International Journal of Semantic Computing
Vol. 16, No. 1 (2022) 29–45

DOI: 10.1142/S1793351X22400025

World Scientific
www.worldscientific.com

Toward Autonomous Detection of Anomalous GNSS Data Via Applied Unsupervised Artificial Intelligence

Mike Dye

Ronin Institute, 127 Haddon Pl, Montclair, New Jersey 07043, USA
P.O. Box 56, Nederland, Colorado 80466, USA
mike.dye@ronininstitute.org

D. Sarah Stamps

Department of Geosciences, Virginia Tech, 926 W. Campus Drive
Blacksburg, Virginia 24061, USA
dstamps@vt.edu

Myles Mason

Academy of Integrated Science, Virginia Tech, 800 W. Campus Drive
Blacksburg, Virginia 24061, USA
mylesm18@vt.edu

Elifuraha Saria

Department of Geospatial Sciences and Technology
Ardhi University, P.O. Box 35176, Observation Hill
Plot No. 3, Block L, University Road
Dar es Salaam, Tanzania
saria.elifuraha@gmail.com

Artificial intelligence applications within the geosciences are becoming increasingly common, yet there are still many challenges involved in adapting established techniques to geoscience data sets. Applications in the realm of volcanic hazards assessment show great promise for addressing such challenges. Here, we describe a Jupyter Notebook we developed that ingests real-time Global Navigation Satellite System (GNSS) data streams from the EarthCube CHORDS (Cloud-Hosted Real-time Data Services for the geosciences) portal TZVOLCANO, applies unsupervised learning algorithms to perform automated data quality control ("noise reduction"), and explores autonomous detection of unusual volcanic activity using a neural network. The TZVOLCANO CHORDS portal streams real-time GNSS positioning data in 1 s intervals from the TZVOLCANO network, which monitors the active volcano Ol Doinyo Lengai in Tanzania, through UNAVCO's real-time GNSS data services. UNAVCO's real-time data services provide near-real-time positions processed by the Trimble Pivot system. The positioning data (latitude, longitude and height) are imported into the Jupyter Notebook presented in this paper in user-defined time spans. The positioning data are then collected in sets by the Jupyter Notebook and processed to extract a useful calculated variable in preparation for the machine learning algorithms, of which we choose the vector magnitude for further processing. Unsupervised K-means and Gaussian Mixture machine learning algorithms are then utilized to locate and remove data points ("filter") that are likely caused by noise and unrelated to

volcanic signals. We find that both the K-means and Gaussian Mixture machine learning algorithms perform well at identifying regions of high noise within tested GNSS data sets. The filtered data are then used to train an artificial intelligence neural network that predicts volcanic deformation. Our Jupyter Notebook has promise to be used for detecting potentially hazardous volcanic activity in the form of rapid vertical or horizontal displacement of the Earth's surface.

Keywords: TZVOLCANO; CHORDS; UNAVCO; artificial intelligence; machine learning; noise reduction; unsupervised learning; volcanic hazards assessment.

1. Introduction

The use of artificial intelligence (AI) applications in the geosciences has been increasing in the past few decades [1, 2]. A particularly useful application is in geohazards assessment. In this work, we make progress towards applying AI to the application of volcanic hazards assessment for the active volcano Ol Doinyo Lengai in Tanzania by developing a Jupyter Notebook [1] that can filter noisy Global Navigation Satellite System (GNSS) positioning data that are openly accessible.

Ol Doinyo Lengai is the only active carbonatite volcano in the world, and it erupts with a Volcanic Explosivity Index (VEI) greater than 1, on average, every 10 years [7]. The last significant eruption of a VEI 3 began in June 2007 with continued eruptions until October 2010. We, therefore, expect another significant eruption to occur in the near future. Hazards associated with Ol Doinyo Lengai include potential disruption to air traffic at the Kilimanjaro International Airport located approximately 140 km away from the volcanic crater, as well as hazards to the local populations, visitors and agriculture from lava flows, pyroclastic flows and volcanic ash. Developing tools to identify hazardous volcanic deformation that could indicate an impending volcanic eruption would help local decision makers determine if an evacuation is warranted.

To monitor Ol Doinyo Lengai, the TZVOLCANO network was installed beginning in 2016 (Fig. 1). The TZVOLCANO network has open data and currently consists of six GNSS stations [8–13] that are capable of streaming real-time positioning data and one broadband seismometer [14]. The raw GNSS data are available through UNAVCO (https://www.unavco.org/) and the seismic data are available through IRIS (https://www.iris.edu/).

This project takes advantage of the EarthCube CHORDS cyberinfrastructure [15, 16], which powers the TZVOLCANO CHORDS portal [17]. GNSS positioning data (longitude, latitude and height) are from the active Ol Doinyo Lengai volcano in Tanzania, which are made available through UNAVCO's real-time GNSS data services. UNAVCO's real-time GNSS data services provides positions (latitude, longitude and height) processed by the Trimble Pivot system. Real-time GNSS data from several instruments are streamed into the TZVOLCANO portal using brokering scripts developed in Python [18] and using shell scripting and awk [19], which makes the positions available in near-real-time via the CHORDS data API service.

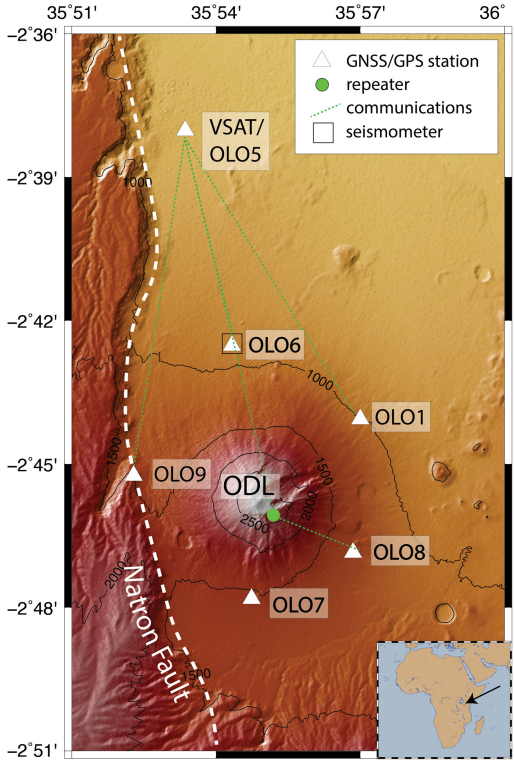

Fig. 1. The TZVOLCANO Network that monitors the active volcano Ol Doinyo Lengai in Tanzania. The Jupyter Notebook uses data from the GNSS station OLO1.

