Machine Learning Project Documentation

Project Title: Dengue Fever Weekly Risk Alert System

Team: Group 8

Member:

1. Sue Sha Htunn (sueshahtunn2002@gmail.com)
2. Mya Moe Wai (myamoewai2002@gmail.com)
3. Shwe Sin Phoo (shwesinphoo2431@gmail.com)
4. Kyaw Soe Lwin (klwin7@my.smccd.edu)
5. Phyo Zay Yar Kyaw (kophyozayarkyaw@gmail.com)

1. Motivation and Context

Dengue fever continues to be one of the most serious and recurring public health challenges in Thailand. With strong seasonal behavior, climate sensitivity, and large regional variations, weekly risk prediction has high value for hospitals, local authorities, and community health systems. Early warning enables faster resource planning, outbreak mitigation, and targeted awareness campaigns.

This project aims to build a machine learning system capable of predicting weekly dengue risk levels using weather signals, epidemiological trends, and population search behavior. The system must generalize across provinces and seasons and support future deployment as an automated alert tool.

2. Data Pipeline Summary

A complete end-to-end data pipeline was designed to support this prediction system. It consists of eight major components developed in separate but connected scripts:

1. ERA5 climate data collection for Thailand from 2020 to 2024
2. Provincial weather aggregation from gridded climate outputs

3. Google Trends collection for dengue-related keywords
4. Dengue case collection or simulation consistent with MOPH formats
5. Data cleaning, imputation, merging, and temporal alignment
6. Feature engineering including lagged climate, rolling weather indicators, epidemiological lags, seasonality encodings, and search-behavior features
7. Model training with advanced tuning for multiple algorithms
8. Model evaluation, interpretability, and reporting tools

This pipeline ensures high-quality, weekly-level, multi-source data for robust machine learning training.

3. Current Progress and Challenges

At the time of this report, the most computationally intensive stage is under way: **downloading, processing, and aggregating ERA5 climate data for 2020–2024**.

Because ERA5 provides hourly reanalysis data, it must be downloaded month by month. For five years, this results in 60 separate files and several gigabytes of raw input. After collection, each file is processed into weekly provincial weather summaries. This step takes time but is essential because dengue is highly sensitive to temperature, humidity, rainfall, and pressure.

Other components of the pipeline, such as Google Trends collection, dengue case generation, cleaning, and feature engineering, are already fully implemented. Model training and evaluation pipelines are complete and ready to run once all weather records are prepared.

4. System Architecture Overview

The dengue forecasting system follows a modular architecture:

1. **Data Ingestion**
   Climate data, search trends, and dengue case records are collected from external sources and stored in raw form.

2. **Preprocessing and Cleaning**
   Each dataset is standardized by handling missing values, aligning dates, harmonizing province names, removing anomalies, and validating time continuity.

3. **Feature Engineering**
   Many additional signals are generated, including lagged weather indicators, rolling climate windows, seasonal encodings, mosquito-breeding conditions, and autoregressive case patterns.

4. **Model Training and Refinement**
   Multiple machine learning algorithms are evaluated using time-based splitting. Refinement uses class balancing, hyperparameter tuning, and ensemble models.

5. **Testing and Deployment Preparation**
   Trained models are applied to future weeks. Metrics, interpretability plots, and model files are exported for future use.

This design ensures clarity, reusability, and scalability for future expansions.

Model Refinement

1. Overview

The model refinement phase focused on improving the predictive performance, stability, and generalization ability of the dengue fever risk alert system. After establishing the initial baseline models, refinement aimed to address weaknesses identified during earlier evaluations. This phase emphasized better handling of class imbalance, enhanced hyperparameter tuning, improved temporal validation, and selection of more effective algorithms. Refinement ensured that the final models were more robust for real-world seasonal disease forecasting.

2. Model Evaluation

The initial evaluation used a time-based split separating training, validation, and testing periods to capture real temporal patterns. Early baseline results showed reasonable accuracy for low-risk and medium-risk classes but weaker sensitivity for high-risk periods. Confusion matrices and preliminary F1-scores indicated that high-risk classes were underrepresented, causing misclassification toward lower-risk categories. Visualizations from the earlier exploration phase, including distribution plots, seasonality patterns, and lag correlations, guided the decision to apply stronger imbalance handling and more complex models.

3. Refinement Techniques

Multiple refinement techniques were introduced to address the limitations found in early results. These included algorithm-level improvements, data-level balancing, and additional feature engineering. SMOTE was applied to balance the underrepresented high-risk class. Several advanced algorithms were tested, including XGBoost, LightGBM, and Random Forests, on top of the logistic regression baseline. Ensemble-based tree models were prioritized because they aligned well with the temporal and nonlinear relationships present in dengue data. The refinement also used improved lag features, rolling indicators, and epidemiological variables from the feature engineering pipeline.

