Real-Time Anomaly detection for the Secure Water Treatment

No Author Given

No Institute Given

Abstract. Anomaly detection refers to identifying the pat- terns in data that deviate from expected behavior. These non- conforming patterns are often termed as outliers, malwares, anomalies or exceptions in different application domains. The anomaly dedection is done by performing critical comparative analysis using existing approaches, such as SVM, Local Outlier Factor, isolation forest, DBSCAN algorithms. In this report, we had provided information about the Real-Time Anomaly detection for the Secure Water Treatment data-set and the system architecture of the implementation.

Keywords: Anomaly detection · System architecture · Stream Mining.

1 Introduction

Real-time anomaly detection aims to capture abnormalities in system behavior in real time. The abnormalities/anomalies appear in the form of malicious network intrusion, malware infection, over-utilization of system resources due to design defects, etc. Machine learning algorithms have been incorporated into various anomaly detection application areas, such as net- work monitoring, power distribution, IoT sensors devices, healthcare, fraud detection, and transportation traffic anomaly. As such, this paper addresses Real-Time Anomaly detection for the Secure Water Treatment data-set and the system architecture of the implementation. . Accuracy will be improved by hyper-parameter tuning unsupervised machine learning algorithms, which is capable of analyzing data in real-time. Furthermore, data visualization will be presented via Grafa.

The rest of this paper is organized as follows. Section 2 critically overall system architecture and anomaly detection techniques in real time using open source technologies. Section 3 presents experimental setup, data set, features selection, parameter, and problem analysis by evaluating the accuracy rate, performance evaluation, results, and analysis for current anomaly detection approaches. Section 5 evaluates and compares the result with the existing anomaly detection approach. Finally, Section 6 provides concluding remarks.

2 Overall system architecture

This section critically analyzes various clustering algorithms used for real-time anomaly detection. Unsupervised algorithms focus on the data that do not con-

tain any labeling information, and it does not need a separate training and testing phase. Many unsupervised machine learning algorithms have been used for anomaly detection. Unquestionably, clustering helps to classify the patterns into groups and further supports to obtain insight, classification, and compressing of data.

Additionally, this section will show overall system architecture and importance of open source technologies.

2.1 Clustering algorithms for anomaly detection

Isolation trees constitute the isolation forest, where each instance is separated from one another. The tree structures of the learned groups produce anomaly records, this approach evades the computation of costly distance or density measures. One study, isolation forest was used for dimensionality reducing preprocessing step to enhance the interpretability of isolation forest model and improve the performance on anomaly detection.

In the HDBSCAN, the number of clusters is calculated automatically and it is able to handle clusters with different density and shapes. Furthermore, noise and outliers were identified easily.

Support Vector Machine (SVM) — a supervised machine learning algorithm frequently cited and used in classification problems. SVMs use hyperplanes in multi-dimensional space to separate one class of observations from another. Naturally, SVM is used in solving multi-class classification problems. However, SVM is also increasingly being used in one class problem, where all data belong to a single class. In this case, the algorithm is trained to learn what is "normal", so that when a new data is shown the algorithm can identify whether it should belong to the group or not. If not, the new data is labeled as out of ordinary or anomaly.

LOF is an unsupervised (well, semi-supervised) machine learning algorithm that uses the density of data points in the distribution as a key factor to detect outliers. LOF compares the density of any given data point to the density of its neighbors. Since outliers come from low-density areas, the ratio will be higher for anomalous data points. As a rule of thumb, a normal data point has a LOF between 1 and 1.5 whereas anomalous observations will have much higher LOF. The higher the LOF the more likely it is an outlier. If the LOF of point X is 5, it means the average density of X's neighbors is 5 times higher than its local density.

2.2 Open source technology and System architecture

The frameworks we are about to use are Kafka, Spark, Influx DB, and docker to communicate. Kafka provides guaranteed message delivery with proper ordering. This means messages published by a producer to a particular topic partition will be delivered in the order they are sent, and a consumer can subscribe to topics and receive messages. Moreover, Kafka can form a cluster to handle a high

volume of data with low processing latency and server failures without losing messages.

As the stream data comes continuously and is voluminous, we require a scalable distributed framework. To address the scalability issue, we can use a distributed solution like Hadoop, MapReduce, etc. Hadoop runs in batch mode and cannot handle real-time data. As we are looking for real-time data analysis in a distributed framework, Thus we can use Apache Spark which is fault-tolerant and supports distributed real-time computation systems for processing fast, large streams of data.

