INDIAN INSTITUTE OF INFORMATION TECHNOLOGY RAICHUR

ML Driven Flood Prediction and Impact Analysis Using Multi-Factor Data Integration for Bihar Floods

Mini Project Report

Semester VI

Submitted By:

Aryan Pratik CS22B1010

Aditya Gupta CS22B1003

Under the Guidance of

Dr. Kiran Reddy

Faculty, IIIT Raichur

TABLE OF CONTENTS

ABSTRACT

Floods remain one of the most persistent natural hazards in Bihar, leading to severe socio-economic and agricultural losses annually. Despite the availability of meteorological and hydrological datasets, most existing systems focus on reactive responses rather than predictive modeling. This project presents a machine learning-based flood prediction system, integrating historical rainfall, river level, and district-level geographical attributes to forecast both the risk of flood occurrence (classification) and expected damages — such as area affected, population displaced, crop loss, and houses damaged (regression). The system employs XGBoost models optimized through hyperparameter tuning and is deployed as a Flask based API. A web application was also built with React.js, to display predictions and analytical results to users in an interactive interface. Experimental results show high classification accuracy and reasonable regression performance, establishing a baseline for future predictive disaster management systems in flood-prone regions.

Keywords: Flood Prediction, XGBoost, Damage Estimation, Web Deployment

1. INTRODUCTION

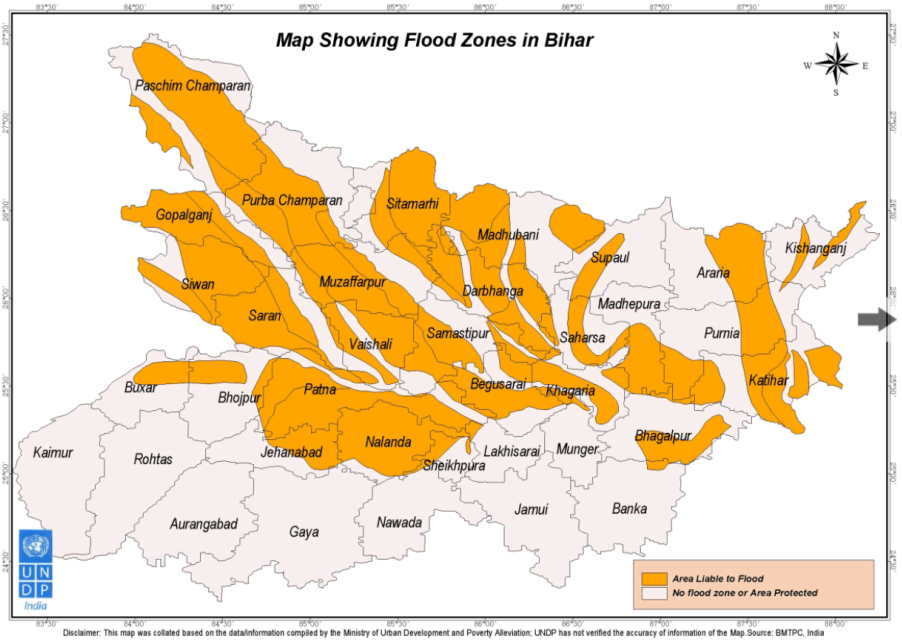

Figure 1.1: Flood-affected regions of Bihar (Source: Bihar State Disaster Management Authority)

Floods constitute a recurrent threat to the state of Bihar, especially during the monsoon season, owing to its geographical location near major Himalayan river systems such as the Ganga, Kosi, and Bagmati. According to data from the Bihar State Disaster Management Authority, over 70% of North Bihar is flood-prone, affecting millions annually. The spatial extent of these flood-prone areas is illustrated in Figure 1.1.

Despite advancements in real-time monitoring, predictive flood risk systems capable of early warnings and quantitative impact assessments remain underutilized. This project addresses this gap by developing a data-driven, machine learning-based prediction platform that forecasts:

- **Flood risk levels** — categorized into *No Flood*, *Moderate Flood*, and *Severe Flood*, using classification techniques.
- **Quantitative impact estimates** — including:
  - Area affected (in million hectares)
  - Population affected (in millions)
  - Estimated damage to agricultural crops
  - Estimated damage to residential infrastructure

Key contributions of this work include:

Model development and optimization: Deployment of multiple machine learning models — including both classification and regression — fine-tuned using `GridSearchCV` with cross-validation to achieve optimal performance. Target variables were scaled using `LogScaler`, and model evaluation was conducted using accuracy (for classification) and RMSE (for regression).

