DATS 6202
GROUP# 1

TSUNAMI PREDICTION USING MACHINE LEARNING TECHNIQUES

—

Jack McMorrow, Alex Khater, Alejandra Mejia

TABLE OF CONTENT

Introduction **3**
Literature review **4**
Description of the dataset **5**
Data Overview 5
Data Dictionary 5
Data Cleaning 6
Exploratory Data Analysis 6
Considerations for model selection 11
Alignment with project goal 11
Dataset size 11
Dimensionality 11
Inference time 11
Machine Learning Network and training algorithm **12**
Experimental setup **12**
XGBoost 12
Additional Models 13
MLP 13
KNN 14
SVM 14
Random Forest 15
AdaBoost 16
Results **17**
XGBoost 17
MLP 19
KNN 19
SVM 21
Random Forest 22
AdaBoost 23
Summary and conclusions **24**
Limitations **25**
References **26**

Introduction

Although tsunamis are generally infrequent, their unpredictable nature makes them a potentially devastating natural hazard. Being able to develop accurate methods for tsunami prediction quickly could become key to saving lives. Recent research has focused on the use of artificial intelligence (AI) algorithms (Cardiff University) and deep-learning models (Los Alamos National Laboratory) combined with real-time data.

However, the use of machine learning algorithms in this field still seems novel and a work in progress. In this project, we aimed to add to the current research by exploring seismic research dataset which contains earthquake information for the past 22 years. We used this data to develop a diverse classification model that can have the capability to predict whether or not an earthquake poses a significant risk of a tsunami. Our approach will focus on using historical data to test such predictions using six different machine learning models and compare them based on various accuracy metrics.

The report is structured in eight sections:

1. The first section provides a literature review of the existing efforts in using machine learning for natural disasters predictions.
2. Section two provides a description of the dataset, including a detailed data dictionary and an exploratory data analysis to help us understand the nature of the data before we tackle the machine learning problems.
3. Section three discusses the background information of the machine learning algorithms selected.
4. The fourth section deep dives in the implementation of the selected machine learning techniques previously discussed in section three.
5. Section five provides insight into the results obtained from each of the techniques implemented and provides a comparison between each technique performance.
6. The sixth section provides a summary of the results, lessons learned, and future improvements that would be necessary to enhance the prediction power of the algorithms.
7. Finally, the last sections provide references and appendices.

Literature review

Due to the catastrophic impact of tsunamis, there has long been interest in predicting whether or not a tsunami will follow an earthquake in order for people to prepare for possible damage. Consequently, there has been extensive research on utilizing machine learning. While our dataset is relatively simple when compared to other seismographic datasets, our goal of predicting tsunami's seeks to use simpler earthquake data to make predictions. Many researchers have put more complicated datasets and models in service to this goal.

Two major issues that have plagued the task of tsunami prediction have been the complexity of collecting and interpreting seismic data and the corresponding complexity of the models. In order to ensure high accuracy, data collection and handling can be quite difficult and expensive, as useful data often requires undersea data collection with complex instruments. Although high accuracy in tsunami prediction is expensive and arduous this has not stopped major breakthroughs in the field. A major study on the topic was released in December of 2022.

In the aftermath of the 2011 Tōhoku earthquake and tsunami in Japan, 150 undersea seismic research stations were built deep in the ocean around the Greater Japan Trench with the intention of acquiring data that could be used to predict the severity and onset of tsunamis. A group of researchers in association with the RIKEN Pioneering Project[1] published a paper in which they implemented a neural network model to use offshore/undersea physics-based data, seismographic techniques, including using Stochastic Slip Models, and then feeding into a 150 Neuron nonparametric regression model through TensorFlow to predict various factors about tsunami's to produce some astoundingly accurate results meant to supplement the tsunami early warning systems in Japan.

Due to the relative simplicity of our dataset and relative lack of domain expertise in comparison to this research team, we seek to create a simpler model and see how it uses simple metrics to create a predictive model for tsunami's following earthquakes as a comparison to the accuracy of the more in depth models of the previously discussed study.

[1] https://www.nature.com/articles/s41467-022-33253-5#Sec11

Description of the dataset

Data Overview

The dataset used in the current project was retrieved from Kaggle and contains data records of 782 different earthquakes. The amount of earthquakes recorded was large enough to perform machine learning techniques on. The variables contained in the database include more than 20 features and a binary tsunami variable that we will use as our target. The following section provides further details regarding the data dictionary.

Data Dictionary

Our dataset consisted of 18 variables detailed below:

1. Title: title name given to the earthquake
2. Magnitude: The magnitude of the earthquake
3. Date_time: date and time
4. cdi: The maximum reported intensity for the event range
5. mmi: The maximum estimated instrumental intensity for the event
6. alert: The alert level - “green”, “yellow”, “orange”, and “red”
7. tsunami: "1" for events in oceanic regions and "0" otherwise
8. sig: A number describing how significant the event is. Larger numbers indicate a more significant event. This value is determined on a number of factors, including: magnitude, maximum MMI, felt reports, and estimated impact
9. net: The ID of a data contributor. Identifies the network considered to be the preferred source of information for this event.
10. nst: The total number of seismic stations used to determine earthquake location.
11. dmin: Horizontal distance from the epicenter to the nearest station
12. gap: The largest azimuthal gap between azimuthally adjacent stations (in degrees). In general, the smaller this number, the more reliable the calculated horizontal position of the earthquake. Earthquake locations in which the azimuthal gap exceeds 180 degrees typically have large location and depth uncertainties
13. magType: The method or algorithm used to calculate the preferred magnitude for the event
14. depth: The depth where the earthquake begins to rupture
15. latitude / longitude: coordinate system by means of which the position or location of any place on Earth's surface can be determined and described

16. location: location within the country
17. continent: continent of the earthquake hit country
18. country: affected country

Prior to starting any analysis, some data cleaning was required in order to ensure that our machine learning methods would be able to produce robust results. The data cleaning process is detailed in the following section.

Data Cleaning

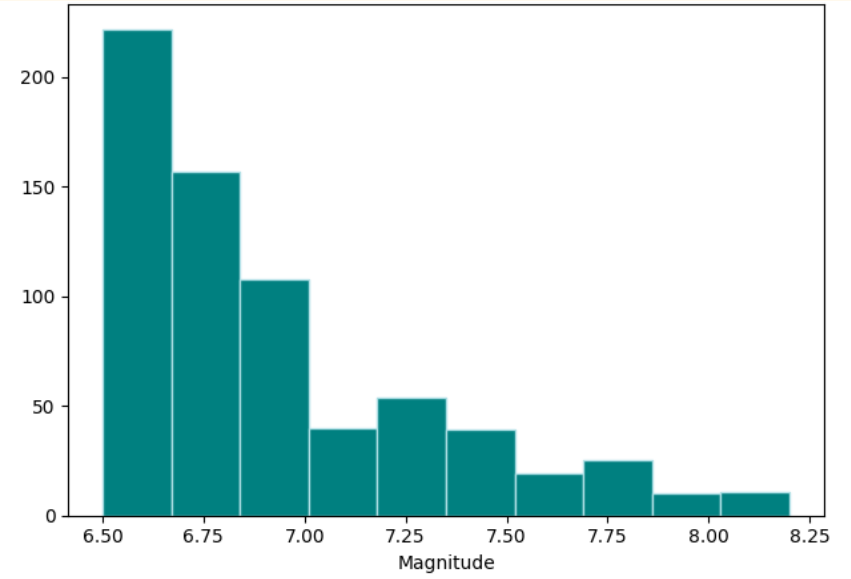
The data cleaning process consisted of three phases, the first one was focused on removing missing values and ensuring that the data did not have labeling mistakes regarding locations. Overall the dataset had missing values in four variables, alert (367), continent (576), country(298), location (5). For the first three variables the columns were dropped and a new variable called Country2 was created. Country2 variable used the information from the location variable after dropping the 5 NA values and identified the countries where the event happened . Second, geospatial information was validated in order to consider using these variables to provide further robustness. The third step consisted in cleaning the data based on the need of the models used for instance in the case of the multi-layer perceptron(MLP), K-Nearest Neighbors (KNN) and Support vector machines (SVM) data standardization for the magnitude was required. On the other hand, the Random Forest Model and AdaBoost additionally required replacing categorical variables by dummy variables.

