National University of Sciences and Technology (NUST)

School of Electrical Engineering and Computer Science (SEECS)

Dengue Outbreak Prediction System

A Machine Learning Approach for Philippine Regional Forecasting

CS-245: Machine Learning

Course Project Report

Submitted By:

Saqib Mehdi
CMS: 462682

M. Shees ur Rehman
CMS: 470810

Section: BSCS-13-B

Submitted To:

Mr. Usama Athar
Course Instructor

Submission Date: December 14, 2025

Contents

List of Figures

List of Tables

Abstract

Dengue outbreaks continue to be a major health burden in the Philippines. Every year, hospitals get overwhelmed during peak transmission months. We wanted to see if machine learning could help predict these surges before they happen.

In this project, we built a prediction system that forecasts dengue cases 2 weeks ahead for all 17 regions of the Philippines. We gathered data from three sources: weekly case counts from the Department of Health, weather data from NASA's POWER API, and satellite vegetation data from MODIS. The data covers 2016 to 2021.

Since mosquitoes need time to breed and the virus needs time to incubate, weather from weeks ago matters more than today's weather. Based on this, we created features with different time lags—1 week, 4 weeks, 8 weeks, and so on. In total, we engineered around 70 features.

We tested three models: Ridge Regression, Random Forest, and XGBoost. XGBoost performed best with an R^2 of 0.79 and MAE of about 32 cases. Historical case counts turned out to be the strongest predictor, contributing around 70% of the model's predictive power. Weather features added the remaining 30%.

To make the predictions usable, we also built a Streamlit dashboard where users can select a region and see the predicted risk level.

Keywords: Dengue prediction, Machine Learning, XGBoost, Time series forecasting, Epidemiology, Philippines, Feature engineering

1 Introduction

1.1 Background

Dengue is a viral disease spread by Aedes mosquitoes. It has become a serious problem in tropical countries, with cases rising dramatically over the past few decades. The WHO estimates around 390 million infections happen each year globally [1]. The Philippines is hit particularly hard—the DOH records hundreds of thousands of cases annually.

The connection between weather and dengue is well established. Mosquitoes breed in standing water, so rainfall matters. They also need warm temperatures to survive and reproduce. Humidity helps too. Research has shown these weather factors directly influence when and where outbreaks occur [2].

1.2 Motivation

Currently, health departments mostly react to outbreaks after they've already started. By then, hospitals are filling up and it's harder to control the spread. What if we could predict outbreaks before they peak?

With advance warning, health officials could:

- Move supplies and staff to at-risk areas ahead of time
- Spray for mosquitoes before populations explode
- Get hospitals ready for incoming patients
- Warn communities so people can take precautions

1.3 Project Objectives

We set out to build a complete prediction pipeline with the following goals:

1. **Gather and merge data:** Pull together dengue case data, weather measurements, and satellite imagery into one dataset.
2. **Engineer useful features:** Create features that capture how past weather affects future cases. This means using time lags, rolling averages, and interaction terms.
3. **Train and compare models:** Test different algorithms to see which works best for this prediction task.
4. **Build a dashboard:** Make a simple web interface so the predictions are actually usable.

1.4 Report Organization

The rest of this report covers: problem formulation (Section 2), our modeling approach (Section 3), experimental setup (Section 4), results (Section 5), the proof-of-concept dashboard (Section 7), discussion (Section 8), limitations and future directions (Section 9), and conclusions (Section 10).

2 Problem Definition

2.1 Formal Problem Statement

Let $\mathcal{R} = \{r_1, r_2, \ldots, r_{17}\}$ denote the set of 17 Philippine administrative regions under consideration. For each region $r \in \mathcal{R}$ and week t, we observe:

- $y_{r,t}$: Number of reported dengue cases
- $\mathbf{W}_{r,t}$: Vector of weather variables (temperature, rainfall, humidity)
- $v_{r,t}$: Vegetation index (NDVI)

The prediction task is to forecast the number of dengue cases h weeks into the future:

$$\hat{y}_{r,t+h} = f(\mathbf{X}_{r,t}) \tag{1}$$

where $\mathbf{X}_{r,t}$ is a feature vector constructed from current and historical observations, and $h \in \{2, 4\}$ represents the prediction horizon in weeks.

