STAR LION COLLEGE OF ENGINEERING

AND TECHNOLOGY

Program : Earthquake Prediction model using python
Name : S. Bharaniga
Date : 18.10.2023
Register No : 822021104004
Nanmuthalvan ID :au822021104004

EARTHQUAKE PREDICTION MODEL USING PYTHON

Earthquake prediction :

Earthquake prediction is a branch of the science of seismology concerned with the specification of the time, location, and magnitude of future earthquakes within stated limits, and particularly "the determination of parameters for the *next* strong earthquake to occur in a Earthquake prediction is sometimes distinguished from *earthquake forecasting*, which can be defined as the probabilistic assessment of *general* earthquake hazard, including the frequency and magnitude of damaging earthquakes in a given area over years or decades.

Content :

- Causes
- **Characteristics**
- **Measurement**
- Prediction

Causes :

Fault (geology):

In geology, a **fault** is a planar fracture or discontinuity in a volume of rock across which there has been significant displacement as a result of rock-mass movements. Large faults within Earth's crust result from the action of plate tectonic forces, with the largest forming the boundaries between the plates, such as the megathrust faults of subduction zones or transform faults. Energy release associated with rapid movement on active faults is the cause of most earthquakes. Faults may also displace slowly, by aseismic creep.

The vector of slip can be qualitatively assessed by studying any drag folding of strata, which may be visible on either side of the fault.[12] Drag folding is a zone of folding close to a fault that likely arises from frictional resistance to movement on the fault.[13] The direction and magnitude of heave and throw can be measured only by finding common intersection points on either side of the fault (called a piercing point). In practice, it is usually only possible to find the slip direction of faults, and an approximation of the heave and throw vector.

Volcano tectonic earthquake :

A volcano tectonic earthquake or volcano earthquake is caused by the movement of magma beneath the surface of the Earth. The movement results in pressure changes where the rock around the magma has experienced stress. At some point, this stress can cause the rock to break or move. This seismic activity is used by scientists to monitor volcanoes. The earthquakes may also be related to dike intrusion or occur as earthquake swarms.

Cause of volcano tectonic earthquakes:

One possible scenario resulting in a possible volcano tectonic earthquake is tectonic subduction zones. The compression of plates at these subduction zones forces the magma beneath them to move.Magma can not move through the newly compressed crust in as easily a manner. This means it tends to pool in magma chambers beneath the surface and between the converging tectonic plates.

Use in monitoring volcanoes :

Nearly every recorded volcanic eruption has been preceded by some form of earthquake activity beneath or near the volcano. Due to the relation between magma movement, earthquakes, and possible eruptions, approximately 200 of the world's volcanoes are seismically monitored.

Induced seismicity :

Induced seismicity is typically earthquakes and tremors that are caused by human activity that alters the stresses and strains on Earth's crust. Most induced seismicity is of a low magnitude. A few sites regularly have larger quakes, such as The Geysers geothermal plant in California which averaged two M4 events and 15 M3 events every year from 2004 to 2009.

Causes :

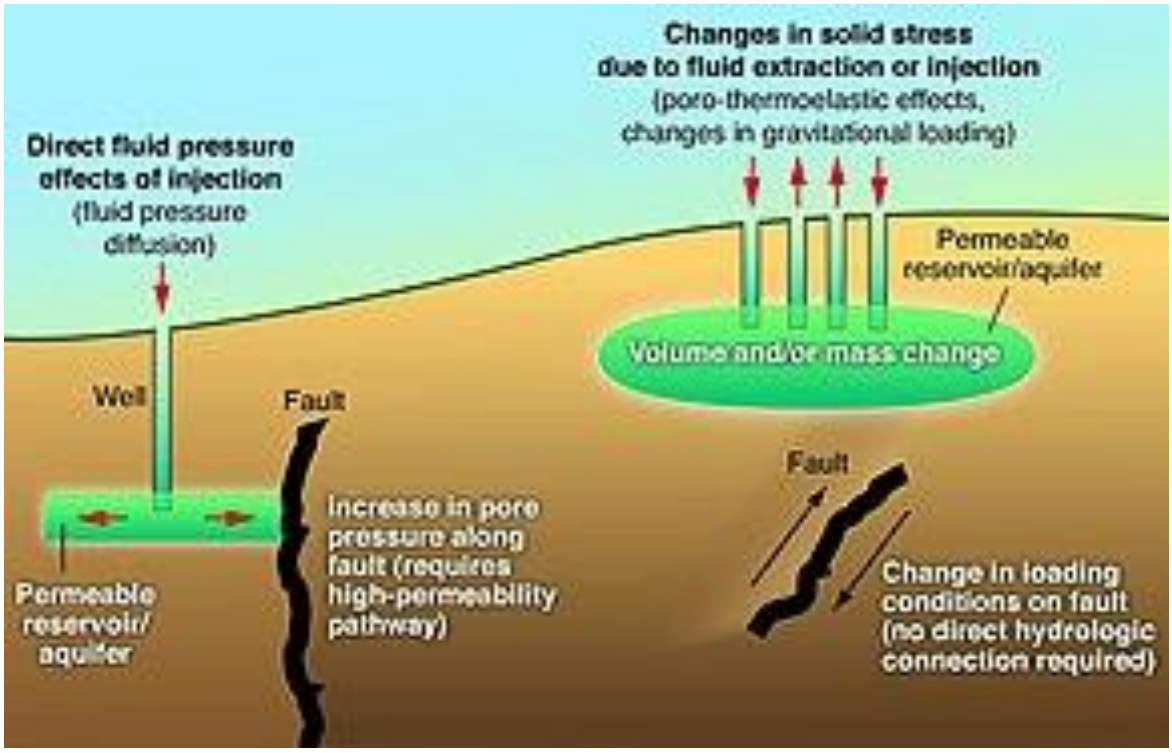


Diagram showing the effects that fluid injection and withdrawal can have on nearby faults can cause induced seismicity

Though understanding of reservoir-induced seismic activity is very limited, it has been noted that seismicity appears to occur on dams with heights larger than 330 feet (100 m).

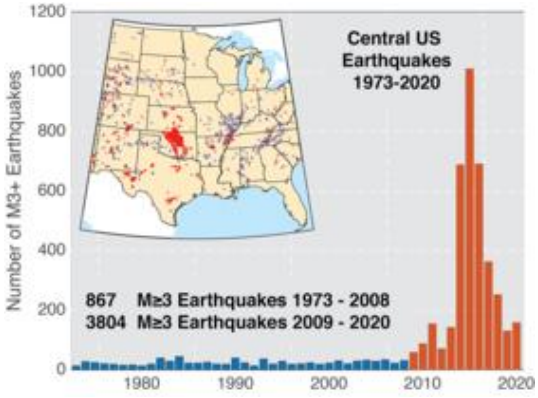
The extra water pressure created by large reservoirs is the most accepted explanation for the seismic activity. When the reservoirs are filled or drained, induced seismicity can occur immediately or with a small time lag.

Artificial lakes :

The largest earthquake attributed to reservoir-induced seismicity occurred at Koyna Dam

Waste disposal wells :

Injecting liquids into waste disposal wells, most commonly in disposing of produced water from oil and natural gas wells, has been known to cause earthquakes. This high-saline water is usually pumped into salt water disposal (SWD) wells. The resulting increase in subsurface pore pressure can trigger movement along faults, resulting in earthquakes.



Cumulative number of earthquakes in the central U.S. The red cluster at the center of the map shows an area in and around Oklahoma which experienced the largest increase in activity since 2009.

