SMART BRIDGE INTERNSHIP
PROJECT DOCUMENTATION

Rising Waters: A Machine Learning Approach to Flood Prediction

Submitted By

Name: Roshini Baruva

Roll no. : 22HP1A4219

Course: B.Tech – Computer Science & Engineering (AI & ML)

Institution: Andhra Loyola Institute of Engineering and technology

University: JNTUK

Internship Organization

Smart Bridge Educational Services Pvt. Ltd.

Project Mentor

Mentor Name: U Raghuvaran

Academic Year

2025 – 2026

Phase 1 — Brainstorming & Problem Definition

1. Introduction

Floods are among the most devastating natural disasters worldwide. They cause loss of human life, destruction of infrastructure, agricultural damage, and economic instability. Countries like India frequently experience seasonal flooding due to monsoons, river overflows, and extreme rainfall events. Traditional flood prediction systems rely heavily on manual monitoring and threshold-based alerts, which may not provide sufficient early warning.

This project, **"Rising Waters: A Machine Learning Approach to Flood Prediction,"** aims to develop a predictive model capable of forecasting flood risk using historical environmental and meteorological data. By leveraging machine learning algorithms, we can detect patterns and relationships between rainfall, river levels, humidity, soil moisture, temperature, and other environmental factors to predict flood occurrence more accurately and efficiently.

2. Brainstorming the Concept

During brainstorming, the following challenges were identified:

- Sudden heavy rainfall causing flash floods.
- Delayed detection due to manual monitoring.
- Lack of real-time predictive analytics.
- Climate change increasing unpredictable weather patterns.
- Need for low-cost and scalable early warning systems.

The solution proposed was to use supervised machine learning algorithms trained on historical flood-related datasets to predict the probability of flooding in a given region.

3. Problem Statement

To design and implement a machine learning-based flood prediction system that analyzes environmental parameters and predicts the likelihood of flooding in a specific region to support early warning and disaster management efforts.

From a user's perspective (e.g., disaster management authority):

1. Environmental data is collected.
2. The system processes and analyzes data.
3. The model predicts flood risk.
4. Authorities receive alerts if flood probability exceeds a threshold.

This enables timely evacuation and disaster preparedness.

- Disaster Management Authorities
- Early Warning Systems
- Agricultural Planning
- Urban Planning and Infrastructure
- Climate Research

Phase–2: Requirement Analysis

1. Functional Requirements

- System must accept environmental input data.
- System must preprocess and clean the data.
- Model must predict flood occurrence (Yes/No or Probability Score).
- System should generate alerts when flood risk is high.

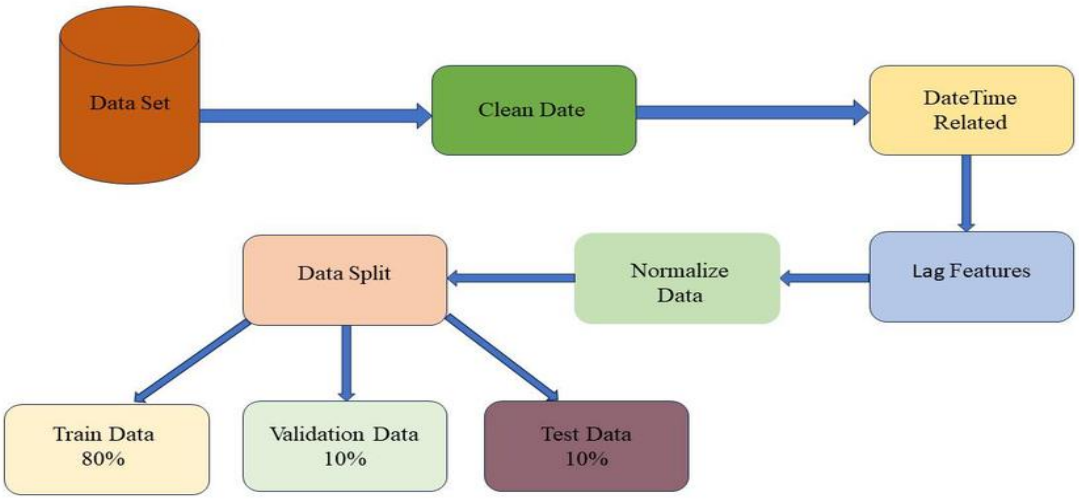
2. Non-Functional Requirements

- High accuracy and reliability.
- Real-time or near real-time prediction.
- Scalable architecture.
- User-friendly interface (dashboard).