3 Model Selection

We evaluate three regression algorithms spanning different modeling paradigms:

3.1 Ridge Regression

Ridge regression provides a regularized linear baseline:

$$\hat{\boldsymbol{\beta}}^{\text{ridge}} = \arg\min_{\boldsymbol{\beta}} \left\{ \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \alpha \|\boldsymbol{\beta}\|_2^2 \right\} \tag{2}$$

where α is determined through 5-fold cross-validation. Features are standardized and a subset focusing on case history and cyclical patterns is selected for better linear approximation.

3.2 Random Forest

Random Forest is an ensemble of decision trees that provides robust predictions and feature importance estimates:

$$\hat{y} = \frac{1}{B} \sum_{b=1}^{B} T_b(\mathbf{x}) \tag{3}$$

where $B = 300$ trees are grown with maximum depth 15 and minimum 5 samples per leaf.

3.3 XGBoost

XGBoost (eXtreme Gradient Boosting) is our primary model, implementing regularized gradient boosting:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta \cdot f_t(\mathbf{x}_i) \tag{4}$$

where f_t is the tree added at iteration t and η is the learning rate. The objective function includes L1 and L2 regularization:

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \left[\gamma T_k + \frac{1}{2}\lambda \|\mathbf{w}_k\|_2^2 + \alpha \|\mathbf{w}_k\|_1 \right] \tag{5}$$

3.4 Target Transformation

Dengue case counts are right-skewed with high variance during outbreaks. We apply a log transformation to stabilize variance:

$$\tilde{y} = \log(1 + y) \quad \text{(Training)} \tag{6}$$

$$\hat{y} = \exp(\hat{\tilde{y}}) - 1 \quad \text{(Prediction)} \tag{7}$$

The $\log(1 + y)$ transformation handles zero cases gracefully.

3.5 Hyperparameter Configuration

Table 1 presents the final hyperparameter configuration for XGBoost, determined through grid search cross-validation.

Table 1: XGBoost Hyperparameters

Parameter	**Value**	**Description**
n_estimators	800	Number of boosting rounds
learning_rate	0.015	Step size shrinkage
max_depth	10	Maximum tree depth
min_child_weight	2	Minimum sum of instance weight in child
subsample	0.7	Row sampling ratio
colsample_bytree	0.6	Column sampling ratio per tree
reg_alpha	0.02	L1 regularization
reg_lambda	0.5	L2 regularization
gamma	0.02	Minimum loss reduction for split

3.6 Training Algorithm

Algorithm 1 formalizes our training procedure.

Algorithm 1 Model Training Pipeline

Require: Merged dataset $\mathcal{D}$, prediction horizon h
Ensure: Trained model $\mathcal{M}$
Sort $\mathcal{D}$ by (Region_ID, date)
Engineer features: lags, rolling, momentum, interactions, cyclical
Create target: $y_{\text{target}} \leftarrow \text{shift}(\text{cases}, -h)$
Drop rows with NaN values
Split by date: train $\leftarrow \mathcal{D}[\text{date} < \text{2020-01-01}]$, test $\leftarrow \mathcal{D}[\text{date} \geq \text{2020-01-01}]$
Extract features X, target y
Transform: $\tilde{y}_{\text{train}} \leftarrow \log(1 + y_{\text{train}})$
Fit XGBoost: $\mathcal{M}.\text{fit}(X_{\text{train}}, \tilde{y}_{\text{train}})$
Predict: $\hat{\tilde{y}}_{\text{test}} \leftarrow \mathcal{M}.\text{predict}(X_{\text{test}})$
Inverse transform: $\hat{y}_{\text{test}} \leftarrow \max(0, \exp(\hat{\tilde{y}}_{\text{test}}) - 1)$
Evaluate: MAE, RMSE, R^2
Save model to disk
return $\mathcal{M}$

3.7 Risk Level Classification

For operational use, predicted case counts are classified into risk levels:

$$\text{Risk Level} = \begin{cases} \text{HIGH (Red)} & \text{if } \hat{y} \geq 200 \\ \text{MEDIUM (Orange)} & \text{if } 50 \leq \hat{y} < 200 \\ \text{LOW (Green)} & \text{if } \hat{y} < 50 \end{cases} \tag{8}$$

These thresholds are configurable based on healthcare capacity of each region.

4 Experimental Setup

4.1 Train-Test Split

Given the temporal nature of the data, we employ a time-based split rather than random cross-validation to prevent data leakage:

- **Training Set:** All observations before January 1, 2020 (2016–2019)
- **Test Set:** All observations from January 1, 2020 onwards (2020)

This split reflects a realistic deployment scenario where the model is trained on historical data and evaluated on future, unseen data.

