Multivariate Unsupervised Machine Learning for Anomaly Detection in Enterprise Applications

Daniel Elsner
Technical University of Munich
daniel.elsner@tum.de

Pouya Aleatrati Khosroshahi
BMW Group
pouya.aleatrati-khosroshahi@bmw.de

Alan D. MacCormack
Harvard Business School
amaccormack@hbs.edu

Robert Lagerström
KTH Royal Institute of Technology
robertl@kth.se

Abstract

Existing application performance management (APM) solutions lack robust anomaly detection capabilities and root cause analysis techniques, that do not require manual efforts and domain knowledge. In this paper, we develop a density-based unsupervised machine learning model to detect anomalies within an enterprise application, based upon data from multiple APM systems. The research was conducted in collaboration with a European automotive company, using two months of live application data. We show that our model detects abnormal system behavior more reliably than a commonly used outlier detection technique and provides information for detecting root causes.

1. Introduction

Today, many organizations offer their services and products through the use of online platforms. The performance of these platforms is therefore critical to a user's experience. Poor user experiences caused by technical failures, such as a slow response time for a website, can have a high impact on the business success of these organizations: For example, Google, lost 20% of its user traffic in 2006 due to an increased website response time of only 500 ms [1].

The controlled monitoring of an enterprise's application portfolio with respect to its performance is called Application Performance Management (APM) [2]. By continuously recording operational metrics from ongoing information technology (IT) operations, APM systems seek to identify performance-related issues and inform the user in case of anomalies [2]. Current APM methods are naturally reactive, e.g., before an APM system reports a long response time for an application, this abnormality must first occur and be identified by the APM system. Only afterwards, can a user begin to identify and resolve the root cause of this abnormality. In response, Panwar [3] proposes a shift from reactive towards proactive and more analytical APM techniques, by harnessing current developments in machine learning (ML), statistical learning, and artificial intelligence (AI) [4].

There are a wide range of software-based solutions, such as AppDynamics[1], Dynatrace[2] and Nagios[3] that support APM in organizations [5]. Most of these solutions follow the same logic: they 1) collect different APM related data from different monitoring sources, 2) store and process the data to make them usable for further analysis, 3) make the data understandable through intuitive visualizations, and 4) enable interpretation and decision-making using notifications/alerts and features for exploring root cause analysis [2, 6]. While these solutions help experts to identify application performance bottlenecks in their enterprise applications, they are mostly limited to alert and visualization functions, rather than providing advanced anomaly detection capabilities. Furthermore, root cause analysis can often only be performed manually, by experts with significant domain knowledge about the applications being assessed [7].

In this paper, we examine the extent to which ML methods can be used to enable the identification of abnormal system behaviors in an enterprise application, using monitoring data from a variety of different APM systems as input. In particular, we address the following research questions (RQs):

- **RQ**$_1$: What are the requirements for a statistical model that can detect abnormal enterprise application behaviors using APM metrics?
- **RQ**$_2$: How reliable is the developed statistical model in terms of its ability to detect "real" enterprise application anomalies?

[1] https://www.appdynamics.com/
[2] https://www.dynatrace.com/
[3] https://www.nagios.org/

URI: https://hdl.handle.net/10125/60018
ISBN: 978-0-9981331-2-6

This paper presents a novel approach for identifying anomalies, based on a density-based clustering algorithm. The key output is a statistical model which enables the detection of enterprise application anomalies, using APM metrics as inputs. The research was conducted in cooperation with a large European automobile company. The development and evaluation of the statistical model was based upon real application performance data from this company, gathered over a two month time period. In our results, we show that our model recognizes "real" anomalies more effectively than a commonly used outlier detection technique. Importantly, real anomalies were identified by the owners of the system, and reflect times during the operation of the system where either there was an outage (e.g., the application was unavailable) or there were corrupt business transactions (e.g., the business logic was erroneous hence errors were returned).

This paper is structured as follows: Chapter 2 gives an overview of related work, and is followed by a brief explanation of our research approach in Chapter 3. The development of our model and the results of our evaluation are presented in Chapter 4. We discuss our results in Chapter 5 and end with a brief conclusion and outlook for further research in Chapter 6.

2. Related Work

Our literature review followed the guidelines established by Webster and Watson [8] and consisted of three structured steps: 1) identify relevant literature, 2) structure the review, and 3) define the research gap. To identify relevant literature, we defined two search queries and examined the online search catalog Scopus, the largest abstract and citation database for scientific contributions. This search revealed 19 contributions. After filtering out unrelated contributions, we were left with 8 articles that directly related to the topics of data mining/ML and anomaly detection.