Influx-DB is used as a data store for any use case involving large amounts of time-stamped data, it works smoothly with Spark-structured streaming to process, store and visualize data in real-time. Data is transported through a Kafka queue continuously as a stream. It is then split into micro batches (DStreams).

The stream comes in a raw format, which needs to be processed (e.g. data normalization and noise elimination) in order to be analyzed. After processing, it is converted to multi-dimensional time-series data. The real-time framework applies machine learning algorithms to analyze stream data. It uses unsupervised clustering algorithms (e.g., k-means, incremental clustering) to build clusters on training data. And after clustering, it predicts data using its cluster information Note: At each micro-batch processing step, the generated immutable data set is stored in memory and the results are propagated to the next micro-batch processing module. As the stream data evolves in nature, the training model should be updated periodically. Finally, the framework sends a request to the resource scheduler to update its resources if it detects any abnormal behavior.

3 Experimental study : problem analysis

In this experiment, the anomaly detection of real-time analytics for the secure water treatment have been evaluated by using four different types of algorithms such as DBSCAN, isolation forest, One class SVM, and Local Outlier Factor. The following subsections will further provide details of the experiment setup.

3.1 Algorithms

To conduct our experiment, we selected four different frequently used anomaly detection algorithms, especially DBSCAN, isolation forest, One class SVM, and Local Outlier Factor (refer to Section 2 for explanation and related work on these algorithms).

3.2 Dataset description

The dataset consists of 13661 samples with 51 attributes which were collected for 6 continuous days. All the data was logged continuously once every second for 136 hours into a Historian server. Data recorded in the Historian was obtained from the sensors and actuators of the testbed. The dataset describes the physical

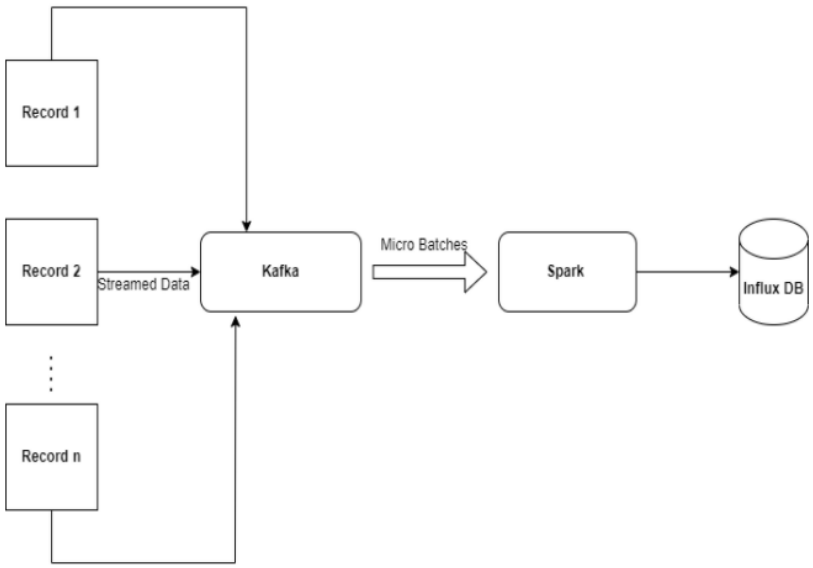

Fig. 1. System architecture

properties of the testbed in operational mode. We infer that the dataset consists of 21 int type, 21 float type, and 3 object type attributes, where 3 object type attributes are converted to float.

3.3 Data preprocessing

The columns with one specific value are checked in the dataset and we find there are 12 attributes with one value. This type of attribute doesn't affect our final outcome. Moreover, it is good to reduce the dimensionality for unsupervised learning techniques. The attributes namely AIT401, P102, P202, P204, P206, P302, P402, P403, P404, P502, P601, P603 are dropped as they are the attributes with one specific value.

We checked with the categorical variables in the dataset and found that there are 15 categorical attributes and the rest are numerical data. Thus standardization takes place over numerical data then Principal Component Analysis(PCA) introduced to reduce the dimensionality of the dataset and on other hand one-hot encoding is applied to categorical data. Finally the preprocessed numerical and categorical data are combined for further analysis.