Once the dataset was clean, an exploratory data analysis was performed in order to further understand the data and assess the modeling techniques that could be more appropriate to undertake. The following section provides the details.

Exploratory Data Analysis

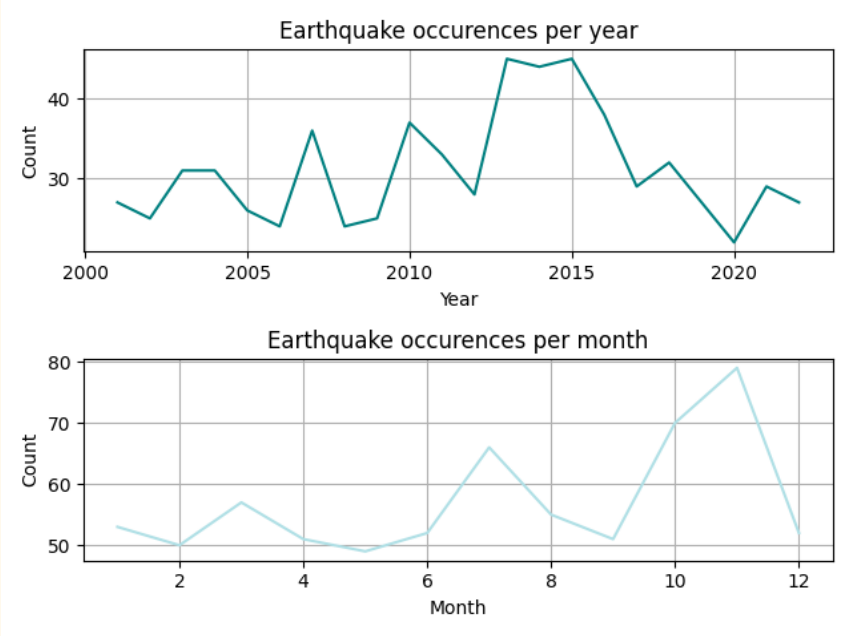
Given that our research question is focused on assessing the predictability of tsunamis based on earthquake occurrence, it became necessary to understand the distribution of earthquake magnitude in our sample. Especially, given that according to the National Weather Service, an earthquake must exceed magnitude 8.0 to generate a dangerous distant tsunami. From the data we were able to depict that the most common magnitude in earthquakes is actually 6.5 and only a few number of instances a magnitude greater than or equal to 8.0 has been observed (Figure 1).

Figure 1. Distribution of earthquake magnitude in the sample



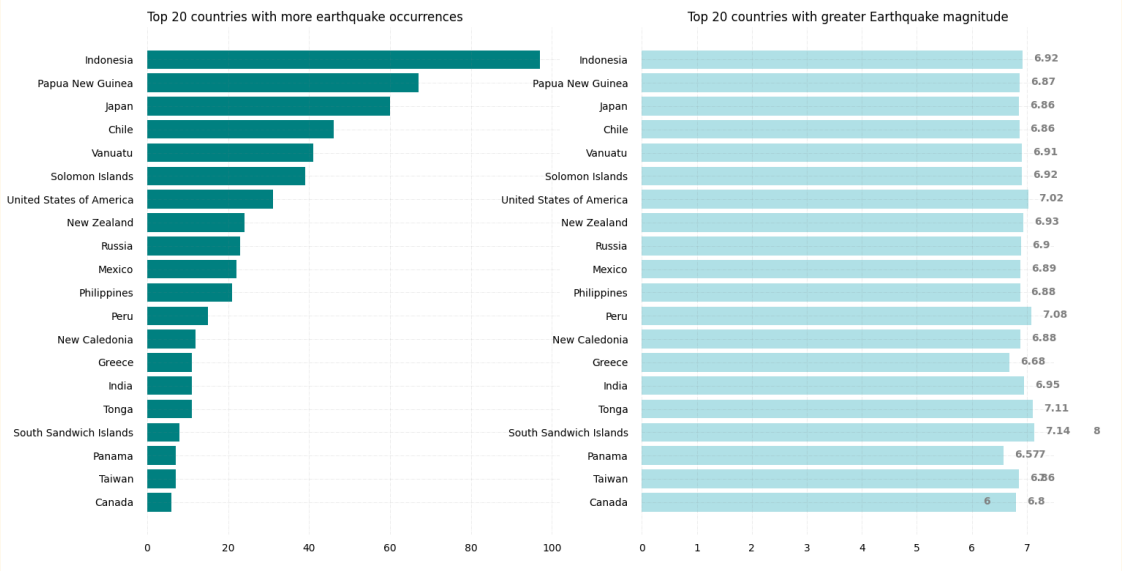
Similarly, we assess if the frequency of events followed a specific trend over the years. According to the available data the years 2013 and 2015 saw the most number of earthquakes, while the month of November seems to be the one with more recorded occurrences (Figure 2). This provides an interesting insight into adding variables in future research that could be related to specific environmental factors that occur during those years or in that month, like sea level rise.

Figure 2. Earthquakes occurrence per year and month



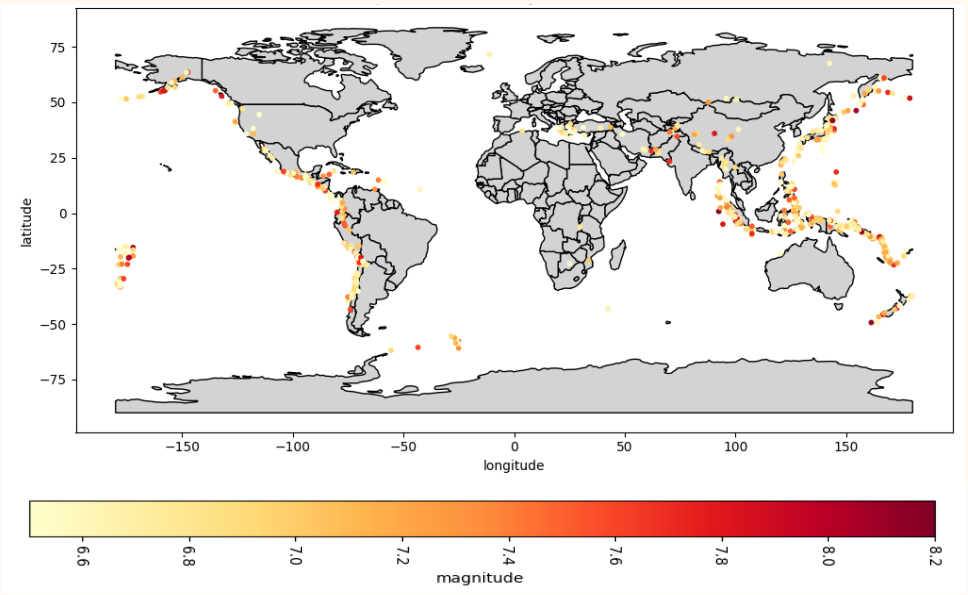
Moreover, our dataset included information on the location of the events. We could see from the data that the events are spread out in different continents, with Indonesia being the country with the highest number of earthquakes. However, it is relevant to notice that on average a higher number of earthquakes does not mean a higher magnitude.

