Urban Flood Detection in South Africa

50.039 Deep Learning Project Report - Group 3

Github: https://github.com/leemzius/flood-prediction

Name	Student ID
Leemah Bisht	1006054
Nellie Khoo Nee Liq	1006213
Tay Jia Jiun Joshua	1005904

1. Introduction

Problem context: The challenge focuses on urban flood detection in South Africa using deep learning. Many past floods, particularly in cities, were underreported or lacked precise records, limiting predictive capabilities. Traditional hydrological models often miss localised events due to infrastructure and topographical variations.

Addressing the problem: Build a model that identifies the exact day a flood occurred using CHIRPS precipitation data and Earth observation imagery.

2. Problem

2.1 Task Definition

The dataset consists of the following:

- **Composite Images:** Each event/location has a composite cloudless image (6 bands: Blue, Green, Red, NIR, SWIR, and slope). Note that some events might not have an image.
- **Precipitation Data:** For each event, there are 730 days (2 years) of daily aggregated CHIRPS precipitation data (over a 5km radius).
- **Flood Label**: For each event in the training dataset, there is a flood label, where 1 corresponds to a flood occurring and 0 otherwise.

The goal of the project is to develop a model that outputs the probability that a flood occurred at that time step (binary classification) when given a sequence of precipitation readings and associated six-channel satellite composites for each event ID and time step.

2.2 Dataset

Dataset source: *“Inundata: Mapping Floods in South Africa” Competition*

Dataset size: Data collected over 730 days

Dimension and format: Composite images of 128x128 resolution with 6 channels/bands per image in npz format

Due to the sparsity of identified floods, the given dataset is imbalanced, where there are some "events" or "locations" that do not have a flood in any of the 730 days.

- Details on how the imbalance is handled?
- Any additional remarks about dataset collection and preparation techniques

3. Approach

Our approach is grounded in understanding how **spatial features extracted from satellite imagery** interact with **continuous precipitation data** to predict flood occurrence. Rather than treating each image–timestamp pair in isolation, we designed the dataset to be **location-centric**, grouping flood and non-flood samples by geographic region. This strategy addresses class imbalance more meaningfully than naive resampling methods and ensures that model learning is driven by **spatial variability and environmental context**. For further details, refer to the **EDA notebook**.

System Pipeline

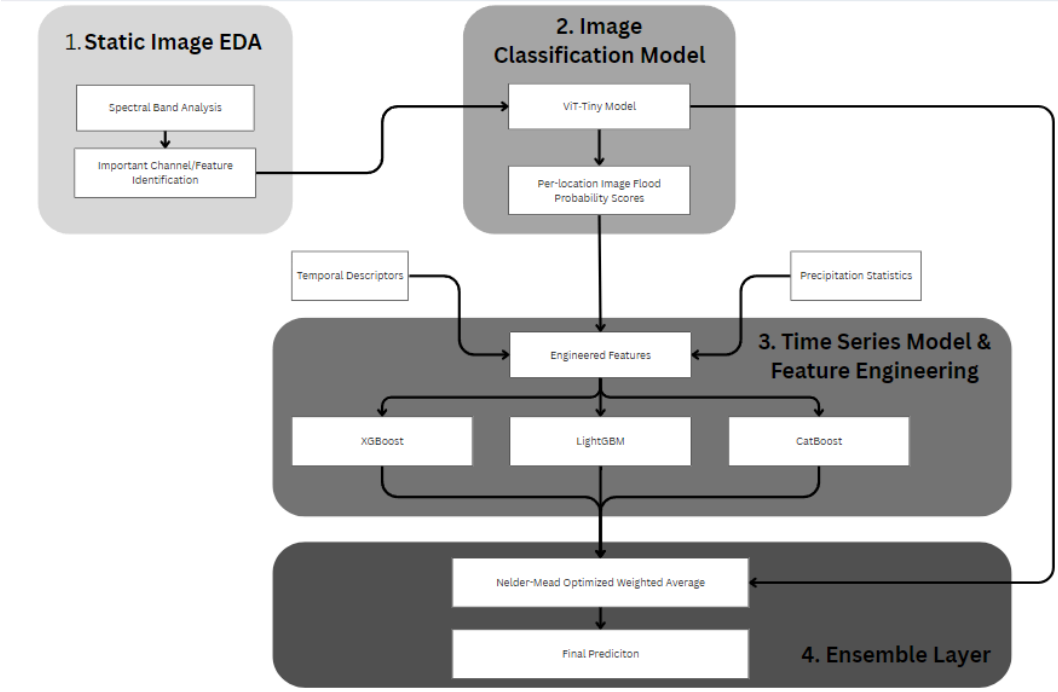

Fig 3.1: Model architecture

The modelling pipeline is structured into four key stages, with each stage contributing features to the next:

