Online event detection for sensor data

Eduardo Ogasawara[1], Rebecca Salles[1], Luciana Escobar[1], Lais Baroni[1], Janio Lima[1], Fabio Porto[2].

[1]*CEFET/RJ - Federal Center for Technological Education of Rio de Janeiro*
Rio de Janeiro/RJ, Brazil
eogasawara@ieee.org, {rebecca.salles, luciana.escobar, lais.baroni, janio.lima}@eic.cefet-rj.br
[2]*LNCC - National Laboratory for Scientific Computing*
Petropolis/RJ, Brazil
fporto@lncc.br

Abstract.

In streaming time series analysis, it is often possible to observe the occurrence of a significant change in behavior at a certain point or time interval. Such behavior change generally characterizes the occurrence of an event. An event can represent a phenomenon with defined meaning in a domain of knowledge. The event detection problem becomes particularly relevant in this context, especially for applications based on sensor data analysis. The algorithms for detecting events online or in real-time run simultaneously with the process they are monitoring, processing each data point as they become available. Online event detection for streaming applications is a challenging problem that creates an increasing demand for high-performance computing and advanced machine learning techniques. Although there is a wide variety of methods, no silver bullet technique exists for event detection. In this context, this work contributes by providing a taxonomy for online detection of events in time series, including incremental and adaptive learning and some of the main methods addressed in the literature.

Keywords: Time Series, Event Detection, Online Event Detection, Machine Learning, Sensors

1 Introduction

Online event detection is the process of identifying the occurrence of events in streaming data. Streaming applications impose particular challenges for time series modeling. These applications involve analyzing a continuous sequence of data occurring in real-time (sensor data analysis [1]). In contrast to offline event detection (batch processing), the full dataset is not available. The system needs to observe each data record in sequential order as they arrive. Let X be a streaming time series. The model receives a continuous stream of inputs: $<\ldots, x_{t-2}, x_{t-1}, x_t, x_{t+1}, x_{t+2}, \ldots>$. At each point in time t, it should be determined whether the system presents an event. This determination must be made in real-time while seeing the next input x_{t+1}. For this, the algorithm must consider the current and previous states to decide whether the system behavior is anomalous, as well as perform any model updates and retraining [2].

Unlike offline event detection, online event detection seeks to report all anomalous events as soon as they occur in the input stream. Practical applications impose additional challenges. The sensor streams might be large and constantly evolves. It leaves little opportunity for expert intervention, manual parameter tweaking, and data labeling are usually not viable. Thus, operating in an unsupervised, automated fashion is almost a need [2].

Furthermore, in streaming applications, early detection of events is often critical. Detecting an anomaly in advance is usually better than detecting it during or after its occurrence. Detection of events can give information that is critical to decision-making. This information must be given early enough to enable preventing possible system failure. However, there is still a tradeoff between early detections and false positives. An algorithm that often makes false detections is likely to be ignored [2].

In this context, this work contributes by providing a taxonomy for the online detection of events in time series. The remainder of this paper is organized as follows. Section 2 briefly introduces background. Section 3 shows the taxonomy for online event detection, while Section 4 presents mostly used methods. Section 5 concludes.

2 Background

A time series is a sequence of observations of an object of interest collected over time. Commonly, the behavior of a time series is studied as a function of its past data [3]. Generally, one may consider a time series X as a stochastic process, that is, a sequence of n random variables, $<x_1, x_2, x_3, \ldots, x_n>$, where x_1 represents the value assumed by the series at the first (oldest) time point and x_n represents the value of the series at the newest time point [4, 5].

Most methods applied for time series modeling assume that the behavior of a time series is stationarity [5, 6]. In a stationary time series, the probabilistic behavior of every possible sequence of values $<x_{t_1}, x_{t_2}, \ldots, x_{t_k}>$ is equal to that of the time shifted sequence $<x_{t_{1+h}}, x_{t_{2+h}}, \ldots, x_{t_{k+h}}>$. Therefore, Equation 1 is valid for all $k = 1, 2, \ldots$, all arbitrary integer time points $t_1, t_2, \ldots, t_k$, all arbitrary numbers $c_1, c_2, \ldots, c_k$, and all possible time shifts $h = 0, \pm 1, \pm 2, \ldots$. Usually, such a definition of stationarity is considered too strong for most applications, and can be substitute for an evaluation for constant mean, variance, and covariance [5].

$$P\{x_{t_1} \leq c_1, \ldots, x_{t_k} \leq c_k\} = P\{x_{t_{1+h}} \leq c_1, \ldots, x_{t_{k+h}} \leq c_k\} \tag{1}$$

Suppose a time series X violates any of the constraints imposed by a stationary process. In that case, it is considered a nonstationary time series. Nonstationarity may manifest in many different ways. Generally, it implies that the mean or variance functions of a time series are non-constant and vary over time, *i.e.*, they are dependent on time t. The changes in mean or variance in time series are often due to events. They might be due to deterministic trends changes, structural breaks, level shifts, or changing variances (a condition known as heteroscedasticity). They can also be due to the presence of unit roots [3].

