Behavior Analysis Using Unsupervised Anomaly Detection

Markus Goldstein, Seiichi Uchida
Kyushu University, Fukuoka, Japan

{goldstein, uchida}@ait.kyushu-u.ac.jp

***Abstract*—The detection of anomalous behavior in log and sensor data is an often requested task for many data mining applications. If there are no labels available in the dataset as in many real-world setups, unsupervised anomaly detection would be the method of choice. Since these algorithms are not directly applicable on the data in general, an appropriate transformation has to be performed first. This paper describes how such "data views" could be generated with respect to the detection goal. It is also shown how contexts and associated events are taken into account correctly when creating the data view. Furthermore, a comparative evaluation of 11 different unsupervised anomaly detection algorithms on standardized datasets reveal useful strategies for selecting an appropriate algorithm. Finally, a real-world example of anomaly detection in power consumption data proves the usefulness of the presented methodology.**

I. Introduction

The detection of suspicious activities in log or sensor data is an often requested analysis procedure in many application domains. In particular, such applications include network intrusion detection, credit card and payment fraud detection, data leakage prevention, monitoring complex systems, medical data analysis, and many others. All of these very different domains have in common that the goal is often similar: rare events, which deviate from the norm, should be found. The most common term for this procedure is *anomaly detection*, but according to the application domain, often synonyms are used as well. This includes the terms fraud detection, outlier detection, misuse detection and also behavior analysis. Although there is no clear and commonly agreed definition, the term behavior analysis already might imply that an anomaly could comprise of more than just a single event. From an algorithmic perspective, anomaly detection algorithms are meant to detect single events only.

This paper describes how to bridge this semantic gap. For this reason, a few formal definitions are given first. Then, the generation of an appropriate data representation called *data view* for a given behavior analysis task is presented. Additionally, a comparative analysis of different unsupervised anomaly detection algorithms supports to select proper algorithms for practical behavior analysis tasks. Finally, a behavior analysis of real-world power consumption sensor data shows the relevance of the presented work.

II. Categorizing Anomaly Detection Tasks

Categorizing an anomaly detection task is a very important first step in order to select a proper data view and a suitable algorithm [1]. In the following, we use the categorization of [2] and [3]. In this context, anomalies are often defined by having two important characteristics:

1) Anomalies are different with respect to their feature values compared to normal data instances, and
2) In a dataset, the total amount of anomalies is much lower compared to the occurrence of normal instances.

The latter characteristic rises the question, how large the percentage of outliers should be at most. Unfortunately this is not easy to answer due to the fact that multiple algorithmic setup exist with respect to the availability of labels. One rule of thumb says that there should be not more that 5% of outliers in a dataset [3]. However, this rule is very important in unsupervised anomaly detection, somehow important when dealing with semi-supervised anomaly detection and could even be neglected when using supervised anomaly detection. These three different learning scenarios are explained in the following.

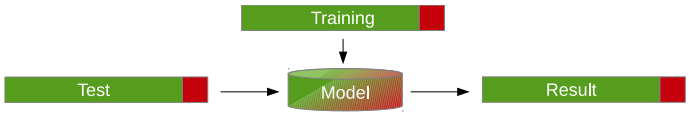
A. Availability of Labels

One main criterion for selecting a proper anomaly detection algorithm depends on the availability of labels of the dataset to be processed. If labels are present in the dataset for both, normal instances and anomalies, *supervised anomaly detection* can be used. If a dataset only contains instances labeled as normal, a *semi-supervised* approach can be used. If no labeling information is available at all, typically *unsupervised anomaly detection* is the method of choice. The three different anomaly detection modes are illustrated in Figure 1.

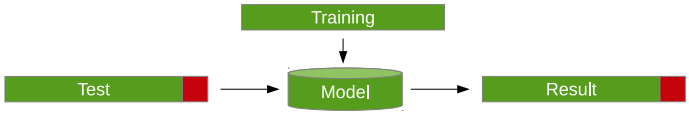
1) Supervised Anomaly Detection: Having normal and anomalous instances in a dataset makes supervised anomaly detection to a well-defined problem. Typically, it does not differ much from traditional machine learning except for having a strong prior difference of the classes. In this context, well-known classification algorithms, such as Support Vector Machines (SVMs) [4], Artificial Neural Networks [5], Bayesian Networks [6], or the k-nearest-neighbor algorithm could be used. Algorithms not being able to deal with a strong class bias, such as decision trees [7] should be avoided.

