World Electric Vehicle Journal

Article

Data Analysis and Visualization Platform Design for Batteries Using Flask-Based Python Web Service

Zuyi Liang [1,2], Zongwei Liang [1,2], Yubin Zheng [1,2], Beichen Liang [1,2] and Linfeng Zheng [1,2,*]

[1] International Energy School, Jinan University, Zhuhai 519070, China; AkashiK1220@outlook.com (Z.L.); liang_zongwei888@163.com (Z.L.); zheng_yubin911@163.com (Y.Z.); polaris11200@163.com (B.L.)
[2] Institute of Rail Transportation, Jinan University, Zhuhai 519070, China
* Correspondence: lfzheng@jnu.edu.cn

Abstract: Battery operating data of electric vehicles is becoming increasingly quantified and complicated. A data analysis platform is necessary to excavate high-value battery status information for more efficient battery management. This paper proposes a Flask framework and Pyecharts-based lithium-ion data analysis and visualization platform. The design processes including the front-end and back-end frameworks, data preprocessing, data visualization, and data storage are elaborated. In the proposed data platform, a case study of battery state of charge estimation using different machine learning methods is demonstrated, and most of the estimation errors are less than 2.0%, highlighting the effectiveness of the platform.

Keywords: lithium-ion battery; visualization platform design; data analysis; machine learning method; state of charge (SOC) estimation

Citation: Liang, Z.; Liang, Z.; Zheng, Y.; Liang, B.; Zheng, L. Data Analysis and Visualization Platform Design for Batteries Using Flask-Based Python Web Service. *World Electr. Veh. J.* **2021**, *12*, 187. https://doi.org/10.3390/wevj12040187

Academic Editors: C. C. Chan, Yoichi Hori, James L. Kirtley, Jr., Joeri Van Mierlo, Myoungho Sunwoo and Xuhui Wen

Received: 15 September 2021
Accepted: 11 October 2021
Published: 14 October 2021

1. Introduction

The development of electric vehicles (EVs) has become a global consensus to solve the problems of environmental pollution and resource scarcities [1]. Affected by the fast development of battery management technology and sensor technology, battery operating data of EVs is becoming quantified and complicated [2]. However, the traditional offline inspection tools are still at the stage of functional checking, which is unable to support deep data processing and data mining [3]. The conventional battery management system (BMS) focuses on the real-time monitoring of the battery, and its display information is relatively limited [4–6]. The concepts of battery data analysis platforms and battery real-time monitoring platforms have been reported in some literature [7,8], but the detailed design of a data visualization platform with human-computer interaction functions for battery development and maintenance personnel is comparatively rare.

It is essential to introduce new and efficient data analysis technologies into battery systems of EVs and build a corresponding platform that can dig out high-value battery status information, significantly improving battery management technology [9], as well as being helpful for EVs companies.

Due to its strong compatibility, Python is suitable for building a data analysis platform [10]. Therefore, this paper proposes a Python-based battery data analysis and visualization platform design scheme, and then applies the platform to estimate the state of charge (SOC) of lithium-ion batteries.

Furthermore, several SOC estimation methods have been developed by researchers in recent years. The various methods which have been proposed to estimate the SOC can be categorized into the ampere-hour counting method (AHCM) [11], the model-based method (MBM) [12–16], and the machine learning-based SOC inference method (MLM) [17–20]. Through the comparison in Refs. [21,22], MLM and MBM are considered to have high accuracy and fast convergence in the SOC estimation. In order to unify the technical route, this paper uses MLM with Python.

The remainder of this paper is organized as follows. Section 2 introduces a comprehensive designing scheme, including overall framework, front-end and back-end technology applications and construction processes, data preprocessing, visualization and storage methods. Section 3 demonstrates how to apply this platform to actual projects, such as dynamic data transmission, real-time drawing of charts, and SOC estimation based on MLM, and gives a brief introduction to the various modules of the platform. Section 4 finally draws the conclusions and points out the area worthy of improvement.

2. System Design

2.1. Framework

The platform architecture, as shown in Figure 1, is mainly divided into three layers including the application layer, service layer, and data layer. As the bottom layer, the data layer can process, store, and analyze battery data. Based on Flask and HTTP, the service layer can support asynchronous response and dynamic interaction. The web application layer can realize customer-oriented data display and query. Among them, the service layer is called by the application layer, and after receiving relevant instructions, the data layer transmits data to the application layer through the service layer, which meets the platform's needs for stability and scalability.

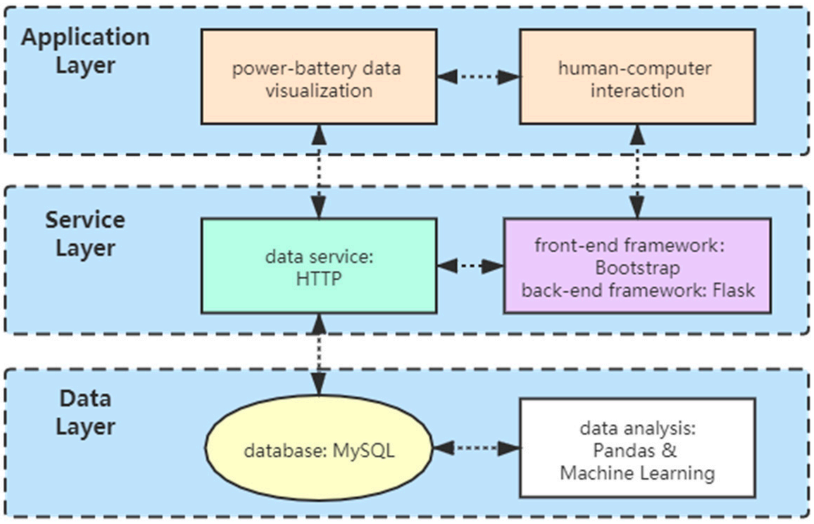

Application Layer
power-battery data visualization
human-computer interaction
Service Layer
data service: HTTP
front-end framework: Bootstrap back-end framework: Flask
Data Layer
database: MySQL
data analysis: Pandas & Machine Learning

Figure 1. The architecture diagram of this platform.

