1. INTRODUCTION

1.1 Project Overview

Rising Waters: A Machine Learning Approach to Flood Prediction is a predictive analytics project that uses historical weather and rainfall indicators to classify whether flood conditions are likely to occur. The system is built by training supervised machine learning models on a structured dataset containing meteorological variables such as temperature, humidity, cloud cover, and aggregated rainfall values across seasons (e.g., Jan–Feb, Mar–May, Jun–Sep, Oct–Dec). After training, the best-performing model is saved and integrated into a user-friendly Flask web application, where users can enter input values and instantly receive a flood-risk prediction. This end-to-end pipeline includes data loading, preprocessing, feature scaling, model training and evaluation, model serialization, and deployment through a simple web interface.

1.2 Purpose

The purpose of this project is to provide an early, data-driven indication of flood risk using machine learning so that decision-makers and users can take timely preventive actions. Floods cause major damage to life, infrastructure, agriculture, and the economy, and accurate early prediction plays a crucial role in preparedness and mitigation. By learning patterns from historical environmental conditions and rainfall distribution, the project aims to reduce reliance on manual judgment and enable fast, consistent predictions. Additionally, the project demonstrates a complete machine learning workflow—from model development to deployment—making it valuable both as a practical flood-risk tool and as a real-world demonstration of applied ML engineering.

Problem Statement

Team ID: LTVIP2026TMIDS47616

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

Problem Statements (Customer-Centric)

The following table captures two key customer problem statements aligned with the project scope.

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	a district/city disaster management officer	predict flood risk early so I can issue warnings and deploy resources in advance	I do not have a fast, consistent, data-driven way to estimate flood likelihood from changing weather and rainfall conditions	manual assessment is slow and forecasts don't translate clearly into actionable flood-risk decisions	uncertain and pressured because delayed action can lead to avoidable damage and safety risks
PS-2	a resident living in a flood-prone area	understand whether current weather and rainfall conditions could lead to flooding soon so I can take precautions	I cannot easily interpret weather patterns or rainfall summaries into a clear flood-risk decision	information is scattered and not presented as a simple "risk/no risk" output for my locality	anxious and unprepared because I may not get enough time to protect my family and property

Empathy Map Canvas

Team ID: LTVIP2026TMIDS47616

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

User Persona: District/City Disaster Management Officer (Flood Response & Preparedness)

SAYS	THINKS
• "I need a clear yes/no risk signal, not just raw rainfall numbers." • "We need quicker decisions for alerts and resource deployment." • "Give me something simple I can explain to officials and the public." • "I need early indicators to prepare shelters and evacuation routes."	• "If we miss a flood warning, the consequences are huge." • "Are these forecasts reliable enough to act on?" • "I want a tool that is consistent and reduces manual judgment." • "How do I prioritize which areas to respond to first?"
DOES	**FEELS**

DOES	FEELS
• Monitors rainfall reports, weather updates, and past flood patterns. • Coordinates with local agencies for warnings, evacuation, and relief camps. • Uses checklists/manual thresholds and communicates alerts. • Reviews historical incident reports after events.	• Responsible and pressured (high-stakes decisions). • Anxious during extreme weather events. • Frustrated when information is scattered or unclear. • Wants confidence and clarity before issuing public warnings.

Pain Points (Challenges)

• Flood risk assessment is often manual and time-consuming.

• Weather/rainfall data is available but not converted into a clear actionable decision.

• False alarms reduce public trust; missed alerts cause severe damage.

• Difficult to prioritize response across multiple regions quickly.

Gains (Needs / Expected Outcomes)

• A fast, consistent flood prediction from key weather + rainfall indicators.

- Simple outputs like “Flood Chance / No Flood Chance” for quick decision-making.
- Better preparedness: earlier alerts, evacuation planning, and resource allocation.
- An easy web tool that can be used by non-technical staff.

Ideation Phase – Brainstorm & Idea Prioritization

Date: 16 February 2026

Team ID: LTVIP2026TMIDS47616

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

Step-1: Team Gathering, Collaboration and Select the Problem Statement

Selected Problem Statement (PS-1):

• I am a district/city disaster management officer, I'm trying to predict flood risk early so I can issue warnings and deploy resources in advance, but I do not have a fast, consistent, data-driven way to estimate flood likelihood from changing weather and rainfall conditions because manual assessment is slow and forecasts don't translate clearly into actionable flood-risk decisions, which makes me feel uncertain and pressured.

Step-2: Brainstorm, Idea Listing and Grouping

A) Data & Features

• Use seasonal rainfall aggregates (Jan–Feb, Mar–May, Jun–Sep, Oct–Dec) plus temperature, humidity, cloud cover as core predictors.

• Add rolling/derived indicators (e.g., rainfall deviation from average, monsoon intensity index).

• Handle class imbalance with stratified split and (optional) class weights or sampling methods.

B) Modeling & Evaluation

• Train multiple models (Decision Tree, Random Forest, KNN, XGBoost) and compare accuracy + confusion matrix.

• Tune decision threshold to reduce false negatives (missed floods) even if false positives rise slightly.

• Use additional metrics (precision, recall, F1) to justify model selection beyond accuracy.

C) Deployment & Integration

• Serialize best model and scaler (floods.save, transform.save) for consistent inference.

• Build Flask web app for quick input → prediction → result page (Flood Chance / No Flood Chance).

• Provide simple API endpoint (/predict) for integration with other dashboards or alert systems.

D) UX / Decision Support

• Modern, clean UI with grouped inputs, validation, and clear result screens (risk vs no risk).

• Display model confidence (probability %) to support decision-making.

• Include a 'Use Sample Values' button for quick testing and demo during evaluation.

E) Operations & Safety

• Log inputs and predictions for monitoring, auditing, and future model improvements.

• Add a disclaimer and guidance steps for 'High Risk' output (monitor alerts, prepare resources).

• Plan periodic retraining when new data becomes available.

Step-3: Idea Prioritization

Prioritization method: Impact vs Effort (High/Medium/Low).

Idea	Impact	Effort	Priority Decision
Deploy Flask UI for instant prediction (form → result pages)	High	Medium	High Priority
Save scaler + model for consistent inference (transform.save, floods.save)	High	Low	High Priority
Train and compare multiple ML models; select best with recall focus	High	Medium	High Priority
Show probability/confidence score to support decisions	Medium	Medium	Medium Priority
Logging of predictions + inputs for monitoring	Medium	Low	Medium Priority

Advanced feature engineering (deviation indices, rolling indicators)	Medium	High	Lower Priority
Scheduled retraining pipeline	Low	High	Lower Priority

Final shortlisted solution for implementation:

• End-to-end ML pipeline (data loading → preprocessing → scaling → model training/evaluation).

