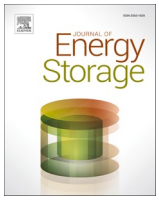
Journal of Energy Storage 72 (2023) 108268

Contents lists available at ScienceDirect

Journal of Energy Storage

journal homepage: www.elsevier.com/locate/est



Research papers

SoC estimation of lithium-ion batteries based on machine learning techniques: A filtered approach

Mehmet Korkmaz *

Aksaray University, Department of Electrical & Electronics Engineering, Aksaray, Turkey

ARTICLE INFO

Keywords:
Lithium-ion battery
Machine learning
Outlier removal
State of charge (SoC) estimation

ABSTRACT

Accurate state-of-charge (SoC) estimation is an essential requirement for many situations where Li-Ion batteries (LiBs) are used. This ensures an efficient battery management system (BMS), so the battery can be protected from excessive discharge, and its life span can be maximized. But when it comes to electrified vehicles (xEVs), the SoC estimation accuracy becomes a more critical and indispensable prerequisite. Because the safety of xEVs during driving and the remaining range, which is an indicator of how far the vehicle can go, are directly related to the accurate SoC. However, the complex electrochemical reactions in the battery and the dependence on environmental variables make SoC estimation a challenging task. Traditionally, this is tackled by establishing either electrochemical or electrical battery equivalent models. Both methods suffer from some limitations, such as parameter identification, complex calculations, and model mismatching due to the aging factor. On the other hand, data-driven methods have recently become a popular choice for SoC estimation since they enable building data-based models rather than chemical reactions or equivalent circuit calculations. The model is built based on battery parameters such as current, voltage, battery type, and then used for SoC estimation. However, many studies in the literature examine only a few methods for SoC estimation. Also, these data-driven black box models can lead to outlier data as they are not observers. Thus, the aims of this study are twofold: First, to make a comprehensive comparison based on most of the ML methods. Second, to utilize several filters for outlier removal and measure their effectiveness. For these purposes,18 ML algorithms were handled in three main groups, and SoC estimation results were analyzed. Additionally, five different filters were used to improve the SoC estimation of these methods, and their comparisons were realized. From the results, it is clear that Bagging and ExtraTree algorithms are substantially better than other ML methods for SoC estimation since their Interquartile Range (IQR) is smaller than 3%, performance indices are the lowest ones, and curve matches are the best. Also, Rloess is the best filter among the others, although they all achieved high performance in outlier removal.

1. Introduction

Countries aim to reduce greenhouse gas emissions due to concerns about climate change and global warming. Most of them establish targets to achieve a carbon neutrality level in the upcoming decades [1]. However, transportation is still a major contributor to this threat, and it is a growing source of emissions around the globe according to the Emissions Gap Report [2]. Governments have begun to take actions to deal with this problem. For example, UK Government bans the sale of new internal combustion engine (ICE) vehicles as of 2030. Electric Vehicles (xEVs) herein come to prominence since their carbon emissions are lower than conventional ones over the lifespan. They are also naturally quieter than conventional ones. Furthermore, the transition to xEVs will prevent air pollution and improve air quality in city centers. These advantages of xEVs will contribute to the governments meeting their goals. The batteries of xEVs are primarily based on Lithium-Ion or its derivatives. These batteries are characterized by their relatively high power-to-weight ratio, meaning even small battery packs can provide high power. Their energy density is higher, which allows them to reduce the overall size and weight of the battery pack. Fast charging, long service life, low memory effect, low pollution, and low self-discharge rate are also other merits of these batteries [3,4]. Although these specialties are considered as positive and desirable sides, research shows that these batteries suffer from internal state estimation, especially two significant parameters, state of health (SoH) and state of charge (SoC), since they are directly related to the battery performance [4,5]. SoH is a

* Corresponding author.
E-mail address: mehmetkorkmaz@aksaray.edu.tr.

https://doi.org/10.1016/j.est.2023.108268
Received 19 December 2022; Received in revised form 2 June 2023; Accepted 1 July 2023
Available online 7 July 2023

measurement that indicates the battery's general condition relative to its initial value, while SoC can be described as the amount of available energy in a battery relative to its nominal capacity. Both are usually expressed as a percentage value. Particularly, SoC plays a crucial role in both drivers' comfort and control of the Battery Management System (BMS). First, this is critical for drivers because they must be informed correctly about the remaining driving range displayed on the car's dashboard to arrange their trips. Secondly, reliable, safe, and effective BMS conditions can only be provided with the help of accurate SoC estimation values [6,7]. These examples highlight the significance of accurate SoC estimation. However, SoC estimation is considered a rigorous task due to difficulties in exact modeling, aging effects that alter the performance of the batteries, and other problems [4,8]. This turns out to be even more problematic because of the highly nonlinear nature of the batteries. A large number of existing studies in broader literature have been summarized in the next section.

1.1. Review of SoC estimation methods

Having accurate SoC estimation is of great importance as correct values of SoC can determine the best operating strategies. One step to overcome the accurate SoC estimation problem is the correct modeling of the battery. Electrochemical model (EM) is the first option for true battery models. This method considers the battery's internal physical and chemical properties, therefore, it is possible to reveal the battery degradation clearly [9]. In contrast, its computational burden is high and somewhat complex as it uses partial differential equations (PDEs). These disadvantages confine its practical implementation and make it a less preferable choice in SoC estimation studies [7,10,11].

On the other hand, the second and relatively more common approach for battery modeling is the electrically equivalent circuit models (ECMs). With these methods, the battery can be represented by electrical circuit components such as resistance, capacitor, voltage source, etc.; thus, the working characteristics of the battery can be analyzed over a circuit network. This has been discussed by a great number of authors in literature and several techniques are reported to address this issue [12,13]. For instance, the most widespread approach to simulate the battery behavior is the circuits consisting of several *n-RC* branches accompanied by a series resistor and voltage source. This technique is also called Thevenin circuits by *n-RC* branches. The coefficient *n* determines the number of parallel branches, which refers to the different battery states. It is usually known as the R_{int} model when *n* is equal to *zero*, OTC (one-time constant) model when *n* is *one*, and TTC (two-time constant) model when *n* is selected as *two*. These representations have different names in several studies, such as the Thevenin model for *1-RC* (OTC), *DP* (Dual Polarization) model for *2-RC* (TTC) etc. [9,14]. Adding more *RC* branches generally improves the model accuracy. However, this significantly increases computational complexity and makes the model parameter identification more difficult and complicated [9]. Efforts to address this include Ding et al., who proposed an improved Thevenin model considering the temperature influence [15]. They claimed that the parameter identification error could be less than 1 % using their model. Kai et al. offered ASRUKF algorithms for SoC estimation using the second order Thevenin ECM [16]. More comprehensive and detailed descriptions of ECMs can be found in literature review papers [12,13,17].

In addition to this, fractional components have been introduced to improve the model accuracy in several studies. The modification usually occurs by replacing the capacitor in *RC* branches with constant phase elements (CPEs) [18,19]. Some papers also offer an extra component called Warburg [18], which is connected to the equivalent circuit model in series. All these fractional order-based models come into prominence with the extra parameters to adjust the order of the fractional elements. Therefore, it is possible to define the battery model with high accuracy. For instance, in [18], Liu et al. established a fractional order circuit model for SoP estimation and used the HPSO algorithm to identify the parameters of the model. In [19], Tian et al. proposed least squares and gradient-based methods to determine the parameters of the fractional order model for Li-Ion batteries. Zou et al. gives a detailed review of fractional-order techniques applied to lithium-ion batteries, lead-acid batteries, and supercapacitors [20].

Together with the battery model, estimation algorithms also directly affect the accuracy of SoC estimation. The most popular way to estimate the SoC is the Kalman-based filters (KF) that consider the estimation problem in a recursive process. High nonlinearity and hysteresis effects in the battery require the nonlinear versions of Kalman filters. EKF (Extended Kalman Filter) takes the lead in these methods and linearizes around the current mean and covariance of the current state. UKF (Unscented Kalman Filter) is an alternative way to overcome linearization errors encountered in the EKF. This method is based on the Unscented Transform (UT) representing the whole distribution with the so-called sigma points. These methods have been discussed by a multitude of papers in the literature [16,17]. In UT transform, the sigma point calculation is performed using the state covariance matrix at each time update phase, and this operation puts an extra computational burden on the algorithm. To handle this, the square-root UKF (SR-UKF) algorithm was proposed, which benefits from effective linear algebra techniques such as QR decomposition and Cholesky factorization. Liu et al. proposed SR-UKF algorithm on Lithium cobalt oxide battery for SoC estimation, and they claimed that robustness was increased and SoC estimation error reduced to the traditional ones [21]. All these methods consider the process and measurement covariances as constant. Choosing inappropriate values of these parameters may lead to more significant errors and divergence in the overall estimation process. To handle this, adaptive versions of these algorithms have also been offered. These adaptive-based schemes update the process and measurement covariances, and different strategies are used to determine the values. For example, in [22], Chen et al. combined residual covariance function with measurement noise covariance matrix, Zhu et al. utilized from fractional order calculus [23]. A detailed review of the KF-based methods, including adaptive schemes, can be found in the review papers [17,24].

As stated in the above explanations, in some cases, neither the battery model nor the estimation algorithm would ensure sufficient accuracy in SoC estimation due to nonlinearities, aging factors, environmental conditions, incorrect modeling, etc. To address this problem, numerous recent works have shown that the SoC estimation problem can also be tackled by using Machine Learning (ML) algorithms. These methods offer a significant advantage as they are generally less dependent on the battery model. The benefits of these systems are the direct modeling using input/output (I/O) data, and they are almost suitable for all types of batteries regardless of their technology [25,26]. Therefore, it is possible to estimate the SoC values with a trained model using the related battery I/O data. The connection between SoC and input can be established by $SoC = \xi(I, V, T)$. $\xi(\bullet)$ is a map that transforms input data to output data, and it usually consists of current (I), voltage (V), and temperature (T) values. Some papers also use average current and voltage, battery type, power etc. [3,6]. The structure of $\xi(\bullet)$ is related to the ML algorithm and can be rule-based, linearly connected or tree-based.

Linear methods are the oldest and most trivial approach known among ML methods. These algorithms predict unknown values by creating the best hyperplane between the input and output data. Although it is an old-fashioned method, it emerges as a reference choice in ML applications due to its easy implementation, fast training, and straightforward interpretation. There are some studies using linear models for SoC estimation. For example, voltage and current values were used as input data of Linear Regression to estimate the SoC in [27]. The results were compared with other ML methods, Random Forest and ANN, over RMSE values, and it was emphasized that the Random Forest method gave better results. Linear regression and other ML methods were used for SoC estimation of Smartphone batteries in [28]. According

to the RMSE performance metric values, it was mentioned that the Linear Regression method had worse results than other methods, especially tree-based methods.

Another main branch of ML algorithms is tree-based ones which generally benefit from if/then rule-based blocks. Unlike linear models, they map non-linear relationships reasonably well [29,30]. Among tree-based models, ensemble methods become different by creating multiple models and combining them to produce more accurate solutions. Gradient Boosting, XGBoost, and Bagging algorithms are the most well-known ensemble methods. Ensemble methods are preferred in many studies, especially because of the representation of non-linear relations, the high accuracy of the results, and the excellent handling of missing data. Different types of ensemble learning algorithms are used for SoC estimation as well. For example, ensemble Bagging and Boosting methods were compared with SVR and ANN in [30]. Their results were evaluated in terms of MSE and RMSE performance metrics. The study stated that the ANN method results were better than the ensemble methods. Although they adjusted the parameters of ANN, they did not mention the parameter setting of ensemble methods. Ipek et al. used SVR and XGBoost algorithms to estimate SoC for Lithium iron phosphate battery [31]. Algorithm results were compared over RMSE and R2 performance metrics, and the results of training times were also presented. Accordingly, the XGBoost algorithm had better results and completed the algorithm training in a shorter time than the SVR algorithm. A hybrid model of SVR and ensemble learning approaches was proposed in [6]. A clustering algorithm divided the original data set into multiple subsets. Then, an SVR estimation model was established for each data subset. The accuracy of the proposed model was evaluated by different driving cycles.