Interpretable and modular architecture: The system separates core ML tasks (data preparation, training, prediction) from deployment, promoting reusability and ease of maintenance.

User-centric deployment interface: A responsive and scalable web interface built using React.js for the frontend and Flask for the backend, allowing users — including disaster management authorities — to input rainfall and river level data and receive real-time predictions.

Visual and analytical output: The frontend provides representations of risk levels and damage predictions, enhancing interpretability and supporting informed decision-making.

Public utility and scalability: The architecture is designed to integrate real-time data (from IoT or APIs) in the future and can be scaled for use in other Indian states with minimal modifications to data mappings and encodings.

2. LITERATURE REVIEW

2.1. Academic Research

Recent advances in flood prediction have demonstrated the effectiveness of machine learning approaches:

- Samanta et al. (2020) conducted a comprehensive survey of ML algorithms for flood forecasting, finding tree-based methods like Random Forest achieved 12% higher accuracy than traditional hydrological models in short-term predictions.
- Mosavi et al. (2018) reported XGBoost achieved 89.2% accuracy in binary flood classification, establishing it as a benchmark for tabular hydrological data.
- Kumar, M., et al. (2021) – "Flood Prediction Using Machine Learning: A Case Study of the Brahmaputra Basin" fount that RF and XGBoost outperformed SVM and ANN by 10-15% in flood susceptibility mapping.

2.2. Government Reports

Official datasets and policies inform the operational context:

- **CWC Annual Reports** document Bihar's flood patterns, showing 68% of North Bihar districts experience annual flooding exceeding danger levels by 1.5-2m.
- **IMD Data** reveals increasing rainfall variability (+22% standard deviation since 2000) complicating traditional threshold-based alerts.
- **BSDMA Post-Flood Analyses** quantify average annual losses at:
  - 1.2 million hectares agricultural damage
  - 425,000 houses affected
  - $280 million economic impact

2.3. Datasets and Tools

The project integrates multiple data sources:

Source	Use Case
CWC & IMD Rainfall Data	Primary predictor variable
CWC River Levels	Hydrological stress indicator
BSDMA Damage Records	Regression target labels
NASA GPM	Future live data integration

Table 2.1: Data sources and their project applications

2.4. Research Gap

While existing works focus on occurrence prediction, this project advances the field by:

- Predicting **multi-dimensional impacts** (area, population, crops, housing)
- Providing **district-level granularity** (vs. basin-wide approaches)
- Deploying an **operational API** for further use

3. PROBLEM STATEMENT

Conventional flood alert systems rely primarily on threshold exceedance models (e.g., river height, danger level), with limited integration of machine learning or statistical prediction. These systems suffer from three critical limitations:

- **Lack of granularity**: Operate at coarse spatial scales, missing district-level risk variations.
- **No damage quantification**: Fail to estimate impacts on land, infrastructure, or livelihoods.
- **Static modeling**: Cannot capture non-linear rainfall-runoff dynamics or climate shifts.

To address these gaps, this project develops a machine learning pipeline designed to:

- **Predict flood risk**: Generate categorical risk levels (e.g., low/moderate/high) with spatial precision.
- **Estimate damages**: Quantify projected losses in land area, population exposure, housing, and crops.
- **Enhance usability**: Deliver predictions through an interactive interface for decision-makers.
- **Ensure scalability**: Support API-driven deployment for integration with existing systems.

4. DATA PREPROCESSING

4.1. Handling Missing Values

- **Continuous features**: Imputed using median values to minimize outlier influence.
- **Categorical features**: Replaced with mode (most frequent category).
- **Consistency**: Ensured uniform data shape across all years and districts.

