



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Machine learning based forecasting for renewable resources

by

Shenglin Su

Supervisor: Terry Walcott

Second Marker:

Project submitted as part of the requirements for
the award of MSc Data and Artificial Intelligence

September 2022

DECLARATION

I, **Shenglin Su**, declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed :

A handwritten signature in black ink that reads "Shenglin Su".

Date :

A handwritten date in black ink that reads "07/09/2022".

FORM OF CONSENT

I, **Shenglin Su**, hereby consent that my Project, submitted in candidature for the M.Sc. Data and Artificial Intelligence degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed :

A handwritten signature in black ink that reads "Shenglin Su".

Date :

A handwritten date in black ink that reads "07/09/2022".

Abstract

The concepts of "Peak carbon dioxide emissions" and "Carbon neutrality" have come to represent the international agreement in the context of phasing out fossil fuels and reducing global warming. Renewable energy is crucial as a means of achieving the two carbon reduction objectives mentioned above. However, since renewable resources are unpredictable and erratic, managing them effectively has emerged as one of the major issues limiting their growth. To overcome this challenge, this study employs three time series forecasting algorithms to anticipate its resources in short term. First, common resources, machine learning methods, and systems for renewable energy are presented. The approaches used in this paper—Prophet, recurrent neural network, and long-short term memory neural network—are then suggested based on the prior literature review. Then case studies are carried out, and the training, validation, testing, and assessment of machine learning models are finished. This is done using historical data on renewable resources of a certain location. Finally, comparison and analysis of the prediction outcomes produced by various algorithm models are conducted. The findings demonstrate that the long-short-term neural network has a strong short-term prediction impact in the area of renewable energy.

Table of Contents

1. Introduction	1
2. Literature review	4
3. Research methodology	11
3.1 Python-related components	11
3.2 Time series forecasting models	16
3.3 Data collection and preprocessing	23
3.4 Model training and evaluation	26
4. Results and discussion	29
4.1 Case A: Prophet forecasting model for renewable resources	29
4.2 Case B: RNN forecasting model for renewable resources	31
4.3 Case C: LSTM forecasting model for renewable resources	35
4.4 Discussion	37
5. Conclusions and recommendations	39
6. Reflection	39
Reference list	40
Appendices	43
A. Prophet	43
B. RNN	45
C. LSTM	47

1. Introduction

Fossil fuels are utilized extensively as industrial civilization develops, producing copious amounts of carbon dioxide that have a significant warming effect on the environment. The melting of glaciers, the rise in sea level, the incidence of severe weather events, and even direct effects on human physiology are all caused by the increase in global temperature. In light of this, it is suggested that "Peak carbon dioxide emissions" and "Carbon neutrality" be used as a basic solution to the issue, which has attracted the attention of governments all over the globe [1].

Renewable energy (RE) is now being actively developed in order to meet these two carbon-related objectives [2]. Renewable energy sources include solar, wind, hydraulic, biomass, tidal, geothermal, and other forms of clean and environmentally friendly energy. It is often used in its early phases on a modest scale for the production of hydroelectric power and the provision of electricity in outlying locations. The use of renewable energy is expanding and being progressively incorporated into the power grid as a result of the recent fast growth of wind energy and solar energy. The forecast and management of renewable resources, however, are especially crucial since RE are often underutilized in this process because of their unpredictability and uncertainty [3].

Time series data is thought of as a special kind of data with the following characteristics [4]. 1) The relationship between the data value at the present instant and the data value at the previous moment is the main distinction between time series data and other forms of data. This characteristic shows that the development and change rules of the present or future data, which mostly comprise trend, periodicity, and irregularity, have been inferred by the previous data. The direction of a time series' development over an extended period of time is reflected in its trend, which may be shown as an ongoing upward, downward, or constant trend over a sizable period of time. A periodic fluctuation with definite length and amplitude is known as periodicity and is brought on by numerous periodic causes. The irregularity depicts time series changes that are non-trending and non-periodic as a result of different unforeseen occurrences and unintentional influences. 2) Time series' stationarity and non-

stationarity. The non-stationarity of the time series suggests that the mean and variance will fluctuate over time, while the stability of the time sequence shows that the mean and variance of the time series do not change consistently from time to time. In other words, the stationarity of time series guarantees that the fundamental properties of time series continue into the future in addition to the present. 3) The size of time series data is always expanding. On the one side, as hardware technology continues to advance in a variety of ways, data sampling frequency in real-world applications also keeps rising, which extends the length of the time series. Dimensional catastrophe and other issues are inescapable when time series are treated as straightforward one-dimensional vector data. On the other hand, time series data are not just one-dimensional data but often comprise a set of values and linkages between these values in many practical applications. New criteria for time series prediction were presented by multidimensional time series.

Specifically, renewable energy prediction models are one kind of time series prediction model. Physical models, statistical models, and machine learning (ML) models are the three categories under which renewable energy prediction models fall as one sort of time series prediction model [5]. Physical models often refer to a type of model, such as numerical weather prediction (NWP), that employs computers to solve physical equations. Statistical models, such as the autoregressive moving average (ARMA) model, are predictions derived by statistical analysis of various input variables. Support vector machines (SVM), artificial neural networks (ANN), recurrent neural networks (RNN), and other ML models employ a lot of data to train and learn the principles of many phenomena. Classical statistical models predict future trends by studying the evolution of time data. Machine learning models have a strong nonlinear fitting ability and have been widely utilized in time series analysis in recent times. With the advent of the era of big data, multivariable and multi-channel massive time series data are growing exploitatively. And multivariate time series data are characterized by high and Spatio-temporal correlation or contain noisy data, which makes it difficult for classical statistical methods to effectively model. In addition, traditional methods also have limitations when dealing with big data, especially

massive and complex multivariable temporal data. However, deep learning (DL) models have a strong adaptive ability, flexible nonlinear modeling ability, massive learning and parallel computing ability, and are effective methods to solve complex nonlinear questions [6]. In theory, they have similarities with statistical models, but they are more effective at resolving nonlinear issues. They offer the benefits of quicker computation and lower cost in short-term prediction compared to physical models [7]. As a result, the major focus of this study is on machine learning algorithms that enable renewable resource prediction.

The forecasts of renewable resources-related papers are discussed and summarized in this research. The technique for this work is then proposed, based on the literature review, and includes the time series prediction models like Prophet, RNN and LSTM. Following that, two case studies of forecasting for renewable resources—one for solar energy and one for wind energy—will be conducted. Various time series forecasting models will be trained, verified, tested, and assessed throughout this process. Finally, in order to get a general conclusion, the generated prediction results from various machine learning models will be compared and examined. It is important to note that this study predicts renewable resources using a range of machine learning techniques, including deep learning means and compares the outcomes. This reference may be used in further research to increase the predictability of renewable resources.

2. Literature review

Time series forecasting could be categorized as long period, medium period, short period, and very-short period predictions in the subject of renewable energy projection. When it comes to power market pricing and real-time monitoring, very-short-period prediction often refers to a window of time between a few seconds and half an hour [8]. A short-period forecast often refers to the window of time between half an hour and one day [9]. It is used to accomplish daily load adjustment and power grid security. A few days to a month's worth of medium-term projections may be utilized to create maintenance schedules that will result in the lowest possible operational expenses [10]. The phrase "long-period forecast" relates to a few months to a year, and it is used to aid in yearly planning, power plant selection, and bidding [11]. Each temporal dimension may be divided in any number of ways, and this number can be dynamically altered depending on the circumstances of the application. Short-term time series prediction is selected as the primary research strategy in this study due to the volume of data and upcoming work.

Solar energy has lately gained a lot of popularity since it can be used indefinitely from local resources, has no moving parts, and is simple to install and maintain. The precise estimation of its producing capability, however, impedes its future expansion. ML techniques work effectively to address this issue. A short-term solar irradiance forecast approach built on SVM and adjusted for ramp-down events was put out by Cheng et al. [12]. The outcome demonstrates that SVM could have a decent prediction effect. And it is important to note that various built-in kernel functions of the algorithm provide distinct predictions with varied results. In order to predict the global solar radiation (GSR) in Tehran province, Iran, Ramedani et al. [13] employed radial basis function (RBF) and polynomial as the kernel function of SVM, respectively. The experimental findings demonstrate that the SVM's RBF kernel is superior at predicting GSR. Additionally, ANN has recently drawn a lot of notice. In order to train an ANN model, Ongsakul et al. [14] used historical intervals of solar irradiance, moving average relative humidity, moving average temperature, day of the year, and time of

day index as inputs and solar radiation as an output. ANN has a greater correlation coefficient and a smaller prediction error, according to the results. In order to accomplish the improvement of solar irradiation forecasting, Sivaneasan et al. [15] suggested an ANN model with fuzzy logic pre-processing. Additionally, this model has a better error correction factor that reduces prediction error by adding the error of the output anticipated in the first five minutes to the input layer. Besides, deep learning methods are also utilized to enhance solar forecasting. To forecast sun radiation accurately, Bamisile et al. [16] developed a deep neural network (DNN) model with many hidden layers. Jaihuni et al. [17] examined five different types of RNN, including basic RNN, gated recurrent unit (GRU), long-short term memory (LSTM), bidirectional GRU (BI-GRU), and bidirectional LSTM (BI-LSTM) when it comes to the estimation of solar irradiance. In general, basic and bidirectional RNN models are very resilient to low data and high temporal data fluctuation.

On the other side, wind energy, which is a readily available and plentiful kind of renewable energy, is seeing a meteoric rise in the rate at which it is being installed. In addition to this, it is considerably more susceptible to unpredictability and ambiguity than solar energy. The forecasting of the wind speed must be accurate in order to ensure its effective and appropriate use of it. The authors of Zhou et al. [18] predicted wind speed using a least-squares support vector machine (LS-SVM) and conducted a comprehensive modification of the model. It has been discovered that the prediction accuracy of LS-SVM is better than that of the persistence technique in the majority of instances, particularly in situations in which the wind speed demonstrates significant temporal variability. Kong et al. [19] used principal component analysis (PCA) to reduce dimensions and introduced the particle swarm optimization (PSO) algorithm to realize the parameters optimization of the SVM model in order to improve the accuracy of predictions. The findings of the simulation provided evidence that the strategy that was presented is successful. However, it is difficult for the forecast accuracy of a single model to achieve the standards that are required by the industry. Optimizing the model via an algorithm is one approach that may be used. For instance, the author of Reference [20] processed the data using the variational mode

decomposition (VMD) technique, and they employed the genetic algorithm (GA) to improve the artificial neural network prediction model. Using a hybrid model is still another approach that may be used. An adaptive neural fuzzy inference system (ANFIS) and ANN hybrid technique were suggested to increase the accuracy of prediction, as shown in references such as [21]. In addition to that, there are many approaches based on deep learning that is suggested for forecasting wind speed. In order to train the RNN and generate successful prediction results, Cao et al. [22] used wind data from many heights throughout the study area. Besides, Elsaraiti et al. [23] used LSTM to solve the RNN's disadvantage of short-term memory, which allowed them to tackle the issue of gradient information vanishing or exploding during the RNN training procedure.

Additionally, cutting-edge techniques in other areas of renewable energy forecasting, namely output power, user load, and electricity price prediction of renewable energy systems, may be leveraged to enhance solar and wind resource forecasting. 1) As for the output power forecasting of renewable energy systems, Heinemann et al. [24] suggested a heterogeneous combined ML strategy employing DT and SVM for the prediction of wind power production. The findings indicate that the suggested method has a faster computation time and a higher degree of accuracy. It can also be used to predict heterogeneous sets more accurately by taking into account more previous steps and making more spatiotemporal data available when using a higher-dimensional model. A hierarchical approach was put up in the literature [25] to increase forecasting accuracy by integrating the SVM and ANN algorithms. Instead of the conventional prediction model based on information about power production at the power plant grid level, this technique builds numerous tiny prediction models for various levels using the micro information of power plants, such as inverters. By comparing the outcomes of multi-step forward prediction, it can be discovered that the approach has more accuracy and universality. More deep learning techniques are also being used at the same time to forecast the output power of renewable energy systems. In Ref. [26], numerical weather prediction (NWP) is combined with a deep belief network (DBN), a deep learning technique with the benefit of processing complicated data and taking the spatial correlation of location and topography, to forecast wind

power. The findings demonstrate that this technique has superior prediction accuracy when compared to other deep neural networks like LSTM and GRU. 2) When it comes to the prediction of user load, ANN has been employed in the literature [27–29] to anticipate demand and supply concurrently in renewable energy systems. The findings demonstrate that the basic neural network topology, which has both generalizability and a quick running time, can often handle the prediction issue. It's interesting to note that article [30] also used the DT approach to accurately anticipate the energy demand level of buildings. When compared to other machine learning techniques, DT produces outcomes and a judgment process that is more explicative overall. There are other papers that use deep learning techniques for load predictions that might be used as references. Two traditional deep learning models, CNN and RNN, were put out in literature [31] to forecast the multi-step load of commercial buildings. According to the simulation findings, the gated 24h CNN model is much more accurate than seasonal ARIMAX. 3) To forecast day-ahead power prices, Windler et al. [32] employed the weighted closest neighbor approach, the TBATS method, and the deep feedforward neural network (DFNN). The findings reveal that although both techniques had respectable accuracy, DFNN performed better, offering accurate forecasts up to a month in advance. A superior iterative training technique based on ANN was presented in the literature [33] for short-term power price prediction, which performs better in simulation. A dynamic choice ANN (DCANN) that could dynamically identify poor samples was suggested to set up an energy price prediction system in reference [34]. It has been discovered that this strategy efficiently addresses the issue of multi-step prediction in advance while also improving the quality of input data.

