Automated Real-Time Anomaly Detection of Temperature Sensors through Machine-Learning

Debanjana Nayak and Harry Perros
Department of Computer Science
North Carolina State University

Fast identification of faulty sensors is necessary for guaranteeing their robust functions in diverse applications ranging from extreme weather prediction to energy saving to healthcare. We present an automated machine-learning based framework that can detect anomalies of temperature sensor data in real-time. We adopted a purely temporal approach that utilizes a univariate time-series (UTS) generated by a single sensor. The framework divides the UTS into subsequences, models each subsequence stochastically as an autoregressive function, and finally mines the function parameters with a One-Class Support Vector Machine (OC-SVM) that classifies any outlier as an anomaly. Extensive experimentation showed that the framework identifies correctly with high degree of accuracy normal and anomalous data.

1. Introduction

The Internet-of-Things (IoT) has attracted a lot of popularity and promises to play an important role in the future. Its applications are manifold extending from smart connected homes to healthcare to transportation [1]. With advancements in the IoT technologies, a variety of sensors with computing and communication capabilities are increasingly being deployed to facilitate diverse energy and environmental applications. With the rising concern of global warming and climate change, the development of sensors that can accurately indicate drastic changes in local weather and distinguish patterns of gradual climate change is emerging as an indispensable task for the protection of the environment [2-4]. Herein, we present an automated real-time machine-learning based framework that can detect anomalies in the sensor data of a critical weather variable, temperature. This is crucial for rapid identification of faulty sensors and initiating corrective measures.

Anomaly detection techniques are often applied to the data collected from sensors to allow prompt identification of failures (or impending failure), misconfigurations, and degradations. Data-driven anomaly detection techniques are widely used. However, a number of them focus on spatial and spatiotemporal methodologies [5-7], where failure detection for each sensor is influenced by data observed by its neighbors. Such approaches are formulated on the assumption that spatially co-located sensors are observing similar values for the same parameters. They may not work in the cases of sparsely distributed sensor networks; where neighboring sensors are geographically far apart and may end up monitoring quite different conditions. In such scenarios, spatial techniques may introduce errors in the anomaly detection.

Here, we propose an anomaly detection framework that is purely temporal in nature and utilizes a univariate time-series (UTS) generated by a single sensor. There are several existing methods proposed for anomaly detection in an UTS [8-10], including One-Class Support Vector Machines (OC-SVM) [11-15] and different kinds of autoregressive models [16-20]. It is generally observed that a highly imbalanced binary class distribution exists between normal and anomalous behavior and is an important classification problem [21, 22]. Anomalies are rare and all characteristic features may not appear in training instances. OC-SVM is a well-known technique for dealing with this problem—it is a sparse solution maximizing the separation between normal and anomalous data [23]. Extensive research has also been carried out with prediction-based statistical approaches such as univariate autoregressive model (AR), autoregressive and moving average model (ARMA), and autoregressive integrated moving average model (ARIMA). Such models attempt to predict future values of the UTS based on past observed values. The deviation between predicted and observed values help in identifying anomalies. Our approach mines the temporal nature of an UTS. First, it divides the UTS into subsequences through a sliding window so as to capture the recent nature of the UTS. Then, it models each of them stochastically using an autoregressive function, and finally mines the parameters of the stochastic models with an OC-SVM. As in prediction-based models, we also assume an underlying stochastic process that defines the time-series. However, predictions are error-prone and need to be avoided. So, we mine the stochastic models instead of using them for predictions. The assumption of an underlying stochastic process governing the UTS enables the clustering and classification of the model parameters. The OC-SVM forms a tight boundary around the normal class and any outlier is regarded as an anomaly.

We have investigated our anomaly detection framework on the temperature data collected by the State Climate Office of North Carolina [24]. The results of our experiments reflect that our approach was effective in capturing the trends and seasonality of temperature and identifying anomalies. Our work can be extended further for anomaly detection in other weather variables such as air pressure, humidity, and wind velocity. Note that each such variable may be governed by a distinct stochastic process. Therefore, the same approach may be inadequate for all weather variables, which may warrant specific modeling for each.

The remaining content is organized as follows: Section 2 describes the proposed anomaly detection framework. Section 3 outlines the different experiments carried out and evaluates its performance. Finally, we provide our concluding remarks in section 4.

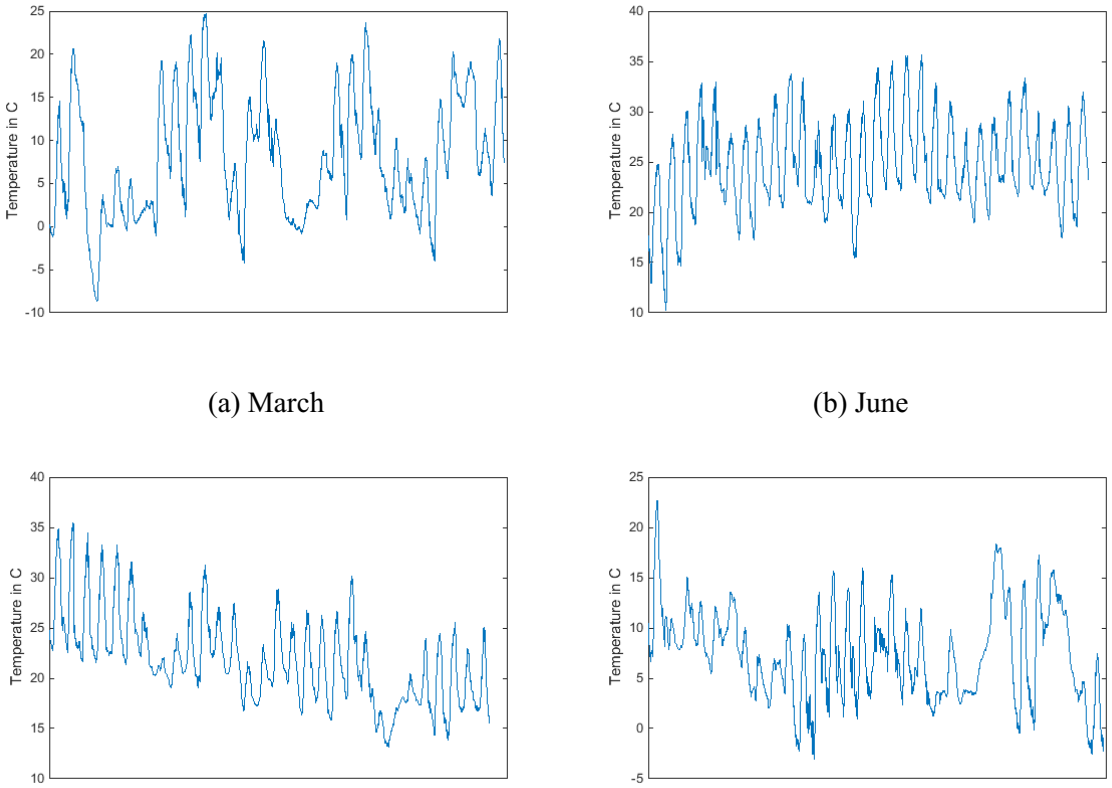
2. Anomaly Detection Framework

2.1 The Autoregressive Nature of UTS

First and foremost, we need to analyze the nature of the UTS temperature. This will aid in an understanding of our design decisions. We have access to the temperature data collected by the State Climate Office of North Carolina [24]. Our study was conducted on data collected by the weather station LAKE - Lake Wheeler Road Field Lab. It is situated in the city of Raleigh, Wake County, North Carolina (Latitude: 35.72816°, Longitude: -78.67981°, Elevation: 382 ft. above sea level). The results provided in section 2 are from experiments conducted on data collected in the year 2014. Similar experiments were also conducted for the years 2011-2018 and we obtained similar results. The experiments described in section 3 were conducted on the entire dataset, comprising of data collected in the years 2011-2018.

Temperature is observed and recorded every minute. Each datapoint undergoes quality control by the climate office and is eventually marked ‘good’ or ‘bad’. The database does occasionally suffer from missing and bad data. Since temperature always changes gradually, we use interpolation for their reconstruction. Such occurrences are rare and have no significant impact on our methodology.

In Fig. 1, we have presented temperature against time to provide a sense of its behavior. Since temperature is recorded every minute, the dataset is extremely large. For convenience, instead of presenting the data for the whole year of 2014 (i.e., 365*24*60 = 525600 minutes), we give the data for March, June, September, and December. Experiments and analysis, however, were conducted with the entire dataset and identical conclusions were reached.



(a) March (b) June

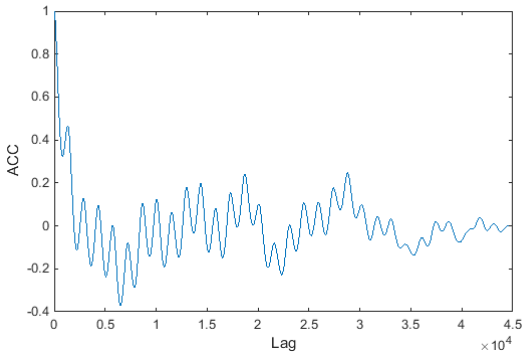
(c) September (d) December

Fig. 1: Temperature vs Time

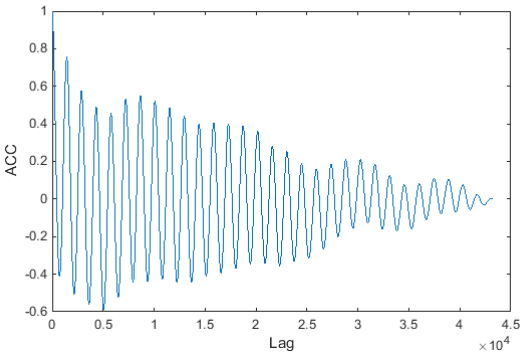
A few observations can be drawn form Fig. 1. The data clearly displays a distinct seasonality. Nights are usually cooler than days. As the night progresses to dawn to daylight, temperature keeps rising. Conversely, as the day advances to dusk into darkness, temperature starts falling. If perceived carefully, disticnt spikes can be noticed in each month’s data, where indicates the number of days in each month. Also, change in temperture is gradual and drastic changes were never noticed in the entire dataset.

