious patterns and potential security incidents. Automated alerting notifies security team of anomalous activities such as unusual access patterns, failed authentication attempts exceeding threshold, and unauthorized data access attempts. Incident response procedures define clear escalation paths, containment strategies, and recovery protocols ensuring rapid mitigation of security breaches. Regular security audits conducted quarterly assess system vulnerabilities and validate control effectiveness. Penetration testing performed annually by independent security firm identifies exploitable weaknesses before malicious actors discover them.

6. MONITORING AND LOGGING

Comprehensive monitoring and logging infrastructure ensures operational visibility, performance optimization, and rapid issue detection. Our monitoring strategy encompasses system health metrics, model performance tracking, user activity logging, and automated alerting enabling proactive system management.

Monitoring Architecture

We implement three-tier monitoring architecture providing comprehensive system observability. Infrastructure monitoring through cloud provider native tools (CloudWatch for AWS) tracks server health including CPU utilization, memory consumption, disk I/O, and network throughput identifying resource bottlenecks before performance degradation. Application monitoring using Application Performance Management (APM) tools tracks API endpoint response times, error rates, request throughput, and database query performance enabling optimization of application code. Model performance monitoring implements custom metrics tracking prediction accuracy against observed outcomes, feature drift detection, and confidence score distribution identifying model degradation requiring retraining.

Key Performance Indicators

Metric Category	Specific Metrics	Target Threshold
System Availability	Uptime percentage, Response time	99.5% uptime, <500ms response
Prediction Accuracy	Monthly validation accuracy, Sensitivity	>85% accuracy, >82% sensitivity
Data Quality	Missing data rate, Feature drift	<10% missing, <15% drift
User Engagement	Active users, Dashboard sessions	80% counties active monthly
Resource Utilization	CPU usage, Memory consumption	<70% average utilization

Logging Framework

Structured logging using JSON format enables efficient log analysis and automated processing. Application logs capture all API requests including endpoint accessed, user identity, request parameters, response status, and execution time. Error logs record exceptions with full stack traces, contextual information, and severity classification facilitating rapid troubleshooting. Prediction logs document each prediction generated including input features, model outputs, confidence scores, and timestamps enabling retrospective analysis and model validation. Audit logs track security-relevant events such as authentication attempts, authorization decisions, data access, and configuration changes supporting compliance requirements and forensic investigation.

Alerting and Notification System

Automated alerting ensures rapid response to system issues and model performance degradation. Critical alerts for system downtime, prediction accuracy dropping below 85% for two consecutive months, data pipeline failures preventing prediction generation, and security incidents trigger immediate SMS and email notifications to technical team requiring acknowledgment within 15 minutes. Warning alerts for elevated error rates, resource utilization exceeding thresholds, feature drift indicators, and unusual access patterns

generate email notifications investigated within 2 hours. Informational alerts for successful monthly prediction completion, model retraining completion, and scheduled maintenance windows provide operational awareness through dashboard and weekly digest emails.

Model Performance Monitoring
Continuous model performance tracking detects degradation requiring retraining. Prediction validation compares forecasted outbreaks against observed outcomes calculated monthly with 1-month lag providing real-world accuracy metrics. Confidence calibration analyzes whether predicted probabilities align with actual outbreak frequencies ensuring reliable uncertainty estimates. Feature importance stability tracks whether model feature rankings remain consistent identifying potential data quality issues or environmental changes. Prediction distribution monitoring detects shifts in predicted outbreak probabilities suggesting model drift or changing epidemiological patterns.

Dashboard and Visualization
Real-time monitoring dashboard provides operational visibility for system administrators and health officials. System health panel displays current uptime, API response times, and resource utilization with 5-minute refresh interval. Model performance panel shows rolling validation accuracy, sensitivity trends, and feature drift indicators updated monthly. Geographic visualization maps county-level outbreak predictions with color-coded risk levels and historical trends. Alert panel lists active alerts with severity classification and acknowledgment status enabling rapid incident triage. Usage analytics track user engagement, prediction requests, and dashboard access patterns informing system optimization priorities.

7. DEPLOYMENT SCHEDULE AND MILESTONES

1. **Phase 1: Pilot Deployment (Months 1-3)**
    - **Week 1-2:** Infrastructure setup (cloud resources, database, API server)
    - **Week 3-4:** Deploy prediction system to three pilot counties (Montserrado, Bong, Grand Bassa)
    - **Week 5-8:** Generate initial predictions and train county health officers
    - **Week 9-12:** Collect feedback, validate predictions, and refine system
2. **Phase 2: Expansion (Months 4-6)**
    - **Week 13-14:** Integrate with DHIS2 for automated data exchange
    - **Week 15-18:** Expand to eight additional counties
    - **Week 19-22:** Implement automated monthly prediction scheduling
    - **Week 23-24:** Conduct mid-deployment evaluation and adjustments
3. **Phase 3: National Rollout (Months 7-12)**
    - **Week 25-28:** Deploy to the remaining four counties, achieving national coverage
    - **Week 29-36:** Establish routine monitoring and quarterly model retraining
    - **Week 37-44:** Implement feedback mechanisms and continuous improvement process
    - **Week 45-52:** Conduct annual comprehensive evaluation and system optimization

CONCLUSION

This deployment documentation establishes comprehensive framework for operationalizing our malaria outbreak prediction system in rural Liberia. The deployment strategy addresses technical infrastructure requirements, security considerations, integration challenges, and operational monitoring needs specific to resource-constrained healthcare settings.

