Machine Learning Final Report: Earthquake-Triggered Tsunami Prediction

Group 19: 111550189, 112550054, 112550137, 112652021

Abstract

Tsunamis are rare but highly destructive natural disasters that can cause severe loss of life and extensive damage to coastal infrastructure. Because tsunami waves can reach shorelines within minutes after an earthquake, early warning systems must operate under extremely tight time constraints. As a result, there is a strong need for fast screening methods that can assess tsunami risk immediately after an earthquake occurs.

In this project, we study a binary classification problem that aims to predict tsunami occurrence using basic seismic features available after an earthquake. We use a global earthquake dataset covering events from 2001 to 2022 and evaluate several classical machine learning models, including Logistic Regression, Support Vector Machines (SVM), Random Forests, and Gradient Boosting.

A key challenge we identify is a strong temporal label distribution shift caused by missing tsunami records in early years. We show that a naive year-based data split leads to unstable performance and poor ranking ability, as reflected by low ROC–AUC scores. By adopting a randomly stratified split with appropriate preprocessing, we obtain more stable and reliable results.

Among the evaluated models, Gradient Boosting achieves the best overall generalization, while SVM and Logistic Regression achieve the highest recall. Our results highlight the importance of data preprocessing, evaluation strategy, and metric selection when applying machine learning methods to disaster prediction tasks.

1. Dataset Description

We used the **Global Earthquake–Tsunami Risk Assessment Dataset**, which is available on Kaggle. The dataset contains earthquake events recorded worldwide between **2001 and 2022**, with a total of **782 samples**. Each sample corresponds to one earthquake event.

1.1. Target Variable

- 1: the earthquake generated a tsunami
- 0: no tsunami occurred.

The dataset is imbalanced, with 38.9% tsunami events and 61.1% non-tsunami events.

1.2. Input Features

We use 10 numeric seismic features, which can be grouped into three categories:

- Magnitude and intensity-related features: magnitude, cdi, mmi, sig, nst
- Geometry and distance-related features: dmin, gap, depth
- Location features: latitude, longitude

The temporal features (Year, Month) are excluded from the input features of our models after performing the year-based split experiments.

1.3. Features Correlation

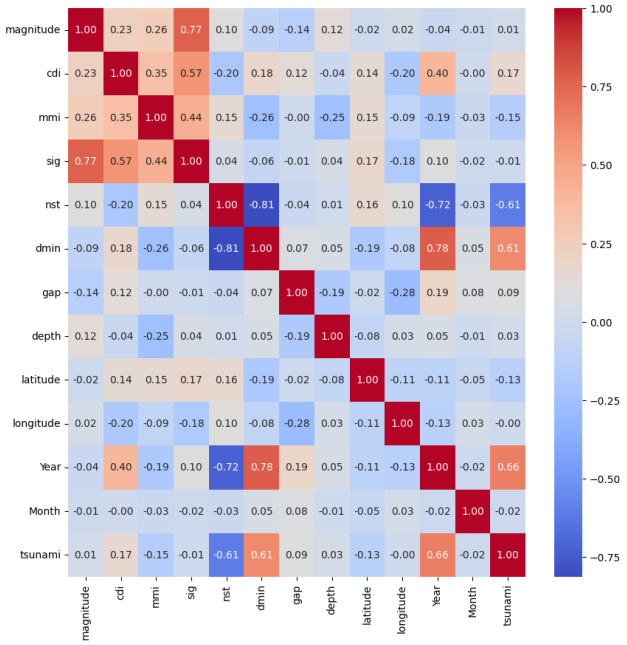

Figure 1. Feature correlation matrix

Figure 1 illustrates the correlation matrix among the input features and the target tsunami variable. We observed

that the ”Year” feature exhibits the strongest positive correlation with tsunami occurrence, followed by the distance to the nearest seismic station (dmin). In contrast, the number of seismic monitoring stations (nst) shows the strongest negative correlation with tsunami occurrence.

Additionally, several pairs of input features display strong correlations. For example, earthquake magnitude and event significance (sig) are highly positively correlated, while ”dmin” and ”nst” exhibit a strong negative relationship.

