Journal of Computational and Applied Mathematics 433 (2023) 115305

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

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

Data-driven state-of-charge prediction of a storage cell using ABC/GBRT, ABC/MLP and LASSO machine learning techniques

J.C. Álvarez Antón [a], P.J. García-Nieto [b,*], E. García-Gonzalo [b], M. González Vega [a], C. Blanco Viejo [a]

[a] Department of Electrical Engineering, Electronics and Communication Systems, 33204 Gijón, Spain
[b] Department of Mathematics, Faculty of Sciences, University of Oviedo, 33007 Oviedo, Spain

ARTICLE INFO

Article history:
Received 5 May 2021
Received in revised form 26 January 2023

Keywords:
Lithium-ion battery state of charge (SOC)
Gradient boosting regression tree (GBRT)
Multilayer Perceptron (MLP)
Artificial Bee Colony (ABC)
Least Absolute Shrinkage and Selection Operator (LASSO)
Dynamic Stress Test (DST)

ABSTRACT

Electric vehicles (EVs) will be the dominant technology for the automobile industry due to efficiency and environmental reasons. Lithium-ion batteries lead the energy supply business for the most recent group of EVs and many other electronic consumer devices. One of the most important pieces of information for EV users is the state-of-charge of the battery, also known as SOC. The SOC works like the *fuel gauge* for the battery. Information about remaining battery capacity is essential to avoid running out of battery power. Battery remaining charge is not easy to estimate, due to non-linear phenomenon inside the battery. This work is concerned with SOC prediction using machine learning techniques. Three machine learning tools, called Artificial Bee Colony-Multilayer Perceptron (ABC/MLP), Artificial Bee Colony gradient boosting regression tree (ABC/GBRT) and Least Absolute Selection and Shrinkage Operator (LASSO) have been used to build models that enable the prediction of the SOC of a storage cell. The predictive results confirm the enhanced performance of the ABC/GBRT-based model over the other methods for SOC prediction. SOC errors remain below 1%, 10% and 17% for ABC/GBRT, ABC/MLP and LASSO, respectively. The goodness of fit, calculated using R^2, was 0.99, 0.95 and 0.81 for the three methods, respectively. A comparison of the results obtained using all the methods has also been carried out.

1. Introduction

The electric car stock reached 16,2 million in 2021 [1]. This figure is still low (9% of cars on the road worldwide), but points to very promising growth. The existence of profitable and efficient battery technologies has currently become a real necessity for the implantation of and maintaining the demand for electric cars [2,3].

One of the key issues to achieve electric vehicles (EVs) with longer driving ranges is higher energy density for their batteries. Li-ion is expected to be the dominant chemistry for battery of BEVs and PHEVs in the foreseeable future, as extensive research is being conducted in the field [4–6]. Li-ion technology provides relatively high power and energy for a given weight. Furthermore, this technology maintains reduced levels of self-discharge and is not affected by the memory effect present in other technologies. Li-Ion is also broadly used for low power applications such as consumer electronics.

A relevant parameter in all EV applications is the state of charge (SOC) of the cell [7–10]. SOC provides the electric car user with essential information about the available capacity in the battery and the time frame, during which it

* Corresponding author.
E-mail address: pjgarcia@uniovi.es (P.J. García-Nieto).

https://doi.org/10.1016/j.cam.2023.115305

can continue circulating before having to be recharged. The SOC expresses the maximum capacity of the battery as a percentage, where 100% is totally charged and 0% means totally exhausted.

The SOC cannot be measured directly, there is not exist such a sensor, so several techniques have been developed [4]. The most common method used for this purpose is *Coulomb counting*, where SOC is computed according to the concept of extracted charge, which is defined as:

$$Q_e(t) = \int_0^t \eta\, I_m(\tau)\, d\tau \tag{1}$$

where η is the Coulombic efficiency (the efficiency at which the charge is transferred) and $I_m(\tau)$ is the charge/discharge electric current (with the current sign changing accordingly).

The SOC can be defined in percentage, considering a fully charged cell at $t = 0$, as:

$$\mathrm{SOC} = 1 - \frac{Q_e}{C_N}\,[\%] \tag{2}$$

where C_N corresponds to the nominal cell capacity.

If the cell is not fully charged at $t = 0$, then, SOC definition is:

$$\mathrm{SOC} = \mathrm{SOC}_{t=0} - \frac{1}{C_N}\int_0^t \eta\, I(t)\, dt \tag{3}$$

where $\mathrm{SOC}_{t=0}$ represent the SOC at the initial-time. In summary, actual SOC depends on initial SOC and the evolution of current over time, i.e., $\mathrm{SOC}(\mathrm{SOC}_{t=0}, I, t)$.

Nevertheless, the accounting of charge, or electrical balance, presents several problems, including: (1) the inherent uncertainty in the determination of the initial SOC ($\mathrm{SOC}_{t=0}$); (2) errors due to inaccuracy of current measurement transducers (mainly sensor offset) that it is accumulated over time and require periodic recalibration; (3) the adoption of a capacity in the denominator, which in the case of nominal capacity, each manufacturer establishes the agreement with a test more or less arbitrarily; (4) the influence of the magnitude of the current and temperature on the discharge capacity; and (5) the battery's intrinsic aging effects that affect the battery capacity. In summary, all these factors prevent obtaining high precision measurements using this technique.

To solve the problems described above and improve the reliability of the measures, the proposed solutions go by contrasting Coulomb counting with other techniques. One of the most common considers the biunivocal relationships between the SOC and cell voltage in thermodynamic equilibrium, also known as OCV–SOC relationship (Open Circuit Voltage–SOC) [11–13]. The OCV–SOC, or EMF–SOC (ElectroMotive Force–SOC) to be more precise, is quite noticeable, with the exception of LFP technology, which has a very flat discharge curve [14]. For this technology, it is very difficult to obtain SOC values from voltage measurement, although OCV–SOC relation is virtually immune to temperature for virtually all lithium battery technologies. However, this technique requires the cell to acquire thermodynamic equilibrium, making it difficult to apply in practice. In the time elapsed from when the current becomes zero until thermodynamic equilibrium is reached, the cell manifests a dynamic component known as OCV relaxation that is dominated by the phenomenon of diffusion of ions into the cell. During relaxation, the cell voltage changes slowly and considering that voltage as the OCV (EMF) it can carry out an error in the estimation of the SOC. The problem is that the relaxation can take hours, especially when the battery is almost empty, when the temperature is low or when the cell is previously charged or discharged at high currents. Measuring the EMF of the battery requires no current flow for a period of time long enough for the OCV to relax and reach equilibrium, which reduces the possibility of applying the method on-line. In summary, the greatest challenge of this technique lies in being able to estimate the EMF realistically under dynamic conditions, in real time, without the need to wait for the relaxation time. Although different techniques have been proposed to estimate the value of EMF from the value of OCV during the first stages of the relaxation process [15], other battery models such as the equivalent circuit models [16] or the electrochemical models [17] are required to obtain accurate SOC estimations. Electrochemical models are based on reproducing the chemical processes that take place inside the cell. Therefore, they require a very deep understanding of the phenomena that occur inside the cell, but they provide very accurate results. Although high-precision electrochemical modeling provides a complete view of the state of the cell, its implementation also has limitations: the model parameters are difficult to obtain for a commercial cell (manufacturers are reticent to reveal confidential information that affects its competitive advantage). On the other hand, many of these parameters change as the battery ages, so they must be adapted, which makes them difficult to use. Furthermore, electrochemical models are governed by a set of coupled nonlinear differential equations whose solution requires considerable computational effort, which prevents their implementation in a low-cost microcontroller-based battery management system (BMS). Its on-line predictive utility is also limited, except for very simplified models in which its reliability is more doubtful.

SOC estimation based on data (data-driven) is a relatively new approach, with increasing utility due to the availability of large amounts of data and powerful computers. Data-driven models, also known as black box models, are based on the empirical observation of the variables that are part of a system to build a model that reveals the relationships between the variables of interest [18]. This type of modeling requires minimal or no expert knowledge about the behavior of the battery as it does not require the existence of a previous model (electrical or electrochemical) to serve as a support. One of the advantages of these methods is that the chemical phenomenon inside the cell is not necessarily to be known. The

algorithms used by these methods can estimate the SOC by monitoring directly and easily measurable variables of a cell such as: current, voltage and temperature, without requiring any electrochemical parameter. As there is no underlying model, the computational load involved in the usual parameter estimation of previous models is avoided. In general, the structure and parameters of the data-driven model are obtained directly through the modeling algorithm itself using the input data. In machine learning models, the parameters are the variables that are estimated during the training process with the data sets. The values of these parameters do not need to be provided a priori but are obtained from the data. Several machine learning techniques as fuzzy-logic [19,20], neural networks [21] and support vector machines [22,23] are used for this goal with different degrees of accuracy.

