COD492 : Flood Prediction system for the river kosi basin

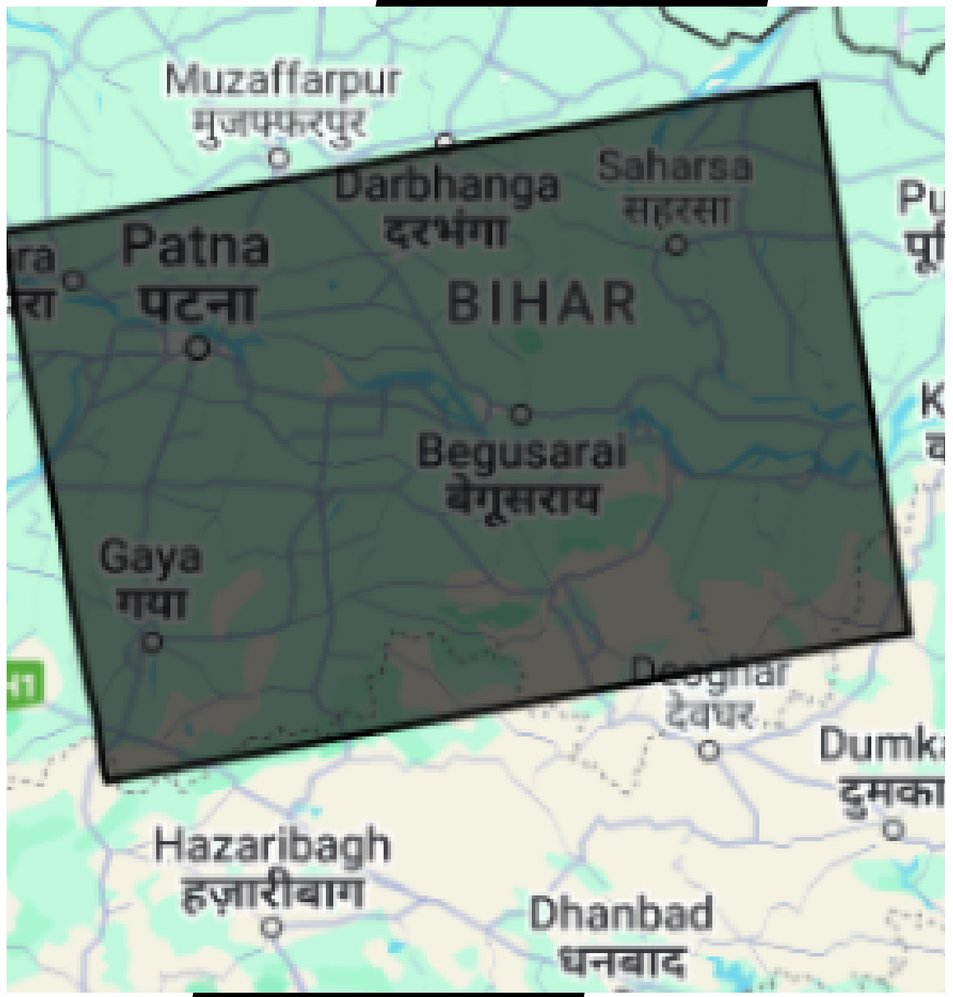
Vibhanshu Lodhi

Supervised by - Prof. Aaditeshwar Seth

Overview

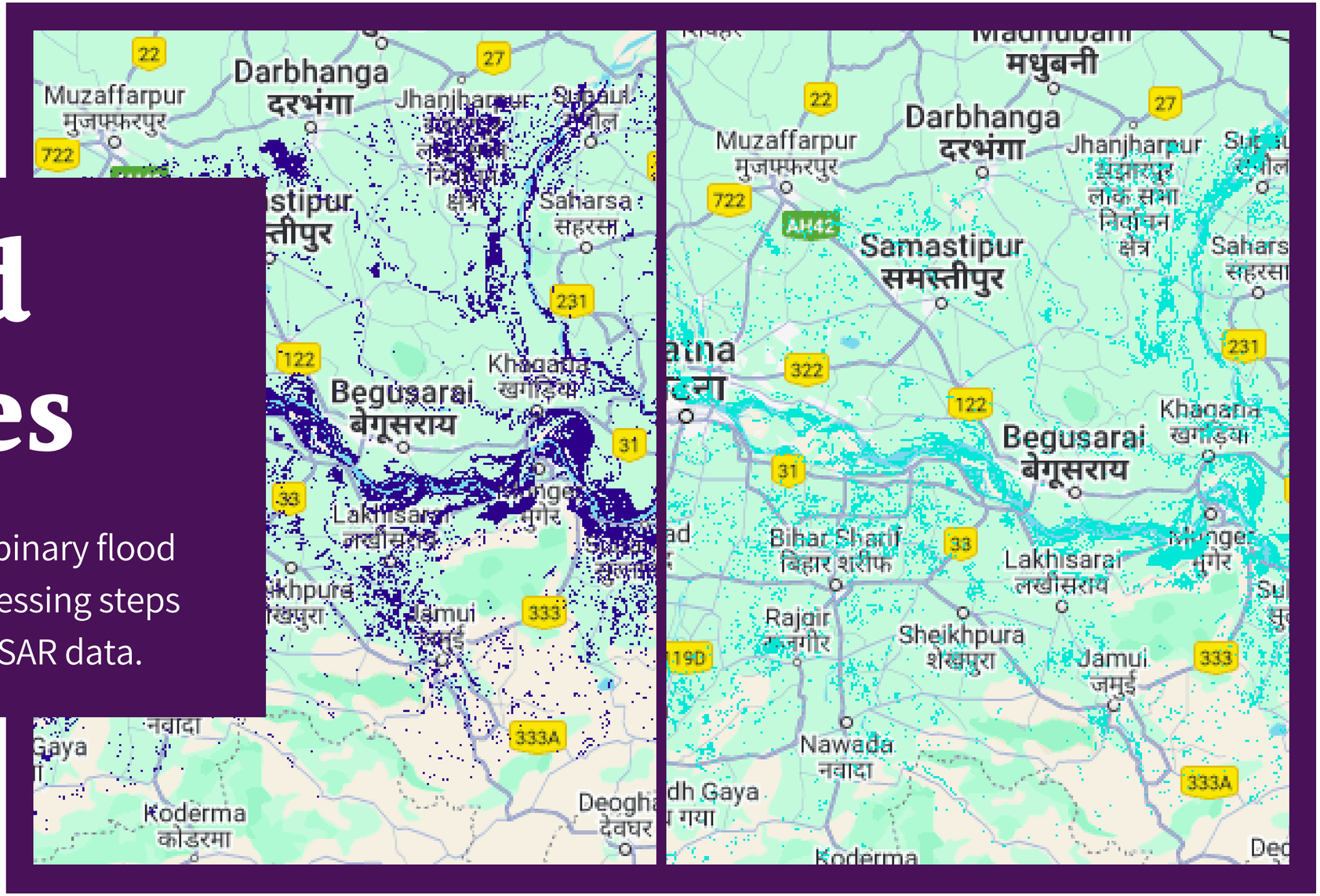
- Develop a machine learning flood predictive model for area situated near the River Kosi basin.
- Actual flood incidents would act as class labels for the model, and data on precipitation, runoff, terrain type etc. would act as feature.

Defining the ROI -



Flood Images

Here, the flood images are binary flood maps derived after preprocessing steps performed on Sentinel-1 SAR data.



Preprocessing Steps to Extract Flood Images

01

Data Source

- Satellite Data: Sentinel-1 (SAR) data, Chosen for its ability to capture surface details under all weather conditions, even during cloud cover or heavy rainfall.
- Auxiliary Data: JRC Global Surface Water dataset or similar datasets used to mask permanent water bodies.