Figure 3. Top 20 countries with more earthquakes and their magnitudes



Moreover, with the latitude and longitude data available we were able to map the reported earthquakes and be able to distinguish a pattern. A great majority of the earthquakes follow the path of the Ring of Fire, which is a path along the Pacific Ocean characterized by active volcanoes.

Figure 4. Earthquakes in the World Jan 2001 to Nov 2022

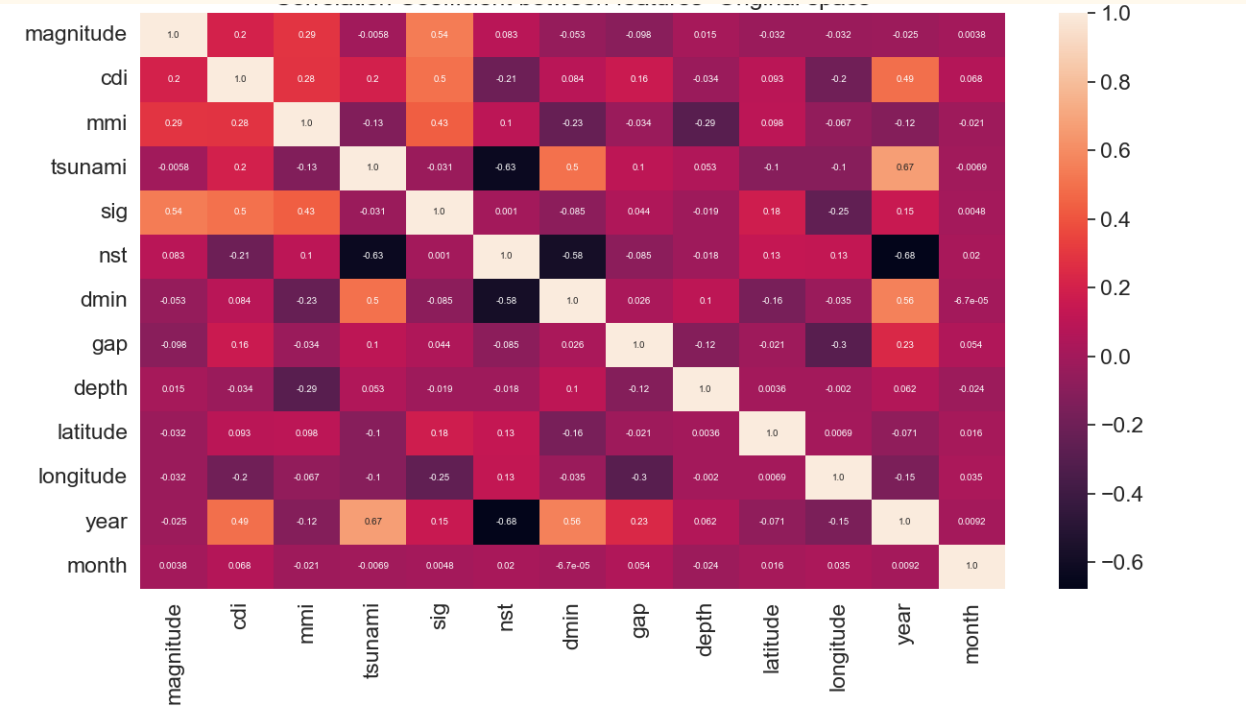


Additionally, we produced a pairplot that allowed us to understand the best set of features to explain the relationship between two variables and to determine how unbalanced our dataframe was. Specifically, given the low occurrence of tsunamis as previously mentioned, our dataset seems slightly unbalanced with 37 percent of tsunamis occurrences against 63 percent of occurrence with earthquakes but no tsunamis. Given this characteristic and that classification accuracy is almost universally inappropriate for imbalanced classification we will focus on sensitivity and precision metrics to better evaluate our models.

Figure 5. Pairplot

Moreover, in order to make sure our features were not correlated between them or with our target variable we produced a correlation matrix. We were able to confirm that the correlation was fairly low, even in variables that based on the dataset dictionary description could be perceived as highly correlated (eg. sig).

Figure 6. Correlation Matrix original space



In order to further confirm our results and make sure our dataset can be used for modeling we also performed some t-tests and some data normalization.

The t-test indicated non-significant p-values which indicated low levels of correlation. However, after performing some normality test we found that the data was non normal in all columns and required data transformation. We applied multiple forms of data normalization, ranging from using the sklearn.normalize function, to the mean and standard deviation normalization formula. Though the p-values were higher, many of our dataset's columns still failed our Shapiro-Wilks test for normality so we decided to carry on, noting this potential limitation. Even though random forest and Adaboost models do not require the normalization of the data, our MLP, SVM, KNN and additional models can benefit from this preprocessing to obtain robust predictions.

After pre-processing our dataset we also deep dive in four criterias in order to select the most appropriate model, which we specify below.

Considerations for model selection

Alignment with project goal

Diverse research points out that machine learning algorithms can handle multidimensional, large (ML) volumes of data usually occurring in thematics related to disaster and pandemic management (Camola et al, 2021). Recent developments in ML and deep learning have been used to better cope with the severe and often catastrophic impacts of disasters (Linardos et.al, 2022). Moreover, ML approaches seem to be more effective at handling complicated and nonlinear high-dimensional data sets (Kainthura, 2022).

Dataset size

The number of training data available is one of the main factors when considering a model. Therefore, given our dataset is not extremely big, a K-Nearest Neighbors (KNN) model seemed as one of the possible modeling methods appropriate for the dataset.

Dimensionality

Given that the number of features is not extremely abundant and that we have tested out multicollinearity, a high-dimensional datasets would not necessarily be a problem for our exercise and therefore we could build a neural network that does not have a high level of complexity.

Inference time

The longer the inference time, the greater the cost of running the model. Usually, KNN processing needed to develop predictions happens during inference time, making it more expensive than approaches like Decision Tree which have lower inference time and require more time during training. Support vector machines are simpler and faster to train.

Given the following considerations and wanting a model that provides the most robust result we have selected a XGBoost approach for our predictions and compared it with five machine learning approaches detailed in the following section.

Machine Learning Network and training algorithm

We intend to use six different Machine Learning Models for our binary classification problem: XGBoost, MLP, KNN, SVM, Random Forest, and Adaboost. Since this is a relatively simple binary classification problem, we can use confusion matrices and F1 scores to compare these models' performances on the data. For now we intend to use these models in standard form, but some modifications for our data may be necessary further in.

Experimental setup

XGBoost

We decided to implement XGBoost because of our data's non-normality and the power and efficiency of the XGBoost in binary classification problems. After processing and cleaning the data, we would apply two rounds of grid search CV to our model to optimize the hyperparameters (detailed below).

XGBoost is an ensemble learning method that takes the results of a series of decision trees built in parallel seeking to minimize the following loss function (Figure 7).

Figure 7. Loss function in XGBoost.