3 Taxonomy

Figure 1 depicts the taxonomy for online event detection. It is inspired on Habeeb et al. [7]. It is divided into seven categories based on the set of parameters found in most literature reviews. The categories are Types of events, Granularity of events, Learning modes, Strategies for detections, Approaches for modeling, Big Data tasks, and Methods. Each category is detailed in this section, except for Methods, which are presented in more detail in Section 4.

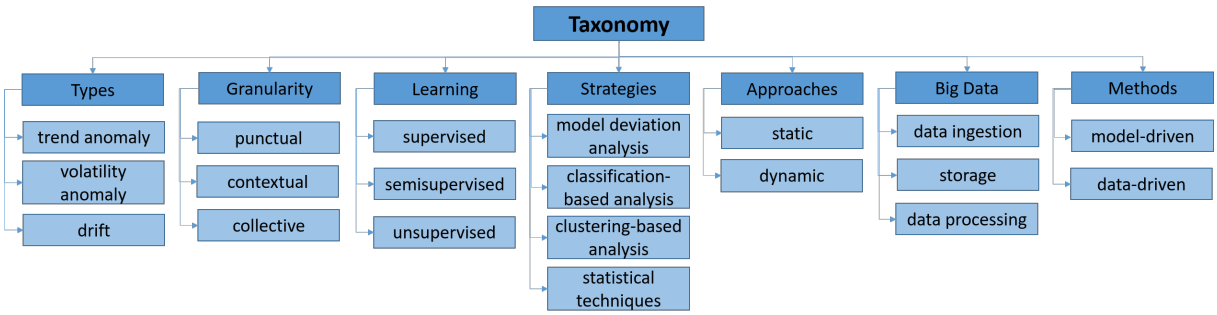

Figure 1. Taxonomy of online event detection techniques.

Types of events. Commonly, events detected in time series refer to anomalies. Anomalies are observations that stand out because they appear not to be generated by the same process as the other observations in the time series. They refer to a special kind of outlier, particularly an outlier of interest [8]. Anomalies can be modeled as isolated observations of the remaining data based on similarity or distance functions. In the time series context, there is a particular interest in detecting anomalies that may represent the occurrence of an event that escapes the trend inherent in the X generating process. They are referred to as trend anomalies.

The literature presents several methods for detecting trend anomalies. Among them are those based on moving average, and K-Nearest Neighbors (KNN-CAD), and decomposition. The decomposition method adopts an approach that comprises decomposing the time series into three components: trend, seasonality, and residuals, on which the search for anomalies occurs [9].

Time series anomalies may present themselves not as deviations from an inherent trend but as variations in data volatility. Most financial time series exhibit non-linear properties, as the volatility of these series varies widely over time, and they are commonly associated with risk. This interpretation is also relevant in many other domains. Thus, there is a demand for the study of the volatility of time series. Econometric models appear to

address the non-linearity of data, including stochastic volatility, such as ARCH and GARCH, the latter being the most well-known and applied [10]. For detecting volatility anomalies, GARCH based models are often adopted. GARCH-type models involve estimating volatility based on previous observations.

In non-stationary scenarios, data is expected to evolve or change over time, including the underlying data distribution. When it happens, we have a phenomenon known as concept drift. Formally, concept drift between time point t_0 and time point t_1 can be defined as $\exists\ X\ :\ p_{t_0}(X, y) = p_{t_1}(X, y)$, where p_{t_0} denotes the joint distribution at time t_0 between the input time series X and the target variable y (class normal or anomalous) [11].

The general assumption is that concept drifts happen unexpectedly and unpredictably. However, there are cases in which they can be known beforehand based on particular environmental events. Moreover, the change may take different forms. It can happen suddenly/abruptly, incrementally, gradually, or even previously seen concepts may reoccur after some time [11]. In such cases, models must unsupervised adapt to a new definition of *normality* [2]. In that case, adaptive learning refers to updating predictive models online during their operation to react to concept drifts [11].