We utilize the Jupyter Notebook platform, which is an open-source web-based application that enables scientists and developers to integrate executable code with plots, diagrams and text in a flexible and reproducible integrated development environment. Jupyter Notebooks make sharing and reproducing scientific workflows easier. The Jupyter Notebook presented here demonstrates a process derived from general concepts described in "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems" by Géron [20] by which processed GNSS data (longitude, latitude and height) obtained from the TZVOLCANO CHORDS portal [17] can be analyzed to filter data that are not likely to be volcanic signals with minimal human input. These prepared and cleaned ("filtered") data are then used to train a neural network that can be used for detecting potentially hazardous volcanic activity in the form of rapid vertical or horizontal displacement of the Earth's surface. Rapid vertical or horizontal displacement on the flanks of a volcano could be an indicator of expanding gases in a shallow magma reservoir or magmatic movement in the subsurface that could lead to an eruption.

2. Data

The positioning data (latitude, longitude and height) ingested by the Jupyter Notebook originate from GNSS stations monitoring the Ol Doinyo Lengai volcano in Tanzania (Fig. 1). Real-time data from the TZVOLCANO network are transmitted via a satellite internet link to UNAVCO for initial processing with the Trimble Pivot system, and are then transmitted using the broker GNSS2CHORDS [18, 19] to the TZVOLCANO CHORDS portal which makes the data available via an API to the Jupyter Notebook (Fig. 2). The UNAVCO processing occurs with the Trimble Pivot system and, ideally, results in 2–5 cm precision positions [21], however, this is not always the case. For the TZVOLCANO stations, it is common for the noise to greatly exceed the desired 2–5 cm precision by an order of magnitude or more during certain periods which masks any useful volcanic signals. The Jupyter Notebook is a means to investigate AI methods that can, with very little human intervention, automatically identify and filter these periods of high noise.

3. Methods

3.1. *Data acquisition and usage*

Using an interface inspired by the EarthCube Brokered Alignment of Long-Tail Observations (BALTO) GUI [22], the raw data are downloaded directly from the TZVOLCANO CHORDS repository located at http://tzvolcano.chordsrt.com/ in a GeoJSON format. The GeoJSON file contains variables for height, longitude, latitude and height, as well as the time at which each measurement was recorded. While the Jupyter Notebook has the ability to access available data for any date range, by default, the Notebook uses data from the day January 1, 2021. The file for this

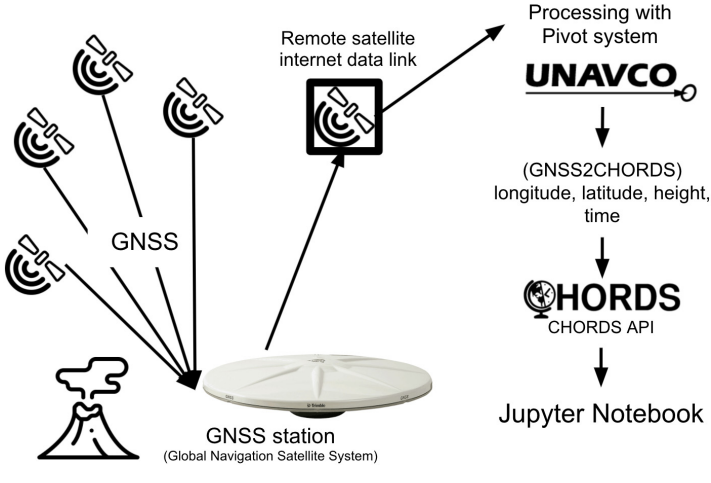

Fig. 2. Overview of the data acquisition pipeline.

particular day is provided with the Jupyter Notebook. Similarly, while the Notebook is capable of downloading data for any of several GNSS stations deployed on the volcano, the station "OLO1" is used by default as seen in Fig. 1. We note that the TZVOLCANO CHORDS portal only retains data for the three months prior to the current date.

While the full data set is used in training and filtering by the Gaussian Mixtures and K-means clustering algorithms, the neural network is trained on a smaller subset of these data. This reduction is necessary in order to decrease computation time because the Jupyter Notebook is designed to run on computers with limited resources. Training with more than 10,000 data points would exponentially increase processing time, taking potentially hours or even days to complete execution on small computers. A larger number of training points would very likely increase the performance of the neural network, but this approach was beyond the scope of this project.

3.2. *Initial data preparation*

3.2.1. *Data scaling*

Many clustering algorithms are highly sensitive to minimum and maximum ranges including both the K-means and Gaussian Mixtures algorithms. Thus, to make the data usable by the clustering algorithms, each feature (also known as a variable) was re-scaled using normalization so all values are between 0 and 1. Here, we are referring to the latitude, longitude and height as the features.

3.2.2. *Calculation of a derived feature: Vector magnitude*

Creating a derived feature is a common technique employed in machine learning. Using features can make detecting patterns in correlated features easier for the algorithms. In our case, the feature assists in identifying regions of localized high-noise areas within the time-series. In addition to the scaled latitude, scaled longitude and scaled height, we calculate the vector magnitude as a derived feature, as shown in the following equation:

$$\mathbf{V} = \sqrt{X^2 + Y^2 + Z^2}, \tag{1}$$

where $\mathbf{V}$ is the vector magnitude, X is scaled longitude, Y is scaled latitude and Z is scaled height.

Figure 3 shows the plots of all variables after scaling and the vector magnitude. The vector magnitude shows outlying data points are significantly further away from the baseline values and that there is less variation within the baseline than for the scaled height. The vector magnitude is, therefore, a good indicator that the clustering algorithms will more easily discern the periods of high noise.