In Fig. 2, we have further examined the data by plotting its auto-correlation function. As is evident from the plots, temperature at time is highly correlated with that of and starts decaying as time progresses.

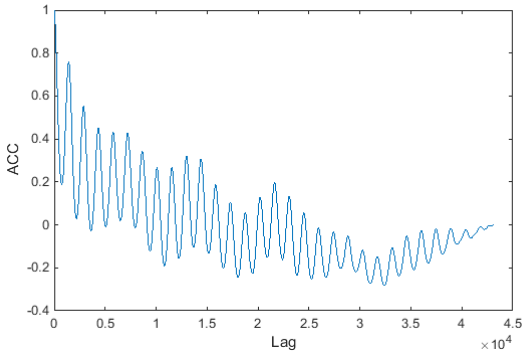
Thus, it can be concluded that temperature at any point of time is strikingly similar to that in the recent past; and therefore, the past may indicate the future. distinct spikes can be noticed in these plots, as well. This is because temperature during daytimes over multiple consecutive days are positively correlated with each other, while being negatively correlated to that at nights.



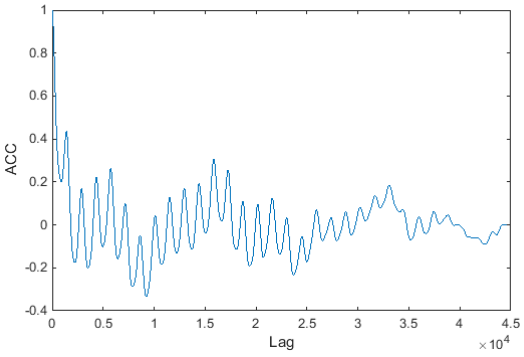
(a) March



(b) June



(c) September

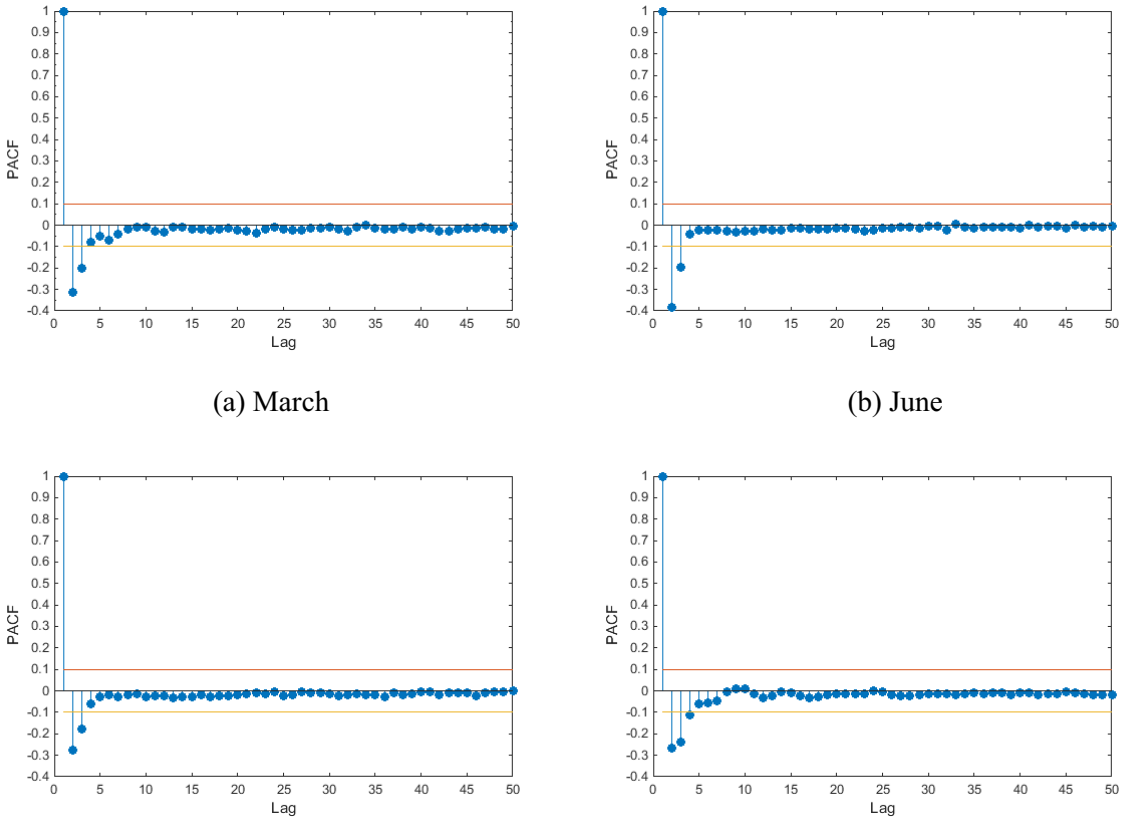


(d) December

Fig. 2: Auto-correlation function of Temperature

Clearly, the data exhibits an autoregressive nature. To further reinforce our inference, we plot its partial auto-correlation function in Fig. 3. The PACF indicates that temperature at time is highly correlated with that at time and moderately correlated to that at and . The remaining past values can be ignored in this case as they have insignificant influence. In Fig. 3, the lines corresponding to PACF = 0.1 and PACF = -0.1 identify the same. Hence, we may conclude that temperature can easily be modelled as an autoregressive process of order 3 or AR(3), as described below:

where is temperature at time , is a constant, and ,, are the coefficients of the AR(3) model. The sequence consists of independent and identically distributed (i.i.d.) random variables, known as residuals (or errors), that give the AR its stochastic nature. The residuals are uncorrelated and they are normally distributed with zero mean and variance .



(a) March (b) June

(c) September (d) December

Fig. 3: Partial auto-correlation function of Temperature

2.2 Overview of the Proposed Framework

In this section, we outline the anomaly detection framework that executes in real time. We divide each day into two successive non-overlapping 12-hr periods, and . Let be the set of all consecutive 12-hr periods in one year (365 consecutive days) preceding :

is the period of inspection for anomalies, and is used to train the anomaly detection framework. Once has been inspected and validated without any anomalies, becomes a part of , so that the new becomes,

The new is then used to validate . If anomalies are discovered in , an alarm is raised and appropriate action is taken immediately. The anomalous data in the new is either rectified, or ignored in case rectification is not possible. For the rest of our discussion, we use the UTS as the training period and the UTS as the inspection period.

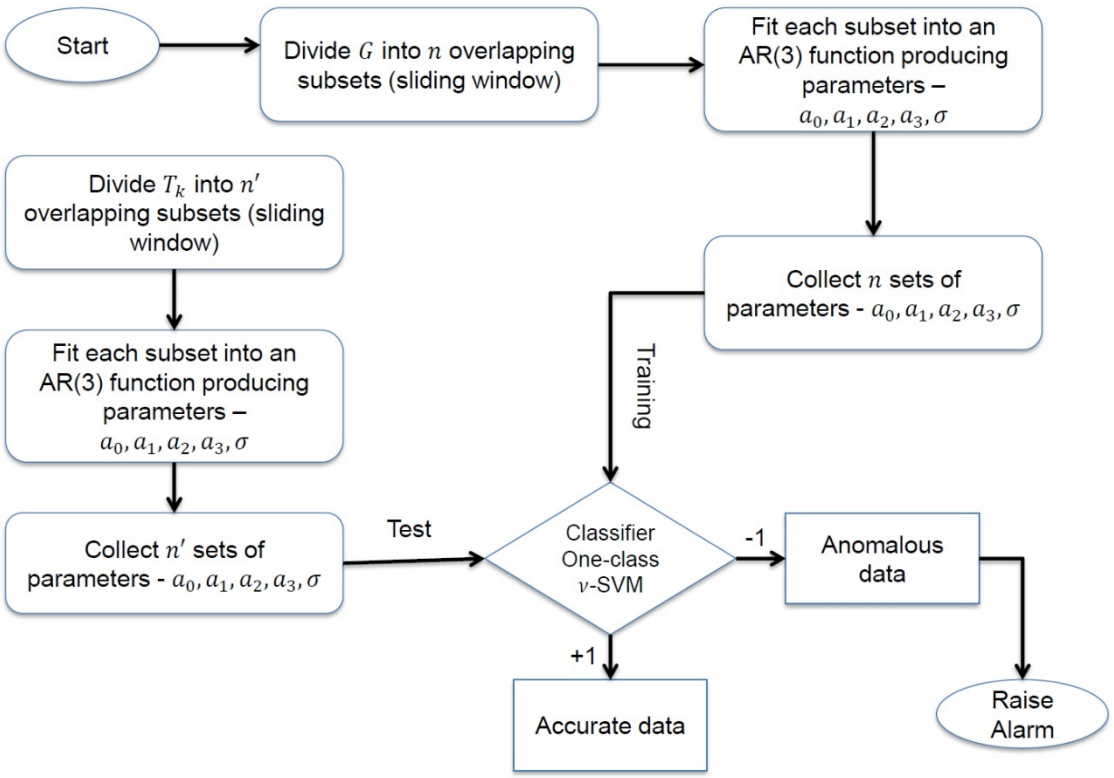


Fig. 4: Broad Overview of the Anomaly Detection Framework

2.3 Generation of the subsets using a sliding window

Fig. 4 provides a graphical overview of the framework. We start by first dividing the UTS into smaller overlapping subsets using the sliding window scheme. As shown in Fig. 5a, each subset has a duration of 6 hours, with each data point being one minute. Suppose signifies the start of . The first subset consists of the data points corresponding to minutes to . Now, we shift the timeframe by 2 hours, so that the second subset cosists of the data from the last 4 hours of the first subset followed by the subsequent 2 hours. That is, the second subset contains the data points from to , the third subset contains the data points from to , and so on. In this manner, is divided into smaller overlapping subsets, where each subset overlaps with the last 4 hours of the preceeding one. We fit a separate separate AR(3) model to each subset and the respective parameters are recorded, thus producing separate sets of five parameters .