Adaptive learning algorithms are primarily based on either an active or passive approach. Algorithms following the active approach specifically aim at detecting concept drift. It is related to the detection of change point events. Change points are described as the points or intervals in time that represent a transition between different states in a process that generates the time series data [12]. Algorithms following the passive approach update the model every time new data is given as input, regardless of whether or not a drift occurs. Both active and passive approaches intend to provide an up-to-date model [13].

The seminal method of detecting change points (SCP) has become a reference in the literature [14]. Other approaches use inertial functions to detect change points, such as the work of Raza et al. [15], who use Exponentially Weighted Moving Average (EWMA).

Granularity of events. Generally, anomalies fall into one of three different categories: punctual, contextual, or collective [7]. Punctual anomalies are individual data instances that differ from the residual of the data. Contextual anomalies are data instances that are anomalous in a specific context. Then it is referred to as a conditional anomaly. Collective anomalies are data instances that are anomalous concerning the entire data set. Data instances in a collective anomaly may not be anomalies by themselves. However, their occurrence together as a collection is anomalous. Mainly the collective category is used for the real-time or online anomaly detection [7].

Learning modes. Based on the availability of the *regular* or *anomalous* labels, event detection techniques can be created in three different learning processes. In supervised learning, the techniques assume the availability of a training data set that has labeled instances. In semi-supervised learning, they assume that the training data has labeled instances only for the regular class. On the other hand, techniques that operate in unsupervised learning do not require labels in training data and thus are most widely applicable. The techniques in this category implicitly assume that regular instances are more frequent than anomalies [8].

Strategies for detections. Event detection methods found in the literature are usually based on four general strategies: (i) model deviation analysis, (ii) classification-based analysis, (iii) clustering-based analysis, or (iv) statistical techniques [8]. The model deviation is one of the most commonly adopted and is based on an analysis of model deviation. First, a model (such as statistical or machine learning) is fitted to the available data. Then events are identified as the observations that most deviate from the fitted model. Classification-based and clustering-based represent event detection as a problem of classification or clustering. In that case, abnormalities are identified by comparing them to samples previously learned or clustered. Analogously, domain-based strategies compare new data samples against what is expected based on expert knowledge. Finally, statistical techniques identify deviations from the data distribution.

Approaches for modeling. Despite the detection of events in real-time data having practical and substantial applications across several industries, there are still few solutions to this problem [2, 7]. Current approaches for online event detection can be categorized into two major categories. In the first, a static model is trained on large data samples. Then, it is deployed on data streams. In the second, the event detection model is initialized on a data sample and then learned incrementally as new data arrives, so it is called dynamic [7].

Indeed, online detection is a much more complicated problem that presents several challenges. Accommodating large volumes of streaming data in the main memory of a machine is impractical and often infeasible. Hence, online processing is required. In this case, data models can be retrained using recent batches of data or trained incrementally by continuous updates. Incremental algorithms process input examples one by one (or batch by batch). It updates the data model after receiving each example. Typically, for any new batch of data, the update operation of the model is based on the previous one. However, incremental algorithms may also have random

access to previous examples or the most representative ones. In this case, these algorithms are said to have partial memory [11].

Learning algorithms often need to operate in environments that are dynamic and can change unexpectedly. A desirable property of these algorithms is their ability to incorporate new data. It, in turn, brings us to the stability-plasticity dilemma [11, 13]. That is, how can a learning system be designed to remain adaptive in response to significant changes and yet remain stable in response to irrelevant changes?

Suppose the data-generating process is not strictly stationary. In that case, the underlying concept, which is being predicted, may be changing over time [11]. Learning in nonstationary environments requires adaptive or evolving approaches to monitor the underlying changes and adapt a model to accommodate these changes. Given that it is the case for most real-world applications, there is a strong demand for effective and efficient algorithms for learning from (and adapting to) evolving or drifting environments [13]. Adapting to such changes can be seen as a natural extension for the incremental learning systems that learn predictive models example by example. Adaptive learning algorithms can be seen as the next step to incremental learning ones that can adapt to the evolution of the data-generating process over time [11].

Big Data tasks. It is crucial to immediately process the data collected to detect any potential threat in the network. However, the existing traditional monitoring tools are not well suited to handle Big Data streaming. In that case, it becomes essential to combine machine learning algorithms and Big Data technologies to process real-time Big Data to detect events efficiently [16]. Machine learning helps the analysis of collected data to detect and monitor the network. At the same time, various real-time Big Data technologies can help to process and stream the enormous amount of network data in real-time and near real-time constraints [7]. Most Big Data online event detection applications involve phases of sensor data ingestion, storage [17] and processing [18].

