Earthquakes prediction model using python

Phase 3: Development part 1

Table of contents

Earthquakes basic concepts

Earthquakes are well-studied events, with plenty of academic studies coverage, so only the basic concepts will be described here.

The majority of seismic activity happens between the movement of lithospheric plates (a.k.a. tectonic plates). This movement accumulates energy in the form of rock stress, and then it is suddenly released.

After the quake happens, it can be determined the location (longitude, latitude, and depth), time, and magnitude. Magnitude is the physical size of the earthquake, and the energy released can also be roughly estimated by converting the moment magnitude [1].

Earthquakes can cause destruction and loss of lives. Not only by the ground shaking event but also by secondary effects such as landslides, fissures, avalanches, fires and tsunamis [2].

Between 1998–2017, earthquakes caused nearly 750,000 deaths globally, more than half of all deaths related to natural disasters. More than 125 million people were affected by earthquakes during this time period, meaning they were injured, made homeless, displaced or evacuated during the emergency phase of the disaster.

- *World Health Organization*

Building a pre-emptive warning system can greatly increase risk management effectiveness. Being able to prepare for those rare events would help to minimise the harm caused, with actions such as local community alert and government provisioning.

Related studies

To the best of my knowledge, 2 studies try to predict when the next earthquake will happen with machine learning [3][4]. Both conclude that is very difficult to predict the next occurrence, due to its randomness and difficulty to prove that earthquakes follow a specific pattern.

It is important to note that both studies use the table of recorded earthquakes to build the machine learning model. Please refer to the Problem statement sub-section for further discussion.

Other ML applications have also been explored:

A study achieved nice results focusing on predicting aftershock events, which happen after larger ones, and is an important subject since aftershocks cause a lot of damage as well [5]. Some discussions have arisen about the data science methodology used [6][7][8].

Laboratory earthquakes experiments are studied through the application of ML trying to predict time to failure [9][10].

Another paper found patterns in energy signals from low-amplitude seismic waves to the timing of slow slip events [11].

The lateral spreading prediction has been explored [12]. A competition for modelling earthquake damage has also been held [13].

