Group #1

EARTHQUAKE AND TSUNAMI PREDICTION USING MACHINE LEARNING TECHNIQUES

Jack McMorrow, Alex Khater, Alejandra Mejia

Source: Tenor

Why Tsunami prediction?

- According to the WHO, tsunamis have caused of 250,000 deaths between 1998-2017
- According to the IMF, the 2011 tsunami in Japan caused between 200 and 300 billion dollars in damages
- Over 700 Million people live in high risk areas for tsunamis
- Early warning systems are key to minimizing damage and loss of life
- Tsunamis can be difficult to predict, given the many factors that contribute toward their formation.

Before and After Photo from the 2011 Japanese Tsunami
Source: https://aqworks.com/en/blog/2011/03/23/infographics-roundup-2011-tohoku-earthquake-and-tsunami/

About the dataset

- The dataset used in the current project was retrieved from [Kaggle](#) and contains data records of 782 different earthquakes.
- It contains information about various earthquakes that have occurred from 2001 to 2022.
- It contains 18 different columns including:
    - Geo-location data
    - Earthquake Magnitude and Projected Damage
    - Various measurement metrics showing what predictive data was available and how extensive it was at the time of the event
    - Date/Time Data
    - Our target variable: a binary variable showing whether or not a tsunami was caused

Quick Pre-Processing Rundown

- Dropped the columns 'Alert', 'Country' and 'Continent' due to high quantity of null value
- Removed Outliers per the mean+3*standard Deviation formula
- Dropped 'Title' because of lack of usage for modeling
- Decided against PCA because of low multicollinearity concerns and low dimensionality
- Applied a Kolmogorov-Smirnov test for normality found non- normal data in most of our columns.
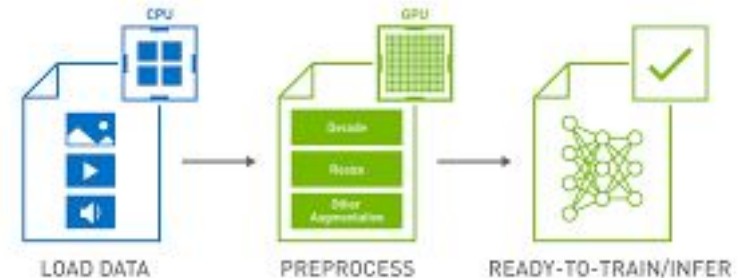

