Mini-Project (ML for Time Series) - MVA 2024/2025

Antoine Sicard antoine.sicard96@gmail.com
Nil Mostefai Danymost@protonmail.com

January 8, 2025

1 Introduction

Earthquakes are the consequence of the brutal rupture and displacement of lithosphere compartments linked to constraint accumulations. The energy release takes the form of different seismic waves which propagate from the core and can cause devastator effects on landscapes and human activities. The first detected waves are the P waves. They are compression-dilation waves, which move particles parallel to the direction of propagation. Then come S waves, which displace particles perpendicularly to the direction of propagation (Figure S1) [2, 5].

The most destructive waves are the latest to arrive, making early earthquake detection a critical task in seismology, as it could prevent many damages with an early detection [1]. It is yet still a challenge since earthquakes can vary a lot, in term of amplitude, fequency and magnitude. We found in the literature that deep learning methods are the state-of-the-art in terms of earthquake detection [3]. However, more traditional machine learning methods are also a huge subject of interest due to their more direct interpretability their good accuracy, and their easier trainability, which are high quality criteria especially for geologists working directly on the field. For these reasons, we decided to develop a traditional machine learning method to perform earthquake classification.

Dictionary learning consists in learning a sparse representation of a signal [6]. This method seems well suited for our problem and several research studies have performed dictionary learning on seismic data [9, 7]. This method could be used to perform classification by learning one representation (i.e. one dictionary) of the wave signal per class. But performing dictionary learning directly on raw seismic waveform could be inefficient due to the high variability of seismic signals [8]. The study by Zhou et al., 2019 [8] proposed an innovative approach that combines feature selection with dictionary learning. In their study, they segmented the signal into time windows and they extracted a meticulous feature list selected via multiple selection metrics. They then built separate dictionaries for seismic and non-seismic events and they were able to classify accurately earthquake from non-earthquake events.

However, their approach is applied on an artificial dataset of seismic waves, simulated in a laboratory. In addition, they didn't make any distinction between the different types of waves (notably P or S waves). In this report, we try to adapt their methodology to detect and classify different kins of seismic waves on a real dataset.

Contributions: we both equally contributed to the design of the project, literature review, code production, result analysis and writing of the report. There was no code provided with the article so we needed to write the entire code (except for some used libraries which are explicitly listed in the attached notebook). We tried to implement the method from the article [8] on a real seismic dataset with a focus on identifying P and S seismic waves.

2 Method

2.1 Features generation and selections

In this scetion, we present our method to geenrate the features from the signal. As mentioned in [8], two types of features can be generated: experiential features (expert knowledge), and features generated using the TSfresh Python package, which automatically extracts a large set of features from time series. However, the latter are typically less interpretable or meaningful. A main limitation of the article, is that the authors don't mention in details which experiential features they used and the only one they spoke about can already be computed with TSFresh. For these reasons, in our study, we only relied on TSFresh features.

The TSFresh package extracts more than 700 features and it is so necessary to do a selection among these features in order to compress the dimension of the input data, reduce irrelevant or redundant information and ensure a good accuracy of machine learning models. To select high-quality features, Zhou et al., 2019 [8] used 2 kinds of selection methods: filter methods and wrapper methods. Filtering the features consists in raking them accordingly to defined criteria, and select only the the best ones using a manually defined threshold. They used 3 different filtering selection methods: ReliefF, Gini index and Kullback-Leibler Divergence. For each of them, they only selected the top 75% features. For our study, we chose to only focus on ReliefF, which is particularly powerful for complex data. RefielfF is an improvement of the Refielf method. Briefly, for a dataset of N points of D features, the Relief algorithm attributes to each feature a weight that is low when the feature is more different to its nearest neighbor of the same class than to its nearest neighbor for a different class, meaning that this feature is highly variable within a same class, and should not be used to discriminate. However, ReflieffF is computational demanding and takes time to run on our own student laptopseven when using the one from the specialized "skrebate" library. To overcome this point, we also decided to use a built-in feature selection method provided directly by the TSFresh library to further streamline the process. This method relies on multiple statistical tests, incorporating a correction to control the false discovery rate (FDR) and balance the curse of dimensionality. Features with p-values below a defined threshold are then selected as relevant. Wrapper methods are computationally demanding because they require training a machine learning model multiple times to evaluate the relevance of different subsets of features. For this reason, we only chose to focus on filtering methods in our study.

