DengueWatch: A System for Real-Time Dengue Monitoring and Forecasting in Iloilo City

A Special Problem Proposal
Presented to
the Faculty of the Division of Physical Sciences and Mathematics
College of Arts and Sciences
University of the Philippines Visayas
Miag-ao, Iloilo

In Partial Fulfillment
of the Requirements for the Degree of
Bachelor of Science in Computer Science by

AMODIA, Kurt Matthew A.
BULAONG, Glen Andrew C.
ELIPAN, Carl Benedict L.

Francis D. DIMZON
Adviser

November 8, 2024

Abstract

In response to a marked rise in dengue cases, Iloilo City and Province are enhancing control measures. As of August 10, 2023, the Iloilo Provincial Health Office reported 4,585 cases and 10 fatalities, reflecting a 319% increase from last year's 1,095 cases and one death. This research includes the development of a centralized system for monitoring and forecasting dengue trends in the Iloilo region. This study explores the application of artificial intelligence (AI) for dengue prediction, using a deep learning approach with Long Short-Term Memory (LSTM) networks. The LSTM model is compared with traditional statistical methods, including non-seasonal and seasonal Autoregressive Integrated Moving Average (ARIMA) models and the Kalman Filter for state estimation algorithm in noisy data conditions. Forecasting was based on climate variables such as temperature, rainfall, relative humidity, and previous monthly case counts, with performance evaluated using Root Mean Square Error (RMSE). This research, aimed at supporting public health agencies like the Department of Health (DOH), advocates for AI-driven solutions that improve outbreak response beyond traditional reporting systems.

Keywords: ARIMA, artificial intelligence, dengue prediction, LSTM, Kalman Filter, deep learning, climate variables, public health, outbreak mitigation

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

From 2020 to 2022, dengue cases declined due to reduced surveillance during the COVID-19 pandemic, but cases surged in 2023 as restrictions were lifted. This year saw an increase in dengue outbreaks worldwide, with over five million cases and more than 5,000 deaths reported in over 80 countries. (Bosano, 2023) Dengue is endemic in the Philippines, leading to longer and more widespread seasonal outbreaks. Globally, dengue infections have increased significantly, posing a major public health challenge. The World Health Organization (Organization, 2024) reported a ten-fold rise in cases between 2000 and 2019, with a peak in 2019 when the disease spread across 129 countries.

Iloilo City and Province are intensifying efforts to curb the rising dengue cases. As of August 10, 2023, the Iloilo Provincial Health Office recorded 4,585 cases and 10 deaths, a 319% increase from last year's 1,095 cases and one death. Governor Arthur Defensor Jr. confirmed that the province has reached the dengue outbreak threshold based on Department of Health (DOH) criteria, and a formal declaration is pending. Local government units (LGUs) have been informed, and the province's disaster management office is on blue alert, indicating disaster mode. (Lena, 2024)

In Iloilo City, 649 dengue cases were recorded during the same period, with two deaths. Cases cluster in 40 out of 180 barangays, meaning multiple cases are being reported in these areas over several weeks. The city's health officer, Dr. Roland Jay Fortuna, reported high utilization of non-COVID-19 hospital beds, reaching over 76%, prompting concerns about hospital capacity.

This study explores the monitoring and forecasting of dengue outbreaks by analyzing key factors such as temperature, relative humidity, and historical dengue cases, using different models. The findings aim to provide an advanced, AI-driven alternative for dengue prevention and control, targeting agencies like the Department of Health (DOH). By aligning with the national AI Roadmap, particularly in Iloilo City, this research aspires to improve outbreak responses through cutting-edge technology rather than traditional reporting methods.

1.2 Problem Statement

The problem being addressed here is that dengue cases remain a critical public health issue worldwide, with rising cases attributed to the easing of COVID-19 restrictions and increased global mobility. From 2020 to 2022, dengue cases saw a temporary decline due to reduced surveillance efforts amidst the pandemic. However, 2023 witnessed a significant resurgence, with over five million cases and more than 5,000 deaths reported across 80 countries, indicating the continued vulnerability of dengue-endemic regions like the Philippines. In Iloilo City and Province, dengue cases surged dramatically by 319% as of August 2023, with local health systems struggling to manage the influx. High hospitalization rates due to dengue, with over 76% of non-COVID-19 hospital beds occupied, have raised concerns about healthcare capacity and the need for enhanced predictive measures.

1.3 Research Objectives

1.3.1 General Objective

This study aims to develop an AI-based dengue forecasting and monitoring system for Iloilo City and Province. The system will use Long Short-Term Memory (LSTM) to predict dengue case trends based on climate data and historical dengue cases to help public health officials in possible dengue case outbreaks.

1.3.2 Specific Objectives

Specifically, this study aims to develop a system that can:

1. Gather dengue data from the Iloilo Provincial Health Office and climate data from online sources. Combine these data into a unified dataset to facilitate comprehensive dengue case forecasting.

2. Develop and evaluate deep learning models, including LSTM, ARIMA, Seasonal ARIMA, and Kalman Filter, for predicting dengue cases. Compare the performance of these models to determine the most accurate forecasting approach.

3. Integrate the predictive model into a web-based data analytics dashboard. This dashboard will include features such as data visualizations and data entry, offering public health stakeholders an interactive tool for analyzing dengue trends and making informed decisions.

1.4 Scope and Limitations of the Research

This study aimed to develop an AI-based dengue forecasting and monitoring system specifically designed for Iloilo City. The system focuses on two major features: dengue case prediction and risk area identification. The dengue case prediction feature utilizes climate variables—such as temperature, rainfall, and relative humidity—along with historical dengue case data to forecast monthly dengue cases. The results will be displayed in a user-friendly interface, providing public health officials with actionable insights to enhance outbreak management and resource allocation. However, this study has several limitations. The accuracy of the dengue case predictions heavily relies on the quality and completeness of the input data. Inconsistent or incomplete historical data may lead to reduced prediction accuracy. Additionally, the model's performance may fluctuate based on variations in climate patterns, which are not always predictable. The model utilizes advanced machine learning techniques, but it cannot account for all factors influencing dengue transmissions, such as socio-economic conditions or public health interventions, which may further impact case dynamics. Finally, the dataset used for training the predictive models has not undergone peer review but has been validated by local public health experts to ensure its relevance and accuracy for the study's context. As a result, the findings should be interpreted with caution, and ongoing validation and adjustments may be necessary to enhance the model's robustness and applicability in real-world settings.