02

Pre - Processing Steps

- Step 1: Radiometric Terrain Correction
  - Correct the radar backscatter values to adjust for terrain-induced distortions.
  - Improves data quality and ensures consistent reflectance values across the region.
- Step 2: Region of Interest Masking
  - Mask pixels outside the Kosi Basin to focus computations on the study area.

Preprocessing Steps to Extract Flood Images

Step 3: Filtering and Noise Reduction

- Apply a speckle filter to remove noise inherent in radar imagery while preserving edges and critical features.
- Reduces artifacts that could misclassify land as water.

Step 4: Otsu’s Thresholding Method

- Compute the backscatter histogram and apply Otsu’s Method to separate pixels into two classes:
  - Low Backscatter (Water): Flooded areas.
  - High Backscatter (Non-Water): Land or other non-flooded regions.
- The algorithm maximizes the variance between the two classes to create a clear binary separation.

Feature Preparation for Flood Prediction

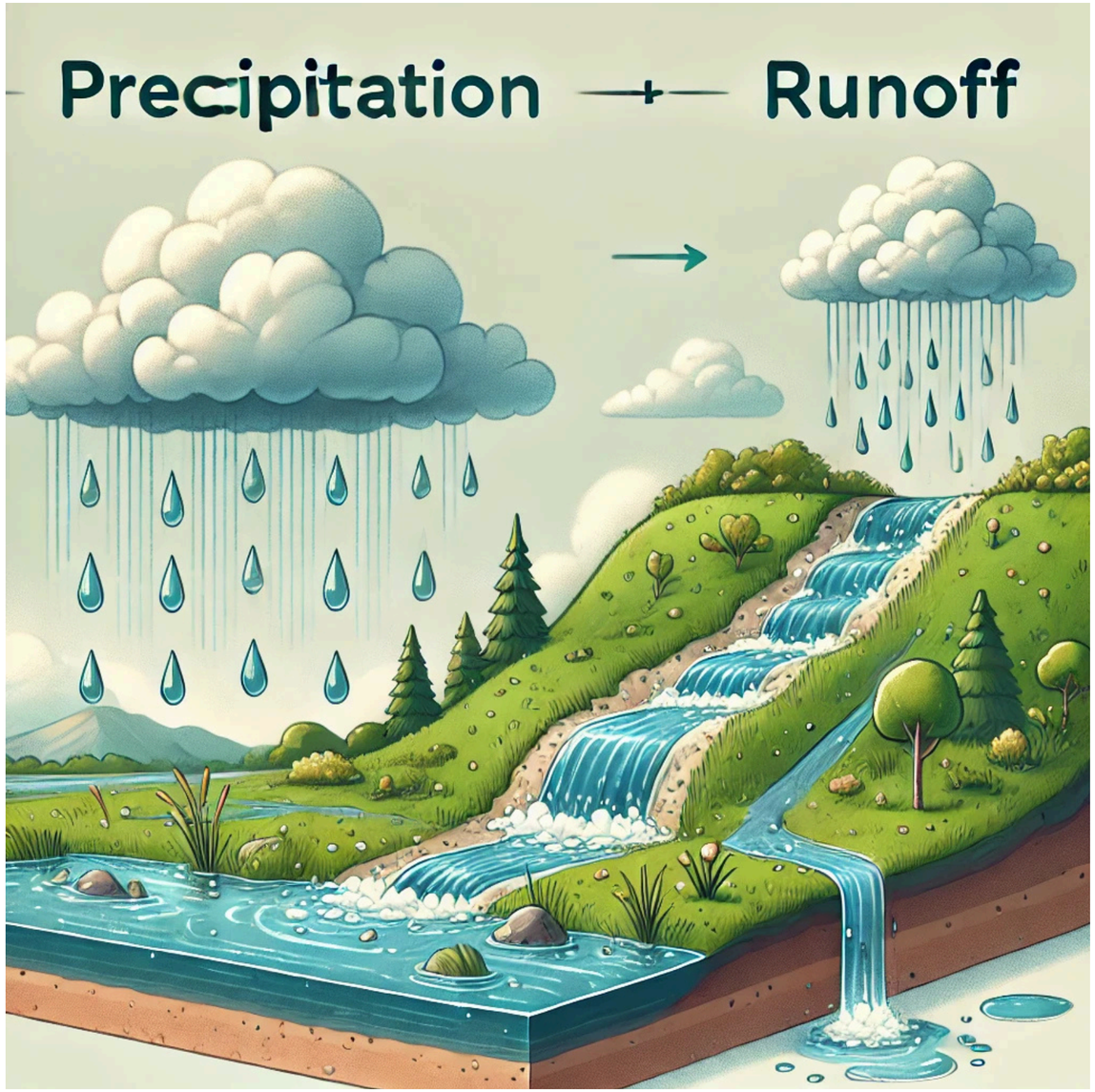
Key Features for Machine Learning Model

01

Features Used: Precipitation, Runoff, Elevation, Slope, TPI (Topographic Position Index), etc.

02

Objective: Preparing relevant features to improve model accuracy.



Categories of Features Used:

Key Features for Machine Learning Model

Meteorological: Total precipitation, runoff.

Topographical: Elevation, slope, TPI.

Custom Features: Derived combinations (e.g., relative slope).