• Best model selection with emphasis on reducing missed floods.

• Model + scaler serialization for deployment.

• Flask-based web application with a modern UI to generate flood-risk predictions.

Customer Journey Map (CJM)

Team ID: LTVIP2026TMIDS47616

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

Primary Persona: District/City Disaster Management Officer

Scenario: Using the Flood Prediction Web App to assess risk, issue warnings, and review outcomes

This Customer Journey Map follows the structure of the reference example (steps from awareness to post-use improvement) and captures what the primary user experiences, their interactions, emotions, and improvement opportunities.

Journey elements	**Entice (Awareness)**	**Enter (Access)**	**Engage (Use & Decide)**	**Exit (Action Taken)**	**Extend (Follow-up & Improve)**
Steps (What happens?)	Learns about the tool through internal circulars, prior flood events, or demo by IT team.	Opens web app; prepares latest weather/rainfall values (or receives them from field staff).	Inputs values; clicks Predict; views risk result (Flood Chance / No Flood Chance).	If high risk, issues warnings, mobilizes resources, and communicates to stakeholders.	Reviews outcome after event; logs results; requests updates or retraining improvements.
Interactions (Things/Places/People)	Emails/meetings; coordination with IT/analytics team; disaster management briefing room.	Laptop/phone; web browser; office control room; may call field officers for data.	Flask web UI; model output page; communicates with officials for approval if needed.	SMS/calls/public alerts; coordination with police/health/shelter teams; field deployment.	Debrief meeting; data team; update dataset/model; documentation and reporting.
Goals & motivations (Help me...)	Help me discover an easier way to estimate flood risk.	Help me access the tool quickly and get the right input data.	Help me get a clear, reliable risk decision fast.	Help me take timely action to minimize harm and losses.	Help me learn from outcomes and improve future preparedness.
Positive moments	Confidence that the process can	Quick access and simple form; clear	Instant result; clear classification; can	Early warning enables better preparedness;	Better reporting and

	be standardized; reduces manual guesswork.	input fields.	justify decision with data-driven output.	smoother coordination.	continuous improvement; improved trust over time.
Negative moments	Skepticism about ML reliability; uncertainty about data quality.	Missing/late input values; confusion about units; connectivity issues.	Result without confidence score; worry about false negatives; borderline cases.	Public trust damage if false alarm; high pressure during emergencies.	Lack of logs/feedback loop; limited historical coverage for some regions.
Areas of opportunity	Short demo + one-page guide; clarify data requirements and when to use tool.	Auto-fill sample/test values; input validation; tooltips for each field.	Show probability/confidence; highlight top contributing features; threshold tuning for recall.	Integrate alert checklist and recommended next actions on High Risk screen.	Store predictions & outcomes; periodic retraining; dashboard for trends and model monitoring.

Note: The journey emphasizes reducing missed floods (false negatives) by improving confidence display, threshold tuning, and integrating recommended actions.

Solution Requirements (Functional & Non-functional)

Date: 16 February 2026

Team ID: LTVIP2026TMIDS47616

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

Functional Requirements

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Data Ingestion	Load dataset (Excel) from Dataset folder Validate required columns (features + flood label) Handle basic data quality checks (nulls, data types)
FR-2	Data Preprocessing	Split features (X) and label (y) Train-test split with fixed random state for reproducibility Standardize input features using StandardScaler Persist scaler for deployment (transform.save)
FR-3	Model Training & Selection	Train multiple classifiers (Decision Tree, Random Forest, KNN, XGBoost) Compare model performance on test set (accuracy + confusion matrix) Select best model based on evaluation results
FR-4	Model Evaluation	Generate confusion matrix and classification report (precision, recall, F1) Highlight missed floods (false negatives) as critical cases Store evaluation summary for reporting
FR-5	Model Serialization	Save selected trained model for inference (floods.save) Ensure saved artifacts are reusable without retraining Load saved model and scaler during app startup
FR-6	Web Application (Flask UI)	Provide input form for required feature fields

		Validate numeric inputs and handle errors gracefully Run prediction using saved model + scaler
FR-7	Result Presentation	Display user-friendly output pages: Flood Chance / No Flood Chance Provide navigation back to input screen Optional: show confidence/probability score (future enhancement)
FR-8	Maintainability & Updates	Allow replacing model/scaler files without code changes Document steps for retraining and redeploying updated artifacts Maintain consistent feature order between training and inference

Non-functional Requirements

Following are the non-functional requirements of the proposed solution.

NFR No.	**Non-Functional Requirement**	**Description**
NFR-1	Usability	UI should be simple and intuitive; input labels must be clear; results must be displayed in an easily understandable format.
NFR-2	Security	Validate and sanitize user inputs; do not expose internal file paths or stack traces in production; restrict debug mode in deployment.
NFR-3	Reliability	Prediction must consistently use the same preprocessing (saved scaler) and model artifact; handle missing/invalid inputs gracefully.
NFR-4	Performance	Prediction should complete within a few seconds on a standard laptop; application should remain responsive under normal usage.
NFR-5	Availability	Web app should be available during usage windows; restart should restore service quickly since model/scaler are stored locally.
NFR-6	Scalability	Design should allow future scaling (e.g., hosting on a server or container); code should support API integration and more concurrent users.

Data Flow Diagram (DFD)