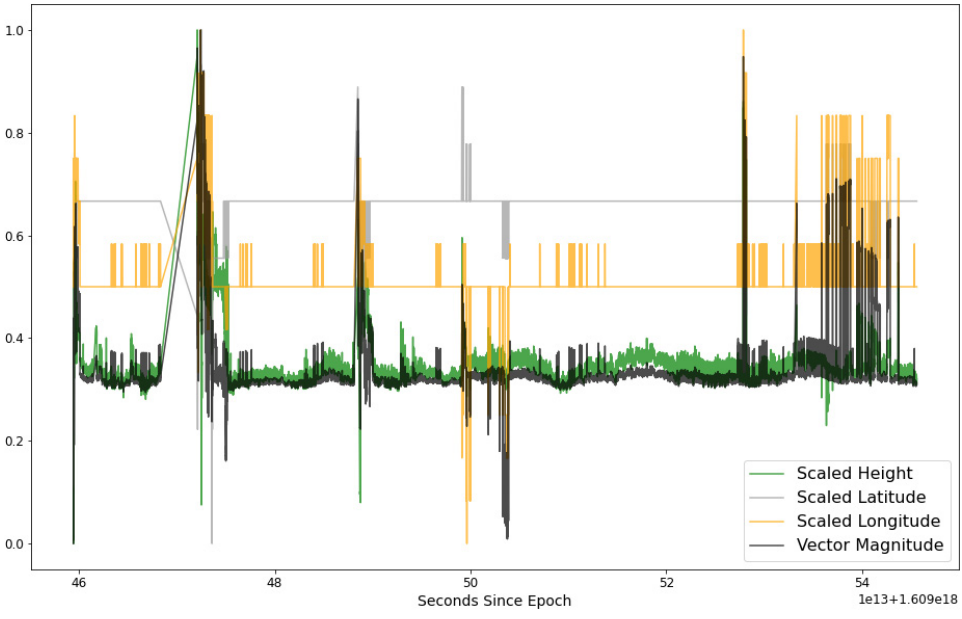

Fig. 3. (Color online) A time-series showing seconds since the start of the day as the X-axis with the vector magnitude (green), scaled longitude (yellow), latitude (red) and height (blue) before filtering. The vector magnitude combines height, longitude and latitude into a single feature that allows for easier identification of noise spikes in the data.

3.2.3. *Imputing*

Many machine learning algorithms including Gaussian Mixtures and K-means cannot function properly if there are gaps of missing data within the time-series being analyzed. To address this, a frequently used approach for anomaly detection and noise reduction is to impute missing data points in a time-series before processing. This means inserting values for each missing point in time with either a null value or with a calculated value such as the mean of all data points or the most frequently occurring value [23]. Imputing using the "most frequent" approach is performed on data used to train both the Gaussian Mixtures and K-means algorithms.

The Jupyter Notebook does not, however, use imputing during preparation of and data used in training the neural networks. Both imputing and nonimputing approaches were explored during development of the Notebook and it was found that training the neural networks on data that had not been imputed resulted in greater prediction accuracy.

3.3. *Gaussian mixtures filtering*

The Gaussian Mixtures algorithm is a probabilistic model that assumes a given set of data points were generated by merging one or more Gaussian distributions [20]. In this case, the Gaussian Mixtures model is built with the assumption that there is

only one Gaussian distribution. This assumption allows the trained Gaussian Mixtures model to classify each point by its distance from a mean value. In turn, this allows the user to specify a "Density Threshold", which is a percentage value that determines whether or not an individual point will be categorized as being "high noise" and flagged for removal.

A Gaussian Mixtures model is created and trained for each feature following the pattern:

- Designate which feature is being used
- Designate a density threshold percent that determines the percentage of data points that will be classified as having "high noise"
- Perform transformations that re-scale the input data
- Impute the data for the designated feature, filling in missing values with the most frequently occurring value for that feature
- Train a Gaussian Mixtures model on the imputed data
- Generate a plot showing which points are flagged for removal.

With the Gaussian Mixtures approach, human intervention is required in that for each feature, the user must input a density threshold percent based on their (1) knowledge of the data and (2) knowledge about what percentage of the overall data should be considered "high noise". Figure 4 shows the output of the trained

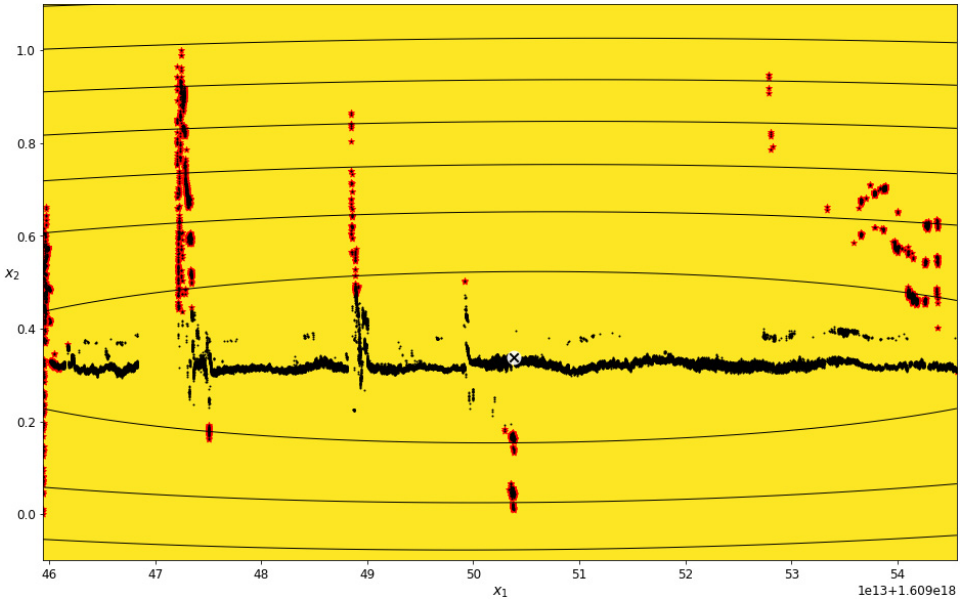

Fig. 4. (Color online) Points outlined in red have been identified by the Gaussian Mixtures algorithm as occurring outside the specified density threshold and are flagged as "high noise" areas for removal.

