### **Rising Waters: Predicting Floods Using Machine Learning**

**1. Abstract :** Floods cause devastation to ecosystems, urban infrastructure, and human life. In this project, we leverage meteorological data and apply machine learning algorithms to predict the possibility of severe flooding. A web-based interface (Flask) allows users to input values and get real-time predictions. The system is designed for quick access and efficient decision-making.

**2. Dataset Description :**

* **File**: flood dataset.xlsx
* **Source**: Provided by SmartBridge
* **Total Columns**: 11
* **Target Column**: flood
* **Features Considered**:
  + Cloud Cover
  + ANNUAL Rainfall
  + Jan-Feb Rainfall
  + Mar-May Rainfall
  + Jun-Sep Rainfall

### 3. Tools & Libraries Used :

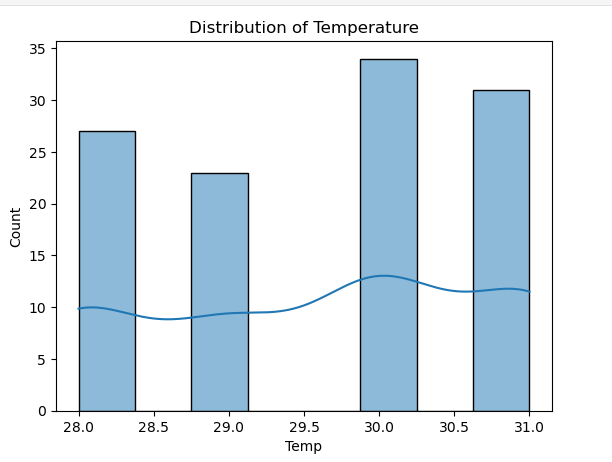
* Python 3.11
* pandas, numpy
* seaborn, matplotlib
* scikit-learn
* XGBoost
* Flask
* joblib

### 4. Data Preprocessing

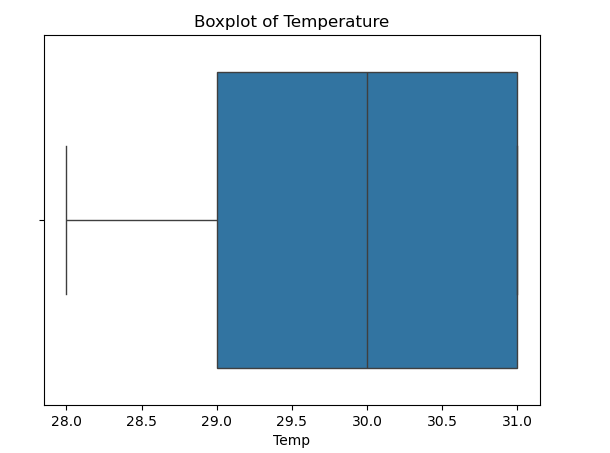
* Checked and confirmed no missing values
* Selected top 5 features based on correlation
* Split data into train and test (80-20)
* Standardized features using StandardScaler

### 5. Exploratory Data Analysis (EDA)

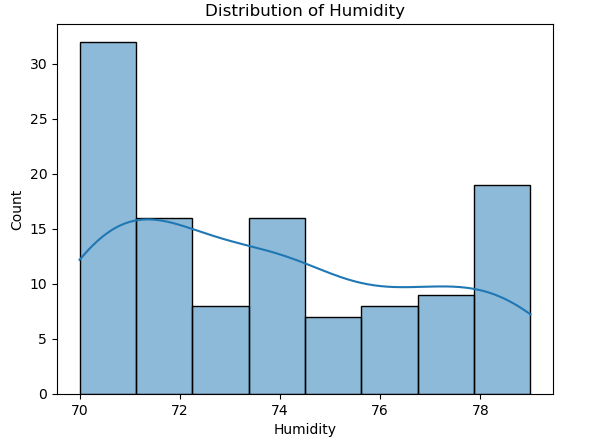
#### 🔹 Temperature Distribution:



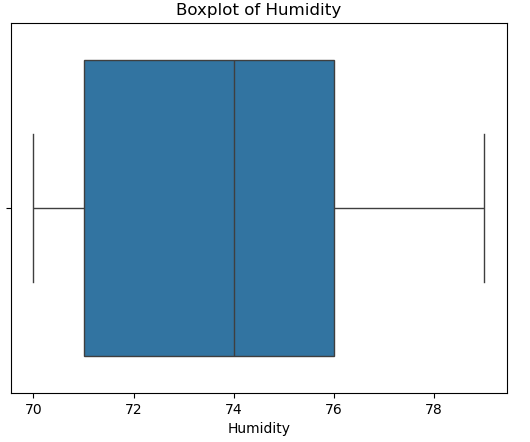
🔹 **Temperature Boxplot:**



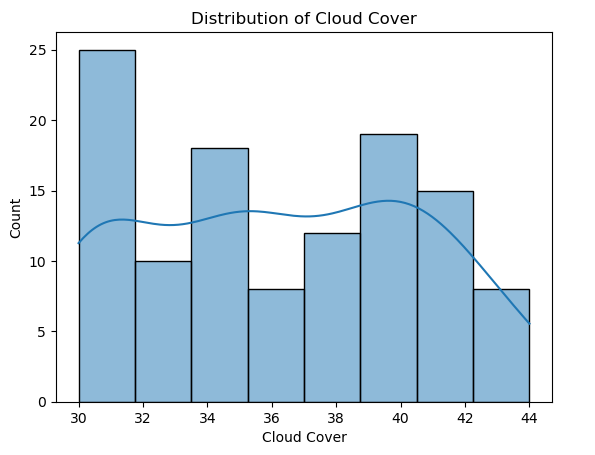
🔹 **Humidity Distribution:**



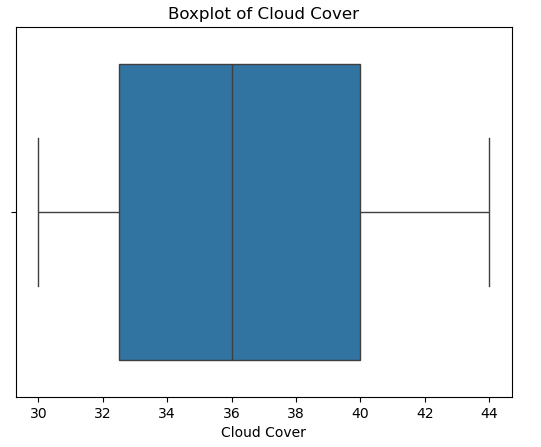
🔹 **Humidity Boxplot:**



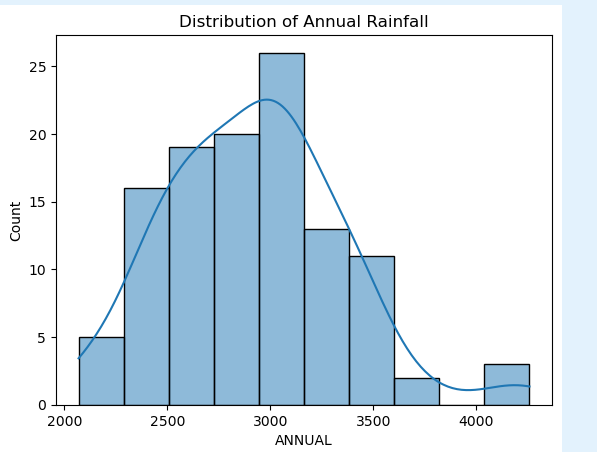
🔹 **Cloud Cover Distribution:**



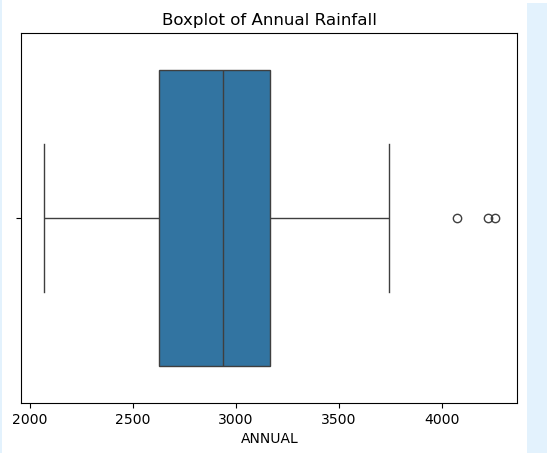
🔹 **Cloud Cover Boxplot:**



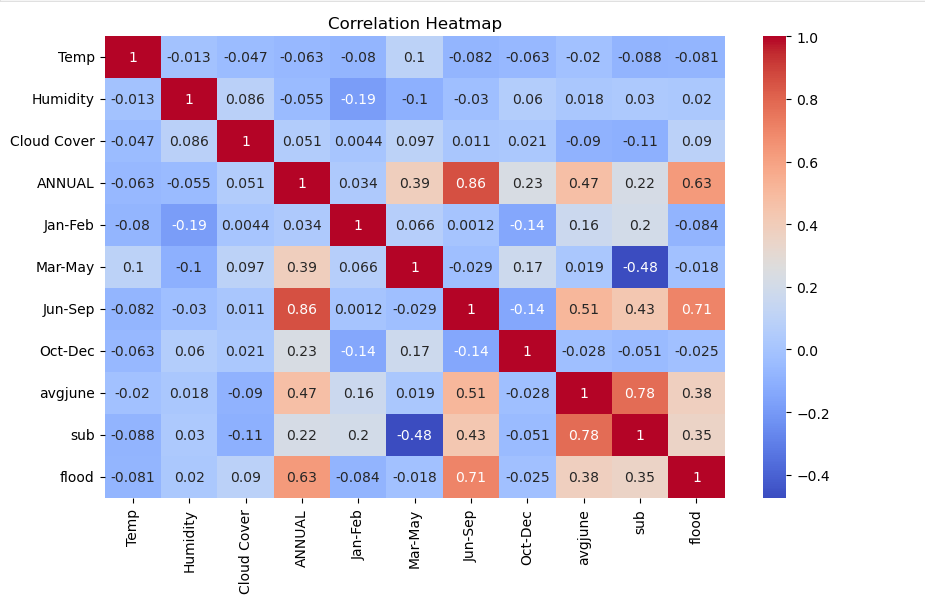
🔹 **Annual Rainfall Distribution:**



🔹 **Annual Rainfall Boxplot:**



