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. Since false negatives (predicted no-tsunami but actual tsunami) are more costly than false positives, we view recall as a priority over precision and accuracy.

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 Logistic Regression achieves 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

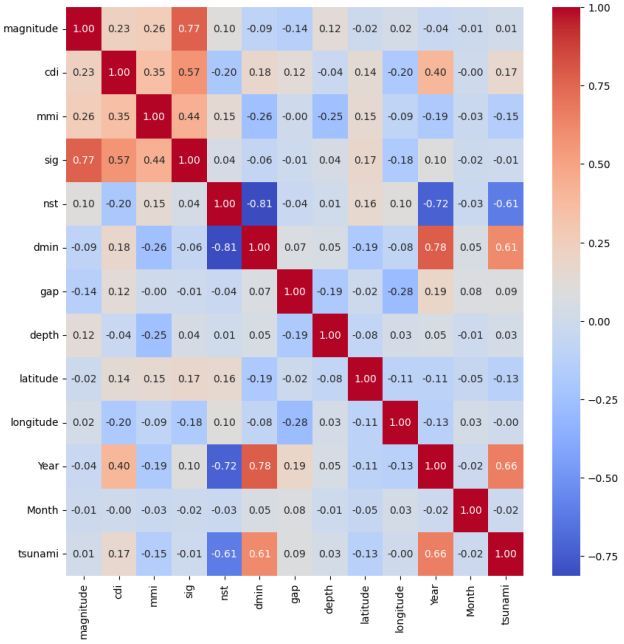

Figure 1. Feature correlation matrix

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.

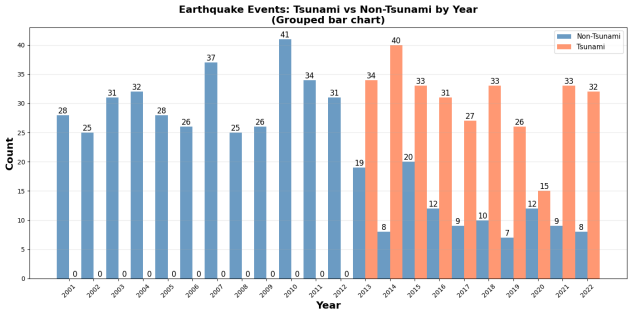

Figure 2. Earthquake Events: Tsunami vs Non-Tsunami by Year

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%**.

3.2. Class Distribution by Year

To understand this imbalanced 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 validation or 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 80/20 training and testing set split**. We did not include a validation set because cross-validation is applied for selecting the best hyperparameters in all models.

In addition, 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. For all models, a grid search with 5-fold stratified cross-validation is applied to find the best combination of the specified parameters. We chose F1-score as the scoring metric, since it provides a balanced measure of precision and recall, which is more indicative than accuracy for this problem.

4.1. Logistic Regression

Logistic Regression serves as a linear baseline. We first trained a model without regularization, which achieved a test accuracy of 0.8153 and a test recall of 0.8852. 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.

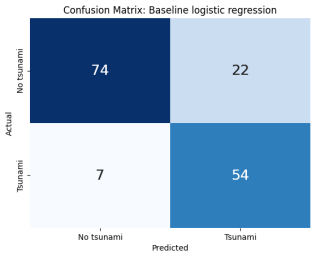

Figure 3. Confusion matrix of Baseline Logistic Regression Model

Then, a 5-fold cross-validation is applied on different values of the parameter C with L1 and L2 penalty, respectively.

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.

