# Predictive Modeling for Tsunami Risk Assessment from Seismic Data

**Name:** Justin Xiao

**Affiliation:** Brown University

**Github Repository:** https://github.com/JustinXre2020/Earthquake-Tsunami/

## Introduction

The objective of this project is to develop a machine learning pipeline capable of predicting whether a seismic event (earthquake) will generate a tsunami. Tsunami events, though less frequent than standard earthquakes, carry catastrophic potential for coastal communities; therefore, early and accurate prediction is vital for disaster mitigation. The dataset utilized for this study includes features such as magnitude, depth, location coordinates, and earthquake alert levels. Previous work in this domain typically focuses on geophysical simulations or classification models with predictive accuracy ranging from 85% to 95%[6]. This report evaluates four machine learning algorithms to determine the most effective approach for high-recall disaster prediction.

## EDA (Exploratory Data Analysis)

Exploratory analysis shows that the dataset is imbalanced, with approximately 67% of events resulting in no tsunami. Key findings from the EDA include:

- **Magnitude and Depth:** Tsunami-generating events tend to occur at higher magnitudes and shallower depths compared to non-tsunami events.
- **Alert Levels:** The alert feature shows a strong ordinal correlation with the target, where "Red" and "Orange" alerts are highly indicative of tsunami risk.
- **Geographic Clustering:** Data visualization highlights "hotspots" in the Pacific Ring of Fire, particularly near Indonesia and Alaska, where subduction zones increase the probability of displacement-driven tsunamis.

# Methods

### Data Assembly and Splitting

Two earthquake-event datasets were concatenated after identifying shared columns, and duplicate events were removed. The target variable was tsunami, with all remaining columns used as features. For each run, the data was split into training/validation (80%) and a held-out test set (20%) using stratification to preserve class proportions. Hyperparameter tuning was performed only within the 80% subset using 5-fold StratifiedKFold. The full pipeline was repeated across ten random seeds [0,1,7,13,21,42,66,88,123,2025] to assess robustness.

### Feature Engineering

Two engineered ordinal features were created:

1. tsunami_potential_bin, computed from magnitude and clipped depth and binned into {none, low, medium, high};
2. gap_bin, derived by binning azimuthal gap (gap) into {poor, fair, good, excellent} to reflect measurement quality.

In addition, a set of non-modeling or redundant fields (e.g., metadata/text/location variables) was dropped after feature creation.

### Data Preprocessing

A ColumnTransformer was used with remainder='drop':

- Ordinal branch: tsunami_potential_bin and gap_bin were imputed with a constant and encoded using OrdinalEncoder with fixed orderings (unknown categories encoded as −1).
- Min–max branch: longitude, latitude, mmi, cdi were median-imputed and scaled with MinMaxScaler.
- Standardization branch: depth and magnitude were median-imputed and scaled with StandardScaler.

Transformations were fit only on training folds/splits and then applied to validation/test data during each K-Fold cross validation.

**Model Pipelines**

All models used the same upstream data construction and feature preparation unless otherwise noted: the two source tables were concatenated, duplicates were removed, y was set to tsunami, and X contained all remaining columns. A deterministic feature-engineering step created two ordinal features—tsunami_potential_bin and gap_bin—and then dropped non-modeling and metadata fields (including the raw gap). A ColumnTransformer was used for preprocessing with remainder='drop':

- Ordinal branch: tsunami_potential_bin, gap_bin → constant imputation → OrdinalEncoder with fixed order (unknown categories encoded as -1)
- Min–max branch: longitude, latitude, mmi, cdi → median imputation → MinMaxScaler
- Standardization branch: depth, magnitude → median imputation → StandardScaler

Baseline (DummyClassifier)

A DummyClassifier using the "most frequent" strategy was trained as a performance floor using the same train/test split protocol.

Logistic Regression

A scikit-learn pipeline was constructed as:

Pipeline(preprocessor → LogisticRegression(max_iter=2000, random_state=seed)).

Support Vector Classifier (SVC)

A scikit-learn pipeline was constructed as:

Pipeline(preprocessor → SVC(random_state=seed)) (kernel and other settings tuned during search).

Random Forest

A scikit-learn pipeline was constructed as:

Pipeline(preprocessor → RandomForestClassifier(random_state=seed)).

XGBoost

XGBoost used the same feature-engineered inputs and the same preprocessing transformer, but preprocessing was applied explicitly within each fold/split (i.e., fit_transform on training partitions and transform on validation/test partitions), followed by training an XGBClassifier with early stopping.

**Machine Learning Pipeline and Hyperparameter Tuning**

Hyperparameter selection was performed within the training/validation portion of each run using stratified cross-validation, with F1-score as the primary selection metric. The full procedure was repeated across ten random seeds [0,1,7,13,21,42,66,88,123,2025].

Logistic Regression (GridSearchCV)

- Tuning method: GridSearchCV over a preprocessing+model pipeline
- CV: StratifiedKFold(n_splits=5, shuffle=True, random_state=seed)
- Scoring: scoring='f1'
- Grid: C ∈ {0.001, 0.01, 0.1, 1, 10, 100, 1000}
- solver ∈ {liblinear, saga}
- Model selection: best mean CV F1-score and final evaluation on the held-out test set.

SVC and Random Forest (GridSearchCV)

- Tuning method: GridSearchCV over preprocessing+model pipelines
- CV: 5-fold stratified (StratifiedKFold) with shuffling per seed
- Scoring: scoring='f1'
- Model selection: best mean CV F1-score; final evaluation on the held-out test set.

XGBoost (Manual Grid + Early Stopping)

- Tuning method: manual enumeration via ParameterGrid with 5-fold StratifiedKFold

Within each fold:

1. Fit preprocessing on the fold's training partition and transform validation
2. Train XGBClassifier(..., early_stopping_rounds=50) using the fold validation set as eval_set

3. Score using fold F1-score on validation predictions
4. Selection criterion: highest mean F1-score across folds

- Class imbalance handling: scale_pos_weight was searched around the empirical ratio computed from the training/validation subset (negative/positive), including ±20% variants
- Final training: retrained on the full 80% training/validation subset with an internal 10% split for early stopping; evaluated once on the held-out test set.

**Evaluation Metrics**

Final performance was reported on the held-out test set using Accuracy, Precision, Recall, and F1-score. Given class imbalance and the cost of missed tsunami events, F1-score and Recall were emphasized. Mean ± standard deviation across the ten random seeds summarized robustness. Also given the high stakes of tsunami prediction (where missing a positive case can be catastrophic), Recall is particularly critical to ensure we capture as many true tsunami events as possible.

**Machine Learning Algorithms & Hyperparameters**

We evaluated one baseline and four machine learning algorithms, including two linear and two non-linear models:

- Baseline: A DummyClassifier using the "most frequent" strategy was used to establish a performance floor.
- Logistic Regression (Linear): We tuned regularization strength and penalty types to handle potential multicollinearity.
- Support Vector Classifier (Non-linear): We explored different kernel functions (RBF, linear) and tuned C (regularization) and gamma to control the decision boundary flexibility.
- Random Forest (Non-linear/Ensemble): We tuned the number of trees (n_estimators), maximum tree depth (max_depth), and minimum samples per split to prevent overfitting.
- XGBoost (Non-linear/Ensemble): We performed extensive tuning on parameters including learning_rate, max_depth, subsample, colsample_bytree, and scale_pos_weight (to address class imbalance).