Apart from these mentioned ML groups, there are also various ML algorithms with different types of design that are not classified within these groups. For instance, ANN is one such ML method that is analogous to a biological neural network. Artificial neurons represent the neurons in the biological brain, and connections between them transmit the data, and these connections create the ANN model. This algorithm has been used in several studies to assess the SoC estimation performance of batteries. Normalized current and voltage were used as input and SoC was used as output data to train the network in [32]. The algorithm was supported by the UKF algorithm to reduce the error. The proposed method was validated by Federal Driving Schedule and dynamic stress testing, and the RMSE of SoC was within 2.5 %, and the maximum error was 3.5 % for different temperatures. The relationship between current, voltage, temperature, and SoC was formed with the SVM method in [33]. While building the model, a moving window was used to overcome the missing data problem. The accuracy of the model was validated by different current profiles.

All the above methods can be used separately to estimate the SoC, and in some situations, they can be merged to improve accuracy and performance. Those hybrid structures usually outperform those with a single algorithm without compromising the overall efficiency [34]. Several studies even add filters to enhance the SoC estimation performance by eliminating the noisy values. Jia et al. proposed a Savitzky-Golay filter to process the estimated SoC [35]. Sidhu et al. offered hybrid ML techniques together with the Gaussian filter and claimed that the proposed model reduced the MAE by 2.88 compared to the SVR and ANN [29].

1.2. Contribution of the study

As stated in Section 1.1, accurate SoC estimation is crucial for safe operation and efficient BMS. Many studies on this subject are filter-based methods that include classical Kalman or its derivatives. Thanks to their recursive nature, these methods continuously observe the system states and make accurate SoC predictions. However, their performance tightly relies on true battery representation and parameter identification. As mentioned in the review section, in most cases, a true representation of the battery is a common problem due to its nonlinearity and uncertain parameters. Myriad problems in battery modeling make the ML algorithms an alternative to accurate SoC estimation since they do not require to impose either chemical or electrical models. Well-trained ML methods often produce better results than classical filter-based methods [25]. Previous studies have shown that many take into account only a few ML methods to examine their proposed model. They offer the best ML technique for their works with the best parameters and do not present a fair comparison with other ML models. To the best of our knowledge, there is no systematic comparison involving many ML methods for the SoC estimation of LiBs. In addition, the output filter design and its effects are rarely analyzed in the literature. In order to provide researchers with comprehensive and systematic guidance for future research in the related field, this work presents recent ML methods along with the output filter techniques for SoC estimation of LiBs used in xEVs, in which the key contributions are as follows:

(1) In order to reveal which ML group and method is the best for SoC estimation and to obtain comprehensive comparison results between ML algorithms, SoC estimation is performed using 18 different ML methods divided into three different subgroups.
(2) To overcome the problem of meaningless output data, filters are applied to ML outputs, and five different filters from various subgroups are compared to see how the system output will be affected and which is the best choice for the output of that system.

2. Materials and methods

2.1. Machine learning algorithms

Machine learning (ML) is a type of artificial intelligence (AI) that allows machines the ability to automatically learn from data and previous experiences to identify patterns and make predictions. Using data for system predictions and classifications comes in handy, especially when the models are highly nonlinear, and it takes work to define system equations. One such system is a battery which is not easy to describe mathematical models since its internal chemical reactions are highly dependent on several variables such as temperature, instantaneous current, and voltage. This makes accurate SoC estimation difficult, which is hugely significant for BMS. From this point of view, ML methods have been used for battery SoC estimation in recent years. By building correct models, ML methods usually outperform classical techniques. Table 1 summarizes the latest studies on the SoC estimation of batteries with ML algorithms.

As stated in the first Section and seen in Table 1, many works chose only a couple of ML techniques to make SoC estimation, and these were usually not compared to other ML algorithms. In this study, 18 different ML algorithms under three different subclasses were used to make SoC estimation. Groups and methods are briefly summarized in the following explanations:

2.1.1. Group 1 (linear models)

This subclass is devoted to linear models, and the first algorithm in this group is Linear Regression (1). This model attempts to make the relationship between input and output data by fitting a model with coefficients. The training process is completed by obtaining the best coefficients that minimize the input/output. The second one is Elastic net (2) which is similar to the first one but considers the L1 and L2 norms to determine the coefficients. The third one is Gradient Descent (SGD – (3)). It takes the gradient of the objective function to minimize error between input/output. The fourth algorithm is the Bayesian Ridge (4). This method is created by determining some coefficients in Bayesian regression by a gamma distribution. This distribution is used to optimize the loss function. The next one is the Lasso (5), which combines the regularization and selection of variables. Lastly, the last method is the RANSAC (6) regression. This algorithm distinguishes inlier and outlier

Table 1
Recent studies on SoC estimation of batteries using ML methods.

Ref.	Method	Input/Output	Performance index	Battery
[29]	Rand. For., SVR, ANN	I, V, dV / SoC	MAE, COD	18650 LNMC
[6]	SVR, Ensemble	I, V, T / SoC	RMSE, MAE, MAPE	18650-20R
[30]	ANN, SVR, Lin. Reg. Bagging, Boosting	I, V / SoC	MSE, RMSE, MAPE, MAE, NRMSE	18650FP; Panasonic
[31]	SVR, XGBoost, Bagging	I, V, T / SoC	RMSE, R2	26650 M1
[36]	Adaboost and its derivatives	I, V, T / SoC	MAE, RMSE, MAPE	18650
[25]	ANN, UKF	I, V, T / SoC	RMSE, R2	AA1F329TS/2-B; Samsung
[37]	SVR, ANN, LSTM	I, V, T, W (motor, differential, wheels, breaks, loss rolling resistance, loss drag) / SoC	MSE	Tesla S/ Nissan Leaf battery packs
[4]	SVR, ANN, GPR	I, V, T / SoC	RMSE, MAE	Mitsubishi Electric Corporation, 4,93 Ah
[3]	DNN	I, V, T, Iavg, Vavg / SoC	MAE	NCR18650PF; Panasonic

data, and training is provided with the inlier ones. It particularly produces good results when the data has several outlier sections.

2.1.2. Group 2 (ensemble models)

Ensemble learning ML methods are subject to this subclass. The first technique is the Gradient Boosting Algorithm (7). This is one of the fundamental ensemble ML algorithms that creates weak learners. Bagging (8) is the second algorithm of this Group that makes the training and predictions by combining different decision trees. The training process is also carried out by different random subclasses. The third algorithm is the XGBoost (9) which is an enhanced version of (7) in a way using the advanced L1 and L2 norms. The fourth one, Histogram Based Gradient Boosting (10), is a faster solution to train the trees. The acceleration is provided with discretizing of the continuous input data. The next technique is the LightGBM (11) which is again an advanced version of the (7). It combines the tree vertically, usually consumes less memory, and is faster than (7). The last method of this group is the AdaBoost (12) algorithm that adjusts the weights of different trees evaluated at each step.

2.1.3. Group 3 (other models)

ML regression algorithms other than the above two groups are examined under this Group. The first method is a tree-based Random Forest (13). It considers the mean values of trees as regression output data. Although it is a kind of ensemble method, it is not evaluated under Group 2 since its decision tree creation and aggregation are not similar to the methods investigated in Group 2. The second algorithm is the Decision Trees (14) which creates a set of decision rules to learn the model. The third technique is the kNN regression (15). This method makes neighbor sets and calculates the new data point value in the regression problem considering the data on the training set. The fourth method is the ANN (Artificial Neural Network, (16)) which is entirely different from any methods introduced so far in building models since it simulates the workings of neurons in the human brain. The next regression algorithm is the SVR (17) which is also known as one of the conventional ML methods. Input and output data are linked over a hyperplane. The final member of this subclass is the ExtraTree (Extremely Randomized Trees (18)). This is also similar to (13), but it randomly chooses the split point instead of calculating the optimal one and uses the whole original sample. Some important parameters of the ML algorithms used in the study are shown in Table 2. Other than these values several significant hyper-parameter sets were also investigated for each algorithm in the study, but no remarkable differences were observed in terms of SoC estimation results.

2.2. Filters

Battery SoC estimation involves collecting battery data such as current, voltage, temperature and estimating it using a model or an algorithm based on these data. These physical values can be directly exposed to external noise while measuring or evaluating in the microprocessor and affect the SoC estimation results. Therefore, this problem necessitates the use of filters. This situation is usually ignored when traditional Kalman-based filters are used since they behave as observers while making state predictions. On the other hand, this problem has to be considered if ML algorithms are used as an estimator. Because outliers may occur in any data-driven method due to different reasons. For example, a dataset may have missing values at some points. Furthermore, the data may have a highly complex relationship between its variables. Above all training process is entirely a black box that is defined by the input-output relationship for all data-driven methods. These all may affect the training process and can lead to learning wrongly in those regions. As a result, the prediction at that step(s) may unexpectedly vary from previous or future predictions which causes to outliers [26]. Therefore, better SoC estimation can be obtained using the filters. From this point of angle, this study considers five different filter algorithms to make better SoC estimations and compare these filters with one another. The following lines give a brief overview of these methods.

Table 2
ML algorithms in groups and some important parameters.

Groups	#	Algorithms	Parameters
Group 1 (Linear Models)	1	Linear Reg.	**sample weigth:** individual samples for each sample
	2	ElasticNet	α: 1, L1_ratio: 0.5, **max_iter**: 300, **tolerance**: 1e-4
	3	SGD	α: 0.0001, **max_iter**: 300, **tolerance**: 1e-4, **loss**: squared_error, **penalty**: L2 epsilon: 0.1, eta0: 0.01, power_t: 0.25
	4	Bay. Ridge	**max_iter**: 300, **tolerance**: 1e-4,α_1: 1e-6, α_2: 1e-6, λ_1: 1e-6, λ_2: 1e-6
	5	Lasso	α: 1, **max_iter**: 300, **tolerance**: 1e-4
	6	RANSAC	**max_trials:** 300, **stop_probability**: 0.9, **loss**: absolute_error
Group 2 (Ensemble Models)	7	Grad. Boost	**loss**: squared_error, **tolerance**: 1e-4, **learning_rate**: 0.1
	8	Bagging	**loss**: squared_error, **tolerance**: 1e-4, **learning_rate**: 0.1
	9	XGBoost	**loss**: squared_error, **tolerance**: 1e-4, **learning_rate**: 0.1
	10	Histogram	**loss**: squared_error, **tolerance**: 1e-4, **learning_rate**: 0.1
	11	LightGBM	**loss**: squared_error, **tolerance**: 1e-4, **learning_rate**: 0.1
	12	AdaBoost	**loss**: squared_error, **learning_rate**: 0.1
Group 3 (Other Models)	13	Rand. Forest	**criterion**: squared_error, **max_leaf_nodes**: 50
	14	Dec. Tree	**criterion**: squared_error, **max_leaf_nodes**: 50
	15	kNN	**neighbours**: 5, **weights**: uniform, **metric**: minkowski
	16	ANN	**hidden_layer**: 100, **activation**: ReLu, **solver**: adam, **max_iter**: 300, **tolerance**: 1e-4, **momentum**: 0.9
	17	SVR	**kernel**: rbf, **tolerance**: 1e-4, **c**:1
	18	ExtraTree	**criterion**: squared_error, **max_leaf_nodes**: 50

2.2.1. Gauss filter

It deals with the mean values within a specific window. This case is also known as the mean filter, but Gauss filter selects the values in terms of Gaussian distribution [38].

2.2.2. Savitzky-Golay (Sgolay) filter

Polynomial and the least squares method are used for curve fitting. Although its efficiency is slightly less than Gauss Filter in noise reduction, it effectively preserves the signal shape. This filter allows outlier detection, keeping the original curve as much as possible and increasing data precision [35].

2.2.3. Rlowess & Rloess filters

Both are extended and robust versions of lowess and loess filters. Their computational load is higher than the conventional ones, but they are more robust. Rlowess and Rloess methods use locally weighted linear regression [39,40].

