Earthquake Detection Model Using Python

Abstract:

We attempt to automatically detect earthquake events in distributed acoustic sensing (DAS) data via.

A supervised learning approach. Detecting earthquakes with different magnitudes could potentially.

Provide the ability of predicting major catastrophic events.

Introduction:

Distributed acoustic sensing (DAS) is an emerging technology used to record seismic data that employs.

Fiber optic cables as a probing system. By measuring the backscattered energy of a pulsing laser transmit.

Ted down a fiber optic cable, it is possible to measure the strain rate occurring within different sections.

Of the cable [1]. DAS recording systems have been shown to measure data comparable with conventional.

Geophones [2] and have been successfully used in exploration and earthquake seismology settings [3, 4].

Dataset And Preprocessing:

The fiber optic cable is deployed in Stanford's telecommunication tunnels in a double loop pattern.

Every 8 meters of cable acts as a receiver and records vibration at a sampling rate of 50Hz, creating a data.

Matrix of 300 channels distributed in space, each continuously recording cable strain since September.

2016. The array generates contiguous time series that conveniently lend themselves to image processing.

Labeling The Data :

We labeled the various categories by using complementary data from other sources. For the cars.

Proceeded with a methodology similar to [9], where a clustering algorithm (K-means) was applied.

Data in the continuous wavelet domain (CWT) and was able to separate different types of seismic signals.

By projecting the data over the array's geometry, a human supervisor can easily hand pick the clusters.

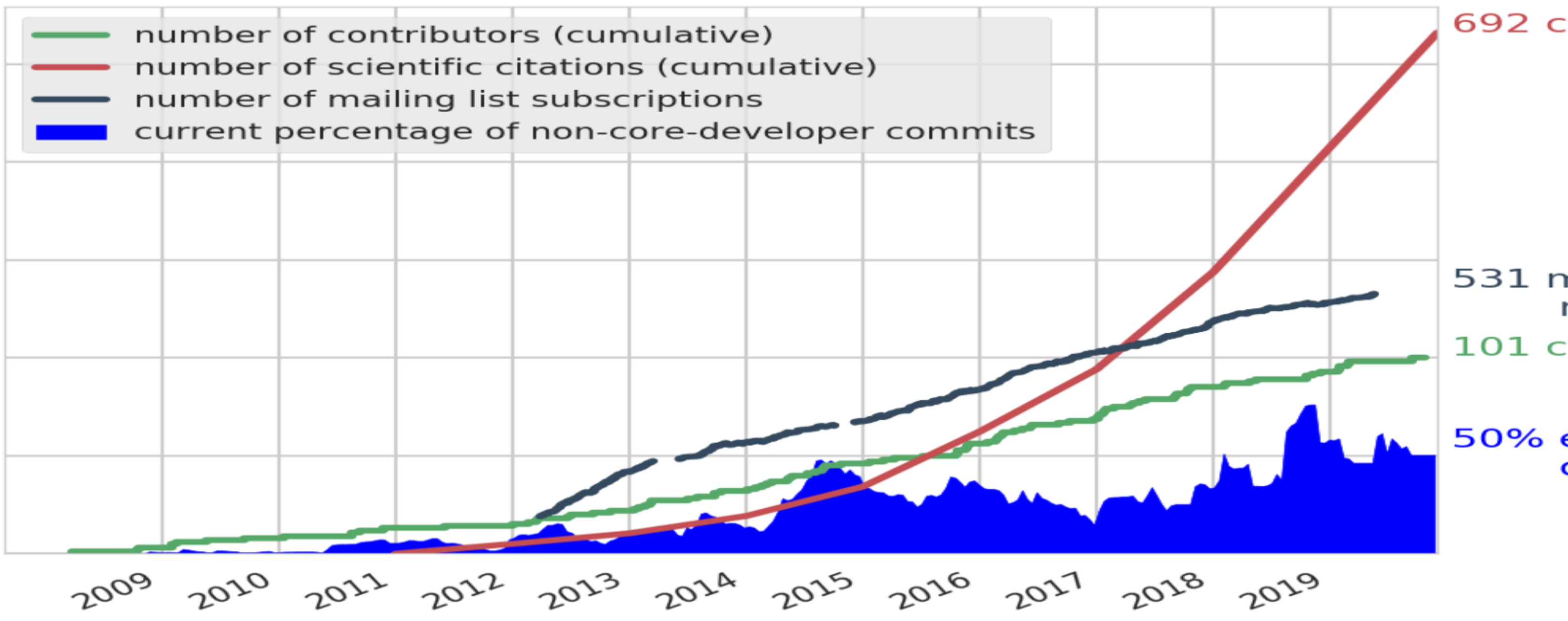
Corresponding to traffic noise.

