SN Computer Science (2022) 3:26
https://doi.org/10.1007/s42979-021-00862-8

ORIGINAL RESEARCH

Addressing IT Capacity Management Concerns Using Machine Learning Techniques

Hendrik Müller[1] · Andrey Kharitonov[1] · Abdulrahman Nahhas[1] · Sascha Bosse[1] · Klaus Turowski[1]

Received: 4 December 2020 / Accepted: 6 September 2021 / Published online: 29 October 2021

Abstract
The digitization of various industries is emerging from and being supported by the Information Technology (IT) industry. However, bringing about the practical implementation of the fourth industrial revolution will require different fields of IT to undergo their own transformations. One of the research fields that draws a lot of attention nowadays is machine learning and its application in different areas. Therefore, in this paper, we present an analysis on the applicability of various machine learning techniques to address different problems in the field of capacity management for Commercial-off-the-shelf enterprise applications. Our investigation of the selected machine learning techniques is based on real monitoring data from over 18,000 SAP applications and database instances that are hosted on more than 16,000 different physical servers. These data are used to train various performance models, such as support vector machines with different kernels, random forests, and AdaBoost, for standard business functions. Boosted trees achieve sufficient accuracy to predict mean response times for ten frequently used transactions. To evaluate the suggested models, we applied them successfully to address different concerns in the context of capacity management. The evaluation includes multiple scenarios in the fields of server sizing, load testing, and server consolidation, with the objective to identify cost-effective designs. Based on the same monitoring data, we also present an anomaly detection scenario. In this scenario, we aim to demonstrate the use of machine learning techniques on historical data to detect possible performance anomalies for a suggested design or even predict possible anomalies in future scenarios. Results strongly emphasize the need to integrate monitoring data from standardized business applications to allow for novel and cost-effective capacity management service offerings.

Keywords Commercial-off-the-shelf enterprise applications · Response time prediction · Capacity management as a service · Performance anomaly detection

Introduction

The fulfillment of requirements for information processing is becoming increasingly complex. The digital transformation affects many industries with the ongoing trends of automation and computerization [23, 25, 42]. According to [21], 47% of the US employment may be computerized by 2033. Particularly, *routine tasks involving explicit rule-based activities* are at highest risk to be automated as such rules may be learned by algorithms much quicker than by humans (in some cases, seconds instead of years). Even [21] used machine learning (ML) techniques to classify jobs into computerization risk groups. Here, the IT industry takes on a pivotal role as it is, on the one hand, vigorously driving the outlined trend with rapid innovations, e.g., in the fields of big data, artificial intelligence, and the internet of things [37, 38, 58]. On the other hand, IT processes themselves

This article is part of the topical collection “Advances on Signal Image Technology and Internet based Systems” guest edited by Albert Dipanda, Luigi Gallo and Kokou Yetongnon.

✉ Hendrik Müller
hendrik.mueller@ovgu.de

Andrey Kharitonov
andrey.kharitonov@ovgu.de

Abdulrahman Nahhas
abdulrahman.nahhas@ovgu.de

Sascha Bosse
sascha.bosse@ovgu.de

Klaus Turowski
klaus.turowski@ovgu.de

1 Faculty of Computer Science, Otto von Guericke University, Universitätsplatz 2, 39106 Magdeburg, Germany

are severely affected by automation and data-driven decisions [28], which necessitate the transformation of IT. One of the crucial IT processes to support cost-effective operations is the capacity management process [31]. This paper, therefore, investigates the utility of machine learning techniques to support typical capacity management scenarios in the domain of enterprise applications (EA). Analyses of machine learning techniques and corresponding related work, described in "Related work", "Performance prediction model" and "Performance prediction model", are based on our previous work presented in [52] and originally published in [48]. We studied the utility of machine learning techniques to deal with capacity management related tasks such as the sizing of new servers (also referred to as capacity planning) and the consolidation of existing servers (e.g., in the fields of orchestration and allocation). In "Anomaly detection model", we extend our analyses further to address anomaly detection problems in the field of capacity management. After server sizing, load testing, and server consolidation, the capacity planner must take into account the detection of potentially anomalous behavior of business-critical transactions to avoid degradation in the overall performance of a system.

An essential requirement for many learning algorithms is the existence of (labeled) training data to learn rules and patterns from historical observations. By means of the data, objective values are to be explained, enabling classification or regression. In capacity management, the main objective is to ensure acceptable levels of performance at the lowest costs taking into account possible anomalies in the required computational power. Strategies to achieve the objective currently rely on the implementation and testing of planned systems or their prototypes [24, 43]. Hence, performance is evaluated rather reactively when a system is about to go live [2]; engineers claim to require *something running to measure* to evaluate performance aspects of an application [66]. Furthermore, discovering and amending performance bottlenecks through measuring a running system is inefficient and expensive due to the high correction costs in the late stages of the development lifecycle [44, 65]. Therefore, model-based approaches were developed to rely on simulation engines or analytical solvers. However, such concepts are largely unused in practice due to their complexity, and personnel costs of modeling [4, 46]. Furthermore, the credibility of results remains questionable until validation in later stages. To combine the advantages of model-based approaches (early applicability) with the ease of measurement-based approaches, machine learning is proposed as an alternative technique to predict performance aspects of planned or managed systems in their design stage. Those predictions can also account for anomaly behaviors of the designated planned or managed systems in their design stage. Here, the main challenges lie in the data acquisition and data preparation phase to generate a sufficient amount of training data for subsequent learning of models.

In the domain of EAs, commercial-of-the-shelf (COTS) software is widely used [56, 64]. Consequently, a system similar to the one which is being planned most likely is already in production. Furthermore, EAs are required to be monitored continuously due to their exceptional significance for business continuity [49]. Therefore, observations that describe the system behavior under particular load and hardware conditions are present and may be used as training data for some machine learning techniques to suggest the required server sizing, load testing, server consolidation, and anomaly detection. The subsequent learning phase runs in a black-box manner without the need for costly expert knowledge. Error metrics that describe the model's accuracy provide clear indicators for the credibility of results. In order to evaluate the outlined strategy, extensive evaluation of different machine learning techniques for performance prediction, and anomaly detection in response time is conducted and presented in this paper. We further apply the investigated machine learning models that delivered sufficient quality to address different problems in the field of capacity management.

In "Related work", related work is studied by classifying existing means to predict EA performance. In addition, the utilization of machine learning techniques for anomaly detection in the field of EA is examined. Then, four typical scenarios of capacity management are introduced in "Capacity management scenarios". To address the scenarios, performance models, and anomaly detection models are to be constructed. In "Training data", we discuss data preparation and model training by means of three different machine learning techniques for performance prediction and four machine learning techniques for anomaly detection. All algorithms are trained using real monitoring data. The subsequent model evaluation, discussed in "Model evaluation", provides insights into the model accuracy and applicability for each technique to predict performance and detect anomalies in response time. In "Model application", the utility of the models is demonstrated on the basis of the introduced capacity management scenarios in which the models are applied. Finally, the proposed strategy is discussed in "Conclusion". A number of future research activities are outlined in "Limitations and future work" with intention to carry forward the novel field of model-based capacity management using machine-learning.

Related Work

The essential challenge of capacity management is to predict the performance of a studied system under varying load patterns [31]. The complex component-based architecture of EAs hampers to predict user performance as requests are affected by tightly integrated components on both the application and the database layer [71]. On a high level, the following two quantitative strategies may be carried out to evaluate EA performance [3, 12]:

- Measurement-based performance evaluation,
- Model-based performance evaluation.

The former requires the implementation of a system prototype. Its performance is then tested by means of scripts that generate workload in a configurable manner. The process is also referred to as benchmarking. Standardized benchmarks exist for different types of COTS EAs, e.g., the sales and distribution (SD) benchmark in the domain of SAP [62]. Relevant performance metrics, termed performance counters, are measured during the benchmark, e.g., by means of built-in monitors (software instrumentation), by distributed software agents, or by passive monitoring systems [31, 72]. In the IT infrastructure library (ITIL), this strategy is also referred to as simulation modeling.