Why these features?
They represent environmental factors affecting floods (e.g., precipitation is a direct trigger; terrain influences water flow).

Overview of Training Data Sampling

Objective: To generate a dataset for training a model using random points within a region of interest and associating them with feature values (e.g., precipitation).

Process Steps:

- Generate random sampling points.
- Assign feature values to these points using a collection.
- Prepare labeled data for training.

Generate Random Points

Generates 10,000 random points within the specified region of interest (table).

The points serve as the initial set of sampling locations.

Map Feature Values to Points

For each random point:

Use reduceRegion to get the feature value (constant) from the precipitation collection.

Assign the value to the point as a label property.

Reducer Used: ee.Reducer.first() selects the value at the first pixel within the point's geometry.

Scale: 30 meters determines the resolution of feature extraction.

Prepare Training Data

sampleRegions extracts feature values for the labeled random points.

Retains the property 'label' for training.

Scales data to match the resolution of the precipitation collection (30 meters).

Refine Training Data

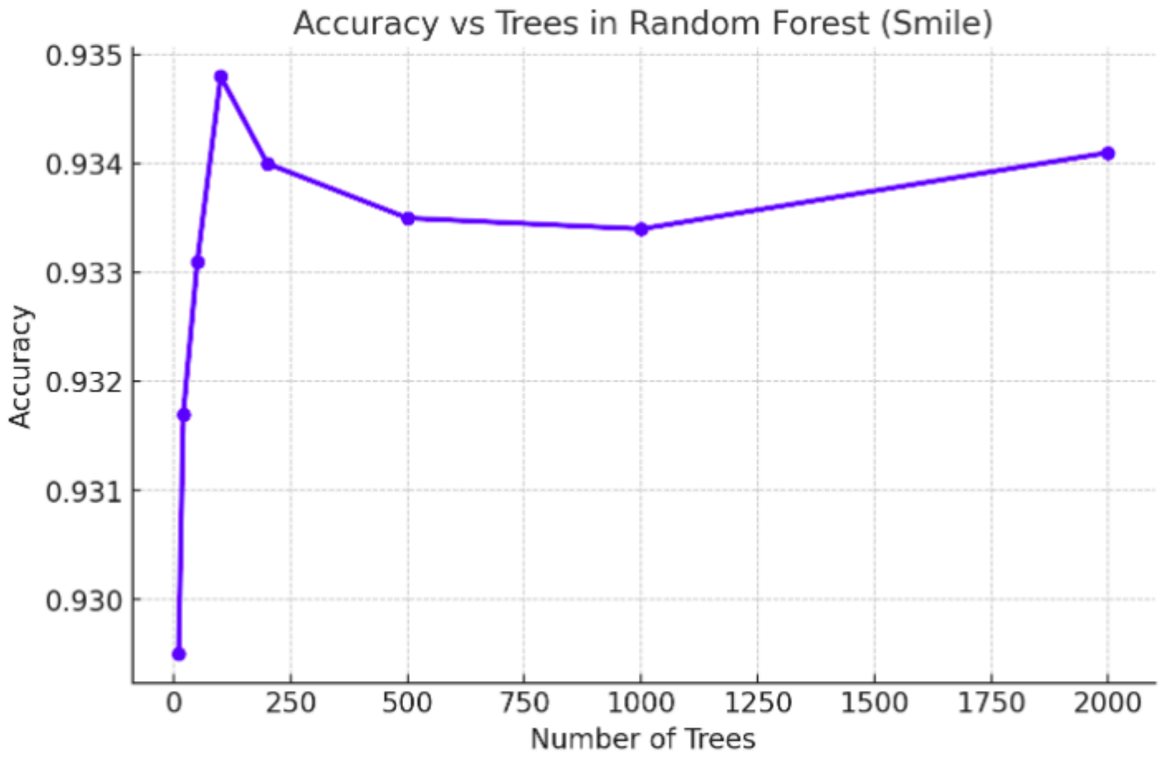
Ensures consistency in feature-label mapping by re-evaluating the feature value for each training point.