#### 🔹 Correlation Heatmap:



### 6. Model Training & Evaluation

#### Models Trained:

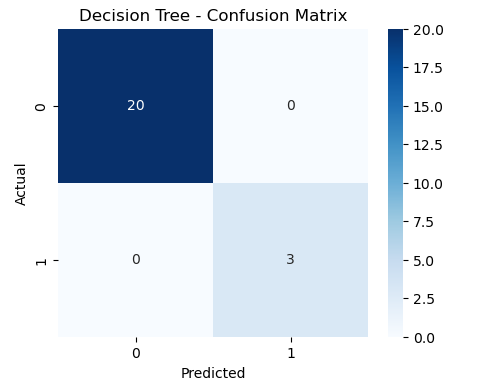
* Decision Tree
* Random Forest
* KNN
* XGBoost

**Accuracy Results:**

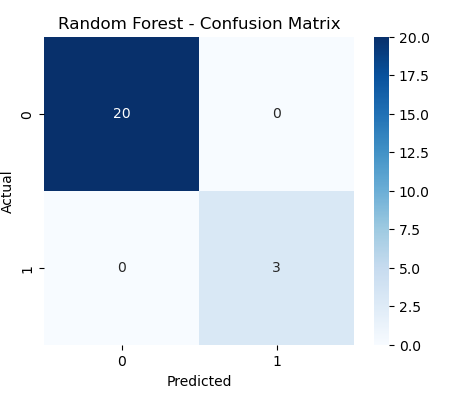
| **Model** | **Accuracy** |
| --- | --- |
| Decision Tree | 100% |
| Random Forest | 100% |
| XGBoost | 100% |
| KNN | 95.6% |

#### 🔹 Confusion Matrices:

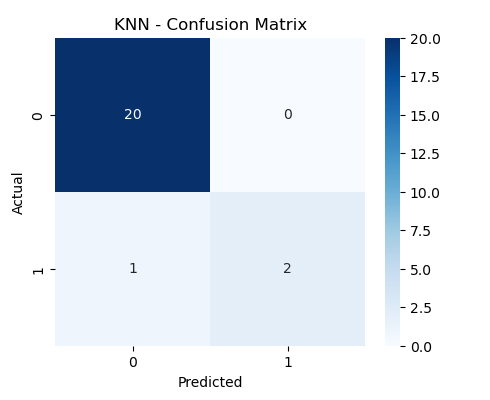
* Decision Tree :



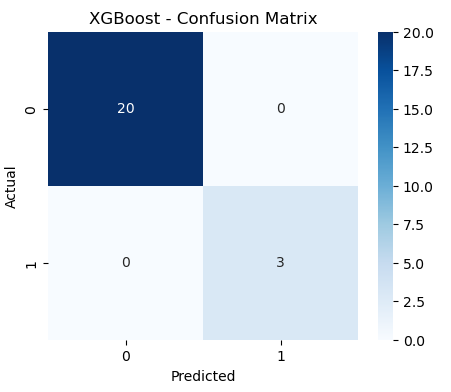
* Random Forest:



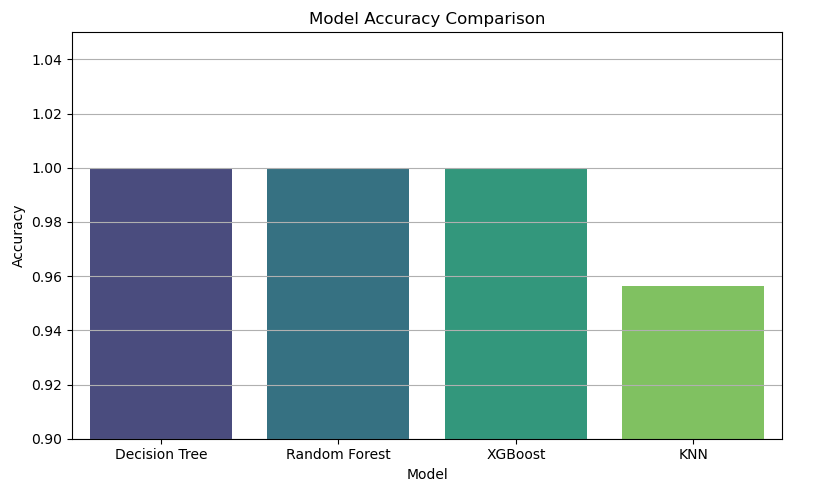
* KNN:



* XGBoost:

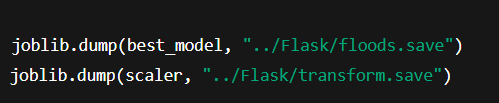


🔹 **Accuracy Comparison Chart :**



### 7. Model Saving

* Saved best model (Random Forest) as **floods.save**
* Saved StandardScaler as **transform.save**
* Both placed inside **Flask/ folder**



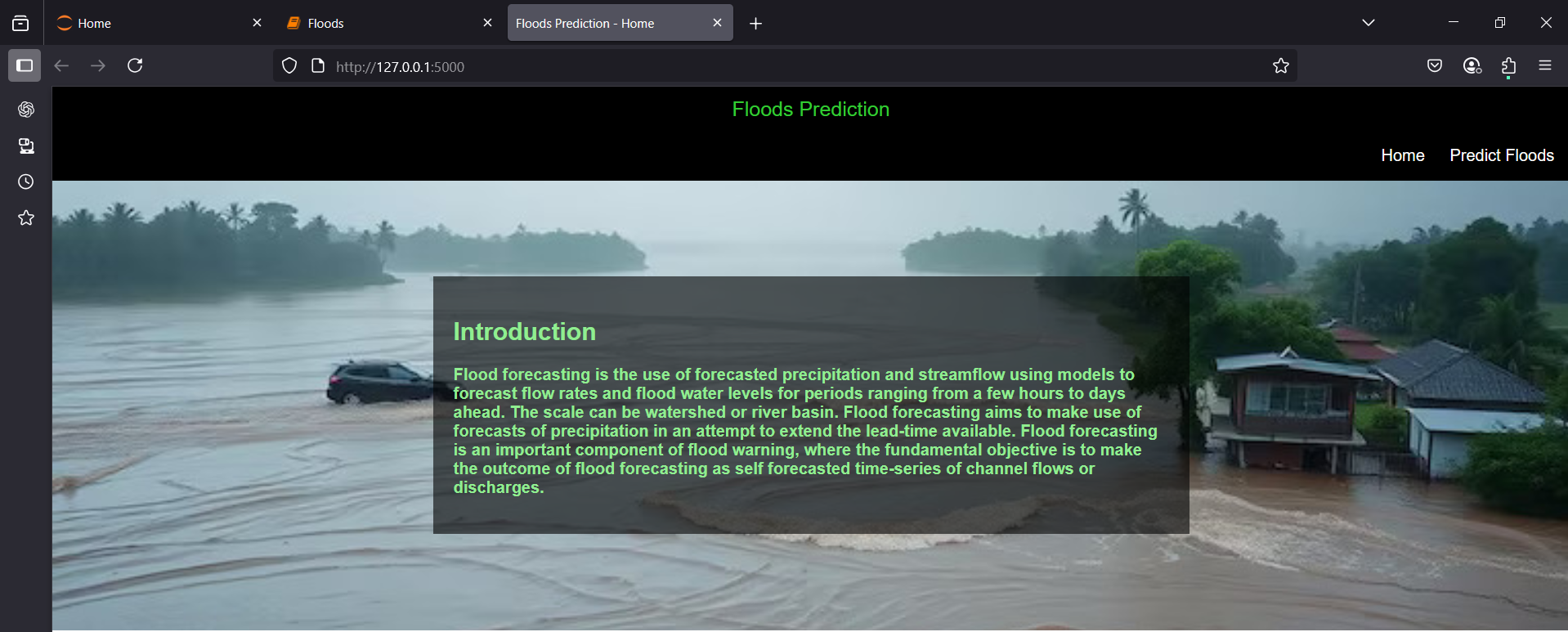
### 8. Web Application (Flask):

