Hindawi
Complexity
Volume 2022, Article ID 8843000, 20 pages
https://doi.org/10.1155/2022/8843000

WILEY | Hindawi

Research Article

Modelling on Car-Sharing Serial Prediction Based on Machine Learning and Deep Learning

Nihad Brahimi,[1] **Huaping Zhang**,[1] **Lin Dai**,[1] **and Jianzi Zhang**[2]

[1]*School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China*
[2]*Chang'an Automobile Co., Ltd., Chongqing, China*

Correspondence should be addressed to Huaping Zhang; kevinzhang@bit.edu.cn

Received 19 August 2020; Revised 29 November 2021; Accepted 7 December 2021; Published 5 January 2022

Academic Editor: Rosa M. Benito

The car-sharing system is a popular rental model for cars in shared use. It has become particularly attractive due to its flexibility; that is, the car can be rented and returned anywhere within one of the authorized parking slots. The main objective of this research work is to predict the car usage in parking stations and to investigate the factors that help to improve the prediction. Thus, new strategies can be designed to make more cars on the road and fewer in the parking stations. To achieve that, various machine learning models, namely vector autoregression (VAR), support vector regression (SVR), eXtreme gradient boosting (XGBoost), k-nearest neighbors (kNN), and deep learning models specifically long short-time memory (LSTM), gated recurrent unit (GRU), convolutional neural network (CNN), CNN-LSTM, and multilayer perceptron (MLP), were performed on different kinds of features. These features include the past usage levels, Chongqing's environmental conditions, and temporal information. After comparing the obtained results using different metrics, we found that CNN-LSTM outperformed other methods to predict the future car usage. Meanwhile, the model using all the different feature categories results in the most precise prediction than any of the models using one feature category at a time

1. Introduction

Predicting the future is considered as one of the most challenging tasks in applied sciences. Computational and statistical methods are used for deducting dependencies between past and future observed values in order to build effective predictors from historical data. Transport answers to people's desire to participate in different activities in different places [1]. Cars have become a part of the mobility ecosystem owing to the flexibility and freedom that they provide [2]. People are more dependent on cars for both intercity and intracity transit, causing traffic congestion and parking difficulties [3]. "Looking for a parking space creates additional delays and impairs local circulation. In central areas of large cities, cruising may account for more than 10% of the local circulation as drivers can spend 20 minutes looking for a parking spot", said by Dr. Jean-Paul Rodrigue Department of Global Studies and Geography of Hofstra University. Many rental models are emerged to solve these parking problems as one of them is the car-sharing model, which aims to distribute cars within a city for use at a low cost. In this fashion, individuals can exploit all the benefits of a private vehicle without the hassles of lease payments, maintenance, or parking. The program comprises one-way or round-trip, depending on whether the pick-up and the drop-off stations are the same or not [4].

The car-sharing system provides an option to numerous people who opt not to own a vehicle, and they use this system whenever a private vehicle is needed. This system usually fixes the cost on the price per minute that includes a quote of variables such as fuel, price per kilometre, and the share of fixed costs for the operator like maintenance, rebalancing, insurance, and parking [5]. Besides helping in decreasing the level of congestion and managing the lack of parking lots, car-sharing systems have many other advantages such as the reduction of vehicle ownership that leads to efficient use of road and infrastructure, economical savings for the users, and diminution of air and noise pollution [5].

However, this program is facing many issues [6], one of them is the nonsuitable distribution of vehicles within car-sharing systems. As a result, cars tend to be easily accessible in low-demand parking lots in excess whereas an insufficient number of vehicles are available in high-demand parking lots [6]. For car-sharing companies, this problem causes a major financial loss. To improve the car usage rate, the companies employ a variety of techniques that hold great promises in car-sharing predictions.

Over the last few years, machine learning and deep learning have proved their efficiency and got recognition in different fields. Machine learning approaches make use of learning algorithms that make inferences from data to learn new tasks [7] and are widely adopted in a number of massive and complex data-intensive fields such as medicine, astronomy, and biology [8–11]. Deep learning models yield good results in the fields of computer vision and natural language processing [12–15] where it can automatically extract multidimensional features and effectively extract the data patterns for classification or regression [16].

In our work, a multivariate time series approach is presented, and it aims to predict the car usage in the short term and to investigate the factors that help to improve its prediction accuracy. Multiple machine learning models and deep learning models have already fulfilled their promises for multivariate time series prediction approaches and also have proved their ability in extracting meaningful understandings that are hard for humans to analyze and infer [5]. Those models were performed with different features set including the past usage levels, Chongqing's environmental conditions, and the temporal information.

The rest of the paper is organized as follows. Section 2 presents a literature review of current studies on times series models. Section 3 gives a description of the studied problem. A time series analysis is presented in Section 4. Section 5 demonstrates the framework of our approach. Section 6 describes the experimental framework used to evaluate the performance of used models for the multivariate time series approach. Finally, Section 7 concludes the paper and outlines future work directions.

2. Literature Review

Car-sharing has become one of the most popular research subjects in transportation. Many studies have been conducted, but to the best of our knowledge, no work has been done in the scientific literature that compares different machine learning and deep learning models in predicting the future usage of the car-sharing system and in investigating the factors that help improve its prediction accuracy. Interesting works are instead the following:

Studies related to this topic can be categorized into numerous subgroups including the followings [5, 17]: (i) user characteristics: it investigates the ways the users interact with the service [18, 19]; (ii) characterizing the shaping service in charge of the provisions and distribution of the cars around the city [20, 21]; (iii) car demands prediction level [22, 23].

Car demands prediction levels for car-sharing systems can be formulated as a time series prediction problem. Time series uses many approaches, such as the autoregressive integrated moving average (ARIMA) model that focuses on extracting the temporal variation patterns of the traffic flow and uses them for prediction; support vector regression (SVR) model, that captures complex nonlinearities, [20] demonstrated that this approach generally performs better on traffic flow time series; extreme gradient boosting (XGBoost), [21] showed that this model improves the prediction's precision and efficiency. Before starting the calculation, XGBoost sorts the traffic data according to the feature values and also realizes parallel computing on feature enumerations.

Recently, machine learning methods have been challenged by deep learning methods on traffic prediction. Deep learning approaches have a strong ability to express multimodal patterns in data, in order to reduce the overfitting problem and to obtain high prediction accuracy. In addition, as a traffic flow process is complicated in nature, deep learning algorithms can represent traffic features without prior knowledge, which has good performance for traffic flow prediction.

Xu and Lim [22] used an evolutionary neural network to prove the effectiveness of this algorithm and its possible usage as a tool for forecasting the net flow of a car-sharing system in order to offer the vehicle in the shortest time possible with the best accuracy; [23] attempted using the deep belief network (DBN) to define a deep architecture for traffic flow prediction that learns features with limited prior knowledge.

The abovementioned models require the input length to be predefined and static, and they cannot automatically determine the optimal time lags. To remedy these problems, many works have been done such as [24] used a model called long short-term memory recurrent neural network (LSTM RNN) that capture the nonlinearity and randomness of traffic flow more effectively and automatically determine the optimal time lags; [25] presented a novel long short-term memory neural network to predict travel speed using microwave detector data, where the future traffic condition is commonly relevant to the previous events with long time spans; Mo et al. [26] predicted the future trajectory of a surrounding vehicle in congested traffic by using the CNN-LSTM. To the best of our knowledge, no work is found in the literature on car-sharing time series prediction using CNN-LSTM.

Regarding the investigation of factors improving the prediction, [6] conducted the study about the effect of seasonal factors on the bookings of cars in Montreal, and after analysing the results, it was concluded that the usage outcomes scored better in summer season.

With respect to the above works, our approach presents the following highlights:

(i) Comparison of various machine learning and deep learning models to predict the future number of bookings made by car-sharing users using different metrics

(ii) Investigation of the factors that help predicting the car-sharing usage by estimating the relationship between data features and model performances

3. Problem Description

The aim of this study is to predict the number of vehicles that are going to be used in the parking stations at a given moment and to investigate the factors that improve the accuracy of predictions.

The number of vehicle usage at a given time t_x is likely to be correlated with a set of features [6], which are as follows:

(i) The past usage (F^x_{usage}): the history usage is tracked to build prediction models. It comprises of the number of car-sharing transactions based on the data from a car-sharing operator located in Chongqing, China.

(ii) Temporal information (F^x_{time}): the time at which the past usages have been acquired. Since the car demands may vary over time, we partition the time period into segments to capture different temporal trends (e.g., holidays/working days, 1 h timeslots) [6].

(iii) The environmental conditions (F^x_{weather}) at that time: the user transportation habits are usually affected by the weather conditions.

Table 1 summarizes the description of the feature categories of the car rental time series.

3.1. Car Rental Prediction Problem. Hereafter, we formulate the multivariate regression problem addressed in this paper [6]. It consists of predicting the car usage based on the values of features belonging to categories F^x_{usage}, F^x_{time}, and F^x_{weather}. Let T be the historical time period considered for training, t_1, ..., t_{k-1} be the past time points in T, and t_k be the current sampling time. We will denote usage (t_j) $[1 \le j \le k]$ the usage level at time t_j. We will give 1 h timeslots as a prediction horizon [6].

Since the future car usage is related to multiple features, the multivariate regressor R is expressed as follows:

$$\text{usage}(t_{k+1}) = R\left(F^x_{\text{usage}}, F^x_{\text{time}}, F^x_{\text{weather}}, 1 \le x \le k\right), \quad (1)$$

where F^x_{usage} is the usage levels of cars, F^x_{time} is the temporal information at t_x, and F^x_{weather} is the weather conditions in the area at time t_x [6].

3.2. Factors Investigation Problem. Another objective of this research is to determine the factors that help to predict vehicle usage. The features considered in this study are classified into different categories, namely, the past usage, temporal information, and the environmental conditions. We studied numerous machine learning and deep learning models, merging the different feature categories or using them separately one by one, aiming to find features that improve the prediction accuracy of the models.

4. Time Series Analysis

Time series is a sequential collection of recorded observations in consecutive time periods, and they can be univariate or multivariate [27, 28]. We may perform time series analysis with the aim of either predicting future values or understanding the processes driving them [29].

To address the problems stated in the previous section, multiple machine learning and deep learning models were performed.

4.1. Machine Learning Models

4.1.1. Vector Autoregression (VAR). Vector autoregression is a forecasting algorithm used when two or more time series influence each other. It is considered as an autoregressive model because the predictors are not only lags of the series but also past lags of itself [30]. Suppose we measure three different time series variables, denoted by $x_{t,1}$, $x_{t,2}$, $x_{t,3}$. The vector autoregression model of order 1 denoted as VAR(1) is as follows [30]:

$$x_{t,1} = a_1 + \Phi_{11}x_{t-1,1} + \Phi_{12}x_{t-1,2} + \Phi_{13}x_{t-1,3} + w_{t,1}, \quad (2)$$

$$x_{t,2} = a_2 + \Phi_{21}x_{t-1,1} + \Phi_{22}x_{t-1,2} + \Phi_{23}x_{t-1,3} + w_{t,2}, \quad (3)$$

$$x_{t,3} = a_3 + \Phi_{31}x_{t-1,1} + \Phi_{32}x_{t-1,2} + \Phi_{33}x_{t-1,3} + w_{t,3}. \quad (4)$$

The variable a_i is a k-vector of constants serving as the intercept of the model. Φ_{ij} is a time-invariant $(k \times k)$-matrix, and W_{ti} is a k-vector of error terms.