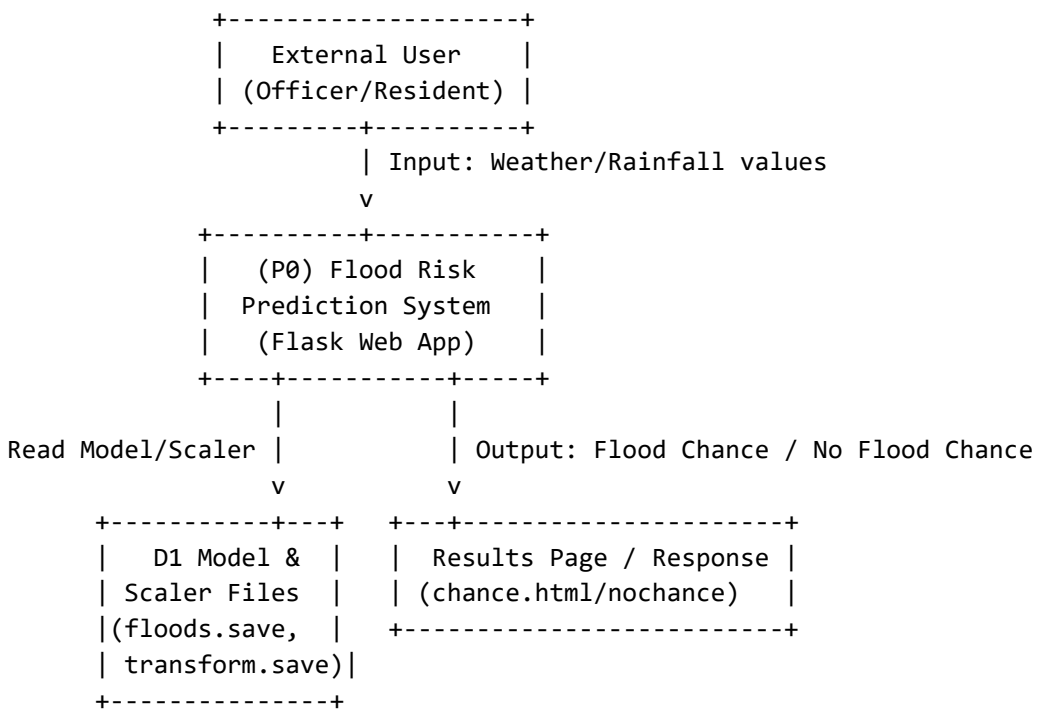
Date: 16 February 2026

Team ID: LTVIP2026TMIDS47616

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

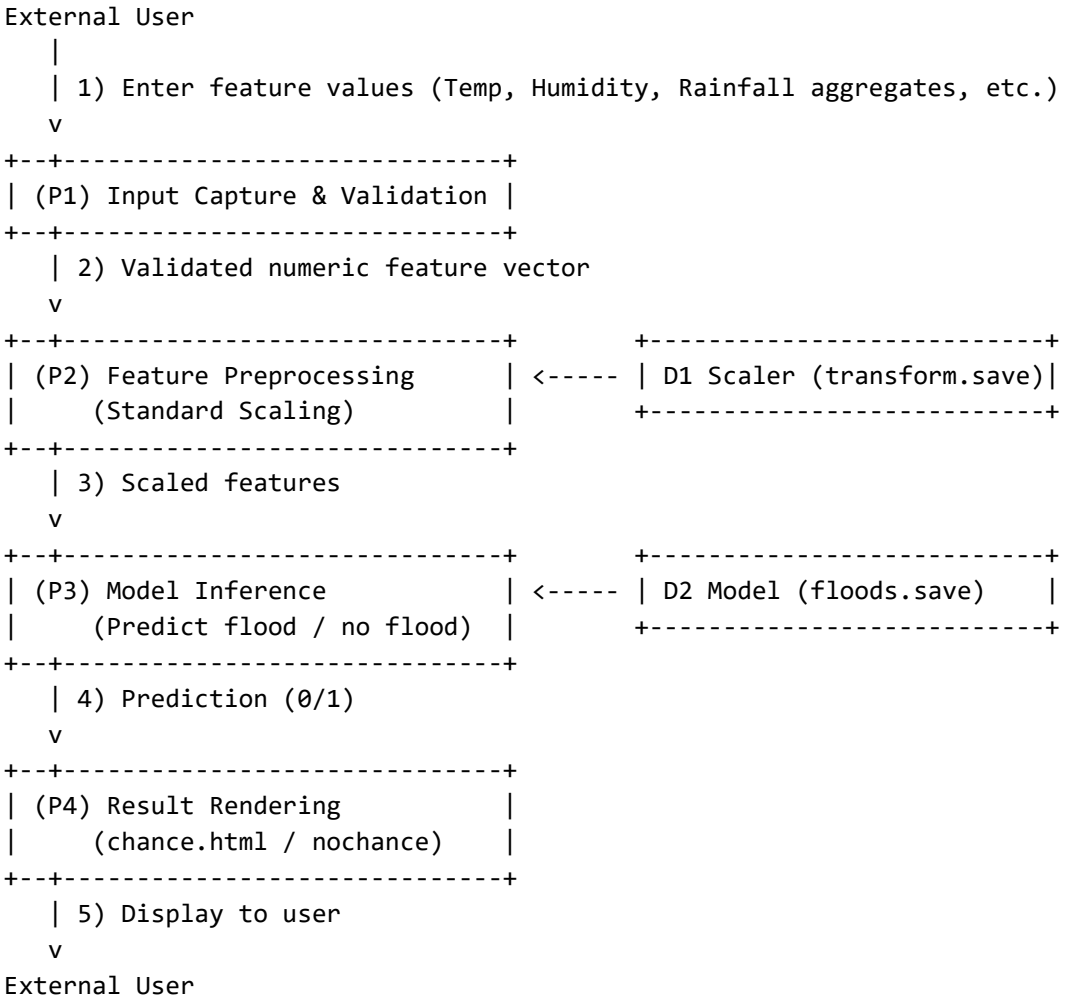
A Data Flow Diagram (DFD) represents how data moves through the flood prediction system, what processes transform the data, and where the data is stored. This document provides a Level 0 (Context) and a Level 1 (Detailed) DFD for the proposed solution.

DFD Level 0 (Context Diagram)



Level 0 shows the system as a single process (P0). The user provides input values via the web UI. The system loads the saved model and scaler from storage (D1) to generate and display the prediction result.

DFD Level 1 (Detailed Diagram)



Offline Training Flow (Model Build):
D3 Flood Dataset (Excel) -> (P5) Train/Test Split + Scaling -> (P6) Train Models -> (P7) Evaluate -> Save to D1 & D2

Level 1 decomposes the prediction system into four main processes: input capture/validation (P1), preprocessing using the saved scaler (P2), model inference using the saved classifier (P3), and result rendering (P4). An offline training flow creates the model and scaler artifacts from the dataset.

Data Stores and Key Data Elements

D1 Scaler File: transform.save – stores StandardScaler parameters (mean and variance) used during training.

D2 Model File: floods.save – stores the best-performing trained classification model (e.g., Decision Tree).

D3 Dataset: flood dataset.xlsx – contains historical feature values and the target label (flood).

Input Features: Temp, Humidity, Cloud Cover, ANNUAL, Jan-Feb, Mar-May, Jun-Sep, Oct-Dec, avgjune, sub.

Output: Binary prediction: 1 = Flood Chance, 0 = No Flood Chance (displayed on result pages).