The conventional approaches to predicting time series include the autoregressive integrated moving average (ARIMA) model, the exponential smoothing (ES) model, the state space model (SSM), and others. The seasonal ARIMA model was considered for the power load prediction by Hong et al. [35] in 2004, and they modeled and projected the power load that would be experienced by a community in Tianjin over the course of the next 15 days. According to the findings, the margin of error between

the anticipated value and the actual value was less than three percent, indicating that the prediction impact was successful. However, this paper only makes predictions for certain plots, and the robustness of the model as well as the application fields it can be used in deserve further discussion. Volkan and Sertac [36] employed a time-series model to anticipate Turkey's coal, oil, gas and other major energy loads. They did this while taking into consideration the fact that projected economic and demographic characteristics were sometimes quite different from reality. The ARIMA and Seasonal ARIMA (SARIMA) models that were developed both produced accurate predictions once they were put into use. For the purpose of the creation of the national energy system, corresponding policies and viewpoints have been put up. The dataset that was utilized in the study that was published in 2007 by Luo et al. [37] was the monthly average price of West Texas Intermediate (WTI) Crude Oil from 2003 to 2006. The ARIMA prediction model was constructed in order to make a forecast regarding the monthly WTI crude oil price in 2007, and the upper and lower limits of the prediction were given with 95% confidence. According to the findings of the projection, the price of WTI crude oil in 2007 moved about within a range of sixty to seventy dollars per barrel [37]. Pappas et al. [38] published an article in 2010 in which they offered a novel approach for power load prediction that was based on the Kalman filter and Auto ARMA models. In addition to this, it is compared to three other established methods of time series analysis, and the accuracy and usefulness of this approach are shown using the case study of power load forecasting in Greece. Ervural et al. [39] developed a combined prediction model of MA and ARMA in 2016, and they used a genetic algorithm to determine the p and q values in the ARMA model (P, Q). Using the consumption of natural gas in Turkey as the object of the prediction, the single model and the combination model are used to forecast respectively, and the results of the comparison reveal that the combination model has a greater prediction accuracy. In 2018, Li Da [40] discussed the benefits of conventional time series in oilfield production prediction and gathered daily oil production data of Xinjiang oilfield from June 13, 2016, to June 12, 2017, as a training set. This data was used to develop a model for predicting oilfield output. The ARIMA model, the back propagation (BP)

neural network model, and the ARIMA-BP model were developed in order to forecast the daily output of the Xinjiang oilfield over the following 20 days. The accuracy of these models' predictions was 98.16%, 98.86%, and 99.28%, respectively. The ARIMA-BP combined model took the best aspects of the ARIMA and BP models and merged them into one. When this model is compared with the ARIMA and BP models, its overall prediction effect is superior [40].

The aforementioned works are merely basic and straightforward applications when employing deep learning models to forecast time series data; they do not combine the properties of time series data. To solve the issue of predicting time series, we find that the hybrid model, which combines the properties of several types of time series data, is the most effective tool to utilize. When pre-testing time series data, literature [41, 42] uses a combination of the ARIMA model and the Multilayer Perceptron (MLP) algorithm and gets good effects. As a result of the advancement of deep network structures, the conventional neural network and recurrent neural network hybrid models are now being used to finish the forecasting task. A hybrid model known as Residual RNN (R2N2) was introduced in the aforementioned piece of literature [43] to forecast multivariate time series issues. R2N2 is first pretested by logarithmic data via a linear model F , like VAR, etc., and forecasted by RNN on the residual acquired by the linear model prediction. This process begins with the logarithmic data and continues through the linear model F . After adding model F to the residual value that was predicted by RNN, the final prediction result may be obtained. The experiment presented in this study just compares the effect difference generated by R2N2 and a single RNN and VAR, which demonstrates that R2N2 is capable of producing a more accurate prediction effect. Literature [44] suggested a novel deep learning frame called the long- and short-term time-series network (LSTNet) to handle the challenge of making accurate predictions based on multivariate time series data. LSTNet implements CNN for the purpose of determining the short-period dependency relationship between the data and RNN for the purpose of determining the long-term mode and trend of the data. LSTNet makes use of both CNN and RNN, bringing their respective benefits to the table. The LSTNet architecture may

be broken down into four distinct units: the volume product unit, the cycle unit, the cycle jump unit, and the autoregressive unit. The CNN and RNN functions are implemented, respectively, by the convolutional unit and the cyclic unit. The primary function of the loop jump unit is to effect modifications on the loop unit in order to accomplish effectively obtaining the periodic pattern of the data. In contrast to the general RNN, the cyclic jump unit's hidden layer makes a connection with the hidden layer in the time before a cycle rather than the hidden layer in the time before the cycle in the general RNN. This connection is made with the hidden layer in the time before the cycle. Because neural networks are insensitive to the size changes of input data, the primary goal of adding autoregressive units is to prevent neural networks from compromising the accuracy of prediction results. This may be accomplished by ensuring that autoregressive units are used.

In earlier research, solar and wind projections were often treated as separate entities. Despite this, they are the most widespread kind of renewable energy source, and they are often utilized together. In addition, since both wind speed and solar irradiance are dependent on large-scale meteorological phenomena, they exhibit a Spatio-temporal correlation [45]. This is the case because of the relationship between the two variables. As a result, there are some strategies that are useful for the implementation of both solar and wind power. According to the research published in [46, 47], when one model is used to forecast the behavior of two distinct renewable resources at the same time, a successful prediction effect is achieved. As a result, in this paper, three different approaches to machine learning are utilized to carry out case studies on solar radiation and wind speed respectively. The purpose of these case studies is to investigate the effects of various models as well as the differences and connections between the two scenarios.

3. Research methodology

3.1 Python-related components

A number of Python open-source frameworks and libraries have been utilized in building time series forecasting models in this paper, including Numpy, Pandas, Matplotlib, Scipy, Sklearn and Pytorch. First of all, it is necessary to introduce it:

(1) "Numeric Python" (NumPy) is the name of a Python open source library. It is made up of procedures for processing arrays and multi-dimensional array objects. The following operations may be carried out by developers using NumPy: mathematical and logic operations on arrays, Fourier transformations, routines for graphical operations, and linear algebra operations. A strong N-dimensional array object called Array, a reasonably developed broadcast function library, a toolbox for integrating C/C++ and Fortran code, practical linear algebra, Fourier transformations, and random number generating functions are the typical built-in features of NumPy. Besides, Scipy's sparse matrix operation package is easier to utilize with NumPy. Additionally, NumPy offers a wide range of complex numerical programming tools, including vector processing, matrix data types, and sophisticated arithmetic libraries. It was developed for precise numeric processing. Numerous major financial institutions as well as key scientific computer institutions like NASA and Lawrence Livermore utilize it to do jobs that would often be completed using M language, C++ or Fortran.

(2) Pandas is a third-party library for Python, which is created for data analysis issues. It is a NumPy-based tool that is often used in conjunction with NumPy and Matplotlib. Pandas integrates a variety of libraries and industry-standard data models to provide the tools necessary for the effective manipulation of massive datasets. In addition, Pandas gives you access to a variety of functions and techniques that make it possible to analyze data in a brisk and uncomplicated manner. Due in large part to this aspect, Python becomes a sophisticated and efficient data analysis environment. Pandas has two primary data types, namely Series and DataFrame, which is an important feature. Series is a data type similar to a one-dimensional array and

DataFrame is a two-dimensional or even multidimensional data type. These two data types are the foundation of Pandas. Pandas provides many functions for data analysis and manipulation, including basic data manipulation, operation of computing, operation of extracting data features, and operation of mining association relationships in data.

(3) Matplotlib is a key auxiliary tool in the whole data analysis. Its ability to properly interpret the data and alter our analysis techniques by visualization. Specially, it is a two-dimensions drawing package in Python that can generate high-quality graphics on various platforms in a diversity of hard copy formats and interactive environments to visualize data, make the data more intuitive, and make the data more objective and convincing. Matplotlib was developed by the Python Software Foundation and accessible for use in Python, IPython shells, Python scripts, Jupyter notebook, and other Web-based servers. In addition to that, it is an effort to make difficult things simpler to do and difficult things feasible. The generation of plots, histograms, power graphs, bar graphs, error graphs, scatter graphs, and other types of graphs just require a small part of writing codes. When coupled with IPython, the PyPlot module offers a user interface that is reminiscent of MATLAB and may be used for straightforward plotting. Through either an object-oriented interface or a collection of methods that are already well-known to MATLAB users, advanced users have complete control over the line styles, font settings, and axis parameters, among other things. In general, we can divide Matplotlib images into three layers: 1. The bottom container layer: mainly includes Canvas, Figure and Axes. An Axes represents a chart and contains a plot, a figure represents a canvas to draw a picture, a picture or a canvas can have multiple charts; Canvas is a tool for placing a canvas. 2. Auxiliary display layer: mainly includes Axis axes, Spines, Tick, Grid, Legend, Title, etc. This layer can be set not to display by `set_AXis_off ()` or `set_frame_on (False)`. 3. Image layer: images drawn through the plot, contour, scatter and other methods.

(4) Scipy is a scientific computing library based on Numpy, which is used in mathematics, science, engineering and other fields. Many high order abstractions and physical models need Scipy. Modules in SciPy cover topics such as optimization,

calculation of linear algebra, integral and interpolation, special or user defined functions, Fourier transforms, signal processing, image processing, solving various differential equations, and other computations that are frequently utilized in the fields of scientific research and engineering practice. It mainly includes the following common modules:

- File read/write: `scipy. io`
- Special or user defined functions: `scipy. special`
- Calculation of linear algebra: `Scipy. linalg`
- Solution of interpolation: `scipy. interpolate`
- Fit and optimization operations: `scipy. optimize`
- Statistics indicators: `scipy. stats`
- Calculation of integration: `Scipy. integrate`
- Fast Fourier transform: `scipy. fftpack`
- Signal/Image processing: `scipy. signal/ ndimage`

(5) Python is the programming language that forms the basis of Scikit-Learn, an open-source machine learning toolkit that is also known as SkLearn. For the purpose of developing fast algorithmic applications, it makes use of Python numerical computing libraries such as NumPy, SciPy, and Matplotlib. Additionally, it encompasses practically all commonly used machine learning methods. In engineering applications, the likelihood of using handwritten Python code to implement an algorithm from scratch is extremely low. This not only consumes time and effort, but also raises the possibility of not being able to write a clear framework or a model with a high degree of robust stability. In the majority of instances, it is necessary to analyze the gathered data, choose the appropriate algorithm according to the characteristics of the data, utilize the algorithm shorthand from the toolkit, tune the parameters of the algorithm, and obtain the required information in order to achieve a balance between the effectiveness and efficiency of the algorithm. Sklearn, on the other hand, is a toolkit that may assist us in effectively implementing algorithms.

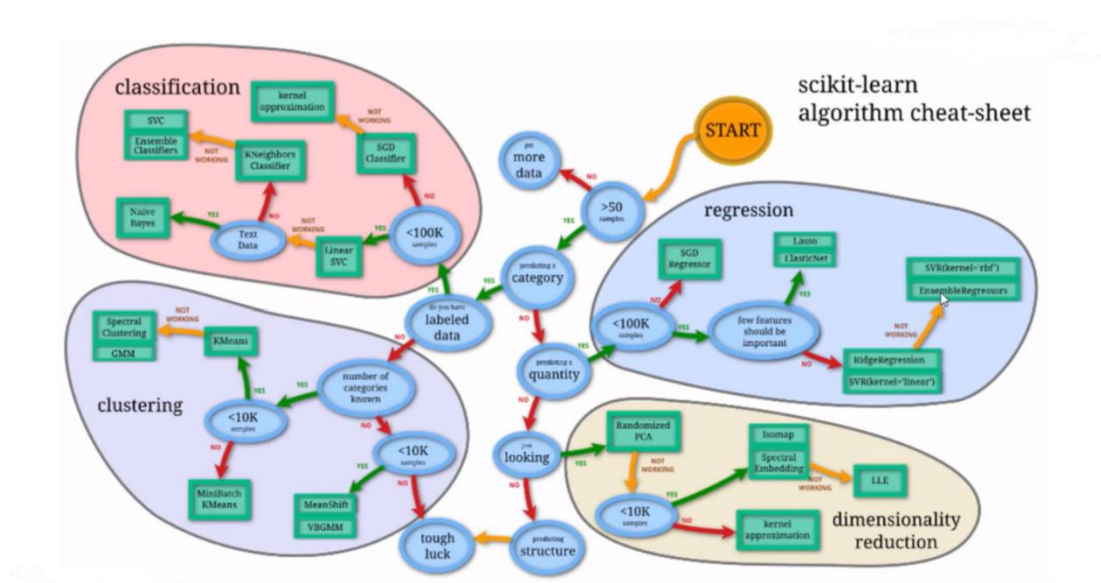


Figure 1. The cheat sheet of Sklearn library

The modules commonly used in Sklearn are classifying, fitting or regressing, clustering, reducing dimensionality, selecting the model's feature and preprocessing. You can refer to the Figure 1 above for how to select the corresponding module in practice. Classification: The support vector machine (SVM), K nearest neighbors (KNN) and decision tree are all examples of popular methods that may be used to determine the category of an item. Typical uses include junk mail and facial recognition software. Regression: To forecast the continuous value qualities that are connected with an item is the purpose of regression. Support vector regression (SVR), Lasso and Ridge regression are examples of common methods. Drug reaction prediction and stock price forecasting are two examples of common uses. Clustering: It is the result of the phenomenon when similar things are automatically grouped together. The k-means method, expectation maximum, and Canopy are instances of common algorithms. The segmentation of customers and the outcomes of experiments conducted in groups are two common use scenarios. Dimension reduction: Reduce the number of random variables that need to be addressed by using the dimensionality reduction technique. Principal component analysis (PCA), factorization of non-negative matrices, and feature selection are common methods to realize this function. And visualization is also used to enhance its productivity. Parameter choose: Model selection involves comparing, validating, and selecting parameters and models;

modules like grid search, cross-validation, and metrics are often used. The objective is to get greater precision by adjustment of the parameters automatically. Preprocessing: The first stage of processing involves the extraction and normalization of features. Preprocessing and feature extraction are two modules that are used often. One example of common use is the conversion of input data (like text, picture or voice) into data that may be used by ML algorithms.

(6) PyTorch is a machine learning package available for public use that is written in Python, based on Torch, and intended for use in applications such as natural language processing. The framework has been updated in Python to make it more versatile, enable dynamic graphs, and give a Python interface, but fundamentally, it is the same as the Torch framework. It is a Python-first deep learning framework that was developed by the Torch7 team. It not only offers tremendous GPU acceleration, but also enables dynamic neural networks.

The following is a list of qualities that PyTorch possesses: 1. Simple to use. PyTorch is meant to be as simple as possible, with the goal of minimizing packing and avoiding the need to reinvent the wheel. In contrast, TensorFlow adheres to three distinct levels of abstraction. These levels are as follows: tensor, variable, and nn.Module. They are intricately connected to one another and each other's representations, which include the tensors of high-dimensional arrays, variables in automated derivations, and layers/modules of neural networks. And it makes it possible to tune parameters to make improvements while simultaneously carrying out activities. 2. Take fewer computation resources. PyTorch's versatility does not come at the price of its performance, and in several evaluations, PyTorch has surpassed other frameworks, such as TensorFlow and Keras, when it comes to speed. 3. Friendly interface to use. PyTorch is an elegant object-oriented design framework currently available, and it is also quite easy to understand. The Torch, which is well-known for its adaptable and simple-to-use interface design, served as the inspiration for the design of the object-oriented interface that is used in PyTorch. Also, the Torch was the initial source of inspiration for Keras. 4. An active community. PyTorch offers users comprehensive documentation, careful guidance, and a forum that is managed by the creator with the

purpose of allowing users to interact and ask questions. PyTorch is strongly supported by the Facebook AI Institute. It mainly includes the following contents:

- Library for automated discrimination based on tape that is compatible with all of Torch's distinct tensors: `Torch. autograd`
- Involve neural network library and is closely connected with Autograd in order to optimize flexibility: `Torch. nn`
- The optimizer packages are used with `Torch. nn`, which includes Adam, SGD, RMSProp, and a variety of other common optimization methods: `Torch. optim`
- Concurrency across many processes in Python, memory sharing amongst torch Tensors processes: `Torch. multiprocessing`
- Data loader that also includes trainers and other useful tasks: `Torch. utils`
- Legacy code imported from Torch to provide compatibility with older versions: `Torch. legacy (.nn/ .optim)`:

3.2 Time series forecasting models

Following an analysis of the relevant previous research, three suitable time series forecasting algorithms, namely Prophet, RNN, and LSTM, have been suggested for use in predicting the behavior of renewable resources. In this section, the definitions, theories, and features of these algorithms will be discussed before being contrasted with one another. This will provide a strong basis for the methodologies that will be used in this study.

1. In a nutshell, Prophet is a time series prediction algorithm developed by Facebook, which can be utilized by the public. It was designed specifically to address the challenges posed by the business prediction tasks that we face at Facebook. The following are some of the usual aspects of these responsibilities: 1) Historical data on hourly, daily, or weekly observations for a minimum of a few months (ideally a year); 2) The presence of robust seasonal tendencies at various human-scale levels, including certain days of the week and certain seasons of the year; 3) Major holidays that are

known to come at unpredictable periods in advance, such as Labor Day. 4) The amount of missing data or significant outlier is under an acceptable range; 5) The data is related to time series and changes with historical trends; 6) The nonlinear growth trend included in the data has a natural limit or a saturation state inside it.

The algorithm model is composed of four parts: growth, seasonality, holidays and the rest. The Prophet algorithm does this by fitting the following variables and adding them up to obtain the forecasted time series values.

$$y(t) = g(t) + s(t) + h(t) + \varepsilon(t)$$

Where, $g(t)$ refers to the trend pattern, which is decided by the alteration trend of time series on the aperiodic level; $s(t)$ is the component of the equation that is periodic or seasonal and is often measured in weeks or years; $h(t)$ represents the holidays' term, which describes the influence that any prospective holidays that occur within a non-fixed period have on the value that is anticipated for the time series; $\varepsilon(t)$, the fluctuations that are not anticipated by the model are represented by the error term, also known as the residual term, and it follows the Gaussian distribution.

(1) The trending term serves two crucial purposes; the first one is determined by the logistic regression function (nonlinear growth), and the second is determined by the piecewise linear function (linear growth).

Trend terms based on logistic regression:

$$g(t) = \frac{C(t)}{1 + \exp\left(-\left(k + a(t)^T \delta\right) \cdot \left(t - \left(m + a(t)^T \gamma\right)\right)\right)},$$

$$\mathbf{a}(t) = (a_1(t), \dots, a_S(t))^T, \delta = (\delta_1, \dots, \delta_S)^T, \gamma = (\gamma_1, \dots, \gamma_S)^T$$

$C(t)$ stands for the carrying capacity, which is a function that varies with the passage of time and establishes a ceiling on the maximum value of the growth; K stands for the growth rate: In the real-world time series, the tendency of the curve will most likely not stay the same, and the curve will change at some particular time or in conjunction with some potential cycle; Change points are what the model calls the appropriate points when the increase rate K alters. And these points can be defined by the model; m denotes the offset. When the increase rate k is modified, each

change points' offset should also be altered correspondingly in order to link the final time point of each segment.

Trend term based on piecewise linear function:

$$g(t) = (k + \mathbf{a}(t)^T \boldsymbol{\delta}) \cdot t + (m + \mathbf{a}(t)^T \boldsymbol{\gamma})$$

Where, k refers to the increase rate, $\boldsymbol{\delta}$ refers to the variation in the increase rate, and m represents the offset. It is worth noting that the biggest difference between linear regression and logistic regression is the different settings of $\boldsymbol{\gamma}$.

(2) Fourier series may be used to approximate the periodic features of time series, which may include seasonal patterns of varying cycle types such as days, weeks, months, and years. Suppose that P represents the time series' period, $P = 365.25$ represents the years' period, and $P = 7$ represents the weeks' period. Its Fourier series is shown in the following form:

$$s(t) = \sum_{n=1}^N \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right)$$

N refers to the number of such intervals that you want to use in the model. A larger value of N can fit a more complex seasonal function, but it will also bring more overfitting problems. According to the empirical value, $N = 10$ for the series with annual cycle ($P = 365.25$); For a sequence of weeks ($P = 7$), $N = 3$. Therefore, the seasonality of the time series is $s(t) = X(t)\boldsymbol{\beta}$, which is controlled by the seasonality prior scale. That is to say, the seasonality impact is made more apparent whenever the value of the seasonality previous scale is increased. When the value is lower, the influence of the season is less noticeable.

(3) In the real life, The time series will be significantly altered by holidays and other significant events; nevertheless, the time points affected by these changes do not often exhibit periodicity, so the analysis of these points is extremely necessary. In Prophet, special holidays are collected for each country. In addition to holidays, users can also set necessary holidays according to their own circumstances, such as Nov 11. Since each holiday or a known major event has a different impact on the time series, for example, the Spring Festival, the National Day is a seven-day holiday, for Labor

Day and other holidays, the length of holiday is shorter. Therefore, the holiday model takes into account the influence of various holidays at various time points as independent models. Additionally, different window values can be set for various holidays, which indicates that the holiday will affect the time series both before and after a certain period of time. For i th holiday, D_i is the period before and after the holiday. In order to express the holiday effect, a corresponding indicator function is needed, and a parameter is needed to express the influence range of the holiday. Assuming L holidays, the holiday effect model is:

$$h(t) = Z(t)\kappa = \sum_{i=1}^L \kappa_i \cdot 1_{\{t \in D_i\}}$$

Where κ is normally distributed and is affected by the index v = holidays prior scale. The default value is 10. When the value is bigger, it shows that the model is more heavily influenced by vacations. The influence of the holidays on the model is less when the value being modeled is lower.

2. ANN is the foundation of RNN. It is essential to get acknowledge of the ANN principle. ANN is a type of self-adaptive approximation system that takes its cues from the structure and function of the neural network in the brain and employs the mathematical model of complex mathematics in order to approximate the workings of unknow things. This system was inspired by the structure and function of the brain's neural network. The MLP, which is the form of neural network that is the most straightforward and widespread, is made up of three layers: an input layer, an output layer, and a hidden layer. Each layer contains several neurons, all of which are linked in a hierarchical fashion to the neurons that reside in the layers under and above it. In addition, as for its activation functions, the ‘Sigmoid’, ‘Tanh’, and ‘Relu’ functions, and their many variants are the ones that are most often utilized. Forward propagation, error reverse transmission, and weight updating are the three components that make up the mathematical idea that underpins MLP.

(1) Forward propagation: Supposed that the input variable is $\mathbf{x} = (x_1, x_2, \dots, x_m)^T$, the weight matrix passed from the first layer (input) to the second layer (hidden) is $\mathbf{w} = [\mathbf{w}_{ij}] \in R^{m \times n}$, and the weight matrix passed from the second layer (hidden) to the last

layer (output) is $v = (v_1, v_2, \dots, v_j)$, and y is output variable. The following is therefore the output produced by the j^{th} neuron in MLP:

$$h_j = f\left(\sum_{i=1}^m w_{ij} x_i - \theta_j\right) = f(u_1), \quad j = 1, 2, \dots, n,$$

The way to determine the outcome of the output layer is as follows:

$$O = f\left(\sum_{j=1}^n v_j h_j - \theta\right) = f(u_2).$$

(2) Error backward propagation: The objective of the learning in the process of error backward propagation is to get the model output O as near as it can come to the target value y by adjusting the weights of the error correction. The least root mean square error (RMSE) criteria was used as the foundation for the learning rule. The following is an illustration of the loss function E :

$$E(W) = \frac{1}{2} \sum_{i=1}^N (y_i - O_i)^2$$

where W represents all of the unknown weight coefficients, such as the weight coefficients for the hidden layer and the output layer; N represents the total number of training examples. During the process of backward propagation, the error collected is used as the backward input of the output layer. Additionally, the partial derivative of the weight value of the output and hidden layer is computed as follows, respectively:

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}} &= \frac{\partial E}{\partial O} \frac{\partial O}{\partial h_j} \frac{\partial h_j}{\partial w_{ij}} = -(y_i - O_i) f(u_2)' v_j f(u_1)' x_i, \\ \frac{\partial E}{\partial v_j} &= \frac{\partial E}{\partial O} \frac{\partial O}{\partial v_j} = -(y_i - O_i) f(u_2)' h_j \end{aligned}$$

(3) Weight update: Adding a weighted gradient that has a specified learning rate dependent on the initial weight is what's meant by "weight update," and this process is referred to as "weight update." The following is the formula for the calculation:

$$W = W - \eta \frac{\partial E}{\partial W}$$

The learning rate may be determined if the η value is greater than 0.

3. A recurrent neural network, or RNN, is a kind of neural network that accepts sequence data as its input, realizes recursion in the progression direction of the sequence, and links all of its nodes (cyclic units) together via a chain [48]. This has to

do with timing tasks that involve using the ideas of the past to predict the future, which is very close. It also indicates that the structure of RNN is very appropriate for the modeling of time series data. Circulation neural networks are superior to ordinary multilayer neural networks in terms of increasing the contact between the hidden layer nodes. The concept of a "cycle" is represented in the fact that the topology of each node in a cyclic neural network is quite similar. In contrast to convolutional networks, which implement weight sharing across networks by using the structure of the receptive field, cyclic neural networks implement weight sharing by adopting the same chain structure of each node.

The expression for the general RNN, which corresponds to the simple RNN is as follows:

$$\begin{aligned} h_t &= H_\theta(h_{t-1}, y_{t-1}) \\ y_t &= O_\theta(h_t, y_{t-1}) + v_t, v_t \sim N(0, \sigma_v^2) \end{aligned}$$

where H_θ and O_θ are any expressions of nonlinear functions. The practical use of time series often involves observations that vary slowly over the course of time, which means that two nearby data are roughly equivalent to one another. In this scenario, the direct use of observed values as target values is troublesome; thus, the only way to practically do this is by adding a residual connection layer.

RNN is able to handle arbitrarily lengthy sequence data in theory; but, in real applications, the conventional RNN model is unable to retain historical knowledge from a great distance in the past and does not have the capacity to build remote information communications. The reason for this is as the gradient disappears, it becomes lesser and lesser, and as for the previous neurons, the learning rate develops too sluggish to make influence the final result. This is the issue with gradient fading, which is also the reason why the learning rate becomes too slow. One of the frequent issues that arise in deep learning is known as gradient extinction. Since the effect of parameter initialization, regularization, and activation function selection is limited, a short and short-duration memory network is proposed as a means of overcoming this challenge. Parameter initialization, regularization, and activation function selection can all contribute to the resolution of the gradient extinction problem.

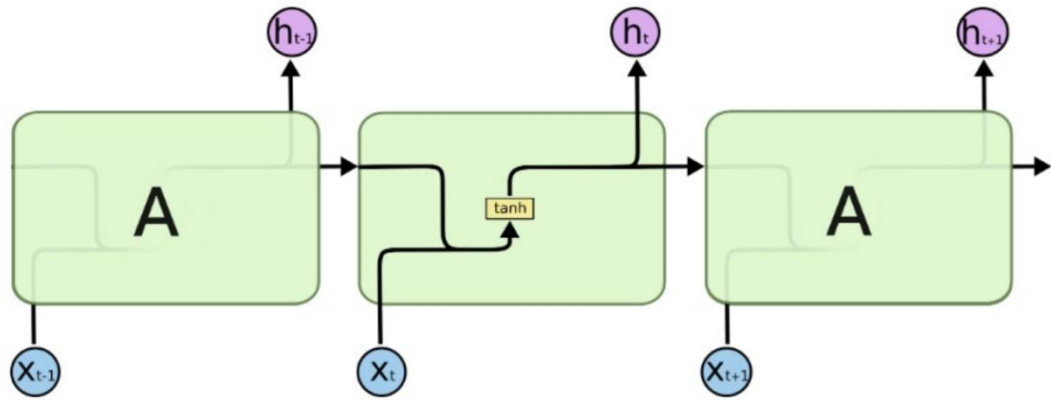


Figure 2. The principle of RNN

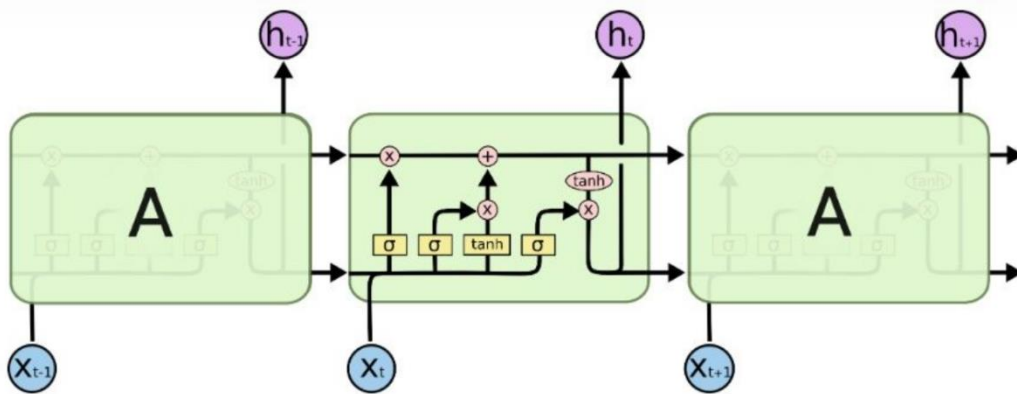


Figure 3. The principle of LSTM

4. The LSTM is a specialized variation of the RNN that has the ability to learn about long-term dependencies. By introducing the self-cycle, the path with the gradient flowing continuously for a long time can be generated to figure out the problem of the vanishing gradient of RNN. It adds input gate, forget gate, output gate and cell state based on RNN. The schematic diagram of RNN and LSTM is shown in the Figure 2 and Figure 3 above. The goal of the RNN's gate is to make the mapping relationship $H_t = H_0(H_{t-1}, Y_{t-1})$ learnable. Additionally, the network structure should be designed in such a way that the gradient propagation over time does not vanish or expand. On the basis of the gating principle, gated RNN provides models the ability to determine when it is appropriate to acquire knowledge and when it is appropriate to forget information. For instance, in LSTM, the topology of the network is purposefully built to preserve short-term memory for as long as possible by preventing the gradient from fading. This helps the model retain information for as long as feasible. The most important contribution made by LSTM was the creation of the memory unit: (1) The information will be written into the memory unit if the input gate is open; (2) The

information will be stored in the memory unit if the forget gate is closed; (3) The information will be read from the memory unit if the read gate is open.

Compared with advanced machine learning algorithms, the Prophet, as a time series of tools, an advantage is don't need to be able to get trend, season and holiday factors from characteristics engineering. But it is also one of its shortcomings, it cannot utilize more information, such as using commodity information, store information, sales information, etc. in the prediction of sales of goods. ANN is established by the classical statistical methods, whose study substance is known as the asymptotic theory when the sample is infinite. However, the sample data used in practical situations is often restricted. It is thus impossible to achieve the optimal application impact when there is a restriction on the sample data. RNN is able to cope with judgment and forecasting of input data, sequence difficulties connected to neighboring data, and short-period memory issues, but it is not sensitive to data from a very long time ago. On the other hand, there are issues with gradient expansion as well as gradient disappearance. LSTM is an improved structure of RNN, mainly to figure out the issues of vanishing gradient in the procedure of long-term sequence training so that recurrent neural network has stronger and better memory performance. In a nutshell, LSTM has the potential to perform far better than regular RNN when dealing with longer sequences. At the same time, improved versions of the RNN version, such as LSTM and GRU, will also be more computationally expensive.

3.3 Data collection and preprocessing

The Prediction of Worldwide Energy Resources project, which is financed by NASA's Earth Science's Applied Sciences Program, is where the data for this research came from [49].

This study is going to choose Dabancheng as the aim since it is abundant in renewable resources. This city is situated in Urumqi, Xinjiang, China. Its coordinates are 43.36 degrees north latitude and 88.31 degrees east longitude. The website collected hourly data on renewable sources of energy about the weather beginning on

1/1/2019 and ending on 31/12/2020. The database includes 17520 entries and 28 characteristics. Following the first filter, some variables that were not pertinent to this investigation were removed. Following this, there will be 17520 records for each of the 25 variables, which will result in 438,000 data points as shown in the Figure 4. The most important characteristics from a variety of perspectives are included in Table 1, and they include temperatures, humidity, precipitation, wind, pressure, solar fluxes, and associated and temporal variables.

Table 1. The major variables of the POWER project

	Parameter	Unit	Explanation		Parameter	Unit	Explanation
Temperatures	T2M	C	Temperature at 2 Meters	Solar fluxes and related	ALLSKY_SFC_SW_DWN	Wh/m ²	All Sky Surface Shortwave Downward Irradiance
	T2MDEW	C	Dew/Frost Point at 2 Meters		CLRSKY_SFC_SW_DWN	Wh/m ²	Clear Sky Surface Shortwave Downward Irradiance
	T2MWET	C	Wet Bulb Temperature at 2 Meters		ALLSKY_KT	/	All Sky Insolation Clearness Index
Humidity /Precipitation	QV2M	g/kg	Specific Humidity at 2 Meters		ALLSKY_SRF_ALB	/	All Sky Surface Albedo
	RH2M	%	Relative Humidity at 2 Meters		SZA	degrees	CERES SYN1deg Solar Zenith Angle
	PRECTOTCORR	mm/hour	Precipitation Corrected		ALLSKY_SFC_PAR_TOT	W/m ²	CERES SYN1deg All Sky Surface PAR Total
Wind/Pressure	PS	kPa	Surface Pressure		CLRSKY_SFC_PAR_TOT	W/m ²	Clear Sky Surface PAR Total
	WS10M	m/s	Wind Speed at 10 Meters		ALLSKY_SFC_UVA	W/m ²	All Sky Surface UVA Irradiance
	WD10M	degrees	Wind Direction at 10 Meters		ALLSKY_SFC_UVB	W/m ²	All Sky Surface UVB Irradiance
	WS50M	m/s	Wind Speed at 50 Meters		ALLSKY_SFC_UV_INDEX	/	All Sky Surface UV Index
	WD50M	degrees	Wind Direction at 50 Meters				
Time	YEAR	a	Year				
	MO	m	Month				
	DY	d	Day				
	HR	h	Hour				

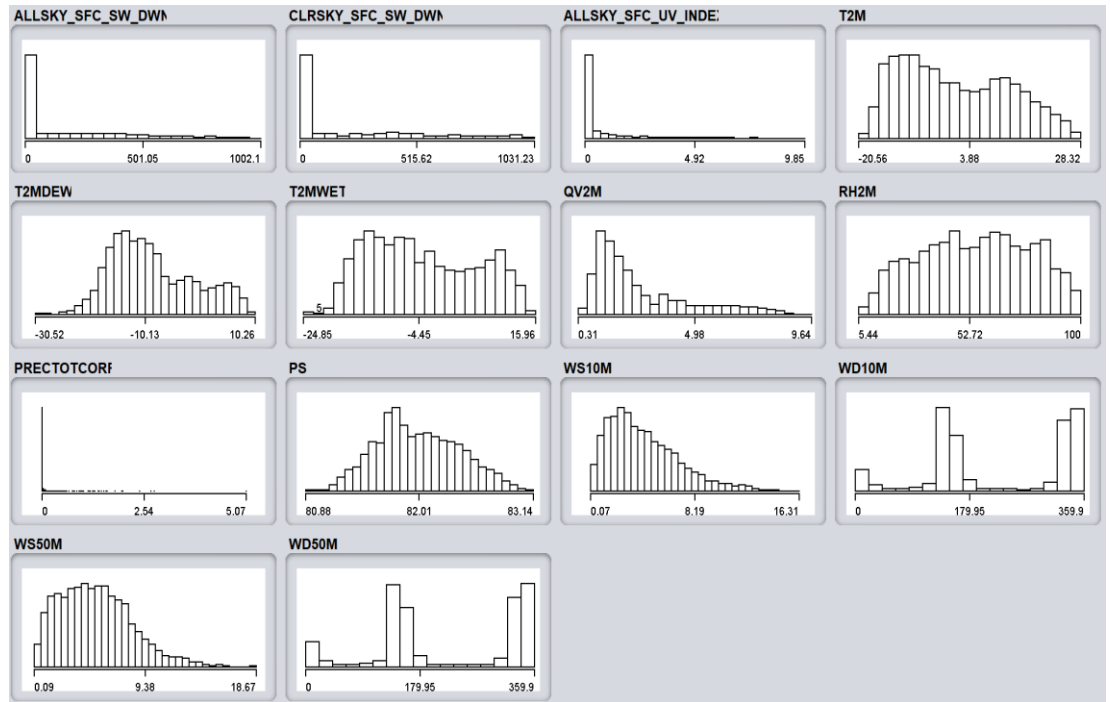


Figure 4. The overall visualization of the main parameters in this dataset

After that, the data will be cleaned up, features will be scaled, features will be

selected, and the dimensions will be reduced as part of the preparation step.

The act of examining data very thoroughly in order to eliminate duplicate information, fix mistakes, and give uniformity is referred to as "data cleaning." There is a possibility that the dataset contains some erroneous values and missing values as a result of input mistakes caused by the survey and coding. These aspects of the dataset need to be treated appropriately. Estimation and deletion will both play a role in the data processing that is described in this paper. Estimation is one of the most straightforward and productive strategies there is. It does this by inserting the sample mean, median, or mode of a variable in lieu of any erroneous or missing values. The fact that this approach does not take into account all of the information that is already included in the data is one of its drawbacks. This will result in a significant increase in the amount of inaccuracy. When using pairwise deletion, a unique code—typically one of the following: 999, etc.—is used to replace faulty values and missing values in the dataset, while at the same time, all variables and samples in the data set are reserved. On the other hand, only samples that have all of the necessary information will be used in the particular calculation scenario; hence, various analyses will have varying effective sample sizes owing to the many factors that are involved. Therefore, it is a very conservative strategy that extracts the maximum amount of information from the dataset.

Data that has been dimensionless transformed has been brought from several standards into a single specification. Interval scaling and normalizing are two examples of popular dimensionless approaches. The assumption behind the process of standardization is that the eigenvalues will behave according to the normal distribution. Once the standardization process is complete, the eigenvalues will be transformed into the standard normal distribution. The interval scaling approach reduces the value range of features to a predetermined range of features, such as $[0, 1]$. This is accomplished with the use of boundary value information. In this context, "feature selection" refers to the process of selecting fundamental characteristics based on earlier studies.

Because the eigenmatrix is too big, there is a possibility that it may create

problems with huge computations and prolonged periods of training. As a result, it is vital to cut down on the number of dimensions that the eigenmatrix has. The principal component analysis (PCA) is one of the most often used linear dimensionality reduction approaches. It has the ability to maintain the integrity of the original data points to the greatest extent feasible. The objective is to reduce the number of data dimensions by mapping high-dimensional data to low-dimensional space via a linear projection of some kind, with the expectation that the projection dimension will contain the most variation in the data.

3.4 Model training and evaluation

In this section, the overarching process of putting together three different models of algorithms will be discussed.

Prophet follows the SkLearn library modeling application program interface. The following is a list of the general stages involved in utilizing the Prophet algorithm: 1) The input quantity that Prophet receives is a two-dimensional array that has two columns named *ds* and *y*. The *ds* column is required to include either a date in the format of YYYY-MM-DD or a particular time in the format of YYYY-MM-DD HH:MM:SS. The *y* column has to be a numerical variable, representing the quantity that should be predicted. So the conversion of input and output data format is the first thing to do; 2) Then setting the Prophet basic parameters needs to be done. As for the setting of trend parameters, we need to have a general understanding of the trend of all the data before making a choice. As for the selection of periodic parameters, we need to apply the annual, seasonal, monthly, weekly and daily cycles based on the periodic properties of the data. Holiday parameters could be initially set according to the dependence of data on a particular date; 3) Finally, the Prophet model parameters are adjusted according to the predicted effect diagram. Specifically, it is to adjust the weight of each cyclical or holiday factor to get the most appropriate results. 4) Get the best prediction result and make the relevant drawing.

The following is a list of the fundamental training phases for the RNN algorithm:

1) Configure the required structural parameters of the RNN; 2) Prepare the training data, validation data and test data (where each group of data in each data set is represented by a series of integers); 3) A series of cyclic numbers with the current number serving as the input and the next number serving as the label (actual value); 4) Each cycle is processed by an RNN model, and the losses incurred by each group are computed and recorded; 5) An optimizer is used to improve the parameters. It is necessary to do steps 1 through 5 n times until the desired outcome is achieved.

The following is a condensed version of the fundamental processes involved in training an LSTM algorithm: 1) Firstly, one-hot coding, normalization and other data preprocessing operations are carried out for input and output data to obtain complete available input and output data; 2) Use libraries or custom functions to construct sequence data structures specifically required by LSTM and complete corresponding encapsulation; 3) Set the basic parameters of the LSTM model initially, including the window size, the number of LSTM layers, the input layer, the output layer, the data dimension of the hidden layer, etc. 4) Set and adjust the basic Settings of model compiler, including optimizer, learning rate, loss function, evaluation index, etc.; 5) Train the model to obtain preliminary prediction results and corresponding evaluation indexes; 6) Change the basic parameters of the model based on the deficiency of the forecasted consequences and evaluation indexes to obtain the optimal prediction effect.

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - f(x_i)|$$

$$MSE = \frac{SSE}{n} = \frac{1}{n} \sum_{i=1}^m w_i (y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{SSE}{n}} = \sqrt{\frac{1}{n} \sum_{i=1}^m w_i (y_i - \hat{y}_i)^2}$$

$$MAPE = \frac{100\%}{n} \sum_{i=1}^n \left| \frac{\hat{y}_i - y_i}{y_i} \right|$$

The quality of a model needs to be judged by evaluation indicators. As for machine learning, different types of tasks have different metrics, and accuracy, the commonly used metric in real life, often fails to accurately judge the model. For classification tasks, TP, FP, FN, TN, confusion matrix, precision, accuracy, recall rate, F1 score, ROC

and AUC curves are often utilized to estimate the quality of the model. However, for the predictive regression task, the evaluation parameters like correlation coefficient, mean absolute percentage error (MAPE), root mean square error (RMSE), mean absolute error (MAE) and mean square error (MSE) is often utilized to evaluate the proposed model. Calculating the mean of the absolute value of the distance between the expected and actual values is how MAE, also known as L1 norm loss, assesses the gap between the values that were predicted and those that were actually observed. MSE, which is also known as L2 norm loss, is a method for measuring the distance that involves computing the mean of the total squares of the difference that exists between the actual values and the values that were anticipated. The RMSE is a measure of the sample standard deviation of the difference between the projected value and the actual value. Compared with MAE, RMSE has a larger penalty for large error samples, but it is sensitive to outliers. MAPE is intuitive and often used in regression evaluation. It is computed in a similar way to MAE, but it has large results for points close to zero or very small values.

4. Results and discussion

4.1 Case A: Prophet forecasting model for renewable resources

In the case A, the time series prediction algorithm, Prophet, will be utilized for wind velocity and solar radiation prediction. This algorithm is a time series forecasting model for commercial forecasting proposed by Facebook, which has the characteristics of short training time and convenient parameter adjustment. Therefore, this paper applies this algorithm to the first attempt at renewable resources prediction.

The algorithm model requires an input of time series in standard format and data to be predicted. In this paper, we first extract the required two columns of data by deep copy and use the command to datetime in Pandas to format the date. As for the data predicted by the target, outliers need to be removed by directly assigning them to None. This is because missing values can be handled by interpolation in Prophet's design, but it is sensitive to outliers. All the data can then be divided into training and testing sets by an 8:2 ratio.

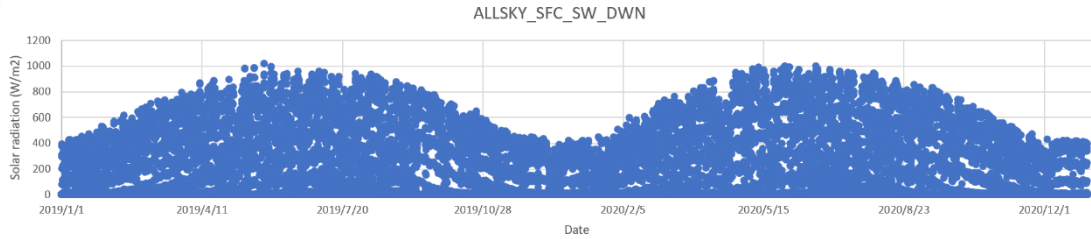


Figure 5. The overall visualization of solar radiation

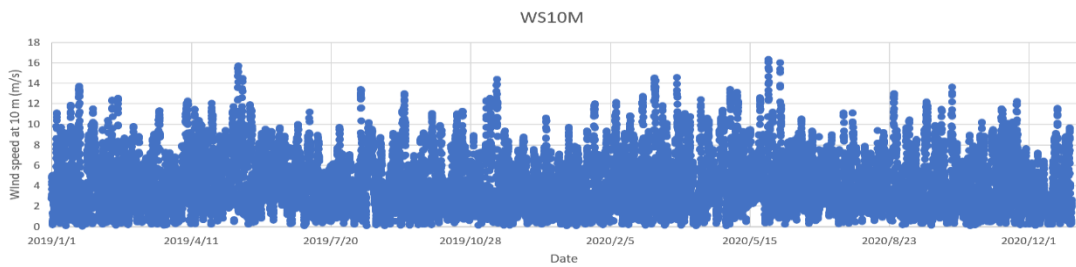


Figure 6. The overall visualization of wind velocity

After preparing the data, the basic parameters of the Prophet model need to be preliminarily set according to the trend of the data shown in the figure above. 1. Trend models including linear and logistic should be selected. For the analysis of wind speed and solar irradiance data, piecewise linear is used, that is, growth= 'linear'. 2. And then

is Changepoint processing. If we know that the trend of the time series will find a shift in some positions, we can manually set it. After observation of the experimental data, there is no such obvious data turning point. However, the Prophet algorithm is flexible and will summarize the changepoint itself based on the actual data. 3. Prophet's most important advantage is the identification of periodic characteristics, mainly including year, week, day, and seasonal periodicity. You can set other monthly and hourly periodicities based on data characteristics, or disable the annual and weekly periodicities. The renewable resource data presented in this paper are from two years and the obvious annual and seasonal cyclicity should be considered. Specifically, the amount of solar radiation varies significantly from day to night, so the periodicity of the day should also be considered. As for the periodicity of the week, this data is not obvious enough, so temporarily set this item to automatic. 4. Prophet algorithm can also identify holiday features. If there is a sudden increase or decrease in data in some periods, the holiday parameter can be set to adjust according to the impact degree and the size of the period. Because the weather data source is located in the basin and there is no significant change related to time, this item is set to None for identification by the model itself.

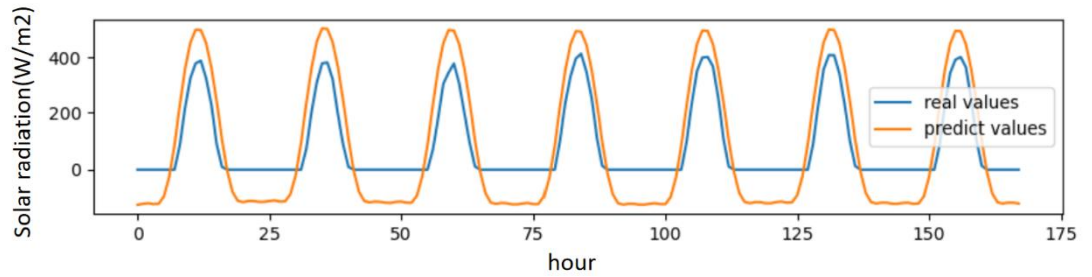


Figure 7. The comparison of predicted and real solar radiation using Prophet

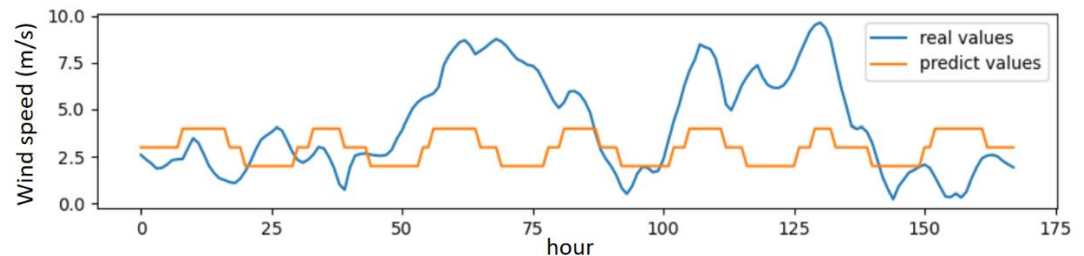


Figure 8. The comparison of predicted and real wind velocity using Prophet

After a preliminary setup, simple mapping observations can be made. Seasonality_prior_scale, Holidays_prior_scale, Changepoint_prior_scale, Seasonality

mode and other parameters are adjusted according to whether the model is overfitted and what components are overfitted. The final forecast for the week's renewable resources data is shown in the Figure 7 and Figure 8 above. It can be found that the model has a poor learning effect on large and complex weather data. For the more periodic solar radiation, the model can generally predict its changing trend, but can not accurately predict the range of peaks and valleys. For wind speed with strong randomness, this model is poor in predicting its numerical value and trend. Therefore, deep learning will be used to solve this problem in the following case study.

4.2 Case B: RNN forecasting model for renewable resources

In the case B, the deep learning method, RNN, is presented to solve the bad performance caused by the complexity of data to realize accurate prediction of renewable resources. Since RNN is able to recall the information from each and every second, it is able to address the issue of time series effectively. At each time, the output of the hidden layer is determined by taking into account both the input layer at that moment and the hidden layer from the moment before.

Deep learning algorithms are more complex, and there are often problems such as overfitting and gradient explosion that seriously affect the prediction performance. So more data processing needs to be done. Similar to Prophet's data processing, temporal data should be transformed as a format of timestamp. And because the data about time is a discrete feature, we need to carry out one-hot coding for it. N-bit status registers are used in one-hot encoding to encode N states. One state may exist because each state has its own independent register bit and only one of these register bits is legitimate. This approach addresses the issue that classifiers struggle to handle discrete data and serves as a feature extender to a certain degree. This paper first uses `LabelEncoder()`. `fit_transform()` to automatically generate one-hot encoding, and then use the `get_dummies` command in Pandas library to convert it into a dummy variable. If the multi-categorical variable has k categories, it can be transformed into k-1 dichotomous variables. This is particularly important in models of nonlinear relationships. On the

other hand, dense features also need format conversion, because a large amount of data and different dimensions will have a great influence on the training of the model. The `MinMaxScaler()` command is used to normalize the data, and the range of all feature data is reduced to $[0,1]$ in this article.

In addition, compared with the Prophet algorithm, which simply inputs time and target data, the input and output structure of RNN is more complex. Like traditional artificial neural networks, RNN also has an input layer, hidden layers, and an output layer. However, RNN propagates a feature sequence of different dimensions instead of a single quantity of features of different dimensions to the next layer network each time. To construct such a data format, the following code in Figure 9 is used. This custom function takes the time order as the standard and converts the data of a certain window size into a list. After the sequence data set is obtained, the whole data set can be processed. The sequence data was automatically split into a training set and test set in the ratio of 8:2, as indicated in the next program in Figure 10, and the data in the training set was shuffled to prevent overfitting.

```
def create_sequences(df, time, all_featuers, window_size, label, seq: list): # 创建序列值
    df.sort_values(time, ascending=True, inplace=True, ignore_index=True) # 值按时间戳排列
    current_arr = df[all_featuers] # 因素列
    risk_arr = df[label] # 所求列
    for i in range(len(risk_arr) - window_size):
        if risk_arr[i + window_size] >= 0: # 把窗口大小的特征数据和对应的一个标签值封起来
            train_seq_arr = current_arr[i: i + window_size]
            train_seq = torch.FloatTensor(train_seq_arr.values)
            train_label = torch.FloatTensor([risk_arr[i + window_size]]).view(-1)
            seq.append((train_seq, train_label))
```

Figure 9. The code for the construction of sequence data

```
def get_dataloader(seq_data, time='time', all_featuers=[], label='', window_size=7, bash_size=64, test_size=0.2, shuffer=True):
    print('获取训练数据')
    seq = []
    create_sequences(df=seq_data, time=time, all_featuers=all_featuers, window_size=window_size, label=label, seq=seq)
    random.seed(10)
    # 先对数据进行划分, 再进行shuffer
    train_len = int(len(seq) * (1-test_size))
    Dtr = seq[0: train_len]
    if shuffer is True:
        random.shuffle(Dtr) # 避免过拟合, 在此处进行shuffle
    Dte = seq[train_len:len(seq)]
    train = MyDataset(Dtr)
    test = MyDataset(Dte)
    Dtr = DataLoader(dataset=train, batch_size=bash_size, shuffle=False, num_workers=0)
    Dte = DataLoader(dataset=test, batch_size=bash_size, shuffle=False, num_workers=0)
    return Dtr, Dte
```

Figure 10. The code for the division of whole dataset

After converting the data format, the basic parameters of the RNN need to be set. This paper uses Pytorch library-related commands to build the RNN model. Input size indicates the dimension of the input features and is set to the number of all features. Hidden size indicates the feature dimensions of the hidden layer. Its size is determined by the complexity of the input feature and is set to 128 according to the empirical value. Num layers indicate the number of layers in the recurrent network. The default value is 1. According to the complication of the input time consequences, 2 is preliminarily set here. Nonlinearity is the selected nonlinear activation function ('Relu' and 'Tanh'), and the default 'Tanh' is adopted. Bias indicates whether bias is used, which is set to True in this paper. Batch first indicates the format of the input data. The default value is False, that is, sequence length first and batch second. Dropout indicates whether dropout is applied at the output layer, which is initially set to 0 in this paper. Bidirectional indicates whether to use bidirectional RNN. The default value is False in this paper.

In addition to the basic setup of the RNN model, compiler-related metrics also need to be selected. The first is the selection of the optimizer, which is used to control gradient clipping. Commonly used are SGD stochastic gradient descent optimizer, RMSprop, Adagrad, Adadelata, Adam/Adamax/Nadam. In this paper, 'Adam' was selected and the learning rate was set to 0.01. The second is the setting of the loss function, which is related to the number of indicators that the computational model ought to reduce during training. Commonly used ones are MSE, MAE, MAPE, and so on. This paper is set to simple and effective MSE; Finally, the evaluation function is selected to assess the performance of the currently trained model. When the model is compiled, the evaluation function should be entered as an argument to metrics. The evaluation function is like the loss function, with the exception that the training procedure does not employ the evaluation function's output. In this paper, MAE and MSE are used for comprehensive evaluation.

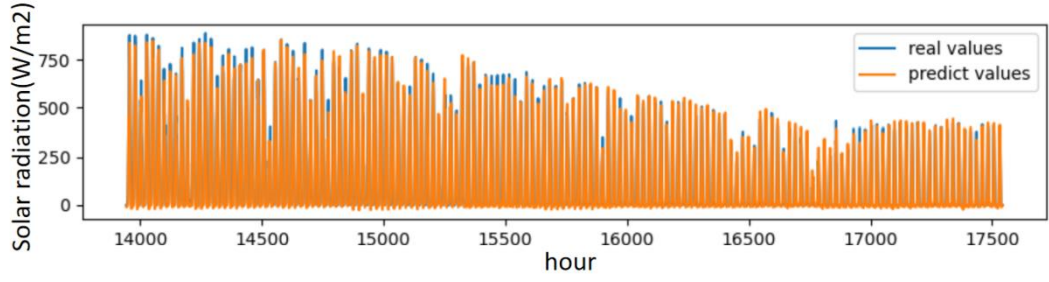


Figure 11. The comparison of the predicted and true solar radiation using RNN

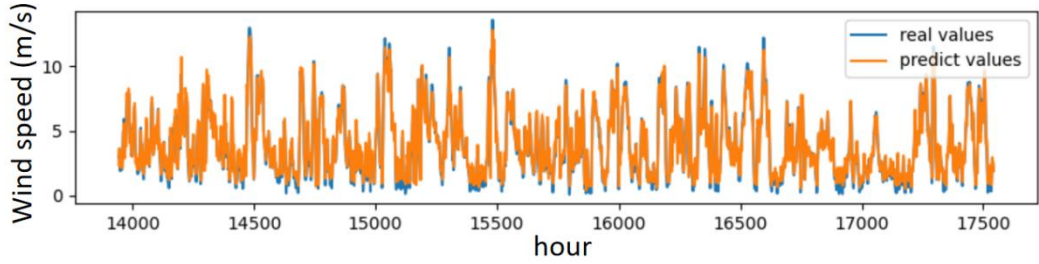


Figure 12. The comparison of the predicted and true wind speed using RNN

After preliminary parameter setting, the prediction results in the Figure 11 and 12 above can be obtained. It can be concluded that RNN has a good feature capture ability for complex time series. Then, more parameter Settings are tried to obtain the optimal evaluation index. Of these, the most important is the length of the encapsulation sequence, which is the window size. In this paper, the parameters are adjusted by big steps, such as 12, 24, 36, 48, 60, 72, etc., and it is found that the large window size costs a lot of computing resources, and the effect is similar or even worse. Therefore, the range of parameters is set between 1 and 24, and the parameters are adjusted in small steps one by one. It is found that when the window size is 8, the evaluation index of the test set is the best: the MAE of the predicted value of solar radiation and wind velocity are 0.019 and 0.028, and the MSE are 0.0008 and 0.0015, respectively. In addition, the epoch is also adjusted, and it is found that setting the value to 5 can achieve a suitable fit and occupy fewer computing resources. Hidden size and Batch size are also set to 128 and 256 respectively to achieve the optimal prediction effect. Although the RNN's prediction performance is good, it is not good enough to predict the long-term details. So in the next case study, LSTM is attempted to further improve the effect.

4.3 Case C: LSTM forecasting model for renewable resources

In the case C, the improved version of RNN, LSTM, is presented to enhance the performance of long-term forecasting by using gate control logic. Because of the vanishing gradient, RNN can only have short-term memory and cannot figure out the long-period dependence issue. Compared with RNN, LSTM adds a forgetting gate. To lessen the likelihood of gradient explosion and disappearance, the sigmoid function is added to the input and output gates, along with the Tanh function.

The data processing of LSTM is quite similar to that of RNN. Because they only transmit data differently between neurons in each layer, but the input and output formats of the data are the same. Firstly, the format of time data needs to be converted into time stamps. `LabelEncoder()`, `fit_transform()` and `get_Dummies ()` are used to generate one-hot encoding and dummy variable for discrete features. Then, the dense features are normalized using the `MinMaxScaler ()` command to reduce the impact of different dimensions of features. In addition, the input of LSTM also requires code to construct sequence data groups under different dimensions as unit input. The data set is split into a training set and a test set in accordance with an 8:2 ratio.

After data preprocessing, LSTM parameters need to be preliminarily set. In this paper, the relevant libraries in the Pytorch framework are applied to construct the LSTM model. Similar to the RNN parameters, the LSTM needs to set the key parameters Input size, hidden size, and the number of LSTM layers. In this paper, the input dimension of LSTM is set as the number of all features, the neurons' number of the hidden layer is set as 128, and the network layers' number of LSTM is set as 2. In addition, some other parameters need to be set, including choosing the default 'Tanh' as the nonlinear activation function, using bias, putting sequence length first and batch second, not applying dropout layer, and not using bidirectional LSTM. In addition, the compiler for the LSTM model also needs to be set specifically. The optimizer selects 'Adam' with the learning rate as 0.01, the loss function is set as MSE, and the evaluation function MAE and MSE are selected to comprehensively evaluate the prediction effect.

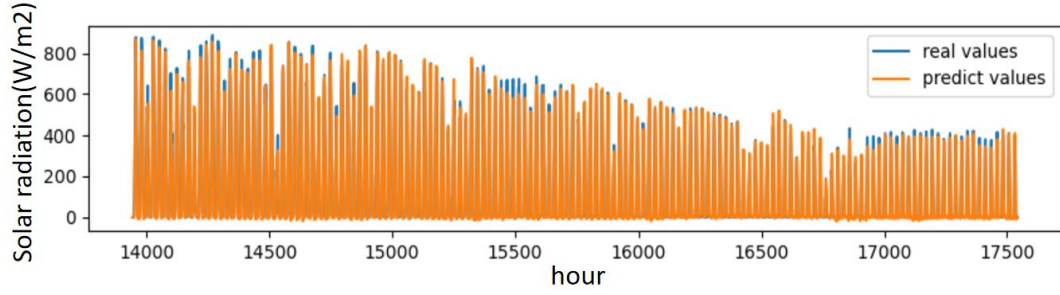


Figure 13. The comparison of the predicted and real solar radiation using LSTM

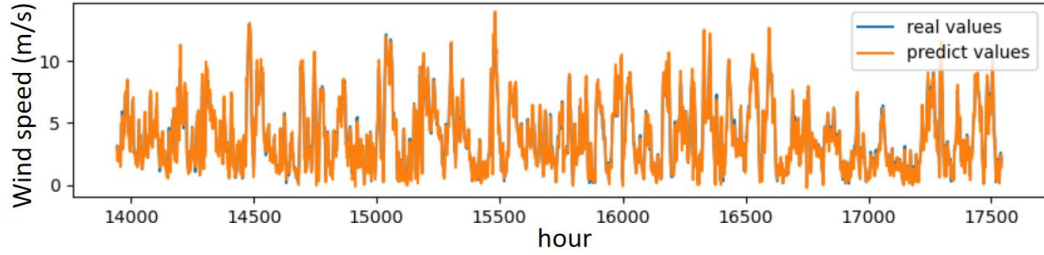


Figure 14. The comparison of the predicted and real wind speed using LSTM

After preliminary parameter adjustment, the prediction results in the Figure 13 and Figure 14 above can be obtained. It can be seen that LSTM can not only capture the trend of time series well but also accurately predict the long-period and short-period time sequence data. More parameter Settings were also attempted to obtain the optimal LSTM model. Similar to RNN, the window size is an important parameter in LSTM. Because the training of LSTM consumes more computing resources than RNN, this paper directly uses the step size of 2 to try the number in the range of 1-48. It is found that when the window size is 8, the evaluation index of the test set is the best: the MAE of the predicted value of solar radiation and wind velocity are 0.015 and 0.020, and the MSE are 0.0007 and 0.0008, respectively. In addition, hidden size and batch size were set to 128 and 256 respectively to achieve the optimal prediction effect. The epoch was also adjusted, and it was found that a value of 5 could be set to achieve a suitable fit with fewer computational resources.

Although LSTM has a strong ability to predict data, its training process will also take a lot of time. This may be because the number and dimension of input features are too large, and there are some useless or low-value feature data. In this case, PCA can be applied to decrease the dimension of input features. The correlation between many variables is investigated using this multivariate statistical approach. The goal is

to derive a few principal components from the original variables that retain as much of the original variables' information as possible and are uncorrelated with one another. This study focuses on how to reveal the internal structure of multiple variables through a few principal components. The standard mathematical procedure is to create a new comprehensive index by creating a linear combination of the original P indexes.

```
def pca(dataMat, topNfeat):
    meanVals = np.mean(dataMat, axis=0)
    meanRemoved = dataMat - meanVals # 标准化 (去均值)
    covMat = np.cov(meanRemoved, rowvar=False)
    eigVals, eigVets = np.linalg.eig(np.mat(covMat)) # 计算矩阵的特征值和特征向量
    eigValInd = np.argsort(eigVals) # 将特征值从小到大排序, 返回的是特征值对应的数组里的下标
    eigValInd = eigValInd[:-(topNfeat + 1):-1] # 保留最大的前K个特征值
    redEigVets = eigVets[:, eigValInd] # 对应的特征向量
    lowDDatMat = meanRemoved * redEigVets # 将数据转换到低维新空间
    # reconMat = (lowDDatMat * redEigVets.T) + meanVals # 还原原始数据
    return lowDDatMat
```

Figure 15. The program of principle component analysis

In this article, PCA is implemented by defining the above functions in the Figure 15: after input data, data standardization, calculation of eigenvalues and eigenvectors, sorting, conversion to low-dimensional space, reduction and other operations. Change the size of topNfeat, you can change the number of remaining feature variables. Since the dimension of the original data is 25, the size of topNfeat is adjusted between 0 and 25. After parameter adjustment, when the remaining dimension is 10, the MAE and MSE of the predicted value of renewable resources are 0.02 and 0.05, 0.001 and 0.006, respectively, and the calculation speed is faster.

4.4 Discussion

The previous three case studies covered different approaches, different preprocessing approaches, different prediction targets, and so on. By comparing the differences and connections between them in this section, the following facts can be found as shown in the following table. 1. About different time series prediction methods: Prophet is suitable for relatively uncomplicated and short-term time series data prediction, and has no ability to iteratively learn previous features; RNN is suitable for short-term or medium-term time series analysis and has the ability to learn

past time features, but it cannot master the details of long-term features. LSTM is suitable for short -, medium - and long-term prediction. It can not only learn the changing trend of long - and short-term time series data but also capture the detailed characteristics. 2. About different renewable resources: solar radiation has strong periodicity and depends on fewer input features, which is relatively simple compared with the temporal trend of wind velocity. However, the daily specific peak and valley values are more uncertain and often cannot be accurately identified according to the previous time. Wind speed has strong randomness and uncertainty, and its change trend is more difficult to identify. However, the maximum and minimum value of its change in a period of time can be accurately judged through time series data. 3. Results of various prediction means: Although the results of the traditional Prophet algorithm in this case study are not good, the trend of the solar radiation quantity with strong periodicity can be identified; Finally, the MAE of RNN for wind speed and solar radiation is 0.019 and 0.028, and the MSE is 0.0008 and 0.0015, respectively. Finally, the MAE of LSTM for the predicted value of solar radiation and wind velocity are 0.015 and 0.020 respectively, and the MSE are 0.0007 and 0.0008 respectively.

Table 2. The comparison of three time series forecasting methods and their results

	Advantage	Disadvantage	Prediction Results	
Prophet	Easy to use, strong ability to forecast cyclical trends	The prediction effect of the complex and large amount of data is poor	Solar radiation	MAE: NA
				MSE: NA
			Wind speed	MAE: NA
				MSE: NA
RNN	It can predict the trend of short- and medium-term time series data well	Requires complex preprocessing, and cannot predict long-term details	Solar radiation	MAE: 0.019
				MSE: 0.0008
			Wind speed	MAE: 0.028
				MSE: 0.0015
LSTM	Have a good effect on short -, medium - and long-term data	Needs more tedious processing and the calculation time	Solar radiation	MAE: 0.015
				MSE: 0.0007
			Wind speed	MAE: 0.020
				MSE: 0.0008

5. Conclusions and recommendations

In this article, the highly related works of literature, theoretical models and execution steps of two renewable resources and three time series forecasting methods are expounded and compared in detail. It can be concluded that both solar radiation and wind speed are suitable for analysis using the time series forecasting method. The prophet is suitable for predicting time series data with strong periodicity and uncomplicated trends. RNNS are suitable for predicting broad trends in complex time-series data in the short or medium term. LSTM is suitable for forecasting short -, medium - and long-term time series data. It is important that the deep learning method LSTM has the best prediction effect on renewable resources. The MAE and MSE of solar radiation prediction are 0.015 and 0.0007 respectively, and the MAE and MSE of Wind speed prediction are 0.020 and 0.0008 respectively. And the utilization of principal component analysis can accelerate the training of the LSTM model. In the next work, more data feature extraction, deep learning algorithms and automatic parameter optimization can be studied.

6. Reflection

This paper is still in the early stage of research, and the data preprocessing is not rich enough, which may affect the feature learning ability of the model. In addition, the relevant parameter adjustment in this paper is tried manually, which may not achieve the optimal prediction effect. Automatic optimization algorithms such as Grid Search can be applied to obtain optimal parameter Settings; In addition, the selection of traditional time series prediction algorithms needs to be enriched, and statistical methods such as ARIMA can be further tried. In addition, other deep learning algorithms such as GRU can be further used, because it has been shown in other kinds of literature that GRU and LSTM have similar prediction effects, but the training of the GRU algorithm consumes less computation. There are many other time series data for renewable resources that can be analyzed, and it would be interesting to see more

types of renewable resource projections.

Reference list

- [1] Y. M. Wei, K. Chen, J. N. Kang, W. Chen, X.Y. Wang, and X. Zhang, "Policy and management of carbon peaking and carbon neutrality: A literature review," *Engineering*, 2022, doi: 10.1016/j.eng.2021.12.018.
- [2] X. Yuan, C. W. Su, M. Umar, X. Shao, and O. R. Lobont, "The race to zero emissions: Can renewable energy be the path to carbon neutrality?," *Journal of Environment Management*, vol. 308, p. 114648, Apr 15 2022, doi: 10.1016/j.jenvman.2022.114648.
- [3] J. P. Lai, Y. M. Chang, C. H. Chen, and P. F. Pai, "A Survey of Machine Learning Models in Renewable Energy Predictions," *Applied Sciences*, vol. 10, no. 17, 2020, doi: 10.3390/app10175975.
- [4] H. M. Yang, Z. S. Pan, and W. Bai, "Review of time series forecasting method," *Computer science*, (01), 21-28, 2019
- [5] C. Sweeney, R. J. Bessa, J. Browell, and P. Pinson, "The future of forecasting for renewable energy," *WIREs Energy and Environment*, vol. 9, no. 2, 2019, doi: 10.1002/wene.365.
- [6] H. D. Nguyen, K. P. Tran, S. Thomassey, et al. "Forecasting and anomaly detection approaches using LSTM autoencoder techniques with the applications in supply chain management," *International Journal of Information Management*, vol. 57, 102282, 2021
- [7] P. Kumari and D. Toshniwal, "Deep learning models for solar irradiance forecasting: A comprehensive review," *Journal of Cleaner Production*, vol. 318, 2021, doi: 10.1016/j.jclepro.2021.128566.
- [8] S. S. Soman, H. Zareipour, O. Malik and P. Mandal, "A review of wind power and wind speed forecasting methods with different time horizons," *North American Power Symposium*, 2010, pp. 1-8, doi: 10.1109/NAPS.2010.5619586.
- [9] U. K. Das *et al.*, "Forecasting of photovoltaic power generation and model optimization: A review," *Renewable and Sustainable Energy Reviews*, vol. 81, pp. 912-928, 2018, doi: 10.1016/j.rser.2017.08.017.
- [10] L. Olatomiwa, S. Mekhilef, S. Shamshirband, K. Mohammadi, D. Petković, and C. Sudheer, "A support vector machine–firefly algorithm-based model for global solar radiation prediction," *Solar Energy*, vol. 115, pp. 632-644, 2015, doi: 10.1016/j.solener.2015.03.015.
- [11] N. D. Kaushika, R. K. Tomar, and S. C. Kaushik, "Artificial neural network model based on interrelationship of direct, diffuse and global solar radiations," *Solar Energy*, vol. 103, pp. 327-342, 2014, doi: 10.1016/j.solener.2014.02.015.
- [12] H.Y. Cheng, C.C. Yu, and S.J. Lin, "Bi-model short-term solar irradiance prediction using support vector regressors," *Energy*, vol. 70, pp. 121-127,

- 2014, doi: 10.1016/j.energy.2014.03.096.
- [13] Z. Ramedani, M. Omid, A. Keyhani, S. Shamsirband, and B. Khoshnevisan, "Potential of radial basis function based support vector regression for global solar radiation prediction," *Renewable and Sustainable Energy Reviews*, vol. 39, pp. 1005-1011, 2014, doi: 10.1016/j.rser.2014.07.108.
 - [14] T. Laopaiboon, W. Ongsakul, P. Panyainkaew and N. Sasidharan, "Hour-Ahead Solar Forecasting Program Using Back Propagation Artificial Neural Network," *International Conference and Utility Exhibition on Green Energy for Sustainable Development (ICUE)*, 2018, pp. 1-7, doi: 10.23919/ICUE-GESD.2018.8635756.
 - [15] B. Sivaneasan, C. Y. Yu, and K. P. Goh, "Solar Forecasting using ANN with Fuzzy Logic Pre-processing," *Energy Procedia*, vol. 143, pp. 727-732, 2017.
 - [16] O. Bamisile, A. Oluwasanmi, S. Obiora, E. Osei-Mensah, G. Asoronye, and Q. Huang, "Application of deep learning for solar irradiance and solar photovoltaic multi-parameter forecast," *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects*, pp. 1-21, 2020, doi: 10.1080/15567036.2020.1801903.
 - [17] M. Jaihuni, *et al.*, "A novel recurrent neural network approach in forecasting short term solar irradiance," *ISA Transactions*, vol. 121, pp. 63-74, Feb 2022, doi: 10.1016/j.isatra.2021.03.043.
 - [18] J. Zhou, J. Shi, and G. Li, "Fine tuning support vector machines for short-term wind speed forecasting," *Energy Conversion and Management*, vol. 52, no. 4, pp. 1990-1998, 2011, doi: 10.1016/j.enconman.2010.11.007.
 - [19] X. Kong, X. Liu, R. Shi, and K. Y. Lee, "Wind speed prediction using reduced support vector machines with feature selection," *Neurocomputing*, vol. 169, pp. 449-456, 2015, doi: 10.1016/j.neucom.2014.09.090.
 - [20] Y. Zhang, G. Pan, B. Chen, J. Han, Y. Zhao, and C. Zhang, "Short-term wind speed prediction model based on GA-ANN improved by VMD," *Renewable Energy*, vol. 156, pp. 1373-1388, 2020, doi: 10.1016/j.renene.2019.12.047.
 - [21] I. Okumus and A. Dinler, "Current status of wind energy forecasting and a hybrid method for hourly predictions," *Energy Conversion and Management*, vol. 123, pp. 362-371, 2016, doi: 10.1016/j.enconman.2016.06.053.
 - [22] Q. Cao, B. T. Ewing, and M. A. Thompson, "Forecasting wind speed with recurrent neural networks," *European Journal of Operational Research*, vol. 221, no. 1, pp. 148-154, 2012, doi: 10.1016/j.ejor.2012.02.042.
 - [23] M. Elsaraiti and A. Merabet, "Application of Long-Short-Term-Memory Recurrent Neural Networks to Forecast Wind Speed," *Applied Sciences*, vol. 11, no. 5, 2021, doi: 10.3390/app11052387.
 - [24] J. Heinermann and O. Kramer, "Machine learning ensembles for wind power prediction," *Renewable Energy*, vol. 89, pp. 671-679, 2016, doi: 10.1016/j.renene.2015.11.073.
 - [25] Z. Li, S. M. Rahman, R. Vega, and B. Dong, "A Hierarchical Approach Using Machine Learning Methods in Solar Photovoltaic Energy Production Forecasting," *Energies*, vol. 9, no. 1, 2016, doi: 10.3390/en9010055.