2.1.1. Front-End Construction

The design of this web platform follows the design of web page structure and content by html, style by css, and script by JavaScript [23], which realizes the separation of structure, performance, and behavior, and satisfies the W3C industry standards exactly. The code of diagrams always appears as the form of Echarts in the front end, which is suitable to be stored in the script file, so charts of each same type should be divided into different js files. For the sake of specification, the inline code of styles in html are separated and changed into independent and unified css files. Among them, the css, js, images, and other folders should be placed in a static folder, and finally the corresponding scripts and style files are imported to implement the calling of those programs, as shown in Figure 2.

In order to optimize the front-end layout and be compatible with multiple terminals and resolutions, a responsive layout Bootstrap framework is built to ensure a fixed container whose width does not change with the screen size, a drop-down menu template, a navigation template for tabs, and the grid system col-lg-x etc.

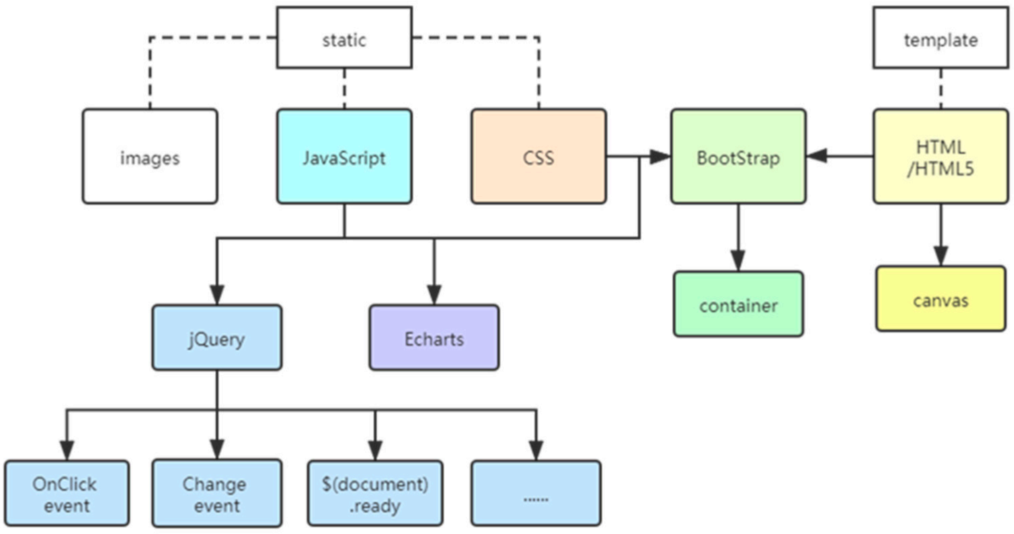


Figure 2. The web front-end architecture diagram.

For the purpose of enhancing the dynamics of the interface, the platform uses the canvas element to set the background of floating fluorescent particles, which uses the *$ (document).ready* method to load a small animation that represents the page loading and uses the *setInterval* function and the Date function to design the system real-time display function. Moreover, the charts query function is realized through the change event of the drop-down selection box and the onclick event of the tab, which helps to save the display space of similar content, as well as improve the simplicity, beauty and human-computer interaction of the web page.

2.1.2. Back-End Construction

Flask is a Python-based micro web framework with strong compatibility and high scalability. The Flask framework is shown in Figure 3. It mainly uses the Werkzeug toolkit that supports routing integration and the jinja2 template engine with sandbox execution functions. Independent developers can design personalized websites flexibly according to their own project requirements, which meets the high-efficiency requirements of the later ops of the data analysis platform [24].

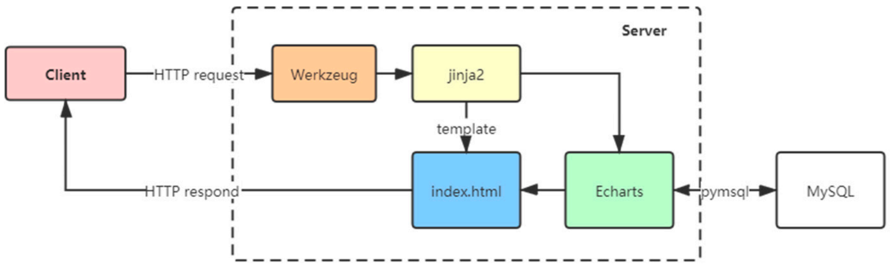


Figure 3. Webpage response flow chart under the Flask framework.

The basic mode of Flask is to assign and bind different view functions to different access paths through the route decorator. Therefore, the routing technology based on Werkzeug is beneficial to save the step of homepage navigation, as well as ensure that a unique URL is generated, so that users can directly access the interface they care about.

When the *app.run* function under the flask framework is executed, the user visits the URL "/test" that is marked on the backend, and then can see a simple login interface. After typing the correct account and password into the input box and clicking the "Submit" button, the page will jump to a new URL "login?name = name &pwd = pwd". Meanwhile, the index.html in the template folder is loaded into the web interface. The technical principle is that Jinja2 redirects the URL login path to the large-screen template “index” that has been rendered, for automatic escape. Thus, according to the HTTP protocol, the user can access the webpage version of this data analysis platform easily, just after a simple login verification operation in the PC browser. At the same time, only the internal personnel with the password can log in to the large visualization screen to view the battery status information, which improves the security and reliability of the platform to a certain extent.

2.2. Data Preprocessing

Lithium-ion battery test data under different working conditions is applied in this work. In the step of data import, it is usually quickly realized by the *read* function of the Pandas module. This function saves the specified data set into a unique data structure DataFrame with row and column index, which is convenient for data display and processing.

In order to standardize the data, using data type conversion functions such as *float*, *int*, etc., could unify the data types. For real-time battery operating data, it is always expressed in the form of floating-point numbers.

Next, different abnormal data requires different processing approaches. For missing values NaN, the Pandas library can use the *dropna* function to delete the rows and columns of the incomplete data, or use the *fillna* function to fill in. Additionally, common replacement values include the mean value, median value, and 0, as well as using some interpolation methods when in pursuit of high precision. For the error value, the processing method is the same as the above, but it is necessary to calculate the difference between the connected data through the *df.diff* function in advance, so as to determine whether there is a data abnormality. For duplicate data, the *df.drop duplicates* function can delete the specified duplicate items and keep the data item that appears at the first time.