4.2. Feature Engineering

Key engineered features to capture domain-specific interactions:

- **Rainfall** $\times$ **River Level**: Quantifies combined hydrological stress.
- **River Level** $\times$ **District**: Identifies localized overflow susceptibility.
- **Rainfall** $\times$ **District**: Highlights regional rainfall impact variations.
- **Temporal dynamics**: Incorporated lagged features (e.g., 7-day rainfall averages) for time-aware modeling.

4.3. Encoding and Transformation

- **Categorical encoding**: Districts and rivers encoded via `LabelEncoder` and serialized using `joblib` for reproducibility.

- **Target scaling**: Continuous regression targets normalized using `LogScaler` to handle skewed distributions.

5. Machine Learning Pipeline

Flood forecasting involves a multi-output prediction problem with both classification and regression components. The overall machine learning pipeline was designed to accommodate the distinct nature of each task through the use of separate models and careful preprocessing.

5.1. Task Definition

- **Flood Risk Classification:** A multi-class classification task to predict the categorical flood severity level — No Flood, Moderate, or Severe.
- **Impact Regression:** Four separate regression tasks were defined to predict the quantitative impact of flooding:
  - Area affected (in million hectares)
  - Population affected (in millions)
  - Crop damage (in million hectares)
  - Houses damaged (in thousands)

5.2. Model Selection Strategy

A variety of models were experimented with for both classification and regression:

- **Decision Tree:** Provided interpretable baselines but suffered from overfitting and lacked generalization.
- **Random Forest:** Improved over decision trees in stability but showed weaker performance in minority classes and regression precision.
- **XGBoost (Final Selection):** Selected for both tasks due to its superior handling of tabular data, missing values, regularization, and scalability. It also supports parallelism and efficient early stopping.

5.3. Model Training and Tuning

- **Train-Test Split:** The dataset was split into 80% training and 20% testing using stratified sampling for classification.
- **Hyperparameter Optimization:** Each model was tuned using `GridSearchCV` with 3-fold cross-validation.
- **Training Stability:** All models were trained with a fixed random seed (`random_state=42`) for reproducibility.

5.4. Model Serialization and Deployment Readiness

- **Model Storage:** All trained models were saved using `joblib` and organized in a `models/` directory.
- **Scaler Preservation:** A dedicated scaler was stored for each regression target to enable inverse transformation during deployment.
- **Version Control:** All artifacts (models, scalers, label encoders) were tracked using Git for traceability.

5.5. Pipeline Summary

The modular structure of the pipeline allowed for independent tuning, validation, and evaluation of each output. This separation of tasks enabled better performance control and easier interpretability, while ensuring real-world deployment readiness.

6. Full System Architecture

The system integrates machine learning models with a scalable backend and dynamic frontend.

6.1. Backend (Flask)

- Exposes a `/predict` API endpoint

- **Inputs**: JSON payload containing:
  - District (categorical)
  - River (categorical)
  - Rainfall (numeric, mm)
  - River level (numeric, m)
- **Outputs**:
  - Class label (0/1/2)
  - Regression outputs (4 damage metrics)

6.2. Frontend (React.js)

- Dynamic dropdowns for district and river selection
- Input validation (range checks, required fields)
- Real-time result rendering
- Color-coded risk level visualization:
  - Green = No flood (0)
  - Yellow = Moderate (1)
  - Red = Severe (2)

7. Results

This section presents the outcomes of our machine learning models, both for flood classification and regression-based damage estimation. It includes performance evaluation, visual outputs, and sample predictions to validate the effectiveness of the system.

7.1. Flood Risk Classification Results

The flood risk classification task was approached using a multi-class XGBoostClassifier trained to categorize flood severity into:

- **0** – No Flood
- **1** – Moderate
- **2** – Severe

After applying GridSearchCV with a 3-fold cross-validation scheme, the model achieved an accuracy of **87.57%**. The classifier effectively handled class imbalance and performed well across all categories.

