

A Machine Learning based Context-aware Prediction Framework for Edge Computing Environments

Abdullah Fawaz Aljulayfi^{1,2}^a and Karim Djemame¹^b

¹*School of Computing, University of Leeds, Leeds, U.K.*

²*Prince Sattam Bin Abdulaziz University, K.S.A.*

Keywords: Edge Computing, Self-adaptive Systems, Machine Learning, Prediction Framework, Linear Regression, Support Vector Regression, Neural Networks, Sliding Window.

Abstract: A Context-aware Prediction Framework (CAPF) can be provided through a Self-adaptive System (SAS) resource manager to support the autoscaling decision in Edge Computing (EC) environments. However, EC dynamicity and workload fluctuation represent the main challenges to design a robust prediction framework. Machine Learning (ML) algorithms show a promising accuracy in workload forecasting problems which may vary according to the workload pattern. Therefore, the accuracy of such algorithms needs to be evaluated and compared in order to select the most suitable algorithm for EC workload prediction. In this paper, a thorough comparison is conducted focusing on the most popular ML algorithms which are Linear Regression (LR), Support Vector Regression (SVR), and Neural Networks (NN) using real EC dataset. The experimental results show that a robust prediction framework can be supported by more than one algorithm considering the EC contextual behavior. The results also reveal that the NN outperforms LR and SVR in most cases.

1 INTRODUCTION

The EC paradigm has emerged to support the Internet of Things (IoT) applications by pushing the computational capabilities towards the edge of the network (Dolui and Datta, 2017; Shi and Dustdar, 2016). Such support requires the efficient management of edge resources to fulfill the IoT applications' requirements such as mobility and low latency. However, the EC resource management process is not a trivial task because of the nature of EC and the rapid increase in the number IoT devices which is estimated to be 41.6 billion devices (Framingham, 2019).

The SASs have seen a significant level of interest in different research areas like autonomic computing and pervasive computing and provide self-management properties and exhibit system properties such as self-awareness to achieve adaptation (Kavanagh et al., 2019). They can monitor resources, state and behavior. Therefore, a SAS is a promising solution to efficiently support the resource management automation in EC as it can adjust itself according to the operation environment (Arcaini et al., 2015; D'Angelo,

2018; Kavanagh et al., 2019; Kramer and Magee, 2007; Krupitzer et al., 2015; Singh and Chana, 2015; Xu and Buyya, 2019). Such adaptation can be either proactive whereby the SAS uses the historical data to forecast the future system behavior or changes in the environment (Al-Dhuraibi et al., 2018; Galante and De Bona, 2012; Moreno-vozmediano et al., 2019), reactive whereby the system is adjusted in real-time by continually monitoring the system behavior and operational environment, or hybrid whereby the system uses both reactive and proactive approaches.

In a proactive adaptation, designing a robust prediction framework for forecasting EC workload and supporting auto-scaling is challenging (Delicato et al., 2017; Gupta et al., 2017; Kaur et al., 2017; B. Liu et al., 2020; Toczé and Nadjm-Tehrani, 2018). An EC environment exhibits a dynamic workload and often has limited resources. In order to design such framework, a deep understanding of EC nature (e.g. workload patterns and users' behavior) and a thorough investigation of the workload prediction methods, their characteristics and impact on their accuracy are required (Islam et al., 2012; Nikravesh et al., 2015b).

^a <https://orcid.org/0000-0002-2262-2340>

^b <https://orcid.org/0000-0001-5811-5263>

The advantages of these activities are twofold: 1) to support the IoT applications' Quality of Service (QoS) by avoiding under-provisioning (Ajila and Bankole, 2013; Aldossary and Djemame, 2018; Calheiros et al., 2015; Kumar and Singh, 2018; Lorigo-Botran et al., 2014; Moreno-vozmediano et al., 2019), and 2) to efficiently utilize the EC resources thus avoiding over-provisioning and improving the system's scalability.

Altogether, a timely research challenge is the design of the CAPF for forecasting EC workload. This paper extends our previous work on the SAS Architecture (Aljulayfi and Djemame, 2019) and focuses on the proactive adaptation support. Further, it analyses a real EC dataset from Shanghai Telecom (Sguangwang.com, 2018) in order to identify workload patterns and propose the most suitable workload prediction model. Moreover, it compares the accuracy of the most well-known ML algorithms: LR, SVR, and NN which includes investigating the effect of window size. Finally, the CAPF is designed in accordance to the investigations' results. The main contributions of this paper are summarized as follows:

- (C1) An analysis of a real EC workload dataset is performed in order to understand the EC workload pattern and train the ML prediction models.
- (C2) A comparison of the most well-known ML algorithms' accuracy considering the window size effect is conducted aiming towards workload prediction framework.
- (C3) A design of CAPF through SAS to support auto-scaling decision using the most accurate and suitable ML prediction algorithms is presented.