### Uncertainty and Robustness Analysis

To measure uncertainty in performance metrics arising from variability in data splitting and the non-deterministic nature of certain algorithms (e.g., Random Forest and XGBoost), the entire training and evaluation process was repeated 10 times using different random seeds (random_state values of 0,1,7,13,21,42,66,88,123,2025). The mean and standard deviation of all evaluation metrics were then computed to provide confidence intervals for the results.

# Results

### Model Performance and Baseline Comparison

All models significantly outperformed the baseline accuracy of 0.67.

| Model | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| XGBoost | 0.7650 ± 0.0270 | 0.6041 ± 0.0369 | 0.8262 ± 0.0492 | 0.6961 ± 0.0235 |
| Random Forest | 0.7725 ± 0.0270 | 0.6590 ± 0.0467 | 0.6262 ± 0.0572 | 0.6409 ± 0.0444 |
| SVC | 0.6775 ± 0.0439 | 0.5042 ± 0.0442 | 0.7431 ± 0.0596 | 0.6001 ± 0.0470 |
| Logistic Regression | 0.6960 ± 0.0232 | 0.5530 ± 0.0629 | 0.3708 ± 0.0512 | 0.4410 ± 0.0445 |
| Baseline | 0.6700 ± 0.0000 | 0.0000 ± 0.0000 | 0.0000 ± 0.0000 | 0.0000 ± 0.0000 |

*Table 1: Performance Summary on Test Set with Standard deviation*

## Performance Analysis

### Baseline Comparison

The baseline model, which always predicted the majority class (“No Tsunami”), achieved an accuracy of 0.6700 ± 0.0000 but failed to identify any positive cases. This illustrates that accuracy alone is not informative under class imbalance. Therefore, all trained models improved substantially over the baseline on recall and F1-score.

### Top Performing Models

Overall, XGBoost delivered the best balance between detecting tsunami events and maintaining overall predictive quality. It achieved the highest Recall (0.8262 ± 0.0492) and the highest F1-Score (0.6961 ± 0.0235), indicating the strongest ability to capture true tsunami events while maintaining moderate precision (0.6041 ± 0.0369).

Random Forest achieved the highest Accuracy (0.7725 ± 0.0270) and the highest Precision (0.6590 ± 0.0467), but its Recall was notably lower (0.6262 ± 0.0572),

suggesting a more conservative decision rule that reduces false positives at the cost of missing more true tsunami cases.

SVC showed strong sensitivity with Recall = 0.7431 ± 0.0596, but its Precision was lower (0.5042 ± 0.0442), yielding a moderate F1-Score (0.6001 ± 0.0470).

### Linear vs. Non-Linear Models

The non-linear models (XGBoost, Random Forest, SVC) consistently outperformed the linear model (Logistic Regression), particularly on Recall and F1-score. Logistic Regression had the weakest detection capability, with Recall = 0.3708 ± 0.0512 and F1-Score = 0.4410 ± 0.0445, despite a moderate Accuracy (0.6960 ± 0.0232). This pattern suggests that the relationship between predictors and tsunami occurrence is not well captured by a linear decision boundary under the current feature set.

### Conclusion on Model Selection

Given the high-stakes nature of tsunami prediction, Recall is the most critical metric because false negatives represent missed tsunami events. Under this criterion, XGBoost is the preferred model, achieving the best Recall (0.8262 ± 0.0492) and the best overall F1-score (0.6961 ± 0.0235). If minimizing false alarms is prioritized instead, Random Forest offers higher Precision (0.6590 ± 0.0467) but with a substantial reduction in Recall.

