A Machine Learning Regression approach for Throughput Estimation in an IoT Environment

Aroosa Hameed, John Violos, Nina Santi, Aris Leivadeas, Nathalie Mitton

A Machine Learning Regression approach for Throughput Estimation in an IoT Environment

Aroosa Hameed *, John Violos *, Nina Santi †, Aris Leivadeas *, Nathalie Mitton †
* Department of Software and IT Engineering, École de technologie supérieure, Montreal, Canada
Email: aroosa.hameed.1@ens.etsmtl.ca, ioannis.violos.1@ens.etsmtl.ca , aris.leivadeas@etsmtl.ca
† Inria Lille-Nord, Lille, France
Email: {nina.santi,nathalie.mitton}@inria.fr

***Abstract*—The success of Internet of Things (IoT) has significantly increased the volume of data generated by various smart applications. However, as many of these applications are characterized by strict Quality of Service (QoS) requirements, there is a growing need for accurately predicting typical performance parameters such as throughput. This prediction should be based on the applications' traffic profiles and at the same time reflect the network uncertainty that IoT access networks add to the overall communication. In this work, we deployed 6 different smart building applications in a real testbed while creating a considerable traffic contention in an IEEE 802.15.4 access network. After preprocessing the raw data and following a feature engineering mechanism, we apply five different regression learning approaches to each application and predict its throughput. By resorting to several prediction error metrics and time metrics such as training and inference time, we show that the multiple linear regression achieves high accuracy while outperforming other well known machine learning methods.**

***Index Terms*—IoT, Machine learning, Throughput prediction.**

I. Introduction

Internet of Things (IoT) is gaining considerable momentum. Even though the term was captured the last decade, we already live in the fourth industrial revolution. The success of IoT lies in the data generated and their analysis to add the necessary intelligence to the applications themselves. Obviously, data analysis can be proved to be a hefty task that cannot be efficiently executed in IoT devices but it rather has to be offloaded to more resource powerful platforms, such as Edge and/or Cloud Computing.

However, the amount of Edge/Cloud resources to be used depends on the volume of the data generated from the IoT devices. Therefore, the latter creates an important challenge related to the accurate QoS traffic profiling of an IoT application. The reason is that IoT access networks are usually wireless, lossy, and unreliable. At the same time, the co-existence of different IoT applications consisting of heterogeneous devices that send data of different contexts and frequencies, and the interference between the devices add several levels of complexity when it comes to predicting their traffic profile and thus their resource demands.

Accordingly, there is a huge interest in classifying the IoT applications based on their networking (e.g., throughput, access network) and functional requirements (e.g., number of devices, transmission frequency, etc.) by applying network analytic mechanisms. In particular, IoT traffic classification can be focused on single applications, where traffic characteristics per device can be extracted [1], or it can be based per application [2]. Nonetheless, these classifications are not efficiently capturing the network behavior and the sources of the uncertainty in the overall traffic sent at the Edge.

To this end, in this paper we aim to predict the throughput for a number of different IoT applications in the context of a smart building by applying a set of different Machine Learning (ML) algorithms. Our goal is to investigate how accurate the throughput can be predicted in a random access network, when a number of heterogeneous applications with various functional characteristics generate a fluctuating traffic dataset. The contributions of this paper can be listed as following:

1) We consider 6 different IoT smart building applications that present different requirements in terms of number of devices, packet length, type of message, and message frequency transmission.
2) We deploy the applications in a real testbed [3] comprised of approximately 300 IoT devices and generate the traffic profile by using an IEEE 802.15.4-2006 access network.
3) We apply five different regression learning techniques with the goal to predict the throughput per application in an access network characterized by high contention and interference that creates a very dynamic traffic profile.

The rest of the paper is organized as follows. Section II presents the related work and their limitations. Section III gives a detailed information on the use case, dataset generation and the feature engineering. Section IV summarizes the proposed models and their operation. Section V illustrates the results and the efficiency of the proposed solutions. Finally, Section VI concludes the paper.

This work was supported in part by the CHIST-ERA-2018-DRUID-NET project "Edge Computing Resource Allocation for Dynamic Networks".

II. Related Work

In the pertinent literature, for IoT traffic profiling and QoS prediction, various studies either applied traditional machine learning algorithms or deep learning frameworks. For example, Akbar et al. [4] focus on the Complex Event Processing (CEP) and historical data prediction using machine learning algorithms. The authors proposed an adaptive predictive model using a regression technique called moving window regression in order to provide a distributed and scalable solutions. The authors in [5] performed the prediction of the IoT network traffic in order to provide a reliable communication. To achieve this, deep learning has been successfully applied using Long Term Short Memory (LSTM). The features of dataset consist of the timestamp, bytes count and the packets count. This work is further extended in [6], for 5G networks using the Recurrent Neural Networks (RNN) and specifically the LSTM.