2.2.4. Median filter

It is a sort of nonlinear filtering method that handles the median values of a specific window so noise in the data can be eliminated.

3. Results and discussion

Evaluation of SoC estimation is a complex task and should be interpreted from different perspectives. Choosing inappropriate evaluation criteria or evaluating the result from one perspective may lead to false judgments about the effectiveness of a method and raise doubts about the study's objectivity. Therefore, this study evaluates 18 different machine learning algorithms from four perspectives: curve consistency, performance indices, statistical, and time, to present a fair, ideal, and quantitative comparison between algorithms. First, SoC estimation curves of algorithms are plotted along with the reference value. That way, the tendency of SoC estimation of ML methods and groups can easily be observed regarding the reference value. Several performance indices which use the estimation values and reference values of data points are utilized as a second quantitative evaluation technique. The formulas are described through eqs. 1 to 3. Along with many other works which calculate MAE and MSE [8], we also take into account the Tweedie deviance error as it is an essential criterion for evaluating regression tasks. This metric has a power parameter, and it is called a mean Poisson deviance error when the power is equal to *one* [41].

$$MSE(y_i, \widehat{y}_i) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2 \tag{1}$$

$$MAE(y_i, \widehat{y}_i) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y}_i| \tag{2}$$

$$D(y_i, \widehat{y}_i) = \frac{1}{n}\sum_{i=1}^{n} 2\left(y_i log\left(\frac{y_i}{\widehat{y}_i}\right) + \widehat{y}_i - y_i\right) \tag{3}$$

where, n refers to the number of data, y_i is the reference value of the sample at point i, and $\widehat{y}_i$ is the predicted value of the sample at point i.

As a third evaluation, time comparisons for training and predicting steps are also presented. This comparison is analyzed as a separate part in Section 3.4. Furthermore, apart from the existing literature, we also evaluate the SoC error change in terms of statistical methods.

3.1. The battery test setup and overall working scheme

To validate the findings in this study, we used the publicly available dataset presented in [42,43,47]. This experimental dataset belongs to the LG HG2 18,650 cylindrical battery, which has 3000 mAh nominal capacity, 3.60 V nominal voltage, 240 Wh/kg energy density, operates at −5– 50 °C for charging and -20– 75 °C for discharging. The experimental procedure to obtain the battery parameters consists of several steps. First, the battery is located in a thermal chamber to prevent temperature oscillation and then, it is charged and discharged over different automotive industry standard driving cycles such as US06, HWFET, and LA92. Finally, battery parameters such as voltage, current, temperature, and consumed energy are measured and collected during the test period. With this data, the ML database was created by combining and normalizing these values. The database consists of normalized values of voltage, current, temperature, average voltage, average current, and SoC. Since the values of variables vary over a vast and irrelevant range, normalized values are chosen for training ML algorithms. The database was then split into training and test sets. Tuned and trained ML algorithms were evaluated with test data that was not seen previously (Fig. 1).

3.2. SoC estimation evaluation

Training and test steps were conducted for different machine learning methods using the experimental data obtained with the setup mentioned in Section 3.1. First, the SoC estimation results of the algorithms within the same group were examined (Figs. 2, 4, and 6). Fig. 2 (a) refers to the SoC estimation results curve of the Group 1 algorithms. When Fig. 2 (a) is examined, it is seen that all algorithms except ElasticNet and Lasso follow the true SoC values, albeit with a certain error. Fig. 2 (b) and (c) are zoomed parts in Fig. 2 (a) at turning points of the graph. The prediction results, especially in those points, are slightly far away from the true values. As stated in Section 1, we speculate that this might be due to state transition of SoC from continuous to flat values. This problem is repeated for all algorithms in Group 1 except Lasso and ElasticNet. Lasso and ElasticNet algorithms have clearly produced the worst results. This probably stems from the multiple coefficients in their objective functions.

On the other hand, this evaluation criterion does not provide explicit results regarding the best algorithm within Group 1, as the curves are closer to each other. Therefore, it is necessary to evaluate the results from different angles. Table 3 presents the performance indices of Group 1 algorithms. It is seen in Table 3 that Bayesian Ridge and Linear Regression algorithms have the best scores regarding the performance indices, and the MSE value of Bayesian Ridge is slightly better than Linear Regression.

On the other hand, clearer results can be obtained by evaluating the algorithms from a statistical view rather than evaluating the SoC curves or performance index values. To do so, SoC estimation errors are calculated for each sample by obtaining the difference between reference and predicted values ($\%error = 100 \times (y_{true} - y_{prediction})$). Fig. 3 illustrates the SoC estimation error change of Group 1 algorithms. In the graph, the dashed red line on the horizontal axis shows the 0% error line. The vertical axis represents the percentage error change, and an absolute increase in this value in a positive or negative direction indicates that the algorithm moves away from correct predictions. Furthermore, error changes for all data points are represented in a violin plot since it depicts both the box plot and kernel density. In the graph, green rectangle boxes represent the interquartile ranges, and the thinner blue lines represent the 1.5 Interquartile Range (IQR). The outer gray line on each side of the graph is a kernel density estimation to illustrate the distribution shape of the data. Wider sections of the plot point out a higher probability while the skinnier ones represent a lower probability. The yellow circles with a plus sign inside the rectangles refer to the median value of all data for the relevant method. Hyphens pointed with a red arrow above and below the distribution respectively represent the maximum and minimum levels of the data. Considering these explanations, the results lead to a similar conclusion that we obtained in Fig. 2 and Table 3. For example, Lasso and ElasticNet have the biggest boxes and most extended distributions, making them the worst algorithm within Group 1. On the other hand, error changes of other algorithms are generally within the range of ±10 %. Linear Regression, Bayesian Ridge, and RANSAC

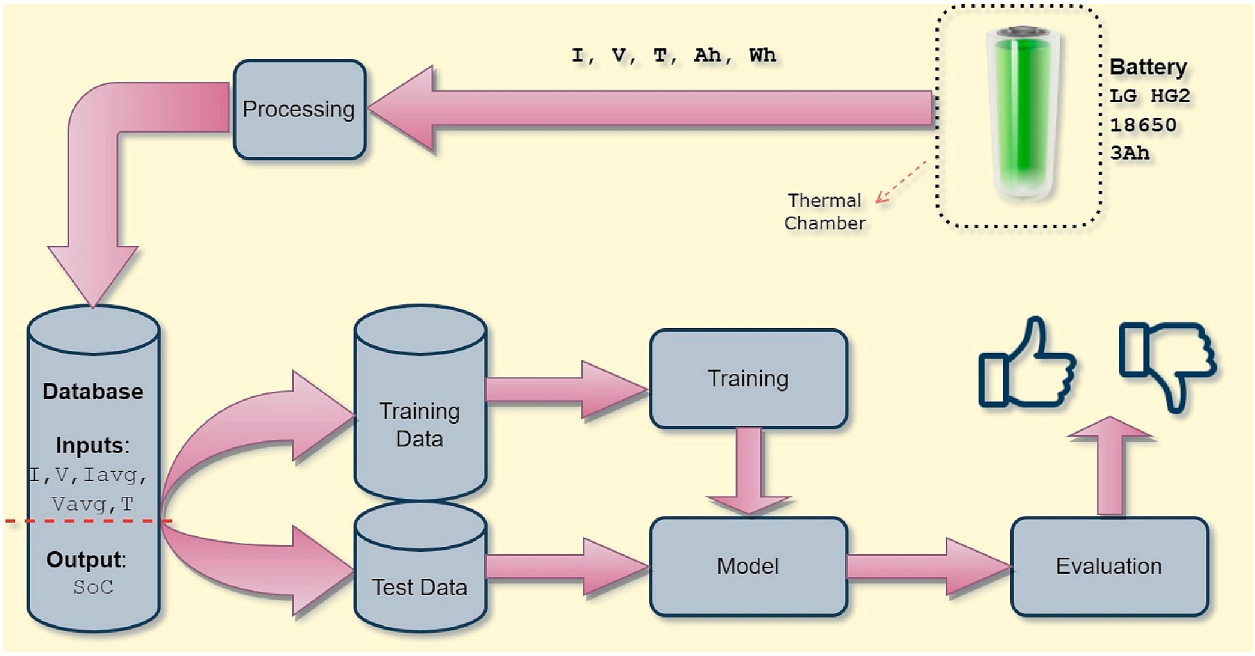


Fig. 1. The overall scheme of the study.

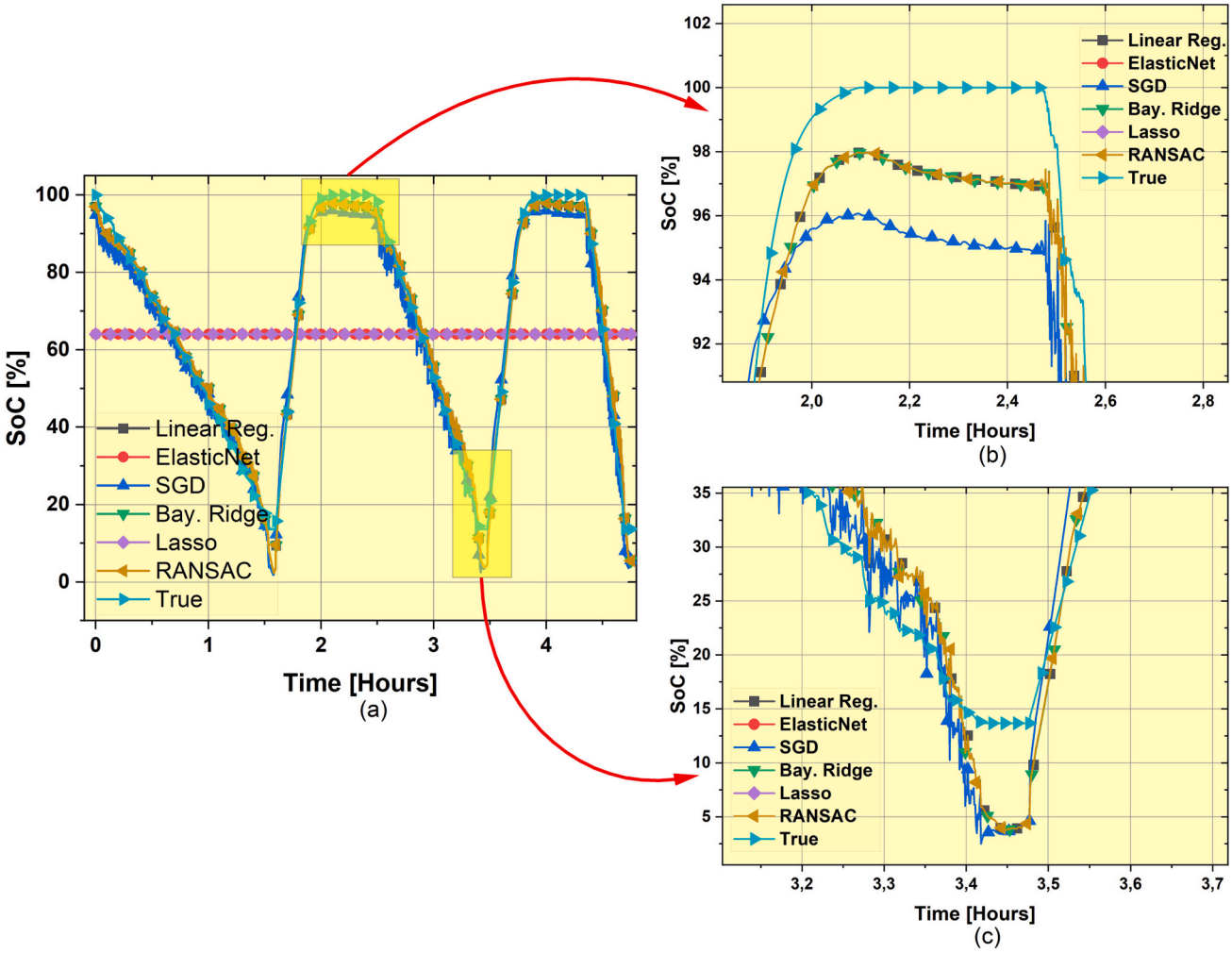


Fig. 2. SoC estimation results of all algorithms in Group 1: (a) overall view; (b) and (c) zoomed parts for yellow regions in (a). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 3
Performance index values of algorithms in Group 1 (best scores in **bold**).

