AI-Powered Climate Risk Prediction and Early Warning System for Africa

1. Overview

The AI-Powered Climate Risk Prediction and Early Warning System for Africa was deployed to make climate-related insights accessible in real time to stakeholders such as farmers, policymakers, and disaster response agencies.  
The deployment process involved:

* Packaging the trained model for production using FastAPI and Docker.
* Hosting the model and API backend on AWS EC2 for scalability.
* Integrating the dashboard front-end (built with React) to visualize predictions and alerts.
* Establishing a CI/CD pipeline using GitHub Actions for automated deployment and updates.

The system provides early warnings on potential droughts, floods, and heatwaves using meteorological and satellite datasets.

1. Model Serialization

The trained model was serialized using Pickle (.pkl) for efficient loading during inference.  
Key considerations included:

* Converting all feature preprocessing pipelines into a scikit-learn Pipeline object before serialization to ensure reproducible predictions.
* Compressing the serialized file using gzip to reduce storage and network transmission time.
* Version-controlling serialized models through MLflow for easy rollback and model comparison.

Example:

*import joblib*

*joblib.dump(model, "climate\_risk\_model.pkl", compress=3)*

1. Model Serving

The serialized model is served through a FastAPI application, containerized using Docker for portability.  
Each container includes the model, inference scripts, and necessary dependencies.  
Deployment environment:

* Platform: AWS EC2 (Ubuntu 22.04 LTS)
* Container Runtime: Docker + Docker Compose
* Scaling: AWS Auto Scaling Group (future enhancement: Kubernetes)

The API exposes prediction endpoints that can process weather, soil, and satellite data inputs to generate climate risk scores.

1. API Integration

The model is integrated into an API that allows programmatic access for external systems and the dashboard.

Example Endpoints:

| Method | Endpoint | Description |
| --- | --- | --- |
| POST | /predict | Submits climate and environmental parameters to generate a risk prediction |
| GET | /health | Returns API health status |
| GET | /model-info | Returns version and metadata about the deployed model |

Input Format (JSON):

*{*

*"temperature": 33.5,*

*"humidity": 65.2,*

*"rainfall": 12.7,*

*"soil\_moisture": 0.45*

*}*

Response Format (JSON):

*{*

*"risk\_level": "High",*

*"predicted\_event": "Flood",*

*"confidence": 0.92*

*}*

The dashboard frontend connects to these endpoints for visualization and alert generation.

1. Security Considerations

Security measures implemented include:

* Authentication: API key-based access control for authorized users.
* Authorization: Role-based permissions for admin, analyst, and public users.
* Encryption: HTTPS with SSL/TLS certificates via AWS Certificate Manager.
* Network Security: Deployed behind AWS Security Groups and Network ACLs.
* Data Privacy: Sensitive meteorological data encrypted at rest using AWS KMS.

Future improvements: OAuth2 token-based authentication for multi-organization use.

1. Monitoring and Logging

To ensure reliability and accountability:

* Monitoring Tools: AWS CloudWatch and Prometheus track model latency, uptime, and resource usage.
* Logging: Application and inference logs stored in AWS S3 and analyzed via ELK (Elasticsearch–Logstash–Kibana) stack.
* Metrics Tracked:
* Model accuracy drift over time
* Response latency (ms)
* Request success rate (%)
* API uptime (%)
* Alerting: Automatic email and SMS alerts triggered when thresholds are exceeded (e.g., low accuracy or API downtime).

These measures ensure consistent model performance and timely system maintenance.