2.2 Dictionary learning

The principle of dictionary learning consists in learning an accurate sparse representation of the data, using a linear combination of a set of basis vectors (called the atoms) [6]. More precisely, for M time series signals x_1, x_2, ... x_M of length N organized in a matrix $\mathbf{X}$ of dimension $N \times M$, the objective is to learn a matrix $\mathbf{D}$ (the dictionary) of dimension $N \times K$ and a matrix $\mathbf{Z}$ of dimension $K \times M$ that gathers all the activation vectors, one for each time series. The problem to solve is:

$$(\mathbf{D}^*, \mathbf{Z}^*) = \arg\min_{\mathbf{D},\mathbf{Z}} \|\mathbf{X} - \mathbf{D}\mathbf{Z}\|_F^2$$

$$\text{subject to } \forall k, \|\mathbf{d}_k\|_2 \leq 1; \forall k, \mathbf{z}_k \text{ is sparse.}$$

A strategy to solve this problem consists in alternating the resolution of two steps iteratively: solve a sparse coding problem with known **D** and solve a dictionary learning problem with known **Z**.

- Solving the sparse coding problem with known **D**. In this step the dictionary **D** is fixed and the objective is to learn **Z**. This problem can be solved using several available methods. The one used by the authors in the article is the OMP (Orthogonal Matching Pursuit) algorithm. We propose an implementation of it in the notebook, so we are not going to expand on it in the report.

- Solving the dictionary learning process with known **Z**. In this step, **Z** is fixed and we want to learn **D**. The details of this step are not provided in the article, so we decided to implement a proximal gradient descent method (see attached notebook).

As we were working on our own laptops, we had access to a limited computational resource. In order to be able to compute the dictionary learning on our dataset, we used the dictionary learning function from scikit-learn while training our real models (see attached notebook). Dictionary learning can be used for many applications. What interests us in this project is to use it to perform classification. It consists in learning as many dictionaries as there are classes. During inference, the signal is "represented" in each dictionary. The class corresponding to the lowest residual is then assigned to the input signal. In our case, we are interested in differentiating pre-seism noise, P waves, S waves and post-seism noise. We are going to learn 4 distinct dictionaries.

As we previously said, the originality of the approach described in the article that dictionary learning is not applied directly to the raw signals, but rather to feature vectors extracted from the data, segmented into time windows [8]. Consequently, each feature vector is represented sparsely in the dictionary.

3 Dataset

We used the STEAD dataset [4], a high-quality, rich and versatile dataset containing 19000 hours of earthquake signals which have been registered from numerous different locations at epicentral distances up to 300 km. Moreover, each earthquake waveform is accompanied by information about the earthquake (magnitude, epicentral distance, etc.) and the three most important labels that interest us in order to classify different types of seismic waves: P-phase, S-phase, Coda-end which are respectively the arrival time of P wave, the arrival time of S wave, the end of the earthquake.

One first problem we faced is the size of the dataset (80 Go). To be able to work with it and train machine learning algorithms, we only took a random small representative subset of this dataset of around 1000 signals with the corresponding labels.

3.1 Preprocessing

We then organized and cleaned the dataset. In order to perform machine learning algorithms efficiently, we then randomly divided our dataset into a train dataset and a test dataset (90%/10%). We noticed that some signals were not properly labeled, we decided to remove them from analysis.

