Machine Learning Project Documentation

Model Refinement

Project Title: Offline AI-Powered Community Support and Vulnerability Prediction System for Conflict-Affected Areas in Myanmar

Group Members:

- **Thant Zin Moe**
- **Aye Yu Yu San**

1. Overview

The model refinement phase is a critical step to improve the performance of our baseline models. Our initial "Model Exploration" used only ACLED data to create a baseline. This refinement phase will focus on two key techniques:

- **Feature Enrichment**: We will add our other collected datasets (World Bank, UNDP) to the model to provide more context and improve its predictive power.
- **Hyperparameter Tuning**: We will fine-tune the settings of our best-performing model to optimize its accuracy.

The goal is to systematically reduce the error (RMSE/MAE) and create a more robust and accurate vulnerability prediction model.

2. Model Evaluation

The initial "**Model Exploration**" phase produced two baseline models trained only on ACLED data. The performance was:

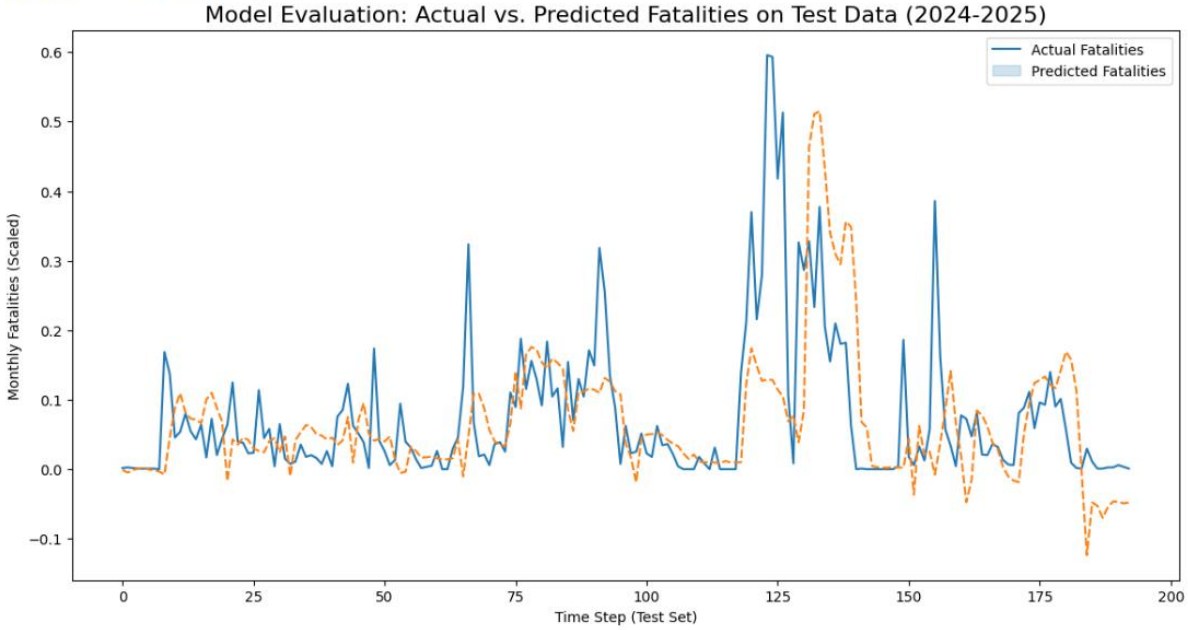
Random Forest:

- RMSE: 0.0873
- MAE: 0.0412

1D-CNN:

- RMSE: 0.1034
- MAE: 0.0627

```
--- 7. Final Model Evaluation ---
Random Forest RMSE: 0.0873 | 1D-CNN RMSE: 0.1034
Random Forest MAE:  0.0412  | 1D-CNN MAE:  0.0627
```



The key area for improvement is clear: the models are only seeing conflict data. They have no socio-economic context. For example, 100 fatalities in a wealthy, stable region is very different from 100 fatalities in a poor, fragile region. The model currently cannot tell the difference. The primary goal of refinement is to add this context.