In this paper, SOC estimation is addressed using three machine learning techniques. The first technique uses a novel combination of the so-called Gradient Boosting Regression Tree (GBRT) method and Artificial Bee Colony (ABC). GBRT is a regression technique based on statistical learning [24] with ABC as the optimal GBRT hyper-parameters locator. The hybrid model obtained from this combination replaces the traditional method for determining parameters using empirical brute force [25]. ABC is based on the smart bees' relationship when they search for food [26]. As occurs with other related optimization methods, like Particle Swarm Optimization [27] or Ant Colony Optimization [28], ABC is motivated by the collective conduct of animal associations [29].

The second method for SOC prediction is a feedforward Artificial Neural Network (ANN) called Multilayer Perceptron (MLP) [30]. In this paper, MLP is likewise optimized using ABC. The third technique is known as LASSO (Least Absolute Shrinkage and Selection Operator). The LASSO technique is able to select the variables of the model and also take into account regularization to avoid overfitting [31].

2. Materials and methods

For artificial intelligence methods to achieve reasonable results, it is necessary to provide them with sufficient experimental data and these data must be truly sufficient and representative. In other words, the results must cover a good part of the possible battery operating scenarios. The need for these results constitutes a complex and laborious task that requires a considerable training time. To avoid this issue, the training data is obtained from a realistic test called Dynamic Stress Test (DST) [32]. The DST specifies a loading and unloading profile that simulates the energy demand of an EV battery while driving. This specific regime can effectively simulate dynamic discharging and can be implemented with equipment at most test laboratories and developers. Thus, a realistic and reproducible battery data profile is used to build the ABC/GBRT, ABC/MLP and LASSO models.

The battery data were obtained using the PEC Corporation's test bench, model SBT 10050 [33] (see Fig. 1). The equipment consists of two main components: (i) charge–discharge equipment with 12 independent channels. Every channel has its own microprocessor and is fully programmable with its own charge/discharge pattern, (ii) data acquisition system used to measure cell current, voltage, and temperature. The equipment can be programmed as a list of consecutive charge/discharge events, and it can achieve a DST profile.

The tested cell was a high-energy Lithium-Iron-Phosphate cell ($LiFePO_4$) manufactured by European Batteries (model EV 42 Ah) with a rated capacity of 42 Ah (see Fig. 1). A temperature NTC sensor is placed at the center of the battery body. Additionally, a temperature-controlled chamber is used during the test to maintain the room temperature under control.

This study uses a dataset that comes from a DST-type dynamic test. The experimental setup follows all the specifications described in the Battery Test Procedures Manual (USABC) [32]. This manual summarizes the procedural information needed to perform the battery testing. DST is a variable power discharge testing used to produce the effects of electric vehicle driving behavior (including regenerative braking) on the performance and life of a battery. This variable power discharge regimen is based on the auto industry standard Federal Urban Driving Schedule (FUDS), a complex 1372 s time–velocity profile originally based on actual driving data [32]. DST is a simplified variable power discharge cycle with the same average characteristics as the FUDS regime. Thus, the DST-type test subjects the battery to a load profile that simulates the demand of an EV under real operating conditions. DST test consists of seven different levels of power that are reproduced successively along 20 steps over a period of 360 s. Table 1 shows the DST test profile, which is applied repetitively up to complete the battery discharge. Negative values in Table 1 mean discharging currents and positive values mean charging currents. Values in the duration column are specified in seconds.

The full test follows this sequence:

(a) Charging: Fully charge the battery to 100% of SOC.
(b) Stabilized temperature after charge: the battery is kept in an open circuit.
(c) DST discharge: The established discharge profile, according to Table 1, is applied.
(d) Test termination: happens when one of the following events occurs during the test: (1) when the battery is not capable of delivering the requested power; or (2) when a net charge exceeding its rated capacity is delivered from the battery.

The data for the analysis were obtained from 31 consecutive DST cycles, and the total number of values processed was about 89,160 (that is, 22,290 samples at intervals of 500 ms of SOC and a total of three variables: voltage, current and temperature).

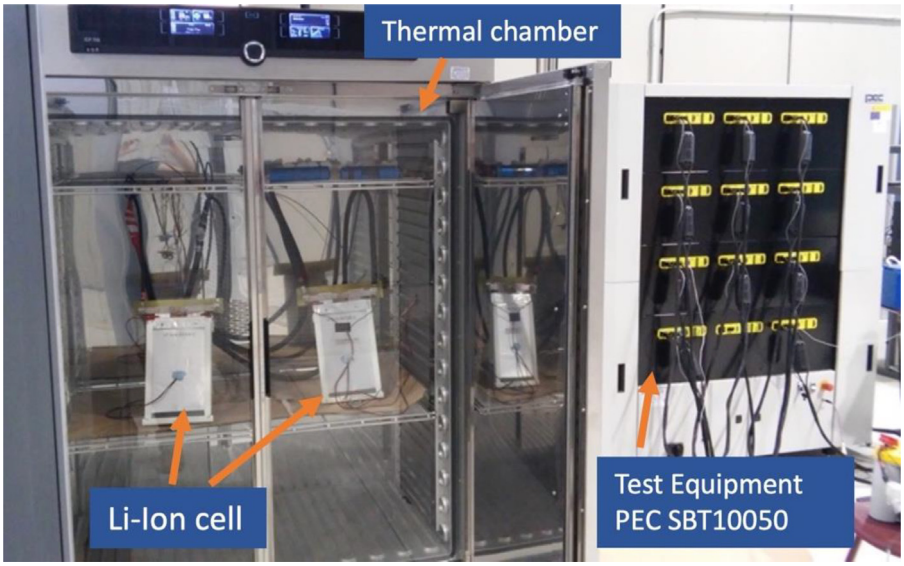

Fig. 1. $LiFePO_4$ cell, thermal chamber and test equipment (PEC SBT10050).

Table 1
Listing of DST power profile.

Step #	Duration (s)	Discharge current (A)	Step #	Duration (s)	Discharge current (A)
1	16	0	11	12	−25
2	28	−12.5	12	8	12.5
3	12	−25	13	16	0
4	8	12.5	14	36	−12.5
5	16	0	15	8	−100
6	24	−12.5	16	24	−62.5
7	12	−25	17	8	25
8	8	12.5	18	32	−25
9	16	0	19	8	50
10	24	−12.5	20	44	0

Cell voltage, current and temperature (independent variables) are the only features used to build the models, thus, internal battery parameters are not required to predict the SOC, information that, on the other hand, is not easily obtained directly from battery manufacturers.

3. Mathematical modeling techniques

3.1. Gradient boosting regression tree (GBRT) technique

Gradient boosting is a mathematical technique based on decision trees with applications in classification and regression problems. Following the usual strategy of the boosting methods, this model is built by stages, so that we obtain a single solid ensemble model looking for the optimum of a differentiable loss function [34–41].

It is possible to describe this approach as a regression method with the goal to instruct a model $\mathcal{F}$ to foretell the values $\hat{y} = \mathcal{F}(x)$, calculating the minimum of the mean squared error $(\hat{y} - y)^2$, so that y are the observed values from the training set.

To fix ideas, we choose an $\mathcal{F}_m$ that foretells the mean y from this training collection for each stage $1 \leq m \leq M$ of gradient boosting. Next, the gradient boosting algorithm gets better $\mathcal{F}_m$ building an $\mathcal{F}_{m+1}$ model. This new model, $\mathcal{F}_{m+1}$, incorporates a term $\hbar$ to ameliorate the previous model, $\mathcal{F}_{m+1}(x) = \mathcal{F}_m(x) + \hbar(x)$. In order to determine $\hbar$, GBRT considers an appropriate $\hbar$ following [36,39–41]:

$$\mathcal{F}_{m+1}(x) = \mathcal{F}_m(x) + \hbar(x) = y \tag{4}$$

with

$$\hbar(\mathrm{x}) = \mathrm{y} - \mathcal{F}_{\mathrm{m}}(\mathrm{x}) \tag{5}$$

Therefore, GBRT carries out $\hbar$ estimation considering the difference $y - \mathcal{F}_m(x)$. This means that $\mathcal{F}_{m+1}(x)$ is constructed from the previous ancestor $\mathcal{F}_m(x)$ in each stage.