Then, we wanted to clean the signals individually. Each recorded waveform is 1 minute long and sampled at 100 Hz. All the signals in the dataset were already detrended and centered (subtraction of the mean) by the authors [4], so we didn't need to put more effort on this part (Figure S2A-C). The authors said that they additionally applied a band-pass filtering to remove potential noise, which could otherwise interfere with analysis. However, in the article they didn't mention precisely the characteristics of their filter. Indeed, several different filters can be used (for example, Bessel that preserves precisely the temporal shape, or Butterworth filters), with different parameters including the cutoff frequencies. Besides, it was not excluded that the signals needed another filtering step to increase the signal to noise ratio. To get an idea, we plotted some spectrograms of the data (data not shown, see attached notebook). We detected frequencies between 0 and 50 Hz (corresponding to the Nyquist frequency) and we noticed that a marge part of frequencies are represented during the P-waves and S-wave phases (Figure S2C). Not to modify the interesting parts of the signals, we therefore decided not to filter more the signals.

To perform our training and predictions on time windows, coming from the segmentation of the seismic time-series, we needed to define our window length. We should pick it such as it is high enough so the features computed from each will be meaningful (too short time window may lack important data), and low enough so to get a precise classification. To do so, we looked the distribution of the lengths of the parts of the signal to choose a correct window length (Figure S2D). We tried different time windows (see the result part). Also, the dataset contains multivariate signals (3 signals corresponding to South-north, East-West and vertical direction). We find that those three signals are not that different statistically just by looking at their means and standard deviations, and they share the same labels. So we choose only one signal (the vertical direction), so that our process is the most convenient.

As a last step of preprocessing, we labeled each of our time windows with one of these four labels: pre-seism (label 1, before the arrival of the P-waves), P-waves (label 2, between the arrival of the P-waves and the S-waves), S-waves (label 3, between the arrival of S-waves and the Coda end - end of the seism) and post seism (label 4, between the end of the seism and the end of the recording) (Figure S3). This enabled us to perform classification for each time window.

4 Results

After characterization and cleaning of the dataset, we divided our signal into windows. Each window was then labeled with one of these 4 labels: pre-seism (label 1), P waves (label 2), S waves (label 3) and post seism (label 4). As in the article [8], we used TSfresh to extract automatically more than 700 features for each time window. We then tried to apply either the ReliefF algorithm or the feature selection by the TSfresh library. The ReliefF algorithm took a lot of time to compute. As a consequence, we did all or tests using the feature selection by TFresh library which is much more faster. Using that technique, we were able to reduce the number of selected features. All the features were then normalized to ensure that no feature dominates because of scale and we performed a dictionary learning model inheritating from scikit-learn estimator that learned 4 different dictionaries. All this process was encapsulated in a custom-built well-structured pipeline in order to provide an automatized way to run the model and test different hyperparameters (using cross-validation).

We first tried to use a time window of 2.5 s (250 time samples). Our choice was motivated by our wish to take a window small enough to capture precise changes in the signal. However, we got a bad average accuracy (52%) likely due to an insufficient number of samples to effectively compute meaningful features. As a consequence, we decided to slightly increase the time window to 6 s (600 samples) to remain within an acceptable detection range. We got a higher accuracy (Table 1).

We used cross-validation with 3 folds to tune the model. We used 333 features for dictionary learning [*See supplementary text if you are interested about some qualitative comments about the selected features*]. We tuned 3 hyperparameters: number of atoms in the dictionary, number of non-zero coefficients, alpha (sparsity hyperparameter). Additionally, each dictionary has its own hyperparameter, which is tunable independently.

We did crossvalidation on many combinations of hyperparameters, Table 1 shows an example of crossvalidation on four different parameters (for details see Table 2). We found that combination of hyperparameter number 3 is the best for our problem, it has the highest scores for label 2 and label 3 (P wave and S wave), but predicts pretty badly the post sismic part of the signal.

Hyperparam	Mean Acc	Std Acc	Label 1	Label 2	Label 3	Label 4
1	0.6823	0.0048	0.7097	0.4593	0.4821	0.6678
2	0.7369	0.0032	0.7428	0.4435	0.2884	0.8148
3	0.5393	0.0401	0.7478	0.5556	0.5993	0.3806
4	0.7209	0.0029	0.7486	0.3657	0.4321	0.7538

Table 1: **Hyperparameters performance for crossvalidation with 3 folds.** We show the mean of the accuracy, its standard deviation, and the accuracy among each true label.