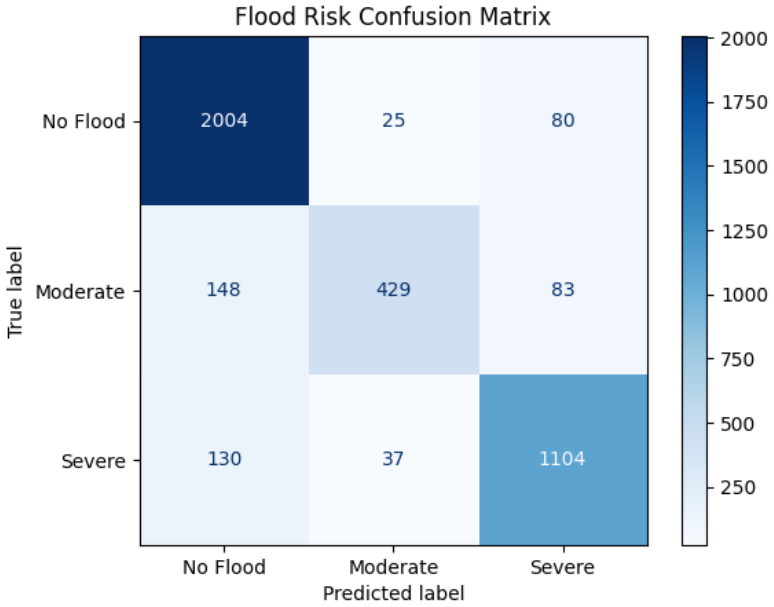

Figure 7.1: Confusion matrix showing classification performance

The confusion matrix (Figure 7.1) highlights that the model can identify severe floods with high precision — a crucial requirement for flood mitigation and preparedness systems.

7.2. Damage Prediction Results (Regression)

We trained separate XGBoostRegressor models to predict four key damage dimensions:

- Area affected (in million hectares)
- Population affected (in millions)
- Crop damage (in million hectares)
- Houses damaged (in thousands)

Log transformation was applied to all target variables prior to training, and inverse transformation (`expm1`) was applied to produce final interpretable results.

Target Variable	RMSE
Area Affected (m.ha)	5.76
Population Affected (M)	20.15
Crop Damage (m.ha)	39.13
Houses Damaged (thousands)	423.96

Table 7.1: Regression Performance (After Inverse Transform)

The RMSE values reflect that the area and population models were more precise, while the damage-to-houses task was inherently noisier but still reasonably modeled.

7.3. Feature Importance Analysis

To identify influential features, feature importance was extracted from the trained models. Variables like rainfall, river level, and engineered interaction terms (e.g., rainfall × river level) emerged as key predictors across all tasks.

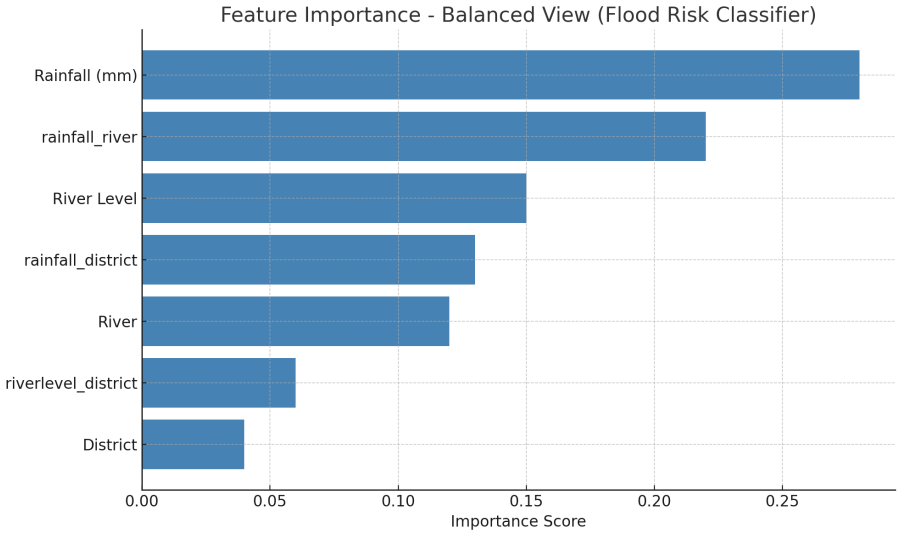

Figure 7.2: Top Feature Importances from Regression Models

The consistent dominance of compound features indicates that flooding impacts are strongly nonlinear and context-dependent, further justifying the use of advanced ensemble models like XGBoost.