1. **Static Image EDA**
   Preliminary analysis identified the most informative spectral bands for flood detection.

2. **Image Classification (Experimentation between Ensemble ResNet18 & ViT-Tiny)**
   Model Option 1: A convolutional neural network (CNN) ensemble extracts features from each modality independently before fusing them for final classification.

   Model Option 2: A lightweight Vision Transformer model generates **per-image flood probability scores**, acting as spatial priors.

3. **Time Series Modeling & Feature Engineering**
   Gradient-boosted decision tree models (XGBoost, LightGBM, CatBoost) are trained

on **engineered features** including precipitation statistics, ViT-derived probabilities, and temporal descriptors.

4. **Ensemble Layer**
   Final predictions are obtained via a **Nelder–Mead–optimized weighted average** of individual model outputs, enhancing robustness across diverse input conditions.

All models are trained using **binary cross-entropy loss**, with Adam used for deep learning components and default optimizers for boosting frameworks.

4. Implementation

4.1 Static-Image EDA

The objective is to determine the optimal combination of any 3 bands/channels for training the image classifier.

4.1.1 Background on the images provided

Referencing, preliminary analysis on the data, we have the knowledge below:

1. Each image has 730 events
2. The images are annual cloud-free composite images from Sentinel-2 satellite imagery. They are of size 128x128 and contain the following 6 channels/bands:

Band	**Use/Explainability the Band Provides**
Sentinel-2 B2 (blue)	Band 2 is useful for soil and vegetation discrimination, forest type mapping and identifying man-made features. It is scattered by the atmosphere, it illuminates material in shadows better than longer wavelengths, and it penetrates clear water better than other colors. It is absorbed by chlorophyll, which results in darker plants.
Sentinel-2 B3 (green)	It gives excellent contrast between clear and turbid (muddy) water, and penetrates clear water fairly well. It helps in highlighting oil on water surfaces, and vegetation. It reflects green light stronger than any other visible color. Man-made features are still visible.
Sentinel-2 B4 (red)	It is strongly reflected by dead foliage and is useful for identifying vegetation types, soils and urban (city and town) areas. It has limited water penetration and doesn't reflect well from live foliage with chlorophyll.

Band	Use/Explainability the Band Provides
Sentinel-2 B8 (NIR)	The near infrared band is good for mapping shorelines and biomass content, as well as at detecting and analyzing vegetation.
Sentinel-2 B11 (SWIR 1)	It is useful for measuring the moisture content of soil and vegetation, and it provides good contrast between different types of vegetation. It helps differentiate between snow and clouds. On the other hand, it has limited cloud penetration.
Slope	Elevation changes of a location.

Given that the images are essentially spatial representations of the environment for that location, it acts like a snapshot. Hence, the image itself cannot provide temporal insights but its spatial features such as topographic features and environmental conditions can be extracted to get insights into features that influence flood occurrence.

This is especially useful as it helps us model weather knowledge such as:

1. Areas with high Normalized Difference Water Index (NDWI) might flood more frequently given the high precipitation.
2. Areas with high slope values might experience flash floods after an intense rainfall.

4.1.2 Experimenting with Different Channel Combinations

Our next step is to visualize if flood and non-flood images have differences visible to the human eye.

Step 1: Generate all possible 3-channel combinations out of the 6 bands available.