Prediction methods :

Earthquake prediction is an immature science – it has not yet led to a successful prediction of an earthquake from first physical principles. Research into methods of prediction therefore focus on empirical analysis, with two general approaches: either identifying distinctive *precursors* to earthquakes, or identifying some kind of geophysical *trend* or pattern in seismicity that might precede a large earthquake.

Electromagnetic anomalies :

Observations of electromagnetic disturbances and their attribution to the earthquake failure process go back as far as the Great Lisbon earthquake of 1755, but practically all such observations prior to the mid-1960s are invalid because the instruments used were sensitive to physical movement.

Since then various anomalous electrical, electric-resistive, and magnetic phenomena have been attributed to precursory stress and strain changes that precede earthquakes,[1] raising hopes for finding a reliable earthquake precursor.

While a handful of researchers have gained much attention with either theories of how such phenomena might be generated, claims of having observed such phenomena prior to an earthquake, no such phenomena has been shown to be an actual precursor.

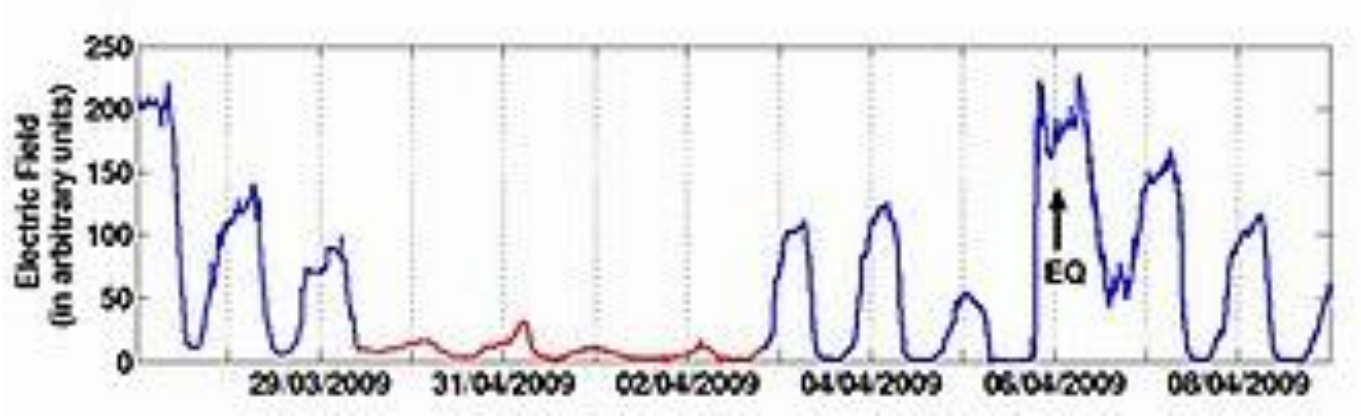
Freund physics :

In his investigations of crystalline physics, Friedemann Freund found that water molecules embedded in rock can dissociate into ions if the rock is under intense stress.

The resulting charge carriers can generate battery currents under certain conditions. Freund suggested that perhaps these currents could be responsible for earthquake precursors such as electromagnetic radiation, earthquake lights and disturbances of the plasma in the ionosphere.The study of such currents and interactions is known as "Freund physics".

Disturbance of the daily cycle of the ionosphere :

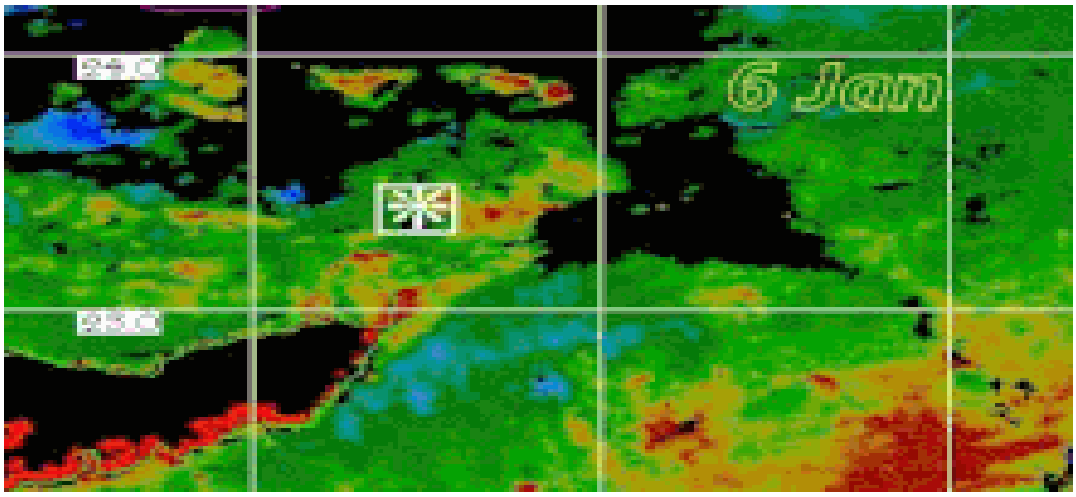
The ionosphere usually develops its lower D layer during the day, while at night this layer disappears as the plasma there turns to gas. During the night, the F layer of the ionosphere remains formed, in higher altitude than D layer.



The ULF* recording of the D layer retention of the ionosphere which absorbs EM radiation during the nights before the earthquake in L'Aquila, Italy on 6/4/2009. The anomaly is indicated in red.

Satellite observation of the expected ground temperature declination :

One way of detecting the mobility of tectonic stresses is to detect locally elevated temperatures on the surface of the crust measured by satellites. During the evaluation process, the background of daily variation and noise due to atmospheric disturbances and human activities are removed before visualizing the concentration of trends in the wider area of a fault. This method has been experimentally applied since 1995.



The thermal night recording on January 6, 21 and 28, 2001 in the Gujarat region of India. Marked with an asterisk is the epicenter of the Bhuj earthquake on January 26 that was of 7.9 magnitude. The intermediate recording reveals a thermal anomaly on January 21 which is shown in red. In the next recording, 2 days after the earthquake, the thermal anomaly has disappeared.

Code:

```
import datetime
import time

timestamp = []
for d, t in zip(data['Date'], data['Time']):
    try:
        ts = datetime.datetime.strptime(d+' '+t, '%m/%d/%Y %H:%M:%S')
        timestamp.append(time.mktime(ts.timetuple()))
    except ValueError:
        # print('ValueError')
```

```
11.         timestamp.append('ValueError')
    12. timeStamp = pd.Series(timestamp)
    13. data['Timestamp'] = timeStamp.values
    14. final_data = data.drop(['Date', 'Time'], axis=1)
    15. final_data = final_data[final_data.Timestamp != 'ValueError']
    16. final_data.head(
```

Output:

	Latitude	Longitude	Depth	Magnitude	Timestamp
0	19.246	145.616	131.6	6.0	-1.57631e+08
1	1.863	127.352	80.0	5.8	-1.57466e+08
2	-20.579	-173.972	20.0	6.2	-1.57356e+08
3	-59.076	-23.557	15.0	5.8	-1.57094e+08
4	11.938	126.427	15.0	5.8	-1.57026e+08