## Figures

### Confusion Matrix

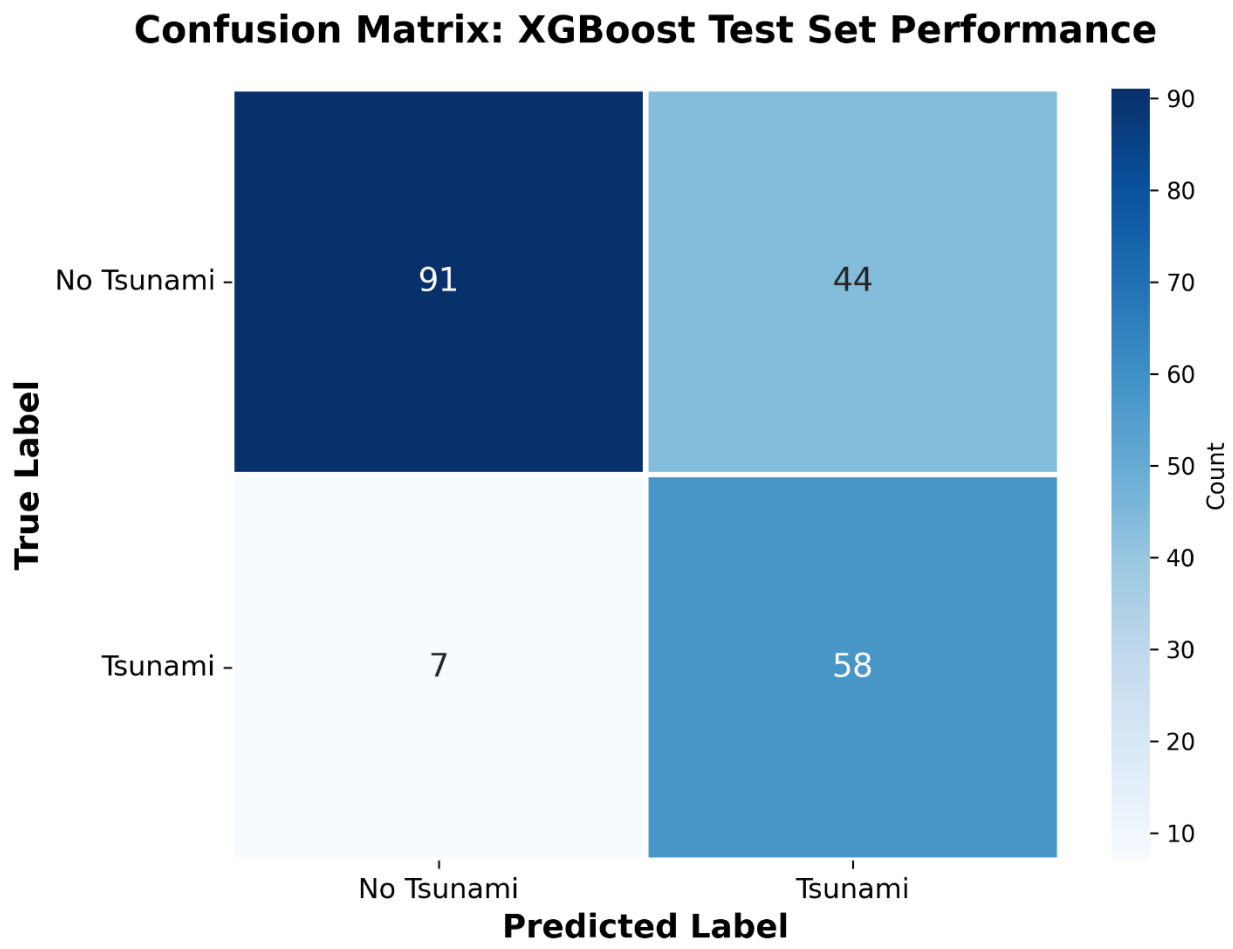


*Figure 1: Confusion Matrix for XGBoost Model.*

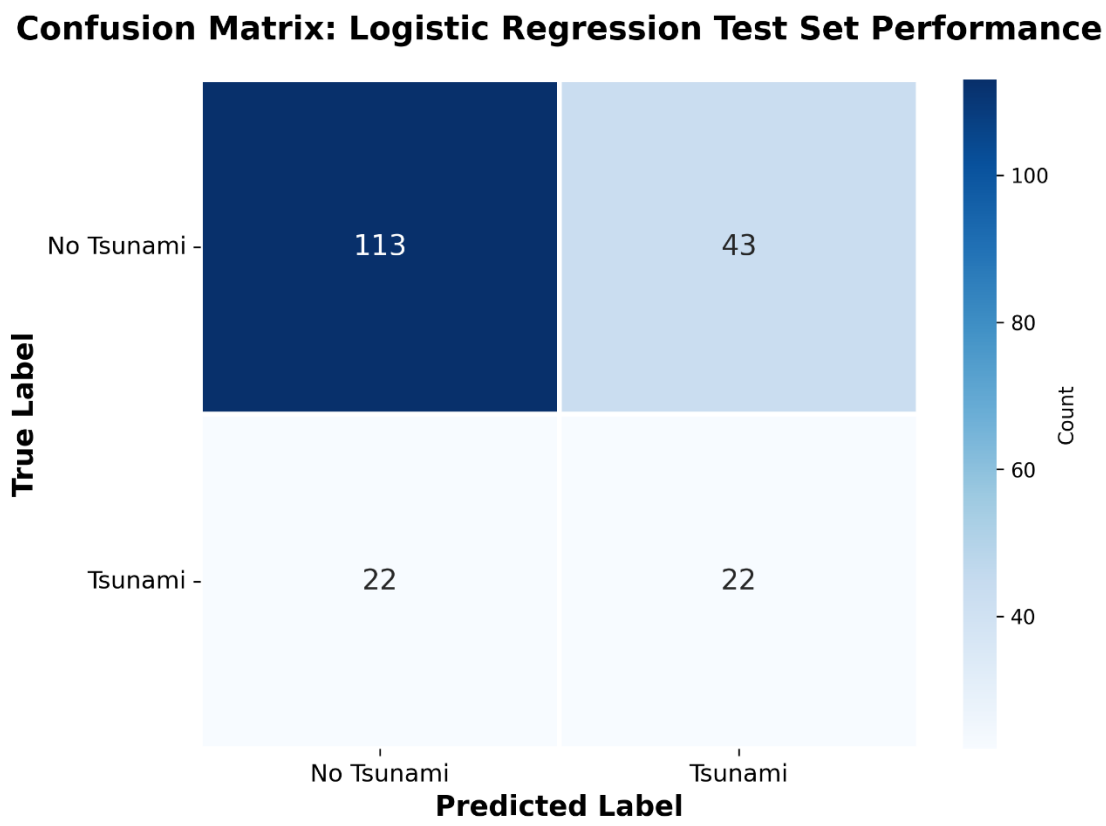


*Figure 2: Confusion Matrix for Logistic Regression Model.*

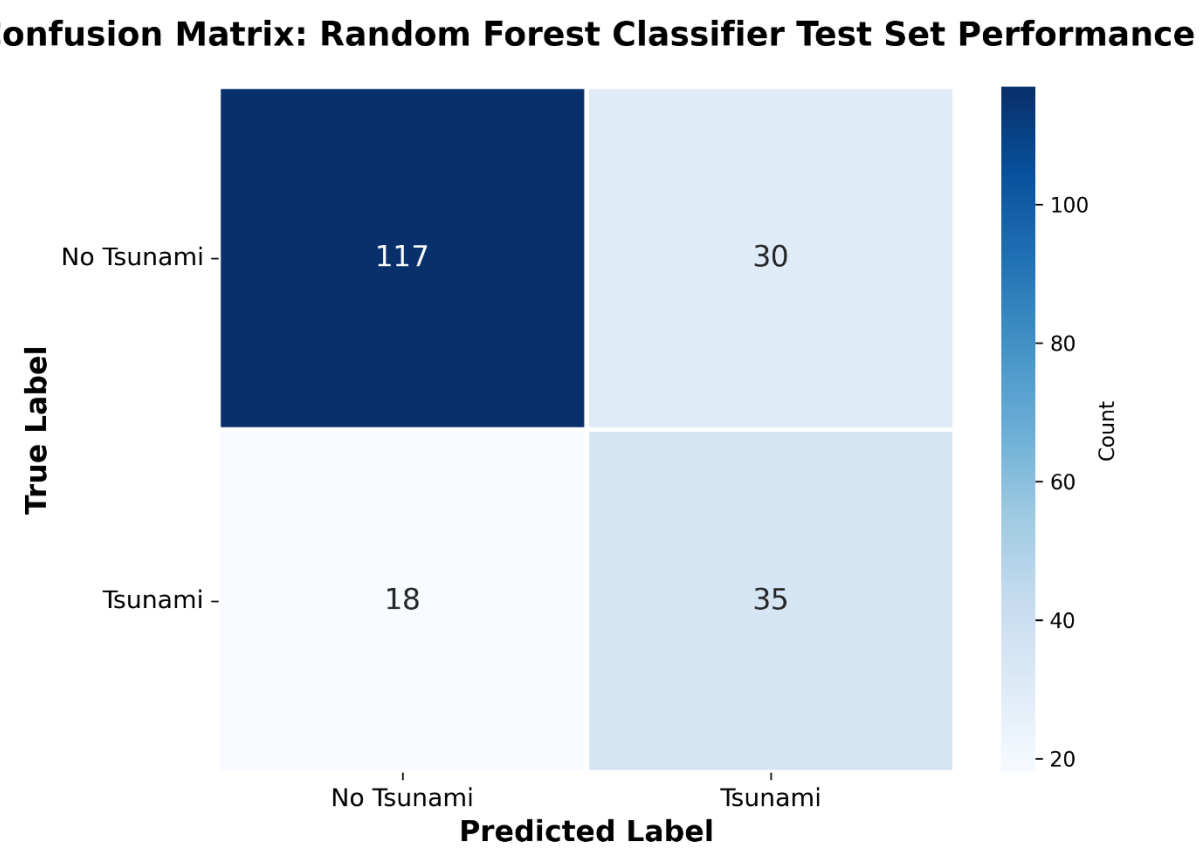