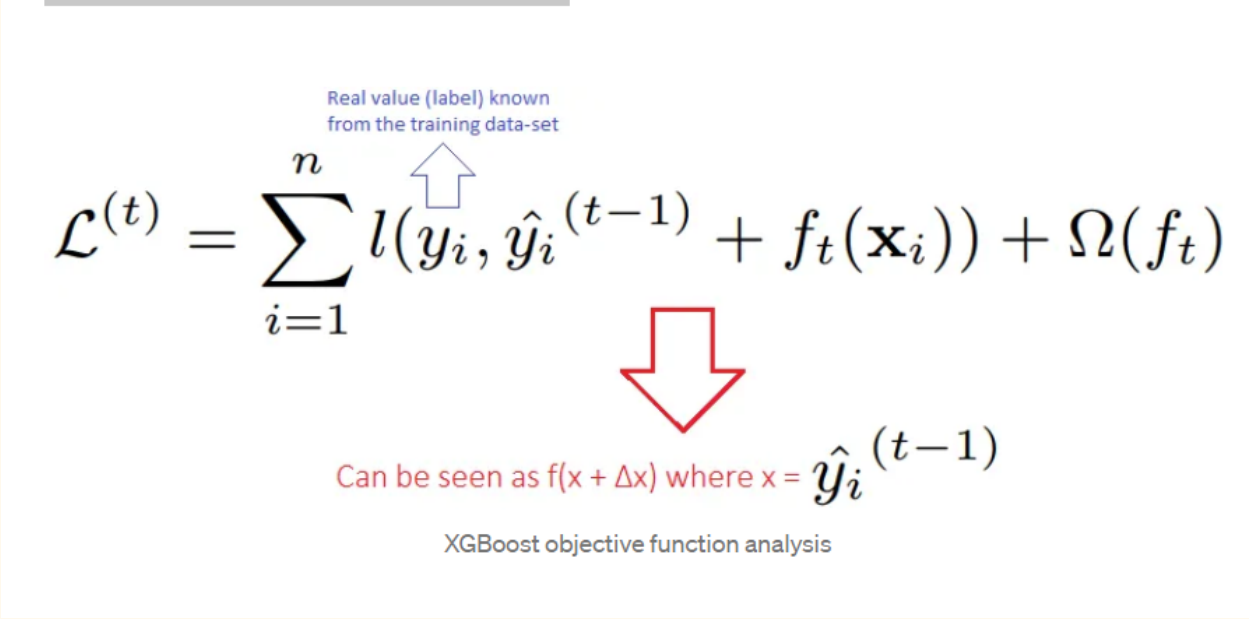


Image Source: https://dimleve.medium.com/xgboost-mathematics-explained-58262530904a

We can see that this loss function used in the XGBoost model involves combinations of the gradient descent algorithm and various decision tree models. More detailed information on the exact mathematics employed by the model can be found below in the footnotes[2]

Additional Models

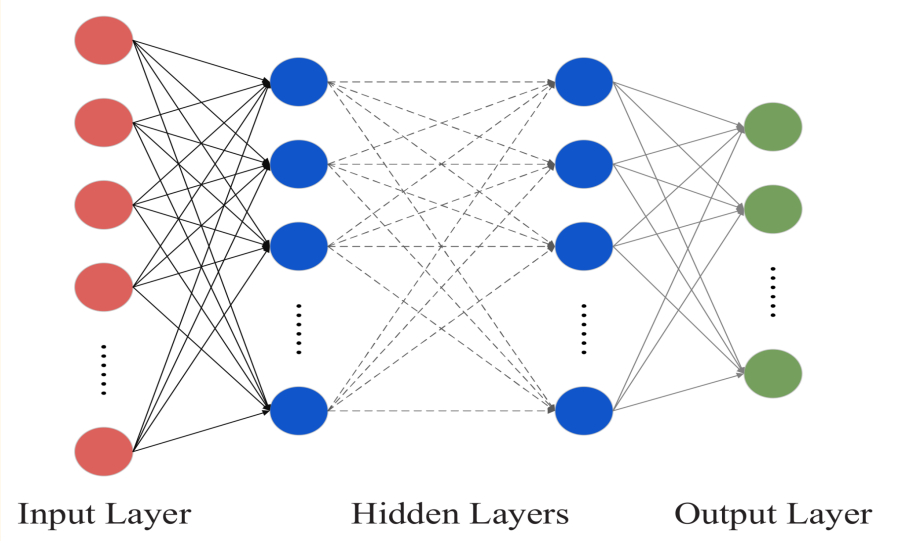
In order to evaluate the efficacy of the proposed NN method, we compared it with five classification approaches: MLP, KNN, SVM, Random Forest and Adaboost. Overall precision and recall were adopted to assess the classification performance of each model.

The aforementioned materials and networks architecture of the proposed methods will now be introduced in detail.

MLP

The Multilayer Perceptron is a widely used neural network that generally is very effective at classifying data. It uses a feedforward approach by passing data through layers of artificial neurons, each of which consists of their own weight, biases, and transfer functions. In this project we'll train different MLP classifiers with different optimization techniques to see how it performs when predicting tsunamis.

Figure 8. Layers of Neurons in a Multilayer Perceptron.



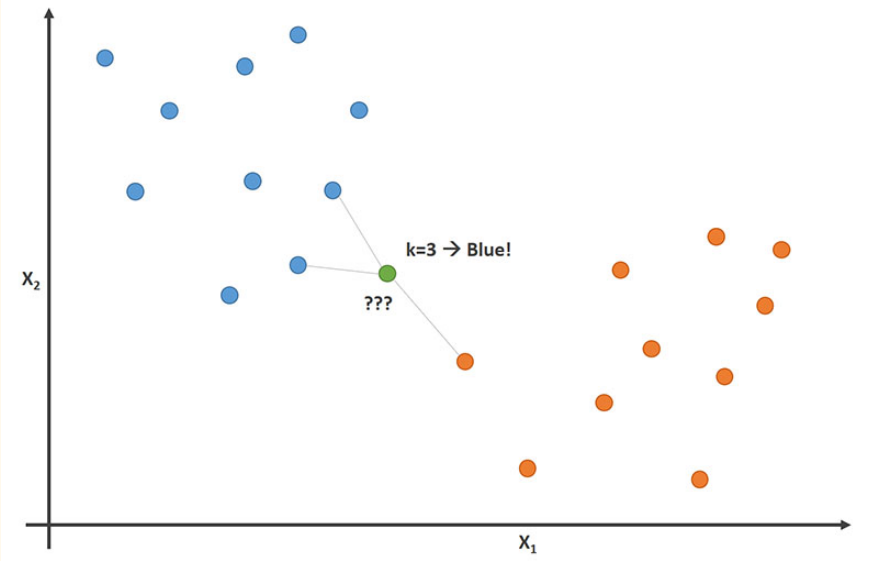
[3]

[2] https://dimleve.medium.com/xgboost-mathematics-explained-58262530904a
[3] https://www.mdpi.com/2072-4292/13/20/4060

KNN

K-nearest neighbors is a supervised machine learning algorithm that classifies input data by measuring its proximity to the training data. This can be done by using Euclidean distance, Manhattan distance, or other measurements. Given that it is a nonparametric method and simply memorizes the input data, it is considered a "lazy learner". The hyperparameter, *k,* denotes the number of neighboring training points that the algorithm uses to compare the input. For classification problems, it uses the highest number of a given class in *k* points to determine the output (Figure 9). In this problem, we'll use KNN to see if it is effective at classifying tsunamis.

Figure 9. Classification with K-Nearest Neighbors.