Technology Stack (Architecture & Stack)

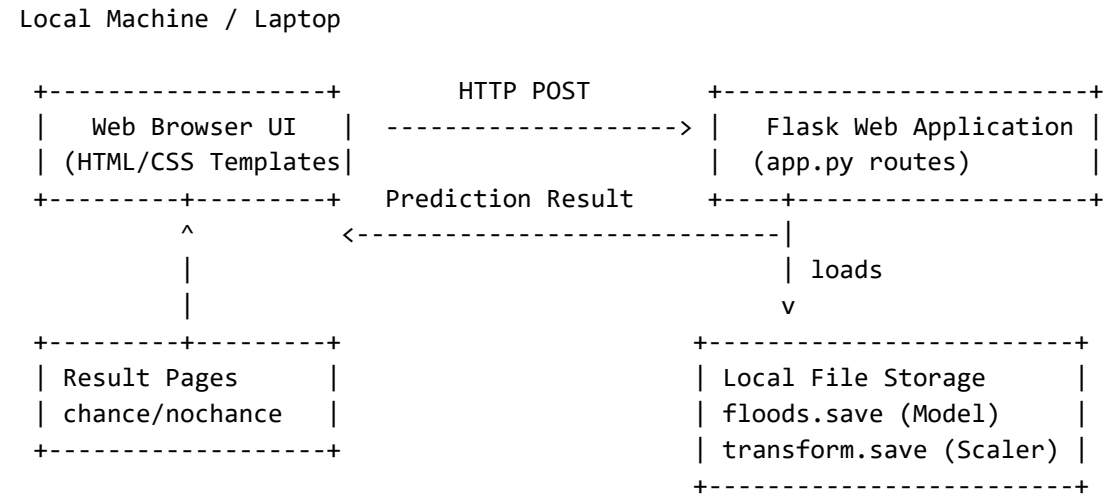
Date: 16 February 2026

Team ID: LTVIP2026TMIDS47616

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

Technical Architecture (High-Level)

The application follows a simple 3-tier flow: Web UI → Flask Application (Inference API) → Saved ML artifacts (model + scaler). Training is performed offline using Python and scikit-learn, then the trained model is serialized and used by the Flask app for real-time prediction.



Infrastructure demarcation:

• Local: Training (Python), model artifacts (files), Flask web app, and UI templates.

• Cloud (optional future): Host Flask app on IBM Cloud / AWS / Azure; store artifacts in cloud storage; add a real-time weather API.

Table-1: Components & Technologies

S.No	Component	Description	Technology
1	User Interface	Web form where user enters feature values and views prediction results	HTML5, CSS3 (modern styling), Jinja2 templates
2	Application Logic-1	Routes for home page and prediction endpoint; handles input validation and response rendering	Python 3.12, Flask
3	Application Logic-2	Preprocessing pipeline for inference using same scaling as training	scikit-learn StandardScaler (joblib)
4	Application Logic-3	Model inference logic (predict flood / no flood)	scikit-learn classifier (Decision Tree / RandomForest / XGBoost)
5	Database	Not required for baseline guided project; could store logs/predictions in future	N/A (optional: SQLite/PostgreSQL)
6	Cloud Database	Optional for cloud deployment to store predictions/audit trail	IBM Cloudant / IBM Db2 (optional)
7	File Storage	Stores dataset (Excel) and serialized artifacts for inference	Local filesystem; pickle + joblib files
8	External API-1	Optional: Real-time weather/rainfall input for automation	IBM Weather API / OpenWeather API (future scope)
9	External API-2	Optional: Disaster alert dissemination (SMS/email)	SMS Gateway / Email service (future scope)
10	Machine Learning Model	Binary classification model to predict flood chance from weather/rainfall features	DecisionTreeClassifier / RandomForestClassifier / XGBoost
11	Infrastructure (Server / Cloud)	Runs locally for demo; can be deployed to cloud as web service	Local PC; optional IBM Cloud Foundry / Docker / Kubernetes

Table-2: Application Characteristics

S.No	Characteristics	Description	Technology
1	Open-Source Frameworks	Libraries used for data processing, ML training, and web deployment	pandas, numpy, scikit-learn, xgboost, Flask, matplotlib, seaborn
2	Security Implementations	Input validation to prevent bad values; disable debug mode in production; basic safe deployment practices	Flask request validation; OWASP best practices (deployment)
3	Scalable Architecture	Separation of UI, app logic, and model artifacts; can be containerized and scaled	3-tier approach; Docker (optional); Kubernetes (optional)

		horizontally	
4	Availability	Local demo availability; cloud hosting can add higher availability with restart policies/load balancing	Local run; optional cloud deployment + load balancer
5	Performance	Low-latency inference since model is preloaded; scaling is fast; responses in milliseconds to seconds	In-memory model; scikit-learn inference; caching optional

PROBLEM – SOLUTION FIT CANVAS

Rising Waters: A Machine Learning Approach to Flood Prediction

Team ID: LTVIP2026TMIDS47616 | Date: 16 Feb 2026

1. CUSTOMER SEGMENT(S)	2. JOBS-TO-BE-DONE / PROBLEMS
• Primary: District/City Disaster Management Officer (control room) • Secondary: Residents in flood-prone areas • Tertiary: Municipal authorities / local administration	• Predict flood risk early from weather & rainfall indicators • Decide when to issue alerts and mobilize resources • Convert raw data into a clear 'Flood / No Flood' decision
3. TRIGGERS	**4. EMOTIONS (BEFORE / AFTER)**
• Heavy/continuous rainfall (monsoon peak, cyclone warnings) • High seasonal rainfall aggregates (Jun–Sep) and cloud cover rise • Past flood history / field reports from low-lying areas	• Before: anxious, uncertain, pressured (high stakes) • After: more confident, faster decisions, better preparedness
5. AVAILABLE SOLUTIONS	**6. CUSTOMER CONSTRAINTS**
• Manual assessment and experience-based judgment • Weather forecast dashboards and reports • Simple rainfall threshold rules (rain > X mm)	• Needs instant output during emergencies • Non-technical users; UI must be simple • Limited compute/IT; runs on a normal laptop • High cost of missed floods (false negatives)
7. BEHAVIOUR	**8. CHANNELS OF BEHAVIOUR**
• Monitors weather/rainfall and local field updates • Coordinates with agencies for warnings and evacuation • Uses SOPs/checklists; communicates alerts to public	• Online: dashboards, WhatsApp/email groups, web tools • Offline: control room briefings, calls, field officer reports
9. PROBLEM ROOT CAUSE	**10. YOUR SOLUTION**
• Flood depends on multiple interacting factors (not rainfall alone) • Gap between raw indicators and actionable decision • Manual decisions are inconsistent under pressure	• Train ML classifier using historical weather/rainfall features • Standardize inputs with saved scaler (transform.save) • Predict with saved model (floods.save) → Flood Chance / No Flood Chance • Deploy via Flask web app with modern UI for quick inference

Note: The implementation prioritizes reducing missed floods (false negatives) and delivering quick, clear decisions via a web-based interface.