Following, Lopez et al. [7] presented a detailed work for the forecasting of IoT traffic volumes based on the stochastic gradient descent algorithm and neural network architectures called gaNET. The dataset included only two features (obfuscated mobile identification (SIM) and the time stamp of traffic records). The authors in [8] proposed a single step ahead and a multistep prediction method for delay prediction in IoT based on NARX recurrent neural network. They simulated an IoT environment and used a simulated dataset. Ateeq et al. [9] implemented a delay prediction in IEEE 802.15.4 network using deep learning multi parametric approach. The features utilized by this work are extracted from the application layer, MAC layer and physical layer of the network.

Furthermore, the focus in [10] is to predict QoS in IoT environments including the service response time and throughput. Nonetheless, the approach used is based on a matrix factorization technique and is limited to missing value predictions in a data matrix containing values for both response time and throughput. The work in [11] also evaluated a number of matrix factorisation approaches and have shown how these approaches can be used to make QoS predictions in an IoT environment. The matrix factorization algorithms used in this work for the evaluation of QoS prediction of the web services includes: (i) CloudPred (ii) EMF (Extended Matrix Factorization) and (iii) LN-LFM (Latent factors models).

However, the above techniques present the following drawbacks: i) Most of the studies utilized a limited number of features for the prediction of the IoT traffic and do not provide detailed fine grained characteristics [4]–[7]; ii) Despite being more and more used, neural network also present drawbacks: they usually require preprocessing of the input data into a very network specific shape and the complex structures of the deep learning models require extensive training time [5]–[9]; iii) Some of the existing studies either predict traffic volume or QoS for a single application and may not provide accurate estimates outside those application [10], [11].

In this work, we solve the above mentioned challenges as follows: (i) First, we perform the feature engineering which is the extraction of additional features from the existing raw features included in each application's dataset; (ii) in order to reduce the training and prediction times, we apply light-weight traditional regression learning approaches for each of the IoT applications; (iii) Finally, we provide throughput prediction for six different and heterogeneous IoT applications that coexist in the same access network.

TABLE I
Experimentation's parameters

Scenario	Nb devices	Message freq.	Packet length	Duration
HVAC	100	1 packet/4 min	60 B	60 min
Lighting	100	1 packet/8 min	30 B	90 min
Emergency	40	1 packet/30 sec	127 B	10 min
Surveillance	30	99 packets/sec	127 B	10 min
AR	10	197 packets/sec	127 B	10 min
VoIP	10	16 packets/sec	127 B	10 min

III. Use case and Dataset description

A. Use case description

The dataset used in this paper represents several smart building applications. Among these applications, there are smart monitoring and response systems for HVAC (heating, ventilation, and air conditioning), lighting, and emergency. The HVAC and lighting systems adapt their activities according to external measurements. The emergency system monitors the building's critical areas, such as gas pipes or fire alarms. Also, there is a surveillance application with cameras and an augmented reality application for themed rooms inside the building. Finally, there is a Voice over IP (VoIP) application for automatic help desks or interactive voice recognition. For each application, we have defined their traffic characteristics in terms of packet sizes, message frequencies, the number of deployments and their duration, as shown in Table I. It should be noted that a typical IEEE 802.15.4 network has a row capacity of 250 kbps. However, in order to be sure that we will saturate the network, we configured the transmission frequencies of the devices in such a way so we can create an environment with high contention. The reason behind this decision is to have a fluctuating traffic profile and to better reveal the efficiency of the used ML techniques in the context of an uncertain communication environment.

B. Dataset Generation Description

Once the traffic characteristics are defined for each application, we launched several experimentations on the FIT IoT-LAB testbed [3]. The platform provides different kinds of sensor nodes connected through a mesh topology and they are remotely reservable and completely programmable. The nodes are IoT-LAB M3 boards, based on STM32 family microcontrollers [1]. For each application under study, we thus run a firmware on a set of representative nodes that have to exchange packets in a broadcast mode using the IEEE 802.15.4-2006 MAC layer and RPL routing protocol. As a result, we have

