Journal of Civil Engineering and Construction 2024;13(1):12-28
https://doi.org/10.32732/jcec.2024.13.1.12

Sea Surface Salinity Forecasting with a Comparison Studying Case of GMM-VSG and FB-Prophet Model

Zhenlin Xiong[1], Elham Farazdaghi[1], Jena Jeong[1], Nicolas Guillou[2], Georges Chapalain[2]

1. Institut de Recherche, École spéciale des travaux publics, du bâtiment et de l'industrie, Cachan, France
2. DTecREM, Centre d'études et d'expertise sur les risques, l'environnement, la mobilité et l'aménagement, Plouzane, France
E-mail: zhenlin.xiong@estp.fr; efarazdaghi@estp.fr; jjeong@estp.fr; nicolas.guillou@cerema.fr; georges.chapalain@cerema.fr

Received: 30 November 2023; Accepted: 15 January 2024; Available online: 5 March 2024

Abstract: With the evolution of artificial intelligence, the utilization of machine learning algorithms for predicting hydrological data has gained popularity in scientific research, especially for the development and operational patterns of marine-related objects in coastal regions. Salinity analysis plays a crucial role in evaluating the resilience and health of marine ecosystems. Traditional numerical models, although accurate, require significant computational resources. Therefore, this study assesses the effectiveness of GMM-VSG proposed by Shanghai University and FB-Prophet created by Meta (Facebook) as rapid alternatives for simulating the nonlinear relationships between salinity and various parameters, like tide-induced free surface elevation, river flows, and wind speed. The algorithms were tested using an eight-year dataset collected at the MAREL buoy at the entrance to bay of Brest. Results indicate that, despite the simplicity of the input data, both algorithms successfully reproduced seasonal and semi-diurnal fluctuations in salinity. This underscores their potential as complementary tools for the ecological monitoring in estuarine environments.
Keywords: GMM-VSG; FB-Prophet; Artificial intelligence; Time-series data of salinity; Estuary ecological monitoring; Non-linear relationships; Sea surface salinity fluctuations.

1. Introduction

Salinity is a key parameter in coastal regions of freshwater influence (ROFI) by reflecting the fate of riverine freshwater input from catchment area into marine saltwater. Salinity is directly associated with hydrodynamics and water quality issues. Thus, it may account for the fate of conservative dissolved pollutants carried by riverine freshwater. Assessing the temporal evolution of the salinity is therefore a valuable goal for the monitoring of marine and estuarine environments.

The salinity in a ROFI is primarily driven by tidal regimes and freshwater input discharges from a river. Freshwater input discharges are variable in time. This variability is characterized by a mix of short-term aperiodicity and seasonality, particularly noticeable in temperate areas. Thus, during late autumn, winter, and early spring, strong episodic freshwater discharges caused by heavy precipitations can lead to severe drops in the salinity [1]. In comparison, tides are continual harmonic phenomena characterized by two major cycles. The first one is the semi-diurnal tidal cycle with two high and low tides per lunar day. The second one is the neap-spring tidal cycle characterized by fortnightly variations of the tidal range and associated currents. Tide-induced free-surface elevation and currents show furthermore an increased spatial variability, particularly noticeable in coastal shelf seas in relation to the combined influences of the coastline, bathymetry, and bottom roughness. Freshwater discharges and tides finally interplay in complex manner at different temporal and spatial scales within the ROFI through dispersion.

Numerical coastal model based on a physical description of hydrodynamic and hydrological processes have proved their ability to simulate the evolution of the salinity distribution in ROFI. However, this kind of approach remains expensive in terms of computational resources and time. In recent years, machine learning methods fed by an extensive amount of field and/or numerical data have appeared to be a promising and cost-effective alternative approach. Thus, a study on the prediction of the temporal evolution of salinity was conducted in the bay of Brest (western Brittany, France) by relying on various well-known machine learning (ML) algorithms [1]. Obtained predictions compared favorably with results of an advanced numerical three-dimensional (3D) process-based physical coastal model. This study yielded interesting outcomes regarding the choice of an ML algorithm and the impact of a number of input parameters to enhance the selected algorithm. However, none of the algorithms exploited can effectively handle empty or missing data resulting from measuring system malfunction and maintenance. Even if the virtual data prediction proves to be more or less satisfactory, it would be necessary to identify another algorithm capable of addressing a significant amount of missing data.

Here, still in the context of the bay of Brest, we aimed to use two machine learning algorithms, GMM-VSG proposed by Shanghai University and FB-Prophet proposed by Meta Company, to predict small samples and large sample time series data, respectively, and generate virtual data to fill in the missing data [2,3]. To adapt the data to the algorithm, we exploited different re-processing methods, such as exploring the intrinsic interaction mechanism between different variables to reduce the data dimension or directly introducing dimensionality reduction and noise reduction algorithms.

2. Research method

The research is principally realized by algorithm FB-Prophet [2]. In addition, we also execute a comparison study by GMM-VSG [4]. The function of manually entering holidays of FB-Prophet is utilized to describe the input conditions by listing all the dates of spring tides and neap tides. Considering the tide is always two or three days later than the phases of the moon, the time limit of impact is set from the corresponding date to the two following days after the date. In addition, the function originally used in the algorithm FB-Prophet can be utilized to simulate the impact of winter floods on the salinity [1]. The different Fourier numbers will be given to the seasons for better fitting. Details are shown in Figures 1 and 2.

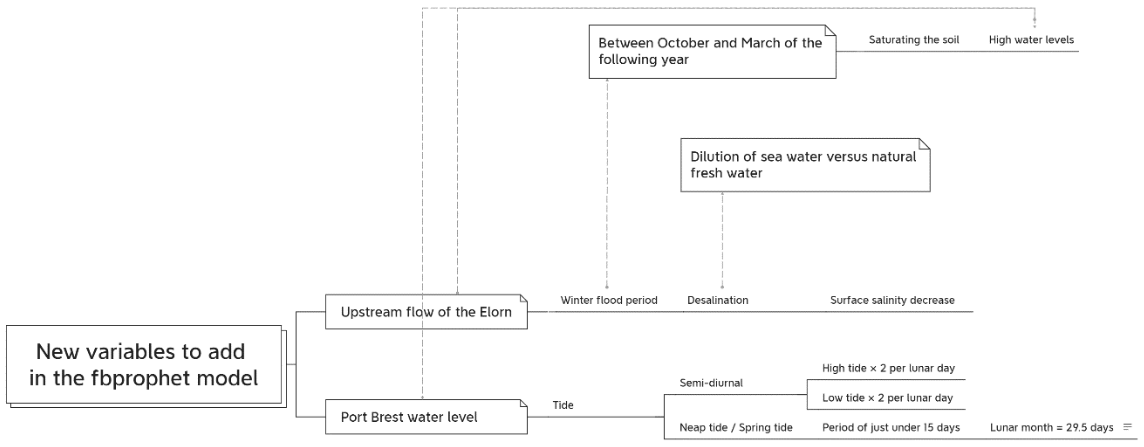
Between October and March of the following year
Saturating the soil
High water levels
Dilution of sea water versus natural fresh water
New variables to add in the fbprophet model
Upstream flow of the Elorn
Winter flood period
Desalination
Surface salinity decrease
Port Brest water level
Tide
Semi-diurnal
High tide × 2 per lunar day
Low tide × 2 per lunar day
Neap tide / Spring tide
Period of just under 15 days
Lunar month = 29.5 days

Figure 1. New variables to add in the FB-Prophet model.

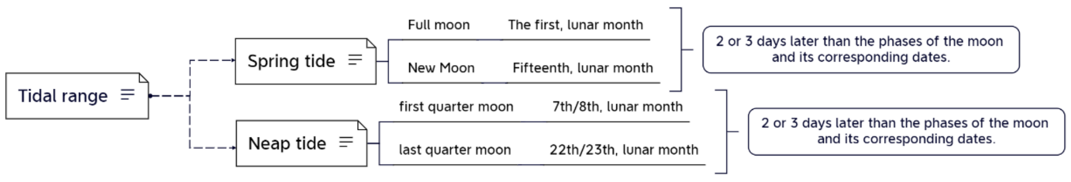

Figure 2. Tidal phenomena and moon phases

2.1 Background description

The unit to describe salinity is psu, which is the abbreviation of Practical Salinity Unit. PSU equals to gram of salt by kg of the sampled seawater. These data are measured at the MAREL buoy at the entrance to bay of Brest. That’s the reason why the data we use is kind of different with that in the article Predicting Sea Surface Salinity in a Tidal Estuary with Machine Learning [1], which is also measured at the mouth of the Elorn River by our co-working enterprise CEREMA. There are in total 181,091 items in the dataset, with salinity in psu and the time point in second (YYYY-MM-DD hh-mm-ss), with free-surface elevation by m and river flow at Brest harbor by m3/s. The dataset lasts for 8 years, from 00:16, January 1st, 2015, to 08:40 September 20th, 2022. The intervals between the 2 time points are 20 or 30 minutes. As for the missing intervals of the dataset, varying from 7 days to 107 days, there are 36 in total (Table 1). We can observe them in the following picture (Figure 3 and 4). The biggest gap, from 31 March 2017 to 15 July 2017, is 107 days. In addition, the missing parts of free-surface elevation and river flow upstream of Elorn are identical from 2015/01/05 07:47:00 (data from 2015/01/01 to 01/05 is missing). Concerning the final purpose, firstly, it's filling the blank intervals with the data generated by designated machine learning algorithms. And then, it's the prediction of the data for the future until 2023 or until 2024.