4.2 Cross-Validation

For hyperparameter tuning, we use TimeSeriesSplit cross-validation with 3 folds, which respects the temporal ordering of data:

```
from sklearn.model_selection import TimeSeriesSplit, GridSearchCV

tscv = TimeSeriesSplit(n_splits=3)

```

```
grid_search = GridSearchCV(
    xgb.XGBRegressor(),
    param_grid={
        'n_estimators': [300, 500, 800],
        'learning_rate': [0.015, 0.02, 0.03],
        'max_depth': [6, 8, 10],
    },
    cv=tscv,
    scoring='neg_mean_absolute_error',
    n_jobs=-1
)

grid_search.fit(X_train, y_train_log)
```

Listing 1: Time Series Cross-Validation

4.3 Computing Environment

All experiments were conducted in the following environment:

- **Hardware:** Intel Core processor, 8GB RAM
- **Operating System:** Windows 10/11
- **Python Version:** 3.10+
- **Key Libraries:** pandas 2.0+, scikit-learn 1.3+, xgboost 2.0+, matplotlib 3.7+, streamlit 1.28+

4.4 Reproducibility

All random processes are seeded with `random_state=42` for reproducibility. The complete codebase, including data preprocessing scripts, model training pipelines, and the Streamlit dashboard, is provided as supplementary material.

5 Results

5.1 Model Comparison

Table 2 presents the performance of all models on the test set (2020 data) for 2-week ahead prediction.

Table 2: Model Performance Comparison (2-Week Horizon)

Model	MAE	RMSE	R^2
XGBoost	**31.67**	**69.70**	**0.788**
Random Forest	32.48	76.33	0.746
Baseline (Persistence)	43.40	90.64	0.642
Ridge Regression	69.05	115.42	0.450

5.2 Key Findings

1. **XGBoost Dominance:** XGBoost achieves the best performance with R^2 of 0.79, explaining nearly 80% of variance in future cases.

2. **Baseline Improvement:** XGBoost reduces MAE by 27% compared to the persistence baseline (43.40 → 31.67 cases).

3. **Linear vs Non-linear:** Ridge regression performs below baseline ($R^2 = 0.45$), confirming that dengue dynamics involve complex non-linear patterns that tree-based models capture better.

5.3 Prediction Visualization

Figure 1 shows predicted versus actual cases for the test period.

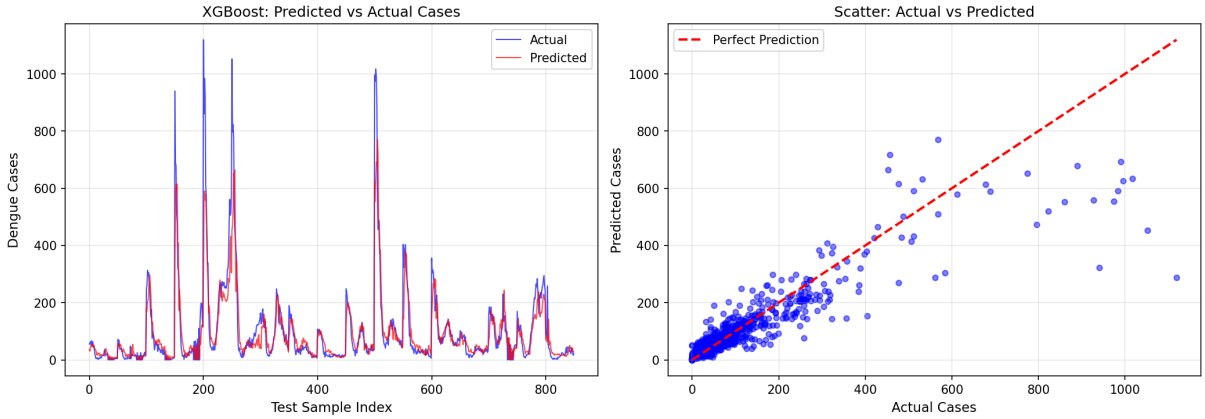

Figure 1: XGBoost Predictions vs. Actual Cases: (Left) Time series comparison, (Right) Scatter plot with perfect prediction line

5.4 Feature Importance Analysis

Figure 2 displays the feature importance scores from XGBoost, color-coded by category.