*Figure 3: Confusion Matrix for Random Forest Classifier Model.*

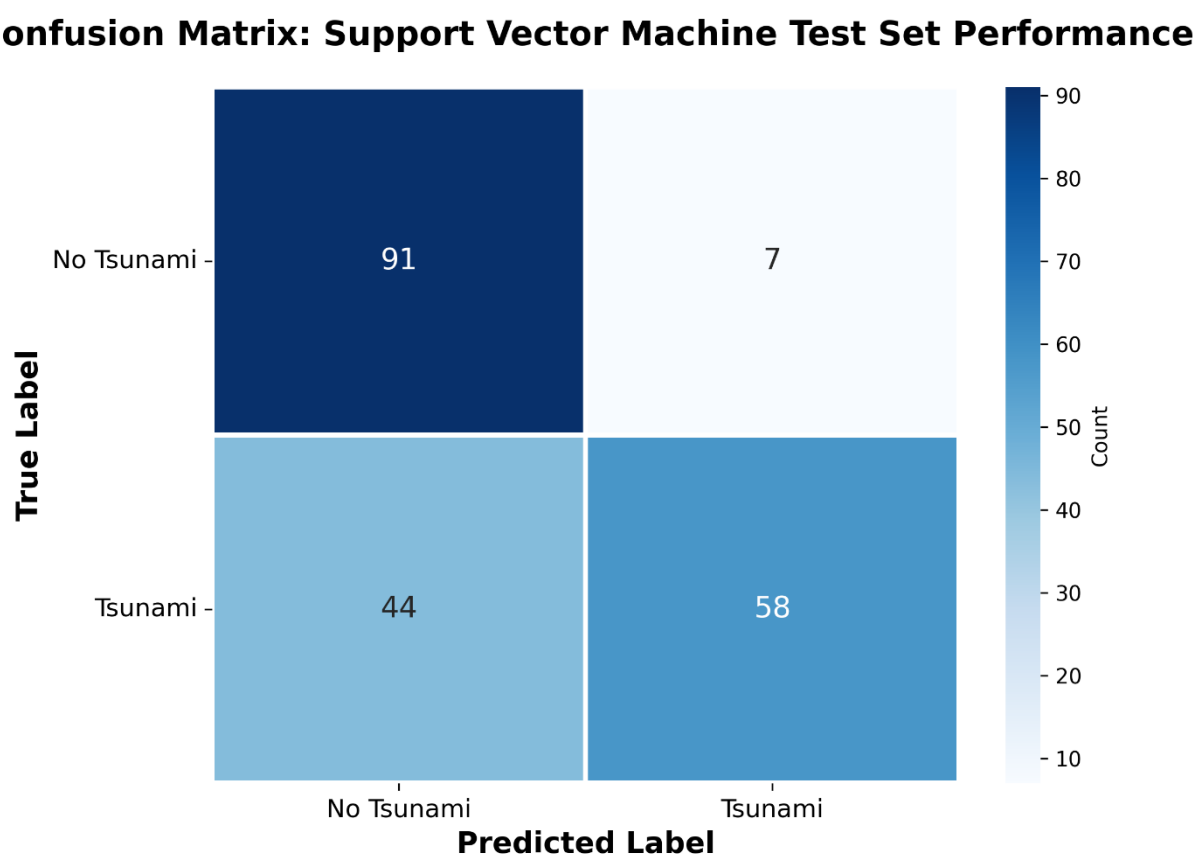


*Figure 4: Confusion Matrix for Support Vector Machine Classifier Model.*

**ROC Curves**

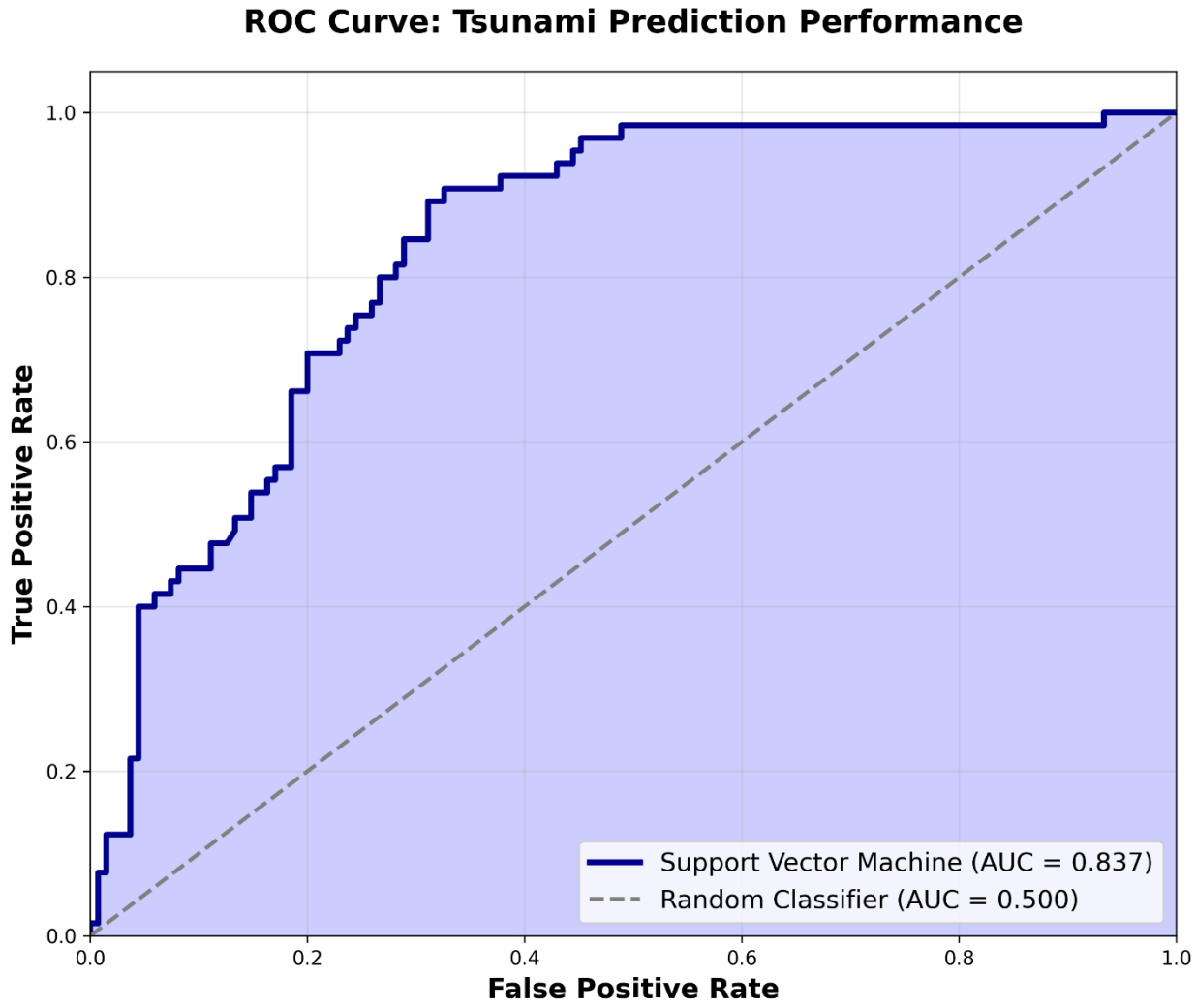


*Figure 5: ROC Curve for the Support Vector Machine showing an AUC of 0.837*