Gaussian Mixtures model for the vector magnitude feature. Points outlined with red stars indicate an area of high noise that will be filtered out.

3.4. *K-means filtering*

K-means is a widely used clustering algorithm known for both its simplicity and speed in identifying partitions and centroids within observations [20]. After training, K-means generates both a centroid of each cluster of data points, as well as boundary line separators between each cluster.

Similar to the Gaussian Mixtures model, human intervention is required, and the user must select a number of clusters to be expected for each feature. Additionally, after the K-means model has been trained and the K-means clusters have been determined, the user must identify which clusters should be considered "high noise", and thus discarded, and which clusters should be retained.

A K-means model is created and trained for each feature following the pattern:

- Designate which feature is being used
- Define the number of clusters expected for each feature
- Perform transformations that re-scale the input data
- Impute the data for the designated feature, filling in missing values with the most frequently occurring value for that feature
- Train a K-means model on the imputed data and calculate the clusters boundaries
- Generate a plot showing the newly computed clusters and cluster boundaries
- Define which clusters should be discarded and which should be retained.

Figure 5 shows the clusters identified for the vector magnitude feature given an input of 10 clusters to expect defined by the user. In the default case, clusters 0, 1, 3 and 7 have been manually identified by the user as points to be retained, and clusters 2, 4, 5, 6, 8 and 9 are considered areas of "high noise" that should be removed.

The areas chosen for removal were selected because they contained large spikes in the data that were inconsistent with reality. For example, on examination of the "Height" feature, it was apparent that the peaks in some of these data spikes were nearly a meter off from the average value. The physical interpretation of this would be that the Earth's surface was rapidly rising by up to 1 m and then falling again by up to 1 m within the time span of a few minutes. This unrealistic behavior of the data highlights the need for some depth of familiarity with the data in order for the K-means filtering method to be meaningfully applied.

3.5. *Apply filtering*

Once the models were created and trained for all four features (vector magnitude, scaled latitude, scaled longitude and scaled height) with both the Gaussian Mixtures

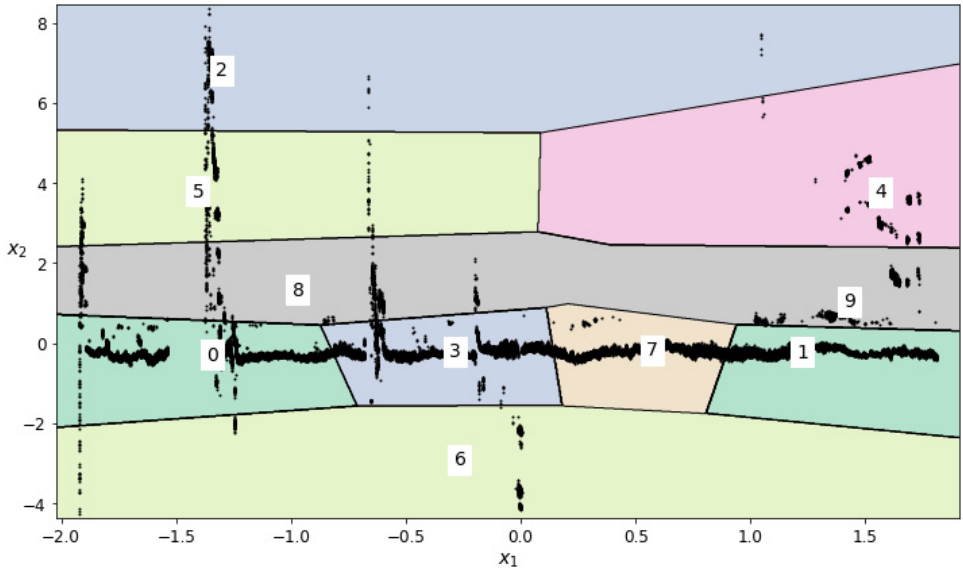

Fig. 5. (Color online) Clusters identified in using the vector magnitude feature by the K-means algorithm. Each cluster is identified with a number and a color.

and the K-means algorithms, data points flagged as having high noise were merged into two sets that included one for each of the clustering algorithms. Two filtered sets of data were then created by removing the high noise sets.

Figures 6 (Gaussian Mixtures) and 7 (K-means) show overlay plots of the unfiltered vector magnitude feature in blue and the filtered data identified by the Gaussian Mixtures and K-means algorithms in red. The filtered data have far fewer large fluctuations in the value ranges.

The plots of the filtered data produced by the algorithms appear quite similar. The K-means plots look to have somewhat larger spikes in several areas. This may indicate that the Gaussian Mixtures approach may do a moderately better job at noise reduction.

While both algorithms were able to identify and remove the larger data spikes, they were not able to locate areas of noise having lesser magnitudes. In Table 1 we compare the filtering results from the two methods. Re-scaling the filtered data output by each algorithm and performing another round of filtering could potentially further reduce the amount of noise.

Table 1. Comparison of unfiltered and filtered data points.

Filtering method	Points remaining after filtering	Points removed	% points removed
Gaussian Mixtures	46,358	4915	9.6%
K-means	46,556	4717	9.2%

Note: Number of data points before filtering: 51,273.