- [26] S. Hu, Y. Xiang, D. Huo, S. Jawad, and J. Liu, "An improved deep belief network based hybrid forecasting method for wind power," *Energy*, vol. 224, p. 120185, 2021, doi: 10.1016/j.energy.2021.120185.
- [27] F. M. N. Ayoub, S. Pokharel and H. A. Gabbar, "ANN Model for Energy Demand and Supply Forecasting in a Hybrid Energy Supply System," *2018 IEEE International Conference on Smart Energy Grid Engineering (SEGE), Oshawa, ON, Canada*, pp. 25-30, 2018, doi: 10.1109/SEGE.2018.8499514.
- [28] K. Murugaperumal and P. Ajay D Vimal Raj, "Feasibility design and techno-economic analysis of hybrid renewable energy system for rural electrification," *Solar Energy*, vol. 188, pp. 1068-1083, 2019, doi: 10.1016/j.solener.2019.07.008.
- [29] W. Zhang, A. Maleki, and M. A. Rosen, "A heuristic-based approach for optimizing a small independent solar and wind hybrid power scheme incorporating load forecasting," *Journal of Clean Production*, vol. 241, p. 117920, 2019, doi: 10.1016/j.jclepro.2019.117920.
- [30] Z. Yu, F. Haghighat, B. C. M. Fung, and H. Yoshino, "A decision tree method for building energy demand modeling," *Energy and Buildings*, vol. 42, no. 10, pp. 1637-1646, 2010, doi: 10.1016/j.enbuild.2010.04.006.
- [31] M. Cai, M. Pipattanasomporn, and S. Rahman, "Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques," *Applied Energy*, vol. 236, pp. 1078-1088, 2019, doi: 10.1016/j.apenergy.2018.12.042.
- [32] T. Windler, J. Busse, and J. Rieck, "One month-ahead electricity price forecasting in the context of production planning," *Journal of Clean Production*, vol. 238, p. 117910, 2019, doi: 10.1016/j.jclepro.2019.117910.
- [33] M. G. Khajeh, A. Maleki, M. A. Rosen, and M. H. Ahmadi, "Electricity price forecasting using neural networks with an improved iterative training algorithm," *International Journal of Ambient Energy*, vol. 39, no. 2, pp. 147-158, 2017, doi: 10.1080/01430750.2016.1269674.
- [34] J. Wang, F. Liu, Y. Song, and J. Zhao, "A novel model: Dynamic choice artificial neural network (DCANN) for an electricity price forecasting system," *Applied Software Computing*, vol. 48, pp. 281-297, 2016, doi: 10.1016/j.asoc.2016.07.011.
- [35] D. H. An, X. Y. Liu, J. K. Liu, and S. J. Xu, "Modeling and forecasting of power load based on seasonal ARIMA model," *Journal of Tianjin University: Natural Science and Engineering Technology Edition*, vol. 2, pp. 184-187, 2004
- [36] V. S. Ediger, S. Akar, "ARIMA forecasting of primary energy demand by fuel in Turkey," *Energy Policy*, vol. 35(3), pp. 1701-1708, 2007
- [37] Z. X. Luo, "Use the ARIMA model to predict oil prices in 2007," *Natural gas technology*, vol. 1, pp. 46-48, 2007
- [38] S. S. Pappas, L. Ekonomou, P. Karampelas, et al. "Electricity demand load forecasting of the Hellenic power system using an ARMA model," *Electric Power Systems Research*, vol.80 (3) 1, pp. 256-264, 2010

- [39] B. C. Ervural, O. F. Beyca, S. Zaim, "Model Estimation of ARMA Using Genetic Algorithms: A Case Study of Forecasting Natural Gas Consumption," *Procedia - Social and Behavioral Sciences*, vol. 235 1, pp. 537-545, 2016
- [40] D. Li, " Prediction and application of field yield based on time series analysis method," Lanzhou science and technology, 2018
- [41] A. Jain, A. M. Kumar, "Hybrid neural network models for hydrologic time series forecasting," *Applied Software Computing*, vol. 7(2), pp. 585-592, 2007
- [42] G. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50(1), pp. 159-175, 2003
- [43] H. Goel, Melnyki, A. Banerjee, "R2N2: Residual recurrent neural networks for multivariate time series forecasting," *Arxiv*, 1709. 03159, 2017
- [44] G. Lai, W. Chang, Y. Yang, et al. "Modeling long- and short- term temporal patterns with deep neural networks," *Arxiv*, 1703. 07015, 2017
- [45] R. Tawn and J. Browell, "A review of very short-term wind and solar power forecasting," *Renewable and Sustainable Energy Reviews*, vol. 153, 2022, doi: 10.1016/j.rser.2021.111758.
- [46] R. A. Gupta, R. Kumar, and A. K. Bansal, "BBO-based small autonomous hybrid power system optimization incorporating wind speed and solar radiation forecasting," *Renewable and Sustainable Energy Reviews*, vol. 41, pp. 1366-1375, 2015, doi: 10.1016/j.rser.2014.09.017.
- [47] W. Zhang, A. Maleki, M. A. Rosen, and J. Liu, "Sizing a stand-alone solar-wind-hydrogen energy system using weather forecasting and a hybrid search optimization algorithm," *Energy Conversion and Management*, vol. 180, pp. 609-621, 2019, doi: 10.1016/j.enconman.2018.08.102.
- [48] I. Goodfellow, Y. Bengio, A. Courville, "Deep learning," Cambridge: MIT Press, vol. 1, pp. 367-415, 2016
- [49] National Aeronautics and Space Administration, "The POWER project", Online source: <https://power.larc.nasa.gov/#resources>

Appendices

A. Prophet

```
data = pd.read_csv("./data/dataset.csv")
labels = ['WS10M', 'ALLSKY_SFC_SW_DWN']
data['ds'] = data[['YEAR', 'MO', 'DY', 'HR']].apply(lambda x: f'{x[0]}-{x[1]}-{x[2]}{x[3]}:00:00", axis=1)
data['ds'] = pd.to_datetime(data['ds'])
for label in labels:
```

```

tmp_df = data[['ds', 'label']].copy(deep=True) # 深拷贝不影响源数据, 纯复制

tmp_df.columns = ['ds', 'y'] # 两列, 时间戳和真实值

tmp_df.sort_values(by = ['ds'], ascending = True, inplace = True, ignore_index =
True) # 复制完可能乱序的, 按时间戳排序

test_len = 3504
train_df = tmp_df[:-test_len]
pred_df = pd.DataFrame()
pred_df['ds'] = tmp_df['ds']

m = Prophet (growth = 'linear', changepoints = None, n_changepoints = 10,
changepoint_range = 0.5,
              yearly_seasonality = True, weekly_seasonality = True,
daily_seasonality = True,
              holidays = None, seasonality_mode = 'additive',
seasonality_prior_scale = 10.0,
              holidays_prior_scale = 10.0, changepoint_prior_scale = 0.05,
mcmc_samples = 0,
              interval_width = 0.8, uncertainty_samples = 1000, stan_backend =
None)

m.fit(train_df)
tmp_df['pred'] = m.predict(tmp_df) ['yhat'] .apply(lambda x: int(x))

tmp_df = tmp_df [-168:] # 看预测一周的效果

plt.figure (figsize = (10, 2))

# plt.axvline (x = len(tmp_df) - test_len, c = 'r', linestyle = '--') # 训练和测试
的分界线

plt.plot (range (len(tmp_df)), tmp_df ['y'], label = "real values")
plt.plot (range (len(tmp_df)), tmp_df ['pred'], label = "predict values")
plt.legend()
plt.savefig(f"./images/{label}_prophet.png")

```

```
plt.close()
```

B. RNN

```
def main (label = 'WS10M'):
```

```
    window_size = 8 #最佳 3 0.018 0.025
```

```
    hidden_size = 128
```

```
    num_layers = 2
```

```
    output_size = 1
```

```
    batch_size = 256
```

```
    device = 'cpu'
```

```
    epochs = 5
```

```
    lr = 0.01
```

```
    test_size = 0.2
```

```
    data = pd.read_csv("./data/dataset.csv")
```

```
    print(data.columns)
```

```
    data['ds'] = data [['YEAR', 'MO', 'DY', 'HR']].apply(lambda x: f"{x[0]}-{x[1]}-{x[2]} {x[3]}:00:00", axis = 1)
```

```
    all_features = []
```

```
    sparse_features = ['MO', 'DY', 'HR'] # 稀疏特征
```

```
    dense_features = ['YEAR', 'ALLSKY_SFC_SW_DWN',  
                      'CLRSKY_SFC_SW_DWN',  
                      'ALLSKY_KT', 'ALLSKY_SRF_ALB', 'SZA',  
                      'ALLSKY_SFC_PAR_TOT',  
                      'CLRSKY_SFC_PAR_TOT', 'ALLSKY_SFC_UVA',  
                      'ALLSKY_SFC_UVB',  
                      'ALLSKY_SFC_UV_INDEX', 'T2M', 'T2MDEW',
```

```

'T2MWET', 'QV2M', 'RH2M',
                                'PRECTOTCORR', 'PS', 'WS10M', 'WD10M', 'WS50M',
'WD50M'] # 稠密特征

# 离散特征进行 one-hot 编码

for feature in sparse_features:

    data[feature] = LabelEncoder().fit_transform(data[feature]) # 训练

LabelEncoder,使编码为 0,1,2..., 再转化

    tmp_onehot = pd.get_dummies(data[feature], prefix=feature)
    data[tmp_onehot.columns] = tmp_onehot
    all_features.extend(tmp_onehot.columns)

label_mms = MinMaxScaler()
data[label] = label_mms.fit_transform(data[label].values.reshape(-1, 1))
data[dence_features] = MinMaxScaler().fit_transform(data[dence_features])
all_features.extend(dence_features)

data['ds'] = pd.to_datetime(data['ds'])

Dtr, Dte = get_dataloader(data, time = 'ds', all_features = all_features, label =
label, window_size = window_size,

                                test_size = test_size)

input_size = len(all_features)

# 模型结构

model = RNN (input_size, hidden_size, num_layers, output_size, batch_size =
batch_size, device = device,

                                window_size = window_size).to(device) # 输入维度是特征的
数量，输出就这一个，

# 模型评价

```

```

model.compile("adam", "mse",
              metrics = ['mae', 'mse'],
              lr = lr)

# 模型训练

model.fit (train = Dtr, batch_size = batch_size, epochs = epochs, validation_data
= Dte)

del Dtr, Dte
gc.collect()

# 验证和绘图

test_len = int(len(data) * test_size)

all_data, _ = get_data_loader(data[int(len(data) * (1 - test_size))-100:], time='ds',
all_featuers = all_features, label = label, window_size = window_size,
                                test_size = 0, shuffer = False)

y_pred = model.predict(all_data)
y_pred = label_mms.inverse_transform(y_pred)
data[label] = label_mms.inverse_transform(data[label].values.reshape(-1, 1))

plt.figure(figsize = (10, 2))
# plt.axvline(x = len(data) - test_len, c = 'r', linestyle = '--')
plt.plot(range(len(data) - len(y_pred), len(data)), data[label][len(data) -
len(y_pred):len(data)], label = "real values")
plt.plot(range(len(data) - len(y_pred), len(data)), y_pred, label = "predict values")
plt.legend()
plt.savefig(f'./images/{label}_rnn_values_pred.png')
plt.close()

if __name__ == '__main__':

```

```

for label in ['ALLSKY_SFC_SW_DWN', 'WS10M']:
    print(f'label = {label} *****')
    main(label)

```

C. LSTM

We can change the RNN model to the following code to realize LSTM forecasting:

```

model = LSTM (input_size, hidden_size, num_layers, output_size, batch_size =
batch_size, device=device, window_size=window_size).to(device)

```