1.5 Significance of the Research

This study's development of an AI-based dengue forecasting and monitoring system has wide-reaching significance for various stakeholders in Iloilo City:

- Public Health Agencies: Organizations like the Department of Health (DOH) and local health units in Iloilo City and Province stand to benefit greatly from the system. With dengue predictions, we can help these agencies optimize their response strategies and implement targeted prevention measures in high-risk areas before cases escalate.

- Local Government Units (LGUs): LGUs can use the system to support their disaster management and health initiatives by proactively addressing dengue outbreaks. The predictive insights allow for more efficient planning and resource deployment in barangays and communities most vulnerable to outbreaks, improving overall public health outcomes.

- Healthcare Facilities: Hospitals and clinics, which currently face high bed occupancy rates during dengue season will benefit from early outbreak forecasts that can help in managing patient inflow and ensuring adequate hospital capacity.

- Researchers and Policymakers: This AI-driven approach contributes valuable insights for researchers studying infectious disease patterns and policymakers focused on strengthening the national AI Roadmap. The system's data can support broader initiatives for sustainable health infrastructure and inform policy decisions on resource allocation for dengue control.

- Community Members: By reducing the frequency and severity of outbreaks, this study ultimately benefits the community at large. This allows for timely awareness campaigns and community engagement initiatives, empowering residents with knowledge and preventative measures to protect themselves and reduce the spread of dengue.

Chapter 2

Review of Related Literature

2.1 Existing System: RabDash DC

RabDash, developed by the University of the Philippines Mindanao, is a web-based dashboard for rabies data analytics. It combines predictive modeling with genomic data, enabling local health authorities to optimize interventions and allocate resources more effectively. RabDash's modules include trend visualization, geographic hotspot mapping, and predictive forecasting, utilizing Long Short-Term Memory (LSTM) models for time-series forecasting (RabDashDC, 2024).

For DengueDash, RabDash serves as a strong inspiration, particularly in its monitoring, historical trend visualization, and forecasting capabilities. These features align well with the needs of dengue control efforts, providing real-time insights into outbreak trends and enabling more effective, data-driven decision-making. RabDash's architecture is relevant to the DengueDash, as dengue outbreaks similarly require time-series forecasting models. By using LSTM, RabDash effectively models trends in outbreak data, which provides a framework for adapting LSTM to dengue forecasting. Research indicates that LSTM models outperform traditional methods, such as ARIMA and MLP, in handling the complexities of time-dependent epidemiological data (Ligue & Ligue, 2022).

2.2 Deep Learning

The study of Kim Dianne Ligue and Kristine Joy Ligue highlights how data-driven models can help predict dengue outbreaks. The authors compared traditional statistical methods, such as non-seasonal and seasonal autoregressive integrated moving average (ARIMA), and traditional feed-forward network approach using a multilayer perceptron (MLP) model with a deep learning approach using the long short-term memory (LSTM) architecture in their prediction model. They find that the LSTM model performs better in terms of accuracy. The LSTM model achieved a much lower root mean square error (RMSE) compared to both MLP and ARIMA models, proving its ability to capture complex patterns in time-series data (Ligue & Ligue, 2022). This superior performance is attributed to LSTM's capacity to capture complex, time-dependent relationships within the data, such as those between temperature, rainfall, humidity, and mosquito populations, all of which contribute to dengue incidence (Ligue & Ligue, 2022).

2.3 Kalman Filter

The Kalman Filter is another powerful tool for time-series forecasting that can be integrated into our analysis. It provides a recursive solution to estimating the state of a linear dynamic system from a series of noisy measurements. Its application in epidemiological modeling can enhance prediction accuracy by accounting for uncertainties in the data(Li et al., 2022). Studies have shown that Kalman filters are effective in predicting infectious disease outbreaks by refining estimates based on observed data. A study published in Frontiers in Physics utilized the Kalman filter to predict COVID-19 deaths in Ceará, Brazil. They found that the Kalman filter effectively tracked the progression of deaths and cases, providing critical insights for public health decision-making (Ahmadini et al., 2021). Another research article in PLOS ONE focused on tracking the effective reproduction number (R_t) of COVID-19 using a Kalman filter. This method estimated the growth rate of new infections from noisy data, demonstrating that the Kalman filter could maintain accurate estimates even when case reporting was inconsistent(Arroyo-Marioli, Bullano, Kucinskas, & Rondón-Moreno, 2021).

Our study will compare ARIMA, seasonal ARIMA, Kalman Filter, and LSTM models using our own collected dengue case data along with weather data to identify the most effective model for real-time forecasting.

2.4 Weather Data

The relationship between weather patterns and mosquito-borne diseases is inherently nonlinear, meaning that fluctuations in disease cases do not respond proportionally to changes in climate variables(Colón-González, Fezzi, Lake, & Hunter, 2013) Weather data, such as minimum temperature and accumulated rainfall, are strongly linked to dengue case fluctuations, with effects observed after several weeks due to mosquito breeding and virus incubation cycles. Integrating these lagged weather effects into predictive models can improve early warning systems for dengue control(Cheong, Burkart, Leitão, & Lakes, 2013). A study also suggests that weather-based forecasting models using variables like mean temperature and cumulative rainfall can provide early warnings of dengue outbreaks with high sensitivity and specificity, enabling predictions up to 16 weeks in advance(Hii, Zhu, Ng, Ng, & Rocklöv, 2012).

We will utilize weather data, including variables such as temperature, rainfall, and humidity, as inputs for our dengue forecasting model. Given the strong, non-linear relationship between climate patterns and dengue incidence, these weather variables, along with their lagged effects, are essential for enhancing prediction accuracy and providing timely early warnings for dengue outbreaks.

