Hoeffding Regression Trees for Forecasting Quality of Experience in B5G/6G Networks

José Luis Corcuera Bárcena, Pietro Ducange, Francesco Marcelloni, Alessandro Renda and Fabrizio Ruffini

[1]*Department of Information Engineering, University of Pisa, Largo Lucio Lazzarino 1, 56122 Pisa, Italy*

Abstract

Online data stream analysis is becoming more and more relevant, as the focus of daily life analyses shifts from offline processing to real-time acquisition and modeling of massive data from remote devices. In this paper, we focus our attention on the domain of telecommunications, in particular the video streaming services for moving devices (e.g., a passenger enjoying a movie during a car trip). Since the streaming service must provide a satisfactory level of quality of experience to the user, it is important to predict incoming problems on video quality. We used the well-known Hoeffding Decision Tree (HDT) for streaming data, tailored to regression problems, and we compared its performance with standard Regression Trees (RTs) to evaluate the potentiality of HDTs to forecast the quality of experience in terms of accuracy, time for learning, and memory used. Results show that, during the online learning process, the standard RT outperforms HDT in terms of accuracy, but is prone to under-performance in terms of timings and memory when applied to potentially massive data streaming scenarios.

Keywords

Data Stream Mining, Regression Tree, QoE forecasting, Explainable AI, Hoeffding Decision Tree

1. Introduction

Quality of Experience (QoE) is a measure of end-user satisfaction in enjoying a service and is typically used in the context of telecommunications [1]. The fulfillment of QoE metrics is a primary goal in current, i.e., fourth and fifth generations, and future mobile networks. Beyond 5G (B5G) and 6G networks are indeed currently under development as pointed out, for instance, by the commitment of institutions, industry and academia in the framework of international projects such as Hexa-X[1]. Such next generation wireless networks are expected to be much more complex than current ones and will support innovative functionalities such as holographic communication, high precision manufacturing, and smart automotive applications [2]. Notably, the capability to play high-definition videos in real-time may represent a key enabler toward such new functionalities. Thus, being able to forecast the perceived quality of video experience

OLUD 2022: First Workshop on Online Learning from Uncertain Data Streams, July 18, 2022, Padua, Italy

joseluis.corcuera@phd.unipi.it (J. Corcuera Bárcena); pietro.ducange@unipi.it (P. Ducange); francesco.marcelloni@unipi.it (F. Marcelloni); alessandro.renda@unipi.it (A. Renda); fabrizio.ruffini@ing.unipi.it (F. Ruffini)

0000-0002-9984-1904 (J. Corcuera Bárcena); 0000-0003-4510-1350 (P. Ducange); 0000-0002-5895-876X (F. Marcelloni); 0000-0002-0482-5048 (A. Renda); 0000-0001-6328-4360 (F. Ruffini)

[1]https://hexa-x.eu/, accessed June 2022

may be fundamental to avoid the degradation of end-users' satisfaction or to determine whether a specific functionality should be provided or not.

In the context of video streaming services, QoE metrics include startup delay, rebuffering events and video quality [1] which clearly depend on contextual factors and typically vary over time; several works [1, 3, 4] have recently addressed the QoE prediction task by exploiting Machine Learning (ML) techniques and leveraging Quality of Service (QoS) metrics, i.e., quantitative measures that characterize the service offered by the network, such as packet loss and channel quality. Interestingly, only one of these works [4] has framed the issue of QoE prediction as a timeseries forecasting problem, yet disregarding important challenges of data stream mining: the whole dataset is typically not available for offline processing and the distribution of data may change over time due to a phenomenon known as *concept drift* [5], making it essential to adapt the model to avoid performance degradation.

In the last decades, various approaches for incremental learning of ML models have been proposed; here, we focus on the field of eXplainable Artificial Intelligence (XAI) and specifically on a class of *inherently interpretable* models, capable of explaining, by design, how decisions have been taken. Indeed, transparency (i.e., the capability of understanding the structure of the model itself) represents a key requirement towards *trustworthy* AI (AI) [6] which in turn is deemed as a major pillar in the design of next generation wireless networks. In this framework, the Hoeffding Decision Tree (HDT) [7] represents a reference approach: it has been widely exploited for both classification and regression tasks. In the context of classification tasks, HDT has also recently been extended with fuzziness to handle vague and noisy data and enhance interpretability [8].

In this paper, we present a preliminary experimental evaluation of the Hoeffding Regression Tree (HRT) for a QoE forecasting task in the frame of next generation wireless networks: specifically, we resort on a recently published QoS-QoE forecasting dataset and compare the performance of HRT and classical Regression Tree (RT) from different perspectives: modelling capability, training time and memory required.

The rest of the paper is organized as follows: in Section 2 we summarize the key aspects of HRT model; in Section 3 we describe the experimental setup, highlighting the different learning schemes being compared and the evaluation strategies. Section 4 reports the experimental results, whereas in Section 5 we analyze the robustness of the HRT model to hyperparameter configuration. Finally, Section 6 draws some conclusions.

2. Hoeffding Regression Tree: background

HDT, also known as "Very Fast Decision Tree" [7], is a reference model to solve classification problems over an input data stream. In a nutshell, it allows growing a binary decision tree incrementally: a leaf is considered for a split only if it contains a minimum number of samples and a condition based on the Hoeffding's theorem is met. The theorem guarantees, within a certain level of confidence, that the selected attribute would have been the same in the case of an infinite number of available samples. In the case of classification, the condition is met when the difference between the two highest values of the information gains computed for the attributes available at the leaf node is higher than a bound, dubbed the Hoeffding's bound.

Although the adoption of the Hoeffding's theorem in relation to the splitting criterion has received some criticism [9], HDT generally provides satisfactory results and can be regarded as a valid heuristic method.

HRT represents an adaptation of HDT to solve regression problems given an input data stream. Unlike its classification counterpart, HRT relies on calculating the reduction of variance of the target variable to decide among the splitting candidates. Let $\Delta\text{Var}(a)$ and $\Delta\text{Var}(b)$ be the reduction of variance associated to the best and the second best splitting attribute, respectively. The Hoeffding condition, for a leaf node L, is defined as follows:

$$\frac{\Delta\text{Var}(b)}{\Delta\text{Var}(a)} < 1 - \varepsilon_L \tag{1}$$

and the term ε_L, i.e. the Hoeffding bound for the leaf node L, is evaluated according to the following equation:

$$\varepsilon_L = \sqrt{\frac{\ln(1/\delta)}{2N_L}} \tag{2}$$

where δ (split confidence) is equal to 1 minus the desired probability of choosing the correct attribute, and N_L is the number of samples in node L.

The value assigned to a leaf node is the average of the target values of the training samples contained in the leaf node, and, given an incoming input sample, is used to predict the output at inference time. As any tree-based model, HRT features a high level of interpretability, which is a crucial requirement in many applications, including those within next generation wireless networks. Thus, we adopt HRT for tackling our QoE forecasting problem, leveraging an implementation available in the `scikit-multiflow` library [10].

3. Experimental analysis

In this section, we first introduce the problem and the dataset; then, we describe the models and learning schemes involved in the experimental comparison. Finally, we provide details about the experimental setup.

3.1. Problem description: the QoE forecasting dataset

As the scenario of our investigation, we consider the publicly available QoS-QoE forecasting dataset[2], introduced in one of our previous works [4]. A client-server video-streaming application is simulated within Simu5G [11], a dedicated open-source *model library* for realistic 5G network simulations: while experiencing the video, each of the 15 simulated clients, also referred to as user equipment (UE), measures or collects a set of time-tagged QoS and QoE metrics. We formulate the QoE prediction task as in [4], replicating the preprocessing and features extraction steps. Specifically, a simulation lasts approximately 120 seconds: for each user, during such time frame, we collect the timeseries related to 12 metrics (QoS, QoE and

[2] http://www.iet.unipi.it/g.nardini/ai6g_qoe_dataset.html, accessed June 2022

contextual). Then, we obtain any tuple of the preprocessed dataset as follows: for a timestamp t, the input variables consist in 11 statistics (i.e., mean, median, max, min, variance, standard deviation, kurtosis, skewness, Q1 and Q3, number of samples) measured for each metric in the time window $[t - W, t]$ (with $W = 10s$), whereas the output variable consists in the mean of the target QoE metric over the time horizon of one second (i.e., in $[t, t + H]$, with $H = 1s$). As the target QoE metric, we consider the average percentage of arrived frames at the time of its display. The subsequent tuple is obtained by sliding the two windows W and H with a step of 1 second. To summarize, each instance in the dataset is represented in $\mathbb{R}^{132}$, resulting from 11 statistics evaluated over window of size W on 12 timeseries. The 120-seconds video-streaming simulation is repeated 24 times.

We consider the following setting: we aim to learn the mapping between QoS and QoE in order to tackle the QoE forecasting problem. We assume that the data generated by different UEs within a simulation can be gathered for training the model; however, the data from the various simulations are not immediately available but arrive in chunks, each corresponding to one of the 24 simulations. Basically, one can think of the 24 simulations as representing temporally consecutive scenarios in which each of the various UEs experiences, from time to time, different situations. The overall dataset consists of 28758 samples, with a chunk size ranging from 972 to 1466 samples (the variability is induced by the removal of missing values). Such a setting demands for ad-hoc strategies for incremental model training: in the following, we describe two learning schemes based on classical RT and HRT models, respectively, along with the evaluation strategies adopted for assessing the performance of the models.

3.2. Learning schemes and evaluation strategies

Let $chunk_i$ indicate the chunk of data of the i-th simulation, with $i = 1, 2, \ldots, 24$. Each $chunk_i$ contains the samples of all the UEs from the i-th simulation. We compare two learning schemes using two evaluation strategies.

Learning schemes. **HRT** supports an incremental learning scheme: it consists in updating the model at each incoming chunk. In other words, at each step i the model is updated considering only the current $chunk_i$. Conversely, the classical **RT** does not support an incremental learning scheme: the model is retrained from scratch at each newly collected chunk of data. At each step i the previous model is replaced with a new one trained on $\bigcup_{j=1}^{i} chunk_j$, i.e., the union of the chunks collected so far.

Evaluation strategies. Both learning schemes are evaluated using two approaches, widely adopted in data stream applications. **Prequential evaluation**, or interleaved-test-then-train, can be formalized as follows: once a new $chunk_i$ is collected (with $i = 2, \ldots, 24$) we first assess the performance of the current model on $chunk_i$ and then exploit it to train/update the model. For example, the first evaluation step consists in using the first chunk ($chunk_1$) for training and the $chunk_2$ for testing. **Hold-out evaluation** consists in assessing the performance of a model after updating it using each $chunk_i$ on a fixed test set. To carry out this experiment, we assume that 4 chunks are immediately available as test set (specifically: $chunk_{21}$, $chunk_{22}$, $chunk_{23}$,

and $chunk_{24}$). At each step of the analysis the updated model will always be tested on the same data.

To summarize, in our experimental campaign, we refer to the various approaches using the following notation:

- **HRT-preq** indicates the HRT model, i.e., incremental learning scheme, evaluated using the prequential strategy.
- **HRT-hold-out** indicates the HRT model, i.e., incremental learning scheme, evaluated using the hold-out strategy.
- **RT-preq** indicates the RT model, i.e., retraining learning scheme, evaluated using the prequential strategy.
- **RT-hold-out** indicates the RT model, i.e., retraining learning scheme, evaluated using the hold-out strategy.

3.3. Experimental setup

Both HRT and classical RT have publicly available Python implementations: HRT is available in `scikit-multiflow`[3], whereas the classical RT is implemented in `scikit-learn`[4]. Tables 1 and 2 report the values of the main configuration parameters for HRT and RT models. As per the former, we adopt the default parameter configuration, whereas for the latter we set the parameters coherently with our previous study [4], pursuing a fair comparison of the results.

We executed our experiments on a computer featuring an x86_64 architecture, 16 cores, Intel Xeon Processor (Cascadelake) - 2.194*GHz* and 32*GB* RAM.

Table 1
HRT configuration parameters.

Parameter	Value
Split confidence	10^{-7}
Tie threshold	0.05
Grace period	200

Table 2
RT configuration parameters.

Parameter	Value
Max. depth	10
Split criterion	MSE
Min_samples_split	0.01
Min_samples_leaf	0.001

4. Experimental Results

In this section, we report the results of our experimental analysis from a threefold perspective: regression metrics, memory used, and time for learning/updating the model.

[3] https://scikit-multiflow.readthedocs.io/en/latest/api/generated/skmultiflow.trees.HoeffdingTreeRegressor.html, accessed June 2022

[4] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html, accessed June 2022

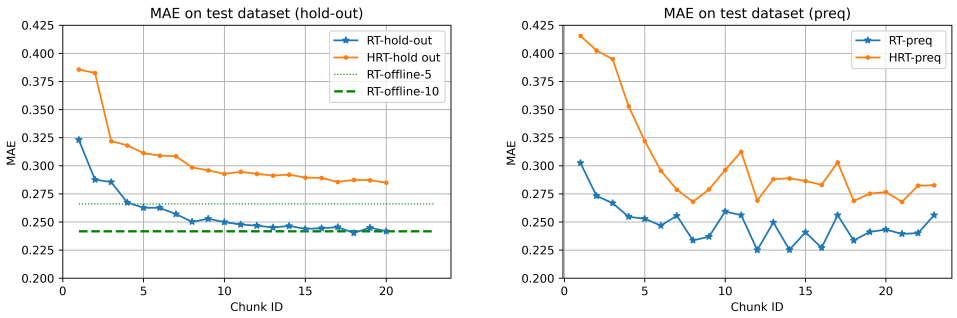
4.1. Regression metrics and model complexity

In the following, we compare the HRT models with their RT counterparts, considering the prequential and the hold-out evaluation strategies independently.

Figure 1 shows the trends of the Mean Absolute Error (MAE) along the sequence of processed chunks considering the two evaluation strategies, namely *hold-out* (Fig. 1a) and *prequential* (Fig. 1b). In the *hold-out* setting, two "offline" versions of the decision tree described in [4] are considered as reference baselines. These models are generated considering a global training dataset composed by all training chunks (ie, $chunk_1$ to $chunk_{20}$). The results of these "offline" decision trees (RT-offline-5 and RT-offline-10, induced by setting the maximum depth at 5 and 10, respectively) are obtained evaluating the models on the same hold-out test set made up of the last 4 chunks of the dataset.

In general, we can observe how the RT-models outperform their HRT counterparts along the whole model updating process in streaming. As regards HRT-hold-out model (Fig. 1a), after an initial phase of "start-up" corresponding more or less to the first five chunks, it reaches a plateau in performance on the hold-out test set, approaching, but unfortunately not reaching, the performance of the baseline models.

As for the prequential evaluation strategy (Fig. 1b), HRT-prequential closely trails the performance of RT-prequential: again, however, re-training the traditional model leads to consistently superior performance compared to incremental training of HRT.



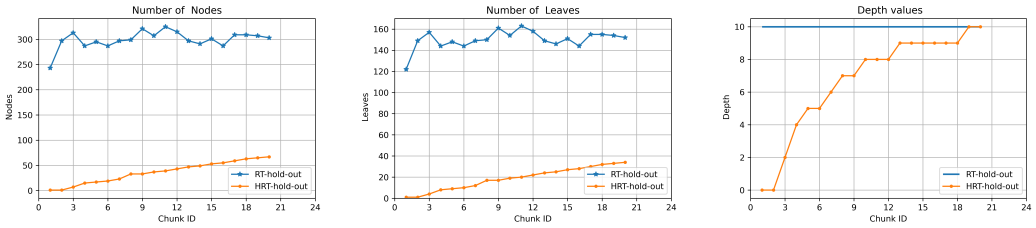
(a) Hold-out evaluation strategy

(b) Prequential evaluation strategy

Figure 1: MAE metrics measured on the test set.

In the following, we discuss in details the trend of the complexity for both the HRT and RT models. As the training stage is analogous among hold-out and prequential setting (at least for the first 20 chunks), we just consider the former, but the same considerations apply for the latter. Figure 2 shows the trends of the complexity of the HRT-hold-out and RT-hold-out along the sequence of processed chunks. As regards the number of nodes (Fig. 2a) and the number of leaves (Fig. 2b), it is worth noticing that RT-hold-out is always more complex than the HRT-hold-out, up to one order of magnitude. We can observe that the RT models entail a large number of nodes even at the first chunks, while the number of nodes in the HRT models keep steadily increasing almost linearly with the number of chunks. The depth of the tree (Fig. 2c) is relevant as well, since it is associated with maximum number of conditions in the antecedent of

the rules that can be extracted from the trees: we can observe that the HRT-hold-out reaches the same depth of RT-hold-out just at the end of the stream of chunks. We recall that, to obtain an easier comparison, we constrained the RT models to have the maximum depth equals to 10 as per the best model we found in the previous study reported in [4].



(a) Number of nodes (b) Number of leaves (c) Tree depth

Figure 2: Hold-out models complexity.

Tables 3 and 4 reports the performance of the models for the hold-out and prequential evaluation strategy, respectively, after training the models up to the final available chunk (from $chunk_1$ to $chunk_{20}$, in the case of hold-out, and from $chunk_1$ to $chunk_{23}$ in the case of prequential). The performance of the models are measured in terms of Mean Squared Error (MSE), MAE, and coefficient of determination (R^2). Furthermore, we report the complexity of the model measured in terms of number of nodes, leaves, maximum depth, and number of features selected by the induced tree.

Table 3
Global results and model complexity for HRT-hold-out and RT-hold-out approaches. Regression is evaluated on test sets. The different baselines refer to different models where we fixed the maximum depth to 5 (i.e., RT-offline-5) and 10 (i.e., RT-offline-10, the best result from [4]).

model	**Regression metrics**			**Model Complexity**			
	MSE	MAE	R^2	Nodes	Leaves	Features selected	Max depth
HRT-hold-out	0.120	0.285	0.300	67	34	22	10
RT-hold-out	0.102	0.242	0.407	303	152	65	10
RT-offline-10	0.102	0.242	0.407	303	152	65	10
RT-offline-5	0.111	0.266	0.357	57	29	18	5

Obviously, the results obtained with the RT-hold-out learning scheme after processing the final chunk (i.e., $chunk_{20}$) are equivalent to those obtained with the more complex among the two baselines, namely RT-offline-10: in fact, the last step of the RT-hold-out strategy is essentially the same scenario as the baseline strategy where the whole dataset (from $chunk_1$ to $chunk_{20}$) is used for training.

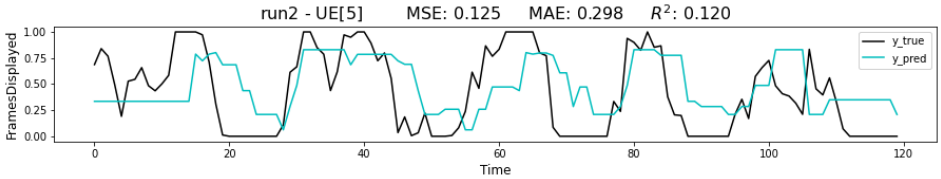
In general, results confirm that the HRT-hold-out and HRT-preq strategies are characterized by a worse performance in terms of MAE, MSE, and R^2 than their RT-counterparts. However, HRT models are characterized by the lowest levels of complexity, in terms of number of nodes, number of leaves, number of selected features and maximum depth of the trees, thus ensuring a

Table 4
Global results and model complexity for HRT and RT prequential (preq) approaches. Regression metrics are evaluated on the last testing chunk (i.e., $chunk_{24}$).

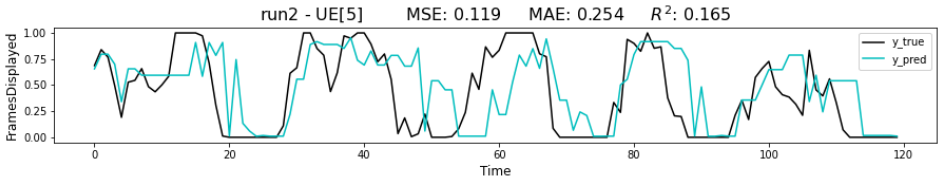
model	Regression metrics			Model Complexity			
	MSE	MAE	R^2	Nodes	Leaves	Features selected	max depth
HRT-preq	0.122	0.283	0.264	77	39	22	9
RT-preq	0.111	0.256	0.332	305	153	56	10

higher level of interpretability than RT models.

Figures 3 and 4 report examples of QoE test timeseries for different UEs, overlapping the ground-truth with the predicted values obtained by the different models in the test datasets, after processing the last available chunk of training data. The visual analysis suggests that the different models provide reasonable predictions in different conditions; in particular, the HRT-based models show a worse predictive performance than their RT counterparts, possibly due to their lower complexity.



(a) HRT-hold-out



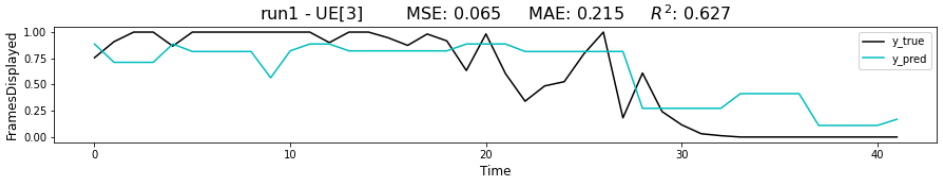
(b) RT-hold-out

Figure 3: Real and predicted values of QoE for an example UE of the test set: hold-out evaluation strategy.

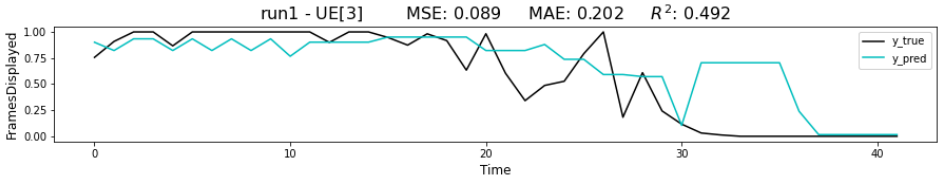
To summarize, the accuracies obtained by the HRT models are smaller by a 10-26% (depending on the MAE, MSE or R^2 metric considered) than the corresponding RT counterpart models. However, this decrease in performance is counter-balanced by the time-for-learning and memory-used values, that are aspects of utmost importance in a streaming scenario; for this reason, they are detailed in the following.

4.2. Memory occupancy

Figure 5 shows the training set sizes (i.e., the number of samples) used for updating both the RT and HRT models when processing a new chunk of data. We just discuss the prequential



(a) HRT-prequential



(b) RT-prequential

Figure 4: Real and predicted values of QoE for one example UE of the test set: prequential evaluation strategy.

setting: in the hold-out strategy, the learning phase is analogous and the same considerations apply. As expected, the RT model memory occupancy rapidly exceeds the HRT model one: in a real-case where we have massive input data stream, this would lead to large training set sizes, thus making the retraining learning scheme an impractical and very computationally intensive approach.

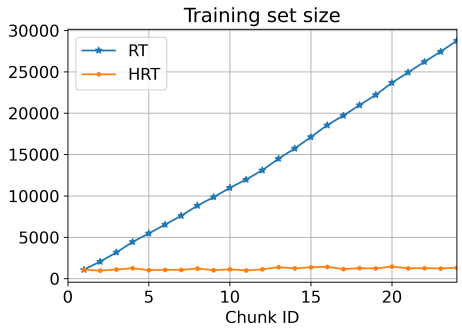


Figure 5: Training set sizes during the model updating process in streaming.

4.3. Time for model updating

Figure 6 reports the trends of the updating times for the RT and HRT models. Also in this case, we just discuss the prequential setting. The plot shows how, after about 22 chunks, the time for the RT learning exceeds the time for HRT learning. This is important, because for the HRT models we need to reduce as much as possible the dependency of the training time on the number of chunks, with the aim of ensuring minimum latency in the operative real-time application. In addition, for the RT-model we can observe an almost linear relationship between

the number of chunks and the learning time: in fact, a simple linear fitting on the trend related to RT-preq model yields R^2=0.99 and p-value=3.88e^{-23}. On the other hand, the HRT-models do not show a strong increasing behaviour in function of the chunk number, but it can depend on the dimension, namely the number of samples, of each single chunk.

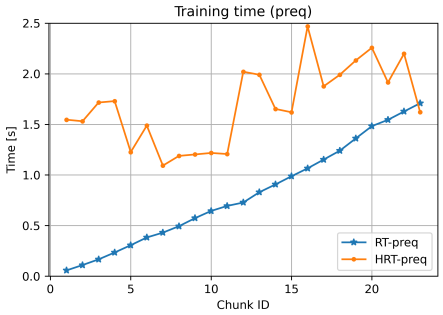


Figure 6: Training times of RT-prequential and HRT-prequential.

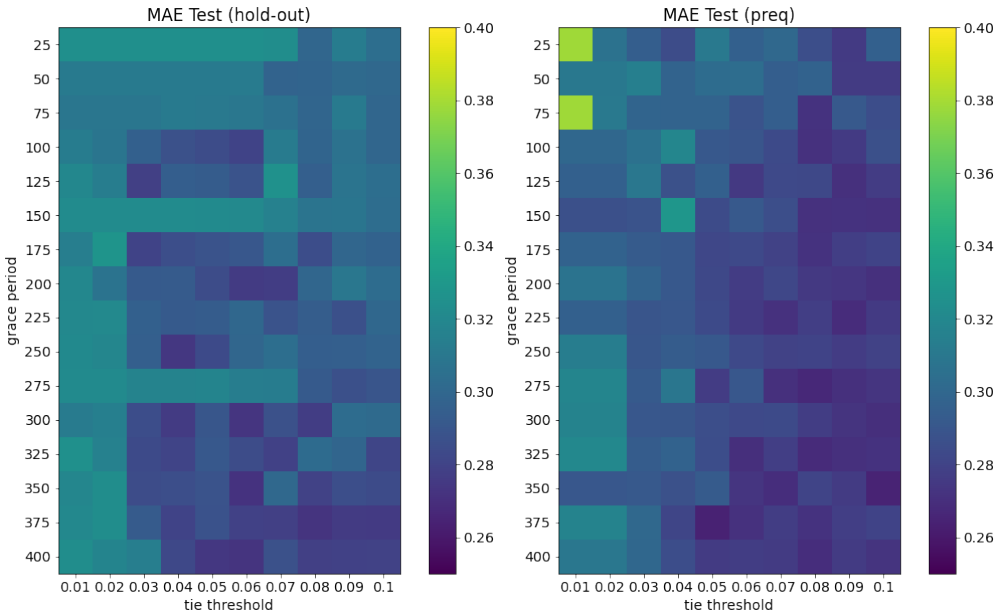
5. HRT sensitivity analysis

In this section we analyze the sensitivity of HRT models with respect to two aspects: parameter setting and order of chunks in the streaming process.

5.1. Sensitivity with respect to parameter configuration

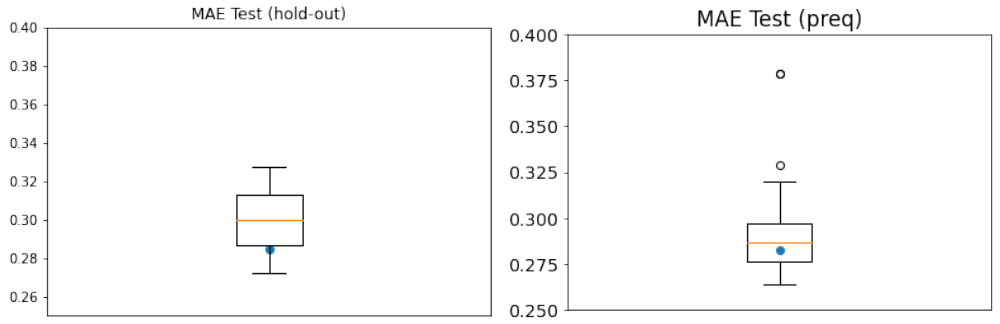
We analyzed the suitability of the default configuration of the HRT training in terms of model parameters, reported in Table 1. In particular, we compared the MAE values for different values of the grace period and of the tie-threshold, after the whole dataset has been incrementally processed. We aim to analyse if the default values of the model parameters are a "robust" choice, and ensure good performances. We recall that *grace period* defines a threshold on the number of instances contained in a leaf node before considering it for a split, whereas *tie threshold* consists in a threshold below which a split will be forced to break ties. Intuitively, lower values of grace period and higher values of tie threshold will foster easier node splitting, thus leading to more complex trees. Notably, hyperparameter tuning for finding optimal parameters configuration is not viable in an operative scenario, where the model cannot rely on a static dataset but rather learns from an incoming data stream.

Figures 7 and 8 show the MAE values on the test set for the HRT-models. It is worth highlighting that the value of the metric measured at the end of the training conveys only a partial insight into the behaviour of the model, but can still be considered a proxy for the quality of the parameter configuration. From the heatmaps, we can observe a slight indication of the presence of a better-performing area, in the bottom right of the plot, where the grace period and the tie-threshold have values greater than 300 and 0.08, respectively. The boxplots show how, for the default configuration (grace period=200 and tie-threshold=0.05), the value of the MAE score, even if not optimal, lies below the median values for both the evaluation strategies.



(a) HRT-hold-out (b) HRT-prequential

Figure 7: MAE results on the test set for different choices of model parameters for HRT models.



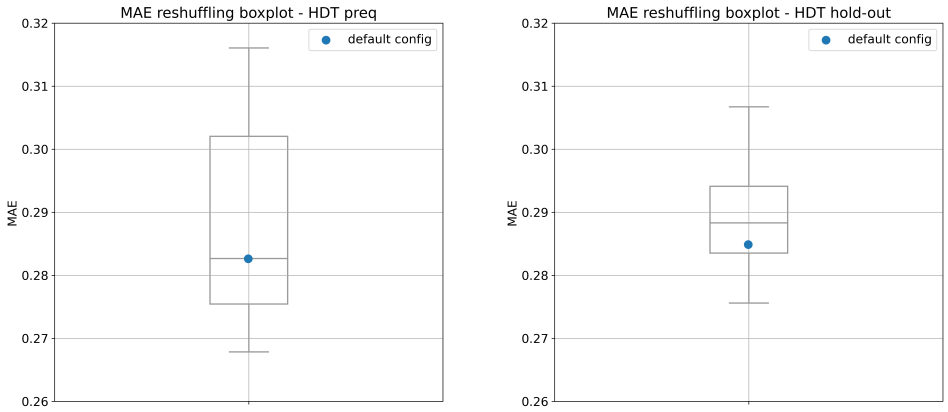
(a) HRT-hold-out (b) HRT-prequential

Figure 8: Boxplot of MAE values on the test set for different choices of model parameters for HRT models. The value obtained with the default configuration is marked with a blue dot.

5.2. Sensitivity with respect to chunk order

In HRT the initial structure of the model (e.g., the root) is determined based on the initial chunks and cannot be reassessed subsequently. As a consequence, the order of the chunks may impact on the performance of the model throughout the whole data stream. To quantify the performance variation, we performed ten tests where we randomly shuffled the input chunks order. Figure 9 shows the MAE values for the last test dataset (i.e., after processing $chunk_{23}$ for the prequential strategy and $chunk_{20}$ for the hold out strategy), suggesting that the order of

input chunks does not significatively affect the resulting performance: the maximum (minimum) MAE values are different of about 12% (5%) with respect to the median value of the distribution. Such variability is not negligible in absolute terms, but it is still comparable to the variations of MAE values we observe, for instance, during model training in the prequential case (see Fig. 1b), after the first "start-up" 5 chunks.



(a) HRT-prequential

(b) HRT-hold-out

Figure 9: Boxplot of MAE values for ten repetitions of the experiment with different random shuffling of chunks. Values obtained with the default order (analyzed in the rest of the paper) are marked with a blue dot.

6. Conclusion

In this paper, we have discussed an application of streaming methods to a realistic 5G network simulation for QoE forecasting. We applied a Hoeffding Decision Tree for data stream regression to predict incoming QoE, and we compared the results with standard regression trees. From the results, we observed that HRT models have proven to be better strategies regarding memory usage and learning time aspects, at the cost of having worse accuracies than RT models. However, we experimentally highlighted how HRT models, after an initial start-up phase where the models complexity increase, approach the performance of the standard RT models with comparable complexity. This can be explained by the kind of strategy used by the Hoeffding Decision Tree: by construction, the structure of the tree is strongly affected by the initial data input, and the resulting tree is typically more shallow with respect to "traditional" decision trees. These considerations represent the initial steps for future works, where ad-hoc methods could be designed to take the discussed shortcomings into account. In particular, a further study will aim to shed some light on the relationship between complexity and performance in the streaming approaches, by refining the tree updating strategy and investigating techniques to select appropriate parameter configurations. Furthermore, we plan to assess if concepts from

fuzzy set theory can help improve the performance of HRT models in this kind of applications.

Acknowledgments

This work has been partly funded by the Italian Ministry of University and Research (MIUR), in the framework of the Cross-Lab project (Departments of Excellence) and PON 2014-2021 "Research and Innovation", DM MUR 1062/2021, Project title: "Progettazione e sperimentazione di algoritmi di federated learning per data stream mining" and by the EU Commission through the H2020 projects Hexa-X (Grant no. 101015956).

References

[1] V. Vasilev, J. Leguay, S. Paris, L. Maggi, M. Debbah, Predicting QoE Factors with Machine Learning, in: 2018 IEEE Int'l Conf. on Communications (ICC), 2018, pp. 1–6. doi:10.1109/ICC.2018.8422609.

[2] K. Sheth, K. Patel, H. Shah, S. Tanwar, R. Gupta, N. Kumar, A taxonomy of AI techniques for 6G communication networks, COMPUT COMMUN 161 (2020) 279–303. doi:10.1016/j.comcom.2020.07.035.

[3] A. Renda, P. Ducange, G. Gallo, F. Marcelloni, XAI Models for Quality of Experience Prediction in Wireless Networks, in: 2021 IEEE Int'l Conf. on Fuzzy Systems (FUZZ-IEEE), 2021, pp. 1–6. doi:10.1109/FUZZ45933.2021.9494509.

[4] J. L. Corcuera Bárcena, P. Ducange, F. Marcelloni, G. Nardini, A. Noferi, A. Renda, G. Stea, A. Virdis, Towards Trustworthy AI for QoE prediction in B5G/6G Networks, in: First Int'l Workshop on Artificial Intelligence in Beyond 5G and 6G Wireless Networks (AI6G 2022), (accepted).

[5] J. a. Gama, I. Žliobaitè, A. Bifet, M. Pechenizkiy, A. Bouchachia, A Survey on Concept Drift Adaptation, ACM Comput. Surv. 46 (2014). doi:10.1145/2523813.

[6] Ethics Guidelines for Trustworthy AI, Technical Report, 2019. European Commission. High Level Expert Group on AI. https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai.

[7] P. Domingos, G. Hulten, Mining high-speed data streams, in: Proc. of the sixth ACM SIGKDD Int'l Conf. on Knowledge discovery and data mining, 2000, pp. 71–80.

[8] P. Ducange, F. Marcelloni, R. Pecori, Fuzzy Hoeffding Decision Tree for Data Stream Classification, INT J COMPUT INT SYS 14 (2021) 946–964. doi:10.2991/ijcis.d.210212.001.

[9] L. Rutkowski, L. Pietruczuk, P. Duda, M. Jaworski, Decision Trees for Mining Data Streams Based on the McDiarmid's Bound, IEEE T KNOWL DATA EN 25 (2013) 1272–1279. doi:10.1109/TKDE.2012.66.

[10] J. Montiel, J. Read, A. Bifet, T. Abdessalem, Scikit-multiflow: A multi-output streaming framework, J MACH LEARN RES 19 (2018) 2915–2914.

[11] G. Nardini, D. Sabella, G. Stea, P. Thakkar, A. Virdis, Simu5G–An OMNeT++ Library for End-to-End Performance Evaluation of 5G Networks, IEEE Access 8 (2020) 181176–181191. doi:10.1109/ACCESS.2020.3028550.