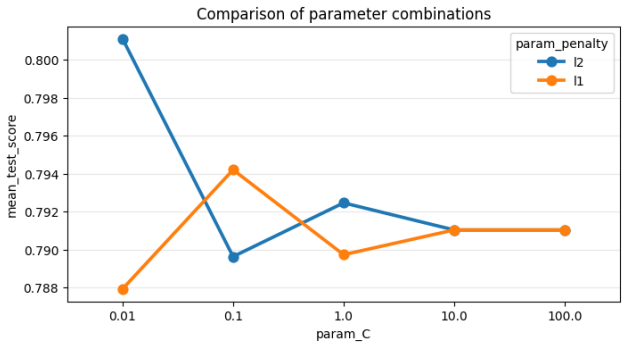

Figure 4. Logistic Regression: Cross-Validation F1-score for different parameters

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 showed a slight improvement in recall, although precision remains relatively low. Overall, the improvement is 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**. The default L2 penalty is used for this experiment.

We explored a range of values for C and γ with 5-fold cross-validation, F1-score selected as the evaluation metric. In general, a smaller γ value is the main contributor to higher cross-validation scores, while the regularization parameter has a limited influence on the SVC model.

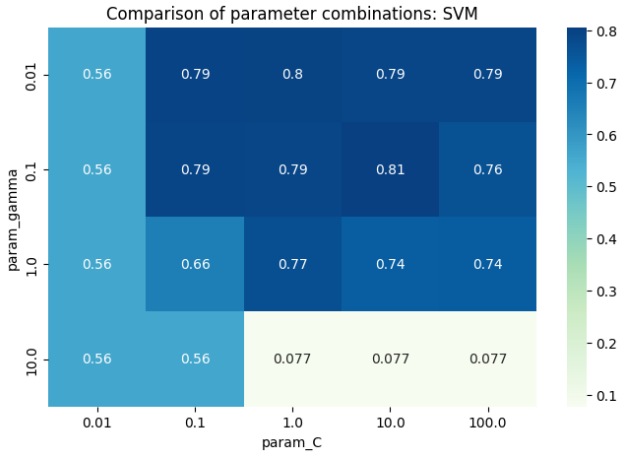

Figure 5. SVM: Cross-Validation F1-Score for different parameters

The optimal configuration identified was **C=10 and γ =0.1** with the best score of 0.8058. Under this setting, the model achieved a test recall of 0.8852 and an ROC–AUC score of 0.91825. In Figure 11, the confusion matrix of the tuned SVM has noticeably fewer false positives than logistic regression, albeit at the cost of slightly more false negatives.

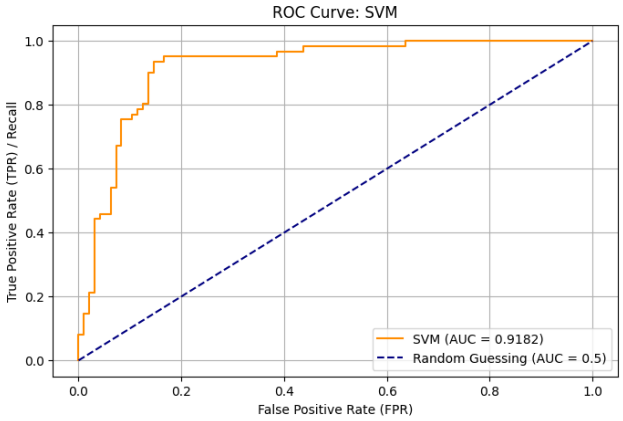

Figure 6. SVM: ROC Curve

4.3. Random Forest

Random Forest is an ensemble learning technique that integrates several decision trees trained independently with a random selection of data points and features. Compared to a single decision tree, random forests are more robust to overfitting as each individual tree captures different perspectives of the data. However, it can also result in overly complex models for relatively small datasets.

Similarly, we applied a 5-fold cross-validation method with F1-score as the evaluation metric to investigate how the number of estimators (trees) and the maximum depth of each tree can affect performance.