2.5 Chapter Summary

This chapter reviewed key literature relevant to our study, focusing on existing systems, predictive modeling techniques and the role of weather data in forecasting dengue outbreaks. We examined systems like RabDash DC, which integrates predictive modeling with real-time data to inform public health decisions, providing a foundational structure for our Dengue Watch System. Additionally, deep learning approaches, particularly Long Short-Term Memory (LSTM) networks, were highlighted for their effectiveness in time-series forecasting, while alternative methods such as ARIMA and Kalman Filters were considered for their ability to model complex temporal patterns and handle noisy data.

The literature further underscores the significance of weather variables—such as temperature and rainfall—in forecasting dengue cases. Studies demonstrate that these variables contribute to accurate outbreak prediction models. Leveraging these insights, our study will incorporate both weather data and historical dengue case counts to build a reliable forecasting model.

Chapter 3

Research Methodology

This chapter lists and discusses the specific steps and activities that will be performed to accomplish the project. The discussion covers the activities from pre-proposal to Final SP Writing.

3.1 Research Activities

3.1.1 System Development Framework

The Agile Model is the birthchild of both iterative and incremental approaches in Software Engineering. It aims to be flexible and effective at the same time by being adaptable to change. It's also important to note that small teams looking to construct and develop projects quickly can benefit from this kind of methodology. As the Agile Method focuses on continuous testing, quality assurance is a guarantee since bugs and errors are quickly identified and patched.

3.1.2 Design, Building, Testing, and Integration

Design and Developlment

After brainstorming and researching the most appropriate type of application to accommodate both the prospected users and the proposed solutions, the team has decided to proceed with a web application. Given the time constraints and available resources, we believe this is the most pragmatic and practical move. The next step is to select modern and stable frameworks that align with the fundamental ideas we have learned at the university. The template obtained from WVCHD was meticulously analyzed to create use cases and develop a preliminary well-structured database that adheres to the requirements needed to produce a quality application. The said use cases serve as the basis of general features. Part by part, these are converted into code, and with the help of selected libraries and packages, it resulted in the desired outcome that may still modified and extended since it is continuously being developed.

Testing and Integration

Each feature is rigorously user-tested to ensure quality assurance, with particular emphasis on prerequisite features, as development cannot progress properly if these fail. Moreover, integration between each feature serves as a pillar for a cohesive user experience. Presently, we have not been able to use performance metrics to measure the system's performance, as developing and connecting the core features is the utmost priority.

3.2 Model and Algorithm

Data Preprocessing and Feature Selection:

- Clean and preprocess the dataset, handling missing values and outliers.
- Select important features such as temperature, rainfall, and humidity.

Model Selection and Justification:

- Train an LSTM model due to its efficiency in handling long-term data relationships.
- Compare performance with ARIMA, SARIMA, and Kalman Filter.

Seasonal ARIMA (SARIMA):

- Set seasonal parameters based on observed seasonality in cases.

- Apply SARIMA to capture short-term and seasonal trends.

Kalman Filter:

- Use Kalman Filter for real-time updates and handle missing values dynamically.

LSTM (Long Short-Term Memory) Neural Network:

- Build a sequential LSTM model with tuned hyperparameters.

Model Evaluation:

- Evaluate performance with Mean Absolute Error (MAE), RMSE, and R-squared metrics.

3.3 Development Tools

3.3.1 Software

Github

GitHub is a cloud-based platform that tracks file changes using Git, an open-source version control system (*About GitHub and Git - GitHub Docs*, n.d.). In an article by Sawers (2023), it is the most popular collaboration software for software development with 100 million users and hosting over 420 million repositories as of January 2023. Due to its dependability, GitHub is used in the project to store the application's source code, manage the system's source version control, and serve as a repository for the Latex files used in the actual research.

Visual Studio Code

Visual Studio Code is a free, lightweight, and cross-platform source code editor developed by Microsoft (*Why Visual Studio Code?*, 2021). While it is not a full-fledged traditional Integrated Development Environment (IDE), it still supports features such as syntax highlighting, code completion, and debugging. With the vast array of extensions available through its store, the said code editor can be an IDE-like environment experience. As VS Code supports this project's programming and scripting languages, it was chosen as the primary source code editor.

Django

Django is a free and open-sourced Python-based web framework that offers an abstraction to develop and maintain a secure web application. Because of this, developers can focus on coding a more straightforward solution without minding the more intricate hassles of web development (*Django introduction - Learn web development — MDN*, 2024). In addition, Django offers well-defined documentation and a large community.

As this research aims to create a well-developed and maintainable application, it is in the best interest to follow an architectural pattern that developers and contributors in the future can understand. Since Django adheres to Model-View-Template (MVT) that promotes a clean codebase by separating data models, business logic, and presentation layers, it became the primary candidate for the application's backbone.

Next.js

A report by Statista (2024) claims that React is the most popular front-end framework among web developers. However, React has limitations that can be a nuisance in rapid software development, which includes routing and performance optimizations. This is where Next.js comes in—a framework built on top of React. It offers solutions for React's deficiency, making it a rising star in the framework race. Moreover, it also offers a more enhanced developer experience through hot module replacement, Typescript support, and a rich plugin ecosystem, which facilitates efficient development workflows and reduces configuration overhead.

Postman

As the application heavily relies on the Application Programming Interface (API) being thrown by the backend, it is a must to use a development tool that facilitates the development and testing of the API. Postman is a freemium API platform that offers a user-friendly interface to create and manage API requests (*What is*

Postman? Postman API Platform, n.d.).

3.3.2 Hardware

The web application is continuously being developed on laptop computers with minimum specifications of an 11th-generation Intel i5 CPU and 16 gigabytes of RAM.

3.3.3 Packages

Django REST Framework

Django Rest Framework (DRF) is a third-party package for Django that provides a comprehensive suite of features to simplify the development of robust and scalable Web APIs. These services include Serialization, Authentication and Permissions, Viewsets and Routers, and a Browsable API (Christie, n.d.).