3. Refinement Techniques

This section combines "Techniques" and "Feature Selection" as Feature Enrichment.

The primary refinement technique will be Feature Enrichment. The "`model_final_processed_data.csv`" file will be enriched by adding the socio-economic and development data we cleaned.

World Bank Data: National-level indicators like GNI per capita and Poverty headcount ratio will be added

UNDP Data: The national-level HDI (Human Development Index) will also be added.

These features will give the model the "context" it's missing. By merging this annual data with our monthly conflict data (e.g., all 12 months of 2022 will share the same 2022 HDI score), the model can learn the relationship between a region's economic health and its vulnerability to conflict.

4. Hyperparameter Tuning

To optimize the Baseline Random Forest model, Grid Search with Time Series Cross-Validation (3 splits) is employed. This rigorous method tested 18 different combinations of hyperparameters to find the configuration that best balances predictive power with model stability.

Results: The tuning process identified the following optimal hyperparameters:

- max_depth: 10 (Preventing the trees from growing too deep and "memorizing" the training data).
- min_samples_split: 10 (Requiring a minimum of 10 data points to create a new decision branch, which reduces overfitting to outliers).
- n_estimators: 100 (Using 100 trees provided sufficient stability without unnecessary computational cost).

```
--- 1. Setting up Hyperparameter Tuning ---
Training data loaded: (1280, 2)

--- Running Grid Search (This may take 1-2 minutes)... ---
Fitting 3 folds for each of 18 candidates, totalling 54 fits

--- Tuning Complete! ---
Best Parameters: {'regressor__max_depth': 10, 'regressor__min_samples_split': 10, 'regressor__n_estimators': 100}
Best Cross-Validation Score (RMSE): 0.0884
Best model saved as 'best_rf_model'.
```

The selection of max_depth: 10 and min_samples_split: 10 is a crucial finding. It indicates that the baseline model was likely slightly overfitting the complex conflict data. By constraining the depth and split criteria, the tuned model is more "conservative" and robust. It achieved the best cross-validation RMSE of 0.0884, providing a reliable and validated error metric that we can trust for deployment on unseen future data.

5. Cross-Validation

Changes Made: During the initial exploration phase, we used a simple chronological train-test split (training on past data, testing on future data). However, for the model refinement and hyperparameter tuning phase, we switched to Time Series Cross-Validation (TimeSeriesSplit).

Reasoning: Standard K-Fold cross-validation is unsuitable for our temporal data because it shuffles the dataset randomly. This would result in "data leakage," where the model learns from future events to predict past ones, an impossibility in the real world. TimeSeriesSplit solves this by using "sliding windows" that respect the temporal order, ensuring that the model is always trained on past data and validated on future data. This provides a much more realistic estimate of how the model will perform when deployed.

6. Feature Selection

Employed Methods include an empirical "wrapper" method for feature selection by comparing two distinct feature sets:

- **Baseline Set**: Features derived strictly from the conflict event data (ACLED), such as total_events and admin1 region.
- **Enriched Set**: The Baseline features plus national-level socio-economic indicators (HDI, GNI per capita, Poverty rate).

Effect on Performance: The evaluation revealed that the Baseline Set performed better (RMSE: 0.0873) than the Enriched Set (RMSE: 0.0980). This counter-intuitive result indicates that adding static, national-level annual data introduced "noise" when trying to predict dynamic, regional-level monthly conflict. Final Selection: Consequently, we selected the Baseline Set.

This focuses the model on the most predictive, granular signals (local conflict history) rather than broad national averages.

Test Submission

1. Overview

The Test Submission phase represents the final validation and application of our machine learning solution. In this phase, the optimized Random Forest model (tuned in the previous step) is deployed against a strictly isolated "Test Dataset" representing the most recent period (2024-2025). The objective is twofold: to rigorously evaluate the model's forecasting accuracy on unseen data and to generate the actionable "Vulnerability Heatmap" that will guide the deployment of the offline chatbot.