Figure S4 and Figure S5 show the difference between the predictions and the real labels on a signal. The model is confused between S wave and the post seismic noise part. That's because we choose n_nonzero_coefs $= [10, 20, 20, 10]$ where the last element in the list is the number of non zero coefficients for the label 4. So our model tried to be too much sparse and underfitted for label 4. However, we notice that it predicts pretty well the arrival of the P wave, indicating the start of the earthquake. This is particularly valuable, as detecting the P wave—the earliest signal of an earthquake—enables critical early warnings to mitigate the impact of the more destructive waves that follow.

When we try to put a greater hyperparameter on label 4, our model overfitted on this label as shown by Figure S6 and Figure S7 for a model trained with base hyperparameters: based on the histograms, we observe that the predictions for labels 1 and 4 are quite similar. However, the accuracy for label 3 has significantly deteriorated, and there is also a slight degradation for label 2, indicating overfitting.

The model has improved its accuracy for the last label. This is likely because it relies too heavily on its knowledge of class 4 (which is much more prevalent than the other classes) and tends to predict class 4 when it is uncertain whether an example belongs to class 3 or 4.

4.1 Opening

Our model is improvable (try to tune hyperparameters, improve the method of features selection, use overlapping window - we startes to implement that but unsuccessfully for now). Current state of the art on this problem is an attention model referenced here [3], which has an impressive ability of detecting P waves and S waves. On perspective of our work would be to implement and improve both methods

References

[1] Gemma Cremen and Carmine Galasso. "Earthquake early warning: Recent advances and perspectives". In: *Earth-Science Reviews* 205 (2020), p. 103184. ISSN: 0012-8252. DOI: https :

//doi.org/10.1016/j.earscirev.2020.103184. URL: https://www.sciencedirect.com/science/article/pii/S0012825220302300.

[2] Irshad Khan and Young-Woo Kwon. "P-Detector: Real-Time P-Wave Detection in a Seismic Waveform Recorded on a Low-Cost MEMS Accelerometer Using Deep Learning". In: *IEEE Geoscience and Remote Sensing Letters* 19 (2022). Conference Name: IEEE Geoscience and Remote Sensing Letters, pp. 1–5. ISSN: 1558-0571. DOI: 10.1109/LGRS.2022.3161017. URL: https://ieeexplore.ieee.org/document/9739015.

[3] S. Mostafa Mousavi et al. "Earthquake transformer—an attentive deep-learning model for simultaneous earthquake detection and phase picking". In: *Nature Communications* 11.1 (Aug. 7, 2020), p. 3952. ISSN: 2041-1723. DOI: 10.1038/s41467-020-17591-w. URL: https://www.nature.com/articles/s41467-020-17591-w.

[4] S. Mostafa Mousavi et al. "STanford EArthquake Dataset (STEAD): A Global Data Set of Seismic Signals for AI". In: *IEEE Access* 7 (2019), pp. 179464–179476.

[5] S. Yeats Robert, E. Sieh Kerry, and R. Allen Clarence. *The Geology of Earthquakes by, Kerry Sieh, and Clarence R. Allen*. Vol. 68. Oxford University Press, Sept. 1, 1997. URL: https://pubs.geoscienceworld.org/srl/article/68/5/778-779/142248.

[6] Karl Skretting and Kjersti Engan. "Recursive Least Squares Dictionary Learning Algorithm". In: *IEEE Transactions on Signal Processing* 58.4 (Apr. 2010). Conference Name: IEEE Transactions on Signal Processing, pp. 2121–2130. ISSN: 1941-0476. DOI: 10.1109/TSP.2010.2040671. URL: https://ieeexplore.ieee.org/document/5382523.

[7] Caixia Yu, Jingtao Zhao, and Yanfei Wang. "Seismic detection method for small-scale discontinuities based on dictionary learning and sparse representation". In: *Journal of Applied Geophysics* 137 (Feb. 1, 2017), pp. 55–62. ISSN: 0926-9851. DOI: 10.1016/j.jappgeo.2016.12.005. URL: https://www.sciencedirect.com/science/article/pii/S0926985116306012.