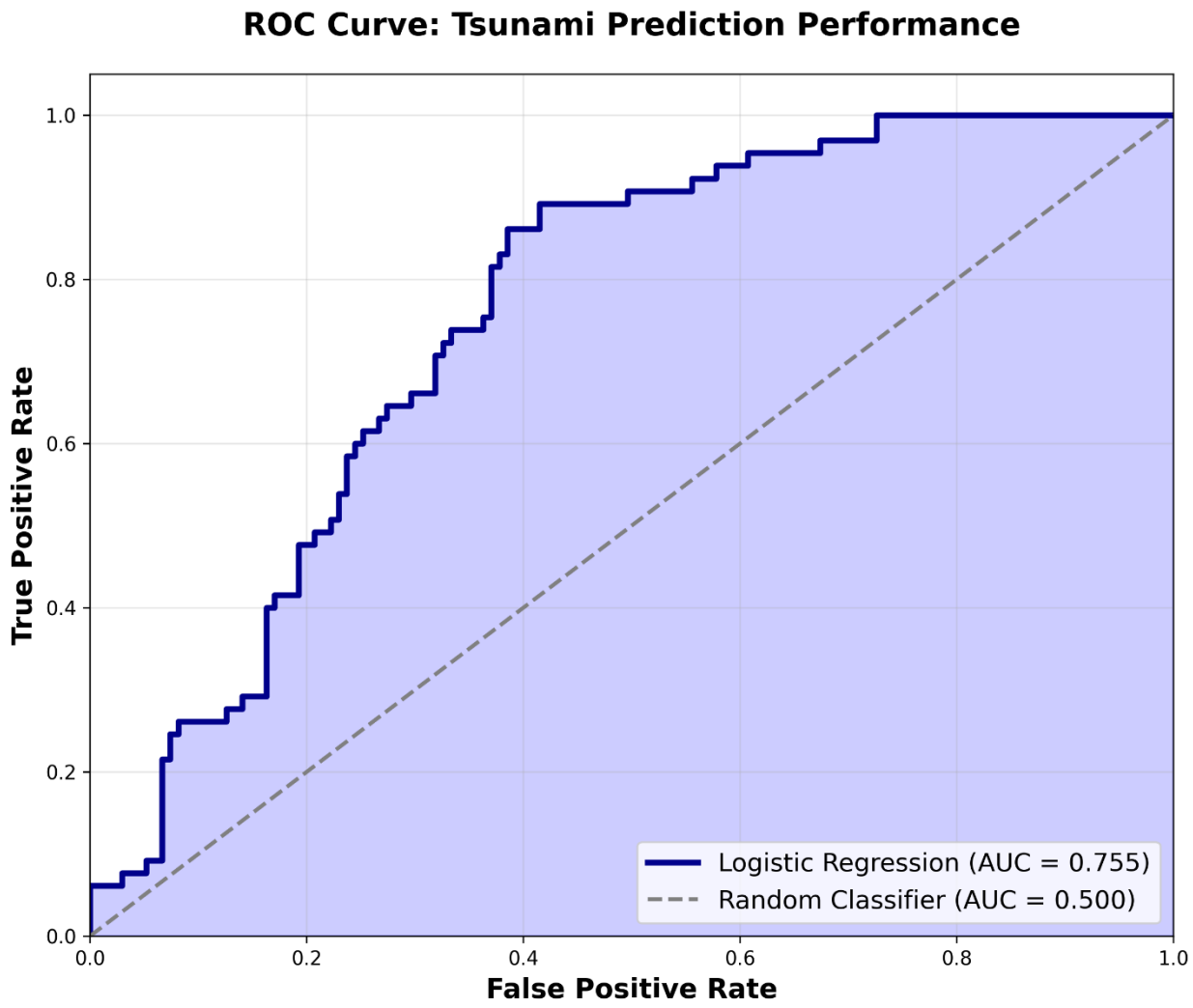


*Figure 6: ROC Curve for the Logistic Regression showing an AUC of 0.755*

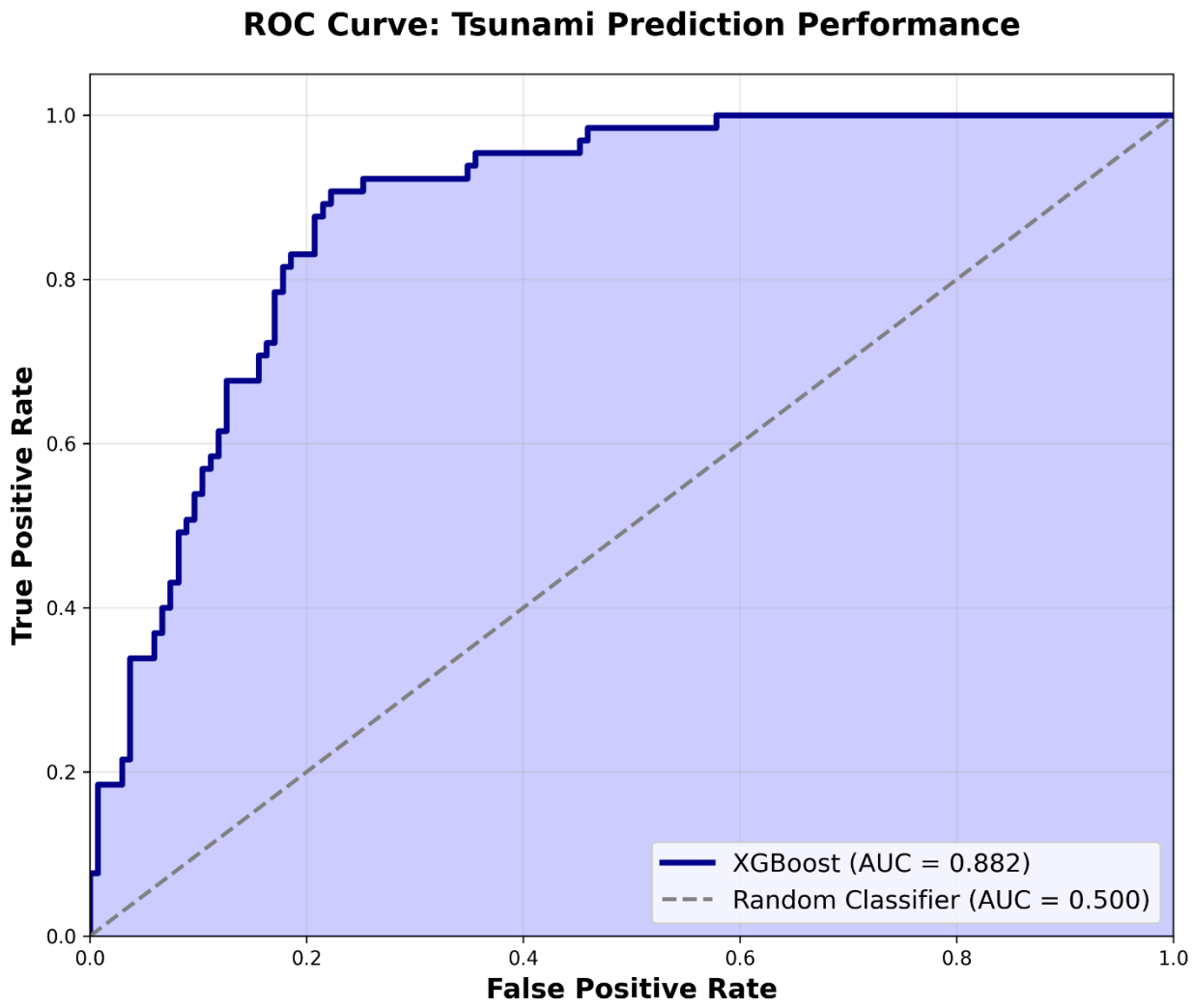


*Figure 7: ROC Curve for the XGBoost showing an AUC of 0.882, demonstrating excellent class separation capabilities.*