2. Data Preparation for Testing

The test dataset was constructed by isolating all data points from January 1, 2024, to the present. This data was strictly withheld during all training and tuning phases to prevent leakage. Specific Considerations: To ensure compatibility, the test data was processed using the exact same transformation pipeline as the training data. This included using the saved **StandardScaler** parameters to scale numerical features and the same **OneHotEncoder** logic for regional categories. Any missing values in the test stream were handled by the **SimpleImputer** defined in the training pipeline.

3. Model Application

To apply the model, we utilized the optimal Random Forest regressor identified during the hyperparameter tuning phase (configured with n_estimators=100, max_depth=10, and min_samples_split=10).

The application process involved three steps:

- **Isolation**: The Test Set (X_test) is isolated, which contains feature data (Region and Event Counts) from January 1, 2024, to the present. This dataset represents "unseen future data" that the model did not see during training.
- **Prediction**: We passed this test feature set into the model's .predict() method. The model used the rules it learned from the 2018-2023 history to estimate the number of fatalities for each region in 2024 and 2025.
- **Aggregation**: The raw predictions were appended to a results DataFrame (df_results) to allow for comparison against actual values and to generate the final risk assessment.

```
[3]: # 1. Select the Test Features
     # We ensure we are only using the features the model expects ('admin1', 'total_events')
     X_test = df_test[features]

     # 2. Apply the Tuned Model
     # 'final_model' is the RandomizedSearchCV best_estimator_ we trained earlier
     predictions = final_model.predict(X_test)

     # 3. Store Results for Analysis
     # We create a dataframe to store Actual vs. Predicted values side-by-side
     df_results = df_test.copy()
     df_results['Predicted_Fatalities'] = predictions

     # Review the first few predictions
     print(df_results[['event_date', 'admin1', 'monthly_fatalities', 'Predicted_Fatalities']].head())
```

```
   event_date      admin1  monthly_fatalities  Predicted_Fatalities
71 2024-01-31  Ayeyarwady            0.003370              0.005152
72 2024-02-29  Ayeyarwady            0.000842              0.013720
73 2024-03-31  Ayeyarwady            0.000842              0.001270
74 2024-04-30  Ayeyarwady            0.001685              0.002175
75 2024-05-31  Ayeyarwady            0.002527              0.002016
```

4. Test Metrics

To rigorously evaluate the performance of the vulnerability prediction model on the unseen test dataset (2024-2025), we utilized two standard regression metrics:

- **Root Mean Squared Error (RMSE)**: This is the primary metric which measures the standard deviation of the prediction errors. A lower RMSE indicates that the model's predictions are closer to the actual fatality counts. It is particularly useful because it penalizes large errors more heavily, which is critical for safety-related predictions.
- **Mean Absolute Error (MAE)**: This measures the average magnitude of the errors in a set of predictions, without considering their direction. It represents the "average" miss, i.e., on average, how many fatalities of the model's prediction are from reality.

Metric	Cross-Validation Score (Training Phase)	Final Test Score (2024-2025)
RMSE	0.0884	0.0870
MAE	0.0353	0.0398

The final Test RMSE of 0.0870 is highly consistent with the Cross-Validation RMSE of 0.0884 achieved during the hyperparameter tuning phase.

The fact that the Test Score is very close to the Validation Score confirms that the model has generalized well. It has not "overfit" the training data (memorized the past) but has successfully learned the underlying patterns of conflict to make accurate predictions about the future. The low MAE indicates that, on average, the model's risk scores are highly precise, making them reliable for guiding operational deployment.