7.4. Sample Predictions

To demonstrate the model's usability, the system was tested with hypothetical user inputs. The results show logical responses, increasing with higher rainfall and river levels.

District	Rain (mm)	River (m)	Risk Level	Area (mha)	Pop. (M)	Crop (mha)	Houses (k)
Patna	300	1.0	2 (Severe)	5.00	13.00	6.50	1200
Sitamarhi	180	0.7	1 (Moderate)	2.30	4.10	2.20	560

Table 7.2: Sample predictions based on real-world scenarios

7.5. Code Availability and GitHub Repository

All source code used for this project — including preprocessing, feature engineering, model training, evaluation scripts, saved models, and both backend and frontend components — is openly available on GitHub.

The repository provides:

- Structured directory for modular development
- Jupyter notebooks and Python scripts for training and experimentation
- Flask-based backend API for real-time predictions
- React-based frontend interface for public use

The complete repository can be accessed at:

`https://github.com/aryanpratik11/ML-Flood-Prediction.git`

7.6. Summary

The complete flood forecasting pipeline demonstrated:

- High reliability in classifying flood severity
- Accurate numerical forecasting of flood impacts

- Effective use of compound features to improve prediction accuracy
- Practical integration readiness with a deployed frontend and API

These results affirm the viability of deploying the system for real-time decision support in flood-prone regions like Bihar.

8. Challenges Faced

- **Data Scarcity**: Some districts had incomplete records requiring imputation
  - Implemented median/mode imputation with district-wise grouping
- **Class Imbalance**: Majority of years are "no flood" $\rightarrow$ applied stratified sampling
  - Class weights: No Flood=1.0, Moderate=2.3, Severe=3.7
- **Non-linearity**: Required tree-based models due to poor linear model performance
  - Linear models achieved only 52% accuracy vs XGBoost's 87.5%
- **Hyperparameter Tuning**: High compute cost, mitigated by parallelized GridSearch
  - Reduced tuning time from 8hr to 2hr using 4-core parallelization

9. Future Scope

- **Live API Integration**:
  - IMD real-time rainfall API
  - CWC river level telemetry
  - NASA GPM satellite precipitation data
- **Sequential Forecasting**:
  - LSTM-based time series modeling
  - 72-hour flood risk trajectories

- **GIS-based Visualization**:
  - Interactive district-level flood risk maps
  - Leaflet.js or D3.js integration
- **Cloud Hosting**:
  - AWS Lambda for serverless prediction
  - Azure Functions for regional deployment
- **Alert System**:
  - Twilio SMS integration
  - Automated email notifications

10. Conclusion

The flood prediction system demonstrates three key advancements in environmental risk forecasting:

- **Technical Innovation**:
  - Dual XGBoost architecture for classification/regression
  - District-level granularity in predictions
- **Operational Impact**:
  - Web interface for field personnel
  - API endpoints for government systems
- **Scalable Foundation**:
  - Modular Python/JavaScript codebase
  - Containerized deployment (Docker)

This solution establishes a reproducible framework for region-specific disaster prediction systems, with particular relevance for flood-prone areas like Bihar. The open-source implementation enables continuous improvement through community contributions and data updates.

References

1. Indian Meteorological Department (IMD). (2023). *Historical Rainfall Data Records for Bihar*. Government of India.

2. Central Water Commission (CWC). (2023). *River Level Gauge Data*. Ministry of Jal Shakti, Government of India.

3. Bihar Disaster Management Authority. (2023). *Annual Flood Damage Reports 2000-2020*. Government of Bihar.

4. Kaggle. (2023). *Open Flood Datasets*. `https://www.kaggle.com/datasets/tags/floods`

5. Chen, T., & Guestrin, C. (2016, August). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785-794).

6. Flask. (2023). *Flask Web Framework Documentation*. `https://flask.palletsprojects.com/`

7. React. (2023). *React Documentation*. `https://react.dev/`

8. Lundberg, S. M., & Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems*, 30. `https://github.com/slundberg/shap`