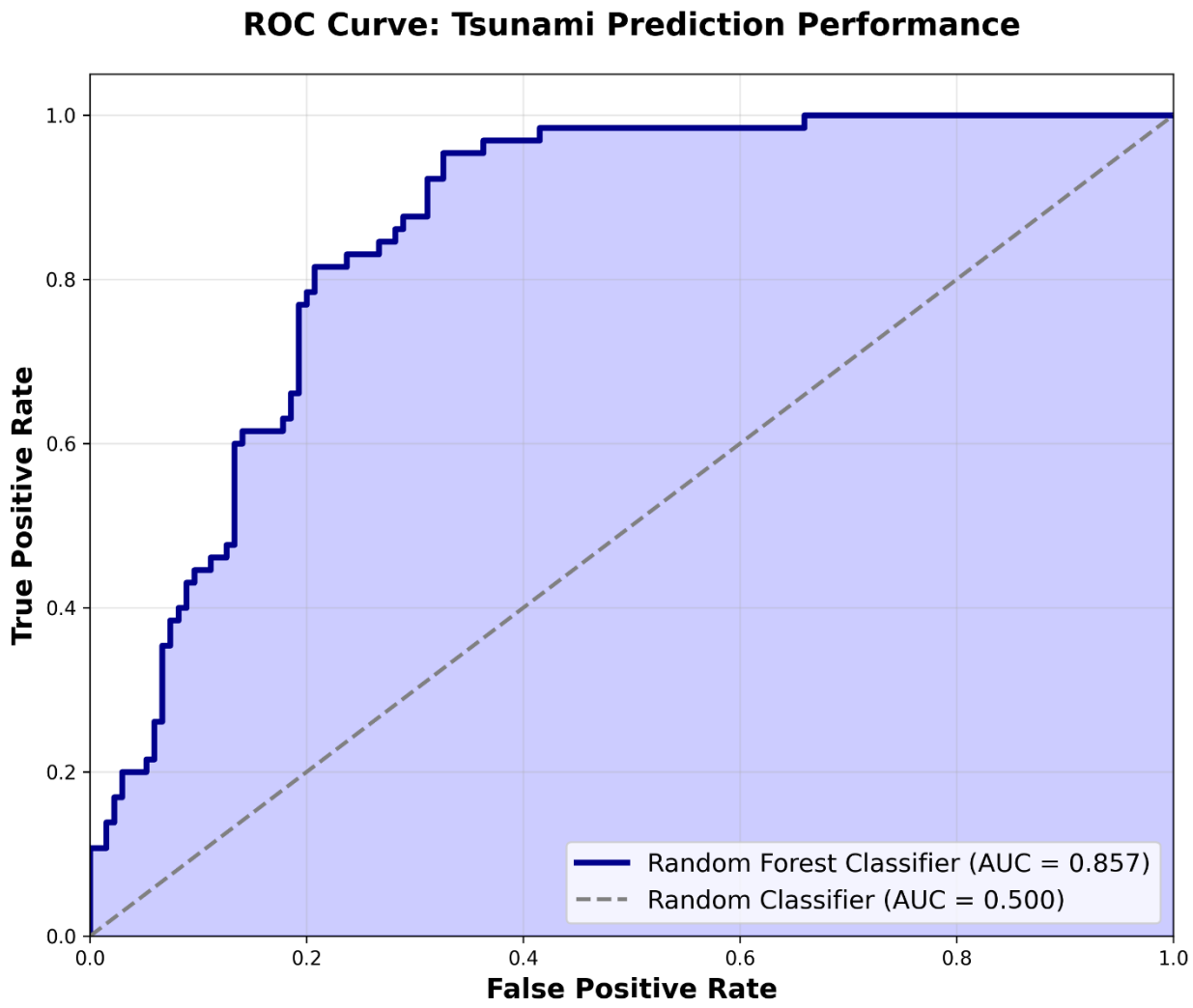


*Figure 8: ROC Curve for the Random Forest showing an AUC of 0.857.*

### Model Interpretation and Feature Importance

As for model interpretation, there are three global feature importances (Gain, Weight, and SHAP) and local SHAP values.

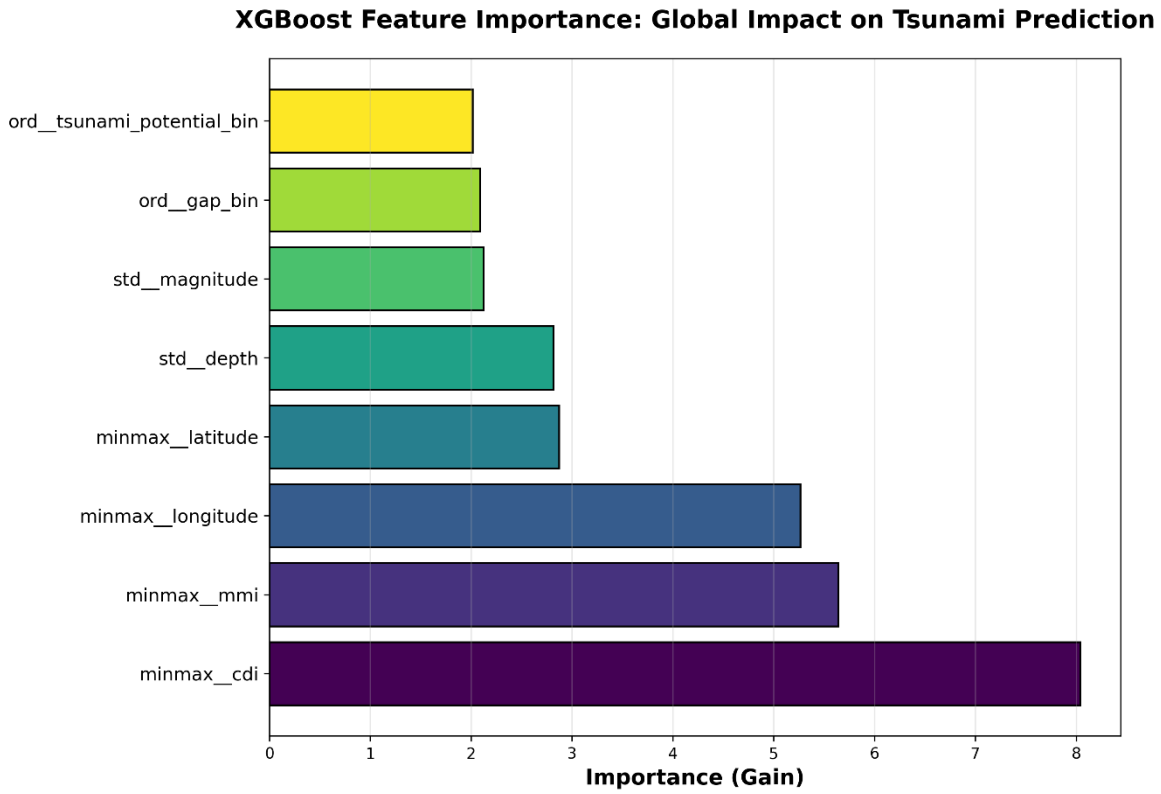


*Figure 9: XGBoost Feature Importance (Gain). This chart illustrates global feature importance using the Gain metric.*

