Water Access Prediction Model
Deployment Phase Documentation

1 Overview

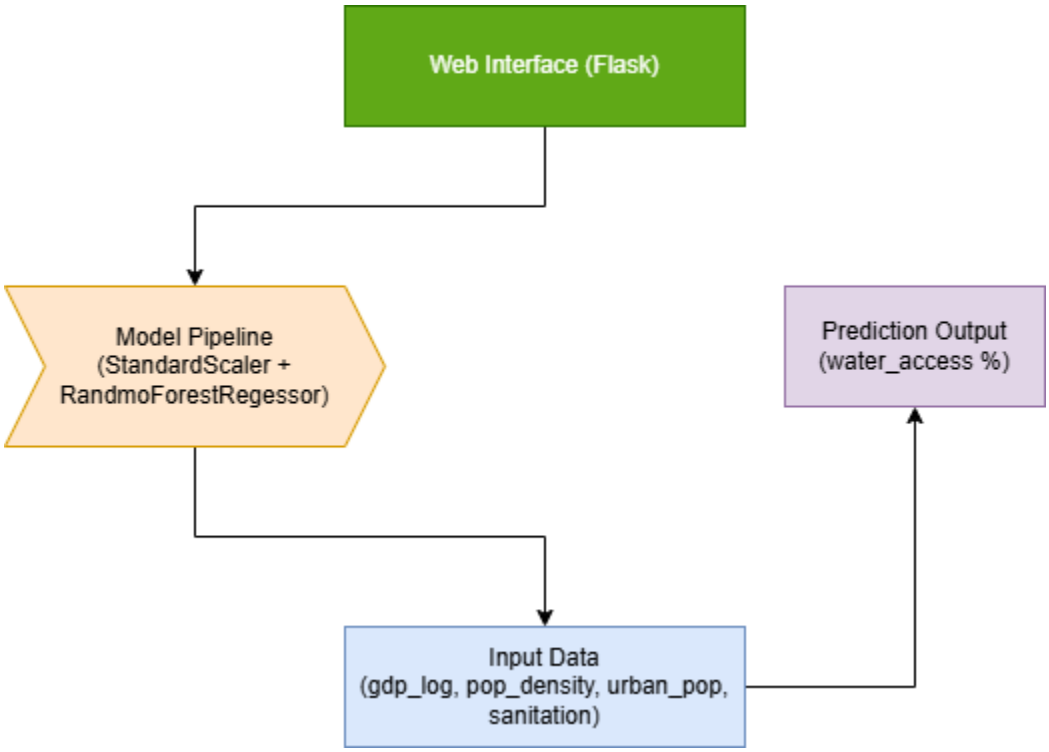
The deployment phase focuses on transforming a trained machine learning model into a production-ready, user-friendly application that can be used in real-world scenarios (e.g., policy planning, resource allocation). The goal is to ensure the model is:

- Accessible to end-users via a web interface.
- Consistent in preprocessing and prediction logic.
- Robust against invalid input and edge cases.
- Scalable for future enhancements or integration with larger systems.

Tools and Frameworks

Component	Tool
Model Format	Joblib for serialized pipeline
Web Framework	Flask (Python-based lightweight web framework)
Data Handling	Numpy for numerical operations
Error Handling	Flash messages, input validation, logging
Deployment Environment	Local(development)

2 Model Serialization



The trained model was serialized using joblib.dump(), which is optimized for scikit-learn models and handles large numpy arrays efficiently.

- **Pipeline Inclusion:** The serialized file includes:
    - **StandardScaler** (for consistent input preprocessing)
    - **RandomForestRegressor** (trained model with hyperparameters)
- **Efficient Storage:** joblib compresses arrays using Numpy's memory mapping, reducing file size.

3 Model Serving

Deployment Platform

The model was served via a Flask-based web application, hosted locally for development and can deploy on cloud platforms for production use.

Deployment Options

During the development of the model we used Local Flask server after finishing our work we plan to use Docker for Containerized deployment for consistency across en

Model Loading

```
# Load the full pipeline (scaler + model)
try:
    pipeline = joblib.load('optimized_water_access_model.joblib')
except Exception as e:
    print(f"Error loading pipeline: {e}")
    pipeline = None
```

4 API Integration

The model was integrated into a **RESTful API** for programmatic access.

Input and Output Format

```
# Get input values
# Change the number into whole number
gdp_log = float(requ
pop_density = float(request.form['pop_density'])
urban_pop = float(request.form['urban_pop'])
sanitation = float(request.form['sanitation'])

# Prepare input to make sure they got the same order as the training
input_data = np.array([[gdp_log, pop_density, urban_pop, sanitation]])
```

(variable) request: Request

```
# Get prediction from flash message
messages = get_flashed_messages()
prediction = messages[0] if messages else None
return render_template('index.html', prediction=prediction)
```

5 Security Considerations

Authentication and Authorization

- Secret Key: set in **app.secret_key** for flash message security.
- Input Validation: Prevents non-numeric inputs.

Data Privacy

- No Data Logging: Inputs are not stored or tracked.
- Anonymous Use: No user authentication required for basic