Data ingestion is how data is moved from a source to a destination, where it can be stored and analyzed later. Given the usual large volume of data, online event detection is commonly done in batches. The ability to ingest and process data at speed and scale is critical. Cloud-based services need to efficiently ingest data into on-premises systems, cloud repositories, and messaging hubs like Apache Kafka. Such ingestion enables real-time processing. Apache Kafka is mainly used to construct data pipelines in real-time and develop online streaming applications [16].

Data storage systems must have high storage capacity, performance, fail-safe features. There is a need to support record requests with consistency. They should enable fast, low-cost, reproducible reads and writes of large data streams. The Hadoop Distributed File System (HDFS) and data lake are widely used examples of these types of data storage. HDFS allows fast data transfer between compute nodes. Conversely, a data lake is a centralized repository that enables structured and unstructured data to be stored under different scales [19]. It is a flexible and cost-effective option for storing event data, but it can be challenging to get operational.

The data processing phase is responsible for consuming data from the storage layer. Data processing can be parallel, distributed, and performed in real-time [20]. Some examples are tools currently used to facilitate this process, such as Hadoop, Spark, and Apache storm. Hadoop is one of the most popular Big Data technology frameworks used to store large amounts of data across multiple computing nodes. Spark receives data from Kafka and processes it in real-time, using machine learning algorithms for anomaly detection. The storm is a framework (similar to Spark) used to write applications for processing Big Data [16, 18].

4 Methods

Due to the increasing demand for event detection in real-world applications, many event detection methods have been developed and are currently available in the literature. This section presents a selected subset of relevant methods for online event detection. The methods are divided into model-driven or data-driven according to the approach adopted. While model-driven techniques are more traditional and correlated with statistics, data-driven techniques are enabled by machine learning.

4.1 Model-driven methods

In real-time event detection, model-driven techniques are frequently used due to the smaller computational requirements. Some of these techniques include sliding windows size, outlier tests (such as extreme studentized deviate and k-sigma), change point detection, statistical hypotheses testing, exponential smoothing, and eccentricity anomaly detection [2, 7].

The main model-driven methods are based on one of these techniques to model time series: linear regression, autoregressive models, state-space, moving average, empirical mode decomposition (EMD), or volatility.

There are many different methods for detecting events. However, three stand out for being references in detecting trend anomalies, volatility anomalies, and change points, respectively: Exponentially Weighted Moving Average (EWMA), Generalized ARCH (GARCH), and Seminal Change Point (SCP).

EWMA is a well-referenced method for event detection, forecasting, and estimating trend changes in time series. It can detect small changes in the moving average of a time series [15]. To establish the importance of current and past observations in a time series, the EWMA assigns a weight constant (λ). As shown in Equation 2, in addition to the weight constant we have y_t as the time observation value t.

$$z_t = \lambda y_t + (1 - \lambda)\, z_{t-1}, \tag{2}$$

In the Equation 2 we have that z is the exponentially weighted moving average (EWMA), with z_0 being the average of the initial data [21]. λ is a smoothing constant such that $0 \leq \lambda \leq 1$. Such constant is determined considering the size of the change to be detected [22]. Therefore, EWMA assigns a higher weight value to more recent data, while older data receives lower weight values.

The GARCH model is commonly used to estimate the volatility of time series. It is a nonlinear time series model, where a time series X is given from μ_t, which is the average component. GARCH is presented in Equation 3, where w_t is the noise sequence given by i.i.d. N$(0, 1)$, and the conditional distribution of $\tilde{x_t} = x_t - \mu_t$, given $\tilde{x}_{t-1}, \tilde{x}_{t-2}, \ldots$ is N$(0, \sigma_t^2)$ [10].

$$x_t = \mu_t + \sigma_t w_t \tag{3}$$

The seminal method of detecting change points (SCP) method has become a reference in the literature related to the detection of change points [14]. It adopts the following strategy. For each data window, a model is adjusted. Then, considering a reference point, two models are adjusted: before and after the reference point. A change point is detected if the fitting errors are significantly reduced compared to whether the observation in the reference point is not considered [12].