Therefore, in the face of various data types, skilled use of Pandas modules can efficiently complete data cleaning, extract the most critical information, and eventually obtain more accurate charge and discharge experimental data classified by temperature and charge rate. Among them, the *df.describe* function can be used to quickly view the comprehensive analysis and statistics of the data set, which helps to prepare for the subsequent data visualization and mining modeling.

2.3. Data Visualization

According to the characteristics of the lithium-ion battery test data, this platform uses the Pyecharts library to draw various charts to complete the preliminary data visualization and improve the legibility of big data.

Charging rate is a common measure of charging speed, which refers to the current value required by the battery to discharge its rated capacity within a specified time. It is equal to the multiple of the battery’s rated capacity in data value, and is usually represented by the letter C. The discharging rate can be obtained in the same way.

The charging and discharging curves at various temperatures and charging rates are more appropriately drawn as a broken line graph. Using the Line function can pass the time parameter onto the x-axis and the gradually changing voltage value during charging and discharging onto the y-axis. Furthermore, using the Timeline function can call the carousel chart function, which is conducive to combine multiple similar charts and get an intuitive and convenient data comparison, as shown in Figure 4. Among them, the player button can realize the automatic carousel charts display, enhancing the dynamic and interactive visualization. The figures like “0.05C” and “0.33C” below belong to the control buttons of Timeline, which can also switch charts by choosing different charging rates.

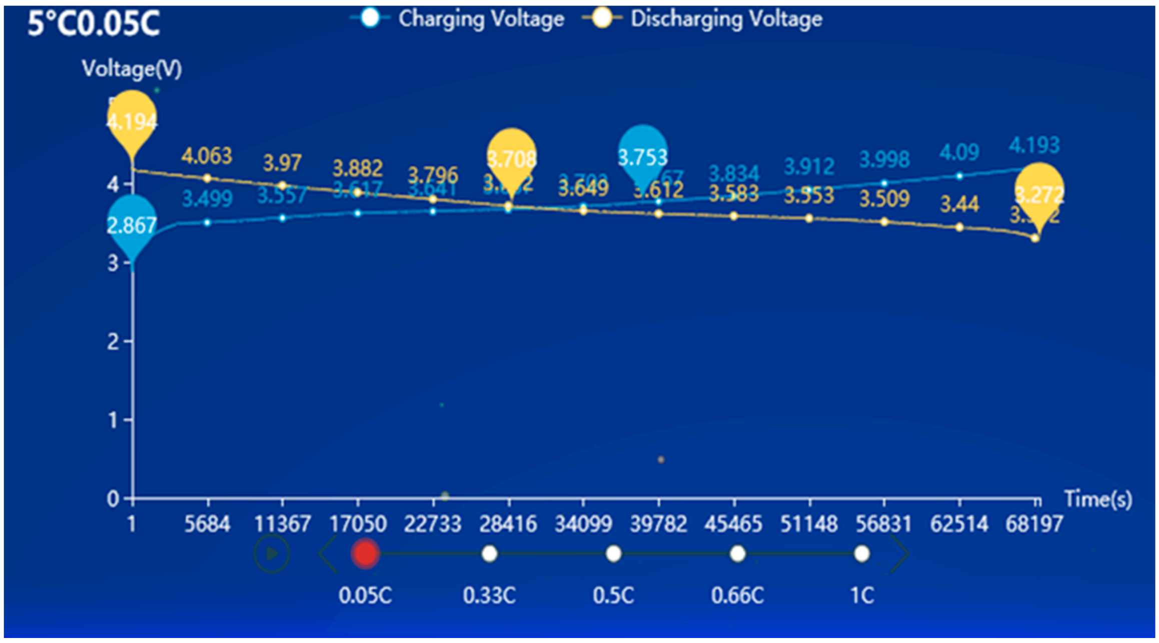


Figure 4. Example of carousel line charts of charge and discharge curves.

If there is no need to pass in a large amount of data or involve a percentage, it is more appropriate to use a histogram for displaying the difference of total data by calling the Bar function. Choosing the tile mode for visualization and using different colors and heights to mark the battery data can surely enhance the intuitiveness of the data. After comparing the chart data, it can be found that different parameters are affected by the charging rate. The length of charging time is negatively associated with the charging rate, while the size of the charging current has a positive correlation with the charging rate.

In addition, the chart comes with convenient multi-dimensional legend switches. As shown in Figure 5, clicking the built-in legend switch can turn off or turn on the display of a certain data in the chart, which helps users to focus on some data series at any time and remove objects that they don't care about. It reflects the high interactivity of the interface and the personalization of the platform.

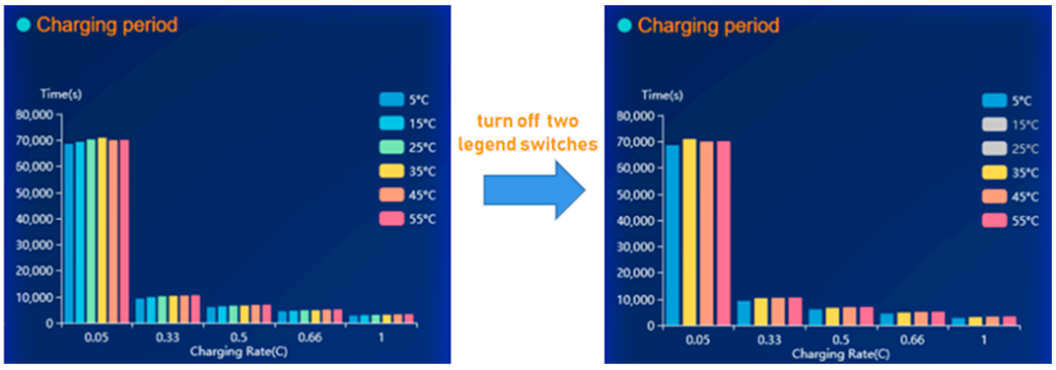


Figure 5. A histogram in tiled mode and its visualization when two of the legend switches are turned off.

