SOUTHAMPTON SOLENT UNIVERSITY

DEPARTMENT OF SCIENCE AND ENGINEERING

MSc Applied AI and Data Science
Academic Year 2024-2025

Anandhakrishnan Madathil Remesh

AI-Driven Flood Prediction and Mapping System

Supervisor: Dr. Raza Hasan
[May] [2025]

This report is submitted in partial fulfilment of the requirements of Southampton Solent University for the degree of MSc Applied AI and Data Science.

Abstract

Flooding remains one of the most devastating natural disasters, leading to severe loss of life, economic disruption, and environmental damage. Traditional flood prediction models, which primarily rely on static hydrological and meteorological data, often fail to capture the complex and nonlinear interactions influencing flood events. This research presents an AI-driven flood prediction and mapping system, integrating XGBoost and Random Forest models for flood risk classification and a U-Net deep learning model for flood extent mapping using satellite imagery. The methodology involves data collection from meteorological, hydrological, and remote sensing sources, followed by preprocessing, feature engineering, and model training to improve predictive accuracy. The XGBoost and Random Forest models analyze structured hydrological data, including rainfall, drainage capacity, and dam conditions, to predict flood probabilities. Simultaneously, the U-Net model processes satellite imagery for semantic segmentation, mapping flood-affected areas with high precision. The trained models are deployed in an interactive web-based dashboard using Streamlit, enabling real-time flood risk assessment and visualization. Experimental results demonstrate that the AI-based approach significantly enhances flood prediction accuracy compared to traditional hydrological models. The system provides timely, data-driven insights for disaster preparedness, aiding authorities in decision-making and resource allocation. This research underscores the potential of machine learning and geospatial analysis in improving flood resilience and disaster mitigation strategies.

Table of Content

Table of Figures

List of Tables

Acronyms

AUC: Area Under the Curve

AUC-ROC: Area Under the Receiver Operating Characteristic Curve

AI: Artificial Intelligence

ANN: Artificial Neural Network

API: Application Programming Interface

BCE: Binary Cross-Entropy

BRT: Boosted Regression Trees

CART: Classification and Regression Tree

CNN: Convolutional Neural Network

DL: Deep Learning

F1-Score: Harmonic mean of precision and recall

GloFAS: Copernicus Global Flood Awareness System

IoU: Intersection over Union

LR: Logistic Regression

LSTM: Long Short-Term Memory networks

ML: Machine Learning

PR: Precision-Recall

RGB: Red-Green-Blue color format

RF: Random Forest

SHAP: Shapley Additive Explanations

SVM: Support Vector Machine

WHO: World Health Organization

XAI: Explainable Artificial Intelligence

XGBoost: Extreme Gradient Boosting

Chapter 1: Introduction

This chapter outlines the background (Section 1.1) and problem context (Section 1.2) of the research, along with its purposes (Section 1.3). Section 1.4 defines the scope of this research work and presents the objectives. Section 1.5 highlights the research questions considered to address the problem. Ethical considerations are discussed in Section 1.6. Finally, Section 1.7 provides an outline of the remaining chapters of this dissertation.

1.1 RESEARCH BACKGROUND

Flooding is one of the most destructive natural disasters, with profound impacts on human lives, property, and ecosystems worldwide (United Nations Office for Disaster Risk Reduction, 2022). According to the World Meteorological Organization (2021), floods account for over 40% of all natural disasters globally, causing extensive damage to infrastructure and agriculture. These catastrophic events are increasingly driven by a combination of factors, including climate change, deforestation, rapid urbanization, and inadequate drainage systems. As the frequency and intensity of floods continue to rise, the challenge of effectively predicting and managing these events becomes more pressing for governments, disaster response agencies, and affected communities.

The consequences of flooding extend far beyond immediate damage. In addition to loss of life, floods frequently lead to the displacement of entire communities, creating long-term humanitarian crises. Reports from the Centre for Research on the Epidemiology of Disasters (2023) reveal that floods have claimed over 200,000 lives globally in the past five decades, making it one of the deadliest natural hazards. Furthermore, the economic toll of flooding is substantial, with damages to infrastructure, agricultural lands, and businesses causing severe financial strain on local and national economies (Kundzewicz et al., 2019). Beyond economic losses, the environmental impact of flooding cannot be overlooked. Floodwaters contaminate water sources, spread waterborne diseases, and destroy critical ecosystems, exacerbating existing environmental challenges and complicating recovery efforts (James C. Young et al., 2018).

Despite advancements in meteorology and hydrological modeling, traditional flood prediction systems remain inadequate. These methods primarily rely on historical

rainfall data, river discharge measurements, and static topographical maps, but struggle to capture the complex, dynamic, and nonlinear nature of climate-induced flood events. As a result, flood forecasting remains highly inaccurate, leading to delayed responses and suboptimal disaster management. The urgent need for real-time, data-driven flood prediction systems has never been clearer.

In recent years, machine learning (ML) and deep learning (DL) techniques have shown great potential in addressing these challenges. XGBoost and Random Forest models have been widely applied for structured data analysis, improving flood forecasting accuracy by identifying patterns in historical flood data. Additionally, Convolutional Neural Networks (CNNs) and U-Net architectures have been applied successfully in satellite image analysis, enabling precise flood detection and mapping of flood extents (Dev Raj Paudyal et al., 2023). These AI-driven models offer the possibility of more accurate flood predictions, efficient early warning systems, and optimized disaster response strategies, ultimately minimizing loss of life and property.

1.2 PROBLEM STATEMENT

Floods are not just natural disasters; they are relentless forces of destruction that uproot lives, dismantle economies, and alter landscapes within hours. In many parts of the world, communities are caught off guard by these catastrophes, left with little to no time to react. While meteorologists and hydrologists have worked tirelessly to improve flood prediction models, current approaches remain alarmingly inadequate.

Traditional flood prediction methods rely heavily on historical data, river discharge measurements, and topographical mapping. However, in an era where climate change is altering weather patterns unpredictably, these static models fail to capture the dynamic and nonlinear nature of flood events. The consequences are dire delayed evacuation plans, overwhelmed emergency responders, and infrastructure unable to withstand the impact of sudden flooding. Rapid urbanization and deforestation further exacerbate the situation, rendering old methodologies obsolete.

The need for a transformative solution has never been more urgent. Flood prediction must evolve beyond its conventional form into an intelligent, real-time, and adaptive system that can integrate multiple data sources, identify risks dynamically, and provide actionable insights. Without significant advancements in predictive technologies, floods will continue to devastate vulnerable communities, leading to loss of lives, displacement of populations, and extensive economic damage. This research takes a bold step forward, leveraging artificial intelligence to build a flood prediction and mapping system that empowers communities and disaster response teams with reliable, real-time insights because preparedness is the key to resilience.

1.3 PROPOSED SOLUTION

To address the limitations of traditional flood prediction methods, this study proposes the adoption of machine learning (ML) and deep learning (DL) techniques specifically XGBoost, Random Forest models, and U-Net architectures to enhance flood prediction, mapping, and early warning systems.

XGBoost and Random Forest models, known for their ability to handle structured data, will be used to analyze historical and real-time hydrological parameters such as rainfall, drainage capacity, and river discharge levels. These models have been widely applied for predictive analytics due to their efficiency in handling large-scale datasets and capturing non-linear relationships in flood-related variables.

In addition, U-Net, a deep learning model specialized for image segmentation, will be used for flood extent mapping. By processing high-resolution satellite imagery, U-Net can delineate flood-prone regions, providing an accurate representation of affected areas. This is particularly crucial for real-time decision-making and emergency response planning.

The proposed AI-driven system will integrate these models into an interactive Streamlit-based dashboard that provides real-time flood predictions, visual risk assessments, and early warning notifications. This solution will:

- Improve the accuracy of flood forecasting by leveraging advanced machine learning techniques.
- Enhance situational awareness through dynamic, real-time flood mapping.
- Support disaster management agencies and communities by providing actionable insights for preparedness and response.

By combining predictive analytics with geospatial mapping, this research aims to develop a comprehensive and adaptive flood prediction system that minimizes the devastating impact of floods, ultimately saving lives, reducing economic losses, and strengthening disaster resilience.

1.4 AIMS AND OBJECTIVES

The aim of this research is to develop an advanced artificial intelligence-based flood prediction and monitoring system that significantly enhances early warning

mechanisms and supports proactive disaster preparedness. By integrating structured hydrological data with unstructured satellite imagery, the proposed system seeks to deliver high-accuracy flood risk predictions and real-time flood extent mapping. This integrated approach is intended to strengthen the decision-making capabilities of disaster response teams, policymakers, and local communities, ultimately contributing to reduced flood impact and improved disaster resilience.

To achieve this aim, the research is guided by the following objectives:

- To develop a machine learning-based flood prediction model using XGBoost and Random Forest algorithms for analyzing structured hydrological data.
- To implement a U-Net deep learning model for segmenting and mapping flood-prone regions using satellite imagery.
- To integrate structured and unstructured data into a unified prediction framework that enhances accuracy, spatial coverage, and reliability.
- To design an interactive, real-time dashboard using Streamlit to visualize flood risk assessments and support timely decision-making for disaster management.

The system's effectiveness will be critically evaluated through a comparison with traditional hydrological models using standard performance metrics such as Accuracy, F1-Score, AUC-ROC, and IoU. This evaluation will validate the accuracy, interpretability, and operational readiness of the proposed AI-driven flood prediction system.

1.5 RESEARCH QUESTIONS

This study explores the application of artificial intelligence to enhance flood risk prediction and spatial mapping by integrating structured hydrological data with unstructured satellite imagery. The focus lies in evaluating the predictive accuracy, reliability, and practical usability of AI-driven models to support effective flood management and early warning systems. The research is guided by the following questions:

1. How effectively can an AI-driven system predict flood risks and identify vulnerable regions using a combination of structured hydrological data and unstructured satellite imagery?
2. How do traditional hydrological models compare with machine learning algorithms in terms of predictive accuracy, precision, and generalization across diverse environmental conditions?
3. How can deep learning models like U-Net be used to accurately segment and map flood-prone areas from satellite imagery, and what is their performance in terms of spatial evaluation metrics such as IoU and Dice Coefficient?

1.6 ETHICAL CONSIDERATION

Ethical considerations play a vital role in the responsible development and deployment of AI-driven flood prediction systems. This study ensures that all data used for model training and analysis comply with relevant data privacy regulations, avoiding the misuse of sensitive information. Hydrological and geospatial datasets will be sourced from reputable and publicly available databases, ensuring transparency and accountability in data handling.

One of the key ethical concerns in AI applications is bias in predictive models. To address this, efforts will be made to ensure that the machine learning models are trained on diverse datasets that account for regional variations in flood patterns, preventing disparities in prediction accuracy across different geographical locations. Explainable AI (XAI) techniques will also be integrated into the system to enhance interpretability, allowing stakeholders to understand how predictions are generated and reducing the risk of opaque decision-making.

Moreover, fairness and accessibility are prioritized in this research. The developed system aims to benefit both governmental agencies and local communities, ensuring that flood prediction insights are accessible to all relevant stakeholders. The system's user interface will be designed to be intuitive and inclusive, catering to a wide range of users with varying levels of technical expertise.

Lastly, ethical AI deployment must align with sustainability and environmental responsibility. The computational resources used for training deep learning models will be optimized to minimize energy consumption, contributing to eco-friendly AI practices. By adhering to these ethical principles, this research ensures that AI-driven flood prediction technology is deployed responsibly, benefiting society while upholding data integrity and fairness.

1.7 DISSERTATION OUTLINE

This dissertation is structured as follows:

- Chapter 1: Introduction – Provides an overview of the research, including background, problem statement, proposed solution, aims and objectives, research questions, ethical considerations, and the dissertation structure.
- Chapter 2: Literature Review – Examines existing research related to flood prediction, machine learning techniques, remote sensing, and disaster management approaches.
- Chapter 3: Methodology – Outlines the research design, data collection process, machine learning model development, implementation framework, and evaluation metrics.
- Chapter 4: Implementation and Results – Details the process of training and deploying the models, performance evaluation, and experimental findings.
- Chapter 5: Discussion – Analyzes the results in the context of existing literature, highlights key insights, and discusses challenges and limitations.
- Chapter 6: Conclusion and Future Work – Summarizes the research contributions, discusses implications, and suggests potential future improvements.

This structured approach ensures a clear and logical flow of the dissertation, guiding the reader through each phase of the study, from conceptualization to practical implementation and evaluation.

Chapter 2: LITERATURE REVIEW

Flood prediction is a crucial aspect of disaster management, with the potential to minimize loss of life and economic damage. Traditional methods, including hydrological and meteorological models, have long been the foundation of flood forecasting. However, recent advancements in artificial intelligence (AI) and machine learning (ML) have introduced more sophisticated, data-driven approaches to flood prediction. This chapter critically evaluates existing literature on flood prediction methodologies, particularly focusing on the evolution from traditional hydrological models to modern ML and deep learning techniques.

The literature review will explore key themes, including traditional flood prediction models, ML-based forecasting, deep learning applications, explainable AI (XAI) techniques, and ensemble learning approaches. Each section will critically assess the strengths and limitations of existing studies, highlighting the research gap that this study aims to address.

2.1 TRADITIONAL FLOOD PREDICTION MODELS

Historically, flood prediction has relied on hydrological models that use river discharge, precipitation, and topographic data to estimate flood risk. One of the widely used models is the Copernicus Global Flood Awareness System (GloFAS), which integrates meteorological data with hydrological simulations (Nearing et al., 2024). While effective for large-scale flood monitoring, these models often lack precision when applied to smaller regions with unique hydrological characteristics.

Mosavi et al. (2018) reviewed various hydrological models and found that while they provide a foundational understanding of flood risks, they struggle with real-time adaptability and are heavily dependent on historical datasets. This reliance on past data makes them less effective in predicting extreme flood events influenced by climate change. Another key limitation is their inability to integrate multiple data sources, such as satellite imagery and environmental parameters, which could improve prediction accuracy.

2.2 MACHINE LEARNING IN FLOOD PREDICTION

Machine learning has revolutionized flood forecasting by enabling predictive models that can learn complex relationships between multiple flood-related variables. Several studies have demonstrated the superior accuracy of ML models over traditional hydrological approaches. Kabir et al. (2020) introduced a deep convolutional neural network (CNN) model trained on hydraulic model outputs to

predict rapid flood inundation. Their study showed that ML models could effectively identify flood-prone areas and estimate flood depths with high precision, making them valuable tools for real-time forecasting.

One of the key challenges in ML-driven flood prediction is the availability of high-quality, labeled data for training models. Liu et al. (2022) successfully applied a Transformer-based dual-encoder model to track river discharge with high accuracy, outperforming Long Short-Term Memory (LSTM) networks. Meanwhile, Xu et al. (2022) enhanced LSTM performance by integrating the Particle Swarm Optimization algorithm, further optimizing flood forecasting capabilities. These studies demonstrate that ML models can offer a viable alternative to traditional hydrological models, especially when trained on diverse, high-resolution datasets.

2.3 DEEP LEARNING AND REMOTE SENSING FOR FLOOD MAPPING

Deep learning models, particularly Convolutional Neural Networks (CNNs) and U-Net architectures, have been widely used in flood extent mapping using satellite and drone imagery. The ability of deep learning models to process large-scale spatial data has significantly improved flood prediction and response strategies.

Paudyal et al. (2023) demonstrated the effectiveness of U-Net in segmenting flood-affected areas from Sentinel-1 SAR imagery. Their study highlighted how deep learning models can achieve high spatial accuracy in detecting flooded regions, providing essential data for emergency response teams. Similarly, Zeng and Bertsimas et al. (2023) proposed a multimodal ML approach that integrated geospatial data with historical disaster information to predict global flood risks, showcasing the advantages of combining multiple data sources for more reliable forecasting.

Despite the advancements in deep learning, challenges remain, particularly concerning the computational requirements for training deep neural networks and the interpretability of model predictions. The black-box nature of deep learning models raises concerns about their usability in critical decision-making processes, necessitating the integration of explainable AI techniques.

2.4 EXPLAINABLE AI (XAI) AND MODEL INTERPRETABILITY

One of the significant challenges in applying AI to flood prediction is ensuring model interpretability. Explainable AI (XAI) techniques aim to make machine learning predictions more transparent and trustworthy. Islam et al. (2021) demonstrated that integrating SHAP (Shapley Additive Explanations) and feature attribution methods could help stakeholders understand how flood prediction models arrive at their conclusions.

By enhancing interpretability, XAI techniques can improve stakeholder confidence in AI-driven flood forecasting systems. Furthermore, the integration of explainability into predictive models allows for better disaster preparedness by making model outputs more actionable for decision-makers.

2.5 ENSEMBLE LEARNING FOR FLOOD PREDICTION

Ensemble learning techniques have emerged as robust solutions for flood prediction, as they combine multiple models to improve forecasting accuracy. Hybrid ensemble models, such as those combining Random Forest (RF) with Support Vector Machines (SVM) and Artificial Neural Networks (ANN), have outperformed traditional single-model approaches (Islam et al., 2021).

Prăvălie and Costache (2013) and Mishra and Sinha (2020) emphasized that climate change and urbanization are major factors contributing to increasing flood risks. Ensemble models can account for these factors by incorporating multiple predictive techniques, making them more resilient to uncertainties in flood forecasting. Dagging and Random Subspace models have been particularly effective in handling complex flood prediction tasks, providing adaptive forecasting solutions.

2.6 FLASH-FLOOD SUSCEPTIBILITY MAPPING BASED ON XGBOOST, RANDOM FOREST AND BOOSTED REGRESSION TREES

Abedi et al. (2021) conducted a comprehensive study to map flash flood susceptibility in the Bâsca Chiojdului River Basin in Romania, a region prone to frequent flash floods. The study utilized a dataset of 962 documented flash flood events and applied four machine learning models: Classification and Regression Tree (CART), Random Forest (RF), Boosted Regression Trees (BRT), and Extreme Gradient Boosting (XGBoost). A total of eleven environmental and topographic factors, including slope, land use, and topographic wetness index, were used as predictors. Among these, slope was found to be the most influential factor.