2. Data Preprocessing

2.1. Missing Values

We inspected the dataset using summary statistics and confirmed that no missing values are present in any column. Therefore, no imputation or row removal was required, and all models are trained on the full set of 782 events.

2.2. Feature Selection

The original dataset contains 13 columns: 10 numeric seismic features, two temporal columns (Year, Month), and the target label tsunami. We use all 10 numeric features as input variables and treat tsunami as the binary target. We exclude Year and Month from the final feature set because they do not represent physical earthquake properties and may introduce temporal bias, given the incomplete tsunami records in early years, as seen in Figure 2.

2.3. Feature Scaling

Since all 10 input features are numeric and have different scales, we applied **z-score standardization using StandardScaler**. The scaler is fitted only on the training data and then applied to the validation and test sets to avoid data leakage. This preprocessing step is especially important for distance-based and margin-based models such as Logistic Regression and SVM.

3. Data Splitting Strategy

3.1. Year-Based Split Experiment

We first conducted a year-based split to mimic a realistic deployment scenario:

- Training set: 2001–2016
- Validation set: 2017–2019
- Test set: 2020–2022

The same 10 numeric features were used, excluding Year and Month.

However, this split revealed a severe class imbalance across time. In the training period, tsunami events account for only **24.6%** of samples, while in the validation and test periods, tsunami events dominate at approximately **70–75%**.

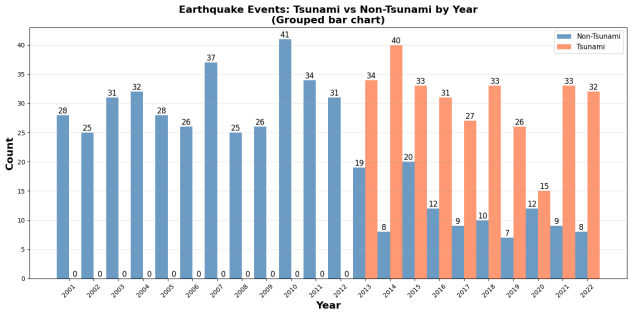

Figure 2. Earthquake Events: Tsunami vs Non-Tsunami by Year

Using Random Forest as an example, the year-based split model achieved a test accuracy of around **0.70** and an F1-score of **0.80**, but the **ROC-AUC dropped to 0.67**, which is relatively low. This made us suspect that something might be wrong with the data distribution over time.

3.2. Class Distribution by Year

To understand this behavior, we analyzed the class distribution by year. Figure 2 shows that **before approximately 2013, there are almost no tsunami records in the dataset**, even though tsunamis may have occurred in reality. If we use data from earlier years as train dataset and data from recent years as val/test datasets, then the model would not see the features of a tsunami outcome. This leads to a strong label distribution shift between early and later years, which explains the unstable performance observed under the year-based split.

3.3. Switching to Randomly Stratified Split

Because of this issue, we do not use the year-based split as our main evaluation setting. Instead, we adopt a randomly stratified 70/15/15 split for training, validation, and testing. Stratification ensures consistent class proportions across all splits, enabling fair model comparison and more stable evaluation.

4. Model Implementation

We evaluated four classical machine learning models: Logistic Regression, Support Vector Machine, Random Forest, and Gradient Boosting. The models are tuned viewing recall as the first priority and the precision as the second priority.

4.1. Logistic Regression

Logistic Regression serves as a linear baseline. We first trained a model without regularization, which achieved a test accuracy of 0.79 and a test recall of 0.87. Examining the confusion matrix, we can see that while most positive samples are detected, the model produces a considerable

number of false positives, resulting in a high recall with low precision.

Then, we applied a grid search with 5-fold stratified cross-validation, optimizing for F1-score. The F1-score was chosen because it provides a balanced measure of precision and recall, which is important for this classification task.

The parameter C in logistic regression is the inverse of the regularization strength; therefore, smaller values of C correspond to stronger regularization. In general, we observed that smaller C values led to improved model performance, suggesting that regularization helps reduce overfitting in this data set.

