Alexandria Engineering Journal (2023) 72, 635–647

HOSTED BY

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

ORIGINAL ARTICLE

A comparative study of machine learning-based load balancing in high-speed

Emre Gures [a,*], Ibrahim Yazici [a], Ibraheem Shayea [a,*], Muntasir Sheikh [b], Mustafa Ergen [a], Ayman A. El-Saleh [c]

[a] *Department of Electronics and Communication Engineering, Faculty of Electrical and Electronics Engineering, Istanbul Technical University (ITU), 34467 Istanbul, Turkey*
[b] *Electromagnetics Engineering, Faculty of Engineering, King AbdulAziz University (KAU), Saudi Arabia*
[c] *Department of Electronics and Communication Engineering, College of Engineering, A'Sharqiyah University (ASU), Ibra 400, Oman*

Received 2 November 2022; revised 8 March 2023; accepted 7 April 2023

KEYWORDS

Load balancing;
Machine learning;
High-speed railways;
Regression;
Handover margin;
Time-to-Trigger;
Radio link failure;
Handover ping-pong

Abstract With the rapid developments of fifth generation (5G) mobile communication networks in recent years, different use cases can now significantly benefit from 5G networks. One such example is high-speed trains found in several countries across the world. Due to the dense deployment of 5G millimetre wave (mmWave) base stations (BSs) and the high speed of moving trains, frequent handovers (HOs) occur which adversely affect the Quality-of-Service (QoS) of mobile users. User association for load balancing is also a key issue in 5G networks. Therefore, HO optimisation and resource allocation are major challenges in the mobility management of high-speed train systems. Handover Margin (HOM) and Time-to-Trigger (TTT) parameters are crucial for the HO process since they affect the key performance indicators (KPIs) of high-speed train systems in 5G networks. To manage system performance from the aspect of predictive analytics, we have modelled system performance of mobility management through machine learning (ML). First, the HO management process of a high-speed train scenario is framed as a supervised ML problem. The inputs for the problem are regression task, HOM and TTT and the outputs are key performance indicators (KPIs). Second, data processing is accomplished after generating a simulation dataset. Several methods are employed for the dataset, such as Adaptive Boosting (AdaBoost), Gradient Boosting Regression (GBR), CatBoost Regression (CBR), Support Vector Regression (SVR), Multi-layer Perceptron (MLP), Kernel Ridge Regression (KRR) and K-Nearest Neighbour Regression (KNNR). Tenfold cross validation is then applied for choosing the best hyperparameters. Finally, the deployed methods are compared in terms of the Mean Absolute Error (MAE), Mean Square

* Corresponding authors.
E-mail addresses: gures.emre@gmail.com (E. Gures), shayea@itu.edu.tr (I. Shayea).

☆ This publication/paper has been produced benefiting from the 2232 International Fellowship for Outstanding Researchers Program of TÜBİTAK (Project No: 118C276).
Peer review under responsibility of Faculty of Engineering, Alexandria University.

https://doi.org/10.1016/j.aej.2023.04.013

Error (MSE), Maximum Error (Max E), and R^2 score metrics. From the MAE results, CBR achieves the best outcomes for load level and throughput KPIs with 0.003 and 0.0144, respectively. On the other hand, GBR achieves the best results for call dropping ratio (CDR), radio link failure (RLF) and spectral efficiency KPIs with 0.354, 0.082 and 0.354, respectively. CBR also follows GBR for the three KPIs with 0.356, 0.082 and 0.357, respectively. Only a slight difference in estimations is present. MLP achieves the best results for HO ping-pong (HOPP) and HO probability (HOP) KPIs with 0.0045 and 0.011, respectively. This is followed by GBR and CBR. From the MSE outcomes, CBR and GBR exhibit the best results for load level and throughput KPIs with 2e-5 and 3e-5, respectively. GBR attains the best results for CDR, RLF and spectral efficiency KPIs with 0.25, 0.011 and 0.025, respectively. Accordingly, CBR follows GBR with slightly different errors for the three KPI estimations. MLP achieves the best results for HOPP and HOP KPIs with 5e-5 and 3.6e-5, respectively. Again, this is followed by GBR and CBR for the estimation of these results. This indicates that CBR and GBR can capture relationships between inputs and KPIs for the dataset used in this study, outperforming all other methods generally used for solving this problem.

1. Introduction

High-speed railways are becoming increasingly popular due to their advantages such as lower energy consumption, reduced environmental pollution, greater transport capacity, time efficiency, more comfort, higher safety, etc. [1–3]. Broadband communication for high-speed trains is becoming the main trend due to the significant demand for various mobile wireless services, including online gaming, video calls, etc. [3,4]. The mmWave frequency can be used by 5G and beyond (B5G) cellular networks to potentially accommodate exponentially increasing data rates thanks to the new spectrum bands. However, there are significant obstacles to overcome in mmWave bands, such as high path loss and vulnerability to rapidly changing channel conditions and occlusions [5–11]. The dense deployment of mmWave cells is required to ensure proper coverage and capacity. It is anticipated that the implementation of edge computing will further exasperate issues as well [12]. HOs will frequently occur due to the dense deployment of 5G mmWave BSs as well as the high speed of moving trains. Signalling overhead and latency of service will increase due to more frequent channel estimation and cell association [13,14]. Therefore, HO and resource allocation during user association are crucial issues in mmWave networks.

In recent years, the topic of applying user association to achieve load balancing in mmWave networks has drawn significant attention [15–17]. Load balancing is a promising approach for effectively handling higher data rates, increasing system capacity by controlling cell congestion and managing wireless resource allocation across multiple links [18]. The cell bandwidth is shared by all of its connected users. The cell becomes overloaded when its workload exceeds capacity or approaches near capacity since the number of current users in the cell have reached the threshold limit. The network resources in lightly loaded cells are not fully utilised while users in heavily loaded cells compete for scarce network resources. Therefore, data rates are unfairly distributed across users within the network, and the quality of experience (QoE) becomes inconsistent. In this case, load balancing is initiated, and some users at the boundaries of overlapping or adjacent cells are handed over from overloaded cells to lightly-loaded cells.

HO is the mechanism that switches an active call from one cell to another while the user is moving within the coverage area of a cellular system [19]. HO is a key process that must be properly managed in high-speed rail networks since it poses multiple threats to QoS, such as reductions in the average throughput and service interruptions [20]. Load balancing achieves a fair load distribution between cells by optimising HO control parameters (HCP), such as HOM and TTT.

HOM is one of the main parameters that control the HO process between the serving and target cells. Basically, the cell coverage is expanded or narrowed by including the HOM value to the cell's pilot power value. With this inclusion, the underloaded target cell becomes more attractive in user association than the overloaded serving cell. As a result, user traffic is offloaded from overloaded cells to underloaded cells. In this way, the overall system handles more traffic and users obtain higher throughput. However, since HOM only mimics the received signal quality, this can lead to lower SINR for users that have been handed over, mostly users at the cell edge. Therefore, appropriately setting the HOM value is a crucial issue in any HO process.

Another significant metric used to regulate the HO process is the TTT interval. The TTT interval is a predetermined amount of time that the system will use to repeatedly test the measured received signal strength prior to performing HO. Low TTT time allows users to HO from overloaded cells to underloaded cells in a shorter time, increasing average throughput and reducing the call dropping ratio (CDR). However, this setting can also cause an increase in HO probability (HOP) and HO ping-pong (HOPP) [21]. A high TTT setting may prevent the offloading of users from overloaded cells and may also cause delays in handing over users experiencing poor communication quality.

Various mobility management algorithms are available throughout the literature for high-speed train scenarios. In [22], a distributed architecture with an access point on each wagon was proposed to avoid any temporary interruptions for 5G connectivity, improving the QoE. The proposed distributed architecture is where each train carriage uses one (or

more) Wi-Fi access points for its train-to-ground connection. HO on each high-speed train carriage is modelled as a queuing system with service interruptions triggering load balance between adjacent queues. Each carriage uses the local information of neighbouring carriages and forwards its accumulated packets to neighbours according to the current service speed of neighbouring carriages. In [20], an adaptive switching algorithm based on random suppression was proposed in high-speed rail scenarios. The algorithm establishes an elliptical function relationship between HOM and train speed. HOM can be adjusted according to train speed which reduces the difficulty of forward transition of high-speed trains. However, the proposed vertical HO algorithm fails to fully consider uncertain factors in the network environment. It is easily affected by dynamic network changes which causes further problems, such as HO performance degradation.