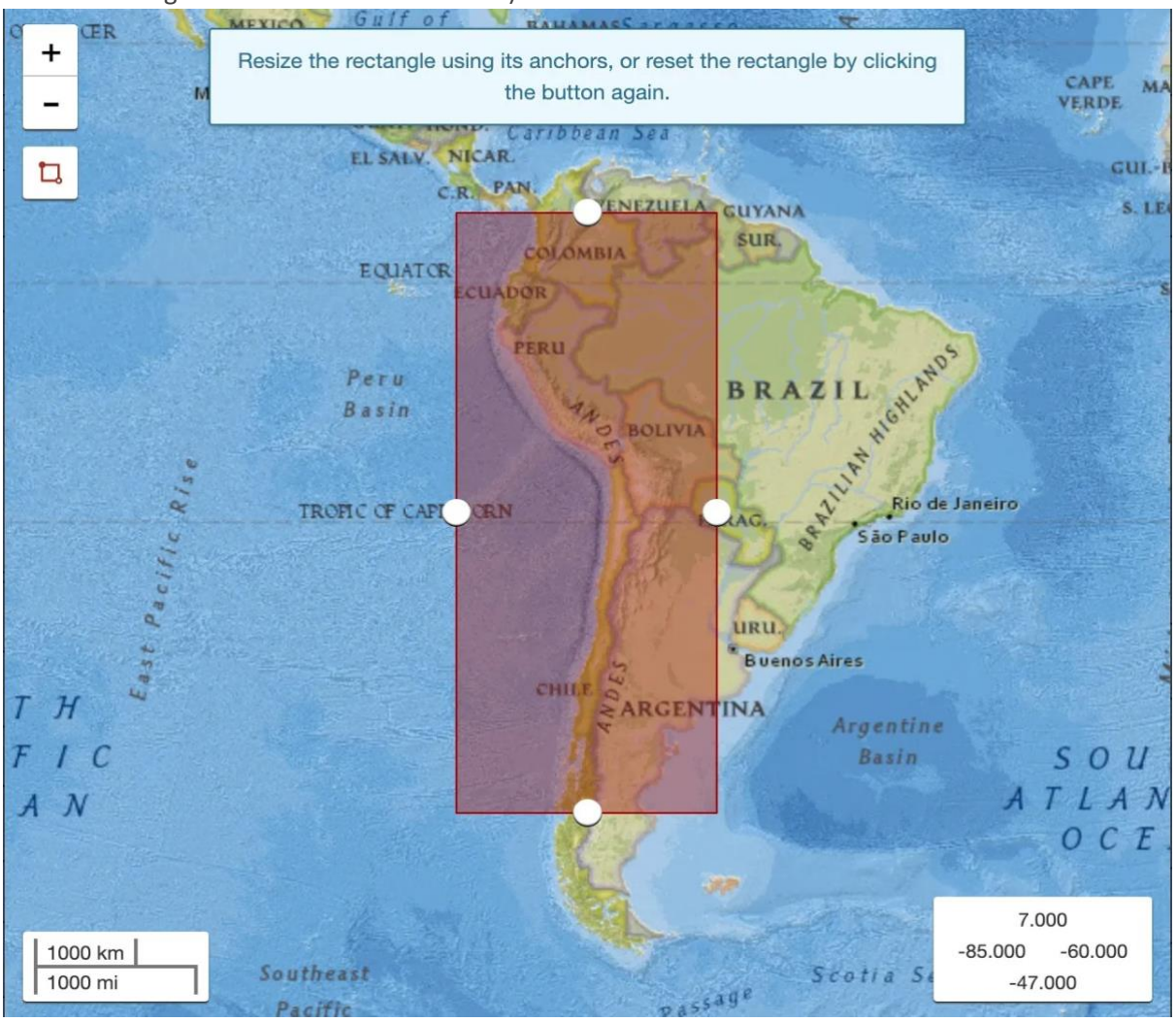
Earthquake detection and phase picking automation [14].

Data

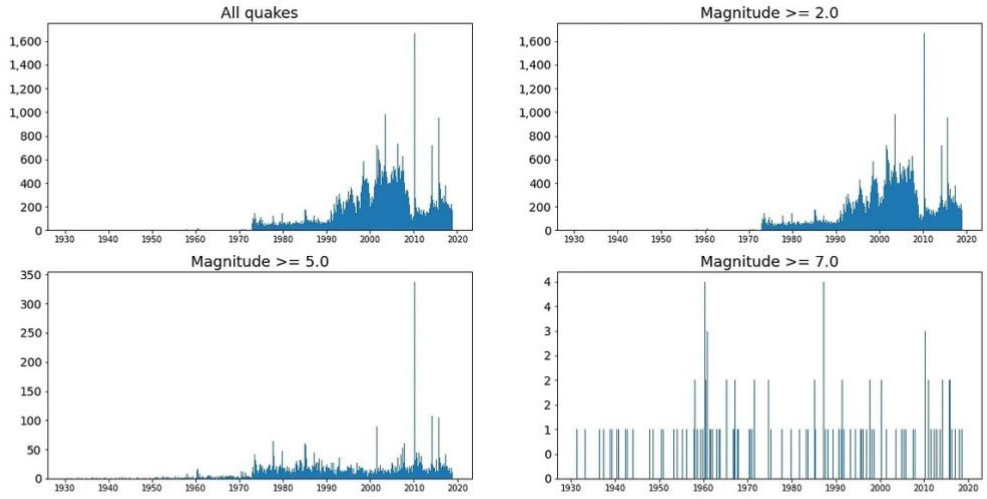
The raw data was sourced from The United States Geological Survey (USGS) earthquake catalog [15]. All worldwide earthquakes from the beginning of records until the end of 2018 were downloaded and later filtered as described below.