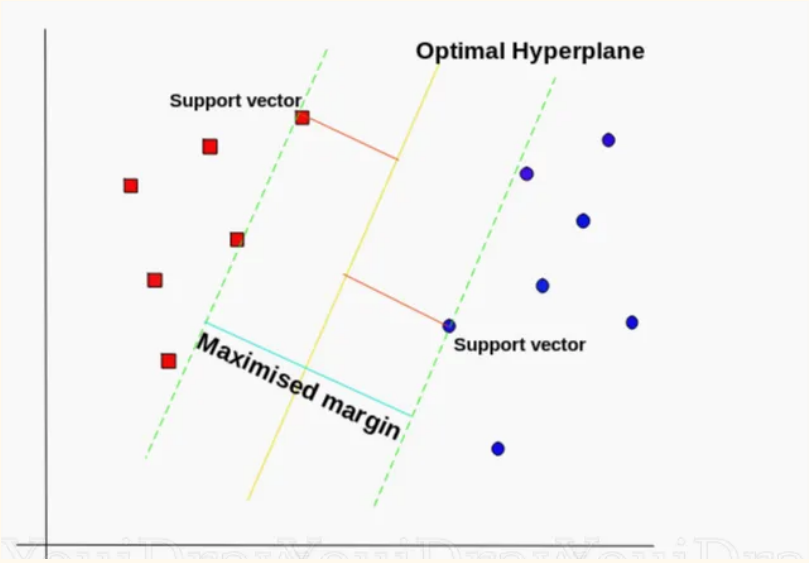


SVM

SVM is a linear classification/regression algorithm that we have applied to this problem. Simply put, an SVM creates a decision hyperplane that separates the classes of data. The SVM finds the points of each class that are closest to a decision boundary, These are called the "Support Vectors" and seeks to maximize the distance between the decision boundary to create a decision boundary with the least ambiguity (Figure 10):[4]

[4] https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989
[5]https://www.google.com/url?sa=i&url=https%3A%2F%2Frapidminer.com%2Fblog%2Fk-nearest-neighbors-laziest-machine-learning-technique%2F&psig=AOvVaw0chnVgjFO3g_zGTb1bzqs4&ust=1682980521295000&source=images&cd=vfe&ved=0CBAQjRxqFwoTCNCK6JTV0v4CFQAAAAAdAAAAABAD

Figure 10. Optimal Hyperplanes in SVMs.

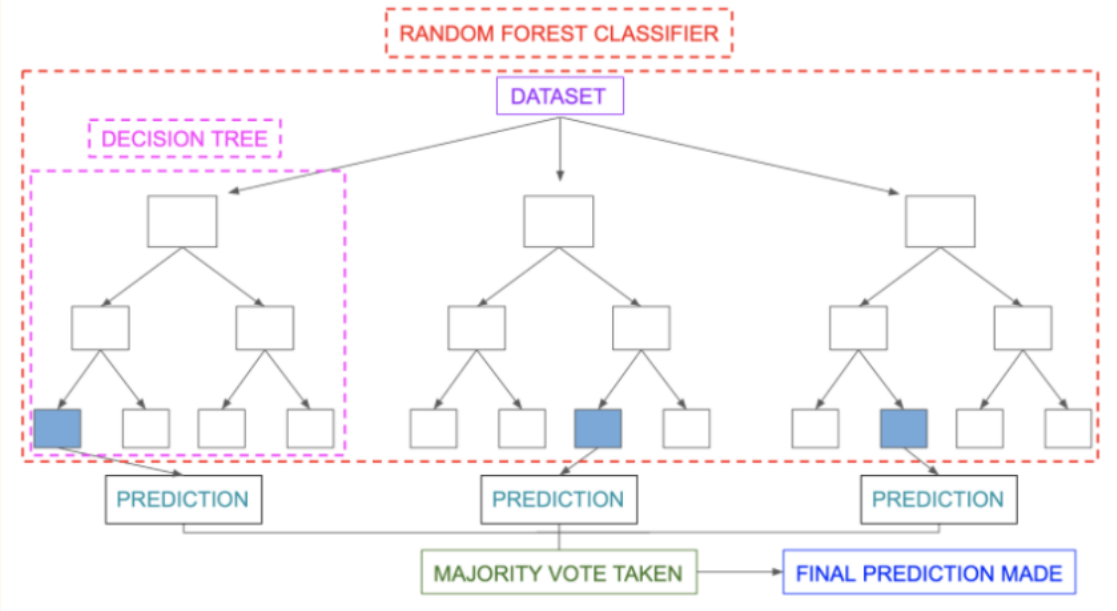


SVM's can be quite powerful and efficient in high dimensional data such as image processing, though our data is not particularly high dimensional. It does not handle large datasets well, which is fine because of our relatively small dataset. Like our XGBoost Model, we used a grid search cross validation approach to optimize hyperparameters, which we will detail in the results section.

Random Forest

Given that decision trees and neural network algorithms can fall very easily in the overfitting trap, we decided to test out the ensemble method of Random Forest. Random Forests are known for their good scalability and ease of use, and given that they are able to average the high variance that individual trees suffer, they are able to build more robust and less susceptible to overfitting models (Figure 11). Moreover, they are not affected by multicollinearity that much since every model sees a different set of data points. An additional advantage of using random forest relies in not requiring the need of choosing good hyperparameter values given ensemble models are robust to noise. Although not so common in practice we optimized the hyperparameters of the Random Forest, by using a randomized search. 70% of the reference data were randomly selected for the training stage and the other 30% were used for the testing. The proposed architecture was implemented on the machine learning framework Sklearn. Initializing with 100 n_estimators and max_depth of none (the nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.)

Figure 11. Random Forest Classification Algorithm

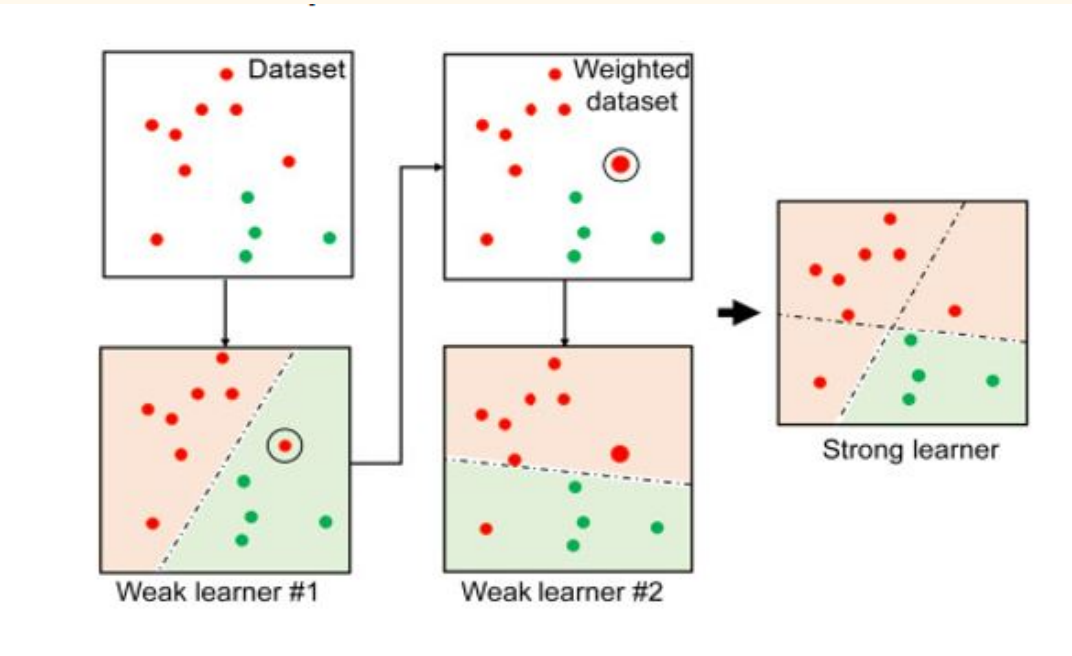


Source: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

AdaBoost

A more powerful approach than Random Forest suggested by the literature is the boosting approach (AdaBoost) which trains weak models and gives higher priority to the observations predicted incorrectly by previous models. Moreover, AdaBoost is often much better at making accurate classifications. The proposed architecture was implemented on the machine learning framework scikit-learn. Initializing with 100 n_estimators and using the decision tree like the base_estimator. A grid search was also undertaken to find the best hyperparameters.