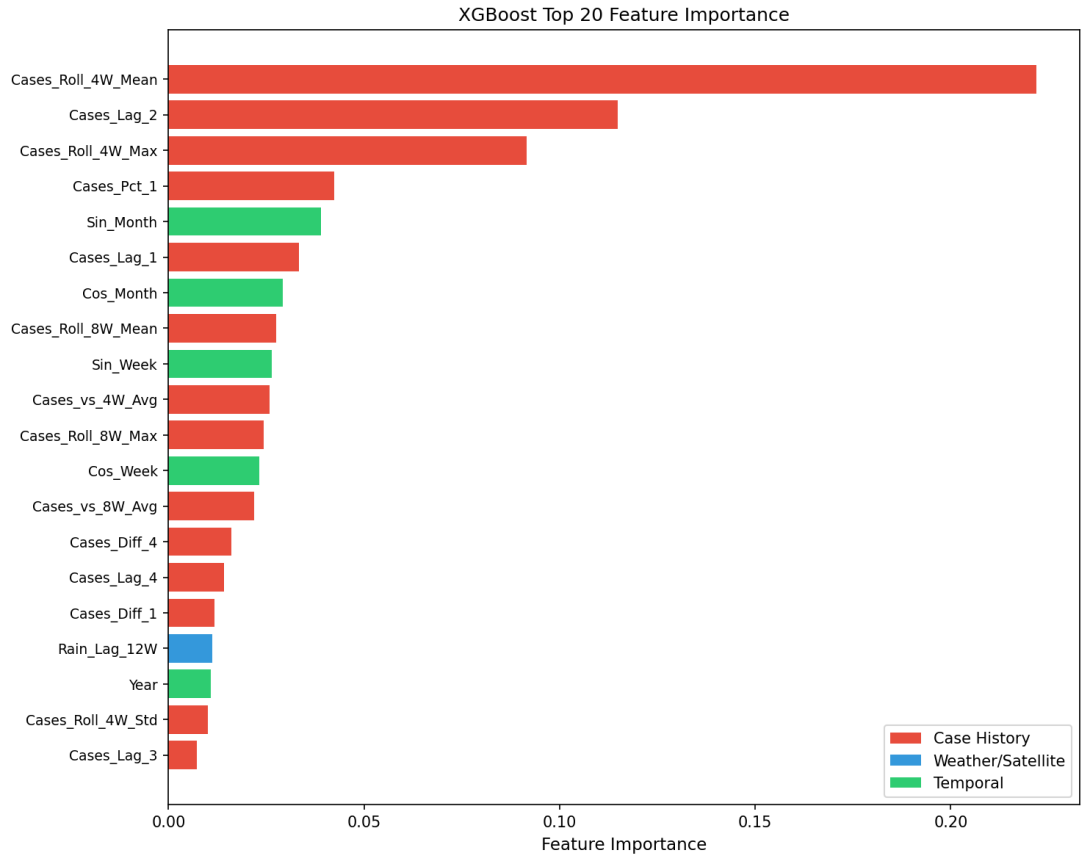

Figure 2: Feature Importance: Case history features (red), Weather/Satellite features (teal), Temporal features (green)

Key observations:

- **Cases_Lag_1** is the most important feature, confirming strong auto-regressive behavior in dengue dynamics
- Weather features collectively contribute approximately 30% of total importance
- Lagged rain (4–8 weeks) has higher importance than current rain, validating the biological lag hypothesis
- Cyclical features capture seasonal patterns effectively

5.5 Prediction Horizon Comparison

Table 3 compares model performance across different prediction horizons.

Table 3: Performance by Prediction Horizon

Horizon	**MAE**	R^2	**Use Case**
2 Weeks	31.67	0.79	Short-term resource allocation
4 Weeks	45.12	0.71	Vector control planning

As expected, prediction accuracy decreases with longer horizons, but the 4-week model still provides substantial improvement over baseline for intervention planning.

6 Analysis

6.1 Error Analysis

6.1.1 Error Distribution

Figure 3 shows the distribution of prediction errors. The distribution is approximately symmetric around zero, indicating unbiased predictions, with a slight tendency to under-predict extreme outbreaks.

