Supplementary Materials

1 Appendix A: Performance comparison with other machine learning algorithms

The benefit of the XGBoost algorithm we adopted in the simulation is that it can be used for modeling non-linear functions. However, other novel machine learning (ML) algorithms are widely applied for regression analysis in many applications. To this end, we compare the performance of XGBoost with other five algorithms, and the specific parameters of these algorithms are summarized in Table 1. We conduct experiments on both urban scenario and highway scenario, and the threshold δ is set to 0.2.

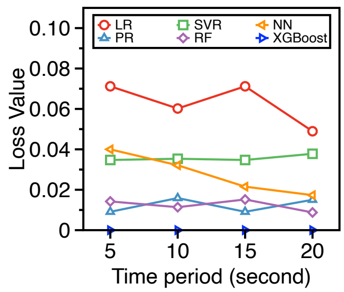
Table 1: Parameter values of algorithms

Algorithm	**Value**
Linear Regression (LR)	-
Polynomial Regression (PR)	Degree: 5
Support Vector Regression (SVR)	Kernel: RBF
Random Forest (RF)	Number of trees: 500
Neural Network (NN)	Number of hidden layer: 2 Total neurons: 600
XGBoost	Number of estimators: 1000 Learning rate: 0.1 Subsample: 0.7

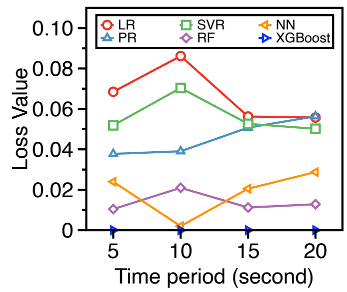
Both the loss values in the urban scenario and the highway scenario are almost below 0.1 shown in Figure 1 and Figure 3. Overall linear regression produces the highest loss values since it is difficult to model non-linear functions. On the other hand, the XGBoost algorithm produces the lowest loss values, thereby having the best performance in data fitting.

In terms of the overheads, results in Figure 2a and Figure 4a show that all these algorithms consume only a few memory storage (less than 4 MB), which is acceptable for the framework deployed in the RSU. Note that the memory costs presented here are the usage of algorithms, rather than the total memory usage of machine learning framework (e.g., PyTorch and TensorFlow). The system administrators could decide which framework is suitable for the RSU operation.

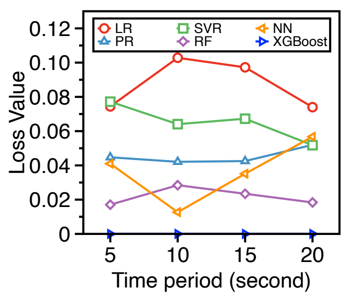
Moreover, as shown in Figure 2b and Figure 4b, the neural network technique spends 2.85 seconds and 4.5 seconds generating the utility function in the two scenarios, while other techniques consume less than 1 second.



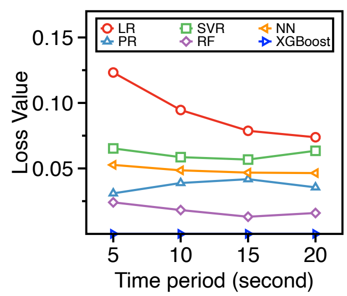
(a) Utility Function of Velocity



(b) Utility Function of Acc

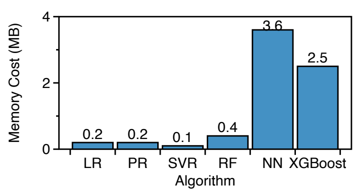


(c) Utility Function of Heading

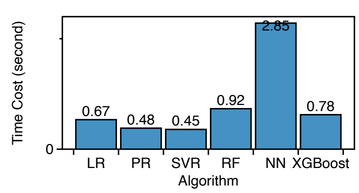


(d) Utility Function of Distance

Figure 1: Urban scenario: loss values of utility functions



(a) Memory Cost

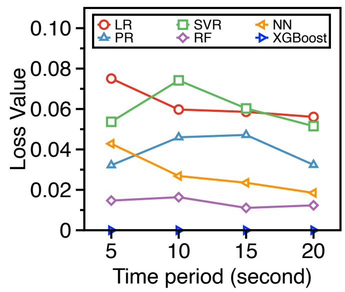


(b) Time Cost

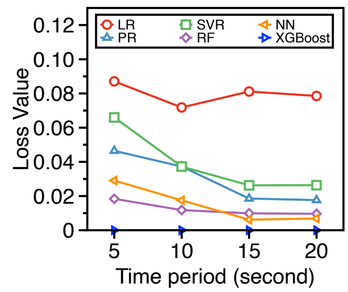
Figure 2: Urban scenario: overheads of different ML algorithms

2 Appendix B: Maps for different scenarios

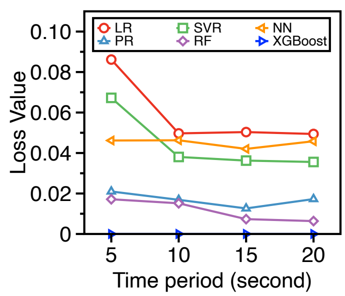
The maps for the urban scenario and highway scenario are illustrated in Figure 5a and Figure 5b, respectively. These maps are produced using the online tool available in `https://www.openstreetmap.org`.