The performance of each model was evaluated using the Area Under the Curve (AUC) metric. The Random Forest model demonstrated the highest predictive capability, achieving an AUC of 0.956, outperforming BRT (0.899), XGBoost (0.892), and CART (0.868). The study effectively showcased the superiority of ensemble-based models, particularly RF, for accurate flood susceptibility mapping. The findings provide a strong foundation for implementing data-driven flood risk management and early warning systems in vulnerable areas.

2.7 PREDICTION OF FLOOD SENSITIVITY BASED ON LOGISTIC REGRESSION, EXTREME GRADIENT BOOSTING, AND RANDOM FOREST MODELING METHODS

Wu et al. (2024) conducted a flood sensitivity assessment focused on Suqian City, Jiangsu Province, China, using three prominent classification models Logistic Regression (LR), Random Forest (RF), and eXtreme Gradient Boosting (XGBoost). Their goal was to identify high-risk flood areas and evaluate the performance of traditional versus ensemble machine learning models in predicting flood sensitivity. The study used a dataset comprising 12 influential factors, including precipitation, land use, elevation, slope, soil type, and drainage density. These variables were carefully selected through statistical correlation and multicollinearity analysis to ensure model stability and accuracy.

The models were evaluated using a combination of accuracy, precision, recall, F1 score, and AUC (Area Under the Receiver Operating Characteristic Curve). Among the three models, Random Forest outperformed the others with an AUC of 0.932 on the testing dataset, followed closely by XGBoost. Logistic Regression, while still effective, demonstrated lower predictive accuracy, showcasing the limitations of linear models in complex, non-linear flood dynamics.

This study is especially relevant, which focuses on the application of AI models to predict flood risks using structured hydrological data. The comparative approach taken by Wu et al. validates the selection of ensemble models like Random Forest and XGBoost as highly capable tools for flood prediction tasks. Their methodology for variable selection and robust model evaluation provides a clear framework that can be adapted for my own structured data experiments. Furthermore, the urban context of their study closely aligns with my research objective of creating early warning systems and vulnerability mapping in urban or semi-urban flood-prone regions.

By replicating their approach and adapting it with localized data and further enhancements such as synthetic oversampling or integration with unstructured data, my project work can be built upon proven techniques and contribute further to the field of intelligent flood risk prediction systems.

Table 1 Summary of Similar Studies in AI-based Flood Prediction

Study Authors	Models Used	Advantages	Identified Gaps
Kabir et al. (2020)	CNN	Accurate flood depth estimation from hydraulic model outputs	Requires high-quality labeled data; limited generalizability to new regions
Liu et al. (2022)	Transformer-based Dual Encoder	Outperformed LSTM in tracking river discharge	High computational cost; limited interpretability
Xu et al. (2022)	LSTM + Particle Swarm Optimization	Enhanced long-term forecasting performance	May struggle with real-time or abrupt flooding events
Paudyal et al. (2023)	U-Net (Sentinel-1 SAR imagery)	High spatial segmentation accuracy for flood detection	Requires extensive image preprocessing and labeled datasets
Zeng & Bertsimas (2023)	Multimodal ML (geospatial + historica)	Fused multiple data sources for improved global flood prediction	Focused on large-scale global models; lacks regional granularity
Abedi et al. (2021)	CART, RF, BRT, XGBoost	RF achieved highest AUC (0.956). excellent for flash flood susceptibility maps	Heavily dependent on topographical and environmental input data
Wu et al. (2024)	Logistic Regression, RF, XGBoost	RF had AUC of 0.932; suitable for structured urban flood risk prediction	Logistic regression underperformed; ensemble models required more fine-tuning
Nearing et al. (2024)	GloFAS (Hydrological model)	Effective for broad-scale flood forecasting and global alert systems	Less adaptable for local forecasting; cannot integrate unstructured data like imagery

2.8 SUMMARY AND IMPLICATIONS

The literature review highlights the evolution of flood prediction from traditional hydrological models to advanced AI-driven forecasting systems. Traditional models, while useful, lack the adaptability and precision of machine learning approaches. ML and deep learning models, particularly those leveraging remote sensing data, have significantly improved flood forecasting accuracy. However, challenges such as model interpretability, computational efficiency, and data availability remain areas for further exploration.

This research contributes to the field by integrating XGBoost and Random Forest models for structured flood prediction and U-Net deep learning models for flood extent mapping. By leveraging AI and geospatial data, this study aims to enhance real-time flood forecasting capabilities and disaster preparedness. The findings from this literature review provide a foundation for developing a comprehensive methodology, which will be explored in the next chapter.

Chapter 3: METHODOLGY

This chapter outlines the methodological approach used to develop an AI-driven flood prediction and mapping system. The study integrates both structured data-based machine learning models (XGBoost & Random Forest) and unstructured data-based deep learning (U-Net segmentation for flood mapping). The methodology is structured to ensure that the research objectives are met while maintaining scientific rigor and reproducibility. This chapter covers data collection, preprocessing, model development, evaluation metrics, and system deployment to justify the research design and its effectiveness in answering the research questions.

3.1 RESEARCH DESIGN AND RATIONALE

This study follows an applied, experimental, and quantitative approach that leverages artificial intelligence techniques for flood risk prediction and flood extent mapping. The study is experimental in nature as it involves testing and evaluating different AI models to determine their effectiveness in improving flood forecasting. By combining machine learning (ML) and deep learning (DL), the study enhances traditional flood risk assessment by introducing real-time predictive modeling and automated flood segmentation from satellite imagery.

This hybrid approach ensures a comprehensive flood risk management system by combining AI-based structured prediction and deep learning-based visual segmentation.

3.2 METHODOLOGY WORKFLOW

The methodology follows a structured workflow that allows for systematic implementation, evaluation, and deployment of the models:

1. Data Collection: Obtaining structured hydrological and meteorological data and unstructured satellite images.
2. Data Preprocessing: Cleaning structured data, normalizing numerical values, resizing, and augmenting image data.
3. Feature Selection: Identifying relevant attributes using correlation analysis and feature importance ranking.
4. Model Development:
    - Training XGBoost and Random Forest for flood risk prediction.
    - Training U-Net for flood extent segmentation.
5. Model Training & Optimization: Using hyperparameter tuning and cross-validation to improve accuracy.

6. Model Evaluation: Assessing ML and DL models based on key performance metrics.
7. Deployment & GUI Integration: Implementing a Streamlit-based dashboard to provide an interactive platform for real-time flood analysis.

3.3 METHODOLOGICAL CONSIDERATIONS

The methodology is based on the following key considerations:

- Reliability of Data Sources: Ensuring that datasets used for training the models are accurate and well-represented.
- Model Generalizability: Ensuring that AI models can make predictions for new flood events beyond the training dataset.
- Computational Efficiency: Optimizing model training and inference speed to enable real-time flood risk prediction.
- Scalability and Deployment: Developing models that can be integrated into a user-friendly dashboard for accessibility by different stakeholders.

3.4 RESEARCH OBJECTIVES

The research objectives are aligned with the system's development and deployment, ensuring a robust flood prediction framework. The objectives are:

- To develop an AI-driven flood prediction model using XGBoost and Random Forest to analyze structured hydrological data and predict flood risks.
- To implement a U-Net deep learning model for processing satellite imagery and mapping flood-affected regions.
- To integrate structured and unstructured data into a unified flood prediction framework for improved forecasting accuracy.
- To develop an interactive Streamlit dashboard that provides real-time flood risk assessments, predictive alerts, and geospatial flood mapping.
- To evaluate the system's performance using multiple statistical and deep learning evaluation metrics.

3.5 METHODOLOGICAL ASSUMPTIONS

This study is based on several key assumptions:

- Data Accuracy: The structured and unstructured datasets are assumed to be reliable and representative of real-world flood conditions.
- Model Generalizability: The trained ML and DL models can generalize well to new flood events based on historical and remote sensing data.
- Feature Relevance: Selected features from correlation analysis contribute significantly to flood prediction accuracy.
- AI-based Predictions are Interpretable: Explainability techniques like feature importance in ML models make predictions understandable for users.

3.6 DATA COLLECTION AND PREPROCESSING

3.6.1 Data Sources

This research incorporates data from multiple sources to build a robust flood prediction system. The dataset includes:

- Structured Data (Flood Risk Prediction): Obtained from Flood Prediction Dataset on Kaggle. It consists of meteorological and hydrological parameters such as:
    - Rainfall intensity
    - River discharge levels
    - Soil moisture
    - Humidity and temperature
    - Flood probability labels (used as the target variable for classification models)
- Unstructured Data (Flood Segmentation): Obtained from Flood Area Segmentation Dataset on Kaggle. It includes:
    - Satellite images of flooded and non-flooded regions
    - Corresponding segmentation masks that highlight affected areas

By these datasets, the system can predict flood risks based on meteorological conditions and visually map flood extents from satellite imagery.

3.6.2 Preprocessing for Flood Image Segmentation (U-Net)

Before training the U-Net model for flood segmentation, the raw satellite images and their corresponding masks must undergo a series of preprocessing steps to ensure consistency, efficiency, and optimal learning performance. Proper preprocessing helps standardize the dataset, making it more suitable for deep learning training while reducing noise and inconsistencies. The key preprocessing steps include image resizing, normalization, mask binarization, and data augmentation.

3.6.2.1 Standardization of Image Dimensions

Satellite images vary significantly in size, which can introduce inconsistencies during training. Deep learning models, particularly convolutional neural networks (CNNs), require inputs of a fixed size to process data efficiently. Therefore, all images and masks are resized to 256×256 pixels to create a uniform dataset while maintaining spatial integrity. This ensures that the network learns patterns effectively across all images.

```
# Load and preprocess image
img = Image.open(img_path).convert('RGB')
img = img.resize(target_size)
img = np.array(img, dtype=np.float32) / 255.0
```

Figure 1 Loading dataset

By converting the images to RGB format, the model is provided with three channels (Red, Green, and Blue), which are essential for feature extraction. The resizing operation preserves the aspect ratio while fitting the image to the required dimensions, ensuring that no critical information is lost.

3.6.2.2 Normalization of Pixel Values

Raw pixel values in images typically range from 0 to 255, which can lead to high variance and unstable gradients during training. To improve model convergence and stability, pixel values are normalized to a range between 0 and 1. Normalization helps the network process images more effectively by reducing the impact of extreme pixel values and making the optimization process more stable.

By dividing each pixel value by 255.0, all values are mapped to a standardized range, ensuring that no pixel dominates the training process. This transformation significantly improves training efficiency and generalization.

3.6.2.3 Binarization of Segmentation Masks

The segmentation masks, which indicate flooded and non-flooded areas, must be converted into a binary format to ensure that the model can effectively learn the classification task. Since the U-Net model performs pixel-wise classification, the masks must clearly differentiate between the two classes, where 1 represents flooded areas and 0 represents non-flooded areas.

```
# Load and preprocess mask
mask = Image.open(mask_path).convert('L')  # Grayscale
mask = mask.resize(target_size)
mask = np.array(mask, dtype=np.float32) / 255.0

# Binarize the mask
mask = (mask > 0.5).astype(np.float32)

# Expand dims to make it (H, W, 1)
mask = np.expand_dims(mask, axis=-1)
```

Figure 2 Binarize

By converting the mask to grayscale, we ensure that each pixel contains only a single intensity value rather than multiple color channels. The mask is then thresholded at 0.5, meaning that any pixel value greater than 0.5 is classified as 1 (flooded), while any pixel value below 0.5 is set to 0 (non-flooded). This transformation simplifies the learning process for the model by providing a clear, well-defined objective.

3.6.2.4 Data Augmentation for Generalization

To further enhance model robustness, data augmentation is applied to artificially expand the dataset. Augmentation introduces variations into the dataset, making the model more resilient to real-world distortions such as changes in perspective, lighting conditions, and environmental factors. The applied transformations include:

• Horizontal and Vertical Flips: Randomly flipping images ensures that the model does not develop positional bias.

• Rotations: Random 90-degree rotations help the model recognize flood patterns from different orientations.

• Scaling and Shifting: Slight scaling and shifting of images prevent the model from overfitting to specific locations within the frame.

• Brightness and Contrast Adjustments: Helps the model handle variations in lighting conditions in different satellite images.

• Gaussian Noise Addition: Introduces random noise to improve robustness against artifacts.

The augmentation process is implemented using the Albumentations library, which provides efficient and high-performance transformations:

```python
def augment_image_and_mask(image, mask):
    transform = A.Compose([
        A.HorizontalFlip(p=0.5),
        A.VerticalFlip(p=0.5),
        A.RandomRotate90(p=0.5),
        A.ShiftScaleRotate(shift_limit=0.1, scale_limit=0.1, rotate_limit=30, p=0.5),
        A.RandomBrightnessContrast(p=0.2),
        A.GaussNoise(p=0.2)
    ])

    augmented = transform(image=image, mask=mask)
    return augmented['image'], augmented['mask']
```

Figure 3 Augmentation

Each transformation is applied with a certain probability (e.g., p=0.5 means the transformation is applied to 50% of the images), ensuring that augmentation is randomized and does not introduce repetitive patterns into the dataset.

3.6.2.5 Efficient Data Loading Using a Generator

Deep learning models, especially U-Net, require large amounts of data for training, which can lead to memory constraints when loading the entire dataset at once. To address this issue, a data generator is implemented to load and process images in small batches, reducing memory consumption and ensuring efficient training.

```python
def data_generator(images, masks, batch_size=8, augment=False):
    while True:
        for i in range(0, len(images), batch_size):
            batch_images = images[i:i + batch_size]
            batch_masks = masks[i:i + batch_size]

            if augment:
                augmented_images, augmented_masks = [], []
                for img, mask in zip(batch_images, batch_masks):
                    img, mask = augment_image_and_mask(img, mask)
                    augmented_images.append(img)
                    augmented_masks.append(mask)

                batch_images = np.array(augmented_images)
                batch_masks = np.array(augmented_masks)

            yield np.array(batch_images), np.array(batch_masks)
```

Figure 4 Data Generator

The generator:

• Loads images and masks batch by batch instead of loading the entire dataset into memory.

• Applies augmentation dynamically, ensuring that each training batch contains unique variations.

• Yields the processed images and masks for seamless integration with the U-Net training loop.

3.6.2.6 Splitting the Dataset for Training and Validation

To evaluate the model's ability to generalize to unseen data, the dataset is split into 80% training data and 20% validation data. This ensures that the model is trained on a large set of examples while being validated on separate data that it has never seen before.

```
# Assuming 80% train, 20% validation
train_images, val_images = images[:232], images[232:]
train_masks, val_masks = masks[:232], masks[232:]
```

Figure 5 Split Dataset

By ensuring a balanced training-validation split, the model's performance can be effectively monitored, allowing early stopping mechanisms to prevent overfitting.

The preprocessing pipeline is a critical component of the U-Net training process, ensuring that input data is standardized, optimized, and augmented for deep learning. Through resizing, normalization, mask binarization, augmentation, and efficient data loading, the dataset is prepared for segmentation tasks, allowing the U-Net model to learn robust flood detection patterns. These preprocessing steps enhance model accuracy, improve generalization, and reduce training inefficiencies, making the segmentation process more reliable in real-world applications.

3.6.3 Preprocessing for Flood Prediction (Structured Data)

In the initial preprocessing phase of this research project, I began by importing the raw flood dataset into Python. This task was performed using the pandas library, which facilitates efficient handling and analysis of structured data. The following code demonstrates the initial step of loading the dataset:

```
import pandas as pd

#  dataset
df = pd.read_csv("/Users/anandhu/Downloads/Final Project/combined_dataset/structured/flood_data.csv")
```

Figure 6 Loading Dataset Raw

After importing the data, I conducted a thorough inspection to verify its integrity, completeness, and accuracy, specifically emphasizing the Flood Probability column. Initially, flood probabilities were presented in a decimal format (e.g., 0.23 representing a 23% chance). Recognizing the potential difficulty in interpreting

decimal probabilities directly, I transformed these values into percentage terms to enhance readability and interpretability. This transformation was conducted by multiplying each probability by 100 and rounding to two decimal places, as shown in the following code snippet:

```
# Convert FloodProbability to percentage values by multiplying by 100
df["FloodProbability_Percentage"] = (df["FloodProbability"] * 100).round(2)
```

Figure 7 Converting into Percentage value

To confirm the accuracy and effectiveness of this conversion, I carefully reviewed the transformed data side-by-side with the original decimal values:

```
Before conversion:
0    0.450
1    0.475
2    0.515
3    0.520
4    0.475
Name: FloodProbability, dtype: float64

After conversion to percentages:
   FloodProbability  FloodProbability_Percentage
0             0.450                         45.0
1             0.475                         47.5
2             0.515                         51.5
3             0.520                         52.0
4             0.475                         47.5

Updated dataset saved as 'flood_data_updated.csv'.
```

Figure 8 Verifying

Following successful verification, I saved the updated and refined dataset into a new CSV file named flood_data_updated.csv. This action ensured that the original data remained intact for reference while facilitating organized data management for subsequent analytical tasks. The code for this step is as follows:

```
# Save the updated dataset
df.to_csv("/Users/anandhu/Downloads/Final Project/combined_dataset/structured/flood_data_updated.csv", index=False)
print("\nUpdated dataset saved as 'flood_data_updated.csv'.")
```

Figure 9 Saving the Dataset

This structured approach to data preprocessing laid a strong foundation for accurate and efficient data analysis, ensuring clarity and usability in subsequent modeling phases of this dissertation.