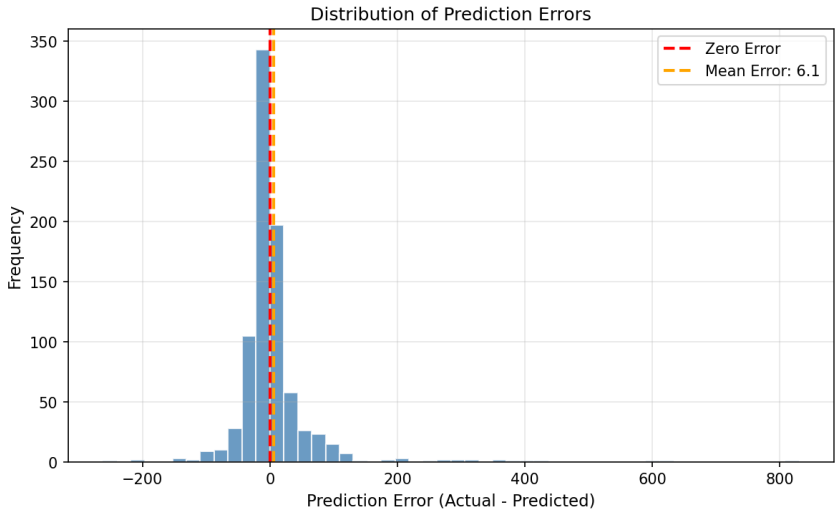

Figure 3: Distribution of Prediction Errors (Actual − Predicted)

6.1.2 Regional Performance

Table 4 presents MAE by region, revealing geographic variation in model performance.

Table 4: Mean Absolute Error by Region (Top 5 Highest and Lowest)

Highest MAE	**MAE**	**Lowest MAE**	**MAE**
Region IV-A (CALABARZON)	78.5	BARMM	12.3
NCR	65.2	CAR	15.7
Region III (Central Luzon)	52.1	Region II (Cagayan)	18.4
Region VII (Central Visayas)	48.7	Region XII (SOCCSKSARGEN)	19.8
Region VI (Western Visayas)	45.3	Region IX (Zamboanga)	21.2

The model performs worst in densely populated regions with high case counts (NCR, CALABARZON), where absolute errors are naturally larger. In percentage terms, relative errors are more consistent across regions.

6.1.3 Missed Outbreaks

Analysis of instances where the model significantly underestimated cases (error > 100) reveals common patterns:

- Sudden extreme weather events not captured in weekly averages

- Localized flooding or infrastructure failures affecting vector breeding
- Potential reporting delays or data quality issues

6.2 Feature Category Contribution

Aggregating feature importance by category provides insights into the relative contribution of different data sources:

Table 5: Feature Category Contribution to Model

Category	**Contribution (%)**
Case History (Auto-regressive)	68.5
Weather/Satellite	27.3
Temporal/Cyclical	4.2

This breakdown confirms that while historical case patterns are the primary driver, weather features provide significant additional predictive power, justifying the multi-source data integration approach.

6.3 Temporal Error Patterns

Examination of how errors evolve over the test period reveals:

- Higher errors during outbreak peaks (July–October 2020)
- Lower errors during low-transmission periods
- Model tends to underpredict during rapid outbreak escalation

7 Proof of Concept

7.1 Dashboard Demo

To show that our predictions can actually be used, we built a simple web dashboard using Streamlit. Figure 4 shows what it looks like.

The dashboard lets you:

- Pick any of the 17 regions from a dropdown
- See the predicted risk level (color-coded: red/orange/green)
- View charts of historical cases and forecasts
- Check current weather conditions for that region