As illustrated in Fig. 5b, the UTS is also divided into smaller overlapping subsets in a similar manner. However, unilke in , the first subset consists of the first 2 hours of and the preceeding 4 hours, i.e., the last 4 hours of . It spans over the minutes to . Now we proceed as in G, so that the second subset spans over the minutes to , the third over the minutes to , and so on. Let be the total number of subsets. Each subset is again fitted into separate AR(3) functions, thus generating separate sets of parameters .

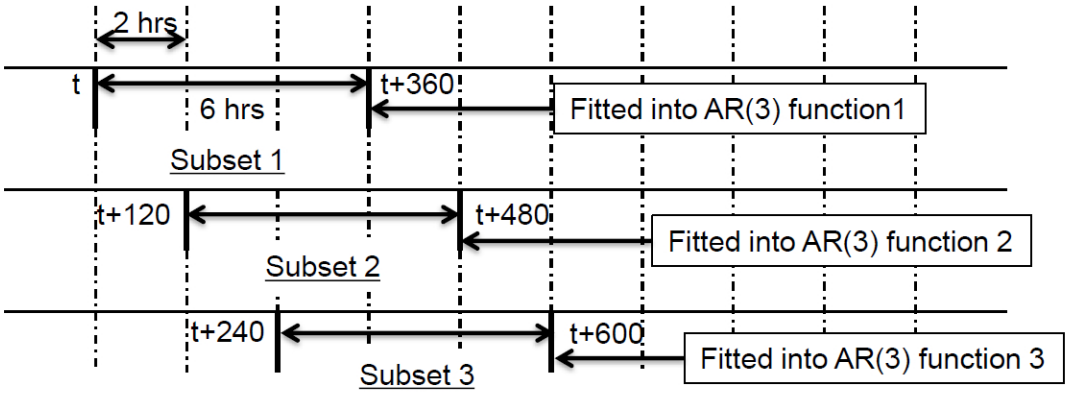


Fig. 5a: is divided into overlapping subsets

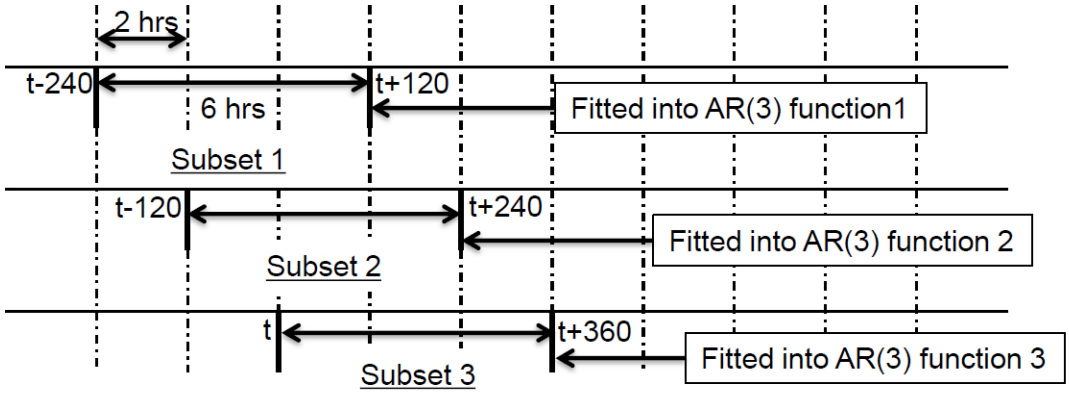


Fig. 5b: is divided into overlapping subsets

As shown in Fig. 4, the next step is to classify the anomalous data in accordingly. However, temperature is a continuous random variable, varying with time, rendering the raw data points unsuitable for classification. Therefore, we need to define a feature space, which can be utilized as an input to the classification problem.

2.4 The Feature Space

Through Fig. 6, we visualize the sets of parameters generated from a sample , in order to see how the parameters are clustered. As it is impossible to display the five parameters in a single plot, we reduce the original space to a new space through principal component analysis. Fig. 6 presents a 3D plot of the principal components. All other experiments were conducted on the original space.

Theoretically, the entire dataset should be representable by a single fixed model, where do not change with time. Anything else can be classified as anomalous. Had the data followed an AR(3) process strictly, we would have ended up with sets of identical parameters and the scatterplot in Fig. 6 would consist of a

single point. However, practical datasets suffer from different types of observational and environmental errors. Some errors may also creep in during various estimations that we need to perform on the data. Again, since we train a separate AR(3) function on each subset in , which are of 6 hours each, local trends or seasonality may influence the estimates slightly. Nevertheless, we observe that the scatterplot in Fig. 6 is very dense, indicating that the AR(3) models are not very different. It also indicates that the sets of parameters can be easily clustered and the probability that anomalies get classified accordingly is high. Hence, the 5-tuple data set of observations forms our feature space that we will use for classification.

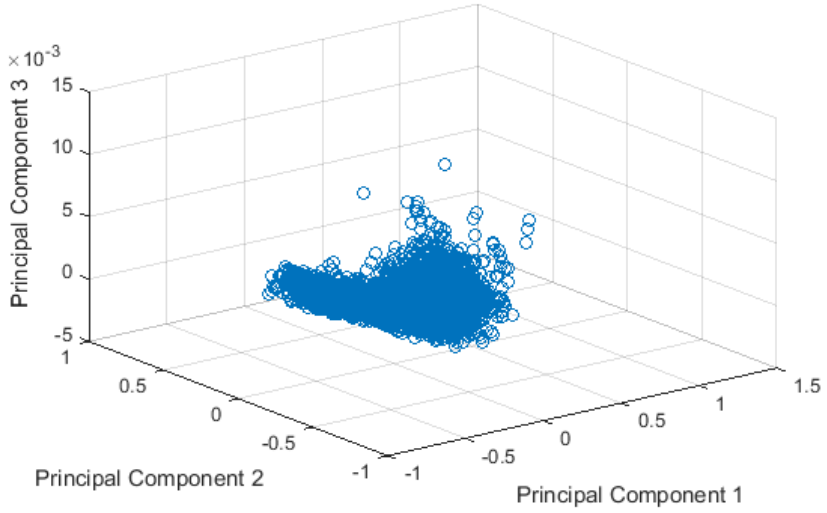


Fig. 6: Scatterplot of the three principal components

2.5 Classification Based on One-class -SVM

As shown in Fig. 4, a Support Vector Machine (SVM) is first trained on the sets of the 5-tuple feature vector from . It then classifies each of the sets of feature vector from as accurate (classified as +1) or anomalous (classified as -1). We use LIBSVM [25] for our experiments.

We use a -SVM [26], with only one class [27] representing all accurate feature vectors. A -SVM involves a regularisation parameter , which helps implement a penalty on the misclassifications committed while separating the classes. is always between and represents the upper bound on the fraction of training errors and lower bound on the fraction of support vectors in the trained model. The kernel of the -SVM is the radial basis function (RBF): , where is the squared Euclidean distance between the two feature vectors, and , and is a parameter of the RBF function. is used to tune the kernel.

Both and can be used to tweak the trained SVM for higher accuracy. It is necessary to find suitable (,) pairs. is varied between and and between and . For each (,) pair, a 10-fold cross-validation is performed on the sample and the corresponding accuracy is recorded. We must keep in my mind that the framework does not classify each raw data point as accurate or inaccurate. Rather each subset is evaluated and classified accordingly. The accuracy of classification is the ratio of the number of correct classifications divided by the total number of classifications. Since training data consists of only accurate values, the correct classification of each subset is and indicates a misclassification. The results are

presented in Fig. 7. High accuracy is obtained for a wide range of and values. The highest accuracy obtained is 99.93%, corresponding to the (,) pairs in TABLE I. These were selected for the remaining experiments. All other (,) pairs are ignored.

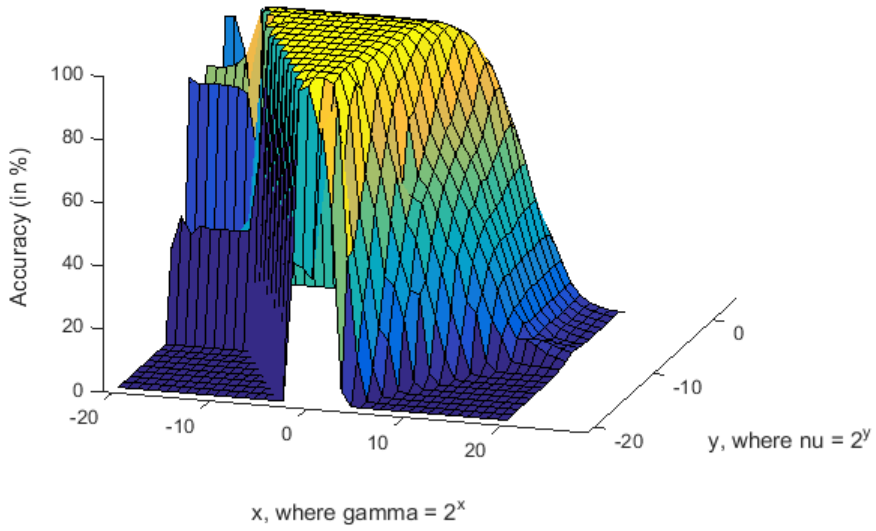


Fig. 7: 3D surface plot of the accuracy for different (,) pairs

TABLE I: (,) pairs with the highest accuracy

		99.93%	99.93%		99.93%		99.93%
		99.93%	99.93%	99.93%	99.93%	99.93%	
		99.93%	99.93%	99.93%	99.93%		
		99.93%		99.93%			

3. Experiments and Results

Experiments were conducted on the data available for 2011-2018. Two kinds of experiments were carried out, namely, a) on accurate data, with only accurate values, and b) on anomalous data, which may consist of single anomalies or a mixture of anomalies introduced to the accurate data.