The remainder of this paper is organized as follows: Section 2 discusses the related work. This is followed by Section 3 which presents the SAS architecture. Section 4 shows the research methodology. The results and discussion are in Section 5. Section 6 illustrates the CAPF. Finally, the paper's conclusion and future work in Section 7.

2 RELATED WORK

This section discusses the proposed proactive adaptation models to support auto-scaling systems, which can be classified into resource utilization- and workload-based. The resource utilization-based studies predict resource utilization e.g. CPU utilization to support the auto-scaling decision. A considerable body of research adopts this method. For example, a CPU-utilization prediction model using the Regression-Markov chain (RMC) method targeting the applications' QoS is proposed in (Li et al., 2018). The

results show that the RMC provides better accuracy as compared to LR due to large fluctuation and randomness. Some other studies adopt ML methods. Three prediction models for CPU-utilization, throughput, and response time for Cloud Computing (CC) are proposed by (Bankole and Ajila, 2013) using a synthetic linear workload. Furthermore, LR, SVR, and NN ML methods are used. The results show that the SVR outperforms LR and NN in predicting both CPU-utilization and throughput whereas the NN outperforms other methods in predicting the response time. This work is extended in (Ajila and Bankole, 2013) by considering the random workload pattern.

In (Islam et al., 2012) a synthetic linear workload pattern is generated in order to develop prediction models to support scaling decisions. Moreover, this work compares the accuracy of LR and NN with and without the sliding window consideration. It reports that the sliding window has a positive impact on the models' accuracies. The effect of the NN on the auto-scaling decision technique is also evaluated using a threshold and compared with SVR (Nikravesh et al., 2014). Additionally, an investigation is conducted to select the best proportion of the dataset split considering e.g. 50%/50% for training/testing. This work is extended in (Nikravesh et al., 2015a) and aims to investigate the effect of different workload patterns (i.e. growing, periodic, and unpredicted). Besides the sliding window technique is considered.

A workload prediction model using SVR and NN for growing, periodic, and unpredicted workload patterns is proposed in (Nikravesh et al., 2015b). Moreover, the influence of window size on the selected algorithms is considered. The adopted hypothesis is claiming that the prediction auto-scaling system accuracy can be improved by selecting the best prediction algorithms based on the workload pattern. The research is extended in (Nikravesh et al., 2017) to investigate the risk minimization principle using the same methods and workload patterns. In addition, the SVR, NN Multi-layer Perceptron (MLP), and MLP with Weight Decay (MLPWD) are considered. The NN is also adopted in (Kumar and Singh, 2018) to develop a workload prediction model that is able to learn the best mutation strategy along with optimal crossover rate. The model is evaluated using two real datasets and compared with maximum, average, and back propagation network methods.

In (Moreno-vozmediano et al., 2019), an auto-scaling system using the SVR model is introduced. Besides, a performance model based on queuing theory is proposed to determine the number of resources that must be provisioned. The SVR model is compared with e.g. LR method. Further, several SVR con-

figurations are investigated considering the kernel type. The results reveal that the SVR using different configurations outperform the other methods. In (B. Liu et al., 2020) both Autoregressive Moving Average (ARMA) and Elma Neural Network (ENN) are used where the ENN is responsible for correcting the prediction error of ARMA and providing the final prediction value.

Most of the presented studies focus on the CC environment and use a synthetic workload. To the best of our knowledge, this paper is the first to propose a CAPF for the EC environments based on real EC workload with a support of a proactive SAS. This framework is designed thanks to a thorough comparison of the most effective ML algorithms used in the literature with consideration of the window size effect to improve prediction accuracy.

3 SELF-ADAPTIVE SYSTEM ARCHITECTURE

This section briefly illustrates our SAS architecture by zooming in to show auto-scaling components only due to the page limit. It is shown in Figure 1 where the full version including the research roadmap can be found in (Aljulayfi and Djemame, 2019). The SAS uses the MAPE-based (Monitor, Analyse, Plan, Execute) loop with a focus on the Analyse activity as it is the paper’s scope. The use of MAPE-based allows the system to have a full and continuous management over the operational environment thanks to MAPE loop. Additionally, the design of the SAS architecture aims to have a hybrid adaptation, but this paper only focuses on the proactive side.