Since Django utilizes models that bundle data together, DRF serves as a bridge to convert these complex datatypes into primitive Python datatypes, which can be easily rendered into JavaScript Object Notation (JSON), Extensible Markup Language (XML), or other content types. On the other hand, it also includes a variety of authentication policies that control access to API endpoints. These include Basic, Session, and Token Authentication. It is also worth noting that a custom permission class can also be created as needed. DRF also simplifies the process of creating RESTful APIs by combining the logic for a group of related views into a single class, which can be accessed through a URL route. Lastly, the implementation of a web-browsable API is a game-changer since it allows developers to interact with the API through a user-friendly interface that becomes a breeding ground for testing and exploration.

Leaflet

One of the features of the web application is the ability to map the number of cases using a Choropleth Map. Leaflet is the only free, open-sourced, and most importantly, stable JavaScript package that can do the job. With its ultra-lightweight size, it offers a comprehensive set of features that does not trade off performance and usability (*Leaflet — an open-source JavaScript library for* *interactive maps*, n.d.). Open-source contributors have also created extensions of the package itself to support multiple frameworks. React-Leaflet is currently being supported and improved to satisfy the changes with the most up-to-date version of React (*Introduction — React Leaflet*, n.d.).

Chart.js

Another feature of the application is to provide users with informative, approachable data storytelling that is easy for everyone to understand. The transformation of pure data points and statistics into figures such as charts is a big factor. Thus, there is a need for a package that can handle this feature without compromising the performance of the application. Chart.js is a free and open-source JavaScript package that is made to meet this criteria. It supports various types of charts, including bar, line, pie, scatter, and more. Since the package utilizes HTML5's canvas element, it is guaranteed to work across all modern browsers (*Chart.js*, n.d.). It is also worth noting that due to its simplicity and flexibility, the charts generated are highly customizable bringing data visualization to new heights.

Tailwind CSS

Using plain CSS in production-quality applications can be counterproductive. Therefore, CSS frameworks were developed to promote consistency and accelerate the rapid development of web applications (Joel, 2021). One of these is Tailwind, which offers low-level utility classes that can be applied directly to each HTML element to create a custom design (*tailwind-no-date*, n.d.). Given the limited timeline for this project, using this framework is a wise choice due to its stability and popularity among developers.

Shadcn

Shadcn offers a collection of open-source UI boilerplate components that can be directly copied and pasted into one's project. With the flexibility of the provided components, Shadcn allows developers to have full control over customization and styling. Since this is built on top of Tailwind CSS and Radix UI, it is supported by most modern frontend frameworks, including Next.js (Shadcn, n.d.).

3.4 Calendar of Activities

A Gantt chart showing the schedule of the activities is included below. Each bullet represents approximately one week of activity.

Table 3.1: Timetable of Activities

Activities (2024)	Aug	Sept	Oct	Nov	Dec
Project Initiation and Team Formation	●●				
Literature Review and Data Gathering	●●	●●●●			
Data Cleaning and Feature Selection		●●		●	
Creating System Dashboard		●●	●●●●	●	
Analysis and Interpretation of Results			●		
Documentation		●	●●●●	●	

Chapter 4

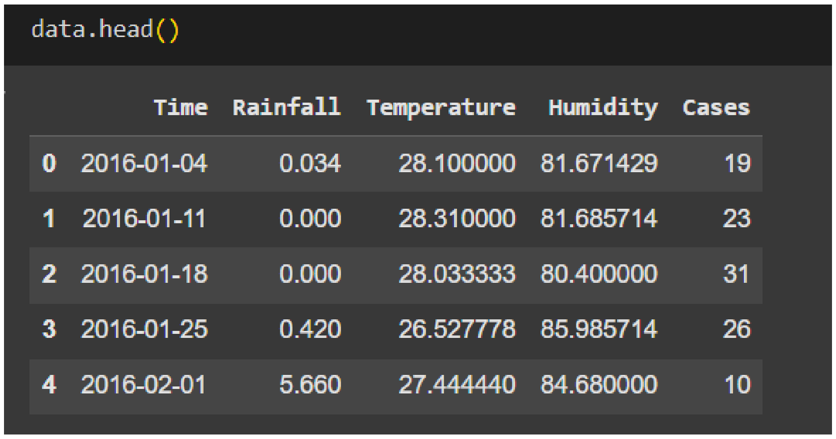
Preliminary Results/System Prototype

4.1 Data Gathering

The data for dengue case prediction was gathered from a variety of reliable sources, enabling a comprehensive dataset spanning from January 2016 to September 2024. This dataset includes 452 rows of data, each containing weekly records of dengue cases along with corresponding meteorological variables, such as rainfall, temperature, and humidity.

1. Dengue Case Data: The primary source of historical dengue cases came from the Humanitarian Data Exchange and the Western Visayas Center for Health Development (WVCHD). The dataset, accessed through Freedom of Information (FOI) requests, provided robust case numbers for the Western Visayas region. The systematic collection of these data points was essential for establishing a reliable baseline for model training and evaluation.

2. Weather Data: Weekly weather data was obtained by web scraping from Weather Underground, allowing access to rainfall, temperature, and humidity levels that correlate with dengue prevalence.



```
data.head()

         Time  Rainfall  Temperature   Humidity  Cases
0  2016-01-04     0.034    28.100000  81.671429     19
1  2016-01-11     0.000    28.310000  81.685714     23
2  2016-01-18     0.000    28.033333  80.400000     31
3  2016-01-25     0.420    26.527778  85.985714     26
4  2016-02-01     5.660    27.444440  84.680000     10
```

Figure 4.1: Snippet of the Combined Dataset

4.2 Preliminary Model Training

The proposed Dengue Watch system utilized four distinct models to forecast weekly dengue cases: Long Short-Term Memory (LSTM) networks, Autoregressive Integrated Moving Average (ARIMA), Seasonal ARIMA (SARIMA), and Kalman Filter. Each model was trained on a dataset containing 452 weeks of historical dengue cases from 2016 to 2024, with meteorological variables such as temperature, humidity, and rainfall.