3.4 Anomaly detection with algorithms

We apply 4 algorithms for anomaly detection and also hyper-parameter tuning is used for developing of accuracy. 1. Local Outlier Factor: This algorithm takes

two inputs, which are the k value and data points. We define two methods that calculate the k distance and the reachability distance. For each data point, we store two variables named KNN and lrd, which calls the methods kDistance and reachabilityDist, respectively. Furthermore, for each point in KNN, we hold a temporary sum of the lrd and get the max of the lof variable and the temporary lof variable. Finally, we return the lof data.

Algorithm 2 Local Outlier Factor

```
1:  INPUT: k, dp
2:  OUTPUT: lof values
3:  kDistance (dp, pt)
4:  reachabilityDist(pt)
5:  lof = null
6:  for each data point pt do
7:      KNN = kDistance(dp, k)
8:      lrd = reachabilityDist(KNN, k)
9:      for each p in KNN do
10:         lofhold[i]=sum(lrd[o ∈ N(p)])/lrd[i]/|N(p)|
11:         lof = max(lof, lofhold)
12:     end for
13: end forreturn lof
```

Fig. 2. Local Outlier Factor

2. DBSCAN In that algorithm, we also apply hyper parameter tuning process for finding optimal anomaly detection. In DBSCAN, epsilon is the important parameter. So, from 1 to 5 by 0.5 epsilon range values show average silhoutte score for each value. We get maximum average silhoutte score when epsilon is equal to 5.

3.Isolation Forest For the streaming data set Zi, the initial anomaly detection model is build, at first, we build the iTrees in terms of the bootstrap sampling from the Zi . The ensemble detection model E is composed of L iTrees, namely, E=E1,E2,...EL,which is built from the data in i-th sliding window. The detail algorithms and evaluation of iForest can be seen in [Liu et al., 2008], here, we only describe the main part of iForest algorithm.

4. One Class SVM The approach is based on , where the authors attempted to make the supervised SVM more robust in case of existing outliers in the training data. The key idea is the minimization of the of the Mean Square Error (MSE) for tackling outliers using the center of class as an averaged information. The conducted experiments showed that the generalization performance improved and the num- ber of support vector decreased compared to the traditional SVM.

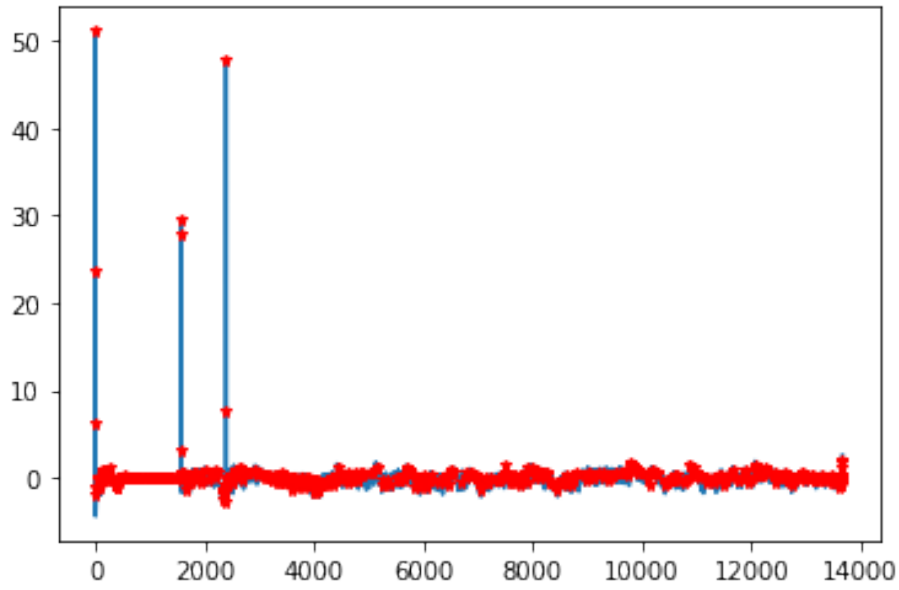

Fig. 3. Local Outlier Factor result

Algorithm 18.1: Density-based Clustering Algorithm

dbscan **(**$\mathcal{D}$**,** ϵ**,** *minpts***)** :
foreach $x \in \mathcal{D}$ **do**
    Compute $N_\epsilon(x)$
    Classify x as core, border, or noise
$id = 0$
foreach $x \in \mathcal{D}$, *such that* x *is core and unmarked* **do**
    $id = id + 1$
    DensityConnected(x, id)
return *Clustering* $\{\mathcal{D}_i\}_{i=1}^{id}$, *where* $\mathcal{D}_i = \{x \in \mathcal{D} : x \text{ has label } i\}$