Considering y and x as input and output variables. The purpose is to discover an estimation $\hat{\mathcal{F}}(x)$ that minimizes a lost function $\mathcal{L}(y, \mathcal{F}(x))$ given the training set of x and y values, [38–45]:

$$\hat{\mathcal{F}} = arg_F min E_{x,y}[\mathcal{L}(y, \mathcal{F}(x))] \tag{6}$$

In this way, the gradient boosting method determines roughly y by means of a weighted sum of functions $\hbar_i(x)$ belonging to some class $\mathcal{H}$, termed soft learners (i.e., weak trainees):

$$\mathcal{F}(x) = \sum_{i=1}^{M} \gamma_i \hbar_i(x) + const \tag{7}$$

Hence, this procedure searches $\hat{\mathcal{F}}(x)$ minimizing a cost function using training values through the empirical risk minimization principle. Indeed, it begins with a methodology that chooses a constant function $\hat{\mathcal{F}}_0(x)$, and then develops its value in an avaricious form [41,44,45] in stages:

$$\hat{\mathcal{F}}_0(x) = arg_\gamma min \sum_{i=1}^{n} \mathcal{L}(y, \gamma) \tag{8}$$

$$\hat{\mathcal{F}}_m(x) = \hat{\mathcal{F}}_{m-1}(x) + arg_{\hbar \in \mathcal{H}} min \sum_{i=1}^{n} \mathcal{L}(y_i, \hat{\mathcal{F}}_{m-1}(x_i) + \hbar(x_i)) \tag{9}$$

It is necessary to consider that the selection of $\hbar$ at every step for a random cost function $\mathcal{L}$ becomes in an infeasible optimization problem. However, it is possible to reduce this problem so that a gradient descent method can be applied here. In this sense, the model can be upgraded according to [41,44,45]:

$$\hat{\mathcal{F}}_m(x) = \hat{\mathcal{F}}_{m-1}(x) + \gamma_m \sum_{i=1}^{n} \nabla_{\hat{\mathcal{F}}_{m-1}} \mathcal{L}(y_i, \hat{\mathcal{F}}_{m-1}(x_i)) \tag{10}$$

$$\gamma_m = arg_\gamma min \sum_{i=1}^{n} \mathcal{L}\left(y_i, \hat{\mathcal{F}}_{m-1}(x_i) - \gamma \frac{\partial \mathcal{L}(y_i, \hat{\mathcal{F}}_{m-1}(x_i))}{\partial \hat{\mathcal{F}}_{m-1}(x_i)}\right) \tag{11}$$

where partials of lost functions are calculated with respect $\hat{\mathcal{F}}_i$.

If the collection $\mathcal{H}$ is not infinite (i.e., a limited set), we will choose the applicant function $\hbar$ closest to $\partial\mathcal{L}/\partial\mathcal{F}$ and γ determined using Eqs. (7) and (8). It is important to realize that this approximation is of a heuristic type and it will not produce a precise solution to the problem, but a high-quality approach.

A description of the operating principles of the computer program corresponding to the typical gradient boosting method is indicated as follows [38,41,44,45]:

- Input:
  - training samples of x_i and y_i
  - $\mathcal{L}(y_i, \hat{\mathcal{F}}(x_i))$ the cost function (must be differentiable).
  - M, iteration number.
- Algorithm:
  - Begin using:

$$\hat{\mathcal{F}}_0(\mathrm{x}) = \arg_\gamma \min \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathcal{L}(\mathrm{y_i}, \gamma)$$

  - Repeat with $m = 1:M$
    - First, calculate residuals:

$$r_{im} = -\left[\frac{\partial \mathcal{L}(\mathrm{y_i}, \hat{\mathcal{F}}(\mathrm{x_i}))}{\partial \hat{\mathcal{F}}(\mathrm{x_i})}\right]_{\hat{\mathcal{F}}(x) = \hat{\mathcal{F}}_{\mathrm{m}-1}(x)} \quad i = 1, \ldots, n$$

    - Second, adjust soft learner $\hbar_m(x)$ to the pseudo-residuals, using the training set $\{(x_i, r_{im})\}_{i=1\ldots n}$
    - Third, we determine γ_m as:

$$\gamma_\mathrm{m} = \arg_\gamma \min \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathcal{L}\left(\mathrm{y_i}, \hat{\mathcal{F}}_{\mathrm{m}-1}(\mathrm{x_i}) - \gamma \hbar_m(\mathrm{x_i})\right)$$

∗ Finally:

$$\hat{\mathcal{F}}_{\mathrm{m}}(\mathrm{x}) = \hat{\mathcal{F}}_{\mathrm{m}-1}(\mathrm{x}) + \gamma_{\mathrm{m}} \hbar_m(x)$$

– Obtain:

$\hat{\mathcal{F}}_{\mathrm{m}}(\mathrm{x})$

It is very common to use the gradient boosting technique with decision trees, specifically with CART trees, of a given limited size as soft learners. In this case, Friedman suggests a variant of the gradient boosting technique that increases the goodness-of-fit of each soft learner [38,41,44–46].

In this way, a typical gradient boosting fits a decision tree $\hbar_m(x)$ to the pseudo-residuals in the mth stage. Let $\mathcal{J}_m$ as the model leaves number, it separates the input space into $\mathcal{J}_m$ divided zones $\mathcal{R}_{jm}, \ldots, \mathcal{R}_{\mathcal{J}_m m}$ and acquires a constant value for each zone. Therefore, $\hbar_m(x)$ can be written for the input x as the summation [39–41,44,45]:

$$\hbar_m(x) = \sum_{i=1}^{\mathcal{J}_m} \mathcal{b}_{jm} \mathcal{I}, x \in \mathcal{R}_{jm} \tag{12}$$

being $\mathcal{b}_{jm}$ the term determined for the region $\mathcal{R}_{jm}$.

The $\mathcal{b}_{jm}$ coefficients are multiplied by γ_m, determined by line search method. Next, we upgraded the model according to:

$$\hat{\mathcal{F}}_m(x) = \hat{\mathcal{F}}_{m-1}(x) + \gamma_m \hbar_m(x) \tag{13}$$

$$\gamma_m = arg_\gamma min \sum_{i=1}^{n} \mathcal{L}\left(y_i, \hat{\mathcal{F}}_{m-1}(x_i) - \gamma \hbar_m(x_i)\right) \tag{14}$$

Friedman suggested a variation selecting a distinct optimum γ_{jm}, in place of only one γ_m for the entire tree. This customized algorithm is termed TreeBoost.

Next, an upgraded model is obtained as indicated below [39–41,44,45]:

$$\hat{\mathcal{F}}_m(x) = \hat{\mathcal{F}}_{m-1}(x) + \sum_{i=1}^{\mathcal{J}_m} \gamma_{jm} \mathcal{I} \quad x \in \mathcal{R}_{jm} \tag{15}$$

$$\gamma_{jm} = arg_\gamma min \sum_{x_i \in \mathcal{R}_{jm}}^{\mathcal{L}} \left(y_i, \hat{\mathcal{F}}_{m-1}(x_i) + \gamma\right) \tag{16}$$

Being $\mathcal{J}$ terminal nodes in trees. This parameter can be configured for the training data collection. Indeed, it manages the interaction level among variables in the model. If $\mathcal{J} = 2$, the variables do not interact. If $\mathcal{J} = 3$ interactions are allowed between two variables, and so forth. If $\mathcal{J}$ value ranging from 4 to 8, this technique functions properly, so that the results are independent of $\mathcal{J}$. In most cases $\mathcal{J}$ ranges from 3 to 10, obtaining accurate results in the calculations.

In order to avoid overfitting, regularization techniques are used during the training process. Several approaches permit us to achieve this goal [39–41,44,45]. Specifically, the *Xgboost* algorithm contains a penalty function in order to restrict the overfitting in the loss function:

$$\mathcal{L}(x) = \mathcal{E}(x) + \mathcal{P}(x) \tag{17}$$

with:

- $\mathcal{E}$ is the mean squared error or any other appropriate kind of error function; and
- $\mathcal{P}$ is the penalty term and controls the complexity of the model. Indeed, if the model complexity increases, this function penalizes the loss function enlarging its value, with the purpose of the overfitting decrease.

In this research study, *Artificial Bee Colony* (ABC) is used to estimate model parameters. The approach adopted here, which combines GBRT with ABC, avoids the *empirical* estimation of the GBRT parameters. Fundamentals about the ABC algorithm are explained in the next section.

3.2. The Artificial Bee Colony (ABC) algorithm

Artificial Bee Colony (ABC) was developed by Karaboga [26,29]. This technique is founded on honeybee foraging behavior and belongs to the type of algorithms with a behavior based on the interchange of information among entities that make up a group [47,48]. It is a flexible algorithm able to solve real-world problems where an optimization process is required. Although the initial applications of ABC were in numerical optimization, current research topics extend ABC to the optimization of hybrid functions, engineering design problems, multi-objective optimization problems, neural network training and image processing problems, amongst others [49–51].

An ABC swarm is a set of bees able to collaborate and communicate among themselves to perform the task of collecting. ABC uses bees with three different roles: (a) employed; (b) onlooker; and (c) scout bees [52]. Employed bees are related to the source of food. They share information such as distance and direction from the hive with the onlooker bees [53]. They choose sources of food through the information shared by employed bees. Onlooker bees select high quality food sources with more probability than low quality food sources. The scouts search for new sources of food around the hive [53].