#	Algorithms	MAE	MSE	Poisson
1	Linear Reg.	**0,02822**	0,00121	0,00631
2	ElasticNet	0,25808	0,08476	0,14514
3	SGD	0,03419	0,00170	0,00846
4	**Bayesian Ridge**	**0,02822**	**0,00120**	0,00631
5	Lasso	0,25808	0,08476	0,14514
6	RANSAC	0,02816	0,00120	**0,00629**

perform better than SGD since their median value is almost at the 0 % line while the median value of SGD is slightly far from the reference line. Other properties of these three algorithms show similarities, but considering the other two criteria, it might be said that Bayesian Ridge is the best algorithm of this group.

As mentioned in Section 2.1, Group 2 algorithms are methods based on the ensemble learning technique. To compare the algorithms in this Group, the same analysis performed for Group 1 was also carried out for Group 2. First, all algorithms in the group were trained using experimental data to create the model. These models were later tested using the test data. The comparative graph for SoC estimation is as in Fig. 4. When Fig. 4 (a) is examined, it is seen that all algorithms generally follow the reference line successfully. Superior results are seen for all algorithms of Group 2 except the AdaBoost method. The most important reason why the Adaboost method is less successful than the other methods in Group 2 is thought to be related to the sensitivity of the algorithm to outliers. Table 4 also lists the different performance index values of algorithms. The bagging algorithm comes to prominence, having the lowest MAE and Poisson values, while XGBoost is slightly closer to it. Furthermore, this Table also shows how the Gradient Boosting method is worse than the best ones Bagging and XGBoost, while it cannot be easily observed in Fig. 4.

SoC error changes were computed as in Group 1 to evaluate the results statistically. Fig. 5 illustrates SoC error changes of all algorithms for Group 2 in a violin plot. When the error distributions in Fig. 5 are examined, it is seen that the error changes of Adaboost and Gradient Boosting methods are almost two times higher than the other algorithms. The findings are directly in line with previous findings in Table 4. It is seen that the other algorithms in Group 2 have error changes in a very narrow range, and the wide region of the distribution line, which refers to the higher probability areas, concentrates around the reference error line. Moreover, the median values of all these algorithms are nearly on the reference 0 % line. Taking into account the 1.5 IQR of all these four best methods of Group 2, it is obviously seen that the range length is very short, and the Bagging algorithm has the minimum one.

When SoC estimation, error change graphs and Table 4 performance value comparisons are analyzed, it can be said that four algorithms produce good results in Group 2. However, the Bagging method produces the best results among Group 2 ensemble learning algorithms, albeit with a slight difference.

Fig. 6 shows the SoC estimation results for the other algorithms in Group 3. When the SoC estimation results are analyzed, it is seen that all algorithms except the SVR method successfully track the reference value. This is also seen in the magnified graphs of rotating parts where the state transitions are abrupt. The SVR method predicted approximately 5–8 % more inaccurate results than the reference values at these points.

A more detailed statistical analysis of the Group 3 algorithms is given in Fig. 7. As seen in the previous graphs and Fig. 7, the SVR method's error margin is ±10 %. On the other hand, the prediction errors of the other algorithms are in a very narrow range, and the error margins are small. In particular, the results of Random Forest and ExtraTree algorithms are in the range of ±3 %, which is very close to the true value. Furthermore, 25–75 % of the data is very close to the reference point, and the median value is almost at the reference point. Similarly, the 1.5IQR margins of errors are represented in a very narrow range which implies how robust the predictions are.

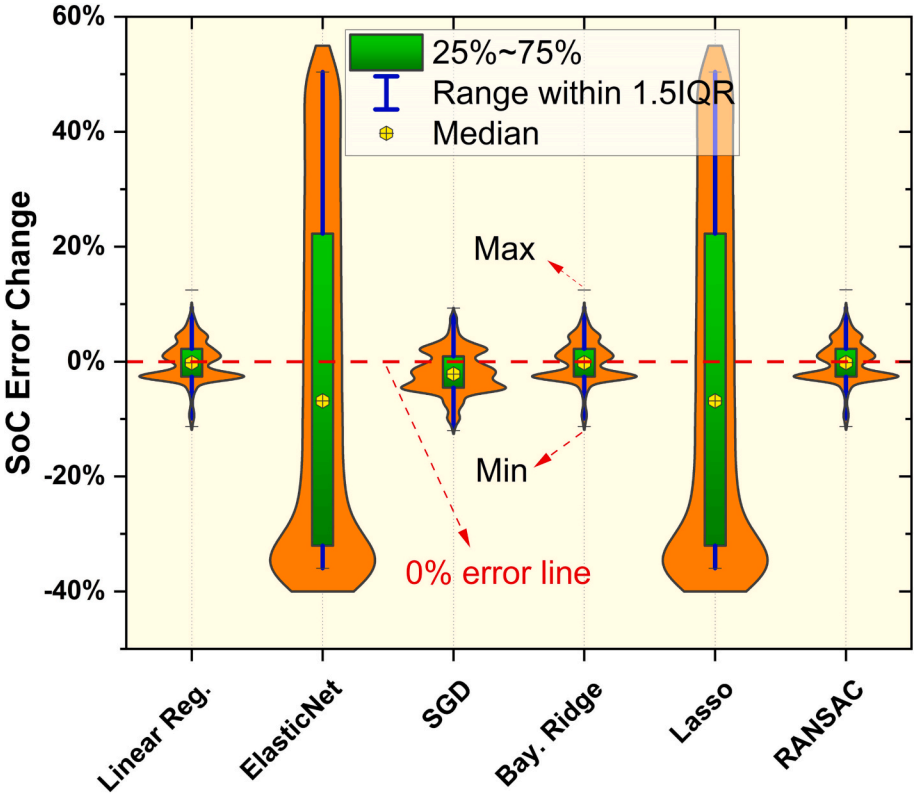


Fig. 3. Group 1 algorithms SoC estimation error changes in a violin plot.

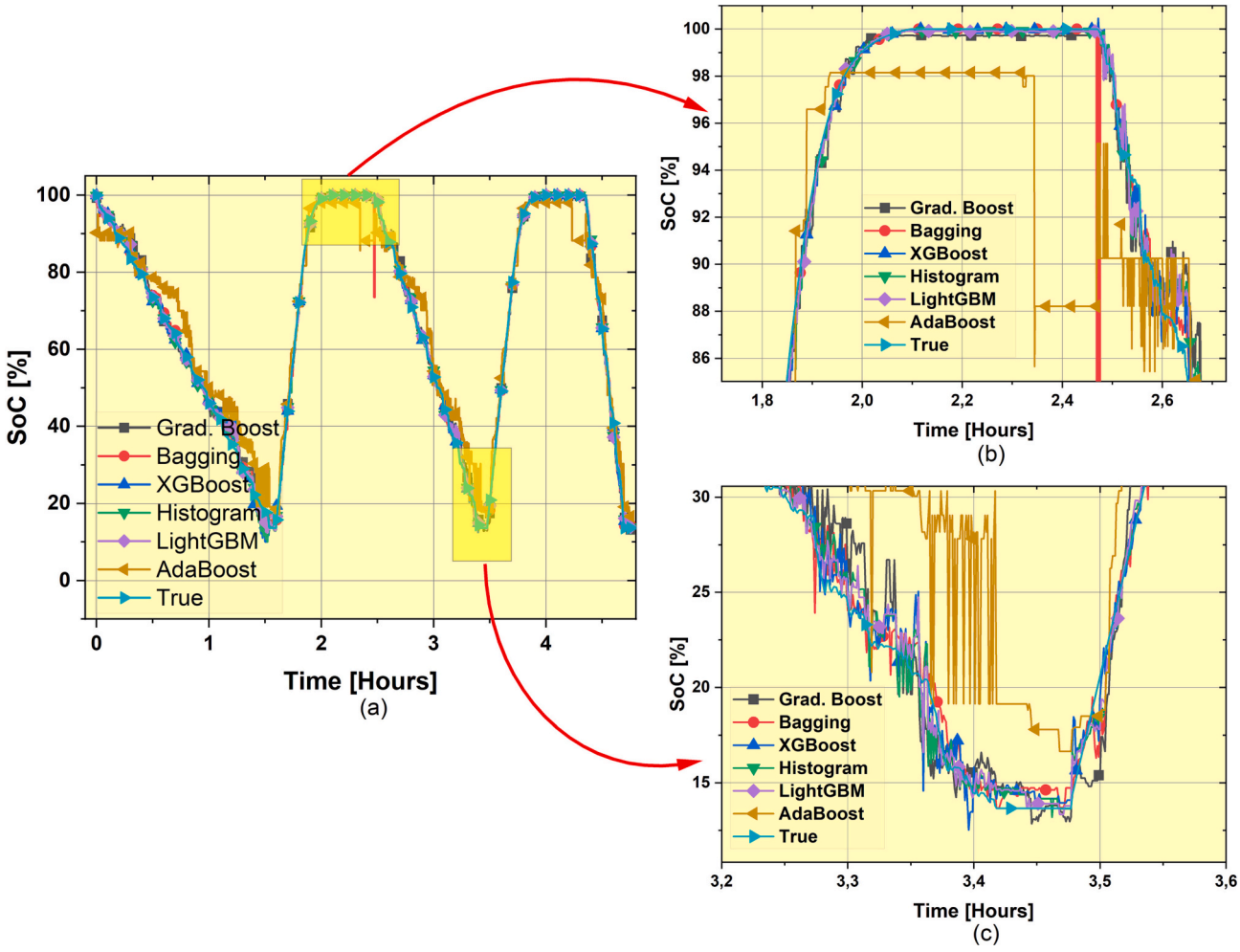


Fig. 4. SoC estimation results of all algorithms in Group 2: (a) overall view; (b) and (c) zoomed parts for yellow regions in (a). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 4
Performance index values of algorithms in Group 2 (best scores in **bold**).

#	Algorithms	MAE	MSE	Poisson
7	Gradient Boosting	0,01303	0,00036	0,00096
8	**Bagging**	**0,00658**	0,00018	**0,00038**
9	XGBoost	0,00802	**0,00017**	0,00058
10	Histogram	0,00888	0,00019	0,00064
11	LightGBM	0,00887	0,00018	0,00050
12	AdaBoost	0,04684	0,00294	0,00661

SoC estimation results and error change analyses show that four algorithms in Group 2, except SVR and ANN, have closer predictions to the reference curve. However, the performance indices in Table 5 clearly prove the efficiency of the ExtraTree method, having at least 15 % lower MAE and 2 and 1.5 times better MSE and Poisson values than the other five algorithms.

When all these results are analyzed, it is seen that the algorithms within the groups make SoC estimations that are very close to the correct value, irrelevant or incorrect. On the other hand, even some methods that make better estimations are affected by the state transition of data, resulting in outliers in some cases (See Fig. 4 (b), 6 (b)). In order to further improve these results produced by ML methods, the algorithms that give the best results among the groups are selected, and the prediction values are subjected to a filtering process. Their results are analyzed in the next section.

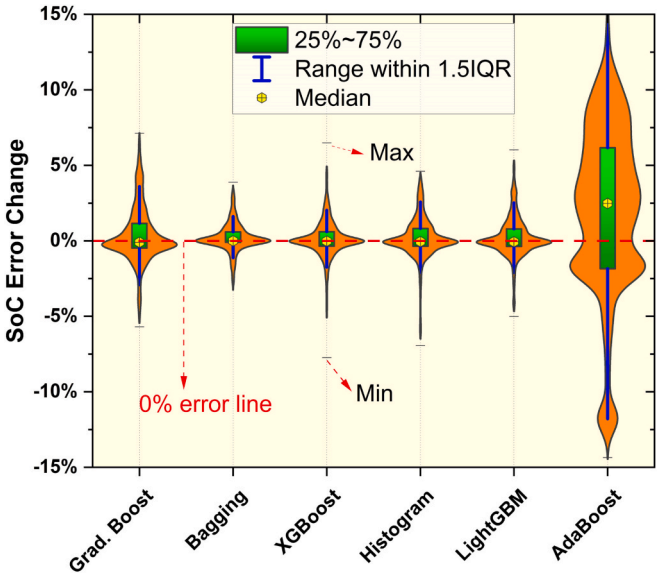


Fig. 5. Group 2 algorithms SoC estimation error changes in a violin plot.