Since labeling information about anomalies are often not available in practice, supervised anomaly detection does not play a major role when dealing with real-world problems.

2) Semi-supervised Anomaly Detection: When only normal data is available for training and no assumption about the anomalies can be made, semi-supervised anomaly detection



(a) Supervised anomaly detection uses normal instances and anomalies for training and testing.



(b) Semi-supervised anomaly detection requires solely normal data for training. The learned model detects then deviations in the test data from that norm.

(c) Unsupervised anomaly detection uses no labeling information at all. The algorithm only takes intrinsic information of the data into account in order to detect anomalous instances being different from the majority.

Fig. 1. Depending on the availability of labels, a proper anomaly detection mode has to be selected.

is the method of choice. Semi-supervised anomaly detection algorithms are often based on the so called one-class classification idea [8]. In this context, one-class SVMs [9], replicator neural networks (also known as autoencoders) [10] or clustering could be used as semi-supervised anomaly detection algorithms. In general, also density estimation methods such as Gaussian Mixture Models (GMMs) [11] or Kernel Density Estimation (KDE) [12] can be used for estimating a probability density function for normality. A very well-known application scenario of semi-supervised anomaly detection is network intrusion detection. Please note that anomalies, which are accidentally present in the normal training data, may lead to a bad anomaly detection performance. In practical applications, training data could sometimes not be guaranteed to be anomaly free. Some algorithms, especially density estimation approaches, might deal better with this issue than others.

3) Unsupervised Anomaly Detection: If the dataset contains normal data and anomalies and no labels exist at the same time, unsupervised anomaly detection algorithms can be used. Here, the main idea is to use intrinsic information only, for example a density estimation of the dataset. Then, every single instance is scored based on the density of the area it resides in. Unsupervised anomaly detection is the most flexible methodology, especially in practice when data has been collected and should be analyzed without any further knowledge. On the other hand, it is very sensitive to the input data. A proper preprocessing and data view generation is essential for success. The remainder of this paper focuses on this anomaly detection mode.

B. Type of Anomalies

As already mentioned previously, a semantic gap between anomalies processable by algorithms and anomalies meant by humans exists. Algorithms always process data such that they detect single anomalous instances, so called point anomalies. More complex anomalies, comprising of multiple instances need to be converted in such a point anomaly detection problem first.

1) Point Anomalies: Point anomalies are single instances deviating from the mass of instances with respect to their features. When using an appropriate visualization, point anomalies can be easily spotted by humans when the data has at most three dimensions.

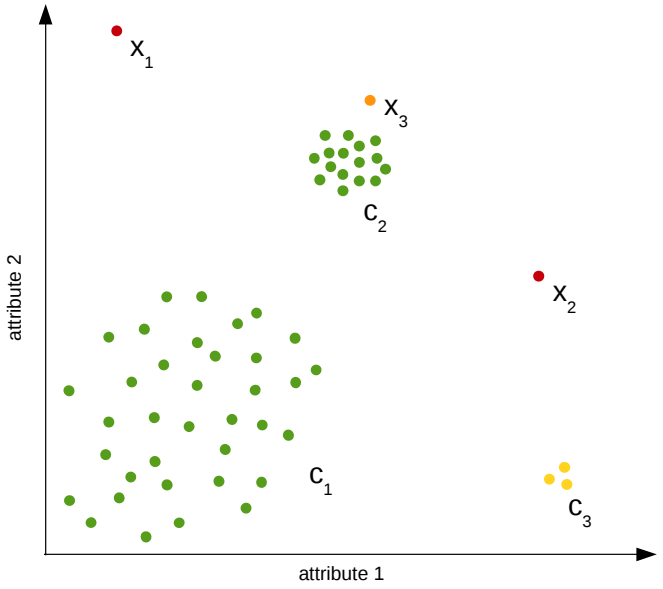


Fig. 2. A two-dimensional example of a point anomaly detection problem. The two global anomalies x_1 and x_2 are easily identifiable, x_3 is a local anomaly with respect to its direct neighborhood and c_3 is a micro-cluster.