Considering APM and anomaly detection from a data mining/ML point of view, research has examined the topic using numerous techniques. For instance Baraglia and Palmerini [9], Hussain et al. [10], and Fend and Vij [11], investigate APM with web usage mining algorithms to optimize web server performance. In their contribution, they classify web visitors, generate suggested links for user user navigation, or perform web cache pre-fetching to improve web server performance optimization. Hyndman et al. [12] approaches the topic with another focus and technique: they collected monitoring data from Yahoo, perform a principal component analysis (PCA) to reduce the dimensionality of multivariate time series data and then apply a density-based and α-hull based multi-dimensional outlier detection algorithm. Wang [13] focuses on forecasting the utilizations of the central processing unit (CPU) by proposing an auto-regressive integrated moving average with back-propagation neural network. Further contributions are provided by Hyndman et al. [14], Jiang et al. [15] and Cunha and Silva [16, 17]. Since these contributions are beyond the scope of this research, they are not discussed here. However, considering contributions from ML and data mining, our literature review reveals that only a few contributions have used unsupervised ML to detect enterprise application anomalies.

The second relevant research field is the anomaly detection community, which focuses on discovering enterprise application anomalies, based on APM metrics. As outlined earlier, numerous vendors provide software-aided solutions for APM. In this context, Ahmed et al. [18] recently compared multiple solutions in terms of detecting performance regressions and anomalies. They conduct a case study with three commercial and one open source APM tool to analyze the performance of three open source applications. Although the contribution shows that existing tools can support practitioners in detecting performance regressions, it also shows that they lack state-of-the-art mining approaches. Ara et al. [19] suggest a bivariate time series model for statistical monitoring of web servers to detect abnormal error rates (connection synchronization and rejection errors). Takashi et al. [20] focuses on network related performance anomalies by analyzing packet loss ratios. The results reveal implementation errors in web server technologies. A further approach is presented by Guo et al. [21]: they used modified support vector machines to detect abnormal patterns in server performance monitoring data and derived thresholds for an alarm system. Finally, although the APM anomaly detection research community offers numerous research fields, only Hyndmann et al. [12] use bivariate outlier detection techniques to detect anomalies.

Based on the literature review, we were able to identify a research gap, which was also relevant to application owners inside our automotive company partner: to investigate multivariate density-based anomaly detection techniques to identify application anomalies and root causes for this behaviour. We used this gap to derive requirements for developing anomaly detection models (see Section 4).

3. Research Approach

We aimed to create and evaluate a research artifact (i.e., a model to detect application anomalies), hence our research approach aligns to the design science approach described by Hevner et al. [22] and Peffers et al. [23]. As noted in Section 1, this work was conducted in collaboration with a large European automotive company. The company employs around 100,000 people and operates a portfolio of around 5,000 applications. The IT department consists of around 4,500 employees, with a dedicated department that deals with APM topics within the organization, as well as a "big data" department that deals with data analytics topics across the entire organization. Experts from both departments assisted during this research. Hence, our work is based upon the integration of prior academic knowledge (see Section 2) with managerial practice.

The anomaly detection model (i.e., the research artifact) was designed, developed, and evaluated iteratively over an eight-month period. Figure 1 illustrates the research steps conducted.

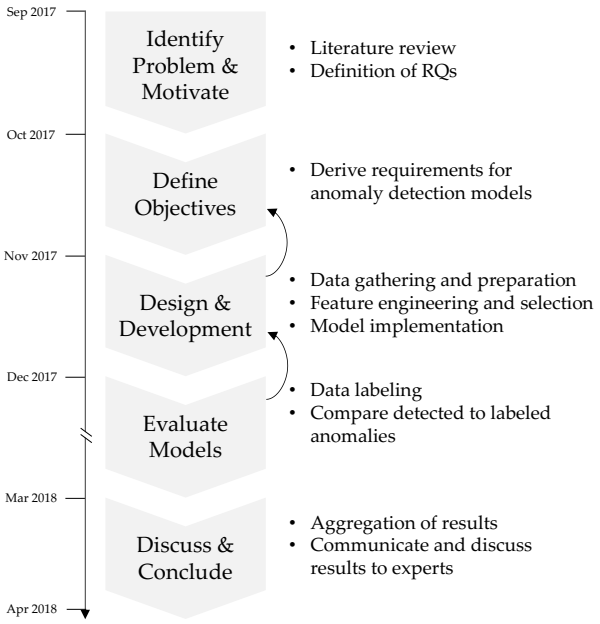


Figure 1. Research approach motivated by Hevner et al. [22] and Peffers et al. [23]

Identify Problem & Motivate: Based on our literature review and discussions with domain experts within the automotive company, we derived two RQs. The review followed best practices by Webster and Watson [8] and reveals contributions in APM-related fields, data mining/ML, and anomaly detection. The results of the review are given in Section 2.