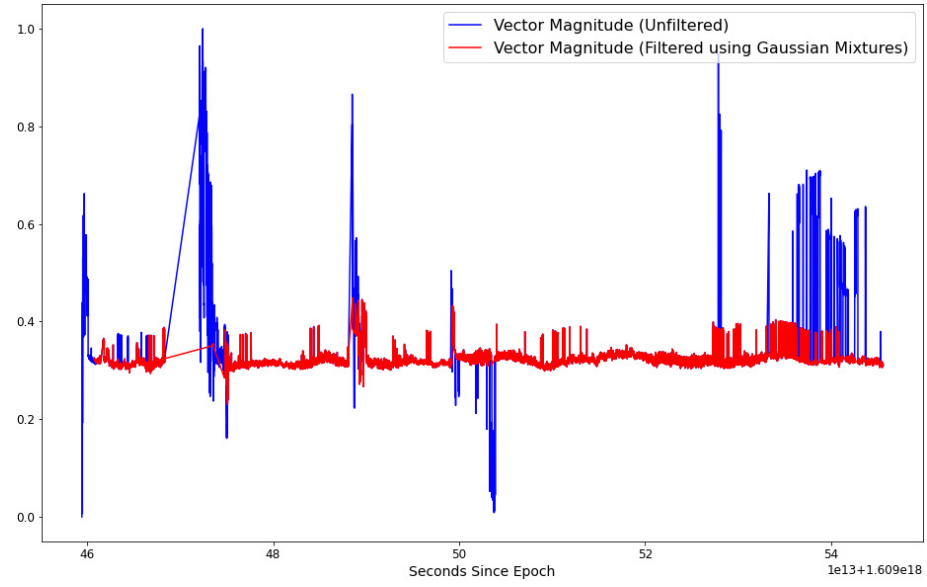

Fig. 6. (Color online) Comparison of the vector magnitude feature before and after the application of Gaussian Mixtures filtering. The largest signal fluctuations in the unfiltered data (blue) have been removed in the data that have been filtered with the Gaussian Mixtures model (red).

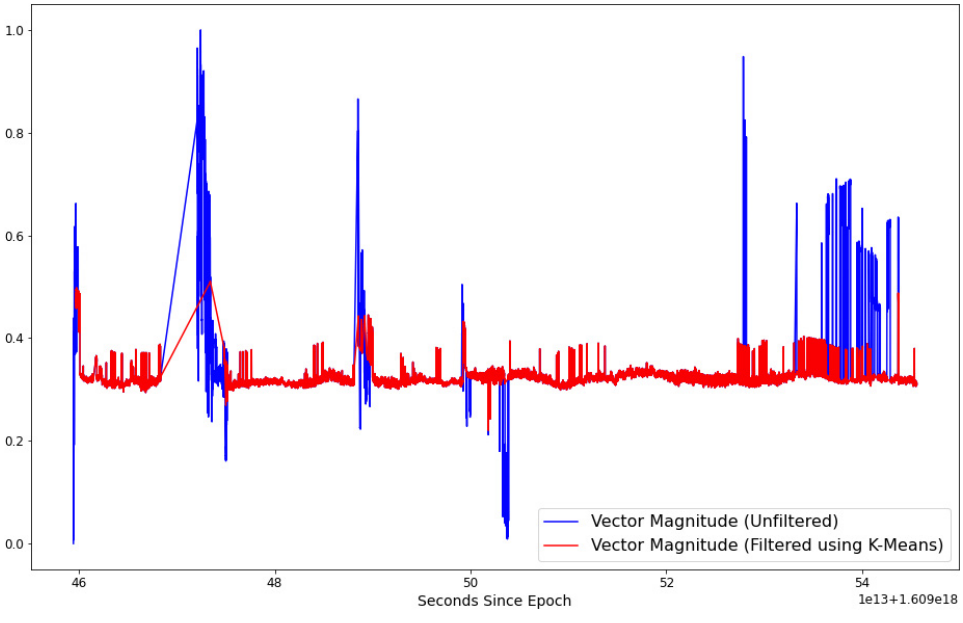

Fig. 7. Comparison of the vector magnitude feature before and after the application of K-means filtering. While the K-means filtered data are quite similar to that produced by Gaussian Mixtures filtering there are noticeable differences.

3.6. *Definition and training of the neural network*

After the filtered data sets have been created, neural network models are defined and trained for the following:

- Unfiltered data
- Data filtered using the Gaussian Mixtures algorithm
- Data filtered by the K-means algorithm

3.6.1. *Defining a recurrent neural network*

Many different types of neural networks were explored during development of the Jupyter Notebook. By iterative testing, it was determined that a recurrent neural network (RNN) depicted in Fig. 8 containing the following layers yielded the best performance:

- One-dimensional (1D) convolutional layer (input)
- A gated recurrent units (GRU) layer (20 neurons)
- A dropout layer
- A GRU layer (20 neurons)
- A dense layer (output)

Although we focused on the RNN depicted above, it is possible for users of this Jupyter Notebook to explore other types of neural network models.

3.6.1.1. Convolutional layer (Input)

Convolutional neural networks are a specialized form of neural networks that uses mathematical convolutions to calculate weights and output values. Convolutions, or the mixing of two functions or sources of information to create a third, have long been used in image processing. First proposed by LeCun *et al.* [3], multi-dimensional convolutional neural networks are used widely in processing spatial and imaging data such as handwriting recognition and computer vision. 1D CNN layers as they are used in this Notebook are very useful for feature generating feature maps in 1D time-series data. This assists the deeper layers in the network to identify longer-term patterns in the data.

3.6.1.2. GRU layer

GRU networks are closely related to long short-term memory networks. They are somewhat more simple, and have greater computational efficiency but in most applications perform just as well.

First proposed by Cho *et al.* [4] GRU addressed several issues. First, to address the vanishing and exploding gradient problem where large and rapid changes in input fail to be recognized by a neural network and result in undesirable weighting adjustments during training. Second, the GRU layer was designed to be more

computationally efficient and execute faster than previous solutions to the vanishing gradient problem.

Additionally, the GRU is better at remembering longer-term patterns. Many forms of neural networks are good at learning and recognizing short-term patterns, but fail to remember these patterns over greater spans of time. Using two GRU layers as we do in the Notebook helps to better detect longer-term patterns in our time-series data. This ability to better model the long-term behavior is further assisted by the initial 1D convolutional layer.

3.6.1.3. Dropout layer

A dropout layer randomly sets the input value for a neuron to zero during each epoch in the training of a neural net. When inserted between other layers in a neural network, a dropout layer introduces an element of randomization and is useful in preventing a neural network over-fitting training set data. First proposed by Hinton *et al.* in 2012 [5] and expanded upon by Srivastava *et al.* in 2014 [6], the dropout technique is widely used to improve a neural networks ability to generalize its learning.