Figure 2 illustrates such a point anomaly detection task as a two-dimensional example. Here, two anomalies could easily be identified: The instances x_1 and x_2 deviate clearly from the two larger clusters c_1 and c_2 and are also therefore called *global anomalies*. The instance x_3 is a *local anomaly*. When looking at the whole dataset globally, x_3 can be considered as normal. However, with respect to its direct local neighborhood, the cluster c_2, it can be considered as an anomaly. The three instances in cluster c_3 are another special case: A micro-cluster can be seen as an anomaly or as normal instances, depending on the application scenario. The color of the instances in Figure 2 already illustrates a possible result of an anomaly detection algorithm: Green instances should be detected as normal, red instances as anomalies and yellow as well as orange instances somewhere in between. This example also shows that it is in general a good idea for the output of a good anomaly detection algorithm to use a *score* instead of a class label.

2) Contextual Anomalies: For behavior analysis, the context of instances is very important. Therefore, we can define contextual anomalies as anomalies, which can only be identified within a specific context. As an example, suppose we are able to measure the power consumption of a building. Typically, the power usage follows the human daily routine. In this building, a power usage of 10 kWh might be normal. However, if such

a consumption is measured in the middle of the night where typically everybody is asleep, it might be an anomaly. Since this anomaly is only detectable when taking the context "time" into account, it is a contextual anomaly. In practice, many different contexts can be used. Besides time, also IDs, for example from users or devices are often used. Furthermore, any sensor reading such as temperature, counting persons, GPS coordinates, IP addresses and many more can be such contexts.

When applying anomaly detection algorithms in practice, the context needs to be taken into account when creating proper *data views*. Also, it is common that more than only a single context has to be taken into account as shown later in Section III.

3) Collective Anomalies: The last and most complex type of anomalies are collective anomalies. Here, a combination of many instances causes an anomaly, whereas each single one of these contributing instances is not a (point) anomaly itself. Consequentially, these kind of anomalies occur in datasets where the instances have a certain relationship with each other. In practice, log data or any sequentially collected datasets are often collective anomaly detection tasks. Of course, the correlation between this instances cannot be found automatically, since there is an extremely large number of possible combinations. Furthermore, in the unsupervised mode it is even impossible to evaluate these combinations due to missing labels. This means that similar to the contextual anomalies, proper data views have to be generated, aggregating the data such that potential correlating instances are grouped together. In almost all cases, domain knowledge is required for the generation of a data view.

III. Data View Generation

In order to detect the relevant anomalies, the generation of a proper data view is essential. Especially when unsupervised anomaly detection should be applied, the data view generation preprocessing step influences, which kind of anomalies are detected. For this reason, we focus in the following on unsupervised anomaly detection.

A. Entity

As a first step, it needs to be determined for which entity anomalies should be detected. In this context, an entity is the event or item, to which an anomaly should refer to. For example, if log data from electrical power consumption measuring devices ("smart taps") should be analyzed, multiple entities could be defined. One could try to find anomalies among the single devices, trying to answer the question whether one device (the entity) uses an uncommon amount of energy. Also, it is possible to aggregate the total consumption of all devices within a specific time frame (another entity) and check whether the whole building uses a suspicious amount of energy. If user information is additionally available, the entity to be modeled can also be a person. As we can see from this simple example, the entity definition always implicitly answers the question, which kind of anomaly should be focused on.

In some applications, where the analysis target is not clear in the beginning, also multiple data views, which use different entities each, can be used and analyzed in parallel [13].

B. Converting a contextual/collective task to a point anomaly detection task

As already mentioned, no contextual or collective anomaly detection algorithm exist in general for multidimensional data and data views need to be used for converting the task into a point anomaly detection problem. However, for one dimensional data and the context time, *time series analysis* methods could be used to detect anomalies, such as ARMA/ ARIMA [14] or artificial neural networks [15].

In practice, the context has to be integrated by data aggregation and discretization. In particular, for the context time, it is common to aggregate events for a specific time interval, for example the power consumption within one hour. Of course, multiple contexts may exist in parallel. Typically they are also taken into account by aggregation and discretization (binning). For example, to aggregate the total power consumption of all devices within one room for one hour takes two contexts into account, the time and the place. Also, it is possible to aggregate the power consumption on a per-device basis, which is illustrated in Figure 3. Here, the power consumption is binned for hourly intervals and for different devices, marked as the dark green intersection area of the hyperplanes. When applying the unsupervised anomaly detection algorithm, the device ID is then typically only used as an identifier, but not as a feature.