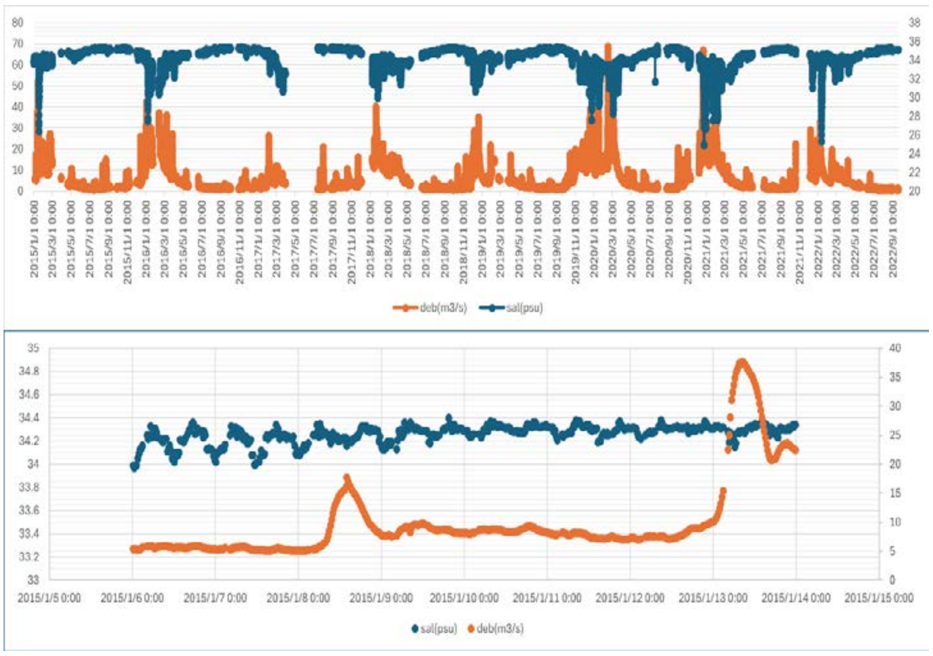

Figure 3. Time-evolution of measured Elorn river flow and MAREL buoy salinity over 8 years from 2015 to 2023 (top) and 3 days from month day, year to month day, year (bottom).

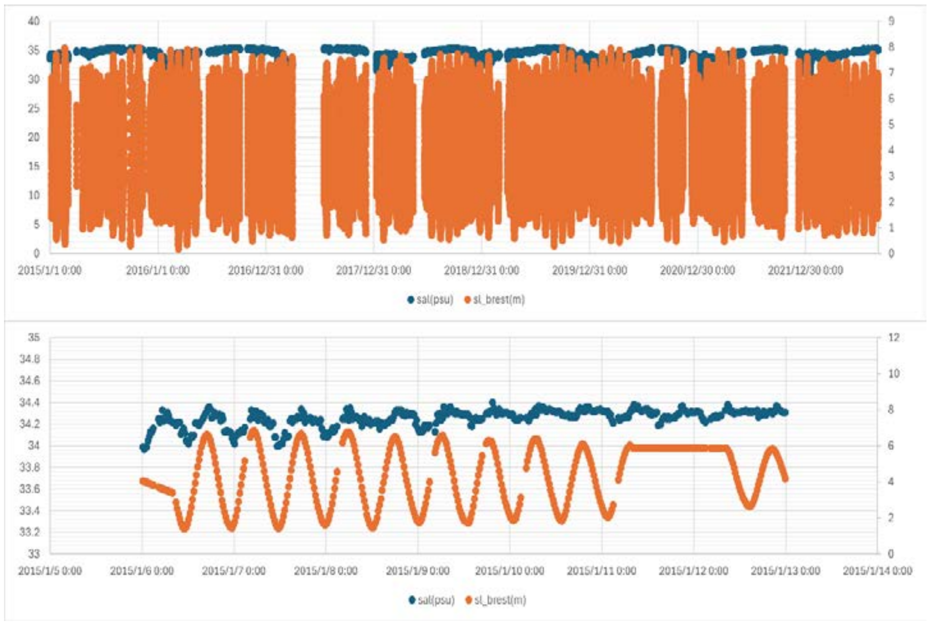

Figure 4. Time-evolution of measured free-surce height at Brest and MAREL buoy salinity over 8 years from 2015 to 2023 (top) and 3 days from month day, year to month day, year (bottom).

in which the missing dates of beginning and end of the depicted short measurement period need to be completed:

Table 1. The information of missing parts of the data

N(Year)	Gap start time	a (row)	Gap finish time	b (row)	General information
2015	2015/2/1 0:17	2095	2015/2/10 23:17	2812	0201-0210 10 days
	2015/3/3 9:15	4126	2015/3/31 10:15	6145	0303-0331 29 days
	2015/3/31 19:07	6172	2015/4/21 12:07	7663	0331-0421 21 days
	2015/7/1 0:00	12376	2015/7/9 23:40	13023	0630-0710 9 days
	2015/7/23 2:36	13801	2015/8/1 9:16	14469	0723-0801 9 days
	2015/9/7 15:38	16745	2015/9/29 1:13	18029	0907-0929 23 days
	2015/10/1 0:10	18140	2015/10/9 23:46	18679	1001-1009 9 days
	2015/11/10 9:41	20466	2015/12/2 11:16	21790	1110-1202 22 days
2016	2016/1/25 2:53	24879	2016/2/12 4:28	25963	0125-0212 18 days
	2016/5/19 1:09	31448	2016/6/20 19:57	33415	0519-0620 33 days
	2016/8/1 0:06	35719	2016/8/9 0:06	36199	0731-0809 8 days
	2016/10/3 16:46	39474	2016/10/30 11:10	41080	1003-1030 28 days
2017	2017/3/31 10:59	49695	2017/7/15 2:59	56035	0331-0715 107 days
	2017/8/7 0:22	57158	2017/8/26 13:33	58331	0807-0826 19 days
	2017/11/1 0:22	61673	2017/11/10 23:57	62272	1101-1111 10 days
	2017/12/4 9:00	63373	2018/1/8 9:23	64901	1204-0108 36 days
2018	2018/2/24 18:49	66898	2018/3/7 23:09	67598	0224-0307 12 days
	2018/5/13 9:06	72266	2018/6/20 12:46	75013	0513-0620 39 days
	2018/12/11 19:20	86779	2018/12/20 1:47	87346	1211-1220 9 days
2019	2019/1/10 23:37	88786	2019/1/18 13:57	89307	0110-0118 8 days
	2019/2/2 11:07	90333	2019/2/11 8:07	90972	0202-0211 9 days
	2019/2/11 14:19	90990	2019/2/24 9:19	91911	0211-0224 14 days
	2019/2/28 18:56	92145	2019/3/27 17:22	93992	0228-0327 29 days
	2019/11/3 1:53	108965	2019/11/15 23:12	109893	1103-1115 13 days
2020	2020/2/1 0:15	114820	2020/2/10 23:55	115539	0131-0211 10 days
	2020/7/1 0:10	125403	2020/7/7 23:49	125906	0630-0708 7 days
	2020/7/27 19:10	127328	2020/8/26 20:50	129493	0727-0826 30 days
	2020/11/11 10:30	134872	2020/12/9 11:09	136890	1111-1209 29 days
2021	2021/1/11 13:50	139260	2021/1/21 22:50	140007	0111-0121 11 days
	2021/6/1 12:40	148697	2021/7/8 1:40	151228	0601-0708 37 days
	2021/8/31 23:00	155089	2021/9/8 23:20	155666	0831-0908 8 days
	2021/10/20 7:00	158250	2021/12/5 23:00	161610	1020-1205 47 days
2022	2022/2/1 0:00	165635	2022/2/9 23:39	166282	0131-0210 11 days
	2022/5/20 11:20	172997	2022/6/9 11:40	174438	0520-0609 21 days
	2022/9/1 21:40	179791	2022/9/15 19:40	180793	0901-0915 15 days
	2022/9/20 9:00	181093	2022/12/31 23:40	183555	0920-0101 101 days

A virtual environment is built for the FB-Prophet in Anaconda3 with Spyder 5.3.3. Python 3.7.16 is used for running the model. Pystan 2.19.1.1 and Plotly 5.9.0 are installed previously for the visualization and uncertainty interval estimation. The saving of the model is realized by Joblib 1.1.1. The operating system of the device is a 64-bit Windows OMEN Laptop equipped with an AMD core Ryzen 7 4800H CPU @2.90 GHz with Radeon Graphics, 8 cores, 16 Logical Processors, and 16GB Ram. The data is pre-processed to be input into the model and utilized.

2.2 GMM-VSG

GMM-VSG is originally selected from the article “A virtual sample generation algorithm supporting machine learning with a small-sample dataset: A case study for rubber materials” [3]. The algorithm GMM-VSG is composed of two parts, which are PCA (Principal Component analysis) [5] and GMM (Gaussian Mixture Model). First of all, the PCA will transfer the matrix composed of several features into one single feature while retaining most of the original variance with less loss of information quantity. It's been widely used in the visualization of high-dimensional data and noise reduction in the Machine Learning field.