Proposed Solution

Date: 16 February 2026

Team ID: LTVIP2026TMIDS47616

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

S.No.	Parameter	Description
1	Problem Statement (Problem to be solved)	Flood preparedness often depends on manual assessment of rainfall and weather conditions, which can be slow and inconsistent. Decision makers need a fast, reliable, and easy-to-use system that converts meteorological indicators into a clear flood-risk decision to enable timely warnings and resource deployment.
2	Idea / Solution description	Develop an end-to-end machine learning system that predicts flood likelihood (Flood Chance / No Flood Chance) using historical features such as temperature, humidity, cloud cover, annual rainfall, and seasonal rainfall aggregates. The solution includes: (i) data preprocessing and standard scaling, (ii) training and comparing multiple classifiers (Decision Tree, Random Forest, KNN, XGBoost), (iii) selecting and saving the best model and scaler, and (iv) deploying a Flask-based web app where users input values and get instant predictions.
3	Novelty / Uniqueness	Unlike simple threshold rules (e.g., rainfall > X), the proposed approach learns combined patterns across multiple weather and rainfall indicators. It packages the full ML workflow (training → evaluation → deployment) into a lightweight web app that non-technical staff can use quickly during emergencies.
4	Social Impact / Customer	Improves disaster preparedness

	Satisfaction	by enabling earlier alerts, better evacuation planning, and faster coordination. By reducing delays and standardizing decisions, the solution can help minimize loss of life and property and improve public confidence in warning systems.
5	Business Model (Revenue Model)	Primarily a public-safety solution that can be adopted by government bodies. Potential revenue options include subscription-based licensing for municipalities, custom deployment/integration services for agencies, and enterprise dashboards for organizations managing infrastructure (utilities, transport, insurance).
6	Scalability of the Solution	The design is modular and scalable: models can be retrained with new data, the Flask app can be hosted on a server/cloud, and the prediction endpoint can be integrated into larger alerting platforms. The solution can expand to multiple regions by incorporating region-specific datasets and additional real-time data sources.

Solution Architecture

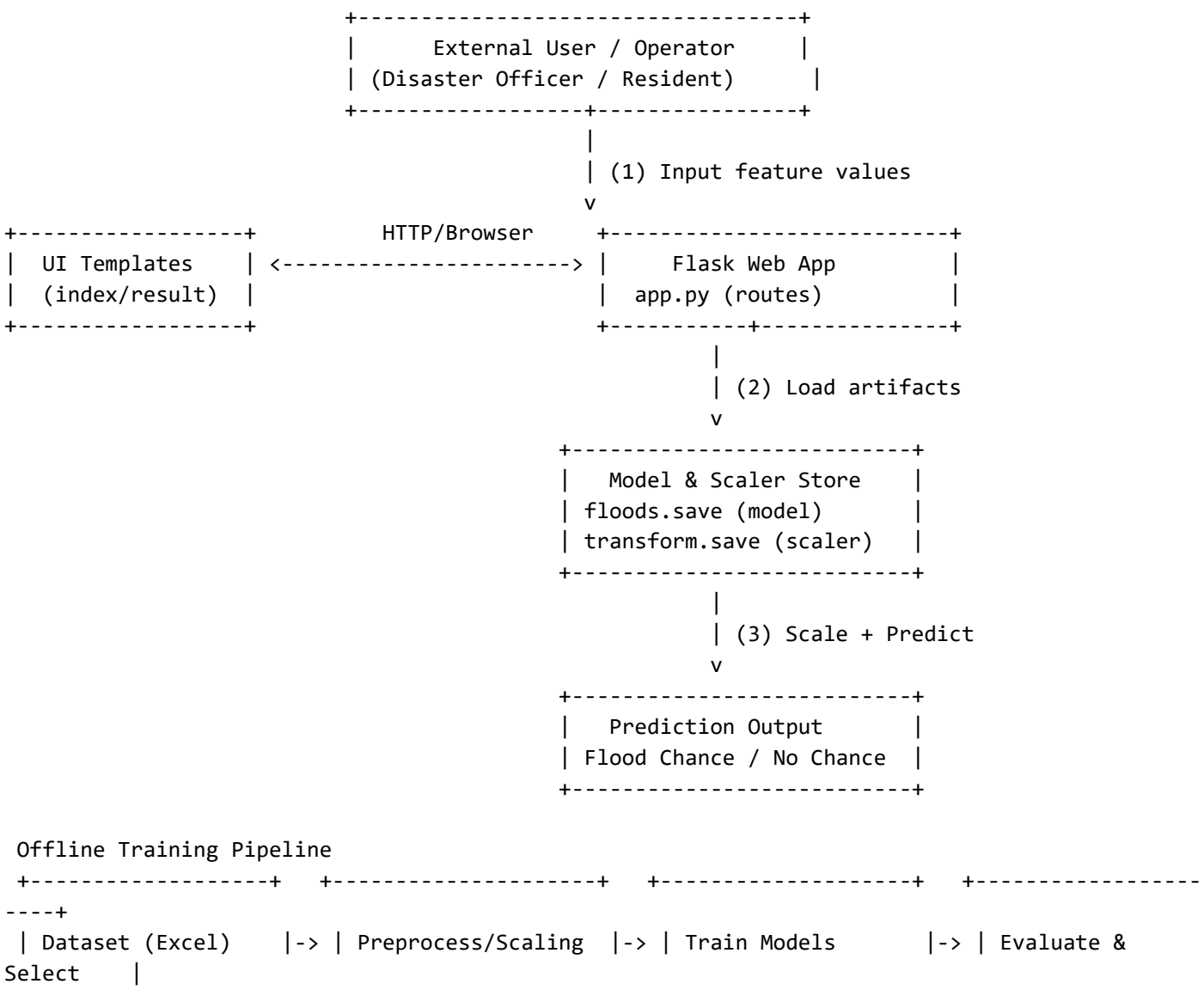
Date: 16 February 2026

Team ID: LTVIP2026TMIDS47616

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

Solution architecture describes the high-level structure of the system, the major components, and the flow of data from input to prediction output. The Rising Waters solution uses an offline training pipeline to build the model artifacts and a Flask-based web application to serve real-time predictions.