DensityConnected (x, id):
Mark x with current cluster id
foreach $y \in N_\epsilon(x)$ **do**
    Mark y with current cluster id
    if y *is core* **then**
        DensityConnected(y, id)

Fig. 4. DBSCAN Algorithms

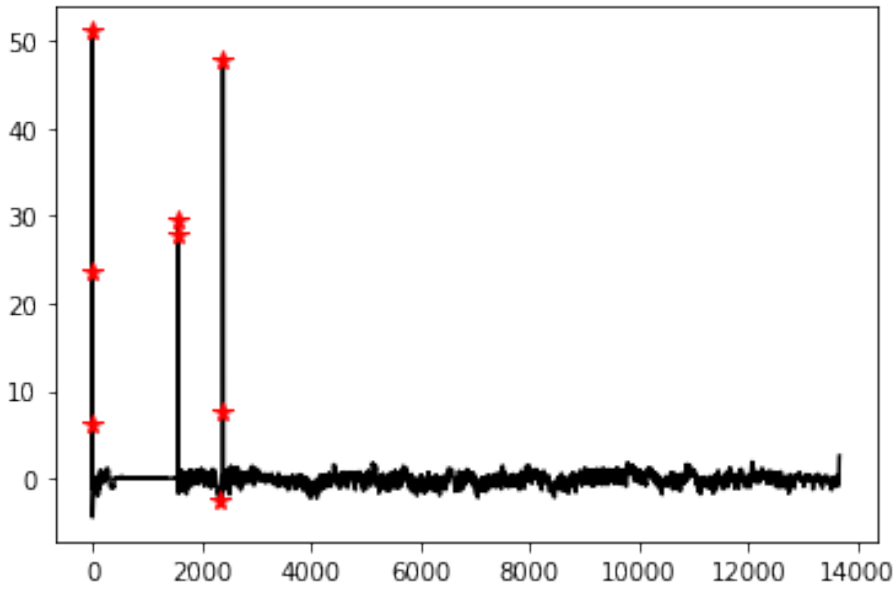

Fig. 5. DBSCAN Algorithms

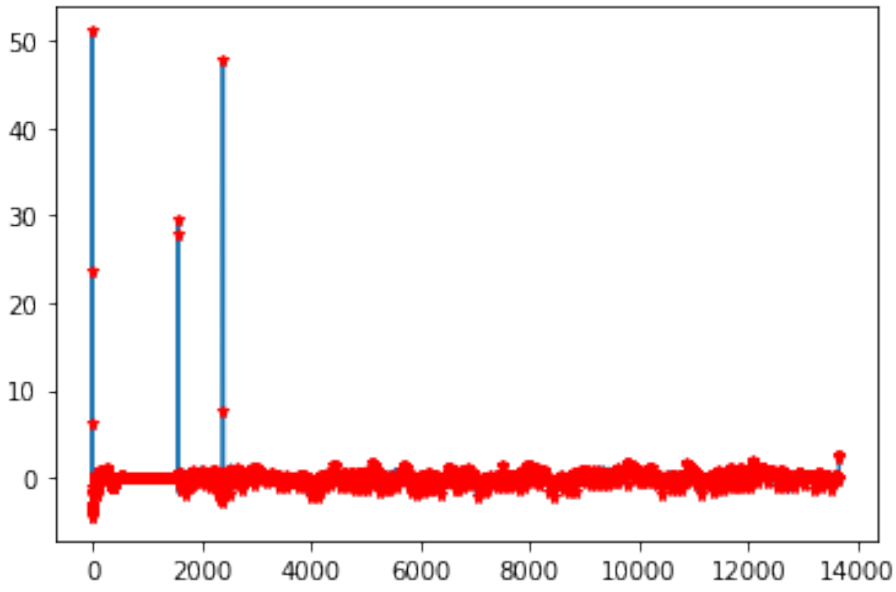

Fig. 6. Isolation Forest

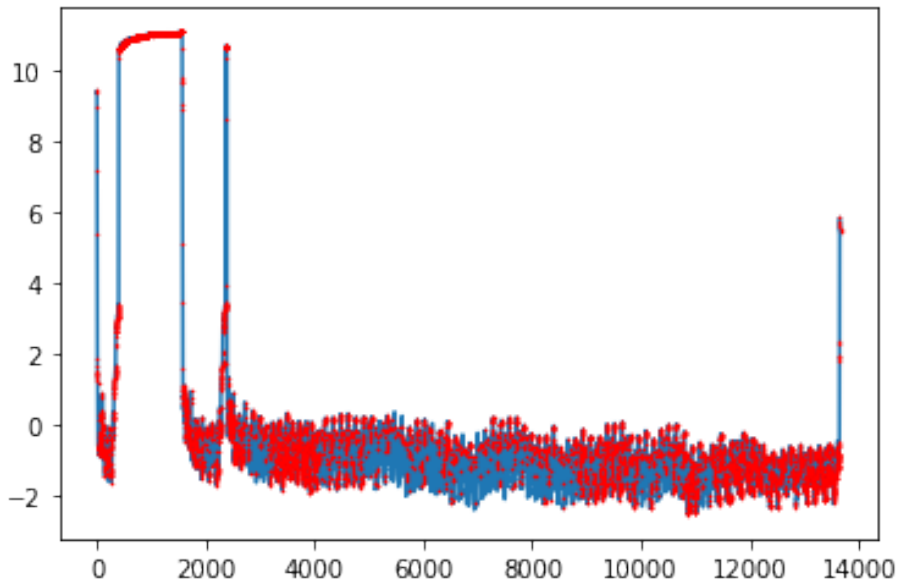

Fig. 7. One Class SVM

References

1. Ariyaluran Habeeb RA, Nasaruddin F, Gani A, et al. Clustering-based real-time anomaly detection—A breakthrough in big data technologies. Trans Emerging Tel Tech. 2019;e3647.
2. M. Solaimani, M. Iftekhar, L. Khan, B. Thuraisingham and J. B. Ingram, "Spark-based anomaly detection over multi-source VMware performance data in real-time," 2014 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), 2014, pp. 1-8, doi: 10.1109/CICYBS.2014.7013369.
3. S. Zhao, M. Chandrashekar, Y. Lee and D. Medhi, "Real-time network anomaly detection system using machine learning," 2015 11th International Conference on the Design of Reliable Communication Networks (DRCN), 2015, pp. 267-270, doi: 10.1109/DRCN.2015.7149025.
4. Ahmad S, Lavin A, Purdy S, Agha Z. Unsupervised real-time anomaly detection for streaming data. Neurocomputing. 2017;262:134-147.