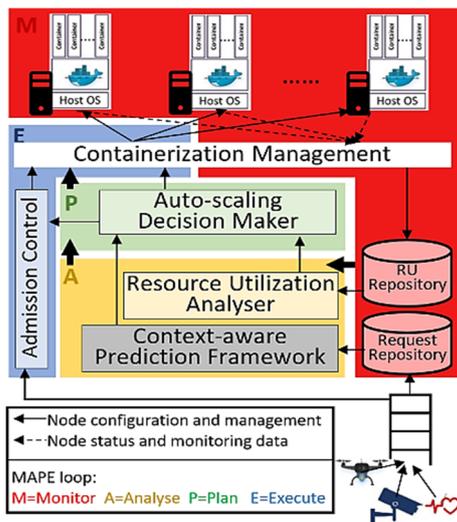


Figure 1: Self-adaptive system architecture.

The data analyser (i.e. analyse activity) is responsible for analysing the monitoring data which is provided by Monitor activity. In order to support hybrid adaptation, this activity is divided into two main components as follows: 1) *Context-aware Prediction Framework (CAPF)*: is responsible for predicting the number of tasks requests in the future by consuming the historical workload that stored in Request Repository where these requests will be scheduled as containers. This component (highlighted in grey) supports C3. Further details about its internal components is available in Section 6 as it is designed after conducting the paper’s investigation. 2) *Resource Utilization Analyser*: is responsible for reactive adaptation process which is used as a back-up for the CAPF in case of events are not predicted.

4 METHODOLOGY

This section presents the methods that are used towards achieving the research objectives.

4.1 Dataset Analysis

The paper makes use of the Shanghai Telecom dataset which simulates the EC workload (Sguangwang.com, 2018) and reported in (Guo et al., 2019; Wang, Guo, et al., 2019; Wang, Zhao, Huang, et al., 2019; Wang, Zhao, Xu, et al., 2019). It provides six months of mobile phones records accessing the Internet via base stations which are distributed over Shanghai city. The dataset has 7 attributes: *month, date, start time, end time, latitude, longitude, and user ID*. The analysis of the full data set shows that it has 6,952,921 records, 9739 mobile devices, and 3042 base stations. Further, a thorough analysis of the dataset is conducted in order to understand the workload patterns and mobile phone users’ behavior. However, this section only presents part of the workload analysis that is related to this paper.

A preliminary data analysis revealed the workload of the first month (i.e. June) has the lowest percentage of records with missing data e.g. base station location. Therefore, we decide to select the second day from

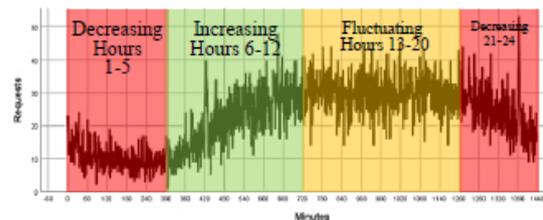


Figure 2: Workload pattern.

the same month as it is representative for the rest days in the same month in the sense that the overall workload pattern is periodic. The workload of this day is shown in Figure 2 per minute after removing the extreme outliers which hide the data pattern. From Figure 2, it can be seen clearly that the overall pattern of the data is a fluctuation with decreasing, increasing, and steady (fluctuating) behavior. Therefore, to propose a robust context-aware prediction model, the day will be divided into three categories based on the workload. These categories as shown in Figure 2 are 1) *decreasing* which includes late night and early morning (red), 2) *increasing* which includes morning (green), and 3) *Fluctuating* which includes afternoon to evening (orange). Further, one hour from each category is selected (i.e. 2nd, 12th, 14th hours) which will be used in training and testing the prediction models. The training and testing splitting percentage will be discussed in Section 4.4.

4.2 Machine Learning Algorithms

Three of the most popular and widely used ML algorithms, LR, SVR, and NN are considered. These algorithms are able to predict the future workload efficiently based on historical data (Baig et al., 2020; Islam et al., 2012; C. Liu et al., 2017; Sapankevych and Sankar, 2009).

The LR is the simplest and most widely used supervised ML algorithm for prediction (Baig et al., 2020; James et al., 2017). In this paper, the simplest case of LR is used because we have only one input variable. The SVR is an efficient learning method that implements the Support Vector Machine principle but produces continuous variable. The advantage of using SVR is its ability to map the time-series to a higher dimension using kernel function (Nikravesh et al., 2017). The NN or Artificial Neural Network (ANN) is a supervised learning algorithm that can be used for both regression and classification problems (Nikravesh et al., 2015b). A type of ANN is MLP which is a feed-forward network that is used for a range of problems including forecasting (Nikravesh et al., 2017; Zhang et al., 1998). This network architecture is adopted in this paper because it is the most popular and efficient network architecture that is used for forecasting.