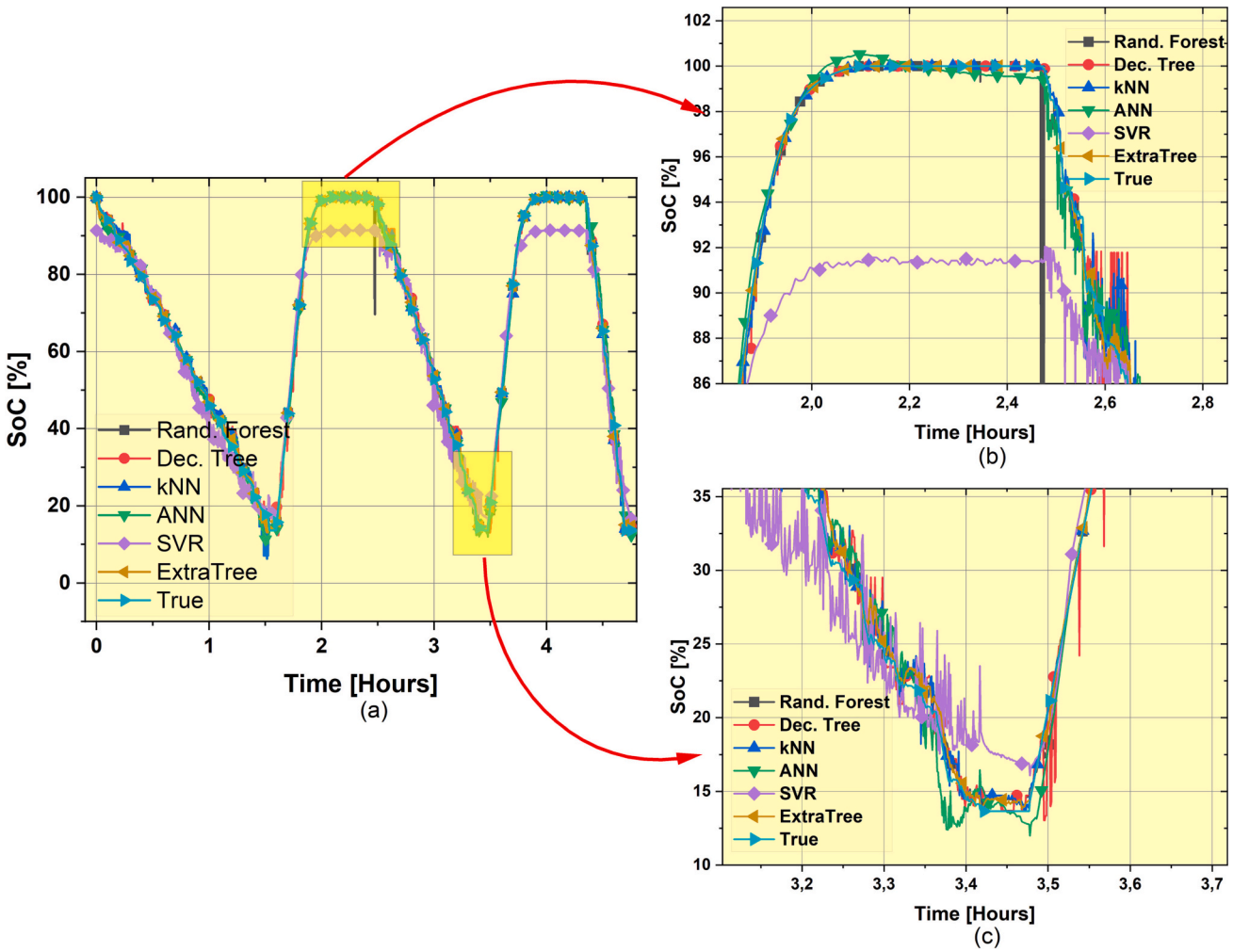


Fig. 6. SoC estimation results of all algorithms in Group 3: (a) overall view; (b) and (c) zoomed parts for yellow regions in (a). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3.3. Filter effect

Machine Learning algorithms are trained on input and output data to create a prediction model. This approach, also called black box modeling, offers significant advantages over traditional mathematical modeling or classical Kalman filter-based structures in state estimation of systems with complex models or complex nonlinear structures. However, in some cases, since these models do not rely on the observer, they may produce values that are irrelevant to the previous or subsequent states of the system [44]. This is one of the most important disadvantages of black box modeling. In the literature, this is investigated as a separate research topic called Explainable AI (XAI), or Explainable Machine Learning (XML), which humans can understand the decisions by the AI [45] . Nevertheless, in systems with signal-like predictions and whose results are predicted by machine learning methods, it is common to use filters that remove outlier or meaningless data from the predictions [29,46]. Thanks to these filters, some outlier data that ML algorithms may generate can be analyzed and eliminated from the model prediction values. As seen in the previous sections, there are algorithms in all groups that follow the SoC estimation well or produce irrelevant results. Even in some algorithms that make good predictions for SoC, there are parts where outlier data occurs (See Fig. 4 (b), 6 (b).). In order to enhance system performance and eliminate these outlier data, different filters were used in this study and their comparative analysis was carried out.

To apply the filters and compare the results, ML methods were divided into three groups in terms of working principle and model, and the models that gave the best results within the groups were selected. In this context, the Bayesian Ridge algorithm was selected from Group 1, the Bagging algorithm from Group 2, and the ExtraTree algorithm from Group 3. Gaussian, Sgolay, Rloess, Rlowess and median filters are chosen, and two different coefficients are considered to see the effect of the filter parameter (*P1*, *P2*). Since it is difficult to distinguish the difference between algorithms in the SoC estimation graph, the results are compared based on performance indices.

In Figs. 8, 9, and 10, the first part contains the values of the best algorithm of Group 1, Bayesian Ridge; the second part represents the values of the best algorithm of Group 2, Bagging; and the third part point outs the values of the best algorithm of Group 3, ExtraTree (purple blocks separated by thick solid black lines). Y-axis refers to the performance index value of the method, and X-axis in cyan represents the name of the filters. N/A refers to the values obtained without any filtering and has the same value as those found in Section 3.2. These values are also drawn in multiple horizontal lines to show the effectiveness of the filters. The red dashed line along the X-axis indicates the value of Bayesian Ridge, the best algorithm in Group 1; the green dash-dotted line refers to the value of Bagging, the best algorithm in Group 2; and the brown dotted line indicates the value of ExtraTree, the best algorithm in Group 3. The yellow and dark blue bars are performance index values of different filter coefficients, respectively.

Considering all the performance indices in Figs. 8 to 10, it is obviously seen that Group 2 and 3 best algorithms have better results than

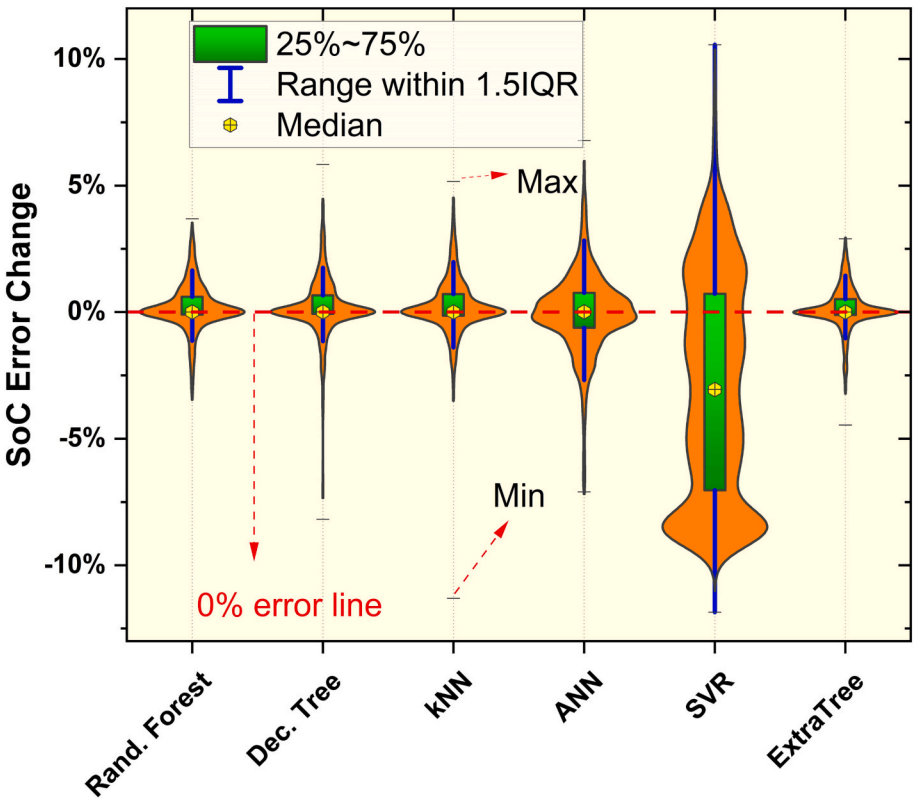


Fig. 7. Group 3 algorithms SoC estimation error changes in a violin plot.

Table 5
Performance index values of algorithms in Group 3 (best scores in **bold**).

#	Algorithms	MAE	MSE	Poisson
13	Random Forest	0,00639	0,00017	0,00033
14	Decision Tree	0,00727	0,00016	0,00044
15	kNN	0,00748	0,00017	0,00075
16	ANN	0,01105	0,00025	0,00089
17	SVR	0,04345	0,00273	0,00447
18	**ExtraTree**	**0,00554**	**0,00008**	**0,00022**

Group 1 algorithm. This situation has not changed whether a filter was applied or not. For example, the MAE values of the Bagging and Extra-Tree algorithms are almost five times lower than the Bayesian Ridge values for the case without filters. The same results are consistent when the filters are applied, and these algorithms are again almost five times better than Bayesian Ridge. The parallel results are also observed for other performance indices. For instance, without any filter, the MSE results of the algorithms belonging to the Groups are *1,20E-3*, *1,88E-4*, and *8,52E-5*, respectively. Even in the worst case, the Group 2 and 3 algorithms outperform Group 1's best algorithm by eight times. As stated above, the Poisson performance index is a good indicator for regression problems. When Poisson performance index values of the algorithms are analyzed, the difference between groups reaches about *20* times. On the other hand, it can be understood from the figures that the results of the Group 2 and 3 best algorithms are almost very close to each other for different parameters. A closer look to each group reveals that filtered values are generally better than the unfiltered ones in terms of the performance index value, and filtered values are also under the reference line for both different parameters. For example, in terms of the MAE index, for Group 1 almost five different filter results with two different filter coefficients are better than the situation without a filter, and the best performance is achieved by the Gauss filter with a parameter coefficient *P1*. The Rloess filter with a parameter *P1* is also the best

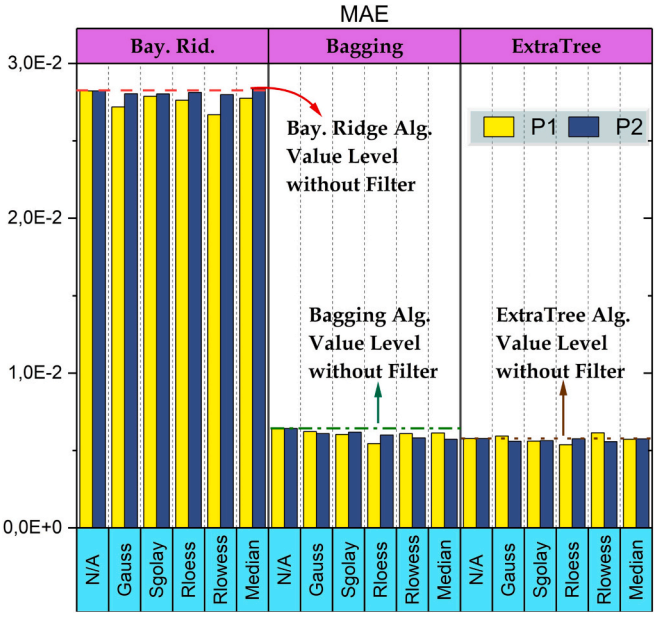


Fig. 8. MAE results of all best algorithms for two different filter parameters.

one for Group 2 and 3. Considering the MSE index, the Rlowess filter has the minimum value for Group 1, and Rloess filter has the lowest value for Group 2 and 3. For the Poisson criterion, the minimum values are obtained with the Gauss filter for Group 1 and the Rloess filter for Group 2 and 3.