Each variable is a linear function of the lag 1 values for all variables in the set. In general, for a VAR(p) model, the first p lags of each variable in the system would be used as regression predictors for each variable [31, 32].

4.1.2. eXtreme Gradient Boosting (XGBoost). XGBoost is an efficient and scalable implementation of gradient boosting framework by Friedman et al. [33] and Friedman et al. [34] The package includes an efficient linear model solver and tree learning algorithm [35]. XGBoost fits the new model to new residuals of the previous prediction and then minimizes the loss while adding the latest prediction [36]. What makes it unique is that it uses "a more regularized model formalization to control overfitting, which gives it better performance"—Tianqi Chen. XGBoost is used for supervised learning problems, where we use the training data x_i to predict a target variable y_i. After choosing the target variable y_i, we need to define the objective function to measure how well the model fits the training data, and it consists of two parts, training loss and regularization term, as follows:

$$\text{obj}(\theta) = L(\theta) + \Omega(\theta), \quad (5)$$

where θ denotes the parameters that we need to learn from data, L is the training loss function, and Ω is the regularization term. A common choice of L is the mean-squared error, which is given by:

$$L(\theta) = \sum_i (y_i - \overline{y_i})^2. \quad (6)$$

And the regularization term controls the complexity of the model, which helps us to avoid overfitting [27].

4.1.3. Support Vector Regression (SVR). The foundations of support vector machines (SVM) have been laid by Vapnik and Chervonenkis, and the methodology is gaining in popularity. The foundations of SVM that deal with classification problems are called support vector classification (SVC) and those of SVM that deal with modelling and prediction are called support vector regression (SVR)) [28].

Most real-world problems cannot be modelled by using linear forms [31]. SVR methodology allows to handle real-world problems. Here are some common kernels used in the SVR modelling [25]:

(1) Linear kernel: $x * y$

(2) Polynomial kernel: $[(x * x_i) + 1]d$

(3) Radial basis function (RBF): exp{-ϒ $|x - x_i|^2$}

4.1.4. K-Nearest Neighbors (kNN). K-nearest neighbors (kNN) is an efficient and intuitive method that has been used extensively for classification in pattern recognition [32]. It is a distance-based classifier, which implies that it implicitly presumes that the smaller the distance between two points is, the more similar they would be [37]. KNN classification algorithm is by far more popular than KNN regression [37]. In the KNN regression model, the derived information from the observed data is applied to forecast the amount of predicted variable in real time [38]. In other words, it estimates the response of a testing point X_t as an average of the responses of the k closest training points, $X_{(1)}, X_{(2)}, \ldots, X_{(k)}$, in the neighborhood of X_t [32]. Let $X = \{X_1, X_2, \ldots, X_M\}$ be a training data-set consisting of M training points, each of which possesses N features [32]. The Euclidean distance is used to calculate how close each training point X_i is to the testing point X_t using

$$d(x_t, x_i) = \sqrt{\sum_{n=1}^{N} (x_{t,n} - x_{i,n})^2}, \quad (7)$$

where N is the number of features, $x_{t,n}$ is the nth feature values of the testing point X_t, and $x_{i,n}$ is the nth feature values of the training point X_i. Some other methods are Manhattan, Minkowski, and Hamming distance methods.

4.2. Deep Learning Models

4.2.1. Long Short-Term Memory (LSTM). Long short-term memory neural network (LSTM NN) was initially introduced by Hochreiter and Schmidhuber (1997) [21]. The primary objectives of LSTM NN are to overcome the vanishing gradients problem of the standard recurrent neural network (RNN) when dealing with long-term dependencies [39]. Its features are especially desirable for traffic prediction in the transportation domain [40]. Figure 1 shows the architecture of long short-term memory cell. The core concept of LSTM is to be composed of recurrently connected memory blocks, each of which contains one or more memory cells, along with three multiplicative "gate" units: One input gate i_t with corresponding weight matrix W_{xi}, W_{hi}, W_{ci}, b_i; one forget gate f_t with corresponding weight matrix W_{xf}, W_{hf}, W_{cf}, b_f; one output gate o_t with corresponding weight matrix W_{xo}, W_{ho}, W_{co}, b_o. All of those gates are set to generate some degrees, by using current input x_i, the state h_{t-1} that the previous step generated, and the current state of this cell c_{t-1}, for the decisions whether to take the inputs, forget the memory stored before, and output the state generated later. Just as these following equations demonstrate [39]:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i), \quad (8)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f), \quad (9)$$

$$g_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + W_{cc}c_{t-1} + b_c), \quad (10)$$

$$c_t = i_t g_t + f_t c_{t-1}, \quad (11)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o), \quad (12)$$

$$h_t = o_t \tanh(c_t). \quad (13)$$

The network controls the flowing information by its sigmoid layer which outputs numbers between zero and one ($S(t) = 1/(1 + e^{-t})$).

4.2.2. Gated Recurrent Unit (GRU). The gated recurrent unit (GRU) architecture contains two gates: an update gate z_t decides how much the unit updates its activation or content and a reset gate r_t allows to forget the previously computed state [41]. The model is defined by the following:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \quad (14)$$

$$r_t = \sigma(W_r x_r + U_r h_{t-1} + b_r), \quad (15)$$

$$H_t = \tanh(W_H x_t + U_H (h_{t-1} r_t) + b_H), \quad (16)$$

$$h_t = z_t H_t + (1 - z_t) h_{t-1}, \quad (17)$$

where h_t represents the output state vector at time t, while H_t is the candidate state obtained with a hyperbolic tangent, x_t represents the input vector at time t, and the parameters of the model are W_z, W_r, W_H (the feed-forward connections), U_z, U_r, U_H (the recurrent weights), and the bias vectors b_z, b_r, b_H [42].

4.2.3. Convolutional Neural Network (CNN). Convolutional neural networks (CNNs) are analogous to traditional artificial neural networks (ANNs) as they are comprised of neurons that self-optimize through learning

Table 1: Categories of features.

Feature category (abbrv.)	Feature description
Usage level (F^x_{usage})	#Rented-cars shows the hourly variation in the number of car rentals in Chongqing between 2017 and 2019
Time (F^x_{time})	Take-date: car-sharing rental date Take-hour: car-sharing rental hour Take-month: month of the year for sharing rental Season: the seasons of the year (winter, spring, summer, autumn)
Weather (F^x_{weather})	Temperature, precipitation, humidity, wind speed, etc.

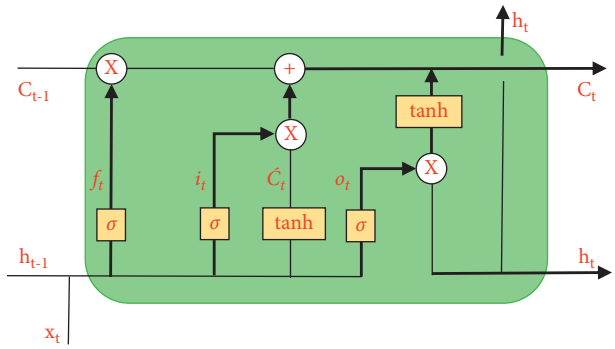

Figure 1: The architecture of long short-term memory cell.

[43]. They were initially developed for computer vision tasks; nevertheless, there have been a few recent studies applying them to time series forecasting tasks. CNNs are comprised of three types of layers including convolutional layers, pooling layers, and fully connected layers as shown in Figure 2.

The convolutional layer will determine the output of neurons connected to local regions of the input through the calculation of the scalar product between their weights and the region connected to the input volume [43]. There are two important techniques used in the convolutional layers to accelerate the training process: local connectivity and weight sharing. The two techniques are implemented using a filter with a specific kernel size which defines the number of nodes that share weights. Their usage decreases significantly the number of learned and stored weights and allows the network to grow deeper with fewer parameters. The pooling layer is usually incorporated between two successive convolutional layers [44].

The main idea of pooling is to reduce the complexity for further layers by down-sampling [45]. Max-pooling is one of the most common types of pooling methods as it performs better [44]. It consists in partitioning the image to subregion rectangles and only returning the maximum value of the inside of that subregion [45].

The fully connected layers are simply, feed-forward neural networks. They form the last few layers in the network [46]. The input to the fully connected layer is the output from the final pooling or convolutional layer, which is flattened and then fed into the fully connected layer in order to perform the same duties found in standard ANNs [43, 46].

4.2.4. CNN-LSTM Model. CNN-LSTM is a hybrid model built by combining CNN with LSTM for improving the accuracy of forecasting [47]. Figure 3 shows the architecture of the CNN-LSTM model. The model comprises of two main components: the first component consists of convolutional and pooling layers in which complicated mathematical operations are performed to filter the input data and extract the useful information. More specifically, the convolutional layers apply convolution operation between the raw input data and the convolution kernels, producing new feature values [48]. The convolution kernel can be considered as a window that contains coefficient values in a matrix form. This window slides all over the input matrix applying convolution operation on each subregion of it. The result of all these operations is a convolved matrix that represents a feature value.

The convolutional layers are usually followed by a nonlinear activation function and then a pooling layer. A pooling layer is a subsampling technique that extracts certain values from the convolved features and produces new matrices (i.e., summarized versions of the convolved features that are produced by the convolutional layer).

The second component exploits the generated features by LSTM which possesses the ability to learn long-term and short-term dependencies through the utilization of feedback connections and dense layers [48].

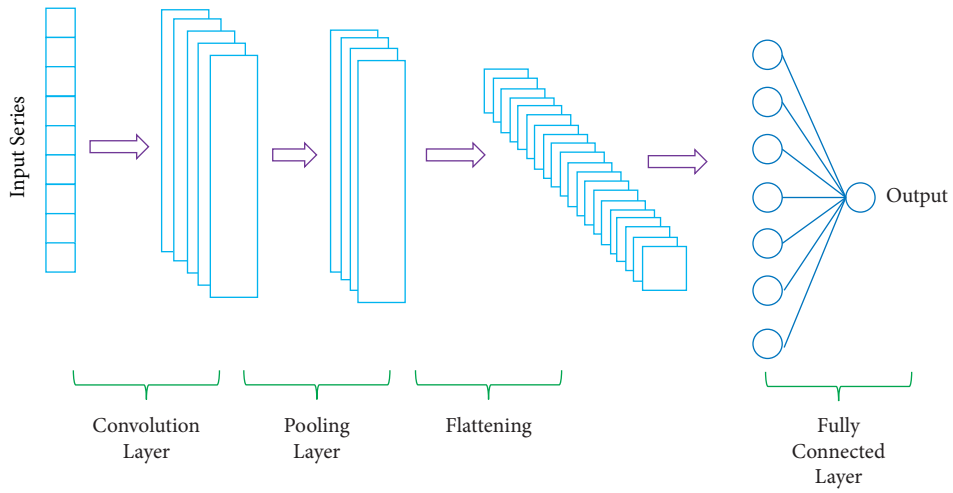

Figure 2: A simple architecture of convolutional neural network.