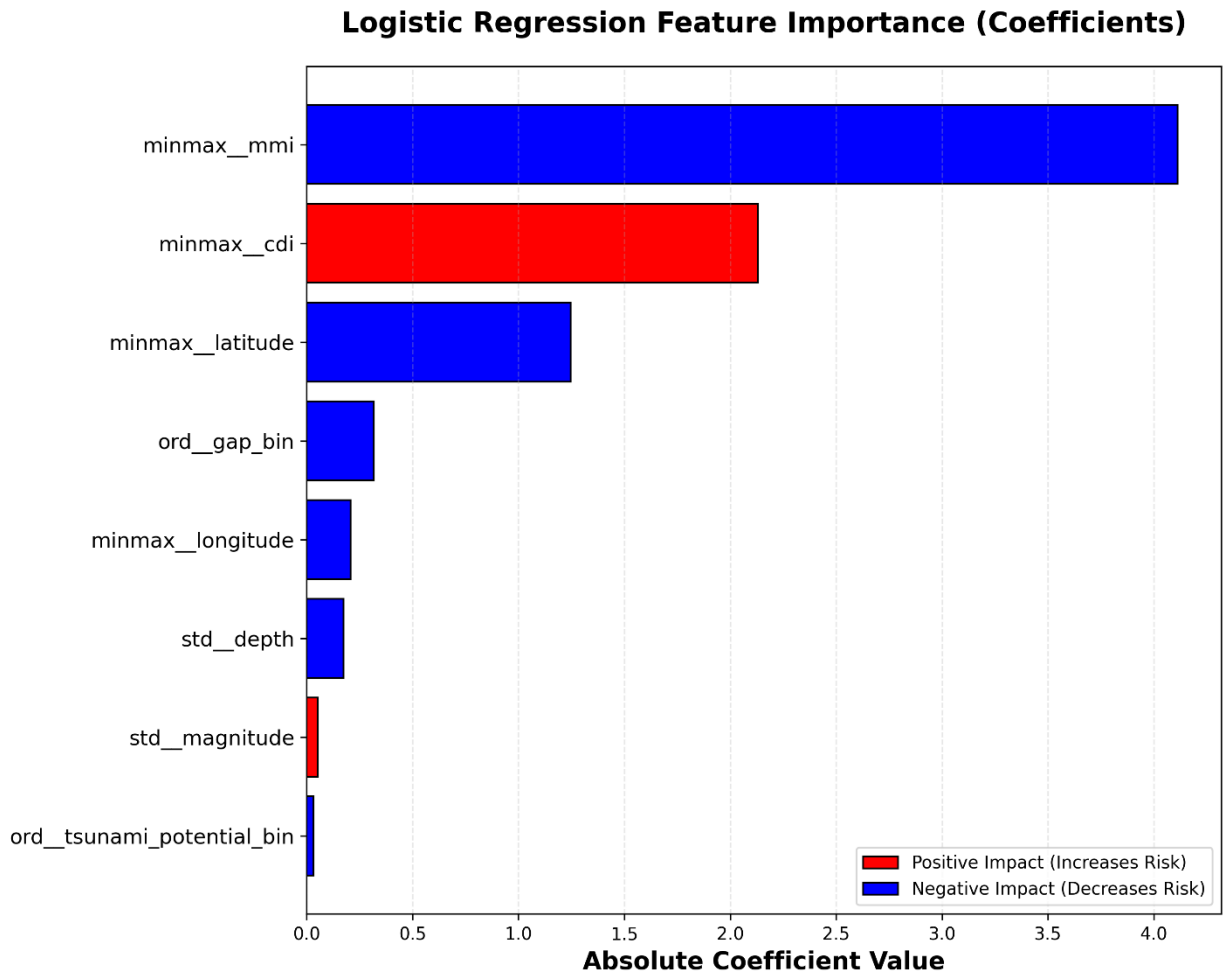


*Figure 10: Logistic Regression Feature Importance. This chart ranks features by their absolute coefficient values, revealing a strong dependence on specific categorical location variables rather than general physical metrics.*

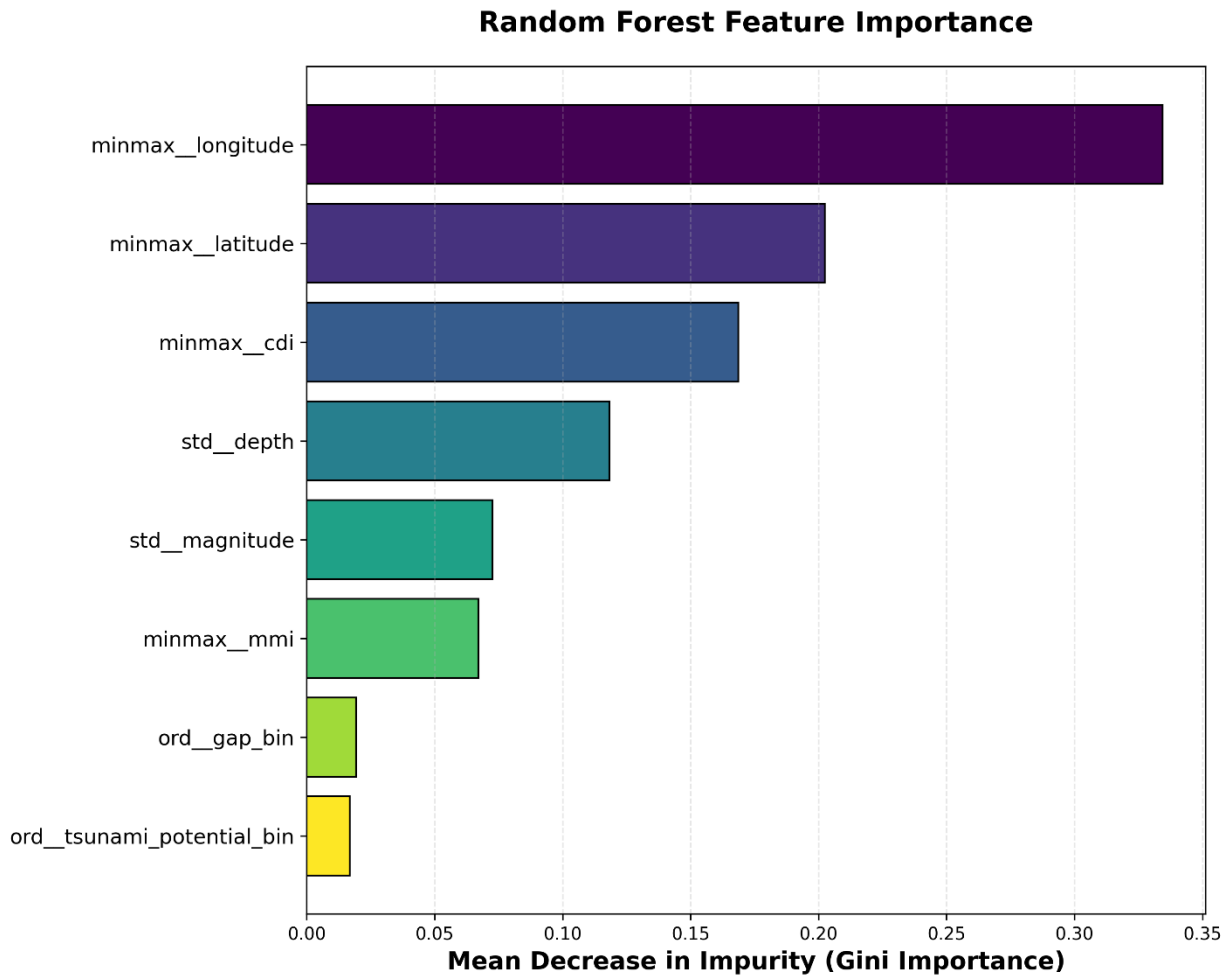


*Figure 11: Random Forest Feature Importance. This horizontal bar chart visualizes feature importance based on the Mean Decrease in Impurity (Gini Importance).*