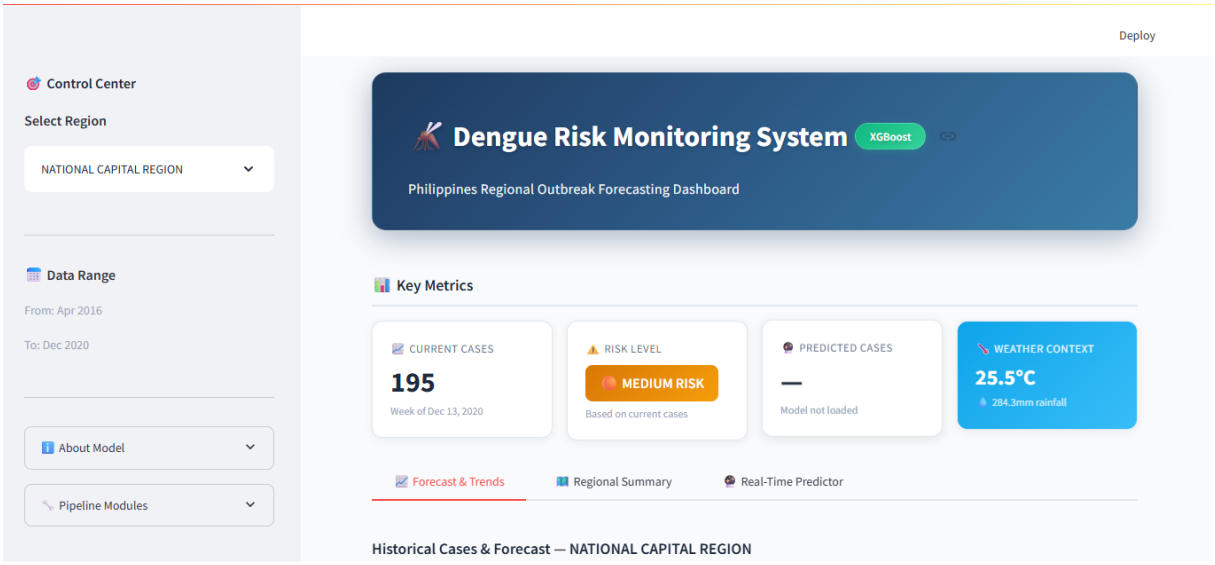

Figure 4: Dashboard Overview: Risk level indicator and key metrics

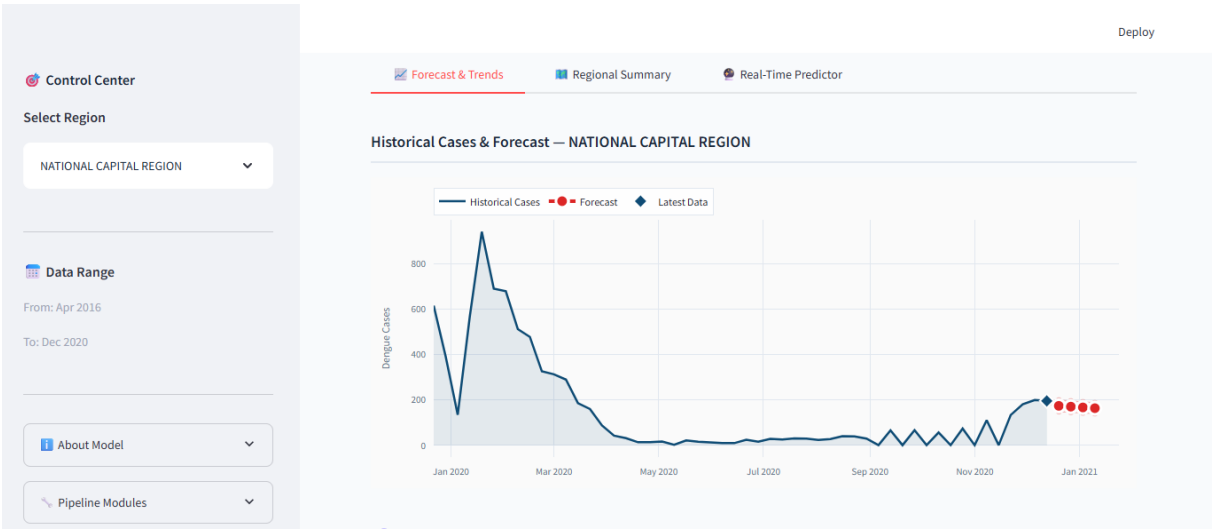

Figure 5: Dashboard Charts: Historical cases, forecasts, and weather patterns

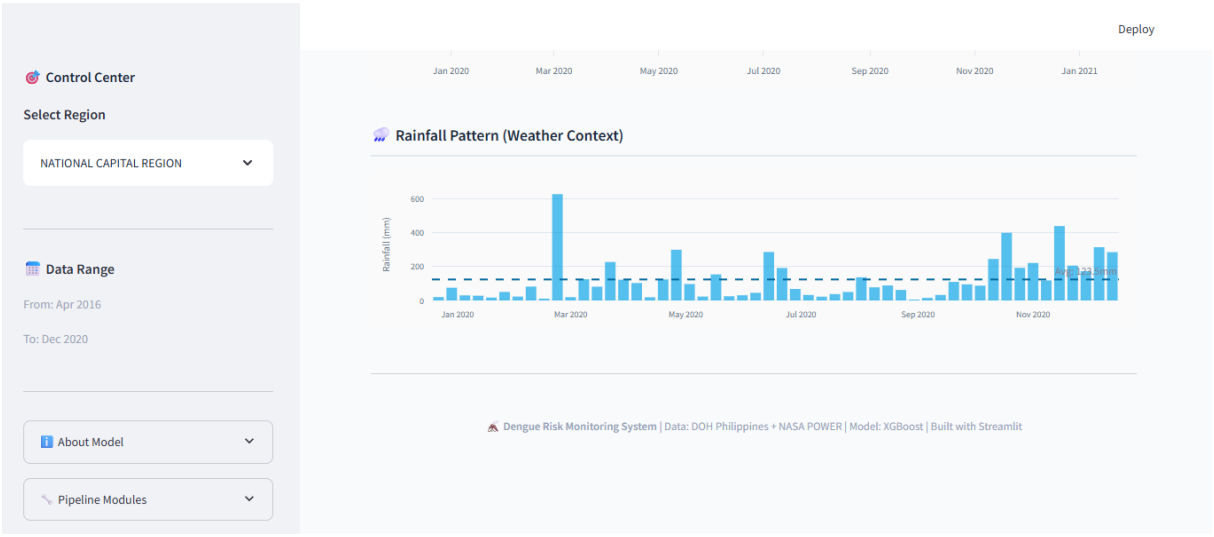

Figure 6: Dashboard Controls: Region selection and date range filters in sidebar

Figure 7: User Input Interface: Custom parameters for generating predictions

7.2 How It Works

Behind the scenes, the app:

1. Pulls recent weather data from NASA's POWER API
2. Converts daily data to weekly averages
3. Runs it through the same feature engineering pipeline we used for training
4. Feeds the features to our saved XGBoost model
5. Labels regions as HIGH/MEDIUM/LOW risk based on predicted case counts

You can run it locally with:

```
streamlit run app.py
```

It opens in a browser at `localhost:8501`. For a real deployment, you'd want to host it on Streamlit Cloud or a server.

8 Discussion

8.1 What We Learned

The main takeaway from this project is that predicting dengue outbreaks with machine learning is definitely possible. We got an R^2 of 0.79 for 2-week predictions, which is pretty solid for this kind of problem.

8.1.1 Feature Engineering Matters

The biggest lesson was how much feature engineering matters. Just feeding raw weather data into a model doesn't work well. But once we added lagged features (especially 4–8 week lags), performance jumped significantly. This makes biological sense—mosquitoes don't appear instantly when it rains; they need weeks to breed and mature.

8.1.2 Why Weather Data Helps

Adding weather features improved predictions compared to using case history alone. Weather contributed about 27% of the model's predictive power. This is useful because weather can be forecasted—potentially letting us predict outbreaks even further ahead.

8.1.3 Why XGBoost Won