In ABC, the hive is made up of the same number of onlooker and employed bees. The swarm's food sources, or tentative solutions, depend on the number of onlooker and employed bees. Initially, ABC creates a random population of SN food sources or solutions. If the *i*th food source is expressed as:

$$X_i = \{x_{i1}, x_{i2}, \ldots, x_{in}\} \tag{18}$$

a V_i is generated by every employed bee in the proximity of its position following the equation [47,52]:

$$V_{ik} = X_{ik} + \Phi_{ik} \cdot \left(X_{ik} - X_{jk}\right) \tag{19}$$

where X_j is a solution ($i \neq j$) that is selected randomly, k is an index randomly selected from the set $\{1, 2, \ldots, n\}$ that indicates the chosen dimension and Φ_{ik} is a number generated from an uniform distribution in the interval $[-1, 1]$. Once the V_i candidate solution is obtained, if the new value of V_i improves that of its father X_i, then X_i is updated with V_i. Otherwise, the value of X_i is kept unchanged. Once the search process is completed by the employed bees, they share the information about the position of their food sources with the onlooker bees by means of their dances. Then, the onlooker bee assesses the information on the collected nectar and picks up a food source taking into account a probability that is related to the amount of nectar in it. The probabilistic selection constitutes a mechanism of selection of roulette which is described in the following equation [52,54]:

$$P_i = \frac{fit_i}{\sum_j fit_j} \tag{20}$$

where fit_i is the fitness value for the *i*th food source. Therefore, the better the food source *i*, the higher the probability it remains in the set of tentative solutions. If one of the food sources does not improve after a fixed number of cycles, it is discarded.

The scout bee is responsible for finding a replacement for the discarded source, following the equation [52,54]:

$$X_{ik} = lb_j + ran\,(0, 1) \times \left(ub_j - lb_j\right) \tag{21}$$

so that $ran\,(0, 1)$ is a random number in the interval [0, 1] relied on a uniform distribution being *lb* and *ub* the lower and upper boundaries that correspond to the *i*th dimension, respectively.

3.3. Artificial neural network (ANN): multilayer perceptron (MLP)

The second technique used in this paper for SOC prediction is Artificial Neural Networks (ANN). ANN are processing algorithms modeled like the neural structure of the cerebral cortex but on a smaller scale. Neural networks can be used to determine relationships and patterns between inputs and outputs [55,56]. ANN are usually organized in layers, which are made up of a number of interconnected nodes and which contain an activation function. Patterns are introduced to the network via the input layer. This layer can be communicated with hidden layers where the actual processing is performed via a system of weighted connections. A popular type of neural networks is the multilayer perceptron (MLP) [30,56]. MLP is built with an input, a hidden and output layers. What makes a MPL different is the use of a nonlinear activation function (AF).

The function implemented by the network is $f : X \subset \Re^n \rightarrow Y \subset \Re^c$, represented as [55–57]:

$$\begin{aligned} &f\,(\mathbf{X}) = \phi\,(\psi\,(\mathbf{X})) = (\phi^{\circ}\psi)\,(\mathbf{X}) \\ &\phi : X \subset \Re^n \rightarrow U \subset \Re^h \\ &\psi : U \subset \Re^h \rightarrow Y \subset \Re^c \end{aligned} \tag{22}$$

where U is hidden variables space. Considering this architecture [55,56], it results:

- $\psi_j\,(\mathbf{X}) = \psi\left(\mathbf{W}_j^T\mathbf{X} + W_{j0}\right)$: ψ represents the AF; $\mathbf{W}_j \in \Re^n$ *is* a parameter; and $W_{j0} \in \Re$ is a threshold value. There are three types of AF: hyperbolic tangent, sigmoid and logistic AF.
- $\phi_j\,(\mathbf{u}) = \phi\left(\mathbf{c}_j^T\mathbf{u} + c_{j0}\right)$: ϕ is the AF of the output layer neurons; $\mathbf{c}_j \in \Re^h$ are the weights; and $c_{j0} \in \Re$ is the threshold. Generally, ϕ is the Heaviside or dichotomous function.

The MLP implements the following function [55–57]:

$$f\,(\mathbf{X}) = \sum_{j=1}^{h} c_j \psi\left(\mathbf{W}_j^T\mathbf{X} + W_{j0}\right) + c_0 \tag{23}$$

3.4. Least absolute shrinkage and selection operator (LASSO)

The third technique used in this paper for SOC prediction is the LASSO regression [31,34,35]. Typically, we have n observations, that is, we have a set of training data $(x_1, y_1), \ldots, (x_n, y_n)$, each of which consists of p features (number of variables) and a single output. Let y_1 the output and $x_i = \left(x_{i1}, x_{i2}, \ldots, x_{ip}\right)^T$, the covariate vector for the ith sample. A popular regression technique, least squared, is based on minimizing a cost function given by [31,34,35]:

$$RSS = \sum_{i=1}^{n}\left(y_i - \theta_0 - \sum_{j=1}^{p}\theta_j x_{ij}\right)^2 \tag{24}$$

where $\boldsymbol{\theta} = \left(\theta_1, \theta_2, \ldots, \theta_p\right)^T$ are the coefficients that must be calculated in order to minimize Eq. (24). There is a way to improve the classical linear regression, the regularized linear regression, by including a regularization term in Eq. (24), constituted by squares of the coefficients to compute (excluding the intercept). Then, we choose $\boldsymbol{\theta}$ to minimize [31,34,35]:

$$\sum_{i=1}^{n}\left(y_i - \theta_0 - \sum_{j=1}^{p}\theta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\left|\theta_j\right| \tag{25}$$

where $\lambda \geq 0$ is the regularization parameter or complexity parameter, that address the overfitting, keeping all the features, but reducing the magnitude/values of parameters θ_j. In fact, Eq. (25), known as LASSO regression, trades off two different criteria. The first seeks a fitting function that is as close as possible to the data, minimizing RSS. However, the second term, $\lambda\sum_{j=1}^{p}\left|\theta_j\right|$, which is the shrinkage penalty, seeks to obtain small $\theta_1, \theta_2, \ldots, \theta_p$, and hence shrinks the coefficients θ_j. The parameter λ establishes the relative importance of these two terms in the final solution. If $\lambda = 0$, the second term is cancelled out and, as $\lambda \to \infty$, the effect of the shrinkage increases. Thus, the solution will depend on this parameter and different coefficient estimates, $\hat{\theta}_{\lambda}^{LASSO}$, will be obtained for each value of λ. It is important to select a good λ value. Grid-search with cross-validation will be used here to tune this parameter.

The intercept θ_0 is not included in the set of coefficients to shrink because it is an estimation of the average value when $x_{i1} = x_{i2} = \cdots = x_{ip} = 0$. If the data has been scaled, i.e., a linear transformation has been applied (mean value is zero for all the independent variables), then $\theta_0 = \overline{y} = \frac{\sum_{i=1}^{n} y_i}{n}$.

LASSO regression controls the overfitting, reducing the magnitude of large θ coefficients. It could be said that the penalty term in LASSO uses L_1- norm, because the L_p- norm of a vector $\boldsymbol{\theta}$ is given by:

$$\|\boldsymbol{\theta}\|_p = \left(\sum_{j=1}^{p}\left|\theta_j\right|^p\right)^{\frac{1}{p}} \tag{26}$$

Therefore, LASSO regression shrinks the coefficients in the regression function towards zero, but the use of the L_1- norm penalty forces some of the coefficients to vanish when λ is large enough. Hence, LASSO is able to drop variables and thus perform variable selection. This results in a model that is easier to interpret. LASSO can be said to obtain sparse models, i.e., models with comparatively few variables.

3.5. Approach accuracy

As is well known, the SOC is the dependent variable to be predicted. To predict SOC from the input variables with sufficient confidence, it is essential to select the best model fitted to the observed dataset. Although several possible statistics can be used to ascertain the goodness-of-fit, the principal rule employed in this investigation was the coefficient of determination R^2 [58,59], as it is a statistic employed in the scope of a statistical model whose principal objective is to predict upcoming results or to check an assumption. Next, the observed values are referred to as t_i and the values predicted by the model y_i, making it possible to define the following sums of squares given by [58,59]:

- $SS_{tot} = \sum_{i=1}^{n}\left(t_i - \overline{t}\right)^2$: is the overall sum of squares, proportional to the sample variance.
- $SS_{reg} = \sum_{i=1}^{n}\left(y_i - \overline{t}\right)^2$: is the regression sum of squares, also termed the explained sum of squares.
- $SS_{err} = \sum_{i=1}^{n}\left(t_i - y_i\right)^2$: is the residual sum of squares.

where $\overline{t}$ is the mean of the n observed data:

$$\overline{t} = \frac{1}{n}\sum_{i=1}^{n} t_i \tag{27}$$

Table 2
GBRT search space.