3.1 Experiments on Accurate Data

These experiments were performed on real temperature values, recorded by the climate office. As described in section 2.1, missing and bad data are reconstructed using interpolation. This ensures that the resulting data set contains only accurate values. In these experiments, we measure how accurately the anomaly detection framework classifies accurate data as , while indicates misclassification. As can be seen in TABLE II, a high accuracy is achieved.

TABLE II: Accuracy of classification of accurate data

		99.98%	99.98%		99.98%		99.98%
		99.98%	99.98%	99.98%	99.98%	99.98%	
		99.98%	99.98%	99.98%	99.98%		
		99.98%		99.98%			

3.2 Experiments on Anomalous Data

For these experiments, we created anomalous data artificially by introducing errors in accurate data. These errors are generated using mathematical error models. The decision to synthesize anomalous data rather than use those available naturally was due to several reasons. First of all, not a lot of anomalous data is available to us. Also, we wished to study the performance of the framework for various types of errors, and prior knowledge of the error models help us understand the results better. Finally, since we have used mathematical models for error generation we were able to tweak the parameters of the error models and evaluate the extent till which the framework performs efficiently.

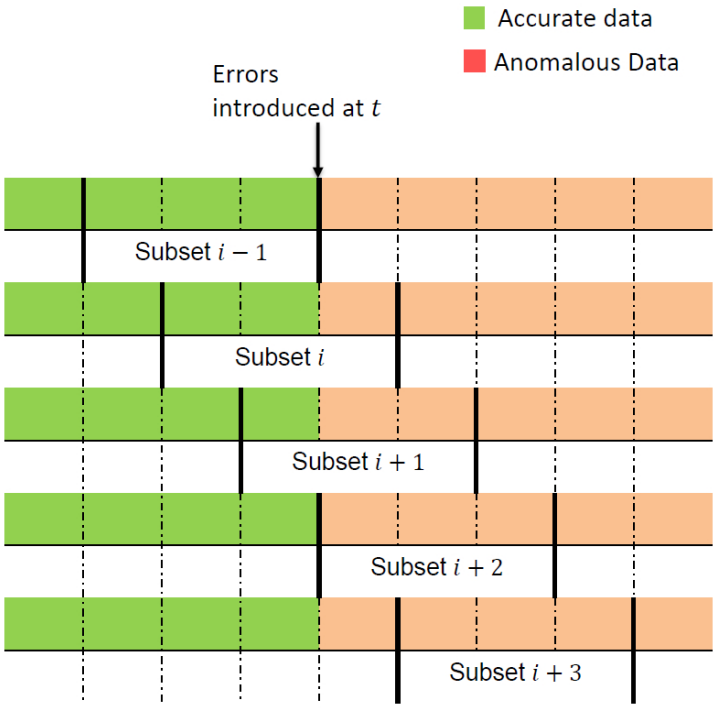


Fig. 8: Generating anomalous data by introducing errors in accurate data

Fig. 8 illustrates how errors are introduced into accurate data at time , so that the rest of the data is anomalous. Subset , created with data points before , contains only accurate data. However, anomalies start creeping into the subsequent subsets. Subset has a third of its data anomalous; subset has two-thirds of its data anomalous; while subset is entirely anomalous and so is each succeeding subset. We introduced errors only at the start of any subset. Errors can be introduced anywhere within the subset producing similar results.

A subset is considered anomalous if it is classified as In the example shown in Fig. 8, subset with only accurate data should be ideally classified as , while the sebsequent subsets as since anomalies are present in them. Therefore, theoretically, the earliest detection of an anomaly is possible at , when subset is classified as . However, practically, the framework may or may not be able to correctly classify anomalous subsets right away. Detections may happen later. Nevertheless, anomalies should ideally be detected onwards, since the entire subset is anomalous. Table III demonstrates the classification results produced by the framework in certain scenarios of anomaly detection with respect to the example in Fig. 8.

TABLE III: Few scenarios of anomaly detection (Reference: Fig. 8)

Class of Subset				
				Detection by
				Detection by
				Detection by

The following kinds of error models were investigated: a) Gaussian errors, b) Exponential growth and decay, c) Linear growth and decay, d) Exponential errors, and e) Linear errors.

3.2.1 Gaussian Errors

In the first set of experiments, Gaussian error is introduced in the accurate data. Gaussian error reflects white noise and is defined as:

where is Gaussian error, and and are the mean and standard deviation of the Gaussian distribution. For our experiments, we have used the standard Gaussian distribution () to reflect white noise. Also,

where and are respectively, anomalous data generated and accurate data extracted at time .

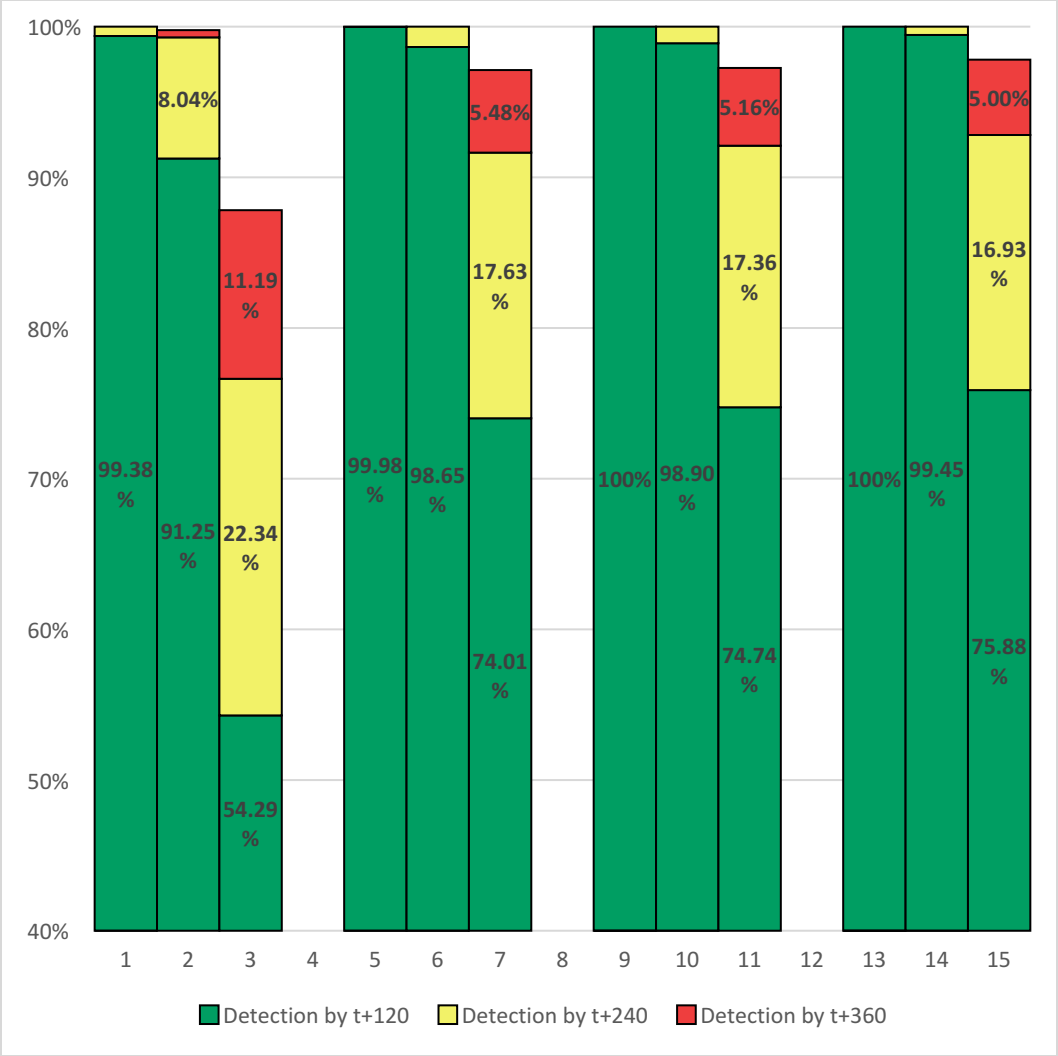


Fig. 9: Percentage of anomalies detected in case of Gaussian errors

Fig. 9 demonstrates how many of the anomalies are detected and how soon. We experimented with all the (,) pairs in Table I, but for presentation purposes we only provide the results for four pairs. The results for the other pairs are similar. We varied from 1.0 to 0.2. For each value of , we show how many of the anomalies are detected by , , and . Obviously, we need to detect anomalies as soon as possible. As evident in Fig. 9, for higher , large errors are introduced and the framework can easily detect most of them by . However, as decreases, producing smaller errors, more time is required.

The performance of the framework deteriorates considerably as goes below 0.2. These results are not documented here. This is expected as such low standard deviations produce very small errors so that the anomalous data greatly resembles the accurate data, making it almost impossible to detect. Results for experiments with are not documented either as high standard deviations generate large errors which are easily detectible.

