Rising Waters: A Machine Learning Approach to Flood Prediction

1. Introduction

Project Title: Rising Waters: A Machine Learning Approach to Flood Prediction

Team ID: LTVIP2026TMIDS65140

Team Size: 4

- **Team Leader**: Jillela Revanth Kumar - Project Lead & Model Developer
- **Team Member**: Thupakula Divya - Data Engineer
- **Team Member:** Chembedu Hema Latha - Frontend & Backend Developer
- **Team Member**: Bethu Hema Sri – Evaluation & Deployment Specialist

2. Project Overview

2.1. Purpose

Rising Waters is a machine learning-based flood prediction system designed to analyze environmental parameters such as rainfall, humidity, and water level to predict the likelihood of flood occurrence.

The main objective is to improve disaster preparedness, reduce risks, and provide early warning insights using predictive analytics

2.2. Features

- Machine Learning Based Prediction
- Real-Time User Input Analysis
- Simple Web Interface using Flask
- Fast Prediction Response
- Expandable for Weather API Integration
- User-Friendly Interface

3. Architecture

3.1. Frontend

- Developed using HTML5 and CSS3
- Provides input form for environmental parameters
- Displays prediction results dynamically using Flask

3.2. Backend

- Developed in Python using Flask Framework
- Loads trained model (model.pkl)
- Accepts user input
- Performs prediction using ML algorithm
- Returns results to frontend

Dataset:

- Rainfall
- Water Level
- Humidity
- Flood occurrence

Model

Logistic Regression / Random Forest Classifier

Model Training Pipeline:

- Data preprocessing using Pandas
- Train-test split
- Model training using Scikit-learn
- Model saved using Joblib (.pkl format)

3.3. Data Preprocessing:

- Handling missing values
- Feature scaling
- Dataset splitting

3.4. Database

No database used in this prototype. Predictions are generated in real time.

4. Setup Instructions

4.1. Prerequisites

- Python 3.8+
- Flask
- Scikit-learn
- Pandas
- NumPy
- Joblib

4.2. Installation

pip install -r requirements.txt
python app.py

5. Folder Structure

5.1. Client (Flask Frontend)

templates/

├── index.html

├── chance.html

5.2. Server (Flask Backend)

App.py

Model.pkl

Floods.ipynb

Requirements.txt

6. Running the Application

Python app.py
http://127.0.0.1:5000

7. API Documentation

This application primarily works through image upload via form, but internally uses:

Route: /
Method: POST

Input:

- Rainfall
- Water Level
- Humidity

Output:

Flood / No Flood prediction

8. Authentication

Not applicable – This is a prototype application.

9. User Interface

Two pages:

- index.html → Input page
- chance.html → Result page

UI Highlights:

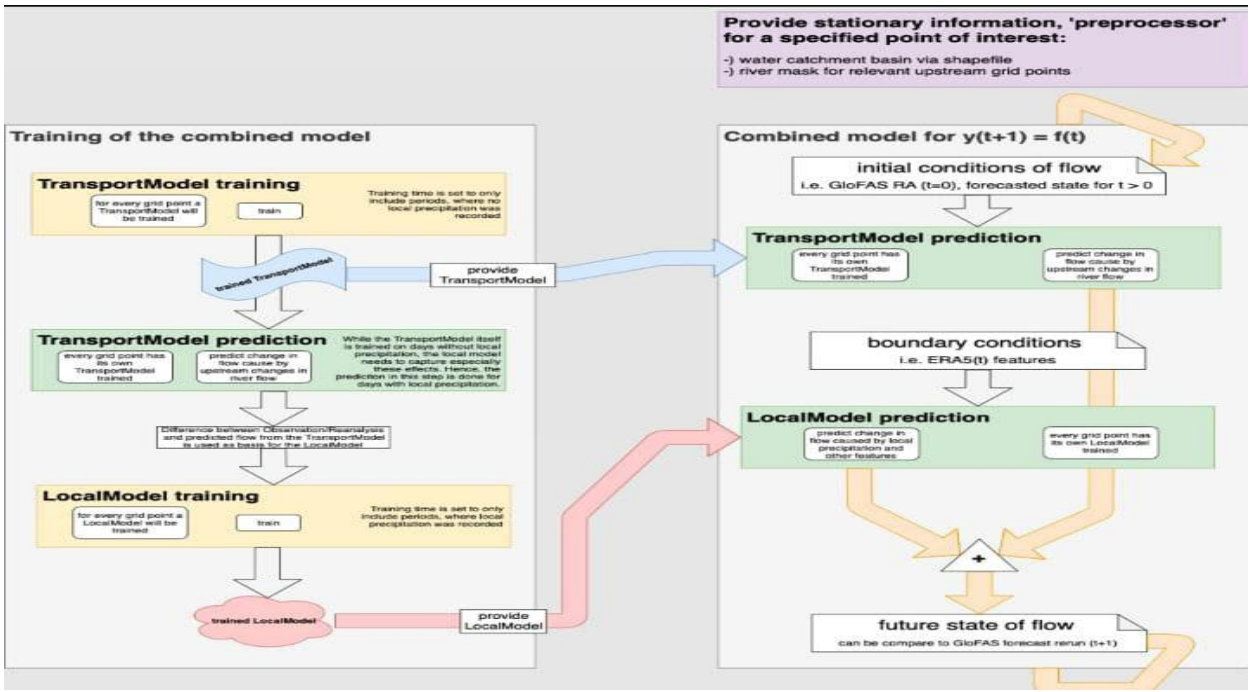
- Simple design
- Easy navigation
- Quick response

10. Testing

- Accuracy evaluation using test dataset
- Manual testing with multiple inputs
- Confusion matrix generated

11. Screenshots

- User enters environmental values
- System predicts flood risk
- Result displayed on screen



12. Known Issues

- Accuracy depends on dataset quality
- No live weather integration
- Limited dataset size

13. Future Enhancements

- Integration with weather APIs
- Deep learning models
- Mobile application
- Cloud deployment
- IoT sensor integration