[1]https://www.iot-lab.info/docs/boards/iot-lab-m3/

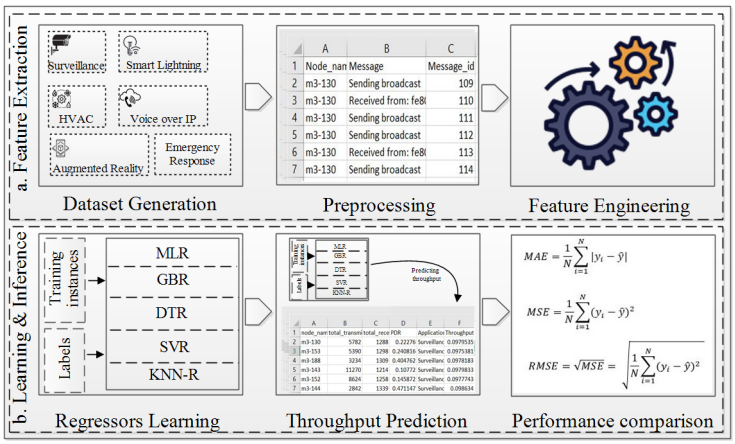

Fig. 1. Proposed Regression Framework

obtained files containing the data described in Table II for each received or transmitted packet.[2]

TABLE II
DESCRIPTION OF RAW FEATURES IN DATASET

Data	Feature	Description
Received Data	node_name	name of sensor node
	timestamp	message reception time
	message_id	message unique identifier
	reception delay	reception delay in milliseconds
Transmitted data	node_name	name of sensor node
	timestamp	message transmission time
	message id	message unique identifier
	success	transmission success

C. Feature Engineering

One of the main objectives of this work is to extract the most relevant QoS features for an IoT application. Therefore, we engineered/extracted several features as described below:

1) Node name: The sensor name used for generating real time IoT traffic under several applications.
2) $total_{msg\_trans}$: Total number of messages transmitted by a node.
3) $total_{msg\_rec}$: Total number of messages received by a node.
4) $time_{first\_msg}$: Time of the first transmitted message in the experimentation.
5) $time_{last\_msg}$: Time of the last transmitted message in the experimentation.
6) Transmission success: Total number of messages successfully transmitted by a node without dropouts.
7) Average transmission latency: The average time taken by a transmitted message to be successfully received :

$$Latency = \frac{\sum (time_{msg\_rec} - time_{msg\_trans})}{total_{msg\_trans}} \quad (1)$$

where $time_{msg\_rec}$ is the message reception time, and $time_{msg\_trans}$ is the message transmission time.

8) Packet Delivery Ratio (PDR): The ratio of received packets to transmitted packets per node:

$$PDR = \frac{total_{msg\_rec}}{total_{msg\_trans}} * 100 \quad (2)$$

9) Throughput: The rate of successful message delivery over a network channel:

$$Throughput = \frac{total_{msg\_trans}}{time_{last\_msg} - time_{first\_msg}} \quad (3)$$

All of the above mentioned features, except node name, are calculated using the raw received and transmitted data features illustrated in Table II. Furthermore, the calculation of features is done per node within each of IoT application and the time is measured in milliseconds.

IV. Regression Methodology

The proposed regression framework consists of two main steps as shown in Fig. 1. The first step is the feature extraction which consists of raw data preprocessing and feature engineering (as discussed in Section III-C). The second step i.e., learning and inference is comprised of training regression models for each application. Finally throughput is predicted and different models are compared using appropriate metrics.

A. Predicting QoS with Machine Learning

Throughput is a well known QoS metric, and with respect to the terminology of data science, it belongs to the quantitative data and more specifically to the ratio data type. This urged us for the use of a Machine Learning (ML) model, more specifically regression, which is a prominent solution following the supervised learning approach. In the context of IoT and Edge Computing, the heterogeneity and dynamicity of the devices and workload renders the estimation and prediction of QoS a black box problem. IoT devices can connect and disconnect dynamically over time, different applications can create different transmission patterns, while

[2]The data are in open access along with the used method at : https://gitlab.irisa.fr/0000H82G/traces

the unpredictability of the access network can add several additional levels of complexity in an already fluctuating traffic profile.

Profiling the data transmission in an IoT enabled Edge computing environment provides valuable information. ML regression models work like approximator functions and are capable of approaching feasible solutions to the QoS black box problem leveraging on the historical datasets. The important characteristic of a ML regression is that instead of making a simple memorization of the historical data, it builds a prediction mechanism with strong generalization abilities. This means that the regression model can provide accurate predictions for observations it has never seen before.

Neural network techniques can also be a solution, but as such methods deals better with the nonlinearities and require large training data, they could not be directly applied for the particular use case. Specifically, we performed experiments with models such as vanilla LSTM, bidirectional LSTM and stacked LSTM but the results found were rather poor. Hence, in the rest part of this section we only emphasize on ML-based models. More specifically, some of the most efficient and widely used regression models in the ML area are described. These models have also been experimentally evaluated for the needs of the QoS prediction and discussed in the next sections.