Define Objectives: The results of the literature review were aggregated and discussed with experts inside the company. Based on the practical relevance of the literature and data availability inside the firm, requirements were derived for the model development. These requirements are discussed in Subsection 4.1.

Design & Development: APM monitoring data was gathered from a variety of sources, and then prepared for processing. Based on the defined objectives and data availability, feature engineering and selection was conducted by incorporating expert domain knowledge. The implemented statistical models addressed the requirements from prior research steps by incorporating density-based unsupervised ML techniques. Model design and development are described in detail in Subsection 4.2.

Evaluate Models: The evaluation of the models relied on obtaining labels for "abnormal" time periods in the operation of the enterprise application that we studied. These labels were provided by the owners of the system. The results were presented and discussed with the big data department and APM experts of the automotive company. Based on their insights, the models and selected features were adjusted iteratively. Subsection 4.3 gives detailed information about the evaluation.

Discuss & Conclude: Based on the results of the evaluation, the defined RQs were answered and presented to the company. The model, which incorporates a novel approach for detecting anomalies, as well as an indication for the root cause of these anomalies, performs more precisely than a baseline model that is based upon a commonly used approach for outlier detection. We also identified limitations of our work, and suggest future work that builds upon this research. These are presented in Section 6.

4. Anomaly Detection

4.1. Define Objectives

Based upon our literature review, we derived the following research requirements (Reqs) for our anomaly detection model to address the defined research gap:

- **Req$_1$:** The model should harness density-based unsupervised ML techniques to detect anomalies in enterprise applications.

- **Req$_2$:** The model should accept a multivariate feature vector as input in order to incorporate multiple APM metrics of different types.

- **Req$_3$:** The model should output a continuous measure of the degree of abnormality in the system over time, called the Anomaly Index.
- **Req$_4$:** The model should provide an indication of the root cause of detected anomalies, by showing which APM metrics most impact system state.
- **Req$_5$:** The model should be evaluated with respect to labelled time periods where anomalies were known to exist by the organization.

4.2. Design & Development

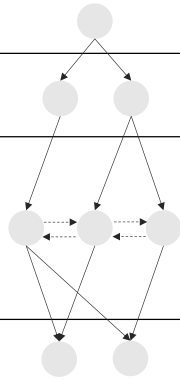
The monitoring data used for building the models comes from three different APM systems of a focal enterprise application. The application was chosen based upon its high business relevance, as well as the quality and availability of monitoring data. In cooperation with experts from the APM department, the scope of our analyses was defined as follows:

- **Time Period of Collected Data:** Two months, from October 1, 2017 to November 30, 2017.
- **Input:** Fifteen APM metrics as time series data of varying collection and aggregation intervals, from 1 data point per minute to 1 data point per 60 minutes (step size defined as 1 minute). The source systems for the data were Dynatrace, Nagios, and JMX monitoring on web server, application server, and infrastructure levels.
- **Output:** A continuous numerical measure, which reflects the degree of anomaly, i.e., the Anomaly Index, in the enterprise application over time.

4.2.1. Data Source Identification In figure 2, the APM measures collected by the monitoring systems across the four-layer enterprise architecture are outlined with their respective collection intervals. Since client data was not available, we only consider server monitoring data from the point where a client request arrives at the web server layer, where a load balancer sends the request to an application sever instance. The 15 available APM measures are as follows:

- **Requests per Minute:** The amount of requests from clients arriving at the web server layer.
- **Traffic per Minute**: The amount of data in bytes transferred through the web server layer.
- **CPU Load:** The average CPU utilization in percentage.
- **MQS Queuelength:** The average length of the message queue.
- **JVM Threads:** The average amount of JVM threads in the JVM running the application.
- **JVM Heap Used:** The average heap memory used in the JVM running the application in megabytes.
- **Response Time:** The response time of application servers in milliseconds.
- **Path Duration:** The complete duration a path was executed by the application server in milliseconds. Additionally to the response time, this measure can contain asynchronous subsequent tasks.
- **Failed Transactions:** Amount of failed transactions includes every transaction resulting in an HTTP error.
- **Path Amount:** Amount of processed execution paths (i.e., traces).
- **Exception Count:** Amount of exceptions thrown by the application server.
- **Node Count:** Total amount of nodes (i.e., Dynatrace agents) visited in all processed paths.
- **DB Sessions:** Amount of current database sessions.
- **DB Physical Reads:** Amount of reading operations in database.
- **DB Physical Writes:** Amount of writing operations in database.