Image Source: NVIDIA developers

EDA

- 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
- The years 2013 and 2015 saw the most number of earthquakes,
- The month of November seems to be the one with more recorded occurrences
- Interesting insight into considering adding variables in future research related to specific environmental factors that occur during those years or in that month, like sea level rise.
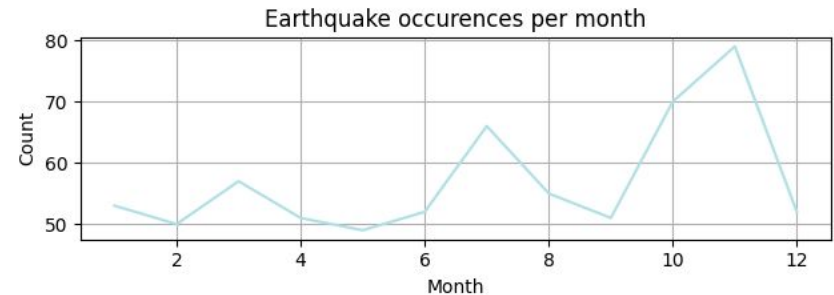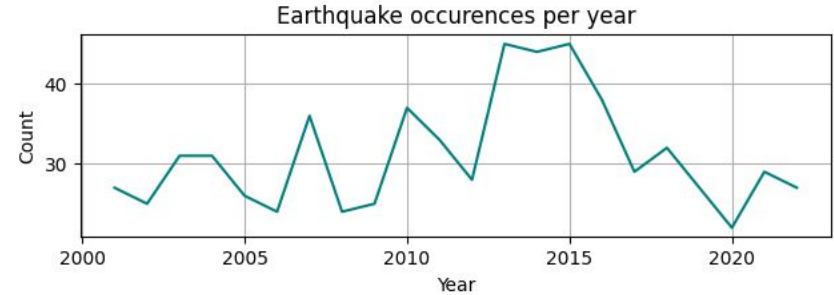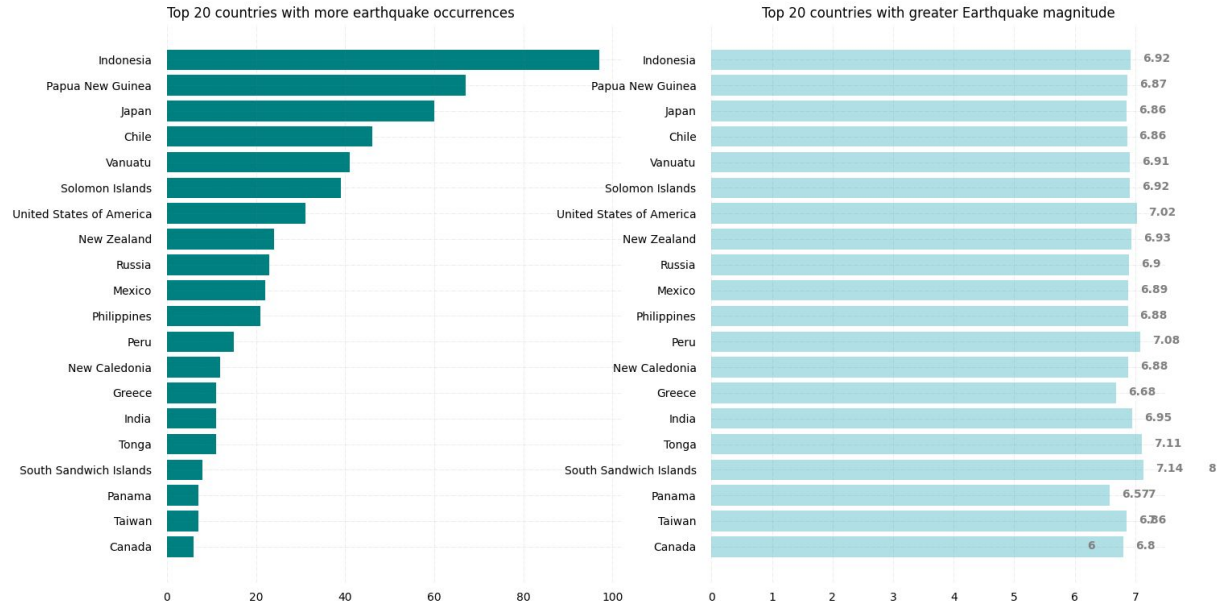

Earthquakes occurrence per year and month

EDA

Events are spread out in different continents:

- Indonesia has the highest number of earthquakes.
- On average a higher number of earthquakes does not mean a higher magnitude.

Top 20 countries with more earthquake occurrences

Country	
Indonesia	
Papua New Guinea	
Japan	
Chile	
Vanuatu	
Solomon Islands	
United States of America	
New Zealand	
Russia	
Mexico	
Philippines	
Peru	
New Caledonia	
Greece	
India	
Tonga	
South Sandwich Islands	
Panama	
Taiwan	
Canada	

Top 20 countries with greater Earthquake magnitude

Country	Magnitude
Indonesia	6.92
Papua New Guinea	6.87
Japan	6.86
Chile	6.86
Vanuatu	6.91
Solomon Islands	6.92
United States of America	7.02
New Zealand	6.93
Russia	6.9
Mexico	6.89
Philippines	6.88
Peru	7.08
New Caledonia	6.88
Greece	6.68
India	6.95
Tonga	7.11
South Sandwich Islands	7.14
Panama	6.577
Taiwan	6.736
Canada	6.8

EDA

- Slightly unbalanced with 37 percent of tsunamis occurrences against 63 percent of occurrence with earthquakes but no tsunamis.
- Features were not correlated between them or with our target variable.
- T- test indicated non-significant p-values which indicated low levels of correlation

