Universiteit Leiden

Master Computer Science

Application of Automated Machine Learning Pipeline for the Classification of Volcanic Time Series Data

Name:	Yingjie Li
Student ID:	s3126161
Date:	28/02/2023
Specialisation:	Computer Science: Data Science
1st supervisor:	Dr. Furong Ye
2nd supervisor:	Dr. Anna V. Kononova
3nd supervisor:	J.P. de Nobel

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Volcán de Colima is an active volcano that is located on the Ring of Fire and it is reported that hundreds of eruptions occur here each year. It brings a lot of hazardous The eruptions introduce numerous dangerous substances into the local ecosystem, which is detrimental to the native vegetation and local economy. However, the transitional volcanic eruption forecasting ways require related domain knowledge and extensive human work, and are somewhat sluggish. Compared to this circumstance, Automated machine learning can address the issue more quickly with less human labor and subject expertise.

In this thesis, we aim to tackle the challenge of volcanic eruption forecasting using time series data as well as the Automated machine learning pipelines from previous work. First of all, we post preprocessing methods on data collected from Volcán de Colima: we filled in the missing data, and applied a filter to extract fixed frequency from the raw data. Additionally, We also use the downsampling technique to downsample the rate, which helps to save time in future steps. Then, the sliding window algorithm was applied to split the continuous signal into separated time windows. Subsequently, we extract features from the sliding windows by generating massive statistic features. And we apply three feature selection methods to remove the features with the low contribution to the prediction: FRESH algorithm, Variance Threshold, and Boruta. Furthermore, to balance the dataset, we compare three Imbalanced learning techniques: Randomly dropping, Synthetic Minority Oversampling Technique(SMOTE), and Four Types of Samples. And we choose the latter two and add them to our pipeline before Modeling and Hyperparameter Optimization step. Finally, we compared the performance of two Modeling and Hyperparameter Optimization methods: Tree-Based Pipeline Optimization Tool(TPOT) and Automated tuned Random forest, and choose the latter one. To conclude, in this work, we mainly applied previous work and techniques to tackle the challenge of volcanic eruption forecasting based on the data collected from Volcán de Colima.

KEYWORDS: Automated Machine Learning, Imbalanced Learning, Time Series Classification, Volcanic Eruptions, Forecasting

Contents

1 Introduction **6**

2 Related work **11**

2.1 Volcanic eruption forecasting with machine learning 11

2.2 Automated machine learning pipeline 11

2.3 Imbalance learning 12

3 Dataset and Problem Statement **14**

3.1 Data collection 14

3.2 Problem Statement 16

4 Methods **18**

4.1 Automated machine learning pipeline 18

4.2 Preprocessing 19

4.3 Feature Extraction 24

4.4 Feature Selection 25

4.5 Imbalanced Learning 28

4.6 Modeling and Hyperparameter Optimization 31

5 Experiment setup and Result **36**

5.1 Design of Experiment 36

5.2 Objective Functions 37

5.3 Experiment Setup 37

5.4 Result 40

6 Conclusion **51**

1 Introduction

Volcanic eruptions forecasting is challenging because volcanoes are complex and stochastic systems. And the eruptions are controlled by many different non-linear processes. Therefore, it is hard to give a detailed description of them at a certain time. The conventional method of explosion prediction heavily relies on handheld devices or aircraft to collect data from the ground or the air, and they are thought to be dangerous, labor-intensive, and uneconomical. Nevertheless, manual input and analysis of these methods are required to predict explosions, which is time-consuming, while a quick result is needed sometimes for an upcoming eruption.

Volcanologists have traditionally focused focus on a small range of frequencies called tremor(0.5-15 Hz) which proved to be the most crucial range for eruption precursor recognition [1]. A tremor is continuous seismic energy and contains precursors of eruptions and other volcanic activities.

Volcanologists employ a variety of graphical tools to visualize and analyze the seismic signals collected from sensors when focusing on the tremor. Real-time Seismic-Amplitude Measurement (RSAM) and Seismic Spectral Amplitude(SSAM) are important and commonly used tools to extract useful information from signals, and they can obviously show the peak of signals which indicate the highest energy from the volcano. The peak values show the volcanoes may be active. Additionally, SSAM can help volcanists to recognize the source of the signal. And other tools can also tell important information as well. The graph Mean of Dominant Frequency shows the domain and highest frequencies and the outliers in a given period. And Frequency Index shows the ratio of the average value of one part in valid frequencies to another part and helps the volcanists to discover the outliers. Volcanologists rely on these tools to analyze the tremor and predict volcanic eruptions. But each of the graphic tools requires the corresponding domain knowledge of volcanology and is also labor-intensive. So, it is not very convenient for people who do not major in this field.

Therefore easy-to-use and quick methods should be introduced to solve the challenge. And researchers try to dig out information from big data and find patterns that can predict volcanic eruptions without analyzing the figures every time. A lot of machine learning methods are developed to help [2] [3] [4] [5]. They applied different Machine learning models and algorithms, such as Convolutional Neural Networks (CNN), K-means clustering, and Random Forest, to solve the problem and make some success. Machine learning reduces receptive human labor, for once a model is trained, future prediction can be done by the model without human intervention, and it requires less subject expertise in volcanology. Even though researchers use different Machine learning techniques to solve the problem of volcanic eruption forecasting, which are proven to be feasible in this field, there are still some human interventions needed, for example, model selection, and parameter tuning. They are strange to researchers from other fields, and they require both experience and domain expertise as well. Therefore, researchers need to learn a lot of relevant Machine learning knowledge to tackle the problem. However, Automated machine learning (AML) can help to solve this problem because it requires less domain knowledge and fewer human interventions. AML can automatically construct the pipeline with the best-performed model, algorithms, and hyperparameters.

Researchers can use Automated machine learning techniques to explore many practical methods requiring less domain knowledge. And it only needs researchers to preprocess the data, which only needs limited domain knowledge in both the domain field, for example, volcanology, and the machine learning field. Researchers just put the preprocessed data in the constructed machine learning pipeline and the result. And there are already many successful automated machine learning applications on time-series data in many fields, and machine learning also performs relatively well on the volcano dataset [6] [7] [8] [9]. For example, M Kefalas [7] put up with an Automated machine learning pipeline on the aircraft field to predict the Remaining Useful Life (RUL) of engines. And D. E. Dempsey [6] proposed a machine learning pipeline to forecast the probability of explosion for the volcano Whakaari in New Zealand. But their dataset only contains no more than 10 explosions.

Based on the previous work [10], we applied an Automated machine learning (AML) pipeline on our dataset which has a large number of eruptions collected from Volcán de Colima. This pipeline originally was used to solve the problem of vehicle on-board. Then it also proves to be effective on EEG (electroencephalogram) data to recognize the cognitive function of Parkinson's disease patients. We adjusted it and combined it with other Imbalanced learning techniques, such as the Synthetic Minority Oversampling Technique(SMOTE) and Four Types of Samples [11]:

- We posted preprocessing methods on our dataset: We used an interpolation technique to fill in the missing data throughout the day by computing the average value of the remaining data on that day. We aggregated data of a single sensor from different geographic directions, while sensors collect data from three different directions. Then, we adopted a Bandpass filter to remove the high-frequency signals and low-frequency signals. Next, we used the downsampling technique to reduce the sampling rate, which decreases the consumption time for future steps, such as feature selection. we also applied the Sliding window algorithm to split data into small time windows, and each window contains a piece of continuous data. Additionally, we test out the suitable parameters for the time windows and sampling rate.

- We applied the feature extraction methods which generate abundant statistical features based on data in time windows from pre-defined mathematical formulas. Then, in order to remove the redundant generated features, we applied three feature selection methods: FRESH algorithm, Variance Threshold, and Boruta. And find that when the sampling rate is relatively small, all of them are needed to remove redundant features. When the sampling rate is large, the FRESH algorithm and Variance Threshold are able to remove redundant features. But in this circumstance, the influence of Boruta is limited to the feature selection result.

- Based on the fact that our data are imbalanced, we tested three imbalanced learning techniques on our dataset: Randomly dropping, Synthetic Minority Oversampling Technique(SMOTE), and Four Types of Samples. we first adjusted Four Types of Samples and made it available on multi-class problems, and we prove that it can improve the performance of our models based on experiments. Additionally, we find the combination of Four Types of Samples and SMOTE performed much better in processing the problem of the imbalanced dataset than the combination of Four Types of Samples and Randomly dropping. However, Randomly dropping and SMOTE can not be used together because one will generate synthetic data and another one will delete data from the dataset. Also, we find the performance of variants of SMOTE is relatively close based on experiments.

- We finally compared the performance of two Modeling and Hyperparameter Optimization methods: Tree-Based Pipeline Optimization Tool(TPOT) and Automated tuned Random forest. And find their performance is very close, and Automated tuned Random forest costs less time than TPOT. Furthermore, Automated tuned Random forest is better for future development because of its fixed model.

The remaining parts of the paper are structured as follows: Section 2 lists some relevant work in volcanic eruption prediction, AML, and techniques for imbalanced learning, Section 3 provides a detailed task definition and describes the dataset's source and collection method. In Section 4, we provide the methods used in the thesis. In addition, the construction of the AML pipeline is described in depth. In Section 5, we list the experimental design, outcomes, and interpretations. Finally, Section 6 ends with a brief conclusion.

2 Related work

2.1 Volcanic eruption forecasting with machine learning

There are already a lot of works about volcanic eruption forecasting in the field of machine learning [2] - [6]. CX Ren [2] applies the Spectral clustering technique which is an unsupervised learning method to the data of Piton de la Fournaise. And supervised learning methods like Random Forest [6], SVM (support vector machine) [4] [5], are also applied in the field of volcanoes to detect eruption. Furthermore, some researchers use Convolutional Neural Networks, CNN for the prediction [3]. In conclusion, researchers try to solve the challenge of volcanic eruption forecasting with different kinds of machine learning techniques. And there is also a lot of work based on automated machine learning (AML) as well.