The filter coefficients *P1* and *P2* are 51 and 11, respectively. Almost in all cases, when the filter parameter is increased, the performance gets better, up to 10 %. This is an expected result since it also increases the filter order. The upper limit of the maximum value of the filter

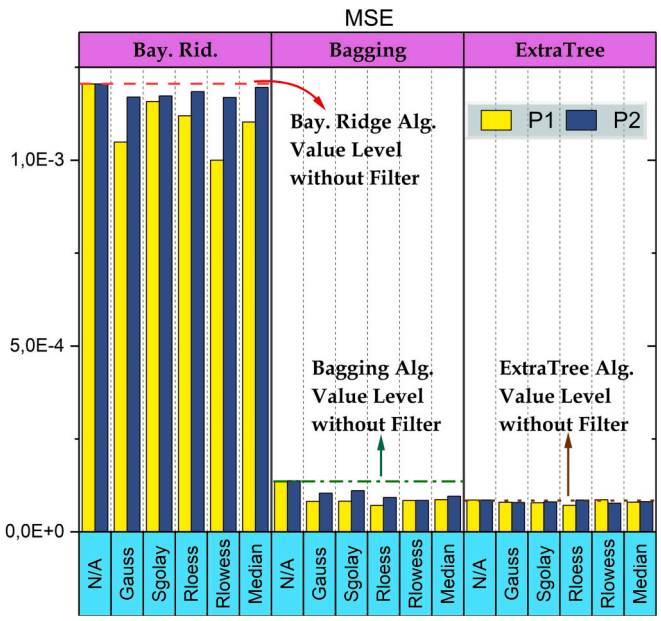


Fig. 9. MSE results of all best algorithms for two different filter parameters.

coefficient was determined by experimentally controlling the filter result until it reached the unchanged level.

The use of filters provides quite an improvement in performance index values. On the other hand, the most significant advantage of filters is eliminating outlier values. As mentioned earlier, ML algorithms make predictions for each set of inputs and the previous or subsequent outputs of the system do not directly affect the prediction results. This is due to the black box modeling of ML algorithms, and in some cases, it can lead to the generation of outlier, or nonsense data that can affect the system's overall performance. Therefore, filters are used to correct the system outputs. Fig. 11 shows the overall SoC prediction graphs for the Bagging algorithm without filters and with different filters and some enlarged parts. We have chosen two regions to show the filter's effect on the SoC estimation curve. First, Fig. 11 (b) shows the effect of the filters for the initial state and monotonically decreasing values of SoC. Second, Fig. 11 (c) illustrates the effect of the filters on sudden jumps in SoC. In Fig. 11 (b) and (c) X-axis represent the time interval, Y-axis stands for the filter name, and *Z*-axis is the percentage SoC value. The names of the filters are Median, Rlowess, Rloess, Sgolay, Gaussian, and unfiltered from frontmost to backmost. Both enlarged regions clearly show that the raw SoC estimation values fluctuate with time for the unfiltered case. After applying the filters, the SoC curves become smooth. This does not make a significant difference for Fig. 11 (b), where the curve changes monotonically, but not for Fig. 11 (c), where the curve has sudden jumps, i.e., outlier data. These outliers were up to 20 % more inaccurate than average values, which can affect the system's overall performance. When filters were applied to remove these data, it was witnessed that all filters successfully removed outliers. However, the best values were obtained with the Rloess filter. These results are in line with the performance index comparison above.

3.4. Time evaluation

Training and prediction time criteria are important parameters in evaluating the effectiveness of ML algorithms. As the data size increases, it may take forever to train the model, or data may not fit in memory. Furthermore, slow prediction time may result in current system variables not being captured in time. To this aim, the time efficiency of ML algorithms used in this study was compared. Fig. 12 spider chart shows

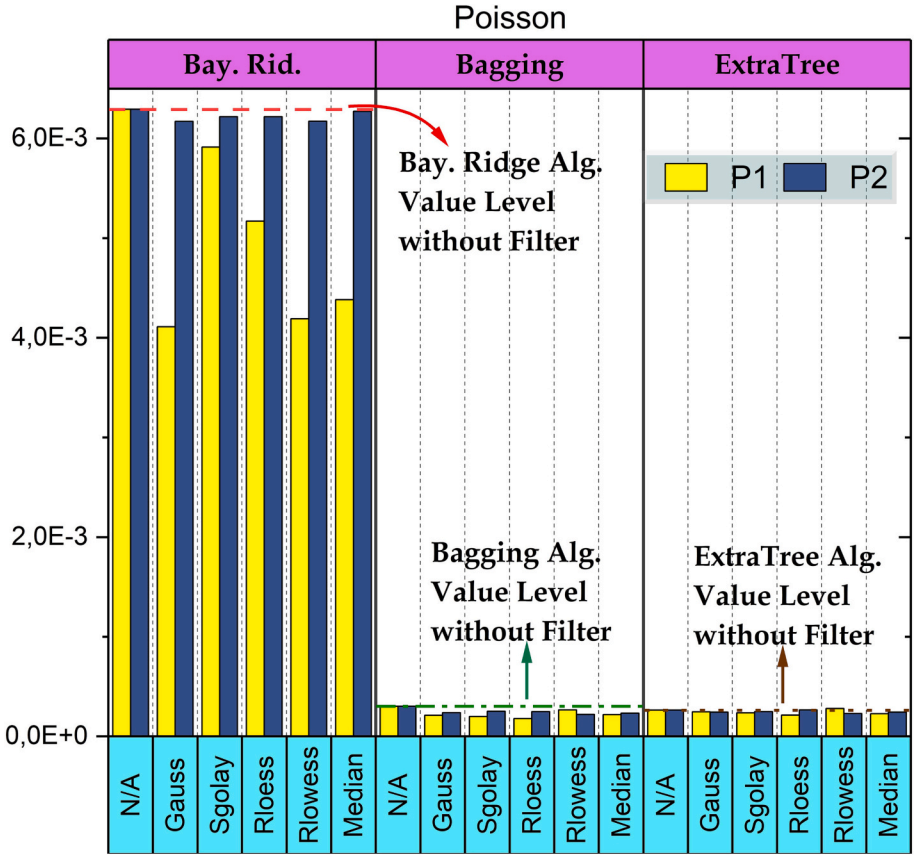


Fig. 10. Poisson results of all best algorithms for two different filter parameters.

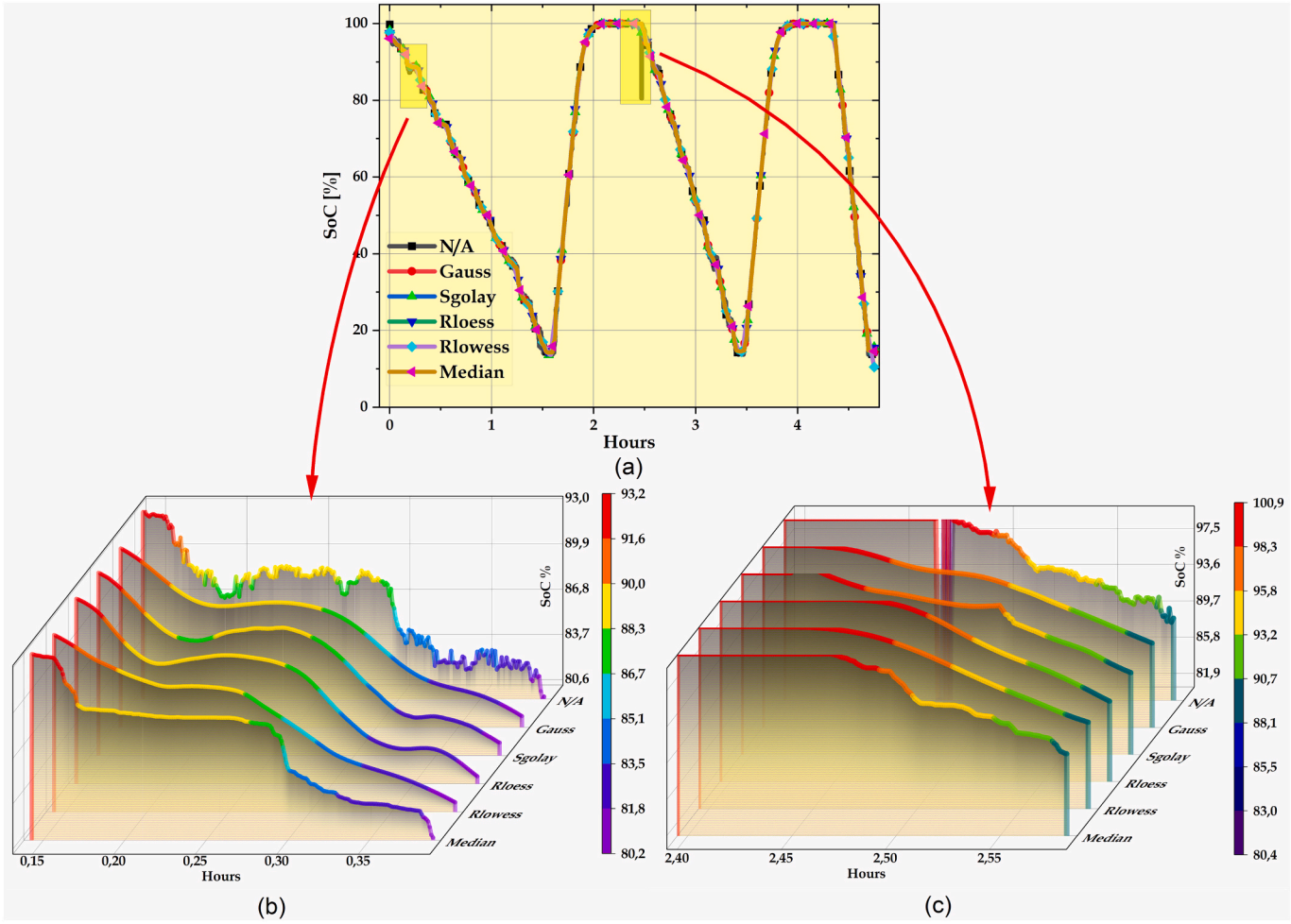


Fig. 11. SoC estimation results of Bagging algorithm for different filters: (a) Overall view; (b) and (c) zoomed parts for yellow regions in (a). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the training and prediction time values for different groups and algorithms. Since the values have an extensive range, a logarithmic scale is chosen to show all values compactly. Major axis intervals in this polar chart correspond to the power of *10*, meaning that moving a unit of distance along the scale multiplies the number by *10*. In addition, the farther towards the end of the spike, the larger the value and the closest to the center means closer to the minimum value. In Fig. 12, those in yellow indicate the algorithms in Group 1, those in red represent the

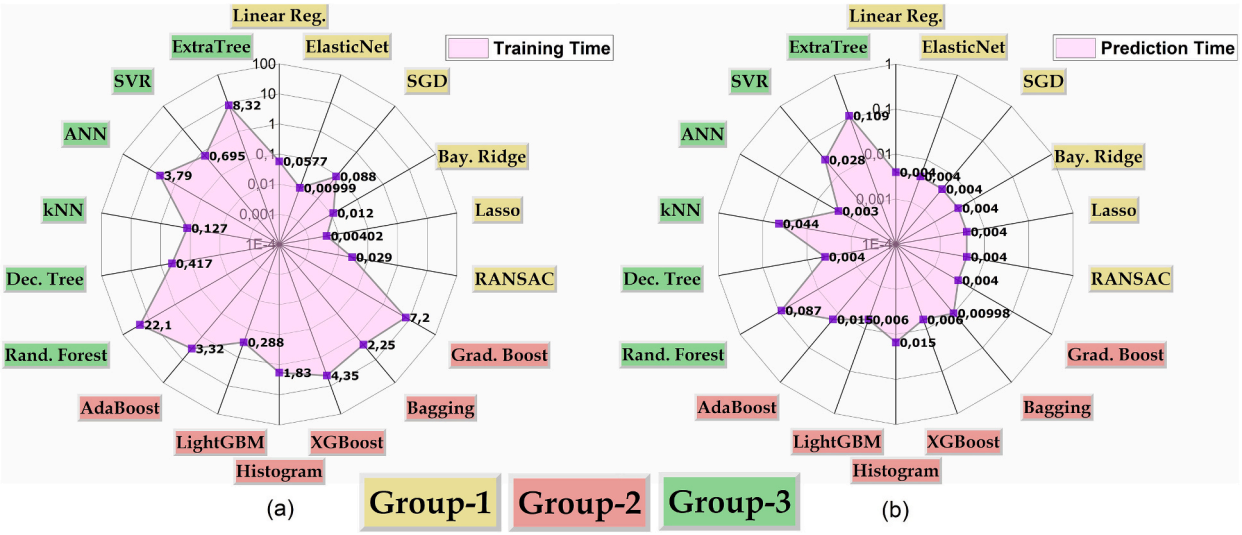


Fig. 12. Time values of machine learning algorithms in logarithmic scale (a) training process; (b) prediction process.

algorithms in Group 2, and those in green refer to the algorithms in Group 3. The end of spikes refers to the name of the relevant ML method, and the purple squares in Fig. 12 represent the time values of algorithms in logarithmic scale, while real values are given in Table 6.

The evaluation between the groups demonstrates that Group 1 methods show the best performance in training and prediction time, while Group 2 algorithms show the worst time performance. This significant difference between groups arises from the fact that Group 1 is Linear, and Group 2 is ensemble-based. All algorithms in Group 1 have the same or very close values in prediction time. On the other hand, the lowest training time in Group 1 was obtained with the Lasso method, followed by ElasticNet and Bayesian Ridge algorithms.

Among the Group 2 algorithms, the LightGBM algorithm has the lowest training time. This algorithm is followed by Histogram and Bagging algorithms. The training time of the LightGBM algorithm with the lowest time in group 2 is even higher than the algorithm with the worst training time in Group 1. The lowest prediction time is obtained with gradient boost and lightGBM, while all algorithms in Group 2 have the closest prediction time values.

Since Group 3 algorithms include different techniques, the time difference between the algorithms is more noticeable. The lowest training time in this group is obtained with the Decision Tree algorithm, which is quite high compared to the Group 1 algorithms and half of the Group 2 algorithms. The random forest algorithm in this group has the worst training time among all algorithms. This is followed by the ExtraTree algorithm with the second worst time in this group.

3.5. Effect of sampling rate

The general idea in training machine learning algorithms is to use more data, as it often leads to building better models. More data leads to a better understanding of the model by the algorithm. A typical battery charge/discharge regime usually takes hours, and the data collected from this test can reach up to massive sizes if the sampling time is too frequent. It is obvious that this will bring difficulties in data collection and processing. For this purpose, an investigation was carried out to see the effect of sampling frequency. Accordingly, the effectiveness of ML algorithms for sampling rates of *1*, *10* and *100* Hz was analyzed. Fig. 13 illustrates the performance of the algorithms giving the best values in three groups for three sampling rates. Since the calculations obtained from the performance indices are small and have a wide range, logarithmic values are chosen to illustrate the results better ($-logP$). P here represents the MAE, MSE or Poisson performance index values of the algorithms. The true values of these calculations are shown inside the boxes, while their heights refer to the logarithmic values. In Fig. 13, yellow values show the performance index values of the algorithms for *1* Hz, pink values for *10* Hz and purple values for *100* Hz in logarithmic scale. When the values and graph are analyzed, for example, for the best algorithm of Group 1, Bayesian Ridge, the values are *0.0283* at *1* Hz, *0.0282* at *10* Hz, and *0.0283* at *100* Hz. It was observed that this was similar for the other algorithms and sampling rates and reducing the number of samples for training ML algorithms did not affect the results much. This shows that the change in battery values during charging or discharging is tiny compared to the sampling time, so training can be successfully performed even with small sample values. On the other hand, when the time efficiency is examined, for example, for the Bagging algorithm, the training times were *29.059 s* at *1* Hz, *2.246 s* at *10* Hz and *0.244 s* at *100* Hz.

Table 6
Real values of training and prediction process of algorithms.

Groups	#	Algorithms	Training time	Prediction time
Group 1 (Linear Models)	1	Linear Regresion	0,058	0,004
	2	ElasticNet	0,010	0,004
	3	SGD	0,088	0,004
	4	Bayesian Ridge	0,012	0,004
	5	Lasso	0,004	0,004
	6	RANSAC	0,029	0,004
Group 2 (Ensemble Models)	7	Gradient Boosting	7,195	0,004
	8	Bagging	2,246	0,010
	9	XGBoost	4,349	0,006
	10	Histogram	1,827	0,015
	11	LightGBM	0,288	0,006
	12	AdaBoost	3,320	0,015
Group 3 (Other Models)	13	Random Forest	22,102	0,087
	14	Decision Tree	0,417	0,004
	15	kNN	0,127	0,044
	16	ANN	3,789	0,003
	17	SVR	0,6954	0,02799
	18	ExtraTree	8,31816	0,10905

3.6. Overall evaluation & discussion

Detailed analysis and comparison of different ML methods for SoC estimation of batteries have been performed. In this context, algorithms were divided into different groups and comparisons were made between and within groups. The comparison was evaluated from four different perspectives. These criteria are the match of SoC prediction curves with the true value, training and prediction time evaluation, performance index values and statistical analysis. Table 7 summarizes the scores for all these assessments. The number of stars in the table shows the success of the algorithms according to the relevant evaluation criterion. The higher the number of stars, the better the performance on the criterion. A red cross indicates that the method scored poorly on this criterion and produced unacceptable results.

Considering the SoC curve match, it is seen that algorithms follow the true values except for the ElasticNet, Lasso, and SVR algorithms, and some made it with high accuracy. Group 1 and some Group 3 algorithms have the highest scores for the time evaluation while Group 2 algorithms perform poorly. From the perspective of MAE, MSE, and Poisson criteria, some algorithms of Group 2 and 3 have superior performance, while none of the Group 1 algorithms give acceptable results. The number of stars for the last two performance criteria was calculated based on the normalization of the true values of algorithms. In addition, statistically, 1.5IQR margins have been taken into account to evaluate the algorithms. The sum of absolute values in both the positive and negative direction greater than *20* is considered unacceptable, while less than three is considered three stars.

With battery data obtained from numerous driving cycles, different situations to which the battery can be exposed can be represented. ML models can therefore be trained offline. In this case, the training time is ignored. However, prediction time is directly related to the sampling rate and can only be ignored when it is lower than the sampling rate. A comparison of the sampling rate and time values described in Sections 3.4 and 3.5 proves that the prediction time is still lower than the sampling rate, even when it has the highest value at *1* Hz. According to these explanations, the overall score can be generated by the other three criteria. To generate the overall score, an algorithm with three stars in all criteria is rated as the best. In contrast, an algorithm with a red cross in any criterion is considered unacceptable for SoC estimation. Therefore, any algorithms in Group 1, Adaboost in Group 2, and SVR in Group 3 are considered unacceptable for SoC estimation since they perform poorly in one or more criteria. From the results in Table 7, it is clear that the Bagging algorithm in Group 2 and the ExtraTree algorithm in Group 3 are the most suitable algorithms for SoC estimation of battery.

As stated in Section 3.3, using filters in ML-based prediction problems can decrease the fluctuations in the estimation curve and eliminate the outlier data. This situation was clearly observed in both the estimation graphs and performance indices. For almost all conditions, all filters outperformed the unfiltered cases. Especially Rloess filter had the best performance index values and curve matching. This might have arisen from the fact that this filter utilizes quadratic polynomials and

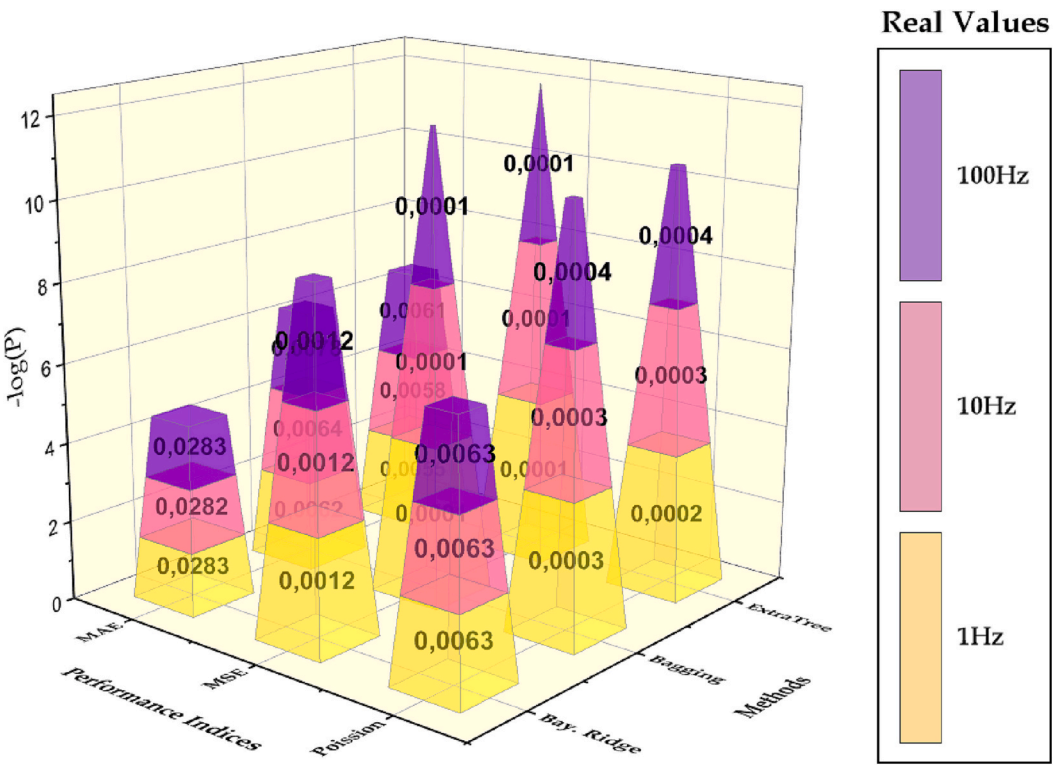


Fig. 13. Effect of sampling rate for different performance indices.

Table 7
Overall scores of algorithms.

Groups	#	*Algorithms*	*SoC Curve*	*Time*	*Performance Criteria*	*Statistical*	*Overall Score*
Group 1 (Linear Models)	1	Linear Reg.	★★	★★★	★	✗	👎
	2	ElasticNet	✗	★★★	✗	✗	👎
	3	SGD	★	★★★	★	✗	👎
	4	Bay. Ridge	★★	★★★	★	✗	👎
	5	Lasso	✗	★★★	✗	✗	👎
	6	RANSAC	★★	★★★	★	✗	👎
Group 2 (Ensemble Models)	7	Grad. Boost	★★★	★	★★	★	👍👍
	8	Bagging	★★★	★	★★★	★★★	👍👍👍
	9	XGBoost	★★★	★	★★★	★★	👍👍
	10	Histogram	★★★	★	★★★	★★	👍👍
	11	LightGBM	★★★	★★	★★★	★★	👍👍
	12	AdaBoost	★★	★	★	✗	👎
Group 3 (Other Models)	13	Rand. Forest	★★★	★	★★★	★★	👍👍
	14	Dec. Tree	★★★	★★★	★★★	★★	👍👍
	15	kNN	★★★	★★★	★★★	★★	👍👍
	16	ANN	★★★	★★	★★	★	👍
	17	SVR	✗	★★★	★	✗	👎
	18	ExtraTree	★★★	★	★★★	★★★	👍👍👍

robust weight functions. On the other hand, Sgolay and Median filters were the least successful ones since their performance index values were usually worse than the others. Furthermore, Median filter estimations had some fluctuations along the whole prediction process, while Sgolay still had some sudden jumps where the state had an abrupt change. This seems to depend on unsuitable median points for the Median filter and not fitted polynomials for the Sgolay filter.