To optimize predictive performance, hyperparameter tuning was conducted individually for each model, refining parameters to achieve the most accurate and reliable forecasts. Following training, the models were rigorously evaluated against the dataset using a set of key performance metrics, including Mean Squared Error (MSE) and Root Mean Squared Error (RMSE).

The table below provides a summary and comparative analysis of each model's results across these metrics, offering insights into the strengths and limitations of each forecasting technique for dengue case prediction in Iloilo City.

Model	MSE	RMSE
LSTM	**342.59**	**18.51**
Seasonal ARIMA (2, 0, 2) (0, 1,1)	**1198**	**34.62**
ARIMA (2, 0, 3)	**1983.16**	**44.53**
Kalman Filter	**2755.77**	**52.49**

Table 4.1: Comparison of Models

4.2.1 LSTM Model

The LSTM model architecture consisted of an input layer, a single LSTM layer with 64 units and ReLU activation, followed by a dense layer with a single output neuron to predict the dengue case count. Key hyperparameters included:

- Window Size: 10 weeks, representing the time steps used in the sequence data for each prediction.
- Epochs: 50 epochs were used for training, balancing sufficient training time with computational efficiency.
- Batch Size: 1, allowing the model to process one sequence at a time, which is beneficial for small datasets but increases training time.
- Optimizer: The Adam optimizer was chosen for its adaptive learning capabilities and stability in training. A custom learning rate of 0.0001 was set to ensure gradual convergence and minimize risk of overfitting.

Data Division for Training and Testing The dataset was split into training and test sets to evaluate the model's performance and generalizability:

- Training Set: 85% of the data (385 sequences) was used for model training, enabling the LSTM to learn underlying patterns in historical dengue case trends and their relationship with weather variables.
- Test Set: The remaining 15% of the data (67 sequences) was reserved for testing

Upon testing, the LSTM model produced an MSE of 342.59 and an RMSE of 18.51, showing its capability to generalize to unseen data. The close alignment of actual and predicted values in the test set plot supports this observation, suggesting that the model successfully captured critical temporal trends that correlate with dengue outbreaks.

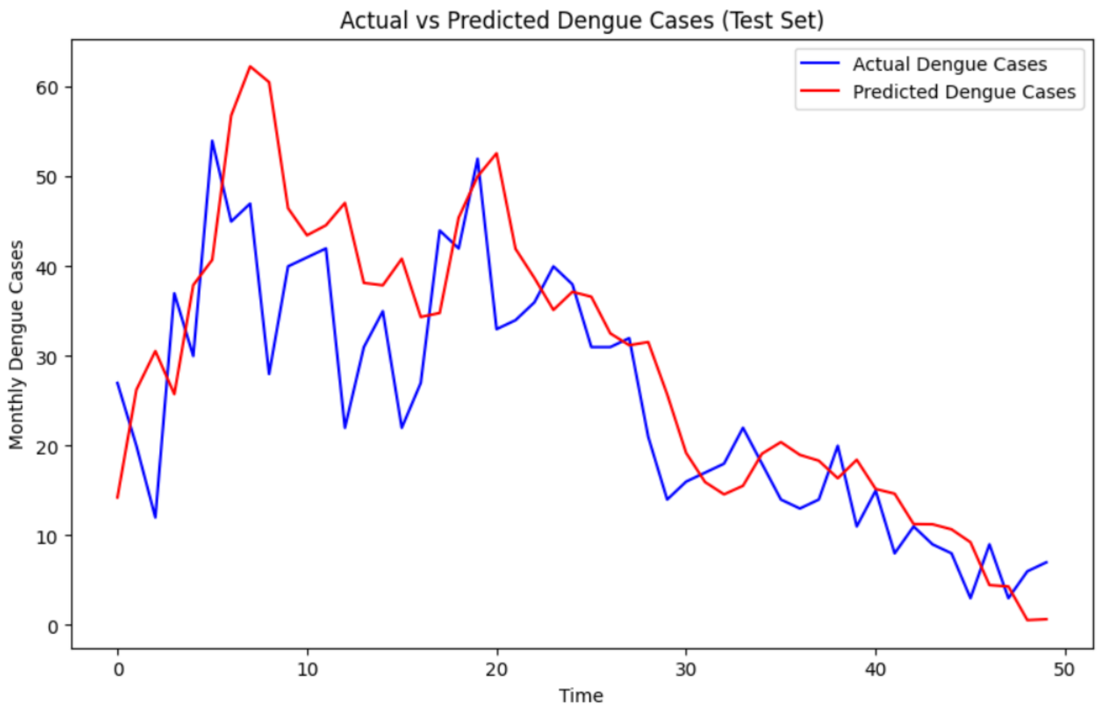


Figure 4.2: LSTM Prediction Results for Test Set

Training and Testing Data Division for ARIMA and Seasonal Arima

Both models utilized an **80%-20% split** to evaluate generalizability:

- **Training Set**: 80% of the data was used for training, allowing the models to learn underlying patterns in the dataset.
- **Test Set**: 20% of the data was reserved for testing, providing an unbiased assessment of the models' performance on unseen data.