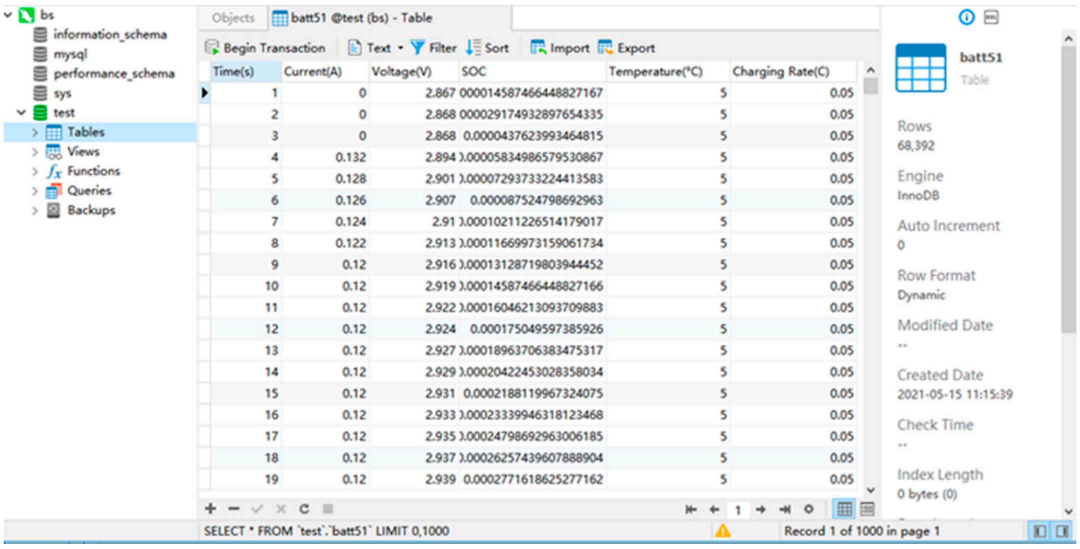
In the end, the Page function can integrate all the above charts, and organize them on the same page, as well as using the drag-and-drop function to design its layout. Among them, the personalized layout can be stored as a json file through the *save-json* function, and then the *save resize html* function can be executed to re-call the layout, namely re-render the html webpage. Furthermore, using functions such as *find all* or *select* under the BeautifulSoup library can directly arrange the layout.

2.4. Data Storage

A complete data analysis platform should set up a corresponding database, which is conducive to the realization of data centralized management and interface sharing, data security, repair faults, and system development easily [25]. The MySQL with open-sources and high-performance that can provide safe and reliable data storage support for a web platform, as well as improving data independence, is employed in this work.

Among them, sqlalchemy is an open source tool library that provides a data mapping model in the Python environment. Combining the functions of sqlalchemy and Pandas library leads to a simple but efficient way to interact with the database; *read sql* function for reading, and *to sql* function for writing.

The first parameter of the *to sql* function is often a custom new data table name, and its connection to the database is successfully established by the "engine" parameter through the sqlalchemy library. Another parameter "if exists" should be set to "replace", which could directly overwrite the original data table, in order to avoid data being repeatedly written into the same table and reduce data redundancy. An example of a written data table is shown in Figure 6. The *read sql* function generally has only two parameters, including the "engine" mentioned above and the SQL statement used as a query to obtain a specific row and column in the database.



Time(s)	Current(A)	Voltage(V)	SOC	Temperature(°C)	Charging Rate(C)
1	0	2.867	000014587466448827167	5	0.05
2	0	2.868	000029174932897654335	5	0.05
3	0	2.868	0.0000437623993464815	5	0.05
4	0.132	2.894	).00005834986579530867	5	0.05
5	0.128	2.901	).00007293733224413583	5	0.05
6	0.126	2.907	0.000087524798692963	5	0.05
7	0.124	2.91	).00010211226514179017	5	0.05
8	0.122	2.913	).00011669973159061734	5	0.05
9	0.12	2.916	).00013128719803944452	5	0.05
10	0.12	2.919	).00014587466448827166	5	0.05
11	0.12	2.922	).00016046213093709883	5	0.05
12	0.12	2.924	0.000175049597385926	5	0.05
13	0.12	2.927	).00018963706383475317	5	0.05
14	0.12	2.929	).00020422453028358034	5	0.05
15	0.12	2.931	0.0002188119967324075	5	0.05
16	0.12	2.933	).00023339946318123468	5	0.05
17	0.12	2.935	).00024798692963006185	5	0.05
18	0.12	2.937	).00026257439607888904	5	0.05
19	0.12	2.939	0.0002771618625277162	5	0.05

```
SELECT * FROM `test`.`batt51` LIMIT 0,1000
Record 1 of 1000 in page 1
```

batt51
Table

Rows	68,392
Engine	InnoDB
Auto Increment	0
Row Format	Dynamic
Modified Date	--
Created Date	2021-05-15 11:15:39
Check Time	--
Index Length	0 bytes (0)

Figure 6. Example of written data table.

3. Application Results

3.1. Asynchronous Refresh Based on Ajax

ECharts is an open source visualization chart library based on Javascript, which has favorable characteristics of dual rendering engines, high compatibility, vivid and elegant responsive design, and support of data refresh asynchronously [26]. The predecessor of Pyecharts mentioned above is ECharts, which means that the two visualization libraries are actually the same. Therefore, just using the front-end technology to modify the html document directly can realize the dynamic drawing of charts. It can use the inline writing of css style to set its width and height, and set an empty list for incremental data in the script, according to the data loading feature of Echarts.

Ajax is a web data interaction technology independent of the server and browser under JavaScript. It can transmit data with the web server in small batches and quickly load the increment in the partial interface [27] instead of refreshing the entire page. Thus, the asynchronous update of the web page is realized. Its core timing polling function mainly relies on the set Interval function of js. The procedure is set to trigger an Ajax request every 1500 milliseconds, which connect with the URL address of the corresponding route and obtain the json format data interactively in the back-end Flask framework. If it is successfully obtained, the code of Ajax function would be executed. Then the setOption item of the Echarts chart would be updated; namely, the battery incremental data would be passed in in the form of name and value pairs. Finally, both the data transmission between the back-end and the front-end browser and the real-time dynamic drawing of the chart are realized through Ajax, as shown in Figure 7.