The main limitation of the study is the lack of a comparison with deep learning (DL) methods. The reason why we did not compare the ML algorithms with DL is twofold. First, DL requires large amounts of labeled data. The data we use in the manuscript at hand would not be enough to build DL models, and therefore it may result in poor performance. Secondly, although it is a subset of ML, DL is usually referred to as deep neural networks since it is an enhanced version of the neural networks. However, it differs considerably from ANN in that it requires large amounts of data and has numerous hidden layers. Due to these properties of DL, it would not be fair to compare it with the ML algorithms used in this study. Instead, it should be considered as a separate

study addressing the comparison of different DL structures. Another limitation of our implementation when training ML methods is not to use filtered values as input variables of ML algorithms. Such a practice may contribute to the performance of the algorithms. However, we overcame any performance degradation problem by using the normalized data when training the algorithms, and the results obviously show that algorithms, especially two of them, have superior performance.

4. Conclusion

The accuracy of SoC estimation is a key parameter for the safe and comfortable drive of xEVs. In pursuit of this, many methods, from perfect modeling to best filters, have been offered in the literature. On the other hand, data-driven methods have recently become a good alternative choice for SoC estimation since they can be characterized by input/output data relationships. However, studies considering the ML algorithms for SoC estimation have addressed the problem only for a few methods and the effects of filters have rarely been studied directly. We, therefore, analyzed 18 different ML methods in three main groups and investigated their efficiency according to four different criteria: training and prediction time comparisons, SoC estimation curve matching, statistical evaluation, and performance indices. The results confirm that ensemble algorithms are a good choice for SoC estimation. Among all methods, superior results were especially seen for Bagging and Extra-Tree algorithms. Statistically, the 1.5IQR of these algorithms was below 3 %, and performance index values were also at least 16 % better than the other algorithms. Moreover, their SoC estimation curve matched the reference line best. To examine filter impact on SoC estimation, we also applied five different filters. Slightly superior results were achieved with the filters. They removed the outlier data from the prediction curve and smoothed the ordinary subsequent values. Although there was no remarkable change among the filters, the Rloess method was nearly found to be the best one in removing the outliers and increasing the performance index values. The results provide a good starting point for discussion and further research. For instance, findings from the study can be applied to the different batteries to see the effect of ML and filters. Furthermore, it is another question of future research to investigate the comparison of deep learning methods.

CRediT authorship contribution statement

Mehmet KORKMAZ: Conceptualization, Methodology, Data Curation Writing – original draft and review & editing, Visualization, Investigation.

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.

Data availability

Data will be made available on request.

References

[1] C. Zhao, et al., Is smart transportation associated with reduced carbon emissions? The case of China, Energy Econ. 105 (2022), 105715.
[2] Z. Zhongming, et al., The Emissions Gap Report, 2021, p. 2021.
[3] E. Chemali, et al., State-of-charge estimation of Li-ion batteries using deep neural networks: a machine learning approach, J. Power Sources 400 (2018) 242–255.
[4] G.O. Sahinoglu, et al., Battery state-of-charge estimation based on regular/recurrent Gaussian process regression, IEEE Trans. Ind. Electron. 65 (5) (2018) 4311–4321.
[5] Y.H. Li, et al., A hybrid machine learning framework for joint SOC and SOH estimation of lithium-ion batteries assisted with fiber sensor measurements, Appl. Energy 325 (2022).
[6] H.X. Tian, A. Li, X.Y. Li, SOC estimation of lithium-ion batteries for electric vehicles based on multimode ensemble SVR, J. Power Electron. 21 (9) (2021) 1365–1373.
[7] W. He, et al., A physics-based electrochemical model for lithium-ion battery state-of-charge estimation solved by an optimised projection-based method and moving-window filtering, Energies 11 (8) (2018) 2120.
[8] B. Yang, et al., Classification, summarization and perspectives on state-of-charge estimation of lithium-ion batteries used in electric vehicles: a critical comprehensive survey, J. Energy Storage (2021) 39.
[9] X. Lai, Y.J. Zheng, T. Sun, A comparative study of different equivalent circuit models for estimating state-of-charge of lithium-ion batteries, Electrochim. Acta 259 (2018) 566–577.
[10] L.X. Wu, et al., Online SOC estimation based on simplified electrochemical model for Lithium-ion batteries considering current bias, Energies 14 (17) (2021).
[11] Y.T. Liu, et al., A nonlinear observer SOC estimation method based on electrochemical model for lithium-ion battery, IEEE Trans. Ind. Appl. 57 (1) (2021) 1094–1104.
[12] S.S. Madani, E. Schaltz, S.K. Kaer, A review of different electric equivalent circuit models and parameter identification methods of lithium-ion batteries, in: 19th International Conference on Advanced Batteries, Accumulators and Fuel Cells (Abaf 2018) 87(1), 2018, pp. 23–37.
[13] M. Adaikkappan, N. Sathiyamoorthy, Modeling, state of charge estimation, and charging of lithium-ion battery in electric vehicle: a review, Int. J. Energy Res. 46 (3) (2022) 2141–2165.
[14] A. Rahmoun, H. Biechl, Modelling of Li-ion batteries using equivalent circuit diagrams, Przeglad Elektrotechniczny 88 (7b) (2012) 152–156.
[15] X. Ding, et al., An improved Thevenin model of lithium-ion battery with high accuracy for electric vehicles, Appl. Energy 254 (2019), 113615.
[16] K. Wang, et al., State of Charge (SOC) estimation of lithium-ion battery based on adaptive square root unscented Kalman filter, Int. J. Electrochem. Sci. 15 (9) (2020) 9499–9516.
[17] P. Shrivastava, et al., Overview of model-based online state-of-charge estimation using Kalman filter family for lithium-ion batteries, Renew. Sust. Energ. Rev. 113 (2019).
[18] C.H. Liu, et al., State of power estimation of lithium-ion battery based on fractional-order equivalent circuit model, J. Energy Storage (2021) 41.
[19] J.P. Tian, et al., Online simultaneous identification of parameters and order of a fractional order battery model, J. Clean. Prod. 247 (2020).
[20] C.F. Zou, et al., A review of fractional-order techniques applied to lithium-ion batteries, lead-acid batteries, and supercapacitors, J. Power Sources 390 (2018) 286–296.
[21] Q.H. Liu, Q.Q. Yu, The lithium battery SOC estimation on square root unscented Kalman filter, Energy Rep. 8 (2022) 286–294.
[22] J.X. Chen, et al., State of charge estimation for lithium-ion batteries using gated recurrent unit recurrent neural network and adaptive Kalman filter, J. Energy Storage (2022) 55.
[23] Q. Zhu, et al., A state of charge estimation method for lithium-ion batteries based on fractional order adaptive extended kalman filter, Energy 187 (2019).
[24] I.B. Espedal, et al., Current trends for State-of-Charge (SoC) estimation in lithium-ion battery electric vehicles, Energies 14 (11) (2021).
[25] H. Ben Sassi, et al., Comparative study of ANN/KF for on-board SOC estimation for vehicular applications, J. Energy Storage (2019) 25.
[26] M.F. Ng, et al., Predicting the state of charge and health of batteries using data-driven machine learning, Nat. Mach. Intell. 2 (3) (2020) 161–170.
[27] I. Akhil, et al., State of charge estimation using different machine learning techniques, J. Inf. Optim. Sci. 43 (3) (2022) 543–547.
[28] H. Li, X. Liu, Q. Mei, Predicting smartphone battery life based on comprehensive and real-time usage data, in: arXiv Preprint arXiv:1801.04069, 2018.
[29] M.S. Sidhu, D. Ronanki, S. Williamson, State of Charge Estimation of Lithium-ion Batteries using Hybrid Machine Learning Technique. 45th Annual Conference of the Ieee Industrial Electronics Society (Iecon 2019), 2019, pp. 2732–2737.
[30] V. Chandran, et al., State of charge estimation of lithium-ion battery for electric vehicles using machine learning algorithms, World Electr. Veh. J. 12 (1) (2021) 38.
[31] E. Ipek, M.K. Eren, M. Yilmaz, State-of-charge estimation of li-ion battery cell using support vector regression and gradient boosting techniques, in: 2019 International Aegean Conference on Electrical Machines and Power Electronics (Acemp) & 2019 International Conference on Optimization of Electrical and Electronic Equipment (Optim), 2019, pp. 604–609.
[32] W. He, et al., State of charge estimation for Li-ion batteries using neural network modeling and unscented Kalman filter-based error cancellation, Int. J. Electr. Power Energy Syst. 62 (2014) 783–791.
[33] J.H. Meng, G.Z. Luo, F. Gao, Lithium polymer battery state-of-charge estimation based on adaptive unscented Kalman filter and support vector machine, IEEE Trans. Power Electron. 31 (3) (2016) 2226–2238.
[34] Z.C. Xu, et al., Improving the state of charge estimation of reused lithium-ion batteries by abating hysteresis using machine learning technique, J. Energy Storage (2020) 32.
[35] M. Jiao, D. Wang, The Savitzky-Golay filter based bidirectional long short-term memory network for SOC estimation, Int. J. Energy Res. 45 (13) (2021) 19467–19480.
[36] H. Li, et al., A novel state of charge estimation method of lithium-ion batteries based on the IWOA-AdaBoost-Elman algorithm, Int. J. Energy Res. 46 (4) (2022) 5134–5151.
[37] M. Ragone, et al., Data driven estimation of electric vehicle battery state-of-charge informed by automotive simulations and multi-physics modeling, J. Power Sources 483 (2021).

[38] C. Lopez-Molina, et al., Multiscale edge detection based on Gaussian smoothing and edge tracking, Knowl.-Based Syst. 44 (2013) 101–111.
[39] J.W. Liu, et al., Remaining useful life prediction of PEMFC based on long short-term memory recurrent neural networks, Int. J. Hydrog. Energy 44 (11) (2019) 5470–5480.
[40] A. Nurunnabi, G. West, D. Belton, Robust locally weighted regression techniques for ground surface points filtering in mobile laser scanning three dimensional point cloud data, IEEE Trans. Geosci. Remote Sens. 54 (4) (2016) 2181–2193.
[41] B. Jorgensen, The Theory of Dispersion Models, CRC Press, 1997.
[42] C. Vidal, et al., Robust xev battery state-of-charge estimator design using a feedforward deep neural network, SAE Int. J. Adv. Curr. Pract. Mobil. 2 (2020-01–1181) (2020) 2872–2880.
[43] C. Vidal, et al., Li-ion battery state of charge estimation using long short-term memory recurrent neural network with transfer learning, in: 2019 IEEE Transportation Electrification Conference and Expo (ITEC), IEEE, 2019.
[44] T. Yehia, et al., Removing the outlier from the production data for the decline curve analysis of shale gas reservoirs: a comparative study using machine learning, ACS Omega 7 (36) (2022) 32046–32061.
[45] F. Xu, et al., Explainable AI: a brief survey on history, research areas, approaches and challenges, in: CCF International Conference on Natural Language Processing and Chinese Computing, Springer, 2019.
[46] M. Tranmer, M. Elliot, Multiple linear regression, Cathie Marsh Centre Census Surv. Res. (CCSR) 5 (5) (2008) 1–5.
[47] P. Kollmeyer, C. Vidal, M. Naguib, M. Skells, Lg 18650HG2 li-ion battery data and example deep neural network xEV SOC estimator script, Mendeley Data 3 (2020) 2020.