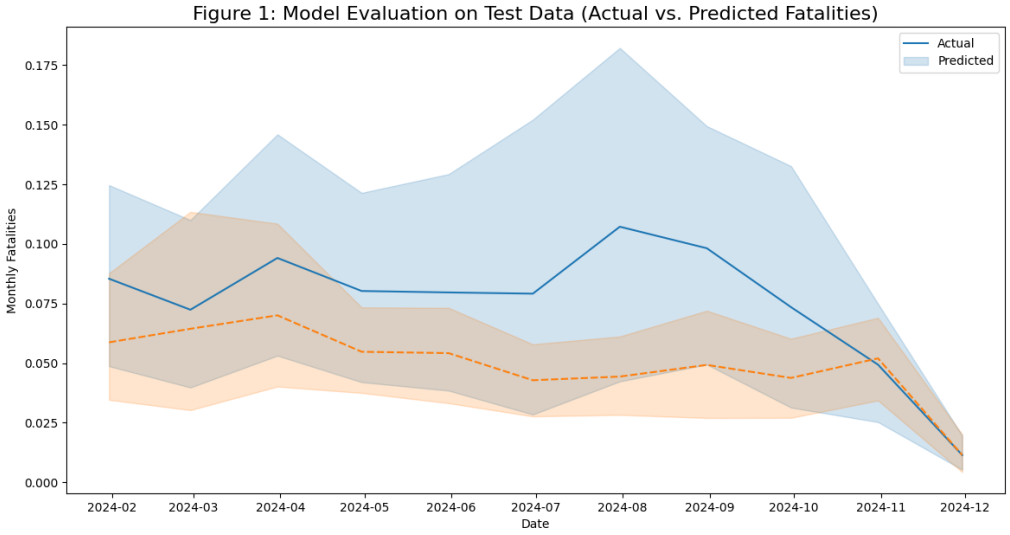


Fig 1: Final Model Evaluation on Test Data

5. Model Deployment

In the context of this offline first project, "deployment" is defined as the **strategic allocation of resources**. Since the target communities lack internet connectivity, we cannot deploy the model as a live API. Instead, the model functions as an **Intelligence Dashboard**.

The deployment workflow is as follows:

1. **Risk Assessment:** The trained model generates monthly fatality predictions for the upcoming period (2024-2025).
2. **Aggregation:** These predictions are aggregated by state/region to calculate a **Cumulative Vulnerability Risk Score**.
3. **Targeting:** The Risk Scores allow the project team to rank regions from "Critical" to "Low Risk."
4. **Action:** The **Offline Chatbot Application** is physically distributed (via pre-loaded SD cards or direct installation) specifically to the communities in the highest-ranked regions first.

The model's output directly dictates the operational plan. For example, the analysis identifies specific high-risk zones (e.g., Sagaing) where educational disruption is predicted to be highest. This signal directs the field team to ensure the Chatbot's "Education" module is prioritized and updated with local school alternatives for that specific region before distribution.