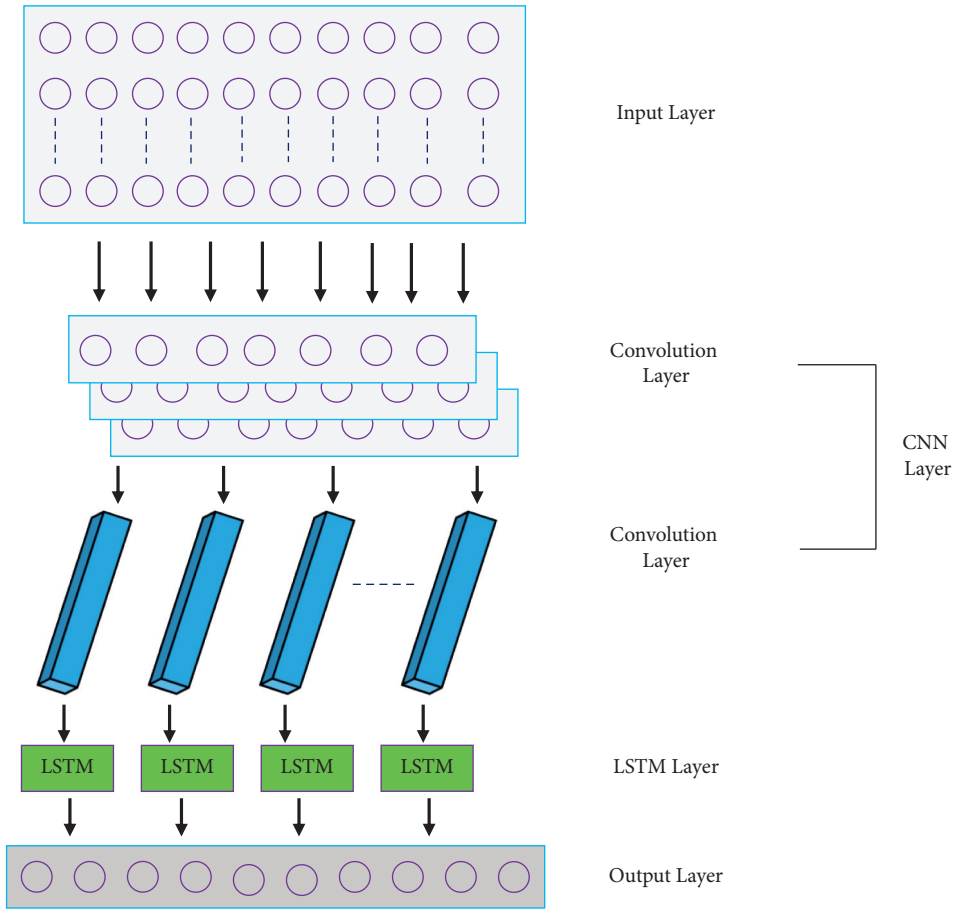

Figure 3: Architecture of CNN-LSTM.

4.2.5. Multilayer Perceptron (MLP). Multilayer perceptrons (MLPs) are deep artificial neural networks and are often applied to supervised learning problems [49]. As we can see from Figure 4, a multilayer perceptron consists of three types of layers: An input layer to receive the signal, an output layer that performs the required tasks to make a decision or prediction about the input, and an arbitrary number of hidden layers that are the true computational engine [49, 50]. In MLP, the data flow in the forward direction from the input to output layer, and the neurons are trained with the backpropagation learning algorithm on a set of input-output pairs. The training involves adjusting the parameters, or the weights and biases, of the model in order to minimize errors [49].

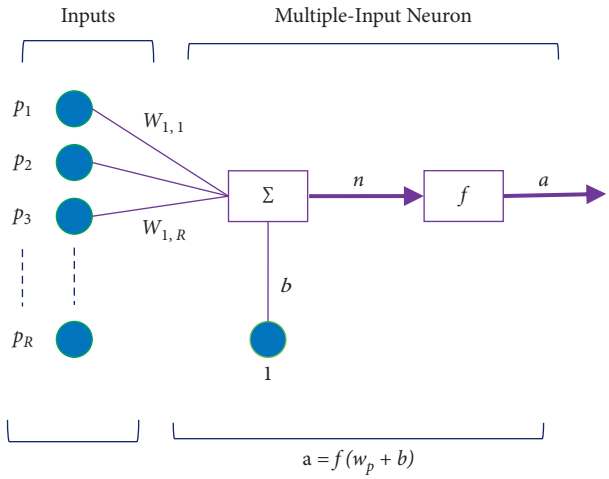

Figure 4: Multilayer perceptron architecture.

4.3. Vanishing Gradients Problem. Artificial neural networks often experience training problems due to vanishing and exploding gradients. The training problem is amplified exponentially especially in deep learning due to its complex artificial neural network architecture [51]. The vanishing gradient is one example of unstable behavior that may be encountered during the training with gradient-based methods (e.g., back propagation) [52]. The neural network's weights receive an update proportional to the partial derivative of the error function with respect to the current weight in each iteration of training. In some cases, the gradient tends to get smaller as we move backward through the hidden layers. This means that neurons in the earlier layers learn much more slowly than neurons in later layers preventing the weight from changing its value [52].

Several approaches exist to reduce this effect in practice, for example, through careful initialization, hidden layer supervision, and batch normalization [53]. In our work, batch normalization has been used, as it was effective in augmenting the performance of the deep neural network.

5. Framework of the Approach

Figure 5 shows the process of car-sharing usage prediction and factors investigation approach based on machine and deep learning models.

5.1. Collecting Chongqing's Car-Sharing Operator Data. Chongqing car-sharing operators' data set contains more than 1 M records of car-sharing usage over 860 parking lots, from January 1st, 2017, 00:00:00 to January 31st, 2019, 23:00:00. The initial records were obtained at different time intervals, and for study purposes, the data are aggregated by hours, days, and weeks, for the whole network.

5.2. Collecting Chongqing's Weather Data. The web crawling technology with selenium was used to extract the hourly Chongqing's weather conditions data from January 1st, 2017, 00:00:00 to January 31st, 2019, 23:00:00 [54].

5.3. Data Preprocessing. A data preprocessing is performed on the data set to improve the performance [55].

5.3.1. Processing Missing Values. Some values were missing in the data set from Chongqing's car-sharing operator. Due to its numerical meaning, we replaced the missing values by the mean of the previous and next hour number of car-sharing usage. This method yields better results compared to the removal of rows. The detailed calculation is shown as follows:

$$C^i_{j,k} = \frac{\left(C^i_{j,k-1} + C^i_{j,k+1}\right)}{2}, \tag{18}$$

where $C^i_{j,k}$ represents station i's missing value on the k^{th} hour of jth day of the year. After handling the missing values of Chongqing's car-rental operator data set, we merged it with Chongqing's weather conditions data set based upon dates and time.

5.3.2. Encoding the Categorical Data. Since the final data set, combining Chongqing's car-sharing operator and weather data, contains some categorical data such as weather condition and season, we converted the categorical data to numerical data using one-hot encoding method. It consists of representing each categorical variable with a binary vector that has one element for each unique label and marking the class label with a 1 and all other elements 0 [56].

5.3.3. Clustering Car-Sharing Parking Stations. To identify and understand the car-rental behaviors across stations and reveal the relationships between the time of a day and usage [57], we organized the parking stations with similar patterns into five distinct classes as follows:

(i) Class A: daily rented cars
(ii) Class B: frequently used cars
(iii) Class C: sometimes used cars
(iv) Class D: occasionally used cars

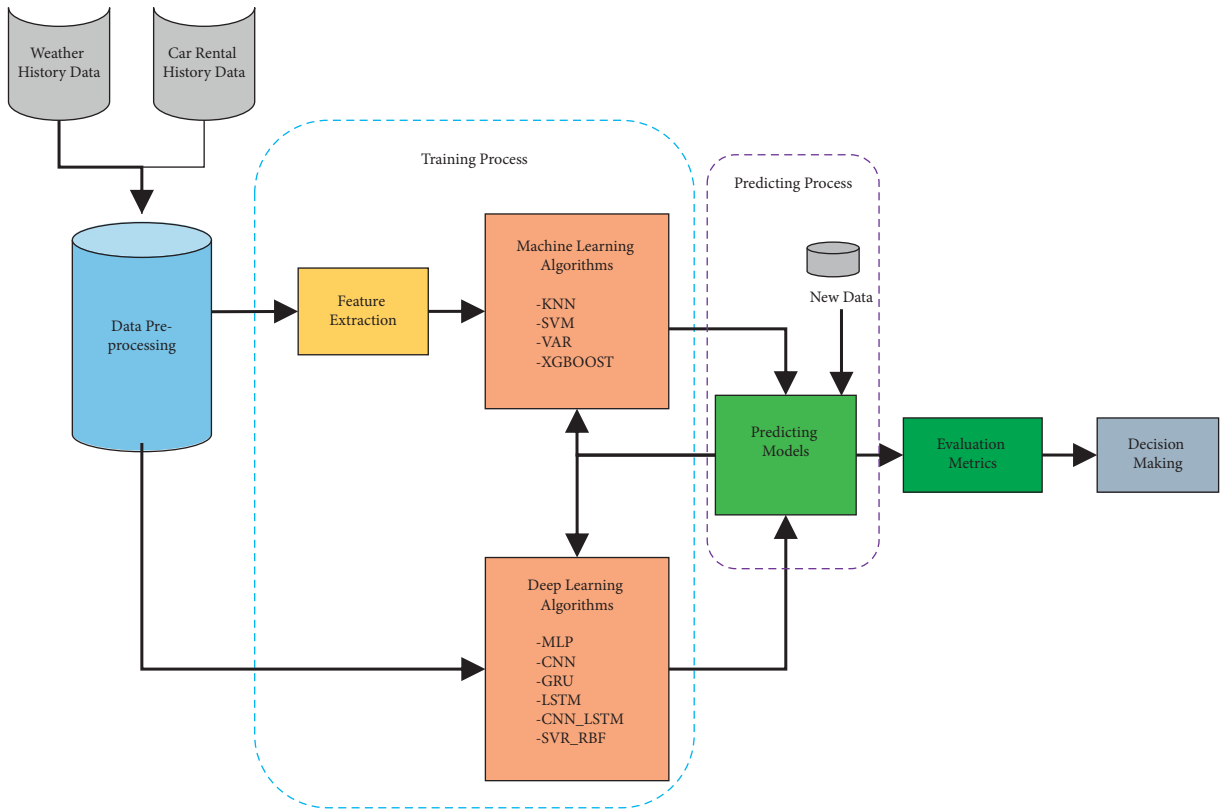

Figure 5: The framework of the proposed approach.

(v) Class E: unlike other parking stations, cars of this class are cars rarely used

where Class A, B, C, D, and E have different parking stations' IDs such as 16, 104, 6, 28, and 25, respectively. To simplify the large set of data and make it understandable, we used the grouped frequency table to cluster the 860 parking stations into five classes [58].

First, the usage frequencies of the parking stations were put in order, and then, the range was calculated as below [59]:

$$\text{range} = \text{largestfrequency value} - \text{smallest frequency value}. \tag{19}$$

Second, an approximate class width was calculated by dividing the range by the number of classes:

$$\text{class width} = \frac{\text{range}}{\text{number of classes}}. \tag{20}$$

The lowest usage frequency represents the first minimum data value.

Third, the next lower class value was calculated by adding the class width to the lowest usage frequency.

$$\text{lower class value} = \text{lowest usage frequency} + \text{class width}. \tag{21}$$

This step was repeated for the other minimum data values until the chosen classes number was created.