B. Regression methods

1) Decision Tree Regression (DTR): DTR [12] consists of hierarchical successive layers of nodes which are formed with a recursively subdivision of the feature space into smaller areas. The subdivision of the feature space is aligned with the partition of the trained data observations into nodes with the principle that starting from the root node down to the leaf nodes the node purity is increased. The traverse of the DTR takes place based on the attribute splits and eventually the leaf nodes output the predicted values.

The advantages of DTR include that they are inexpensive to construct, extremely fast in the inference, easy to interpret for small-sized trees, robust to noise and they can easily handle redundant or irrelevant attributes. The disadvantages are that the space of possible decision trees is exponentially large and greedy approaches are often unable to find the best tree. The DTR does not take into account interactions between attributes. Finally, the DTR can be unstable because small variations in the data might result in a completely different tree being generated.

2) Gradient Boosted Regression Tree (GBRT): GBRT [13] is an ensemble of decision trees as estimators with an iterative functional gradient descent algorithm that minimizes the loss function over the hypothesis space. The cost optimization takes place by iteratively selecting a weak DTR with a negative gradient direction. The loss function is minimized iteratively by adding at each step a new DTR that reduces the loss while leaving unchanged the previous DTR. Every new DTR is fitted to the residuals of the previous DTR with a contribution weight shrunk by a learning rate smaller than one. The final GBRT is a linear combination of the DTR following a forwards stage-wise procedure.

GBRT is robust to overfitting and can fit complex nonlinear relationships between variables. GBRT handles different types of estimator variables and interaction effects among estimators. In most of the cases, a large number of estimators results in better performance and it is one of the GBRT hyperparameters. GBRT can also mitigate the problem of missing data and there is no need for the outliers to be discarded.

3) Support Vector Regression (SVR): SVR [14] is a flexible model that learns from data observations to draw hyperplanes in an n-dimensional feature space and provides predictions with a function f as given in Eq. (4), based on a decision boundary and an error margin ϵ. The learning process tunes the ϵ to gain an acceptable accuracy minimizing the coefficients w and satisfying the constraints in Eq. (5)

$$f(x) = \langle w, x \rangle + \ b, \ with \ w \in X, \ b \in \mathbb{R} \tag{4}$$

$$minimize \ \ \tfrac{1}{2}\|w\|^2$$
$$subject\ to = \begin{cases} y_i - \langle w, x_i \rangle - b \leq \epsilon \\ \langle w, x_i \rangle + b - y_i \leq \epsilon \end{cases} \tag{5}$$

Eq. (5) can be solved using the Lagrange multiplier method. In case some data observations are not linearly separable but a linear decision boundary is feasible, we can introduce slack variables in order to apply a soft version of margins. If there are not linear decision boundaries, then we can transform the data observations into a higher dimensional space using the kernel trick.

SVR is effective in high dimensional spaces and has very good generalization capabilities in unseen data. Its computational complexity does not depend on the dimensionality of the input space but it has the disadvantage that it is computational heavy for training a new model. In addition, it is not suitable for large datasets and missing values.

4) K-Nearest Neighbor Regression (KNNR): KNNR [15] is a non parametric method, namely the data observations distributions do not require to meet certain assumptions. KNNR is based on the principle that the k closest neighbors of a testing observation in an n-dimensional space can predict the testing value by averaging their outputs. The k neighbours can also be weighted by the inverse of their distances to the testing observation. The training data are located in the n-dimensional space and are capable of selecting the k neighbours using a distance metric like the Minkowski distance. Different metrics have also been proposed in the literature like Mahalanobis and cosine distance. The number of k is predefined by the user or chosen by a technique like cross-validation.

KNNR has the advantage that is an instant regression model. This means that there is no need for training, but the computations take place in the prediction stage by comparing the distances of the testing observation with the training data. It also implies that new training observations can be added seamlessly. The disadvantage of KNNR is that it cannot be efficient in large datasets and in high dimensional space. In addition, it is sensitive to noise, outliers and missing values.

5) Multiple Linear Regression (MLR): MLR [16] was developed in the field of statistics, but its applicability and efficiency to describe a wide range of problems with a simple to implement and well understood way makes it a straightforward approach by data scientists. MLR models the relationship between variables by fitting a linear equation to the training data and minimizing the sum of the squared residuals. The formula of MLR is given in Eq. (6):

$$f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k + \epsilon \quad (6)$$

where β_0 represents the intercept, β_k represents the coefficients and ϵ represents a random error. The β_0, β_k are estimated with a method like Ordinary Least Squares. Lasso and Ridge regression are variations of MLR that address some of the problems that introduce the Ordinary Least Squares, like the multicollinearity, by imposing a penalty on the size of β_k with L1 and L2 regularization respectively. In addition, MLR can mitigate the problem of overfitting using the dimensionality reduction.

An important advantage of MLR is that both training and inference are computational efficient. But, linear models do not have enough capacity to model complex relationships between variables. This can be a reason that MLR is prone to underfitting, noise and outliers.

V. Experimental evaluation

A. Experimental setting

A total of 11171 labeled instances for VoIP virtual assistant, 238050 instances for video surveillance, 27536 instances for smart lightning, 38258 instances for HVAC, 32981 instances for an emergency response and 46843 instances for AR application were collected. The classification was implemented in Python (version 3.8.2), while we split each application dataset into two groups of 70% training and 30% testing instances.

B. Evaluation Metrics

The evaluation of regression models is done using prediction error metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) along with training and inference time. All of the error metrics are positive and their smaller value means better prediction models. MAE is the sum of the absolute value of differences between the target values y_j and predicted values $\overline{y}_j$, divided by the total number of predictions as given in Eq. (7).

$$MAE = \frac{1}{n}\sum_{n=1}^{n} |y_j - \overline{y}_j| \quad (7)$$

MSE is the average of the squared errors between the predicted values and the targeted values divided by the total number of predictions. RMSE is the square root of MSE as given in Eq. (8).

$$RMSE = \sqrt{\frac{1}{n}\sum_{n=1}^{n} (y_j - \overline{y}_j)^2} \quad (8)$$

As RMSE assigns a higher weight to larger prediction errors, it is more useful when large errors of QoS are undesirable. However, MAE is preferred when all errors have the same importance. Furthermore, RMSE is equal or greater than MAE and may increase more than MAE as the dataset increases. Finally, since the prediction process should not be computationally heavy, we focus on light weight ML models with smaller training time as compared to deep learning models.

TABLE III
Performance metrics of all algorithms for IoT applications

Application	Algorithm	MAE	MSE	RMSE
Augmented Reality	MLR	0.0045641	2.9297e-05	0.0054127
	GBR	0.005969	6.5315e-05	0.0080818
	DTR	0.0049392	5.4937e-05	0.0074119
	SVR	0.0058981	4.2406e-05	0.0065121
	KNNR	0.0052350	3.8610e-05	0.0062137
Emergency Response	MLR	2.6155e-09	8.8448e-18	2.9740e-09
	GBR	2.6301e-07	1.0277e-13	3.2058e-07
	DTR	2.8198e-07	1.3504e-13	3.6747e-07
	SVR	3.3816e-07	1.3879e-13	3.7254e-07
	KNNR	5.9421e-08	1.0169e-14	1.0084e-07
HVAC	MLR	3.0885e-09	1.6375e-17	4.0467e-09
	GBR	6.8290e-08	7.8697e-15	8.8711e-08
	DTR	1.0306e-07	1.6541e-14	1.2861e-07
	SVR	9.5958e-08	1.3207e-14	1.1492e-07
	KNNR	2.0620e-08	1.1094e-15	3.3308e-08
Smart Lightning	MLR	4.5123e-09	4.1255e-17	6.4230e-09
	GBR	4.5025e-08	3.4404e-15	5.8655e-08
	DTR	9.1376e-08	1.7043e-14	1.3055e-07
	SVR	5.9834e-08	6.2366e-15	7.8972e-08
	KNNR	3.7146e-08	2.8440e-15	5.3329e-08
VoIP virtual assistance	MLR	0.0002211	7.2603e-08	0.0002694
	GBR	0.0001546	2.8749e-08	0.0001695
	DTR	9.6463e-05	1.3520e-08	0.0001163
	SVR	0.0001392	2.3750e-08	0.0001541
	KNNR	0.0001483	2.7738e-08	0.0001665
Video Surveillance	MLR	0.0012548	3.2583e-06	0.0018050
	GBR	0.0018372	9.6768e-06	0.0031107
	DTR	0.0019151	0.0000103	0.0032104
	SVR	0.0086434	0.0000827	0.0090953
	KNNR	0.0013595	5.0101e-06	0.0022383

C. Performance Comparison and Discussion

In Table III, we provide the regression results using the above error metrics. However, we also plotted the same results using MAE and MSE in a different format to make it easier to compare the different algorithms for each application as shown in Fig. 2 and Fig. 3. To extract the results, we firstly performed an application-wise comparison by applying all algorithms per application to see which application gives the best regression results in term of MAE, MSE and RMSE. Following, we compared the five algorithms within each application to see which regressor gives the best accuracy for each application.

Specifically, in Table III, we have grouped the regression models into different classes corresponding to the 6 applications. Firstly, we perform the comparison between applications using the error metrics presented above. We can observe that regression models in the classes: emergency response, HVAC and smart lightning have obtained the best results by considering the MAE, MSE and RMSE, while the models for augmented reality class obtained worst results followed by the video surveillance and VoIP classes for all error metrics.

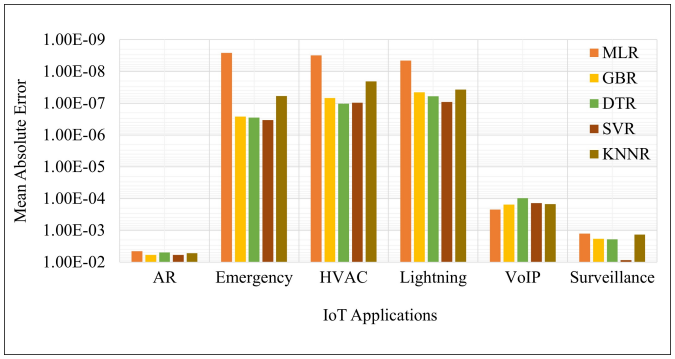

Fig. 2. Comparison of MAE results for all algorithms

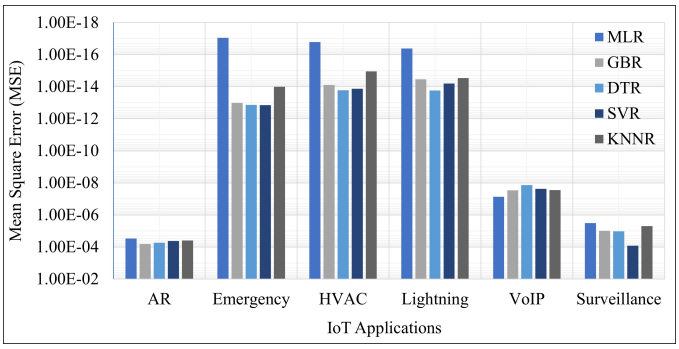

Fig. 3. Comparison of MSE results for all algorithms

The reason for having less accurate results for augmented reality, VoIP and surveillance applications are the following: (i) the number of sensor nodes within these applications are less as compared to other highly accurate applications; (ii) the duration of traffic produced for such application is also less compared to other applications as shown in Table I. However, ML algorithms require a lot of training data to give a good prediction; (iii) the predicted throughput may present a skewed behavior and therefore, skewness can affect the regression model's prediction accuracy for such applications (classes).

Following, we concentrate on the comparison of the best predicted application's throughput i.e., emergency, lightning and HVAC, and we focus on Fig. 2 and 3 illustrations. Specifically, Fig. 2 shows the prediction accuracy of throughput by plotting the MAE of the different regression models for each IoT application. For the emergency response, HVAC and smart lightning applications, we observe that the MLR provides the best performance in terms of MAE. The reason that MLR provides good results compared to other algorithms is that firstly, it deals better with linear dependencies and secondly, it is a parameteric approach so it works well despite the size of the dataset. For the latter, a ML model can be either parametric or non-parameteric. Parameteric models assume some finite set of parameters while non-parameteric models assume that the data distribution cannot be defined in terms of such a finite set of parameters, and thus the size of the dataset.

After that there is almost one to two orders of magnitude difference with the KNNR and even more with the GBR and DTR. Finally, the SVR gives the worst results, since the particular algorithm is not that suitable for large datasets and the particular applications contains a large number of devices compared to other applications (see Table I).

Fig. 3 presents the algorithm comparison with respect to MSE. Once more the MLR with MSE of 8.84E-18, 1.64E-17 and 4.13E-17, for the three applications under consideration, outperforms all other algorithms. Next is the KNNR that achieves a MSE of 1.02E-14, 1.11E-15, 2.84E-15 for all three applications. It is worth noting that we have also conducted experiments with Lasso and Ridge regression that are regularized versions of the MLR. However, they had a degradation close to 22% for MAE, 36% for MSE and 16% for RMSE and we decided not to illustrate them.

For our prediction problem, it is not only important to achieve high accuracy, but to also get small training and inference (prediction) times. Both times are measured in seconds. Fig. 4 and Fig. 5 illustrate the training time and prediction time of all regression models applied to each application category. It is interesting to see that SVR takes the least training time for almost all applications i.e., 0.00097s for AR, 0.00103s for emergency, 0.00100s for HVAC, 0.00200s for lighting, 0.00103s for VoIP and 0.00099s for surveillance as compared to all other algorithms. However, its prediction accuracy is worse as described above. Further, the GBR provides the highest training time for all applications. The reason is because it uses a large number of estimators during the training time, which helps to better learn the data in the expense of high training times. It is also to be noted that we used the default value of the n-estimator parameter for GBR (i.e., 100) for all applications. However, the prediction performance of GBR is not that bad and ranking third just after linear regression and KNNR for the top three applications.

Regarding the time to predict the throughput (inference time), the best models are the SVR for emergency response and VoIP applications, DTR for lighting and surveillance applications, GBR for AR and both SVR and DTR for HVAC. Focusing on the different results, we conclude that the throughput can be best predicted for the three applications: emergency response, HVAC and smart lighting by using the MLR, KNNR and GBR approaches. Finally, the SVR is the best in terms of training time. However, the inference time of algorithms differs from application to application.

Lastly, in Table IV we provide the predicted versus actual values of the average throughput for each application and for each regressor. It should be noted that initially throughput values are predicted per node for each application before computing the average throughput per application in Mbps. We can see that there are no significant differences in the throughput values predicted by the regressors and actual throughput. However, MLR provides the highest accuracy for most of the applications. For example, MLR provides the best predicted throughput value for VoIP, Smart Lighting, HVAC and emergency response. For video surveillance, SVR gives the highest throughput accuracy followed by MLR. Finally, for augmented reality, DTR predicts better the throughput value.

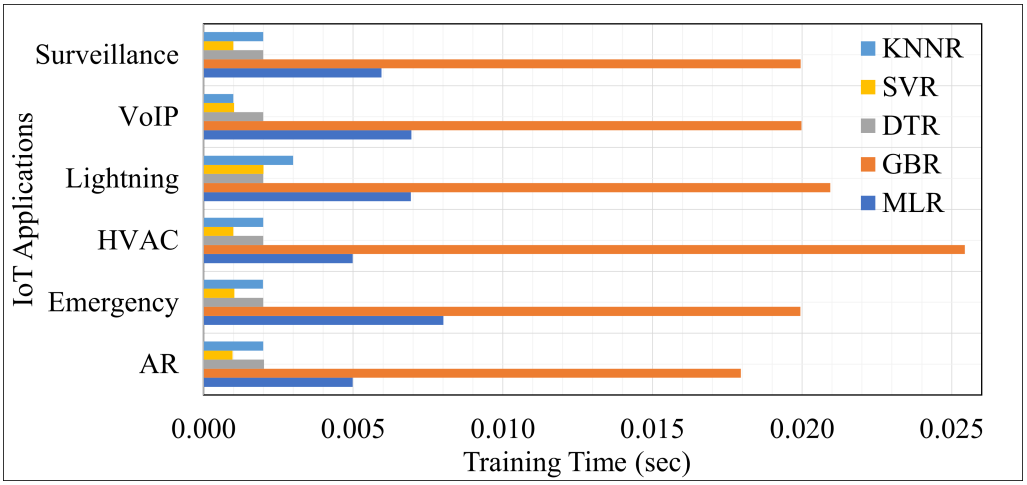

Fig. 4. Comparison of training times for all algorithms

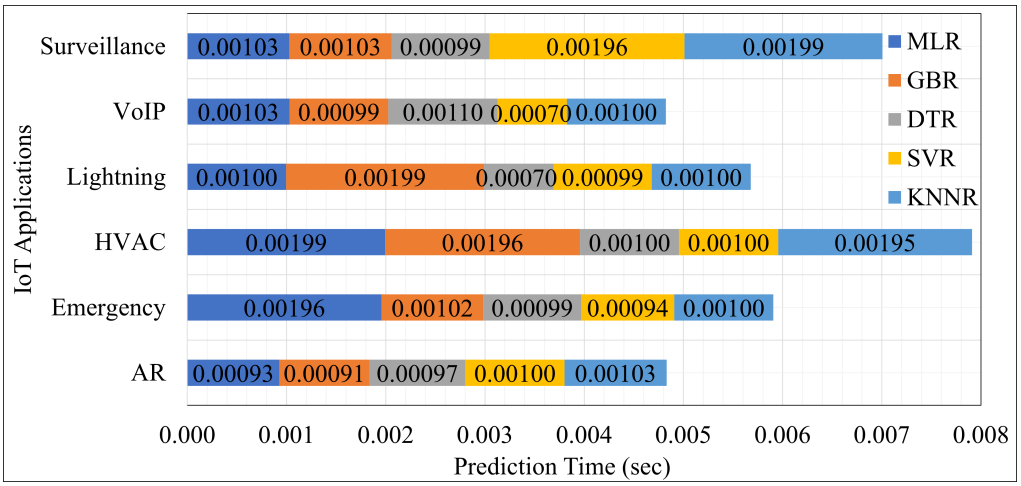

Fig. 5. Comparison of prediction times for all algorithms

TABLE IV
PREDICTED VS. ACTUAL AVERAGE THROUGHPUT VALUE PER APPLICATION IN MBPS

Application	MLR	GBR	DTR	SVR	KNNR	Actual
Video Surveillance	0.0987812848	0.0980375623	0.0981972347	0.0994706215	0.0987835169	0.1030300316
Voice over IP	0.0149721937	0.0150054310	0.0149943455	0.0151100497	0.0151578104	0.0149655950
Augmented Reality	0.1928382299	0.1986622426	0.1986393116	0.1940642805	0.1943529015	0.1968919514
Smart Lightning	0.0000023553	0.0000023423	0.0000023554	0.0000024036	0.0000023642	0.0000023552
HVAC	0.0000045310	0.0000045188	0.0000045444	0.0000044814	0.0000045264	0.0000045370
Emergency Response	0.0000339790	0.0000337916	0.0000338419	0.0000341214	0.0000339717	0.0000339787

VI. CONCLUSION AND FUTURE WORK

In this paper, we have analyzed the performance of several machine learning regression techniques to predict the throughput per device for 6 different IoT applications. To do so, we generated a dataset containing raw features in a real smart building environment. After a basic preprocessing of the data, the final features are extracted. Finally, the comparison of the regression techniques was performed in terms of prediction errors and time analysis. The application level comparison showed that emergency, HVAC and lighting applications can achieve good results in term of all prediction errors. However, linear regression is the algorithm that performed better than all other algorithms while SVR is the algorithm with the least training time. Future direction of this work includes incorporation of feature selection methods such as correlation based feature selection and mutual information based feature selection along with prediction of other QoS metrics such as delay.

REFERENCES

[1] A. Hameed and A. Leivadeas, "Iot traffic multi-classification using network and statistical features in a smart environment," in *IEEE Int. Workshop on Comp. Aided Modeling and Design of Comm. Links and Networks (CAMAD)*, 2020, pp. 1–7.

[2] A. Pekar, J. Mocnej, W. K. G. Seah, and I. Zolotova, "Application domain-based overview of iot network traffic characteristics," *ACM Comput. Surv.*, vol. 53, no. 4, Jul. 2020.

[3] C. Adjih *et al.*, "Fit iot-lab: A large scale open experimental iot testbed," in *IEEE World Forum on IoT (WF-IoT)*, 2015, pp. 459–464.

[4] A. Akbar, A. Khan, F. Carrez, and K. Moessner, "Predictive analytics for complex iot data streams," *IEEE Internet of Things J.*, vol. 4, no. 5, pp. 1571–1582, 2017.

[5] A. R. Abdellah, V. Artem, A. Muthanna, D. Gallyamov, and A. Koucheryavy, "Deep learning for iot traffic prediction based on edge computing," in *Distributed Computer and Communication Networks: Control, Computation, Communications*, 2020, pp. 18–29.

[6] V. Artem, A. A. Ateya, A. Muthanna, and A. Koucheryavy, "Novel ai-based scheme for traffic detection and recognition in 5g based networks," in *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*, 2019, pp. 243–255.

[7] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Neural network architecture based on gradient boosting for iot traffic prediction," *Future Generation Computer Systems*, vol. 100, pp. 656–673, 2019.

[8] A. R. Abdellah, O. Abdulkareem Mahmood, and A. Koucheryavy, "Delay prediction in iot using machine learning approach," in *Int. Con. on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 2020, pp. 275–279.

[9] M. Ateeq, F. Ishmanov, M. K. Afzal, and M. Naeem, "Predicting delay in iot using deep learning: A multiparametric approach," *IEEE Access*, vol. 7, pp. 62 022–62 031, 2019.

[10] G. White, A. Palade, C. Cabrera, and S. Clarke, "IoTPredict: Collaborative qos prediction in iot," in *IEEE Int. Conf. on Pervasive Computing and Communications (PerCom)*, 2018, pp. 1–10.

[11] ——, "Quantitative evaluation of qos prediction in iot," in *IEEE/IFIP Int. Conf. on Dependable Systems and Networks*, 2017, pp. 61–66.

[12] M. Batra and R. Agrawal, "Comparative Analysis of Decision Tree Algorithms," in *Nature Inspired Computing*, ser. Advances in Intelligent Systems and Computing. Singapore: Springer, 2018, pp. 31–36.

[13] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001, publisher: Institute of Mathematical Statistics.

[14] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, Aug. 2004.

[15] N. S. Altman, "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, Aug. 1992, publisher: Taylor & Francis.

[16] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*. John Wiley & Sons, Jan. 2021.