Figure 12. AdaBoost Classification Algorithm



Source: https://www.sciencedirect.com/topics/engineering/adaboost

After discussing the experimental setups of each of the algorithms used, the following section details the results obtained.

Results

XGBoost

We utilized Gridsearch CV for our hyperparameter tuning. We set the following parameters for XGBoost: We set it to sample 50% of the datasets columns per tree, with a max tree depth of 7. Our learning rate, α, was set to 0.009 and our gamma was set to 0.25. Our reg_lambda parameter was set to 0.25 and our scale_pos_weight parameter was set to 3 and our sub sample parameter was set to 0.8. See the results section for information on model performance.

After fitting the optimal hyperparameters per the Gridsearch CV process, the XGBoost model's accuracy, precision, recall, and F1 only slightly improved, but when applying the ROC_AUC (Area Under Receiver Operator Characteristic Curve) score we saw that there was a notable improvement in model performance over the pre-tuned XGBoost model. The XGBoost had the best accuracy of all the models we tried (until we tried the ADABoost model) and showed very

encouraging results. The high recall score was especially encouraging given the seriousness of a false negative with regard to the damage and destruction a tsunami could cause. The high F1 score despite the non-normal data and relatively minimal preprocessing truly demonstrated the power of gradient boosting methods in classification tasks.

Figure 13. Confusion Matrix XGBoost

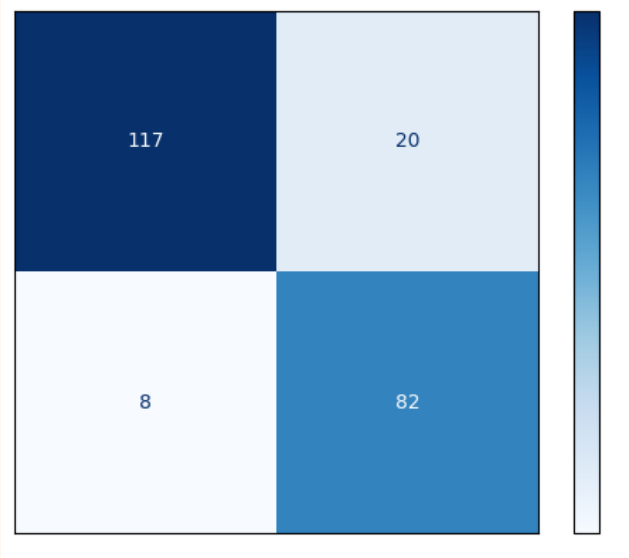
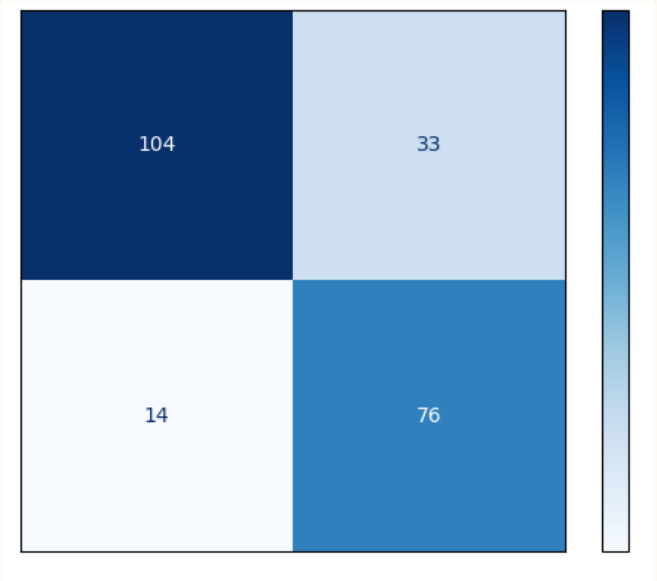


Table 1. Performance Metrics XGBoost

Precision	Recall	ROC_AUC	F1 Score
0.8540	0.9360	0.9514	0.8932

MLP

Figure 14. Confusion Matrix MLP



As addressed above, this multilayer Perceptron Model was used as a comparison case and was run with "out of the box" hyperparameters from the Sklearn package. (100 hidden layers, Adam Stochastic Gradient Optimizer, α= 0.0001, Tolerance= 0.0001, max iterations = 200). This model performed reasonably well given its lack of tuning, but was outclassed by the other models. The multilayer perceptron is the baseline for binary classification problems like ours, but is not as powerful and flexible as other models that were applied to this project. Consider this model a baseline to compare the other models.

Table 2. Performance Metrics MLP

Precision	Recall	ROC_AUC	F1 Score
0.7591	0.8814	0.8616	0.8156

KNN

The KNN model performed well at classifying the data. We examined the model at different levels of k, which is reflected in the graph below. We ultimately went with a model that uses 11 nearest neighbors, whose scores and confusion matrix are shown below. However, this model did not perform as well as the others.

Figure 15. Accuracy scores of KNN at different levels of *k*

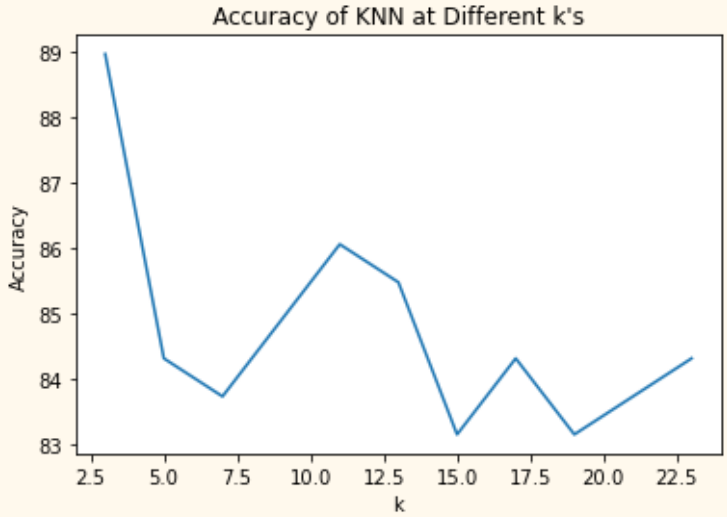


Figure 16. Confusion Matrix KNN

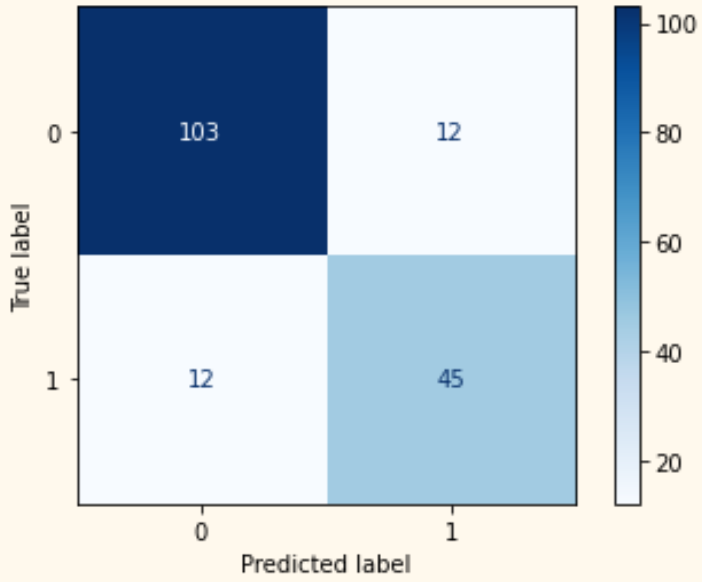


Table 3. Performance Metrics KNN

Precision	Recall	ROC_AUC	F1 Score
0.79	0.79	0.84	0.79

SVM

Figure 17. Confusion Matrix SVM

After applying hyperparameter optimization onto our SVM, we arrived at the following hyperparameters: Regularization C=2, Kernel Function Degree= 1, Gamma= 0.025, kernel function= Radial Basis. The hyperparameter tuning noticeably increased the model's performance. The F1 score of 0.82 was satisfactory and the recall was good considering the severity of false negatives in this particular classification problem. The SVM proved to be our weakest model that received any form of hyperparameter tuning. This makes sense as the SVM is best at handling high dimensional data and our dataset is relatively low dimensional. There were other models better suited to this task, but we decided that our findings when applying the SVM to this particular dataset would be useful as a comparative metric. Given more complex seismic data such as the kind used in the previously mentioned RIKEN Pioneering Project study, the SVM may have outperformed other models, but given the dataset that we are using, it seems there are stronger choices for a classification model.