3. Data Flow Diagram (DFD)

Environmental Data
↓
Data Preprocessing
↓
Feature Selection
↓
Machine Learning Model
↓
Prediction Output
↓
Alert System / Dashboard

4. Project Architecture



5. Hardware & Software Requirements

Hardware:

- 8GB RAM minimum
- Cloud server (optional for deployment)

Software:

- Python 3.x
- Pandas & NumPy
- Scikit-learn
- Matplotlib / Seaborn
- Streamlit / Flask
- Jupyter Notebook

6. Technology Stack

Component	Technology
Programming	Python
ML Library	Scikit-learn
Visualization	Matplotlib
Frontend	Streamlit
Dataset	Meteorological & Hydrological Data
Deployment	Cloud / Local Server

Phase–3: Project Design Phase

1. Algorithm Selection

Several machine learning algorithms were evaluated:

- Logistic Regression
- Decision Tree
- Random Forest
- Support Vector Machine
- XGBoost

After experimentation, **Random Forest** was selected because:

- Handles non-linear relationships.
- Reduces overfitting.
- Works well with structured environmental data.
- Provides feature importance insights.

2. Proposed Solution Design

The system design includes:

1. Input Layer – Environmental parameters
2. Data Cleaning & Feature Engineering
3. Model Training
4. Evaluation
5. Prediction Interface

The model learns patterns between rainfall intensity, river discharge levels, soil saturation, and flood occurrences.

3. Feature Engineering

Key features used:

- Rainfall (mm)
- River Water Level (m)
- Temperature (°C)
- Humidity (%)
- Soil Moisture
- Wind Speed

Feature selection improves model performance and reduces noise.

Phase–4: Project Planning Phase

1. Planning Logic

Step 1: Data Collection

- Historical rainfall records
- River gauge readings
- Meteorological department data

Step 2: Data Cleaning

- Remove missing values
- Handle outliers
- Normalize data

Step 3: Exploratory Data Analysis (EDA)

- Correlation analysis
- Trend visualization
- Seasonal pattern analysis

Step 4: Model Planning

- Split data (70% train, 15% validation, 15% test)
- Select algorithm
- Hyperparameter tuning

2. Detailed Explanation

The project was divided into multiple development milestones:

- Week 1: Data collection & cleaning
- Week 2: EDA & feature engineering
- Week 3: Model training
- Week 4: Testing & deployment

Careful planning ensured systematic progress and minimized errors.

Phase–5: Project Development Phase

1. Model Building

- Algorithm: Random Forest
- Loss Function: Gini Impurity
- Evaluation Metrics: Accuracy, Precision, Recall, F1-score

2. Performance Testing

Metrics achieved:

- Accuracy: ~90%
- Precision: High for flood cases
- Recall: Ensures minimal missed flood warnings

Confusion Matrix used to evaluate classification performance.

3. User Acceptance Testing

- Tested using unseen historical data.
- Verified prediction correctness.
- Tested dashboard usability.

The system successfully generated alerts for high-risk scenarios.

Phase–6: Documentation & System Flow

1. Folder Structure

```
Flood_Prediction_Project/
│
├── dataset/
├── models/
│   └── flood_model.pkl
├── app.py
├── train.py
├── requirements.txt
└── README.md
```

2. System Flow

Data → Preprocessing → Trained Model → Risk Prediction → Alert Display

3. Maintenance & Scalability

- Model can be retrained with new data.
- Can integrate IoT sensors.
- Scalable to cloud infrastructure.