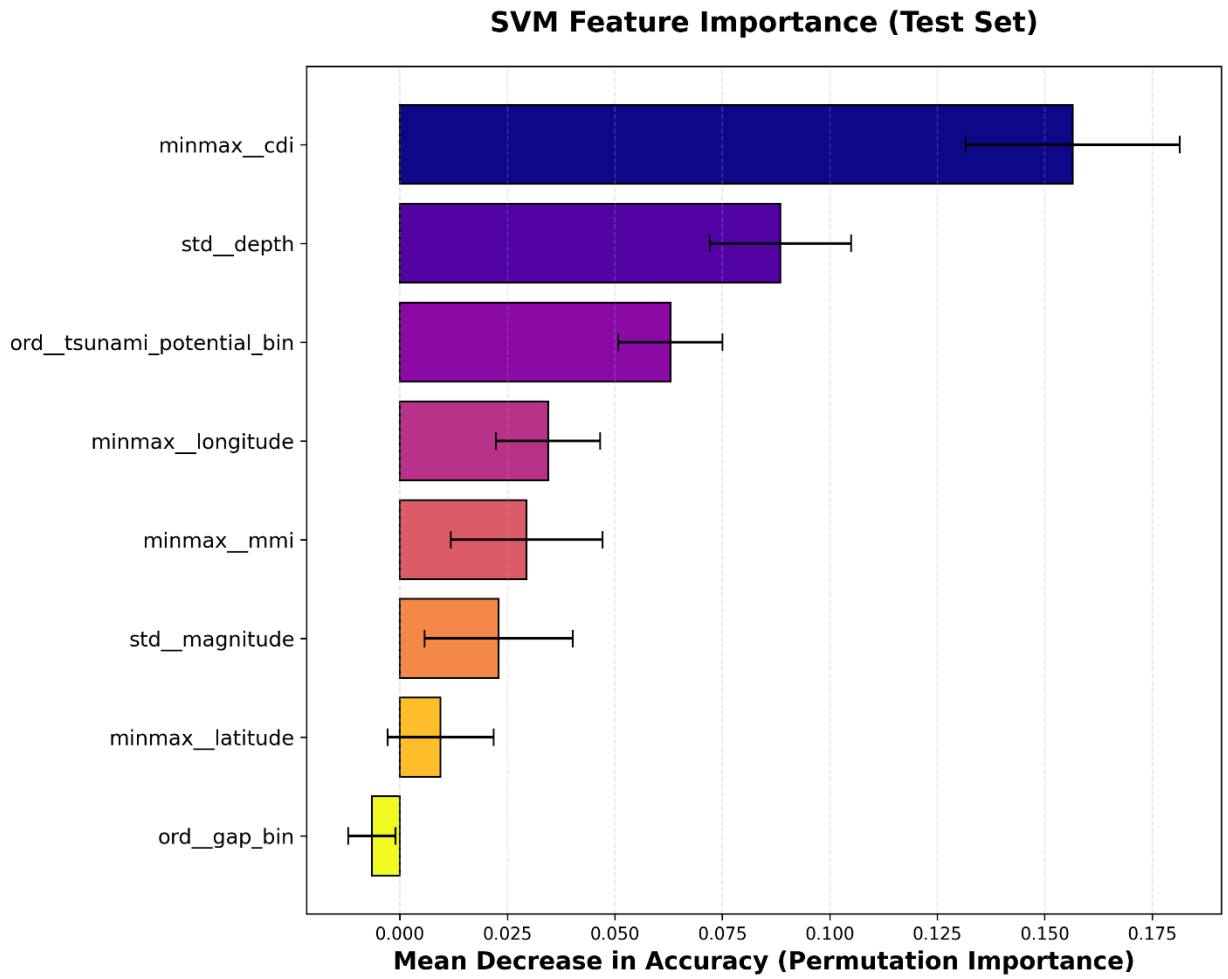


*Figure 12: SVM Feature Importance*

**Key Interpretations:**

- **Most Important:** cdi and mmi are the most influential features. These shaking intensity measures capture the felt impact of earthquakes at the surface, which directly correlates with the vertical seafloor displacement necessary to generate tsunamis — earthquakes that are strongly felt at the surface are more likely to displace water.
- **Geographic Dominance:** longitude and latitude rank among the top features, outperforming raw seismic measurements like magnitude. This reveals that location is a critical predictor — earthquakes occurring along specific tectonic boundaries (particularly in the Pacific Ring of Fire) have inherently higher tsunami probabilities regardless of magnitude.

# Outlook

While the XGBoost model demonstrates high performance, particularly with a Recall of 82.6% and an AUC of 0.876, several critical avenues for future improvement exist to enhance its reliability as a disaster warning system.

- **Data Scarcity:** The study is limited by a small sample size (~1,000 observations), which hinders generalizability and increases variance. Future research requires larger datasets (exceeding 100,000 earthquake-tsunami events) to ensure robust training and validation.
- **Precision Refinement:** Although the model effectively catches over 90% of tsunami events, the current Precision of 65.5% indicates a significant number of false alarms. It should prioritize reducing these false positives through collecting more oceanic data and the fine-tuning of decision thresholds to move Precision toward a target of 75% or higher.

# References

**National Centers for Environmental Information.** *Global Historical Tsunami Database, 2100 BC to Present*. National Oceanic and Atmospheric Administration, 2024, https://www.ncei.noaa.gov/access/metadata/landing-page/bin/iso?id=gov.noaa.ngdc.mgg.hazards:G01215. [3]

U.S. Geological Survey. *Earthquake Hazards Program: Technical Documentation*. U.S. Department of the Interior, 2024, https://www.usgs.gov/programs/earthquake-hazards/science/technical-documentation. [4]

**Chen, Tianqi, and Carlos Guestrin.** "XGBoost: A Scalable Tree Boosting System." *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785-794. *ACM Digital Library*, https://dl.acm.org/doi/10.1145/2939672.2939785. [5]

**Pedregosa, Fabian, et al.** "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research*, vol. 12, 2011, pp. 2825-2830, https://scikit-learn.org/stable/about.html. [6]

**Lundberg, Scott M., and Su-In Lee.** "A Unified Approach to Interpreting Model Predictions." *Advances in Neural Information Processing Systems 30*, 2017, https://shap.readthedocs.io/en/latest/index.html. [7]