As for the GMM, the essence of the mixed Gaussian model is to merge several simple Gaussian models to make the model more complex, thus generating more complex samples. Theoretically, if the number of Gaussian models merged by a certain mixed Gaussian model is large enough, this mixed model can fit samples of any distribution.

$$p(x) = \sum_{k=1}^{K} p(k)p(x|k) = \sum_{k=1}^{K} \Pi_k N\left(x \middle| u_k, \textstyle\sum_k\right) \tag{1}$$

p(x|k) is the probability density function of the number k Gaussian model, which can be thought of as the probability that the model produces X after selecting the kth model. p(k) = pik is the weight (comparative importance) of the Kth Gaussian model referred to as the prior probability of selecting the kth model and satisfying the sum of pk is 1.

Here we can see clearly how GMM works. Different Gaussian models together form a curved surface in multiple dimensions, which can describe or simulate any kind of data.

$$X \sim N(\mu, \sigma^2) \;\; f(x) = \frac{1}{\sigma\sqrt{2\Pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2}$$

But it's difficult to find out the parameters μ, σ of the mixture Gaussian distribution. So, we introduce the Expectation-Maximization algorithm (EM) to solve that with the tool MLE (Maximum likelihood estimation). From all these known samples, we need to find a suitable Gaussian mixture distribution (determined by the parameters μ, σ of the Gaussian distribution), to produce as much probability as possible for this set of samples. It's more like an iterative process.

Initialize, perform the E-step, execute the M-step, and iterate until the Maximum Likelihood Estimation (MLE) is sufficiently close [6].

$$\hat{\theta} = arg \max_{\theta} \sum_i \sum_{z^{(i)}} Q_i \left(z^{(i)} \log \frac{p\left(x^{(i)}, z^{(i)}; \theta\right)}{Q_i(z^{(i)})} \right) \tag{3}$$

There is the Akaik Information Criterion (AIC)/ Bayesian Information Criterion (BIC) strategy aiming at balancing accuracy and the performance of the model [7]. Just like in the chart, the value of X in the lowest point is the best. In the article mentioned above, 6 features are being researched at the same time. GMM-VSG is not sensitive to the inner relationship between features. Even though several features are relative to each other, and the features are arranged parallelly without pre-treatment, the performance of the algorithm is still perfect. In our case, we have 3 features to be researched.

As for the environment for algorithm GMM-VSG, to save time, another computer is used at the same time. The environment is the default base(root) environment in Anaconda3(2022.10) with Conda 23.1.0 and Spyder 5.2.2. Python 3.9.13 is used for running the model. The scikit-learn 1.0.2 is required to realize the program. The operating system of the device is a 64-bit Windows Desktop computer equipped with an Intel(R) Core (TM) i7-7700 CPU @3.60 GHz, 4 cores, 8 Logical Processors, 16GB RAM.

2.3 FB-Prophet

FB-Prophet was developed in 2017 during their work in the Core Data Science group at Meta (formerly known as Facebook). All the principles and usages of the algorithm are explained in the article published in the 6 journals The American Statistician for the name as Forecasting at Scale [2].

FB-Prophet is a time series forecasting model designed to handle the common features of business time series. Importantly, it is also designed to have intuitive parameters that can be adjusted without knowing the details of the underlying model. This is necessary for the analyst to effectively tune the model. The implementation is available as open-source software in Python and R. The time series model in the FB-Prophet is decomposable, consisting of three main components: Trend, Seasonality, and Holidays. They're supposed to be combined by a multiplicative model or GAM (generalized additive model) [8]. In case the period and amplitude of the data don't vary from the time, the GAM (Generalized Additive Model) is the preferable choice.

The equation expression is:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t \tag{4}$$

where, g(t) represents the trend function capturing non-periodic changes in the time series value, s(t) accounts for periodic changes such as weekly or yearly seasonality, and h(t) reflects the effects of holidays occurring potentially on irregular schedules over one or more days. The error term εt represents any idiosyncratic changes not accommodated by the model. Additionally, there is usually a parametric assumption that εt follows a specific distribution. There are two different built-in growth models, which are the saturating logistic growth model with a carrying capacity of C and the piecewise linear growth model without saturating growth. The expression of the logistic growth model is (SIGMOID function) [9]:

$$g(t) = \frac{C}{1 + e^{-k(t-m)}} \tag{5}$$

The expression of the linear growth model is:

$$g(t) = kt + m \tag{6}$$

In this case, the trend of the salinity data obviously does not conform to the linear law. As a result, logistic growth model is the preferable choice. Considering that the coefficients vary from time to time, it's necessary to assign the coefficients as variables, which is like:

$$g(t) = \frac{C(t)}{1 + e^{-\left(k+\alpha(t)^T\delta\right)\left(t-\left(m+\alpha(t)^T\gamma\right)\right)}} \tag{7}$$

where k represents the growth rate, m is the offset variable, δ represents the vector of growth rate adjustments, γ represents the vector of correction modifications at change points aiming to make the trend function differential at every point on the timeline, and a(t) is the vector of modification parameters.

Concerning the selection of the changepoints, the default quantity is 25 changepoints located at the former 80 percent of the timeline, which is also recommended. The selection is automatically executed by putting a sparse prior on δ. The prior δj observes Laplace distribution, which is δj ~ Laplace (0, τ). The parameter τ is relevant to the parameter changepoint before scale, which can be adjusted directly during the operation. As for the changepoints in the prediction part, the quantity is calculated by the ratio of the changepoints in the time points in the timeline. The variance of Laplace distribution of the changepoints in the prediction part is determined by the variance of the changepoints in historical data. FB-Prophet fits the seasonality by Fourier transform. The seasonal components are all periodic, and the final form of a time series may be the result of a variety of seasonal components of different cycles acting on the trend component, and each seasonal component can be fitted with a set of sine waves.

The expression is:

$$s(t) = \sum_{n=1}^{N} (a_n \cos(\frac{2\pi nt}{P}) + b_n \sin(\frac{2\pi nt}{P})) \tag{8}$$

Such a set of sine waves requires 2N parameters, which will be calculated during the transformation process. However, it's required to determine an N to ensure the accuracy of the transformation process while 7 preventing over-fitting and reducing the pressure on computing power. The default seasonality is yearly, weekly, and daily seasonality. If there is a need to analyze periodic regularity on month or on specific period, it is possible to add by hand. Normally, for yearly components, using N = 10, P = 365.25 is a suitable for the most case. For weekly components, it turns to N = 3, P = 7, which can achieve good results on most problems. Both parameter N and P is adjustable. In addition, FB-Prophet also operates a Gaussian Smoothing process on these 2N parameters. The degree of smoothing will also determine the speed of seasonal component changes, which is adjustable parameter seasonality prior scale. This parameter can also be used to regulate the intensity of seasonal components influencing the final shape of the time series.

$$s(t) = X(t)\beta \tag{9}$$

$$X(t) = \left[\cos\left(\frac{2\pi(1)t}{P}\right), \ldots, \sin\left(\frac{2\pi(N)t}{P}\right)\right] (assume\ N\ as\ an\ even) \tag{10}$$

$$\beta \sim Normal(0, \sigma^2) \tag{11}$$

Some events are closely related to time, such as certain holidays, which do not appear periodically on a long-time scale. However, it still meets certain rules of occurrence. Such important influences are fed into the model in the form of data tables. The prophet counts the occurrence time of each holiday, attaches an influence value k to each holiday (sampled from a normal distribution), and adds an influence value of holiday at the corresponding time of each holiday (or all the time within a window centered on the holiday), as the value of h(t) in the previous equation. Details are shown in Figure 5:

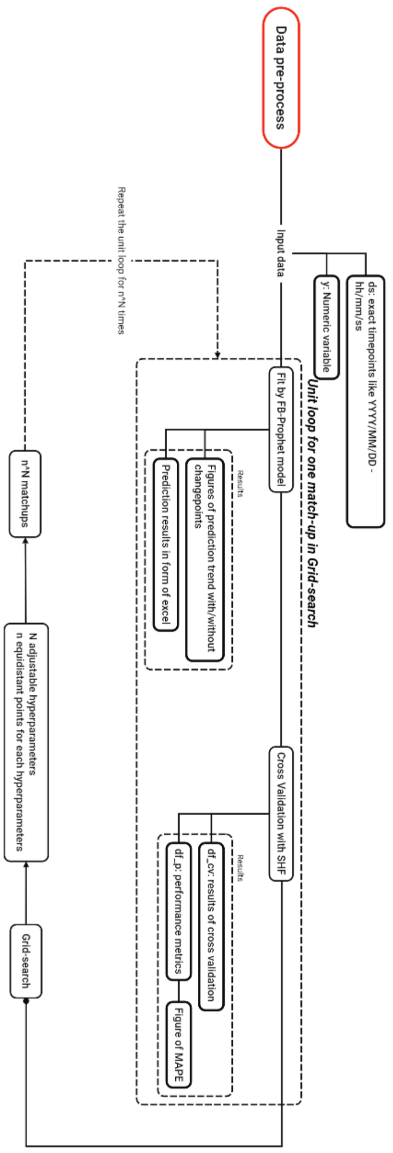
Data pre-process
Input data
ds: exact timepoints like YYYY/MM/DD - hh/mm/ss
y: Numeric variable
Unit loop for one match-up in Grid-search
Fit by FB-Prophet model
Results
Figures of prediction trend with/without changepoints
Prediction results in form of excel
Cross Validation with SHF
Results
df_cv: results of cross validation
df_p: performance metrics
Figure of MAPE
Repeat the unit loop for n^N times
n^N matchups
N adjustable hyperparameters
n equidistant points for each hyperparameters
Grid-search

Figure 5. The flow chart of FB-Prophet model illustration indicating the sequential steps and components involved in the modeling process.

2.4 Evaluation

In terms of metrics for the research, we have selected MAPE (Mean Absolute Percentage Error) and RMSE (Root Mean Square Error). MAPE, commonly used in time series prediction problems, is also a calculation method. MAPE is a measure of the prediction accuracy of a forecasting method in statistics. It usually expresses the accuracy as a ratio defined by the formula:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \tag{12}$$

where *At* is the actual value and F_t is the forecast value. The result of MAPE is expressed in the form of a percentage, which increases the convenience of figuring out the difference between two datasets [10]. RMSE measures the average difference between a statistical model's predicted values and the actual values. Mathematically, it is the standard deviation of the residuals. Residuals represent the distance between the regression line and the data points. RMSE quantifies how dispersed these residuals are, revealing how tightly the observed data clusters around the predicted values.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2} \tag{13}$$

where y_i indicates the forecast value and xi indicates the actual value. The RMSE has a range [0, +∞), with a value of 0 indicating a perfect consistency between the forecast and the actual values representing a perfect model. As the error increases, the RMSE value also increases.

3. Results and discussion

3.1 Preprocessing of the data

Prior to inputting the data into the FB-Prophet, it is necessary to preprocess the original data. This preprocessing includes two steps. First, the names of the original data must be changed. The name of timestamp column, which shall be in the date format of Pandas, should be replaced by 'ds'. Meanwhile, the name of the column of time series value, which shall be in numerical attribute, should be replaced by 'y'. This demonstrates that FB-Prophet is a highly encapsulated package, providing comprehensive functionality within a self-contained framework. Secondly, although FB-Prophet is an algorithm package which is insensitive to missing values, and is robust to changes in trends, it is still necessary to add the timestamps for the missing data manually [2].

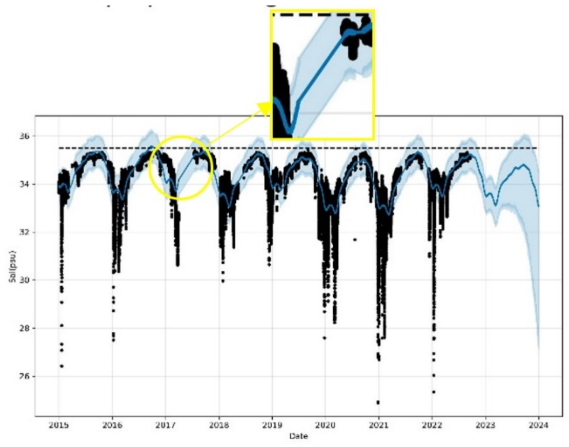

Figure 6. Result of the Logistic Saturating Growth model with default treatment for missing data by FB-Prophet.

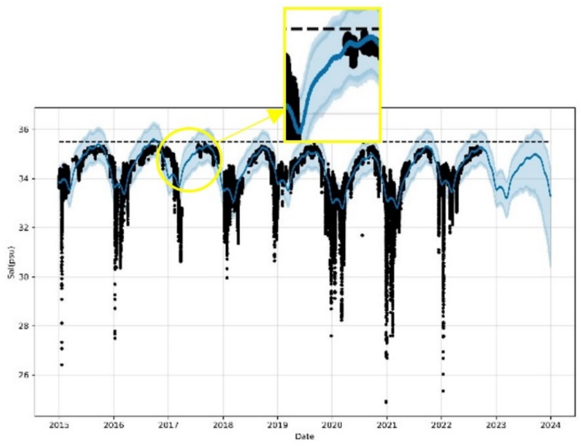

Figure 7. Result of the Logistic Saturating Growth model with manually adding the missing data timestamps.

Since the treatment by default of FB-Prophet towards missing data is to perform a pure linear fit between the known values at both ends of the missing value and set the corresponding value of the missing value abscissa in the linear function to the 'y' value. With the default method, neither the piecewise linear growth model nor the logistic saturating model is able to function on the missing data parts. As a result, we add the timestamps for the missing data before the input. From Figures 6 and 7, we could clearly observe the difference. Not only the missing data, but also the prediction part is influenced by the choice of preprocessing methods.

The deep blue line means the baseline, which is values obtained by FB-Prophet time series fitting. The black dots mean the discrete dots of the original time-series data. The area enclosed by the light blue line is the uncertainty interval, which is obtained by MAP (Maximum a posterior estimation) or MCMC (Markov chain Monte Carlo) Sampling [11].

3.2 Optimization of FB-Prophet

The optimization of FB-Prophet involves fine-tuning parameters and adjusting settings to enhance its forecasting accuracy and efficiency. After performing uniform preprocessing, one can observe the difference in results between the logical and linear models. In Figures 8 to 11, we can observe the difference in two parts:

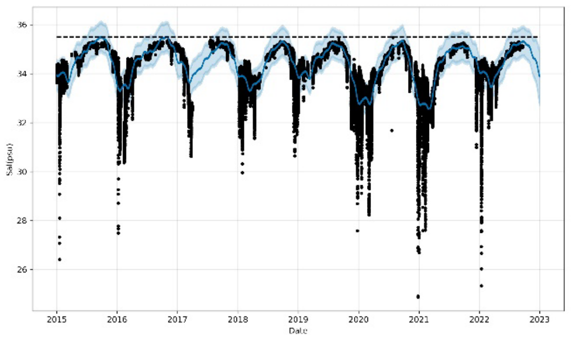
Figure 8. Results of linear growth model – prediction until 01/01/2023.

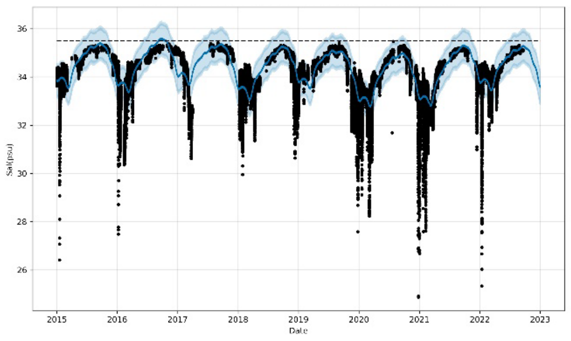
Figure 9. Results of logistic growth model – prediction until 01/01/2023.

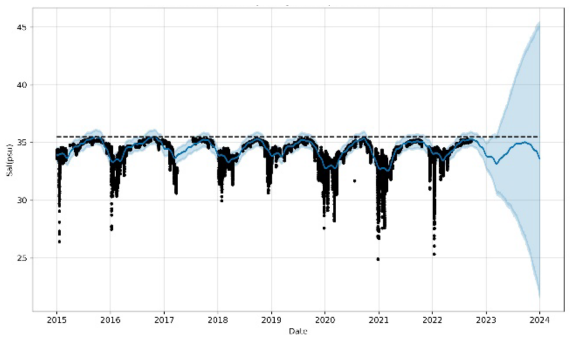
Figure 10. Results of linear growth model – prediction until 01/01/2024

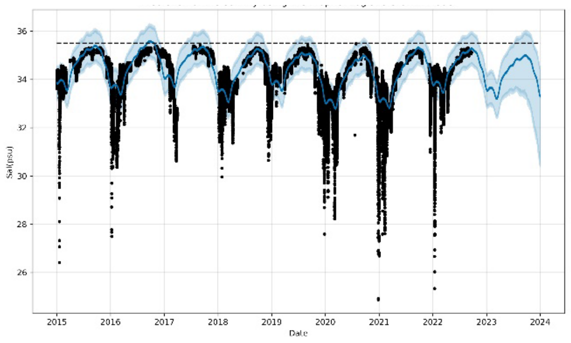
Figure 11. Results of logistic growth model – prediction until 01/01/2024.

The first part concerns missing data. Specifically, the missing data of the linear growth model is filled using piecewise linear growth functions, while that of the logistic growth model is filled using generalized nonlinear growth functions, resulting in a curved shape rather than a straight line. We can observe the difference between the two models in Figures 12 and 13 below:

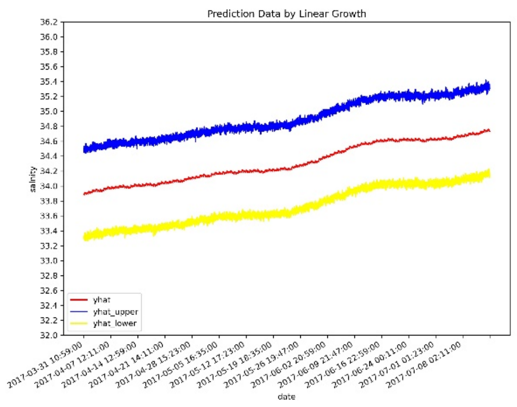

Figure 12. Missing data (2017/03/31 - 2017/07/15) generated by Linear growth model.

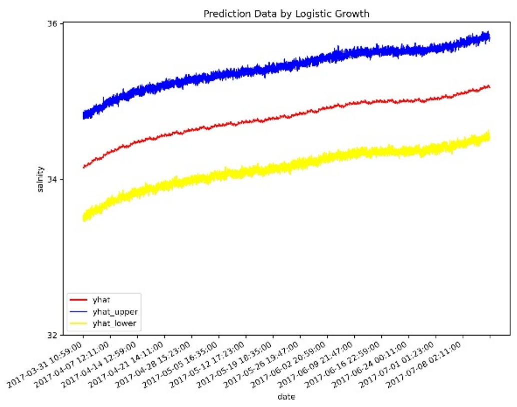

Figure 13. Missing data (2017/03/31 - 2017/07/15) generated by Logistic growth model.

The second part concerns the prediction part. The primary trend of the prediction, mainly reflected in the monotonicity, is similar. As the prediction time increases, the prediction accuracy tends to decrease. Notably, in the case of a linear growth model without a saturation upper cap, the uncertainty interval can rapidly expand, making the forecast meaningless. However, for the predicting values on baseline, there are only some subtle differences between the two models.

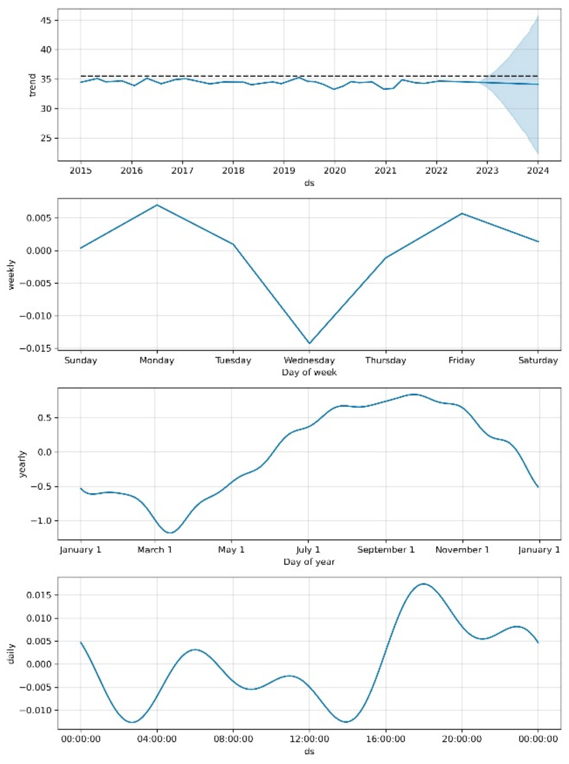

Figure 14. The periodic and non-periodic trend of the results obtained by the linear growth model until 2024

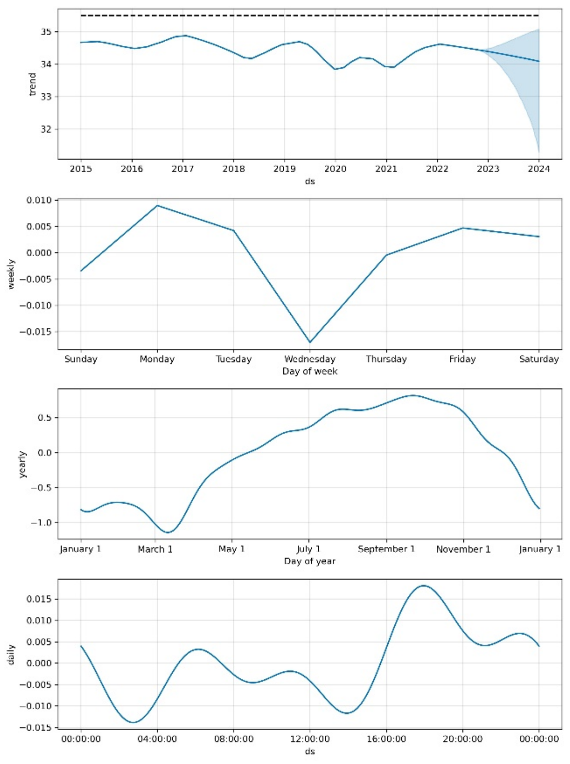

Figure 15. The periodic and non-periodic trend of the results obtained by the logistic growth model until 2024

Additionally, based on the component graphics in Figures 14 and 15, we observe that the difference between the two models primarily lies in the trend after eliminating the daily, yearly, and weekly effects. In the following steps, the SHF (Simulating Historical Forcast) cross-validation shall be executed using Grid-Search optimization [12]. SHF means that N time points are selected from the historical data resulting in isometric intervals (N-1 segments in total). Even if the data between two time points is known, cross-validation will still be implemented by training the data before the selected time point to get a predicted dataset for comparison with the known actual data. Theoretically, the data in the training part is supposed to be three times more than that in the testing part. This ensures that a a result is obtained, and its performance shares similarity with the k-fold cross-validation. Grid-search is a method for automatically searching for the best hyperparameter matchups using a 'for' loop statement. Technically, in our case, the adjustable hyperparameters as changepoint_prior_scale (ranging from 0.001 to 0.5), seasonality_prior_scale (ranging from 0.01 to 10), and holidays_prior_scale (ranging from 0.01 to 10) will be divided equidistantly into n(n>0) segments, respectively. As a result, n+1 values are obtained for each hyperparameter, resulting in a total of $(n+1)^3$ matchups. After evaluating all possible matchups, the one with the least MAPE value shall be chosen as the best-fitted one. However, the grid-search process demands significant computational resources or can result in extended processing time. In such cases, it's recommended to consider adopting the dichotomy method to ensure the normal computation operation, even though it may involve manual intervention. Based on the MAPE value of each matchup, the best-fit matchup is selected. In our case, the best-fit matchup among all 64 possibilities for the parameters is changepoint_prior_scale = 0.001, seasonality_prior_scale = 0.01, and holidays_priority_scale = 1. Figures 16 to 21 illustrate the results of SHF cross-validation for the best-fit matchup.

Particularly noteworthy is the cross-validation result for Period 3, where the fit performance is significantly better than that of the default parameter matchup (Demonstration in following Figures 22 and 23).

With the best-fit parameter matchup, we present the figure of MAPE values, which provides both intuitive and quantitative insights into the performance of cross-validation.

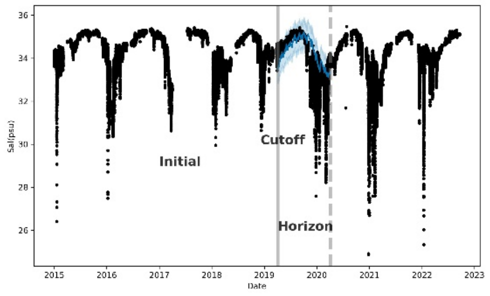

Figure 16. Period 1

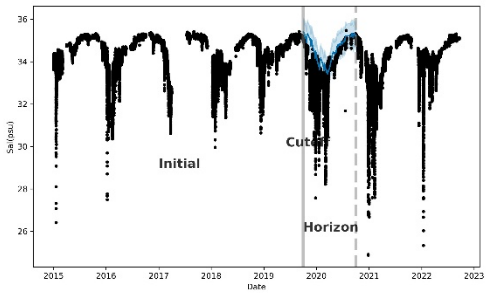

Figure 17. Period 2

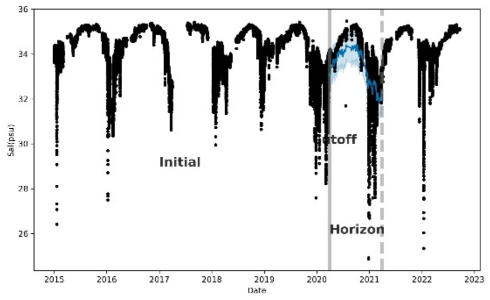

Figure 18. Period 3

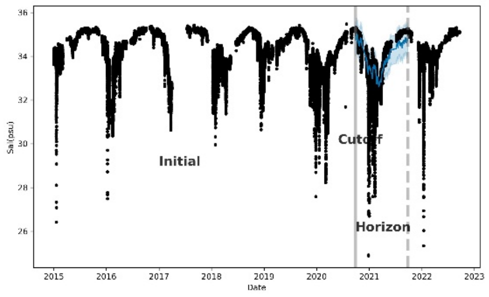

Figure 19. Period 4

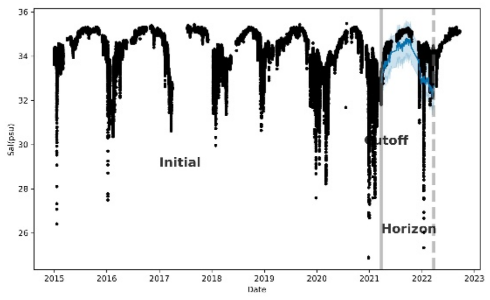

Figure 20. Period 5

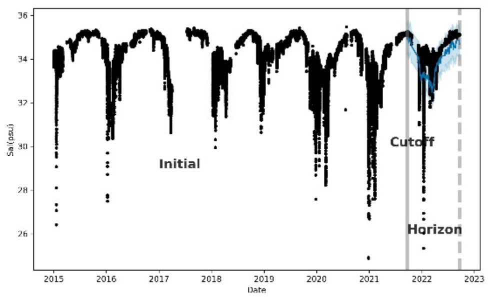

Figure 21. Period 6

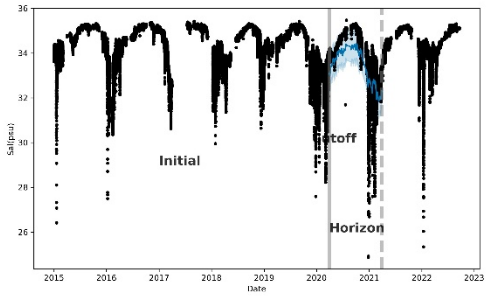

Figure 22. Cross-validation period 3 after Grid-Search

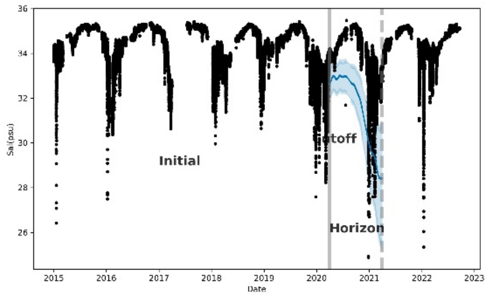

Figure 23. Cross-validation period 3 before Grid-Search

From Figures 24 and 25, it is evident that the MAPE values for the default parameter matchup in Figure 24 increase from around 0.02 at the beginning to almost 0.05 at the end of year (Cross-validation horizon).

Additionally, there are numerous large bubbles in the figures. In contrast, the MAPE values for the best-fit parameter matchup in Figure 25 remains consistently stable ranging from 0.2 to 0.25, which is clearly better than that of default parameters. Furthermore, the MAPE values are well distributed and relatively concentrated, mitigating the negative influence of outliers and rendering the results more convincing. At last, we will consider the impact of river flow and tidal phenomenon. The results of the fit, considering these new variables, are shown in Figure 26.

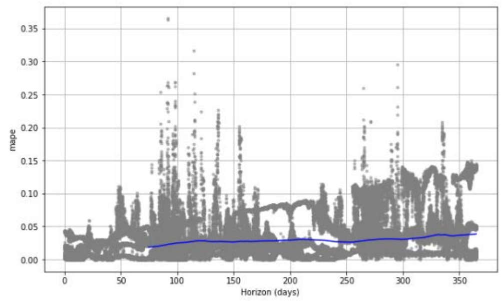

Figure 24. Figure of MAPE values with default parameter matchup

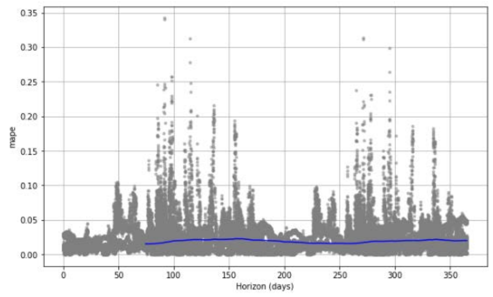

Figure 25. Figure of MAPE values with best fit parameter matchup

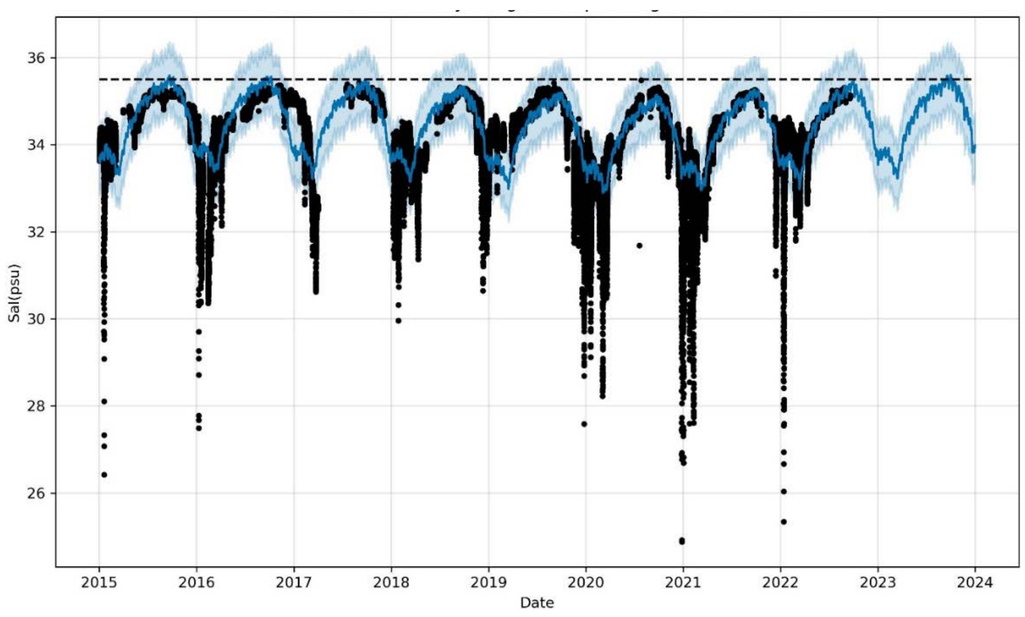

Figure 26. Results of salinity considering the new variables are presented.

In Figures 27 and 28, we can observe that after a more detailed decomposition, the monotonicity of the general trend tends to simplify, and the uncertainty width for the prediction part becomes smaller and narrower. Meanwhile, the yearly and daily trends do not show too much difference: the yearly trend increases continuously until October and then decreases rapidly due to dilution caused by the occurrence of winter floods. Regarding the daily trend, we can clearly observe two peaks and two troughs each day, in line with the law governing semi-diurnal occurrences of two high tides and two low tides. The trough is expected to be between two neighboring peaks and vice versa.

Regarding the ‘holidays’, after decomposition, the values of salinity are in staggered distribution each lunar month on the day of the spring tide, strong tidal currents eliminate the vertical layering of freshwater floating above denser seawater, leading to an increase in salinity at the surface seawater in the port. the two negative values represent that on the day of the neap tide, the current of natural water from the upstream river still dominates the tidal current, resulting in relatively lower salinity at the surface water in the port. (Note: The plus or minus on the Y-axis does not represent the absolute values of salinity; it only indicates the relative relationship.). Upon closer examination of Figure 27, holiday part, each bar of the day of tide is composed by several almost coincident bars, since the impact of both neap tides and spring tides will last for at least two days, especially there is a delay of two or three days between the phase of moon and the corresponding tide.

Concerning the lunar monthly trend, we can see that the troughs and peaks are also in staggered distribution influenced by the regularity of spring tide and neap tide. Generally, the two peaks are located on the 6th and 19th of every month, while the two troughs are located on the 10th and 24th, which is reasonable. Regarding the trend of monthly flood season and monthly normal season, the regularity is not strong. What can be only observed is that during the flood, the amplitude of salinity is significantly bigger than that of another normal season.

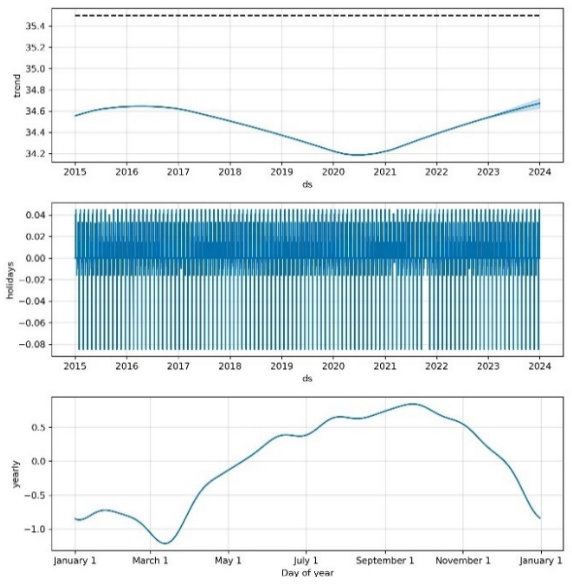

Figure 27. General trend – 'Holidays' (Dates of Spring tides and neap tides) - Yearly trend

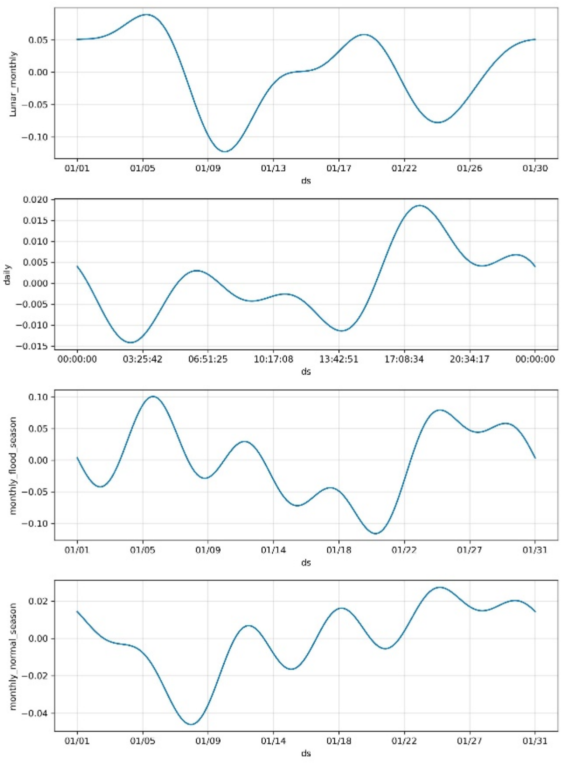

Figure 28. Lunar monthly trend - Peak/off season trend -Daily trend

In Figure 26, we can observe that both the baseline and limitation lines become more jagged after the building of the model by inputting all the dates with certain phases of the moon, which is expected (Table 2). As the tide rises, ocean water surges into the river mouth, elevating salt levels and causing an increase in estuarine water salinity. Conversely, during low tide, the receding ocean water leads to a decrease in salinity. The subtle changes caused by tidal phenomena occur four times every day. Objectively, it is impractical to pinpoint all these subtle tidal changes on specific dates. However, we can settle the spring tides and neap tides as 'holidays' to build a model that more precisely simulates the impact of tides on salinity. The jagged lines just show the exceptional changes happening during the neighboring days of spring tides and neap tides. Both the historical data, which is already known, and the prediction part will be fitted according to the 'holidays' model.

3.3 Comparison study

According to the relevant article published by Lijun Shen and Quan Qian of Shanghai University [3], GMM-VSG algorithm is suitable for the virtual sample generation of a small dataset with several features. It has been applied to the case study for predicting the wear resistance of rubber materials based on six relevant variables.

In our case, the values to be predicted are X – time (YYYY-MM-DD, HH:mm:ss), Y1 – Salinity (psu), Y2 – River flow (m^3/s), Y3 – Free surface level (m). Among these, the independent value is X – time (Day). First, we need to perform pre-processing, which involve converting the timestamps from the standard time format to the day format. Afterward, we organize the different variables into a consistent form for further calculation. Considering the relatively low computational efficiency of GMM-VSG, we have chosen to focus on the data from January 5, 2015, 07:47:00 to February 28, 2015, 13:01:00. This dataset comprises a total of 3704 items (timestamps) with 2986 items being known, and 718 items requiring generation. It is noteworthy that the missing data is concentrated within the same time interval, occurring between February 01, 2015, 00:17:00 and February 10, 2015, 23:17:00. we plan to significantly increase the dataset size to approximately tenfold, totaling 30,000 entries. Simultaneously, we intend to address and fill in the missing data during this expansion.
After inputting the data and processing of GMM-VSG for 32129 seconds, around 8.92 hours, the results are shown in the following Figures 29 and 30.

Table 2. The phases of moon (spring tides/neap tides) and the dates

Phase / Year	Full moon	First quarter moon	New moon	Last quarter moon
2015	01-05, 02-04, 03-05, 04-04, 05-04, 06-02, 07-02, 07-31, 08-29, 09-28, 10-27, 11-25, 12-25,	01-27, 02-25, 03-27, 04-26, 05-25, 06-24, 07-24, 08-22, 09-21, 10-20, 11-19, 12-18,	01-20, 02-19, 03-20, 04-18, 05-18, 06-16, 07-16, 08-14, 09-13, 10-13, 11-11, 12-11,	01-13, 02-12, 03-13, 04-12, 05-11, 06-09, 07-08, 08-07, 09-05, 10-04, 11-03, 12-03,
2016	01-24, 02-22, 03-23, 04-22, 05-21, 06-20, 07-20, 08-18, 09-16, 10-16, 11-14, 12-14,	01-17, 02-15, 03-15, 04-14, 05-13, 06-12, 07-12, 08-10, 09-09, 10-09, 11-07, 12-07,	01-10, 02-08, 03-09, 04-07, 05-06, 06-05, 07-04, 08-02, 09-01, 10-01, 10-30, 11-29, 12-29,	01-02, 02-01, 03-02, 03-31, 04-30, 05-29, 06-27, 07-27, 08-25, 09-23, 10-22, 11-21, 12-21,
2017	01-12, 02-11, 03-12, 04-11, 05-10, 06-09, 07-09, 08-07, 09-06, 10-05, 11-04, 12-03,	01-05, 02-04, 03-05, 04-03, 05-03, 06-01, 07-01, 07-30, 08-29, 09-28, 10-28, 11-26, 12-26,	01-28, 02-26, 03-28, 04-26, 05-25, 06-24, 07-23, 08-21, 09-20, 10-19, 11-18, 12-18,	01-19, 02-18, 03-20, 04-19, 05-19, 06-17, 07-16, 08-15, 09-13, 10-12, 11-10, 12-10,
2018	01-02, 01-31, 03-02, 03-31, 04-30, 05-29, 06-28, 07-27, 08-26, 09-25, 10-24, 11-23, 12-22,	01-24, 02-23, 03-24, 04-22, 05-22, 06-20, 07-19, 08-18, 09-17, 10-16, 11-15, 12-15,	01-17, 02-15, 03-17, 04-16, 05-15, 06-13, 07-13, 08-11, 09-09, 10-09, 11-07, 12-07,	01-08, 02-07, 03-09, 04-08, 05-08, 06-06, 07-06, 08-04, 09-03, 10-02, 10-31, 11-30, 12-29,
2019	01-21, 02-19, 03-21, 04-19, 05-18, 06-17, 07-16, 08-15, 09-14, 10-13, 11-12, 12-12,	01-14, 02-12, 03-14, 04-12, 05-12, 06-10, 07-09, 08-07, 09-06, 10-05, 11-04, 12-04,	01-06, 02-04, 03-06, 04-05, 05-05, 06-03, 07-02, 08-01, 08-30, 09-28, 10-28, 11-26, 12-26,	01-27, 02-26, 03-28, 04-27, 05-26, 06-25, 07-25, 08-23, 09-22, 10-21, 11-19, 12-19,
2020	01-10, 02-09, 03-09, 04-08, 05-07, 06-05, 07-05, 08-03, 09-02, 10-01, 10-31, 11-30, 12-30,	01-03, 02-02, 03-02, 04-01, 04-30, 05-30, 06-28, 07-27, 08-25, 09-24, 10-23, 11-22, 12-22,	01-24, 02-23, 03-24, 04-23, 05-22, 06-21, 07-20, 08-19, 09-17, 10-16, 11-15, 12-14,	01-17, 02-15, 03-16, 04-15, 05-14, 06-13, 07-13, 08-11, 09-10, 10-10, 11-08, 12-08,
2021	01-28, 02-27, 03-28, 04-27, 05-26, 06-24, 07-21, 08-22, 09-21, 10-20, 11-19, 12-19,	01-20, 02-19, 03-21, 04-20, 05-19, 06-18, 07-17, 08-15, 09-13, 10-13, 11-11, 12-11,	01-13, 02-11, 03-13, 04-12, 05-11, 06-10, 07-10, 08-08, 09-07, 10-06, 11-04, 12-04,	01-06, 02-04, 03-06, 04-04, 05-03, 06-02, 07-01, 07-31, 08-30, 09-29, 10-28, 11-27, 12-27,
2022	01-18, 02-16, 03-18, 04-16, 05-16, 06-14, 07-13, 08-12, 09-10, 10-09, 11-08, 12-08,	01-09, 02-08, 03-10, 04-09, 05-09, 06-07, 07-07, 08-05, 09-03, 10-03, 11-01, 11-30, 12-30,	01-02, 02-01, 03-02, 04-01, 04-30, 05-30, 06-29, 07-28, 08-27, 09-25, 10-25, 11-23, 12-23,	01-25, 02-23, 03-25, 04-23, 05-22, 06-21, 07-20, 08-19, 09-17, 10-17, 11-16, 12-16,
2023	01-07, 02-05, 03-07, 04-06, 05-05, 06-04, 07-03, 08-01, 08-31, 09-29, 10-28, 11-27, 12-27	01-28, 02-27, 03-29, 04-27, 05-27, 06-26, 07-26, 08-24, 09-22, 10-22, 11-20, 12-19	01-21, 02-20, 03-21, 04-20, 05-19, 06-18, 07-17, 08-16, 09-15, 10-14, 11-13, 12-13	01-15, 02-13, 03-15, 04-13, 05-12, 06-10, 07-10, 08-08, 09-07, 10-06, 11-05, 12-05

It can be observed that each figure contains only one line, indicating that only one feature is adequately considered, while the other two features are automatically excluded. Additionally, it appears that no virtual samples have been generated in the blank portion of the missing data.

As mentioned earlier in section 2.2 on GMM-VSG, the GMM-VSG algorithm consists of PCA and GMM components. Consequently, PCA has the capability to identify and automatically eliminate noise in the data.

Clearly, in Figure 31, it is evident that the values of Y1 – Salinity and Y3 – Free surface level exhibit minimal fluctuations compared to Y2 – River flow. This characteristic renders Y1 and Y3 as seemingly irrelevant information for predicting the data, categorized as noise [13]. This is the reason why in both figures, before and after running the algorithm, only one line is visible. Additionally, the high concentration of points in the figures might create a misunderstanding that no new virtual samples were generated in the blank part. It's because the probabilistic model of GMM is generated according to the distribution of original data, which leads to the peak of probability density p(x|k) located at the centralized part; on the contrary, the trough at the blank missing part.

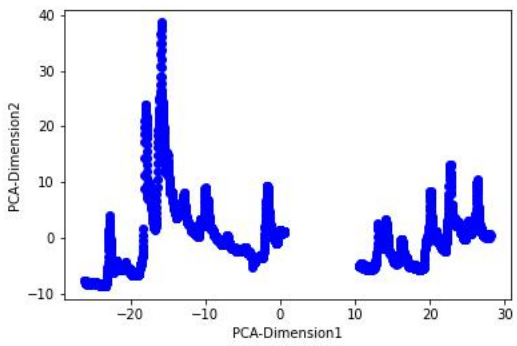

Figure 29. Original data before GMM-VSG with PCA dimension reduction

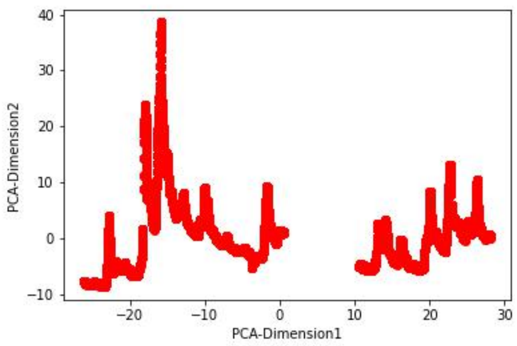

Figure 30. Virtual data generated after GMM-VSG with PCA dimension reduction.