Key Deployment Achievements

Our deployment architecture successfully transforms the refined ensemble model achieving 89.8% test accuracy and 86.4% sensitivity into production-ready early warning system. Cloud-based infrastructure provides scalability, reliability, and global accessibility essential for rural county access. RESTful API design enables seamless integration with existing DHIS2 surveillance platform. Comprehensive security implementation protects sensitive health data through encryption, authentication, and access control. Robust monitoring and logging framework ensures operational visibility and rapid issue detection. Phased rollout strategy validates system effectiveness while building stakeholder confidence and operational capacity.

Operational Impact

This deployment enables transformative shift from reactive malaria response to proactive prevention. County health officers gain 3-month advance warning of outbreak probability facilitating strategic resource allocation, targeted intervention deployment, and community mobilization before outbreak peaks. Automated integration with DHIS2 reduces manual data entry burden enabling health workers to focus on clinical care. Offline capability ensures continuous access to predictions despite intermittent connectivity common in rural areas. Standardized API interface supports future development of mobile applications and integration with additional health information systems.

Sustainability and Future Enhancements

Deployment sustainability requires ongoing technical maintenance, user support, and continuous improvement. Automated monitoring and alerting minimize manual system oversight requirements. Quarterly model retraining incorporating recent data maintains prediction accuracy as epidemiological patterns evolve. Cloud infrastructure scales cost-effectively with usage preventing over-provisioning of resources. Comprehensive documentation and training materials enable local technical capacity development reducing dependence on external expertise.

Future enhancement opportunities include extending prediction horizon beyond 3 months through incorporation of sea surface temperature and oceanic climate indicators, developing mobile applications for field-level access by community health workers, incorporating vector surveillance data and intervention coverage metrics to improve prediction accuracy, implementing machine learning model explainability features for transparency and stakeholder trust, and establishing feedback mechanisms capturing user experiences and operational insights for continuous system improvement.

Alignment with Sustainable Development Goals

This deployment directly advances SDG 3 (Good Health and Well-being) through provision of actionable early warning system reducing malaria burden in vulnerable rural populations. System implementation supports SDG 9 (Industry, Innovation, and Infrastructure) demonstrating application of advanced machine learning technology addressing critical public health challenges in resource-constrained settings. Collaborative development approach aligns with SDG 17 (Partnerships for the Goals) bringing together technical expertise, health system knowledge, and community engagement achieving shared objectives.

Final Remarks

The successful deployment of this malaria outbreak prediction system represents significant milestone in applying machine learning to public health challenges in Sub-Saharan Africa. By providing health authorities with reliable, accessible, and actionable outbreak predictions, this system has potential to transform malaria control from crisis response to strategic prevention. The comprehensive deployment framework established through this documentation ensures system reliability, security, and sustainability while accommodating unique constraints of rural Liberian healthcare infrastructure. As the system matures

through operational deployment and continuous refinement, it serves as replicable model for similar applications across resource-limited settings facing endemic infectious disease challenges.

REFERENCES

Balogun, A.L., et al. (2021). Prediction of malaria incidence using climate variability and machine learning. Informatics in Medicine Unlocked, 22, 100508.

Jaiteh, F., et al. (2024). Predicting malaria outbreak in The Gambia using machine learning techniques. PLOS One, 19(5), e0304289.

Martineau, P., et al. (2022). Predicting malaria outbreaks from sea surface temperature variability up to 9 months ahead in Limpopo, South Africa. Frontiers in Public Health, 10, 962377.

Merkord, C.L., et al. (2021). Predicting malaria epidemics in Burkina Faso with machine learning. PLOS One, 16(6), e0253302.

Woldegiorgis, A.B., et al. (2023). Machine Learning Techniques for Predicting Malaria in Sub-Saharan Africa. In Artificial Intelligence and Machine Learning for Healthcare, Springer.

Liberia Ministry of Health DHIS2 System - National Malaria Control Program surveillance data

World Bank Climate Change Knowledge Portal - Liberia climate datasets

Liberia Institute of Statistics - Population and demographic data

Flask Documentation - Web framework for Python API development

Joblib Documentation - Model serialization and persistence

AWS/GCP Documentation - Cloud infrastructure deployment guides