3.6.3.1 Outlier and Data Filtering

In the subsequent preprocessing step, I performed additional data cleaning by addressing potential outliers within the dataset. This involved identifying and filtering rows with exceptionally high feature values, as these could skew analytical results or predictive modeling.

Initially, I reloaded the previously processed dataset (flood_data_updated.csv) into Python for further processing:

Next, I identified all numeric feature columns while explicitly excluding the target variables (FloodProbability and FloodProbability_Percentage) to ensure accurate filtering focused only on input features:

```
# Identify numeric feature columns excluding the target variables
features = [col for col in df.select_dtypes(include=["number"]).columns
           if col not in ["FloodProbability", "FloodProbability_Percentage"]]
```

Figure 10 Identifying Numeric Values

I then created a Boolean mask to filter out rows that contained values greater than 10 in any of these feature columns. This threshold was selected to maintain data consistency and minimize the influence of potential outliers

```
# Create a boolean mask that is True for rows where all feature values are <= 10
mask = (df[features] <= 10).all(axis=1)

# Filter the DataFrame using the mask
df_filtered = df[mask]
```

Figure 11 Filtering out the values

After applying this filtering process, I compared the dimensions of the original and filtered datasets to verify the impact of the removal process:

```
Original dataset shape: (50000, 22)
Filtered dataset shape: (37893, 22)
Filtered dataset saved as 'flood_data_filtered.csv'.
```

Figure 12 Again Verifying

Lastly, the filtered dataset was saved into a new CSV file named flood_data_filtered.csv, ensuring a clear distinction between the cleaned data used for detailed analyses and the previously saved datasets This meticulous filtering step significantly enhanced data quality by reducing potential distortions due to outlier values, thereby preparing a robust foundation for subsequent analytical and predictive modeling stages of my dissertation.

3.6.3.2 Feature Selection and Synthetic Data Generation

Continuing the preprocessing stage, I performed a focused feature selection to refine the dataset further for meaningful analysis. After reloading the previously filtered dataset (flood_data_filtered.csv), I removed the original FloodProbability column, retaining only the percentage-based probability column (FloodProbability_Percentage). This decision aimed to standardize the target variable clearly.

```
# Drop the 'FloodProbability' column while keeping 'FloodProbability_Percentage'
df = df.drop(columns=["FloodProbability"], errors="ignore")
```

Figure 13 Dropping Flood Probability

I selected 15 key flood-related features based on graphical correlation analysis, feature importance plots, and domain knowledge. The graphical analysis involved both correlation heatmaps and feature distribution graphs. These visual tools provided clear insights into which features most significantly influenced flood probabilities Features such as MonsoonIntensity, ClimateChange, DeterioratingInfra structure, IneffectiveDisasterPreparedness, and Inadequate distinct patterns and strong associations with high flood probability scenarios. Conversely, features like Dams Quality displayed clear inverse relationships, indicating their role in flood mitigation.

```python
# Generate synthetic high probability data based on feature conditions
def generate_synthetic_high_cases(num_samples):
    data = pd.DataFrame({
        "MonsoonIntensity": np.random.randint(7, 10, num_samples),
        "ClimateChange": np.random.randint(6, 10, num_samples),
        "Landslides": np.random.randint(7, 10, num_samples),
        "DamsQuality": np.random.randint(0, 2, num_samples),
        "CoastalVulnerability": np.random.randint(7, 10, num_samples),
        "IneffectiveDisasterPreparedness": np.random.randint(7, 10, num_samples),
        "InadequatePlanning": np.random.randint(7, 10, num_samples),
        "Deforestation": np.random.randint(6, 10, num_samples),
        "Urbanization": np.random.randint(6, 10, num_samples),
        "Encroachments": np.random.randint(6, 10, num_samples),
        "WetlandLoss": np.random.randint(6, 10, num_samples),
        "AgriculturalPractices": np.random.randint(6, 10, num_samples),
        "DeterioratingInfrastructure": np.random.randint(7, 10, num_samples),
        "PoliticalFactors": np.random.randint(2, 5, num_samples),
        "Watersheds": np.random.randint(6, 10, num_samples),
    })
    data["FloodProbability_Percentage"] = np.random.randint(70, 100, num_samples)
    return data

# Generate synthetic low probability data based on feature conditions
def generate_synthetic_low_cases(num_samples):
    data = pd.DataFrame({
        "MonsoonIntensity": np.random.randint(1, 4, num_samples),
        "ClimateChange": np.random.randint(0, 3, num_samples),
        "Landslides": np.random.randint(0, 3, num_samples),
        "DamsQuality": np.random.randint(7, 10, num_samples),
        "CoastalVulnerability": np.random.randint(0, 3, num_samples),
        "IneffectiveDisasterPreparedness": np.random.randint(0, 3, num_samples),
        "InadequatePlanning": np.random.randint(0, 3, num_samples),
        "Deforestation": np.random.randint(0, 3, num_samples),
        "Urbanization": np.random.randint(0, 3, num_samples),
        "Encroachments": np.random.randint(0, 3, num_samples),
        "WetlandLoss": np.random.randint(0, 3, num_samples),
        "AgriculturalPractices": np.random.randint(0, 3, num_samples),
        "DeterioratingInfrastructure": np.random.randint(0, 3, num_samples),
        "PoliticalFactors": np.random.randint(4, 10, num_samples),
        "Watersheds": np.random.randint(0, 3, num_samples),
    })
    data["FloodProbability_Percentage"] = np.random.randint(0, 30, num_samples)
    return data
```

Figure 14 Generating High and Low synthetic Data

```python
# Generate synthetic high and low probability cases
high_cases = generate_synthetic_high_cases(target_size)
low_cases = generate_synthetic_low_cases(target_size)

# Combine synthetic cases with the existing dataset
df_final = pd.concat([df_final, high_cases, low_cases], ignore_index=True)

df_final.loc[:, "FloodProbability_Percentage"] = (df_final[selected_features].sum(axis=1) / len(selected_features) * 10).round(0).astype(i
```

Figure 15 Combining the data

```
Before synthetic data - Highest Flood Probability: 73%
Before synthetic data - Lowest Flood Probability: 27%

Updated dataset saved as 'synthetic_flood_data.csv'.
After synthetic data - Cases above 60%: 1788
After synthetic data - Cases below or equal to 30%: 1009
```

Figure 16 View of cases added

3.6.3.3 Key Observations Behind Feature Selection

- Monsoon Intensity and Climate Change: Graphs indicated consistently high values correlated with increased flood probabilities, aligning with known environmental impacts.
- Landslides and Coastal Vulnerability: Distribution graphs showed these geographical vulnerabilities as significantly affecting flood risks.
- Dams Quality and Deteriorating Infrastructure: Feature importance analysis revealed these infrastructure factors critically impact flood outcomes.
- Urbanization and Encroachments: Graphical analysis highlighted urban growth and settlements as increasing flood risks by obstructing natural water pathways.
- Wetland Loss and Deforestation: Loss of these natural defenses showed clear visual relationships with higher flood probabilities.
- Political Factors IneffectiveDisasterPreparedness, and InadequatePlanning: visual graphs and plots revealed governance and disaster management as pivotal in determining flood resilience.
- Agricultural Practices and Watersheds: The graphs indicated that mismanagement of agricultural practices and watersheds substantially heightened flood risks.

Due to moderate representation of extreme flood scenarios identified through visual analyses, I generated synthetic data to address this imbalance. This synthetic dataset was crafted to realistically represent very high-risk (70-100%) and very low-risk (0-30%) scenarios:

Finally, the enriched dataset containing both original and synthetic data was saved as synthetic_flood_data.csv, enhancing the dataset's comprehensiveness and reliability for predictive modeling and analysis This comprehensive preprocessing approach, informed by both visual analyses and domain expertise, significantly improved the dataset's balance, representativeness, and overall analytical quality, preparing a robust foundation for subsequent predictive analyses.

Key Observations from Feature Selection:

- Features highly correlated with flood probability (e.g., rainfall, river discharge, soil moisture) were retained for model training.
- Features with low variance or minimal impact on flood prediction (e.g., unrelated environmental factors) were removed.
- Multicollinearity among independent variables was assessed to ensure that redundant features were dropped, reducing overfitting risks.

By selecting only, the most relevant and non-redundant features, the flood prediction model is optimized for better accuracy and generalization.

3.6.3.4 Oversampling and Dataset Balancing

Following the synthetic data generation step, I conducted further preprocessing by addressing the dataset's class imbalance to ensure more robust and reliable predictive modeling. The previously enriched dataset (synthetic_flood_data.csv) was reloaded to carry out this balancing process.

The objective was to achieve a balanced representation of cases across various flood probability levels (low, moderate, and high risk). To accomplish this, I implemented an oversampling technique that created slight variations of existing cases. These variations maintained realistic constraints, with each feature value adjusted minimally to avoid data distortion.

```
def oversample_cases(df, target_size):
    oversampled_data = []
    while len(oversampled_data) < target_size:
        sample = df.sample(n=1, replace=True).iloc[0].copy()
        for col in selected_features:
            variation = np.random.randint(-1, 2)  # Small random variation (-1, 0, or +1)
            sample[col] = max(0, min(10, sample[col] + variation))
        oversampled_data.append(sample)
    return pd.DataFrame(oversampled_data)
```

Figure 17 Oversampling the data

I defined the target size for balancing by aligning the number of moderate-risk cases (probability between 30% and 60%) to guide the volume of oversampling needed for low-risk (≤30%) and high-risk (>60%) scenarios:

```
After oversampling - Cases above 60%: 38884
After oversampling - Cases below or equal to 30%: 38105
After oversampling - Cases between 30% - 60%: 37096
```

Figure 18 Oversampled data

```python
# Define target size to balance classes
target_size = df_final[(df_final["FloodProbability_Percentage"] > 30) & (df_final["FloodProbability_Percentage"] <= 60)].shape[0

# Separate cases by flood probability categories
high_cases = df_final[df_final["FloodProbability_Percentage"] > 60]
low_cases = df_final[df_final["FloodProbability_Percentage"] <= 30]

# Oversample high and low probability cases
high_cases_oversampled = oversample_cases(high_cases, target_size)
low_cases_oversampled = oversample_cases(low_cases, target_size)

# Combine all cases into a balanced dataset
df_balanced = pd.concat([df_final, high_cases_oversampled, low_cases_oversampled], ignore_index=True)
```

Figure 19 Defining the target size and combining

These oversampled cases were then combined with the existing dataset, resulting in a comprehensive, balanced dataset, which was saved as balanced_flood_data.csv:

The final dataset class distribution was verified to confirm successful balancing across the different flood probability categories This meticulous oversampling procedure significantly improved data representativeness and class balance, setting a robust foundation for precise and effective flood risk prediction in subsequent analytical phases of my research.

Chapter 4: Model Development

In this chapter, I describe the process of developing machine learning and deep learning models for predicting flood risk and mapping flood-prone areas. The models were constructed using the preprocessed and balanced dataset prepared in the preceding phase. The objective was to create a system that not only forecasts the probability of flooding based on hydrological and environmental data but also provides visual representations of the area's most susceptible to flooding. This chapter is divided into two main sections: Section 4.1 focuses on the machine learning models, specifically Random Forest and XGBoost, utilized for flood risk prediction based on structured data. Section 4.2 presents the deep learning model, U-Net, which was applied to satellite imagery for the spatial mapping of flood-affected regions.

4.1 MACHINE LEARNING MODELS FOR FLOOD PREDICTION

In this phase, I developed machine learning models aimed at accurately predicting flood risk based on the prepared and balanced dataset. After careful consideration of various algorithms suitable for classification tasks, I chose two robust models: Random Forest Classifier and XGBoost Classifier. These models were selected based on their proven ability to handle complex, non-linear relationships, high-dimensional data, and their interpretability, making them highly suitable for this research project on flood prediction.

4.1.1 Random Forest Classifier

The Random Forest Classifier is an ensemble learning method widely recognized for its stability and accuracy in classification tasks. It builds multiple decision trees during training and aggregates their results to improve overall predictive performance and reduce overfitting. To optimize its performance on my dataset, I applied hyperparameter tuning using GridSearchCV, exploring combinations of parameters such as the number of estimators (n_estimators), tree depth (max_depth), and minimum samples required for splits (min_samples_split) and leaf nodes (min_samples_leaf).

```python
# Split into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Hyperparameter Tuning for Random Forest
rf_param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

rf_grid_search = GridSearchCV(RandomForestClassifier(random_state=42), rf_param_grid, cv=3, n_jobs=-1, verbose=2)
rf_grid_search.fit(X_train, y_train)
```

Figure 20 Defining Hyperparameter's

The chosen hyperparameters were selected based on their influence on the model's ability to generalize to unseen data. For instance, n_estimators determines how many decision trees are built; a higher number typically enhances stability. max_depth controls the complexity of each tree, balancing between overfitting and underfitting, while parameters like min_samples_split and min_samples_leaf directly affect how trees grow, further controlling the generalization capacity.

4.1.2 Xgboost classifier

The XGBoost (Extreme Gradient Boosting) Classifier was selected for its strong predictive performance, efficient computation, and effective handling of missing values and data imbalance, factors particularly relevant to flood prediction. Similar to Random Forest, XGBoost is a tree-based ensemble method but utilizes gradient boosting, systematically improving predictions by focusing more on incorrectly classified instances from previous iterations.

```python
# Hyperparameter Tuning for XGBoost
xgb_param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 6, 10],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}

xgb_grid_search = GridSearchCV(XGBClassifier(use_label_encoder=False, eval_metric='mlogloss', random_state=42),
                               xgb_param_grid, cv=3, n_jobs=-1, verbose=2)
xgb_grid_search.fit(X_train, y_train)

# Best XGBoost Model
best_xgb_model = xgb_grid_search.best_estimator_
y_pred_xgb = best_xgb_model.predict(X_test)
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
```

Figure 21 Hyperparameters for XGBoost

Hyperparameter tuning for XGBoost included adjustments to parameters like the number of estimators (n_estimators), learning rate (learning_rate), maximum depth of trees (max_depth), and sampling ratios (subsample, colsample_bytree), enhancing its predictive accuracy and robustness.

These hyperparameters were carefully selected due to their crucial roles in model performance: n_estimators balances model complexity and computational cost, while max_depth and learning_rate directly affect how quickly and accurately the model learns from data. Parameters such as subsample and colsample_bytree were included to ensure diversity among trees, preventing overfitting and promoting robust predictions.

Following model training, the best-performing models were saved for future deployment and validation purposes, ensuring reproducibility and facilitating potential real-world applications.

```
# Save the trained models
joblib.dump(best_rf_model, "/Users/anandhu/Downloads/Final Project/combined_dataset/structured/random_forest_model.pkl")
joblib.dump(best_xgb_model, "/Users/anandhu/Downloads/Final Project/combined_dataset/structured/xgboost_model.pkl")
print("Models saved successfully!")
```

Figure 22 Saving the model

Through careful selection, tuning, and validation, this model development stage provided a solid foundation for effectively predicting flood risks, significantly contributing to the research objectives of this dissertation.

4.2 DEEP LEARNING MODEL FOR FLOOD MAPPING

Flood extent mapping requires a semantic segmentation model capable of identifying flood-affected areas from satellite imagery. U-Net is chosen due to its ability to preserve spatial information while segmenting flooded and non-flooded regions.

4.2.1 U-Net Model Architecture and Compilation

Image segmentation is a crucial task in satellite image analysis, requiring a model capable of distinguishing between flooded and non-flooded regions with high precision. In this study, a U-Net architecture is employed due to its efficiency and effectiveness in image segmentation tasks, particularly in medical and satellite imagery. U-Net is a fully convolutional neural network (CNN) that follows an encoder-decoder structure, making it particularly well-suited for pixel-wise classification problems.

4.2.2 U-Net Architecture

The U-Net model consists of three main components: the encoder (downsampling path), the bottleneck layer, and the decoder (upsampling path).

1. Encoder (Downsampling Path):

The encoder is responsible for extracting important spatial features from input images. It comprises multiple convolutional layers, each followed by ReLU activation and max-pooling operations. As the input image progresses through the encoder, its spatial resolution decreases while its feature depth increases, allowing the model to capture high-level representations. The encoder efficiently encodes the essential structural and contextual information needed for segmentation.

2. Bottleneck Layer:

The bottleneck layer represents the deepest and most compressed representation of image features. It serves as a bridge between the encoder and the decoder, containing the most informative features extracted from the image. This layer consists of two consecutive convolutional layers, ensuring that relevant spatial information is preserved while reducing the computational complexity of subsequent upsampling operations.

3. Decoder (Upsampling Path):

The decoder is responsible for reconstructing the segmented output by gradually increasing the spatial resolution of the feature maps. To retain fine-grained details lost during downsampling, the decoder incorporates skip connections, which directly transfer feature maps from corresponding encoder layers to the decoder. These skip connections allow the model to retain essential spatial information, ensuring

accurate segmentation boundaries. The final output layer employs a sigmoid activation function, producing a binary mask where each pixel is classified as either flooded (1) or non-flooded (0).