Various mobility management algorithms based on ML in B5G networks are also present throughout the literature. In [23], an adaptive optimisation method based on the Q-Learning algorithm was proposed to achieve real-time estimation of HCPs for long term evaluation (LTE) railway (LTE-R) system. A performance situation map of HCPs for different speeds is established to enhance the HO performance. In [24], an intelligent HO scheme based on the Elman network in LTE-R networks was proposed. A neural network model in different regions is trained by the Elman network to obtain the variations of measured HO decision parameters in a future period of time. In [25], the proposed HO decision algorithm based on Bayesian regression assumes that HO conditions are independently distributed in LTE-R networks. The algorithm predicts the cell boundary crossing time. According to the prediction, the algorithm in the serving BS decides when to initiate the HO process, assuming constant delay of HO preparation and HO execution. However, this algorithm assumes that it is too idealistic and does not consider the influence of multiple attributes in the HO results.

Although these algorithms enhance handover performance, they are not robust nor ideal in selecting appropriate HCP values in the 5G system. One of the main reasons is that most of these algorithms were developed for the fourth generation (4G) technology, which has different specifications and requirements as compared to the 5G technology. The existing algorithms developed for previous cellular networks are not efficient for 5G network implementation. Thus, further research with various mobility and deployment scenarios is required for the 5G network. All previously mentioned algorithms were designed to operate on a central optimisation model. This model may lead to increased HO issues for some users [26]. Not all mobile users require the optimisation process at the same time and in the same direction. This issue can become more critical due to the small coverage offered by 5G BSs and the required support for high mobility speeds (400 km/h). Another problem is the type of HCPs considered for optimisation. Several existing algorithms do not optimise all HCPs. For instance, some algorithms only optimise one HCP (HOM or TTT), such as in [20,23,25].

The contributions of this paper offer versatile solutions to the current literature. Firstly, to the best of our knowledge, this study is the first attempt to establish a predictive method for microcells and railway network scenarios using ML methods. For such scenarios, the simulated system is framed as a supervised learning problem, and the estimation of system KPIs are performed by ML methods from the aspect of predictive analytics. Secondly, we use and deploy ML methods to acquire results that forge a direction for future research, further enhancing the reproducibility of our applications in this paper. Finally, this investigative study contributes to the understanding of ML deployment methods and their effectiveness in estimating system performance of high-speed trains in B5G mobile networks. This can greatly assist in determining which learning algorithms should be applied for mobility load balancing in a high-speed railway scenario. The resultant prediction model may also be used for proposing efficient load balancing algorithms that improve system performance by optimising or adapting HCPs.

The organisation of this paper is as follows. Section II introduces the system and simulation models. Section III briefly presents the ML methods used in this paper. Section IV highlights the data collection, data processing and implementation details of the methods used for solving the problem. Sections V displays the outcomes of deploying the ML methods and Section VI presents the concluding remarks.

2. System and simulaton models

This section introduces the network and simulation models. The deployment model and mobility scenario are described, and the simulation settings are summarised. The HO decision process used in load balancing is introduced and the modification made in target cell identification is explained.

2.1. Network and simulation models

The simulation environment has been developed in MATLAB 2020b to simulate the B5G network according to microcells and railway network scenarios. The network layout consists of thirty-nine evolved NodeB (eNB) with three sectors for each cell, and the distance between the two BSs is 400 m (each eNB covers 200 m). The network is modelled according to the specifications of LTE-Advanced Pro 3GPP Rel. 16. Table 1 displays the network parameters. The number of hexagonal cells can automatically increase depending on the simulation time. Fig. 1 presents an example of the B5G network deployment scenario where the pink-coloured points and hexagonal shapes represent the user equipment (UE) and the eNB, respectively. The frequency reuse factor is assumed to be one.

A collection of mobile users is initially constructed with random coordinates inside the hexagonal boundaries of each cell to simulate the environment of the actual B5G network. The number of users randomly and periodically fluctuates over the simulation cycles in each cell. This means that each eNB's traffic load is periodically and automatically modified to reflect the current state of the network. The simulation model is created to mimic a random generation of traffic load throughout the simulation, taking into account the complete enablement of admission control functionality in the target cell during user mobility. Each user is equipped with an omnidirectional antenna to communicate with the serving cell.

In this study, the Directional Mobility Model (DMM) is proposed for all measured mobile users to simulate the movement of high-speed trains. Mobile users are only allowed to move in one direction, as shown in Fig. 1. The measured users are initially generated at random coordinates in cell number

Table 1 Simulation settings [16,26,31–35].

Parameter	Assumption
Environment	5G Rel. 16 System, micro cells and urban areas
Cellular layout	Hexagonal grid
No. of sectors for each cell	3 Sectors
Path loss model	$L = 58.8 + 37.6x\log_{10}(d) + 21\log_{10}(f_c)$
eNB's antenna height	15 m
Cell radius (d)	200 m
Carrier frequency (f_c)	28 GHz
Total power TX eNB	46 dBm
System bandwidth	500 MHz
Shadowing	8 dB
Tested UE number	15 UEs randomly distributed
eNB noise figure	5 dB
White noise power density N_t	−174 dBm/Hz
Thermal noise power (N_p)	$N_p = N_t + 10\log_{10}(BWx10^6)dB$
No. of tested UEs	15 UEs randomly distributed
UE noise figure	9 dB
UE height	1.5 m
UE antenna gain	0 dB
UE's antenna	1, omni-directional
Mobility model	DMM
No. of PRBs	2500 PRBs
Resource distribution	Equally across all active UEs
UE's speed	400 km/h
Simulation cycle	40 ms
Min. desired level of RX in the cell	−101.5 dBm
TTT intervals	Changes from (0 to 5120) ms
HOM values	Changes from (0 to 16) dB
Modulation scheme	Adaptive Modulation Coding scheme

one, as shown in Fig. 1. Afterwards, users move between solid blue lines in only one direction and on different parallel paths to each other. The simulation periodically updates the movements of the UEs. The periodic interval is specified as 40 ms. To accurately depict the characteristics of a high-speed train, the user speed for the simulation study was set to 400 km/h.

In this research, 15 users were selected to investigate the performance of the proposed algorithms. The average values from 15 users represent all results considered in the measurements made throughout this study. 15 users are randomly generated in cell #1 in the first simulation cycle. Since performance is independently measured for each user during each simulation cycle, the accuracy of results is improved. The average value was taken for all users measured in each simulation cycle. Therefore, the average values of the cells' load level, throughput, CDR, spectral efficiency [27], HOP, HOPP and radio link failure (RLF) are calculated in each simulation cycle. The simulation began according to the parameter settings depicted in Table 1.

2.2. Handover decision

The main objective of load balancing is to ensure that the total network traffic load is equally shared among cells by offloading the user traffic from overloaded cells to underloaded cells. For this to be accomplished, a bias value is added to the target cell's pilot power value. The load balancing function increases the HOM to initiate an early HO to another cell when the serving cell is overloaded. Underloaded target cells with biased power during the HO decision stage become more attractive than overloaded cells. Therefore, the total system serves more traffic and users obtain higher throughput [18]. However, this unanticipated increase may lead to RLF and HOPP, causing poor QoS for users [28]. An automated model is therefore needed to separately predict HOM settings based on the cell load for each user. The conventional A3 HO event depending on a power-based margin is initiated when the neighbouring cell's power is greater than that of the serving cell for one TTT period. It is mathematically expressed in Equation (1), as follows:

$$RSRP_t + HOM > RSRP_s \tag{1}$$

where $RSRP_s$ and $RSRP_t$ represent the RSRP of the serving cell and the target cell, respectively. If the load balancing function is inactive (the serving cell is not overloaded, that is, the load level is less than 65 %), the HOM value is set to zero [29,30].

