Weather Prediction

Based on ARIMA, RNN and Prophet

Xiangqing Wang

2024-05-31

Contents

1 Introduction

Weather prediction is an essential tool that impacts almost every aspect of modern life. It enhances safety, supports economic activities, aids in disaster management, and contributes to public health and welfare. As technology advances, the accuracy and reliability of weather forecasts continue to improve, providing even greater benefits to society.

In this study, I focus on the city of Shenyang in China as a sample to establish a model for analyzing model composition and predicting weather conditions. Specifically, the study examines the **temperature** and **wind speed** data from the Shenyang weather station over a period of 2184 days, from January 1, 2017, to December 30, 2022.

To achieve a good prediction of these daily sequences, I began with the **ARIMA** model and explored three different models in total. The initial **ARIMA** model proved limited in fitting the daily weather data and did not perform well in day-level forecasting, as it failed to efficiently utilize past information. Due to the poor fit of the **ARIMA** model for the temperature sequence, I employed a **Recurrent Neural Network (RNN)** to achieve a better fit. Although the **RNN** model provided a satisfactory fit to the data, its forecasts for the future still needed improvement. Similar to **ARIMA**, the **RNN** model struggled with utilizing more current information and relied heavily on past data for forecasting. Finally, I turned to the **Prophet** model to enhance the forecasts. The results indicate that the **Prophet** model indeed offers a satisfactory fit for the data and effectively explores the underlying seasonal patterns. This model serves as a robust tool for predicting sequences with clear periodicity, providing more accurate and reliable weather forecasts for Shenyang.

2 Data Introduction

2.1 Data Source

The data I use in this project is from National Center for Environmental Information (NCEI) in National Oceanic and Atmospheric Administration's (NOAA)[1]. Specifically, I choose the weather data by day of Shenyang weather station from 1st January 2017 to 30th December 2022, together 2184 days. Here, I focus on the **temperature** and **wind speed** for analysis.

For the wind speed sequence, there are few extreme values, which I firstly dropped to avoid the distrubance on analysis. Table 1 is the summary statistics for the data.

2.2 Visualization

Here I plot the two sequence as well as the *ACF* and *PACF* plots, which is shown in Figure 1. We could directly see from the figure that the there is a strong seasonal pattern in both these two sequence. Since there are per day sequence, the seasonal period should be no bigger

[1]https://www.noaa.gov/

Table 1: Summary Statistics

Variable	N	Mean	Std. Dev.	Min	Pctl. 25	Pctl. 75	Max
Temperature	2184	48.64	23.36	-9.00	29.05	69.62	90.70
Wind Speed	2184	6.22	42.61	1.00	3.10	5.30	999.90

than 365. At the same time, the *ACF* plots have very slowly decaying pattern, indicating that there is a need for us to take differencing for the sequences.

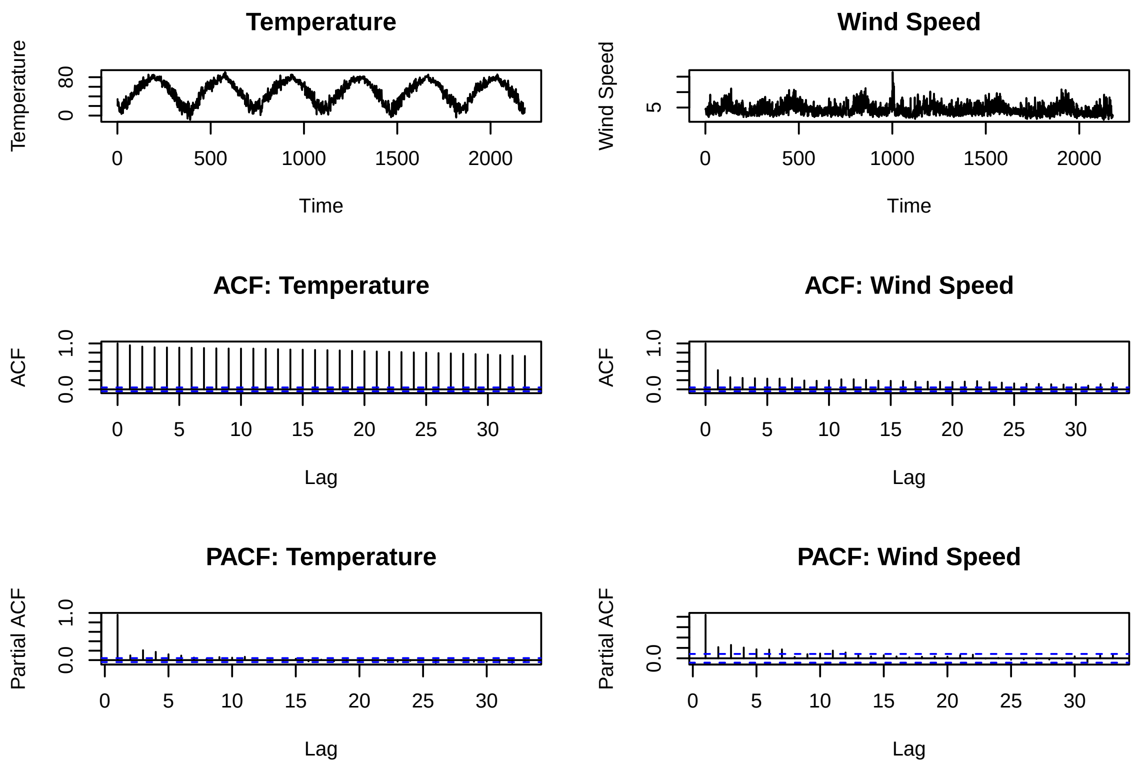

Figure 1: Data Visualization

3 Methodology

3.1 ARIMA

3.1.1 Model Introduction

The first model I implement on this data is $ARIMA$ model introduced in our course. Here the key parameter of the model is (p, d, q), where p is the *Autoregressive* coefficient, d is the step of differencing and q is the *Moving Avergae* coefficient. The general form of $ARIMA(p, d, q)$ model is in Equation (1).

$$(1-L)^d\phi(L)(y_t-\mu)=\theta(L)\epsilon_t \tag{1}$$

3.1.2 Correlation Plot

The reason that I use the **ARIMA** model separately instead of **VAR** model for these two sequence is that the correlation of these two sequence is generally 0, as shown in Figure 2. Thus, we indeed could fit these two sequence separately.

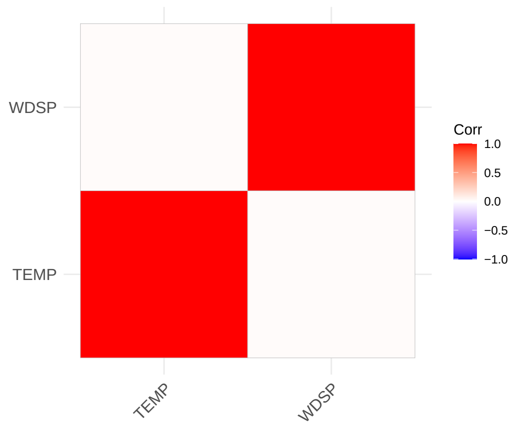

Figure 2: Correlation

3.1.3 Model Fitting, Residuals Checking and Forecast

From the former visualization of the data, we get to know these two sequence are generally nonstationary. Therefore, I firstly tried take the first order differencing for both two sequence. Then I use the `auto.arima` function from `forecast` package to select the best p and q.

Firstly, the sequences after differencing is generally stationary from *ACF*, *PACF* plots and `Augmented Dickey-Fuller test`, as shown in Figure 3.

```r
require(fUnitRoots)
adfTest(d.wind)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -52.2304
```

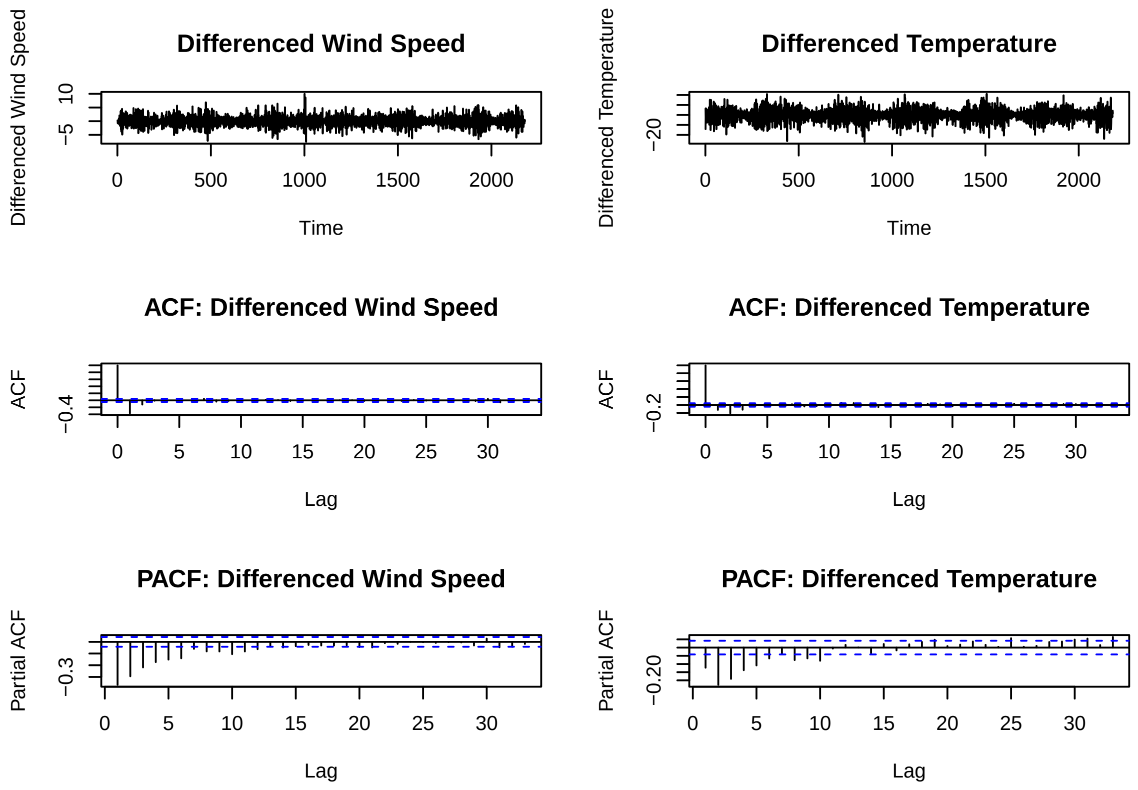

Figure 3: Visualization of Differenced Data

```
##   P VALUE:
##     0.01
##
## Description:
##  Sat Jun  1 16:25:48 2024 by user:
```

```r
adfTest(d.temp)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -44.0737
##   P VALUE:
##     0.01
##
## Description:
```

```
##  Sat Jun  1 16:25:48 2024 by user:
```

Then, I use the `auto.arima` function to fit the differenced sequence and get corresponding results.

Wind Speed Fitting

```
fit.wind <- auto.arima((d.wind))
fit.wind
```

```
## Series: (d.wind)
## ARIMA(0,0,2) with zero mean
##
## Coefficients:
##           ma1      ma2
##       -0.6883  -0.2053
## s.e.   0.0206   0.0209
##
## sigma^2 = 2.459:  log likelihood = -4071.79
## AIC=8149.58   AICc=8149.59   BIC=8166.64
```

From the output, the fitted **ARIMA** model for the **differenced wind speed** sequence is actually **MA(2)**, or equivalently **ARIMA(0,1,2)** for the **wind speed** sequence. Therefore, the expression of the fitted model is of Equation (2). Here, since we have already take the difference for the wind speed sequence, the mean is equal to 0.

$$\begin{gathered}(1-L)\times y_t = \epsilon_t - 0.69\times\epsilon_{t-1} - 0.21\times\epsilon_{t-2} \\ \epsilon_t \sim WN(0, 2.46)\end{gathered} \tag{2}$$

Then, I check whether the residuals are following the *White Noise* sequence. Specifically, I use the `tsdiag` function from `TSA` package to implement the analysis. This function shows the direct and the *ACF* plot of residuals as well as the `Ljung.Box` results plot. The checking plot for **wind speed** data is shown in Figure 4.

From the plot, we could generally conclude that it passed the residuals checking. So this *ARIMA(0,1,2)* model for the wind speed sequence. Then, I try to forecast 1 more year, namely out of sample forecast for 365 days. The forecast results are shown in Figure **??**. From the figure, we could only see subtle variation among the first serval steps of forecast, with later forecast generally become a horizontal line with the same width of confidence interval. The information here is that though the **ARIMA** model for wind speed could pass the residuals checking, the forecast outcome is really not satisfying and there is a need for us to develop a better model to do the forecast.

Temperature Fitting

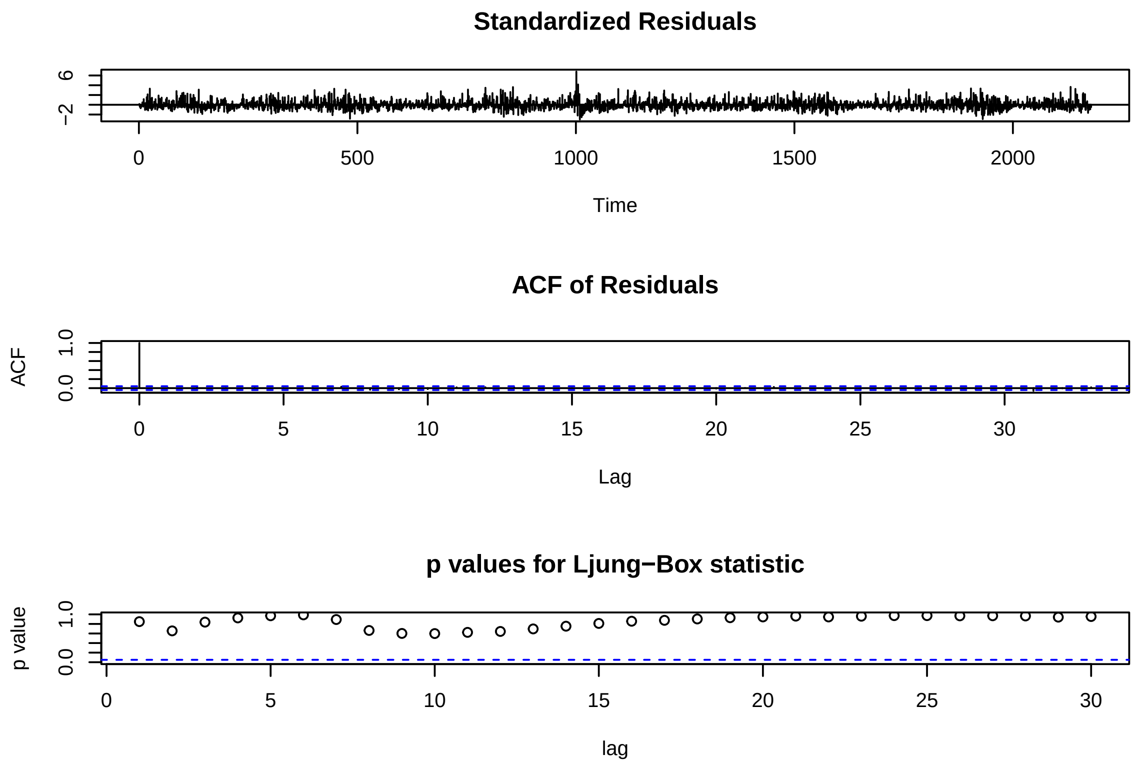

Figure 4: Residuals Checking for Wind Speed

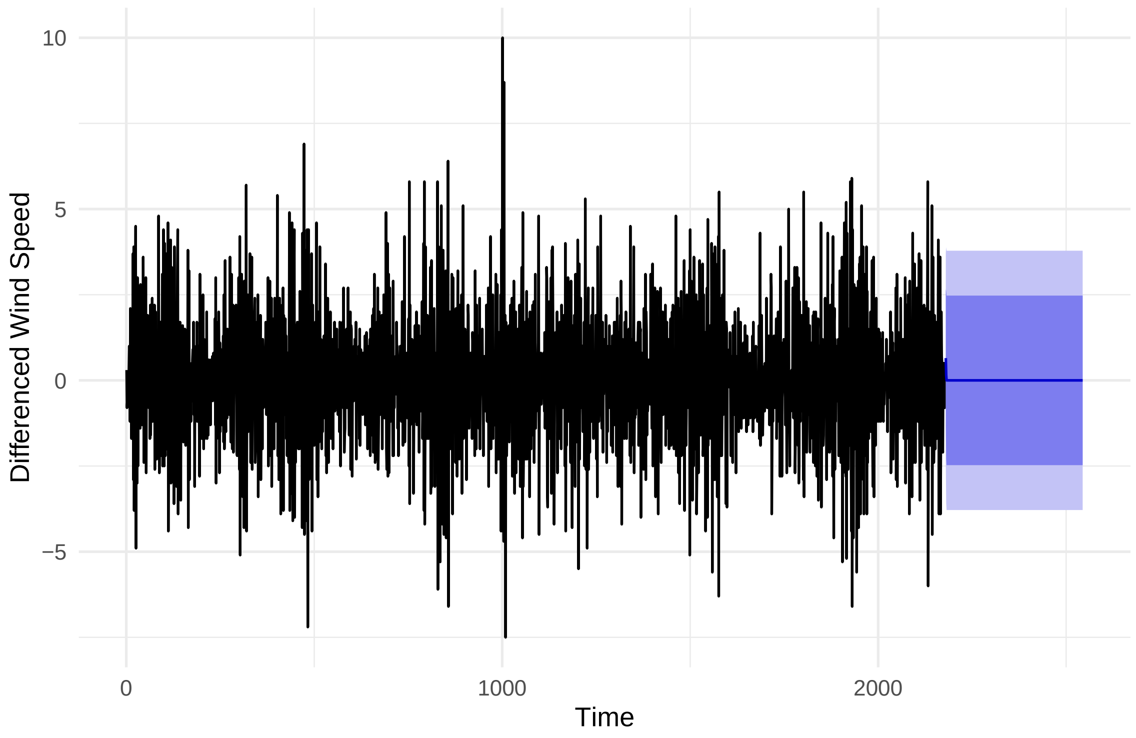

Figure 5: Forecast for the Differenced Wind Speed Based on ARIMA

I also tried to fit the **ARIMA** model for **temperature** sequence by the same procedure as introduced in fitting the **wind speed** sequence. The fitted model for the differenced temperature sequence is **ARIMA(3,0,1)**, or equivalently **ARIMA(3,1,1)** for the original temperature sequence. The expression of the fitted model is shown in Equation (3).

```r
fit.temp <- auto.arima((d.temp))
fit.temp
```

```
## Series: (d.temp)
## ARIMA(3,0,1) with zero mean
##
## Coefficients:
##          ar1      ar2      ar3      ma1
##       0.5108  -0.1500  -0.0323  -0.7680
## s.e.  0.0345   0.0244   0.0255   0.0274
##
## sigma^2 = 36.31:  log likelihood = -7016.52
## AIC=14043.05   AICc=14043.07   BIC=14071.49
```

$$(1-L)\times y_t = 0.51(1-L)\times y_{t-1} - 0.15(1-L)\times y_{t-2} - 0.03(1-L)\times y_{t-3} + \epsilon_t - 0.77\epsilon_{t-1} \quad (3)$$

$$\epsilon_t \sim WN(0, 36.31)$$

Though the `auto.arima` gives the best model **ARIMA** model could develop, the model selection criterion *AIC* and *BIC* are extremely large for temperature sequence. This phenomenon force me to find a better model to fit the temperature data, which I will introduced in the later section.

Since the fitting is not good, we can also image a bad results in the residuals checking (in Figure 6) as well as the forecasting (in Figure 7).

3.2 Recurrent Neural Network

In order to obtain a better fit and look for a better forecast as well, I tried a prevailing machine learning technique, the **Recurrent Neural Network (RNN)** to fit the temperature sequence. I will start with the model description.

3.2.1 Model Introduction

Recurrent Neural Networks (RNNs) are a state-of-the-art neural network which is mainly used to deal with sequential data. Therefore, the first feature of RNN, sequential data processing requires the correct order of data points. Secondly, RNN maintains a hidden

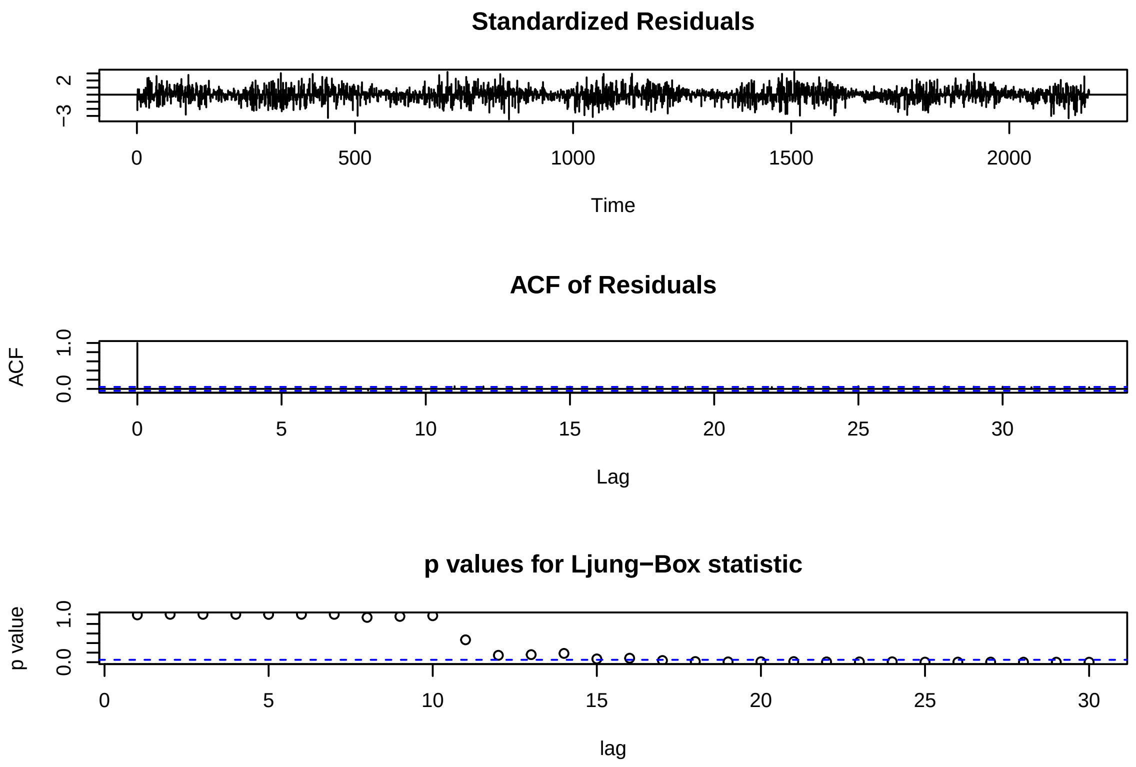

Figure 6: Residuals Checking for Temperature

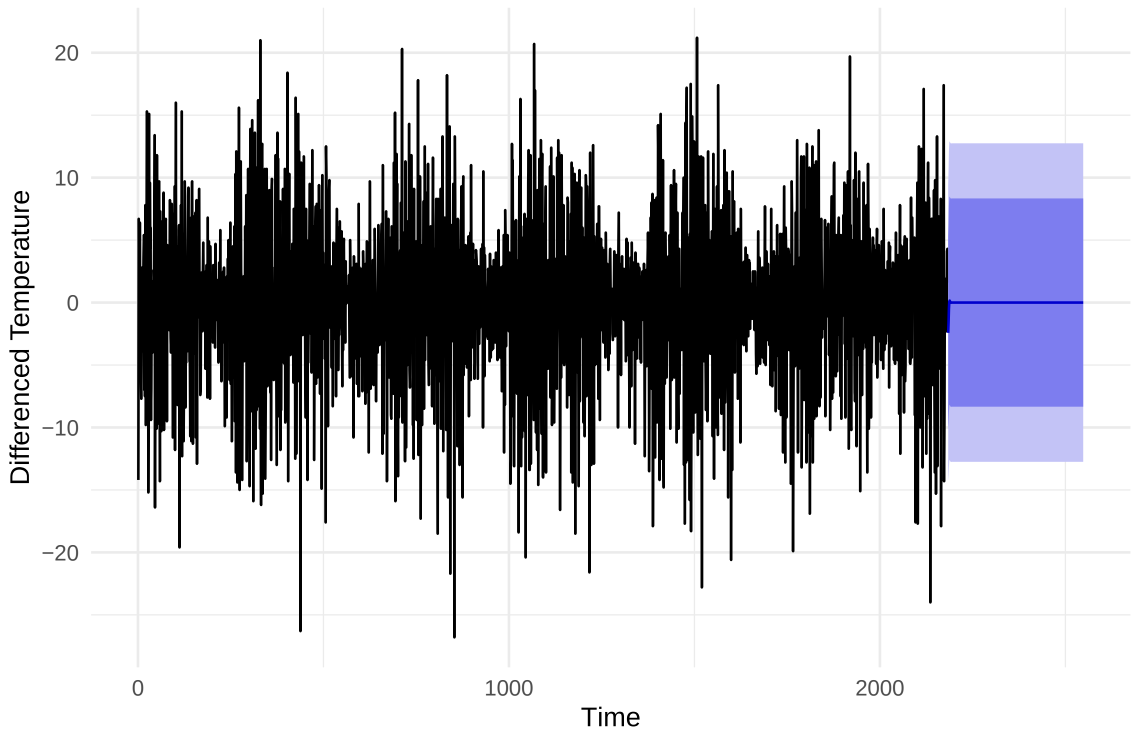

Figure 7: Forecast for the Differenced Temperature Based on ARIMA

state that is updated at each time step. This hidden state acts as a memory that captures information from previous time steps, allowing the network to learn temporal dependencies. Thirdly, RNN loops back from the hidden state to itself. This feedback loop enables the network to retain information over time and leverage past data to make predictions about future events. Finally, RNNs share parameters across different time steps, which reduces the number of parameters to be learned and makes the model more efficient. In a word, the RNN could synthesize the former information to help predict the current condition.

In mathematical introduction, we first define the the input sequence, hidden state and output sequence.

- **Input Sequence**: $\mathbf{x} = \{x_1, x_2, \dots, x_T\}$
- **Hidden State**: $\mathbf{h}_t$ at time step t
- **Output Sequence**: $\mathbf{y} = \{y_1, y_2, \dots, y_T\}$

The hidden state $\mathbf{h}_t$ is updated using the input at the current time step x_t and the hidden state from the previous time step $\mathbf{h}_{t-1}$ as shown in Equation (4)

$$\mathbf{h}_t = \sigma(\mathbf{W}_{xh} x_t + \mathbf{W}_{hh} \mathbf{h}_{t-1} + \mathbf{b}_h) \tag{4}$$

Notations:

- σ: a non-linear activation function (e.g., tanh or ReLU).
- $\mathbf{W}_{xh}$: the weight matrix for the input to hidden state connection.
- $\mathbf{W}_{hh}$: the weight matrix for the hidden to hidden state connection.
- $\mathbf{b}_h$: the bias term for the hidden state.

Then the output at each time step y_t can be computed as Equation (5)

$$y_t = \mathbf{W}_{hy} \mathbf{h}_t + \mathbf{b}_y \tag{5}$$

Notations:

- $\mathbf{W}_{hy}$: the weight matrix for the hidden state to output connection.
- $\mathbf{b}_y$: the bias term for the output.

For my study, I perform the study in `python` and follow the steps as shown below:

1. Scale the Data using `MinMaxScaler`
2. Convert Time Series to Supervised Learning Problem
3. Reshape Data for RNN Input: [samples, time steps, features].
4. Build the RNN Model using `tensorflow.keras`
    - Use a Sequential model

- Add a SimpleRNN layer with 50 units
- Add a Dense layer with 1 unit for the output
- Compile the model with the **Adam optimizer** and **Mean Squared Error (MSE)** loss function

5. Train the Model:
    - 100 epochs with a batch size of 32
6. Make Predictions
7. Model Evaluation using `sklearn.metrics`

For the model evaluation part, since the model couldn't automatically generate the *AIC* and *BIC* value and make our comparison with **ARIMA** hard, I manually calculate the *AIC* (Equation (6)) and *BIC* (Equation (7)) based on *Mean Squared Error (MSE)* value.

$$AIC = n * log(mse) + 2 * k \tag{6}$$

$$BIC = n * log(mse) + k * log(n) \tag{7}$$

3.2.2 Model Fitting and Forecast

As the RNN model couldn't generate the specific fitting formula, I use evaluation measures to show the goodness of fit and figure to show both the fitting and forecast.

Model Evaluation

I summarize some common model evaluation measures, which is shown in Table 2. We could see both the *AIC* and *BIC* decrease a lot compared to the **ARIMA** model for temperature sequence.

Table 2: RNN Evaluation

Model	Value
MSE	78.277
MSA	6.990
R^2	0.855
AIC	-6770.927
BIC	8296.996

Fitting and Forecast Figure

Then I draw the figure to show the model fitting and forecast for another 1 year, namely out-of-sample prediction for 365 days, which is shown in Figure 8. We could see the fitting for RNN is quite good, but the out-of-sample prediction is still unsatisfactory. The main reason is that the model couldn't get enough information, especially more current information and, it only forecasts using a lot former information. Thus, there is still a need for us to develop a model to help us better forecast these daily sequence.

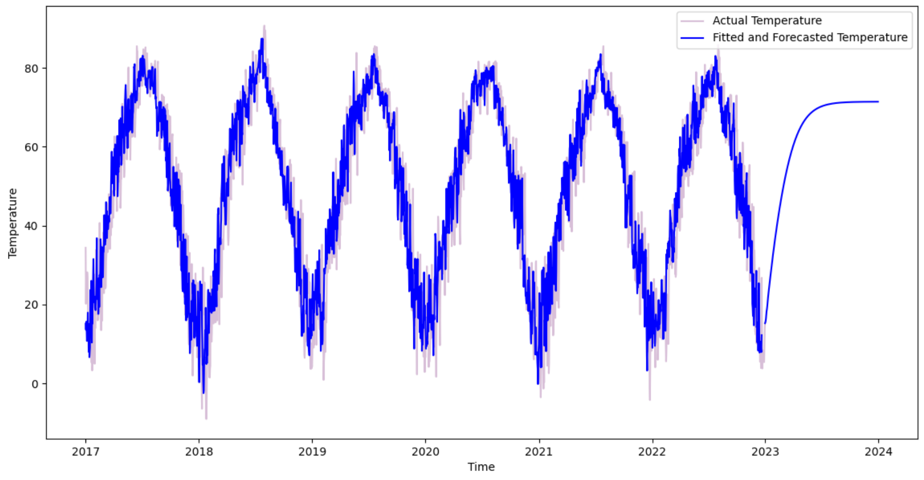

Figure 8: Temperature Fitting and Forecast from RNN

3.3 Prophet

Since both the **ARIMA** and **RNN** couldn't give a good illustration for the out-of-sample prediction of 365 days, I find another model called **Prophet Model** to generate a better forecast.

3.3.1 Model Introduction

The **Prophet** model or "Facebook Prophet", is an open-source library for univariate time series forecasting developed by Facebook. It is particularly suitable for forecasting time series with obvious *seasonal periodicity* such as temperature, commodity sales, traffc flow, etc.

The usage of **Prophet** model in *R* firstly requires the input data requires columns of `ds` and `y`, containing the *date*[2] and *numeric value* respectively. Also, *R* has already developed a package called `prophet` and we could use the `prophet()` function to fit the data. As for forecast, I use the `make_future_dataframe()` and `predict()` function.

[2]The `ds` column should be YYYY-MM-DD for a date, or YYYY-MM-DD HH:MM:SS for a timestamp

3.3.2 Model Fitting and Forecast

The fitting and forecast figure of **wind speed** is shown in Figure 9 and the figure of **temperature** is shown in Figure 10. From both figures, we could see a huge developement of the out-of-sample prediction and the inclusion of seasonality. Therefore, we get to find a good model to predict the daily weather sequence.

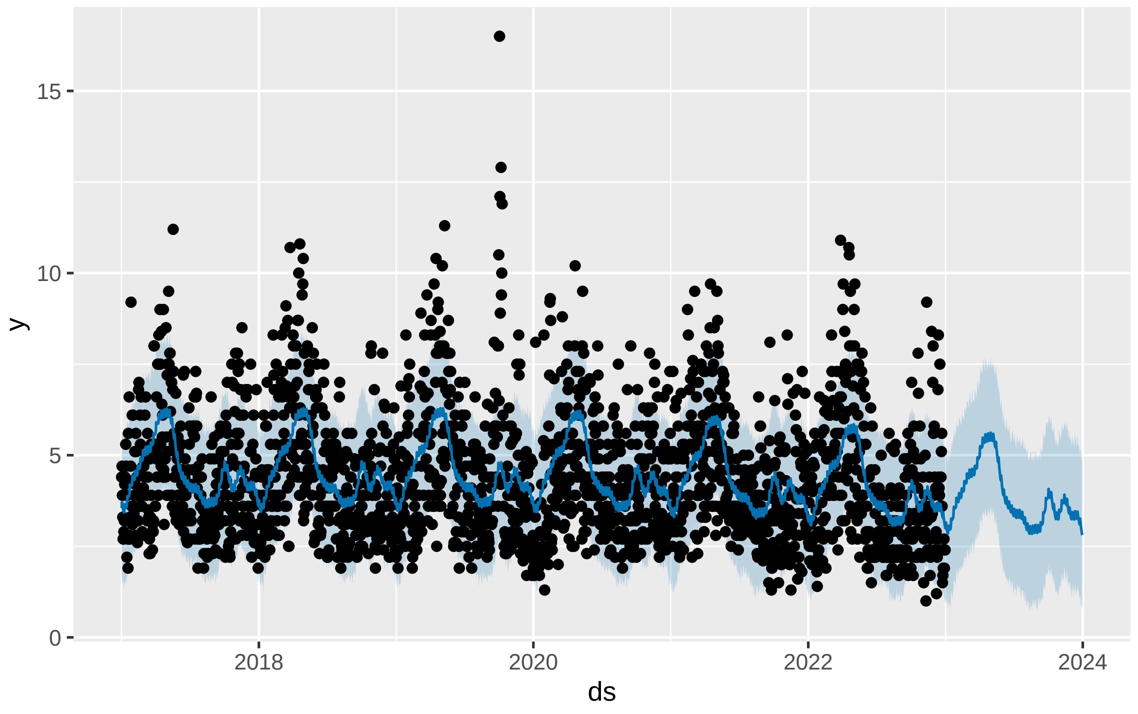

Figure 9: Prophet Model Fitting: Wind Speed

4 Conclusion

Weather prediction is an essential tool that impacts almost every aspect of modern life. In this study, I focus on the city of Shenyang in China as a sample to establish a model for analyzing model composition and predicting weather conditions. Specifically, the study examines the **temperature** and **wind speed** data from the Shenyang weather station over a period of 2184 days, from January 1, 2017, to December 30, 2022.

In order to get a good prediction of these daily sequence, I started with **ARIMA** model and altogether tried for 3 models. The first **ARIMA** model is limited at fitting the daily weather data. Also the forecasting is not that good at the level of day as well and couldn't efficiently utilize the former information. Due to the extremely bad fit for the **temperature** sequence based on **ARIMA**, I used the **Rucurrent Neural Network (RNN)** and tried to get a better fit. The result shows an satisfactory model to fit the data but the forecasts for the future still need to be developed. Similar as **ARIMA**, I think the main reason is that the **RNN** model also couldn't get enough information, especially more current information, and

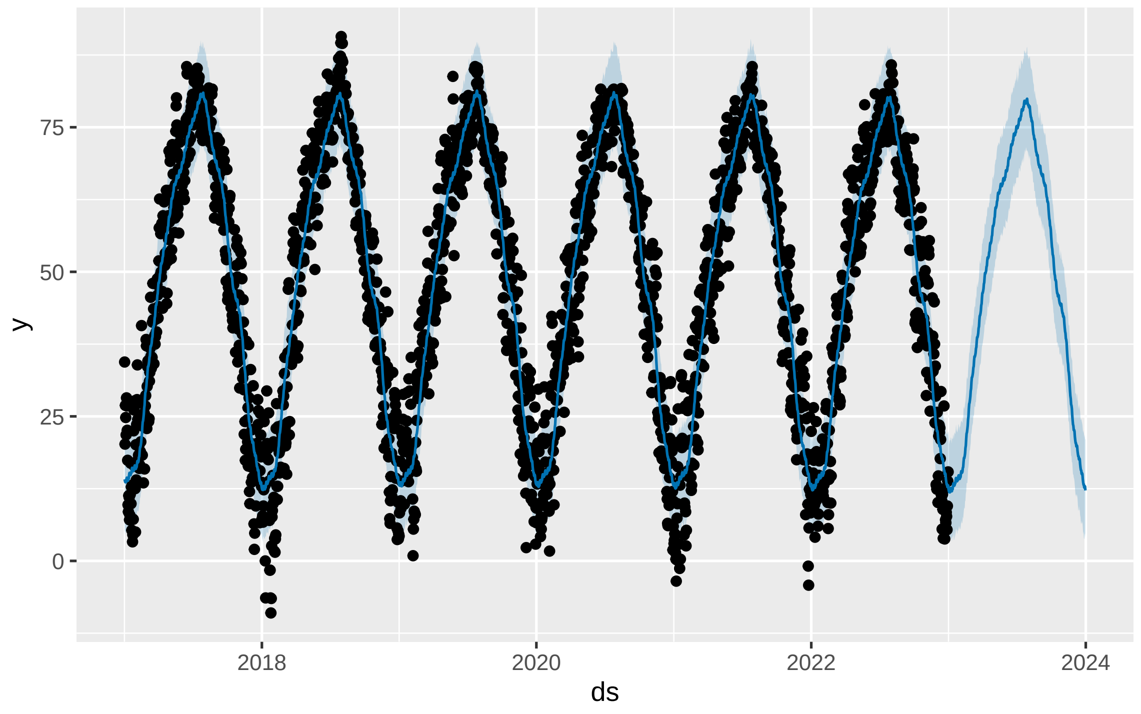

Figure 10: Prophet Model Fitting: Temperature

it only forecasts using a lot former information. Thus, I finally based on the **Prophet** model to find the better forecast. From the figures, we can tell the**Prophet** model indeed gives an satisfactory model to implement forecasting and explore the seasonal pattern behind it, which could serve as a strong tool to help predict sequence with obvious periodicity.