2.2 Automated machine learning pipeline

Aiming to solve the machine learning problem without human effort, AML is an efficient technique to generate optimal machine learning framework from a given search space, optimizer, and objective function [12]. AML pipeline mainly aims to solve the problem of CASH(combined algorithm selection and hyperparameter optimization) where the model tries to find out suitable algorithms, preprocessing methods, and hyperparameters. This can help future users who may have little domain expertise in machine learning because the hyperparameter and algorithm setting requires both domain knowledge and experience in machine learning. And in recent years, automated machine learning has been well developed in many different areas, for example, defect detection [8] in industrial applications and disease recognition in the healthcare field [9]. And typically, an AML pipeline consists of Preprocessing, Feature extraction, Feature selection, Modeling, Hyperparameter optimization, and Evaluation.

2.3 Imbalance learning

Oversampling tries to solve the problem of imbalanced data distribution. And there are a lot of methods for addressing this problem, for example, creating minority class data and dropping some majority class data to make the dataset balanced. The minority class is the label with the least amount of data, while the majority class is the opposite. Randomly dropping is the commonest way for the imbalanced datasets [13], and it always works well in many circumstances, while it is simple and intuitive. Then, The synthetic minority oversampling technique (SMOTE) [14] and its variants solve the problem of oversampling by creating synthetic minority class data. And there are many variant SOMTE algorithms for solving the problem. SMOTE is the most famous method in the resampling field. Based on the K-nearest neighbors and randomly chosen minority data, SMOTE can generate enough minority data to make the dataset balanced. And more variants are developed based on SMOTE, for example, the adaptive synthetic (ADASYN) [15]. Compared to SMOTE, ADASYN mainly gives higher weight to the data which is difficult to learn, and reuse them more often than the normal data. And there are other variants as well, for example, BorderlineSMOTE. BorderlineSMOTE is also based on SMOTE, but it first divides the minority data into 3 types based on labels of their neighbors: Safe, Danger, and Noise. And it just focuses more on the data near the boundary. So, most of the new data is generated based on Danger data which is near the boundary.

In Kong's work [11], the authors put up with methods to create additional attributes for binary classification based on Anomaly detection, named Four Types of Samples.

Anomaly detection aims to detect rare items in datasets and is used in many different fields, for example, healthcare, machine vision, and so on. And researchers define the anomaly in different ways. For example, data that does not follow the normally confirmed definition and the pattern is thought to be an anomaly [16]. Or anomaly can also be defined as a point that is out of its cluster. Four Types of Samples computes the ratio of the given point's minority neighbors to the number of K-neighbors. Then, based on the value, the points are divided into 4 types: Safe(S), Borderline(B), Rare(R), and Outerlier(O). The smaller value the point has, the more likely it is an outlier. They want to solve the problem of imbalanced data by recognizing outliers in a given dataset.

3 Dataset and Problem Statement

3.1 Data collection

The raw seismic data contains 19 sensors located around the volcano, and each sensor collects signals from 3 geographical directions (east-west, north-south, and vertical directions). From 2013 to 2017, there are a total of 11569 small, 565 middle, and 23 large eruptions by manual records (the detail of labeling will be given in Section 4.1). Even though the signals of sensors are different because of the distance from the sensors to the volcano, the terrain influence, or different noises. They share the statistics of eruptions, which means they share the same labels.

The data is provided by Universidad de Colima. There are 4 short-period vertical seismometers, and 6 Broadband seismic stations collecting data ranging from 0.033Hz (30 Seconds) to 50 Hz. The working sensors collect signals every 0.01 seconds (100 Hz), 864,000 data in total every day for a single sensor. Furthermore, these signals contain the tremor, the frequency from 0.5-15 Hz, and is proven to provide the most important information for volcano explosions.

However, only a small part of sensors are used throughout time. And because of some technical error, sensors fail to work at the same time. So, it is better to pick a sensor and use its data for prediction. As shown in Figure 1, SOMA, WEST, and INCA are the closest sensors to the volcano. Based on the tracks, the sensor WEST has more records from 2013 to 2017 than any other sensor. So, we use the data from WEST for the volcano explosion prediction through all the experiments.

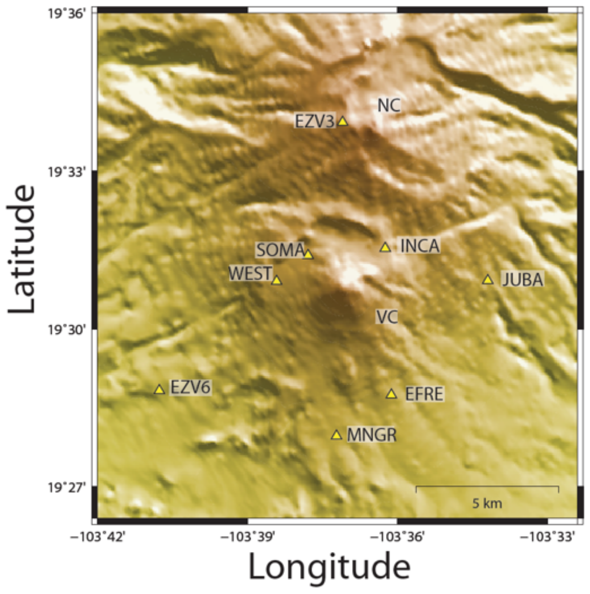

Figure 1: The distribution of sensors around the volcano

Formally, for a single sensor, the data within a day t is denoted as a track s_t, and s_t is denoted as:

$$s_t =< s_{t\_HHE},\ s_{t\_HHN},\ s_{t\_HHZ} > \tag{1}$$

where $s_{t\_HHE},\ s_{t\_HHN}\ and\ s_{t\_HHZ}$ are signals in different directions (east-west direction, north-south direction, and vertical direction separately) because sensors collect signals from different directions.

And there are two types of missing data for a single sensor. The data distribution of WEST is shown in table 1, and we can find there more than half of the days are missing for 2013 and 2017. And in 2015, around a sixth of the data is missing. Except for the data loss throughout the day, there is also some data missing within a day.

	Sequence number of date	Start date	End date
2013	[134, 134]	2013-05-14	2013-05-14
	[141, 365]	2013-05-21	2013-12-31
2014	[1, 365]	2014-01-01	2014-12-31
2015	[1, 171]	2015-01-01	2015-06-20
	[182, 283]	2015-07-01	2015-10-10
2016	[19, 366]	2013-01-19	2016-12-31
2017	[1, 15]	2017-01-01	2017-01-15
	[41, 151]	2017-02-10	2015-05-31

Table 1: The data distribution of sensor WEST from 2013-2017

3.2 Problem Statement

Volcanic eruption forecasting is crucial for protecting the local environment and economy around the volcano area. However, it is not an easy task because of the complex data collected from the volcano which usually contains a lot of noise and is always imbalanced. There may be too many eruptions for an active volcano, while too few records for an inactive one. In our work, we use the current few hours of data to predict the degree of the eruption in six hours for an active volcano. The degree of the eruption is determined by the strength of the signal picked up by the sensors. When the value of the signal is large, it suggests there is a large explosion, while a small value means a small explosion or no explosion in the feature.

However, there are some problems that make the dataset complex, such as a large explosion that may be hard to recognize if it only lasts for a short time. Or, small explosions and large explosions appear in an adjacent time period, and it is hard to distinguish them. Also, we need to test and determine suitable techniques for preprocessing the raw data which is not feasible as the input of the pipeline. Then, we also should find suitable methods to solve the problem of imbalanced data for the active volcano where there are too many explosions. So, in the next sections, we aim to find an AML pipeline to solve these problems.

4 Methods

4.1 Automated machine learning pipeline

In this thesis, the pipeline we tested and used is based on AML applications in different application fields. For example, Kefalas, M, etc. [7] put up with an AML pipeline to solve a regression problem of predicting the remaining life of an engine based on the time series recordings of engines. And M.Kefalas, etc. [17] developed an industrial AML pipeline and solved the classification problem of disease prediction. And we combined the techniques they used. Then, we modified the details and make them also feasible in multi-class classification problems because some of their tasks are binary-class classification problems. Besides, we also compare the performance of Imbalanced learning techniques: Randomly dropping, Synthetic Minority Oversampling Technique (SMOTE), and Four Types of Samples. And the pipeline consists of the following parts:

- Preprocessing: generate labels, filter and downsample raw data, generate time windows by sliding window algorithm, and fill in the missing data.
- Feature Extraction: with time series data.
- Feature Selection: in several algorithms: FRESH algorithm, Variance Threshold, and Boruta.
- Imbalanced learning: Randomly dropping, SMOTE, and an additional attribute technique, called Four Types of Samples.
- Modeling and Hyperparameter Optimization: by Tree-Based Pipeline Optimization Tool (TPOT) and Automated tuned Random forest.

First of all, we draw the pipeline we use to solve the problem of volcanic explosion forecasting in Figure 2: Preprocessing techniques, Algorithm for feature engineering, Imbalanced learning techniques, and Modeling and hyperparameter optimization techniques.