4.3 Sliding Window Technique

The sliding window technique uses the last n samples of the data feature in order to forecast the future value

(Nikravesh et al., 2017). The use of the sliding window technique is important to perform a supervised ML when having only one feature in the dataset aiming to train the prediction algorithm (Nikravesh et al., 2015b). In this paper, the number of requests per time unit feature is only available. Therefore, in order to apply ML algorithms, the sliding window technique is used. Indeed, the window size is an important factor which has a significant influence on the ML prediction accuracy. However, selecting the appropriate window size is challenging because we have to find the best window size that allows the model to capture the data pattern and application behavior (Amiri and Mohammad-khanli, 2017). This means a small window size might not be representative while a large window size might cause overfitting (Nikravesh et al., 2015b). Therefore, this paper aims to investigate the effect of window size on the ML algorithms accuracy.

4.4 Experimental Design

This section presents the design of the experiments and the overall approach. As mentioned, three prediction models will be proposed, each model targets a part of the day. In order to do so, firstly, we must find the best splitting percentage that allows the ML algorithms to capture the data pattern and relationship. Further, this is done for each day part (i.e. decreasing, increasing, and fluctuating). Then, based on the best splitting percentage, the ML algorithms' accuracy will be evaluated using: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). This means for each workload pattern, the prediction models are trained and tested with consideration of the effect of the window size on the prediction accuracy. Finally, the CAPF will be designed based on these investigations.

Table 1: SVR and NN configurations.

Method	Parameter	Value
SVR	C	1.0
	Kernel	RBF
	RegOptimizer	RegSMOimproved
NN	Learning rate	0.38
	No. of hidden layers	1
	Number of hidden neurons	4
	Momentum	0.2
	Epoch/training time	10000

As part of the experimental design, the implementation of ML algorithms and their configurations must be considered. In this paper, all the selected algorithms are implemented using the well-known ML tool WEKA 3.8¹. In terms of the configurations, one predictor is used in LR simplest case. For SVR and MLP

¹ <https://www.cs.waikato.ac.nz/ml/weka/>

(i.e. NN), we used the same configuration of (Nikraves et al., 2015a, 2015b) as shown Table 1 because we have same scenario and workload pattern.

5 RESULTS AND DISCUSSION

This section presents and discusses the experimental results. The discussion for workload will be separate as each workload represents a different pattern.

5.1 Percentage Splitting

Before comparing the ML models, it is important to specify the best training duration that allows the models to capture and learn the data pattern. This section presents the results of the experiments considering the proportion of the dataset to include in the train split: 80/20 (i.e. 80% training and 20% testing) and 70/30 (i.e. 70% training and 30% testing). This means each workload (i.e. decreasing, increasing, and fluctuating) is split and evaluated using these percentages.

The overall results show that the 80/20 is the best split percentage as it allows the selected algorithms to capture the data pattern and provide the most accurate results. Further, the 80/20 split outperforms the 70/30 overall evaluation metrics and different window sizes. Therefore, the 80/20 splitting percentage results will be considered in the following sections.

5.2 ML Algorithms Comparison

This section compares the accuracy of the ML prediction algorithms considering the testing results and addresses accordingly contribution (C2).

Data with the *decreasing workload* reveals that SVR outperforms both LR and NN in overall prediction accuracy metrics. This can be seen clearly from Table 2 which shows the evaluation metrics for decreasing workload. Additionally, the best ML prediction value is also provided by the SVR when the window size is 3 using MAE and RMSE. If LR is compared with NN, the LR outperforms NN in most cases.

For the *increasing workload* pattern, the prediction results in Table 3 show that SVR outperforms both LR and NN in most cases; this is similar to the decreasing workload pattern. However, by looking closely at the results, the best prediction values over the evaluation metrics are provided by NN when the window size equals 9. Although SVR has better accuracy in most cases, the NN provides the best accuracy in the increasing data pattern, thus, NN will be adopted for this data pattern.

Table 2: MAE, MAPE, and RMSE values (decreasing).