The SCP gave rise to several other methods related to the detection of events in time series. An example is the *ChangeFinder* (CF)[12], which searches for anomalies and change points through two phases. In the first, a α model is fitted to the time series X resulting in $\hat{X}$. From the residuals of the series s, defined in Equation 4, the anomalies are marked. The second phase consists of defining a new series $\overline{s}_p$, which is defined from the moving averages of s with p terms. Finding anomalies in this new series $\overline{s}_p$ results in the detection of change points.

$$s_i = (\hat{x}_i - x_i)^2\,,\ \hat{x}_i = \alpha(x)_i \tag{4}$$

4.2 Data-driven methods

In contrast to the methods presented previously, the data-driven methods based on machine learning are not necessarily restricted to certain kinds of applications/problems. In the same way, they are not restricted to detecting a specific type of event. The main data-driven methods used to events online event detection are K-Nearest Neighbors Conformal Anomaly Detector (KNN-CAD), Feed-Forward Neural Network (NNET), Convolutional Neural Networks (CNN), Support Vector Machine (SVM), Extreme Learning Machine (ELM), and K-MEANS.

Feed-Forward Neural Networks (NNET) are widely explored in time series prediction studies. This network is composed of nodes organized in layers, where each node represents an artificial neuron. Its structure consists of several interconnected neurons, where the input is modified by weight and added to all other inputs. The resulting value is passed by a transfer function (linear or non-linear) to one or more neurons in the next layer. The process can be adopted in research involving the prediction of univariate time series or non-linear signals, such as speech and images [23]. The weights are adjusted to map a certain input pattern to the desired output in the learning process. At the same time, the network size involves the number of layers, the number of neurons per layer, and the number of connections [24].

It combines feature extraction and classification in a single network. The composition of a CNN model consists of several layers for specific tasks, being composed of three distinct layers: the input layer, the convolutional layer, and the pooling layer. The last is responsible for reducing the size of the input data. Convolution layers in datasets can be applied as an extractor of implicit features in the data. A convolution process is presented in Equation 5, where g is the input layer, h is one of the k filters that a CNN optimizes for an objective function during the learning process. It considers the moment t, $*$ is the convolution operator, and n is a hidden layer in the neural network. The goal of deep learning is to discover multiple levels of representation, looking for high-level features that can represent the most abstract semantics of the data [25].

$$(g * h)[n] \equiv \sum_{t=0}^{k} g[k-t]h[t] \tag{5}$$

SVM (Support Vector Machines) is a machine learning method commonly used in pattern recognition and classification tasks. This method maps observations to a high-dimensional space using kernel transformations and aims to recognize the data patterns. SVM separates a dataset into classes, forming groups with similar characteristics. Furthermore, it seeks to find classifiers with the greatest distance between observations of different classes through support vectors [26]. SVM has a counterpart for regression problems, which is commonly called Support Vector Regression (SVR) [27].

An Extreme learning machine (ELM) is a learning method with good results in event detection and pattern recognition. Among the advantages of its implementation is a simple structure, low computational cost, and good versatility [28]. ELM excels in terms of performance in the following aspects: *(i)* extremely fast learning speed; *(ii)* better generalization performance compared to gradient-based learning algorithms; and *(iii)* ability to work with smooth and non-differentiable activation functions [29]. It has less dependence on parameter adjustment, which is another feature that improves its effectiveness [30].

K-MEANS is an unsupervised algorithm used to identify clusters. The parameter k establishes the number of clusters. All observations are associated with a particular cluster. The clustering is performed based on the distance of the observations to the centroids of each cluster. Each centroid is iteratively defined by averaging the observations within each cluster [31].

5 Final Remarks

This work reviews the basic categories related to online event detection according to the proposed taxonomy. Even though the state-of-the-art methods in online event detection have progressed in recent years, there is still no silver bullet. More effort is needed to obtain adequate event detection models, especially when balancing between accuracy and performance.

Acknowledgements. The authors thank CNPq, CAPES (finance code 001), Petrobras, and FAPERJ for partially funding this research.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] M. Pimentel, D. Clifton, L. Clifton, and L. Tarassenko. A review of novelty detection. *Signal Processing*, vol. 99, pp. 215–249, 2014.

[2] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, vol. 262, pp. 134–147, 2017.

[3] D. M. Hanssens, L. J. Parsons, and R. L. Schultz. *Market Response Models: Econometric and Time Series Analysis*. Springer, Boston, Mass., 2nd edition edition, 2003.

[4] P. Esling and C. Agon. Time-series data mining. *ACM Computing Surveys*, vol. 45, n. 1, 2012.

[5] R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer, New York, NY, 4 edition, 2017.

[6] D. N. Gujarati and D. C. Porter. *Basic Econometrics*. McGraw-Hill Publishing, 2008.