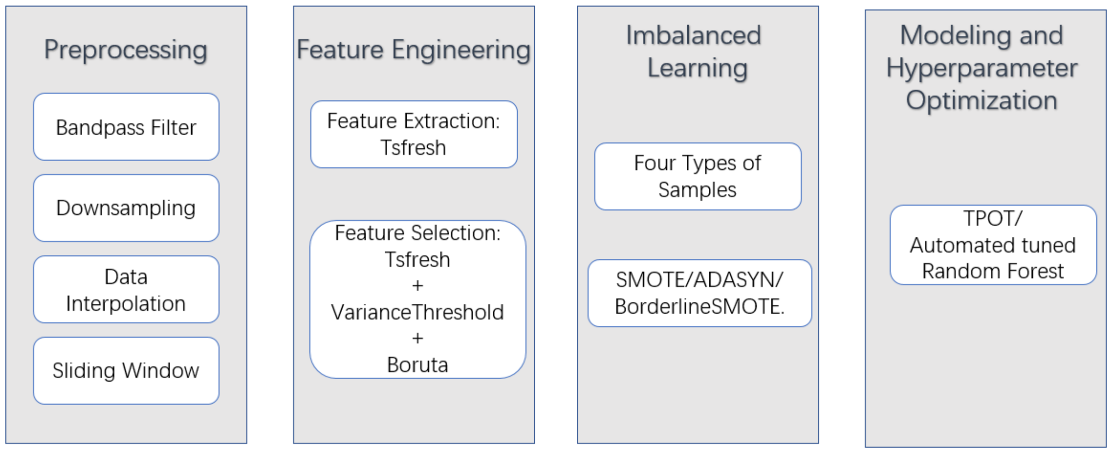
Preprocessing
Bandpass Filter
Downsampling
Data Interpolation
Sliding Window
Feature Engineering
Feature Extraction: Tsfresh
Feature Selection: Tsfresh + VarianceThreshold + Boruta
Imbalanced Learning
Four Types of Samples
SMOTE/ADASYN/ BorderlineSMOTE.
Modeling and Hyperparameter Optimization
TPOT/ Automated tuned Random Forest

Figure 2: Pipeline overview

4.2 Preprocessing

Data preprocessing in time series mainly aims to turn the raw data into a machine-understandable format because the raw data can not be used directly in most cases. Furthermore, researchers also interpolate or drop data in this step to make sure the quality of the data before the next step. And in the time series field, there are many commonly used techniques in the step, for example, downsampling and sliding window algorithm.

For data preprocessing in our dataset, the first step is to preprocess the raw signal. We need to adjust the format of the signal, such as keeping a fixed frequency signal and downsampling. Then, we applied the Sliding window algorithm to split the continuous signal. Finally, we filled in the missing data, and generate labels for corresponding windows.

Volcanologists mainly focus on a fixed range of frequencies called tremor (0.5-15 Hz) which contains important information for the prediction. And the signals out of bounds are thought to have noise as well. The bandpass filter is always used to restrain the range of frequency. First of all, in order to preprocess the raw data, we added a bandpass filter that can remove both low-frequency and high-frequency signals. In the bandpass, Fast Fourier Transform(FFT) is the core algorithm that turns the signal from the time domain to the frequency domain. Then, it removes frequencies that are out of range. And for us, we keep the frequency from 0.5Hz to 15Hz, and remove signals out of bounds. However, in our experiment, we also set up a normal group that does not remove low-frequency and high-frequency signals. By this, we aim to verify the influence of noise in the Machine learning field.

After filtering the signal, there is also another challenge that the size of the signal is not feasible for the next steps, such as, Sliding window algorithm and Feature extraction, because the sampling rate is too large and it costs too much for subsequent steps. To overcome this, we applied the technique, Downsampling. And there are mainly two kinds of downsampling methods: Downsampling by an integer factor, or by a rational factor. If the integer reduction rate is M, The former method will keep every M_{th} data from the original signal, and a lowpass filter is added to remove the high-frequency signal before this [18]. Then for the latter one, when the sampling rate is $\frac{M}{N}$ it first resamples the signal by a factor N and by the methods upsampling. Then keep every M_{th} data from the original signal. And in this way, information from given data can be saved maximally, while the size can be reduced.

And, in our experiments, we applied downsampling because of the large data size.

Otherwise, it will cost too much for the feature step. We used downsampling by an integer factor which regularly extracts features from the original signal with a fixed interval. However, it is hard to say which value is suitable for our experiment. So, we set a group of sampling rates, $R \in \{0.5, 0.01, 0.1\}$, and test them in our experiments. In conclusion, downsampling reduces the size and maximally keeps the original information.

After applying the bandpass filter and downsampling technique, we get a processed signal track with a fixed frequency and with a smaller size. Then, the next step is to split the continuous track into separate blocks, because a long track is not feasible to be the input of the model because there is only one track and a lot of labels, and they can not be matched well. Therefore, we need to manually turn it into a machine-understandable format. In order to split the track, we apply the Sliding window algorithm. Sliding window is a practical and commonly used algorithm in the machine learning field for time-series datasets. Starting with a beginning time point, it picks up data from the raw signal in a fixed interval, and each piece of extracted data is called a time window [19]. For example, in a long track s_t with d days(see Figure 3, 11 days for example) in our dataset, we can extract a set of time series windows **T** $= \{T_0, T_1 ... T_n\}$, where

$$n = (d * 24 - L)/S \tag{2}$$

T_i is the $i-th$ window in this long track. L is the length (L hours) of the windows. and S is the interval, called step (S hours) between two windows. For example, for a given window T_i, the start time point is t_i (h). If $L = 4$, it suggests the length of a time window is 4, and the end time point is $t_i + 4$. And the data within the four hours belong to the window T_i. Therefore, the length of the time window T_i is 4 hours, from t_i to $t_i + 4$. In addition, if $S = 1$, it suggests the start time point of a window is an hour before the start time point of its next window. For example, the next time window for T_i is T_{i+1}, and the start time point of the next window is $t_i + 1$ which is 1 hour behind the start time point of T_i. We say the interval between the start time points of T_{i+1} and T_i is called step S. However, just like the sampling rate, it is hard to determine the value for step S and length L. So, in the experiment, we test different settings where $S \in \{1, 2, 3, 4\}$ and length $L \in \{1, ..., 12\}$.

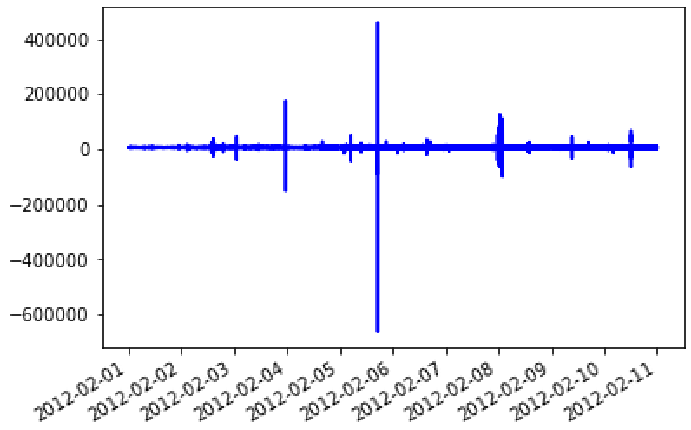

Figure 3: Raw signal in INCA sensor from 01-02-2012 to 11-02-2012

And as we mentioned in Section 3.1, a single long track $s_t =< s_{t\_HHE},\ s_{t\_HHN},\ s_{t\_HHZ} >$ where $s_{t\_HHE},\ s_{t\_HHN}\ and\ s_{t\_HHZ}$ are signals in different directions (east-west direction, north-south direction, and vertical direction separately). So, for a single track, we need to combine the signal from different directions. And the time windows are also denoted as:

$$T_i =< T_{i\_HHE},\ T_{i\_HHN},\ T_{i\_HHZ>} \tag{3}$$

$T_{i\_HHE}, T_{i\_HHN}, T_{i\_HHZ}$ are windows from different geographic directions. Therefore, T_i is the combination of three time windows, and all the windows have the same amount of data. In conclusion, Sliding window algorithm is usually applied to time series data because it splits the continuous data into windows of identical length which is feasible to be the input of the next step.

However, during the step where we apply the Sliding window algorithm, we find that except for the data missing throughout the day, some data is also missing within a day. For the former situation, we just ignore the missing day and split the period into two parts. And we apply the Sliding window algorithm in each part. And the latter situation will influence the application of the Sliding window algorithm. For example, if there are five minutes of missing data in a time point, all windows after the time point will use data from the next window in order to make all windows the same length. We check the dataset, and find that the missing data is always at the beginning or the end of a day. And for this situation, we fill the missing data with the average value of the remaining data in the day.

After generating the time windows, we also need to generate corresponding labels for them. For a given window T_i, we will get its corresponding features F_i by feature extraction algorithm in the next step. Then, we denote these features F_i by X_i which is the input of the model. So, the corresponding label is denoted by:

$$y_i = Max(D_{t+6}, D_{t+7}...D_{t+12}) \tag{4}$$

where D_t is the eruption degree at time t. t is the start time of the window T_i. And the default D_t is 0 in the dataset. And if the maximal energy captured is less than 10,000,000 between time t to $t+1$, D_t should be labeled with 1 which means small eruption. And D_t is 2 which means middle eruption while its energy is between 10,000,000 and 100,000,000. The label is 3 when energy is more than 100,000,000, and 3 indicates a large eruption. Finally, We aim to apply an automated machine learning pipeline by the input feature F_i to predict the scale y_i at time t. From 2013 to 2017, there are a total of 11569 small, 565 middle, and 23 large eruptions.

4.3 Feature Extraction

After preprocessing, a lot of separated windows are generated. However, signals in windows are still not efficient and interpretable in the future. Feature extraction in time series tries to extract features from the raw data [20]. It removes redundancies in the dataset and makes it more informational. Besides, the new statistical features are proven to be more interpretable [21]. And many feature extraction algorithms in time series are proposed, for example, Shapelet Transform [22] is a shapelet-based algorithm where shapelet describes a subset of given time series data. It is based on the distance between a shapelet and all the shapelets from the same time series data.

And in our experiments, we adopt the method of extracting massive statistical features from the time series data by pre-defined formulas. By generating a large number of features, we are able to deal with most of the time series scenarios. In our project, we define the feature extraction function as f. And for a given time window T_i at time t, the function will generate a k-dimensional feature vector f_t with the recorded signals at time t as the input. And there are already published packages implementing our feature extraction method, such as tsfresh. tsfresh is a python package that can automatically extract time series features from the raw data by statistical formulas. And these features are pre-defined in tsfresh. It uses k=794 different statistic formulas to extract features for a single time window. And features are extracted from different properties, for example, in the aspect of distribution: mean, standard deviation, in the aspect of correlation: FFT and power spectral density, in the method of linear regression: gradient and standard error, and so on. In our data, a time window is denoted as $T_i = T_{i\_HHE} + T_{i\_HHN} + T_{i\_HHZ}$, which means there are three separate windows and 3*k features generated in the end. Then, for a single time window, T_i, we combine the features extracted from all the separate windows together as the result of the Feature extraction. However, some features may contribute less to the prediction. So, we need to filter them in the next step, Feature selection.