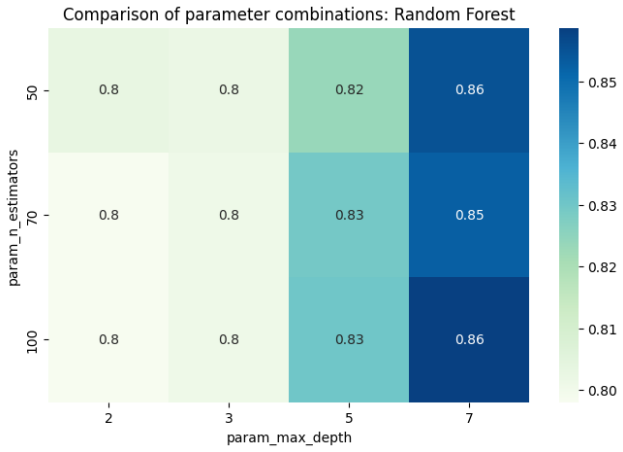

Figure 7. Random Forest: Cross-Validation F1-Score for different parameters

Figure 7 illustrates that increasing the maximum depth has a greater impact on the cross-validation score, whereas the number of estimators only improve the model performance slightly.

The optimal settings found for the random forest model were **100 estimators with a maximum tree depth of 7**. On the test set, this model achieved a recall of 0.9016 and an AUC score of 0.9366.

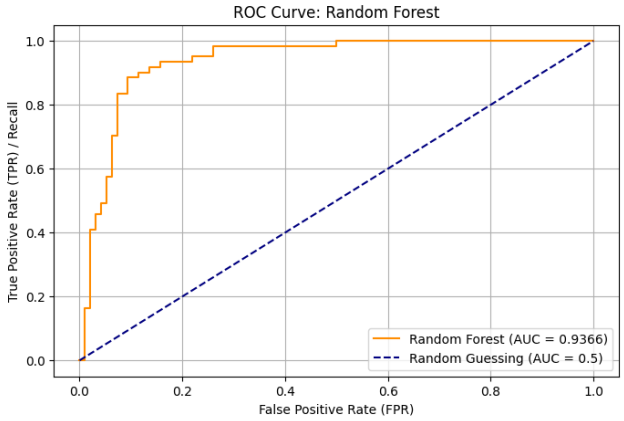

Figure 8. Random Forest: ROC Curve

4.4. Gradient Boosting

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

We implemented the same training and tuning process, which included the use of GridSearchCV with 5-fold cross-validation, once again emphasizing F1-score.

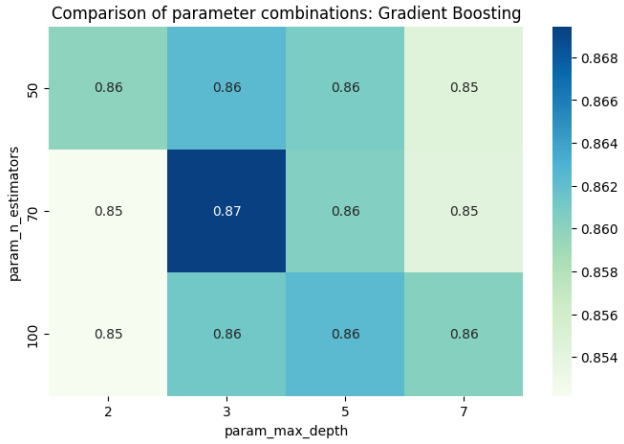

Figure 9. Gradient Boosting: Cross-Validation Recall Score for different parameters

From the result of Figure 9, increasing the maximum tree depth and the number of estimators does not improve the cross-validation recall score, as it can lead to overfitting by making the model overly complex. The opti-

mal model configuration reduced **the number of boosting stages (n_estimators) to 70, and the maximum depth to 3**. Given the relatively small size of the dataset, these constraints help control the model complexity and prevent overfitting.

As illustrated in Table 1, Gradient Boosting attains the highest ROC-AUC score of 0.95 and the best F1-score of 0.88 among all the models evaluated. Figure 10 reveals that the ROC curve for gradient boosting is the closest to the top-left compared to the other three models. A possible further implementation is to adjust the decision threshold value with respect to the ROC curve.