The U-Net architecture is implemented as follows:

```
def build_unet(input_shape=(256, 256, 3)):
    inputs = Input(input_shape)

    # Encoder (Downsampling)
    c1 = Conv2D(16, (3, 3), activation='relu', padding='same')(inputs)
    c1 = Conv2D(16, (3, 3), activation='relu', padding='same')(c1)
    p1 = MaxPooling2D((2, 2))(c1)

    c2 = Conv2D(32, (3, 3), activation='relu', padding='same')(p1)
    c2 = Conv2D(32, (3, 3), activation='relu', padding='same')(c2)
    p2 = MaxPooling2D((2, 2))(c2)

    c3 = Conv2D(64, (3, 3), activation='relu', padding='same')(p2)
    c3 = Conv2D(64, (3, 3), activation='relu', padding='same')(c3)
    p3 = MaxPooling2D((2, 2))(c3)

    c4 = Conv2D(128, (3, 3), activation='relu', padding='same')(p3)
    c4 = Conv2D(128, (3, 3), activation='relu', padding='same')(c4)
    p4 = MaxPooling2D((2, 2))(c4)

    # Bottleneck
    c5 = Conv2D(256, (3, 3), activation='relu', padding='same')(p4)
    c5 = Conv2D(256, (3, 3), activation='relu', padding='same')(c5)

    # Decoder (Upsampling)
    u6 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(c5)
    u6 = concatenate([u6, c4])
    c6 = Conv2D(128, (3, 3), activation='relu', padding='same')(u6)
    c6 = Conv2D(128, (3, 3), activation='relu', padding='same')(c6)

    u7 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(c6)
    u7 = concatenate([u7, c3])
    c7 = Conv2D(64, (3, 3), activation='relu', padding='same')(u7)
    c7 = Conv2D(64, (3, 3), activation='relu', padding='same')(c7)

    u8 = Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(c7)
    u8 = concatenate([u8, c2])
    c8 = Conv2D(32, (3, 3), activation='relu', padding='same')(u8)
    c8 = Conv2D(32, (3, 3), activation='relu', padding='same')(c8)

    u9 = Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='same')(c8)
    u9 = concatenate([u9, c1])
    c9 = Conv2D(16, (3, 3), activation='relu', padding='same')(u9)
    c9 = Conv2D(16, (3, 3), activation='relu', padding='same')(c9)

    outputs = Conv2D(1, (1, 1), activation='sigmoid')(c9)

    model = Model(inputs, outputs)
    return model
```

Figure 23 U-net Architecture

4.2.3 Model Compilation and Optimization

To ensure effective learning and stable optimization, the model is compiled with the Adam optimizer, a widely used optimization algorithm known for its adaptive learning rate capabilities. The loss function is set to binary cross-entropy dice loss, a combination of binary cross-entropy (BCE) and the Dice coefficient loss. This hybrid loss function enhances segmentation performance by addressing both pixel-wise classification accuracy (BCE) and overall shape similarity (Dice loss).

The model is compiled using the following configuration:

```
unet_model = build_unet(input_shape=(256, 256, 3))

unet_model.compile(
    optimizer=Adam(learning_rate=0.0001),
    loss=sm.losses.bce_dice_loss,
    metrics=['accuracy', iou_metric, dice_metric]
)
```

Figure 24 Adam Optimizer

Adam optimizer: Ensures efficient weight updates and prevents vanishing/exploding gradients.

• Binary cross-entropy dice loss: Combines pixel-wise accuracy with shape-based loss for better segmentation.

• IoU and Dice metrics: Provide robust evaluation of segmentation accuracy.

This structured approach ensures that the U-Net model effectively learns spatial relationships, accurately distinguishing flooded regions in satellite imagery while preserving fine-grained details essential for high-precision segmentation.

4.2.4 U-Net Model Training & Optimization

After constructing and validating the U-Net model, further refinement is necessary to enhance segmentation accuracy and generalization. This refinement process, known as fine-tuning, involves reloading the previously trained model and continuing training with a significantly reduced learning rate. Fine-tuning allows the model to make small but meaningful adjustments to its learned parameters, improving performance without drastically altering its feature representations.

Loading the Pre-Trained U-Net Model

The fine-tuning process begins by loading the pre-trained U-Net model, previously saved as unet_custom_best.h5. Since the model was trained with custom evaluation metrics such as Intersection over Union (IoU) and Dice score, as well as a custom binary cross-entropy dice loss function, these components must be explicitly defined when reloading the model to ensure compatibility. The pre-trained model provides a strong foundation, allowing fine-tuning to focus on refining segmentation boundaries rather than learning features from scratch.

```
from tensorflow.keras.models import load_model

unet_model = load_model(
    'unet_custom_best.h5',
    custom_objects={
        'iou_metric': iou_metric,
        'dice_metric': dice_metric,
        'binary_crossentropy_plus_dice_loss': sm.losses.bce_dice_loss
    }
)
```

Figure 25 Pre-Trained Model

Recompiling the Model with a Reduced Learning Rate

Once the model is loaded, it is recompiled using the Adam optimizer, but with a very low learning rate of 5e-6 (0.000005). The purpose of this reduced learning rate is to prevent large weight updates that could destabilize the already learned patterns. Instead, it enables the model to gradually adjust its parameters, ensuring subtle refinements in segmentation performance. The same binary cross-entropy dice loss function is retained, ensuring consistency in optimization while continuing to monitor segmentation accuracy through IoU and Dice metrics.

```
from tensorflow.keras.optimizers.legacy import Adam

unet_model.compile(
    optimizer=Adam(learning_rate=5e-6),  # Very low learning rate
    loss=sm.losses.bce_dice_loss,
    metrics=['accuracy', iou_metric, dice_metric]
)
```

Figure 26 Recompiling

Fine-Tuning the Model

The fine-tuning phase consists of an additional 10 training epochs, during which the model is exposed to the same training and validation data. Data generators are utilized to efficiently load small batches of augmented images and their corresponding masks, thereby reducing memory constraints. Several callback mechanisms are integrated into the training process to enhance efficiency and prevent overfitting. These include early stopping, which terminates training if validation loss stagnates; learning rate reduction, which dynamically lowers the learning rate when improvements plateau; and model checkpointing, which saves the best-performing fine-tuned model.

```
fine_tune_history = unet_model.fit(
    train_gen,
    steps_per_epoch=steps_per_epoch,
    epochs=10,
    validation_data=val_gen,
    validation_steps=validation_steps,
    callbacks=[early_stopping, reduce_lr, model_checkpoint]
)
unet_model.save('unet_finetuned_best.h5')
```

Figure 27 Fine Tuning

This optimized U-Net model can now be deployed for real-world flood segmentation tasks, ensuring precise identification of flooded regions in satellite imagery. The combination of fine-tuning with a reduced learning rate, data augmentation, and optimized callbacks significantly enhances the model's ability to segment images with improved accuracy while maintaining computational efficiency.

4.3 STREAMLIT APPLICATION DEVELOPMENT

As part of my dissertation project, I developed a user-friendly, interactive web application called the "Flood Risk Assessment Tool" using Streamlit. The primary purpose of this application was to translate complex AI models into an accessible, intuitive platform that allows users—including emergency planners, government officials, and the public to easily understand and assess flood risk scenarios.

The decision to use Streamlit was based on its simplicity, flexibility, and seamless integration with Python, enabling real-time predictions and interactive visualization without requiring extensive technical knowledge from users. Recognizing the need for clear, actionable insights, the application was specifically designed with user experience at the forefront, employing interactive sliders and visual feedback to ensure ease of use and engagement.

The key functionalities of this application include:

- Flood Risk Prediction: Users can input relevant environmental and infrastructural details through straightforward, interactive sliders representing factors such as monsoon intensity, climate change impacts, urbanization, and infrastructure quality. The tool immediately provides clear, understandable predictions—categorizing flood risk into Low, Medium, or High—using the trained Random Forest and XGBoost machine learning models.

- Image Segmentation for Flooded Areas: Users can upload images of potentially flood-affected regions. The application uses a deep learning U-Net model to automatically identify and visually highlight areas impacted by flooding, helping users quickly grasp the severity and extent of flooding in a visual and intuitive manner.

- Customized Visual Experience: To further enhance usability, the application includes appealing visuals, clear navigation, and supportive descriptions that guide users step-by-step through the predictive and visualization processes.

```python
elif menu == "🖼️ Image Segmentation":
    st.write("## 🖼️ Upload Image for Segmentation")
    uploaded_image = st.file_uploader("Upload a Flood-Affected Area Image (jpg, png)", type=["jpg", "png", "jpeg"])

    if uploaded_image:
        image = Image.open(uploaded_image).convert("RGB")
        prediction = predict_segmentation(image)

        st.write("### 🎛️ Adjust Visualization")
        threshold = st.slider("Threshold", 0.0, 1.0, 0.5, 0.05)
        opacity = st.slider("Mask Opacity", 0.0, 1.0, 0.5, 0.05)

        # Create a binary mask based on threshold
        mask = (prediction > threshold).astype(np.uint8)

        st.write("### 🖼️ Original Image")
        st.image(image, use_column_width=True)

        st.write("### 🟢 Segmentation Mask")
        st.image(mask * 255, use_column_width=True)

        st.write("### 🖌️ Overlay Image + Mask")
        fig, ax = plt.subplots()
        ax.imshow(image)
        ax.imshow(mask, cmap="jet", alpha=opacity)
        ax.axis("off")
        st.pyplot(fig)
```

Figure 28 Image Segmentation Page

```python
elif menu == "📊 Flood Prediction":
    st.markdown("## 🌊 Flood Risk Assessment")
    st.write("Enter the environmental conditions below to predict the flood risk or select a default case.")

    selected_case = st.selectbox("Choose a default risk case", ["None", "Low Risk", "Medium Risk", "High Risk"])


    if selected_case != "None":
        feature_values = default_cases[selected_case]
    else:
        feature_values = {key: 5 for key in default_cases["Medium Risk"].keys()}

    st.write("### Default Feature Values")
    st.dataframe(pd.DataFrame(feature_values, index=[0]), use_container_width=True)

    col1, col2 = st.columns(2)

    with col1:
        st.markdown("### 🌧️ Weather & Climate Conditions")
        with st.expander("Expand to enter details"):
            monsoon_intensity = st.slider('Monsoon Intensity 🌧️', 0, 10, feature_values["MonsoonIntensity"], format="%d")
            climate_change = st.slider('Climate Change Impact 🌍', 0, 10, feature_values["ClimateChange"], format="%d")
            coastal_vulnerability = st.slider('Coastal Vulnerability 🏖️', 0, 10, feature_values["CoastalVulnerability"], format="%d")
            deforestation = st.slider('Deforestation 🌲', 0, 10, feature_values["Deforestation"], format="%d")
            landslides = st.slider('Landslide Risk ⛰️', 0, 10, feature_values["Landslides"], format="%d")
            watersheds = st.slider('Watershed Condition 🌊', 0, 10, feature_values["Watersheds"], format="%d")
```

Figure 29 Flood Prediction Page

4.4 SUMMARY

This chapter outlines the research design used to develop an AI-based system for flood risk prediction and mapping. The design closely follows the research questions and objectives, focusing on how artificial intelligence can improve the accuracy and reliability of flood forecasting. The approach taken is applied, experimental, and data-driven, using both structured hydrological data and unstructured satellite imagery to train the models.

To build a reliable system, several steps were taken to prepare the data. These included cleaning the dataset, selecting the most important features, handling missing or imbalanced values, and generating synthetic data where needed. Outliers were removed, and flood-related variables were chosen based on visual analysis and feature importance techniques. Oversampling methods were also used to ensure that both high-risk and low-risk flood scenarios were well represented in the dataset.

The model development process involved two types of AI techniques. On the one hand, Random Forest and XGBoost were used to predict flood risks from structured data like rainfall, drainage, and river levels. These models were chosen for their ability to handle complex patterns and deliver accurate results. Their parameters were carefully adjusted to improve performance. On the other hand, the U-Net deep learning model was used to analyze satellite images and identify flood-prone areas on a map. This helped in addressing the third research question, which focuses on the spatial mapping of flood risks.

To make the system practical and user-friendly, an interactive web app was developed using Streamlit. This tool allows users to enter real-world data such as environmental or infrastructure-related factors and receive instant flood risk predictions. It also includes visual flood maps generated by the U-Net model, making the results easy to understand and apply in real-life decision-making.

In all, this chapter presents a complete and practical approach to developing an AI-powered flood prediction system. By carefully processing the data, selecting suitable models, and creating an easy-to-use interface, the research sets the stage for the next chapter, where the models will be tested and evaluated based on performance metrics.

Chapter 5: Results and Discussion

This chapter presents and discusses the findings of the analyses conducted in this study. It begins by outlining key results from data preprocessing and feature selection, including insights derived from graphical and statistical evaluations. Following this, the chapter details outcomes from machine learning models (Random Forest and XGBoost) and the deep learning approach (U-Net), linking these findings directly to the research objectives outlined in Chapter 1.

Results are illustrated using tables, graphs, and charts for clarity. Each section critically evaluates findings in relation to existing literature from Chapter 2, emphasizing practical implications and contributions to artificial intelligence and flood risk management. Finally, the chapter identifies study limitations and proposes directions for future research, presented in clear, accessible language.

5.1 ANALYSIS AND INTERPRETATION OF DATA

The initial analysis of the dataset provided significant insights into flood risk prediction factors. The dataset, comprising 50,000 records and 21 numeric features, included critical elements such as Monsoon Intensity, Climate Change, Dams Quality, Urbanization, and Political Factors. A detailed correlation and graphical analysis revealed strong associations between flood probabilities and environmental variables (Monsoon Intensity and Climate Change), infrastructural factors (Deteriorating Infrastructure and Dams Quality), and socio-political conditions (Ineffective Disaster Preparedness and Inadequate Planning).

Further exploratory data analysis highlighted moderate imbalances, particularly in extreme flood risk scenarios. The histogram illustrating Flood Probability Distribution (Fig. 30) revealed a nearly normal distribution with most records concentrated around moderate flood risk (approximately 0.5 probability). More specifically, a significant number of cases (over 80%) fell within the mid-range of 0.3 to 0.6 flood probability. In contrast, fewer cases were observed at the extremes: only 864 records were below 0.3 and 1,133 records exceeded 0.6. This imbalance presented a challenge for modeling as the AI systems may become biased toward predicting moderate-risk outcomes.

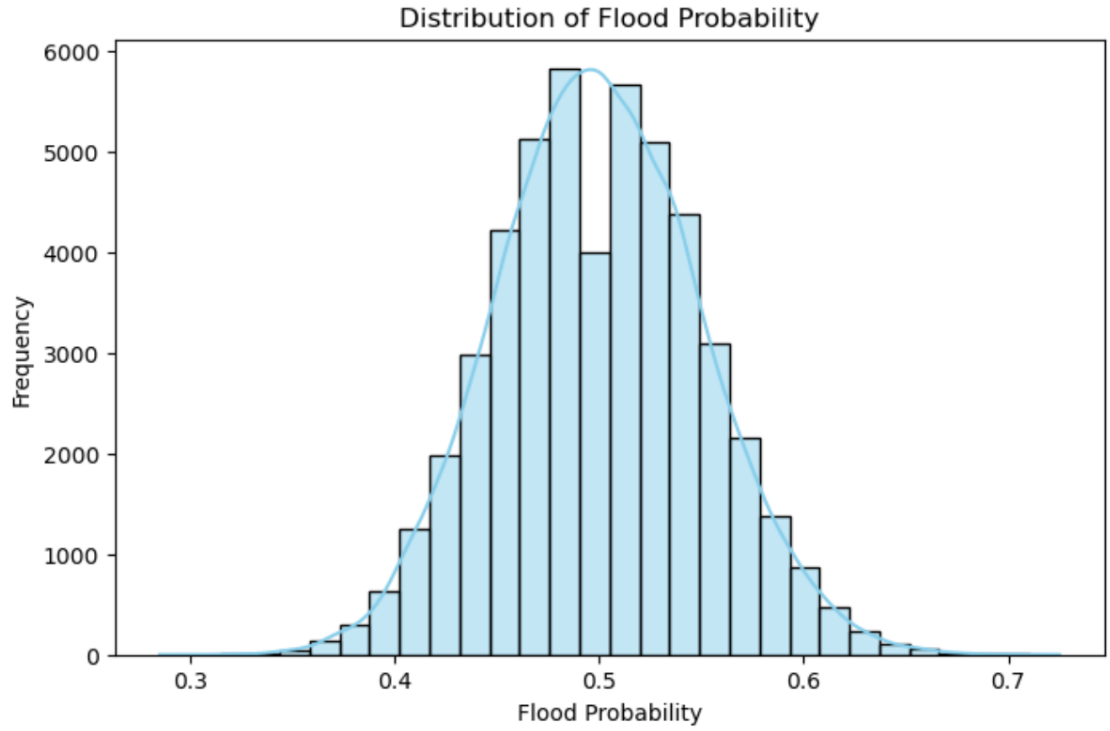


Figure 30 Flood Probability Distribution

To address this issue, synthetic data generation and oversampling techniques were employed, effectively increasing the representation of both high-risk and low-risk scenarios. These enhancements not only improved class balance but also allowed the AI models to learn from a more comprehensive dataset that accurately reflects real-world conditions. This step proved critical for achieving better generalization and ensuring that the predictive models did not underperform in less frequent yet critical scenarios.

Boxplots (Fig. 31 and Fig. 32) provided additional insights by revealing substantial variability and the presence of outliers within critical flood-related features, such as Monsoon Intensity, Climate Change, Dams Quality, and Urbanization. Outliers in these features likely represent significant deviations in environmental and infrastructural conditions associated with extreme flood risk.