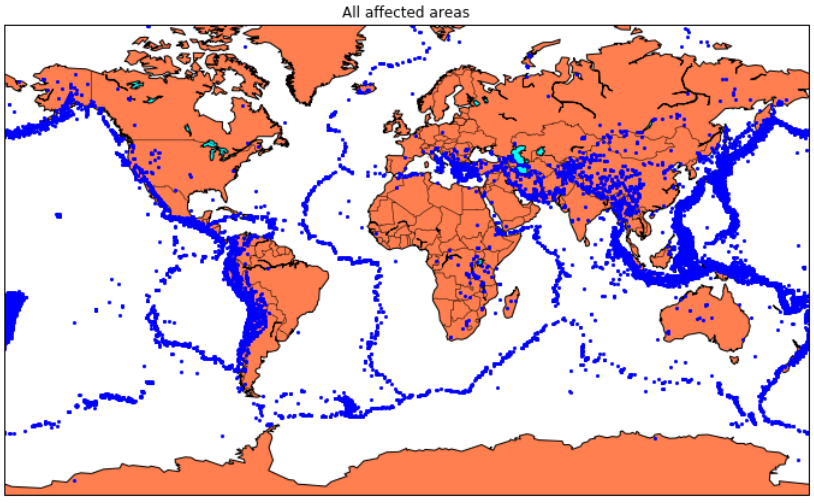
Visualization code :

```
from mpl_toolkits.basemap import Basemap

m = Basemap(projection='mill',llcrnrlat=-80,urcrnrlat=80, llcrnrlon=-180,urcrnrlon=180,lat_ts=20,resolution='c')

longitudes = data["Longitude"].tolist()
latitudes = data["Latitude"].tolist()
#m = Basemap(width=12000000,height=9000000,projection='lcc',
            #resolution=None,lat_1=80.,lat_2=55,lat_0=80,lon_0=-107.)
x,y = m(longitudes,latitudes)
fig = plt.figure(figsize=(12,10))
plt.title("All affected areas")
m.plot(x, y, "o", markersize = 2, color = 'blue')
m.drawcoastlines()
m.fillcontinents(color='coral',lake_color='aqua')
m.drawmapboundary()
m.drawcountries()
plt.show()
```

Output:



Neural Network Model :

```
from keras.models import Sequential
from keras.layers import Dense

def create_model(neurons, activation, optimizer, loss):
    model = Sequential()
    model.add(Dense(neurons, activation=activation, input_shape=(3,)))
    model.add(Dense(neurons, activation=activation))
    model.add(Dense(2, activation='softmax'))

    model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])

    return model
from keras.wrappers.scikit_learn import KerasClassifier

model = KerasClassifier(build_fn=create_model, verbose=0)

# neurons = [16, 64, 128, 256]
neurons = [16]
# batch_size = [10, 20, 50, 100]
batch_size = [10]
epochs = [10]
# activation = ['relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear', 'exponential']
activation = ['sigmoid', 'relu']
# optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']
optimizer = ['SGD', 'Adadelta']
loss = ['squared_hinge']

param_grid = dict(neurons=neurons, batch_size=batch_size, epochs=epochs, activation=activation, optimizer=optimizer, loss=loss)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1)
grid_result = grid.fit(X_train, y_train)

print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

Output :

```
Best: 0.957655 using {'activation': 'relu', 'batch_size': 10, 'epochs': 10, 'loss': 'squared_
hinge', 'neurons': 16, 'optimizer': 'SGD'}
0.333316 (0.471398) with: {'activation': 'sigmoid', 'batch_size': 10, 'epochs': 10, 'loss':
'squared_hinge', 'neurons': 16, 'optimizer': 'SGD'}
0.000000 (0.000000) with: {'activation': 'sigmoid', 'batch_size': 10, 'epochs': 10, 'loss':
'squared_hinge', 'neurons': 16, 'optimizer': 'Adadelta'}
0.957655 (0.029957) with: {'activation': 'relu', 'batch_size': 10, 'epochs': 10, 'loss': 'squ
ared_hinge', 'neurons': 16, 'optimizer': 'SGD'}
0.645111 (0.456960) with: {'activation': 'relu', 'batch_size': 10, 'epochs': 10, 'loss': 'squ
ared_hinge', 'neurons': 16, 'optimizer': 'Adadelta'}
```

Random Forest :

It is a type of machine learning algorithm that is very famous nowadays. It generates a random decision tree and combines it into a single forest. It features a decision model to increase accuracy. These trees divide the predictor space using a series of binary splits (“splits”) on distinct variables. The tree’s “root” node represents the entire predictor space.

The final division of the predictor space is made up of the “terminal nodes,” which are nodes that are not split. Depending on the value of one of the predictor variables, each nonterminal node divides into two descendant nodes, one on the left and one on the right.

If a continuous predictor variable is smaller than right will be the larger predictor points. The values of a categorical predictor variable Xi come from a small number of categories.

To divide a node into its two descendants, a tree must analyze every possible split on each predictor variable and select the “best” split based on some criteria. A common splitting criterion in the context of regression is the mean squared residual at the node.

a split point, the points to the left will be the smaller predictor points, and the points to the

Support Vector Classifier :

There is a computer algorithm known as a support vector machine (SVM) that learns to name objects. For instance, by looking at hundreds or thousands of reports of both fraudulent and legitimate credit card activity, an SVM can learn to identify fraudulent credit card activity.

A vast collection of scanned photos of handwritten zeros, ones, and other numbers can also be used to train an SVM to recognize handwritten numerals.

Gradient Boosting Algorithm :

To provide a more precise estimate of the response variable, gradient boosting machines, or simply GBMs, use a learning process that sequentially fits new models. This algorithm’s fundamental notion is to build the new base learners to have as much in common as possible with the ensemble’s overall negative gradient of the loss function. The loss functions used can be chosen at random.

However, for the sake of clarity, let’s assume that the learning process yields successive error-fitting if the error function is the traditional squared-error loss. In general, it is up to the researcher to decide on the loss function, and there is a wealth of previously determined loss functions and the option of developing one’s own task-specific loss.

Due to their high degree of adaptability, GBMs can be easily tailored to any specific data-driven activity. It adds a great deal of flexibility to the model design, making the selection of the best loss function a question of trial and error.

Model architectures. Additionally, the GBMs have demonstrated a great deal of success in a variety of machine learning and data mining problems in addition to practical applications.

Ensemble models are a helpful practical tool for various predictive tasks from the perspective of neurorobotics since they regularly deliver findings with a better degree of accuracy than traditional single-strong machine learning models.

To detect and identify human movement and activity, for instance, the ensemble models can effectively map the EMG and EEG sensor readings. These models, however, can also be incredibly insightful for memory simulations and models of brain development.

In contrast to artificial neural networks, which store learned patterns in the connections between virtual neurons, in boosted ensembles the base-learners act as the memory medium and successively build the acquired patterns, thereby enhancing the level of pattern detail.

Since the ensemble formation models and network growth strategies can be combined, advances in boosted ensembles can be useful in the field of brain simulation.

The ability to build ensembles with various graph properties and topologies, such as small-world networks, which are present in biological neural networks, will be possible in particular if the base learners are thought of as the network's nodes, which in the context of the connectome will mean the neurons.

It is crucial to first establish the technique and computational framework for these models before moving forward with sophisticated neurorobotics applications of boosted ensemble models.

Project Prerequisites :

The requirement for this project is Python 3.6 installed on your computer. I have used Jupyter notebook for this project. You can use whatever you want.

The required modules for this project are –

- Numpy(1.22.4) – pip install numpy
- Sklearn(1.1.1) – pip install sklearn
- Pandas(1.5.0) – pip install pandas

That's all we need for our earthquake prediction project.

1. Import the modules and all the libraries we would require in this project.

```
import numpy as np#importing the numpy module

import pandas as pd#importing the pandas module
from sklearn.model_selection import train_test_split#importing the train test split module
import pickle #import pickle
from sklearn import metrics #import metrics
from sklearn.ensemble import RandomForestClassifier#import the Random Forest Classifier
```

2. Here we are reading the dataset and we are creating a function to do some data processing on our dataset. Here we are using the numpy to convert the data into an array.