Refines the dataset to avoid potential mismatches in values.

High-quality labeled data ready for model training.

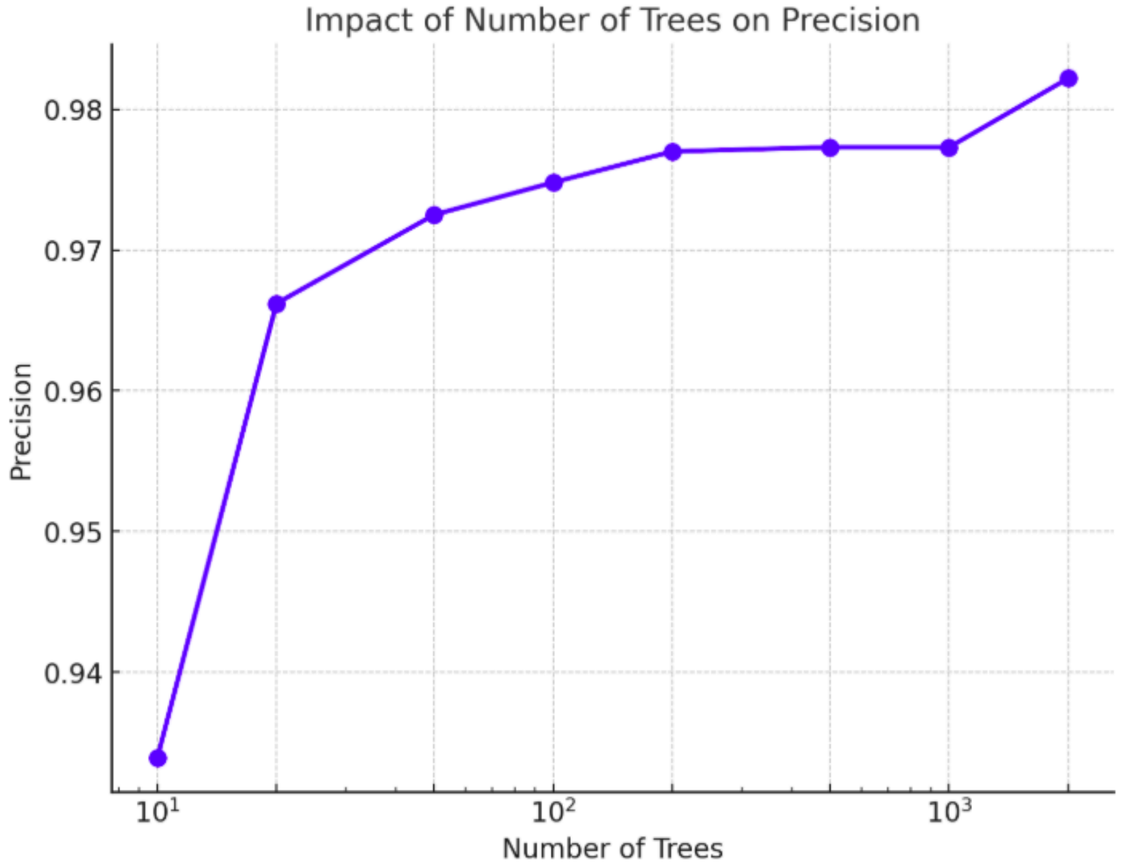
To access the GEE script for preparing the training data, please click here.