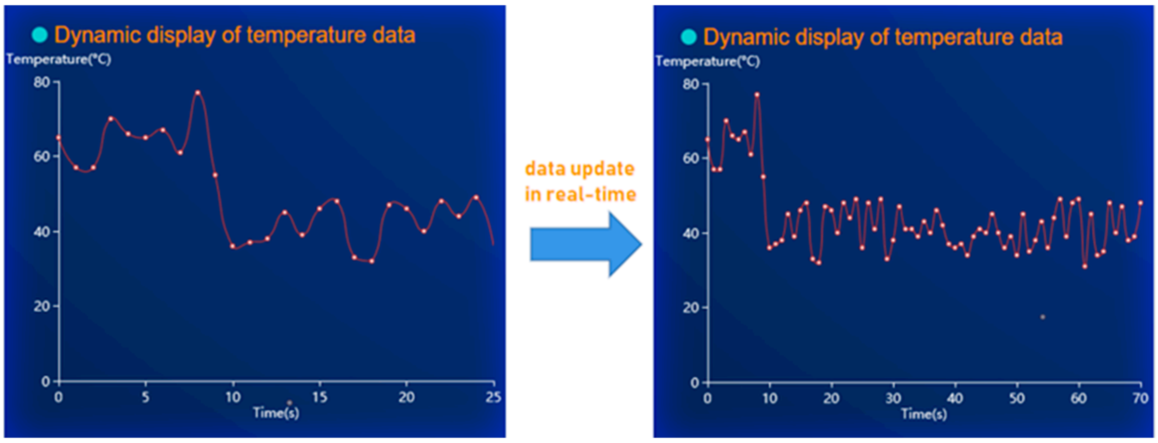


Figure 7. The temperature curve updated in real time.

3.2. SOC Estimation Based on Machine Learning Methods

In-depth data analysis is not only to list the relationships between data, but also to complete data mining; that is, to model data and estimate future data. The core parameter SOC is essential to predict the cruising range of EVs. The reported SOC estimation methods mainly include four types: the ampere-hour integration method, the open circuit voltage (OCV) method, the model-based method, and the data-driven method. The ampere-hour integration method is more reliable when the current acquisition accuracy is guaranteed, but there is still a large error accumulation. The OCV method needs high experimental requirements, and the use of the OCV-SOC curve for estimation has certain limitations. The model-based method is robust, but the development cost is quite high, and the implementation steps are much more complicated [28]. When the mathematical model of the object is unknown, it is suitable to adopt data-driven methods. Because they mainly rely on the mathematical mapping between input and output to create predictive models, they eliminate the consideration about the internal situation of the system.

In a data analysis platform, the SOC estimation problem can be transformed into a regression analysis, since the concept "regression" refers to a method of using approximation technology to predict the true value. In the related definition of machine learning, the classification model is a discretized regression model, thus a supervised classifier can be applied to estimate the SOC.

It is known that there are many kinds of classification algorithms in sklearn, which is a powerful open source machine learning library [29]. This paper chooses five commonly used algorithms with excellent generalization ability, as shown in Table 1.

Table 1. Five commonly used classifiers.

Algorithm	Module	Function
Linear Regression	linear model	LinearRegression
K-NearestNeighbor	neighbors	KNeighborsRegressor
Support Vector Machine	svm	SVC
Decision Tree	tree	DecisionTreeRegressor
Random Forest	ensemble	RandomForestRegressor

For estimating battery SOC, this platform imports the sklearn module, divides the data previously stored in the datadframe into two types of sample, features and labels, and then uses the *train test split* function to train and test the model according to a certain proportion. It is worth noting that only the SOC that is the value to be predicted should be retained as the label item, and the remaining categories of data should be treated as sample feature sets.

After the selection and testing of hyperparameters, it can be found that the linear regression, the k-nearest neighbor, and the decision tree algorithm have good fitting results and high prediction accuracy just with the default values to process the battery data. The other two classifiers require parameter tuning.

Taking the support vector machine (SVM) as an example, it is crucial to select a suitable kernel function [30]. By debugging, it is found that the training effect of the radial basis kernel function *RBF* is better than the polynomial kernel function *Poly* and the linear kernel function *Linear*. In addition, the larger the penalty value *C* has the higher prediction accuracy, but also leads to weaker generalization ability. After equalization, *C* is set to 20 in this program, and the insensitivity coefficient gamma as a kernel function is set to "scale". The SOC estimation results using the SVM-based method under different temperatures and current rates are depicted in Figure 8, where most of estimation errors are less than 2.0%.

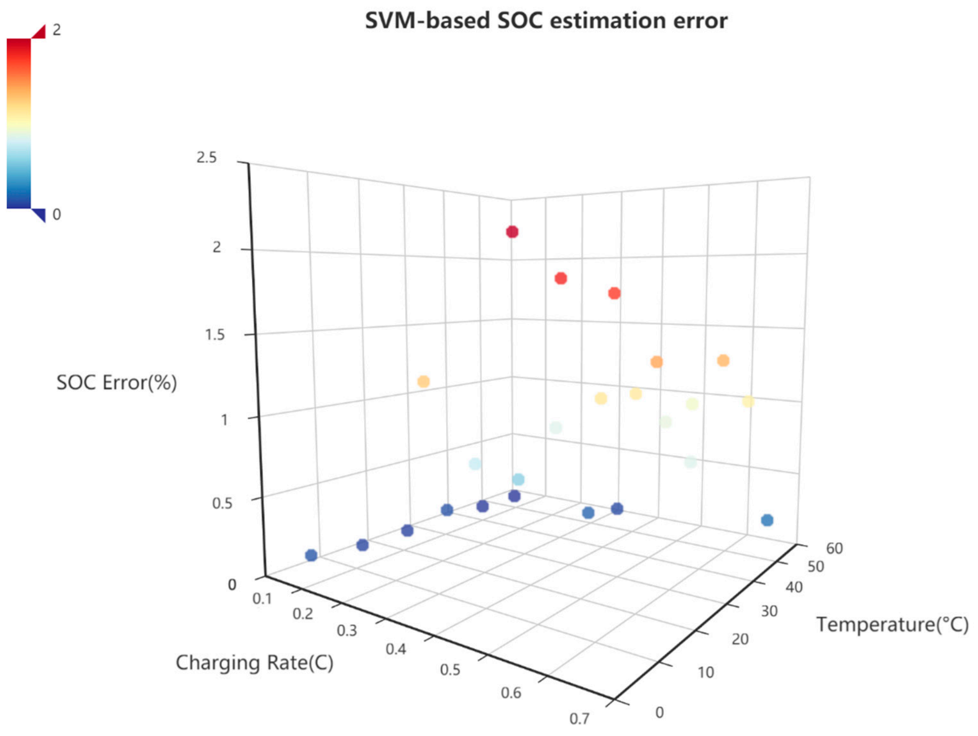


Figure 8. Estimation errors of SVM-based SOC prediction in different working conditions.