XGBoost outperformed Random Forest and Ridge. We think this is because gradient boosting builds trees sequentially, with each tree correcting the mistakes of previous ones. For time series data like ours, this seems to work better than averaging many independent trees.

8.2 How We Compare to Other Studies

Our R^2 of 0.79 is actually quite good compared to published work:

- Lauer et al. (2018) got 0.5–0.7 for Puerto Rico
- Racloz et al. (2012) reported 0.6–0.8 for Southeast Asia

So we're in the upper range of what's been done before.

8.3 Practical Use

Two weeks of advance warning is enough time for health departments to:

- Pre-position medical supplies and hospital beds
- Deploy fogging and larvicide treatments
- Issue community advisories and prevention campaigns
- Coordinate inter-regional resource sharing

The dashboard interface makes these predictions accessible to non-technical users, bridging the gap between machine learning and public health practice.

9 Limitations and Future Directions

9.1 What Could Be Better

This project has some clear limitations that we should acknowledge.

Data is too coarse. We only have regional-level data. The Philippines has 17 regions, and each one is huge. Outbreaks in Manila might look completely different from those in Mindanao, but we're treating each region as a single unit. Ideally, we'd have city or even barangay-level data.

Case counts are probably undercounted. Not everyone who gets dengue goes to the hospital. Some people have mild symptoms and stay home. This means the DOH numbers don't capture the full picture.

NDVI didn't help much. We included satellite vegetation data hoping it would indicate mosquito habitats. But at regional scale, it didn't add much value. Higher-resolution imagery that could identify standing water or urban flooding might work better.

We tested on 2020 (COVID year). The pandemic changed everything—fewer people went to hospitals, surveillance systems were stretched thin. Our test results might not generalize perfectly to normal years.