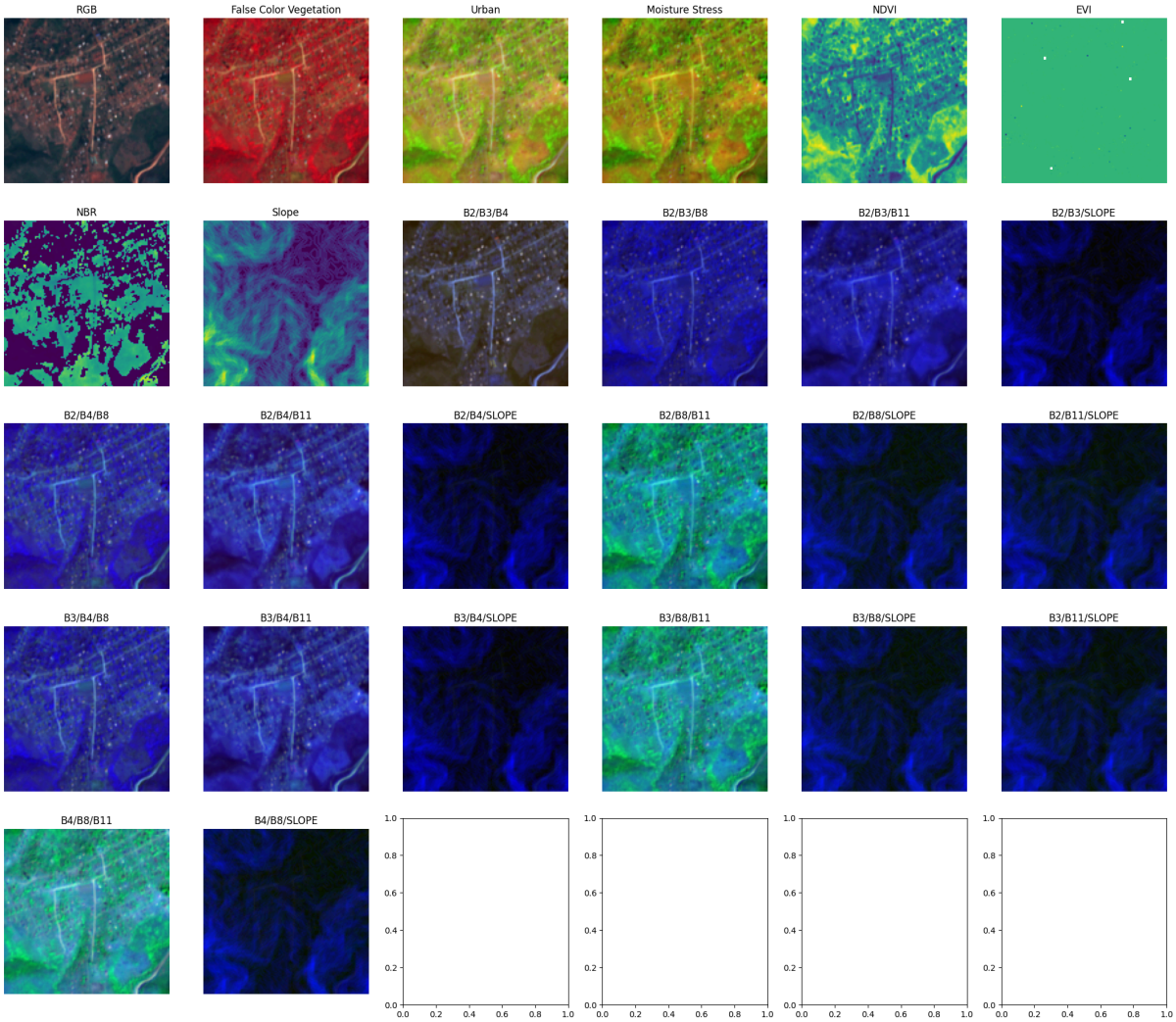

Fig 4.1. For a randomly selected location, the above are the images generated from possible channel combinations.

Step 2: Compare a flooded and non-flooded location image.