GBRT hyper-parameters	Lower limit	Upper limit
Rounds	1	100
η	0.1	1
γ	0	30
Minimum child weight (MCW)	1	30
Maximum Δ step (MDS)	0	30
Subsample ratio	0.5	1

Table 3
GBRT optimal hyper-parameters.

GBRT hyper-parameters	Optimal values
Rounds	100
η	0.29
γ	0.0
Minimum child weight (MCW)	2.4
Maximum Δ step (MDS)	18
Subsample ratio (SR)	1.0

Based on the former sums, the coefficient of determination is specified by the following equation [58,59]:

$$R^2 \equiv 1 - \frac{SS_{err}}{SS_{tot}} \tag{28}$$

The closer the R^2 statistic is to the value 1.0, the smaller the disagreement between the experimental and foretold data. Similarly, the mathematical expressions for the other two statistics used in this study (RMSE and MAE) are as follows [58,59]:

$$RMSE \equiv \sqrt{\frac{1}{n}\sum_{i=1}^{n}(t_i - y_i)^2} \tag{29}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|t_i - y_i| \tag{30}$$

Higher values of R^2 are preferred, i.e. closer to 1 means better model performance and regression line fits the data well. Conversely, the lower the RMSE and MAE values are, the better the model performs.

4. Results and discussion

Fig. 2 shows the ABC/GBRT-based model flow diagram. First, random initialization of parameters is performed. After a phase of training and validation, a fitness function is calculated. If the termination criteria is not satisfied, a new cycle with new parameters is applied to the model based on ABC searching. This cycle is maintained until optimal parameters are calculated.

Table 2 shows the GBRT search space and Table 3 the resulting optimal hyperparameters using ABC.

The ABC/GBRT-based model was built using two packages from the R project: the *ABCoptim* and *Xgboost packages* [60, 61].

The results for the ABC/MLP model are presented next. The optimal parameters related to ABC/MLP are shown in Table 4.

The hybrid ABC/MLP model was built in this study employing the ABC code for MATLAB [62] and functions from the MATLAB Neural Network Toolbox.

Finally, Table 5 shows the result of the LASSO regression method. A good λ value selection is critical. Thus, λ-parameter optimization was performed using the grid-search technique with cross-validation.

LASSO was implemented using the LASSO function from *Python's scikit-learn library* [63].

The graphical results for the three hybrid methods are shown in Figs. 3 to 5.

Fig. 3 shows the simulated results for the ABC/MLP model and Fig. 4 for the ABC/GBRT model. LASSO results are shown in Fig. 5.

It can be observed from Fig. 3 (ABC/MLP model), that the estimation error grows near 70% and 50% of SOC.

Fig. 4 shows the predicted and observed SOC values of ABC/GBRT model. Better accuracy is obtained using the ABC/GBRT technique, as is shown in Fig. 4. A good fit is obtained for SOC values between 100% to 75% and SOC values lower than 40%.

LASSO prediction shows the worst results, as is shown in Fig. 5. A considerable error is obtained around SOC of 60%.

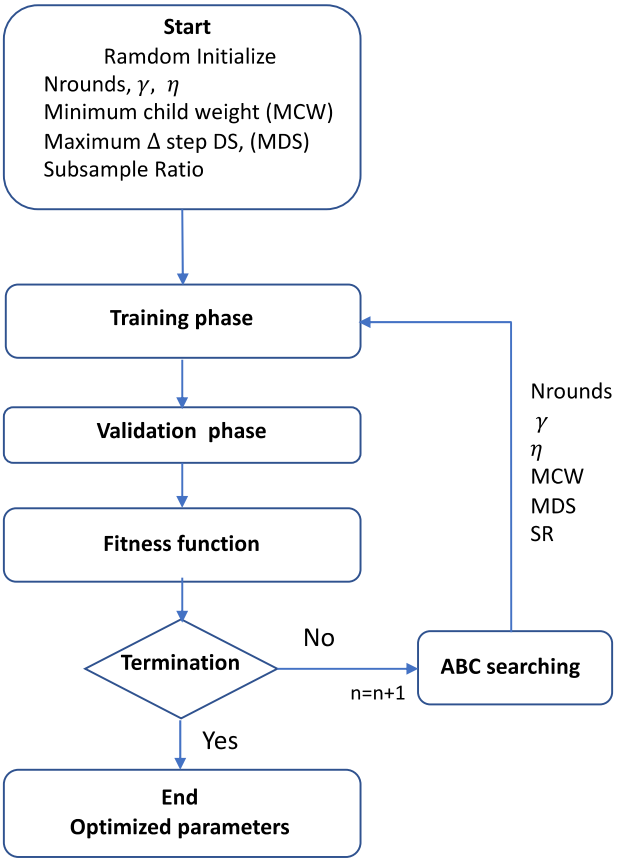

Fig. 2. ABC/GBRT-based model flow diagram.

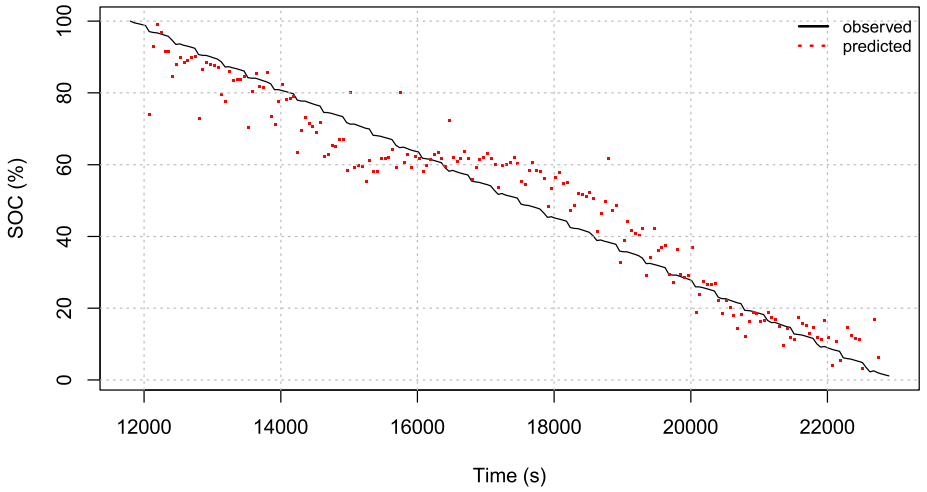

Fig. 3. Observed and predicted SOC results (100% to 0%) for the ABC/MLP model.

In order to analyze the ABC/GBRT results in more detail, Fig. 6 shows experimental and predicted SOC from 100% to 95%.

The experimental voltage, current and temperature values used as input variables for this period are shown in Figs. 7, 8 and 9, respectively. Fig. 8 shows the typical pattern of a DST test described in Table 1, where negative current values mean a cell under discharge and positive values correspond to charge periods.

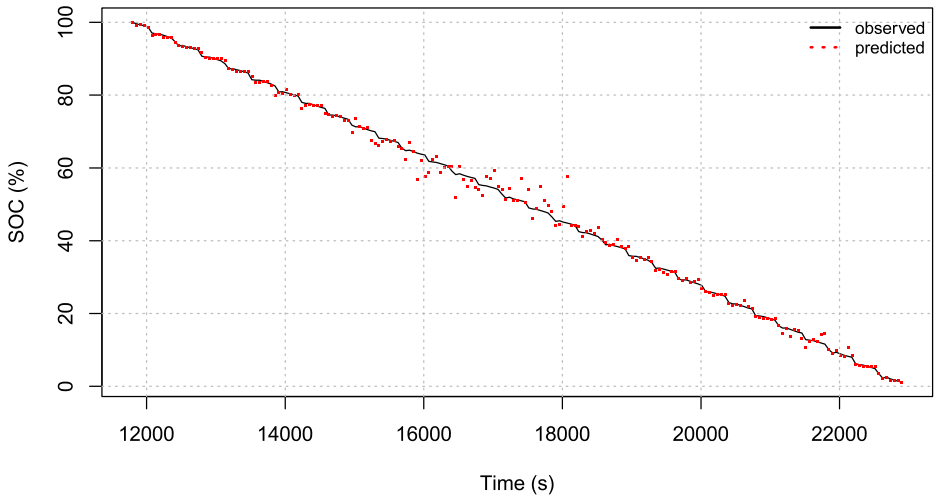

Fig. 4. Observed vs predicted SOC (100% to 0%) using the ABC/GBRT-based model.