5. Ding Z, Fei M. An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. IFAC Proc Vol. 2013;46(20):12-17.
6. Rettig L, Khayati M,Cudre-Mauroux P, Pirkowski M. Online anomaly detection over big data streams. Paper presented at: IEEE International Conference on Big Data (Big Data); 2015; Santa Clara, CA.
7. Falcão, Filipe and Zoppi, Tommaso and Barbosa Vieira da Silva, Caio and Santos, Anderson and Fonseca, Baldoino and Ceccarelli, Andrea and Bondavalli, Andrea. (2019). Quantitative comparison of unsupervised anomaly detection algorithms for intrusion detection. 318-327. 10.1145/3297280.3297314.

8. Papastefanopoulos,V.; Linardatos, P.; Kotsiantis, S. Unsupervised Outlier Detection: A Meta-Learning Algorithm Based on Feature Selection. Electronics 2021, 10, 2236. https://doi.org/10.3390/ electronics10182236
9. Alnafessah, Ahmad and Casale, Giuliano. (2020). AI Driven Methodology for Anomaly Detection in Apache Spark Streaming Systems. 1-2. 10.1109/ICCAIS48893.2020.9096667.
10. Solaimani, Mohiuddin and Iftekhar, Mohammed and Khan, Latifur and Thuraisingham, Bhavani. (2015). Statistical technique for online anomaly detection using Spark over heterogeneous data from multi-source VMware performance data. Proceedings - 2014 IEEE International Conference on Big Data, IEEE Big Data 2014. 1086-1094. 10.1109/BigData.2014.7004343.