#### 8.1 Pages:

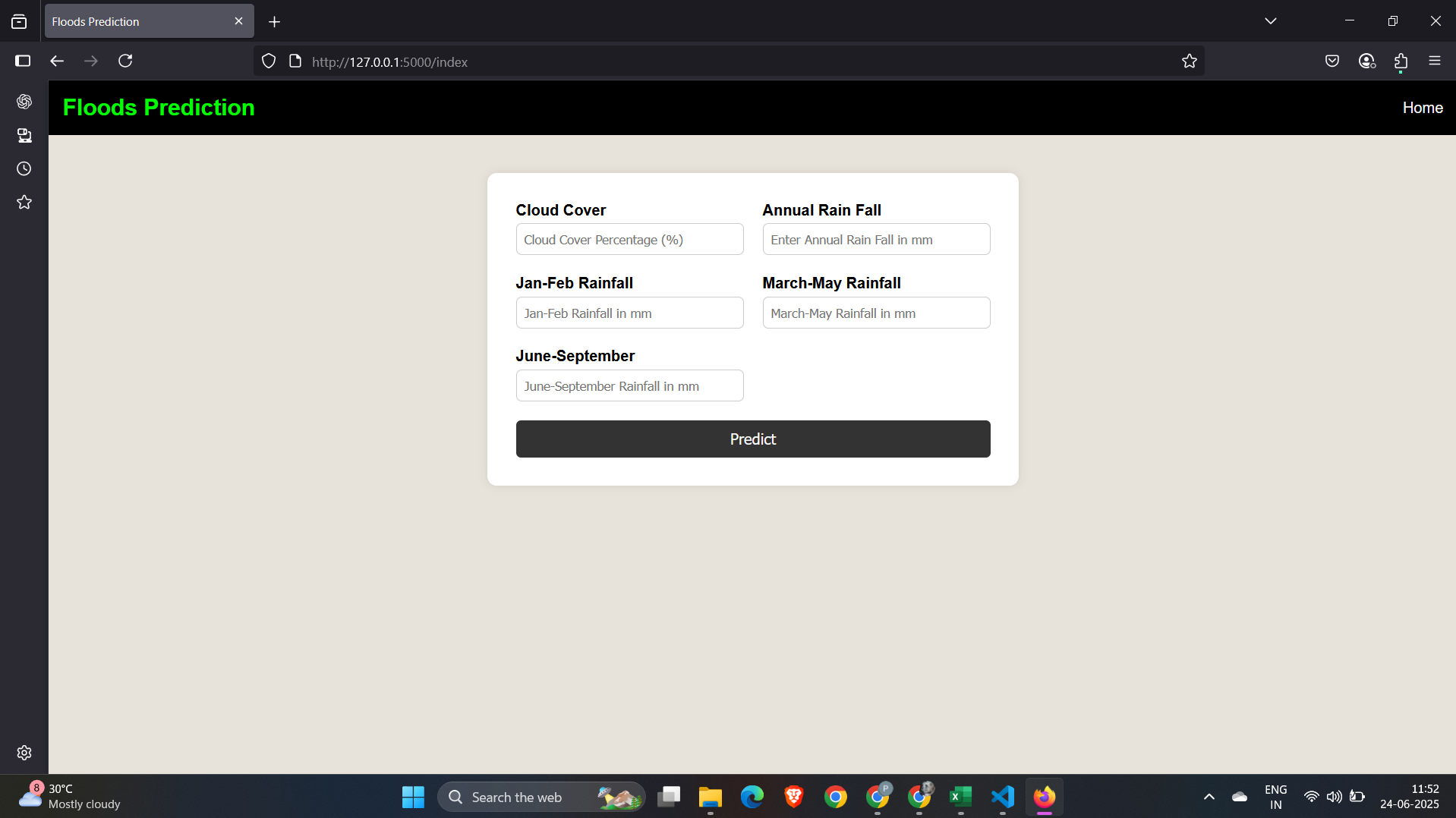
* Home **(home.html**): Intro page with background image.
* Index (**index.html**): Form to enter Cloud Cover, ANNUAL, Jan-Feb, Mar-May, Jun-Sep.
* Prediction Output:
  + **chance.html**: Displays "Possibility of severe flood"
  + **no\_chance.html**: Displays "No possibility of severe flood"