Model-based performance evaluation, on the other hand, deals with the construction of models that represent the system behavior. Commonly, these models are distinguished regarding their solving technique. Simple models may be formulated mathematically in a closed form and, accordingly, are solved analytically. However, due to the complexity of today's EAs, the non-linear application behavior, in many cases, must be simulated, e.g., by means of queuing networks. The model-based strategy does not necessarily involve setting up real systems or prototypes but may require measured data for model calibration.

Hence, measurement-based approaches provide accurate and credible evaluation results but involve costly setups of prototypes. These are to be tested in a various number of load scenarios and configurations to identify cost-effective and suitable designs [16]. Model-based approaches overcome the effort of actual system implementation and testing but introduce modeling effort in terms of time and expert knowledge [10]. The latter refers to both modeling experts and software architects who are aware of the planned system, and its dependencies [68]. For this reason, model-based approaches that are solved analytically or on the basis of simulation represent white-box approaches. A popular white-box modeling technique is the Palladio component model (PCM), proposed by [4]. The technique covers the system architecture, its execution environment, and the usage profile in separate models, which are combined with a PCM instance. The instance may be transformed into analytical solvers or simulation models, which allow making performance predictions. The technique was used in the domain of EAs by [11]. In this domain, [69] proposed another white-box technique which particularly addresses SAP software. The approach relies on queuing networks that are built on the basis of monitoring data and must be calibrated in multiple iterations.

To reduce the costs of white-box modeling, different strategies evolved to reduce modeling effort. For example, [16, 17] built non-linear models while the coefficients are automatically obtained from existing measurement data. However, application-specific attributes were disregarded although known to significantly affect execution times [41, 55]. Furthermore, the model was tested using its own training data, which does not prove the model's capabilities of generalization to unseen data. A similar idea was followed by [33] who research on the automation of building PCM instances. As the approach requires access to the application's source code, [34, 35] developed an alternative approach which only requires bytecode access. Here, a reference platform is to be implemented whose measured performance metrics are used to feed a genetic algorithm (GA). The GA is used to translate user input into bytecode counts and, this way builds the basis of a platform-independent performance model. While the amount of required expertise is effectively reduced, an additional platform-dependent model is still to be constructed manually. Furthermore, application bytecode access may not be given. The detection of performance anomalies and changes in CPU demand patterns was automated by [18] by means of an analytical approach which uses regression. However, the prediction of actual execution times is not supported. The outlined approaches increase automation but still require considerable expert knowledge. Therefore, those are termed gray-box strategies.

In contrast, black-box strategies automate the construction of models by means of machine learning. The technique was successfully applied in the domains of job scheduling and load balancing to predict execution times or resource requirements of incoming requests [20, 29, 53, 67]. The application of traditional supervised and unsupervised machine learning techniques, classification and clustering, respectively, is relatively straightforward when using historical data for performance anomalies detection [32]. Such approaches are well suited for processing past performance metrics during overall system health analysis. For instance, clustering algorithms are useful in cases when no specific anomaly patterns are known in advance. Statistical analysis is performed on the historical data while grouping data points in related clusters. Data points that find themselves outside of the major clusters, outliers, are likely to be anomalies and should be analyzed additionally. This task is assisted by the anomaly detection specific algorithms such as isolation forest (IF) [40] or its modifications. While IF is not a new algorithm, it is shown that with certain modifications it is applicable to the solutions based on modern technologies such as containers [73].

Further analysis can be performed by applying classification [59] if specific types of anomalies are known. Anomalous data points can be compared with a pre-trained classification model to assist the deduction of the cause for analyzed anomalous events. It is worth noting that the previous examples are based on the batch analysis of historical data. These approaches can be useful to discover anomalous events in a later audit. For example, an approach based on recurrent neural networks (RNN) [30] allows one to predict the state of the system at any time point and compare it to the actual values. If the actual state of the system strays too far from the predicted, it can be considered anomalous. Of course, this is but a single example out of a variety of approaches for anomaly detection based on neural networks [36]. As the applicability and accuracy of machine-learning-based approaches strongly depend on the amount of training data, some of the related artifacts involve to feed knowledge bases continuously with historical observations [53, 67, 72]. However, model applicability is still limited by the variety of data. As concluded by [67], it is an open challenge to collect and prepare a sufficient amount of training data in a cost-effective way.

To address these drawbacks of machine-learning-based approaches, we propose an approach that leverages the dominating use of COTS software in the domain of EAs. A pre-analysis to this paper across 1.230 data centers, in the domain of SAP, has shown a mean customization degree of only 20% [48]. In other words, around 80% of the executed business transactions refer to standard features that are offered by the software vendor to a large group of customers. By means of software instrumentation, these executions are typically logged in the background and performance-related metrics are stored in the file system. These data are rarely analyzed statistically, although highly valuable for the outlined purpose [12]. Therefore, we argue for a domain-specific knowledge base to integrate measurement data of different environments that utilize the same type of COTS software. As investigated in our preliminary studies, these data may be used to train standardized performance models for the application in various capacity management scenarios [48, 51]. Hence, initial model training and evaluation surely involves expertise from the data science domain. However, required domain expertise with respect to the field of EA capacity management is significantly reduced when compared to existing white- and gray-box strategies. Furthermore, the approach scales with unknown hardware configurations and evaluated models may be used for a variety of use cases in different environments of the same COTS EA, e.g., sizing, load testing, and server consolidation. We refer to this technique as machine-learning-based using performance counters. In this paper, model training and application is demonstrated using real measurement data from productively operated EAs and by means of a scenario-based evaluation [27]. Table 1 summarizes the characteristics of the outlined techniques to evaluate EA performance.

Capacity Management Scenarios

Capacity management, in accordance with the ITIL service design, is typically carried out in a top–down manner on the following three layers [31]:

- Business capacity management,
- Service capacity management,
- Component capacity management.

It is at the heart of capacity management to provide a sufficient amount of capacity on the component level, which meets the requirements on the business level in a cost-effective manner. In other words, it is the objective to balance performance and costs. Hence, performance predictions are beneficial to answer various what-if questions within capacity management exercises [22, 41]. Input to these questions may be changes of

Table 1 Classification of techniques for predicting EA performance

Technique	Measurement-based	Model-based using simulation engines	Model-based using analytical solvers	Machine learning-based using performance counters
Model type examples	None (implemented prototype)	Queueing networks, Petri nets, Markov chains	Closed-form expressions such as (non)linear models	Random forests, support vector machines
Domain expertise	High	High (white-box modeling)	Medium (gray-box modeling)	Low (black-box modeling)
Training data	No model training	None or low amount to calibrate models	Few user-generated or measured top level parameters	Large amount of low level counters
Degrees of freedom	Low	High	Medium	High
Number of testable design alternatives	Limited to implemented designs	Limited to modeled designs	Limited to modeled designs	Models generalize to unseen designs
Analyzability	High	High	Medium	Low
Literature examples	[39, 44, 45, 62]	[4, 5, 8, 70]	[15, 17, 18]	[20, 22, 26, 29, 53, 67, 72]

workload or hardware characteristics. To demonstrate the utility of machine learning-based techniques which utilize performance counters, the following four scenarios will be addressed in the course of this paper:

Server sizing Before EAs are deployed, their capacity demands are to be estimated. Hence, for workload characteristics which are given on the business layer, suitable hardware components are to be identified. This process is defined as capacity planning [12] and practitioners also refer to it by the term sizing.

Load testing Another example, raised in the ITIL service design publication, refers to workload changes: "What if the throughput [of a service] doubles? [...] What will be the effect on the response times [...]?" [31, p. 173] Performance models allow to predict response times under varying workload conditions and, this way, to explore the borders of a given design.

Server consolidation The management of existing EAs involves to optimize the allocation of running services to servers in a periodical (offline) or continuous (online) manner. To improve utilization levels, services with orthogonal workloads are to be identified. Subsequent relocations, however, are subject to uncertainties since the effect on the service performance is rarely analyzed before solution deployment [51]. Here, machine-learning techniques which were trained on a large number of different hardware configurations may support decisions with respect to the expected quality of service performance.