Results and Discussion:

In order to understand how much each class is separable in the single amplitude feature space, in Figure.

4a we show the amplitude as a function of class number. We clearly notice that the three classes entirely

Overlap, thus they are not linearly separable. In addition, Figure 4b displays the normalized histogramOf the amplitudes for each class.



Feature Extraction:

In the context of ML-based earthquake detection, amplitude and frequency are the two key pieces of information among different statistics of the accelerometer signal.

IQR (Interquartile Range): IQR is the interquartile range $Q3$

$-Q1$

Of the 3 component vector sum VS

;

$VS=X2+Y2+Z2$−−−−−−−−−−−√

(1)where X, Y, and Z are the acceleration components.

CAV

(Cumulative Absolute Velocity): CAV

Feature is the cumulative measure of the VS

In the time window and is calculated as

$CAV = \int s0 | VS(t) | dt$

(2)where s is the total time period of the feature window in seconds, and t is the time. In this work, we used a two-second feature window.

ZC

(Zero-Crossing): ZC

Is the maximum zero-crossing rate of X, Y, and Z component and the zero-crossing rate of component X can be calculated as:

$ZCX = 1N - 1 \sum t = 1N - 11\mathbb{R} < 0(XtXt - 1)$

(3)where N is the total length of the signal X and $1\mathbb{R}<0$

Is indicator function.

IQR and CAV are the amplitude features, while ZC is the frequency feature, and these are proposed in [6,30].

These features detect earthquakes and can discriminate non-earthquake data, but through exhaustive experimental evaluations and also its implementation in the static environment as given in our previous work, we found that in a noisy environment (noisy sensors or external events), its performance can be degraded.

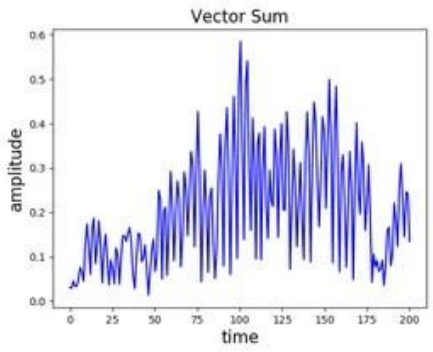
Where, A is an M x N matrix, where M represents two-second points, i.e., 200, and N is 3. SVD provides three new vectors $UMxM, SMxN, and\ VTNxN$

, which, if linearly combined, give back the approximated original vector; where U is the set of singular vectors with singular values in vector S, VT

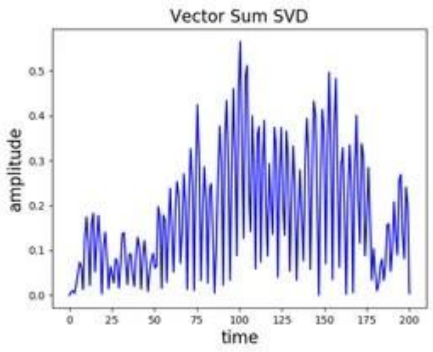
Is the primary direction. The new vectors are ordered, and the first vector explains most of the original acceleration amplitude and frequency information, as shown in Figure 3. Figure 3a depict almost the same structure; therefore, we select the first vector as a primary vector $U[:,0]$

From the given SVD's, along with the first value $S[0]$

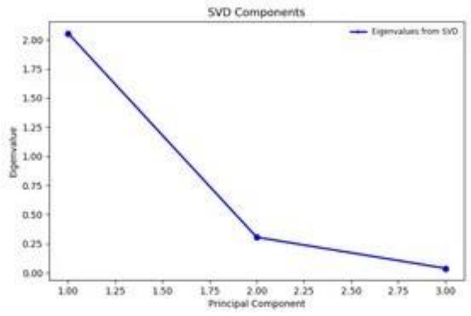
Of S, which is a scaling factor (give amplitude information of the given vector). We extracted the following three additional features.