In this study, an adjustment was made in the target cell selection to provide load balancing and reduce HOP and HOPP. The target cell is determined from a restricted list of neighbouring cells defined by the load level criterion. In this list, the cell with the highest RSRP is designated as the target cell. This adjustment ensures the handover of users to underloaded target cells if adequate conditions are met. In the scenario where the serving cell is overloaded, users are handed over to underloaded target cells with appropriate HCP settings. In this scenario, no matter how high the RSRP value of the serving cell is, it cannot be assigned as the target cell since it cannot meet the load level requirement. Therefore, the probability of HOPP and HOP is reduced.

3. Machine learning-based system performance prediction

ML methods are applied across various fields, producing promising results. We chose ML methods for estimating system performance. Our goal is to deploy machine learning methods to estimate performance of a mobility system. To achieve our goal, the ML methods predict outputs of a high-speed train system based on HOM and TTT input values. The results of this performance estimation approach can also be used in load balancing and HCP optimisation. ML consists of three types of learning: supervised learning, unsupervised learning and reinforcement learning. Our task falls under the supervised learning category which the applied ML methods are trained by labelled data (output) to perform either a regression task or a classification task. In this paper, we applied regression-type supervised methods since our outputs are all continuous. The high-speed train simulation system is framed as a supervised learning problem, and HOM and TTT are used to predict the outputs of high-speed train scenarios in the B5G network. Ultimately, the study conducts predictive analytics for HO management in high-speed train scenarios. Powerful ML methods are utilised in the task of predicting outputs. AdaBoost Regression (ABR), CatBoost Regression (CBR), Gradient Boosting Regression (GBR), K-Nearest Neighbour

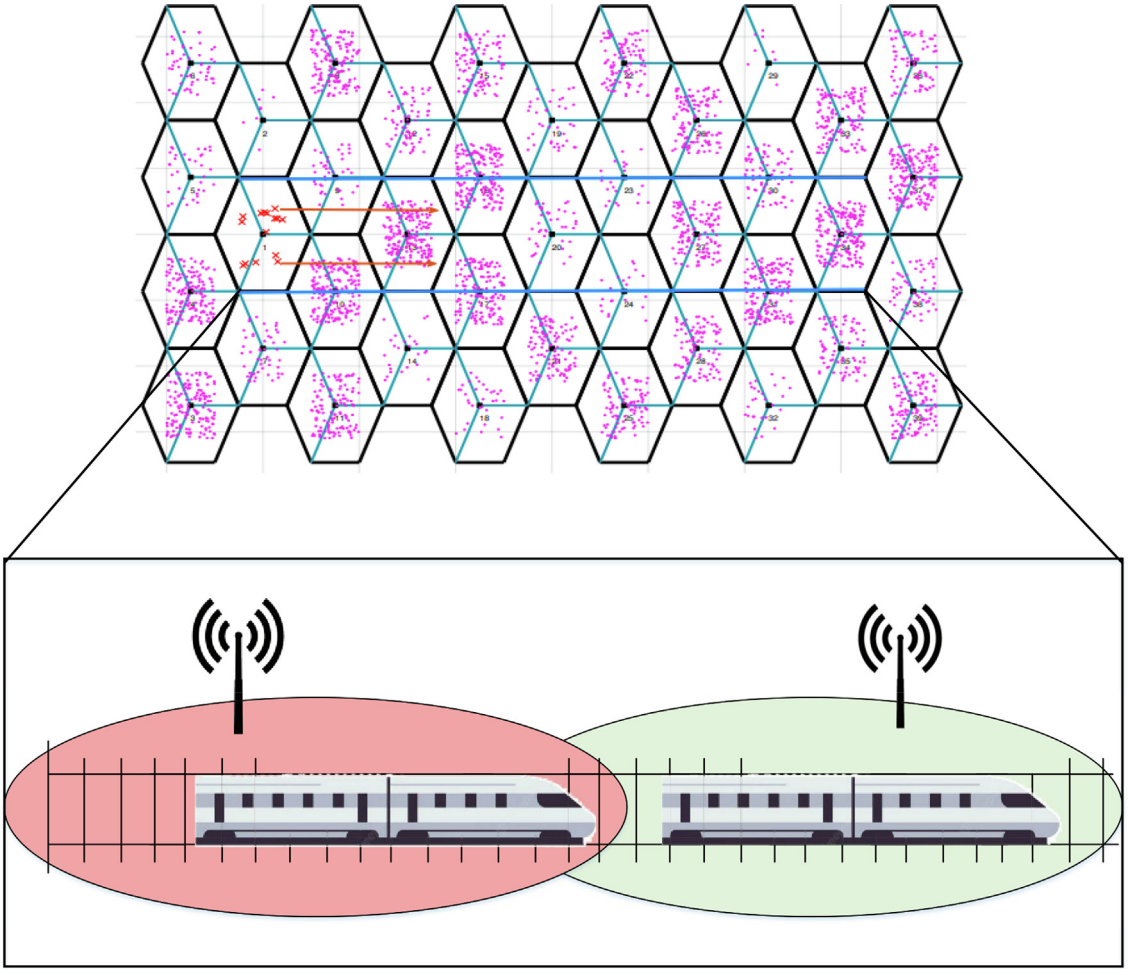

Fig. 1 B5G network deployment scenario.

Regression (KNNR), Kernel Ridge Regression (KRR), Multilayer Perceptron Regression (MLPR) and Support Vector Regression (SVR) are the supervised ML methods used in this paper. A brief overview of these methods is provided in the following subsections.

3.1. Adaboost regression

Ensemble learning in ML methods forms strong estimators (learners) rather than a single base estimator (weak learner) to achieve better performance at the end of a learning process. It consists of two types: bagging and boosting. Boosting constructs its model through sequential learning with each generated model trying to mitigate the weakness of its predecessor model. Bagging trains its model in parallel, learning to construct its model in this manner.

Adaptive Boosting Regression is a type of boosting algorithms frequently used for both classification and regression tasks [36]. The easy-to-implement and bias reducing features of this algorithm support its utilisation in different classification and regression tasks. However, the hyper-parameter selection (which depends on the problem handled), the intense computations for high amounts of data and the avoidance of overfitting issues are all significant issues in the deployment stage.

3.2. Gradient boosting regression

Gradient Boosting is another type of boosting algorithm that can perform both classification and regression tasks. As previously mentioned, the gradient boosting model is constructed by forming strong learners from weak learners, compensating the weakness of the weak learners. A set of weak learners (regression trees) evolves in a manner where a set of strong learners is obtained during this evolution process by learning from the errors of previous learners [37,38]. This method is an extended version of adaptive boosting. Boosting is framed as an optimisation problem that minimises the loss function using a gradient descent type optimisation procedure. Hyperparameter selection and overfitting issues must be carefully managed in the deployment stage.

3.3. Cat boosting regression

Cat Boosting, proposed by Yandex research, is a novel boosting algorithm that conducts both classification and regression tasks [39]. It considers categorical features along with continuous-valued features in a problem, providing scalable and fast GPU implementations for large datasets. It falls under

the category of boosting algorithms similar to AdaBoost and Gradient Boosting.

3.4. K-Nearest neighbour regression

The KNN algorithm is a type of non-parametric supervised ML algorithm that can be used for both classification and regression tasks [40]. This algorithm makes use of the neighbourhood of pre-determined K neighbours in model fitting. In regression, the output is computed by averaging the values of these K neighbours. In model fitting, the model is constructed by using a step function that leads to smoother fits [41]. This algorithm is a promising solution for different use-cases, however, bias-variance trade-off is a significant issue when determining the number of K in the deployment stage.

3.5. Kernel ridge regression

Kernel ridge regression is a combined algorithm that uses classification and ridge regression with the kernel trick in a hybrid form. Kernel ridge regression transforms the space of features into a higher dimensional space to informatively represent the dataset. This algorithm is formed by applying the kernel trick to a parametric model, the ridge regression [42]. The solution vector depends on all training inputs, hence, the model learned by kernel ridge regression is non-sparse [43]. Kernel ridge regression is similar to support vector machines in using kernel tricks, however, they vary in their use of different loss functions. Support vector machine employs epsilon-insensitive loss, while the kernel ridge regression utilises squared error loss.