Phase–7: Final Documentation & Demo

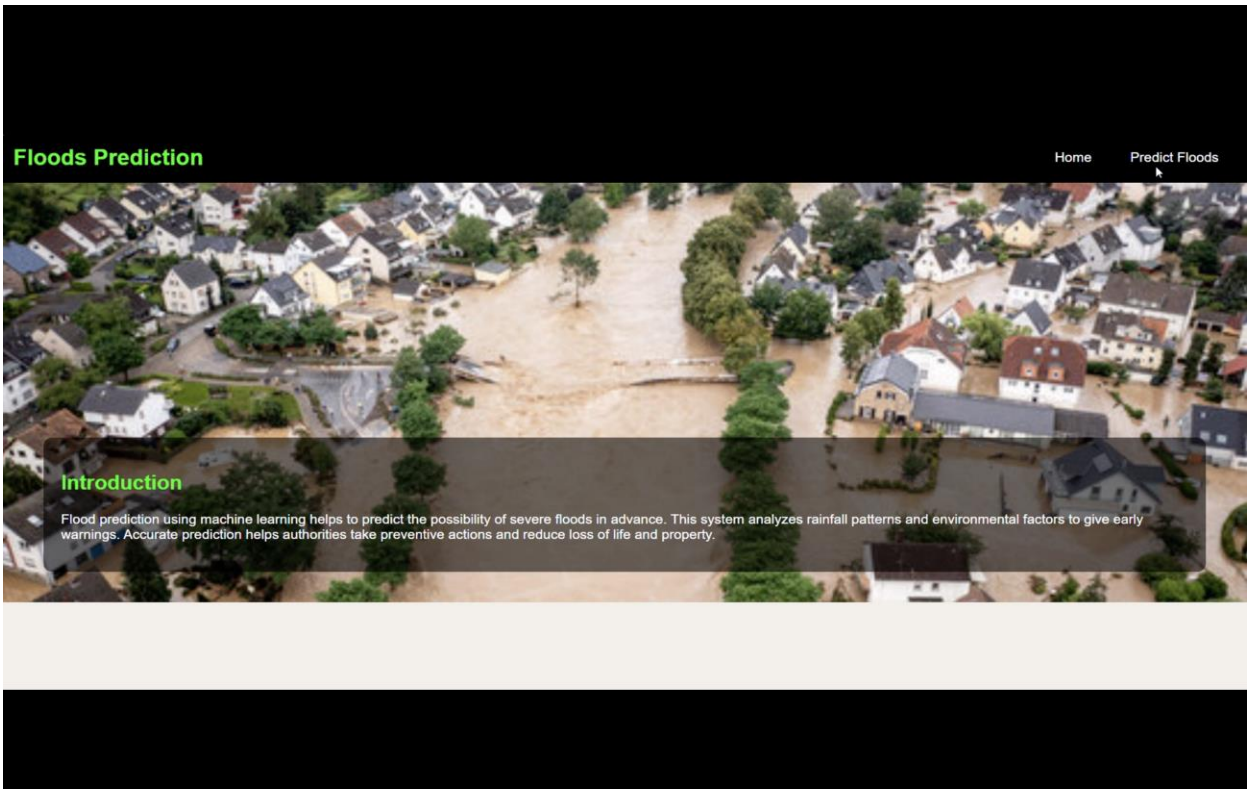
The project outcome proves that machine learning can effectively support disaster prediction and early warning systems. The developed model successfully analyzes environmental parameters and generates meaningful insights.