Key Results



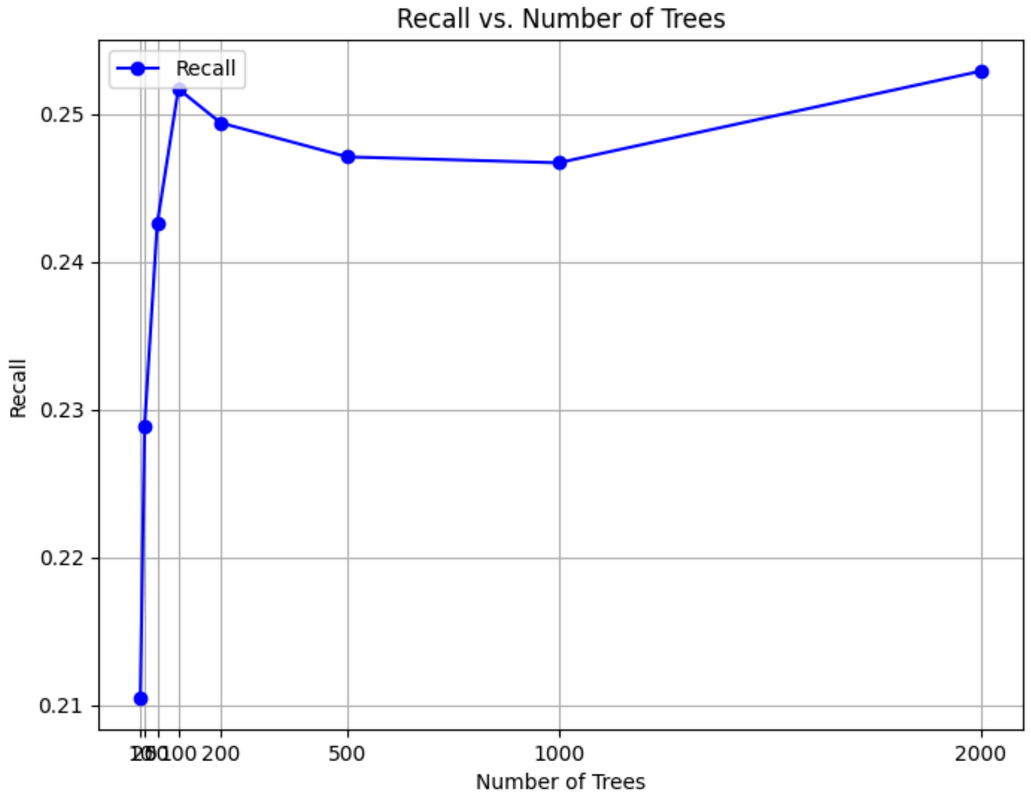
Impact of Number of Trees on Accuracy in *Random Forest (Smile)*

- As the number of trees increases, the accuracy tends to improve but eventually stabilizes.
- After 100 trees, the accuracy does not show significant improvements.
- A balance between computation cost and accuracy can be found around 100–200 trees.