[8] Zheng Zhou et al. "Earthquake Detection in 1D Time-Series Data with Feature Selection and Dictionary Learning". In: *Seismological Research Letters* 90.2 (Mar. 2019), pp. 563–572. ISSN: 0895-0695, 1938-2057. DOI: 10.1785/0220180315. URL: https://pubs.geoscienceworld.org/ssa/srl/article/90/2A/563/568576/Earthquake-Detection-in-1D-TimeSeries-Data-with.

[9] Lingchen Zhu, Entao Liu, and James H. McClellan. "Seismic data denoising through multiscale and sparsity-promoting dictionary learning". In: *GEOPHYSICS* 80.6 (Nov. 1, 2015), WD45–WD57. ISSN: 0016-8033, 1942-2156. DOI: 10.1190/geo2015-0047.1. URL: https://library.seg.org/doi/10.1190/geo2015-0047.1.

Supplementary material

Supplementary text

In the article, they provided a feature characterization into 4 categories: Time domain, Frequency, Energy and Other features. We wondered to which of these categories our selected features belong to (after selection). We only looked at the 50 highest score features, as it is impossible to look by hand at 300 features. We found that 20 of them are time domain features (e.g. maximum, minimum, autocorrelation, etc.) 8 of them were freqeuncy-domain features (performed after FFT transformation for instance) and 3 of them are energy features. This contrasts with the results from the article where they showed that the more represented features are frequency features. In our case, this is only a preliminary result, but this difference could come from the fact that we are making the explicit distinction between wave P and wave S and that we are working on real recorded data unlike in the article.

Supplementary figures

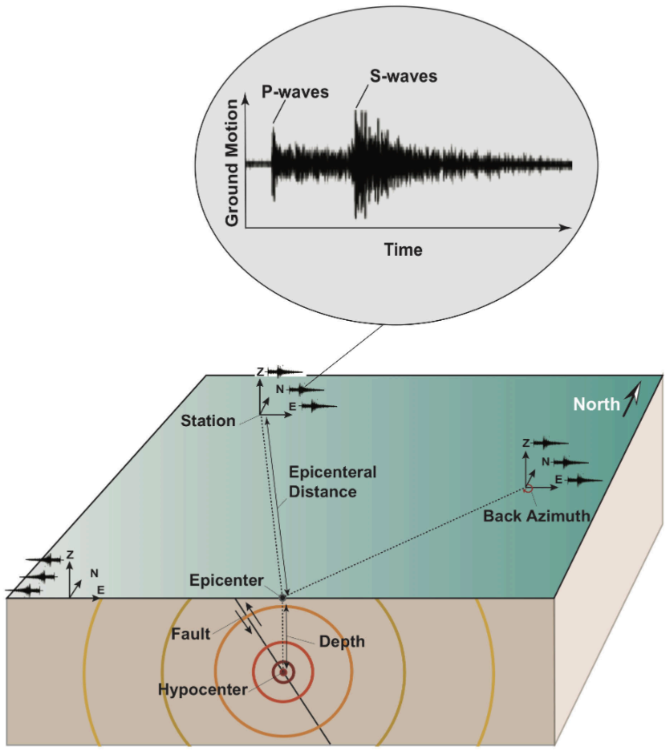

Figure S1: **Propagation and recording of seismic waves**. A zoomed-in window displays an annotated earthquake waveform for detailed analysis.