Anomaly detection Performance analysis and prediction for enterprise applications are sufficient to manage normal operations. However, accounting for the events and behavior that fall outside of the enterprise application landscape's normal operational patterns is equally important. Such events are called anomalies. Anomaly detection is a mature but still active field of research. Anomalies in normal operations of an application or the entire landscape can vary in nature. Anomalies in the system behavior can be caused by security breaches, which can result in abnormal access patterns and transaction executions. Failing hardware or software errors can be the cause of anomalies in increased response times. Additionally, anomalies, specifically in response times, can be caused by sudden load spikes that fall outside of the planned capacity to a large margin. Such spikes are not malicious and not a sign of failing hardware but signify a shortcoming in planning and managing available capacity. Therefore, detection and analysis of anomalies is important in managing capacity and planning further advancement of the applications landscape and infrastructure.

Training Data

The accuracy of the performance model and anomaly detection model highly depends on the variety and quality of the training data. As argued earlier, running EAs often integrate software monitors to track business transaction performance. While the approach benefits from usage profiles that mainly rely on standard business transactions as opposed to custom ones, variety is achieved by a large number of observations on diverse infrastructure environments. The resulting data may undergo an extract-transform-load (ETL) process before being integrated into a central knowledge base that provides the model's attribute space. For this paper's evaluation, we made use of software instrumentation functionality where performance metrics are logged by the EA itself. In the area of SAP, so called statistical records track a number of performance metrics for each screen change, invoked by any user activity. These records include resource utilization metrics with respect to CPU and memory, as provided by the operating system via an additional middleware that typically runs once per server within an SAP landscape [60, 61]. Information on the underlying infrastructure and its capacity was collected as part of additional consulting activities. As a final step of the ETL process, the extracted data sets have been anonymized and structured in a comma-separated format to be loaded into a relational database schema that serves as the knowledge base. This way, real monitoring data from more than 18,000 running SAP instances were extracted. The data refer to application and database instances from different environments, each of them monitored for a period of up to 3 weeks. In total, more than 16,000 servers are covered, ensuring a large number of permutations of workload and hardware parameters.

In a study conducted by [41], several machine learning techniques were explored to facilitate EA execution times. The authors concluded that model accuracy improves if application-specific parameters are included in the training data. Accordingly, the complete set of utilized data comprises attributes from the following dimensions:

- Workload characteristics: A number of application-specific performance indicators are grouped by business transaction type for each monitored hour. Data include the number of dialog steps for each transaction as well as associated CPU time, queue time, lock time, and total response time.
- Resource capacity: Each hardware component is associated with a maximum CPU and memory capacity. While memory capacity is measured in Gigabytes, CPU capacity can be expressed in SAPS. It stands for SAP Application Performance Standard and is a popular metric to normalize the throughput of an SAP EA in business related terms. Therefore, SAPS are derived from a benchmark in the sales and distribution area and describe the number of screen changes a system can handle per hour. These screen changes are referred to as dialog steps. As an

example, a server which is associated with a throughput of 1000 SAPS, may handle up to 20,000 fully business processed order line items per hour [62].
- Meta data: Additional attributes characterize the measured data. For example, the year of the measurement may be important to represent basic server generations. Furthermore, system type labels allow to filter, e.g., for productive EAs only. This dimension also includes system topology information, that is the mapping of services to components.

Model construction was carried out on the basis of the Cross-industry standard process for data mining (CRISP-DM) [14]. Therefore, one of the early steps, before actual modeling, is to gain an understanding of the data and to prepare the data. As part of these activities, observations which describe abnormal application behavior (outliers) are to be excluded from the input data.

For example, maintenance operations such as backup jobs may interfere with performance. Preliminary investigations by [18] have shown a doubled error if typical maintenance time frames are present in the training data. Accordingly, we performed an initial outlier analysis which exposed end user dialog activity to occur typically during office hours while time periods between 11 pm and 4 am show the least dialog activity [48]. As these hours may provide suitable system maintenance slots and periods of increased batch activity, those were excluded from model training. Additional filters were applied to ensure a minimum CPU activity on both application and database instance, a minimum number of active users, and to exclude test measurements of less than 2 days. The target variable describes the hourly mean response time per dialog step, measured in ms. It ranges from 0 to 5000 ms and shows a standard deviation of 465.28. The mean value equals 571.87 ms.

As mentioned earlier, the data used for model training contain monitoring information for running SAP application instances. These instances are grouped into a number of potentially heterogeneous SAP system landscapes. Each system landscape is an individual permutation of specific applications and hardware, which in turn have differences in their workload profiles. However, as we mention in “Related work”, the majority of transactions running within such landscapes reflects standard enterprise application logic. Therefore, the behavior of these transaction types is expected to be comparable. We select and analyze only data that reflect the common transaction types present in all investigated system landscapes. This allows us to perform reasonable comparison, as well as homogenize training data contents across multiple landscapes.

Models were trained using different machine learning techniques. The techniques were selected with the objective to represent fundamentally varying strategies to investigate their suitability for the given use cases. As stated by [53], application behavior is hardly linear so that complex non-linear models which support recursive partitioning were focused. As a starting point, Support vector machines are known to provide feasible results, as tested, e.g., by [41, 53]. Random forests, as a representative of Bagging strategies, were proven to work well out of the box, too. Accordingly, in [13] identified remarkable results using Random forests in an empirical comparison of different ML techniques. Here, another ensemble technique, based on the Boosting strategy, obtained the best accuracy in most of the cases for predicting response time. Both Bagging and Boosting techniques combine a number of learners which are trained on subsets of the data. If Bagging is applied in a regression use case, the final prediction results from an average value across all learners. Hence, model training allows being parallelized. Boosting, on the other hand, follows a rather evolutionary process where weak learners focus on parts of the data which could not be explained by preceding learners. The following machine learning techniques are applied in this paper:

Performance prediction

- Support vector machines with Radial basis function (RBF) and with Polynomial kernel (PK)
- Random forests as representative of Bagging strategy
- AdaBoost as representative of Boosting strategy

Anomaly detection

- Isolation Forest as representative of application-specific clustering strategy
- Elliptic envelope
- One-Class Support vector machines with Radial basis kernel function
- Local Outlier Factor

For model training, we fed the three techniques with the cleaned set of training data. For different business transaction types, individual models were trained. For the sake of demonstration, we limited our analysis to the ten most frequently used business transactions within the module of sales and distribution. Model error metrics are provided in the following for each business transaction.

Model Evaluation

In this section, the evaluation of the performance model and the anomaly detection model is presented. We relied on different approaches based on the considered scenario. For evaluating the performance model, we relied on k-fold cross-validation on separate training sets with $k = 5$. However, anomaly detection usually is an unsupervised learning task. Especially when it comes to field evaluation, this task is usually required to be performed using unlabeled data as we present in this paper using real monitoring data. However, many anomaly detection algorithms are evaluated using pre-labeled datasets, such as KDD CUP 99 and similar datasets [19]. To evaluate the applicability of considered machine learning techniques for detecting anomalies in real workload datasets, we must rely on expert knowledge. For implementation and execution, we relied on the publicly available, open source reference algorithm implementations [54].

Performance Prediction Model

Typical error metrics that support the analysis of a model's accuracy include the mean absolute error (MAE), the mean absolute percent error (MAPE), the root mean squared error (RMSE), and the coefficient of determination (R^2). While the RMSE may be compared to the standard deviation within the training data, R^2 describes the relation between the distances of measured and predicted values and measured values and their average. Hence, the closer R^2 to one, the better the model. All mentioned metrics were calculated as part of model evaluation after appropriate hyperparameter optimization, feature selection, and normalization. Acceptable thresholds of errors are domain-specific and must be defined by the modeler for a specific use case. In capacity management, errors below 30% are generally acceptable for model-based strategies [1, 47].A simple way to compare model accuracy independent of the input data's magnitude, specific to regression use cases, was proposed by [6] as Regression error characteristics (REC) curve. It was also used, e.g., by [41]. The REC curve enables to read the frequency in which a defined threshold of error was met during model testing. Hence, the capacity manager may identify a tolerance level of the error on the *x*-axis of Fig. 1.