Fourth, the upper class limits (that are the highest values possible in the class) were calculated by subtracting 1 from the class width and adding that to the minimum data value.

$$\text{upper class limit} = \text{lowest usage frequency} + (\text{Class width} - 1). \tag{22}$$

Finally, the list of classes is obtained by including in each class the usage frequencies that are greater than the lower class value and smaller than the upper class limit.

5.4. Deseasonalization. Stationarity is an important concept for time series analysis. Some experts believe that neural networks are able to model seasonality directly and that no prior deseasonalization is required, whereas others believe the contrary. The results in [60] show that a prior data processing is required to construct a forecasting model. To test our time series, Augmented Dickey-Fuller Test (ADF Test) was conducted with machine learning and deep learning models to make predictions more accurate [61]. We employed differencing to remove seasonality from the nonstationary time series after the ADF Test [61, 62].

5.5. Scaling. The scaling phase is crucial to move the time series into a reasonable range. In our work, MinMaxScaler was used to scale each feature to a given range.

5.6. Splitting the Data Set. After handling the previously mentioned steps, we prepared our data set properly. We split the data between training and test sets. The training set starts from January 1st, 2017, to December 31st, 2018, and the test set from January 01st, 2019, to January 31st, 2019. Nested cross-validation with an outer loop equal to ten and an inner loop equal to five is used to calculate and to compare each model error. All models had to use the same validation procedure for consistency matters.

6. Experiments

6.1. Data Set. The experiments were performed on the preprocessed Chongqing's car-sharing operator data set combined with Chongqing's weather data set, to extract features that help to predict car usage, and to demonstrate the effectiveness of deep learning, more precisely the CNN-LSTM comparing to other models.

We have implemented the proposed models using a PC with an i7 Intel (R) Core™i7-7500U CPU running at 3.00 GHz and 8 GB RAM with the Windows 10 operating system, under the Python 3.7 development environment. The following packages were installed: TensorFlow 1.14.0; Keras 2.2.4-tf; Pandas 0.23.4; Sklearn 0.21.1; Numpy 1.18.1; Matplotlib 3.1.0; Statsmodels 0.10.1.

The hourly weather observations include time, temperature, humidity, wind speed, pressure, precipitation, and weather conditions. To have a basic idea about what weather conditions are typically associated with the city of Chongqing, we calculated the normalized frequency distributions of every weather condition (Table 2).

From Table 2, fair is the most prevalent meteorological condition in Chongqing, closely followed by fog, light rain, partly cloudy, and cloudy.

From Table 3, July and August are the hottest months of the year with an average temperature of 28°C, and January is the coldest month of the year with the temperature of 7°C.

6.2. Evaluation Metrics. The evaluation metrics are the measure that reflects how close the prediction matches the historical data. They are useful in comparing prediction methods on the same set of data [63].

6.2.1. Mean Absolute Error (MAE). MAE is calculated as the mean of the absolute predicted error values. The MAE is popular as it is easy to both understand and compute.

$$\text{MAE} = \text{mean}\left(\text{absolute}\left(\text{expected}_{\text{value}} - \text{predicted}_{\text{value}}\right)\right). \tag{23}$$

6.2.2. Mean Square Error (MSE). MSE is known for putting more weight on large errors. It is calculated as the average of the squared predicted error values.

$$\text{MSE} = \text{mean}\left(\left(\text{expected}_{\text{value}} - \text{predicted}_{\text{value}}\right)^2\right). \tag{24}$$

6.2.3. Root Square Mean Error (RMSE). The mean-squared error described above is in the squared units of the predictions. It can be transformed back into the original units of the predictions by taking the square root of the mean-squared error score [64]:

$$\text{RMSE} = \text{sqrt}(\text{MSE}). \tag{25}$$

The RMSE is chosen as an evaluation metrics because it penalizes large prediction errors more compared with mean absolute error (MAE).

6.2.4. Mean Absolute Percentage Error (MAPE). Mean absolute percentage error (MAPE) is one of the most widely used measures of forecast accuracy, due to its advantages of scale independence and interpretability [65]. It is calculated using the absolute error in each period divided by the observed values that are evident for that period and consequently averaging those fixed percentages [66]. MAPE indicates how much error in prediction is compared with the real value. The MAPE can be defined by the following formula:

$$\text{MAPE} = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|, \tag{26}$$

where A_t is the actual value, F_t is the forecast value, and n denotes the number of fitted points.

6.2.5. Root-Mean-Squared Log Error (RMSLE). The root-mean-squared log error (RMSLE) is the RMSE of the log-transformed predicted and target values [54]. RMSLE only considers the relative error between predicted and actual values, and the scale of the error is nullified by the log-transformation [67]. The formula for RMSLE is represented as follows:

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\log\left(p_i + 1\right) - \log\left(a_i + 1\right)\right)^2}, \tag{27}$$

where n is the total number of observations in the data set, p_i is the prediction of target, a_i is the actual target for i, and $\log(x)$ is the natural logarithm of x, $\log_e(x)$.

6.3. Experiments and Analysis. Many experiments were conducted on parking stations of different classes to extract which features improved the prediction and to predict the demands accordingly. Before applying the different models, some tests were performed on our time series.

6.3.1. Granger Causality Tests. The Granger causality test is a statistical hypothesis test for determining whether one time series is useful for predicting another [68]. In other words, it

Table 2: Chongqing's weather conditions normalized frequency distributions.

Weather condition	Frequency	Weather condition	Frequency
Fair	0.42962	Light drizzle	0.00135
Fog	0.13210	T-storm	0.00130
Light rain	0.12954	Partial fog	0.00055
Partly cloudy	0.12487	Heavy T-storm	0.00050
Cloudy	0.10191	Wintry mix	0.00025
Mostly cloudy	0.02330	Fair/windy	0.00020
Haze	0.01707	Thunder in vicinity	0.00015
Mist	0.01285	Heavy T-storm/windy	0.00010
Light rain shower	0.01155	Heavy rain	0.00010
Rain	0.00341	T-storm/windy	0.00010
Patches of fog	0.00236	Shallow fog	0.00005
Light rain with thunders	0.00236	Heavy rain shower/windy	0.00005
Thunder	0.00226	Showers in the vicinity	0.00005
Rain shower	0.00190	Light rain/windy	0.00005

Table 3: Chongqing's monthly temperature records.

Month average	Average temperature	Max/min average
January	7	5/10
February	9	7/12
March	14	11/17
April	19	15/23
May	22	18/25
June	25	21/29
July	28	24/33
August	28	24/33
September	24	20/27
October	18	16/21
November	14	11/16
December	9	7/11

is an approach that analyses the causal relationships between different variables of the time series.

After analysing the results shown in Table 4, we observed that all the given p value were smaller than the significance level (0.05). For example, the value of 0.0003 at (row 4) represents the p value of the Grangers causality test for temperature_x causing number_rented_cars_y (p value (0.0003) < significance level of 0.05).

From the results, we can infer that all the variables are good candidates to help predicting the number of rented cars.

6.3.2. Machine Learning Configuration

(1) eXtreme Gradient Boosting. A grid search was created for XGBoost model in order to locate the most optimal hyperparameters for the data set.

(2) Vector Autoregression. To select the right order of the VAR model, we iteratively fit increasing orders of the VAR model and picked the order that gave a model with the least AIC [29]. In our work, we have chosen the lag 4 model. Before predicting the future values of the target variable with VAR, we used serial correlation of residuals to check if our model is able to explain the patterns in the time series. The obtained scores in Table 5 show that our model is able to capture all the patterns without leftover.

(3) Support Vector Regression. The RBF kernel was chosen in this study for its good performance and advantages in time series prediction problem that has been proved in past researches [69, 70]. The penalty parameters are all tuned for the best using of a grid search method. Predictions are computed with optimal combination of cost and gamma parameters.

(4) K-Nearest Neighbors. The most important hyper-parameters for KNN are as follows: the number of neighbors (K) and the distance metric, and they determine the way in which the nearest neighbors are chosen.

To choose the number of neighbors (K), a grid search was performed. Moreover, the distance metric plays a crucial role in the nearest neighbor algorithm. Most of the papers in the references used the Euclidean distance. Reference [71] did a comparison between the Euclidean and Manhattan distance and found that statistically, no distance metric is significantly better than the other. The Euclidean distance has been selected since it is the most used in time series forecasting with KNN regression works.

6.3.3. Deep Learning Configuration

(1) Long Short-Term Memory and Gated Recurrent Unit Models. In this study, LSTM and GRU models have one neuron in the output layer. Both neural network models were designed with only one hidden layer, a number of 50 epochs were chosen, and the learning rate was set to 0.01. The input and hidden neurons for the GRU model were the same as those for the LSTM model and were set to 41 and 50, respectively [56].

(2) Convolutional Neural Network. CNN is one of the most successful deep learning methods, and its network structures include 1D CNN, 2D CNN, and 3D CNN. 1D CNN is used in this paper, and it can be well applied to time series analysis. The detailed process of the 1D CNN is described as follows:

In our experiment, we used various layers including one convolutional layer with a kernel size of 2 and 64 filters, a max-pooling layer, along with that a rectified linear unit

Table 4: Granger causality test results.

Variables	Number_rented_cars
Day_of_week	0.0004
Take_month	0.0289
Season	0.0001
Temperature	0.0003
Humidity	0.0000
Wind_speed	0.0000
Pressure	0.0000
Precipitation	0.0000
Weather_condition	0.0004
Weekdays_weekend_x	0.0003

Table 5: Check for serial correlation of residuals.

Columns	Value
Number_rented_cars	2.03
Day_of_week_x	2.0
Take_month_x	2.0
Season_x	2.0
Temperature_x	2.03
Humidity_x	2.01
Wind_speed_x	2.0
Pressure_x	2.03
Precipitation_x	2.0
Weather_condition_x	2.01
Weekdays_weekend_x	2.0

(ReLU) activation function is applied in the convolutional and output layers. To minimize the mean-squared error, the gradient descent backpropagation algorithm and Adam optimizer were used; following that, a dropout rate of 0.5 was employed to avoid the overfitting [72], and a number of 70 epochs were used for training the model.

(3) CNN-LSTM. In our implementation, we utilized a version of the CNN-LSTM model that consists of two one-dimensional convolutional layers with a kernel size equal to 5, 32, and 64 filters, respectively, followed by a max-pooling layer, a LSTM layer with 50 units, and a dense layer of 32 neurons [48], In order to avoid overfitting during training, the dropout was adjusted to 0.2. A number of 100 epochs were chosen to train the model.

(4) Multilayer Perceptron Regressor. MLP was applied on our multivariate time series, with ReLU as an activation function to train the regression model. Three hidden layers were used with a number of 50, 35, and 10 hidden neurons, respectively. The training was performed for 50 epochs for MLP with a learning rate of 0.01.

6.4. Prediction Result. For the good organization of the paper, and not being redundant in our explanations, we only discussed the result analysis of the class "A" as other classes exhibit the same behavior and lead to the same conclusion.

To perform the comparison while fitting the models with different features, the results of only CNN-LSTM for deep learning models and XGBoost for machine learning models are described in our analysis. In the same way, each of the applied models had the analysis of the same results.

In our investigation, metrics such as MAE, MSE, RMSE, MAPE, and RMSLE are used in respective order to make the comparison. Note that the smallest errors are shown in bold text in Tables 6–13.

6.4.1. Univariate Time Series

(1) Machine Learning Models. As shown in Table 6, XGBoost reduced the error by (93.14%, 99.14%, 93.36%, 69.17%, 93.76%) against VAR; (93.69%, 99.48%, 94.17%, 83.09%, 94.70%) against SVM; and (93.93%, 99.65%, 94.44%, 84.17%, 95.05%) against KNN.

(2) Deep Learning Models. As it can be seen from Table 7, CNN-LSTM yielded the best results and had smaller evaluation error compared to all other models. CNN-LSTM decreased the evaluation error by (44.19%, 48.37%, 34.44%, 1.03%, 24.14%) against LSTM; (51.91%, 57.97%, 35.15%, 11.71%, 40.21%) against GRU; (61.15%, 58.58%, 79.38%, 37.38%, 38.76%) against CNN; and (67.43%, 95.75%, 69%, 88.24%, 61.37%) against MLP.

6.4.2. The Effect of Weekends Information. We might not forget or underestimate the impact of weekends on the prediction while building prediction models with time series data. They are significant since they can add peculiarity to the outcomes.

(1) Machine Learning Models. From Table 8, it can be noticed that using the XGBoost model with weekend features enhanced the improvement rate of (1.14%, 0.62%, 1.08%, 0.42%, 0.24%) compared to univariate time series.

XGBoost outperformed VAR, SVM, and KNN. It reduced evaluation error at the rate of (93.20%, 99.19%, 93.34%, 69.04%, 93.76%) contrary to VAR; (93.74%, 99.47%, 94.14%, 82.98%, 94.46) contrary to SVM; and (94.00%, 99.65%, 94.37%, 84.04%, 95.04%) contrary to KNN.

(2) Deep Learning Models. The addition of weekend features improved prediction accuracy as it can be observed from Table 9, where adding the weekend feature to the univariate time series of class "A" improved the results with a rate of (2.4%, 4.95%, 2.54%, 4.23%, 0.43%) for CNN-LSTM.

Regarding models' comparison, CNN-LSTM achieved the best results with an improvement rate of (43.75%, 46.52%, 28, 50%, 0.88%, 37.98%) against LSTM; (46.47%, 57.96%, 35.16%, 4.87%, 35.45%) against GRU; (40.38%, 59.34%, 67.17%, 48.70%, 40.56%) against CNN; and (65.92%, 89.22%, 67.79%, 75.46%, 60.88%) against MLP.

6.4.3. The Effect of Weather Information. In addition to rental information in the city of Chongqing, we can leverage weather data to improve prediction at different times of the day.

Table 6: Machine learning for univariate time series evaluation results.

	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.07272**	**0.00975**	**0.09868**	**0.30398**	**0.07189**
VAR	1.06025	1.20552	1.4851	0.98598	1.15297
SVM	1.15297	1.86949	1.69395	1.7977	1.3575
KNN	1.19834	2.77452	1.77467	1.91974	1.45087

(a) Class A

	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.09125**	**0.02252**	**0.14942**	0.18352	**0.06025**
VAR	0.45087	0.36788	0.60653	**0.14527**	0.09057
SVM	0.72289	1.04231	1.01927	1.14777	0.960432
KNN	0.76353	1.48786	1.18645	1.25337	0.72234

(b) Class B

	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.1179**	**0.03489**	**0.18489**	**0.29007**	**0.1179**
VAR	0.43931	0.34352	0.5861	0.50319	0.43931
SVM	0.81499	0.99441	0.99721	1.1251	0.59046
KNN	0.83915	1.86306	1.36483	1.22767	1.30815

(c) Class C

	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.03654**	**0.00314**	**0.05391**	**0.04028**	**0.03299**
VAR	0.43459	0.34597	0.58819	0.18361	0.43459
SVM	0.54589	0.69401	0.83355	1.16683	0.54795
KNN	0.61745	1.14995	1.07216	1.18775	0.90985

(d) Class D

	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.04051**	**0.00285**	**0.05338**	**0.00683**	**0.04051**
VAR	0.08543	0.08782	0.29143	0.14278	0.08659
SVM	0.14529	0.11864	0.34444	0.53728	0.08543
KNN	0.27331	0.13296	0.56446	1.9495	0.47325

(e) Class E

(1) Machine Learning Models. Table 10 shows that applying XGBoost reduced the MAE by 2.68%; the MSE by 3.18%; the RMSE by 0.25%; the MAPE by 6.84%; and the RMSLE by 1.56% compared to the univariate time series results. Correspondingly, it reduced the MAE by 1.56%; the MSE by 2.58%; the RMSE by 0.33%; the MAPE by 6.98%; and the RMSLE by 1.32% relative to the weekend results.

XGBoost outperformed the other models as it decreased the error values by (93.28%, 99.21%, 93.35%, 76.26%, 93.72%) compared to VAR; (93.49%, 99.41%, 94.15%, 94.20%, 94.05%) compared SVM; and (94.08%, 99.56%, 94.26%, 94.54%, 95.10%) compared to KNN.

(2) Deep Learning Models. From Table 11, our findings show that the CNN-LSTM model was (37.97%, 37.59%, 35.82%, 5.49%, 23.42%) more accurate than LSTM; (39.82%, 57.73%, 34.93%, 2.62%, 38.70%) more accurate than GRU; (40.21%, 58.84%, 58.46%, 43.80%, 40.76%) more accurate than CNN; and (65.25%, 82.77%, 52.96%, 76.67%, 60.67%) more accurate than MLP.

Moreover, it was observed after a thorough examination that integrating weather data as a feature improved the prediction accuracy. When the CNN-LSTM model was applied to class "A" data with weather features, it reduced the MAE by 2.67%; the MSE by 6.04%; the RMSE by 3.03%; the MAPE by 11.10%; and the RMSLE by 5.02% compared to the univariate time series results. Similarly, it also reduced the MAE by 0.28%; the MSE by 1.16%; the RMSE by 0.51%; the MAPE by 7.17%; and the RMSLE by 5.43% compared to the weekend's results.

Table 7: Deep learning for univariate time series evaluation results.

	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.04083**	**0.00364**	**0.06035**	**0.08108**	**0.04843**
LSTM	0.07316	0.00705	0.09206	0.08192	0.06384
GRU	0.08491	0.00866	0.09306	0.09151	0.081
CNN	0.10512	0.00879	0.2927	0.14614	0.07909
MLP	0.12537	0.08568	0.19468	0.77838	0.12537

(a) Class A

	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.00616**	**0.00714**	**0.0074**	**0.05434**	0.00616
LSTM	0.04043	0.04718	0.04977	0.08423	**0.00404**
GRU	0.04404	0.05247	0.04989	0.18194	0.03956
CNN	0.03939	0.08383	0.09927	0.77724	0.13109
MLP	0.1261	0.05533	0.28953	0.88942	0.1261

(b) Class B

	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.02172**	**0.00126**	**0.03547**	0.09764	**0.02864**
LSTM	0.03332	0.00243	0.06397	**0.07606**	0.03059
GRU	0.0338	0.00252	0.04974	0.65142	0.03546
CNN	0.12709	0.00983	0.14989	0.73808	0.06779
MLP	0.13366	0.0851	0.29171	0.91345	0.12709

(c) Class C

	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.00913**	0.00852	**0.01545**	0.54338	**0.00927**
LSTM	0.01596	**0.00621**	0.02222	**0.49018**	0.01717
GRU	0.01663	0.08175	0.02151	0.74992	0.01612
CNN	0.12113	0.08763	0.28592	0.53462	0.12113
MLP	0.12187	0.15624	0.29603	0.80945	0.12187

(d) Class D

	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.01213**	**0.00198**	**0.04331**	**0.00727**	**0.02345**
LSTM	0.02244	0.00466	0.13236	0.02329	0.02687
GRU	0.03553	0.00475	0.06827	0.33448	0.03836
CNN	0.03746	0.08376	0.06893	0.67174	0.0418
MLP	0.1227	0.08708	0.28941	0.71862	0.1227

(e) Class E

6.4.4. Combined Effect of Historical, Weekends, and Weather Information

(1) Machine Learning Models. The evaluation error of XGBoost was lower as shown in Table 12. It reduced the error by (93.35%, 99.28%, 93.76%, 73.71%, 80.32%) relative to VAR; by (93.57%, 99.47%, 94.47%, 94.21%, 93.81%) relative to SVM; and by (94.12%, 99.59%, 94.62%, 94.54%, 95.16%) relative to KNN.

Our findings show that when all features were used together, XGBoost performed better with rate of (3.96%, 12.92%, 6.65%, 6%, 2.85%). Similarly, it also performed (2.85%, 12.38%, 6.72%, 6.14%, 2.62%) better than when history combined with weekend features were used and (1.31%, 10.6%, 6.41%, 5.48%, 1.31%) better than when history combined with weather features were used.

Table 8: Machine learning for historical data with weekends data evaluation results.

	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.07189**	**0.00969**	**0.09876**	**0.30527**	**0.07172**
VAR	1.05771	1.19652	1.482066	0.98598	1.14989
SVM	1.14782	1.83432	1.68569	1.79397	1.2942
KNN	1.19799	2.77371	1.75573	1.91257	1.44645

(a) Class A

	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.09057**	**0.02244**	**0.15007**	0.18352	**0.05771**
VAR	0.44645	0.36339	0.60281	**0.13388**	0.09025
SVM	0.72168	1.03896	1.01929	1.14777	0.78527
KNN	0.76351	1.40766	1.18639	1.24379	0.7216

(b) Class B

	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.1175**	**0.03476**	**0.18457**	**0.28931**	**0.1175**
VAR	0.43805	0.34159	0.58445	0.50319	0.43805
SVM	0.81479	0.99398	0.99698	1.1251	0.57837
KNN	0.83915	1.86244	1.36463	1.22597	0.93895

(c) Class C

	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.03299**	**0.00314**	**0.05544**	**0.03984**	**0.02959**
VAR	0.43452	0.34574	0.588	0.18361	0.43452
SVM	0.54589	0.69409	0.83352	1.16626	0.54589
KNN	0.61696	1.14954	1.07216	1.18884	0.89971

(d) Class D

	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.01878**	**0.00222**	**0.04717**	**0.00583**	**0.01878**
VAR	0.08543	0.08481	0.29123	0.14278	0.08527
SVM	0.14422	0.11862	0.34426	0.53462	0.08543
KNN	0.27324	0.13282	0.36444	1.94985	0.37284

(e) Class E

(2) Deep Learning Models. From Table 13, our findings reveal that when CNN-LSTM was fitted with all features together, it performed better with rate of (1.41%, 2.34%, 1.28%, 8.82%, 7.98%) than when only history features were used; likewise, it was (4.04%, 8.24%, 4.28%, 8.17%, 12.60%) better than when history combined with weekend features were used, and (1.68%, 3.47%, 1.79%, 8.33%, 12.97%) better than when history combined with weather features were used.

In comparison to all other models, CNN-LSTM has much lower evaluation error. It decreased the error by (36.80%, 23.39%, 31.18%, 46.46%, 29.66%) against LSTM, by (37.03%, 53.09%, 35.41%, 74.91%, 37.21%) against GRU, by (36.939%, 58.25%, 31.52%, 88.65%, 42.98%) against CNN, by (38.79%, 59.17%, 36.02%, 88.52%, 31.97%) when compared to MLP.

6.5. Comparison of the Results. One of our objectives is to compare the accuracy of various machine learning and deep learning models in predicting the future number of car-sharing transactions. After analysing the results obtained in our study, the following are our findings:

6.5.1. Machine Learning. First, with regard to the results obtained with the machine learning models and after the comparison based on the evaluation metrics, it shows that XGBoost gave the best results, followed by VAR, SVR, and KNN. The XGBoost model had several advantages in model prediction such as complete feature extraction, good fitting effect, and high prediction accuracy.

Second, the SVR prediction series failed to capture random and nonlinear patterns. Hence, it failed to perform well, while XGBoost and VAR forecast series were able to capture random walk patterns.

Third, KNN performed the worse compared to the other machine learning models because of the high number of inputs.

6.5.2. Deep Learning. After comparison of results, we can deduce that CNN-LSTM generated better outcomes followed by LSTM, GRU, CNN, and MLP.

The hybrid CNN-LSTM model yielded better performance on the strength of its capability in supporting very long input sequences that can be read as subsequences by the CNN model and then formed together by the LSTM model.

Besides the CNN-LSTM model, the long short-term memory model achieved good results on account of its ability to learn patterns from sequenced data more effectively.

The key difference between the gated recurrent unit model and the long short-term memory model is that GRU is less complex than LSTM, as it only has two gates (i.e., reset and update) while LSTM has three gates (including

Table 9: Deep learning for historical data with weekends data evaluation results.

	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.03985**	0.00346	**0.05882**	**0.07765**	**0.04864**
LSTM	0.07084	0.00647	0.08226	0.07834	0.07843
GRU	0.07445	0.00823	0.09072	0.08235	0.07535
CNN	0.06685	0.00851	0.17917	0.16054	0.08183
MLP	0.11695	**0.0321**	0.18264	0.31643	0.12436
(a) Class A					
	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.00549**	**0.00712**	**0.0062**	**0.02987**	**0.00549**
LSTM	0.03908	0.0343	0.01044	0.03156	0.02043
GRU	0.03774	0.01249	0.04989	0.32077	0.03437
CNN	0.03362	0.03393	0.09543	0.41409	0.07012
MLP	0.11776	0.01509	0.18421	0.74706	0.11776
(b) Class B					
	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.02107**	**0.00126**	**0.03544**	**0.07989**	**0.0274**
LSTM	0.02945	0.00229	0.05796	0.09897	0.02942
GRU	0.03309	0.00249	0.04929	0.56722	0.03524
CNN	0.11495	0.00257	0.13197	0.31063	0.03386
MLP	0.12668	0.03283	0.18118	0.6956	0.11495
(c) Class C					
	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.00841**	0.00543	**0.01444**	0.32857	**0.00848**
LSTM	0.00947	**0.00524**	0.02185	**0.16148**	0.0155
GRU	0.01529	0.0302	0.02081	0.28902	0.00966
CNN	0.10715	0.03437	0.17378	0.67375	0.10715
MLP	0.11859	0.03674	0.18727	0.8023	0.11859
(d) Class D					
	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.00943**	**0.00188**	**0.04145**	**0.00517**	**0.01213**
LSTM	0.02032	0.00464	0.05192	0.01976	0.02659
GRU	0.03531	0.00471	0.06658	0.32383	0.03742
CNN	0.03576	0.03396	0.06862	0.49758	0.03509
MLP	0.11089	0.03635	0.18428	0.70503	0.11089
(e) Class E					

input, output, and forget). By comparing the two models using the different evaluation metrics, it can be concluded that the LSTM model had good memory for longer sequences as compared to the GRU model, and it outperformed in the tasks requiring modelling the long-distance relations.

The CNN produced quite impressive results because of the ability of its convolutional layer in identifying patterns between the time steps. Contrary to the LSTM model, the CNN model is not recurrent, and it can only train the data that are inputted by the model at a particular time step.

Unlike the other models, the multilayers perceptron model performed worse. The model received inputs and didn't treat them as sequence data, which led to temporal dependencies and sequence patterns loss.

6.5.3. Comparison between Machine Learning and Deep Learning Models. It can be inferred from the obtained results that the deep learning models outperformed all the machine learning time series prediction models. From the different results of the different models, we noticed that CNN-LSTM gave the best performance measures and achieved the most accurate prediction results. Furthermore, Figure 6 shows that the dashed line of predicted values almost coincides with the one of real values, which proves that the hybrid CNN-LSTM model generated good results.

Figure 7 shows a comparison between the two best machine learning and deep learning models of Class A, and it can be noticed that CNN-LSTM slightly outperformed the XGBoost model.

Table 10: Machine learning for historical data with weather data evaluation results.

	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.07077**	**0.00944**	**0.09843**	**0.10385**	**0.07077**
VAR	1.05296	1.192	1.480543	0.43748	1.12674
SVM	1.08711	1.60693	1.68354	1.79101	1.1896
KNN	1.19562	2.14944	1.7146	1.90031	1.44379
(a) Class A					
	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.09003**	**0.02233**	**0.1498**	0.14143	**0.05296**
VAR	0.44379	0.36095	0.60079	**0.13328**	0.09003
SVM	0.72164	1.03892	1.02093	1.14312	0.69041
KNN	0.76347	1.4106	1.18769	1.23748	0.72168
(b) Class B					
	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.11705**	0.03418	**0.18679**	**0.27058**	**0.11705**
VAR	0.43726	**0.34063**	0.58364	0.41371	0.43726
SVM	0.80968	0.98371	0.99181	1.12363	0.54982
KNN	0.83909	1.86221	1.36451	1.22429	0.83915
(c) Class C					
	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.02945**	**0.00307**	**0.05401**	**0.03076**	**0.02951**
VAR	0.43389	0.34523	0.58756	0.1595	0.43389
SVM	0.54583	0.69474	0.83312	1.16663	0.54594
KNN	0.61329	1.14941	1.07194	1.18616	0.81517
(d) Class D					
	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.01711**	**0.00215**	**0.04635**	**0.00462**	**0.01711**
VAR	0.08469	0.08491	0.29139	0.1393	0.08519
SVM	0.13407	0.11862	0.34422	0.52147	0.08471
KNN	0.2732	0.13283	0.36446	1.94885	0.37281
(e) Class E					

6.5.4. The Computational Time. The computational time of various machine and deep learning models can be found in the following tables:

Table 14 shows that XGBoost has faster computational time while SVM is the more demanding. For deep learning models, Table 15 shows that the computational time of CNN-LSTM is bigger than the LSTM, GRU, CNN, and MLP models.

Machine learning models exhibit faster computational time, while deep learning models take a longer time because of their high number of parameters and their complex mathematical formulas.

Table 11: Deep learning for historical data with weather data evaluation results.

	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.03974**	**0.00342**	**0.05852**	**0.07208**	**0.046**
LSTM	0.06407	0.00548	0.09118	0.07627	0.06007
GRU	0.06604	0.00809	0.08993	0.07832	0.07504
CNN	0.06647	0.00831	0.14089	0.13935	0.07765
MLP	0.11436	0.01985	0.12441	0.30888	0.11695

(a) Class A

	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.00449**	**0.00505**	**0.00639**	**0.02946**	**0.00449**
LSTM	0.03302	0.01164	0.00767	0.02954	0.03342
GRU	0.00712	0.01249	0.04986	0.10656	0.03386
CNN	0.03309	0.01212	0.0738	0.27765	0.04365
MLP	0.08257	0.01421	0.1101	0.32499	0.08257

(b) Class B

	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.02061**	**0.00111**	**0.03333**	**0.06524**	**0.02502**
LSTM	0.0248	0.00206	0.04984	0.07459	0.02668
GRU	0.02939	0.00247	0.04974	0.28738	0.03112
CNN	0.09692	0.00248	0.04984	0.22379	0.034
MLP	0.12534	0.01729	0.1315	0.94313	0.09692

(c) Class C

	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.0076**	**0.00354**	**0.0142**	0.20089	**0.00822**
LSTM	0.01172	0.00418	0.02168	0.27274	0.01112
GRU	0.01203	0.01016	0.02078	**0.17646**	0.0118
CNN	0.06987	0.01151	0.10078	0.45685	0.06987
MLP	0.07914	0.02542	0.10539	0.5951	0.07914

(d) Class D

	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.00598**	**0.00172**	**0.04102**	**0.00427**	**0.00943**
LSTM	0.01912	0.00461	0.04713	0.01093	0.03641
GRU	0.0316	0.00468	0.06813	0.19444	0.03412
CNN	0.03492	0.01408	0.06839	0.29567	0.03855
MLP	0.09834	0.01498	0.11865	0.62538	0.09834

(e) Class E

Table 12: Machine learning for combined features evaluation results.

	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.06984**	**0.00849**	**0.09212**	**0.10335**	**0.06984**
VAR	1.0498	1.17777	1.47572	0.39307	0.3548
SVM	1.08607	1.59434	1.66544	1.78393	1.12886
KNN	1.18772	2.08258	1.7107	1.89279	1.44327

(a) Class A

	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.08978**	**0.0223**	**0.14396**	**0.11954**	**0.0498**
VAR	0.44327	0.36056	0.60046	0.13314	0.08978
SVM	0.7216	1.03716	1.01919	1.14185	0.61201
KNN	0.76191	1.41458	1.18536	1.23693	0.72143

(b) Class B

	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.11671**	**0.03406**	**0.18643**	**0.25127**	**0.11671**
VAR	0.43631	0.33916	0.58238	0.38869	0.43631
SVM	0.80446	0.97407	0.98695	1.12311	0.27791
KNN	0.839	1.86212	1.36414	1.22258	0.53914

Table 12: Continued.

	MAE	MSE	RMSE	MAPE	RMSLE
			(c) Class C		
	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.01955**	**0.00291**	**0.05102**	**0.02952**	**0.0295**
VAR	0.43306	0.34416	0.58665	0.15809	0.43306
SVM	0.54572	0.69482	0.83307	1.16535	0.54581
KNN	0.61314	1.14905	1.071	1.18552	0.43926
			(d) Class D		
	MAE	MSE	RMSE	MAPE	RMSLE
XGBoost	**0.01691**	**0.00204**	**0.04636**	**0.00572**	**0.01591**
VAR	0.08465	0.08488	0.29134	0.13918	0.07416
SVM	0.1173	0.11817	0.34376	0.52012	0.0847
KNN	0.27281	0.13251	0.16539	1.94874	0.17384
			(e) Class E		

Table 13: Deep learning for combined features evaluation results.