Solution Architecture Diagram (High-Level)



Component Description

Component	Type	Role	Technology
Flood Dataset	Data Store	Stores historical features and flood label for training	Excel (.xlsx)
Training Script/Notebook	Process	Loads data, splits train/test, scales features, trains multiple models	Python, pandas, scikit-learn, xgboost
Scaler Artifact	Model Artifact	Applies identical feature standardization at inference time	joblib (transform.save)
Model Artifact	Model Artifact	Predicts flood/no flood using selected best classifier	pickle (floods.save)
Flask Web App	Application	Accepts user inputs, performs scaling + prediction, renders results	Flask (Python)
HTML Templates	Presentation	Form page + result pages shown to user	HTML/CSS (Jinja2)

End-to-End Data Flow (Inference)

1. User opens the web app and enters feature values (Temp, Humidity, Cloud Cover, seasonal rainfall aggregates, etc.).

2. Flask validates the inputs and converts them into a numeric feature vector in the correct feature order.

3. The saved StandardScaler (transform.save) standardizes the input vector using training-time mean and variance.

4. The saved model (floods.save) predicts the class label: 1 = Flood Chance, 0 = No Flood Chance.

5. Flask renders the appropriate result page (chance.html or nochance.html) to the user.

5.1 Project Planning

Date: 16 February 2026

Team ID: LTVIP2026TMIDS47616

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

Product Backlog, Sprint Schedule, and Estimation

Story points scale used: 1 (Very Easy), 2 (Easy), 3 (Moderate), 5 (Difficult).

Sprint	Functional Requirement (Epic)	User Story No.	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Ingestion & Understanding	USN-1	As a developer, I can load the flood dataset (Excel) into the training script/notebook.	2	High	Team (LTVIP2026TMIDS47616)
Sprint-1	Data Ingestion & Understanding	USN-2	As a developer, I can inspect dataset shape, columns, and basic statistics.	1	High	Team (LTVIP2026TMIDS47616)
Sprint-1	Data Preprocessing	USN-3	As a developer, I can split features (X) and label (y) using the 'flood' column.	2	High	Team (LTVIP2026TMIDS47616)
Sprint-1	Data Preprocessing	USN-4	As a developer, I can perform train-test split with reproducible parameters.	1	High	Team (LTVIP2026TMIDS47616)
Sprint-1	Data Preprocessing	USN-5	As a developer, I can standardize features using StandardScaler and persist the scaler (transform.save).	3	High	Team (LTVIP2026TMIDS47616)
Sprint-2	Model Training & Selection	USN-6	As a developer, I can train a Decision Tree model for baseline flood prediction.	2	High	Team (LTVIP2026TMIDS47616)
Sprint-2	Model Training & Selection	USN-7	As a developer, I can train a	3	High	Team (LTVIP2026TMIDS47616)

			Random Forest model and compare performance.			
Sprint-2	Model Training & Selection	USN-8	As a developer, I can train a KNN model and compare performance.	2	Medium	Team (LTVIP2026TMIDS47616)
Sprint-2	Model Training & Selection	USN-9	As a developer, I can train an XGBoost model and compare performance.	3	High	Team (LTVIP2026TMIDS47616)
Sprint-2	Model Evaluation & Reporting	USN-10	As a developer, I can generate confusion matrix and classification report for the best model.	3	High	Team (LTVIP2026TMIDS47616)
Sprint-2	Model Serialization	USN-11	As a developer, I can save the best model (floods.save) for deployment.	2	High	Team (LTVIP2026TMIDS47616)
Sprint-3	Web Application (Flask)	USN-12	As a user, I can access a web page that shows input fields for flood prediction features.	3	High	Team (LTVIP2026TMIDS47616)
Sprint-3	Web Application (Flask)	USN-13	As a user, I can submit inputs and get a prediction result from the saved model + scaler.	5	High	Team (LTVIP2026TMIDS47616)
Sprint-3	Result Presentation	USN-14	As a user, I can view clear result pages (Flood Chance / No Flood Chance) and navigate back.	2	High	Team (LTVIP2026TMIDS47616)
Sprint-3	UI Enhancement	USN-15	As a user, I get a modern, classy UI with responsive layout and input validation hints.	3	Medium	Team (LTVIP2026TMIDS47616)
Sprint-4	Testing & Documentation	USN-16	As a developer, I can test the app using sample rows and confirm correct preprocessing and predictions.	3	High	Team (LTVIP2026TMIDS47616)
Sprint-4	Reporting	USN-17	As a team, we can prepare the project report	5	High	Team (LTVIP2026TMIDS47616)

			sections and supporting documents.			
Sprint-4	Packaging & Handover	USN-18	As a team, we can package code, artifacts, and run instructions for final submission.	3	High	Team (LTVIP2026TMIDS47616)

Project Tracker, Velocity & Burndown Chart

Sprint duration assumed: 7 days per sprint (can be adjusted as per your schedule).

Sprint	**Total Story Points**	**Duration**	**Sprint Start Date**	**Sprint End Date (Planned)**	**Story Points Completed (Planned End Date)**
Sprint-1	9	7 Days	17 Feb 2026	23 Feb 2026	9
Sprint-2	15	7 Days	24 Feb 2026	02 Mar 2026	15
Sprint-3	13	7 Days	03 Mar 2026	09 Mar 2026	13
Sprint-4	11	7 Days	10 Mar 2026	16 Mar 2026	11

Velocity (Average Story Points per Sprint) = Total Story Points Completed / Number of Sprints = 48 / 4 = 12.00

Burndown Chart: Plot remaining story points vs. days in sprint. (You can create this in Excel or any project tool.)

6.1 Performance Testing (Model Performance Test)

Date: 16 February 2026

Team ID: LTVIP2026TMIDS47616

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

Model Performance Testing: The following metrics summarize the classification performance of the trained flood prediction model on the test dataset.

S.No.	Parameter	Values
1	Model Type	Classification Model (Flood Chance / No Flood Chance)
2	Best Model Selected	Decision Tree Classifier (selected based on highest accuracy tie)
3	Accuracy Score	0.9655 (96.55%)
4	Confusion Matrix	[[26, 0], [1, 2]]
5	Classification Report	Class 0: precision=0.96, recall=1.00, f1=0.98 (support=26) Class 1: precision=1.00, recall=0.67, f1=0.80 (support=3) Overall accuracy=0.97 (support=29) Macro avg: precision=0.98, recall=0.83, f1=0.89 Weighted avg: precision=0.97, recall=0.97, f1=0.96

Interpretation of Results

• The model achieved ~96.55% accuracy on the test set.