[7] R. A. Habeeb, F. Nasaruddin, A. Gani, I. Targio Hashem, E. Ahmed, and M. Imran. Real-time big data processing for anomaly detection: A Survey. *International Journal of Information Management*, vol. 45, pp. 289–307, 2019.

[8] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, vol. 41, n. 3, 2009.

[9] M. Gupta, J. Gao, C. Aggarwal, and J. Han. Outlier Detection for Temporal Data: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, n. 9, pp. 2250–2267, 2014.

[10] R. Carmona. *Statistical Analysis of Financial Data in R*. Springer Science & Business Media, 2013.

[11] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, vol. 46, n. 4, 2014.

[12] J.-I. Takeuchi and K. Yamanishi. A unifying framework for detecting outliers and change points from time series. *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, n. 4, pp. 482–492, 2006.
[13] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar. Learning in Nonstationary Environments: A Survey. *IEEE Computational Intelligence Magazine*, vol. 10, n. 4, pp. 12–25, 2015.
[14] V. Guralnik and J. Srivastava. Event Detection from Time Series Data. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pp. 33–42, New York, NY, USA. ACM, 1999.
[15] H. Raza, G. Prasad, and Y. Li. EWMA model based shift-detection methods for detecting covariate shifts in non-stationary environments. *Pattern Recognition*, vol. 48, n. 3, pp. 659–669, 2015.
[16] L. Rettig, M. Khayati, P. Cudre-Mauroux, and M. Piorkowski. Online anomaly detection over Big Data streams. In *Proceedings - 2015 IEEE International Conference on Big Data, IEEE Big Data 2015*, pp. 1113–1122, 2015.
[17] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The Hadoop distributed file system. In *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST2010*, 2010.
[18] W. Inoubli, S. Aridhi, H. Mezni, M. Maddouri, and E. Mephu Nguifo. An experimental survey on big data frameworks. *Future Generation Computer Systems*, vol. 86, pp. 546–564, 2018.
[19] H. Fang. Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem. In *2015 IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems, IEEE-CYBER 2015*, pp. 820–824, 2015.
[20] M. Zaharia, R. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica. Apache spark: A unified engine for big data processing. *Communications of the ACM*, vol. 59, n. 11, pp. 56–65, 2016.
[21] K. Atashgar, N. Rafiee, and M. Karbasian. A new hybrid approach to panel data change point detection. *Communications in Statistics - Theory and Methods*, 2020.
[22] H. Assareh, I. Smith, and K. Mengersen. Change point detection in risk adjusted control charts. *Statistical Methods in Medical Research*, vol. 24, n. 6, pp. 747–768, 2015.
[23] A. J. Aljaaf, T. M. Mohsin, D. Al-Jumeily, and M. Alloghani. A fusion of data science and feed-forward neural network-based modelling of covid-19 outbreak forecasting in iraq. *Journal of Biomedical Informatics*, vol. 118, pp. 103766, 2021.
[24] F. Riese and S. Keller. Supervised, semi-supervised, and unsupervised learning for hyperspectral regression. *Advances in Computer Vision and Pattern Recognition*, pp. 187–232, 2020.
[25] T. Guo, J. Dong, H. Li, and Y. Gao. Simple convolutional neural network on image classification. In *2017 IEEE 2nd International Conference on Big Data Analysis, ICBDA 2017*, pp. 721–724, 2017.
[26] V. Chauhan, K. Dahiya, and A. Sharma. Problem formulations and solvers in linear SVM: a review. *Artificial Intelligence Review*, vol. 52, n. 2, pp. 803–855, 2019.
[27] Rahul and B. Choudhary. An Advanced Genetic Algorithm with Improved Support Vector Machine for Multi-Class Classification of Real Power Quality Events. *Electric Power Systems Research*, vol. 191, 2021.
[28] J. Ma and C. Yuan. Adaptive safe semi-supervised extreme machine learning. *IEEE Access*, vol. 7, pp. 76176–76184, 2019.
[29] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, vol. 70, n. 1-3, pp. 489–501, 2006.
[30] A. Batool, M. Nisar, J. Hussain Shah, A. Rehman, and T. Sadad. Ielmnet: An application for traffic sign recognition using cnn and elm. pp. 132–137. cited By 0, 2021.
[31] A. Muniyandi, R. Rajeswari, and R. Rajaram. Network anomaly detection by cascading k-Means clustering and C4.5 decision tree algorithm. In *Procedia Engineering*, volume 30, pp. 174–182, 2012.