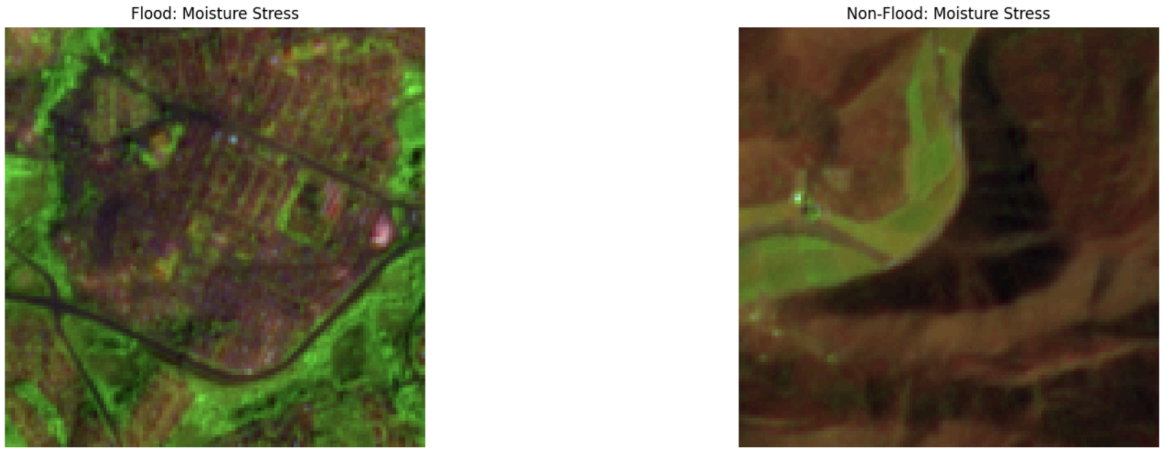

Fig 4.2. Example of a good band combination that has reduced noise yet unique distinguishable features.

Fig 4.3. Example of a poor band combination that is not the most informative on the location because it's mostly a color blur.

Step 3: Selecting Combinations of Channels.
The decision-making rules behind this were:

1. Reduced as much noise as possible
2. Had distinguishable features somewhat visible to the human eye.
3. The combination of bands was explainable.

Hence, the final selected combinations were:

1. Moisture Stress = B2, B8, B11
2. NIR11Slope = B8, B11, Slope

4.2 Image Classifier

Comparing the results between the ViT-Tiny & the ensemble image classifier
Model 1: Ensemble Image Classification Model

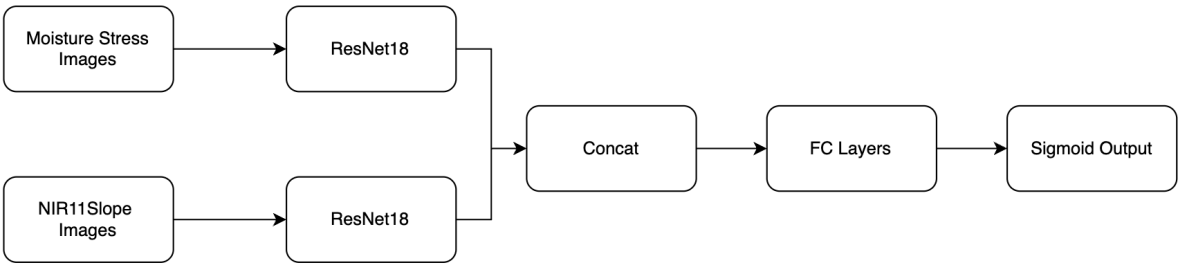

Fig 4.4. Proposed architecture for the image classification model 1.

Rationale: A convolutional neural network (CNN) ensemble extracts features from each modality independently before fusing them for final classification.
Step 1: Preparing the Dataloader
To prepare the dataset:

- All images were resized to 224×224.
- Images were stacked into tensors of shape [2, 3, 224, 224] per sample.
- A custom PyTorch Dataset class (FloodEventDataset) was implemented for dynamic loading.

Step 2: Image Classification Model Architecture
The model (FloodClassifier) is composed of:

a. Image Branches
- Two independent ResNet18 models (one per image modality).
- Each ResNet18 acts as a feature extractor (final layer removed).
- Parameters are frozen to leverage pre-trained weights on ImageNet.

b. Feature Fusion
- Feature vectors (512-dim) from each ResNet are concatenated → [batch_size, 1024].

c. Classification Head
- Fully connected layers: 1024 → 256 → 128 → 1
- Activation: ReLU + Dropout + Sigmoid
- Output: Binary prediction (flood/no flood)

We conducted two training experiments to evaluate the effect of learning rate on model convergence.