3.6. Support vector regression

Support vector regression is a regression form of the support vector machine used for classification tasks [44]. Support vector regression also applies kernel trick and epsilon-insensitive loss function in its algorithm. This loss function only penalises the points which are lying outside the epsilon-tube [43]. The SVR problem is first framed into a constrained optimisation problem and then solved by the quadratic function optimisation. SVR is good at avoiding the overfitting issue, however, it is computationally burdensome when the dataset size is large. This becomes a problem in the deployment stage [45].

3.7. Artificial neural network

Artificial Neural Network is a type of stacked logistic regression method where the last layer changes according to task type, either regression or classification. In regression tasks, the last layer is a linear layer while it is sigmoid, softmax or another classification activation function. ANN consists of one input layer, one hidden layer and one output layer [43,46]. It mimics the function of neurons in the brain during its working principles. In ANNs, the given information is processed throughout hidden layer(s) by means of several activations until finally, an output is produced. A shallow neural network consist of one input layer, one hidden layer and one final output layer. The size of the hidden layer in a shallow neural network may significantly affect the performance of the algorithm by causing overfitting or underfitting, therefore, this parameter must be considered in model construction and deployment stages [45].

4. Implementation

This section introduces the stages of application. The first stage consists of data collection and data processing tasks, followed by the inputs, outputs and prediction scheme. The second stage includes the training procedures, model selection and prediction and performance metrics. Fig. 2 presents the overall implementation steps.

4.1. Data collection

The dataset was created using the MATLAB 2020B software in the B5G network and was designed to consider the microcells and railway network scenarios according to the simulation settings summarised in Table 1. The inputs of the system consist of fixed HOM and TTT values. A total of 528 samples were obtained by simulating the combinations of fixed HOM and TTT values. Input parameters (HOM and TTT) take the values defined by the 3rd generation partnership project (3GPP) [34]. Fixed HOM values range from 0 to 16 dB in 0.5 dB increments. Fixed TTT intervals consist of the following values defined by 3GPP: 0, 40, 64, 80, 100, 128, 160, 256, 320, 480, 512, 640, 1024, 1280, 2560, and 5120 ms. We offer

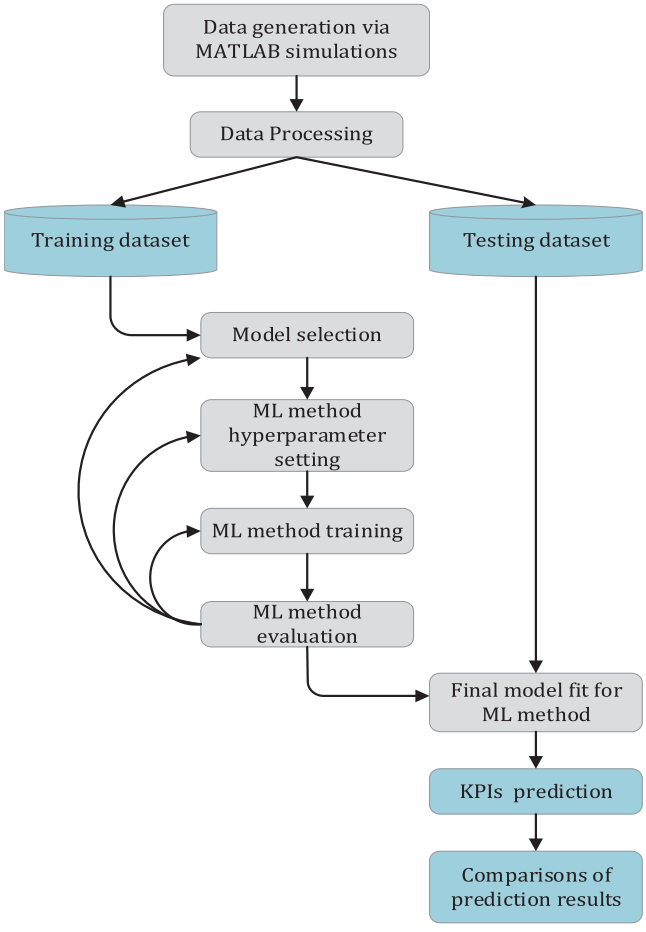

Fig. 2 Procedure for ML-based system performance prediction.

a large set of measurements to provide a more comprehensive research that considers additional KPIs, thereby determining the performance of various networks from different perspectives. Other than the KPIs related to load balancing (load level, throughput, CDR and spectral efficiency), the performance metrics associated with mobility robustness optimisation (RLF, HOPP and HOP) are also included in the analysis due to the high speed scenario of trains. The dataset consists of seven outputs, as explained in Table 2. The outputs are the average measurement values achieved by 15 users in each simulation cycle.

4.2. Data processing

All outputs are real-valued outputs, therefore, regression was applied in the supervised ML method. Since the units of each output and input features are different, they must be normalised to acquire stable results with the deployed methods. This study applies the zero-mean normalisation for each input and output features according to Equation (2):

$$x_i^{'} = \frac{x_i - \mu}{\sigma} \tag{2}$$

where x_i and $x_i^{'}x_i^{'}$ denote the value of i^{th} example and the normalised value of this i^{th} example, respectively. On the other hand, μ and σ correspond to the values of mean and standard deviation of the relevant feature or output, respectively. Normalisations for the testing and training sets are performed with different mean and standard deviation values. This is a significant issue in ML deployment methods. In ML, the testing dataset distribution is unknown by the fitted model that is only trained on a training dataset with possibly different distributions than the testing dataset. Hence, normalising the testing and training data with the same parameters will result in wrong predictions. In this paper, different parameters (mean and standard deviation values) are used for the normalisation of the training and testing datasets.

Table 2 Definiton and purpose of inputs and outputs.

Type	Parameter	Definition
Inputs	HOM	An offset that artificially modifies the HO decision.
	TTT	Predetermined amount of time that the system will use to repeatedly test the measured received signal strength prior to performing HO.
Outputs	Load Level	The ratio of the current number of users in a cell to the maximum number of users that the cell can serve.
	Throughput	The volume of data accurately transferred from one site to another within a predetermined period of time.
	Call dropping ratio	The ratio of the number of calls to the total number of calls due to poor connection quality and insufficient resources in the cell
	Spectral efficiency	Data rate normalized with the cell bandwidth.
	Radio link failure	Drop rate of communication during user mobility due to a deterioration in the RSRP level.
	HO ping-pong	HO of a user back to the serving cell in a time below the critical time after being handed over to the target cell.
	HO probability	Probability of connections exchanged between the source and destination BSs during user mobility.

4.3. Hyperparameter setting and model training

This section presents the stages of training, validation and model selection. In our experiments, we used Python 3.9 and sklearn 1.1.1 library for the deployment method. An ultrabook laptop that has Intel Core i5 CPU with 1.8 GHz was used for conducting the experiments.

ML methods have different hyperparameters that affect their performance, such as the learning rate, hidden layer units in neural networks, C parameter for SVR, number of trees in decision tree method, etc. Choosing the model with the best hyperparameter settings is crucial to ensure that the ML algorithm can achieve promising results.

The use of cross-validation for model selection is therefore essential to learn the hyperparameters of the ML method, making it a viable option. The cross-validation method resamples a dataset used to assess the performance of the ML method on a limited dataset size. Cross-validation, generally dubbed as k-fold cross-validation, partitions a training dataset into k-folds. The ML method is then iteratively trained on each k-fold as a validation set, and the rest are used for the training set. Each fold of the dataset is used to train the ML method. The method learns different input sets in this manner. It then learns to estimate the skill of the ML method to select the best one among different models as a result of the cross-validation process. Therefore, the hyperparameters of an ML method are the parameters learnt during the training stage with the training dataset. Different ML methods have different hyperparameters. We employed the k-fold cross-validation to determine the best hyperparameters for different ML methods in our implementations. Cross-validation further contributes to the generalisation capability of ML methods, allowing them to introduce the k-unseen dataset during the training stage. Hence, the model produces more robust results than one that uses a simple train-test split method. Fig. 3 presents an example of cross-validation in the model selection process. 85 % of the dataset was used for training, and the remainder was used for the testing set. 10 % of the training dataset was used for the validation set by applying 10-fold cross-validation.