To evaluate the models developed, we needed to acquire labels of abnormal behavior within the enterprise application, representing a so-called ”ground truth” [4]. This was achieved through a simple web-based software interface, which allowed application owners to manually label abnormal time periods. In the considered time period of two months, eight abnormal time periods were identified by these domain experts, ranging in duration from 21 to 316 minutes. The labels reflect time periods for the application in which either an outage occurred (i.e., service was unavailable) or corrupt business transactions were prevalent (i.e., business logic was erroneous).

Layer	Exemplary Unidirectional Path	Nagios	JMX	Dynatrace
Client				
Web Server		Requests per Minute (1/1.5min) Traffic per Minute (1/1.5min) CPU Load (1/1.5min)		
Application Server		CPU Load (1/1.5min) MQS Queuelength (1/2min)	JVM Threads (1/1.5min) JVM Heap Used (1/1.5min)	Average Response Time (1/min) Maximal Response Time (1/min) Average Path Duration (1/min) Maximal Path Duration (1/min) Failed Transactions (1/min) Path Amount (1/min) Exception Count (1/min) Node Count (1/min)
Database Server		DB Sessions (1/60min) DB Physical Reads (1/60min) DB Physical Writes (1/60min)		

Logical instance of server or client

Figure 2. Collected APM measures on four-layer architecture from three monitoring systems: Nagios, JMX, Dynatrace

4.2.2. Design Guidelines Our contribution consists of developing statistical models which harness density-based unsupervised ML techniques for detecting anomalies in application performance. The design of our models is inspired by a number of prior efforts. First, Overseer Labs[4] use K-Means clustering for root cause analysis of bad system health [24, 25]. The work is motivated by the fact that individual measures are noisy and can trigger false positives when it comes to detecting anomalous system behaviors. Hence the authors propose the use of multivariate measures to develop an overall indicator of system health. Second, Netflix[5] apply a clustering algorithm called Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to detect unhealthy servers in a server-farm, using univariate time series data [26]. We extend this work by showing how it can be used with multivariate data from multiple APM systems. Finally, Goldstein and Uchida compare a number of unsupervised ML outlier detection algorithms on multivariate data sets, and show that a technique called Local Outlier Factor (LOF), which detects outliers based upon the density of the local neighborhood for each observation, performs well. We use the LOF method as a baseline with which to compare the performance of our new model [27].

Below, we define our design guidelines (DGs), which capture useful aspects of this former work and frame them in terms of concrete implementation steps:

- **DG**$_1$: To address Req$_1$, we apply and compare DBSCAN and LOF, density-based unsupervised ML and outlier detection techniques, respectively. [27].
- **DG**$_2$: In alignment with Req$_2$ and Req$_3$, we emphasize multivariate modeling to output a measure that reflects overall system health (i.e., the Anomaly Index) [24].
- **DG**$_3$: As constituted in Req$_4$, we use the information about individual contributions of metrics to the overall anomaly index to provide an indication about the root cause of the anomaly [24].

Figure 3 depicts the conceptual process for the model development, consisting of data gathering and preparation, feature engineering and selection, model implementation, and data labelling and model evaluation with respect to these designated DGs.

4.2.3. Feature Engineering & Selection After obtaining the data from the source systems described above, several steps needed to be taken. First, we needed to fill in missing data observations for some time periods – we tested the use of both mean and median filling and chose the median method, given it produced more robust results. Second, the APM time series measures had different scales, hence we standardized the scales to enable clustering. Third, to

[4] http://overseerlabs.io/
[5] https://www.netflix.com/

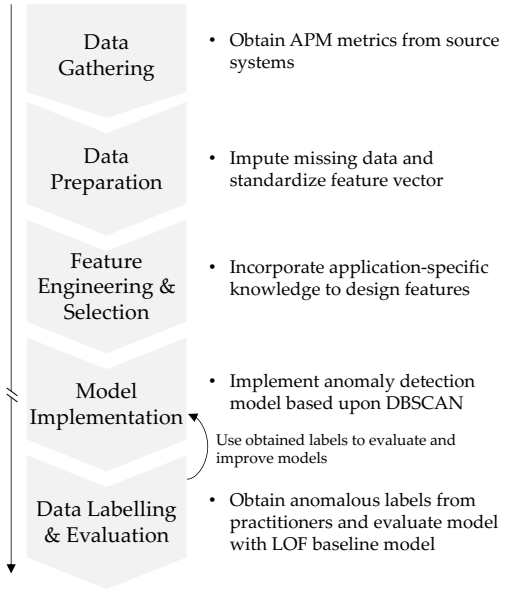


Figure 3. Conceptual process for the model development

supplement the set of 15 APM metrics, we developed four self-engineered features, which combine different APM metrics. These included:

- **Average Failure Rate**
- **Average Exceptions per Path**
- **Average Failures per Path**
- **Average Nodes per Path**

Domain experts at the company selected the most relevant APM metrics for the application, which are used as features in the anomaly detection models. The two models based on DBSCAN and LOF were then trained using the resulting feature vector. After obtaining anomalous labels from system owners, the labels were used to improve and evaluate the implemented models and tune their hyperparameters. Furthermore, the set of selected features was enhanced by conducting correlation analyses on the obtained labels and the available APM metrics. Table 1 shows the best feature set after these analyses.