Experiment 1: learning rate 0.001, epoch = 20

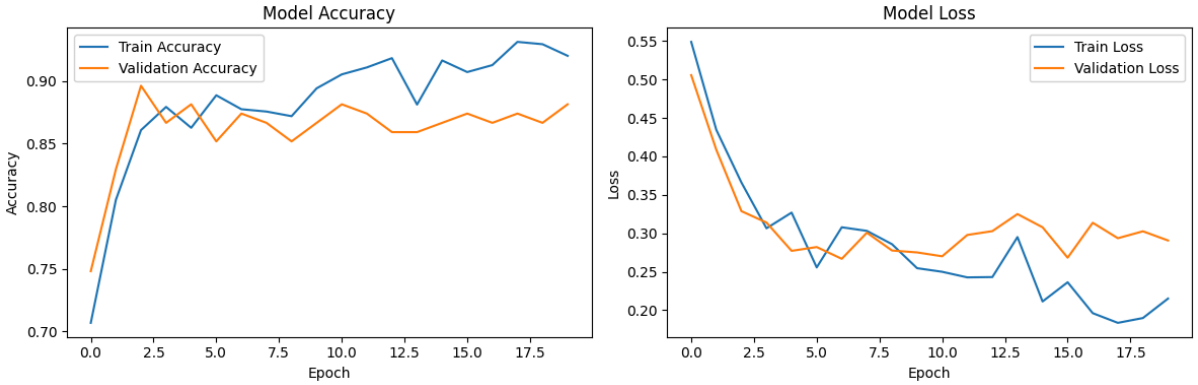

Fig 4.5. Accuracy and model loss for experiment 1.

Experiment 2: learning rate 0.0001, epoch = 20

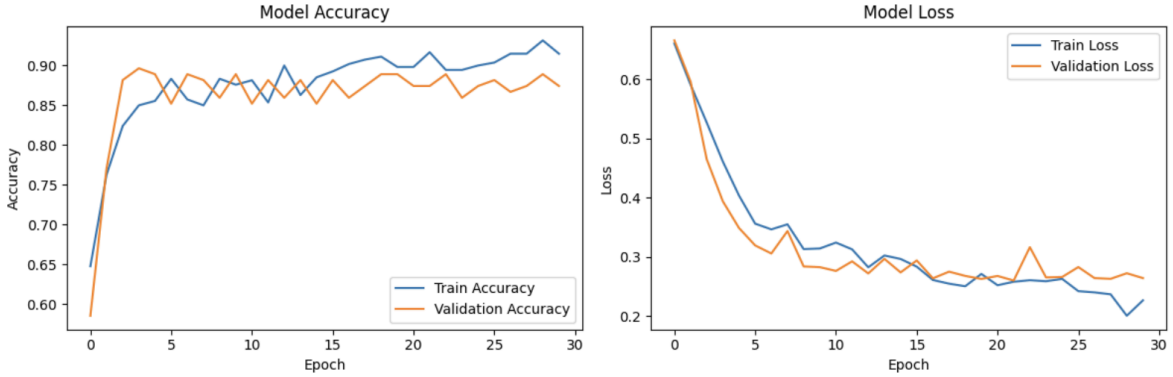

Fig X. Accuracy and model loss for experiment 2.

Conclusion: While the smaller learning rate had slightly weaker performance, the model training was smoother and the divergence between the training and validation set was also lower. Thus, our final picked hyperparameter was a learning rate of .0001.

Step 3: Model Inference for Generating Predictions for Ensemble in Stage 4
To incorporate image-based flood insights into the ensemble model, we generated flood probability predictions for each sample in both training and validation sets. These predictions were used as features for the subsequent model combining continuous CHIRPS precipitation data.

A custom inference function was developed to output a CSV file for the flood predictions based on the CV model. The function uses torch.no_grad() to perform inference without gradient tracking, improving efficiency during prediction. The resulting CSV was saved as flood_predictions.csv and used to enrich the multimodal dataset for the ensemble.

Furthermore, the inference data was standardized using the same normalization statistics as during training.

After inference, we evaluated model performance for both splits:
Train Accuracy: 0.9518
Validation Accuracy: 0.8741

Model 2: ViT-Tiny

Using Moisture Stress images, we train an existing image classifier (eva02_tiny_patch14_224) to predict the probability of flooding at each location.