W. Size	MAE			MAPE			RMSE		
	LR	SVR	NN	LR	SVR	NN	LR	SVR	NN
2	2.56	2.5	3.04	34.43	32.6	42.63	3.1	2.95	3.78
3	2.55	2.41	3.29	34.21	30.32	45.81	3.08	2.84	4.01
4	2.56	2.52	2.52	34.3	33.02	28.44	3.09	2.98	2.93
5	2.57	2.57	4.61	34.61	34.79	62.37	3.11	3.13	5.37
6	2.55	2.49	2.47	34.16	32.46	31.35	3.07	2.94	2.86
7	2.69	2.49	2.82	37.37	32.73	39.43	3.37	2.97	3.53
8	2.53	2.5	3.25	33.65	32.61	45.18	3.03	2.95	3.98
9	2.51	2.42	3.91	33.02	30.71	35.36	2.98	2.86	4.56

Table 3: MAE, MAPE, RMSE values (increasing).

W. Size	MAE			MAPE			RMSE		
	LR	SVR	NN	LR	SVR	NN	LR	SVR	NN
2	4.25	4.24	5.33	16.67	16.83	22.54	4.76	4.84	6.78
3	4.25	4.23	4.53	16.75	16.84	16.35	4.8	4.84	5.21
4	4.37	4.25	5.45	17.54	16.9	22.95	5.2	4.87	6.87
5	4.25	4.25	4.28	16.91	16.92	15.99	4.87	4.87	4.7
6	4.82	4.25	5.72	18.99	16.92	24.08	5.44	4.87	7.13
7	4.25	4.24	5.81	17.02	16.89	24.49	4.93	4.86	7.28
8	4.25	4.25	6.21	16.92	16.91	25.93	4.88	4.86	7.54
9	4.25	4.24	3.87	16.87	16.9	14.87	4.85	4.87	4.57

The prediction results of the *fluctuating workload* are shown in Table 4. Unlike the decreasing and increasing patterns, the result reveals that NN has better accuracy as compared to both LR and SVR in most cases. Moreover, its accuracy is the best when the window size is 9 using MAPE and RMSE. Therefore, the NN with window size 9 will be adopted in the CAPF.

Table 4: MAE, MAPE, RMSE values (fluctuating).

W. Size	MAE			MAPE			RMSE		
	LR	SVR	NN	LR	SVR	NN	LR	SVR	NN
2	4.97	4.84	4.26	20.93	20.41	17.26	6.23	6.11	5.13
3	4.86	4.7	4.29	20.87	19.8	16.96	6.45	5.94	5.06
4	4.74	4.65	5.52	19.92	19.54	23.3	5.97	5.89	6.84
5	4.72	4.6	4.75	19.83	19.31	19.97	5.95	5.81	5.96
6	4.65	4.59	4.57	19.51	19.13	18.97	5.86	5.73	5.61
7	4.63	4.48	4.63	19.31	18.56	19.58	5.77	5.54	5.91
8	4.63	4.66	4.06	19.36	19.41	16.88	5.79	5.8	5.06
9	4.66	4.69	4.15	19.75	19.63	15.88	5.88	5.91	4.93

5.3 Sliding Window Effect

The increase in window size does not have a significant impact on LR and SVR algorithms over all metrics in both decreasing and increasing data as shown in Figures 3 and 4, respectively. To be more specific, from Figure 3 and 4, the changes in the accuracy of both LR and SVR are roughly steady when the window size increase. Although the SVR accuracy is almost steady, it provides the best accuracy when the window size is 3 as compared to other ML algorithms in decreasing data. Additionally, the difference between LR and SVR are neglected in the most cases of different window size values. In terms of the increasing data, the SVR accuracy seems to be steady over

window size as shown in Figure 4. In contrast, the increase of window size causes highly fluctuating NN accuracies over MAE, MAPE, and RMSE in both decreasing and increasing data. Further, in the increasing data, NN provides the best accuracy when the window size equals to 9.

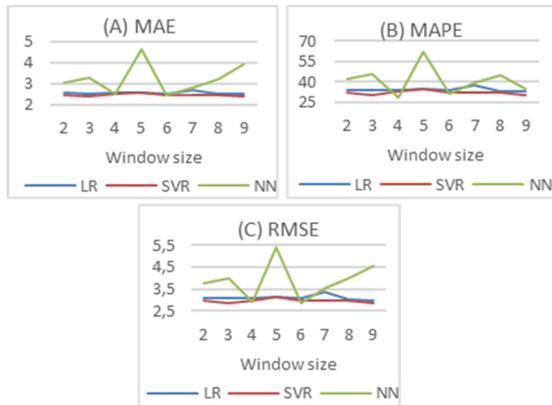


Figure 3: Window size effect (decreasing).