Random Forest (RF) belongs to a type of ensemble algorithm. Its core parameter is the number of classifiers *n estimators*, which, if set to a small number, will lead to under-fitting, while a large number will increase computation burden. After testing, a moderate value of 20 is selected. The SOC estimation results under different temperatures and current rates are depicted in Figure 9, where the estimation errors are limited in a narrow error band of 1.5%.

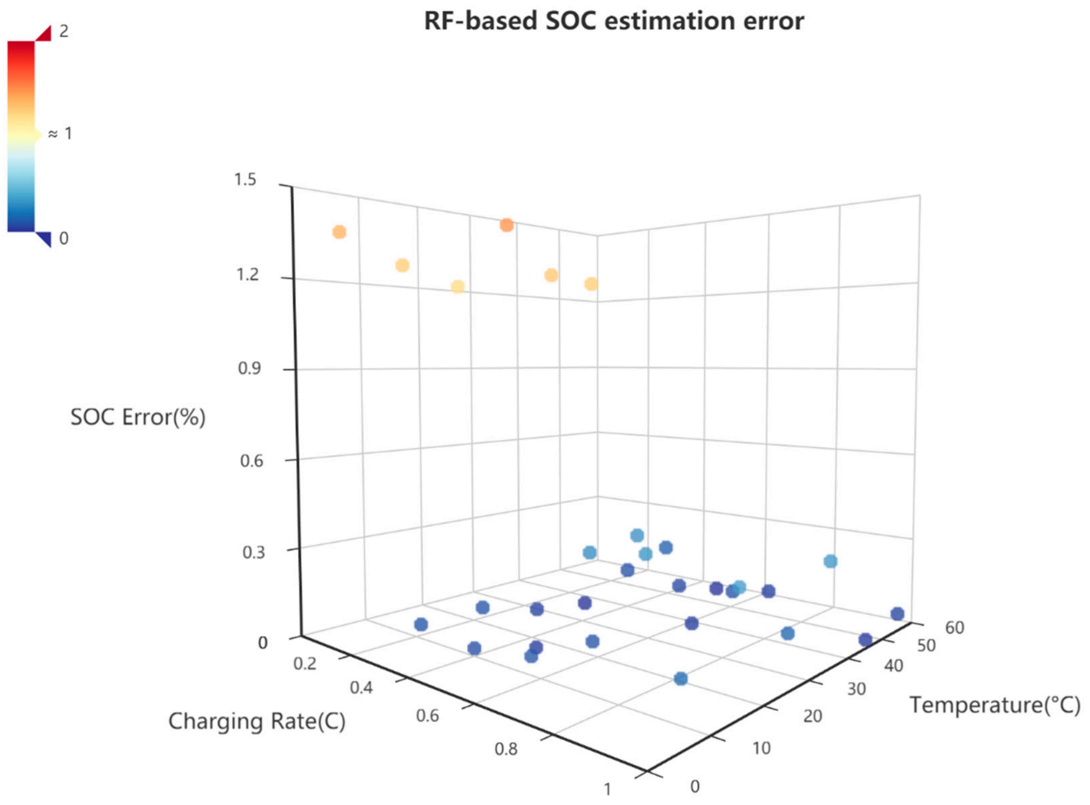


Figure 9. Estimation errors of RF-based SOC prediction in different working conditions.

Through adjusting the algorithm parameters, the previously separated sample feature sets for test are imported into the trained model. The prediction performance of the model is judged by calculating the difference between the predicted value of the model and the sample label set (here is SOC data) for the test. Next, the *sort* function is used to restore the data set disrupted by the *train test split* function as the incremental sorting. Then, the sorted data is converted back to the list format that is suitable for graph drawing, and drawn by using the Pyecharts library. At last, combined with the aforementioned Line function, the trained prediction data and test set data are visualized as fitting curves, and the tab is introduced and bound to the onclick event, so that charts would be integrated into the large web screen.

3.3. Overview of the Proposed Data Platform

The overview of the proposed battery data analysis platform is shown in Figure 10. Based on the dynamic blue particle background, the title is displayed in bold white, and the loading animation would be displayed in the upper left corner before the page is loaded. After that, it can be seen that there are three block diagrams on the left, which are two histograms showing the variation of charging time and charging current values, and the last one is a dynamic line graph of charging temperature that would be refreshed with system time. The middle box is composed of two large diagrams: a top frame and a below frame. The top frame contains a digital table (values of charging rate, working current, and battery temperature) that can be displayed in real time, and a carousel atlas of battery charging and discharging curves with a pull-down selection to realize query function. The frame below also includes carousel charts with query function, but its content is about the change of SOC curves under different charging rates and temperatures. The upper right corner is the calendar box displaying the real system time and date. Then, the first right box is a comparison atlas of the OCV-SOC curves with the *select* query function, and the second right is the combination of SOC estimation curves that could be selected by tab navigation buttons to use various machine learning algorithms, and by a pull-down selection to complete the temperature query.

