**Machine Learning Project 6**

**Group 7:**Members:   
  
1.Thomas S. McCay  
2. Mark Kekula  
3. David D. Samolu  
4. Francis T. Cole  
5. David Paye

**Deployment**

**1. Overview**

The deployment phase focused on transforming the malaria prediction model into a fully interactive and accessible web application.  
The aim was to enable **real-time malaria incidence prediction** based on environmental and geographical inputs (temperature, precipitation, location, etc.) and provide **live analytics dashboards** for monitoring system performance and prediction trends.

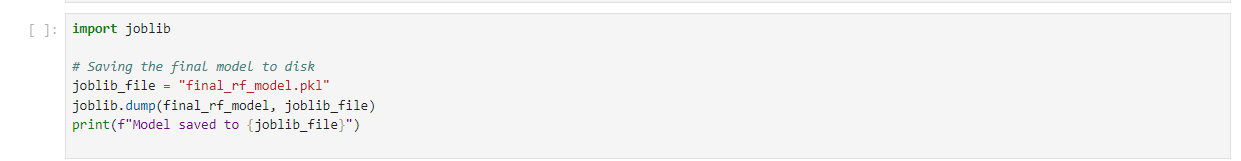
We deployed the application using **Streamlit Cloud**, chosen for its simplicity, scalability, and seamless integration with Python-based machine learning pipelines.

**Key Deployment Goals:**

* Make the trained Random Forest model accessible through a web interface.
* Enable real-time predictions for different African countries.
* Log each prediction request and display interactive analytics dashboards.
* Provide a user-friendly front-end that explains the project before users interact with it.

**2. Model Serialization**

The trained model was serialized using the **Joblib** library, which is optimized for models with large NumPy arrays (such as scikit-learn models).



**Format:** .pkl (Joblib binary)   
**Reason for Joblib:** It preserves large arrays efficiently and provides faster load times than Pickle for scikit-learn estimators.

**Storage:**   
The serialized model is included in the root directory of the deployed Streamlit app repository. This ensures the model is automatically loaded during app initialization.

**3. Model Serving**

Model serving was implemented via a **Streamlit app** that loads the serialized model once at startup and serves predictions in real-time.

### Workflow:

1. **User Input:**   
   Users select a country and enter environmental parameters such as rainfall, average/max/min temperature, longitude, and latitude.
2. **Model Inference:**  
   The inputs are converted into a NumPy array and passed into the final\_rf\_model for prediction:
3. Real**-Time Output:**   
   The predicted malaria incidence value is displayed instantly on the app interface with descriptive labels.
4. **Logging:**   
   Each prediction request (including country, timestamp, and input features) is automatically stored in a local CSV file (logs/prediction\_log.csv).

### Deployment Platform:

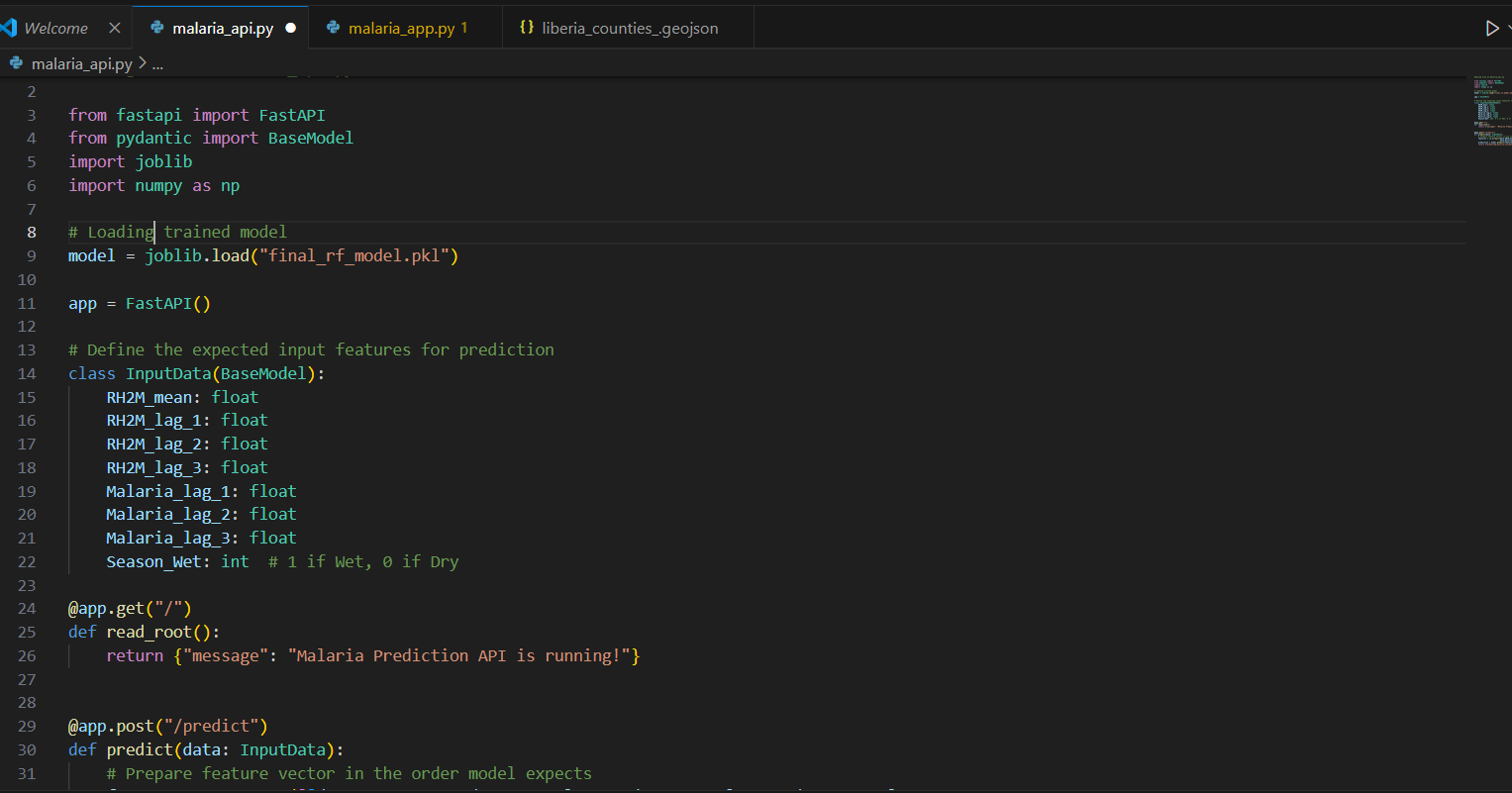
**Streamlit Cloud** — chosen for:

* Fast deployment directly from GitHub.
* Support for Python and machine learning libraries (scikit-learn, pandas, numpy, joblib).
* Built-in scalability and browser accessibility.
* Easy integration with Plotly for analytics dashboards.

**4. API Integration**

Although Streamlit primarily serves a front-end web app, the underlying model can also be exposed as a **RESTful API** if needed.

Currently, predictions are served directly within the Streamlit interface. However, the app can be extended to include an API endpoint using **FastAPI** or **Flask** for programmatic access.



This setup would allow third-party applications, health ministries, or research dashboards to integrate malaria predictions programmatically.

**5. Security Considerations**

During deployment, several key security considerations were taken into account:

|  |  |
| --- | --- |
| Security ASpect | Implementation |

Model Integrity --------------------The model file (final\_rf\_model.pkl) is stored read-only in the Streamlit app root directory to prevent tampering.

Input Validation ------------------User inputs (temperature, rainfall, latitude, etc.) are type-checked and range-limited to prevent invalid or malicious input.

File Safety --------------- ------- No external uploads are allowed. The system only processes numeric fields and pre-selected dropdowns.

Cloud Access --------------- Streamlit Cloud provides HTTPS encryption by default for all deployed apps, ensuring secure data transmission.

Future Enhancements ---------------- Authentication and rate-limiting can be added using an external API gateway for premium or restricted access.

**6. Monitoring and Logging**

A dedicated **Analytics and System Logs** page was implemented in Streamlit to track prediction history, trends, and system activity.

### Key Features:

* **Prediction Logs:**   
  Every prediction (timestamp, country, inputs, and predicted value) is recorded in a CSV log file.
* **Interactive Analytics Dashboard:**  
  Built with **Plotly**, showing:
  + Daily prediction trends (line charts)
  + Distribution of predicted incidence (histograms)
  + Most queried countries (bar charts)
  + Correlation matrix between environmental factors and predictions
* **Real-Time Visualization:**   
  Logs are automatically refreshed each time the user visits the analytics page.



### Metrics Tracked:

* Average predicted incidence (daily/monthly)
* Frequency of model use per country
* Correlation between inputs and model predictions
* System log size and health status

### Future Enhancements:

* Connect to **Google Cloud Storage**, **Firebase**, or **AWS S3** for persistent logging beyond Streamlit session lifetime.
* Integrate **alerting mechanisms** to notify admins if model drift or unusual prediction patterns are detected.

## ****Outcome****

The deployment transformed the machine learning workflow into a **publicly accessible intelligent malaria prediction tool**, empowering researchers, policymakers, and health officials to visualize and predict malaria risk across Africa.

**Technologies Used:**

* Python, Streamlit, Plotly
* Joblib, Pandas, Scikit-learn
* Streamlit Cloud (Deployment Platform)