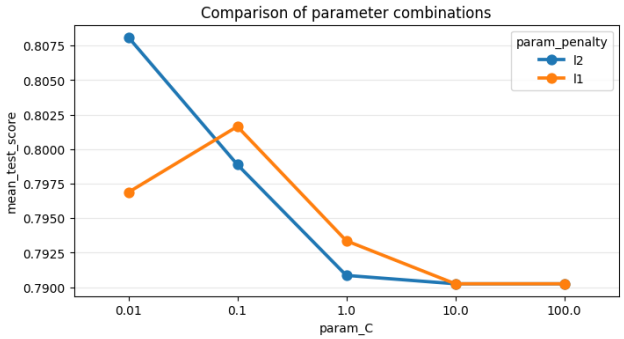

Figure 3. Logistic Regression: Cross-Validation F1-score for different parameters

The best hyperparameter combination identified by the grid search was **L2 penalty with C=0.01**. L2 performs better than L1 because it shrinks model coefficients more evenly, resulting in a more stable model.

Compared to the baseline, the regularized model shows a slight improvement in recall, although precision remains relatively low. Overall, the improvement was limited, which is likely due to the limitation of logistic regression to linear decision boundaries, causing it to struggle with more complex data patterns.

4.2. Support Vector Machine (SVM)

Next, we evaluated a Support Vector Machine (SVM) classifier using the `SVC` implementation from *scikit-learn*. We employed the default radial basis function (RBF) kernel, which is widely used when little prior knowledge about the data distribution is available. The RBF kernel maps the input data into a higher-dimensional feature space, enabling the model to learn complex, non-linear decision boundaries.

The key parameters of the SVM are γ and C. The parameter γ controls the radius of influence of individual training samples, where smaller values lead to smoother and simpler decision boundaries. C is the regularization parameter that balances the margin width and classification errors; lower values of C correspond to a wider margin with stronger regularization.

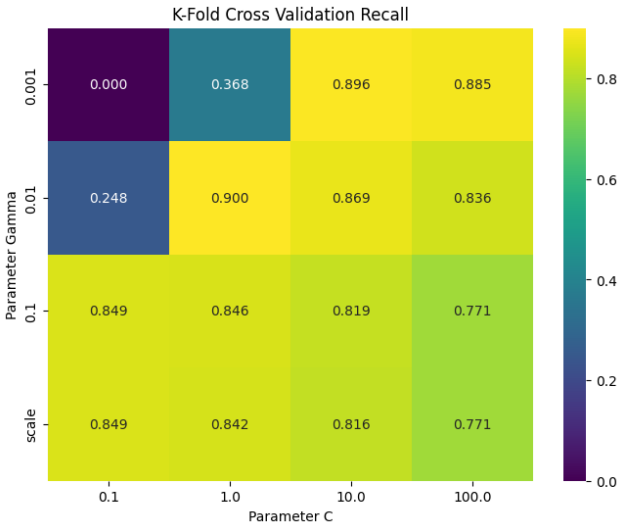

Figure 4. SVM: Cross-Validation Recall Score for different parameters

We explored a range of values for C and γ using GridSearchCV with 5-fold cross-validation, selecting recall as the primary evaluation metric. The optimal configuration identified was **C=1 and γ =0.01**. Under this setting, the model achieved a test recall of 0.89 and a ROC–AUC score of 0.7965. The confusion matrix shows that the tuned SVM successfully identifies most tsunami events, albeit at the cost of a relatively high number of false positives.

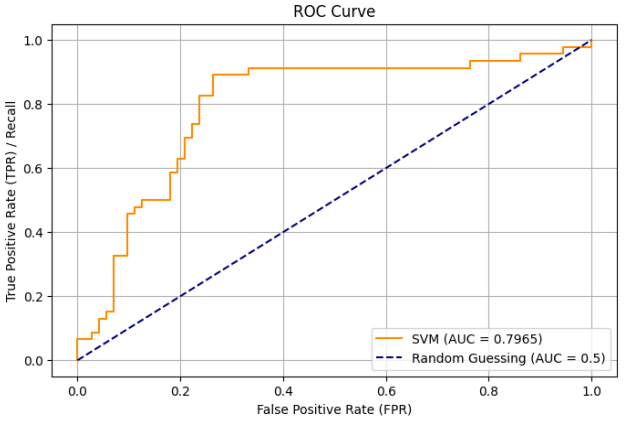

Figure 5. SVM Model: ROC Curve

4.3. Random Forest

Random Forest is an ensemble learning technique that integrates several decision trees that are trained independently. We initially evaluated the model using the default hyperparameter settings, while fixing the random state and enabling balanced class weights.

Surprisingly, this model was significantly overfitting, attaining 100% performance across all metrics on the training dataset, as illustrated in Figure 7. However, this level of performance did not translate well to the validation and test datasets.

To address this issue, we applied 5-fold cross-validation method with recall as the scoring metric. In addition, we adjusted the number of trees (n_estimators) and the maximum depth of each tree (max_depth).

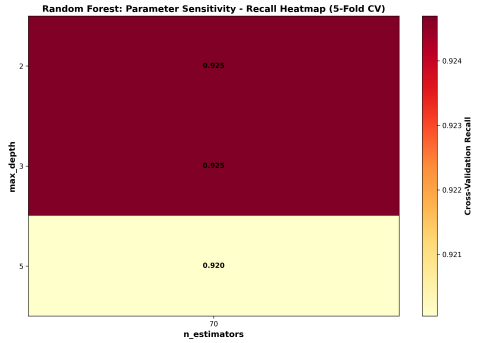

Figure 6. Random Forest: Cross-Validation Recall Score for different parameters

Figure 6 shows the cross-validation recall with respect to different parameters, while the final settings are chosen according to the highest recall on validation set. The modified settings **reduced the number of trees from 100 (default) to 70 and constrained the tree depth to 5**. These constraints effectively reduced model complexity and prevented individual trees from memorizing the training data.

Consequently, the resulting model generalized much better and achieved robust test performance across all evaluation metrics. This highlights the importance of carefully controlling model capacity when applying ensemble tree-based methods to relatively small datasets.

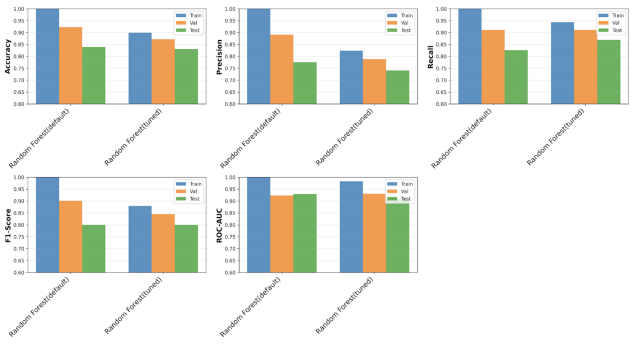

Figure 7. Random forest model: Metric scores for train/val/test data before and after tuning

4.4. Gradient Boosting

Gradient Boosting is a tree-based ensemble technique that combine models sequentially. Each newly added tree corrects the mistakes made by the previous ones, which allows Gradient Boosting to learn complex patterns. However, careful hyperparameter tuning and regularization are required to achieve optimal performance.

We implemented the same training and tuning process that was applied to Random Forest, which included the use of GridSearchCV with 5-fold cross-validation, once again emphasizing recall.

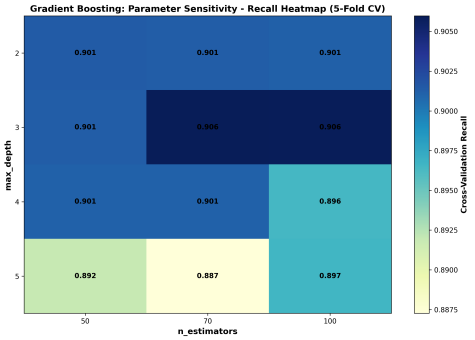

Figure 8. Gradient Boosting: Cross-Validation Recall Score for different parameters

As shown in Figure 8, increasing the maximum tree depth and the number of estimators generally does not improve the cross-validation recall score, as it can lead to overfitting by making the model overly complex. Based on the validation recall, the final model configuration reduced **the number of boosting stages (n_estimators) from 100 (default) to 50, and set the maximum depth of the individual trees to 3**. Given the relatively small size of the dataset, these settings help control model complexity and prevent overfitting.

As illustrated in Table 1, Gradient Boosting attains the highest ROC-AUC score of 0.93 and the best F1-score of 0.83 among all the models evaluated.

5. Results and Discussion

Across all models, we observed that:

- SVM and Logistic Regression has identical confusion matrices; however, SVM has a higher ROC-AUC score, which indicates that the SVM model can perform better at distinguishing the tsunamis across different thresholds.
- SVM and Logistic Regression achieved the highest recall, making them effective at detecting tsunami events. However, they produce more false positives than the tree-based models.
- Gradient Boosting generalizes best overall, achieving the highest ROC-AUC and strong F1-score. It also provides

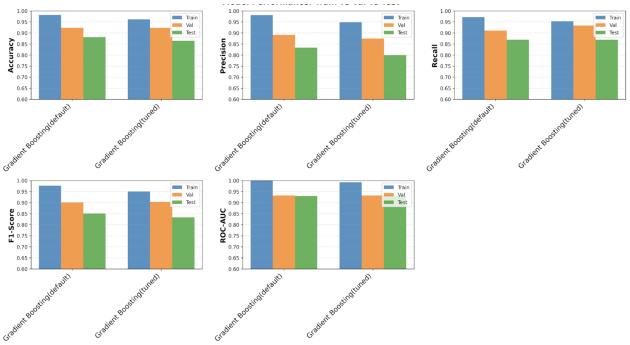

Figure 9. Gradient boosting model: Metric scores for train/val/test data

better discrimination between tsunami and non-tsunami events, resulting in fewer misclassifications overall. This is likely due to the sequential nature of Gradient Boosting, allowing it to capture non-linear and complex relationships, producing accurate results that outperform the other three models.

	Logistic Regression	SVM	Random Forest	Gradient Boosting
Accuracy	0.788136	0.788136	0.830508	0.864407
Precision	0.672131	0.672131	0.740741	0.800000
Recall	0.891304	0.891304	0.869565	0.869565
F1-score	0.766355	0.766355	0.800000	0.833333
ROC-AUC	0.777174	0.796498	0.907005	0.930254

Table 1. Evaluation metrics results of all models

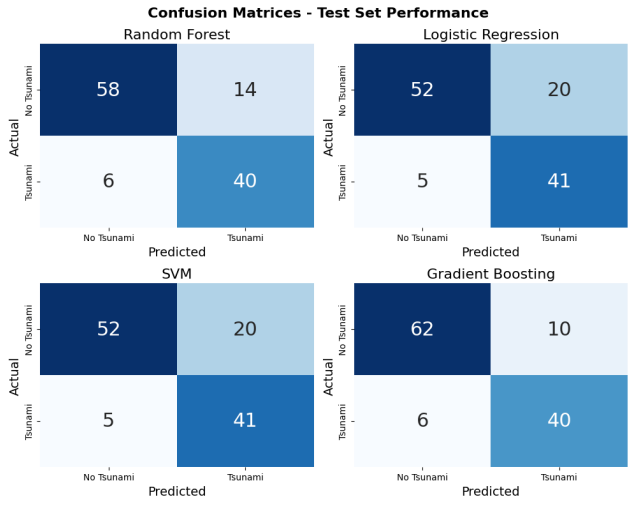

Figure 10. Confusion matrices of all models

6. Conclusion and Future Work

In this project, we studied the tsunami prediction binary classification problem with classic machine learning models. We demonstrated that preprocessing and data splitting strategy play a crucial role in model performance. A naive year-based split leads to severe label distribution shift and unstable evaluation, while a randomly stratified split provides more reliable results. This project also highlights the importance of using multiple evaluation metrics and prioritizing recall in disaster prediction tasks.

For future work, we plan to:

- Explore more advanced hyperparameter tuning strategies
- Experiment with more sophisticated models such as XGBoost or neural networks
- Perform feature engineering
- Collect more tsunami events to reduce the imbalance of dataset.

References

https://www.kaggle.com/datasets/ahmeduzaki/global-earthquake-tsunami-risk-assessment-dataset/data