Correlation Coefficient between features - Original space
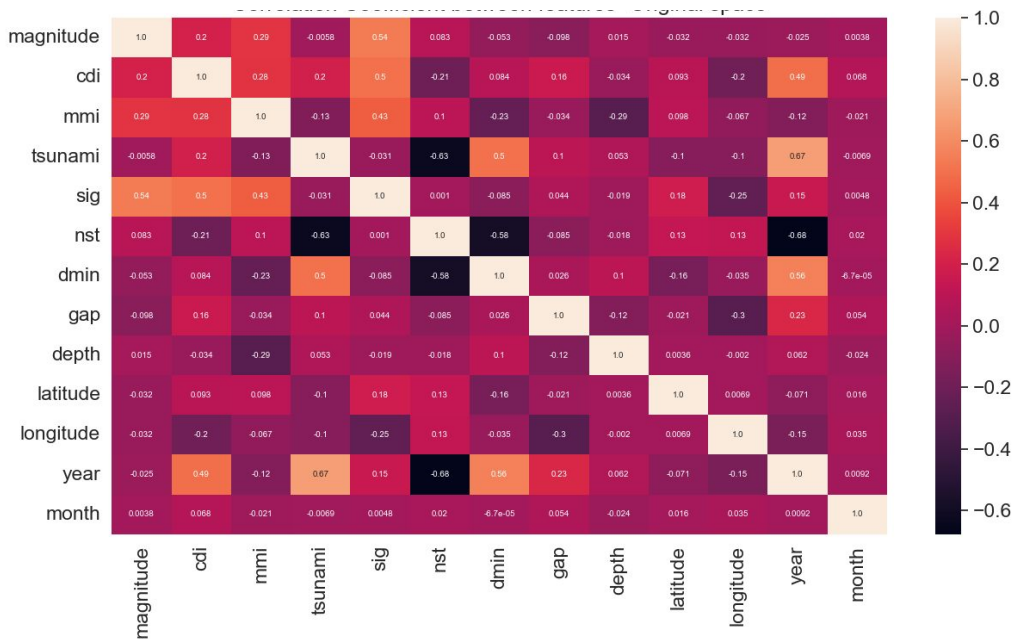
Considerations for model selection

1. **Alignment with project goal**
   - ML algorithms can handle multidimensional, large (ML) volumes of data usually occurring in thematics related to disaster and pandemic management.
   - Been used to better cope with the severe and often catastrophic impacts of disasters
   - More effective at handling complicated and nonlinear high-dimensional data sets

2. **Dataset Size:** dataset is not extremely big ~ a K-Nearest Neighbors(KNN)

3. **Dimensionality**
   - No abundant features, no multicollinearity ~ non-complex NN

4. **Inference time:** SVM simple and faster, XGBoost faster than AdaBoost

5. **Train Test Split:** We decided on a 67-33 split for our training and testing set using Sklearn
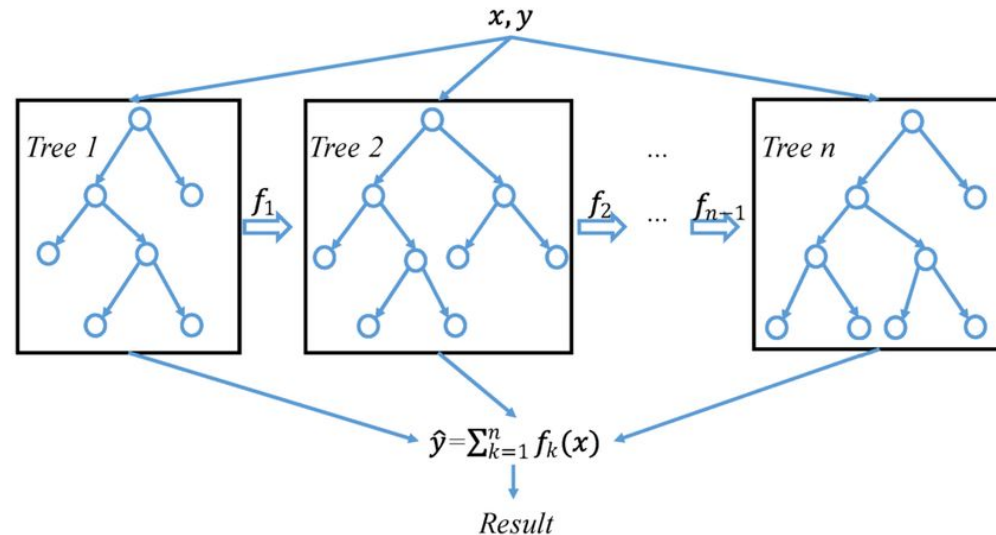
A Note on Model Evaluation