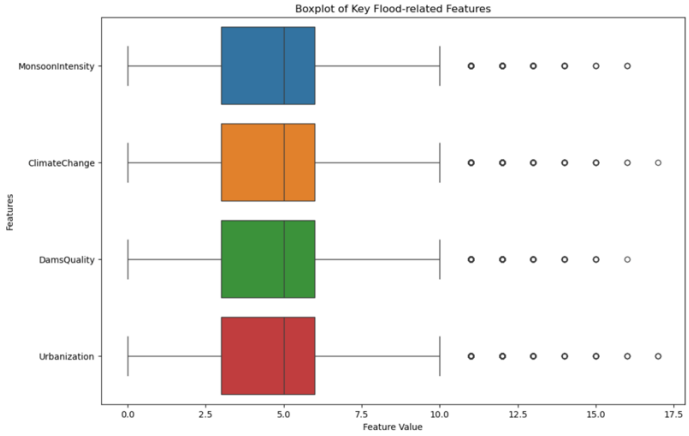


Figure 31 Box plot Flood related Features

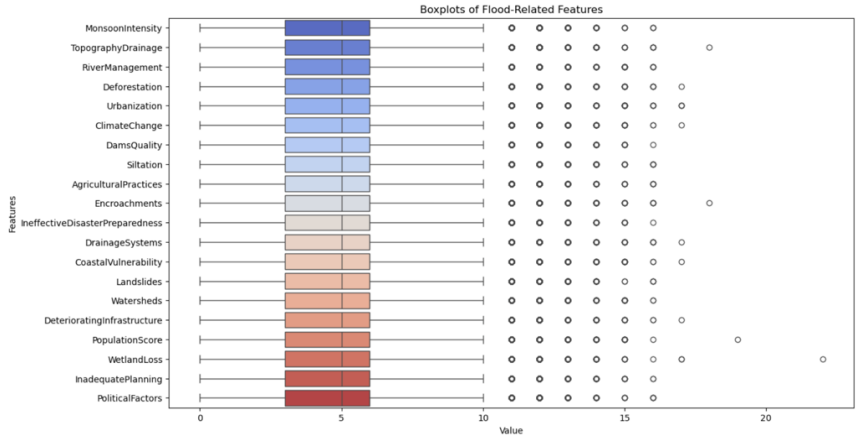


Figure 32 Box plot Flood related Features

A direct comparison between high and low flood-risk cases, specifically regarding Monsoon Intensity (Fig. 33), illustrated clear distinctions. High-risk scenarios showed notably higher intensity, validating the significance of intense monsoon events in elevating flood risks.

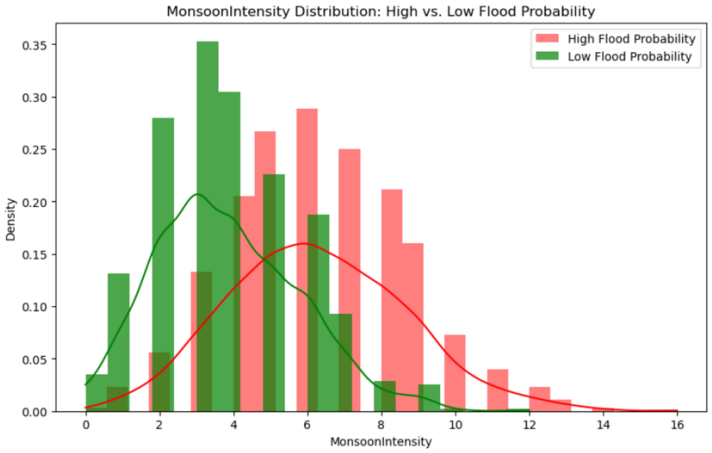


Figure 33 Monsoon Intensity low vs high

The correlation heatmap (Fig. 34) effectively depicted significant relationships among features like Deteriorating Infrastructure, Topography Drainage, River Management, Watersheds, and Dams Quality. These interrelated factors exhibited strong and consistent correlations with flood probability (approximately 0.23), confirming their collective importance in predictive modeling.

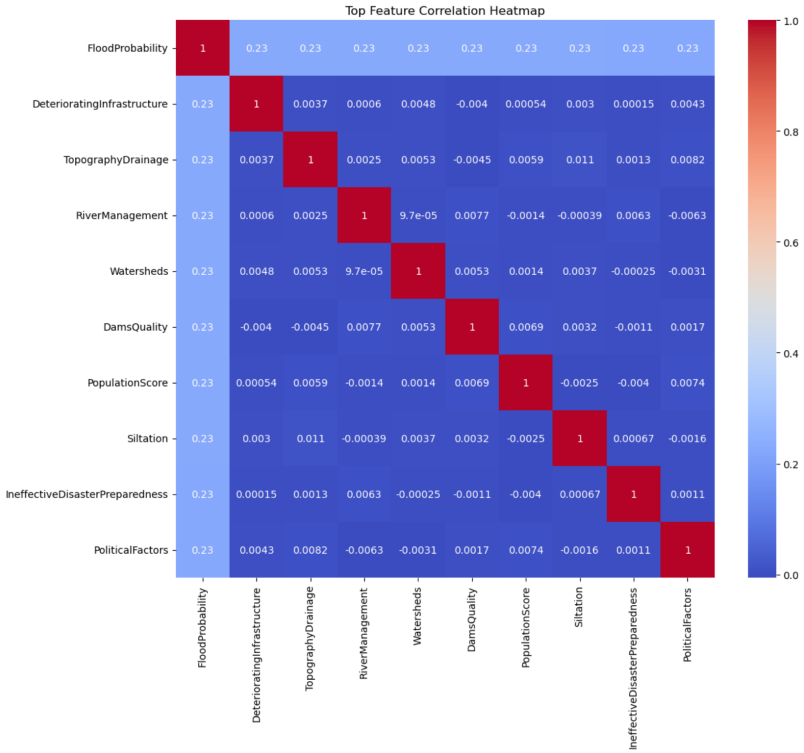


Figure 34 Correlation Heatmap

Finally, a statistical summary (Fig. 35) detailed quantitative distributions across features, further emphasizing considerable variability, particularly notable in Population Score and Wetland Loss. This variability underscores their potential as distinguishing factors in flood risk scenarios.

```
Number of Features: 20
Total Number of Cases: 50000
High Flood Probability Cases (>0.6): 1133
Low Flood Probability Cases (<0.4): 864
```

Statistical Description of Dataset:

	MonsoonIntensity	TopographyDrainage	RiverManagement	Deforestation
count	50000.000000	50000.000000	50000.00000	50000.000000
mean	4.991480	4.984100	5.01594	5.008480
std	2.236834	2.246488	2.23131	2.222743
min	0.000000	0.000000	0.00000	0.000000
25%	3.000000	3.000000	3.00000	3.000000
50%	5.000000	5.000000	5.00000	5.000000
75%	6.000000	6.000000	6.00000	6.000000
max	16.000000	18.000000	16.00000	17.000000

	Urbanization	ClimateChange	DamsQuality	Siltation
count	50000.000000	50000.000000	50000.00000	50000.000000
mean	4.989060	4.988340	5.01536	4.988600
std	2.243159	2.226761	2.24500	2.232642
min	0.000000	0.000000	0.00000	0.000000
25%	3.000000	3.000000	3.00000	3.000000
50%	5.000000	5.000000	5.00000	5.000000
75%	6.000000	6.000000	6.00000	6.000000
max	17.000000	17.000000	16.00000	16.000000

Figure 35 Statistical Description

Overall, these analyses and visualizations collectively demonstrate the importance of comprehensive data examination and rigorous preprocessing in achieving accurate and insightful predictive modeling for flood risk assessment.

5.2 MACHINE LEARNING MODEL EVALUATION

In this section, two robust ensemble-based classifiers Random Forest (RF) and Extreme Gradient Boosting (XGBoost) were evaluated for their performance in classifying flood risk into Low, Medium, and High categories. These models were trained using a structured dataset composed of various hydrological and environmental indicators, with the aim of developing an accurate and interpretable risk classification framework.

Both models were trained using a stratified data split to maintain class balance and tuned using GridSearchCV to optimize hyperparameters such as the number of estimators, maximum depth, learning rate, and regularization terms. This ensured both fairness and robustness in model training and evaluation.

5.2.1 XGBoost (XGB) Evaluation

Extreme Gradient Boosting (XGBoost or XGB) is a robust and scalable ensemble learning technique based on decision trees. It is designed to optimize performance through gradient boosting while maintaining high efficiency. In this study, XGBoost was employed to classify flood risk into three categories — Low, Medium, and High using structured environmental and hydrological data.

5.2.1.1 Hyperparameter Tuning and Final Configuration

Hyperparameter tuning is essential for extracting optimal performance from machine learning models. For XGBoost, a grid search method was applied to test multiple combinations of key parameters, including tree depth, number of trees, and learning rate. The final model was chosen based on cross-validation performance and generalization on the validation set. To maximize the predictive performance of the XGBoost model while maintaining generalization, I conducted systematic hyperparameter tuning using GridSearchCV. XGBoost is a powerful gradient boosting framework known for its ability to handle structured/tabular data with high accuracy, but its performance is highly sensitive to its hyperparameters.

Table 2 Why? GridsearchCV

Reason	**Explanation**
Exhaustive Search	Ensures all Parameter combinations are evaluated fairly.
Cross-validation integration	Uses k-fold validation to prevent overfitting and ensure robust selection.
Handles Bias-variance Tradeoff	Helps balance model Complexity and generalization
Compatible with sklearn	Seamlessly integrates with scikit-learn for pipeline optimization

Table 3 Hyperparameter Values

Hyperparameter	**Values tested**	**Purpose**
N_estimators (Trees)	50,100,150	Controls the numbers of boosting's rounds. More trees can improve accuracy
Max_depth	3,5,6	Controls tree complexity prevents overfitting on small data variations
Learning_rate	0.01,0.1	Determines how quickly instances to use per tree adds randomness
Subsample	0.8,1.0	Fraction of training instances to use per tree. Adds randomness
Colsample_bytree	0.8,1.0	Fraction of features to use per tree. Helps reduce feature dominance

The table below outlines the final tuned hyperparameters used for the XGBoost model:

Table 4 Final tuned hyperparameters

Hyperparameter	**Selected Values**	**Explanation**
N_estimators (Trees)	100	Total Number of trees Affects models complexity and training time
Max_depth	6	Maximum depth of each decision tree Controls model complexity and overfitting
Learning_rate	0.1	Controls the contribution of each tree to final prediction. Smaller values make learning more gradual and stable
Subsample	1.0	Fraction of samples used per tree Helps reduce overfitting
Colsample_bytree	1.0	Fraction of features used per tree. Ensures diversity among trees

These settings were selected after evaluating several combinations. The chosen parameters offered a strong balance between model accuracy, training time, and generalizability. This configuration allowed the XGBoost model to strike a balance between model flexibility and generalization ability. The moderate depth of 6 ensured that trees were not too complex, while the learning rate of 0.1 allowed stable convergence without overshooting minima.

5.2.1.2 Performance Summary

Once tuned, the XGBoost model was tested on the held-out test dataset. It produced highly promising results:

Table 5 XGBoost Performance Summary

Metric	**XGBOOST Result**	**Interpretation**
Accuracy	99.21%	Indicated overall correct classification. Slightly Higher than Random Forest
Log Loss	0.0213	Measures the confidence of predictions. Lower is better this value shows excellent calibration
AUC	0.9998	Represents ability to distinguish between classes. Near-perfect classification capability
Precision (per class)	High	Strong performance, particularly for medium risk class. Slightly less precise in distinguishing High Risk Compared to low/Medium

The XGBoost model demonstrated slightly better performance than Random Forest in terms of both log loss and overall accuracy, indicating more confident and well-calibrated predictions. One of the key strengths of XGBoost lies in its gradient-based optimization and built-in regularization techniques, which contributed to a faster convergence rate during training. This efficiency makes it especially valuable in time-sensitive scenarios, such as flood prediction systems where quick and accurate model training is crucial.

The lower log loss achieved by XGBoost is particularly beneficial for decision support systems that rely not only on accurate classifications but also on the confidence levels of those predictions. This is vital in applications such as early flood warning systems, where overconfident but incorrect classifications could have serious real-world consequences. The ROC curves illustrated that the model achieved near-perfect class separation across all three flood risk levels Low, Medium, and High signifying excellent sensitivity and specificity. Additionally, the Precision-Recall analysis confirmed the model's high reliability, especially in identifying Medium and High-risk zones while minimizing false positives, which is crucial for risk mitigation and emergency response planning.

in conclusion, XGBoost emerged as a fast, highly accurate, and dependable model for classifying structured flood risk levels. Its superior performance validates the power of ensemble-based boosting techniques, particularly when applied to a well-balanced and thoughtfully engineered environmental dataset. The model's ability to produce confident and reliable predictions makes it an excellent candidate for deployment in real-world flood risk management systems. Its integration into early-warning dashboards and disaster response platforms can offer timely, data-driven insights, ultimately aiding authorities in making informed and proactive decisions.

5.2.2 Random Forest (RF) Evaluation

The Random Forest (RF) model was selected for its ensemble-based approach, which aggregates predictions from multiple decision trees to enhance overall performance and reduce overfitting. It was trained on the structured flood dataset using a stratified train-test split to ensure balanced representation of the three flood risk categories: Low, Medium, and High.

Table 6 Final Scores

Metric	score
Accuracy	99.1%
Log Loss	0.077
AUC (one –vs-Rest)	0.99
precision	0.99
Recall	0.99
F1-Score	0.99

These scores demonstrate that Random Forest was highly effective in correctly classifying flood risks across all categories with minimal false predictions. Particularly, the high AUC score (~1.0) confirms the model's strong capability to distinguish between the classes.

5.2.2.1 Hyperparameter Optimization for Random Forest

In machine learning, hyperparameters are external configurations that govern the training process of a model, such as how deep trees can grow or how many trees are used in an ensemble. Choosing the right set of hyperparameters is crucial for achieving optimal model performance and avoiding overfitting or underfitting.

To ensure the best-performing Random Forest model, I employed GridSearchCV, a robust hyperparameter tuning technique that performs an exhaustive search over a predefined grid of parameters. GridSearchCV trains the model for every possible combination of the specified parameters and selects the one that performs best based on cross-validation accuracy.

This approach ensures fairness and objectivity in model comparison and helps uncover the most suitable parameter set, even when parameter interactions are non-obvious.

The following hyperparameters were chosen for tuning based influence on model complexity and learning behavior:

Table 7 Random Forest Testing

Hyperparameter	**Values Tested**	**Reason for selection**
N_estimators	100,150,200	More trees usually improve performance, but increase training time
Max_depth	10,15,20	Controls how complex each tree can get deeper trees may overfit
Min_samples_split	2,5	Prevents small noisy branches from splitting too early.

Reason for using Gridsearchcv	Explanation
Exhaustive Search	It tests all combinations of hyperparameters systematically
Cross-Validation	Uses K-fold cross-validation to avoid overfitting and ensure stability
Automates Selection	Automatically selects the best model configuration based on validation performance
Reliable & Repeatable	It ensures consistency in results and reproducibility

After running GridSearchCV with the above grid, the best-performing model used the following parameters:

Table 8 Finalized the values

Hyperparameter	Best Value
Number of Trees	200
Maximum Tree Depth	20
Min Samples Per split	2

This configuration provided the best trade-off between accuracy and generalization, enabling the Random Forest model to:

• Learn complex flood-risk patterns from the dataset.

• Avoid unnecessary tree splits that might cause overfitting.

• Generalize well to unseen test data.

The deeper trees with more estimators captured important interactions among environmental features like Monsoon Intensity, River Management, and Topography, while the use of cross-validation ensured that these results were consistent and not due to random chance.

5.3 EVALUATION SUMMARY: RANDOM FOREST AND XGBOOST

Both Random Forest (RF) and XGBoost (XGB) demonstrated exceptional performance in classifying flood risk into three categories Low, Medium, and High based on structured hydrological and environmental data. The Random Forest model, trained with 200 estimators and a maximum depth of 20, achieved a strong accuracy of 99.1%, with balanced precision, recall, and F1-scores across all classes. It handled complex, nonlinear relationships well and offered high interpretability due to its ensemble of decision trees. On the other hand, XGBoost, configured with 100 estimators, a max depth of 6, and a learning rate of 0.1, slightly outperformed RF with an accuracy of 99.2%, and most notably, a much lower log loss (0.0213 vs. 0.0771). This lower log loss indicates that XGBoost's probability predictions were better calibrated, which is especially valuable in high-stakes applications such as flood early-warning systems where the confidence of predictions is as important as their correctness.

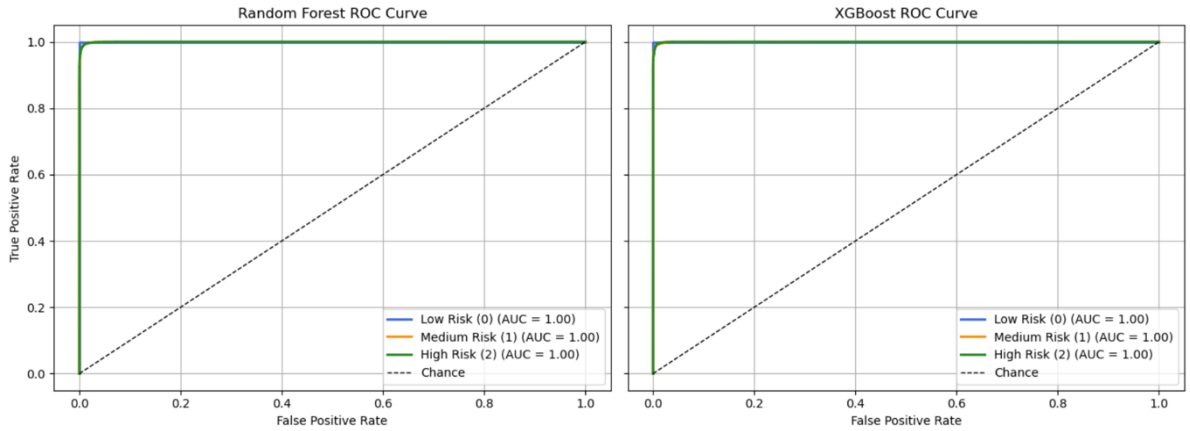


Figure 36 Class wise ROC curves for Random Forest and XGBoost

Visual evaluation further confirmed these findings. As shown in Figure 36, the ROC Curves for both models illustrated near-perfect separability between the three classes, with AUC (Area Under the Curve) values close to 1.00. This indicates a high true positive rate with minimal false alarms across all categories. The curves for XGBoost were slightly smoother, indicating more stable prediction behavior. To complement this, Figure 37 presents the Precision-Recall (PR) curves, which are particularly useful in cases of potential class imbalance. Both RF and XGB achieved average precision scores exceeding 0.98, meaning that when the model predicted a High or Medium flood risk, it was almost always correct. XGBoost showed improved recall for Medium Risk, which is typically the hardest class to predict due to its overlap with Low and High cases.