4.3 Time Series Modeling & Feature Engineering

Our approach integrates information from both static environmental conditions and dynamic precipitation patterns. A core component involves leveraging pre-computed features derived from the provided 6-band composite satellite imagery (Sentinel-2 B2, B3, B4, B8, B11, and SRTM Slope). Statistical summaries (mean, median, standard deviation) of key spectral indices – such as NDVI, NDWI, MNDWI, MSI, EVI – and topographic slope were calculated for each location (`location_id`). Crucially, a static, location-based flood probability estimate, derived from a separate image classification model analyzing moisture-related image characteristics via cross-validation, was incorporated as a powerful indicator of inherent flood risk.

These static spatial and risk features were combined with an extensive set of temporal features engineered from the daily CHIRPS precipitation data. This included:

- Raw daily precipitation values.
- Lagged precipitation values spanning up to 364 days prior.
- Rolling window statistics (mean) over various periods (e.g., 3, 7, 30, up to 296 days).

- Exponentially Weighted Moving Averages (EWMA) to capture recent rainfall trends (e.g., 7-day span).
- Temporal indicators such as the day index (`event_t`) within the 730-day sequence.

This rich, combined feature set formed the input for predicting daily flood probabilities. To capture diverse patterns within the data, we utilized predictions generated from multiple distinct modeling techniques known for high performance on structured data. Specifically, our final ensemble incorporated predictions derived from:

- A LightGBM model.
- A CatBoost model.
- A FastAI Tabular deep learning model.

These underlying predictions were generated using robust `StratifiedGroupKFold` cross-validation in their respective development processes, ensuring the Out-Of-Fold (OOF) predictions used for ensembling were representative of generalization performance.

4.4 Ensemble Layer

To maximize predictive accuracy and robustness, we implemented a sophisticated ensemble strategy combining the outputs from the individual models (LightGBM, CatBoost, FastAI Tabular).

1. **Optimized Weighted Ensembling:** Rather than a simple average, optimal weights for combining the model predictions were determined. We utilized the Nelder-Mead optimization algorithm (`scipy.optimize.minimize`) applied to the Out-Of-Fold (OOF) predictions available for each model from the training set. The objective was to find weights wi that minimized the Binary Cross-Entropy (Log Loss) of the weighted average ∑wi×OOFi against the true labels, effectively learning the best way to combine the models based on their cross-validated performance.

2. **Probability Normalization:** A final calibration step was applied to the optimized ensemble predictions. This step leverages the pre-computed static, location-based flood probability estimate (`static_prob`). A threshold for this static probability was identified by minimizing the Log Loss of the *normalized, weighted OOF ensemble* predictions. For locations identified as having high inherent flood risk (i.e., `static_prob` >= optimized threshold), the weighted ensemble probability for each day was divided by the sum of all daily probabilities for that location (plus a small epsilon). This normalization adjusts prediction confidence based on the baseline spatial risk, demonstrably improving the final Log Loss score.

5. Reproducibility

This section details the steps required to reproduce the final predictions and results generated by our ensemble pipeline.

4.1. Environment Setup

The project requires Python 3. Key dependencies include `pandas`, `numpy`, `scikit-learn`, and `scipy`. A `requirements.txt` file detailing specific versions is provided in the code repository. Ensure these libraries are installed in your environment.

4.2. Data Acquisition

1. Create a new notebook environment (e.g., on Kaggle).
2. Press File > Import Notebook. The project's curated Kaggle dataset should automatically be loaded. If unsuccessful, add our dataset using the "Add Data" feature. Search for dataset name/slug: jaytaykay/flood-prediction.
3. This dataset will be mounted at `/kaggle/input/flood-prediction/` and contains:
    - Original competition files (`Train.csv`, `Test.csv`, `SampleSubmission.csv`).
    - The static flood probability results file (`cv_flood_predictions.csv`).
    - The necessary Stage 3 model prediction files (`fastai_train_with_oof.csv`, `fastai_test_with_oof.csv`, `gbdt_train_with_oof.csv`, `gbdt_test_with_oof.csv`).