The *y*-axis shows the percentage of records which were predicted with an error below this value. The REC curve in Fig. 1 refers to the models for the most frequently used transaction type. In our data set, that is the transaction to change sales orders. As shown, the most accurate predictions were provided by AdaBoost. Here, the targeted level of error ($< 30\%$) was met in around 80% of the cases. Worst results were obtained using SVMs in conjunction with a polynomial kernel (MAPE = 39.30%). Here, the RBF performed significantly better with a MAPE of 25.90%. In our tests, Random forests provide acceptable accuracy without much tuning, using 500 trees and a subset of $p/3$ features which are tested to decide for a tree split where p represents the total number of features. In order to allow for the interpretation of the approach with respect to generalization, Table 2 shows the MAPE for the two best performing model types AdaBoost and Random forest, grouped by the ten most frequently used business transactions. While AdaBoost provides mean percentage errors below 30% in all cases, Random forest achieves the tolerance level in 3 out of 10 cases only. Consequently, a Boosting strategy which utilizes trees with a small depth (stumps) seems to be beneficial for predicting mean response times of EAs on the basis of monitoring data. As machine learning models are supposed to generalize to unseen data, the trained model may be used in various scenarios where performance is to be estimated.

Anomaly Detection Model

For detecting anomalies, we adopted four machine learning techniques, namely, Isolation forest [40], Elliptic envelope [57], Local outlier factor [9] and one-class SVM with nonlinear kernel (RBF) [63]. We evaluate the performance of these techniques for detecting anomalies by training two types of models for each.

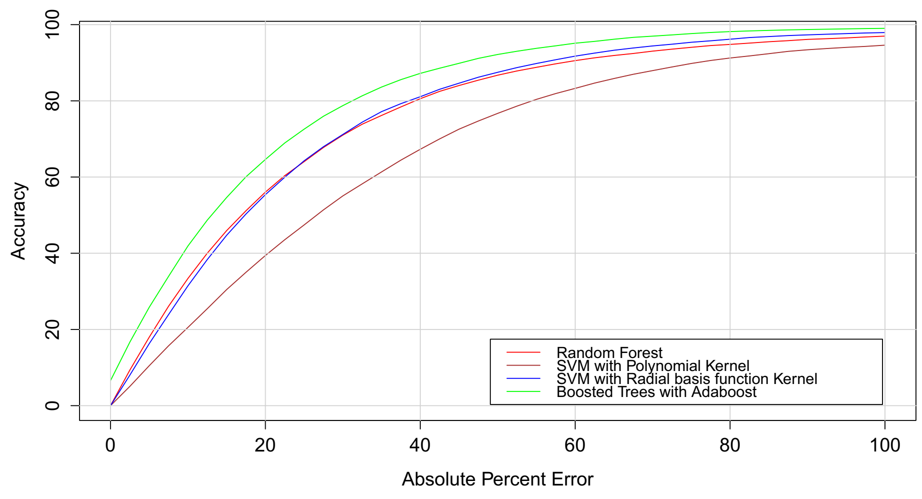

Fig. 1 REC curves for the business transaction to change sales orders

Table 2 Mean absolute percent errors

Business transaction	Random forest	AdaBoost
Create sales order	32.30	25.26
Change sales order	26.08	19.57
Display sales order	28.19	22.93
List of sales orders	26.44	21.95
Create billing document	39.04	28.66
Change billing document	38.97	29.34
Display billing document	33.74	25.75
Create outbound delivery	35.59	26.45
Change outbound delivery	36.25	27.01
Display outbound delivery	36.53	27.38

First, we construct *generalized models* which were trained on the monitoring data for all available system landscapes at the same time, while taking into account the data preparation described in “Training data”. This model type incorporates data not only from the specific single system landscape but expands the pool of statistical data for the models to the industry-wide level.

Second, we train the investigated algorithms to get models that are specific for each landscape and have no knowledge about any other system landscape (*single models*). The main narrative behind this analysis is to observe the percentage of intersected detected anomalies in both scenarios. A high percentage of intersections in the detected anomalies between the single model and the generalized model would increase the confidence in the obtained results. This implies that the applied machine learning techniques are trained on sufficient monitoring data to detect the anomalies in the desired system landscape successfully.

Additionally, different values for the relevant hyperparameters of the considered machine learning techniques are tested to draw conclusions about their utility. In this context, the tuned hyperparameters fundamentally control the threshold that clusters data points as normal or anomaly data points. Specifically, contamination for isolation forest, elliptic envelope, and local outlier factor. One-class SVM was tuned with ν hyperparameter.

The computational results are presented in Table 3. In the table we present the percentage of data points in the entire considered dataset that was detected as anomalous by our selected algorithms in generalized and single modes at different hyperparameter values. Additionally, we present the percentage of the total number of data points that are were commonly detected by both generalized and single models at each algorithm and hyperparameter combinations. The generalized and the single models that are based on Isolation forest and Elliptic envelope reported good results for detecting anomalies. 95.3% of the detected anomalies by the single model of Isolation forest are also detected by the generalized model of the same algorithm. Similarly, 97.4% of the detected anomalies by the single model of Elliptic envelope are also detected by the generalized model of the same algorithm. Oddly, the generalized model of the Local outlier factor algorithm results in a more restrictive detection of anomalies in contrast to Isolation forest and Elliptic envelope generalized models. In addition, the percentage of intersections in the detected anomalies in the generalized model and the single model is fairly low around 57%. This

Table 3 Comparison of the selected anomaly detection algorithms

Algorithm	Hyper-parameter	Generalized model (%)	Single model (%)	Common detected (%)	Intersection (%)
Isolation forest	0.1	14.31	10.01	9.55	95.41
	0.05	9.24	5.01	4.69	93.70
	0.01	1.89	1.01	0.96	94.81
	0.005	0.92	0.51	0.5	97.44
Elliptic envelope	0.1	12.89	10.01	9.63	96.19
	0.05	9.73	5.01	5.01	100
	0.01	2.22	1.01	1	98.7
	0.005	1.09	0.51	0.49	94.87
Local outlier factor	0.1	5.74	9.39	4.06	43.62
	0.05	2.34	4.76	1.46	30.58
	0.01	0.49	0.89	0.24	26.47
	0.005	0.25	0.43	0.13	30.30
One-class SVM (RBF kernel)	0.1	56.61	53.86	30.69	56.98
	0.05	47.25	39.21	19.22	49.01
	0.01	54.2	52.25	27.56	52.75
	0.005	44.15	57.48	27.48	47.82

implies that the nature of the detected anomalies by the generalized model is largely different from the one found by the single model. Therefore, the confidence in the result of this model cannot be guaranteed. Finally, the performance of the one-class SVM was not sufficient for detecting anomalies in the considered data.

Direct comparison of the anomalies detected by isolation forest and elliptic envelope algorithms indicates that both of the algorithms have a high degree of agreement. Results of such comparison for a single model are presented in Fig. 2 and for generalized model in 3. Specifically, 78.9% of anomalies that were detected by a single model of isolation forest were also detected by the single model of elliptic envelope. A higher degree of agreement was displayed by the generalized models, where 86.5% of the anomalies detected by the isolation forest, were also detected by the elliptic envelope.

We have conducted the hyperparameter evaluation, presented in Table 3 considering only the performance metrics, excluding the temporal features because analysis for anomaly detection performed only on a time series with a limited span of 2 weeks. It's important to note that the inclusion of the temporal features from the model did not result in radical differences in the detection of anomalies in the considered evaluation dataset. The percentage of detected anomalies had no significant differences in both cases. The difference between the sets of discovered anomalous data points did not exceed 4%. The only exception is isolation forest at 0.1 contamination value in a single model, where the difference between considering and not considering temporal features led to up to 34% difference in the resulted datasets, however, in both cases, the number of detected anomalies was excessively high, rendering the model impractical to use for any meaningful analysis. However, as can be seen from Figs. 2 and 3 the considered workload clearly correlates to office hours. Interestingly, if we consider a workload with a highly regular pattern, such as an automated task executed on a schedule.