3.2.2 Exponential Growth and Decay

In this set of experiments, we investigated anomalies due to exponential growth or exponential decay of accurate data. The exponential growth or decay may be described as,

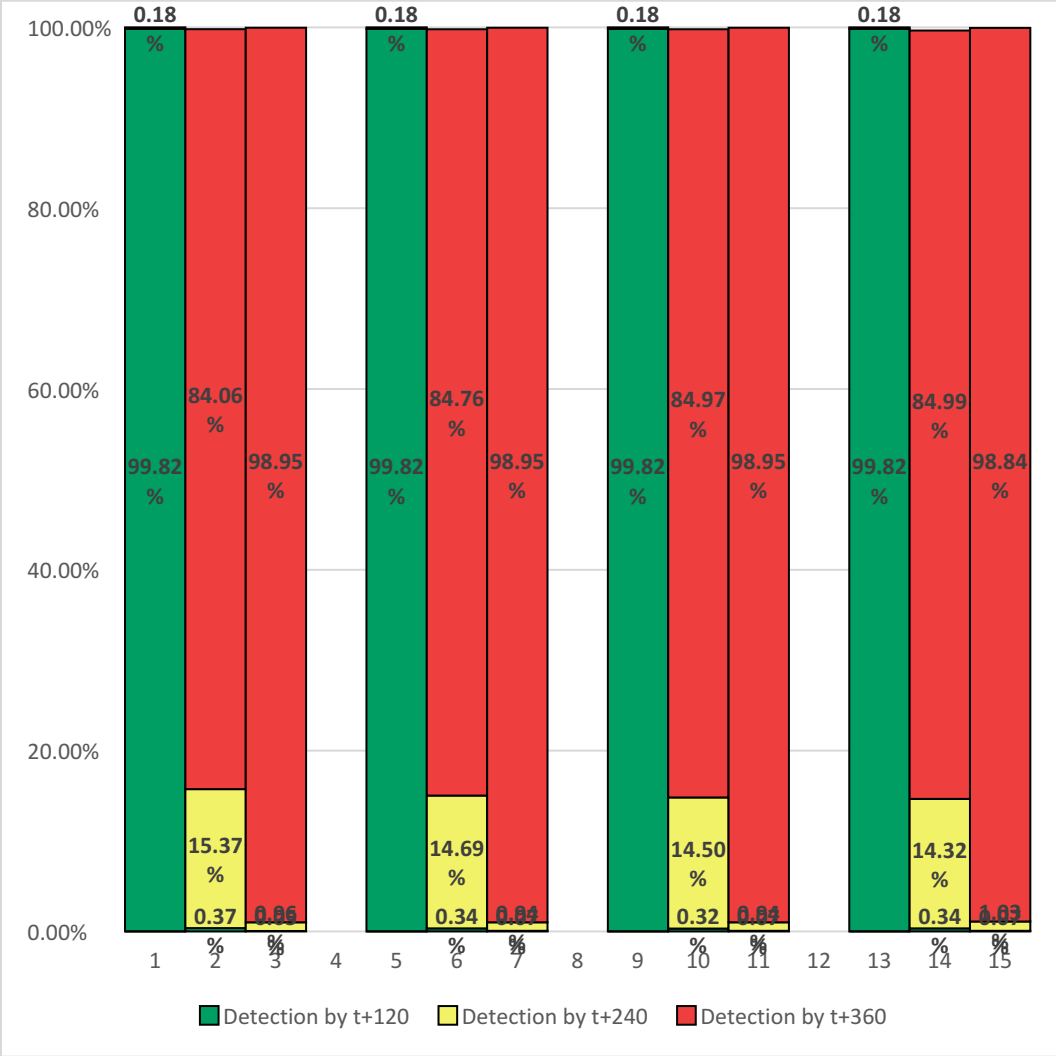


Fig. 10: Percentage of anomalies detected in case of Exponential growth

where is the anomalous data generated at time , is the last known accurate data, and is the rate of exponential growth/decay. If is positive, we have exponential growth and if negative, exponential decay.

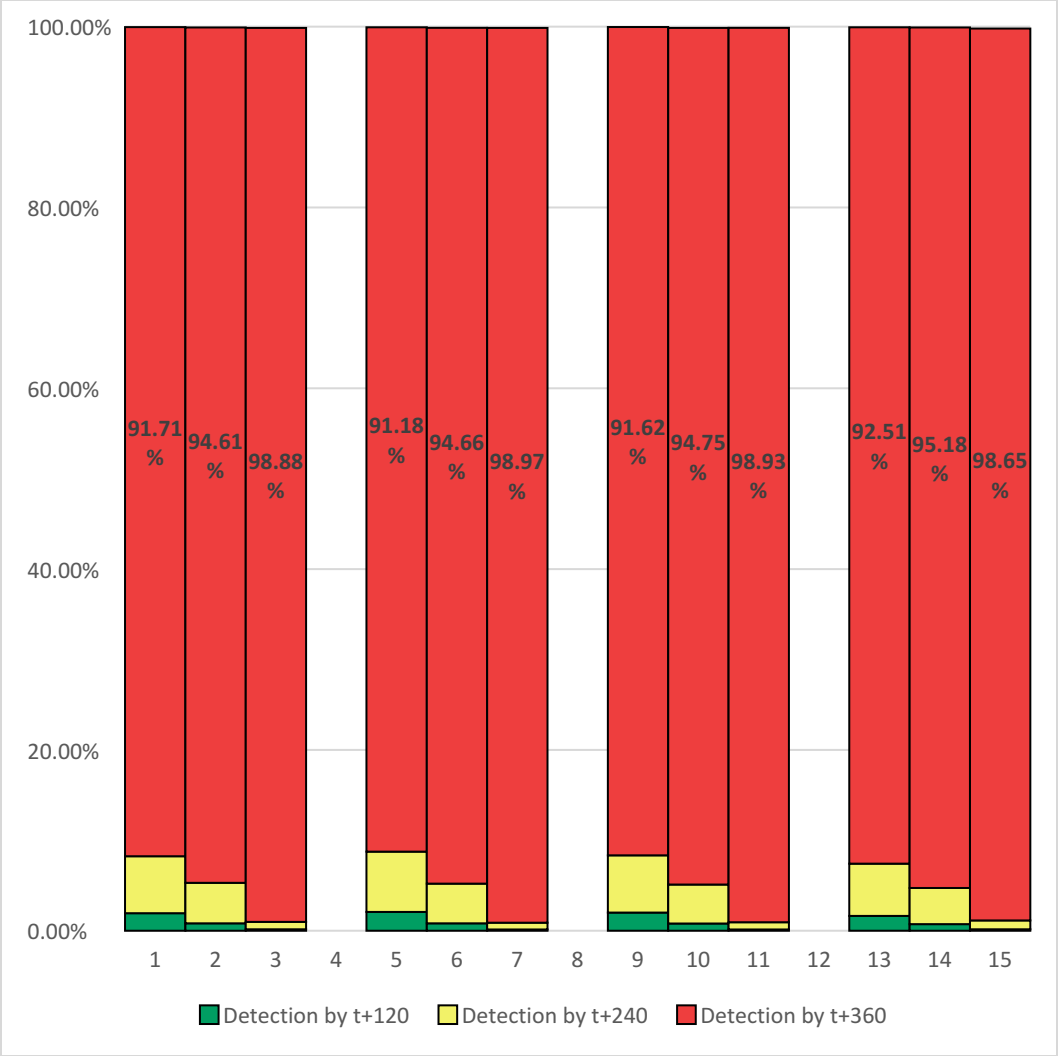


Fig. 11: Percentage of anomalies detected in case of Exponential decay

We varied from 0.1 to 0.001 in case of exponential growth as in Fig. 10 and from -0.1 to -0.01 in case of exponential decay as in Fig. 11. Both in cases of exponential growth and decay, as increases, deviation between anomalous and accurate data increases. Therefore, for higher , detections happen faster. Also, since the very nature of the data changes – autoregressive (accurate) to exponential (anomalous), each and every anomaly gets detected sooner or later.

We experimented with values of beyond the given ranges for both exponential growth and decay. They are not documented here as they follow similar trends.

3.2.3 Linear Growth and Decay

In the third set of experiments, anomalous data was created from linear growth or decay of accurate data, defined as,

where is the anomalous data generated at time , is the last known accurate data, and is the rate of linear growth or decay. If is positive, we get linear growth and if r negative, linear decay.