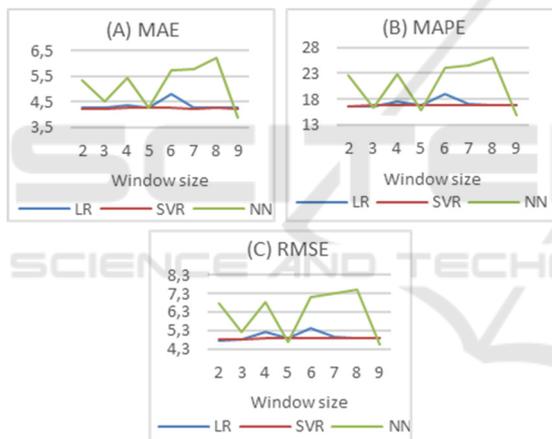


Figure 4: Window size effect (increasing).

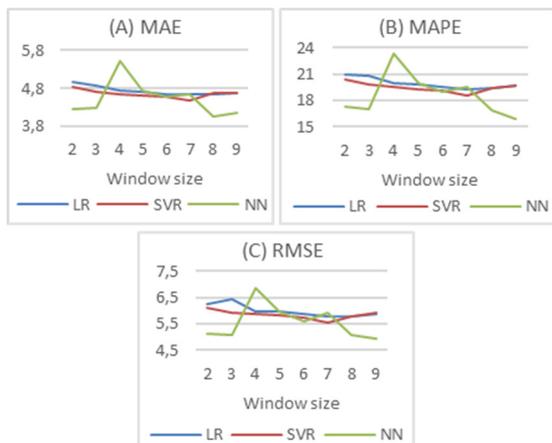


Figure 5: Window size effect (fluctuating).

Unlike decreasing and increasing data, increasing the window size has a positive impact on all ML algorithms overall evaluation metrics in fluctuating data. This effect can be seen clearly in Figure 5 which shows a decreasing trend. Further, the sliding window technique slightly improves the accuracy of both LR and SVR. In the case of NN, it has a significant impact on its accuracy over MAE, MAPE, and RMSE. Additionally, the best accuracy is provided by NN when the window size is 9 using MAPE and RMSE.

5.4 Results Summary

This section highlights the main findings of the experiments and their position in the context of the related work. NN outperforms LR and SVR in both increasing and fluctuating workloads whereas SVR outperforms NN and LR in decreasing workload. The reason NN exhibits the best accuracy is its ability to capture all noise in the data whereas SVR tries to find a smooth curve to cover them (Nikraves et al., 2017). Based on this logic, SVR should outperform NN in increasing workload. However, the increasing workload has some form of fluctuation which reduces the SVR accuracy.

In terms of the sliding window, the results show that for some workload patterns increasing the window may have a significant impact on the prediction accuracy. For example, increasing the window size has a positive impact on the ML algorithms in the fluctuating workload because the large window size allows the models to learn the relationships between features (Islam et al., 2012; Nikraves et al., 2015b). In contrast, the increase of window size does not have an impact on some ML algorithms such as LR and SVR which means that their accuracies are almost steady over the window size values (Nikraves et al., 2017).

6 CONTEXT-AWARE PREDICTION FRAMEWORK

This section proposes the CAPF fulfilling contribution (C3). It is designed according to the above experiments that select the best ML prediction algorithms with consideration of the window size. Further, it is integrated with the SAS architecture in the CAPF component that is shown in Figure 1 (i.e. highlighted in grey). The framework aims to forecast the future workload that will be submitted to the EC by the IoT devices. This framework consists of two main components as shown in Figure 6 (highlighted in grey) which are: 1) *Context analyser*: identifies the context

of the application, including time dependence, in order to select the ML model suits its workload pattern (i.e. decreasing, increasing, and fluctuating). 2) *Algorithm selector*: selects the best ML algorithm based on the workload pattern that is identified by the Context Analyser. The selection of algorithms will be based on the experiments that we performed to select the best ML algorithm for each pattern. In other words, it uses either SVR or NN for predicting the future workload based on the day's part.

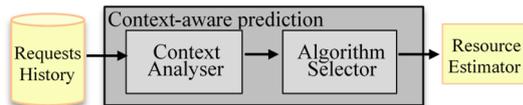


Figure 6: Context-aware prediction framework.

7 CONCLUSION AND FUTURE WORK

This paper has presented a CAPF to support auto-scaling decisions in EC environments. This framework predicts the future EC workload using either SVR or NN ML algorithms. These algorithms are considered the best ML algorithms to be used for EC workloads which is based on the comparison that has been performed. Further, the comparison process includes a thorough investigation on the window size effect. All these steps have done using the Shanghai Telecom dataset which represents a real EC workload. The results reveal that window size has a significant impact on workload patterns and ML algorithms as best size allows the ML algorithms to capture the workload pattern and behavior.