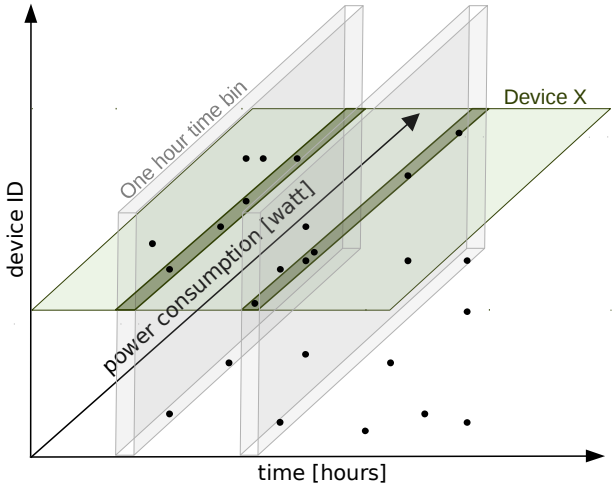


Fig. 3. A data view can be generated by integrating the contexts time and device ID. Here, the vertical gray slice represent an hourly bin and the green hyperplane a particular device.

Dealing with collective anomaly detection problems is very similar. For converting them into a point anomaly detection problem, domain knowledge is used such that events corresponding to a specific entity are aggregated. For example, if defective electrical consumers cause a certain power consumption pattern, this pattern can be coded as a feature for the anomaly

detection process. In sensor and log file data, this is often a specific and known sequence of particular events.

When generating a data view for unsupervised anomaly detection, it might also be a good decision in some cases to neglect features, when it is known that they do not contribute to a useful anomaly detection result. In general, features with no entropy might cause the unsupervised anomaly detection algorithm to fail. Due to the lack of labels, it is not possible that the algorithm can perform a feature selection, such as implicitly done by many supervised machine learning algorithms.

C. Normalization

For unsupervised anomaly detection, normalization of the features plays an important role. If the scales of the features are too different, algorithms will very likely tend to prefer specific features. This is due to the fact that most algorithms are based on distances and larger distances mean a smaller degree of similarity. In some cases this behavior could be desired, but in general, features should be normalized using the same scale. A range transformation to the interval [0.0, 1.0] is very common for numerical features. If categorical data is involved, it should be transferred into a numerical scale when the data is ordinal. For generic categorical data a distance of 0.0 and 1.0 could be used, depending if they share the same category or not. Mixing this kind of converted categorical distances with numerical data might be error-prone. Keep in mind that the two distances of 0.0 and 1.0 separate the data into 2 disjoint subsets, because the distance is always larger than the numeric distance. For dealing with this issue, a weighting between numerical and categorical data could be applied [3].

IV. Comparing Unsupervised Anomaly Detection Algorithms

Although the generation of the data view is the most important step for unsupervised anomaly detection, also the selection of a proper algorithm requires some thoughts.

A. Types of Algorithms

In this paper, we cannot describe all available algorithms in detail. Instead, we briefly explain a few important ones and their main characteristics. Roughly, the algorithms may be categorized in three main classes: (1) Nearest-neighbor based methods, (2) Clustering-based methods and (3) Statistical methods. Statistical methods can be sub-categorized into parametric or non-parametric methods such as histograms [16], Kernel-density estimation [17] or Gaussian Mixture Models [11]. Besides that, other methods based on classification techniques exist, such as Support Vector Machines [18] or autoencoders [10]. In the remainder of this section, we describe some well-known techniques from the first two main classes following [3].