Id	Date	Latitude	Longitude	Depth	Magnitude
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
iscgem907212	02/02/1930	51.39	179.82	25	6.4
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
iscgem907259	06/03/1930	-33.29	-178.01	15	6.3
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

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



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

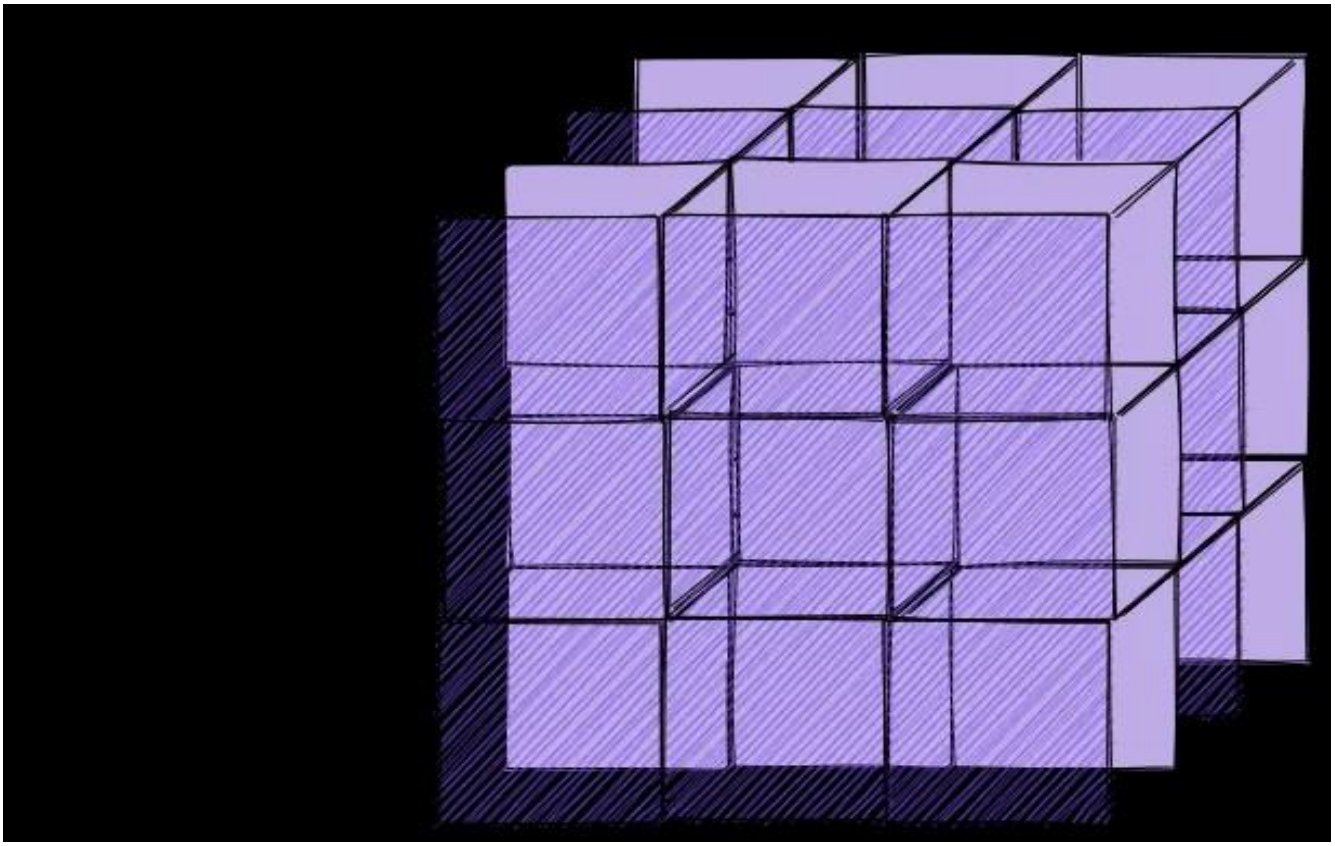
Problem statement

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.