(**a**) Vector sum



(**b**) Vector sum SVD1



(**c**) Scree plot

A two-second window of the strongest portion of the earthquake; (a) The vector sum of X, Y, and Z; (b) vector sum of the primary vector of SVD (center); (c) scree plot of the three components.

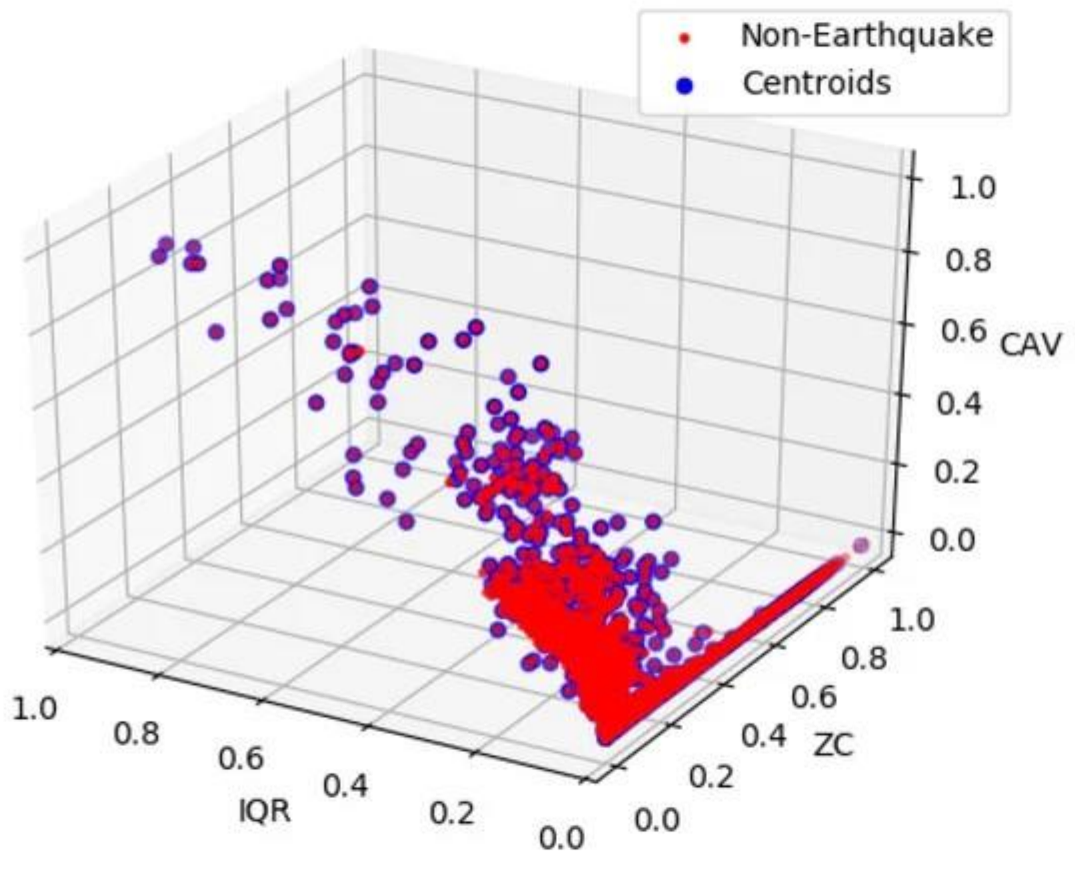
Pre-Processing:

In our methodology, the pre-processing involved balancing the dataset and scaling the features to range from 0 to 1. Balancing is required because the imbalanced datasets greatly affect the performance of the machine learning model [34]. In our case, the non-earthquake dataset (noise and human activities) is much larger than the earthquake dataset.