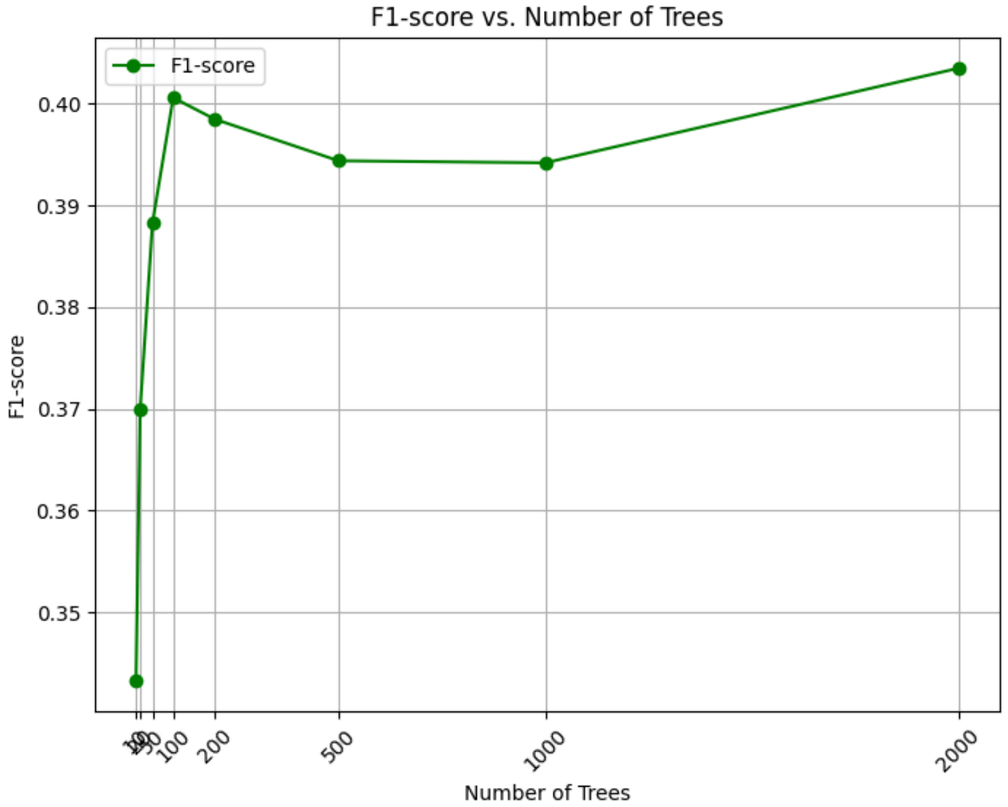
Impact of Number of Trees on Precision

- Precision increases with the number of trees, with diminishing returns after approximately 1000 trees.
- After 1000 trees, the precision stabilizes, and further increases do not lead to significant improvements.
- A good balance between model complexity and performance is achieved around 1000–2000 trees.



Impact of Number of Trees on Recall

- Recall increases with the number of trees, with notable improvement as the number of trees rises from 10 to 1000.
- Around 1000 trees, recall stabilizes, and further increases in the number of trees result in only minimal improvements.
- The highest recall value occurs at 2000 trees, though the change from 1000 to 2000 trees is marginal, suggesting diminishing returns.
- A good balance between model complexity and performance is achieved around 1000–2000 trees, where recall stabilizes and no substantial gains are observed with more trees.



Impact of Number of Trees on F1-score

- F1-score increases as the number of trees grows, showing improvement from 10 trees to 1000 trees.
- After 1000 trees, the F1-score reaches a plateau, with only minimal increases seen when moving from 1000 to 2000 trees.
- The maximum F1-score is observed at 2000 trees, indicating that while the performance improves as more trees are added, the gain becomes marginal after 1000 trees.
- The optimal range for balancing model complexity and performance is between 1000 and 2000 trees, as F1-score stabilizes after this point.