SVR is another ML method used in this study. Fig. 4 presents an exemplary SVR method. In Fig. 4, a regression line is fitted for the inputs, and a boundary that encompasses the maximum inputs (points) in the epsilon-margin error is constructed for SVR modelling.

Fig. 5 displays the architecture of a shallow artificial neural network. The input layers receive the inputs of a neural network model, the hidden layers process the inputs by nonlinear transformations and the output layer then produces the results of the model.

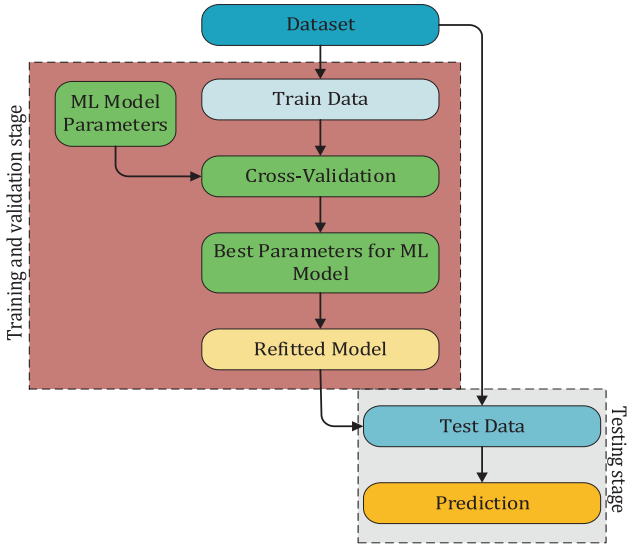

Fig. 3 Cross-validation process.

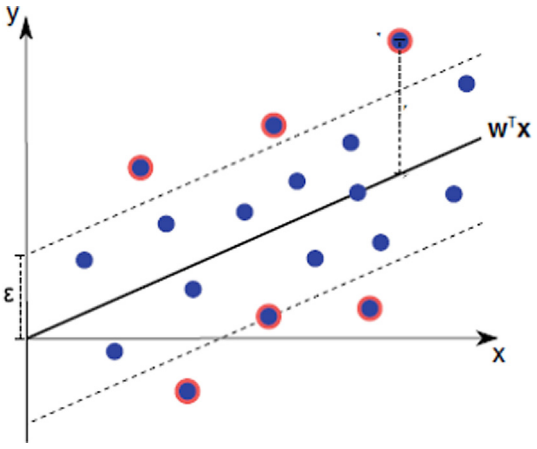

Fig. 4 Example of SVR.

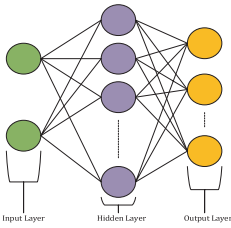

Fig. 5 Example of shallow neural network.

4.4. Prediction

After determining the best hyperparameters in our experiments, they were then set for each ML method. The Multi Input Multi Output (MIMO) prediction strategy was used for all deployed ML methods, as shown in Fig. 6.

To evaluate and confirm the performance of the deployed ML methods, it is necessary to measure the statistical errors between the ground truth and predicted values. Two common

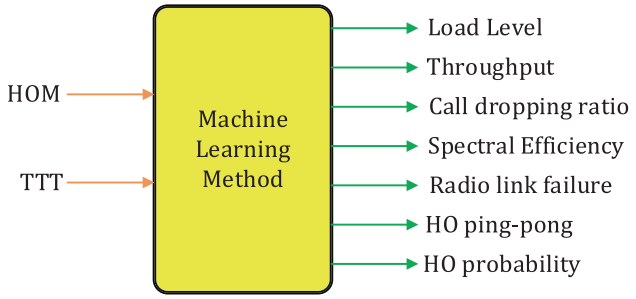

Fig. 6 MIMO prediction strategy.

error metrics were employed for the ML methods: the Mean Absolute Error (MAE) and the Mean Square Error (MSE). The Maximum Error (MaxE) metric was also used to evaluate the error of worst performing method(s). Computations for MAE, MSE and MaxE are provided in Equations 3–5, respectively.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{3}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4}$$

$$MaxE = \max_i(|y_i - \hat{y}_i|) \tag{5}$$

In Equations 2–4, N corresponds to the size of the relevant dataset. y_i and $\hat{y}_i$ denote the ground truth value of element i of an output and the predicted value of element i of the output, respectively. The R^2 score, a coefficient of determination, was also used. This score shows the proportion of total variance in the ML method that is explained by the independent variable(s). Computation for R^2 is provided in Equation (6).

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2}{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}, \bar{y} = \sum_{i=1}^{N} y_i \tag{6}$$

5. Results and discussions

This section provides the results of the deployed ML methods with the best hyperparameters determined through grid search and cross-validation methods. The best hyperparameters are obtained by performing 10-fold cross validation, as explained in the implementation section. The employed methods are compared in terms of MAE, MSE, Max E and R^2 score values. The results for each output are displayed in Tables 3-6, respectively.

The best score for each comparison is highlighted in bold throughout Tables 3-6. In addition, the last row of Tables 2 and 3 provide the error performance difference between the best performing method and the second best as a percentage performance difference for each output. According to the MAE results presented in Table 3, CatBoost Regression is best at predicting load level and throughput. On the other hand, Gradient Boosting Regression is best at predicting CDR, RLF and spectral efficiency. However, in these first five outputs, the performance differences between Gradient Boosting and CatBoost Regression are insignificant, as shown in the last row of Table 3. Lastly, the Multi-layer Perceptron outperforms

Table 3 MAE Results.

	MAE						
Method	**LL**	**T**	**CDR**	**RLF**	**SE**	**HO PP**	**HOP**
ABR	0.00483	0.01565	0.36917	0.0887	0.36726	0.00476	0.01533
GBR	0.00303	0.01447	**0.3540**	**0.08214**	**0.3540**	0.00478	0.01476
CBR	**0.00300**	**0.01442**	0.35678	0.08237	0.35678	0.00478	0.01475
SVR	0.01785	0.05917	2.07203	0.33304	2.07203	0.00848	0.03784
MLP	0.01125	0.01653	1.5876	0.09675	1.5789	**0.00446**	**0.01083**
KNNR	0.01085	0.022	1.13137	0.12361	1.13137	0.00495	0.01506
KRR	0.00906	0.01631	1.13999	0.09224	1.13999	0.01080	0.02622
Diff. (%)	1	0.4	0.8	0.3	0.8	6.3	26.6

Table 4 MSE Results.

	MSE						
Method	**LL**	**T**	**CDR**	**RLF**	**SE**	**HO PP**	**HOP**
ABR	0.00003	0.00036	0.26528	0.01151	0.26405	0.00006	0.00051
GBR	**0.00002**	**0.00033**	**0.25254**	**0.01069**	**0.25254**	0.00006	0.00049
CBR	**0.00002**	**0.00033**	0.25322	0.01079	0.25322	0.00006	0.00049
SVR	0.00089	0.00492	12.96513	0.15663	12.96513	0.00050	0.00448
MLP	0.00019	0.00053	3.80253	0.01854	3.77245	**0.00005**	**0.00036**
KNNR	0.00039	0.00079	4.09487	0.02495	4.09487	0.00006	0.00055
KRR	0.00013	0.00051	2.35721	0.01666	2.35721	0.00029	0.00173
Diff. (%)	33.3	8.3	0.3	1	0.3	16.7	26.5

Table 5 Max E. Results.

	Max Error						
Method	**LL**	**T**	**CDR**	**RLF**	**SE**	**HO PP**	**HOP**
ABR	0.01005	0.03586	0.99988	0.20494	0.99988	0.02656	0.06721
GBR	**0.00794**	**0.03532**	**0.99988**	**0.20193**	**0.99988**	**0.02657**	**0.06709**
CBR	0.00816	0.03612	1.05268	0.20568	1.05268	0.02724	0.06916
SVR	0.08525	0.14891	10.7277	0.85161	10.7277	0.08640	0.23301
MLP	0.02911	0.06205	4.37948	0.37847	4.28590	0.02868	0.07027
KNNR	0.06225	0.07431	6.27021	0.41876	6.27021	0.02675	0.06742
KRR	0.02524	0.06703	3.79483	0.38762	3.79483	0.06049	0.15171

all other methods in the prediction of HOPP and HOP outputs. According to Table 3, the performance differences of this method compared to the second-best method are extremely high. The differences are 6.3 % and 26.6 % for HOPP and HOP, respectively.