9.2 Ideas for Future Work

If we had more time, here's what we'd try next:

Deep learning. LSTMs or Transformers might capture longer-range patterns better than XGBoost. They could also give us uncertainty estimates, which would be useful for risk communication.

Use weather forecasts. Right now, we only use observed weather. But weather can be predicted 1–2 weeks out. If we incorporated forecast data, we could potentially predict outbreaks even earlier.

Add spatial features. Neighboring regions probably influence each other. If cases spike in Region III, Region IV might be next. We didn't model these spatial dependencies but it could help.

Real deployment. For this to actually be useful, it would need to run automatically every week, pull fresh data from DOH, and send alerts when risk is high. That's a whole software engineering project beyond the ML model.

10 Conclusion

We built a machine learning system that can predict dengue outbreaks in the Philippines about two weeks in advance. It works by combining historical case data with weather measurements and satellite imagery.

Our best model (XGBoost) achieved an R^2 of 0.79, beating the naive baseline by about 27% in terms of MAE. The most important predictors turned out to be recent case counts—what happened last week strongly predicts what will happen next week. Weather features, especially rain and temperature from 4–8 weeks ago, added useful predictive power on top of that.

We also made a Streamlit dashboard where you can select a region and see the predicted risk level. It's a simple proof of concept but shows how these predictions could actually be used.

The project has limitations. We only have regional-level data, so we can't predict for specific cities or neighborhoods. The test year (2020) overlapped with COVID, which probably affected how many people visited hospitals. And we only predict case counts—we can't say anything about disease severity.

Still, this shows that data-driven outbreak prediction is feasible. With better data and continued development, systems like this could genuinely help public health officials get ahead of the next outbreak instead of just reacting to it.

References

[1] World Health Organization. (2023). *Dengue and severe dengue*. Retrieved from https://www.who.int/news-room/fact-sheets/detail/dengue-and-severe-dengue

[2] Morin, C. W., Comrie, A. C., & Ernst, K. (2013). Climate and dengue transmission: evidence and implications. *Environmental Health Perspectives*, 121(11-12), 1264–1272.

[3] NASA. (2023). *POWER – Prediction of Worldwide Energy Resources*. Retrieved from https://power.larc.nasa.gov/

[4] NASA. (2023). *MODIS – Moderate Resolution Imaging Spectroradiometer*. Retrieved from https://modis.gsfc.nasa.gov/

[5] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.

[6] Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

[7] Lauer, S. A., et al. (2018). Prospective forecasts of annual dengue hemorrhagic fever incidence in Thailand, 2010–2014. *Proceedings of the National Academy of Sciences*, 115(10), E2175–E2182.

[8] Racloz, V., et al. (2012). Surveillance of dengue fever virus: a review of epidemiological models and early warning systems. *PLoS Neglected Tropical Diseases*, 6(5), e1648.

[9] McKinney, W. (2010). Data structures for statistical computing in Python. *Proceedings of the 9th Python in Science Conference*, 56–61.

[10] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90–95.

A Project Resources

GitHub Repository:
https://github.com/SaqibMehdi123/Dengue-Outbreak-Prediction-System

Video Demo:
https://drive.google.com/drive/folders/17ykPJbeb1BjaAjtbAs9IqG5dn5UyjkVs

B Code Repository Structure

```
dengue_project/
|-- app.py                    # Streamlit dashboard
|-- config.py                 # Configuration settings
|-- pipeline.py               # Main orchestration script
|-- fetch_live_data.py        # NASA POWER API data fetcher
|-- requirements.txt          # Python dependencies
|
|-- src/                      # Core ML modules
|   |-- data_preparation.py      # Phase 1: Data loading & merging
|   |-- feature_engineering.py   # Phase 2: Feature creation
|   |-- model_training.py        # Phase 3: Model training
|   |-- evaluation.py            # Phase 4: Metrics & analysis
|   +-- predict.py               # Phase 5: Predictions
|
|-- data/                     # Data files
|   |-- philippines_dengue.csv
|   |-- weather/              # Weather CSVs by region
|   |-- philippines_dengue_dataset_FINAL.csv
|   +-- dengue_dataset_engineered.csv
|
|-- models/                   # Trained models
|   |-- xgboost_best.joblib
|   +-- random_forest.joblib
|
+-- notebooks/                # Jupyter notebooks
    +-- dengue_project.ipynb
```

Listing 2: Project Directory Structure