4.2.2 ARIMA Model

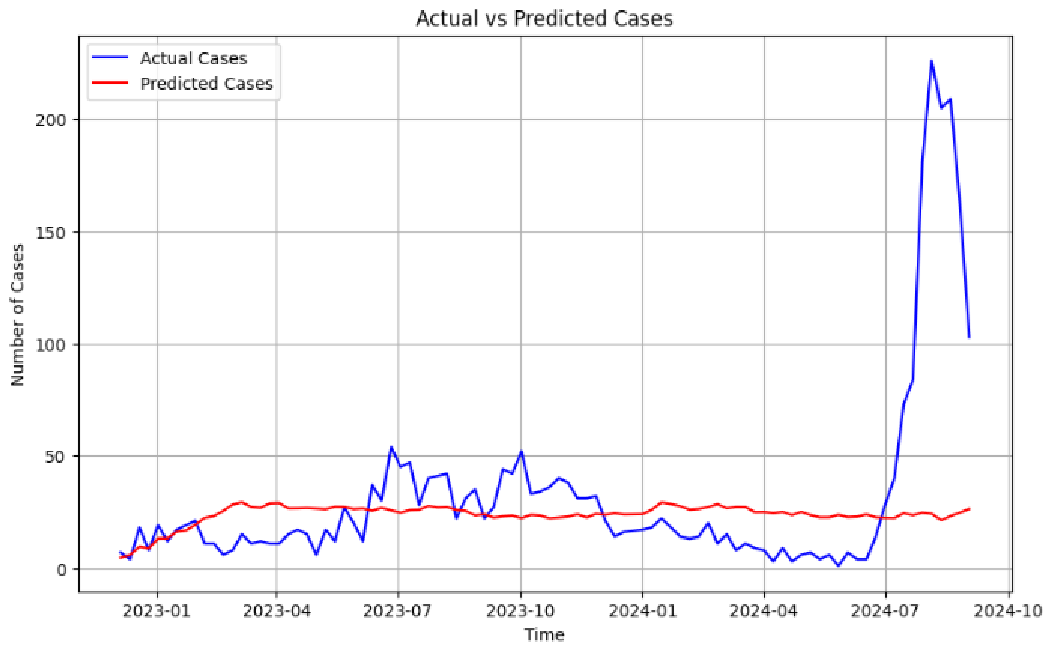


Figure 4.3: ARIMA Prediction Results for Test Set

The ARIMA model was developed to capture non-seasonal trends in the data. To determine the best model configuration, grid search was used to explore various combinations of ARIMA parameters, ultimately selecting **ARIMA(2, 0, 3)**. The model was iteratively refined over **400 iterations** to ensure convergence to an optimal solution. Key details are as follows:

1. **Data Preprocessing:** Prepare the dataset by handling any missing values and scaling the data if necessary to improve model convergence and stability.
2. **Hyperparameter Tuning:** Use a grid search on potential ARIMA parameters (p, d, q) to identify the configuration that minimizes error. The optimal parameters were found to be **(2, 0, 3)**.
3. **Model Training:**
    - Set the number of iterations to 400 to ensure thorough training and convergence.
    - Train the ARIMA model on 80% of the data and reserve 20% for testing.

4. **Evaluation:** After training, the ARIMA model was evaluated on the test data, yielding the following performance metrics:

   - **MSE (Mean Squared Error)**: 1983.16
   - **RMSE (Root Mean Squared Error)**: 44.53

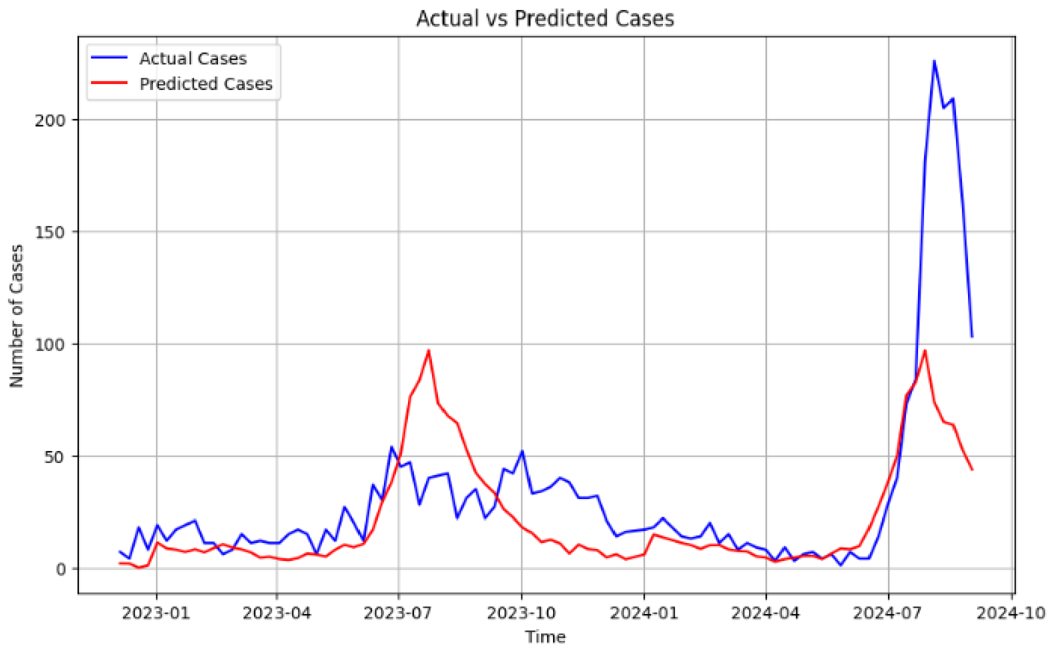


Figure 4.4: Seasonal ARIMA Prediction Results for Test Set

4.2.3 Seasonal ARIMA Model

(a) **Data Preprocessing:** Prepare the dataset by handling any missing values and scaling the data if necessary to improve model convergence and stability.

(b) **Hyperparameter Tuning:** Use a grid search on potential ARIMA parameters (p, d, q) to identify the configuration that minimizes error. The optimal parameters were found to be **(2, 0, 3)**.

(c) **Model Training:**

   - Set the number of iterations to 400 to ensure thorough training and convergence.
   - Train the ARIMA model on 80% of the data and reserve 20% for testing.

(d) **Evaluation:** After training, the ARIMA model was evaluated on the test data, yielding the following performance metrics:

- **MSE (Mean Squared Error)**: 1983.16
- **RMSE (Root Mean Squared Error)**: 44.53

These metrics demonstrate that the ARIMA model effectively captured non-seasonal temporal patterns in the data.