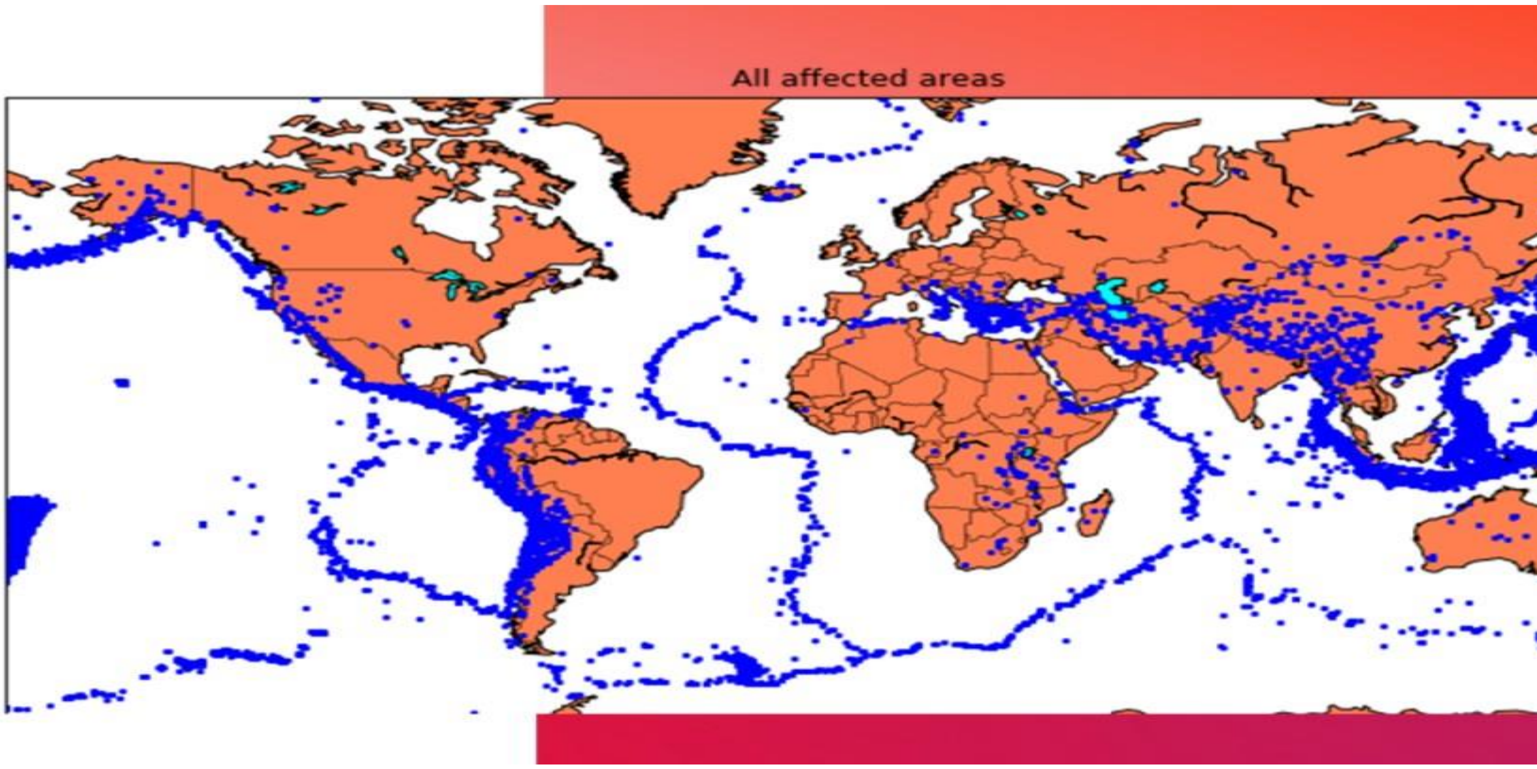
```
From mpl_toolkits.basemap import Basemap
```

```
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:


All affected areas