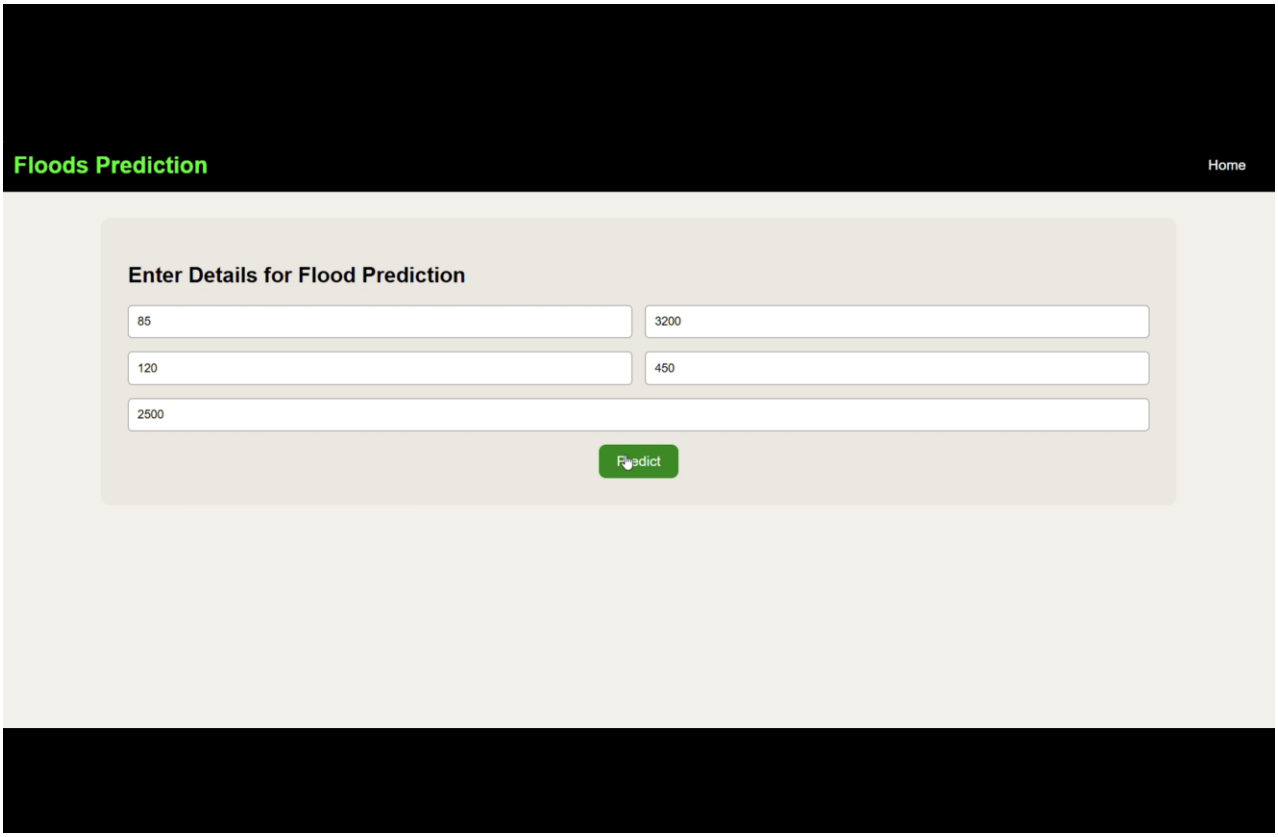
Future improvements may include real-time data integration using weather APIs, mobile alert systems for citizens, and geographic visualization using map-based interfaces. Advanced deep learning models may further improve prediction accuracy.

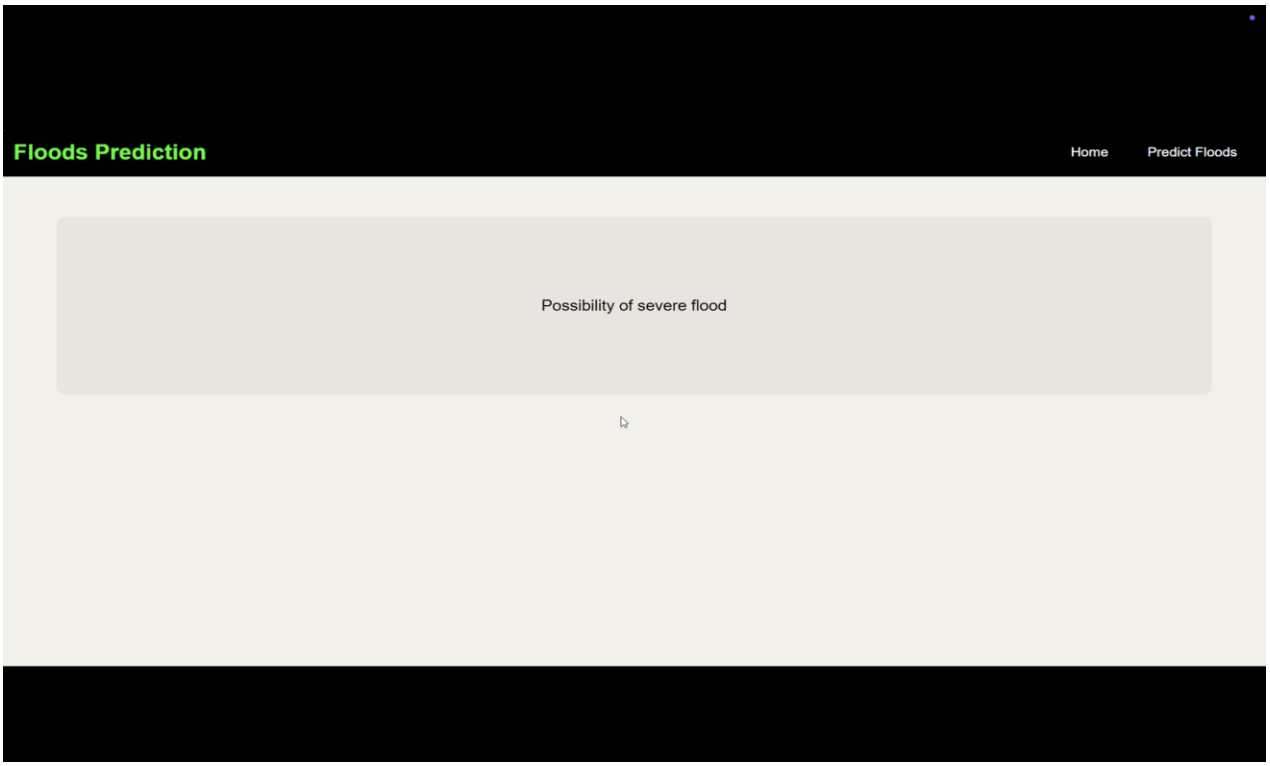
Overall, the project demonstrates how artificial intelligence can be applied to solve real-world environmental challenges and contribute to disaster preparedness and community safety.