4.3. Execution

1. Import the primary execution notebook, `fourth_stage_final.ipynb`, from the Github repository (https://github.com/leemzius/flood-prediction/) into your Kaggle environment.
2. Ensure the file paths in the notebook's `CFG` class point correctly to the input files within `/kaggle/input/flood-prediction/`.
3. Run all cells in the notebook sequentially.
4. The script performs the following automated steps:
    - Loads required prediction and static probability data.
    - Calculates optimal ensemble weights based on loaded OOF data.
    - Applies weights to the loaded test predictions.
    - Calculates the optimal normalization threshold based on OOF data.
    - Applies normalization to the weighted test predictions.
    - Generates the final submission file named `submission.csv` in the `/kaggle/working/` output directory.
5. You may then run the submission file on the hackathon page (https://zindi.africa/competitions/inundata-mapping-floods-in-south-africa, requires login) for the final test score.

4.4. Model Loading The final result is generated by applying the ensemble procedure rather than loading a single monolithic model file. Reproducing the exact submission involves running the execution script (Step 4.3), which deterministically calculates and applies the optimized ensemble weights and normalization threshold based on the provided input prediction files.

6. Results

Our developed multi-modal ensemble pipeline achieved a high ranking in the Zindi "Inundata: Mapping Floods in South Africa" competition. The performance, evaluated using the Binary Cross-Entropy (Log Loss) metric, is as follows:

- **Private Leaderboard Log Loss: 0.002469533**
- **Public Leaderboard Log Loss: 0.002318223**
- **Final Rank: 9th / 315**

This result places our solution within the top 3% of participants, highlighting the effectiveness of our methodology.

Out-Of-Fold (OOF) Performance:

Analysis during development using cross-validated OOF predictions confirms the benefit of each stage:

- Individual Model OOF Log Loss (Examples):
    - LightGBM: 0.002428
    - CatBoost: 0.002672
    - FastAI Tabular: 0.002653
- Uniform Ensemble OOF Log Loss: 0.002307
- Optimized Weighted Ensemble OOF Log Loss: 0.002252
- Final Normalized OOF Log Loss: **0.002241**

7. Discussion

The strong 9th place ranking achieved demonstrates the efficacy of our integrated approach for predicting precise flood event timing. The success can be attributed to several key methodological components:

- **Multi-Modal Feature Synergy:** Combining dynamic temporal precipitation features (raw values, lags, rolling statistics, EWMA) with static spatial context (derived spectral index statistics) and location-based risk assessment (pre-computed static flood probability) provided a comprehensive input for modeling. The static probability, in particular, proved essential for both feature enrichment and final calibration.

- **Ensemble Power:** Leveraging predictions from multiple distinct modeling paradigms (Gradient Boosted Trees via LightGBM/CatBoost and Deep Learning via FastAI Tabular) captured different patterns in the data. The use of optimized weighting, derived from OOF performance, yielded significantly better results than any single model or a simple average, highlighting the benefits of diversity and data-driven weight optimization in ensembling.
- **Probability Calibration:** The final normalization step, guided by the location's static flood risk probability and an optimized threshold, served as an effective calibration technique. By rescaling probabilities in high-risk areas, it refined the predictions and demonstrably improved the Log Loss score on OOF data, likely contributing significantly to the final leaderboard standing.

While successful, the pipeline has limitations. Its performance relies on the quality of the input predictions from the individual models and the accuracy of the static flood probability estimates. The multi-stage nature adds complexity compared to end-to-end models. Future work could explore integrating these components more deeply, potentially using attention mechanisms for feature fusion or incorporating more advanced sequence models like Transformers tailored for multivariate time series within the ensemble. Investigating the sensitivity to the normalization threshold and exploring alternative calibration methods could also yield further improvements.

8. Conclusion

This project successfully developed and executed a high-performance machine learning pipeline for the challenging task of precise urban flood event timing prediction. By integrating multi-modal data sources, employing extensive feature engineering, utilizing an optimized ensemble of diverse model predictions, and applying a final risk-based probability normalization, our solution achieved a top-10 rank (9th/315) in the competitive Zindi "Inundata" challenge, with a Private Leaderboard Log Loss of 0.002469533. This work demonstrates the power of sophisticated ensemble techniques and context-aware calibration in tackling complex, imbalanced spatio-temporal forecasting problems, offering valuable insights for operational flood prediction systems.