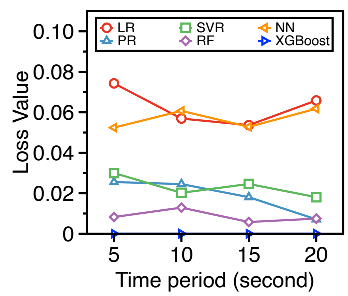
(a) Utility Function of Velocity



(b) Utility Function of Acc

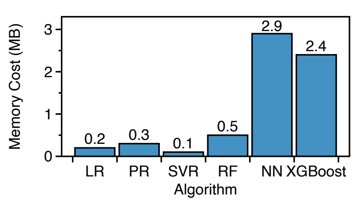


(c) Utility Function of Heading

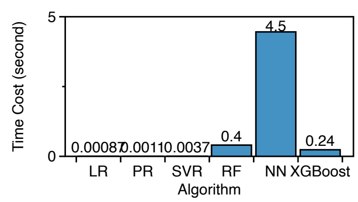


(d) Utility Function of Distance

Figure 3: Highway scenario: loss values of utility functions

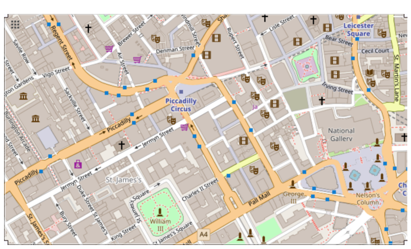


(a) Memory Cost

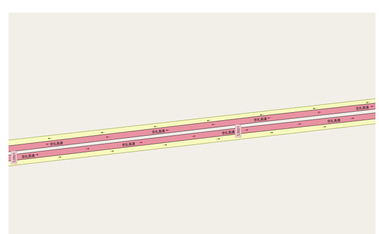


(b) Time Cost

Figure 4: Highway scenario: overheads of different ML algorithms

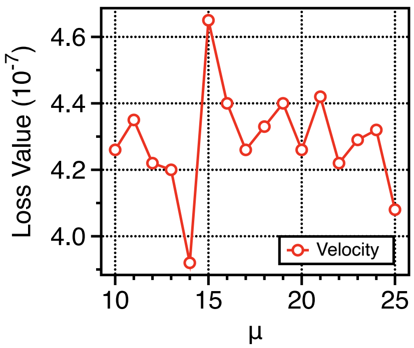


(a) Map of urban area

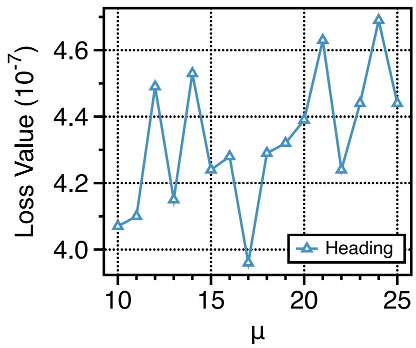


(b) Map of highway area

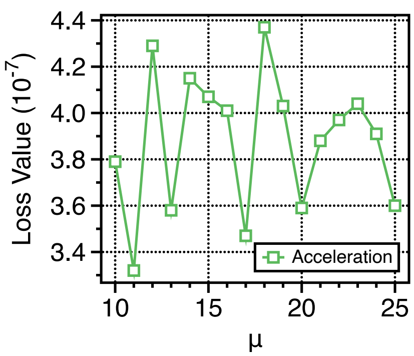
Figure 5: Maps of two scenarios



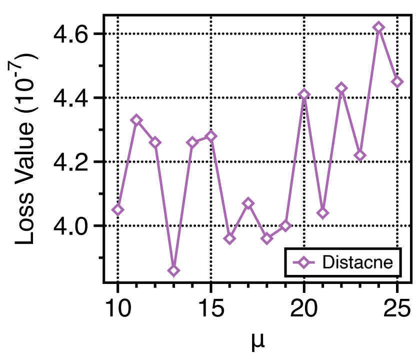
(a) Loss values for attribute of velocity



(b) Loss values for attribute of heading



(c) Loss values for attribute of acceleration



(d) Loss values for attribute of distance

Figure 6: Results of loss values for different attributes

3 Appendix C: Impact of μ for different attributes

Parameters α and β are two predefined parameters to decide the value of μ in the ML training process. Now we show the impact of μ for different decision attributes in Figure 6. It is figured that different values of μ have little impact on the performance of ML training (10^{-7} level).