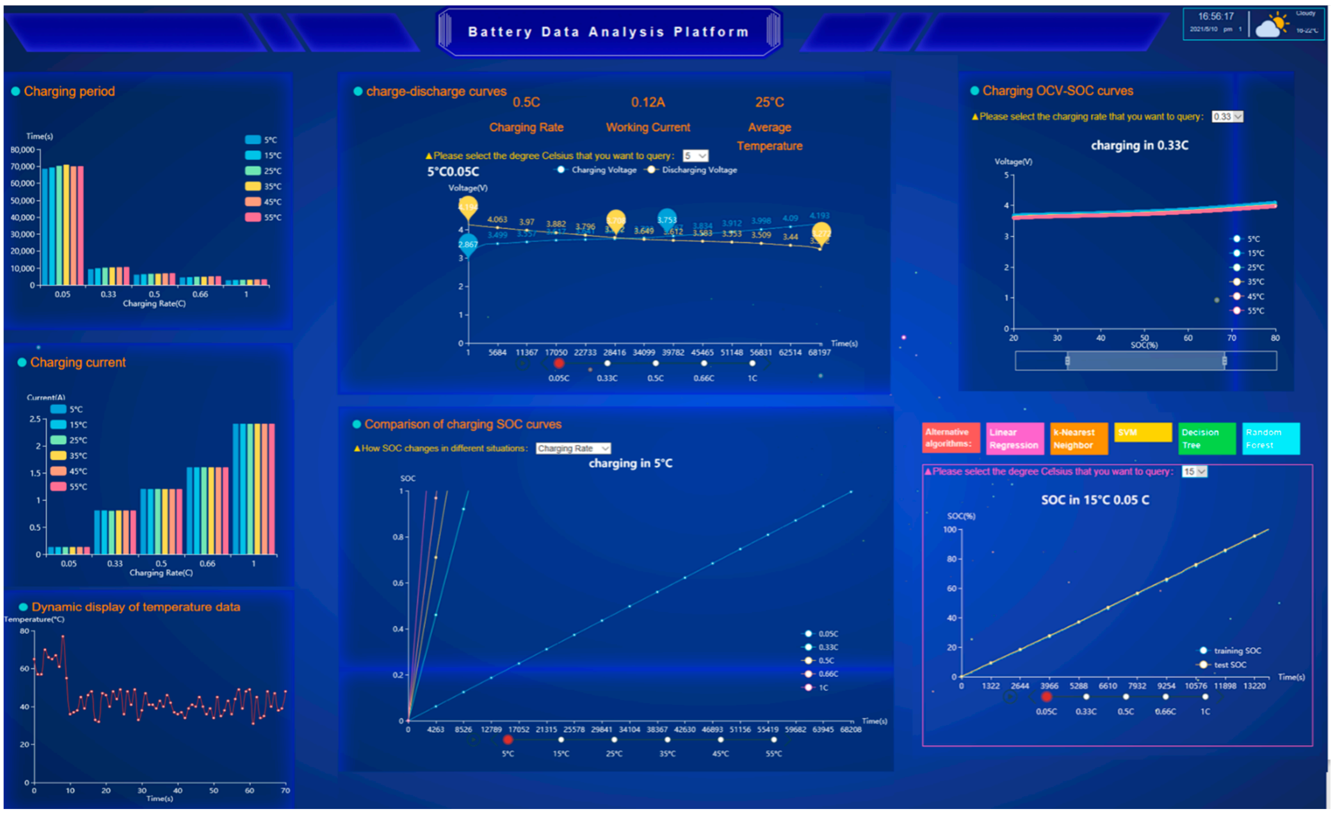


Figure 10. Overview of data analysis web platform for power battery.

4. Conclusions

This paper proposes a Python-based lithium-ion battery data analysis and visualization platform, which has higher human-computer interaction and data legibility for more effective battery management and data analysis. The detailed design processes including the front-end and back-end frameworks, data preprocessing, data visualization, and data storage are elaborated. A vivid visualization design is used to show the relationship between data characteristics of the battery and their influencing factors. In the proposed data platform, it is easy to use different machine learning methods to deeply analyze the battery operating data and estimate battery SOC under various working conditions. Most of the estimation errors are less than 2.0%, which highlights the functional feasibility of the data platform.

At present, on one hand, the proposed data platform is only applied to estimate battery SOC using different machine learning methods. On the other hand, it just used the simulation data dynamically generated by the back-end program. In future work, we would consider cooperating with battery-related hardware projects, through the serial port that helps realize the real dynamic data transmission, real-time data analysis and visualization. More artificially intelligent methods would be integrated to the platform, and more status information, such as state of health (SOH) and state of power (SOP), would be predicted using the platform.