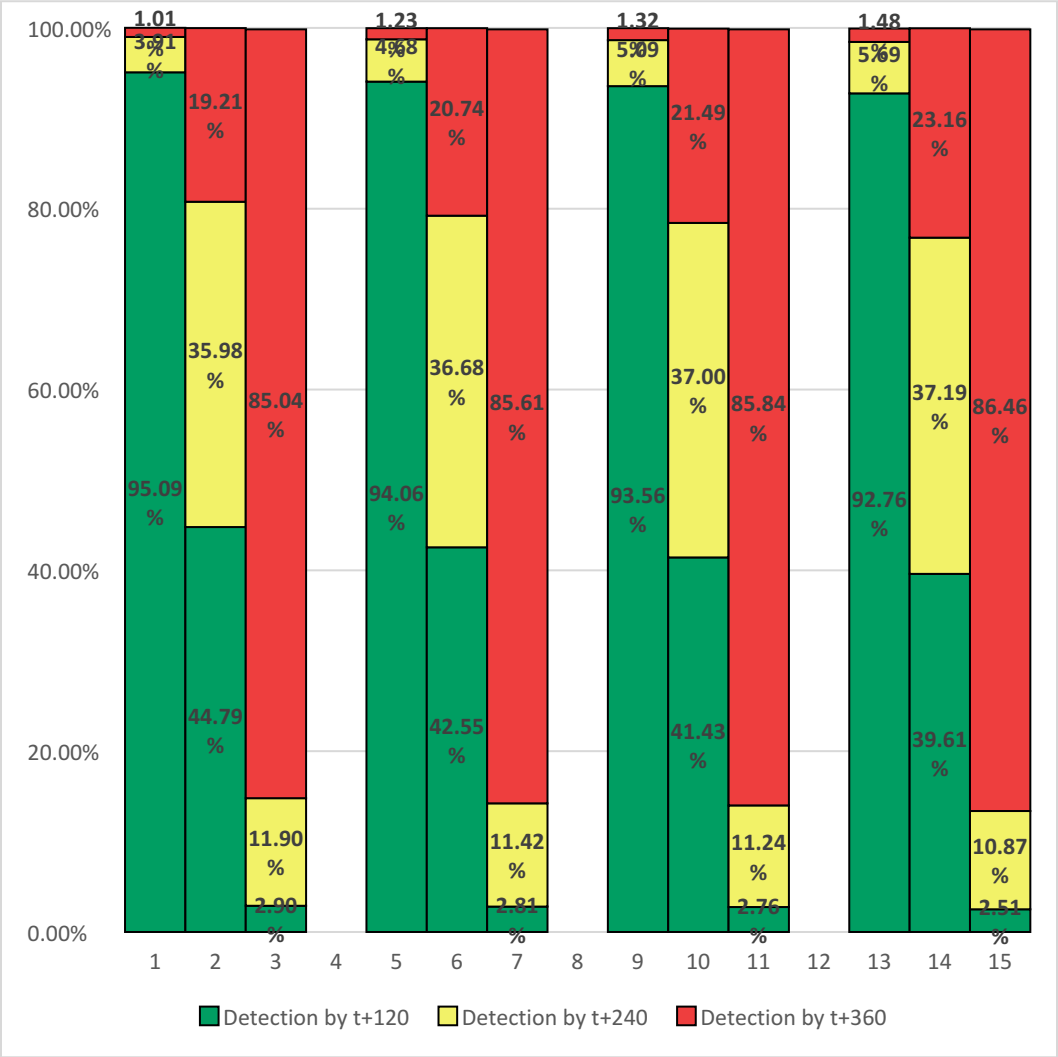


Fig. 12: Percentage of anomalies detected in case of Linear growth

We varied from 3.0 to 1.0 in case of linear growth as in Fig. 12 and from -3.0 to -1.0 in case of linear decay as in Fig. 13. For both linear growth and linear decay, as increases, deviation between anomalous and accurate data increases. Consequently, for higher detections happen faster. As in exponential growth/decay, since the very nature of the data changes – autoregressive to linear, all anomalies get detected sooner or later.

Experiments were also carried out for values of beyond the specified ranges. The results follow similar trends.

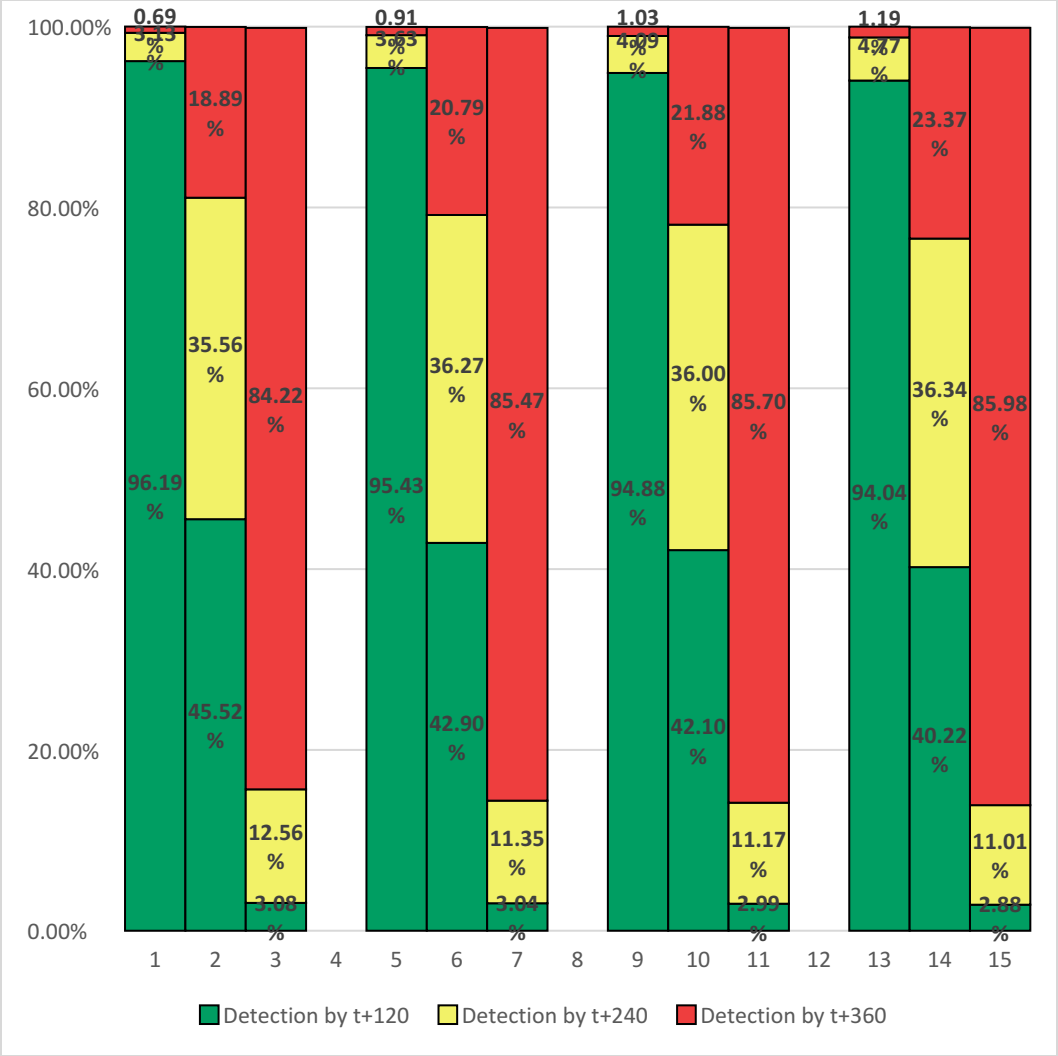


Fig. 13: Percentage of anomalies detected in case of Linear decay

3.2.4 Exponential Errors

Experients were carried out with anomalous data created by introducing exponential error in accurate data. The exponential error may be defined as,

where is error at time , is a constant, and is the rate of exponential error. is always positive, representing exponential growth in consecutive errors. An exponential decay in consecutive errors does not make sense in this scenario. For our experiments, we have chosen , as the set of results produced is convenient for us to be able to express our insights. Also,

where and are respectively, anomalous data generated and accurate data extracted at time . Here, we have documented two sets of results, corresponding to , where an exponential error is added to accurate data and , where an exponential error is subtracted from accurate data.

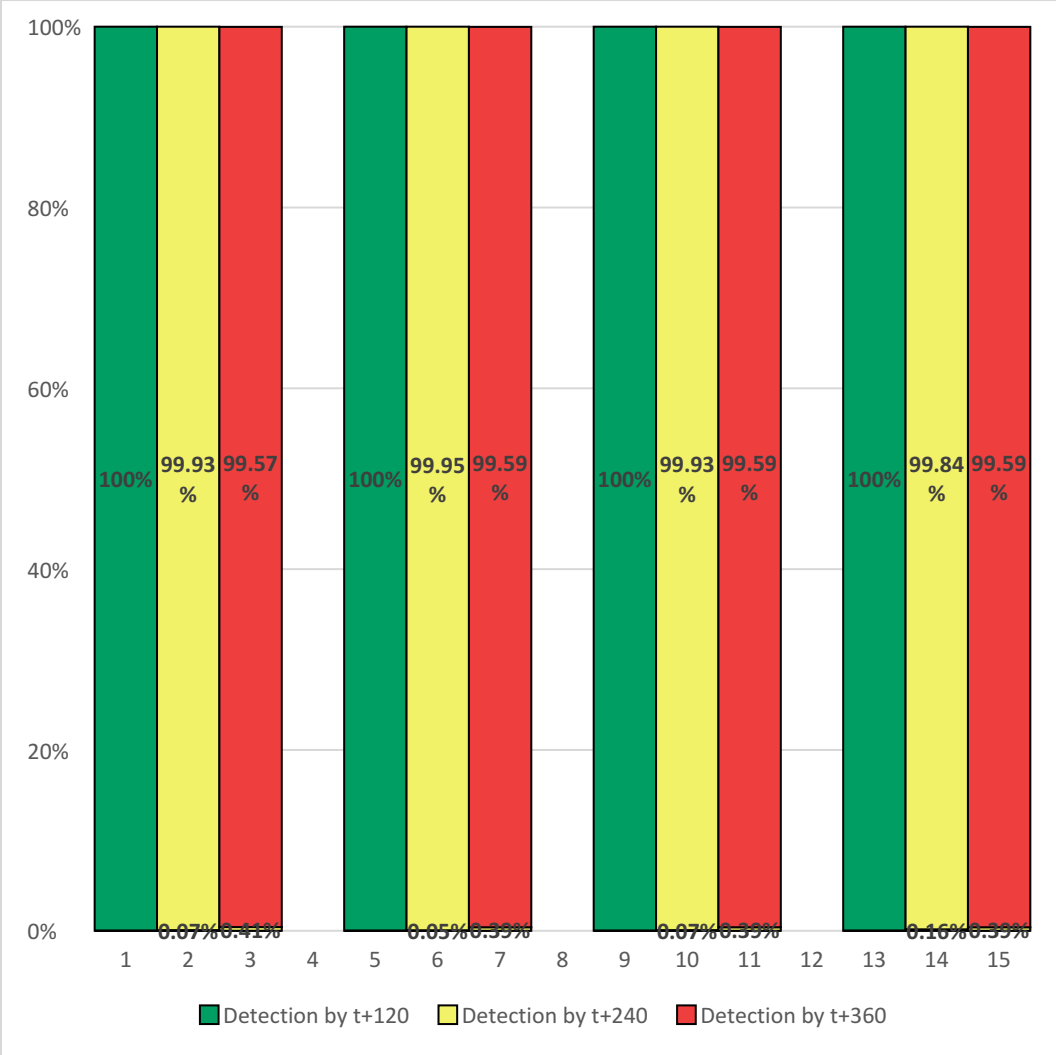


Fig. 14: Percentage of anomalies detected in case of Exponential errors (additive)

In our experiments, we varied from 0.1 to 0.03 both for additive and subtractive exponential errors as shown in Figs. 14 and 15 respectively. In both cases, as increases, larger errors are generated. Consequently, for higher detections happen faster.