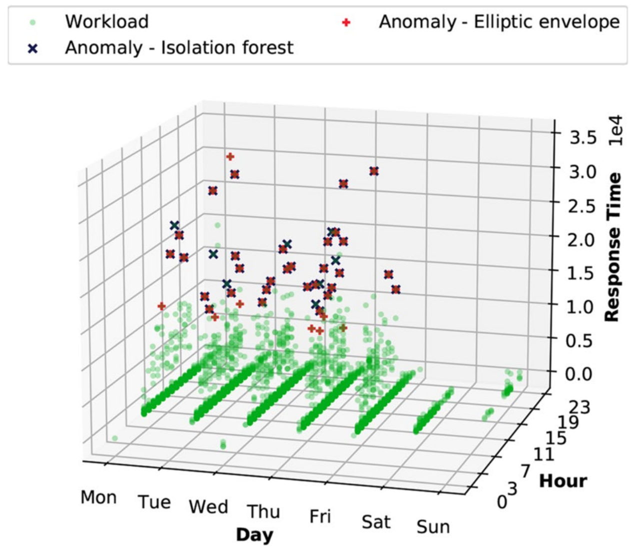

Fig. 2 Compare isolation forest with elliptic envelope: Single model

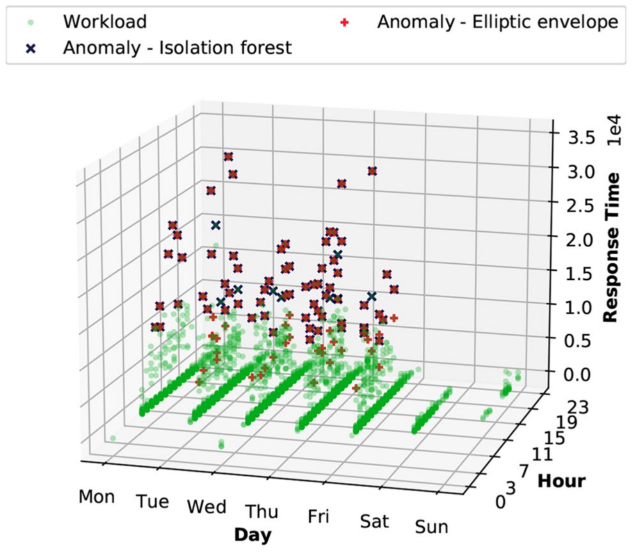

Fig. 3 Compare isolation forest with elliptic envelope: generalized model

It is clear that the isolation forest, *without* considering temporal features in highly regular workloads managed to detect data points that clearly located outside of the regular response time values, while the inclusion of these features resulted in detection of false positive.

Exclusion of the temporal features in elliptic envelope led to failure in detecting any anomalies at all for the given contamination value of 0.005. However, increasing the contamination value for elliptic envelope without including temporal features, resulted in the detection values similar to these of discussed above for isolation forest. Inclusion of temporal features again led to detection of many false positives.

From this evaluation of inclusion or exclusion of temporal features, we can draw the conclusion that the inclusion of these features in anomaly detection limited to a short time series is counterproductive. There is not enough statistical data within our use case in a time series spanning 2 weeks for the considered algorithms to establish strong time bound relationships.

Model Application

Performance Prediction Model

To investigate the utility and applicability, we applied the trained models in three capacity management scenarios, which were introduced in "Capacity management scenarios".

The following subsections exemplify practical use cases for each of the scenarios.

Server sizing As part of the service design, it is the objective of the sizing process to identify sufficient hardware components for given business requirements. We illustrate this scenario using the example of an SAP ERP system, which is distributed vertically on an application server and a database server. In the current server configuration, the system provides mean response times around 800 ms under regular load conditions (load factor = 1). However, in times of load peaks, response times increase to more than 1.400 ms. To address this issue, the capacity planner invokes a sizing process which considers a number of alternative configurations.

The decision maker is interested to know the effect of certain hardware upgrades on both the application and the database layer before their actual deployment. If measurement-based and classical model-based approaches are too expensive due to the setup of a test system and the costs of modeling, a machine learning-based approach may serve as a cost-effective black-box approach. For the sake of demonstration, we transformed the alternative hardware configurations into features of the previously trained models. Mean prediction results are shown in Fig. 4 and allow for comparison of change effects. In accordance with historical load profiles and future business requirements, upgrade decisions may be supported. For example, the concept of a Pareto front may be utilized to identify non-dominated solutions which maximize performance and minimize investment costs.

Load testing Similar to the sizing scenario, alternative load profiles may represent input features to the prediction model. Since performance-related service level agreements (SLA) typically invoke penalty costs for requests which do not meet performance objectives, the prediction results can be utilized to minimize the number (and costs) of SLA violations. To demonstrate the scenario, we define a sample percentile-based SLA which invokes penalty costs c_{sla} of 150 $ for every percentage point of requests which were not processed within less than one second. An additional deadline constraint d_2 requires all requests to be processed within 1.500 ms; respective designs that cause higher response times are to be rejected.

To predict the expected level of SLA fulfillment, we applied the performance models on historical log data with varying load attributes. As a result, Table 4 lists penalty costs which may be associated with each of the load factors. If deadline constraint d_2 applies, load factors of 2 and above cannot be served by the design under study. Hence, additional design alternatives are to be considered. On the basis of the prediction results, Fig. 5 shows the expected number of SLA violations for each of the designed alternatives for historical (load factor = 1) and alternative load scenarios. Each design alternative is characterised by the amount of SAPS it may handle (cf. "Training data"), as used

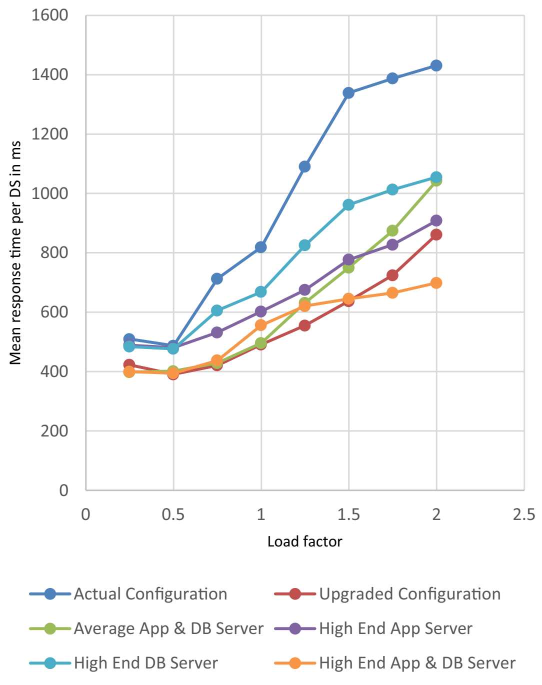

Fig. 4 Predicted mean levels of performance for alternative server designs

Table 4 Penalty costs for alternative load factors

Load factor	Violated hours (d_2)	Fulfillment degree	Violation degree	Penalty costs
1	0	99.32	0.68	0
1.25	0	98.47	1.53	229.5
1.5	0	94.04	5.96	894
1.75	0	73.94	26.06	3,909
2	1	42.59	57.41	8,611.5
2.25	2	25.21	74.79	11,218.5

in SAP-based environments to describe a system's throughput in a hardware agnostic manner. As can be seen, additional capacity generally reduces the risk of SLA violations. However, in our example, improvements are more significant for higher load scenarios. Hence, decisions are to be made in accordance with the business strategy and expected levels of future workload [48].