- The metric we are primarily using to evaluate our models is the F1 Score
- Our data is slightly imbalanced
- Because we mainly care about the positive case (Not much penalty to falsely positives relative to false negatives) the F1 score is the other metric
- We will be showing other metrics (including ROC_AUC), but F1 score is the one we are primarily concerned with

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}}$$

$$= \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Image Source: https://www.v7labs.com/blog/f1-score-guide

A Brief Overview of XGBOOST

- Stands for "Extreme Gradient Boosting"
- Builds a series of weak decision tree models in parallel and forms them level by level
- The final model is a weighted sum of all tree predictions
- It is called gradient boosting because it uses a gradient descent algorithm.

- Very simply put: XGBOOST uses a collection of parallel built decision trees to create a model

A brief overview of the XGBOOST algorithm

Image Source: https://www.researchgate.net/figure/A-general-architecture-of-XGBoost_fig3_335483097

Why XGBOOST

- Very Powerful and Flexible

- Does not need extensive pre-processing

- Does not require normal data because it is a

  composite of decision tree models

- Easy to Use

- Has been dominating Kaggle competitions recently

Our XGBOOST Model

- We used Gridsearch_CV

- It is a cross validation method that tests various values for hyperparameters from a predefined dictionary to find the optimal model performance

```python
#New Parameter ranges since some are at their end
param_grid2 = {
    "max_depth": [7,8,15, 20],
    "learning_rate": [0.005, 0.009, 0.01],
    "gamma": [0.25],
    "reg_lambda": [0.25, 0.5, 0.75, 1, 2],
    "scale_pos_weight": [3],
    "subsample": [0.8],
    "colsample_bytree": [0.5]
}

grid_cv_2 = GridSearchCV(xgb_cl, param_grid2,
                         cv=3, scoring="roc_auc", n_jobs=-1)

_ = grid_cv_2.fit(X_train, y_train)
print("gcv2 best score:",grid_cv_2.best_score_)
```
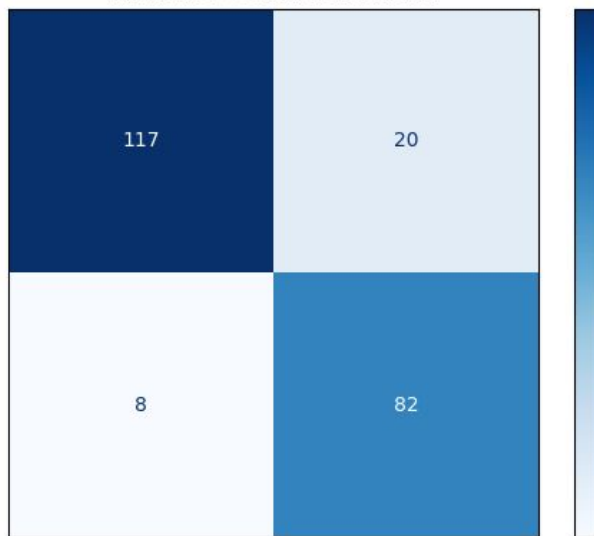
An example from our code of using GridSearch CV

# Our XGBoost Results

Final XGB Confusion Matrix

Method	Precision	Recall	ROC_AUC	F1 Score
MLP	0.7591	0.8814	0.8616	0.8156
XGBoost	0.8540	0.9360	0.9514	0.8932

Basic MLP

We will be using a basic, untuned and unmodified MLP from Sklearn as a comparison for our other models.