	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.03918**	**0.00334**	**0.05777**	**0.01527**	0.13296
LSTM	0.06199	0.00436	0.08394	0.02852	**0.04233**
GRU	0.06222	0.00712	0.08944	0.06087	0.06018
CNN	0.06213	0.008	0.08436	0.13451	0.06741
MLP	0.06401	0.00818	0.09029	0.13296	0.07424
			(a) Class A		
	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.00237**	**0.00101**	**0.00315**	**0.01554**	**0.0029**
LSTM	0.0029	0.00151	0.00364	0.02853	0.03108
GRU	0.00365	0.00245	0.04954	0.10156	0.03227
CNN	0.03153	0.00894	0.05426	0.18847	0.00337
MLP	0.06972	0.00401	0.09456	0.18756	0.06972
			(b) Class B		
	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.01766**	**0.00108**	**0.03282**	**0.01725**	**0.02415**
LSTM	0.00252	0.00142	0.04932	0.07215	0.02534
GRU	0.02914	0.00232	0.09914	0.151	0.02541
CNN	0.07412	0.00243	0.05017	0.17203	0.02956
MLP	0.06779	0.01018	0.10765	0.42891	0.07412
			(c) Class C		
	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.00701**	**0.00147**	**0.01351**	**0.14796**	0.00818
LSTM	0.01175	0.0042	0.02069	0.28051	**0.00795**
GRU	0.00783	0.00443	0.02069	0.15548	0.01206
CNN	0.0584	0.00519	0.0783	0.28875	0.0584
MLP	0.04972	0.00613	0.07204	0.59483	0.04972
			(d) Class D		
	MAE	MSE	RMSE	MAPE	RMSLE
CNN-LSTM	**0.0027**	**0.0016**	0.04046	**0.0009**	**0.0027**
LSTM	0.01707	0.00454	**0.03545**	0.0059	0.02613
GRU	0.03159	0.00443	0.06789	0.19402	0.03386
CNN	0.0314	0.00412	0.06813	0.28588	0.03419
MLP	0.04576	0.00488	0.06422	0.426525	0.04576
			(e) Class E		

6.6. Features Investigations. All the used time series prediction models showed that the prediction results were more accurate when we used the different features categories, namely, the past usage levels feature (e.g., number of car-sharing transactions), temporal information features (e.g., season, weekdays/weekend), and environmental condition features (e.g., temperature, humidity, wind speed, pressure, precipitation, weather conditions) together.

It also showed that the environmental condition features dominated other features, and it was followed by the

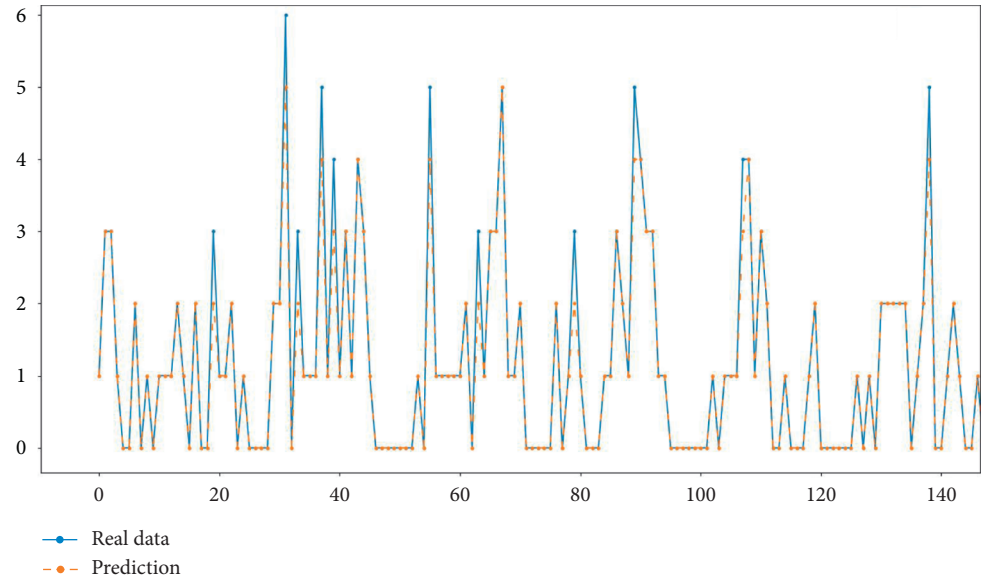

Figure 6: Comparison of the predicted value and the real value for CNN-LSTM.

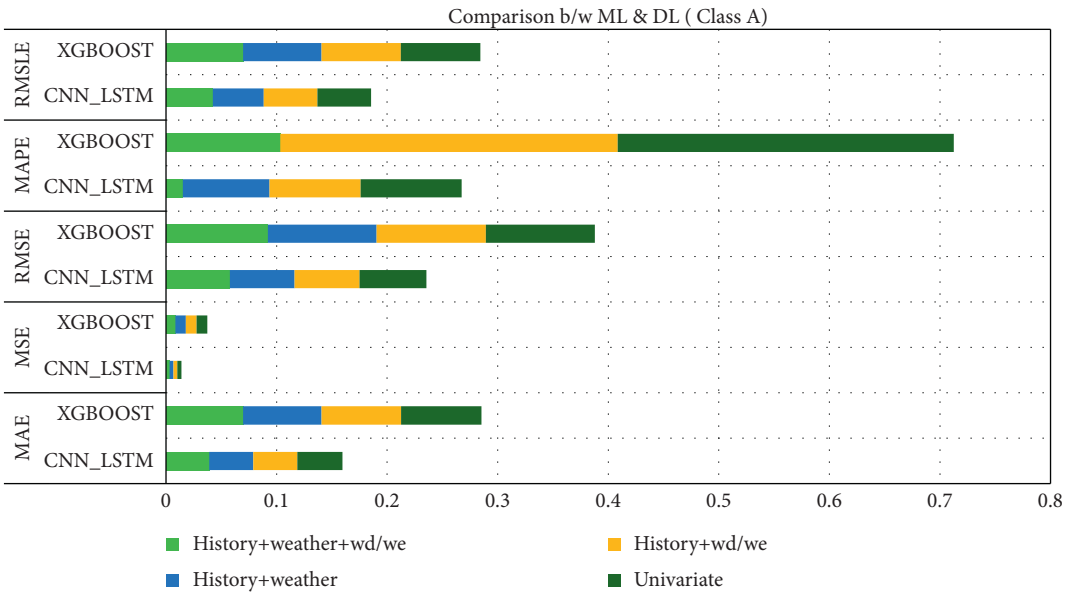

Figure 7: Comparison between best machine learning and deep learning models.

Table 14: Computational time for machine learning models.

Deep learning	Time (s)
XGBoost	10.2203
VAR	34.5798
SVM	65.2421
KNN	20.4125

Table 15: Computational time for deep learning models.

Deep learning	Time (s)
CNN-LSTM	872.2856
LSTM	543.7666
GRU	542.7666
CNN	395.3688
MLP	262.8907

temporal information features. We got the worst results when using only the number of car-sharing transactions based on the data from a Chongqing's car-sharing operator.

7. Conclusions

This research paper, through applying different machine learning and deep learning models to multivariate time series, aims to predict the car usage and to investigate the factors that help to improve the predictions' accuracy.

The evaluation of the different machine learning and deep learning models with MAE, MSE, RMSE, MAPE, and RMSLE reveals that the hybrid model (CNN-LSTM) gives substantially smaller errors as compared to the standalone used models. The experimental results show that the

utilization of the CNN-LSTM model on the number of car-sharing transactions, along with environmental conditions and temporal information features together, yields the highest prediction accuracy. The principal idea of the hybrid model is to efficiently amalgamate the advantages of two deep learning techniques. It exploits the ability of convolutional layers for extracting useful knowledge and learning the internal representation of time series data as well as the effectiveness of long short-term memory (LSTM) layers for remembering events for short and very long time.

Furthermore, through our experimental analysis, we conclude that even though LSTM models constitute an efficient choice for car-sharing time series prediction, their usage along with additional convolutional layers provides a significant boost in enhancing the forecasting performance. Although CNN-LSTM requires high search time due to its sensitivity to various hyperparameters and its high complexity, it shows the highest forecasting accuracy and the best performance.

All the results of the used models confirm that the car-rental usage is more sensitive to environmental conditions than temporal information that means the impact of weather on car-rental transportation deserves more attention at research. However, our work is limited to temporal features. Future studies can extend on adding more features such as the time span of data, spatiotemporal variables, and expand the model to consumers' habits [73–76].

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this study.

Acknowledgments

This research work was funded by the Beijing Municipal Natural Science Foundation (Grant no. 4212026) and National Science Foundation of China (Grant no. 61772075). The authors are thankful to them for their financial support.

References

[1] S. Petit, "World vehicle population rose 4.6% in 2016," *Wards Intelligence*, vol. 17, Jun. 08, 2020, https://wardsintelligence.informa.com/WI058630/World-Vehicle-Population-Rose-46-in-2016.

[2] T. Russell, "Why cars will always Be A main form of transportation," Jun. 08, 2020, https://www.forbes.com/sites/quora/2019/07/30/why-cars-will-always-be-a-main-form-of-transportation/#70d8cab72ab8.

[3] S. Formentin, A. G. Bianchessi, and S. M. Savaresi, "On the prediction of future vehicle locations in free-floating car sharing systems," *2015 IEEE Intelligent Vehicles Symposium (IV)*, vol. 2015, pp. 1006–1011, 2015.

[4] L. Cagliero, S. Chiusano, E. Daraio, and P. Garza, "CarPredictor: forecasting the number of free floating car sharing vehicles within restricted urban area," in *Proceedings of the 2019 IEEE International Congress on Big Data, BigData Congress*, pp. 72–76, Milan, Italy, July 2019.

[5] S. Hosseini, *Data-driven Time Series Demand Forecasting of Car-Sharing Services*, The case study of DriveNow in Munich, Germany, 2018.

[6] A. de Lorimier and A. M. El-Geneidy, "Understanding the factors affecting vehicle usage and availability in carsharing networks: a case study of communauto carsharing system from montréal, Canada," *International Journal of Sustainable Transportation*, vol. 7, no. 1, pp. 35–51, Jun. 2012.

[7] J. Burrell, "How the machine 'thinks': understanding opacity in machine learning algorithms," *Big Data & Society*, vol. 3, no. 1, p. 205395171562251, Jan. 05, 2016.

[8] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP Journal on Applied Signal Processing*, Springer International Publishing, vol. 2016, no. 1, p. 67, Dec. 01, 2016.

[9] G. Battineni, N. Chintalapudi, and F. Amenta, ""Machine learning in medicine: performance calculation of dementia prediction by support vector machines (SVM)," Informatics Med," *Unlocked*, vol. 16, Article ID 100200, Jan. 2019.

[10] E. E. Vos, P. S. Francois Luus, C. J. Finlay, and B. A. Bassett, "A generative machine learning approach to RFI mitigation for radio astronomy," *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, vol. 2019, pp. 1–6, 2019.

[11] M. Mahmud, M. S. Kaiser, A. Hussain, and S. Vassanelli, "Applications of deep learning and reinforcement learning to biological data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2063–2079, 2018.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Aug. 15, 2020, http://image-net.org/challenges/LSVRC/2015/.

[13] A. Agrawal et al., "VQA: visual question answering," Aug. 15, 2020.

[14] H. Peng, J. Li, Y. Song, and Y. Liu, "Incrementally learning the hierarchical softmax function for neural language models.," Aug. 15, 2020, http://www.aaai.org.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[16] J. Li, H. Peng, L. Liu et al., "Graph CNNs for urban traffic passenger flows prediction," in *Proceedings of the 2018 IEEE SmartWorld, Ubiquitous Intell. Comput. Adv. Trust. Comput. Scalable Comput. Commun. Cloud Big Data Comput. Internet People Smart City Innov. SmartWorld/UIC/ATC/ScalCom/CBDCo*, pp. 29–36, Guangzhou, China, October 2018.

[17] C. Morency, M. Trépanier, B. Agard, B. Martin, and J. Quashie, "Car sharing system: what transaction datasets reveal on users' behaviors," in *Proceedings of the IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 284–289, Enshi, China, September 2007.