Server consolidation In the course of server consolidation efforts, service relocation may affect overall service performance. In this regard, the evaluated performance model supports questions similar to the following ITIL example: "What if service B is moved from the current server onto a new server? What will be the effect on the response times [...]?" [31, p. 173] Server consolidation problems are known to be NP-hard; they typically introduce a large solution space

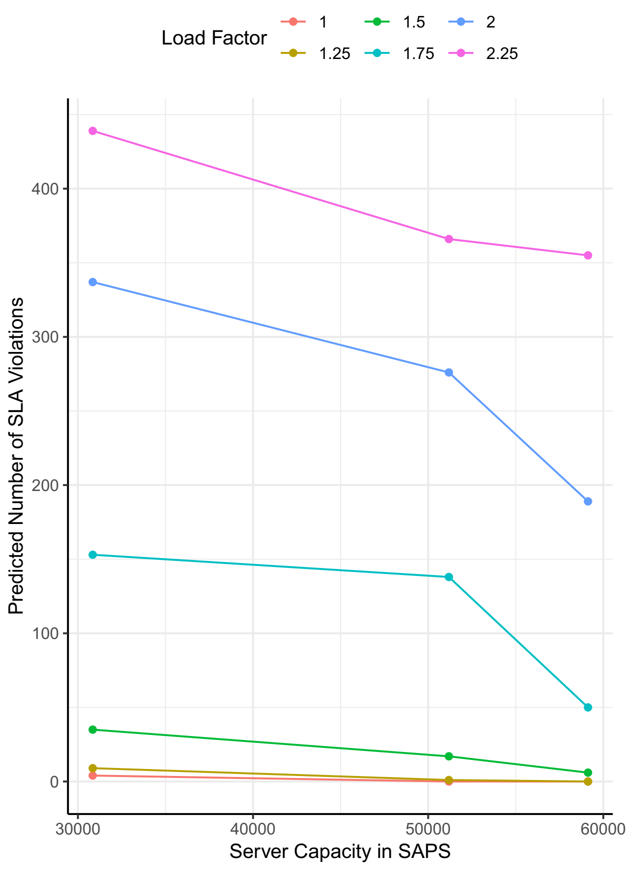

Fig. 5 Prediction of SLA violations for alternative designs

which cannot be explored manually [7, 50]. Hence, solution candidates are identified, e.g., by means of heuristics and metaheuristics. Models, as constructed in "Training data", allow to integrate performance aspects into the problem formulation. A simple example is illustrated in Fig. 6.

Here, three design alternatives were computed by a metaheuristic, which belongs to the class of grouping genetic algorithms (GGA), as part of a preceding field study, presented in [48]. To test the performance effects of these solution candidates before their deployment, we fed our performance model with the resulting allocations of services (and their workload profiles) to servers (and their hardware characteristics). In Fig. 6, Design A represents a solution of lowest capacity and high server utilization. In contrast, Design C provides highest amount of capacity with a rather low mean utilization. In terms of utilization and capacity, Design B lies between A and C. According to the box plots in Fig. 6, higher capacity leads to more stable designs with less variance across the predicted response times. Design C seems to be beneficial, particularly for higher workloads, and reliably catches load peaks. The scenario demonstrates how machine-learning enables to integrate additional decision values into server consolidation problems. Illustrations similar to the one in Fig. 6 effectively reveal change effects. As a result, a great amount of uncertainties is eliminated and decisions remain subject to the risk attitude of the decision maker.

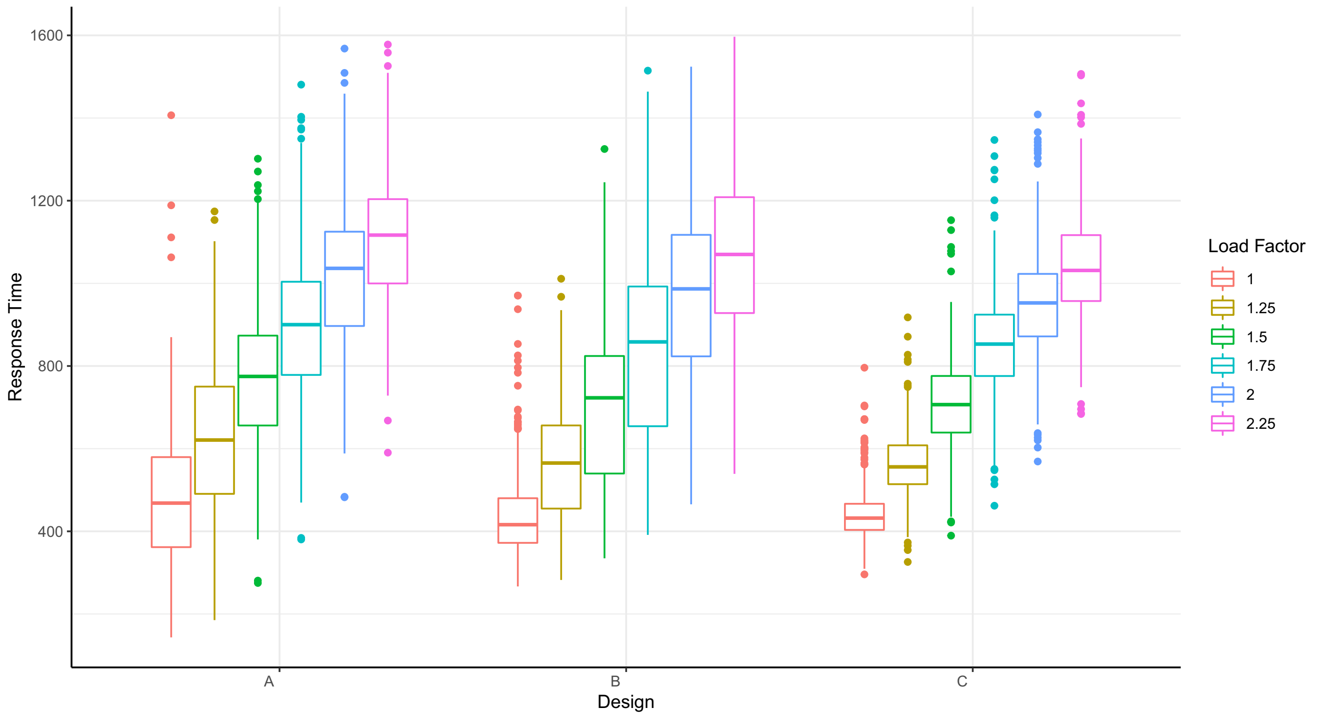

Fig. 6 Consolidation alternatives and their effect on the service performance

Anomaly Detection Model

A typical anomaly detection application is to discover malicious attacks on the IT infrastructure or possible hardware failures. In addition, anomaly detection can be applied within the context of capacity management for standard enterprise applications to avoid sudden degradation in performance. Anomalous peaks in response time for different services and transactions should be evaluated to design the appropriate reactive measures. Even within a short observation period, a peak that occurs in a specific day and time, can be a single-time event or, in fact, a rare but recurrent spike. The use of anomaly detection algorithms allows for clear identification of such spikes at various tasks running within the landscape, as well as a correlation between the response time duration between these.

In Fig. 7, we demonstrate the application of the designed anomaly detection approach on the top transaction type. From this representation, we can clearly observe a detected anomalous spike in response time using, for instance, Isolation forest. We also observed similar behaviour in five more top transaction types in the same landscape. A clear correlation between the increase of response time for different transaction types, most likely indicates a design problem, which requires the attention of the capacity planner to avoid degradation in the overall performance of the system.

The suggested anomaly detection model can be eventually applied to all business-critical transactions to avoid degradation in performance. Detected anomalies in the system can be used by the capacity planner in the forms of warning notifications if those spikes exceed some predefined threshold of response time. Eventually, an increase in response time might lead to violations in the signed SLAs. Based on the suggested system landscape design, the capacity planner need to react to such reoccurring peaks in the workload through, for instance, triggering a new server sizing process.