The SAS architecture is currently under development with the aim of supporting, e.g. elasticity, scalability, and QoS. In term of elasticity support, the predicted workload will be implemented to evaluate the effectiveness of the developed models in operational environment under several applications' scenarios. Further, this involves evaluating the performance of the proposed SAS in the EC with consideration to various adaptation approaches which are proactive, reactive, and hybrid adaptation. The scalability support must also be considered to efficiently utilize the EC resource and maximize the number of running applications in the EC environment in sense that the EC have limited resources. It requires meeting the applications' QoS of IoT devices which have very sensitive requirements such as low latency. The support of scalability and QoS will also involve the consideration of their trade-offs, which is key in service provision.

REFERENCES

- Ajila, S. A., and Bankole, A. A. (2013). Cloud client prediction models using machine learning techniques. *International Computer Software and Applications Conference*, 134–142.
- Al-Dhuraibi, Y., Paraiso, F., Djarallah, N., and Merle, P. (2018). Elasticity in Cloud Computing: State of the Art and Research Challenges. *IEEE Transactions on Services Computing*, 11(2), 430–447.
- Aldossary, M., and Djemame, K. (2018). Performance and energy-based cost prediction of virtual machines auto-scaling in clouds. *44th Euromicro Conference on Software Engineering and Advanced Applications*, 502–509.
- Aljulayfi, A. F., and Djemame, K. (2019). A Novel QoS and Energy-aware Self-adaptive System Architecture for Efficient Resource Management in an Edge Computing Environment. *35th Annual UK Performance Engineering Workshop*, 39–54.
- Amiri, M., and Mohammad-khanli, L. (2017). Survey on prediction models of applications for resources provisioning in cloud. *Journal of Network and Computer Applications*, 82, 93–113.
- Arcaini, P., Riccobene, E., and Scandurra, P. (2015). Modeling and Analyzing MAPE-K Feedback Loops for Self-Adaptation. *10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 13–23.
- Baig, S., Iqbal, W., Lluís, J., and Carrera, D. (2020). Adaptive sliding windows for improved estimation of data center resource utilization. *Future Generation Computer Systems*, 104, 212–224.
- Bankole, A. A., and Ajila, S. A. (2013). Cloud client prediction models for cloud resource provisioning in a multitier web application environment. *2013 IEEE 7th International Symposium on Service-Oriented System Engineering, SOSE 2013*, 156–161.
- Calheiros, R. N., Masoumi, E., Ranjan, R., and Buyya, R. (2015). Workload prediction using ARIMA model and its impact on cloud applications' QoS. *IEEE Transactions on Cloud Computing*, 3(4), 449–458.
- D'Angelo, M. (2018). Decentralized self-adaptive computing at the edge. *IEEE/ACM 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 144–148.
- Delicato, F. C., Pires, P. F., and Batista, T. (2017). *Resource Management for Internet of Things*. Springer. <https://doi.org/10.1007/978-3-319-54247-8>
- Dolui, K., and Datta, S. K. (2017). Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. *2017 Global Internet of Things Summit (GIoTS)*, 1–6.
- Framingham, M. (2019). *The Growth in Connected IoT Devices*. IDC Analyze the Future. <https://www.idc.com/getdoc.jsp?containerId=prUS45213219>
- Galante, G., and De Bona, L. C. E. (2012). A survey on cloud computing elasticity. *2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing A*, 263–270.