The results of the best performing methods in terms of MSE performance (displayed in Table 4) are nearly the same with Table 3. For predicting the load level and throughput, Gradient Boosting and CatBoost outperform all other methods with considerable differences (33 % and 8.3 %). Gradient Boosting outperforms all other methods in predicting CDR, RLF and spectral efficiency. This is followed by the CatBoost method with only a slight performance difference. Lastly, the Multi-layer Perceptron method outperforms all other methods in predicting HOPP and HOP with considerable difference, as seen in the last row of Table 4 (16.7 % and 26.5 %). According to Tables 3 and 4, CatBoost and Gradient Boosting Regression achieve the best results. They outperform all other methods for five outputs out of seven. This outcome is also seen in Figs. 7 and 8 regarding the average MAE and MSE achieved by these methods.

Overall, all methods produced satisfactory results for load level, throughput, RLF, HOPP and HOP according to MAE and MSE metrics. However, SVR, KNNR, KRR and MLP remained behind GBR, CBR and ABR in predicting CDR and RLF according to both metrics. It should be noted that

Table 6 R^2 Scores.

	R^2 *Score*						
Method	**LL**	**T**	**CDR**	**RLF**	**SE**	**HO PP**	**HOP**
ABR	0.9945	0.9796	0.9961	0.9791	0.9961	0.8785	0.9091
GBR	**0.997**	0.981	**0.9963**	**0.9805**	**0.9963**	0.8790	0.9119
CBR	**0.997**	**0.9811**	**0.9963**	0.9804	**0.9963**	0.8787	0.9114
SVR	0.8344	0.7181	0.8095	0.7149	0.8095	−0.0630	0.1988
MLP	0.965	0.9697	0.9441	0.9663	0.9446	**0.8840**	**0.9355**
KNNR	0.9281	0.9548	0.9398	0.9546	0.9398	0.8734	0.9013
KRR	0.9754	0.9705	0.9654	0.9697	0.9654	0.3866	0.69

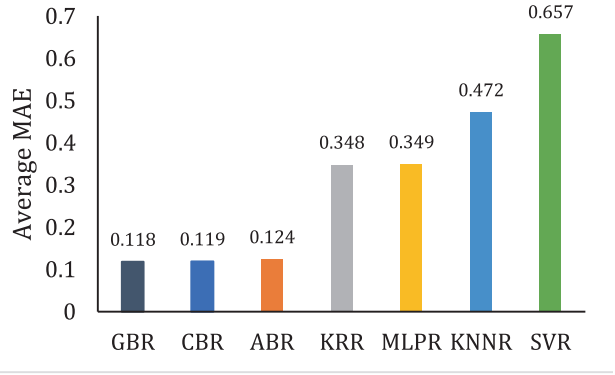

Fig. 7 Average MAE comparisons for each method.

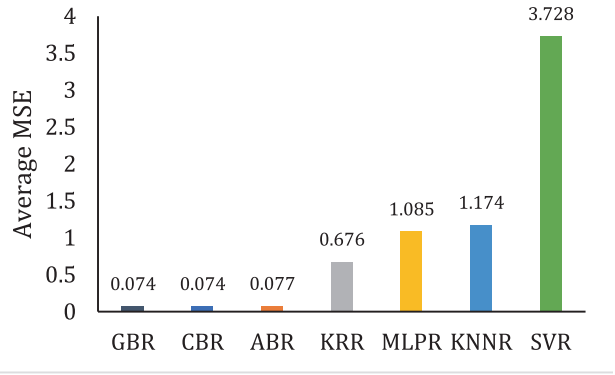

Fig. 8 Average MSE comparisons for each method.

SVR is the worst performing method among all others with high error rates. This may be due to its low generalisation ability for the test set in this study. In Table 4, the SVR errors for CDR and spectral efficiency are very high because MSE quadratically penalises errors. Since SVR diverges from the other methods by producing the worst results for these outputs, this divergence is quadratically penalised by the MSE metric. In Table 5, outperformance of the applied boosting methods is apparent since the maximum error in each output is less compared to the other methods used. This demonstrates the stability of the boosting methods when it comes to predictions. The other methods produce high errors, especially in the prediction of CDR and SE outputs. SVR is the worst performing method among all others. Table 6 demonstrates how much the total variance of the dependent variable is explained by the independent variable(s). The low performance of SVR is also apparent from this table. Even KRR, which is a modified version of SVR, achieved better results since it uses squared loss, which contrasts from the epsilon-insensitive loss applied by SVR. This may be due to KRR's learning ability of the non-sparse model which differs from SVR. The R^2 score of SVR is below 0, indicating that the method is not good at explaining the relationship between input(s) and output(s). A noteworthy point is that KRR's R^2 score is also low for HOPP and HOP as compared to other methods, excluding SVR. For these outputs, KRR performs in parallel with SVR with low R^2 scores, significantly diverging from the other methods.

CatBoost Regression and Gradient Boosting Regression further exhibit superior results in most output predictions, as shown in Table 6. These two boosting methods are directly followed by another boosting method: AdaBoost Regression. It should be noted that MLPR sometimes produces the best prediction outcomes by surpassing the boosting methods. Figs. 7 and 8 display the average results for each method used (including their overall performance) according to MAE and MSE outcomes, respectively. The MAE and MSE errors for the GBR method are 0.118 and 0.074, respectively. Likewise, the same MAE and MSE error values can be seen for CBR (0.0119 and 0.074, respectively). According to the results in the figures, the superiority of CBR and GBR is more apparent. ABR is the boosting method that closely follows the two best performing methods. In contrast, the worst performing methods are MLPR and KNNR. They are worse than KRR yet better than SVR. This indicates that MLPR cannot generalise the dataset well due to its small size. Although KRR and SVR employ nearly the same learning procedure for modelling data in the algorithmic perspective, KRR distinctively outperforms the SVR method in terms of average MAE and MSE results. This implies that KRR can capture patterns in the dataset, hence achieving better predictions.

The overall results reveal the superiority of boosting methods in the regression task for the data applied in this study. Weak learners form strong learners in the training stage to achieve superior results. Boosting methods are generally good at reducing bias in datasets, hence their superiority may stem from bias reduction in the dataset used in this study. The results also provide insight regarding ML modelling relations between HOM, TTT and KPIs (outputs) from the aspect of

predictive analytics in high-speed train scenarios using 5G network systems with different ML algorithms. The limitations of this study include dataset size and the number of features. If the dataset size is larger with a greater number of features, deep learning methods would be applied rather than conventional machine learning methods to obtain better results. Deep learning methods excel at capturing relationships between inputs and outputs in datasets. Further research can address these limitations to acquire improved results.

6. Conclusions

The aim of this paper is to model mobility management of high-speed train system behaviour using ML methods for predictive analysis. ML system modelling enables the prediction of outputs for different HOM and TTT values in high-speed train scenarios in the 5G mobile network. It can also be applied to model different use cases of mobility management scenarios using predictive analytics. Accordingly, data was first collected from a simulation then processed at the start of the application stage. Normalisation was performed for the training and testing datasets using different parameters to provide robust predictions since the testing dataset was unknown by the fitted model. Next, GBR, CBR, ABR, SVR, KNNR, KRR and MLPR methods were selected and applied for the dataset. 10-fold cross validation was conducted for each method, and the methods with the best hyperparameters were chosen according to the cross-validation results. The methods with the best hyperparameters were then applied for the dataset. The performances of the chosen methods were compared in terms of MAE, MSE and Max Error metrics. The R^2 scores were also presented. According to the results, CBR and GBR achieved better MAE and MSE results compared to all other methods for five outputs: LL, T, CDR, RLF and SE. AdaBoost directly follows CBR and GBR in terms of performance. The MLPR method achieved the best results for two outputs: HOPP and HOP.