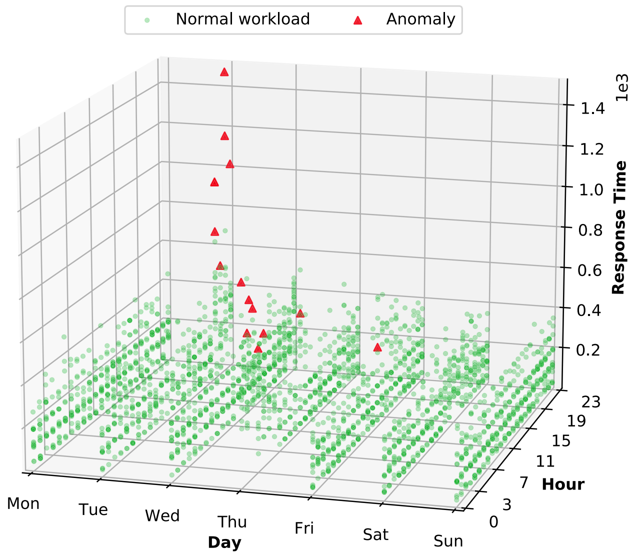

Fig. 7 Application of anomaly detection on the top transaction type in a landscape

Conclusion

In this paper, we investigated various machine learning techniques to address different issues in the field of capacity management. Our two-fold analysis showed a successful utility of different machine learning techniques to predict performance and detect anomalies in response time of EAs. In a scenario-based evaluation, the application of machine learning-based techniques could be demonstrated for the domain of service capacity management. Performance model as well as anomaly detection model that are trained on a variety of monitoring data are evaluated to provide sufficient accuracy to support decisions in the field of sizing, load testing, server consolidation, and anomaly detection. For COTS EAs, a Boosting strategy has proven to depict existing patterns in the data most effectively. As for anomaly detection, Isolation forest and elliptic envelop delivered correlated results between the generalized model and single model, which increase the confidence in the obtained results. In addition, a general agreement on the detected anomalies through both techniques is observed, which in turn validates more with expert knowledge the anomalous nature of the detected points.

Across ten frequently used business transactions, AdaBoost has constantly shown mean absolute percent errors below 30% when predicting the mean hourly response time per dialog step for SAP enterprise applications. In addition, Isolation forest is utilized to detect anomalies of response time in all transactions and presented exemplary results for six top used transactions. Due to the black-box nature of the approach, costs of classical white-box modeling are eliminated, and the dominating measurement-based approaches are complemented by a less expert-driven and more data-driven strategy. Therefore, the findings may be utilized by consultancy organizations to establish cost-effective and profound capacity management services.

Limitations and Future Work

In the investigated scenarios, the performance was analyzed on the level of business transactions. Hence, the most relevant transaction types are to be provided by the decision maker. It is subject to future research to generalize the transaction-based approach to higher levels such as application instance or overall system performance. However, for detecting anomalies of response time, all transactions are analyzed. Furthermore, the performance prediction technique may be integrated into well-known problems in the domain of capacity management. For example, prediction models may be applied by metaheuristics during fitness evaluation of alternative designs. As for anomaly detection, the current analysis is rather designed and can be used as a reactive measure to avoid major degradation in the overall performance of the system. It is of a major interest to extend the current implementation to address real-time anomaly detection on standard enterprise applications. We encourage other researchers and practitioners to push forward the outlined research direction to make ML-based performance evaluation to become an integral part of various capacity management exercises.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of interest On behalf of all authors, Hendrik Müller states that there is no conflict of interest.

References

1. Almeida VA (2002) Capacity planning for web services techniques and methodology. In: IFIP international symposium on computer performance modeling. Measurement and evaluation. Springer, Berlin, pp 142–57.
2. Balsamo S, Marco AD, Inverardi P, Simeoni M. Model-based performance prediction in software development: a survey. IEEE Trans Software Eng. 2004;30(5):295–310.
3. Becker S, Grunske L, Mirandola R, Overhage S. Performance prediction of component-based systems. In: Architecting systems with trustworthy components. Berlin: Springer; 2006. p. 169–92.
4. Becker S, Koziolek H, Reussner R. The palladio component model for model-driven performance prediction. J Syst Softw. 2009;82(1):3–22.
5. Bertolino A, Mirandola R (2004) Cb-spe tool: putting component-based performance engineering into practice. In: International symposium on component-based software engineering. Springer, pp 233–48.
6. Bi J, Bennett KP (2003) Regression error characteristic curves. In: Proceedings of the 20th international conference on machine learning (ICML-03), pp 43–50.
7. Bichler M, Setzer T, Speitkamp B (2006) Capacity planning for virtualized servers. In: Workshop on information technologies and systems (WITS), Milwaukee, WI, USA.
8. Bondarev E, de With P, Chaudron M, Muskens J (2005) Modelling of input-parameter dependency for performance predictions of component-based embedded systems. In: 31st EUROMICRO conference on software engineering and advanced applications, IEEE, pp 36–43.
9. Breunig MM, Kriegel HP, Ng RT, Sander J. Lof: identifying density-based local outliers. SIGMOD Rec. 2000;29(2):93–104.
10. Brosig F, Huber N, Kounev S. Architecture-level software performance abstractions for online performance prediction. Sci Comput Program. 2014;90:71–92.
11. Brunnert A, Krcmar H. Continuous performance evaluation and capacity planning using resource profiles for enterprise applications. J Syst Softw. 2015;123:239–62.
12. Brunnert A, van Hoorn A, Willnecker F, Danciu A, Hasselbring W, Heger C, Herbst N, Jamshidi P, Jung R, von Kistowski J, et al (2015) Performance-oriented devops: a research agenda. arXiv:150804752.
13. Caruana R, Niculescu-Mizil A (2006) An empirical comparison of supervised learning algorithms. In: Proceedings of the 23rd international conference on Machine learning, ACM, pp 161–8.
14. Chapman P, Clinton J, Kerber R, Khabaza T, Reinartz T, Shearer C, Wirth R (2000) Crisp-dm 1.0 step-by-step data mining guide. resreport, The CRISP-DM consortium. https://www.the-modeling-agency.com/crisp-dm.pdf.
15. Chaudhuri S, Narasayya V, Ramamurthy R (2004) Estimating progress of execution for SQL queries. In: Proceedings of the 2004 ACM SIGMOD international conference on Management of data, ACM, pp 803–14.
16. Chen S, Gorto, Liu A, Liu Y (2002) Performance prediction of cots component-based enterprise applications. In: ICSE workshop. 5th CBSE, Citeseer.
17. Chen S, Liu Y, Gorton I, Liu A. Performance prediction of component-based applications. J Syst Softw. 2005;74(1):35–43.
18. Cherkasova L, Ozonat K, Mi N, Symons J, Smirni E. Automated anomaly detection and performance modeling of enterprise applications. ACM Trans Comput Syst. 2009;27(3):6.
19. Divekar A, Parekh M, Savla V, Mishra R, Shirole M (2018) Benchmarking datasets for anomaly-based network intrusion detection: Kdd cup 99 alternatives. In: 2018 IEEE 3rd international conference on computing. IEEE: communication and security (ICCCS), pp 1–8.
20. Duan R, Nadeem F, Wang J, Zhang Y, Prodan R, Fahringer T (2009) A hybrid intelligent method for performance modeling and prediction of workflow activities in grids. In: Proceedings of the 2009 9th IEEE/ACM international symposium on cluster computing and the grid, IEEE Computer Society, pp 339–47.
21. Frey CB, Osborne MA. The future of employment: how susceptible are jobs to computerisation? Technol Forecast Soc Chang. 2017;114:254–80.
22. Ganapathi A, Kuno H, Dayal U, Wiener JL, Fox A, Jordan M, Patterson D (2009) Predicting multiple metrics for queries: better decisions enabled by machine learning. In: Data engineering, 2009. ICDE'09. IEEE 25th international conference on, IEEE, pp 592–603.
23. Gimpel H, Röglinger M (2015) Digital transformation: changes and chances-insights based on an empirical study. Resreport, Fraunhofer Institute for Applied Information Technology. https://eref.uni-bayreuth.de/29908/.
24. Gmach D, Krompass S, Scholz A, Wimmer M, Kemper A. Adaptive quality of service management for enterprise services. ACM Trans Web. 2008;2(1):8.
25. Goudarzi H, Ghasemazar M, Pedram M (2012) Sla-based optimization of power and migration cost in cloud computing. In:

Proceedings of the 2012 12th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGRID 2012), IEEE Computer Society, pp 172–9.
26. Gupta C, Mehta A, Dayal U (2008) PQR: Predicting query execution times for autonomous workload management. In: Autonomic computing, 2008. ICAC'08. International conference on, IEEE, pp 13–22.
27. Hevner AR, March ST, Park J, Ram S. Design science in information systems research. Manag Inf Syst Q. 2004;28(1):75–105.
28. Heyme R, Menge AM (2017) Digitalisierung in Sachsen-Anhalt erfolgreich gestalten. https://library.fes.de/pdf-files/bueros/sachsen-anhalt/13749.pdf. Accessed 14 Jan 2019.
29. Huang L, Jia J, Yu B, Chun BG, Maniatis P, Naik M (2010) Predicting execution time of computer programs using sparse polynomial regression. In: Advances in neural information processing systems, pp 883–91.
30. Huch F, Golagha M, Petrovska A, Krauss A (2018) Machine learning-based run-time anomaly detection in software systems: An industrial evaluation. In: IEEE workshop on machine learning techniques for software quality evaluation (MaLTeSQuE), IEEE, pp 13–8.
31. Hunnebeck L, Rudd C, Lacy S, Hanna A (2011) ITIL service design. The Stationery Office (TSO).
32. Ibidunmoye O, Hernández-Rodriguez F, Elmroth E. Performance anomaly detection and bottleneck identification. ACM Comput Surv. 2015;48(1):1–35.
33. Kappler T, Koziolek H, Krogmann K, Reussner RH. Towards automatic construction of reusable prediction models for component-based performance engineering. Softw Eng. 2008;121:140–54.
34. Krogmann K, Kuperberg M, Reussner R. Using genetic search for reverse engineering of parametric behavior models for performance prediction. IEEE Trans Softw Eng. 2010;36(6):865–77.
35. Kuperberg M, Krogmann K, Reussner R (2008) Performance prediction for black-box components using reengineered parametric behaviour models. In: International symposium on component-based software engineering. Springer, pp 48–63.
36. Kwon D, Kim H, Kim J, Suh SC, Kim I, Kim KJ. A survey of deep learning-based network anomaly detection. Clust Comput. 2019;1–13.
37. Lasi H, Fettke P, Kemper HG, Feld T, Hoffmann M. Industry 4.0. Bus Inf Syst Eng. 2014;6(4):239–42.
38. Lee J, Kao HA, Yang S. Service innovation and smart analytics for industry 4.0 and big data environment. Proced Cirp. 2014;16:3–8.
39. Lilja DJ. Measuring computer performance: a practitioner's guide. Cambridge: Cambridge University Press; 2005.
40. Liu FT, Ting KM, Zhou ZH (2008) Isolation forest. In: 2008 eighth IEEE international conference on data mining, IEEE, pp 413–22.
41. Matsunaga A, Fortes JA (2010) On the use of machine learning to predict the time and resources consumed by applications. In: Proceedings of the 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing, IEEE Computer Society, pp 495–504.
42. Matt C, Hess T, Benlian A. Digital transformation strategies. Bus Inf Syst Eng. 2015;57(5):339–43.
43. Menascé DA. Automatic QOS control. IEEE Internet Comput. 2003;7(1):92–5.
44. Menascé DA. Composing web services: a QOS view. Internet Comput IEEE. 2004;8(6):88–90.
45. Menascé DA, Almeida VA. Capacity planning for web services: metrics, models, and methods. Upper Saddle River: Prentice Hall; 2002.
46. Menascé DA, Ngo P (2009) Understanding cloud computing: experimentation and capacity planning. In: International conference on CMG conference.
47. Menascé DA, Almeida VA, Dowdy LW, Dowdy L. Performance by design: computer capacity planning by example. Upper Saddle River: Prentice Hall Professional; 2004.
48. Müller H (2019) Multi-dimensional server consolidation for commercial off-the-shelf enterprise applications using shared performance counters. PhD thesis, Otto von Guericke University Magdeburg.
49. Müller H, Turowski K (2015) Big data on performance logs—a collaborative monitoring cloud for ERP systems. In: Proceedings on the international conference on internet computing (ICOMP), the steering committee of the world congress in computer science, computer engineering and applied computing (WorldComp), p 75.
50. Müller H, Bosse S, Turowski K (2016) Optimizing server consolidation for enterprise application service providers. In: Proceedings of the 2016 Pacific Asia conference on information systems.
51. Müller H, Bosse S, Turowski K. Capacity management as a service for enterprise standard software. Complex Syst Inform Model Q. 2017;2:1–21.
52. Müller H, Bosse S, Turowski K (2019) On the utility of machine learning for service capacity management of enterprise applications. In: 2019 15th international conference on signal-image technology and internet-based systems (SITIS), IEEE, pp 274–81.
53. Niehorster O, Krieger A, Simon J, Brinkmann A (2011) Autonomic resource management with support vector machines. In: 2011 12th IEEE/ACM international conference on grid computing (GRID), IEEE, pp 157–164.
54. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: machine learning in python. J Mach Learn Res. 2011;12:2825–30.
55. Pinheiro E, Bianchini R, Carrera EV, Heath T (2001) Load balancing and unbalancing for power and performance in cluster-based systems. Tech. rep, Rutgers University.
56. Pollock N, Williams R, Procter R. Fitting standard software packages to non-standard organizations: the ɓiographyóf an enterprise-wide system. Technol Anal Strat Manag. 2003;15(3):317–32.
57. Rousseeuw PJ, Driessen KV. A fast algorithm for the minimum covariance determinant estimator. Technometrics. 1999;41(3):212–23. https://doi.org/10.1080/00401706.1999.10485670.
58. Rüßmann M, Lorenz M, Gerbert P, Waldner M, Justus J, Engel P, Harnisch M (2015) Industry 4.0: the future of productivity and growth in manufacturing industries. Boston Consulting Group 9 .
59. Salman T, Bhamare D, Erbad A, Jain R, Samaka M (2017) Machine learning for anomaly detection and categorization in multi-cloud environments. In: 2017 IEEE 4th international conference on cyber security and cloud computing (CSCloud), IEEE, pp 97–103.
60. SAP (2015) Workload monitor (st03 or st03n). https://wiki.scn.sap.com/wiki/pages/viewpage.action?pageId=17472. Accessed 04 May 2021.
61. SAP (2021a) Os collector saposcol. https://help.sap.com/viewer/9a9875f060404c44a84e2bad1181f80f/LATEST/en-US/4b63175094c64869e10000000a44176f.html. Accessed 04 May 2021.
62. SAP (2021b) Sap benchmark glossary. https://www.sap.com/about/benchmark.html. Accessed 04 May 2021.
63. Schölkopf B, Smola AJ, Williamson RC, Bartlett PL. New support vector algorithms. Neural Comput. 2000;12(5):1207–45. https://doi.org/10.1162/089976600300015565.

64. Somers TM, Nelson K (2001) The impact of critical success factors across the stages of enterprise resource planning implementations. In: System sciences, 2001. Proceedings of the 34th annual Hawaii international conference on, IEEE, p 10.
65. Tertilt D, Krcmar H (2011) Generic performance prediction for ERP and SOA applications. In: ECIS.
66. Tudenhöfner E (2011) Integration of performance management into the application lifecycle. diplom.de.
67. Venkataraman S, Yang Z, Franklin M, Recht B, Stoica I (2016) Ernest: efficient performance prediction for large-scale advanced analytics. In: 13th USENIX symposium on networked systems design and implementation (NSDI 16), pp 363–78.
68. Westermann D, Happe J, Hauck M, Heupel C (2010) The performance cockpit approach: a framework for systematic performance evaluations. In: Software engineering and advanced applications (SEAA), 2010 36th EUROMICRO conference on, IEEE, pp 31–8.
69. Wilhelm K (2001) Capacity planning for sap-concepts and tools for performance monitoring and modelling. CMG J Comput Resour Manag.
70. Wilhelm K (2003) Messung und modellierung von sap r/3-und storage-systemen für die kapazitätsplanung. PhD thesis, University of Duisburg-Essen.
71. Williams LG, Smith CU (1998) Performance evaluation of software architectures. In: Proceedings of the 1st international workshop on software and performance, ACM, pp 164–177.
72. Yoo W, Larson K, Baugh L, Kim S, Campbell RH. ADP: automated diagnosis of performance pathologies using hardware events. ACM SIGMETRICS Perform Eval Rev. 2012;40(1):283–94.
73. Zou Z, Xie Y, Huang K, Xu G, Feng D, Long D. A docker container anomaly monitoring system based on optimized isolation forest. IEEE Trans Cloud Comput. 2019;20:19.