The point of this model is to serve as a comparison to the more advanced models we will be using for this project. Here is its confusion matrix:

MLP Confusion Matrix
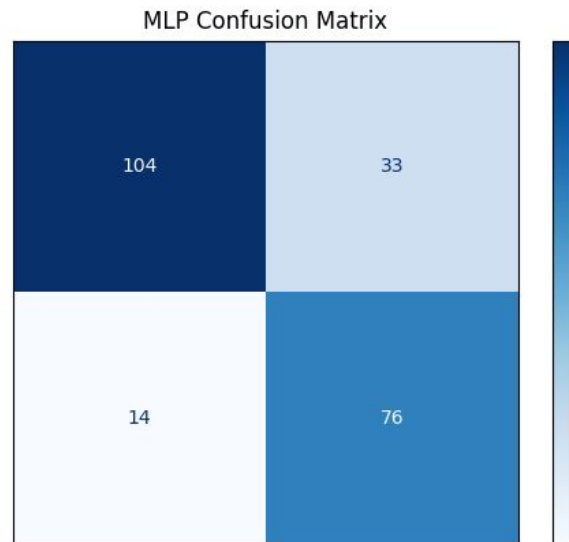
Method	Precision	Recall	ROC_AUC	F1 Score
MLP	0.7591	0.8814	0.8616	0.8156

A Brief Overview of SVM

- Linear Classifier

- Uses selected points of each class as

  "support vector"s to increase a margin

- Well Suited to High Dimensional Data

- Struggles with Large Datasets

Source:
https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989
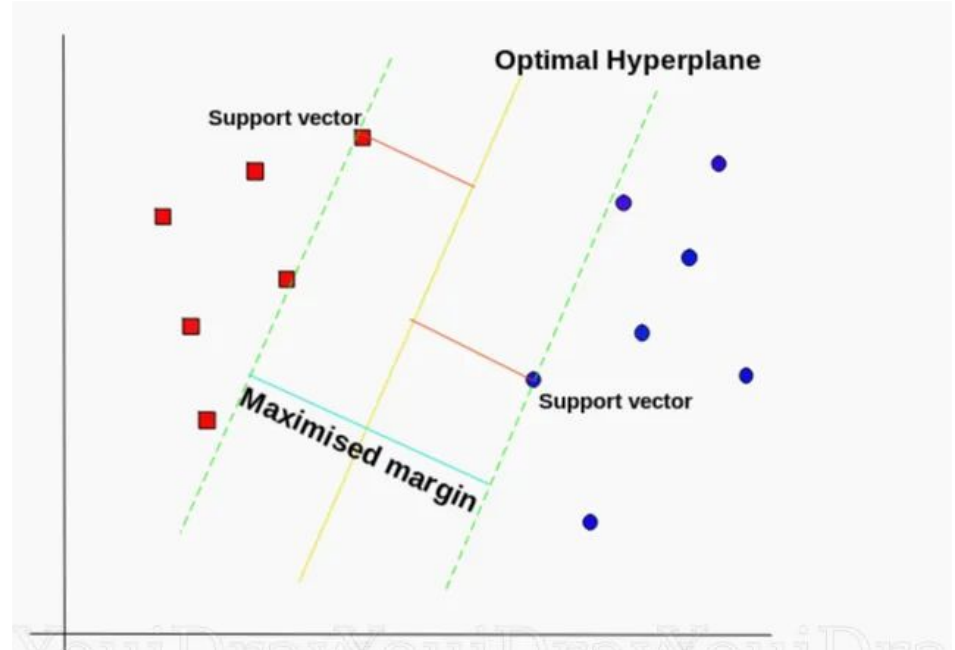
Our SVM Model