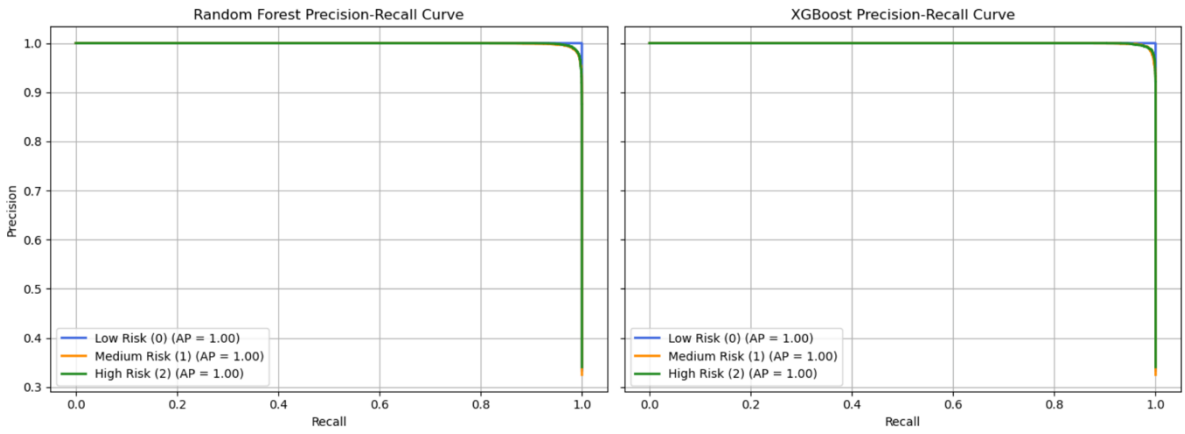


Figure 37 Precision Recall Curves for flood risk categories

5.4 CONFUSION MATRIX INTERPRETATION FOR RANDOM FOREST AND XGBOOST

The confusion matrix provides a detailed breakdown of how each model classified the flood risk levels Low, Medium, and High by comparing the predicted categories with the actual ground truth. It highlights the number of True Positives (TP), False Positives (FP), and False Negatives (FN) for each class.

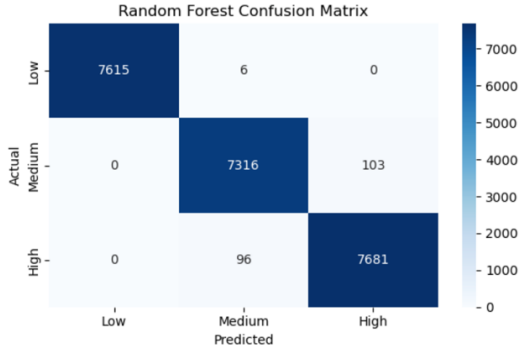


Figure 38 Confusion matrix for Random Forest

In the Random Forest confusion matrix, the model accurately predicted 7615 out of 7621 Low Risk cases, 7316 out of 7419 Medium Risk cases, and 7681 out of 7777 High Risk cases. Only a few Medium and High-risk samples were misclassified, indicating a strong performance across all categories.

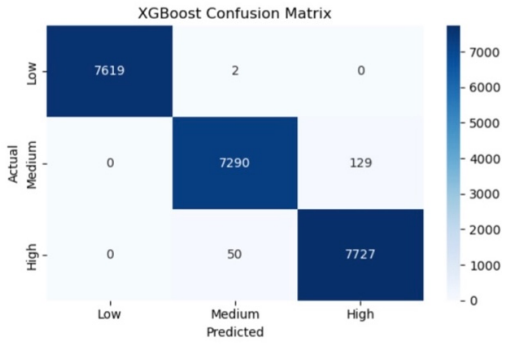


Figure 39 Confusion matrix for XGBoost

In contrast, the XGBoost model demonstrated even more refined classification, especially for the Low and High-Risk categories. It correctly classified 7619 out of 7621 Low Risk cases, 7290 Medium Risk cases, and 7727 High Risk cases. The number of false positives and false negatives was further reduced compared to Random Forest, particularly in the Medium Risk category, where fewer medium samples were wrongly classified as High.This analysis confirms that both models exhibit a high level of precision and recall, with XGBoost slightly outperforming Random Forest in terms of reducing classification errors.

Overall, the confusion matrices illustrate that both models were highly effective, with XGBoost providing slightly better calibration and fewer misclassifications, making it more reliable for critical flood risk prediction scenarios.

In conclusion, both models proved to be highly accurate and robust for flood risk prediction. Random Forest provided strong baseline performance with reliable classification, while XGBoost offered slight improvements in probability calibration and precision, making it especially suitable for real-time, data-driven flood management systems. These results validate the use of ensemble based models on well-preprocessed, balanced environmental datasets, reinforcing their potential in supporting government agencies and disaster response teams with actionable and trustworthy insights.

The evaluation results clearly demonstrate that both Random Forest and XGBoost are highly effective in handling structured environmental and hydrological data for flood risk classification. While both models delivered excellent accuracy and classification performance, XGBoost showed a slight edge over Random Forest in terms of probability calibration and precision, as evidenced by its lower log loss and smoother prediction curves. Thanks to balanced dataset preparation, systematic hyperparameter tuning, and robust validation strategies, minimal overfitting was observed in either model. Furthermore, the combined use of ROC and Precision-Recall curve analysis provided a more complete understanding of how each model

distinguished between borderline cases, enhancing interpretability. These findings confirm that ensemble-based models especially when applied to well-engineered datasets can deliver real-time, high confidence flood risk predictions. This makes them ideal candidates for integration into early warning systems, decision support tools, and flood monitoring dashboards, ultimately empowering emergency responders and policy planners with data-driven insights for proactive flood management.

5.5 DEEP LEARNING EVALUATION – U-NET FOR SATELLITE FLOOD MAPPING

To effectively detect and segment flood-affected areas from satellite imagery, a custom U-Net architecture was implemented and trained on a carefully prepared dataset of image-mask pairs. The dataset was enriched using Albumentations for data augmentation, introducing variations such as rotation, noise, contrast, and flips. These augmentations were crucial to improving the model's ability to generalize across different flood scenarios and terrains.

5.5.1 Initial Training Phase (Epochs 1–50)

The model underwent an initial training phase for 50 epochs, during which both the training and validation losses decreased consistently. By the final epoch, the model had achieved a training accuracy of 85.32%, IoU of 0.6743, and Dice Coefficient of 0.8032. The corresponding validation performance showed strong generalization, with a validation accuracy of 88.45%, IoU of 0.6979, and Dice score of 0.8182. These metrics indicate that the model successfully learned to distinguish between flooded and non-flooded areas, and its performance remained robust across unseen validation data.

5.5.2 Fine-Tuning Phase (Epochs 51–60)

Building upon the strong initial performance, the model was further fine-tuned for 10 additional epochs, this time with a reduced learning rate and early stopping enabled to prevent overfitting. This phase led to incremental improvements in all key metrics. At the conclusion of fine-tuning, the model attained a training accuracy of 86.12%, with an IoU of 0.6887 and a Dice Coefficient of 0.8137. On the validation set, accuracy slightly increased to 88.51%, while IoU rose to 0.6990, and Dice improved to 0.8192.

Table 9 Performance Summary for U-net

Training Phase	Accuracy	IoU	Dice Coefficient	Validation Loss
Initial Training	88.45%	0.69	0.81	0.52
Fine-Tuning (10 epochs)	88.51%	0.69	0.82	0.52

Table 8 shows a comparative summary of the U-Net performance across both stages. Although the improvements might appear marginal numerically, in the context of semantic segmentation, even slight enhancements in IoU and Dice values indicate sharper boundary detection, fewer misclassified pixels, and better spatial alignment, which are vital for real-world flood mapping applications.

5.5.3 Learning Curves and Visual Interpretations

The first phase of training the U-Net model was conducted over 50 epochs. Figure 40 presents both the loss and accuracy curves during this initial phase. On the top graph, the loss values are plotted across epochs, where the blue line represents training loss, and the orange line represents validation loss. The model started with a relatively high training loss of approximately 1.25, which steadily declined to 0.6032 by the 50th epoch. Meanwhile, the validation loss also decreased, reaching 0.5292, showing a healthy trend of model learning and generalization. The convergence

between both curves confirms that overfitting was minimal due to effective data augmentation and regularization strategies.

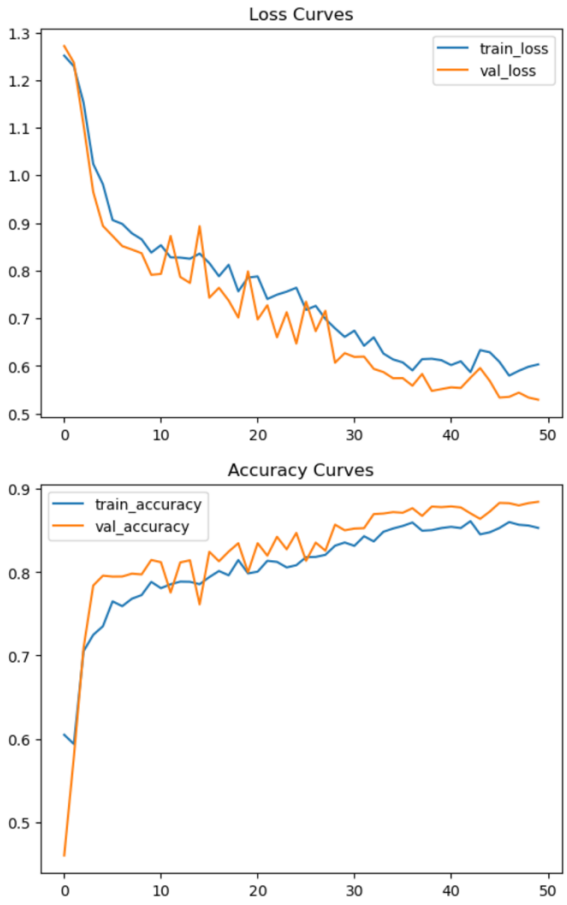


Figure 40 U-Net Training History (Epochs 1–50)

The bottom graph in Figure 41 illustrates the accuracy progression. The training accuracy began near 60% and improved to 85.32%, while the validation accuracy peaked at 88.45%. The consistent upward trend and the close alignment of both curves demonstrate strong generalization, suggesting that the model was successfully learning from spatial patterns in the satellite images and applying them well to unseen data.

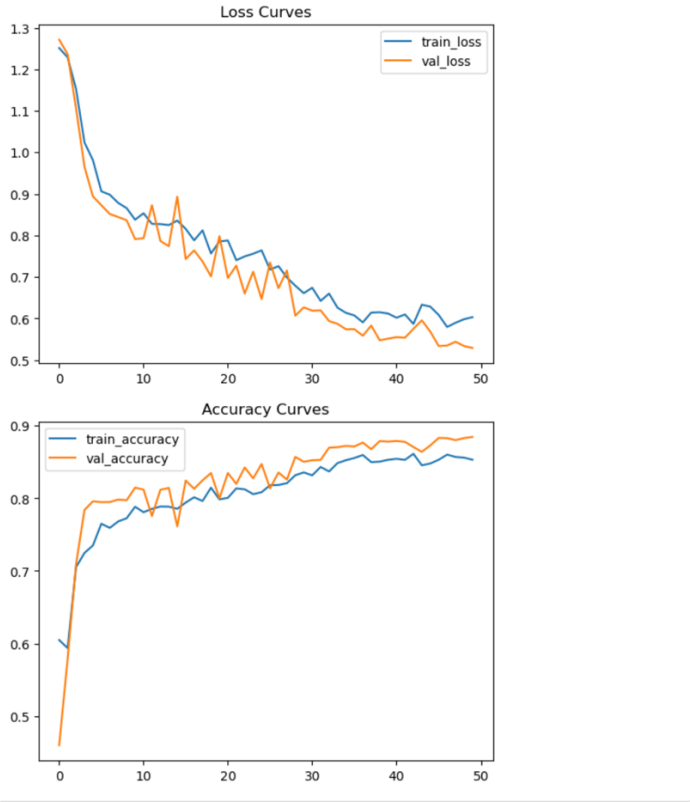


Figure 41 U-Net Fine-Tuning Training History (Epochs 51–60)

After evaluating the model's performance, a fine-tuning phase was introduced for an additional 10 epochs using a lower learning rate to refine the weights. As seen in Figure 37, further improvements were observed. The training loss continued to decrease from 0.6032 to 0.5822, while validation loss slightly improved from 0.5292 to 0.5241, reflecting incremental learning gains and stabilization.

Accuracy metrics also experienced a modest boost. The training accuracy increased from 85.32% to 86.12%, and the validation accuracy improved from 88.45% to 88.51%. These marginals yet important improvements in the fine-tuning stage highlight the advantage of staged training and underscore the effectiveness of early stopping and learning rate scheduling.

Table 10 Summary Table: U-Net Training Metrics

Training Phase	Train Accuracy	Val Accuracy	Train Loss	Val Loss	IoU	Dice
Epochs 1-50	85.32%	88.45%	0.60	0.52	0.69	0.81
Epochs 51-60(FT)	86.12%	88.51%	0.58	0.52	0.70	0.82

These learning curves and metrics collectively demonstrate that the U-Net model was able to robustly learn flood-affected area segmentation. The clear reduction in loss, consistent increase in accuracy, and well-aligned validation performance support the conclusion that the model is not just memorizing but generalizing effectively. The staged training approach enabled better fine-tuning of spatial features, which is particularly crucial for satellite imagery where visual patterns are highly complex.

5.5.4 Visual Predictions and Segmentation Quality

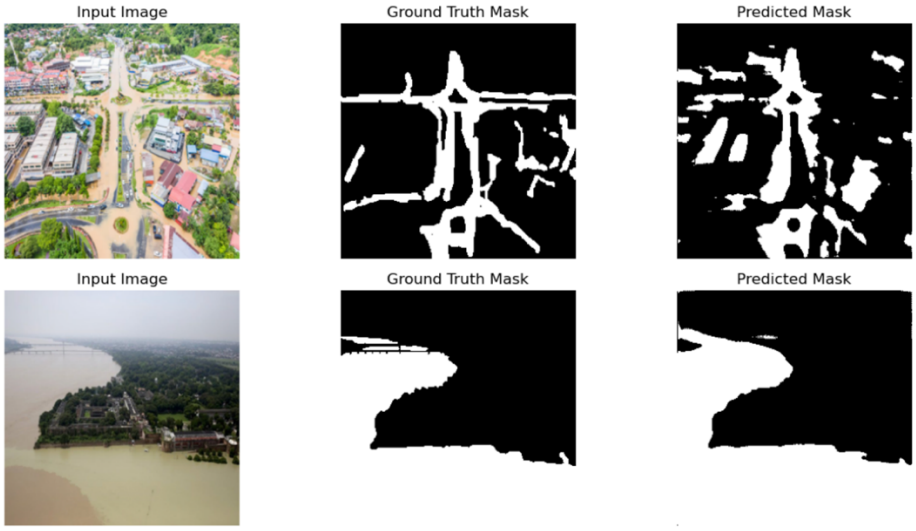


Figure 42 Predicted vs Ground Truth Masks

To visually assess segmentation quality, Figure 42 displays selected results from the validation set. Each row shows an input satellite image, its corresponding ground truth mask, and the predicted mask generated by the U-Net model. In each case, the predicted masks accurately match the labeled flood zones, with clear identification of flooded water bodies, structural outlines, and even partially submerged areas. The segmented regions are not only spatially aligned but also consistent in shape and density, reflecting the model's robustness in real-world geospatial conditions.

5.5.5 Conclusion and Insights

From this two-stage training strategy, several key insights were gathered:

• Data augmentation significantly contributed to generalization, especially in varied terrains.

• IoU and Dice improvements after fine-tuning demonstrate that the model became more precise in edge detection.

• The learning curves reflected stability and consistency, minimizing the risk of overfitting.

• Visual outputs validated the model's ability to replicate human-labeled segmentation with high fidelity.

This deep learning pipeline, combining structured training and visual validation, established U-Net as a reliable tool for satellite-based flood risk analysis, especially when integrated into AI-powered flood monitoring systems

5.5.6 Model generalization evaluation

A machine learning model's true effectiveness isn't only measured by how well it performs on training data, but more importantly, how it behaves on data it has never seen before. This capability is known as generalization. In this section we evaluate how well our trained U-Net model can generalize to a new set of flood images, which were completely unseen during training. This helps us understand how reliably the model would perform in real-world scenarios

The new dataset comprises plain screenshots from Google Satellite View, capturing regions potentially affected by flooding. Unlike the original training dataset, this set lacks manually labeled ground truth flood masks. Quantitative metrics such as accuracy, Intersection over Union (IoU), and Dice coefficient require a comparison to ground truth masks, which act as the definitive reference for what areas are truly flooded. Since the new dataset lacks these reference labels, it is not possible to compute these metrics in a meaningful way. Any such calculation without ground truth would be purely speculative and not grounded in verifiable correctness. Therefore, our evaluation is limited to visual inspection, flood coverage estimation, which still provides meaningful insight into how the model behaves when applied to unfamiliar data.

All these metrics Accuracy, **IoU (**Intersection over Union**)**, and Dice Coefficient require a ground truth reference to calculate how "correct" the model's predictions are.