```
dataframe= pd.read_csv("dataset.csv")#here we are reading the dataset
dataframe= np.array(dataframe)#converting the dataset into an numpy array
print(dataframe)#printing the dataframe
```

3. Here we are dividing our dataset into X and Y where x is the independent variable whereas y is the dependent variable. Then we are using the test train split function to divide the X and Y into training and testing datasets. We are taking the percentage of 80 and 20% for training and testing respectively.

```
x_set = dataframe[:, 0:-1]#getting the x dataset
y_set = dataframe[:, -1]#getting the y dataset
y_set = y_set.astype('int')#converting the y_set into int
x_set = x_set.astype('int')#converting the x_set into int
x_train, x_test, y_train, y_test = train_test_split(x_set, y_set, test_size=0.2, random_state=0)
```

4. Here we are creating our RandomForestClassifier and we are passing our training dataset to our model to train it. Also then we are passing our testing dataset to predict the dataset.

```
Random_forest_classifier = RandomForestClassifier()#creating the model
random_forest_classifier.fit(x_train, y_train)#fitting the model with training dataset
y_pred = random_forest_classifier.predict(X_test)#predicting the result using test set
print(metrics.accuracy_score(y_test, y_pred))#printing the accuracy score
```

5. In this piece of code, we are creating our instance for Gradient Boosting Classifier. The maximum Depth of this Gradient Boosting algorithm is 3. After creating the instance, we pass our training data to the classifier to fit our training data into the algorithm. This is a part of training.

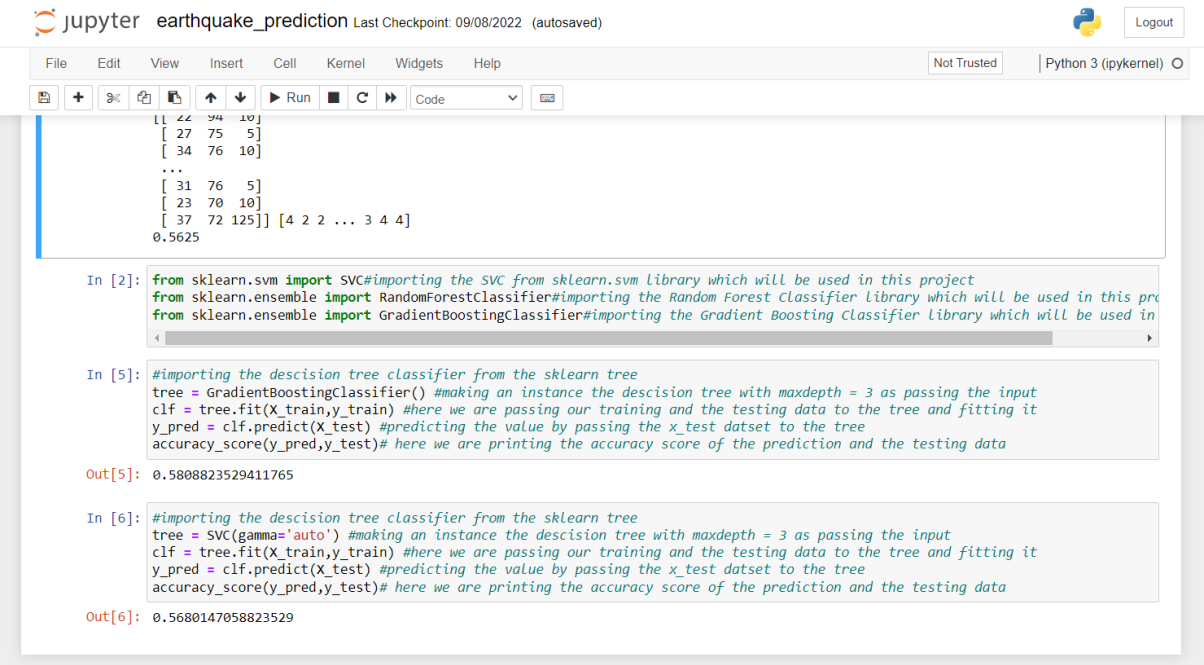
Once we are done with training, we pass the testing data, and we make our predictions on testing data and store it in another variable. After training and testing, it's time to print the score. For this, we are using the accuracy score function, and we are passing the predicted values and the original values to the function, and it is printing the accuracy.

```
#importing the descision tree classifier from the sklearn tree
tree = GradientBoostingClassifier() #making an instance the descision tree with maxdepth = 3 as passing the input
clf = tree.fit(X_train,y_train) #here we are passing our training and the testing data to the tree and fitting it
y_pred = clf.predict(X_test) #predicting the value by passing the x_test datset to the tree
accuracy_score(y_pred,y_test)# here we are printing the accuracy score of the prediction and the testing data
```

6. Here, we are doing the same thing as above. The only difference is that this time, we are using Support Vector Classifier. So, we are creating an instance of Support Vector Classifier and setting the gamma function to "auto". After that, we pass the training data to the classifier.

After training the model, we pass the testing data to our model and predict the accuracy score using the accuracy score function.

```
#importing the descision tree classifier from the sklearn tree
tree = SVC(gamma='auto') #making an instance the support vector tree
clf = tree.fit(X_train,y_train) #here we are passing our training and the testing data to the tree and fitting it
y_pred = clf.predict(X_test) #predicting the value by passing the x_test datset to the tree
accuracy_score(y_pred,y_test)# here we are printing the accuracy score of the prediction and the testing data
```



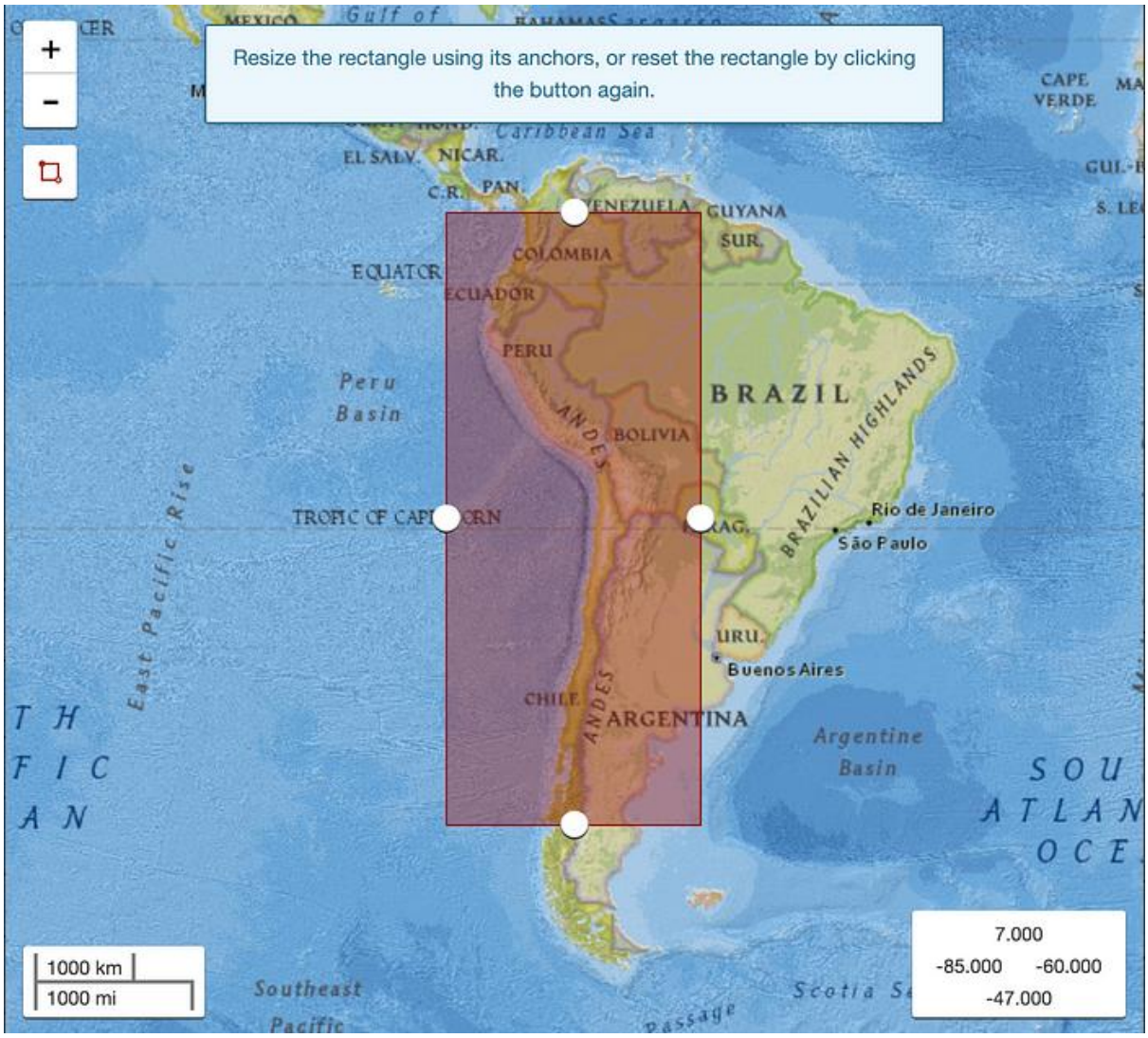
Data :

The raw data was sourced from The United States Geological Survey (USGS) [earthquake catalog](#) .

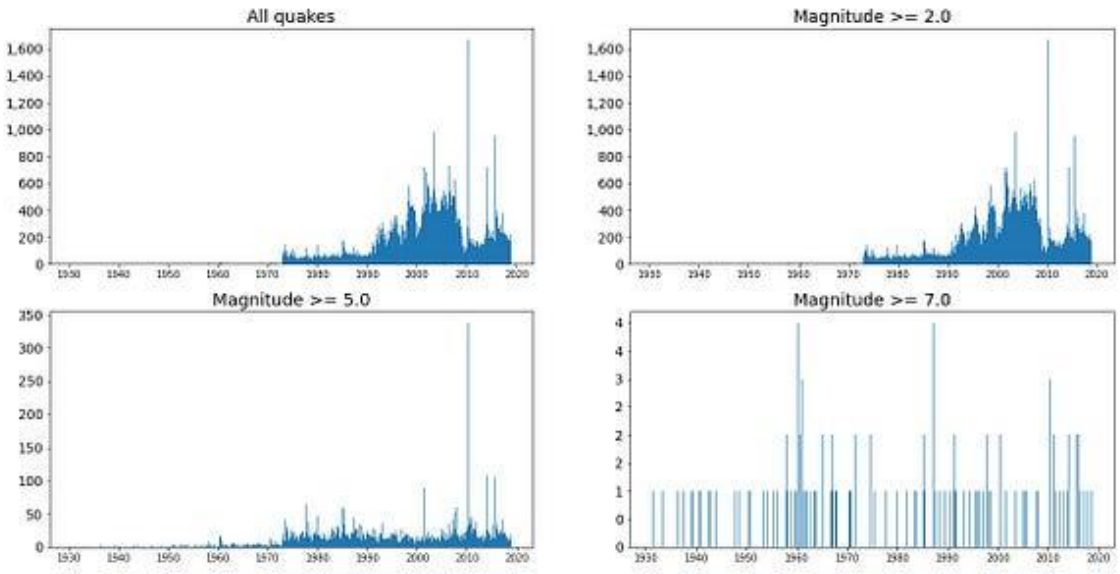
All worldwide earthquakes from the beginning of records until the end of 2018 were downloaded and later filtered as described below.

Id	Date	Latitude	Longitude	Depth	Magnitude
iscgemsup907200	18/01/1930	-4.61	153.18	35	6.5
iscgem907212	02/02/1930	51.39	179.82	25	6.4
iscgem907224	14/02/1930	-21.87	-175.10	35	6.4
iscgem907259	06/03/1930	-33.29	-178.01	15	6.3
iscgem907286	26/03/1930	-7.74	125.81	10	7.0

The Nazca-South American plate boundary area was selected (latitude between -47° and 7°, and longitude between -85° and -60°).



Years between 1973 and 2018 were selected. Comparing the histograms for the number of earthquakes across dates, there is a clear increase in the number of recorded events, mostly for lower magnitudes. This is most likely due to an increase in the number of seismometers, not an actual increase in the number of quakes.



Modelling :

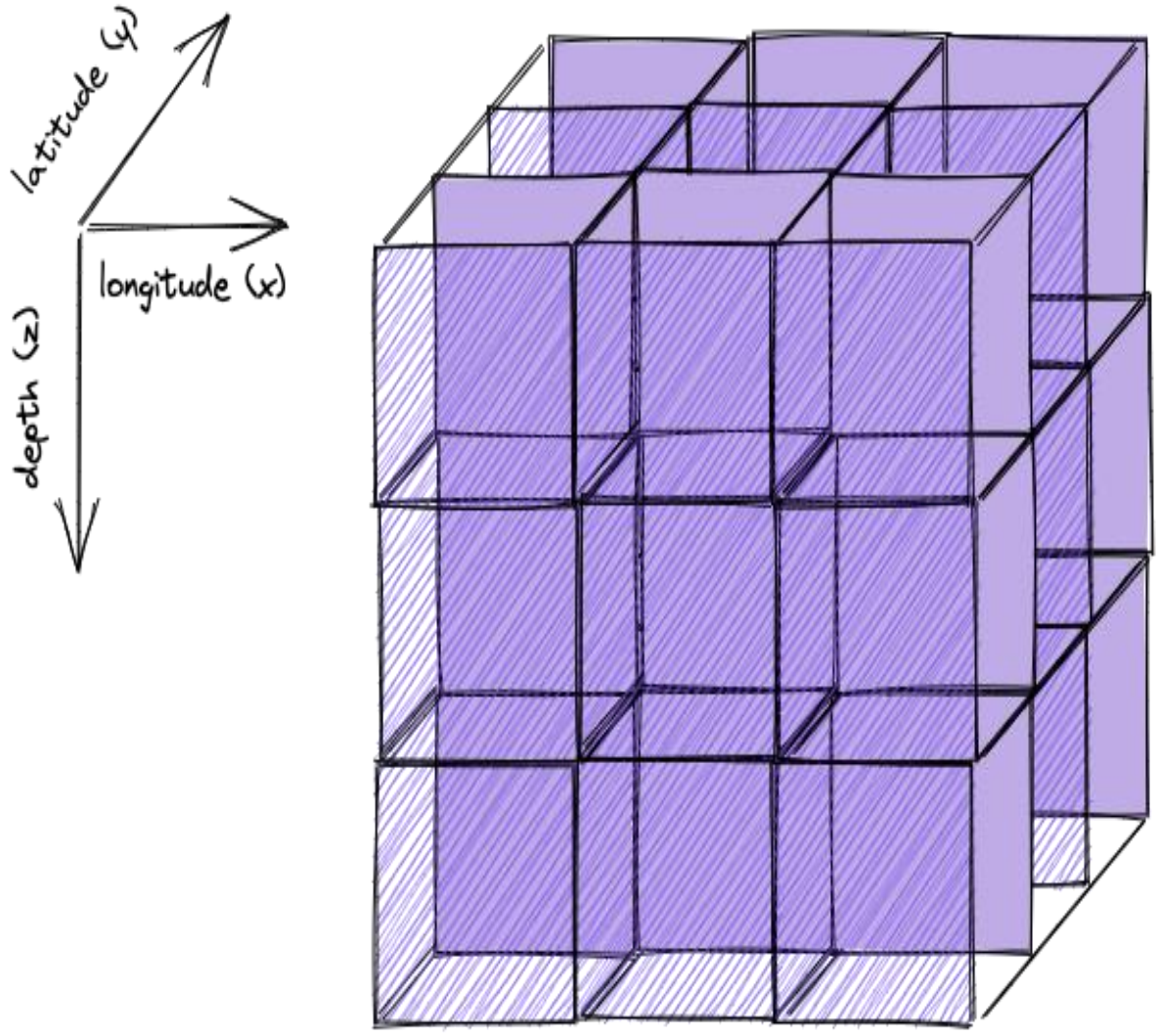
Problem statement :

Instead of using the table of recorded earthquakes to appraise the final model, a different approach was selected.

If the goal is to build a warning system capable of predicting the earthquake risk for any time period and specific areas, in my opinion, a fairer assessment of results is more complex.

We need to replicate a real scenario in our dataset and **add the time periods with no seismic events** in our time frame. That ensures that we will evaluate the predictions even when there is no earthquake.

First, the selected area was divided into a 3D grid. The spatial resolution dimensions selected were 10 degrees latitude, 12 degrees longitude, and 100 km depth.



Second, the data was grouped time-wise for two periods, thus having two final models. One with 7 days (**weekly model**) and another with 1 day (**daily model**). It was also added a range warning of periods, 2 for the **weekly model** and 3 for the **daily model**.

For example, for the **daily model**, if an earthquake will happen on Friday, not only Thursday evening it should make an alert, but also Wednesday and Tuesday. For the **weekly model** , if an earthquake will happen on the third week of a given month, it should make alerts on the first and second weeks.

Lastly, for the actual type of alert, it was chosen to warn for every quake with a magnitude (M) greater than or equal to **5.0**. This magnitude level was chosen because it can cause damage if is close to a population centre (not only regarding the latitude-longitude plane but also the depth, being or not close to the surface).

Even earthquakes with lower magnitudes have already caused deaths, e.g. 4.9 M Afghanistan 1997 killed 15 people , although that is uncommon. Starting from that level, they can become even more destructive, e.g. 5.3 M Tajikistan 1989 killed 274 people .

x	y	z	Date	Quake	Target Event
-61	-17	0	07/02/1979	0	0
-61	-17	0	08/02/1979	0	0
-61	-17	0	09/02/1979	0	1
-61	-17	0	10/02/1979	0	1
-61	-17	0	11/02/1979	0	1
-61	-17	0	12/02/1979	1	0
-61	-17	0	13/02/1979	0	0

Preprocessing and feature engineering :

The core of this modelling is to track energy dispersion. Hence all earthquakes, above or not the selected warning level, were transformed into energy, allowing us to group different events on the same 3D grid point (that cannot be done with magnitude: two events of M = 3 are not the same as one of M = 6).

$$\text{Log } Energy = 5.24 + 1.44\ Magnitude$$

The following step is to reduce the data according to the problem statement. From that is yielded the energy for each point, for every time period, filling periods with no events with 0 energy.

With a well-formatted x-y-z-t dataset, the feature engineering process can be done. Features created are energy moving averages (periods of 30, 60, 90, 180, 330, and 360), ratios of those M. A., and also the moving average for the neighbours' 3D data points. Those last features are created to try to capture the relation of energy between close points.

Another feature created is the tracking of days from the last event, an attempt to capture the frequency of events.

```
* weekly modelBalance: 7.10%
Number of records: 106,265
Number of features: 18* daily modelBalance: 1.76%
Number of records: 744,294
Number of features: 18
```

Model selection :

Since this is a highly imbalanced problem, a better metric to be used is the F-score, which is the weighted harmonic mean between **precision** and **recall**.

In one of my previous articles [18], I explain the difference between those metrics. Precision is penalised from false alarms, and recall is penalised from missed events.

The data split was 90% train and 10% test.

Weekly model

	Records	Balance	Events
Train	95,181 (90%)	6.95%	6,612
Test	11,084 (10%)	8.46%	938

Daily model

Records	Balance	Events
666,688 (90%)	1.72%	11,458
77,606 (10%)	2.16%	1,677

Within the training data, a grid search was performed to find the best models, using a 3-fold cross-validation time-series.

```
############################
# linear models
############################lin_params = {
    'C' : [0.01, 0.1, 1.0],
    'solver': ['lbfgs', 'newton-cg']
}##########################
# random forest models
##########################rft_params = {
    'max_depth': [6, 7, 8],
    'n_estimators': [25, 50, 75, 100, 150],
}##########################
# xgboost models
##########################xgb_params = {
    'max_depth': [5, 6, 7],
    'n_estimators': [15, 25, 35],
}##########################
# additional preprocessing
# all features have the NaN filled and inf values clipped
# for linear models, standard scaling is applied
##########################additional = [None, iforest, kmeans]
```

Cross-validation results:

Weekly model

F0.5	Overfit	Precision	Recall	ROC AUC	Type	Max depth	# estimators	Preprocessing
35%	4%	37%	29%	0.852197	Xgboost	5	15	iforest | kmeans
34%	4%	38%	24%	0.852122	Xgboost	5	15	kmeans
34%	5%	35%	33%	0.853439	Random Forest	7	50	None
33%	4%	37%	26%	0.85451	Random Forest	6	150	kmeans
33%	4%	36%	29%	0.854937	Random Forest	6	25	kmeans

Top 5 cross-validation results for the weekly model. Mean values. Overfit from F0.5 score. Image by author

Daily model

F0.5	Overfit	Precision	Recall	ROC AUC	Type	Max depth	# estimators	C	Solver	Preprocessing
19%	-4%	24%	11%	0.849364	Random Forest	6	25	-	-	kmeans
18%	-4%	23%	11%	0.849544	Random Forest	6	50	-	-	None
18%	-4%	23%	11%	0.849627	Random Forest	6	75	-	-	None
18%	-4%	23%	11%	0.849759	Random Forest	6	100	-	-	None
18%	-7%	28%	15%	0.824601	Linear	-	-	1.0	lbfgs	iforest | kmeans

Results and discussion :

With the best model, the results in the test dataset were determined:

	F0.5	Overfit	Threshold	Precision	Recall	ROC AUC	Type	Max depth	# estimators	Pre-processing
Weekly	36%	-1%	81%	38%	40%	0.835625	Xgboost	5	15	iforest | kmeans
Daily	24%	-7%	84%	31%	12%	0.837831	Random Forest	6	25	kmeans

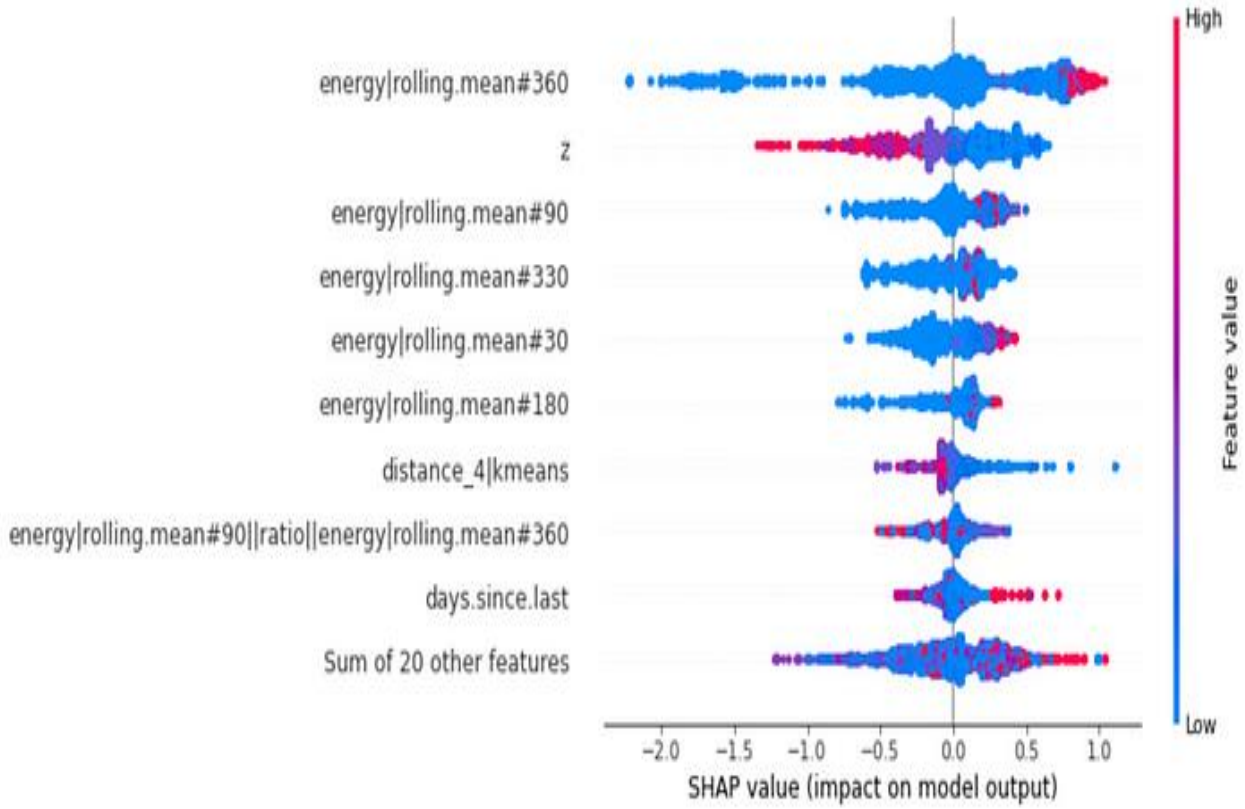
Final results for each model. Results are better than the cross-validation data because in the latter not all 90% training data is used for training, there is a fold for testing. Image by author.

The precision and recall levels achieved for the weekly model are acceptable. Perhaps not suited for alerting the general population of some area. But it can be helpful for either governments or high-risk installations (e.g. nuclear power plants) to plan and be in a better preparedness state.

Confusion matrix and Shapley values are also presented:

Actual \ Predicted	Event	No Event
Event	376	562
No Event	614	9,532

Weekly model confusion matrix. Image by author.



Next steps :

Because this is an initial study, there are a lot of areas for potential improvement and exploration.

Granularity :

Both space and time can be differently discretised, but if it unbalances the model golden source (for example, increasing the spatial resolution) it will perform most likely worse. That also includes the range warning.

The dataset balance is probably one of the most important factors.

Problem statement :

The alert level of 5.0 can be changed, but if set for higher magnitudes, it will decrease the dataset balance, affecting its performance.

An alternative to a binary classification problem is regression, targeting the energy. Then, if the predicted energy reaches a certain level, an alert is produced.

Energy :

Currently, the energy calculation is an approximation, but in reality, it differs depending on the type and level of magnitude. This needs to be analysed if impacts the model.

Adding features :

Moving standard deviations are the next batch of features to be tested.

Geomagnetic field :

Adding more information, like the magnetic field from IMOs, can perhaps improve the results. The data needs to be sourced, transformed and engineered to verify if there is any hidden pattern that could improve the results.

Regions :

Other regions can be explored, and transfer learning is a possibility.

Models :

Neural Networks and Autoencoders can be tested. 3D Convolutional Neural Networks are also an option, making the location (x-y-z) part of the architecture.

Hyperparameter :

More advanced and modern hyperparameter search strategies should be deployed, like Bayesian optimisation.

Explainability :

XAI can be more explored since here only the basics were covered. More conditional features relations can be inspected, like in here [19].

Sampling :

Oversampling and undersampling can also be explored.

GCP pipeline :

All steps can be automated, from data sourcing to hyperparameter search, and automated retraining.

Proposed Methodology:

The proposed model uses latitude, longitude and size to predict depth. The dataset for training the model is extracted from Kaggle and contains data from 1965 to 2016. Use the data trained by the Random Forest regressor to predict depth of earthquakes.

Evaluate model performance using root mean square errors (MSE), root mean square errors (RMSE), and R2 scores. The Random Forest algorithm is a supervised learning technique that used for classification and regression problems in the field of machine learning. The Random Forest algorithm is the widely used technique for regression problems.

The true random regression algorithm is randomly generated and decisions are made through generated random draws. Random Forest Regression calculates the average of all predictions to produce a good estimate of the expected depth of the earthquake.

Random Forest Regression is used to predict real-time methods and business criteria and can be used for future product prices/costs, revenue forecasting, and performance comparison. It offers excellent accuracy, scales data well, is interpretable, and easy to use.

The proposed model consists of the following steps:

1. Data collection: We collected a dataset from Kaggle containing all earthquakes with magnitude greater than 5.5 from 1965 to 2016.

2. Data pre-processing: In this step, we cleaned the data, removed null values and removed unnecessary columns.

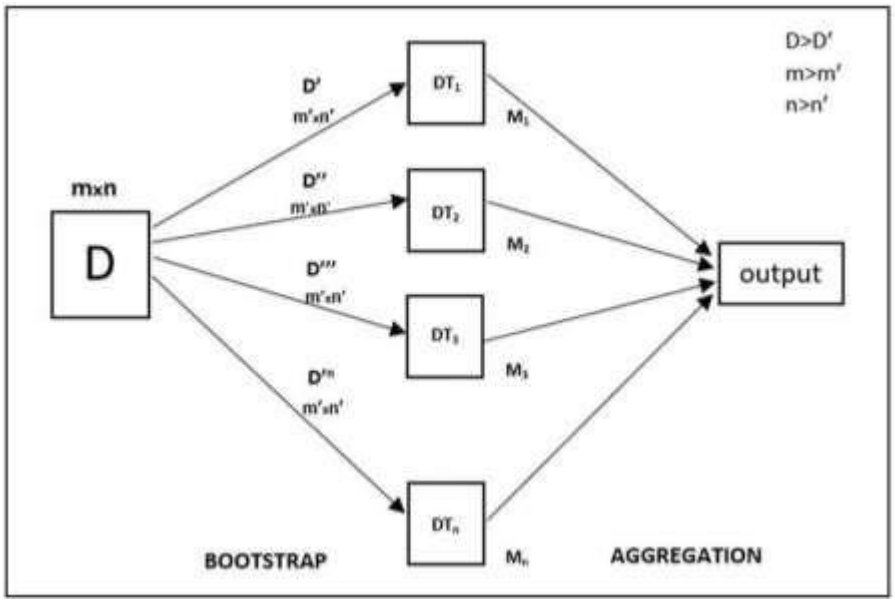
3.Split Data: In this step the dataset is split for training and testing 3. Feature scaling: In this step, we perform standard scaling on the dataset to normalize the independent variables in the dataset.

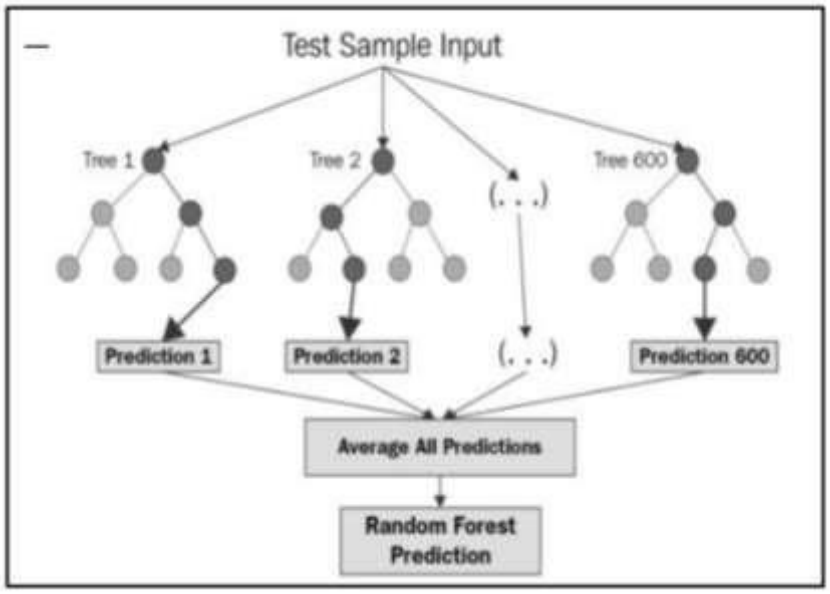
4. Implementation of Algorithm : Here we apply Random Forest Algorithm to train the model to get good accuracy in predicting earthquake depth. The model test accuracy was 98% and the model training accuracy was 88%.

Algorithm Comparison:

Here we compare various kinds of machine learning algorithms such as Multiple Linear Regression, KNN Regression, Decision Trees, and Support Vector Regression. We use the Random Forest algorithm because it offers good accuracy compared to other algorithms. Calculate various metrics such as R2 score, root mean square error for each model.

5. Evaluation of the model: We tested the performance of the model using mean squared errors, mean absolute errors, and R2 score. The training accuracy using the random forest regressor is 0.979053, the R2 score is 0.8744, and the value of root mean square error is 44.877, the average absolute error value is close to 5.


D>D'
m>m'
n>n'
mxn
D
D'
m'xn'
D''
m'xn'
D'''
m'xn'
D'n
m'xn'
DT1
DT2
DT3
DTn
M1
M2
M3
Mn
output
BOOTSTRAP
AGGREGATION

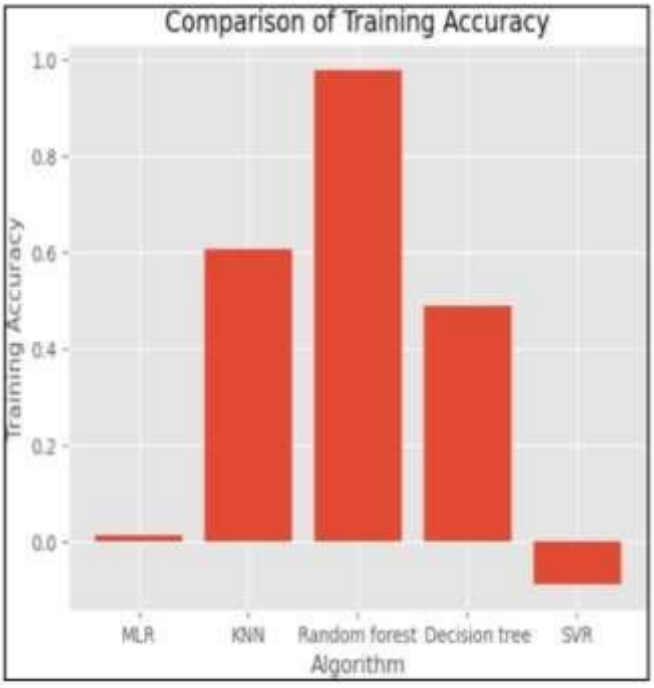

Test Sample Input
Tree 1
Tree 2
(. . .)
Tree 600
Prediction 1
Prediction 2
(. . .)
Prediction 600
Average All Predictions
Random Forest Prediction

Results and Discussions:

We have verified the proposed model using the dataset collected from Kaggle website and the results showed that proposed model obtained a good accuracy in predicting the depth of earthquake. The model obtained an accuracy of 88% in testing the model and 98% accuracy in training the model.

The random forest regressor has a R2 score value of 0.8752, In result comparison module different ml algorithms like multiple linear regressions, KNN regressions, Decision trees and Support vector regression are used and when these algorithms are trained and accuracy is tested, multiple linear regressions gave R2 score of 0.011852, KNN regressions gave R2 score of (0.6080), Decision trees gave R2 score of (0.5165) and support vector regressions gave R2 score(- 0.09170).

The mean square error is low by using the random forest algorithm that is 44.877. The model can predict the depth of the earthquake.



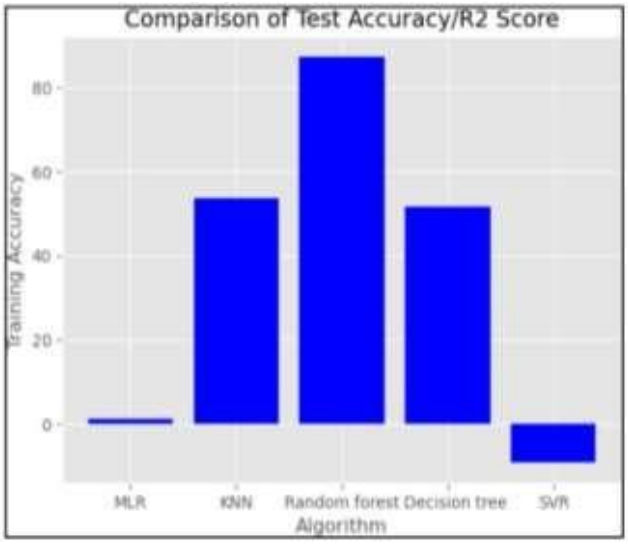


Fig 4Comparision of testing

S.No	ALGORITHM USED	TRAINING ACCURACY	R2 SCORE	MEAN SQUARED ERROR
1	Multiple Linear Regression	0.01182	0.011852	126.1790
2	KNN Regression	0.6080	0.538007	86.27675
3	Random Forest Regression	0.979053	0.8744	44.877
4	Decision Tree	0.4896	0.5165	88.259
5	Support Vector Regression	-0.08850	-0.09170	132.6261

Fig 5.Comparison table of various

Fig 6. Depth predicted at

Fig 7Homepage of the model

Conclusion and Future work:

This paper compares several machine learning algorithms for good accuracy in predicting the depth of earthquakes. Random forest regressor has good accuracy compared to other machine learning algorithms in predicting earthquake depth.

The proposed model takes into account various geographical factors to predict the depth of earthquakes. The proposed model can be used to identify the areas most vulnerable to earthquakes and take appropriate measures to protect human life and infrastructure.

Currently we have used machine learning to predict depth, in the future we may use deep learning techniques to provide more accurate results than this algorithm. And the dataset can also be improved in the future, possibly with more columns of data, providing more data for prediction. It is impossible to develop a model that can satisfy all the necessary needs required by the user. Results can be further improved by including more parameters as prediction inputs.