Experiments for both additive and subtractive exponential errors were performed for values of and similar results were obtained.

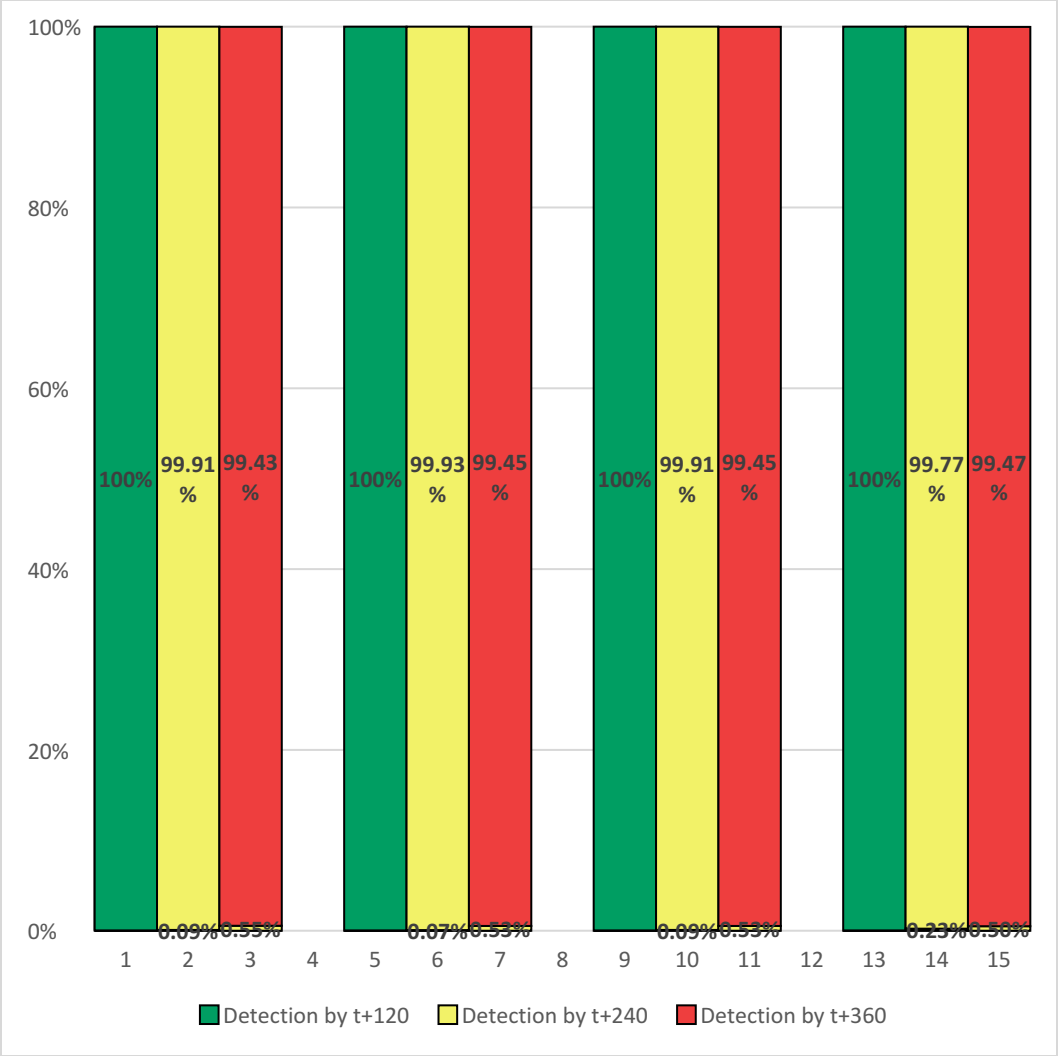


Fig. 15: Percentage of anomalies detected in case of Exponential errors (subtractive)

On the other hand, performance decreases considerably as decreases below 0.03, and those results are not documented here. When we say performance decreses considerably, we mean most of the anomalies are not detected by . They may be detected later. For example, when =0.02, first detection happens at . This is expected as at such a low very small errors are created, so that the anomalous data greatly resembles accurate data, making it almost impossible to detect. We can explain this issue as a tug of war between autoregressive nature of accurate data and exponential nature of errors. As long as the exponential nature supersedes the autoregressive nature significantly, we can expect fast detection of anomalies.

3.2.5 Linear Errors

In the last set of experiments, anomalous data is created by introducing linear error in accurate data. Linear error may be defined as,

where is the error generated at time , is a constant, and is the rate of linear error. As in exponential errors, is always positive as a linear decay in consecutive errors does not make sense here. We also choose the same . Now,

where and are the anomalous data generated and the accurate data extracted at time , respectively. We document two sets of results, corresponding to , where a linear error is added to accurate data and , where a linear error is subtracted from accurate data.

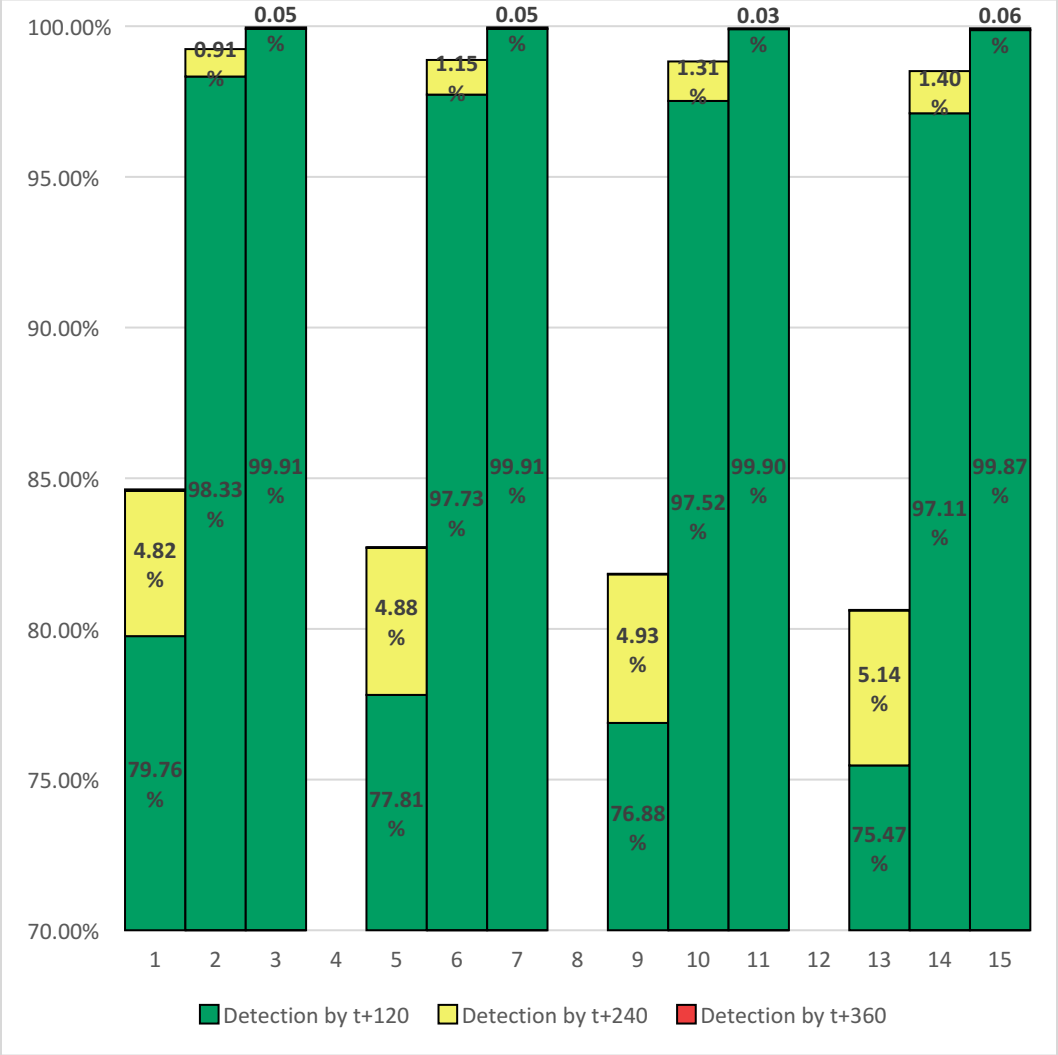


Fig. 16: Percentage of anomalies detected in case of Linear errors (additive)

We varied from 5.0 to 3.0 both for additive and subtractive linear errors as demonstarted in Figs. 16 and 17 respectively. In both cases, as increases, larger errors are generated, leading to faster detections.

Experiments for both additive and subtractive linear errors were performed for values of r > 5.0. They are not documented as they exhibit similar trends. Performance deteriorates considerably as decreases below 3.0, and those results are not documented as well. As for exponential errors, there exists a tug of war

between autoregressive nature of accurate data and linear nature of errors. As long as the linear nature dominates over the autoregressive nature, we can expect fast detection of anomalies.