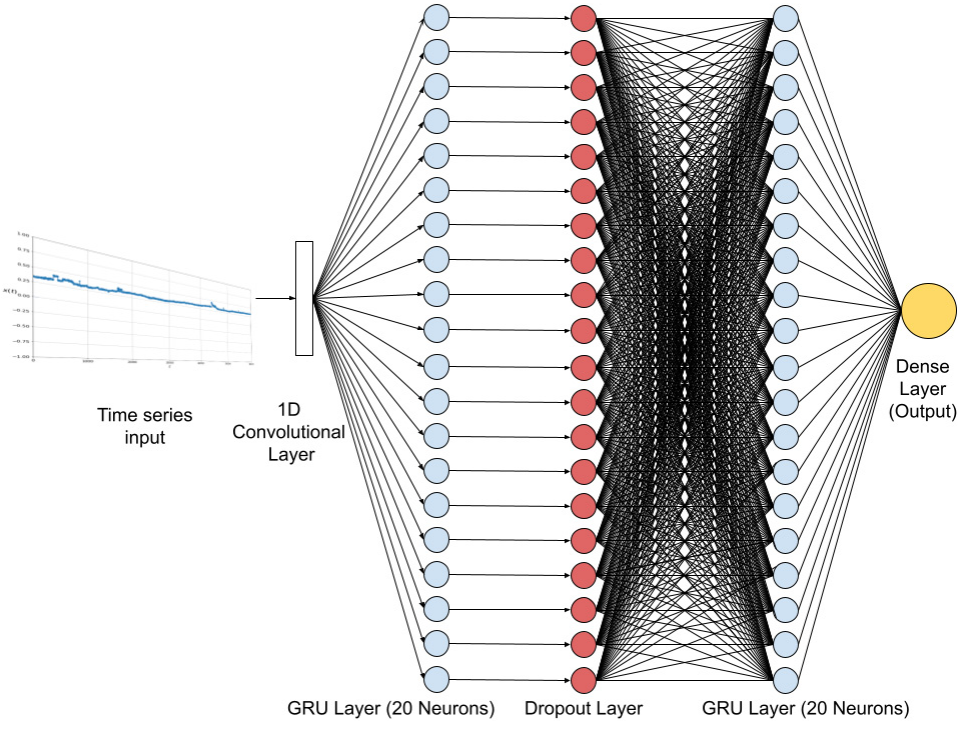

Fig. 8. Information flow from input to output in the RNN.

Our neural networks were tested both with and without the dropout layer. While exact metrics were not gathered, a substantial increase in accuracy in predictions made by the neural net was observed when the dropout layer was in place.

3.6.2. *Training the RNNs*

After the model is created, the filtered data are ingested, and the training process begins. The model is trained for 20 epochs such that the output of one pass through the network is used as input for the next iteration. This process is repeated for 20 iterations. The initial input data sets were truncated to contain a smaller number of points, which is set to 6000 by default. As previously mentioned, this reduction in points was necessary to speed execution time on a computer with limited processing power. After being trained, each RNN is then capable of making predictions of future data points. By default, 100 predictions are generated by the trained neural network for each of the models.

3.7. *Comparing predictions to expected values*

The predictions output by the trained RNNs are then compared to the actual measured values to assess the relative performance of the neural networks. Figures 9–11 show overlay plots comparing the predicted ("Forecast") points to their actual, expected values. Predictions made by the trained neural networks were compared to the actual values using mean squared error (MSE) in the

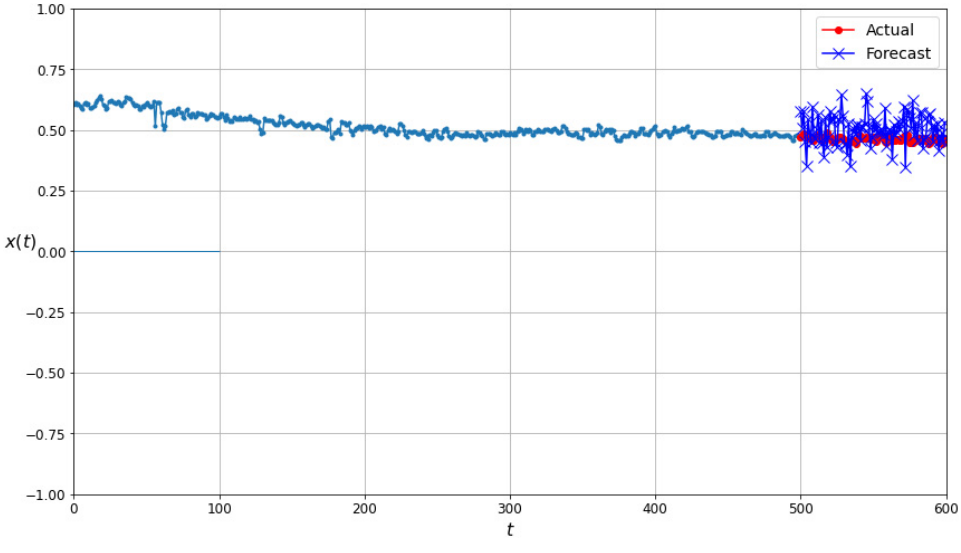

Fig. 9. Comparison of actual measurements to predictions made by the neural net trained on unfiltered data. While generally following the trend of the actual data points, the predictions' baseline is slightly off, and has a noticeably larger range of fluctuation between predicted points.