The following steps were done to achieve that.

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 [16], although that is uncommon. Starting from that level, they can become even more destructive, e.g. 5.3 M Tajikistan 1989 killed 274 people [17].

In summary, the data was translated into a time-series binary classification problem for every point in the 3D grid (x-y-z-t).

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).

$$Log\ energy = 5.24 + 1.44\ Magnitude$$

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.

The resulting datasets characteristics are displayed here:

- *Weekly model*

Balance: 7.10%

Number of records: 106,265

Number of features: 18

- *Daily model*

Balance: 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.

Records	Balance	Events
[illegible]	[illegible]	[illegible]
11,084 (10%)	8.46%	938

Records	Balance	Events
[illegible]	[illegible]	[illegible]
77,686 (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]
```

F0.5	Overfit	Threshold	Precision	Recall	ROC AUC	Ty
[redacted]						
24%	-7%	84%	31%	12%	0.837831	Ran

[redacted]								
33%	4%	37%	26%	0.85451	Random Forest	6	150	kmeans
[redacted]								

Cross-validation results:

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

Top 5 cross-validation results for the daily model. Mean values. Overfit from F0.5 score

Results and discussion

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

F0.5	Overfit	Precision	Recall	ROC AUC	Type	Max depth	# estimators	C	Solver	Preprocessing
[redacted]										
18%	-4%	23%	11%	0.849544	Random Forest	6	50	-	-	None
[redacted]										
18%	-4%	23%	11%	0.849759	Random Forest	6	100	-	-	None
[redacted]										

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.

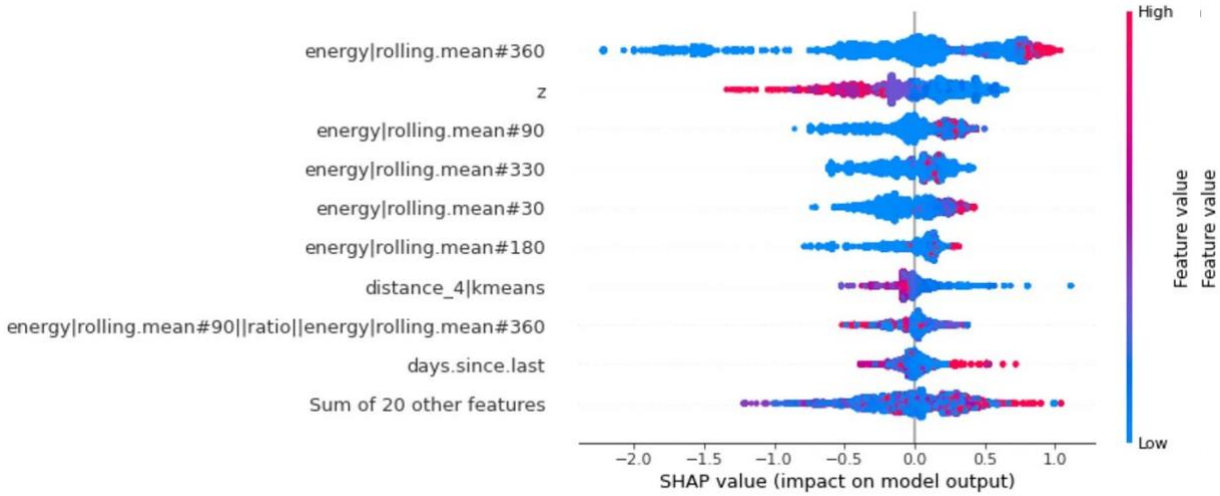
For a Proof of Concept, if put in perspective of the challenge (highly imbalanced), especially for the precision, results are reasonable.

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	[redacted]	[redacted]
No Event	[redacted]	[redacted]

Weekly model confusion matrix



Shapley values for the weekly model