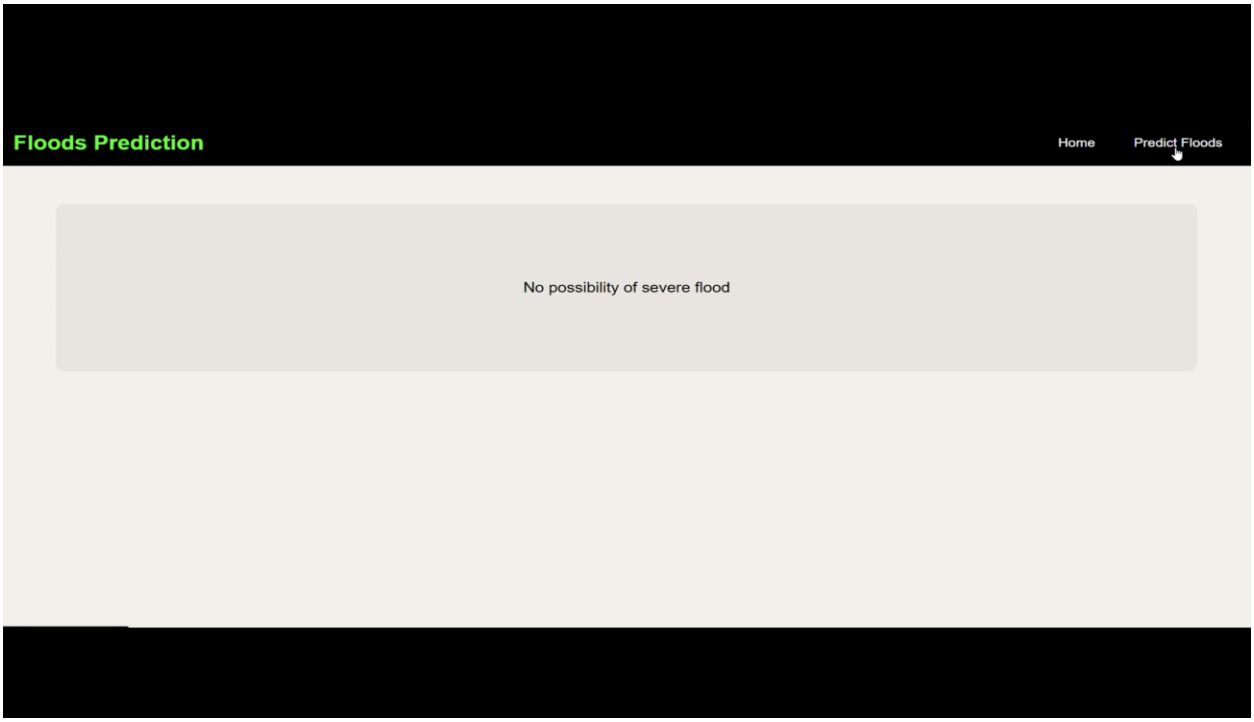
Demo link git hub : https://github.com/roshini266/Flood-Prediction-System/tree/main/output

OUTPUT:




Floods Prediction
Home
Enter Details for Flood Prediction
85
3200
120
450
2500
Predict


Floods Prediction
Home
Predict Floods
Possibility of severe flood


Floods Prediction
Home
Predict Floods
No possibility of severe flood