Seasonal ARIMA (SARIMA) Model

This model incorporates seasonal parameters, which were tuned using grid search to find the best configuration: **SARIMA(2, 0, 2)(0, 1, 1)[52]**. As with ARIMA, **400 iterations** were applied to ensure a robust fit.

Steps to Create the SARIMA Model:

(a) **Data Preprocessing:** Ensure data readiness by filling any missing values and scaling as needed.

(b) **Seasonality Analysis:** Examine the dataset for seasonal patterns. A periodicity of **52 weeks** was identified, making SARIMA a suitable choice for capturing yearly seasonality.

(c) **Hyperparameter Tuning:** Conduct grid search to identify the best set of parameters $(p, d, q)(P, D, Q)[S]$, where:

- **(p, d, q)** are the non-seasonal parameters,
- **(P, D, Q)** are the seasonal parameters, and
- **S** is the season length.

The optimal configuration found was **(2, 0, 2)(0, 1, 1)[52]**.

(d) **Model Training:**

- Set the iteration count to 400 for enhanced model robustness.
- Train the model on the 80% training dataset and reserve the remaining 20% for testing.

(e) **Evaluation:** The SARIMA model yielded the following error metrics:

- **MSE**: 1198
- **RMSE**: 34.62

The SARIMA model outperformed the ARIMA model in terms of lower MSE and RMSE values, indicating its effectiveness in capturing the seasonal patterns in the data.

4.2.4 Kalman Filter Model

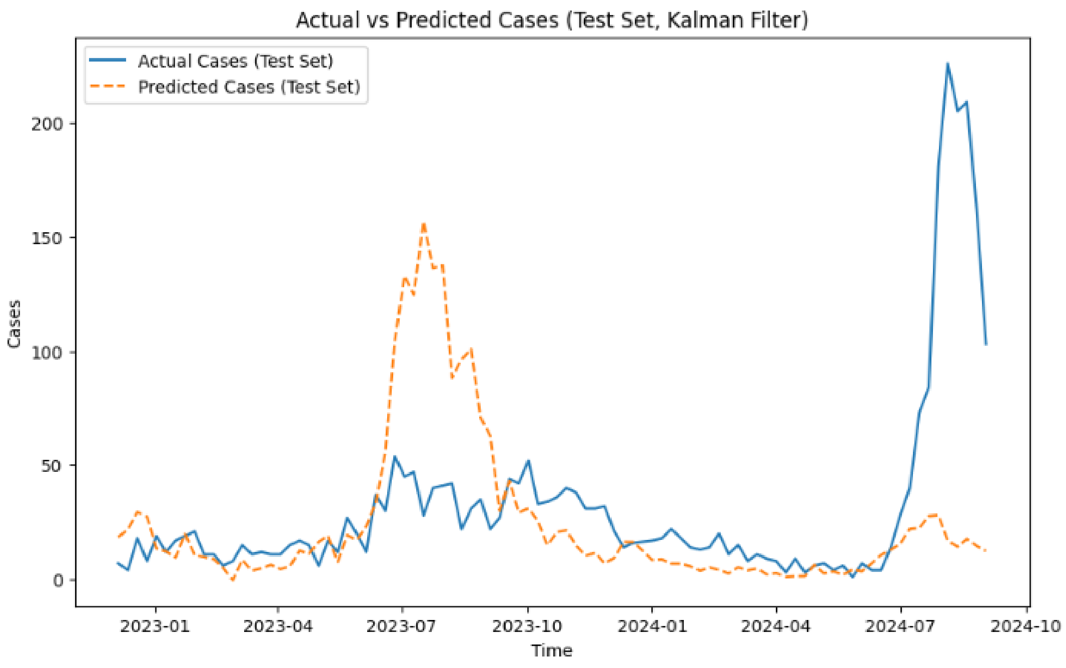


Figure 4.5: Kalman Filter Prediction Results for Test Set

Kalman Filter Methodology with Matrix Calculations

Measurement Acquisition: Obtain the measurement z_k of the system's state with associated confidence. This measurement matrix provides a noisy observation of the true state.

The dataset was split into training and test sets to evaluate the Kalman Filter's performance and generalizability:

- **Training Set**: 80% of the data was used for training, enabling the Kalman Filter model to capture key patterns.
- **Test Set**: The remaining 20% of the data was reserved for testing.

Prediction Step:

- Predict the next state:

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_k$$

where A is the state transition matrix and B is the control matrix.

- Update the state covariance:

$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q$$

where Q is the process noise covariance matrix.

Compute Residual: Calculate the residual

$$y_k = z_k - H\hat{x}_{k|k-1}$$

where H is the observation matrix. This residual represents the new information from the measurement.

Scaling Factor (Kalman Gain):

- Compute the Kalman Gain:

$$K_k = P_{k|k-1}H^T(HP_{k|k-1}H^T + R)^{-1}$$

where R is the measurement noise covariance matrix.

- The Kalman Gain determines the weight of the measurement relative to the prediction.

State Update:

- Update the state estimate:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k y_k$$

blending the prediction and measurement.

- This gives a refined state estimate that considers both the prediction and measurement based on their relative certainties.

Uncertainty Update:

- Update the state covariance:

$$P_{k|k} = (I - K_k H)P_{k|k-1}$$

where I is the identity matrix.

- This updated uncertainty affects the Kalman Gain in the next iteration, reflecting increased confidence in the state estimate after the measurement.

Model Evaluation:

Upon testing, the Kalman Filter produced a Mean Squared Error (MSE) of 2755.77 and a Root Mean Squared Error (RMSE) of 52.49.

References

(n.d.).

About GitHub and Git - GitHub Docs. (n.d.). Retrieved from `https://docs.github.com/en/get-started/start-your-journey/about-github-and-git`

Ahmadini, A. A. H., Naeem, M., Aamir, M., Dewan, R., Alshqaq, S. S. A., & Mashwani, W. K. (2021). Analysis and forecast of the number of deaths, recovered cases, and confirmed cases from covid-19 for the top four affected countries using kalman filter. *Frontiers in Physics*, *9*, 629320.