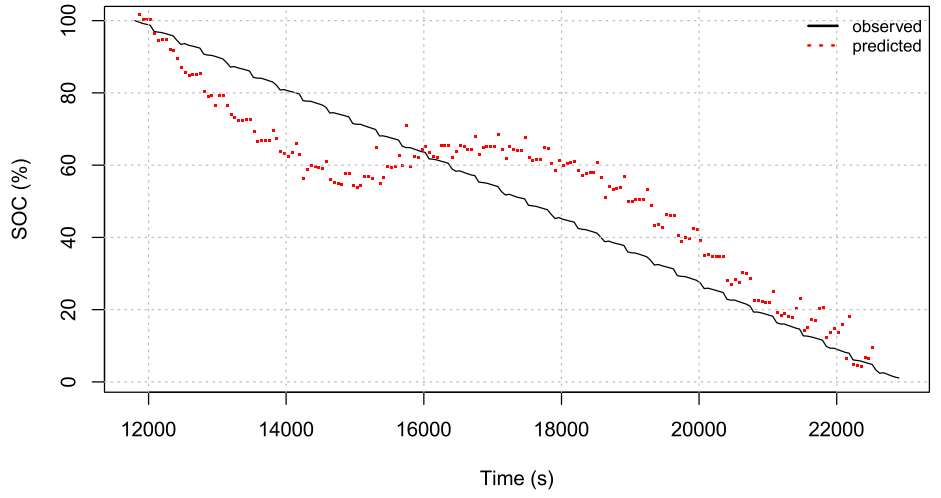

Fig. 5. Observed and predicted SOC (100% to 0%) results for the LASSO model.

Table 4
Optimal hyper-parameters for ABC/MLP model.

Parameters	Values
#hidden-neurons	13
Learning rate	2.47×10^{-3}
Momentum factor	5.15×10^{-1}
Activation function	Tangent sigmoid

Table 5
Optimal parameter for the LASSO model.

Parameter	Optimal	Cross validation R^2
λ	1.4×10^{-7}	0.8173

The absolute errors of the three methods are shown in Fig. 10. SOC errors are kept below 1%, 10% and 17% for ABC/GBRT, ABC/MLP and LASSO, respectively.

Table 6 compares the predictive techniques for SOC using R^2 and r as the most common criteria along with the root mean squared error (RMSE) and mean absolute error (MAE) as complementary criteria [31,34,35].

Additionally, the comparative relevance of the input variables in this approach is illustrated in Table 7 and Fig. 11.

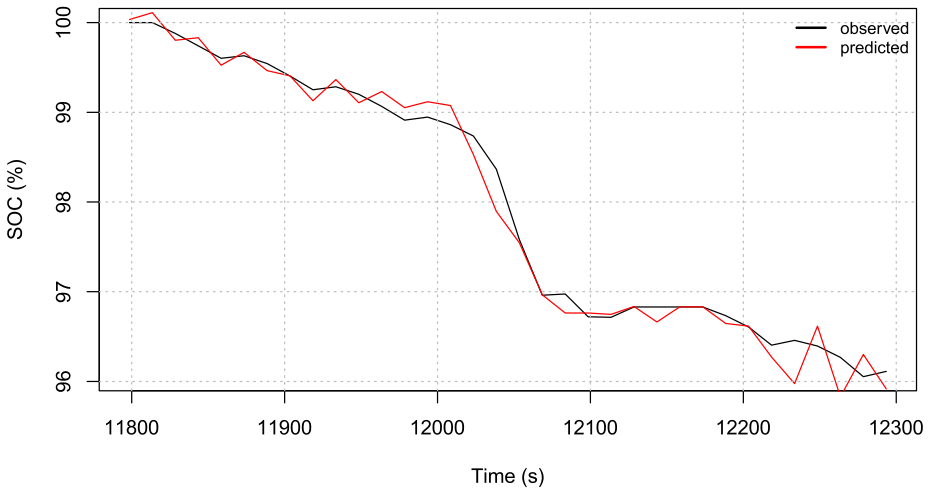

Fig. 6. Observed vs predicted SOC using the ABC/GBRT-based model (from 100% to 95% of SOC).

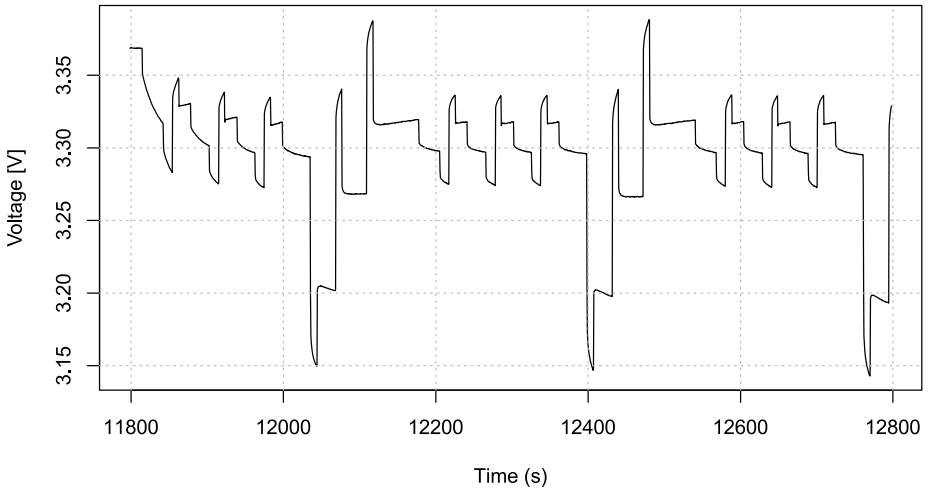

Fig. 7. Experimental cell voltage (100% to 95% of SOC).

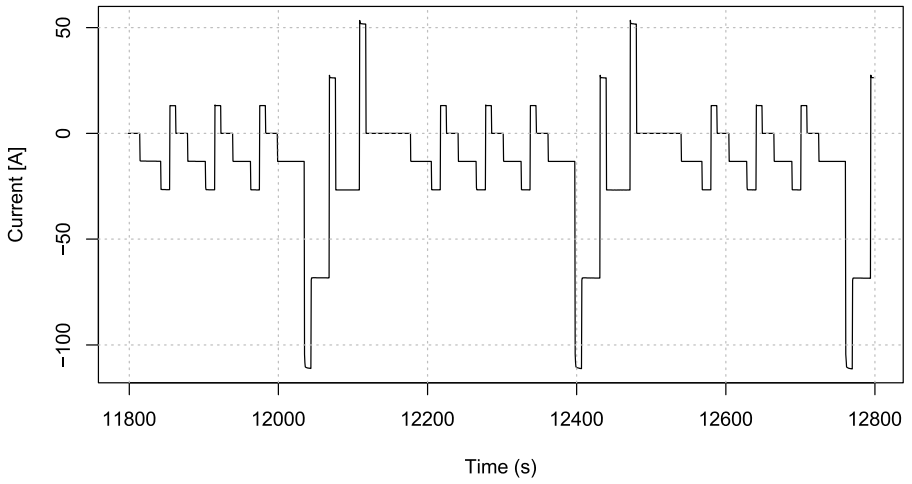

Fig. 8. Experimental cell current during 100% to 95% of SOC.

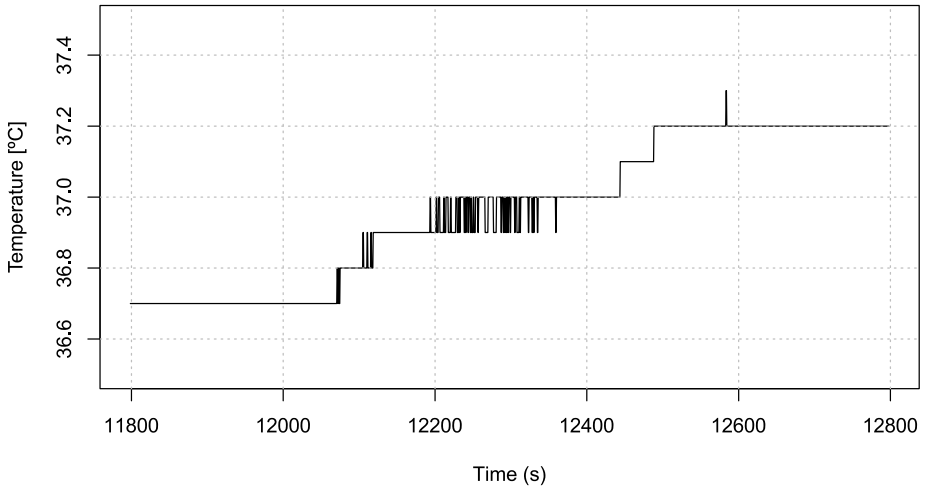

Fig. 9. Experimental cell temperature during 100% to 95% of SOC.