This study provides insight regarding the effectiveness of deploying ML methods in estimating system performance of high-speed trains in B5G mobile networks. It can greatly assist in choosing which learning algorithms should be used for mobility load balancing in a high-speed railway scenario. Predictive load balancing algorithms that are chosen according to examined indices can lead to enhanced results. The ML modelling of a mobility system's performance produces promising results for estimating load balancing KPIs. This offers great potential for future research and applications. Using the outcomes of ML modelling in future studies will create efficient load balancing algorithms that improve system performance by optimising or adapting HCPs in different mobile network scenarios.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

(Emre Gures and Ibrahim Yazici equally contributed to this work.) This research work was funded by Institutional Fund Projects under grant no. (IFPIP: 1039 -135-1443). The authors gratefully acknowledge technical and financial support provided by the Ministry of Education and King Abdulaziz University, DSR, Jeddah, Saudi Arabia.

Appendix. Script for study results is provided after the References section. All codes of all experiments are also available in the link: https://github.com/dervishson/HOM_TTT.

```
import argparse
import pandas as pd
import numpy as np
from sklearn.multioutput import MultiOutputRegressor
from sklearn.svm import SVR
from sklearn.kernel_ridge import KernelRidge
from sklearn.neighbors import KNeighborsRegressor as
KNNR
from sklearn.ensemble import GradientBoostingRegressor as
GBR
from sklearn.ensemble import AdaBoostRegressor as ABR
from sklearn.linear_model import SGDRegressor as SGDR
from sklearn.neural_network import MLPRegressor as MLPR
from sklearn.metrics import r2_score as R
from catboost import CatBoostRegressor as CBR
def data_process(file_name):
data = pd.read_excel(args.file_name) #dataset.xlsx
x = data.sample(frac = 1).reset_index(drop = True)
HOM_mean,TTT_mean = x.mean()[:2]
HOM_std,TTT_std = x.std()[:2]
x["HOM"]=(x["HOM"]-HOM_mean)/HOM_std
x["TTT"]=(x["TTT"]-TTT_mean)/TTT_std
###Train-test split and only second index of j changes when
output changes
X_train,X_test = x.iloc[:450,:2],x.iloc[450:,:2]
y_train,y_test = x.iloc[:450,2:],x.iloc[450:,2:]
X_train,X_test = X_train.values,X_test.values
y_train,y_test = y_train.values,y_test.values
###Output mean and standard deviations
y_tr_means,y_ts_means = y_train.mean(axis = 0),y_test.mean
(axis = 0)
y_tr_stds,y_ts_stds = y_train.std(axis = 0),y_test.std(axis = 0)
#####Normalization of output
y_train,y_test=(y_train-y_tr_means)/y_tr_stds,(y_test-y_ts_me
ans)/y_ts_stds
return X_train,X_test,y_train,y_test,y_tr_means,y_tr_stds,y_ts_
means,y_ts_stds
def work_on(which_model):
X_train,X_test,y_train,y_test,y_tr_means,y_tr_stds,y_ts_means,
y_ts_stds = data_process(args.file_name)
if args.which_model==="SVR":
#define model
model = SVR(C = 10,epsilon = 0.1,gamma = 0.01,kernel=
"rbf")
mor = MultiOutputRegressor(model)
fitted = mor.fit(X_train,y_train)
preds = fitted.predict(X_test)
elif args.which_model==="KRR":
```

(*continued on next page*)

(*continued*)

```
import argparse
#define model
model = KernelRidge(alpha = 0.1,degree = 1,kernel = ''rbf'')
mor = MultiOutputRegressor(model)
#Fit and predict
fitted = mor.fit(X_train,y_train)
preds = fitted.predict(X_test)
elif args.which_model = = ''KNNR'':
#define model
model = KNNR(metric = ''euclidean'',n_neighbors = 1,weigh
ts = ''uniform'')
mor = MultiOutputRegressor(model)
#Fit and predict
fitted = mor.fit(X_train,y_train)
preds = fitted.predict(X_test)
elif args.which_model = = ''GBR'':
#define model
model = GBR(learning_rate = 0.1,max_depth = 5,n_estima
tors = 500)
mor = MultiOutputRegressor(model)
fitted = mor.fit(X_train,y_train)
preds = fitted.predict(X_test)
elif args.which_model = = ''ABR'':
#define model
model = ABR(learning_rate = 0.3,n_estimators = 200)
mor = MultiOutputRegressor(model)
fitted = mor.fit(X_train,y_train)
preds = fitted.predict(X_test)
elif args.which_model = = ''MLPR'':
#define model
model = MLPR(hidden_layer_sizes = 100,max_iter = 1500)
mor = MultiOutputRegressor(model)
fitted = mor.fit(X_train,y_train)
preds = fitted.predict(X_test)
elif args.which_model = = ''CBR'':
#define model
model = CBR(learning_rate = 0.09,max_depth = 6,n_estima
tors = 500)
mor = MultiOutputRegressor(model)
fitted = mor.fit(X_train,y_train)
preds = fitted.predict(X_test)
#Computations
y_test,y_hat = (y_test*y_ts_stds) + y_ts_means,(preds*y_ts_std
s) + y_ts_means
Max_E = np.around(np.max(np.abs(y_test-y_hat),axis = 0),5)
MAE_vals = np.mean(np.abs(y_test-y_hat),axis = 0)
MAE_vals = np.around(MAE_vals,5)
MSE_vals = np.mean(np.power((y_test-y_hat),2),axis = 0)
MSE_vals = np.around(MSE_vals,5)
R_scores = []
for i in range(7):
R_scores.append(R(y_test[:,i],y_hat[:,i]))
a = np.mean(R_scores)
R_scores = np.around(R_scores,4).tolist()
list_1 = [''Load_level'',''Throughput'',''CDR'',''RLF'',''SE'',''
HO PP'',''HO Prob.'']
list_2 = MAE_vals.tolist()
list_3 = MSE_vals.tolist()
list_4 = Max_E.tolist()
list_5 = R_scores
df = pd.DataFrame(list(zip(list_1,list_2,list_3,list_4,list_5)))
df.columns = [''Outputs'',''MAE'',''MSE'',''Max_Errors'',''R
Scores'']
print(df)
```

(*continued*)

```
import argparse
print(''MAE means:'',df[''MAE''].mean())
print(''MSE means'',df[''MSE''].mean())
#print(''Mean absolute values:'',MAE_vals)
#print(''Mean square error values:'',MSE_vals)
def main(args):
np.random.seed(13)
#X_train,X_test,y_train,y_test,y_tr_means,y_tr_stds,y_ts_mean
s,y_ts_stds = data_process(args.file_name)
work_on(args.which_model)
if __name__ = = ''__main__'':
parser = argparse.ArgumentParser(description = 'Result
shows')
parser.add_argument(''--file-name'', type = str,
help = ''Name of the data file'')
parser.add_argument(''--which-model'',type = str,
help = ''Which model will be worked'')
args = parser.parse_args()
main(args)
```

References

[1] Y. Zhou, B. Ai, Handover schemes and algorithms of high-speed mobile environment: a survey, Comput. Commun. 47 (2014) 1–15.

[2] J. Zhao, Y. Liu, C. Wang, L. Xiong, L. Fan, High-speed based adaptive beamforming handover scheme in LTE-R, IET Commun. 12 (2018) 1215–1222.

[3] L. Tian, J. Li, Y. Huang, J. Shi, J. Zhou, Seamless dual-link handover scheme in broadband wireless communication systems for high-speed rail, IEEE J. Sel. Areas Commun. 30 (2012) 708–718.

[4] Y. Zhou, B. Ai, Evaluation of high-speed train communication handover models based on DEA, in: 2014 IEEE 79th Vehicular Technology Conference (VTC Spring), 2014, pp. 1-5.

[5] E. Gures, I. Shayea, A. Alhammadi, M. Ergen, H. Mohamad, A Comprehensive survey on mobility management in 5G heterogeneous networks: architectures, challenges and solutions, IEEE Access, 2020.

[6] T.K. Geok, F. Hossain, S.K.A. Rahim, O. Elijah, A.A. Eteng, C.T. Loh, et al, 3D RT adaptive path sensing Method: RSSI modelling validation at 4.5 GHz, 28 GHz, and 38 GHz, Alex. Eng. J. 61 (2022) 11041–11061.

[7] D. Yang, Y. Zhou, W. Huang, X. Zhou, 5G mobile communication convergence protocol architecture and key technologies in satellite internet of things system, Alex. Eng. J. 60 (2021) 465–476.

[8] I. Shayea, L.A. Nissirat, M.A. Nisirat, A. Alsamawi, T.A. Rahman, M.H. Azmi, et al., Rain attenuation and worst month statistics verification and modeling for 5G radio link system at 26 GHz in Malaysia, Trans. Emerg. Telecommun. Technol., 2019.

[9] I. Shayea, L.A. Nissirat, M.A. Nisirat, A. Alsamawi, T. Abd, M. H. Rahman, Azmi, et al, Rain attenuation and worst month statistics verification and modeling for 5G radio link system at 26 GHz in Malaysia, Trans. Emerg. Telecommun. Technol. 30 (2019) e3697.

[10] A.M. Al-Samman, T.A. Rahman, M.H. Azmi, I. Shayea, Path Loss Model and Channel Capacity for UWB–MIMO Channel in Outdoor Environment, Wirel. Pers. Commun. (2019) 1–11.

[11] I. Shayea, T. Abd. Rahman, M. Hadri Azmi, A. Arsad, Rain attenuation of millimetre wave above 10 GHz for terrestrial

links in tropical regions, Trans. Emerg. Telecommun. Technol., vol. 29, p. e3450, 2018.
[12] M. Ergen, F. Inan, O.E.I. Shayea, M.F. Tuysuz, A. Azizan, N. K. Ure, et al., "Edge on Wheels with OMNIBUS Networking in 6G Technology," IEEE Access, pp. 1-1, 2020 2020.
[13] S. Khosravi, H. S. Ghadikolaei, M. Petrova, Learning-based load balancing handover in mobile millimeter wave networks, in: GLOBECOM 2020-2020 IEEE Global Communications Conference, 2020, pp. 1-7.
[14] F. Jiang, C. Feng, H. Zhang, A heterogenous network selection algorithm for internet of vehicles based on comprehensive weight, Alex. Eng. J. 60 (2021) 4677–4688.
[15] S. Alraih, I. Shayea, M. Behjati, R. Nordin, N.F. Abdullah, A. Abu-Samah, et al, Revolution or Evolution? Technical requirements and considerations towards 6G Mobile Communications, Sensors 22 (2022) 762.
[16] W.K. Saad, I. Shayea, B.J. Hamza, H. Mohamad, Y.I. Daradkeh, W.A. Jabbar, Handover Parameters Optimisation Techniques in 5G Networks, Sensors 21 (2021) 5202.
[17] I. Shayea, M. Ergen, M.H. Azmi, S.A. Çolak, R. Nordin, Y.I. Daradkeh, Key challenges, drivers and solutions for mobility management in 5G Networks: A Survey, IEEE Access 8 (2020) 172534–172552.
[18] E. Gures, I. Shayea, M. Ergen, M.H. Azmi, A.A. El-Saleh, Machine learning based load balancing algorithms in future heterogeneous networks: a survey, IEEE Access, 2022.
[19] A. Kumar, M. Gupta, A review on activities of fifth generation mobile communication system, Alex. Eng. J. 57 (2018) 1125–1135.
[20] Y. Chen, K. Niu, Z. Wang, Adaptive handover algorithm for LTE-R system in high-speed railway scenario, IEEE Access 9 (2021) 59540–59547.
[21] S. Alraih, R. Nordin, I. Shayea, N.F. Abdullah, A. Abu-Samah, A. Alhammadi, Effectiveness of handover control parameters on handover performance in 5G and beyond mobile networks, Wirel. Commun. Mob. Comput. 2022 (2022).
[22] L. Goratti, S. Savazzi, A. Parichehreh, U. Spagnolini, Distributed load balancing for future 5G systems on-board high-speed trains, in: 1st International Conference on 5G for Ubiquitous Connectivity, 2014, pp. 140-145.
[23] X. Cai, C. Wu, J. Sheng, J. Zhang, Y. Wang, A parameter optimization method for LTE-R handover based on reinforcement learning, Int. Wireless Commun. Mobile Computing (IWCMC) 2020 (2020) 1216–1221.
[24] D. Li, D. Li, Y. Xu, Machine learning based handover performance improvement for LTE-R, in: 2019 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW), 2019, pp. 1-2.
[25] J.-H. Bang, S. Oh, K. Kang, Y.-J. Cho, A Bayesian regression based LTE-R handover decision algorithm for high-speed railway systems, IEEE Trans. Veh. Technol. 68 (2019) 10160–10173.
[26] I. Shayea, M. Ergen, A. Azizan, M. Ismail, Y.I. Daradkeh, Individualistic dynamic handover parameter self-optimization algorithm for 5G networks based on automatic weight function, IEEE Access 8 (2020) 214392–214412.
[27] I.M. Shayea Ibraheem, Nordin Rosdiadee, Downlink spectral efficiency evaluation with carrier aggregation in LTE-Advanced system employing Adaptive Modulation and Coding schemes, in: IEEE Malaysia International Conference on Communications (MICC2013) 2013, pp. 98-103.
[28] M. Alhabo, L. Zhang, Load-dependent handover margin for throughput enhancement and load balancing in HetNets, IEEE Access 6 (2018) 67718–67731.
[29] S.S. Mwanje, A. Mitschele-Thiel, A q-learning strategy for lte mobility load balancing, in: 2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), 2013, pp. 2154-2158.
[30] E. Gures, I. Shayea, M. Ergen, A.A. El-Saleh, Fuzzy logic-based load balancing algorithm in heterogeneous networks, Workshop on Microwave Theory and Techniques in Wireless Communications (MTTW) 2022 (2022) 210–215.
[31] A. Goldsmith, Wireless communications, Cambridge University Press, 2005.
[32] I. Shayea, M. Ismail, R. Nordin, H. Mohamad, Handover performance over a coordinated contiguous carrier aggregation deployment scenario in the LTE-advanced system, Int. J. Vehic. Technol. 2014 (2014).
[33] P. Bahl, A. Adya, J. Padhye, A. Walman, Reconsidering wireless systems with Multiple radios, Comput Commun Rev 34 (2004) 1–8.
[34] S. Jeon, S. Lee, A relay-assisted handover technique with network coding over multihop cellular networks, IEEE Commun Lett 11 (2007) 252–254.
[35] R. Pabst, B. Walke, D. Schultz, Relay-based deployment concepts for wireless and mobile broadband radio, IEEE Commun. Mag. 42 (2004) 80–89.
[36] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1997) 119–139.
[37] O. Lyashevska, C. Harma, C. Minto, M. Clarke, D. Brophy, Long-term trends in herring growth primarily linked to temperature by gradient boosting regression trees, Eco. Inform. 60 (2020) 101154.
[38] J.H. Friedman, Greedy function approximation: a gradient boosting machine, Ann. Stat. (2001) 1189–1232.
[39] L. Prokhorenkova, G. Gusev, A. Vorobev, A.V. Dorogush, A. Gulin, CatBoost: unbiased boosting with categorical features, Adv. Neural Inf. Proces. Syst. 31 (2018).
[40] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1967) 21–27.
[41] T. Hastie, R. Tibshirani, J.H. Friedman, J.H. Friedman, The elements of statistical learning: data mining, inference, and prediction vol. 2: Springer, 2009.
[42] T. Gasser, H.-G. Müller, Kernel estimation of regression functions, in: Smoothing Techniques for Curve Estimation: Proceedings of a Workshop Held in Heidelberg, April 2–4, 1979, 1979, pp. 23-68.
[43] K. P. Murphy, "Machine learning: A probabilistic perspective (adaptive computation and machine learning series)," ed: The MIT Press: London, UK, 2018.
[44] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (1995) 273–297.
[45] M.F.A. Fauzi, R. Nordin, N.F. Abdullah, H.A. Alobaidy, Mobile network coverage prediction based on supervised machine learning algorithms, IEEE Access, 2022.
[46] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (1986) 533–536.