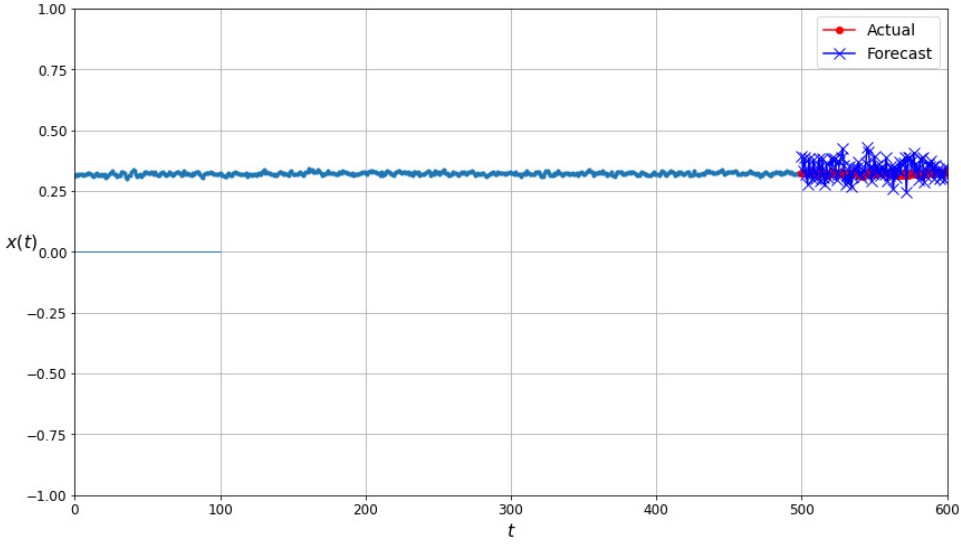

Fig. 10. Comparison of actual measurements to predictions made by the neural net trained on data filtered using the Gaussian Mixtures algorithm. Compared to Fig. 9, the forecast points lay noticeably closer to the actual points and display a tighter grouping with less fluctuation between the predicted points themselves.

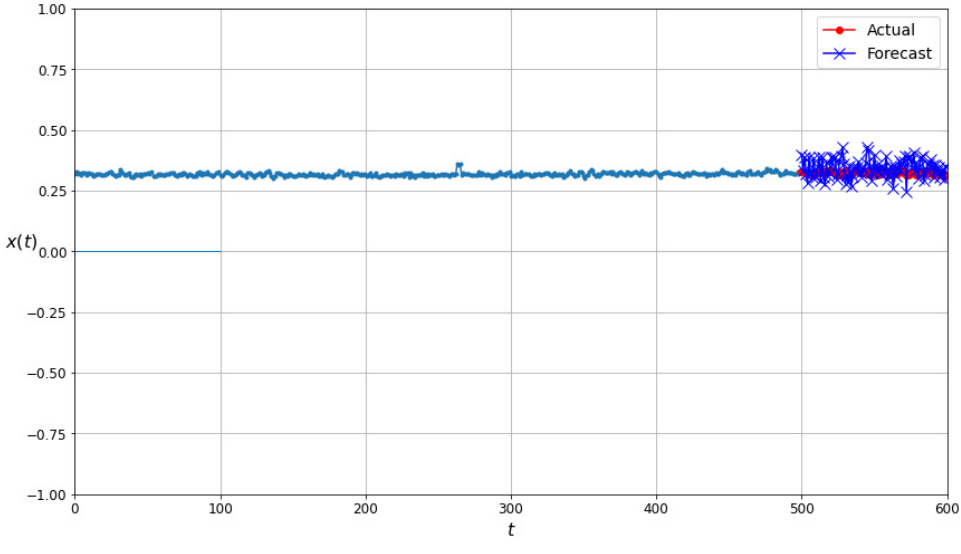

Fig. 11. Comparison of actual measurements to predictions made by the neural net trained on data filtered using the K-means algorithm. Compared to Fig. 9, the forecast points lay noticeably closer to the actual points and display a tighter grouping with less fluctuation between the predicted points themselves. With some variations, the predictions are very similar to those made by the network trained on Gaussian Mixtures filtered data in Fig. 10.

Table 2. MSE for trained neural networks.

Filtering method	MSE
Unfiltered data	0.0052
Gaussian Mixture filtered data	0.0017
K-means filtered data	0.0018

following equation with results shown in Table 2:

$$\text{MSE} = \left(\frac{1}{n}\right)\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2. \tag{2}$$

n is the number of data points.
Y_i are the observed values.
$\hat{Y}_i$ are the predicted values.

4. Results

Comparing the MSEs obtained from the neural network trained on unfiltered data to that of the network trained on filtered data yielded a 67% improvement in prediction accuracy for data filtered using the Gaussian Mixtures algorithm, and a 66% improvement for data filtered using K-means clustering. Therefore, the machine learning algorithms used to filter the data improved the predictive capability of the neural network. The Gaussian Mixtures' and K-means filtered data performed very similarly to one another with only a 1% variation in prediction accuracy.

5. Discussion

Here we describe a number of improvements to the approach taken in this Jupyter Notebook, as well as the necessary next steps that would be required to integrate these techniques in to an application capable of fully autonomous noise reduction and volcanic signal detection.

Of high priority is to identify and implement a method that would allow the noise reduction strategies to be truly autonomous, thus function without any human intervention. Along with this improvement is the need to update, in real-time, both the noise reduction algorithms as well as the trained neural network. To achieve these improvements, additional noise reduction machine learning strategies should also be explored. Specifically, the DBSCAN clustering algorithm may be well suited to this use case as it may be easier to implement to run completely autonomously.

It would also be beneficial to compare the results to more traditional noise reduction strategies (i.e. single spectrum analysis, fading-memory polynomial filters, exponential smoothing). Such comparisons would provide valuable insight into the effectiveness of the algorithms we employed here. The traditional strategies could also be used in tandem with the methods explored in this Jupyter Notebook.

6. Conclusions

This work demonstrates the use of a Jupyter Notebook to apply machine learning algorithms to filter positioning data and to train a neural network capable of predicting volcanic deformation signals. We have shown that of the two machine learning filtering approaches applied, Gaussian Mixtures and K-means, the Gaussian Mixtures approach performs better. We have used a suite of open-source resources, like the EarthCube cyberinfrastructure CHORDS, the TZVOLCANO GNSS data streams and Jupyter Notebooks towards autonomous detection of anomalous GNSS positioning data via applied unsupervised AI.