- We used Gridsearch_CV

- It is a cross validation method that tests various values for hyperparameters from a predefined dictionary to find the optimal model performance

```
#Time for a second Grid CV
param_grid = {
    "C": [2],
    "kernel": ['rbf'],
    "degree": [1],
    "gamma": [0.01, 0.025, 0.075, 0.1],
    "probability": [True]}

grid_cv2 = GridSearchCV(SVM1, param_grid, n_jobs=-1, cv=3, scoring="roc_auc")


_ = grid_cv2.fit(X_train, y_train)
print("Grid CV2 best score:", grid_cv2.best_score_)
💡
print("SVM best Params", grid_cv2.best_params_)
```

An example from our code of using GridSearch CV on the SVM

Our SVM Results

Tuned SVM Confusion Matrix

107	30
17	73
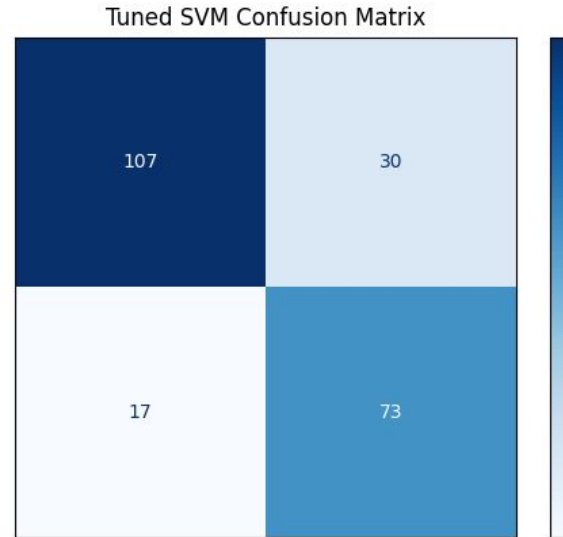
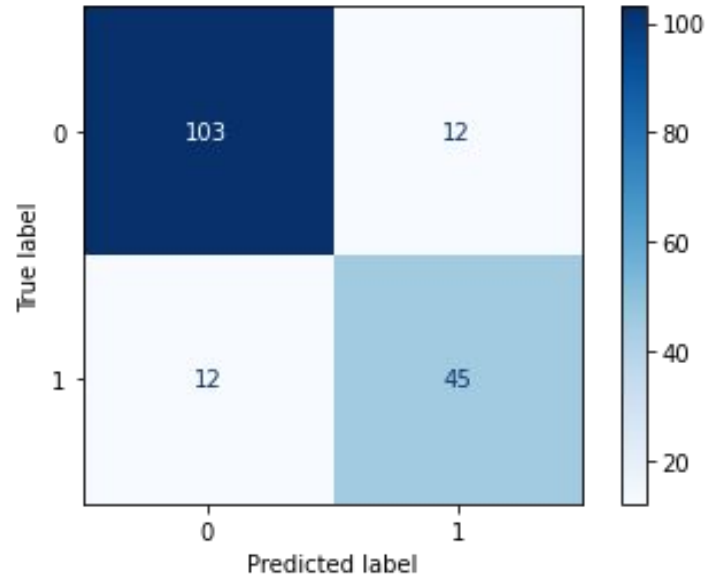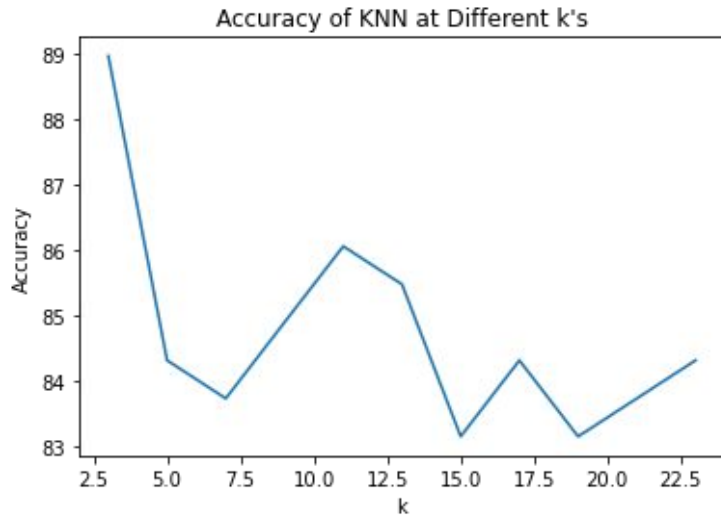| Method	Precision	Recall	ROC_AUC	F1 Score
XGBoost	0.8540	0.9360	0.9514	0.8932
MLP	0.7591	0.8814	0.8616	0.8156
SVM	0.7810	0.8629	0.8716	0.819