For nearest-neighbor based approaches, the global k-NN algorithm [19], [20], the well-known Local Outlier Factor (LOF) [21], the Connectivity-Based Outlier Factor (COF) [22], the Local Outlier Probability (LoOP) [23], the Local Correlation Integral (LOCI) [24] as well as the Influenced Outlierness (INFLO) [25] were selected. Please note that the k-NN algorithm is global and all the remaining ones have been developed for detecting local anomalies. LOF estimates a local density around an instance and compares it with the densities of the k neighbors. This leads to a spherical density estimation, which has been replaced by a minimum spanning tree approach in COF. INFLO is basically another LOF modification, trying to improve some original shortcoming, which appears if clusters of different densities are close to each other. LoOP on the other hand works a little different – here, the local density is estimated by a half-Gaussian distribution. Also, the result of this algorithm is a probability instead of a score.

Among clustering based approaches, the Clustering-based Local Outlier Factor (CBLOF) [26] and a modified version uCBLOF [27] was chosen. The basic idea is to utilize k-means clustering and use the distance from each instance to the cluster center as an anomaly score. Since k-means might lead to different results among multiple runs, its result is not deterministic.

All algorithms are available within an open source anomaly detection plug-in [27] of the RapidMiner [28] data mining software, which has been used for evaluation in the following.

B. Evaluation and Results

Unfortunately, evaluation of unsupervised anomaly detection algorithms is not straightforward. In practice, unsupervised anomaly detection should be applied, when no labels are available. However, for a quantitative evaluation, labeling information is required such that anomaly detection performance can be measured. For this reason, 7 UCI machine learning datasets [29] have been altered to fit the unsupervised anomaly detection constraints: (1) Small amount of anomalies and (2) anomalies are different from the majority. For applying unsupervised anomaly detection, the labels are neglected, but for evaluation they are used. The typical evaluation procedure is as follows: First, the unsupervised anomaly detection algorithm scores all the instances. Then, the instances are sorted according to their anomaly score and finally, a threshold is shifted over all instances, always computing the true positive rate and the false positive rate, resulting in one Receiver Operating Characteristic (ROC) curve. The area under this curve (AUC) serves then as a quality measure for the algorithm. Please note that this is different from classification, where typically a parameter is altered to generate such a curve.

The modified datasets are publicly available and can be used for further evaluations[1]. In particular, they contain medical data, handwritten digits, data from space shuttle machinery, satellite images as well as an intrusion detection dataset. All the data has been preprocessed, features have been extracted and semantically useful anomalies were defined.

The results of the experiments are listed in Table I. Some of the algorithms also appear in the table as a slightly modified version. For k^{th}-NN, only the k^{th} element is used instead of an average over all neighbors. LOF is also evaluated with an

[1] See http://www.madm.eu/downloads

TABLE I
THE RESULTS OF THE UNSUPERVISED ANOMALY DETECTION ALGORITHMS USING THE INTERVAL $10 \leq k \leq 30$. THE AVERAGE AUC AND THE STANDARD DEVIATION IS LISTED FOR EACH DATASET. DUE TO ITS COMPLEXITY, LOCI CANNOT BE COMPUTED IN MOST CASES.

Alg.	breast-cancer	pen-local	pen-global	shuttle	satellite	annthyroid	kdd99
k-NN	0.9783 ±0.0009	0.9852 ±0.0004	0.9852 ±0.0071	0.9370 ±0.0052	**0.9703** ±0.0008	0.6057 ±0.0090	0.9714 ±0.0039
k^{th}-NN	0.9803 ±0.0008	0.9815 ±0.0038	**0.9894** ±0.0050	0.9348 ±0.0048	0.9694 ±0.0010	0.5854 ±0.0083	0.9769 ±0.0027
LOF	**0.9806** ±0.0030	**0.9889** ±0.0008	0.8244 ±0.0863	0.5232 ±0.0059	0.7118 ±0.0309	0.6613 ±0.0159	0.5727 ±0.0168
LOF (UB)	0.9796 ±0.0024	0.9887 ±0.0005	0.8084 ±0.0866	0.5279 ±0.0074	0.7671 ±0.0085	0.6728 ±0.0106	0.5637 ±0.0085
COF	0.9480 ±0.0047	0.9425 ±0.0128	0.7870 ±0.1306	0.5282 ±0.0084	0.6836 ±0.0137	0.6619 ±0.0134	0.5343 ±0.0111
INFLO	0.9558 ±0.0206	0.9827 ±0.0028	0.7614 ±0.0641	0.5064 ±0.0141	0.7596 ±0.0067	0.6671 ±0.0113	0.5328 ±0.0126
LoOP	0.9654 ±0.0139	0.9818 ±0.0082	0.6926 ±0.0860	0.5056 ±0.0041	0.7458 ±0.0125	**0.7013** ±0.0103	0.5500 ±0.0117
LOCI	0.9787	–	0.8877	–	–	–	–
aLOCI	0.8105 ±0.0883	0.8011 ±0.0615	0.6889 ±0.0345	0.9474 ±0.0379	0.8324 ±0.0372	0.6174 ±0.0221	0.6552 ±0.0458
CBLOF	0.2429 ±0.1176	0.7546 ±0.1129	0.2718 ±0.0858	0.8981 ±0.1444	0.5339 ±0.0429	0.5628 ±0.0392	0.7159 ±0.2141
uCBLOF	0.9746 ±0.0228	0.9476 ±0.0090	0.9011 ±0.0412	**0.9826** ±0.0224	0.9614 ±0.0035	0.5349 ±0.0226	**0.9957** ±0.0019