• Confusion matrix shows 1 false negative (flood predicted as no-flood) and 0 false positives.

• Recall for flood class (1) is 0.67 due to very small support (only 3 flood samples in test set).

• This indicates the dataset is imbalanced; reducing missed floods (false negatives) is the key improvement area.

Tune the Model

Hyperparameter Tuning:

• Decision Tree: max_depth, min_samples_split, min_samples_leaf, class_weight

• Random Forest: n_estimators, max_depth, min_samples_leaf, class_weight

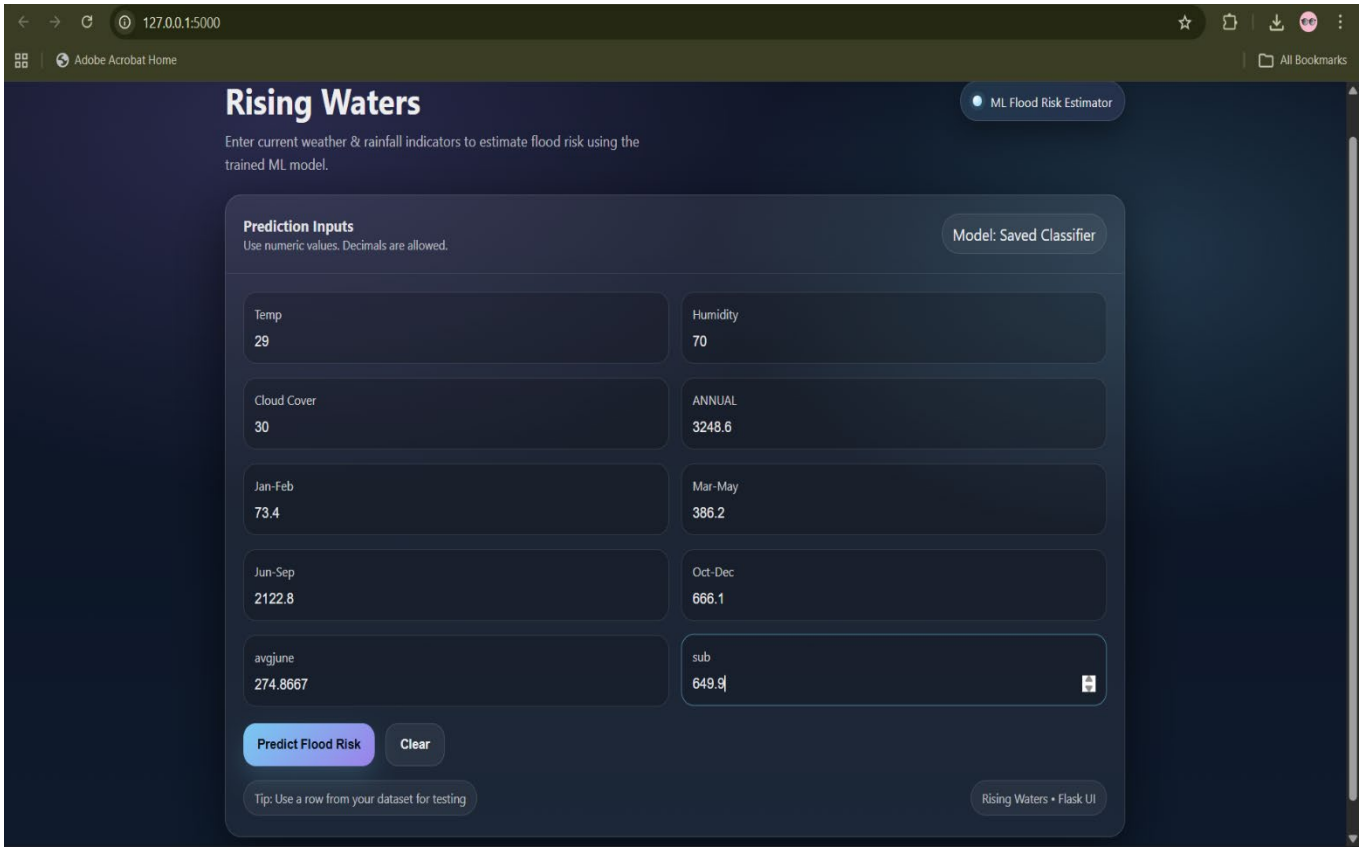
• XGBoost: n_estimators, learning_rate, max_depth, subsample, colsample_bytree, scale_pos_weight