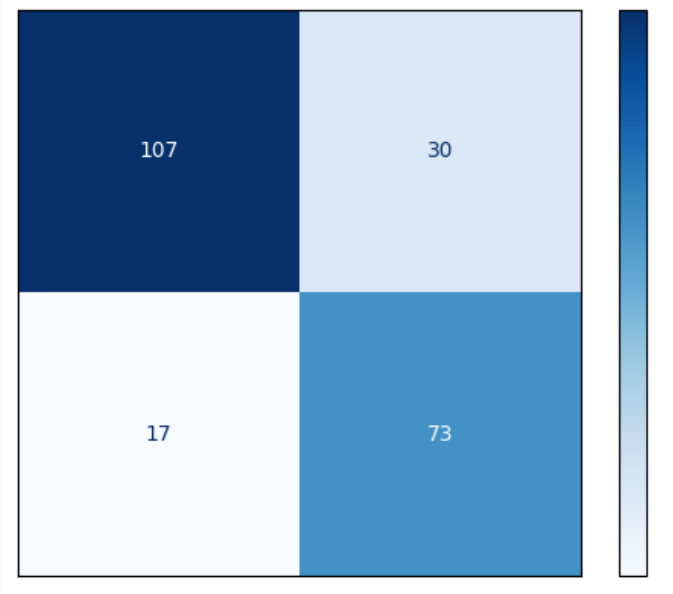


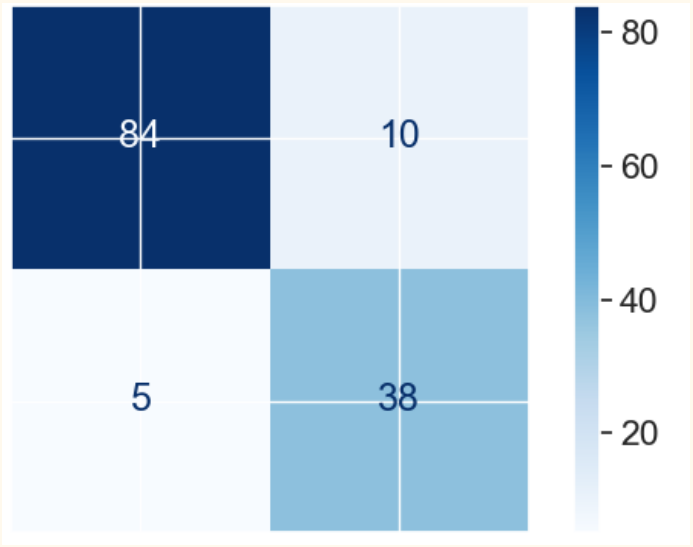
Table 4. Performance Metrics SVM

Precision	Recall	ROC_AUC	F1 Score
0.7810	0.8629	0.8716	0.819

Random Forest

After performing the randomized search it was determined that the best hyperparameters for the model were a max depth of 19 and 495 as number of estimators. Under the best model scenario the performance metric shows a precision of 0.7966, a recall of 0.9216, a ROC_AUC of 0.8910 and aF1 score of 0.8545 (Table1). As it can be seen in the confusion matrix (Figure 9), even though the precision is high, given the unbalance in the data the previously mentioned metrics are a more accurate indicator of the model performance.

Figure 18. Confusion Matrix Random Forest

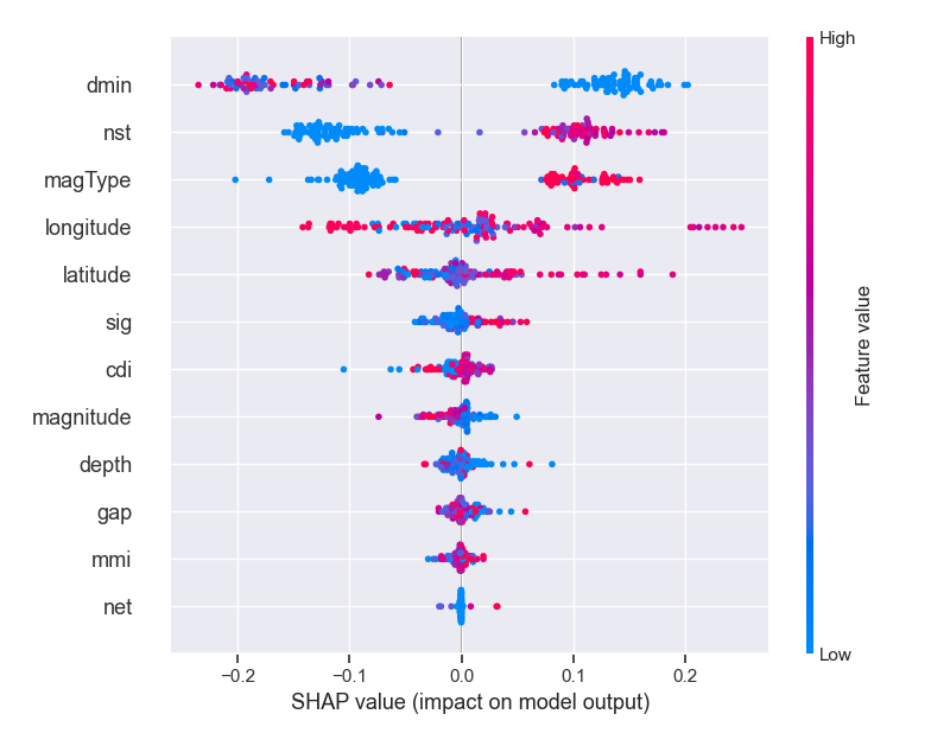


In addition to common performance metrics, it also became relevant to understand the feature importance of the model. In order to do so SHAP values were used. The usefulness of SHAP values relies on them being model-agnostic and being able to get an overview of which features are most important for a model. Figure 19 sorts features by the sum of SHAP value magnitudes over all samples, and uses SHAP values to show the distribution of the impacts each feature has on the model output. The color represents the feature value (red high, blue low). This reveals for example that a high dmin (horizontal distance from the epicenter to the nearest station) lowers the predicted tsunami probability.

Table 5. Performance Metrics Random Forest

Precision	Recall	ROC_AUC	F1 Score
0.7966	0.9216	0.8910	0.8545

Figure 19. Random Forest Feature Importance SHAP values



AdaBoost

In the case of the AdaBoost after performing the grid search it was determined that the best hyperparameters for the model were a learning rate of 0.1 and 100 as number of estimators. Under the best model scenario the performance metric shows a precision of 0.8727, a recall of 0.9412, a ROC_AUC of 0.9299 and a F1 score of 0.9057 (Table2). Similarly to the Random Forest model, it can be seen in the confusion matrix (Figure 10), that even though the precision is high, given the unbalance in the data the previously mentioned metrics are a more accurate indicator of the model performance.