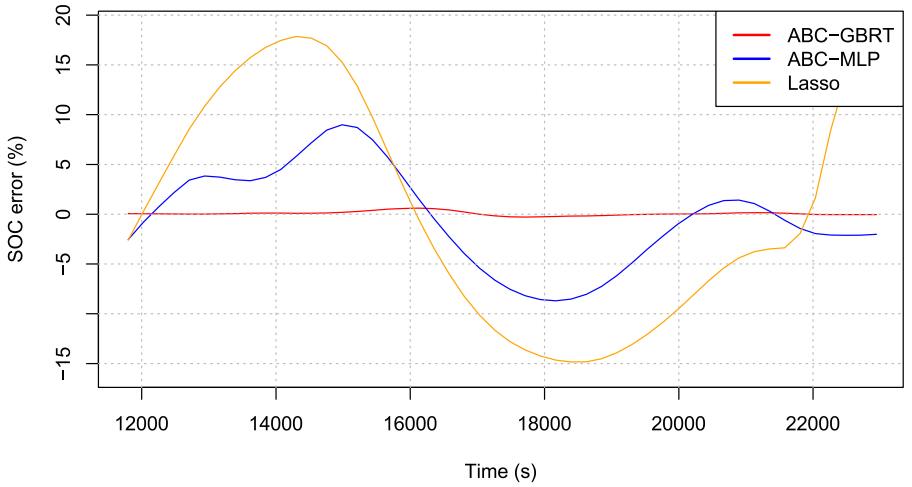

Fig. 10. Absolute error of ABC/GBRT, ABC/MLP and LASSO.

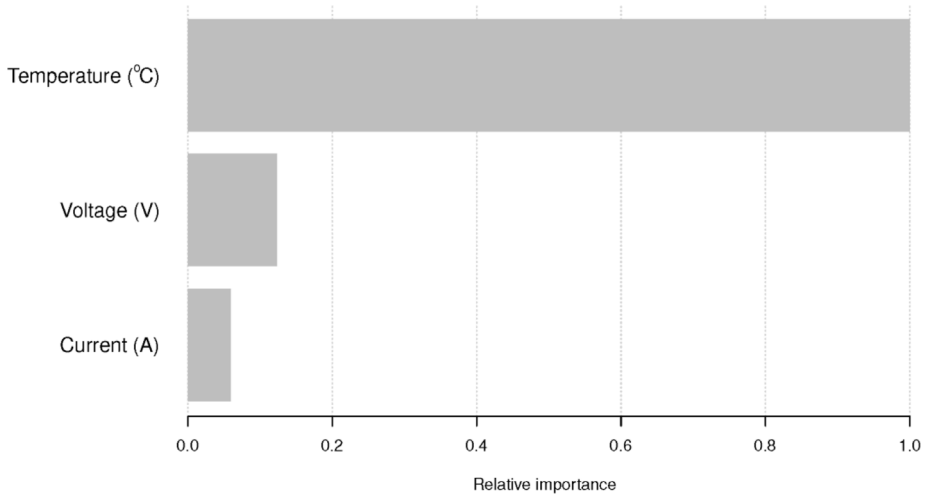

Fig. 11. Relative importance of the input operation variables to predict SOC.

Table 6
Comparison of prediction techniques for the SOC estimation.

Technique	Coeff. of deter. (R^2)/Correl. coef. (r)	RMSE	MAE
ABC/GBRT	0.9936/0.9968	1.5215	0.8786
ABC/MLP	0.9579/0.9787	7.8662	5.9564
LASSO	0.8173/0.9040	12.2594	10.4735

Table 7
Weights connected to each input variable in the ABC/GBRT approach.

Input variable	Weights
Temperature	0.84561313
Voltage	0.10432028
Current	0.05006659

Hence, according to Table 7 and Fig. 11, the most prominent input variable in SOC prediction in the ABC/GBRT approach is temperature, followed by voltage and current. In such a situation, the slight increase in temperature during discharge, associated with the exothermic nature of the discharge chemical reaction, explains the nonlinear relationship between the input variables and the SOC better than other model variables. In fact, temperature effects inside lithium-ion batteries play an important role in the proper battery management [64].

The main motivation for using the ABC/GBRT method here is that it is a procedure that combines the outputs of many weak classifiers to produce a powerful *method*, capable of reproducing the highly nonlinear behavior of the battery successfully. The LASSO approach is formulated for generalized linear regression models. Therefore, the higher the nonlinear behavior of the Li-ion cell, the lower the predictive ability of the LASSO method. The ABC/MLP technique can tackle nonlinear datasets. However, the existence of local minima in the error function makes training considerably difficult, since once a minimum is reached, the training stops even if the fixed rate of convergence has not been reached.

In summary, ABC/GBRT maintains the best accuracy, indicating the relatively strong robustness and performance of ensemble learning models for SOC prediction.

5. Conclusions

The objective of this study was to build a model to estimate the SOC of a power Li-ion cell using three machine learning techniques: GBRT, MLP and LASSO. The two first were hybridized with ABC to obtain their optimal parameters.

Training data is achieved by subjecting the battery to a special regime known as dynamic stress test (DST) described by the USABC. This type of test puts the battery through a set of charging and discharging currents related to the actual current used by an electric vehicle when driving.

Two electric measurements: current and voltage, and cell temperature were the input variables required to obtain the models. Thus, the developed models require no knowledge of the internal battery parameters to predict the SOC.

First, the GBRT model was developed using ABC to obtain the optimal-parameters of the GBRT model. The combination of ABC and GBRT replaces the traditional estimation of parameters based on empirical estimation. Furthermore, the ABC/GBRT approach with the ABC optimizer greatly improves the estimation capacity in comparison to that acquired only with a GBRT regressor without tuning parameters. The ABC/GBRT-based model was built using *Xgboost package* and *ABCoptim* functions from R software.

The second method for SOC prediction uses a feedforward ANN, Multilayer Perceptron (MLP), likewise combined with ABC. The ABC/MLP model was built using the ABC code and functions from the *Neural Network Toolbox* in MATLAB.

The third method was the LASSO regression technique, implemented using *Python scikit/learn* library package.

The predictive results confirm the enhanced performance of the ABC/GBRT-based model over the other models for SOC prediction. SOC errors are kept below 1%, 10% and 17% for ABC/GBRT, ABC/MLP and LASSO, respectively. The goodness of fit was calculated using R^2, obtaining values of 0.99, 0.95 and 0.81 for the three methods, respectively.

Furthermore, the related relevance of the input variables was determined in the SOC prediction. Specifically, the input variable *temperature* is proved as the most outstanding in the forecasting of the SOC. Next, this variable is followed in importance to estimate SOC by the *voltage* and *current* variables.

To sum up, this approach can be employed in distinct types of batteries satisfactorily. However, it is essential to keep in mind the specificities of each battery and the experimental surroundings. Thus, the ABC/GBRT model is a good response to the subject of SOC prediction.

Data availability

Data will be made available on request.

Acknowledgments

The authors wish to acknowledge the computational support provided by the Department of Mathematics at the University of Oviedo. Furthermore, this work was supported by the Spanish State Research Agency (Ministry of Science and Innovation) under Grant MCI-20-PID2019-110955RB-I00. Additionally, we would like to thank Anthony Ashworth for his revision of the English grammar and spelling of the manuscript.