Author Contributions: Methodology, software, validation, formal analysis, visualization, writing—original draft preparation, Z.L. (Zuyi Liang); investigation, data curation, Z.L. (Zongwei Liang), Y.Z. and B.L.; conceptualization, funding acquisition, supervision, writing—review and editing, L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Guangdong Basic and Applied Basic Research Foundation (Grant No. 2019A1515012210) and the National Natural Science Foundation of China (Grant No. 52102424).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sun, X.; Li, Z.; Wang, X.; Li, C. Technology development of electric vehicles: A review. *Energies* **2020**, *13*, 90. [CrossRef]
2. Li, S.; He, H.; Li, J. Big data driven lithium-ion battery modeling method based on SDAE-ELM algorithm and data pre-processing technology. *Appl. Energy* **2019**, *242*, 1259–1273. [CrossRef]
3. Tingfeng, D. Research and Design on Electric Vehicle Power Battery Assembly Testing System. Master's Thesis, Chongqing University, Chongqing, China, 2016.
4. Rahmawatie, B.; Sutopo, W.; Fahma, F.; Purwanto, A.; Nizam, M.; Louhenapessy, B.B.; Mulyono, A.B. Designing framework for standardization and testing requirements of battery management system for electric vehicle application. In Proceedings of the 2017 4th International Conference on Electric Vehicular Technology (ICEVT), Bali, Indonesia, 2–5 October 2017; IEEE: New York, NY, USA, 2018.
5. He, H.; Xiong, R.; Peng, J. Real-time estimation of battery state-of-charge with unscented Kalman filter and RTOS μCOS-II platform. *Appl. Energy* **2016**, *162*, 1410–1418. [CrossRef]
6. Lee, K.S.; Moon, C.J.; Kim, T.G.; Jeong, M.S.; Kim, S.M.; Park, B.J. A development of battery monitoring and management system. In Proceedings of the 2012 IEEE Vehicle Power and Propulsion Conference, Seoul, Korea, 9–12 October 2012; IEEE: New York, NY, USA, 2013.
7. Teng, Z. Building and Applying Online/Offline Data Platform in Data Analysis of Battery with Python Programming Language. Master's Thesis, Beijing Jiaotong University, Beijing, China, 2016.
8. Lee, C.H.; Wu, C.H. Collecting and mining big data for electric vehicle systems using battery modeling data. In Proceedings of the 2015 12th International Conference on Information Technology-New Generations, Las Vegas, NV, USA, 13–15 April 2015; IEEE: New York, NY, USA, 2015.
9. Wu, B.; Chen, Q.-Y.; Liu, B.; Wei, K.-X. Design of the Battery Management System Monitoring Platform. *Electr. Meas. Instrum.* **2013**, *50*, 112–116.
10. Song, Y.; Huang, R.; Wang, J. Research on Data Analysis and Visualization Platform Based on Python. *Mod. Inf. Technol.* **2019**, *3*, 7–9.
11. Zhang, Y.; Song, W.; Lin, S.; Feng, Z. A novel model of the initial state of charge estimation for $LiFePO_4$ batteries. *J. Power Sources* **2014**, *248*, 1028–1033. [CrossRef]
12. Paschero, M.; Storti, G.L.; Rizzi, A.; Mascioli, F.M.F.; Rizzoni, G. A novel mechanical analogy-based battery model for SoC estimation using a multicell EKF. *IEEE Trans. Sustain. Energy* **2016**, *7*, 1695–1702. [CrossRef]
13. He, Z.; Chen, D.; Pan, C.; Chen, L.; Wang, S. State of charge estimation of power Li-ion batteries using a hybrid estimation algorithm based on UKF. *Electrochim. Acta* **2016**, *211*, 101–109.
14. Wassiliadis, N.; Adermann, J.; Frericks, A.; Pak, M.; Reiter, C.; Lohmann, B.; Lienkamp, M. Revisiting the dual extended Kalman filter for battery state-of-charge and state-of-health estimation: A use-case life cycle analysis. *J. Energy Storage* **2018**, *19*, 73–87. [CrossRef]
15. Ren, L.; Zhu, G.; Kang, J.; Wang, J.V.; Luo, B.; Chen, C.; Xiang, K. An algorithm for state of charge estimation based on a single-particle model. *J. Energy Storage* **2021**, *39*, 102644. [CrossRef]
16. Jiang, C.; Wang, S.; Wu, B.; Fernandez, C.; Xiong, X.; Coffie-Ken, J. A state-of-charge estimation method of the power lithium-ion battery in complex conditions based on adaptive square root extended Kalman filter. *Energy* **2021**, *219*, 119603. [CrossRef]
17. Tong, S.; Lacap, J.H.; Park, J.W. Battery state of charge estimation using a load-classifying neural network. *J. Energy Storage* **2016**, *7*, 236–243. [CrossRef]
18. Tian, J.; Xiong, R.; Shen, W.; Shen, W.; Lu, J. State-of-charge estimation of $LiFePO_4$ batteries in electric vehicles: A deep-learning enabled approach. *Appl. Energy* **2021**, *291*, 116812. [CrossRef]
19. Ng, M.F.; Zhao, J.; Yan, Q.; Conduit, G.J.; Seh, Z.W. Predicting the state of charge and health of batteries using data-driven machine learning. *Nat. Mach. Intell.* **2020**, *2*, 161–170. [CrossRef]
20. Deng, Z.; Yang, L.; Cai, Y.; Deng, H.; Sun, L. Online available capacity prediction and state of charge estimation based on advanced data-driven algorithms for lithium iron phosphate battery. *Energy* **2016**, *112*, 469–480. [CrossRef]
21. Shrivastava, P.; Soon, T.K.; Bin Idris, M.Y.I.B.; Mekhilef, S. Overview of model-based online state-of-charge estimation using Kalman filter family for lithium-ion batteries. *Renew. Sustain. Energy Rev.* **2019**, *113*, 109233. [CrossRef]
22. Wang, Y.; Tian, J.; Sun, Z.; Wang, L.; Xu, R.; Li, M.; Chen, Z. A comprehensive review of battery modeling and state estimation approaches for advanced battery management systems. *Renew. Sustain. Energy Rev.* **2020**, *131*, 110015. [CrossRef]
23. Taivalsaari, A.; Mikkonen, T.; Systä, K.; Pautasso, C. Web User Interface Implementation Technologies: An Underview. In Proceedings of the 14th International Conference on Web Information Systems and Technologies (WEBIST 2018), Seville, Spain, 18–20 September 2018; SCITEPRESS–Science and Technology Publications, Lda.: Setubal, Portugal, 2018; pp. 27–136.

24. Vogel, P.; Klooster, T.; Andrikopoulos, V.; Andrikopoulos, V.; Lungu, M. A low-effort analytics platform for visualizing evolving Flask-based Python web services. In Proceedings of the 2017 IEEE Working Conference on Software Visualization (VISSOFT), Shanghai, China, 18–19 September 2017; IEEE: New York, NY, USA, 2017.
25. Opmane, B. Web Application Development Using the Latest Possibilities of MySQL Database Technologies. Master's Thesis, Riga Technical University, Riga, Latvia, 2018.
26. Li, D.; Mei, H.; Shen, Y.; Su, S.; Zhang, W.; Wang, J.; Zu, M.; Chen, W. ECharts: A declarative framework for rapid construction of web-based visualization. *Vis. Inform.* **2018**, *2*, 136–146. [CrossRef]
27. Zhao, H.-G. The realization of ECharts technology with real-time refresh of dynamic data under the support of Ajax technology. *Electron. Technol.* **2018**, *47*, 25–27.
28. Andre, D.; Appel, C.; Soczka-Guth, T. Dirk Uwe Sauer. Advanced mathematical methods of SOC and SOH estimation for lithium-ion batteries. *J. Power Sources* **2013**, *224*, 20–27. [CrossRef]
29. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, A.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
30. Chandran, V.; Patil, C.K.; Karthick, A.; Ganeshaperumal, D.; Rahim, R.; Ghosh, A. State of charge estimation of lithium-ion battery for electric vehicles using machine learning algorithms. *World Electr. Veh. J.* **2021**, *12*, 38. [CrossRef]