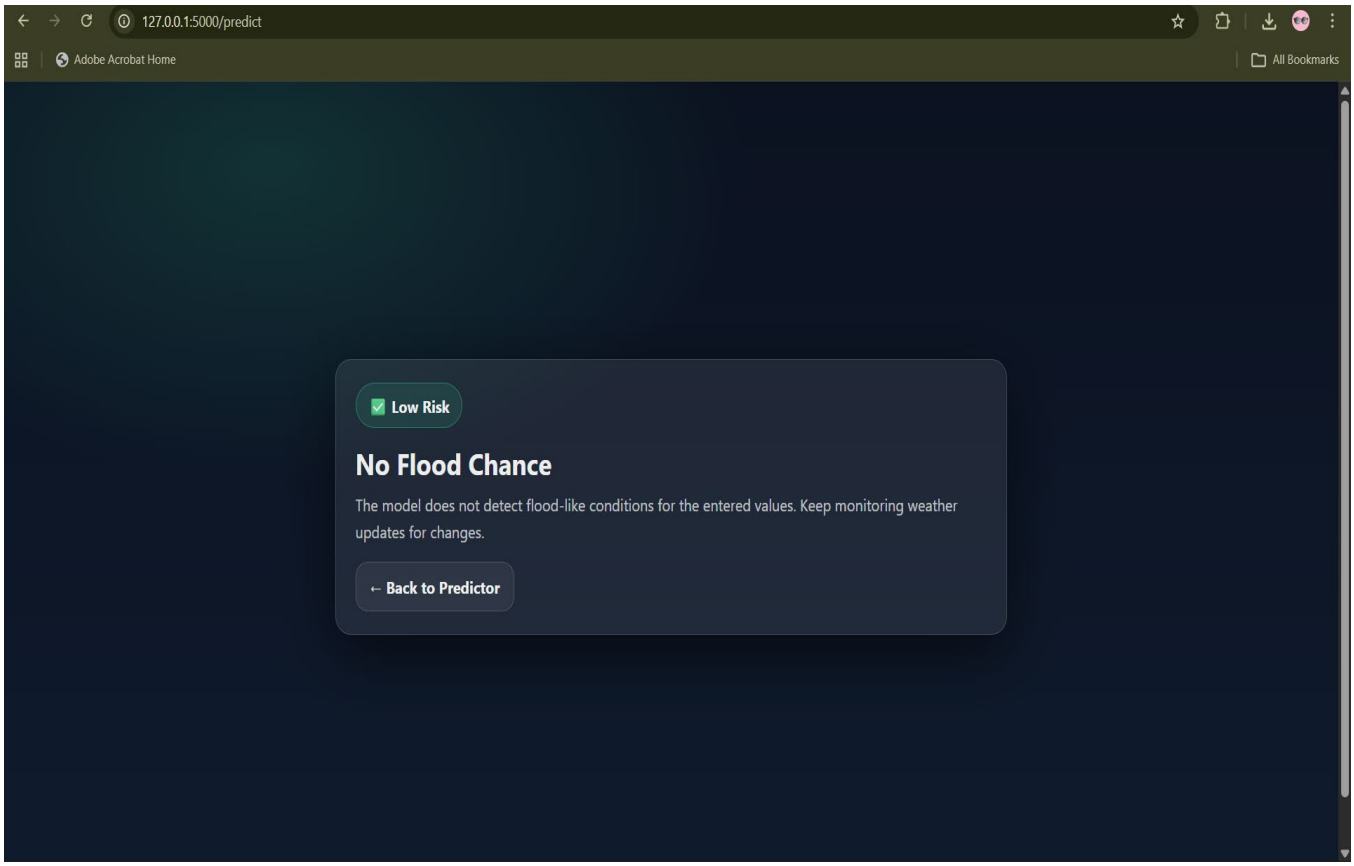
Validation Method:

• Stratified train-test split (test_size=0.25, random_state=10).

• Optional improvement: Stratified K-Fold Cross-Validation (e.g., 5-fold) to stabilize metrics on small datasets.

Screenshots: Attach screenshots of the console output (accuracy, confusion matrix, classification report) as evidence in the final submission.


127.0.0.1:5000
Adobe Acrobat Home
All Bookmarks
Rising Waters
ML Flood Risk Estimator
Enter current weather & rainfall indicators to estimate flood risk using the trained ML model.
Prediction Inputs
Use numeric values. Decimals are allowed.
Model: Saved Classifier
Temp
29
Humidity
70
Cloud Cover
30
ANNUAL
3248.6
Jan-Feb
73.4
Mar-May
386.2
Jun-Sep
2122.8
Oct-Dec
666.1
avgjune
274.8667
sub
649.9
Predict Flood Risk
Clear
Tip: Use a row from your dataset for testing
Rising Waters • Flask UI


127.0.0.1:5000/predict
Adobe Acrobat Home
All Bookmarks
Low Risk
No Flood Chance
The model does not detect flood-like conditions for the entered values. Keep monitoring weather updates for changes.
← Back to Predictor

Advantages and Disadvantages

Project: Rising Waters: A Machine Learning Approach to Flood Prediction
Team ID: LTVIP2026TMIDS47616

Advantages

- Fast, consistent flood-risk estimation from multiple indicators (temperature, humidity, cloud cover, seasonal rainfall aggregates).
- Reduces dependence on subjective/manual judgment and supports quicker decision-making during emergencies.
- End-to-end pipeline (training, evaluation, saving artifacts, deployment) demonstrates real-world ML implementation.
- Lightweight deployment using Flask makes it easy to run on a normal laptop and share within an organization.
- Clear output (Flood Chance / No Flood Chance) improves usability for non-technical users.
- Model and scaler are persisted (floods.save, transform.save), ensuring repeatable inference without retraining.

Disadvantages

- Small dataset (115 rows) and class imbalance can make results unstable and reduce recall for the flood class.
- Binary output may oversimplify risk; borderline conditions may need probability/confidence to support decisions.
- Predictions depend heavily on input data quality; incorrect or delayed inputs can lead to misleading results.
- Generalization may be limited across regions/seasons because the model is trained on a fixed historical dataset.
- No real-time data integration in the baseline system (manual feature entry required).
- Limited interpretability for stakeholders unless feature importance/explanations are added.

9. CONCLUSION

The project **“Rising Waters: A Machine Learning Approach to Flood Prediction”** demonstrates an end-to-end machine learning workflow to classify flood risk using historical weather and rainfall indicators. The dataset was preprocessed, features were standardized using a saved scaler, and multiple classification models were trained and evaluated. The selected model achieved strong test performance (accuracy ≈ **96.55%**) and was deployed as a **Flask** web application that enables users to enter feature values and instantly obtain a clear output (**Flood Chance / No Flood Chance**). Overall, the solution provides a quick, consistent, and practical decision-support tool that can assist in flood preparedness and early risk estimation.

GitHub Repository:
https://github.com/kaushik-1708/Rising-Waters-Machine-Learning-approach-to-flood-prediction/tree/main

10. FUTURE SCOPE

- **Larger & richer datasets:** Train on more samples across multiple regions/years to improve generalization and handle imbalance.
- **Probability/confidence output:** Show prediction probability (%) to support decision-making in borderline cases.
- **Improve flood recall:** Tune thresholds and use imbalance handling (class weights/SMOTE/stratified k-fold) to reduce false negatives.
- **Real-time integration:** Auto-fetch weather/rainfall inputs from APIs to reduce manual entry.
- **Explainability:** Add feature importance/SHAP explanations to justify predictions for stakeholders.
- **Cloud deployment:** Host the app on a cloud platform and provide REST APIs for dashboards/alerts.
- **Operational workflow:** Add recommended actions/checklists when “Flood Chance” is predicted.

Note: Future enhancements should prioritize reducing missed floods (false negatives).

11. APPENDIX

11.1 Source Code

- Training pipeline: dataset loading → preprocessing → scaling → model training & evaluation → saving artifacts (**floods.save**, **transform.save**).
- Deployment: Flask app (**app.py**) loads saved model + scaler, takes user input via UI, and renders result pages.

11.2 Dataset Link

Kaggle – Rainfall Dataset (used in the guided project):
https://www.kaggle.com/datasets/arbethi/rainfall-dataset

11.3 GitHub Link

https://github.com/kaushik-1708/Rising-Waters-Machine-Learning-approach-to-flood-prediction/tree/main