upper bound (UB) modification as introduced in the original publication [21], where the maximum LOF score for multiple different k's was taken. aLOCI refers to a approximate version of LOCI using quad trees for a fast nearest neighbor search. For a fair evaluation, multiple typical values for k have been evaluated. For the nearest-neighbor based methods, k was selected such that $10 \leq k \leq 30$. For the clustering based methods, k refers to the number of clusters and the same interval was used.

In general it can be concluded from the table as a major finding that local unsupervised anomaly detection algorithms typically fail on global anomaly detection tasks, such as kdd99 or the satellite dataset. Thus, it is very important to know before deciding on an algorithm, whether local anomalies are interesting or not with respect to the given task.

Another conclusion is that nearest-neighbor based methods perform often better than clustering based algorithms. Also, the deviation of the results is much smaller for nearest-neighbor based algorithms, which indicates a lower sensibility to the parameter setting. In practice, this means that if the dataset is not too big, nearest-neighbor based algorithms should be preferred. Another advantage is their deterministic nature leading to reproducible anomaly detection results. For very big datasets where it is known that a global anomaly detection task needs to be solved, utilizing a clustering-based approach might be advantageous (c.f. shuttle and kdd99 dataset).

V. EVALUATION USING REAL-WORLD DATA

In the last section, a comparative evaluation was given in order to describe the characteristics of the different algorithms available. For this reason, no data view was generated or it was rather performed prior and is already included in the dataset. Now, a practical example is described briefly to illustrate the data view generation and summarize the guidelines of this paper.

As data source, power usage measurements from smart-tap sensors are used. The data contains two contexts, the time of the measurement and the ID of the device. Therefore, a data view was generated where the entity refers to the power usage of a single device. The resulting data view contains as features the hour of the day, the day of the week and the amount of power used in that hour. Additionally, the device ID of each record is kept for reference but is not used as a feature for anomaly detection.

For algorithm selection, we have now the choice to use a local or a global anomaly detection algorithm. If we are more interested in answering the question whether there is a power consumption of a particular device deviating from the norm in total, we should refer to a global algorithm. In contrast, if we are more interested in finding anomalies with respect to the local neighborhood such as unusual power consumption within a specific daytime, a local algorithm might be preferred. To this end, it was decided to use a local algorithm and LOF was chosen. **[Unfortunately the evaluation results using the data had to be removed from this technical report for legal reasons.]**

VI. CONCLUSION

This paper serves as a generic guideline for detecting suspicious behavior using unsupervised anomaly detection. It is outlined which tasks can be addressed with unsupervised anomaly detection and why the generation of an appropriate data view is essential. From a practical point of view, taking contexts as well as collective events into account is described. After knowing

whether a local or global task should be solved, the quantitative evaluation supports selecting a suitable algorithm. Finally, a practical behavior analysis task complements the theoretical sections and reveals abnormal power usage.

REFERENCES

[1] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[3] M. Goldstein, "Anomaly detection in large datasets," PhD-Thesis, University of Kaiserslautern, München, Germany, 2 2014.

[4] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, ser. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2002.

[5] K. Mehrotra, C. K. Mohan, and S. Ranka, *Elements of Artificial Neural Networks*. Cambridge, MA, USA: MIT Press, 1997.