Random Forest Model

- Decision trees and neural network algorithms can fall very easily in the overfitting
- RF: Good scalability and ease of use, more robust and less susceptible to overfitting, no issue with multicollinearity
- Initializing 100 n_estimators , max_depth of none.
- Randomized search best hyperparameters max depth of 19 and 495 as n_estimators
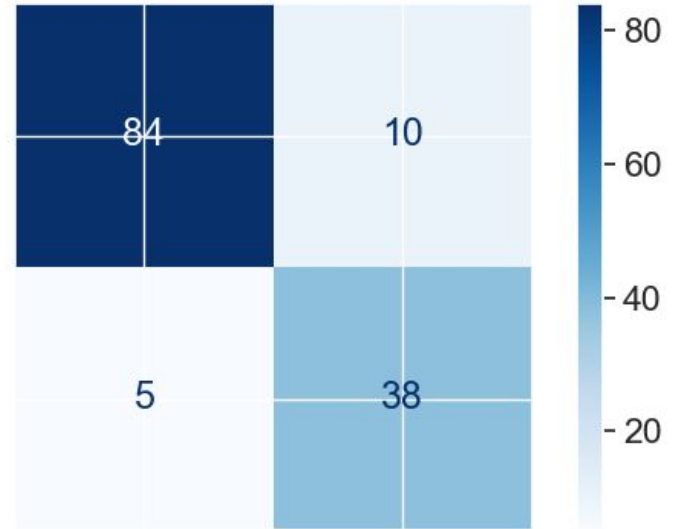
| Precision	Recall	ROC_AUC	F1 Score
0.7966	0.9216	0.8910	0.8545

KNN Model

- We also implemented a K-Nearest Neighbors model to classify the data
- We chose a *k* value of 11 for our final model.

Accuracy of KNN at Different k's

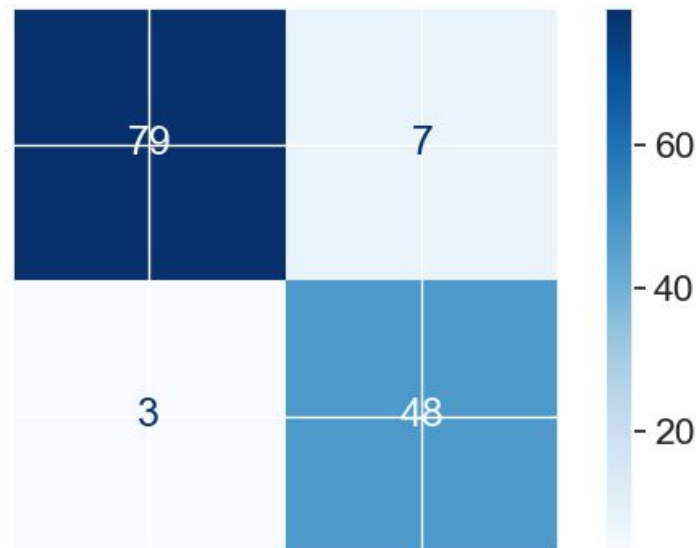| Precision	Recall	ROC_AUC	F1 Score
0.79	0.79	0.84	0.79