4.2.4. Model Implementation Our research approach was to compare our newly developed model – a novel approach incorporating DBSCAN – to a second model using LOF, a well-performing outlier detection technique [27].

For our novel approach using DBSCAN we took the following four steps:

Table 1. Best performing feature set for anomaly detection models

Average Path Duration
Average Response Time
Failure Rate
Maximal Response Time
Maximal Path Duration
MQS Queuelength
Failed Transactions
JVM Threads

1. Train the DBSCAN model on multivariate (n-dimensional), standardized feature vector X with model hyperparameters Eps (the radius of the density neighborhood) and $MinPts$ (the minimal amount of neighbors in the neighborhood). The initial set of hyperparameters is found by applying a heuristic from Ester et al. [28].
2. Select the largest cluster (i.e., containing most observations) and define it as the normal system state. Set its centroid C_{normal} as the reference point for the calculation of the anomaly index.
3. Calculate the euclidean distance d from each observation o to C_{normal} ($d = \sqrt{\sum_{i=1}^{n}(o_i - C_{normal_i})^2}$) in the n-dimensional space, which reflects the anomaly index for the observation.
4. For each observation o, calculate the relative contribution of each of the n dimensions (i.e., features) to the total euclidean distance d.

The procedure is illustrated in figure 4 for the two-dimensional case. Note that each observation in the n-dimensional space reflects one point in time. Therefore, by calculating the euclidean distance for each observation, the anomaly index can be constructed over time. The model achieves the best performance with $Eps = 1$, the radius of the density neighborhood, and $MinPts = 10$, the minimal amount of neighbors in the neighborhood with respect to the evaluation metrics defined in the subsequent subsection.

In comparison to DBSCAN, the LOF model assigns an outlier factor to each observation based on the density of its local neighborhood. The mathematical derivation of the LOF (i.e., outlier factor) can be found in Breunig et al. [29]. The procedure for developing this model is straightforward:

1. Train the LOF model on the multivariate (n-dimensional), standardized feature vector X with hyperparameter $MinPts$.

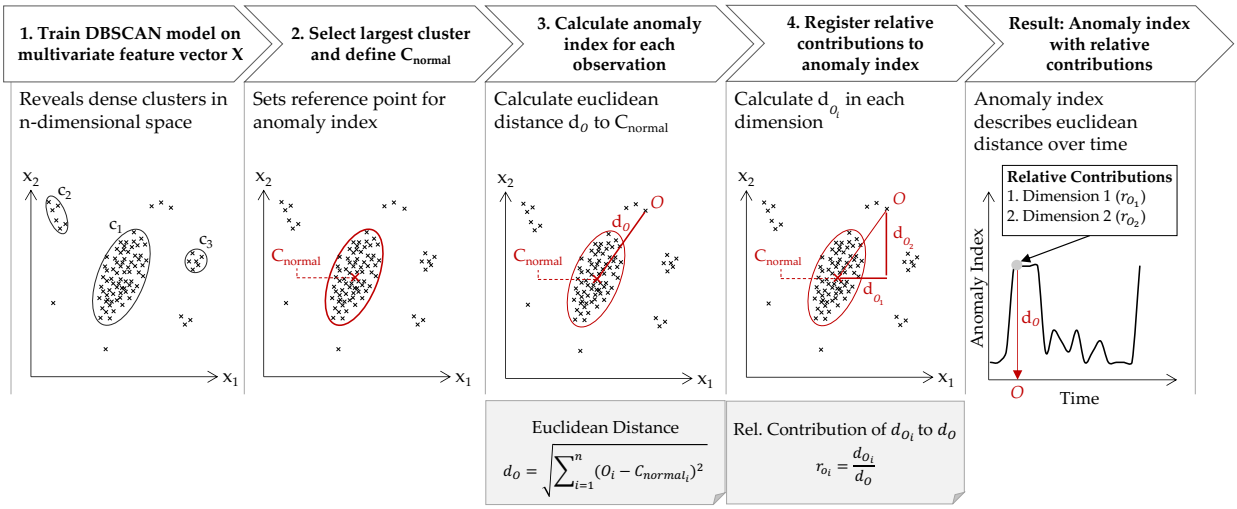


Figure 4. Procedure of DBSCAN model: Derive anomaly index for each time step from euclidean distance to centroid of cluster of normal system state

2. Calculate the LOF for each observation o, which can be interpreted as the anomaly index for the respective observation.