[6] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[7] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*, 1st ed. Chapman and Hall/CRC, 01 1984.

[8] M. M. Moya and D. R. Hush, "Network constraints and multi-objective optimization for one-class classification," *Neural Networks*, vol. 9, no. 3, pp. 463–474, 1996.

[9] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in Neural Information Processing Systems 12 (NIPS)*. The MIT Press, 11 1999, pp. 582–588.

[10] S. Hawkins, H. He, G. J. Williams, and R. A. Baxter, "Outlier detection using replicator neural networks," in *Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2000)*. London, UK: Springer-Verlag, 2000, pp. 170–180.

[11] B. Lindsay, *Mixture Models: Theory, Geometry, and Applications*, ser. NSF-CBMS Regional Conference Series in Probability and Statistics. Penn. State University: Institute of Mathematical Statistics, 1995.

[12] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *The Annals of Mathematical Statistics*, vol. 27, no. 3, pp. 832–837, 09 1956.

[13] M. Goldstein, S. Asanger, M. Reif, and A. Hutchinson, "Enhancing security event management systems with unsupervised anomaly detection," in *Proceedings of the 2nd International Conference on Pattern Recognition Applications and Methods (ICPRAM 2013)*, INSTICC. SciTePress, 2 2013, pp. 530–538.

[14] P. Whittle, "The analysis of multiple stationary time series," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 125–139, 1953.

[15] S. L. Ho, M. Xie, and T. N. Goh, "A comparative study of neural network and box-jenkins ARIMA modeling in time series prediction," *Computers and Industrial Engineering*, vol. 42, no. 2-4, pp. 371–375, 2002.

[16] M. Goldstein and A. Dengel, "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm," in *KI-2012: Poster and Demo Track*, S. Wölfl, Ed. Online, 9 2012, pp. 59–63.

[17] B. A. Turlach, "Bandwidth selection in kernel density estimation: A review," pp. 23–493, 1993.

[18] M. Amer, M. Goldstein, and S. Abdennadher, "Enhancing one-class support vector machines for unsupervised anomaly detection," in *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description (ODD '13)*. New York, NY, USA: ACM Press, 8 2013, pp. 8–15.

[19] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD '00)*. New York, NY, USA: ACM Press, 2000, pp. 427–438.

[20] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *Principles of Data Mining and Knowledge Discovery*, ser. Lecture Notes in Computer Science, T. Elomaa, H. Mannila, and H. Toivonen, Eds. Springer Berlin / Heidelberg, 2002, vol. 2431, pp. 43–78.

[21] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. Dallas, Texas, USA: ACM Press, 05 2000, pp. 93–104.

[22] J. Tang, Z. Chen, A. Fu, and D. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," in *Advances in Knowledge Discovery and Data Mining*, ser. Lecture Notes in Computer Science, M.-S. Chen, P. Yu, and B. Liu, Eds. Springer Berlin / Heidelberg, 2002, vol. 2336, pp. 535–548.

[23] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Loop: Local outlier probabilities," in *Proceeding of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*. New York, NY, USA: ACM Press, 2009, pp. 1649–1652.

[24] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos, "Loci: Fast outlier detection using the local correlation integral," in *Proceedings of the 19th International Conference on Data Engineering*. Los Alamitos, CA, USA: IEEE Computer Society Press, 2003, pp. 315–326.

[25] W. Jin, A. Tung, J. Han, and W. Wang, "Ranking outliers using symmetric neighborhood relationship," in *Advances in Knowledge Discovery and Data Mining*, ser. Lecture Notes in Computer Science, W.-K. Ng, M. Kitsuregawa, J. Li, and K. Chang, Eds. Springer Berlin / Heidelberg, 2006, vol. 3918, pp. 577–593.

[26] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers," *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1641–1650, 2003.

[27] M. Amer and M. Goldstein, "Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer," in *Proceedings of the 3rd RapidMiner Community Meeting and Conferernce (RCOMM 2012)*, I. M. Simon Fischer, Ed. Shaker Verlag GmbH, 8 2012, pp. 1–12.

[28] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler, "Yale: Rapid prototyping for complex data mining tasks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)*. New York, NY, USA: ACM Press, 2006, pp. 935–940.

[29] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml