QoS Predictability in V2X Communication with Machine Learning

Darlan C. Moreira, Igor M. Guerreiro, Wanlu Sun, Charles C. Cavalcante and Diego A. Sousa

***Abstract*—An important use case in fifth generation systems are vehicular applications, where, reliability and low latency are the main requirements. In order to determine if a vehicular application can be used one can apply machine learning (ML) tools to determine if these constraints are met, which open questions such as "which data is available", "which features to use", "the quality of this prediction", etc. In this paper we address some aspects of predicting quality-of-service (QoS) in a cellular vehicular-to-everything scenario, where we employ supervised learning as well as the autoregressive integrated moving average filter to predict if a packet can be delivered within a desired latency window. Particularly, we are interested in the reliability of this prediction, including predicting if a packet generated some time ahead will be delivered in time. Such information is essential when asserting that a vehicular application can indeed be employed safely. We show via simulation results that ML can be used as a prediction tool in vehicular applications. For instance, QoS levels can be predicted two seconds ahead with 85 % reliability.**

***Index Terms*—C-V2X, QoS Prediction, Machine Learning**

I. Introduction

Vehicles have become one of the fastest growing type of connected devices after smart phones and tablets [1]. As a consequence, vehicular-to-everything (V2X) communication has attracted great interest due to its potential of improving traffic safety, reducing energy consumption, and enabling new services related to intelligent transportation systems, such as platooning, extended sensors, advanced driving, and remote driving [2]. In this regard, cellular-V2X (C-V2X) is about being a part of the scale and pace of global Long-term evolution (LTE) network deployment, with a clear evolution path into fifth generation (5G). A fundamental part of C-V2X is to expand the scope of conventional mobile networks to also include support of automotive industries.

Although different V2X applications have diverse transmission requirements in terms of latency, reliability, and data rate, the most challenging scenario lies in safety-related applications with stringent latency and reliability requirements [2]. In this case, two aspects are of particular interests for automotive industries. First, the network can predict in advance what level of quality-of-service (QoS) it can achieve. Second, the network can provide notification to a vehicle in advance so that the vehicle is able to act on the information that the agreed QoS cannot be maintained. These two aspects are referred to as predictability of QoS levels.

Machine learning (ML) has become widespread in recent years and there are many ML techniques developed in the literature [3]. In the context of prediction, ML techniques, such as supervised learning [4], have been considered to bring relevant added values. In [5] the authors seek to predict the instantaneous achievable throughput of a connection in order to adjust the quality of the delivered content.

Focusing on the QoS prediction aspect, this work assesses different off-the-shelf supervised learning tools [6] with respect to classifying whether the network can successfully deliver a periodic packet to a vehicle within some latency constraint. Such tools have provided adequate performance in terms of prediction error. Some of them, namely random forest and neural networks, significantly outperform a classical/baseline auto-regressive model, namely autoregressive integrated moving average (ARIMA) filter [7]. Differently from [5], in this work we are interested not only in an instantaneous prediction, but also in predicting future QoS levels, which arguably are more important in vehicular applications. We show that machine learning can in fact be used to check the availability of vehicular applications in C-V2X.

The rest of this paper is organized as follows. Section II describes the system model. Section III formulates the QoS prediction as a binary classification problem. Section IV described the ML aspects regarding features and training models. Finally, Section V shows numerical results, while Section VI brings the conclusions and some perspectives.

II. System Model

This work considers a typical urban scenario for C-V2X communication, namely Manhattan grid, where vehicles' trajectories and road configurations are well defined [8], [9]. Specifically, each vehicle has a mobility pattern so that it moves only in the streets, and it can turn left/right with 0.25 probability; otherwise, it goes straight.

The cellular network has a wrapped-around seven-site hexagon layout, each site comprising three cells. Each cell is covered by an antenna with two elements with a single cell-specific reference signal port. Moreover, each cell serves a plurality of vehicles equipped with a 2-element antenna using interference rejection combination receiver. Figure 1 illustrates the considered scenario and the layout of the Manhattan grid.

D. C. Moreira, I. M. Guerreiro, C. C. Cavalcante and D. S. Aguiar are with Wireless Telecom Research Group (GTEL), Federal University of Ceará, Brazil. Emails: {darlan,igor,charles,diego@gtel.ufc.br}; W. Sun is with Ericsson Research, Sweden. Email: wanlu.sun@ericsson.com;

The authors acknowledge the financial support from Ericsson Research (technical cooperation contract UFC.46), CNPq (grant 309472/2017-2) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior–Brasil (CAPES) - Finance Code 001.

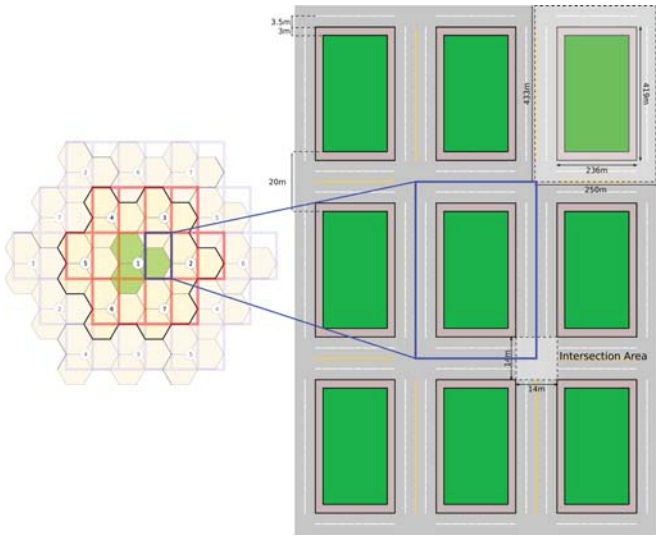

Fig. 1: Network with a wrapped-around seven-site hexagon layout, each site comprising three cells, on a Manhattan grid.

Vehicles, or simply user equipments (UEs), are assumed to be receiving data from the network in downlink unicast. The adopted channel model follows the 3rd Generation Partnership Project specification in [10] for low frequency ranges, namely the 2 GHz carrier with a 5 MHz bandwidth.

The employed traffic model corresponds to periodically sending packets of a fixed size and with a fixed transmit time interval. The data traffic undergoes predefined latency requirement with fixed delay threshold. That is, every sent packet is expected to be delivered with a delay no longer that the latency requirement. Herein, latency is defined from the point of view of the transport layer.

III. Problem Formulation

Communication reliability and low latency are the main requirements for safe vehicular applications in 5G systems. In this context, QoS prediction plays an important role in order to determine when the current state of the network allows some V2X use case, such as lane merging, platooning, etc.

We formulate the QoS prediction as a binary classification problem that aims at classifying whether a packet of size B can be successfully delivered from the base station (BS) to a vehicle within a latency requirement D. At a given time instant t_0, the network is assumed to have access to a set of measurements – from now so referred to as features – collected until then. Based on those features, the network predicts if a packet to be transmitted after some time gap can be delivered within its corresponding latency window. That is, a time gap G means predicting if a packet that will be generated in time $t_0 + G \times D$ will be delivered before $t_0 + (G + 1) \times D$, as illustrated in Figure 2.

In what follows, different solutions for the QoS problem described above are presented.

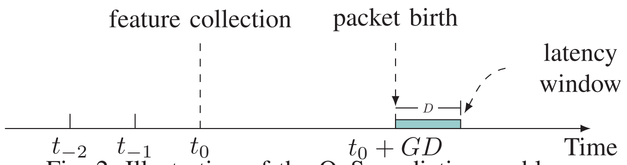

Fig. 2: Illustration of the QoS prediction problem.

IV. Machine Learning Approach

To predict the QoS at a destination vehicle, we employ supervised learning tools with labels defined as:

- **In Time** (or 1): if the packet was delivered within the required latency window;
- **Late** (or 0): if the packet delay exceeded the latency requirement or was lost.

The employed ML algorithms include linear regression (LR), multilayer perceptron (MLP) and random forest (RF), which are compared to the ARIMA filter.

Several different measurements can be employed for this QoS prediction problem. The employed ones are listed below:

- The previous 5 delays;
- The UE x-coordinate value;
- The UE y-coordinate value;
- The reference signal received power (RSRP) value;
- The reference signal received quality (RSRQ) value;
- The index of the best cell for the UE;
- The downlink signal-to-noise ratio (SINR);
- The averaged channel quality indicator (CQI);
- The average hybrid automatic repeat request (HARQ) block error rate (BLER);
- The average throughput of the UE serving cell;
- The load of the UE serving cell;
- The index of the serving cell.

All these features are preprocessed with a “mean inputer”, which replaces any missing values with the mean of available values for that feature, followed by a “max abs scaler”, which normalizes each feature to be in the interval $[-1, 1]$.

A. Dealing with missing entries

One point worth mentioning is that the features “downlink SINR”, “average CQI” and “average HARQ BLER” can have missing values. These correspond to cases where the measurements were not received due to a problem in the control channel, which usually happens when the UE has a bad channel condition. However, this information is lost after the “mean inputer”, since the missing value will be replaced by the average of all other vales of the particular feature. Losing the information about which entries were originally missing can hurt the classifier performance. We can roughly see the importance of knowing which entries were originally missing by comparing the percentage of entries with missing values separately for each label. For instance, in one simulated scenario the proportion of entries with missing values in the “downlink SINR” feature was 0.83 % for the label “one” (packets arrived within the latency window) and 43.84 % for the label “zero”. This clearly indicates that a missing value in the "downlink SINR" feature is more likely to correspond to label "zero" instead of label "one". Therefore, before the inputer pre-processing is performed we create an extra feature for each of the three features that can have missing entries. This extra feature can assume values of either zero or one, indicating if its matching feature had originally a missing value or not. With this information preserved, it will be taken into account by the ML algorithm in the classification problem.

B. *Classic/baseline prediction tool*

Differently from the ML models, the ARIMA filter models the UE specific packet delays as a time series. Therefore, a separated model is required for each UE, while with the ML algorithms all training data is used to train a single model. For each time gap, varying from 0 to 20, with latency window D equal to 100 ms, each ML model was trained with 75 % of the data, while the remaining data was used for testing the model. On the other hand, the ARIMA model for each UE was fit using all delays for packets of that UE, except the last 20 delays that were reserved for testing.

C. *Classification performance metrics*

In order to compare the different ML algorithms as well as the ARIMA filter, we focus on the accuracy and f1-score metrics. Accuracy is defined as the rate of correct predictions of an algorithm. While it is straightforward, it can be misleading due to the class imbalance of both labels. For instance, consider a given binary classification problem where 96 % of the samples have label 1. If an algorithm always predicts label 1, regardless of the features, it will still achieve an accuracy of 96 %, but it would not be useful in practice. A more suitable metric is the f1-score, which in turn is defined as a function of two metrics: *precision* and *recall*.

Both precision and recall metrics are defined separately for each label. They are defined as

$$\text{Precision} = \frac{\text{True positive}}{\text{True positive} + \text{False Positive}}, \tag{1}$$

$$\text{Recall} = \frac{\text{True positive}}{\text{True positive} + \text{False Negative}}, \tag{2}$$

where the meaning of "true/false positive" can be seen in Table I below.

TABLE I: True/False Positive and Negative.

		Predicted/Classified	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
Actual	**Positive**	False Negative	True Positive

Precision can be seen as the ability of the classifier not to label as positive a sample that is actually negative. Therefore, it is a good metric to compare algorithms when the cost of a "false positive" is high. On the other hand, recall can be seen as the ability of the classifier to find all the positive samples. That is, recall is a good metric when the cost of a "false negative" is high. At last, the f1-score metric seeks a balance between precision and recall, which is given by

$$\text{f1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{3}$$

All three metrics take values in $[0, 1]$, where the higher the value the better the learning model performed for the problem at hand. The next section describes the simulated scenario and the obtained results in terms of the accuracy and f1-score.

TABLE II: Simulation Parameters.

Parameter	Value
Number of sites	7
Number of cells per site	3
bandwidth	5 MHz
Central Frequency	2 GHz
Subcarrier Bandwidth	15 kHz
Number of UEs	400, 500, 600, 700, 800 and 900
UE speed	60 Km/h
Latency Window Size D	100 ms
Packet Size	8000 and 12 000 bits
Scheduler	Proportional Fair
Simulated time	80 s (80000 TTIs)
Discarded TTIs	First 40 000 (40 s)
Number of estimators for RF	100
Number of neurons in MLP hidden layer	50 and 100

V. Simulation Results

In order to investigate the QoS predictability we have performed simulations in a scenario with seven cells, each with three sectors, and wrap around as illustrated in Figure 1. The simulation was performed for UE loads of 400, 500, 600, 700, 800 and 900 UEs in the system, as well as packet sizes of 8000 and 12 000 bits. Several independent simulations were performed, each running for 80 000 transmit time intervals (TTIs), but the first 40 000 TTIs were considered as warm-up period and only data obtained after that was employed in the learning process. The investigated models include LR, RF with 100 estimators, MLP (with one hidden layer with 50 neurons and another model with 100 neurons) and ARIMA. All investigated ML algorithms use the default parameters from the scikit-learn library [6], with the exception of the number of estimators of the RF algorithm and the size of the hidden layer of the MLP algorithm. The simulation parameters are summarized in Table II.

Figure 3 illustrates the network performance – considered as the ratio of packets successfully delivered within the latency window – for the different UE loads and packet sizes of 8000 and 12 000 bits. As the network load increases, either when packet size is increased from 8000 to 12 000 or when the UE load is increased, the resources a spread thinner and latency grows-up and more and more packets are delivered outside the desired latency window. The values in Figure 3 correspond to the proportion of each label in our QoS prediction problem. For a low load, almost all packets arrive at the destination within the desired latency window. Therefore, almost all samples fed to the model have been labeled as one. As the network load increases, the number of packets delivered

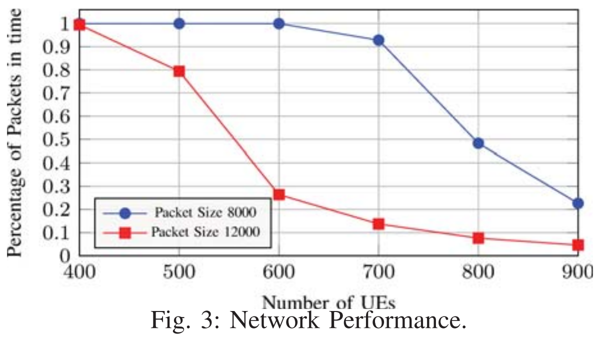

Fig. 3: Network Performance.

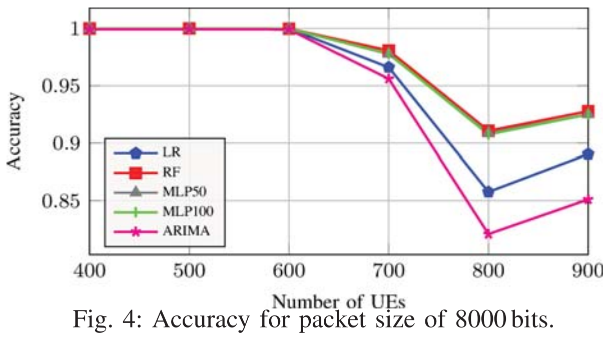

Fig. 4: Accuracy for packet size of 8000 bits.

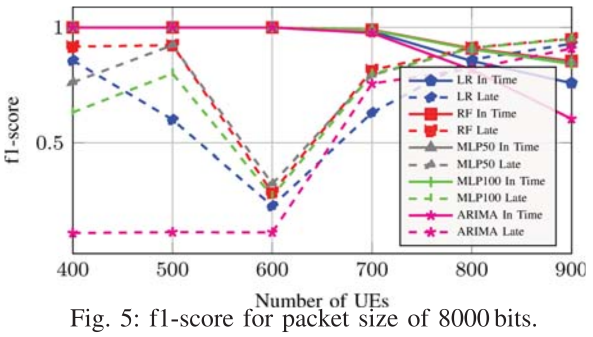

Fig. 5: f1-score for packet size of 8000 bits.

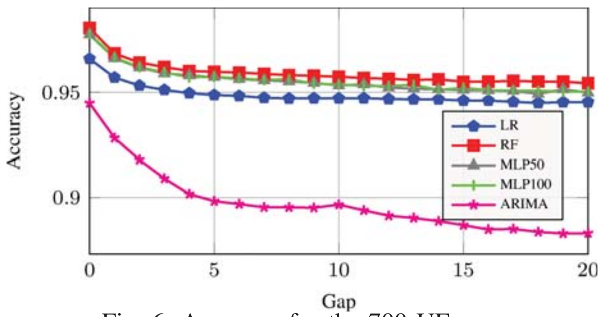

Fig. 6: Accuracy for the 700 UEs case.

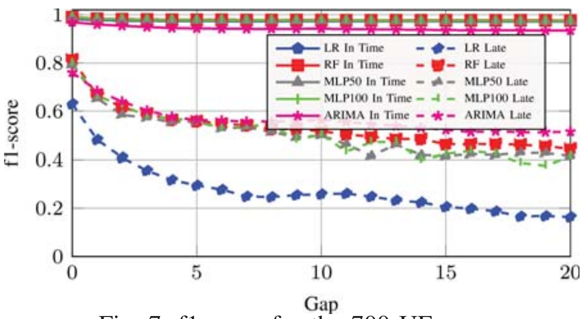

Fig. 7: f1-score for the 700 UEs case.

outside the desired latency window increases and thus we have more samples labeled zero.

Figure 4 illustrates the accuracy for each investigated model for a varying number of UEs with a packet size of 8000 bits. Due to class imbalance, all models have a high accuracy for a low network load, such as when the number of UEs is lower then or equal to 600, even if the model predicts that the packets always arrive in time independently of the feature values. As the network load increases the class imbalance decreased and the accuracy metric becomes more meaningful. As we can see, RF performs better, with MLP being close to it, while the ARIMA filter has the lowest accuracy.

Figure 5 illustrates the f1-score for the same case. The f1-score metric is separated for each class, with the "In Time" class having a better value than the "Late" due to being better represented up to the 700 UEs case. For 800 or more UEs there are more packets that arrive outside the latency window than packets that arrive in time, as can be seen in Figure 4. This inversion makes the f1-score of the "Late" class become higher than the f1-score of the "In Time" class. Nevertheless, the RF algorithm still achieves better performance in terms of the f1-score metric for both classes.

Since we are interested in the predictability of QoS levels in order to assert the safety of employing a vehicular application, a more interesting problem is predicting if a future packet will be delivered within the desired latency window after it has been created. Figures 6 and 7 illustrate the accuracy and the f1-score, respectively, of the different models as the number of gaps varies for the case with 700 UEs. The initial decrease of both accuracy and f1-score is larger for the first five gaps, with a gap of five corresponding to predicting if packets that will be generated 500 ms in the future will be delivered within the latency window of 100 ms. Intuitively, the reason for this initial large drop may be due to the decorrelation of physical layer measurements within the first gaps. Nevertheless, Figures 6 and 7 show that the model is still able to predict the QoS even a few seconds in the future.

One interesting observation in Figure 7 is that the ARIMA filter performs better than the ML algorithms in terms of the "Late" class f1-score and for a number of gaps greater than zero, despite of only using past delays to predict the QoS. The reason for this is that the ML algorithms suffer from the heavy class imbalance. Such class imbalance is also the reason why the "Late" class f1-score in Figure 5 has ups and downs as the number of UEs is increased when it is still below 700 UEs.

To circumvent this issue, data balancing can be used. One manner is to randomly drop samples from the higher represented class such that both classes are more evenly represented. For the 700 UEs case, where the network performance is around 93 %, randomly dropping samples from the "In Time" class still leaves us with around 7600 samples from

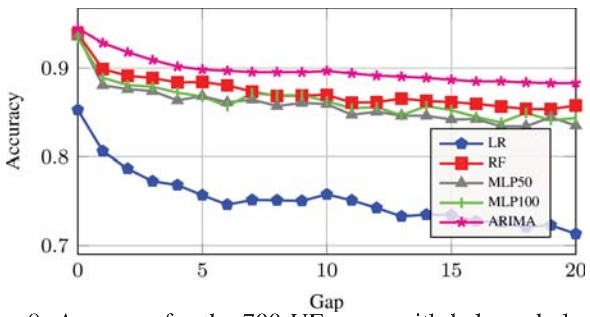

Fig. 8: Accuracy for the 700 UEs case with balanced classes.

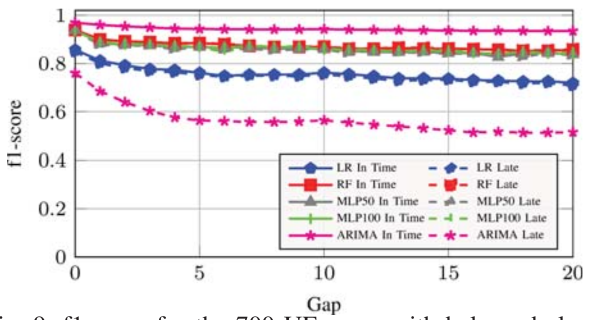

Fig. 9: f1-score for the 700 UEs case with balanced classes.

each class. Figures 8 and 9 illustrate the accuracy and the f1-score, respectively, of the different models for the balanced case as the number of gaps varies. With the classes being balanced, each ML algorithm achieved the same f1-score in both classes, with RF still being the ML algorithm with better performance. Compared with Figure 7, the f1-score of RF jumped from less than 0.5 to more than 0.85 in Figure 9. However, dropping more then 90% of the data in order to balance the classes had an impact on accuracy, as seen in Figure 8. While the network is able to more reliably detect when a packet cannot be delivered, it is also labeling as "Late" more packets that would be successfully delivered. This might be a reasonable compromise, considering safety in vehicular applications. That is, the f1-score of the "Late" class might be the most suitable performance metric from the ones presented in order to compare the ML algorithms in C-V2X applications. However it is also clear that more improvements are possible with more sofisticated approaches to handle the class imbalance.

Another interesting result in Figures 8 and 9 is the performance of the ARIMA filter compared with the ML algorithms. As there is no notion of class imbalance when dealing with time series, the ARIMA filter performs exactly the same as in Figures 6 and 7, since no data is dropped. Thus, it ends up becoming the best one in terms of accuracy and f1-score of the "In Time" class, but at the same time it has a much worse f1-score for the "Late" class than all the ML algorithms. This indicates that the prediction of the ARIMA filter is less useful in C-V2X applications than that of the ML algorithms, even though a separate model is trained for each UE. Nevertheless, the ARIMA prediction could be used as an extra feature when using ML, which might improve the overall performance.

VI. Conclusions

Predicting if a packet can be sent within some latency window is important in V2X scenarios where latency is critical. QoS prediction, herein modeled as a binary classification problem, was assessed by using different supervised learning tools and a time series prediction tool. Reliable predictions using ML can be achieved when the class imbalanced is handled. For example, RF achieved an f1-score around 93 % for gap 0 and around 85 % for gap 20 for both classes. That is, the availability of some vehicular application can be checked some seconds in advance. Besides, ML performed better than the ARIMA filter regarding the prediction of the "Late" class.

As future work, one could investigate if the knowledge of how packet delays evolve in time within future latency windows can improve the feature space of ML models. One possible approach is to add as an extra feature the delays predicted by the ARIMA filter. Furthermore, different strategies for handling class imbalance are also possible, instead of equalizing the class supports by simply discarding the exceeding data from the class with larger support.

References

[1] T. Lohmar, A. Zaidi, H. Olofsson, *et al.*, "Driving transformation in the automotive and road transport ecosystem with 5G," *Ericsson Technology Review*, Sep. 2019.

[2] G. Fodor, H. Do, S. A. Ashraf, *et al.*, "Supporting enhanced vehicle-to-everything services by lte release 15 systems," *IEEE Communications Standards Magazine*, vol. 3, no. 1, pp. 26–33, Mar. 2019.

[3] H. Ye, L. Liang, G. Y. Li, *et al.*, "Machine learning for vehicular networks: Recent advances and application examples," *IEEE Vehicular Technology Magazine*, vol. 13, no. 2, pp. 94–101, Jun. 2018.

[4] G. James, D. Witten, T. Hastie, *et al.*, *An Introduction to Statistical Learning: With Applications in R*, 7th ed. Springer Publishing Company, Incorporated, 2014.

[5] A. Samba, Y. Busnel, A. Blanc, *et al.*, "Instantaneous throughput prediction in cellular networks: Which information is needed?" In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, May 2017, pp. 624–627.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[7] G. E. P. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control*, 5th ed. San Francisco, CA, USA: Holden-Day, Inc., 2015.

[8] 3GPP TR36.885, V14.0.0 (2016-06), "Study on LTE-based V2X Services".

[9] 3GPP TR37.885, V15.1.0 (2018-09), "Study on evaluation methodology of new Vehicle-to-Everything V2X use cases for LTE and NR".

[10] 3GPP TR38.901, V15.0.0 (2018-06), "Study on channel model for frequencies from 0.5 to 100 GHz".