[18] Y. Liu, C. Liu, X. Lu, M. Teng, H. Zhu, and H. Xiong, "Point-of-Interest demand modeling with human mobility patterns," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 947–955, Halifax NS Canada, August 2017.

[19] C. Boldrini, R. Bruno, and M. Conti, "Characterising demand and usage patterns in a large station-based car sharing system," *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, vol. 2016, pp. 572–577, 2016.

[20] M. Castro-Neto, Y.-S. Jeong, M.-K. Jeong, and L. D. Han, "Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions," *Expert Systems with Applications*, vol. 36, no. 3, pp. 6164–6173, Apr. 2009.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] J.-X. Xu and J. S. Lim, "A new evolutionary neural network for forecasting net flow of a car sharing system," in *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, pp. 1670–1676, Singapore, Sep. 2007.

[23] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: deep belief networks with multitask learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2191–2201, 2014.

[24] Y. Tian and L. Pan, "Predicting short-term traffic flow by long short-term memory recurrent neural network," in *Proceedings of the 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, pp. 153–158, Chengdu, China, December 2015.

[25] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 187–197, May 2015.

[26] X. Mo, Y. Xing, and C. Lv, "Interaction-aware trajectory prediction of connected vehicles using CNN-LSTM networks," *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, vol. 2020, pp. 5057–5062, 2020.

[27] "Introduction to boosted trees — xgboost 1.2.0-SNAPSHOT documentation," Jun. 08, 2020, https://xgboost.readthedocs.io/en/latest/tutorials/model.html.

[28] K. Kavaklioglu, "Modeling and prediction of Turkey's electricity consumption using Support Vector Regression," *Applied Energy*, vol. 88, no. 1, pp. 368–375, Jan. 2011.

[29] "Vector autoregression (VAR) - comprehensive guide with examples in Python - machine learning plus," Jun. 08, 2020, https://www.machinelearningplus.com/time-series/vector-autoregression-examples-python/.

[30] "11.2 Vector Autoregressive models VAR(p) models | STAT 510," Jun. 08, 2020, https://online.stat.psu.edu/stat510/lesson/11/11.2.

[31] C.-H. Wu, J.-M. Ho, and D. T. Lee, "Travel-time prediction with support vector regression," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 276–281, 2004.

[32] C. Hu, G. Jain, P. Zhang, C. Schmidt, P. Gomadam, and T. Gorka, "Data-driven method based on particle swarm optimization and k-nearest neighbor regression for estimating capacity of lithium-ion battery," *Applied Energy*, vol. 129, pp. 49–55, 2014.

[33] J. Friedman, R. Tibshirani, and T. Hastie, "Additive logistic regression: a statistical view of boosting," *The Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[34] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[35] T. Chen and T. He, "xgboost: eXtreme Gradient Boosting," 2020, https://medium.com/m/global-identity?redirectUrl=https%3A%2F%2Ftowardsdatascience.com%2Fxgboost-extreme-gradient-boosting-how-to-improve-on-regular-gradient-boosting-5c6acf66c70a.

[36] J. Brownlee, "A gentle introduction to XGBoost for applied machine learning," Jun. 08, 2020, https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/.

[37] "Distance metrics and K-nearest neighbor (KNN) | by luigi fiori | medium," Aug. 23, 2021.

[38] F. Modaresi, S. Araghinejad, and K. Ebrahimi, "A comparative assessment of artificial neural network, generalized regression neural network, least-square support vector regression, and K-nearest neighbor regression for monthly streamflow forecasting in linear and nonlinear conditions," *Water Resources Management*, vol. 32, no. 1, pp. 243–258, 2018.

[39] P. Zhou, W. Shi, J. Tian et al., "Attention-based bidirectional long short-term memory networks for relation classification," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany, August 2016.

[40] J. Jian Zheng, C. Cencen Xu, Z. Ziang Zhang, and X. Xiaohua Li, "Electric load forecasting in smart grids using Long-Short-Term-Memory based Recurrent Neural Network," in *Proceedings of the 2017 51st Annual Conference on Information Sciences and Systems, CISS 2017*, pp. 1–6, Baltimore, MD, USA, May 2017.

[41] T.-T.-H. Le, J. Kim, and H. Kim, "Classification performance using gated recurrent unit Recurrent Neural Network on energy disaggregation," in *Proceedings of the International Conference on Machine Learning and Cybernetics*, pp. 105–110, Jeju, Korea (South), Jul. 2016.

[42] J. Li, X. Wang, Y. Zhao, and Y. Li, "Gated recurrent unit based acoustic modeling with future context,," 2018, https://arxiv.org/abs/1805.07024.

[43] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," pp. 1–11, 2015, http://arxiv.org/abs/1511.08458.

[44] "Convolutional layer - an overview | ScienceDirect topics," Aug. 23, 2021, https://www.sciencedirect.com/topics/engineering/convolutional-layer.

[45] S. Albawi, T. A. M. Mohammed, and S. Alzawi, "Layers of a convolutional neural network," *IEEE*, 2017.

[46] "A comprehensive guide to convolutional neural networks — the ELI5 way | by sumit saha | towards data science," Aug. 23, 2021, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[47] T. Li, M. Hua, and X. Wu, "A hybrid CNN-LSTM model for forecasting particulate matter (PM2.5)," *IEEE Access*, vol. 8, pp. 26933–26940, 2020.

[48] I. E. Livieris, E. Pintelas, and P. Pintelas, "A CNN-LSTM model for gold price time-series forecasting," *Neural Computing & Applications*, vol. 32, no. 23, pp. 17351–17360, 2020.

[49] "Multilayer perceptron - an overview | ScienceDirect topics," Aug. 23, 2021, https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron.

[50] "A beginner's guide to multilayer perceptrons (MLP) | pathmind," Aug. 23, 2021, https://wiki.pathmind.com/multilayer-perceptron.

[51] H. H. Tan and K. H. Lim, "Vanishing gradient mitigation with deep learning neural network optimization," in *Proceedings of the 2019 7th Int. Conf. Smart Comput. Commun. ICSCC 2019*, pp. 1–4, Sarawak, Malaysia, June 2019.

[52] "Vanishing gradient problem - Wikipedia," Aug. 23, 2021, https://en.wikipedia.org/wiki/Vanishing_gradient_problem.

[53] J. Kolbusz, P. Rozycki, and B. M. Wilamowski, "The study of architecture MLP with linear neurons in order to eliminate the "vanishing gradient" problem," *Artificial Intelligence and Soft Computing*, vol. 10245, no. 2013, pp. 97–106, 2017.

[54] *Root Mean Squared Log Error - Machine Learning with Spark - Second Edition [Book]*, https://www.oreilly.com/library/view/machine-learning-with/9781785889936/4317943c-9452-4013-99b9-d267a2820b23.xhtml, Aug. 23, 2021.

[55] X. Ma, R. Cao, and Y. Jin, "Spatiotemporal clustering analysis of bicycle sharing system with data mining approach," *Information*, vol. 10, no. 5, pp. 1–14, 2019.

[56] "3 ways to encode categorical variables for deep learning," Jun. 09, 2020, https://machinelearningmastery.com/how-to-prepare-categorical-data-for-deep-learning-in-python/.

[57] G. Petneházi, "Recurrent neural networks for time series forecasting," *International Journal of Forecasting*, vol. 37, no. 1, pp. 388–427, 2021.

[58] "How to create a grouped frequency table," Jun. 09, 2020, https://sciencing.com/create-grouped-frequency-table-5531910.html.

[59] "Grouped frequency distribution," Jun. 09, 2020, https://www.mathsisfun.com/data/frequency-distribution-grouped.html.

[60] G. P. Zhang and M. Qi, "Neural network forecasting for seasonal and trend time series," *European Journal of Operational Research*, vol. 160, no. 2, pp. 501–514, 2005.

[61] M. Theses and B. T. Ojemakinde, "Trace: Tennessee research and creative exchange support vector regression for non-stationary time series," 2006, https://trace.tennessee.edu/utk_gradthes/1756.

[62] S. Arslankaya and V. Öz, "Time series analysis of sales quantity in an automotive company and estimation by artificial neural networks," *Sakarya University Journal of Science*, vol. 22, no. 3, p. 1, 2018.

[63] C. Chen, J. Twycross, and J. M. Garibaldi, "A new accuracy measure based on bounded relative error for time series forecasting," *PLoS One*, vol. 12, no. 3, p. e0174202, Article ID 0174202, 2017.

[64] J. Brownlee, "Time series forecasting performance measures with Python," Jun. 09, 2020, https://machinelearningmastery.com/time-series-forecasting-performance-measures-with-python/.

[65] S. Kim and H. Kim, "A new metric of absolute percentage error for intermittent demand forecasts," *International Journal of Forecasting*, vol. 32, no. 3, pp. 669–679, 2016.

[66] U. Khair, H. Fahmi, S. A. Hakim, and R. Rahim, "Forecasting error calculation with mean absolute deviation and mean absolute percentage error," *Journal of Physics: Conference Series*, vol. 930, no. no. 1, p. 012002, 2017.

[67] O. P. Abimbola, A. R. Mittelstet, T. L. Messer, E. D. Berry, S. L. Bartelt-Hunt, and S. P. Hansen, "Predicting *Escherichia coli* loads in cascading dams with machine learning: an integration of hydrometeorology, animal density and grazing pattern," *The Science of the Total Environment*, vol. 722, p. 137894, 2020.

[68] W. Wei, "Granger causality test - an overview | ScienceDirect topics," Jun. 09, 2020, https://www.sciencedirect.com/topics/social-sciences/granger-causality-test.

[69] Y. Ding, X. Song, and Y. Zen, "Forecasting financial condition of Chinese listed companies based on support vector machine," *Expert Systems with Applications*, vol. 34, no. 4, pp. 3081–3089, 2008.

[70] S. S. Eslamian, S. A. Gohari, M. Biabanaki, and R. Malekian, "Estimation of monthly pan evaporation using artificial neural networks and support vector machines," *Journal of Applied Sciences*, vol. 8, no. 19, pp. 3497–3502, 2008.

[71] F. Martínez, M. P. Frías, M. D. Pérez, and A. J. Rivera, "A methodology for applying k-nearest neighbor to time series forecasting," *Artificial Intelligence Review*, vol. 52, no. 3, pp. 2019–2037, 2019.

[72] I. Koprinska, D. Wu, and Z. Wang, ""Convolutional neural networks for energy time series forecasting," proc. Int. Jt. Conf," *Neural Networks*, vol. 2018, 2018.

[73] J. Kang, K. Hwang, and S. Park, "Finding factors that influence carsharing usage: case study in seoul," *Sustainability*, vol. 8, no. 8, p. 709, Jul. 2016.

[74] J. Firnkorn and M. Müller, "What will be the environmental effects of new free-floating car-sharing systems? The case of car2go in Ulm," *Ecological Economics*, vol. 70, no. 8, pp. 1519–1528, Jun. 2011.

[75] S. de Luca and R. Di Pace, "Modelling users' behaviour in inter-urban carsharing program: a stated preference approach," *Transportation Research Part A: Policy and Practice*, vol. 71, pp. 59–76, Jan. 2015.

[76] J. Yuan, Y. Zheng, and X. Xie, ""Discovering regions of different functions in a city using human mobility and POIs," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Halifax NS Canada, June 2020.