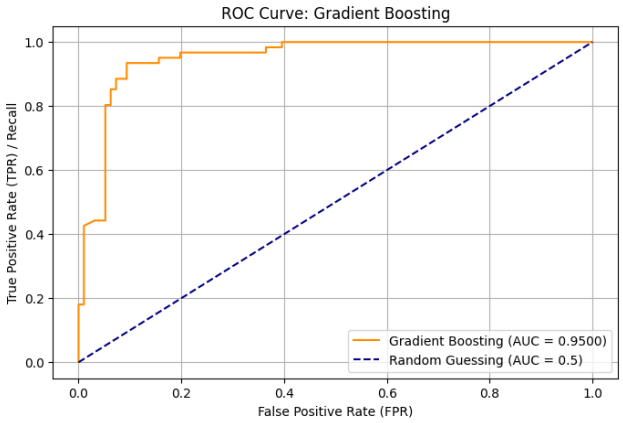

Figure 10. Gradient Boosting: ROC Curve

5. Results and Discussion

In the context of earthquake-triggered tsunami prediction, false negatives (predicted no-tsunami but an actual tsunami occurred) are more costly than false positives. Thus, we prioritize recall and F1-score over precision and accuracy during model evaluation. Across all models, we observed that:

- Logistic Regression achieved the highest recall, making it effective at detecting tsunami events. However, it produced more false positives than SVM and tree-based models, and has the lowest scores for all other evaluation metrics.
- Gradient Boosting generalizes best overall, achieving the highest evaluation scores except for recall. It also provides better discrimination between tsunami and non-tsunami events with an AUC-score of 0.95, 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.
- Random Forest is the second most effective model after Gradient Boosting in terms of precision, F1-score and ROC-AUC score. It is also worth noticing that it has slightly higher recall than Gradient Boosting, making it a suitable choice for the problem.

	Logistic Regression	SVM	Random Forest	Gradient Boosting
Accuracy	0.821656	0.872611	0.891720	**0.904459**
Precision	0.708861	0.805970	0.833333	**0.870968**
Recall	**0.918033**	0.885246	0.901639	0.886246
F1-score	0.800000	0.843750	0.866142	**0.878049**
ROC-AUC	0.841872	0.918204	0.936646	**0.949966**

Table 1. Evaluation metrics results of all models

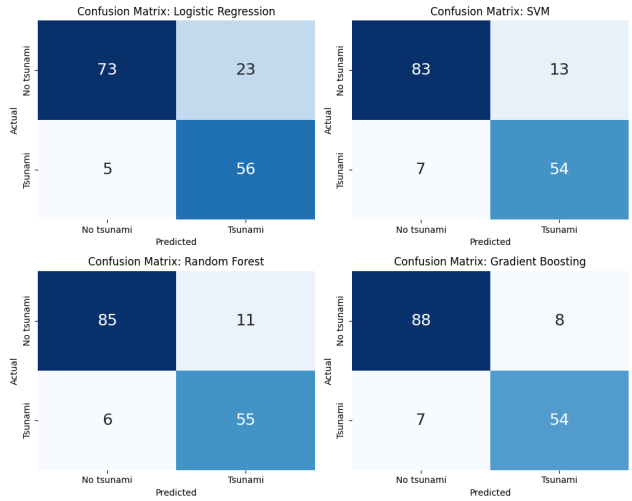

Figure 11. Confusion matrices of all models

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.

In addition, we investigated how various parameters in machine learning models can affect the overall performance. Among the four models implemented, gradient boosting generalized the best, while random forest can also be a reasonable choice with higher recall. This project highlights the importance of using multiple evaluation metrics and prioritizing recall in disaster prediction tasks.

For future work, we plan to:

- Experiment with more sophisticated models such as XGBoost or neural networks

- Perform feature engineering
- Explore more advanced hyperparameter tuning strategies
- Collect more tsunami events to reduce the imbalance of dataset.

7. References

- https://www.kaggle.com/datasets/ahmeduzaki/global-earthquake-tsunami-risk-assessment-dataset/data
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html