- Guo, Y., Wang, S., Zhou, A., Xu, J., Yuan, J., and Hsu, C. H. (2019). User allocation-aware edge cloud placement in mobile edge computing. *Software - Practice and Experience, January 2019*, 1–14.
- Gupta, H., Dastjerdi, A. V., Ghosh, S. K., and Buyya, R. (2017). iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software - Practice and Experience, 47(9)*, 1275–1296.
- Islam, S., Keung, J., Lee, K., and Liu, A. (2012). Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems, 28(1)*, 155–162.
- James, G., Witten, D., Hastie, T., and Tibshirani, R. (2017). *Introduction to Statistical Learning with Applications in R*. Springer.
- Kaur, K., Dhand, T., Kumar, N., and Zeadally, S. (2017). Container-as-a-Service at the Edge: Trade-off between Energy Efficiency and Service Availability at Fog Nano Data Centers. *IEEE Wireless Communications, 24(3)*, 48–56.
- Kavanagh, R., Djemame, K., Ejarque, J., Badia, R. M., and Garcia-perez, D. (2019). Energy-aware Self-Adaptation for Application Execution on Heterogeneous Parallel Architectures. *IEEE Transactions on Sustainable Computing, 1–15*.
- Kramer, J., and Magee, J. (2007). Self-Managed Systems: an Architectural Challenge. *2007 Future of Software Engineering, 259–268*.
- Krupitzer, C., Roth, F. M., VanSyckel, S., Schiele, G., and Becker, C. (2015). A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing, 17*, 184–206.
- Kumar, J., and Singh, A. K. (2018). Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Generation Computer Systems, 81*, 41–52.
- Li, G., Song, J., Wu, J., and Wang, J. (2018). Method of Resource Estimation Based on QoS in Edge Computing. *Wireless Communications and Mobile Computing, 2018*.
- Liu, B., Guo, J., Li, C., and Luo, Y. (2020). Workload forecasting based elastic resource management in edge cloud. *Computers and Industrial Engineering, 139(0360–8352)*, 1–12.
- Liu, C., Liu, C., Shang, Y., Chen, S., Cheng, B., and Chen, J. (2017). An adaptive prediction approach based on workload pattern discrimination in the cloud. *Journal of Network and Computer Applications, 80*, 35–44.
- Lorido-Botran, T., Miguel-Alonso, J., and Lozano, J. A. (2014). A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments. *Journal of Grid Computing, 12(4)*, 559–592.
- Moreno-vozmediano, R., Montero, R. S., Huedo, E., and Llorente, I. M. (2019). Efficient resource provisioning for elastic Cloud services based on machine learning techniques. *Journal of Cloud Computing Advances, Systems and Applications, 8(1)*.
- Nikravesh, A. Y., Ajila, S. A., and Lung, C. H. (2017). An autonomic prediction suite for cloud resource provisioning. *Journal of Cloud Computing, 6(1)*.
- Nikravesh, A. Y., Ajila, S. A., and Lung, C. H. (2015a). Evaluating Sensitivity of Auto-scaling Decisions in an Environment with Different Workload Patterns. *IEEE 39th Annual International Computers, Software & Applications Conference, 415–420*.
- Nikravesh, A. Y., Ajila, S. A., and Lung, C. H. (2014). Measuring prediction sensitivity of a cloud auto-scaling system. *Proceedings - IEEE 38th Annual International Computers, Software and Applications Conference Workshops, COMPSACW 2014, 690–695*.
- Nikravesh, A. Y., Ajila, S. A., and Lung, C. H. (2015b). Towards an Autonomic Auto-scaling Prediction System for Cloud Resource Provisioning. *Proceedings - 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2015, 35–45*.
- Sapankevych, N. I., and Sankar, R. (2009). Using Support Vector Machines: A Survey. *IEEE Computational Intelligence Magazine, 2*, 24–38.
- Sguangwang.com. (2018). *The Telecom Dataset (Shanghai Telecom)*. <http://sguangwang.com/TelecomDataset.html>
- Shi, W., and Dustdar, S. (2016). The Promise of Edge Computing. *Computer, 49(5)*, 78–81.
- Singh, S., and Chana, I. (2015). QoS-Aware Autonomic Resource Management in Cloud Computing: A Systematic Review. *ACM Computing Surveys, 48(3)*, 1–46.
- Toczé, K., and Nadjm-Tehrani, S. (2018). A Taxonomy for Management and Optimization of Multiple Resources in Edge Computing. *Wireless Commu. and Mobile Computing, 2018*, 1–20.
- Wang, S., Guo, Y., Zhang, N., Yang, P., Zhou, A., and Shen, X. S. (2019). Delay-aware Microservice Coordination in Mobile Edge Computing: A Reinforcement Learning Approach. *IEEE Transactions on Mobile Computing, 1–1*.
- Wang, S., Zhao, Y., Huang, L., Xu, J., and Hsu, C. H. (2019). QoS prediction for service recommendations in mobile edge computing. *Journal of Parallel and Distributed Computing, 127*, 134–144.
- Wang, S., Zhao, Y., Xu, J., Yuan, J., and Hsu, C. H. (2019). Edge server placement in mobile edge computing. *Journal of Parallel and Distributed Computing, 127*, 160–168.
- Xu, M., and Buyya, R. (2019). Brownout Approach for Adaptive Management of Resources and Applications in Cloud Computing Systems. *ACM Computing Surveys, 52(1)*, 1–27.
- Zhang, G., Eddy Patuwo, B., and Y. Hu, M. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting, 14(1)*, 35–62.