Note that the LOF model cannot implement DG_4 nor satisfy Req_4, as the individual contribution of each dimension to the anomaly index cannot be derived from the LOF. The model achieves the best performance $MinPts = 200$ with respect to the evaluation metrics defined in the subsequent subsection.

4.3. Evaluate Models

To quantitatively compare the quality of anomaly detection models, different evaluation metrics can be used. Goldstein and Uchida [27] state that the Area Under The Curve (AUC) based evaluation has evolved to be the standard for unsupervised anomaly detection, given anomaly detection problems are typically imbalanced [30, 27]. The AUC is the integral of the Receiver Operating Characteristic (ROC) curve, which plots the True Positive Rate (TPR) on y-axis against the False Positive Rate (FPR) on the x-axis at various threshold levels. A perfect AUC score is 1, whereas the worst score is 0:

$$AUC = \int_{\infty}^{-\infty} TPR(T)(-FPR'(T)), dT \quad (1)$$

An anomaly that is labelled by a system owner may be completely detected, partly detected, or not detected at all, with respect to the anomaly index generated by the algorithm and a defined critical threshold (see figure 5). In our work, we set the critical

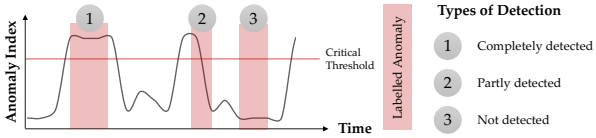


Figure 5. Types of detection for labelled anomalies in enterprise application

threshold to $T_{\mu+3*\sigma} = \mu + 3 * \sigma$, which is derived from statistical process control theory and the six sigma principle [31]. We define two different decision rules for interpreting whether partly detected anomalies qualify as being completely detected for evaluation purposes. Specifically, we define the anomaly as being detected (i) if the maximal anomaly index inside the labelled anomalous time period exceeds the critical threshold, or (ii) if the mean anomaly index inside the labelled anomalous time period exceeds the critical threshold. We evaluate the different results for these different decision rules, in terms of the comparison between DBSCAN and LOF methodologies, below.

To consider collective anomalies rather than point anomalies and address the problem of imbalanced True Positives (TPs) to False Positives (FPs), we use time windows of different sizes to evaluate the quality of detection, as proposed by Ahmad et al. [32]. These windows are then slid across the entire observation period of two months.

Figure 6 depicts the AUC over different window sizes from 1 to 300 minutes with respect to decision rule (i), whereas figure 7 covers decision rule (ii). By examining the AUC over different window sizes, we find that especially for smaller window sizes, the DBSCAN

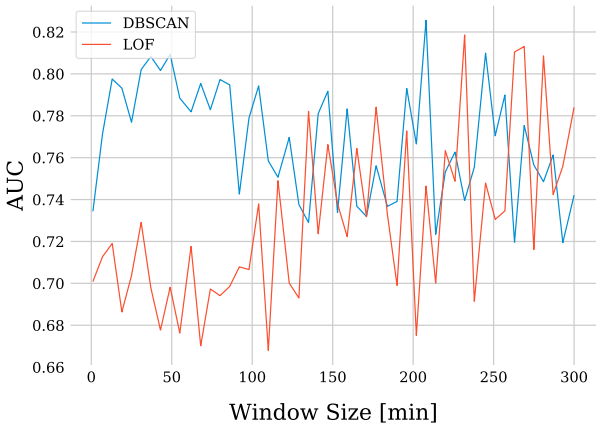


Figure 6. Comparison of AUC evaluation metric for developed anomaly detection models, plotted over varying window size with decision rule (i)

model yields better detection quality than the LOF model. Moving towards larger window sizes the AUC scores are approximately on an equal level. For the largest window sizes ($>$ 250 minutes) the LOF model performs better than the DBSCAN model for decision rule (i). Overall, for decision rule (i) the DBSCAN model has a better AUC score than the LOF model for 39 of the 50 considered window sizes. For decision rule (ii) the DBSCAN model has a better AUC score than the LOF model for 48 of the 50 considered window sizes. We conclude that the DBSCAN model is superior with respect to our primary evaluation metric AUC.