4.4 Feature Selection

Feature Selection aims to remove irrelevant, unimportant, and redundant information from the features in the dataset while keeping the information unchanged [23]. And there are 3 kinds of feature selection algorithms: wrappers, filters and embedded methods [24]. The wrapper algorithm evaluates the features by scoring the subset of the features, for example, the error rate. And Filter methods choose another quickly calculable measure to evaluate the features, for example, mutual information [24]. Then, embedded methods put the feature selection in a part of model construction. Some features are repetitive or contribute little to the prediction. So, it is necessary to remove them from the originally extracted features to prevent the overfitting problem. After this step, theoretically, features are qualified to be the input of the machine learning model. In our dataset, 3*k=2382 features denoted F, are generated for a single time window by our feature extraction method (by tsfresh), which may contain too many redundant features or features that contribute less to the prediction. In addition, too many features may not be beneficial to the prediction, because they may cause the problem of overfitting and decrease the scores of objective functions on the test set. So, it is necessary to filter them, and keep some relevant features. In our thesis, the generated features pass through three steps orderly to remove the redundancy:

- 1. Apply FRESH (Feature Extraction and Scalable Hypothesis testing) algorithm which evaluates the importance of features, to select features in F, and generate a feature set F': $F \to F'$.

- 2. Apply VarianceThreshold to remove features in F' with low variance, and generate another new feature set F'': $F' \to F''$
- 3. Based on the feature set F'', apply Boruta, a model-based algorithm to get the final feature set, F^{Λ}

Initially, we apply the Importance evaluation method called FRESH algorithm [25]. It evaluates the importance of features, called p-value which is evaluated by the Benjamini Hochberg procedure [26]. Benjamini Hochberg procedure is a multiple testing procedure and can be used to evaluate the dependency between a feature and the target, and it will generate a relevance table for the feature matrix to the label. Finally, features with higher p-value remain as the input of the model. However, we find that even though we apply FRESH to select relevant features, there are still hundreds of features kept. And we think there the number of the features, $|F'|$, is still too large to be the input of the model, which means the features should be selected again.

Then we introduced another feature selection method called VarianceThreshold which is intuitive and powerful. Variancethreshold removes features with a variance lower than 1, and low variance means that different tracks share almost the same value in a column (in the same feature). By applying VarianceThreshold after FRESH, the number of remaining features decreases to tens when the sampling rate is low. For example, when the sampling rate is 0.1, we can decrease the number of features, $|F'|$, from more than 700 hundred to $|F''| = 84$. However, it works slightly poorly when the sampling rate is high, for example, 0.5. In this situation, the number of features decreases from more than 700 hundred to around 500 hundred. But this is understandable. Higher sampling means more data in a time window, and also means more relevant information. There, more features are more likely to be relevant to the label with less redundancy when the sampling rate is large. Subsequently, we also try just apply VarianceThreshold without FRESH, and we find the result is even worse. More than 800 hundred features remain. By far, the feature selection methods work relatively well when the sampling rate is low. And we explore more, and try to reduce the dimension of the features when the sampling rate is large. And apply another feature selection method called Boruta after the two methods.

Boruta is a model-based technique and a wrapper algorithm that uses a Random Forest model to select features. It iteratively evaluates the importance, called Z-scores, of each feature by creating shadow features based on real features. Z-scores are the indicators that show the importance of shadow features [27] and will be updated every iteration. Boruta applies the random shuffling algorithm to the real feature vectors and generates shadow features. The corresponding real features will be retained or dropped based on Z-scores. The Z-score also measures the Euclidean distance for a number from the mean of the data point. By iteratively updating, Boruta makes sure of the statistically significant meaning of the procedure of evaluation.

In our dataset, Boruta helps to remove around half of the features when the sampling is relatively high, for example, 0.5. The number of features decreases from $|F'| = 485$ to $|F^{\Lambda}| = 227$ after applying Boruta. It also removes some irrelevant features when the sampling rate is small, for example, 0.1. And the number of features decreases from $|F'| = 84$ to $|F^{\Lambda}| = 70$. This helps to further reduce the chance of overfitting. In addition, we also find we can just use Boruta without Variance Threshold and FRESH, and we still get similar results for the Feature selection step. However, it is too slow to select features because it is a model-based technique where a complex, high-dimensional dataset means a slow training process. So, we decide to keep Variance Threshold and

FRESH before applying Boruta, which can speed up the process of feature selection by Boruta. Variance Threshold and FRESH help quickly sort out the irrelevant features because they rely on computation instead of training models, and computation is quick than training models for the same dataset.

In summary, in this section, after extracting features by tsfresh, we get massive features while some of them are irrelevant to labels. Therefore, we introduce 3 feature selection techniques to remove irrelevant features: FRESH algorithm, Variance Threshold, and Boruta. We put the generated features as the input, and let them go through the three techniques orderly. Finally, we get the filtered features that are feasible as the input of the model. However, this is still another problem to be solved, the dataset is unbalanced. And we compare 3 techniques to solve this problem: Randomly dropping, SMOTE, and Four types of samples.

4.5 Imbalanced Learning

The goal of Imbalanced Learning is to solve the problem of imbalanced data distribution. As we mentioned in Section 4.1, our dataset has a total of 11569 small, 565 middle, and 23 large eruptions from 2013 to 2017. In this step, we process the features by the technique of adding an additional feature, and we compare the technique of generating and dropping data tracks. There are many techniques for help, and here we mainly implement 2 methods to overcome the problem of imbalance: generating synthetic data by Synthetic Minority Oversampling Technique(SMOTE) and its variants, and adding additional features by Four Types of Samples (safe, borderline, rare, and outlier). Then we compare SMOTE with the randomly dropping technique to show its effectiveness.

There are two intuitive but conflicting methods when solving the problem of imbalance: generating and dropping data tracks. And we compare their performance in the experiments. Randomly dropping is a commonly used, intuitive, and simple method to solve the problem, and it performs relatively well in binary classification problems to make the dataset more balanced [13]. However, in our case, it is hard to get the same effect as the binary problem does because the difference between the maximum and the minimum is too wide, and our dataset is quite complex with high-dimensional features. So, we set it as a control group to show the effectiveness of SMOTE. In our dataset, we discard the most numerous labels, so that it has the same number of labels as the second most numerous labels. For example, when Step S=1 and Length L=4, there are 20316 non-explosion data(labeled 0), 10084 small explosions(labeled 1), 1593 middle explosions (labeled 2) and 67 large explosions(labeled 3). In this scenario, we will drop 10232 non-explosion data. So, the number of non-explosion and small explosions are the same and are 10084. However, there still remains a problem that the model can not learn the statistical patterns of middle and large explosion data because they are only a very small part of the data.

Compared to Randomly dropping, SMOTE and its variant may suit our dataset more. SMOTE will generate data to make the number of all labels equal. So, models have more chances to learn the statistical patterns of middle and large explosion data. Based on the minority class in the dataset, SMOTE randomly chooses minority points, measures the Euclidean distance between a selected point and its k-nearest minority neighbors (5 in our experiments), and creates synthetic data. Then, based on SMOTE, many different versions are proposed. For example, ADASYN is another good choice for data oversampling. Compared to SMOTE, ADASYN will give higher weights to the points which are hard to learn and reuse them more frequently. BorderlineSMOTE is also based on SMOTE, but it first divides the minority data into 3 types based on labels of their neighbors: Safe, Danger, and Noise. Safe is the data whose neighbors are the same as it. Noise is the data whose neighbors are all different from it. Danger is between them. And BorderlineSMOTE mainly focuses on the Danger data, and generates synthetic based on it. And it is hard to determine which variant to be used in our experiment through theoretical analysis. So, we set a group of experiments later to compare their performance in our pipeline and determine which variant to use.

In addition, in Section 4.4, when we apply the Boruta algorithm to generate the feature set F^{Λ} in step 3, we also apply SMOTE to produce samples to make the dataset balanced. Because Boruta is a model-based technique to filter the features, the filtered features are only feasible in a biased model if using the original unbalanced dataset as its input. Therefore, we apply SMOTE to automatically generate data to make the dataset balanced. Then, the balanced dataset is feasible to be used in Boruta for feature selection.

Apart from changing the number of data, we also introduce another way to solve the problem of imbalance: adding an additional attribute to distinguish outliers. The technique is called Four types of samples. The technique is developed to solve the problem of binary classification and tries to recognize rare items or outliers in the dataset. However, our project is a multi-classification problem. It divides the samples into 4 categories: Safe(S), Borderline(B), Rare(R), and Outerlier(O). Originally, based on the number of rare items in K-neighbors, we decide what category of a point is. For example, when the K is set to 5, the category is Safe if the number of minorities is more than 3, and the point is thought to be Outerlier if there is no minority. Based on this, we expanded the rule to the multi-class problem, we decide what category of a point is by counting the number of neighbors with the same label. In our experiments, the number of neighborhoods is set to a fixed size of k=5. And the details of the rule are shown in the table below.

Types	Safe(S)	Borderline (B)	Rare (R)	Outlier (O)
Rule	$\frac{k+1}{2k} < R_{\frac{same}{all}} \leq 1$	$\frac{k-1}{2k} \leq R_{\frac{same}{all}} \leq \frac{k+1}{2k}$	$0 < R_{\frac{same}{all}} < \frac{k-1}{2k}$	$R_{\frac{same}{all}} = 0$
E.G. when the size k=5				
Rule	$\frac{3}{5} < R_{\frac{same}{all}} \leq 1$	$\frac{2}{5} \leq R_{\frac{same}{all}} \leq \frac{3}{5}$	$0 < R_{\frac{same}{all}} < \frac{2}{5}$	$R_{\frac{same}{all}} = 0$

$R_{\frac{same}{all}}$ shows the ratio of data with the same label in total neighbors. And k is the number of neighbors.