Acknowledgments

This work was made possible because of the EarthCube cyberinfrastructure CHORDS, which was funded by National Science Foundation grants 1639750, 1639720, 1639640, 1639570 and 1639554. We also leveraged the EarthCube cyberinfrastructure BALTO to develop the data acquisition user-interface in our Jupyter Notebook, which was funded under the National Science Foundation grants 1740704, 1740627 and 1740696. This project has also benefited from the National Science Foundation funded CAREER award Stamps, which is grant 1943681. A grant from the National Geographic Society to Stamps (NGS CP-730R-17) supported the installation of GNSS stations OLO6, OLO7 and OLO8. This material is also based on services provided by the GAGE Facility, operated by UNAVCO, Inc., with support from the National Science Foundation and the National Aeronautics and Space Administration under NSF Cooperative Agreement EAR-1724794. We thank Naibala Komiando Leiyan Laizer, Ntambila Daud and Kristof Nkembo for their continued technical support at the TZVOLCANO network.

References

[1] M. Dye, D. S. Stamps and M. Mason, Toward autonomous detection of anomalous GNSS data via applied unsupervised artificial intelligence GitHub repository, 2021, https://github.com/earthcube2021/ec21_dye_etal.

[2] J. Bortnik and E. Camporeale, Ten ways to apply machine learning in Earth and space sciences, *Eos* **102** (2021).

[3] Y. LeCun and P. Haffner, L. Bottou and Y. Bengio, Object recognition with gradient-based learning, in *Shape, Contour and Grouping in Computer Vision* (Springer, Berlin, 1999), pp. 319–345.

[4] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, Learning phrase representations using RNN encoder–decoder for statistical machine translation, in *Proc. 2014 Conf. Empirical Methods in Natural Language Processing*, 2014, pp. 1724–1734.

[5] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, preprint (2012), arXiv:abs/1207.0580.

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, *J. Mach. Learn. Res.* **15**(1) (2014) 1929–1958.
[7] Global Volcanism Program, Ol Doinyo Lengai (222120) in volcanoes of the world, v. 4.10.1 (29 June 2021), ed. E. Venzke, Smithsonian Institution, 2013. Available at https://volcano.si.edu/volcano.cfm?vn=222120 (accessed on 30 July 2021).
[8] D. S. Stamps, E. Saria, K. H. Ji, J. R. Jones, D. Ntambila, M. Daniels and D. Mencin, TZVOLCANO: OLO1-OLO1_OLO_TZA2016 P.S., UNAVCO, GPS Dataset, 2016.
[9] D. S. Stamps, E. Saria, K. H. Ji, J. R. Jones, D. Ntambila, M. Daniels and D. Mencin, TZVOLCANO: OLO5-OLO5_OLO_TZA2016 P.S., UNAVCO, GPS Dataset, 2016.
[10] D. S. Stamps, E. Saria, K. H. Ji, J. R. Jones, D. Ntambila, M. Daniels and D. Mencin, TZVOLCANO: OLO6-OLO6_OLO_TZA2017 P.S., UNAVCO, GPS Dataset, 2017.
[11] D. S. Stamps, E. Saria, K. H. Ji, J. R. Jones, D. Ntambila, M. Daniels and D. Mencin, TZVOLCANO: OLO7-OLO7_OLO_TZA2017 P.S., UNAVCO, GPS Dataset, 2017.
[12] D. S. Stamps, E. Saria, K. H. Ji, J. R. Jones, D. Ntambila, M. Daniels and D. Mencin, TZVOLCANO: OLO8-OLO8_OLO_TZA2017 P.S., UNAVCO, GPS Dataset, 2017.
[13] D. S. Stamps, E. Saria, K. H. Ji, J. R. Jones, D. Ntambila, M. Daniels and D. Mencin, Tanzania Volcano Observatory — OLO9-OLO9_OLO_TZA2021 P.S., UNAVCO, GPS/GNSS Observations Dataset, 2021.
[14] A. Adams, Ol Doinyo Lengai, TZ volcano monitoring [data set], International Federation of Digital Seismograph Networks, 2017.
[15] M. D. Daniels, B. Kerkez, V. Chandrasekar, S. Graves, D. S. Stamps, C. Martin, M. Dye, R. Gooch, M. Bartos, J. Jones and K. Keiser, Cloud-Hosted Real-time Data Services for the Geosciences (CHORDS) software (Version 0.9), UCAR/NCAR — Earth Observing Laboratory, 2016.
[16] B. Kerkez, M. Daniels, S. Graves, V. Chandrasekar, K. Keiser, C. Martin, M. Dye, M. Maskey and F. Vernon, Cloud hosted real-time data services for the geosciences (CHORDS), *Geosci. Data J.* **3**(1) (2016) 4–8.
[17] D. S. Stamps, E. Saria, K. H. Ji, J. R. Jones, D. Ntambila, M. D. Daniels and D. Mencin, Real-time data from the Tanzania Volcano Observatory at the Ol Doinyo Lengai volcano in Tanzania (TZVOLCANO), UCAR/NCAR — Earth Observing Laboratory, 2016.
[18] J. R. Jones and D. S. Stamps, M. D. Daniels and N. Hook, Optimization of TZVOLCANO to stream real-time GNSS data into Cloud Hosted Real-Time Data for the Geosciences (CHORDS), AGU Fall Meeting Abstracts, IN43D-0925, 2018.
[19] D. S. Stamps, E. Saria, J. R. Jones, M. D. Daniels and D. Mencin, Tectono-magmatic investigations with societal implications: Progress on the Tanzania Volcano Observatory (TZVOLCANO), AGU Fall Meeting Abstracts, A31K-08, 2016.
[20] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd edn. (O'Reilly, Boston, 2019).
[21] R. Leandro, H. Landau, M. Nitschke, M. Glocker, S. Seeger, X. Chen, A. Deking, M. BenTahar, F. Zhang and K. Ferguson, and others, RTX positioning: The next generation of cm-accurate real-time GNSS positioning, in *Proc. 24th Int. Technical Meeting of the Satellite Division of the Institute of Navigation*, 2011, pp. 1460–1475.
[22] S. D. Peckham, M. Stoica, D. S. Stamps, J. Gallagher, N. Potter and D. Fulker, The BALTO Jupyter Notebook GUI, 2020, https://github.com/earthcube2020/ec20_peckham_etal, https://sites.google.com/vt.edu/balto/home.
[23] J. Scheffer, Dealing with missing data, *Res. Lett. Inf. Math. Sci.* **3** (2002) 153–160.