4. Hyperparameter Tuning

Hyperparameter tuning was strengthened by integrating Optuna for automated search across XGBoost and LightGBM parameter spaces. During refinement, the tuning process explored learning rate schedules, maximum depth ranges, subsampling levels, column sampling, minimum child weight, and regularization terms. Tuning produced parameter sets that improved stability over different time periods and provided better separation between high-risk and medium-risk classes. These adjustments supported both multiclass and binary high-risk detection tasks. Insights from tuning highlighted that shallower trees prevented overfitting on past seasonal data while controlled sampling boosted generalization.

5. Cross-Validation

The refinement introduced a time-series-aware validation strategy. Instead of random cross-validation, a temporal split was used, respecting chronological order. This prevented data leakage and simulated real deployment conditions where the model predicts future weeks based on past data. Internal validation during tuning used hold-out data from the 2022–2023 period, ensuring the model could generalize across multiple monsoon cycles. Adjusting cross-validation improved the reliability of the refined model's performance estimates.

6. Feature Selection

Feature selection occurred implicitly through model-based importance scoring and explicit removal of non-predictive or redundant columns. Iterative feature engineering produced weather-based lags, rolling climate indicators, mosquito-related conditions, search behavior metrics, epidemiological lags, and temporal encodings. During refinement, tree-based models helped confirm which features contributed most. Although all engineered features were kept during modeling, importance plots guided the decision to prioritize climatic and lag-based case indicators. This step reduced noise and improved interpretability while maintaining predictive strength.

Test Submission

1. Overview

The test submission phase prepared the refined model for evaluation on unseen 2024 dengue data. This step simulated real operational conditions where predictions must be generated weekly using only past information. The procedure involved preparing the final test dataset, applying transformations, loading the trained models, generating predictions, and calculating performance metrics. Outputs from this step will be integrated into full reporting when real results are completed.

2. Data Preparation for Testing

Test data preparation included selecting all records from weeks beginning in 2024 and applying the same transformations used during training. This involved ensuring standardized date formatting, applying feature scaling for models that required it, and verifying the presence of all engineered features. Missing values were already handled during the cleaning and merging pipeline, so the testing dataset was consistent with the training data structure. Only temporal ordering was preserved to avoid contamination from future records.

3. Model Application

The refined models were applied to the test set by feeding the engineered features into each trained classifier. Scaled versions of the test set were used where required. The following code snippet reflects the typical prediction workflow:

```
# Load models and scaler
xgb_model = joblib.load('xgboost_model.pkl')
lgb_model = joblib.load('lightgbm_model.pkl')
rf_model = joblib.load('random_forest_model.pkl')
scaler = joblib.load('feature_scaler.pkl')

# Prepare test features
X_test = df.loc[test_mask, feature_cols]
X_test_scaled = scaler.transform(X_test)

# Generate predictions
xgb_pred = xgb_model.predict(X_test)
lgb_pred = lgb_model.predict(X_test)
rf_pred = rf_model.predict(X_test)
```

This structure ensured consistency between training and test pipelines.

4. Test Metrics

Metrics such as accuracy, weighted F1-score, macro F1-score, and high-risk detection ability will be computed once final results are available. The test metrics will be compared with training and validation scores to check for overfitting or

underfitting, especially across seasonal transitions and provincial variability. High-risk recall will be a key metric because early detection is essential for public health response.

5. Model Deployment

Deployment considerations included saving models in `.pkl` format, exporting evaluation reports, and storing generated figures. Although full integration into a real-time system is beyond this stage, the current pipeline supports future deployment through consistent preprocessing, stable file structures, and standardized outputs from each model. The system is ready for potential integration with dashboards, health monitoring platforms, or automated alert systems.

6. Code Implementation

The refinement and test submission phases relied on well-structured Python scripts. Important sections included model training with hyperparameter tuning, loading saved models, scaling the test data, applying predictions, and exporting result files. Inline comments within the training scripts document the purpose of each code block and help maintain code readability. These scripts align with the broader project structure built through earlier data collection, cleaning, and feature engineering steps.

Conclusion

The refinement and testing stages strengthened the dengue fever risk alert system by improving model robustness, ensuring accurate temporal validation, balancing class distributions, and applying advanced ensemble algorithms. Although the final quantitative results will be included later, the processes completed so far established a stable and reproducible machine learning pipeline capable of supporting operational forecasting. The project has reached a point where only the final metrics and visualizations need to be inserted.

References

Python Libraries and Tools

pandas
numpy
scikit-learn
matplotlib
seaborn
xgboost
lightgbm
optuna
imbalanced-learn
joblib
shap
cdsapi
pytrends

Data Sources

1. ERA5 climate reanalysis dataset from the Copernicus Climate Data Store
2. Google Trends search data
3. Thai dengue case records in MOPH-compatible format