Random Forest Model

- SHAP values relies on them being model-agnostic and being able to get an overview of which features are most important for a model.
- SHAP values to show the distribution of the impacts each feature has on the model output.
- High *dmin* (horizontal distance from the epicenter to the nearest station) lowers the predicted tsunami probability.
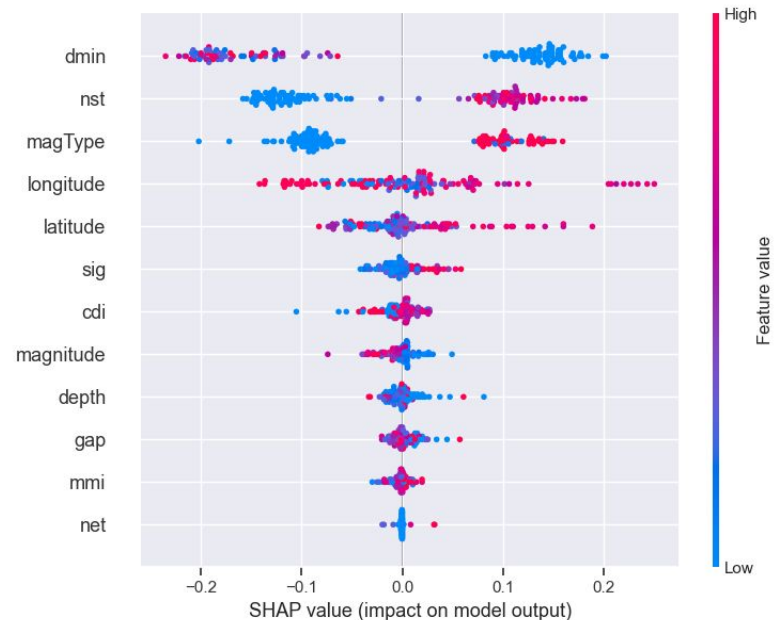
AdaBoost Model

- Trains weak models and gives higher priority to the observations predicted incorrectly by previous models.
- Often much better at making accurate classifications.
- Initialized with 100 n_estimators and decision tree like the base_estimator.
- Best hyperparameters for the model were a learning rate of 0.1 and 100 as number of estimators.

Precision	Recall	ROC_AUC	F1 Score
0.8727	0.9412	0.9299	0.9057

Limitations and future studies

2 Primary Limitations To Be Addressed:

1. Lack of Deep Learning/ more advanced models

2. Usage of More Advanced Seismic Data

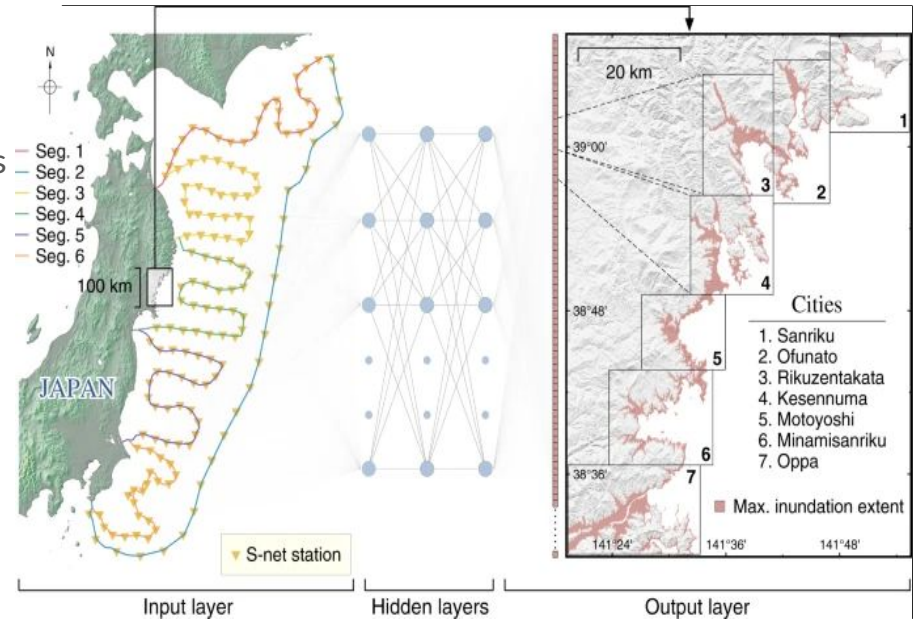3. Lack of information on tsunami severity

Image from the more complex RIKEN Pioneering Project Study using advanced Seismographic Data and Models

Thank you!

Questions?