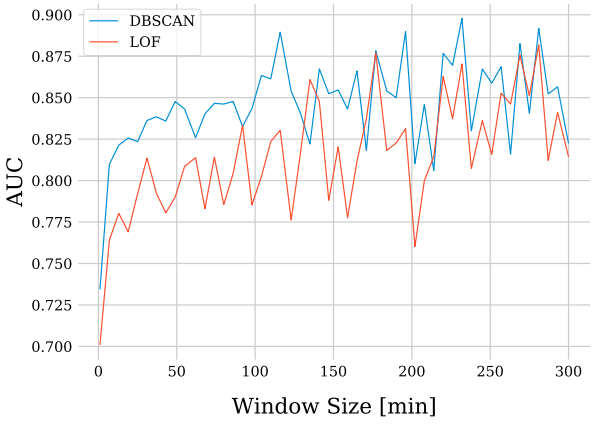


Figure 7. Comparison of AUC evaluation metric for developed anomaly detection models, plotted over varying window size with decision rule (ii)

Significantly, only the DBSCAN model provides an indication of the root cause of high anomaly values, and therefore meets the requirements of the research artifact. The LOF model does not meet the established requirements and also performs worse than the DBSCAN model. However, we should note that we have not evaluated indicators for the root cause of identified anomalies, which are output by the DBSCAN model. Section 6 further discusses this limitation and provides an outlook for future research.

5. Discussion

As outlined in the introduction, we defined two RQs in this research: RQ_1 investigates what requirements should be considered when defining a statistical model to detect abnormal application behavior, whereas RQ_2 investigates how reliable the developed model detects anomalies based on APM metrics.

The research requirements are based on a structured literature review and feedback from practitioners. We identified five requirements and derived three DGs (see Section 4), which contribute to answering RQ_1. These required developing anomaly detection models, which incorporate multiple APM metrics and harness unsupervised density-based ML techniques. In particular, multiple APM metrics are considered simultaneously and their individual values are put into the context a of system state (Req_1, Req_2). The system state and the degree of its abnormality is reflected in a continuous measure, the so-called Anomaly Index (Req_3). Furthermore, our model provides an indication for the root cause of an anomaly by considering respective contributions of selected metrics to the anomaly index (Req_4). The requirements and DGs ensure that the model can detect anomalies of applications regardless of any domain, application type, or other characteristic, based only on APM metrics.

Considering RQ_2, we evaluated our models with real monitoring data. Since there were no labelled anomalies available ex-ante, we obtained labels for eight time periods of abnormal application behavior from the system owners ex-post, which served as a "ground truth" with which to investigate the quality of the statistical models. We use a sliding time window approach to evaluate model quality, using different decision criteria for interpreting how partly detected anomalies should be treated. We find the novel approach of a model incorporating DBSCAN is more accurate than a baseline model LOF, that has been shown in prior research to be effective: furthermore, the DBSCAN model provides an indication of the root cause for the detected anomalies.

6. Conclusion

In this paper, we investigated the extent to which unsupervised density-based ML techniques can be used to identify application anomalies based on APM metrics. The research was conducted in collaboration with a large European automotive company and involved the use of real data for model construction and evaluation. Our final suggested anomaly detection model is based upon the use of DBSCAN and represents a novel approach that builds upon and extends prior work: the definition of requirements and DGs ensures the quality and the generalization of the model and the evaluation reveals better accuracy than a commonly used outlier detection technique called LOF.

There are some limitations in our work that should be noted: First, our research approach was limited in terms of model tuning and feature engineering, due to data availability at the automotive company. Extended data may reveal further interesting features for the detection of anomalies through APM data. In particular, the availability of labelled data (i.e., datasets with labels for anomalies) would allow us to consider other detection techniques (e.g. supervised ML). Second, our evaluation does not compare our defined model with anomaly detection features in existing APM solutions from software vendors. Hence, we do not know whether our techniques outperform these embedded features. This could serve as future work. Third, the measures that were output on the root causes for detected anomalies was not evaluated here. This functionality needs to be evaluated in further research. Last, the developed unsupervised ML approach is to be evaluated regarding its generalization on other data sets and in comparison to other methods mentioned in the literature review.

Significantly, future research could be used to help define a prediction model for the defined anomaly index: forecasting anomalies and potential root causes for these events would be an extremely beneficial topic for researchers and practitioners alike. Finally, the statistical relevance of the model could be increased with further data: in particular, the evaluation we conduct here was based upon a data-set from only one application and company. Examining the techniques across multiple applications and eventually, multiple companies, would help to generalize these techniques.

References

[1] G. Linden, "Marissa Mayer at Web 2.0," 2006.

[2] C. Heger, A. van Hoorn, M. Mann, and D. Okanović, "Application performance management: State of the art and challenges for the future," in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering - ICPE '17*, pp. 429–432, 2017.

[3] M. Panwar, "Application performance management emerging trends," in *2013 International Conference on Cloud & Ubiquitous Computing & Emerging Technologies*, pp. 178–182, 2013.

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

[5] C. Haight, W. Cappelli, and F. De Silva, "Magic quadrant for application performance monitoring suites," *Gartner*, 2015.