They depend on a pixel-by-pixel comparison between:

- What the model predicted
- What is actually true (i.e., labeled by a human expert)

Metric	**Formula**	**Needs Ground Truth?**	**Why?**
Accuracy	(TP + TN) / (TP + FP + TN + FN)	Yes	You need to know which pixels are TP, TN, etc.
IoU	TP / (TP + FP + FN)	Yes	You must count overlapping pixels with the ground truth
Dice	2 * TP / (2 * TP + FP + FN)	Yes	You need to compare sets of predicted vs actual masks

which **only make sense** if you have a "truth" to compare with.

Term	**Full Form**	**Meaning**
TP	True Positives	Pixels correctly predicted as flooded
TN	True Negatives	Pixels correctly predicted as not flooded
FP	False Positives	Pixels incorrectly predicted as flooded (but were not flooded)
FN	False Negatives	Pixels incorrectly predicted as not flooded (but were actually flooded)

- There's no way to know if a pixel marked as "flooded" is actually flooded or not.
- You could predict everything as flooded or nothing at all and the metric would have no way to penalize or reward that.
- The model might appear visually reasonable, but that is qualitative, not measurable.

Evaluation Approach (Without Ground Truth)

Here is the step-by-step process followed to evaluate the model on this new dataset:

- Preprocessing: Each image was resized to 256x256 pixels and normalized to match the model's expected input.
- Prediction: The trained U-Net model (`unet_finetuned_best.h5`) was used to predict flood masks for each image.
- Flood Coverage Calculation: We calculated what percentage of each image the model predicted as flooded.
- Visual Inspection: Overlay images were created by blending predicted masks with original inputs to highlight flood zones.

This process enables an assessment of prediction patterns and consistency, despite the absence of annotated reference data.

While it is technically possible to extract model-derived masks from overlay images for internal consistency checks, this approach was intentionally excluded in this study, as no human-annotated ground truth masks were available for the new dataset. Instead, we rely on flood coverage estimation and visual inspection to assess the model's generalization behavior. This ensures that the evaluation remains grounded in realistic interpretation without introducing unverified approximations.

Quantitative Summary

The model's flood predictions across the dataset were summarized as follows:

```
GENERALIZATION SUMMARY (No Ground Truth):
Average Predicted Flood Area: 7.53%
Minimum Predicted: 3.85%
Maximum Predicted: 9.61%
Overlay images saved in: ./overlay_images
Binary masks saved in: ./predicted_masks
```

Figure 43 summary

Since no ground truth masks were available, estimated model performance based on how much of each image was predicted as flooded. This percentage helps us understand whether the model is making reasonable predictions (e.g., not flooding the entire image or detecting nothing at all). These values indicate that the model is consistently identifying likely flood regions, and it does not over-predict or miss predictions drastically. Most images had a flood prediction in the 5-10% range, which is realistic for satellite flood imagery.

To complement the numeric summary, visually examined the model's predictions. Below are examples that compare the original flood image with the predicted flood overlay: The model correctly highlights areas near the river and low-lying zones as flooded.

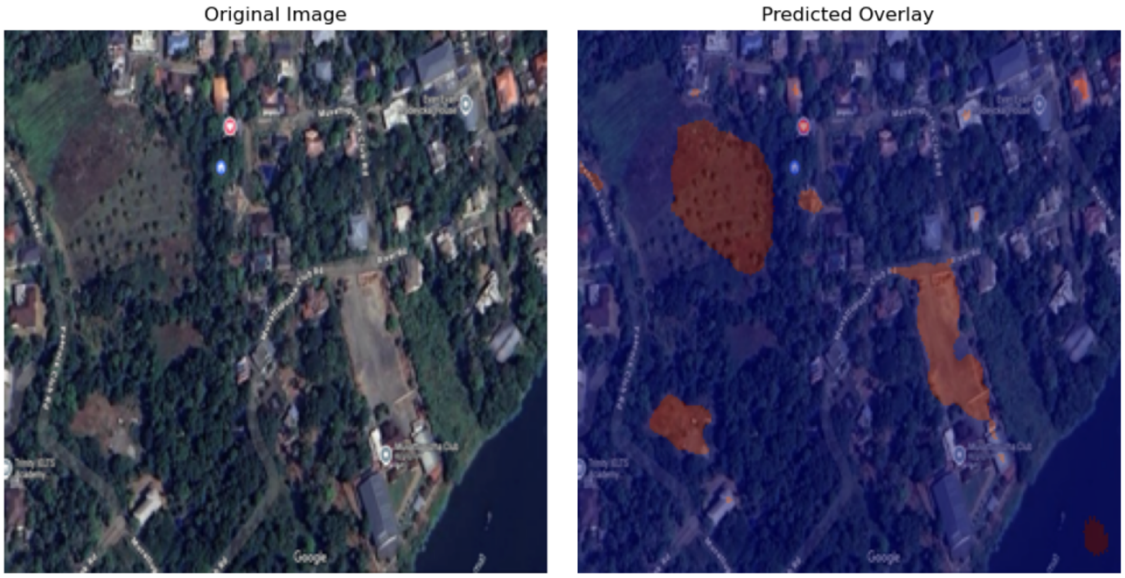


Figure 44 visual evidence

Predictions are well-aligned with real-world terrain features such as water bodies and urban edges. These overlays help validate that the model is not just making random guesses, but instead learning patterns associated with flooding.

From the results above, we can say with confidence that the U-Net model has retained its ability to detect flooded areas, even when applied to new, previously unseen images. The fact that it consistently predicts flood coverage in a realistic range (around 7.5%) and highlights meaningful areas visually supports its generalization capability. However, because we don't have ground truth masks for these images, we can't quantitatively confirm whether these predictions are 100% accurate. This highlights an important point: Without ground truth masks, we can evaluate visual correctness but not calculate true accuracy. Therefore, while the model appears to generalize well visually, a future step could include manually labeling a small sample of the new dataset to measure actual IoU/Dice metrics.

Performance on Training and Validation Sets

As a reference point, the model achieved the following results during training on a labeled dataset:

Training Phase	Train Accuracy	Val Accuracy	Train Loss	Val Loss	IoU	Dice
Epochs 1-50	85.32%	88.45%	0.60	0.52	0.69	0.81
Epochs 51-60(FT)	86.12%	88.51%	0.58	0.52	0.70	0.82

These values demonstrate strong performance on known data but cannot be generalized to the new dataset without reference labels.

Conclusion

This generalization test on an unlabeled, real-world flood dataset shows that the trained U-Net model maintains consistent, logical performance outside its training domain. Despite the absence of ground truth, predicted overlays and coverage estimates demonstrate the model's reliability in identifying flood-prone zones. This builds trust in its real-world applicability and sets the stage for more robust testing using partially labeled or semi-supervised data in the future. Nevertheless, without labeled ground truth masks, definitive statements about the model's accuracy or reliability cannot be made. Future work should focus on the development of a manually annotated subset of the new dataset to facilitate quantitative evaluation. Additionally, the integration of semi-supervised learning or active labeling strategies may further support real-world deployment.

5.6 PROJECT IMPLEMENTATION PLAN

Fig. 45 shows the implementation plan for the dissertation project.

Figure 45: Project implementation plan

Chapter 6: Conclusion

6.1 Concluding remarks

This study set out to develop an intelligent, AI-powered flood prediction and segmentation system that addresses the pressing limitations of traditional hydrological models. By combining structured environmental data with unstructured satellite imagery, the research aimed to enhance flood forecasting accuracy, spatial coverage, and operational readiness. Grounded in the problem context of increasing flood frequency, climate variability, and inadequate real-time prediction tools, the project leveraged machine learning models XGBoost and Random Forest for classification, and U-Net for pixel-level flood segmentation.

The findings strongly support the hypothesis that hybrid AI models can significantly improve both the precision and timeliness of flood risk assessments. The Random Forest and XGBoost classifiers achieved exceptionally high accuracy levels (99.1% and 99.2% respectively), with XGBoost displaying superior probability calibration an essential feature for decision-support systems. Meanwhile, the U-Net segmentation model achieved a validation accuracy of 88.51%, with an IoU **of** 0.6990 and Dice coefficient **of** 0.8192, demonstrating its ability to generalize flood patterns across varied spatial contexts.

Taken together, the integration of ensemble learning for structured prediction and deep learning for image segmentation has shown practical feasibility for deployment in real-world flood early warning systems. The Streamlit based dashboard developed for this study further emphasizes the project's commitment to actionable, accessible, and real-time flood risk intelligence.

These results offer clear answers to the research questions posed:

- AI models effectively predict flood risk and map vulnerable regions by leveraging both numerical and image-based datasets.
- The accuracy and reliability of the ML/DL-based system surpass traditional hydrological models, particularly in dynamically changing or data-scarce scenarios.
- The U-Net architecture, trained on satellite imagery, provides high spatial resolution mapping, enabling timely identification of flood-prone areas for targeted intervention.

6.2 Limitations of the study

Despite the promising outcomes, several limitations warrant consideration:

- Data Diversity: Satellite imagery used in training was limited to specific regions and may not fully represent global variations in terrain, vegetation, or infrastructure.
- Temporal Dynamics: The dataset lacked fine-grained temporal features such as hourly rainfall intensity or river discharge sequences, which limits the model's responsiveness to sudden climatic changes.
- Labeling Assumptions: Segmentation masks were assumed to be ground truth but may contain inaccuracies that impact model training and evaluation.
- Computational Overhead: Training and fine-tuning deep learning models such as U-Net required high-performance hardware, which could hinder scalability for under-resourced regions.

6.3 Recommendations for practical implementation

This research lays a strong foundation for AI-driven flood resilience systems and opens practical avenues for deployment:

- Disaster Management Integration: The system can be integrated into government dashboards or NGO platforms for real-time alerts, visualizations, and risk communication.
- Risk-Based Prioritization: Confidence scores from XGBoost classifiers can be embedded into automated early warning systems, enabling tiered alert mechanisms for resource allocation.
- Community Empowerment: A simplified version of the dashboard could be customized for local communities to promote public awareness and participatory preparedness.

6.4 Recommendations for future research

The study presents several directions for future development:

- Temporal Modeling: Incorporating LSTM or transformer-based models could allow the system to handle sequential data, improving short-term flood forecasting.
- Enhanced Imagery Inputs: Multi-spectral, infrared, or SAR radar imagery can be used to enhance segmentation performance in cloud-covered or night-time conditions.
- Attention-Based U-Nets: Integrating attention mechanisms (e.g., Attention U-Net or SE-UNet) may help the model better focus on flood boundaries, improving segmentation granularity.
- Transfer Learning & Adaptability: Explore domain adaptation techniques to enable model deployment in new regions with minimal retraining or labeled data.
- Real-Time Integration: Future iterations should integrate live weather feeds and hydrological APIs to support dynamic, up-to-date flood monitoring systems.

6.5 Concluding insights: advancing flood prediction through practical and interpretable AI

This research takes a meaningful step forward in bridging the critical gaps identified in previous flood prediction studies. Unlike traditional models that focus on a single data type or operate in isolation, this project brings together both structured hydrological data and unstructured satellite imagery, forming a unified system for predicting and visualizing flood risk. Rather than treating the models as black boxes, the study uses intuitive visual tools like correlation heatmaps and feature importance rankings from Random Forest and XGBoost to clearly show which factors truly drive flood risk. From monsoon intensity to infrastructure quality, the model doesn't just make predictions it explains them. To ensure fairness and accuracy, the dataset was enhanced with synthetic examples and balancing techniques, helping the model learn from both frequent and rare flood scenarios. What sets this work apart is not just the technical innovation, but the practical execution: an interactive, user-friendly Streamlit dashboard that allows decision-makers, planners, or even everyday users to engage with real-time flood forecasts and satellite-based flood detection. By focusing on locally relevant features and building a system that is both powerful and accessible, this study delivers not only a technical solution but a meaningful tool for resilience and preparedness in the face of climate-driven disasters.

6.6 Final reflection

This Project confirms the transformative potential of artificial intelligence in addressing one of the world's most pressing climate-related challenges. The models developed in this study not only achieve high accuracy but also provide interpretability, usability, and operational insight. By integrating structured hydrological data with satellite image segmentation, this system demonstrates how interdisciplinary approaches can produce actionable outcomes for real-world problems. With continued refinement, such AI-powered systems can serve as critical tools in climate resilience planning, emergency response, and sustainable urban development.

Ultimately, the study contributes to the evolving field of disaster analytics, marking a step forward in the quest for smarter, faster, and fairer responses to environmental threats.

Reference

1. **Kundzewicz, Z.W., Su, B., Wang, Y., Wang, G., Wang, G., Huang, J., & Jiang, T. (2019)** 'Flood risk in a range of spatial perspectives – from global to local scales', *Natural Hazards and Earth System Sciences*. doi:10.5194/nhess-19-1319-2019.

2. **Young, J.C., Arthur, R., Spruce, M., & Williams, H.T.P. (2022)** 'Social sensing of flood impacts in India: A case study of Kerala 2018', *International Journal of Disaster Risk Reduction*. doi: 10.1016/j.ijdrr.2022.102908.

3. **Le, X.-H., Ho, H.V., Lee, G., & Jung, S. (2019)** 'Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting', *Water*, **11**(7), 1387. doi:10.3390/w11071387.

4. **Andrew, O., Apan, A., Paudyal, D.R., & Perera, K. (2023)** 'Convolutional Neural Network-Based Deep Learning Approach for Automatic Flood Mapping Using NovaSAR-1 and Sentinel-1 Data', *ISPRS International Journal of Geo-Information*, 12(5). doi:10.3390/ijgi12050194.

5. **Nearing, G., Cohen, D., Dube, V., et al. (2024)** 'Global prediction of extreme floods in ungauged watersheds', *Nature*, 627, pp. 559–563. doi:10.1038/s41586-024-07145-1

6. **Mosavi, A., Ozturk, P., & Chau, K.-W. (2018)** 'Flood Prediction Using Machine Learning Models: Literature Review', *Water*, 10(11). doi:10.3390/w10111536.

7. **Kabir, S., Patidar, S., Xia, X., Liang, Q., Neal, J., & Pender, G. (2020)** 'A deep convolutional neural network model for rapid prediction of fluvial flood inundation', *Journal of Hydrology*. doi: 10.1016/j.jhydrol.2020.125481.

8. **Zeng, C. & Bertsimas, D. (2023)** 'Global Flood Prediction: A Multimodal Machine Learning Approach', *arXiv preprint*, arXiv:2301.12548. doi:10.48550/arXiv.2301.12548

9. **Naik, S., Patil, S.A., Verma, A., & Hingmire, A. (2021)** 'Flood Prediction using Logistic Regression for Kerala State', *International Journal of Engineering Research and Technology (IJERT)*, 9(3). doi:10.17577/IJERTCONV9IS03010.

10. **Li, W., Liu, C., Xu, Y., Niu, C., Li, R., Li, M., Hu, C., & Tian, L. (2024)** 'An interpretable hybrid deep learning model for flood forecasting based on Transformer and LSTM', *Journal of Hydrology: Regional Studies*, 101873. doi:10.1016/j.ejrh.2024.101873

11. **Islam, A.R.M.T., Talukdar, S., Mahato, S., Kundu, S., Eibek, K.U., Pham, Q.B., Kuriqi, A., & Linh, N.T.T. (2021)** 'Flood susceptibility modelling using advanced ensemble machine learning models', *Geoscience Frontiers*, 12(3), 101075. doi:10.1016/j.gsf.2020.09.006

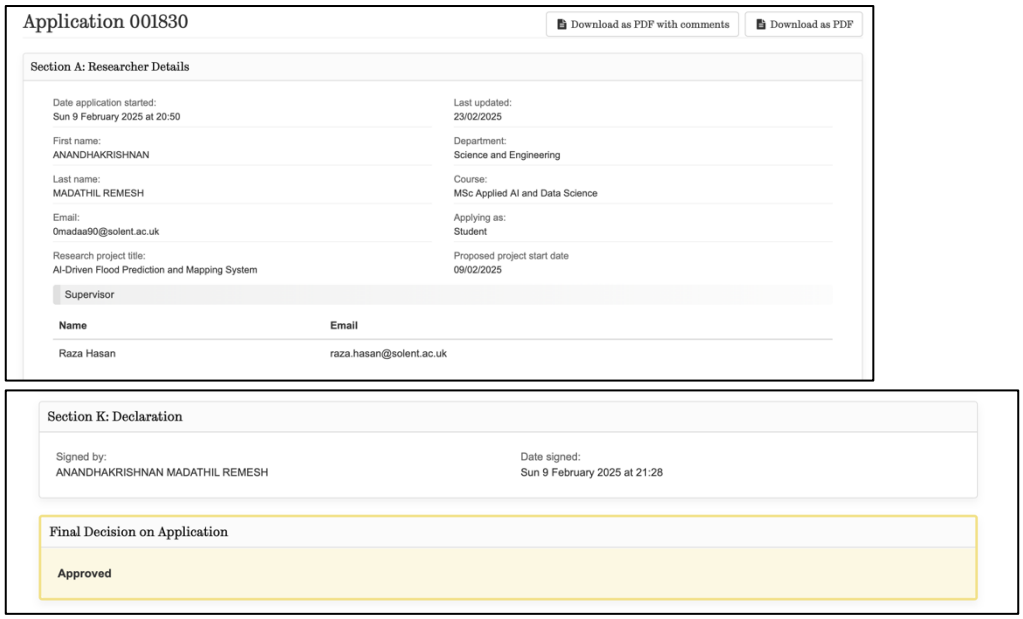
12. **Situ, Z., Wang, Q., Teng, S., Feng, W., Chen, G., Zhou, Q., & Fu, G. (2024)** 'Improving Urban Flood Prediction using LSTM-DeepLabv3+ and Bayesian Optimization with Spatiotemporal Feature Fusion', *Journal of Hydrology*, doi: 10.1016/j.jhydrol.2024.130743.

13. Abedi, R., Costache, R.-D., Shafizadeh-Moghadam, H., and Pham, Q.B. (2021) 'Flash-flood susceptibility mapping based on XGBoost, Random Forest and Boosted Regression Trees', *Geocarto International*,. Available at: https://www.researchgate.net/publication/351080940_Flash-flood_susceptibility_mapping_based_on_XGBoost_Random_Forest_and_Boosted_Regression_Trees

14. Wu, Y., Zhang, Z., Qi, X., Hu, W., & Si, S. (2024). Prediction of flood sensitivity based on Logistic Regression, eXtreme Gradient Boosting, and Random Forest modeling methods. *Water Science & Technology*, 89(10), 2605–2624. https://doi.org/10.2166/wst.2024.146

Appendices

Appendix A: Academic ethics approval



Application 001830

Download as PDF with comments | Download as PDF

Section A: Researcher Details

Date application started: Sun 9 February 2025 at 20:50	Last updated: 23/02/2025
First name: ANANDHAKRISHNAN	Department: Science and Engineering
Last name: MADATHIL REMESH	Course: MSc Applied AI and Data Science
Email: 0madaa90@solent.ac.uk	Applying as: Student
Research project title: AI-Driven Flood Prediction and Mapping System	Proposed project start date 09/02/2025

Supervisor

Name	Email
Raza Hasan	raza.hasan@solent.ac.uk

Section K: Declaration

Signed by: ANANDHAKRISHNAN MADATHIL REMESH	Date signed: Sun 9 February 2025 at 21:28

Final Decision on Application

Approved

Appendix B: Python libraries used in data analysis, model training, and evaluation

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report
import joblib
```

```
import os
import numpy as np
import albumentations as A
from albumentations.core.composition import OneOf
from tensorflow.keras import Input, Model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Conv2DTranspose, concatenate, Dropout
from tensorflow.keras.optimizers.legacy import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint
import tensorflow.keras.backend as K

os.environ["SM_FRAMEWORK"] = "tf.keras"
import segmentation_models as sm
```

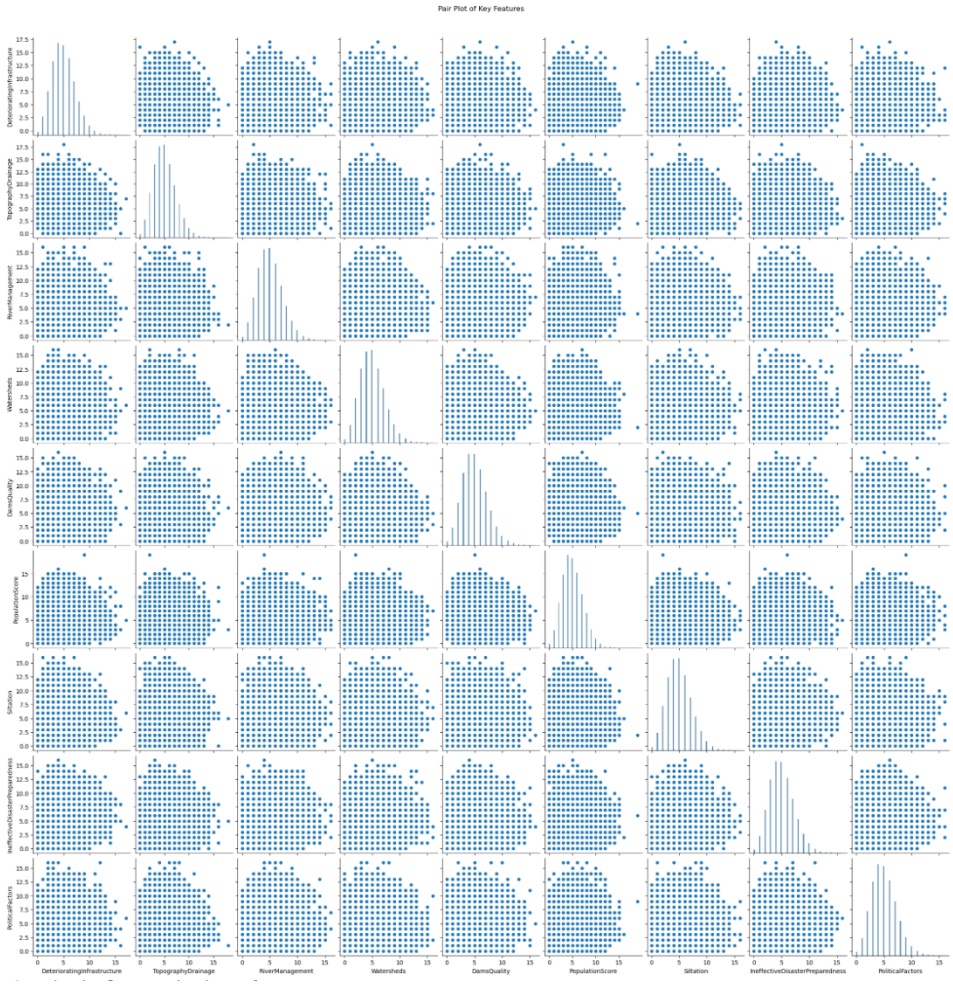
Appendix C: Python Libraries used in Streamlit development.

```
import streamlit as st
import requests
import pandas as pd
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import tensorflow.keras.backend as K
from tensorflow.keras.models import load_model
import joblib
from io import BytesIO
import base64
```

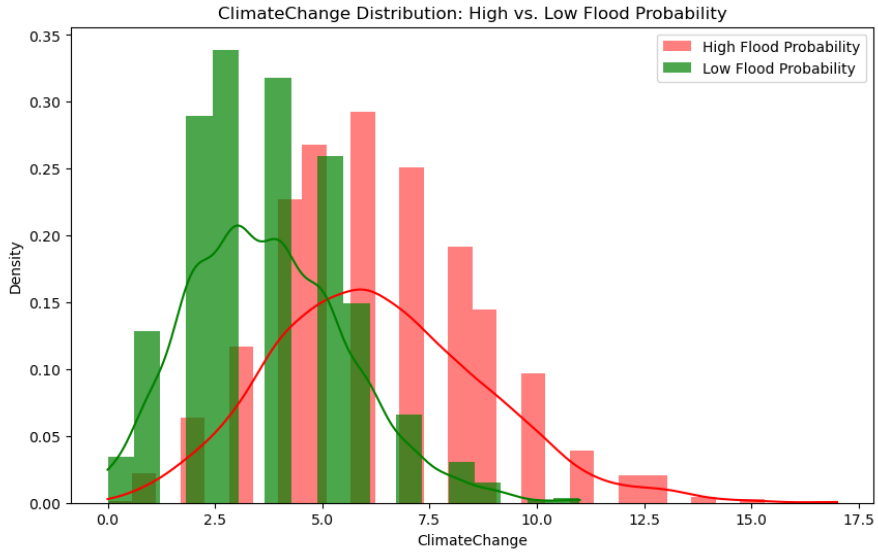
Appendix D: Code snippet for boxplot, pair plot to visualize Key Features and other analyses.

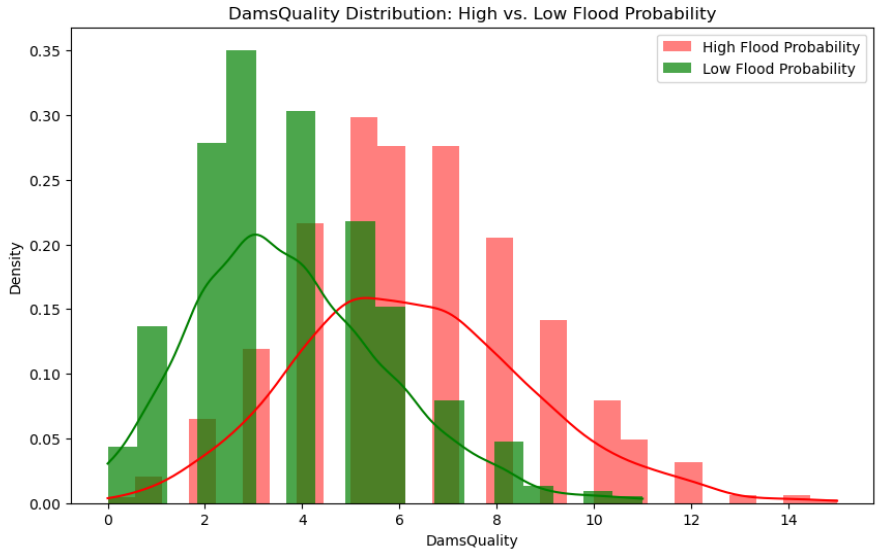
```
#  Boxplots of Key Features
plt.figure(figsize=(14, 8))
sns.boxplot(data=df[features], orient='h', palette="coolwarm")
plt.title('Boxplots of Flood-Related Features')
plt.xlabel('Value')
plt.ylabel('Features')
plt.show()

#  Pair Plot of Selected Key Features
key_features = top_corr_features.drop('FloodProbability')
sns.pairplot(df[key_features])
plt.suptitle('Pair Plot of Key Features', y=1.02)
plt.show()

# Statistical Descriptions
stats_description = df.describe()
print("Statistical Description of Dataset:\n", stats_description)


# Statistical Counts of High and Low Flood Probability Cases
num_high_risk_cases = high_risk.shape[0]
num_low_risk_cases = low_risk.shape[0]
total_cases = df.shape[0]
num_features = len(features)
```

```
# Distribution of Flood Probability
plt.figure(figsize=(8, 5))
sns.histplot(df['FloodProbability'], bins=30, kde=True, color='skyblue')
plt.title('Distribution of Flood Probability')
plt.xlabel('Flood Probability')
plt.ylabel('Frequency')
plt.show()

# Boxplot for key features
key_features = ["MonsoonIntensity", "ClimateChange", "DamsQuality", "Urbanization"]
plt.figure(figsize=(12, 8))
sns.boxplot(data=df[key_features], orient='h')
plt.title('Boxplot of Key Flood-related Features')
plt.xlabel('Feature Value')
plt.ylabel('Features')
plt.show()


#  Feature distributions comparing low and high flood probabilities
high_risk = df[df['FloodProbability'] > 0.6]
low_risk = df[df['FloodProbability'] < 0.4]

# All numeric features excluding FloodProbability
features = df.select_dtypes(include=['number']).columns.drop('FloodProbability')

# Generate feature distribution plots
for feature in features:
    plt.figure(figsize=(10, 6))
    sns.histplot(high_risk[feature], color='red', label='High Flood Probability', kde=True, stat="density", linewidth=0)
    sns.histplot(low_risk[feature], color='green', label='Low Flood Probability', kde=True, stat="density", linewidth=0, alpha=0.7)
    plt.title(f'{feature} Distribution: High vs. Low Flood Probability')
    plt.xlabel(feature)
    plt.ylabel('Density')
    plt.legend()
    plt.show()

# Top correlated features heatmap
plt.figure(figsize=(12, 10))
top_corr_features = df.corr()['FloodProbability'].abs().sort_values(ascending=False).head(10).index
sns.heatmap(df[top_corr_features].corr(), annot=True, cmap="coolwarm")
plt.title('Top Feature Correlation Heatmap')
plt.show()
```



Appendix E: Pair Plot of Key Features

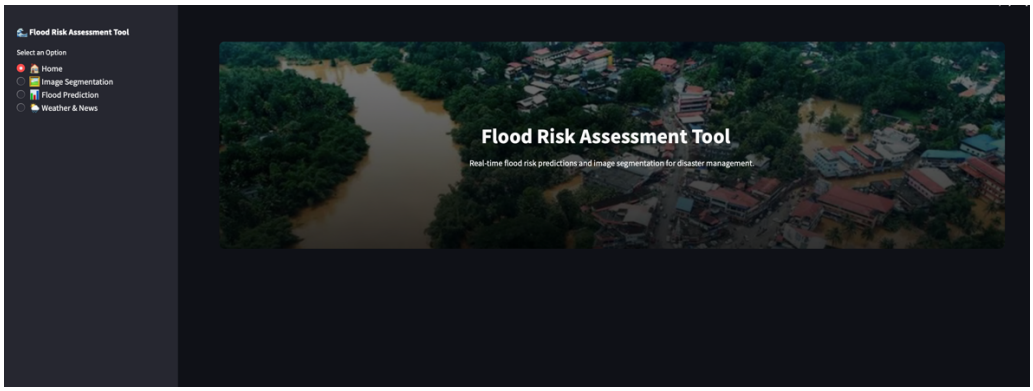


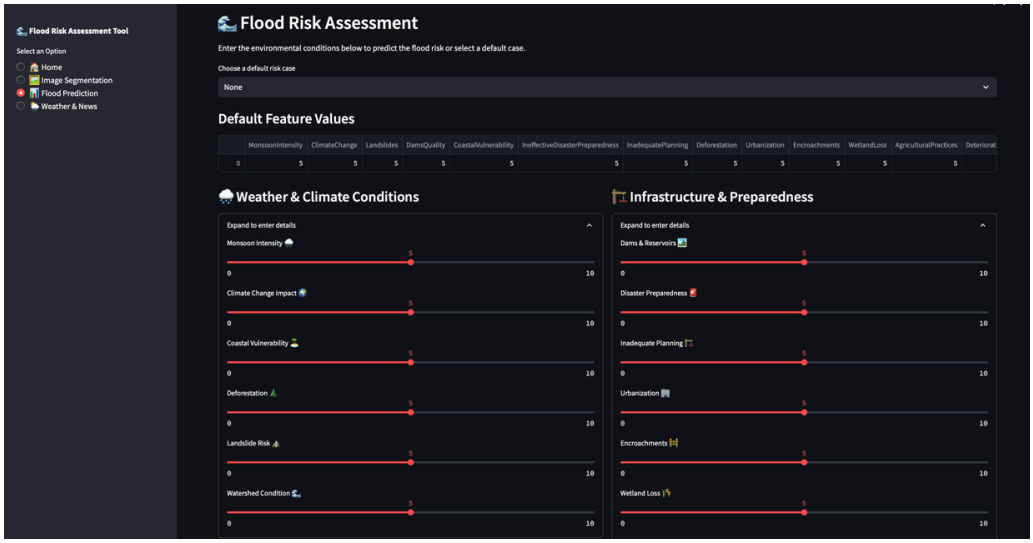


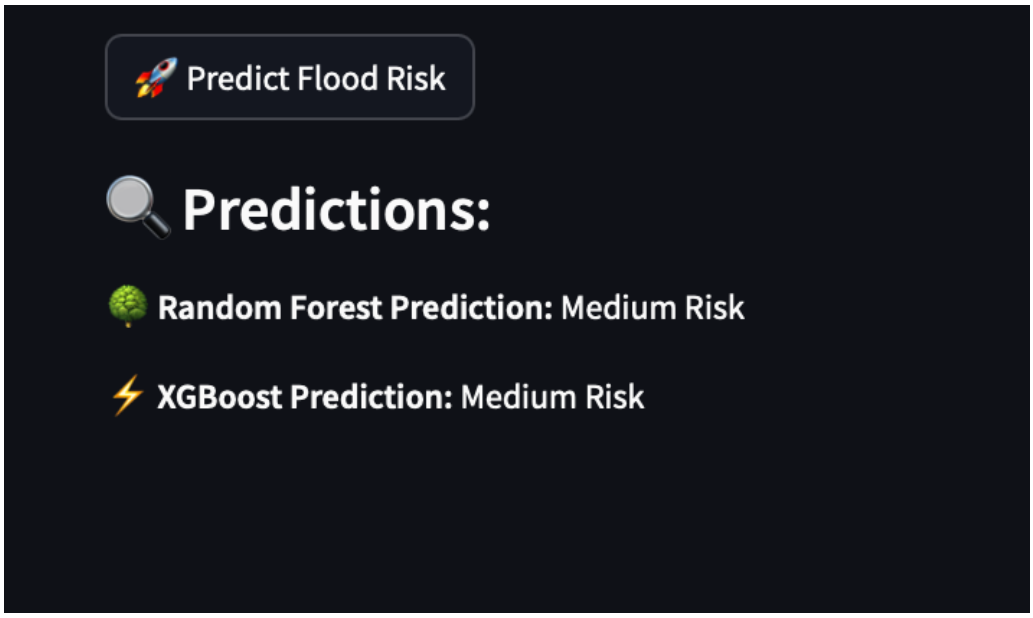
Appendix F: Feature distribution plots

Appendix G: Streamlit Interface

```
# ============================================================
#  Page Configuration & Custom Styling
# ============================================================
st.set_page_config(page_title="Flood Risk Assessment Tool", layout="wide")

st.markdown(f"""
    <style>
    .big-title {{ font-size:40px !important; text-align: center; font-weight: bold; color: #FFFFFF; }}
    .desc {{ text-align: center; font-size: 18px; color: #F8F9FA; }}
    .home-bg {{
        background: linear-gradient(to bottom, rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.8)),
        url("{background_base64}");
        background-size: cover;
        height: 400px;
        width: 100%;
        display: flex;
        align-items: center;
        justify-content: center;
        border-radius: 10px;
    }}
    .sidebar-title {{ font-size: 22px; font-weight: bold; color: #FFFFFF; }}
    </style>
    """, unsafe_allow_html=True)

# ============================================================
# Load Models
# ============================================================

# Flood risk prediction models
rf_model = joblib.load("/Users/anandhu/Downloads/Final Project/combined_dataset/structured/random_forest_model.pkl")
xgb_model = joblib.load("/Users/anandhu/Downloads/Final Project/combined_dataset/structured/xgboost_model.pkl")

# UNet model for image segmentation
@st.cache_resource
def load_unet_model():
    return load_model("unet_finetuned_best.h5", compile=False)

unet_model = load_unet_model()
```






🚀 Predict Flood Risk
🔍 Predictions:
🌳 Random Forest Prediction: Medium Risk
⚡ XGBoost Prediction: Medium Risk