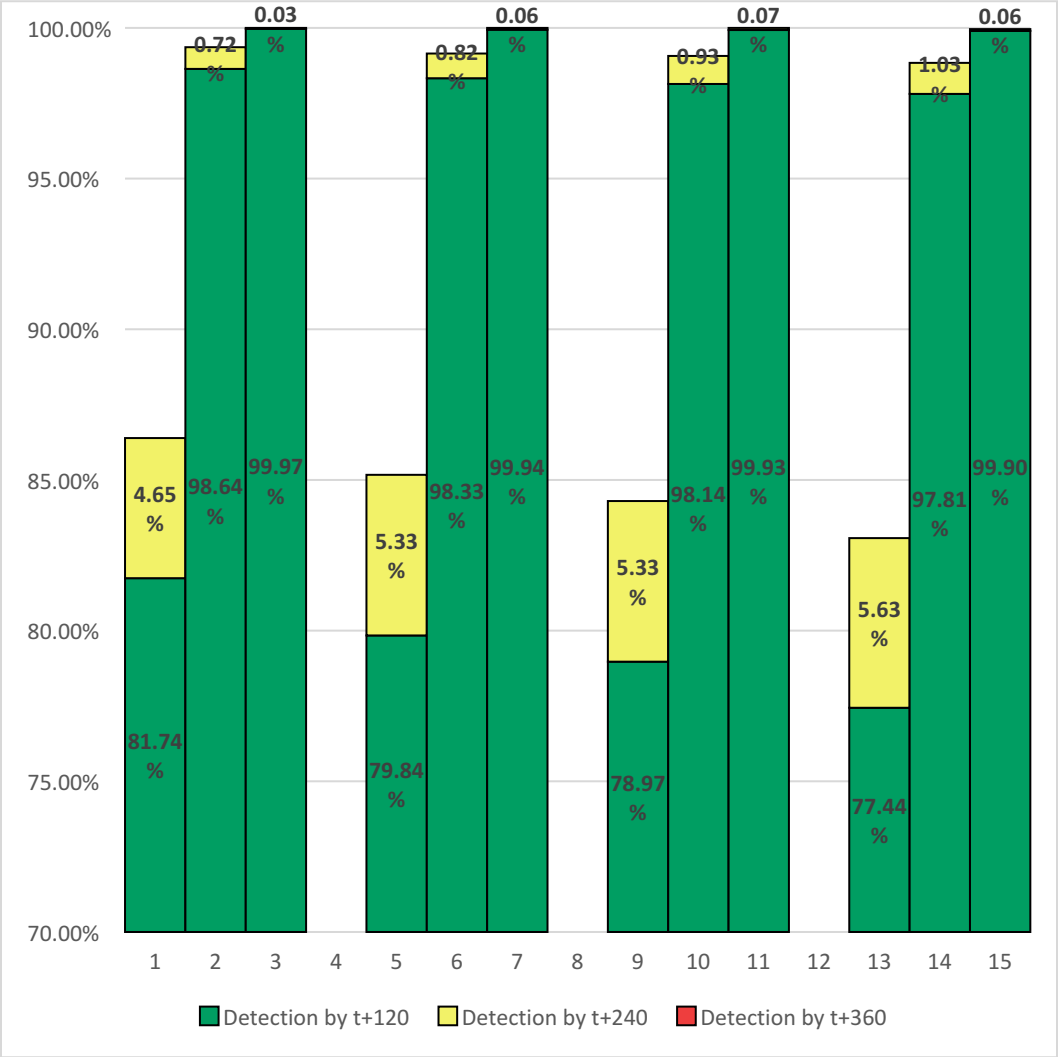


Fig. 17: Percentage of anomalies detected in case of Linear errors (subtractive)

The framework faces an issue in detecting anomalies in case of linear errors, especially for lower values of . In all other experimets, once an anomaly is detected, that is once a subset is classified as , the succeeding subsets get classified as as well. The same is not observed in case of linear errors. Even if an error is detected at some point, there is no guarantee the succeeding subsets get classified as . For example, if a linear error is introduced at time , classifications at, , and may be , , respectively.

4. Conclusion

In this paper, we have proposed an automated machine-learning based anomaly detection framework for identification of a flawed temperature sensor in real-time. We have focused on a purely temporal approach, where the UTS generated by a single temperature sensor is used in monitoring its robust operation. The framework first divides the UTS into overlapping subsequences through a sliding window.

It then models each subsequence stochastically as an AR(3) model, and finally mines the AR(3) parameters with a One-class -SVM. The OC-SVM forms a tight boundary around the normal class and any outlier is regarded as an anomaly. We have evaluated our framework on the temperature data collected by the State Climate Office of North Carolina [24]. Experiments were conducted with both accurate and anomalous data. However, since anomalies are scarce in the available dataset, we synthesized anomalous data by introducing errors in the available temperature data. Our proposed framework exhibits a very good peformance in both cases. The implication of our work on real-time fault detection in temperature sensors will be essential for pattern recognitions in drastic weather and climate change. However, consideration of other weather variables such as air pressure, humidity, and wind velocity will be necessary. Due to their complex and unique stochastic behavior, each of them will require very specific modeling that may go beyond standard regression techniques.

References

[1] Parvaneh Asghari, Amir Masoud Rahmani, and Hamid Haj Seyyed Javadi. 2019. Internet of Things applications: A systematic review. Computer Networks, Volume 148, Pages 241-261.

[2] A. R. Ganguly, and K. Steinhaeuser. 2008. Data Mining for Climate Change and Impacts. 2008 IEEE International Conference on Data Mining Workshops, Pisa, pp. 385-394.

[3] Hossein Hassani, Xu Huang, and Emmanuel Silva. 2019. Big Data and Climate Change. Big Data and Cognitive Computing, Volume 3(1), Article 12.

[4] The IoT and climate change. *The Economist*. Retrieved from https://empoweringspaces.economist.com/the-iot-and-climate-change/

[5] Shashi Shekhar, Zhe Jiang, Reem Y. Ali, Emre Eftelioglu, Xun Tang, Venkata M. V. Gunturi, and Xun Zhou. 2015. Spatiotemporal Data Mining: A Computational Perspective. ISPRS International Journal of Geo-Information, Volume 4(4), Pages 2306-2338.

[6] Gowtham Atluri, Anuj Karpatne, and Vipin Kumar. 2018. Spatio-Temporal Data Mining: A Survey of Problems and Methods. ACM Comput. Surv., Volume 51(4), Article 83.

[7] Zhicheng Shi, and Lilian S.C. Pun-Cheng. 2019. Spatiotemporal Data Clustering: A Survey of Methods. ISPRS International Journal of Geo-Information, Volume 8(3), Article 112.

[8] Manish Gupta, Jing Gao, Charu Aggarwal, and Jiawei Han. 2014. Outlier Detection for Temporal Data. Synthesis Lectures on Data Mining and Knowledge Discovery, Volume 5(1), Pages 1-129.

[9] U. S. Shanthamallu, A. Spanias, C. Tepedelenlioglu, and M. Stanley. 2017. A brief survey of machine learning methods and their sensor and IoT applications. 2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA), Larnaca, pp. 1-8.

[10] Dina ElMenshawy, and W. Helmy. 2018. Detection techniques of data anomalies in IoT: A literature survey. International Journal of Civil Engineering and Technology, Volume 9, Pages 794-807.

[11] J. Ma, and S. Perkins. 2003. Online novelty detection on temporal sequences. In Proc. 9th ACM Int. Conf KDD, New York, NY, USA, pp. 613–618.

[12] T. Kieu, B. Yang, and C. S. Jensen. 2018. Outlier Detection for Multidimensional Time Series Using Deep Neural Networks. 2018 19th IEEE International Conference on Mobile Data Management (MDM), Aalborg, pp. 125-134.

[13] Bouchra Lamrini, Augustin Gjini, Simon Daudin, François Armando, Pascal Pratmarty, and Louise Travé-Massuyès. 2018. Anomaly Detection Using Similarity-based One-Class SVM for Network Traffic Characterization.

[14] A. Yamaguchi, and T. Nishikawa. 2018. One-Class Learning Time-Series Shapelets. 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, pp. 2365-2372.

[15] B. Jin, Y. Chen, D. Li, K. Poolla, and A. Sangiovanni-Vincentelli. 2019. A One-Class Support Vector Machine Calibration Method for Time Series Change Point Detection. 2019 IEEE International Conference on Prognostics and Health Management (ICPHM), San Francisco, CA, USA, pp. 1-5.

[16] D. J. Hill, and B. S. Minsker. 2010. Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. Environ. Modell. Softw., Volume 25(9), pp. 1014–1022.

[17] Lv Y. 2015. An adaptive real-time outlier detection algorithm based on ARMA model for radar's health monitoring. In Proc. the IEEE Autotestcon, pp. 1-7.

[18] Adam Santos, Eloi Figueiredo, M.F.M. Silva, C.S. Sales, and J.C.W.A. Costa. 2016. Machine learning algorithms for damage detection: Kernel-based approaches. Journal of Sound and Vibration, Volume 363, Pages 584-599.

[19] J. Shi, G. He, and X. Liu. 2018. Anomaly Detection for Key Performance Indicators Through Machine Learning. 2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC), Guiyang, pp. 1-5.

[20] S. Duque Anton, L. Ahrens, D. Fraunholz, and H. D. Schotten. 2018. Time is of the Essence: Machine Learning-Based Intrusion Detection in Industrial Time Series Data. 2018 IEEE International Conference on Data Mining Workshops (ICDMW), Singapore, Singapore, pp. 1-6.

[21] N. V. Chawla, N. Japkowicz, and A. Kotcz. 2004. Editorial: Special Issue on Learning from Imbalanced Data Sets. SIGKDDExplor. Newsl., Volume 6(1), pp. 1–6.

[22] H. He, and E. A. Garcia. 2009. Learning from Imbalanced Data. IEEE Trans. on Knowl. and Data Eng., Volume 21(9), pp.1263–1284.

[23] V. Barnett, and T. Lewis. 1994. Outliers in Statistical data. Wiley.

[24] State Climate Office of North Carolina. Retrieved from https://climate.ncsu.edu/

[25] Chih-Chung Chang, and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. ACM Trans. Intell. Syst. Technol., Volume 2(3), Article 27.

[26] Bernhard Schölkopf, Alex J. Smola, Robert C. Williamson, and Peter L. Bartlett. 2000. New Support Vector Algorithms. Neural Comput., Volume 12(5), Pages 1207-1245.

[27] Bernhard Schölkopf, John C. Platt, John C. Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. 2001. Estimating the Support of a High-Dimensional Distribution. Neural Comput., Volume 13(7), Pages 1443-1471.