[6] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *Journal of Grid Computing*, vol. 12, no. 4, pp. 559–592, 2014.

[7] C. Heger, A. V. Hoorn, D. Okanovic, S. Siegl, and A. Wert, "Expert-guided automatic diagnosis of performance problems in enterprise applications," in *Proceedings - 2016 12th European Dependable Computing Conference, EDCC 2016*, pp. 185–188, 2016.

[8] J. Webster and R. T. Watson, "Analyzing the past to prepare for the future: Writing a literature review," *MIS Quarterly*, vol. 26, no. 2, pp. xiii – xxiii, 2002.

[9] R. Baraglia and P. Palmerini, "SUGGEST: A web usage mining system," in *Proceedings - International Conference on Information Technology: Coding and Computing, ITCC 2002*, pp. 282–287, 2002.

[10] T. Hussain, S. Asghar, and S. Fong, "A hierarchical cluster based preprocessing methodology for web usage mining," in *Proc. - 6th Intl. Conference on Advanced Information Management and Service, IMS2010, with ICMIA2010 - 2nd International Conference on Data Mining and Intelligent Information Technology Applications*, pp. 472–477, 2010.

[11] W. Feng and K. Vij, "Machine learning prediction and web access modeling," in *Proceedings - International Computer Software and Applications Conference*, vol. 2, pp. 607–612, 2007.

[12] R. J. Hyndman, E. Wang, and N. Laptev, "Large-scale unusual time series detection," in *Proceedings - 15th IEEE International Conference on Data Mining Workshop, ICDMW 2015*, pp. 1616–1619, 2016.

[13] J. Wang, Y. Yan, and J. Guo, "Research on the prediction model of CPU utilization based on ARIMA-BP neural network," in *MATEC Web of Conferences*, vol. 65, 2016.

[14] R. J. Hyndman, "Computing and graphing highest density regions," *American Statistician*, vol. 50, no. 2, pp. 120–126, 1996.

[15] N. Jiang, R. Villafane, K. Hua, A. Sawant, and A. Prabhakara, "ADMiRe: An algebraic data mining approach to system performance analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 7, pp. 888–901, 2005.

[16] C. A. Cunha and L. Moura E Silva, "Separating performance anomalies from workload-explained failures in streaming servers," in *Proceedings - 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012*, pp. 292–299, 2012.

[17] C. A. Cunha and L. Moura E Silva, "Prediction of performance failures in video-streaming servers," in *Proceedings of IEEE Pacific Rim International Symposium on Dependable Computing, PRDC*, pp. 283–292, 2013.

[18] T. M. Ahmed, C.-P. Bezemer, T.-H. Chen, A. E. Hassan, and W. Shang, "Studying the effectiveness of application performance management (APM) tools for detecting performance regressions for web applications," in *Proceedings of the 13th International Workshop on Mining Software Repositories - MSR '16*, pp. 1–12, 2016.

[19] A. Ara, F. Louzada, and C. A. Diniz, "Statistical monitoring of a web server for error rates: a bivariate time-series copula-based modeling approach," *Journal of Applied Statistics*, vol. 44, no. 13, pp. 2287–2300, 2017.

[20] S. Takashi, K. Eiji, Y. Suguru, and Y. Heiichi, "Performance anomalies of advanced web server architectures in realistic environments," in *8th International Conference Advanced Communication Technology, ICACT 2006 - Proceedings*, vol. 1, pp. 169–174, 2006.

[21] X. Guo, Z. Zhao, and P. Guo, "Studies about convex hull support vector machine in the server performance alarm," in *2011 3rd International Workshop on Intelligent Systems and Applications, ISA 2011 - Proceedings*, 2011.

[22] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[23] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007.

[24] U. Hasan, "How to use machine learning to debug outages 10x faster," 2016. Available at https://engr.overseerlabs.io/how-to-use-machine-learning-to-debug-outages-10x-faster-e19a97946f80.

[25] U. Hasan, "Our challenges in building a useful anomaly detection system," 2017. Available at engr.overseerlabs.io/our-challenges-in-building-a-useful-anomaly-detection-system-1d589de77e60.

[26] P. Fisher-Odgen, G. Burrell, C. Sanden, and C. Rioux, "Tracking down the villains: outlier detection at Netflix," 2015. Available at https://medium.com/netflix-techblog/tracking-down-the-villains-outlier-detection-at-netflix-40360b31732.

[27] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PLoS ONE*, vol. 11, no. 4, 2016.

[28] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, 1996.

[29] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *Proceedings of the 2000 Acm Sigmod International Conference on Management of Data*, pp. 93–104, 2000.

[30] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to roc, informedness, markedness & correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[31] D. Montgomery, *Introduction to statistical quality control*. 2009.

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