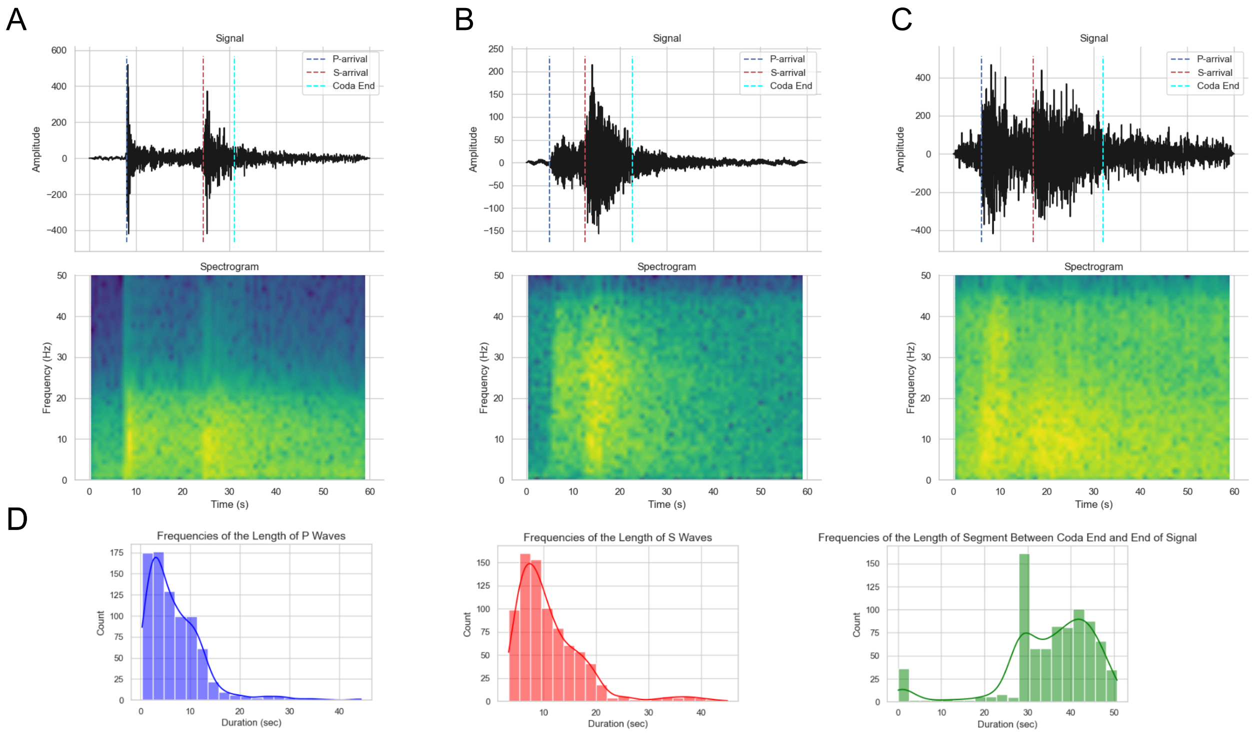

Figure S2: **Visualization of data**. **(A-C)** Visualization of some signals with their corresponding spectrograms. Note the broad distribution of frequencies and the different shapes of signals.**(D)** Distribution of the lengths of three types of waves in the our extracted earthquake dataset.

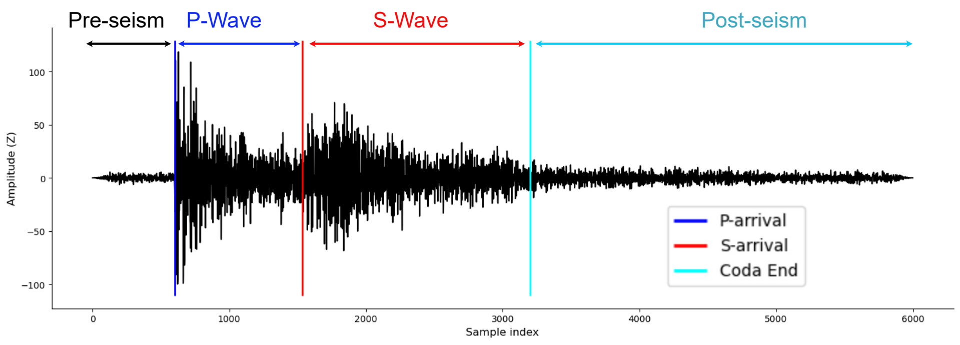

Figure S3: **Class labeling boundaries**. Example of sismogram with ground truth labeling of the 4 classes: pre-seism (label 1, before the arrival of the P-waves), P-waves (label 2, between the arrival of the P-waves and the S-waves), S-waves (label 3, between the arrival of S-waves and the Cuda end - end of the seism) and post seism (label 4, between the end of the seism and the end of the recording). Here, x-axis represent each data point index. The signal was divided into time windows (see main text for details).

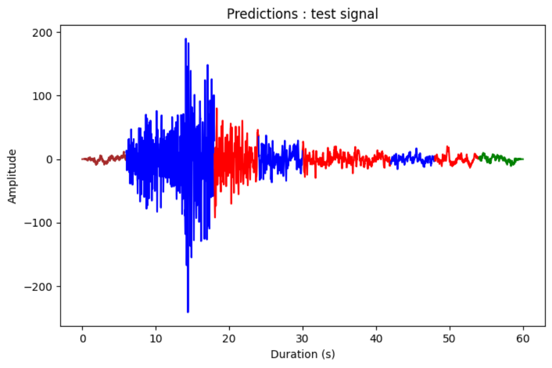

Figure S4: Prediction on a test signal. Color brown is pre sismic label. Color blue is P wave label. Color red is S wave label. Color green is post sismic label.

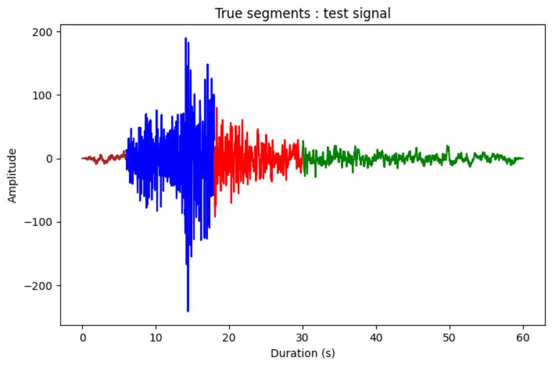

Figure S5: Real labels on a test signal. Color brown is pre sismic label. Color blue is P wave label. Color red is S wave label. Color green is post sismic label.

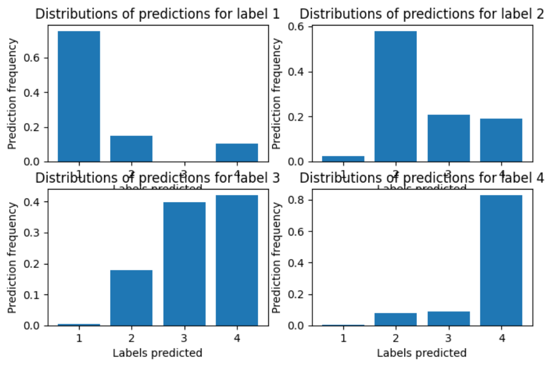

Figure S6: Caption

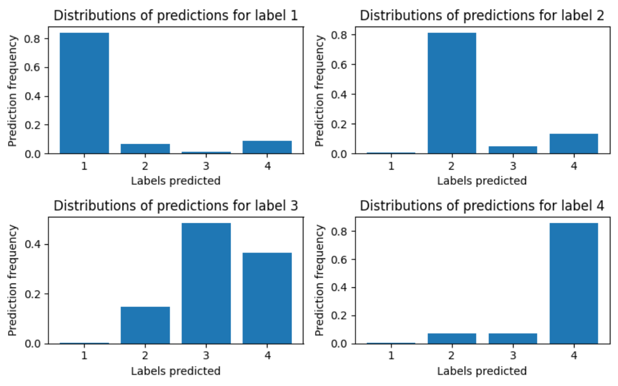

Figure S7: Caption

Parameters	label 1	label 2	label 3	label 4
dico__n_comps				
Param1	100	150	200	100
Param2	100	100	100	100
Param3	100	150	150	100
Param4	100	100	150	100
dico__n_nonzeros_coefs				
Param1	50	60	60	50
Param2	30	30	30	30
Param3	10	20	20	10
Param4	10	10	10	10
dico__alphas				
Param1	1	1	1	1
Param2	1	1	1	1
Param3	1	1	1	2
Param4	1	1	1	1

Table 2: Hyperparameters used for crossvalidation.