Arroyo-Marioli, F., Bullano, F., Kucinskas, S., & Rondón-Moreno, C. (2021). Tracking r of covid-19: A new real-time estimation using the kalman filter. *PloS one*, *16*(1), e0244474.

Bosano, R. (2023). *Who: Ph most affected by dengue in western pacific.* Retrieved Use the date of access, from `https://news.abs-cbn.com/spotlight/12/22/23/who-ph-most-affected-by-dengue-in-western-pacific`

Chart.js. (n.d.). Retrieved from `https://www.chartjs.org/`

Cheong, Y. L., Burkart, K., Leitão, P. J., & Lakes, T. (2013). Assessing weather effects on dengue disease in malaysia. *International journal of environmental research and public health*, *10*(12), 6319–6334.

Christie, T. (n.d.). *Home - Django REST framework.* Retrieved from `https://www.django-rest-framework.org/`

Colón-González, F. J., Fezzi, C., Lake, I. R., & Hunter, P. R. (2013). The effects of weather and climate change on dengue. *PLoS neglected tropical diseases*, *7*(11), e2503.

Django introduction - Learn web development — MDN. (2024, 11). Retrieved from `https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction`

Hii, Y. L., Zhu, H., Ng, N., Ng, L. C., & Rocklöv, J. (2012). Forecast of dengue incidence using temperature and rainfall. *PLoS neglected tropical diseases*, *6*(11), e1908.

Introduction — React Leaflet. (n.d.). Retrieved from https://react-leaflet.js.org/docs/start-introduction/

Joel, C. (2021, 10). *6 reasons to use Tailwind over traditional CSS.* Retrieved from https://dev.to/charliejoel/6-reasons-to-use-tailwind-over-traditional-css-1nc3

Leaflet — an open-source JavaScript library for interactive maps. (n.d.). Retrieved from https://leafletjs.com/

Lena, P. (2024). *Iloilo beefs up efforts amid hike in dengue cases.* Retrieved Use the date of access, from https://www.pna.gov.ph/articles/1231208

Li, X., Feng, S., Hou, N., Li, H., Zhang, S., Jian, Z., & Zi, Q. (2022). Applications of kalman filtering in time series prediction. In *International conference on intelligent robotics and applications* (pp. 520–531).

Ligue, K. D. B., & Ligue, K. J. B. (2022). Deep learning approach to forecasting dengue cases in davao city using long short-term memory (lstm). *Philippine Journal of Science*, *151*(3).

Organization, W. H. (2024). *Dengue and severe dengue.* Retrieved Use the date of access, from https://www.who.int/news-room/fact-sheets/detail/dengue-and-severe-dengue

RabDashDC. (2024). *Rabdash dc.* Retrieved Use the date of access, from https://rabdash.com

Shadcn. (n.d.). *Introduction.* Retrieved from https://ui.shadcn.com/docs

What is Postman? Postman API Platform. (n.d.). Retrieved from https://www.postman.com/product/what-is-postman/

Why Visual Studio Code? (2021, 11). Retrieved from https://code.visualstudio.com/docs/editor/whyvscode

Appendix A

Appendix Title

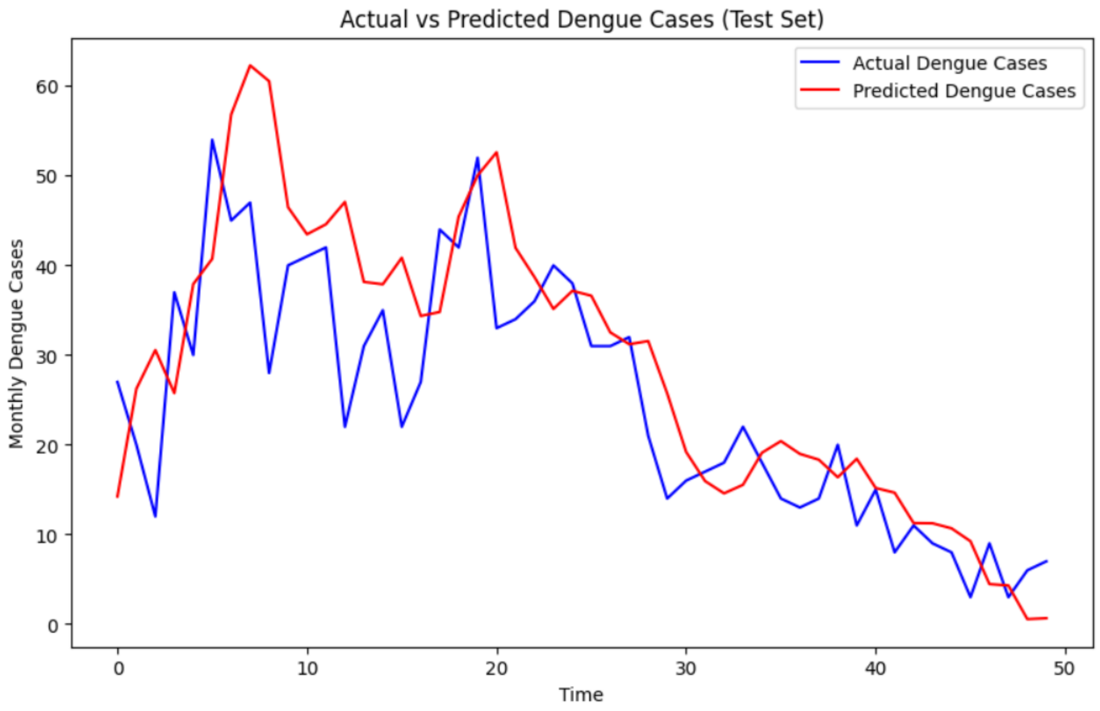


Figure A.1: LSTM Prediction Results for Test Set

Appendix B

Resource Persons

Mr. Firstname1 Lastname1
Role1
Affiliation1
`emailaddr1@domain.com`

Ms. Firstname2 Lastname2
Role2
Affiliation2
`emailaddr2@domain.net`

....