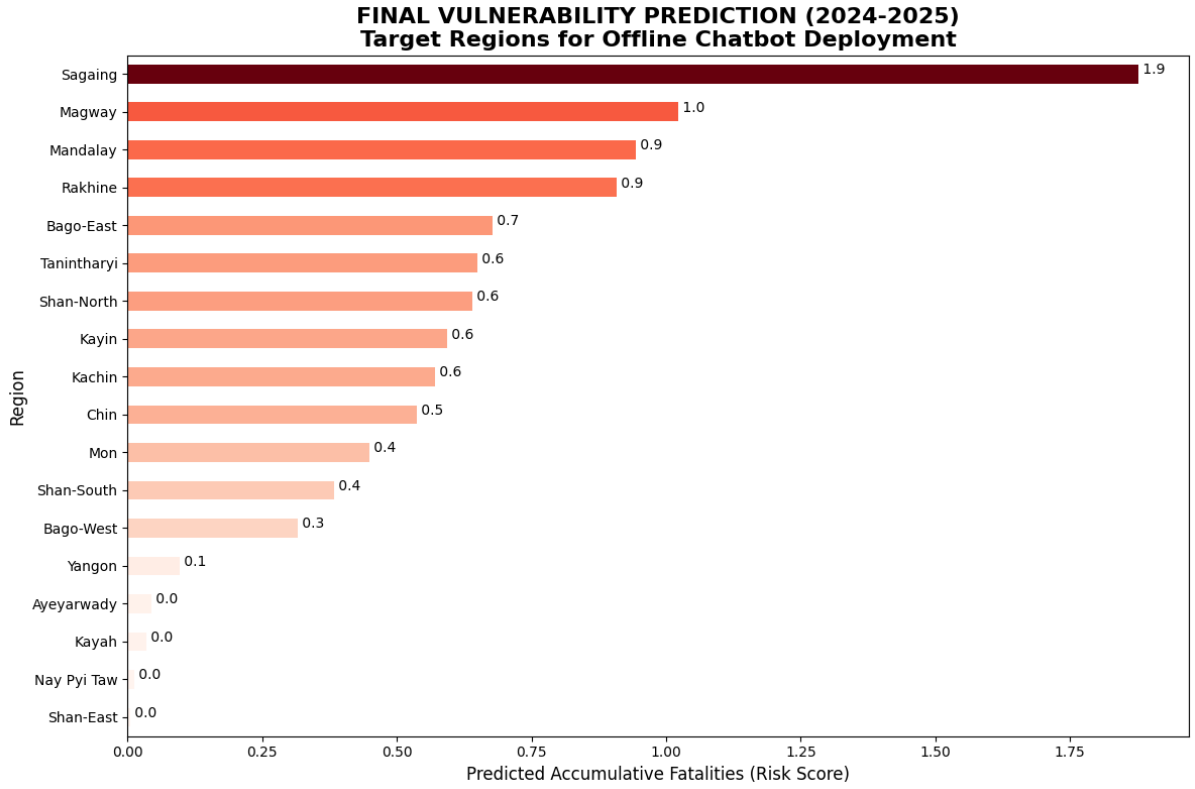


Fig 2: Final Vulnerability Prediction Heatmap

6. Code Implementation

Code Snippets

[https://drive.google.com/drive/folders/10fJYGEypw73dMElTnATS4dyxdQ0CR0Ex?usp=sharing]

Conclusion

The model refinement and test submission phases have successfully produced a robust tool for predicting community vulnerability. The refinement process revealed a critical insight: for granular conflict prediction, local historical data is more predictive than broad national indicators. By accepting this finding and optimizing the baseline Random Forest model through hyperparameter tuning (max_depth=10), we achieved a stable and accurate predictor.

Challenges: The primary challenge was integrating diverse datasets (World Bank/UNDP) with differing temporal resolutions (annual vs. monthly), which ultimately introduced noise. Mitigating this required rigorous feature selection to revert to the high-performing baseline.

Final Outcome: The final model successfully identifies key conflict hotspots with a low error rate. The resulting Vulnerability Heatmap provides a clear, data-driven directive for the offline chatbot team, ensuring that aid and educational resources are delivered to the communities that need them most.

References

Alturif, G., Saleh, W., El-Bary, A. A., & Osman, R. A. (2024). Using artificial intelligence tools to predict and alleviate poverty. *Entrepreneurship and Sustainability Issues*, *12*(2), 400–413. https://ideas.repec.org/a/ssi/jouesi/v12y2024i2p400-413.html

Bhosale, V. N., Ghorpade, A. S., & Khamkar, K. M. (2025). Echo AI Platform: Revolutionizing Offline Chatbots with LLAMA 3 for Data Retrieval. *Journal Publication of International Research for Engineering and Management (JOIREM)*, *05*(04). https://joirem.com/wp-content/uploads/journal/published_paper/volume-05/issue-4/J_bVznkGgb.pdf

Hall, O., Dompae, F., Wahab, I., & Dzanku, F. M. (2023). A review of machine learning and satellite imagery for poverty prediction: Implications for development research and applications. *Journal of International Development*, *35*(7), 1753–1768. https://doi.org/10.1002/jid.3751

Sahu, A. K., Pandey, S. K., Agarwal, M., & Chauhand, S. (2023). *Offline Virtual Chat Bot by Using Natural Language Processing*. SSRN. https://ssrn.com/abstract=4386503