Figure 20. Confusion Matrix AdaBoost

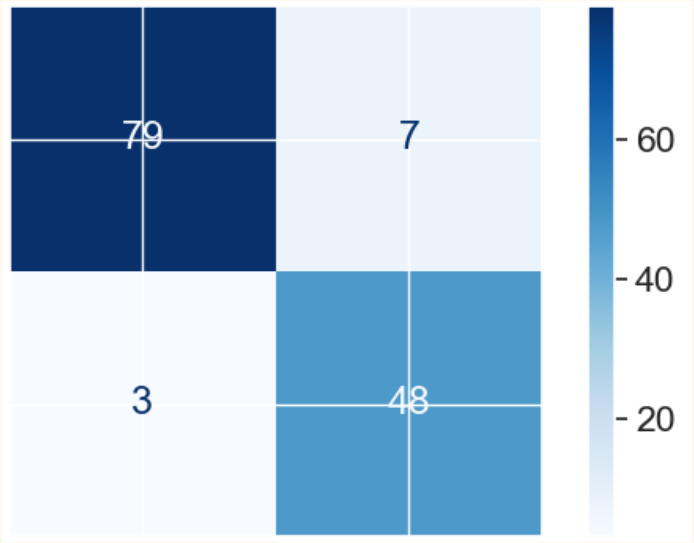


Table 6. Performance Metrics AdaBoost

Precision	Recall	ROC_AUC	F1 Score
0.8727	0.9412	0.9299	0.9057

Summary and conclusions

As it can be seen in Table 7, from the six models implemented XGBoost and AdaBoost have the best performance metrics. However, the AdaBoost is slower than the XGBoost. XGBoost was a good algorithm to choose for this task, but was outperformed slightly by AdaBoost. Both the models are gradient boosters, which explains why algorithms like this are commonly used in Kaggle Competitions. XGBoost's efficiency advantages over the AdaBoost were not perceptible on a dataset this small and clean, but on future iterations of this study with larger and more complex data, the XGBoost's performance advantages would become clear.

It also becomes relevant to point out that our approach has numerous limitations, specified in the posterior section. Given so this research could be improved by the addition of geospatial data, sea-level rise information. Moreover, the approach used in this research could also be adopted to classify other hazards such as earthquakes, landslides, and floods. Additionally, this

research could be expanded with the use of convolutional neural networks and geospatial analysis.

Though more advanced studies with higher levels of domain expertise and more complicated data exist, this study shows what can be done with relatively simple data and models and serves as a baseline for future research into tsunami prediction without extensive oceanic data. Ideally, following studies with similar approaches would find more accurate ways to predict tsunamis without expensive equipment and complex data collection.

Table 7. Summary Performance Metrics

Method	Precision	Recall	ROC_AUC	F1 Score
XGBoost	0.8540	0.9360	0.9514	0.8932
MLP	0.7591	0.8814	0.8616	0.8156
KNN	0.79	0.79	0.84	0.79
SVM	0.7810	0.8629	0.8716	0.819
Random Forest	0.7966	0.9216	0.8910	0.8545
AdaBoost	0.8727	0.9412	0.9299	0.9057

Limitations

While a great deal of work has been done in this project, there were some limitations that we believe are important to address. The first limitation being the inability to apply more powerful deep learning methods to this project. The researchers at the time of development, did not have expertise in the realm of more advanced deep learning models and in future iterations of this project may see fit to implement them to possibly obtain greater accuracy in their predictions.

Another limitation of this study is the relative simplicity of the dataset. While we performed a literature review and studied the dataset, we do not have extensive domain expertise in the realm of oceanic seismography and did not have access to more advanced datasets such as the previously mentioned data from the RIKEN Pioneering Project[5]. This project was purely meant as an exploratory venture into tsunami prediction using simple metrics, but attempts at more advanced models would be good for future inquiry. Finally, the data only allows us to predict

[5] https://www.nature.com/articles/s41467-022-33253-5#Sec11

whether or not a tsunami occurs, not the severity of the tsunami. This could be useful in the future considering that many tsunamis can cause minimal damage.

References

Chamola, V., Hassija, V., & Gupta, S. (2020). *Disaster and Pandemic Management Using Machine Learning: A Survey*. NCBI. Retrieved April 29, 2023, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8768997/

Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

Guha, S., K. Jana, R., & K. Sanyal, M. (2022, July 29). *https://www.sciencedirect.com/science/article/abs/pii/S2212420922004952*. Science Direct. Retrieved April 29, 2023, from https://www.sciencedirect.com/science/article/abs/pii/S2212420922004952

Kainthura, P., & Sharma, N. (2022, July 29). *Hybrid machine learning approach for landslide prediction, Uttarakhand, India*. https://www.nature.com/articles/s41598-022-22814-9. Retrieved April 29, 2023, from https://www.nature.com/articles/s41598-022-22814-9

Leventis, D. (2022, January 2). *XGBoost mathematics explained*. Medium. Retrieved April 30, 2023, from https://dimleve.medium.com/xgboost-mathematics-explained-58262530904a

Linardos, V., Drakaki, M., Tzionas, P., & Karnavas, Y. L. (2022). *Machine Learning in Disaster Management: Recent Developments in Methods and Applications*. MDPI. Retrieved April 29, 2023, from https://www.mdpi.com/2504-4990/4/2/20

NWS. (n.d.). *NWS JetStream - Tsunami Generation: Earthquakes*. National Weather Service. Retrieved April 29, 2023, from https://www.weather.gov/jetstream/gen_earth

scikit learn. (n.d.). *scikit.learn manual*. scikit-learn: machine learning in Python — scikit-learn 1.2.2 documentation. Retrieved April 29, 2023, from https://scikit-learn.org/stable/index.html

Pupale, R. (2019, February 11). *Support vector machines(svm) - an overview*. Medium. Retrieved April 30, 2023, from https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989

Yousefi, S., & Reza Pourghasemi, H. (2022, July 29). *A machine learning framework for multi-hazards modeling and mapping in a mountainous area*. https://www.nature.com/articles/s41598-020-69233-2. Retrieved April 29, 2023, from https://www.nature.com/articles/s41598-020-69233-2

Appendix

Libraries:

Modeling File:

```
import numpy as np
import pandas as pd
from sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.naive_bayes import GaussianNB
import xgboost as xgb
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_auc_score
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, plot_confusion_matrix
from sklearn.svm import SVC
import keras
from keras.models import Sequential
```

```
from keras.layers import Dense, Dropout

from keras.optimizers import Adam

from keras.wrappers.scikit_learn import KerasClassifier

from sklearn.metrics import make_scorer, accuracy_score

from sklearn.compose import ColumnTransformer

import time

import shap

from sklearn import preprocessing

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

import plotly.express as px

import seaborn as sns

import geopandas as gpd

from sklearn import tree

from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score,
ConfusionMatrixDisplay

from sklearn.model_selection import RandomizedSearchCV, train_test_split

from scipy.stats import randint

from sklearn.tree import export_graphviz

from sklearn.metrics import f1_score

from sklearn.ensemble import AdaBoostClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn import metrics

from sklearn.model_selection import RepeatedStratifiedKFold
```

PreProcessing:

```
import time
from sklearn import preprocessing
import plotly.express as px
from tkinter import *
import geopandas as gpd
from scipy.stats import shapiro
import scipy.stats
from statsmodels.stats.outliers_influence import variance_inflation_factor
```