#### 8.2 Web Application UI:

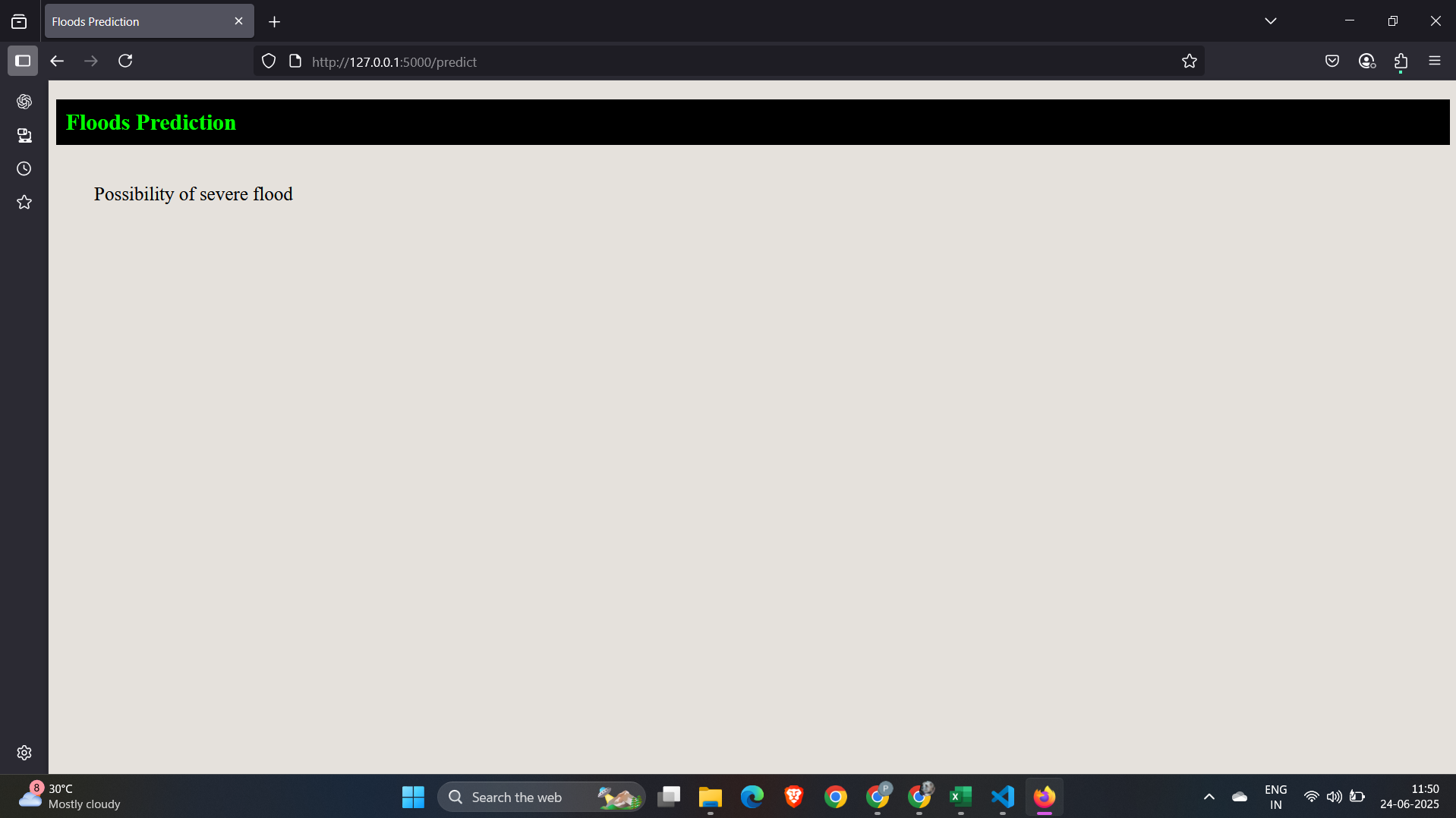
#### 🔹 Home Page:



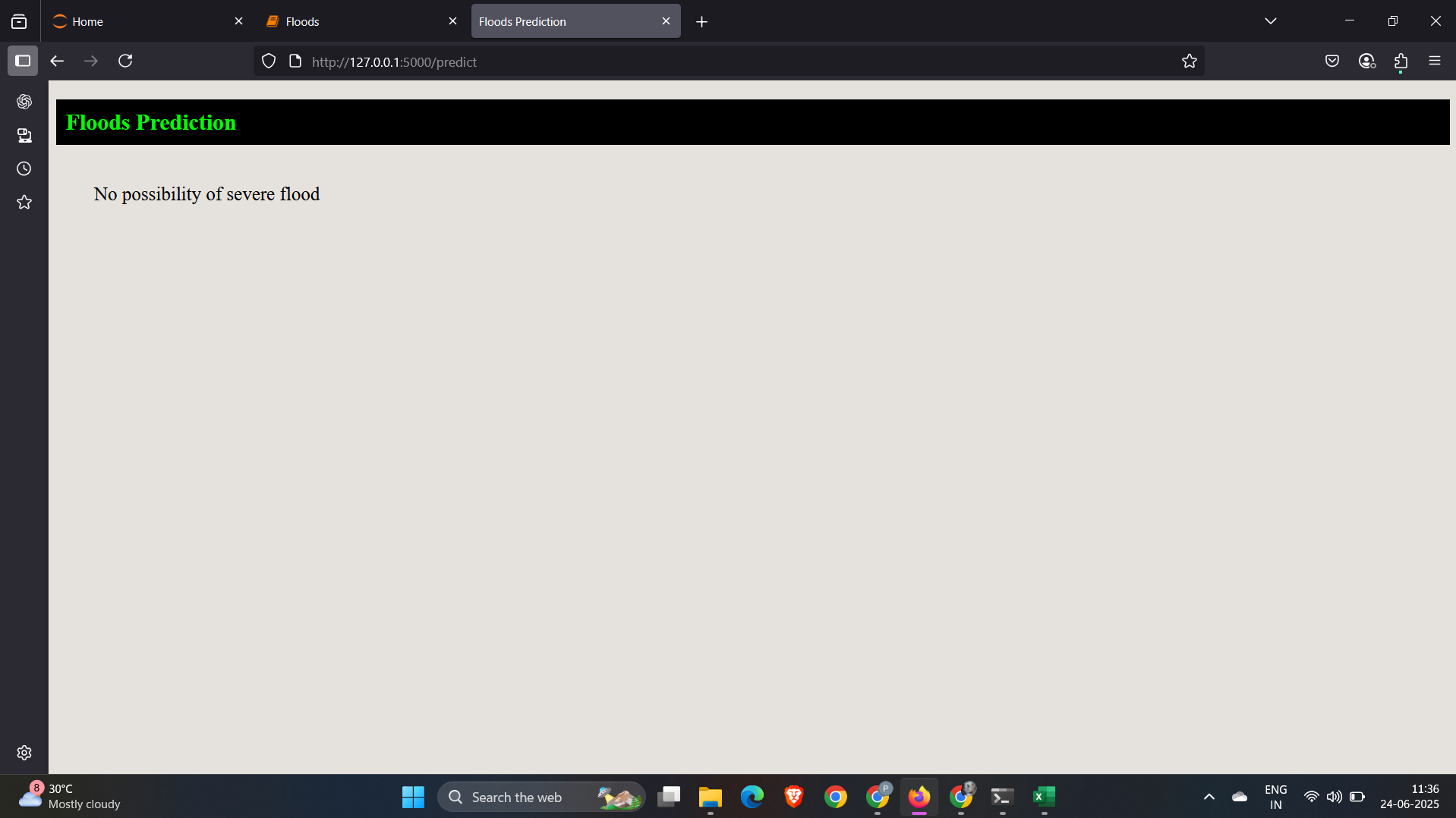
#### 🔹 Prediction Form Page:



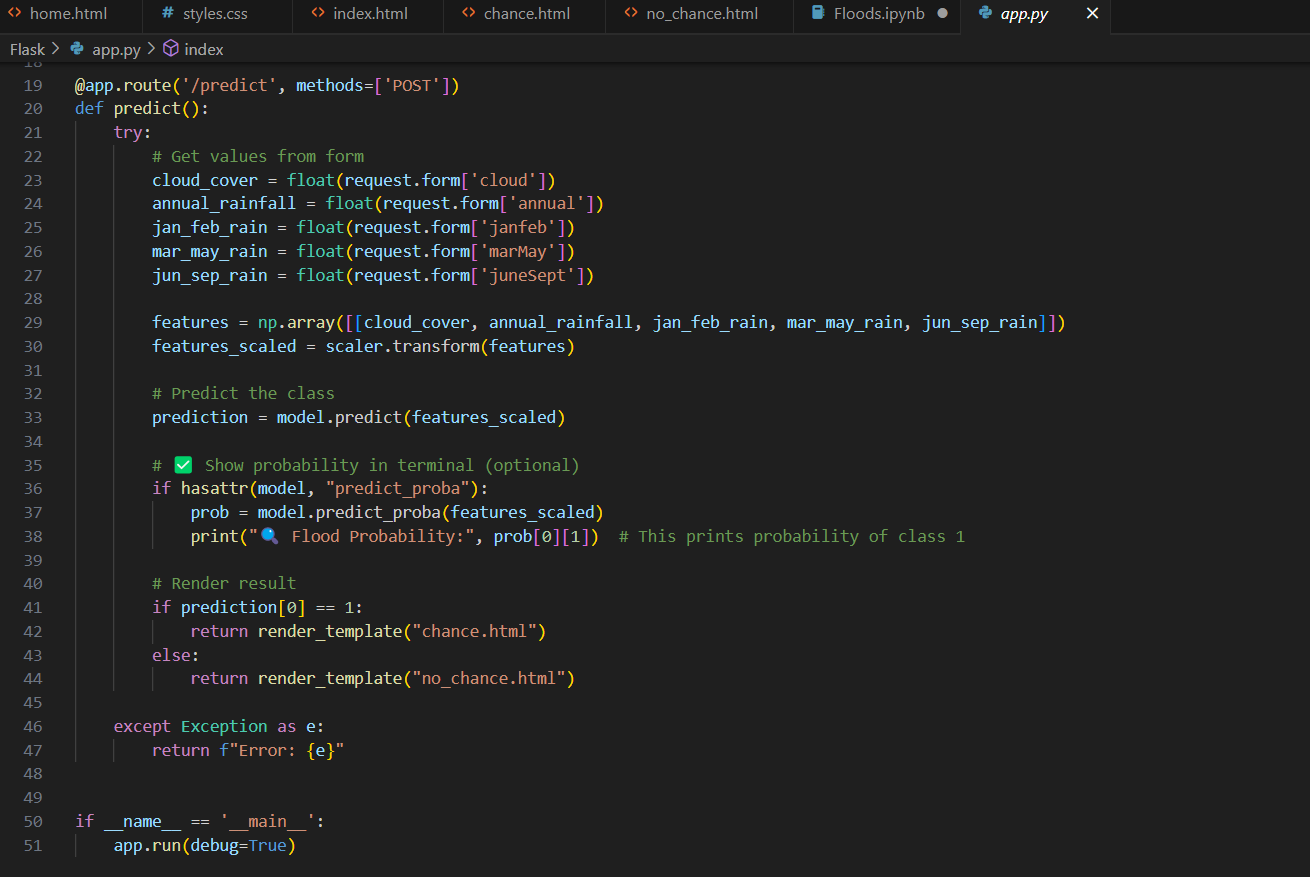
#### 🔹 Flood Prediction Output :



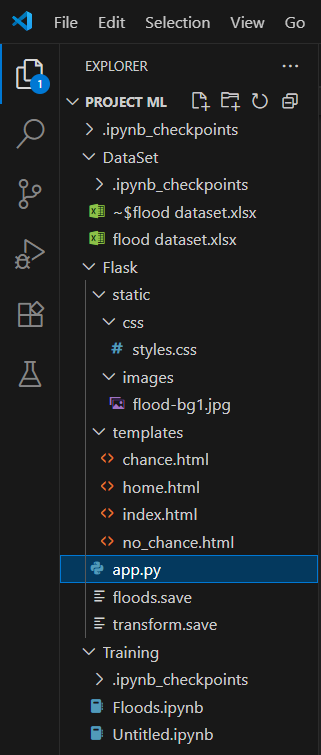
#### 🔹 No Flood Prediction Output:



#### 🔹 Prediction Code :



**9. Folder Structure :**



### **10.** Project Structure (for GitHub):

RisingWaters-MLFloodPrediction/

├── code/ # Jupyter Notebook + app.py

├── data/ # Original dataset

├── screenshots/ # Output plots, web app images

├── report/ # Project\_Report.docx

└── Flask/ # Flask web app folder (with templates/static/model)

### 11. Limitations and Final Notes:

Although the models (Decision Tree, Random Forest, XGBoost) achieved 100% accuracy during testing, it's important to consider the following:

* The dataset contains **725 samples**, which is a good amount for initial training.
* However, the **class distribution is imbalanced** — there are significantly more "No Flood" cases than "Flood" cases.
* This imbalance can lead to models that are biased toward predicting the majority class.
* As a result, high accuracy scores may not reflect true performance in real-world scenarios.

To improve robustness, future work should include:

* **Balancing the dataset** using techniques like SMOTE or undersampling.
* **Validating with real-world flood occurrence data** across different regions.
* **Using performance metrics like precision, recall, and F1-score**, especially for the minority class (flood = 1).