In summary, in this section, in order to solve the problem of the imbalanced dataset, we add two techniques called Four types of samples and SMOTE after the Feature selection step and before Modeling and Hyperparameter Optimization. And in the experiment part, we set up a group of experiments to test the performance of variants of SMOTE and determine which one to be used. And we set a group of experiments to verify the effectiveness of SMOTE by comparing it with Randomly dropping as well. Then, till now, we finish the generation of new features and labels. And the next step is to find a good-performed model, algorithms, and hyperparameter settings.

4.6 Modeling and Hyperparameter Optimization

Modeling and Hyperparameter Optimization are the core of the CASH problem (combined algorithm se- lection and hyperparameter optimization) which is also the goal of Automated machine learning [28]. Modeling and hyperparameter optimization in the automated machine learning field means finding a good-performed model, corresponding algorithms, and parameter settings from a specified search domain that can be pre-defined. And it is conducted by an optimizer, for example, Bayesian Optimizer, that selects from a range of algorithms and hyperparameters [12]. Then based on the evaluation metrics, also called objective functions, the optimizer discovers the best combination. Formally, Hyperparameter Optimization aims to search for the optimal setup of hyperparameters. And there are many Hyperparameter Optimization methods, for example, Grid Search, Random search, Bayesian optimization, Evolutionary optimization, and so on. Grid Search is an intuitive and classic Hyperparameter Optimization method. It conducts an exhaustive searching on the given hyperparameter searching space for learning algorithms, and the searching space should be set manually [29]. Then, Grid Search should be guided by an evaluation metric, and usually cross-validation as well. Cross-validation on the training estimates its generalization performance. However, Grid Search also suffers from the problem of the curse of dimensionality and is in high dimensional. So, it may cost too much time when the dataset is complex and large. Inspired by Grid Search, Random search conduct a random selection on the searching space instead of exhaustive searching, which helps to decrease the dimensionality of the optimization problem. Also, researchers find it outperforms Grid Search when there are only a few crucial variables influencing the result [30].

In our project, we compare two kinds of Modeling and Hyperparameter Optimization techniques: Tree-Based Pipeline Optimization Tool(TPOT) and Automated tuned Random forest. The former searches for both models and hyperparameters in a pre-defined search space(see Table 2). And the latter uses a fixed model and literately searches for hyperparameters based on the provided dataset. We also design a group of experiments to verify their performance on our dataset in next section and decide which technique to be used in our pipeline.

TPOT is tree-based and uses genetic programming to find a well-performed pipeline as the model and the corresponding hyperparameters from more than 1,000 potential pipelines in several or dozens of generations. And in every generation, the parameters of well-performed models will be passed to the next generation. It will end its search when it reaches the maximal generation or when the evaluation score exceeds the threshold. It is easy enough even for a beginner to use, and the users do not need to know knowledge of the parameter and model in machine learning. However, it is out of explicable and is hard for further development. According to the description of the authours [31], the hyperparameter optimization method is stochastic and uses a random search method to search in the possible space. Therefore, the authors believe different pipelines will be generated from different runs if the dataset is theoretically complex. Unfortunately, our dataset is both large and complex as we explained. And we run TPOT for testing several times, and get several different pipelines as the authors said. We will also tell the details of these pipelines in our experiments part. In our dataset, we set the population to 20 and the generation to 30. Population size describes the number of individuals in every generation. Then, the evaluation metric we choose is F1-weighted because it will give different weights to labels according to their proportion.

Many works have proven that Random Forest is always a good choice with low variance and less chance of overfitting [32]. It is an ensemble learning method and a combination of several decision trees, and uses the output predicted by most of the trees as the final result [33]. And it is also effective and simple enough for both the classification and regression problems in different domains and can get relatively good performance.

In our thesis, we use Automated tuned Random forest [17]. It was originally used to process the vehicle dataset, and was developed to solve the problem of cognitive function recognization of Parkinson's disease patients. Automated tuned Random forest uses the Mixed-integer Parallel Efficient Global Optimization (MIP-EGO) as the hyperparameter optimization algorithm. Efficient Global Optimization(EGO) is a global search strategy for expensive black-box functions. MIP-EGO uses a surrogate model to learn and it changes the Gaussian Process Regression in standard EGO into Random forests [34], while Random forest is feasible to handle the mixed integer data. Therefore, MIP-EGO works relatively well in the hyperparameter optimization problem where the hyperparameters are mixed-integer categorical variables [34]. And in our experiments, the search space also contains both numerical values and categorical values (See Table 2). Automated Random Forest uses 10-fold cross-validation after splitting the dataset into the training and test set. And in each fold, the training dataset is used to train the model, and the test dataset is for evaluation. Also, another 10-fold cross-validation is introduced in the training process in each epoch to adjust and optimize the hyperparameter by Bayesian Optimization. Through Automated tuned Random forest, we can get the best-performed hyperparameters for Random forest on our dataset. Also, we use F1-weighted as the evaluation metric.

Parameter	Range
Maximal depth in an individual tree	{None, 2, 4 . . . , 100}
Population of trees	{1, 2, . . . , 100}
The number of features when splitting	{auto, sqrt, log2}
Minimal number of samples required to split a node	{2, 3, . . . , 20}
Minimal number of samples required in the leaf node	{1, 2, . . . , 10}
Whether use bootstrap or not during the training	{True, False}

Table 2: The search space for the automated tuned Random forest

In the next section, we set up experiments and aim to find the suitable values of window length L, window step S, and sampling rate Q. Then, we also want to find the best-perform variant of SMOTE on our dataset, and compare the variants with the technique, Randomly dropping. Finally, we set another group of experiments to compare the performance of TPOT and Automated tuned Random Forest on our dataset.

5 Experiment setup and Result

Here we divide the experiments into 3 parts. In the first part, we mainly aim to search for the best-performed preprocessing parameters: The step S of the time series window, the length L, and also the sampling rate Q. Then, to verify the effectiveness of the technique, Four types of samples, we also add it in this part because we have more comparisons in this step. If it can improve the performance of most experiments, it proves to be effective. And the range of frequency is also taken into consideration in this part. In the second part, we compare the performance of SMOTE and its variants and decide which one should be used in our pipeline. In the third part, we want to compare the performance of Automated tuned Random Forest and Tree-Based Pipeline Optimization Tool (TPOT). First of all, we will introduce a technique, called technique Design of Experiment, and the adoptive objective functions, called the evaluation metrics, in our experiments.

5.1 Design of Experiment

We decide to use both Sliding window and Downsampling techniques for the purpose of preprocessing. However, it is hard to determine the window Length L, window Step S, and the sampling rate Q. Because, for different datasets, the values may be different to make models perform well. However, there are too many applicable combinations of Length L and Step S for Sliding window algorithm. And it is impossible to try them out because prepossessing is a time-consuming procedure that costs really long time, especially for a high sampling rate Q.

So, we also apply the technique Design of Experiment(DOE) [35] which helps to narrow the search space. And here we adopt the method of Latin Hypercube design [36] which divides the possible values into many possible small cells. Each cell has a group of values. and it chooses only a single sample from every small cell. In this way, we do not need to test all the combinations and can still get the effect as exhaustive searching does.

5.2 Objective Functions

In our paper, we mainly use 2 kinds of evaluation metrics to evaluate the model after training: F1-weighted score and accuracy. Accuracy is the most commonly used metric in many machine learning fields.

$$ACC = \frac{TP + TN}{TP + FP + TN + FN}$$

Where TP, TN, FP, FN are short of True Positive, False Positive, True Negative, and False Negative. However, in our dataset, the data is biased, and there are too many small eruptions. So, a high accuracy score does not mean a good model, for it may mistake any incoming data as a small eruption. Even though, it can still get a high score. So, we need to analyze the result with both accuracy and F1-score. F1 score takes both precision and recall score into consideration, so it is a relatively good metric to judge a model:

$$Precision(P) = \frac{TP}{TP + FP}$$

$$Recall(R) = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 * P * R}{P + R}$$

5.3 Experiment Setup

A. Preprocessing Parameters and Four types of samples

In the data preprocessing step, we want to find out the best-performed parameter setup for the Sliding window algorithm, verify the influence of signal frequency, and the performance of the additional attributes technique, Four types of samples. The resulting scores are based on a 10-fold cross-validation (CV). The dataset is randomly split into 10 folds. And every iteration(10 iterations in total), a fold is chosen to be the test set, and the remaining parts are used as the training set. For one of the Feature selection methods, VarianceThreshold, we set the threshold to 1. We use BorderlinesSMOTE in the first experiment and we set the k of k-neighbors to 5. Also, we apply k=5 in Four types of samples as well. And we use Automated tuned Random Forest for Modeling and Hyperparameter Optimization.

We first want to find out the best-performed groups of tuples (step S, length L, sampling rate Q). We set the value of the sampling rate Q to {0.5, 0.1, 0.01}. When the sampling rate is more than 0.5, it will cost too much time to extract features. There are little data remaining when the sampling rate is less than 0.01. Then the length of time windows is {2,4,6,8,12} which has common divisors with 24 hours. Then the step length is set in the range from 1 to 4, {1,2,3,4}, and it should not exceed the length of time windows because no data should be wasted. And as we mentioned before, we use the DOE technique to find a subset (6 groups) from the whole possible parameters. So, when the sampling rate is 0.5, 0.1, or 0.01 we conduct experiments with the length L and step S shown in table 3:

step	1	2	2	4	4	4
length	2	4	6	8	10	12

Table 3: The value of step S and length Q generated from DOE for experiments

We also want to verify the influence of the frequency filter and see if the noise can make the result unchanged or not. With given step S, length L, and sampling rate Q, we apply and test two filters for comparison. One frequency filter aims to extract tremor (0.5-15 Hz) is proven to be a useful range for volcano prediction as we mentioned in Section 1. And we apply another frequency filter ranging from 0.5-30 Hz which will bring the noise.

Finally, we compare the performance of the model with and without the additional attributes technique, Four types of samples. It means that for a model with a given group of sampling rate Q, step S, length L, and frequency, we initially train and evaluate the model without the additional attributes technique. Then, we add the additional attribute and use the same setting again, and record the result.

B. *Variants of SMOTE*

In the first experiment, we select out 5 groups of the best-performed tuples (step S, length L, sampling rate Q). And we also decide which frequency filter to use, 0.5-15 Hz or 0.5-30Hz. Then, we also verify the effectiveness of the Imbalance learning technique, Four types of samples. Based on the result of the first experiment, we are going to conduct experiments on variants of SMOTE, and compare them with the randomly dropping technique. We still set the threshold of VarianceThreshold to 1. The k of k-neighbors keeps unchanged and k=5 in Four types of samples. And we also use And we use Automated tuned Random Forest.

In this part, we mainly focus on three types of synthetic minority oversampling variants: SMOTE, ADASYN, and BorderlineSMOTE. And in the groups of randomly dropping, we randomly discard the highest number of labels so that they are the same as the second highest number of labels. For example, when the step and length are 2 and 6, there are 10,461 small explosions and 4,300 non-explosion data. Here, we randomly drop 6,161 data with label 1 to make them the same number. We test these techniques on 5 groups of the best-performed tuples (S, L, Q).

C. Modelling and Hyperparameter Optimization Techniques

Based on the second experiment, we select out the best-performed variant of SMOTE on our dataset. In this part, we want to compare the performance of Modelling and Hyperparameter optimization techniques: Automated tuned Random Forest and TPOT. We test their performance on 5 groups of the best-performed tuples (S, L, Q) from the first experiment, and on the best-performed variant of SMOTE. We still set the threshold of VarianceThreshold to 1. The k of k-neighbors keeps unchanged and k=5 in Four types of samples.

5.4 Result

A. Preprocessing Parameters and Four types of samples results

a. Summary of the result

We give our results in 3 tables (Table 4, 5, 6 in the end of the part). In each table, we give the evaluation scores on different steps and lengths. And in the left part, we mainly focus on the frequency between 0.5 to 15 Hz, while the right is about the frequency of 0.5-30 Hz. Then, the upper part is about how the model performs when the additional attribute technique is added, and the lower part is reversed. Subsequently, we will list our findings with some figures for illustration.

b. The impact of additional attribute technique

The Imbalanced learning technique, Four types of samples, helps to improve the performance of models. From the three tables, we can see the bold data is the best-performed one among the 72 groups. It shows when the sampling rate is 0.5, step and length are 1 and 4, frequency is 0.5-30 Hz and the model does not use the additional attribute technique, we get the best score for both accuracy and F1 score. However, it does means that the technique is useless. Instead, it is the only group where the model without the technique outperforms the model with this technique. We also draw two figures (Figure 4 and 5) to show the differences between experiments with and without the technique. It is not hard to see that in both frequencies, the additional attribute technique contributes a lot to improving the score for any given S, L, and Q. And the lower the sampling rate is, the more improvement the technique can bring, for the gap between the blue line and green line (accuracy), or between the orange line and red line (F1 score) narrows with the increase of the sampling rate. And when the sampling rate is 0.01, the left part of the graph improves by at least 15 % for each point with the additional attribute technique.

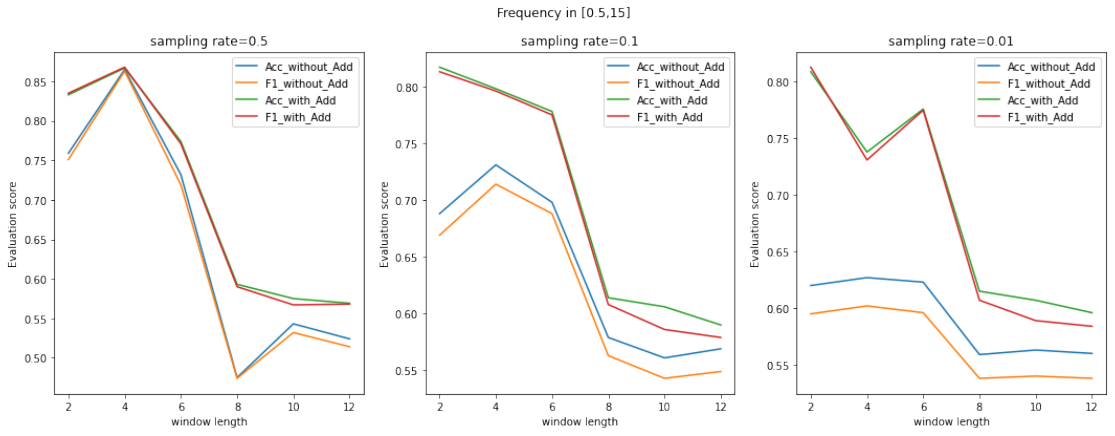

Figure 4: When the frequency is in [0.5-15 Hz], the score of models on given steps and lengths with or without the additional attribute technique

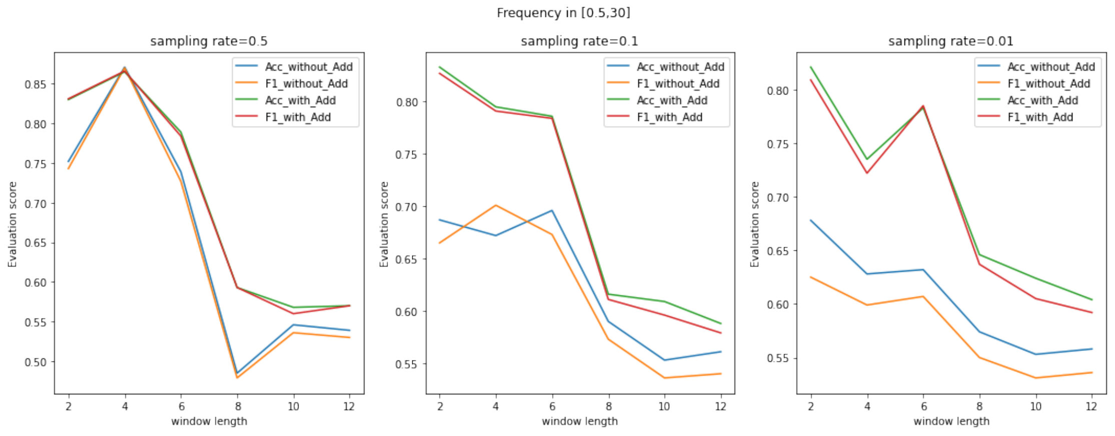

Figure 5: When the frequency is in [0.5-30 Hz], the score of models on given steps and lengths with or without the additional attribute technique

c. The impact of frequency

Subsequently, we can also make a conclusion that **the frequency of the filter does not affect the performance too much**, and the model shows close results with different settings of frequency. We can see from the tables that the differences between groups in frequency from 0.5-15 Hz and in 0.5-30 Hz are really small. Based on the data in the tables, we can also find that the accuracy and F1 score are very close. So, in Figure 6, we mainly focus on the F1 score to analyze the influence of frequencies and the trend of different lengths on different sampling rates. Also, for each line, we use the additional attribute technique, for it always improves the result in our experiments. Then, as shown in Figure 6, it supports the idea that for a given length and sampling rate, different filters do not affect the results too much because models always get close results with different filters, and there is no trend to show one filter is much better than another one. And we decide to fix the frequency to 0.5-15 Hz in the rest parts of the experiments.

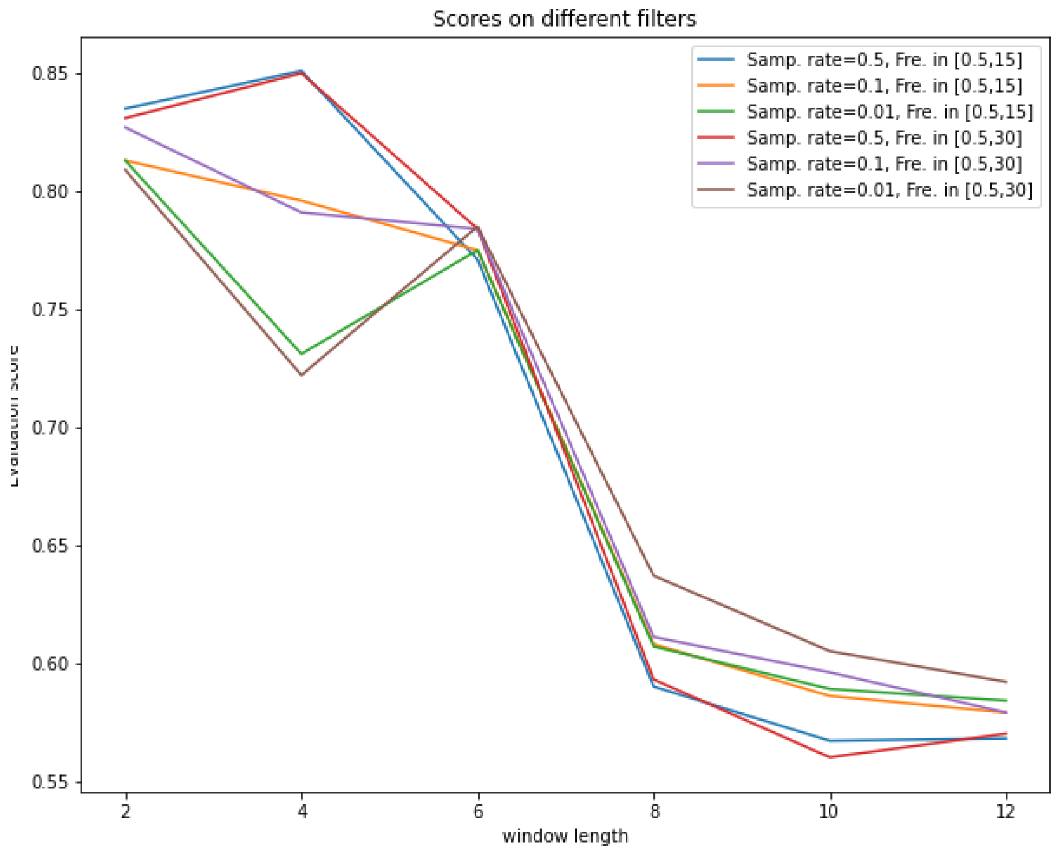

Figure 6: F1 scores on different filters

d. The impact of hyperparameters in Sliding window

Also, from the table, we can draw another conclusion: **when the tuple (step S, length L, sampling rate Q) is $\{$1, 2, 4$\}$, we get the highest score in any other conditions, but models with length 1 perform relatively well and is less sensitive to the sampling rate than the model with length 4.** The models get the highest scores when the tuple is $\{$1, 2, 4$\}$, and models get very close results when the length is 2 and the step is 1. When the length is 4, models are very sensitive to the sampling rate, and when the sampling rate is 0.01, models perform even worse than the model with length 6.

However, the table is not an intuitive representation of their differences. From Figure 7, we can find a few points that can not be seen from the table as well. In this Figure, we use the additional attribute technique, and fix the frequency to 0.5-15 Hz. Focusing on the performance of the models on different steps and lengths, we can find the graph is actually divided into two parts by length L: 2-6 and 8-12. While there is a huge decrease from $L = 6$ to $L = 8$. Therefore, the scores in the left part are always higher than the ones in the right part. Furthermore, From the Figure, we can easily find that when the sampling rate is 0.5 and the window length is 4, the model performs best, and is slightly better than the model with length 2. However, as we mentioned before, the models with length 4 actually perform unstably. There is a large gap between every two experiments when the length is 4. While models with length 2 always get F1 scores ranging from 0.81 to 0.83 for different sampling rates. In addition, when the length is larger than 8, the performance of models is close even with different window lengths, and a higher sampling rate makes models perform worse.

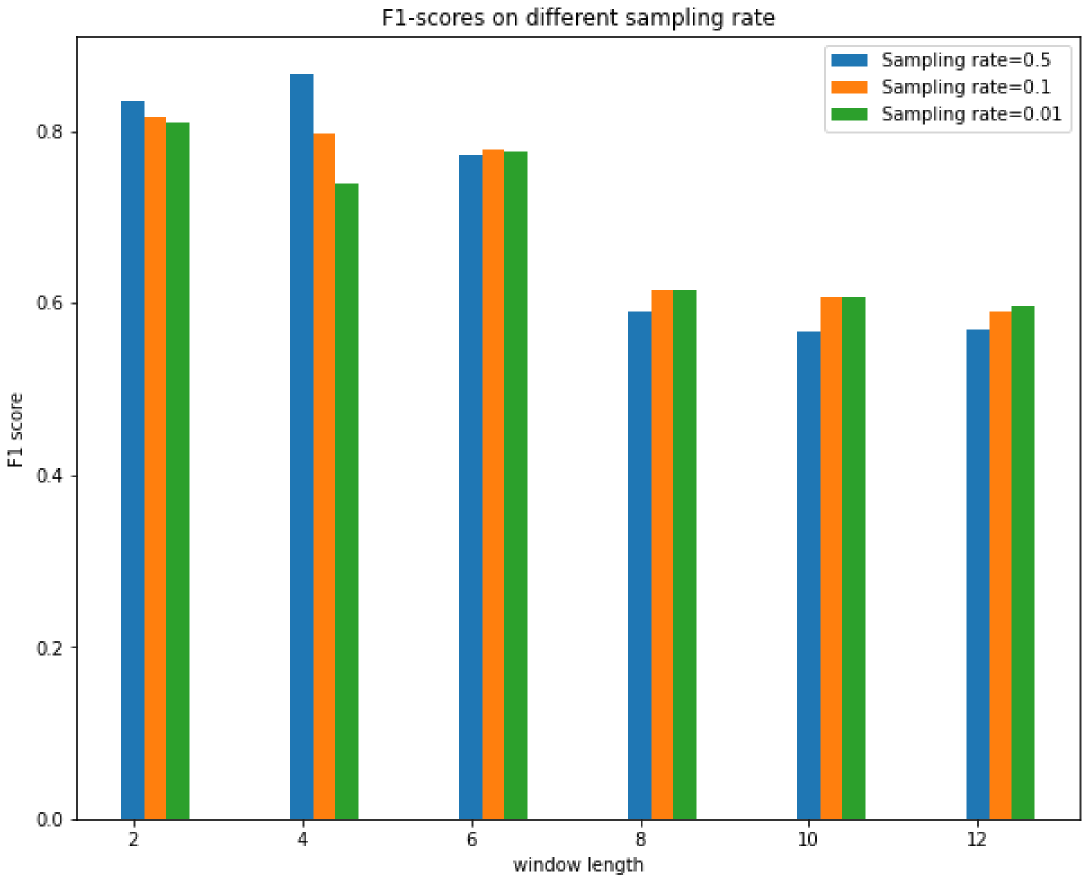

Figure 7: Bar graph of F1 scores

Sampling rate=0.5												
Frequncy	0.5-15 Hz						0.5-30 Hz					
step,length	1,2	1,4	2,6	4,8	4,10	4,12	1,2	1,4	2,6	4,8	4,10	4,12
	With additional attribute						With additional attribute					
Accuracy	0.833	0.867	0.774	0.593	0.575	0.569	0.830	0.865	0.789	0.593	0.568	0.570
F1-weighted	0.835	0.868	0.771	0.590	0.567	0.568	0.831	0.866	0.784	0.593	0.560	0.570
	Without additional attribute						Without additional attribute					
Accuracy	0.759	0.865	0.732	0.475	0.543	0.524	0.752	**0.871**	0.739	0.485	0.546	0.539
F1-weighted	0.751	0.863	0.719	0.474	0.532	0.514	0.743	**0.870**	0.727	0.479	0.536	0.530

Table 4: When sampling rate=0.5, the performance of the model on different steps and lengths with or without the additional attribute technique, using different frequency filters

Sampling rate=0.1												
Frequncy	0.5-15 Hz						0.5-30 Hz					
step,length	1,2	1,4	2,6	4,8	4,10	4,12	1,2	1,4	2,6	4,8	4,10	4,12
	With additional attribute						With additional attribute					
Accuracy	0.817	0.798	0.778	0.614	0.606	0.590	0.833	0.795	0.786	0.616	0.609	0.588
F1-weighted	0.813	0.796	0.775	0.608	0.586	0.579	0.827	0.791	0.784	0.611	0.596	0.579
	Without additional attribute						Without additional attribute					
Accuracy	0.688	0.731	0.698	0.579	0.561	0.569	0.687	0.672	0.696	0.590	0.553	0.561
F1-weighted	0.669	0.714	0.688	0.563	0.543	0.549	0.665	0.701	0.673	0.573	0.536	0.540

Table 5: When sampling rate=0.1, the performance of the model on different steps and lengths, with or without the additional attribute technique, using different frequency filters

Sampling rate=0.01												
Frequncy	0.5-15 Hz						0.5-30 Hz					
step,length	1,2	1,4	2,6	4,8	4,10	4,12	1,2	1,4	2,6	4,8	4,10	4,12
	With additional attribute						With additional attribute					
Accuracy	0.809	0.738	0.776	0.615	0.607	0.596	0.821	0.735	0.783	0.646	0.624	0.604
F1-weighted	0.813	0.731	0.775	0.607	0.589	0.584	0.809	0.722	0.785	0.637	0.605	0.592
	Without additional attribute						Without additional attribute					
Accuracy	0.620	0.627	0.623	0.559	0.563	0.560	0.678	0.628	0.632	0.574	0.553	0.558
F1-weighted	0.595	0.602	0.596	0.538	0.540	0.538	0.625	0.599	0.607	0.550	0.531	0.536

Table 6: When sampling rate=0.01, the performance of the model on different steps and lengths with or without the additional attribute technique, using different frequency filters

Summary

In this part the influence of step S, length L, sampling rate Q, frequency range, and the technique, Four types of samples. And we find that the frequency range does not influence too much, and Four types of samples helps to improve the performance. Then, based on the result, we can find the performances of 5 groups of tuples (step, length, sampling rate) are acceptable, shown in Table 7.

Step/Length	Sampling rate
1/2	0.5
1/2	0.1
1/2	0.01
1/4	0.5
2/6	0.5

Table 7: 5 best-performed groups of $\{S$, L, and $Q\}$ in the first part of experiments

B. Variants of SMOTE results

In this part of the experiment, we want to compare the performance of different Imbalanced learning techniques: Variants of SMOTE and Randomly dropping. We adopt Four types of samples, and set the frequency to 0.5-15 Hz, all called the tremor. And we conduct our experiments based on the data in Table 7.

As shown in Table 8, **the performance of SMOTE and its variants get relatively close results**. And randomly dropping labels makes the performance of models worse than the synthetic minority oversampling techniques. However, the bold data shows that ADASYN gets the highest score so far. And compared to BorderlineSMOTE, it improves by around 1 % with the same step, length, and sampling rate. And it gets relatively good performance on other groups as well. So, we decide to use ADASYN in the next part of the experiments.

Step/Length	Sampling rate	BorderlineSMOTE		SMOTE		ADASYN		Drop	
		Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score
1/2	0.5	0.833	0.835	0.868	0.849	0.844	0.847	0.720	0.699
1/2	0.1	0.817	0.813	0.812	0.810	0.813	0.814	0.590	0.570
1/2	0.01	0.809	0.813	0.810	0.815	0.807	0.812	0.502	0.489
1/4	0.5	0.867	0.868	0.870	0.870	**0.874**	**0.875**	0.798	0.780
2/6	0.5	0.774	0.771	0.810	0.807	0.794	0.794	0.615	0.581

Table 8: The performance of the different Imbalanced learning techniques on different steps, lengths, and sampling rate

C. Modelling and Hyperparameter Optimization Techniques results

In this part of the experiment, We want to compare the performance of different Modelling and hyperparameter optimization techniques: Automated tuned Random Forest and TPOT. We adopt Four types of samples, and set the frequency to 0.5-15 Hz, and use ADASYN for generating synthetic data. And we conduct our experiments based on the data in Table 7.

As shown in Table 8, **the performance of Automated tuned Random Forest is always better than TPOT on our dataset**, especially when the sampling rate is small or the step and length are large. And there is an obvious benefit for Automated tuned Random Forest, the fixed model. And we find that the modeling step generates too many different combinations of models and algorithms in TPOT, which will prevent developers from future development. And developers are hard to design a common way to improve the performance of a flexible pipeline. So, in both of aspects, Automated tuned Random Forest is better than TPOT on our dataset.

Step/Length	Sampling rate	Random Forest Hyperparameter Optimazation			TPOT		
		Accuracy	F1 Score	Modeling	Accuracy	F1 Score	Modeling
1/2	0.5	0.844	0.847	Random Forest	0.840	0.845	StackingEstimator OneHotEncoder LogisticRegression
1/2	0.1	0.813	0.814	Random Forest	0.775	0.772	Normalizer RBFSampler StackingEstimator KNeighborsClassifier
1/2	0.01	0.807	0.812	Random Forest	0.772	0.762	Normalizer OneHotEncoder RBFSampler StackingEstimator GaussianNB
1/4	0.5	0.874	0.875	Random Forest	0.872	0.873	MaxAbsScaler StackingEstimator PCA KNeighborsClassifier
2/6	0.5	0.794	0.794	Random Forest	0.722	0.717	StandardScaler PCA StandardScaler MLPClassifier

Table 9: The performance of the different Hyperparameter Optimization Techniques

6 Conclusion

In this thesis, we apply the automated machine learning pipeline, for example, TPOT and Automated Tuned Random Forest on time series volcanic data. Based on Automated Tuned Random Forest, we explore the influence of hyperparameters in the Sliding window algorithm, and the impact of the techniques, for example, SMOTE, the additional attribute technique, called Four types of samples. First of all, we test out the best-performed hyperparameters for the preprocessing step for both the Sliding window algorithm and Downsampling. And find when tuple (step S, length L, sampling rate Q) is $\{1, 2, 4\}$, we get the highest score for any other situation, while models perform stably when L=2. In addition, when L is too large (more than 6), the resulting scores are far less than the ones of smaller L. Furthermore, we also verify the effectiveness of tremors which is an important precursor of explosions on our dataset. And find that tremors are not crucial to the prediction in our experiments, and models can work with noise as well. Subsequently, based on the situation where our dataset is imbalanced, we adopt two Imbalanced learning techniques before the Modeling step: ADASYN for generating synthetic data, and Four types of samples to add the additional attribute. However, we find that the performances of variants of SMOTE are very close, and the choice of the variant will not influence the result too much. But, the additional attribute technique, Four types of samples, can improve the performance of models especially when the sampling rate Q is small or length L is large. In addition, we compare the performance of the Automated Tuned Random Forest technique with TPOT. And we find that it performs better than TPOT on our dataset in our experiment, while it can show the correlation of features as well. In conclusion, based on those findings, we get the best-performed pipeline with 87.4 % accuracy and 0.875 F1-score on the tuple $\{1, 2, 4\}$.

References

[1] Y. Yukutake, R. Honda, M. Harada, R. Doke, T. Saito, T. Ueno, S. Sakai, and Y. Morita, "Analyzing the continuous volcanic tremors detected during the 2015 phreatic eruption of the hakone volcano," *Earth Planets Space*, vol. 69, no. 1, p. 164, 2017.

[2] C. X. Ren, A. Peltier, V. Ferrazzini, B. RouetEduc, P. A. Johnson, and F. Brenguier, "Machine learning reveals the seismic signature of eruptive behavior at piton de la fournaise volcano," *Geophysical Research Letters*, vol. 47, 2020.

[3] J. Canário, R. Mello, M. Curilem, F. Huenupan, and R. Rios, "In-depth comparison of deep artificial neural network architectures on seismic events classification," *Journal of Volcanology and Geothermal Research*, vol. 401, p. 106881, 2020.

[4] M. Curilem, F. Huenupan, D. Beltran, C. S. Martin, G. Fuentealba, L. Franco, C. Cardona, G. Acuna, M. Chacon, and M. S. Khan, "Pattern recognition applied to seismic signals of llaima volcano (chile): An evaluation of station-dependent classifiers," *Journal of Volcanology Geothermal Research*, vol. 315, no. Apr.1, pp. 15–27, 2016.

[5] M. Malfante, M. Dalla Mura, J.-P. Metaxian, J. I. Mars, O. Macedo, and A. Inza, "Machine learning for volcano-seismic signals: Challenges and perspectives," *IEEE Signal Processing Magazine*, vol. 35, no. 2, pp. 20–30, 2018.

[6] D. E. Dempsey, S. J. Cronin, S. Mei, and A. W. Kempa-Liehr, "Automatic precursor recognition and real-time forecasting of sudden explosive volcanic eruptions at whakaari, new zealand," *Nature Communications*.

[7] M. Kefalas, M. Baratchi, A. Apostolidis, D. Herik, and T. Bck, "Automated machine learning for remaining useful life estimation of aircraft engines," in *2021*

IEEE International Conference on Prognostics and Health Management (ICPHM), 2021.

[8] H. Li, D. Parikh, Q. He, B. Qian, Z. Li, D. Fang, and A. Hampapur, "Improving rail network velocity: A machine learning approach to predictive maintenance," *Transportation Research Part C*, vol. 45, no. aug., pp. 17–26, 2014.

[9] E. Stühler, S. Braune, F. Lionetto, Y. Heer, E. Jules, C. Westermann, A. Bergmann, P. V. Hvell, and N. T. S. Group, "Framework for personalized prediction of treatment response in relapsing remitting multiple sclerosis," *BMC Medical Research Methodology*, vol. 20, 2020.

[10] M. Koch, H. Wang, and T. Bck, "Machine learning for predicting the damaged parts of a low speed vehicle crash," in *2018 Thirteenth International Conference on Digital Information Management (ICDIM)*, 2019.

[11] J. Kong, W. Kowalczyk, S. Menzel, and T. Bäck, "Improving imbalanced classification by anomaly detection," in *Parallel Problem Solving from Nature – PPSN XVI* (T. Bäck, M. Preuss, A. Deutz, H. Wang, C. Doerr, M. Emmerich, and H. Trautmann, eds.), (Cham), pp. 512–523, Springer International Publishing, 2020.

[12] D. A. Nguyen, A. V. Kononova, S. Menzel, B. Sendhoff, and T. Bäck, "An efficient contesting procedure for automl optimization," *IEEE Access*, vol. 10, pp. 75754–75771, 2022.

[13] K. I. Konstantinou and V. Schlindwein, "Nature, wavefield properties and source mechanism of volcanic tremor: a review," *Journal of Volcanology Geothermal Research*, vol. 119, no. 1-4, pp. 161–187, 2002.

[14] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *AI Access Foundation*, no. 1, 2002.

[15] H. He, B. Yang, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, 2008.

[16] H. Gonzalez, J. Han, Y. Ouyang, and S. Seith, "Multidimensional data mining of traffic anomalies on large-scale road networks:," *Transportation Research Record*, 2018.

[17] M. Kefalas, M. Koch, V. Geraedts, H. Wang, M. Tannemaat, and T. Bäck, "Automated machine learning for the classification of normal and abnormal electromyography data," in *2020 IEEE International Conference on Big Data (Big Data)*, pp. 1176–1185, 2020.

[18] F. J. Harris, *Multirate Signal Processing for Communication Systems*. Prentice Hall PTR, 5 2014.

[19] M. N. Norita, "Computational recognition-primed decision model based on temporal data mining approach in a multiagent environment for reservoir flood control decision," 2004.

[20] M. A. Trovero and M. J. Leonard, "Sas 2020-2018 time series feature extraction," 2018.

[21] S. Sarangi, M. Sahidullah, and G. Saha, "Optimization of data-driven filterbank for automatic speaker verification," *Digital Signal Processing*, p. 102795, 2020.

[22] A. Bostrom, "Evaluating improvements to the shapelet transform," 2016.

[23] M. L. Bermingham, R. Pong-Wong, A. Spiliopoulou, C. Hayward, I. Rudan, H. Campbell, A. F. Wright, J. F. Wilson, F. Agakov, and P. Navarro, "Application of high-dimensional feature selection: evaluation for genomic prediction in man," *Scientific Reports*, vol. 5, p. 10312, 2015.

[24] I. M. Guyon, Andrxe, and Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, 2003.

[25] M. Christ, A. W. Kempa-Liehr, and M. Feindt, "Distributed and parallel time series feature extraction for industrial big data applications," 2016.

[26] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: A practical and powerful approach to multiple testing," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 57, no. 1, pp. 289–300, 1995.

[27] M. B. Kursa and W. R. Rudnicki, "Feature selection with boruta package," *Journal of Statistical Software*, vol. 36, no. 11, pp. 1–13, 2010.

[28] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-weka: Combined selection and hyperparameter optimization of classification algorithms," *ACM*, 2012.

[29] C. W. Hsu, C. C. Chang, and C. J. Lin, "A practical guide to support vector classification," *:*, 2003.

[30] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[31] R. S. Olson, R. J. Urbanowicz, P. C. Andrews, N. A. Lavender, L. C. Kidd, and J. H. Moore, "Automating biomedical data science through tree-based pipeline

optimization," in *European Conference on the Applications of Evolutionary Computation*, 2016.

[32] S. Buschjger and K. Morik, "There is no double-descent in random forests," 2021.

[33] Hastie, Trevor, Tibshirani, Robert, Friedman, and Jerome, *THE ELEMENTS OF STATISTICAL LEARNING: DATA MINING, INFERENCE, AND PREDICTION, SECOND EDITION (SPRING*. THE ELEMENTS OF STATISTICAL LEARNING: DATA MINING, INFERENCE, AND PREDICTION, SECOND EDITION (SPRING, 2008.

[34] H. Wang, B. V. Stein, M. Emmerich, and T. Back, "A new acquisition function for bayesian optimization based on the moment-generating function," in *2017 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2017.

[35] M. Uy and J. K. Telford, "Optimization by design of experiment techniques," in *2009 IEEE Aerospace conference*, pp. 1–10, 2009.

[36] Viana and A. C. Felipe, "A tutorial on latin hypercube design of experiments," *Quality and Reliability Engineering International*, vol. 32, no. 5, pp. 1975–1985, 2016.