References

[1] International Energy Agency (Ed.), Global EV Outlook, France, 2022.
[2] G. Pistoia, Electric and Hybrid Vehicles: Power Sources, Models, Sustainability, Infrastructure and the Market, Elsevier, Amsterdam, 2010.
[3] M. Lowe, S. Tokuoka, T. Trigg, G. Gereffi, Lithium-Ion Batteries for Electric Vehicles: The U.S. Value Chain, Center on Globalization, Governance & Competitiveness, Durham, UK, 2010.
[4] L. Lu, X. Han, J. Li, J. Hua, M. Ouyang, A review on the key issues for lithium-ion battery management in electric vehicles, J. Power Sources 226 (2013) 272–288.
[5] M. Corno, N. Bhatt, S. Savaresi, M. Verhaegen, Electrochemical model-based state of charge estimation for li-ion cells, IEEE Trans. Control Syst. Technol. 23 (1) (2015) 117–127.
[6] B. Wang, Z. Liu, S. Eben, S. Moura, H. Peng, State-of-charge estimation for lithium-ion batteries based on a nonlinear fractional model, IEEE Trans. Control Syst. Technol. 25 (1) (2017) 3–11.
[7] V. Pop, H.J. Bergveld, P.H.L. Notten, P.P.L. Regtien, State-of-the-art of battery state-of-charge determination, Meas. Sci. Technol. 16 (12) (2005) 93–110.
[8] K. Cheng, B. Divakar, H. Wu, K. Ding, H.F. Ho, Battery-management system (BMS) and SOC development for electrical vehicles, IEEE T. Veh. Technol. 60 (1) (2011) 76–88.
[9] B. Divakar, K.W.E. Cheng, H.J. Wu, J. Xu, H.B. Ma, W. Ting, K. Ding, W.F. Choi, B.F. Huang, C.H. Leung, Battery management system and control strategy for hybrid and electric vehicle, in: Proceedings of 3rd International Conference on Power Electronics Systems and Applications, IEEE Publisher, Hong Kong, 2009, pp. 20–22.
[10] A. Davide, Battery Management Systems for Large Lithium Ion Battery Packs, Artech House, Boston, 2010.
[11] L. Lavigne, J. Sabatier, J. Mbala, F. Guillemard, A. Noury, Lithium-ion open circuit voltage (OCV) curve modelling and its ageing adjustment, J. Power Sources 324 (30) (2016) 694–703.
[12] S. Tong, M.P. Klein, J.W. Park, On-line optimization of battery open circuit voltage for improved state-of-charge and state-of-health estimation, J. Power Sources 293 (20) (2015) 416–428.
[13] B. Pattipati, B. Balasingam, G. Avvari, K. Pattipati, Y. Bar-Shalom, Open circuit voltage characterization of lithium-ion batteries, J. Power Sources 267 (10) (2014) 317–333.
[14] B. Liu, M. Liu, X. Jiang, X. Tuo, H. Zhou, J. Ren, Design of battery management system based on DSP for BEV, in: Proceedings of 9th International Conference on Modelling, Identification and Control, IEEE Publisher, Kunming, China, 2017, pp. 857–862.
[15] W. Waag, D. Sauer, Adaptative estimation of the electromotive force of the lithium-ion battery after current interruption for an accurate state-of-charge and capacity determination, Appl. Energ. 111 (2013) 416–427.
[16] X. Hua, S. Li, H. Peng, A comparative study of equivalent circuit models for li-ion batteries, J. Power Sources 198 (2012) 359–367.
[17] A. Bartlett, J. Marcicki, S. Onori, G. Rizzoni, X. Guang, T. Miller, Electrochemical model-based state of charge and capacity estimation for a composite electrode lithium-ion battery, IEEE Trans. Control Syst. Technol. 24 (2) (2016) 384–399.
[18] X. Hu, S.E. Li, Y. Yang, Advanced machine learning approach for lithium-ion battery state estimation in electric vehicles, IEEE T. Transp. Electr. 2 (2) (2016) 140–149.
[19] W. He, N. Williard, C. Chen, M. Pecht, State of charge estimation for li-ion batteries using neural network modeling and unscented Kalman filter-based error cancellation, Int. J. Elec. Power 62 (2014) 783–791.
[20] C. Hametner, S. Jakubek, State of charge estimation for lithium ion cells: Design of experiments, nonlinear identification and fuzzy observer design, J. Power Sources 238 (2013) 413–421.
[21] M.A. Hannan, M.S. Hossain Lipu, A. Hussain, M.H.M. Saad, A. Ayod, Neural network approach for estimating state of charge of lithiumion battery using backtracking search algorithm, IEEE Access 6 (2018) 10069–10079.
[22] V. Klass, M. Behm, G. Lindbergth, A support vector machine-based state-of-health estimation method for lithium-ion batteries under electric vehicle operation, J. Power Sources 270 (2014) 262–272.
[23] J.N. Hu, J.J. Hu, H.B. Lin, X.P. Li, C.L. Jiang, X.H. Qiu, W.S. Li, State-of-charge estimation for battery management system using optimized support vector machine for regression, J. Power Sources 269 (2014) 682–693.
[24] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Ston, Classification and Regression Trees, Wadsworth and Brooks/Cole, Monterey, CA, 1984.
[25] R.S. Michalski, J.G. Carbonell, T.M. Mitchell, Machine Learning: An Artificial Intelligence Approach, Springer-Verlag, Berlin, 1983.
[26] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm, J. Global Optim. 39 (2007) 459–471.
[27] M. Farooq, Bee-Inspired Protocol Engineering, Springer-Verlag, Berlin, 2009.
[28] M. Dorigo, T. Stutzle, Ant Colony Optimization, Bradford Publisher, The MIT Press, Cambridge, MA, 2004.
[29] D. Simon, Evolutionary Optimization Algorithms, Wiley, New York, 2013.
[30] A.J. Shepherd, Second-Order Methods for Neural Networks: Perspectives in Neural Computing, Springer-Verlag, London, 1997.
[31] T. Hastie, R. Tibshirani, M. Wainwright, Statistical Learning with Sparsity: The Lasso and Generalizations, CRC Press, Boca Raton, FL, 2016.
[32] U.S. Department of Energy, Electric Vehicle Battery Test Procedures Manual, rev. 2, USABC, MI, USA, 1996.
[33] PEC, Technical Reference Manual SBTXX50, Leuven, Belgium, 2012.
[34] Izenman A.J., Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning, Springer, Berlin, 2013.
[35] V. Vapnik, Statistical Learning Theory, Wiley-Interscience, New York, 1998.
[36] J.H. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: a statistical view of boosting, Ann. Statist. 28 (2) (2000) 337–407.
[37] J.H. Friedman, Greedy function approximation: A gradient boosting machine, Ann. Statist. 29 (5) (2001) 1189–1232.
[38] J.H. Friedman, Stochastic gradient boosting, Comput. Statist. Data Anal. 38 (4) (2002) 367–378.
[39] R.E. Schapire, The boosting approach to machine learning an overview, in: D.D. Denison, M.H. Hansen, C.C. Holmes, B. Mallick, B. Yu (Eds.), Nonlinear Estimation and Classification, in: Lecture Notes in Statistics, vol. 171, Springer, New York, 2003, pp. 149–171.
[40] P. Bühlmann, T. Hothorn, Boosting algorithms: regularization , prediction and model fitting, Stat. Sci. 22 (4) (2007) 477–505.
[41] T. Hastie, R. Tibshirani, J.H. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer, Berlin, 2017.

[42] A. Mayr, H. Binder, O. Gefeller, M. Schmid, The evolution of boosting algorithms: From machine learning to statistical modelling, Method Inform. Med. 6 (1) (2014) 419–427.
[43] A. Mayr, H. Binder, O. Gefeller, M. Schmid, Extending statistical boosting: An overview of recent methodological developments, Method Inform. Med. 6 (2) (2014) 428–435.
[44] S.B. Taieb, R.J. Hyndman, A gradient boosting approach to the kaggle load forecasting competition, Int. J. Forecast. 30 (2) (2014) 382–394.
[45] J. Döpke, U. Fritsche, C. Pierdzioch, Predicting recessions with boosted regression trees, Int. J. Forecast. 33 (2017) 745–759.
[46] G. Ridgeway, Generalized boosted models: a guide to the GBM package, 2007, http://www.saedsayad.com/docs/gbm2.pdf.
[47] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report-TR06, Turkey, 2005.
[48] D. Karaboga, B. Akay, Algorithms simulating bee swarm intelligence, Artif. Intell. Rev. 31 (1) (2009) 68–85.
[49] S.A. Moghaddas, M. Nekoei, E.M. Golafshani, A. Behnood, M. Arashpour, Application of artificial bee colony programming techniques for predicting the compressive strength of recycled aggregate concrete, Appl. Soft Comput. 130 (2022) 109641.
[50] H. Hakli, The optimization of wind turbine placement using a binary artificial bee colony algorithm with multi-dimensional updates, Electr. Power Syst. Res. 216 (2023) 109094.
[51] L. Chen, T. Wu, Z. Wang, X. Lin, Y. Cai, A novel hybrid BPNN model based on adaptive evolutionary artificial bee colony algorithm for water quality index prediction, Ecol. Indic. 146 (2023) 109882.
[52] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artif. Intell. Rev. 42 (1) (2014) 21–57.
[53] V. Tereshko, A. Loengarov, Collective decision-making in honey bee foraging dynamics, Comput. Inform. Syst. 9 (3) (2005) 1–7.
[54] C. Blum, M. Blesa, A. Roli, Hybrid Metaheuristics: An Emerging Approach To Optimization, Springer-Verlag, Berlin, 2008.
[55] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice–Hall, Singapore, 1999.
[56] T.L. Fine, Feed-Forward Neural Network Methodology, Springer-Verlag, New York, 1999.
[57] M. Hassoun, Fundamentals of Artificial Neural Networks, MIT Press, Bradford Book, 1995.
[58] L. Wasserman, All of Statistics: A Concise Course in Statistical Inference, Springer, New York, 2003.
[59] D. Freedman, R. Pisani, R. Purves, Statistics, W.W. Norton & Company, New York, 2007.
[60] Comprehensive R archive network, 2013, ABCoptim: Implementation of artificial bee colony (ABC) optimization, R package ver. 0.13.11, Vienna, Austria.
[61] G. Ridgeway, Gbm: Generalized boosted regression models, r package version 2.1.1, 2017.
[62] D. Karaboga, Artificial Bee Colony (ABC) Algorithm, Intelligent Systems Research Group, 2017, http://mf.erciyes.edu.tr/abc/.
[63] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, É. Duchesnay, Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[64] S. Ma, M. Jiang, P. Tao, C. Song, J. Wu, J. Wang, T. Deng, W. Shang, Temperature effect and thermal impact in lithium-ion batteries: A review, Prog. Nat. Sci. Mater. Int. 28 (6) (2018) 653–666.