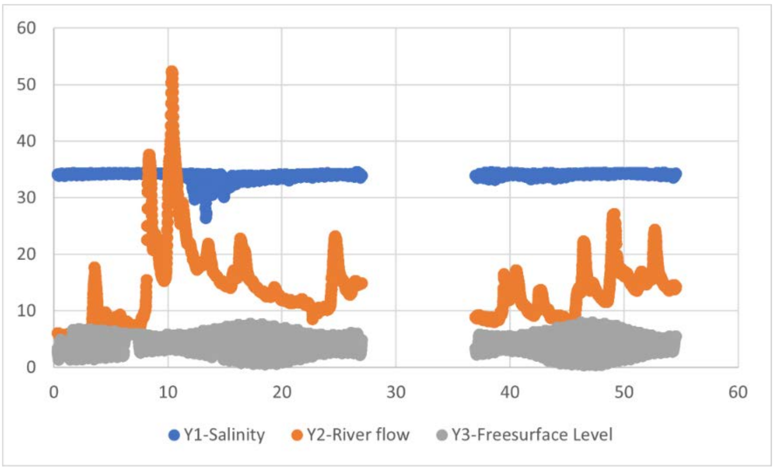

Figure 31. Results output without PCA dimension reduction

As a result, unless repeated numerous times, there won't be a virtual sample generated at the blank part. This implies that neither the missing data will be filled, nor can future data be effectively predicted [14]. Consequently, the actual result may lead to confusion, as the figures before and after processing can easily appear similar, given the prioritization of generating virtual data near the original data.

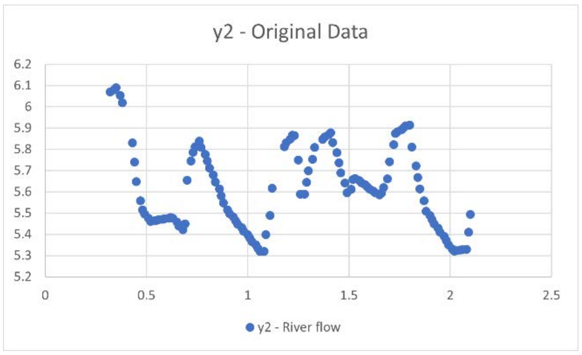

Figure 32. Detailed figure depicting the original data over the initial two days.

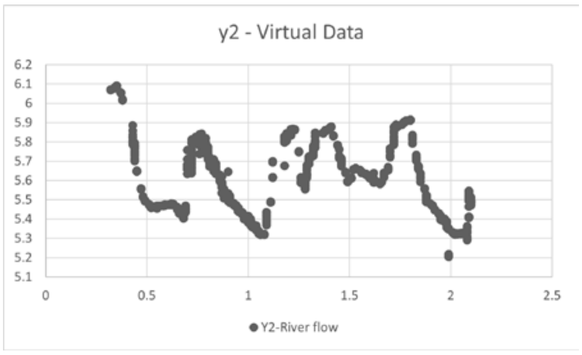

Figure 33. Detailed figure depicting the virtual data over the initial two days.

When focusing on the results from the initial two days, as depicted in Figures 32 and 33, a noticeable distinction between the original and generated data becomes evident. Clearly, the GMM-VSG algorithm can effectively contribute to filling in missing data when the blank section is not overly extensive.

4. Conclusion

After the comparison of the results between the two algorithms, we could easily find out that FB-Prophet exhibits significantly higher efficiency than GMM-VSG. FB-Prophet proves to be 50 to 60 times faster than GMM-VSG. Especially, it’s possible to adjust the accuracy for FB-Prophet, which requires different durations. Additionally, FB-Prophet can perform further statistic calculation automatically.

Table 3. Time statistics of the two algorithms

Time Statistic	FB-Prophet - Grid-Search with 16 matchups		FB-Prophet - Grid-Search with 64 matchups		GMM-VSG
Model Training	12.68 mins	12.68 mins	8.71 mins	8.71 mins	8.92 h
Cross Validation	41.25 mins	7.84 h	10.9 mins	44.26 h	/
Evaluation of Performance	64 mins		105.2 mins		
Visualization of Performance	17.76 mins		19.65 mins		
Grid - Search	5 h 35 mins		41 h mins 51 mins		

Nevertheless, the speed advantage of FB-Prophet does not imply its suitability for all types of data. In the case of small sample datasets, the performance of FB-Prophet is comparable to that of the pure linear fitting method. Conversely, GMM-VSG has a good performance when the data size is small enough to ignore the speed disadvantage. Moreover, FB-Prophet is also better at generating enough virtual data in the blank part of missing data. The prediction of future data automatically involves filling in the missing data using the Logistic/Linear method. Importantly, this process is not influenced by the distribution of a specific probability density.

In addition, the prediction and fitting of the minimum value of FB-Prophet is not always as ideal as that of the maximum value. According to the article published by researchers from Xi’an Jiaotong University, it’s possible to create a new hybrid algorithm by combining FB-prophet with LightGBM to obtain a prediction with more accurate upper and lower bounds [15]. In fact, they have successfully applied this approach to the district heating load forecasting. This could potentially serve as an intriguing avenue for exploration in the future.

The primary limitation of FB-Prophet in the context of our application is its ability to handle datasets with only one independent variable, which is time. If we need to consider other variables, it becomes necessary to establish the relationship between these additional variables and time. This enables us to introduce the effect on time by weekly seasonality, by monthly seasonality or even by customized seasonality. In cases where the extra variables are not time-series data, alternative algorithms like GMM-VSG, support vector machine (SVM), long short-term memory (LSTM), or artificial neural network (ANN) need to be considered.

For our next study, there are two main ways to enhance the FB-Prophet algorithm. One first involves expanding the input parameter features. In the current study, we utilized a limited set of parameters, but it is essential to identify additional independent parameters associated with environmental conditions.

The second approach is to integrate the algorithm with another such as Light Gradient Boosting Machine (LightGBM). LightGBM is an algorithm first proposed by Microsoft, considered as the updated version of the model Gradient Boosting Decision Tree (GBDT). By employing feature parallelization, data parallelization and voting-based parallel, the algorithm could process massive amounts of training data, encompassing both categorical features and continuous features, with reduced time and memory requirements. In contrast to LightGBM, FB-Prophet has the capability to generate multiple features simultaneously even with only one feature being processed.

Consequently, it is necessary to extract the features from FB-Prophet, including the upper and lower bounds of the confidence interval, the daily and weekly seasonalities, the trend and other relevant aspects. The subsequent step involves merging these extracted features with the original features we already have.

In the final step, we process the data by LightGBM. There’s research proving that the improvement of the performance caused by a combination of LightGBM and FB-Prophet could be as high as 31%, compared to the case of FB-Prophet model with the best feature engineering, by performance metric MAE. This approach maximizes the advantages of both algorithms and represents a direction worthy of research in the future [15].

The previously mentioned results and our conclusion, providing a global perspective on machine learning algorithm applications, would be beneficial for researchers involved in coastal event prediction. Indeed, coastal events such as sea level changes, flooding, and salinity are intricately linked to environmental, natural, and weather conditions. The traditional numerical solutions require excessive energy and computing resources.

5. References

[1] Guillou, N., Chapalain, G., Petton, S. Predicting sea surface salinity in a tidal estuary with machine learning. Oceanologia. 2022;65(2):318-332.
[2] Taylor, S. J., Letham, B. Forecasting at scale. The American Statistician. 2018;72(1):37–45.
[3] Shen, L., Qian, Q. A virtual sample generation algorithm supporting machine learning with a small-sample dataset: A case study for rubber materials. Computational Materials Science. 2022;211:111475.
[4] Reynolds D A. Gaussian mixture models. Encyclopedia of biometrics. 2009;741:659-663.
[5] Salih Hasan, B. M., Duhok Polytechnic University, Abdulazeez, A. M., Duhok Polytechnic University. A review of principal Component Analysis algorithm for dimensionality reduction. Journal of Soft Computing and Data Mining. 2021;02(01);20-30.
[6] Laird N, Lange N, Stram D. Maximum likelihood computations with repeated measures: application of the EM algorithm. Journal of the American Statistical Association,1987; 82(397):97-105.
[7] Min, A., Holzmann, H., Czado, C. Model selection strategies for identifying most relevant covariates in homoscedastic linear models. Computational Statistics & Data Analysis. 2010;54(12):3194–3211.
[8] Hastie T J. Generalized additive models. Statistical models in S. Routledge. 2017. p.249-307.
[9] Kyurkchiev N, Markov S. Sigmoid functions: some approximation and modelling aspects. LAP LAMBERT Academic Publishing, Saarbrucken.2015, 4.
[10] McCluskey, W. J., McCord, M., Davis, P. T., Haran, M., McIlhatton, D. Prediction accuracy in mass appraisal: A comparison of modern approaches. Journal of Property Research.2013;30(4):239–265.
[11] Van Ravenzwaaij, D., Cassey, P., & Brown, S. D. A simple introduction to Markov Chain Monte–Carlo sampling. Psychonomic Bulletin & Review. 2018;25(1);143–154.
[12] Jiménez Á B, Lázaro J L, Dorronsoro J R. Finding optimal model parameters by discrete grid search. Innovations in Hybrid Intelligent Systems. Berlin, Heidelberg: Springer Berlin Heidelberg.2008. p.120-127.
[13] Zhu X, Wu X. Class noise vs. attribute noise: A quantitative study. Artificial intelligence review.2004; 22: 177-210.
[14] Yang, J., Yu, X., Xie, Z.-Q., Zhang, J.-P. A novel virtual sample generation method based on Gaussian distribution. Knowledge-Based Systems. 2011;24(6):740–748.
[15] Shakeel, A., Chong, D., Wang, J. District heating load forecasting with a hybrid model based on LightGBM and FB-prophet. Journal of Cleaner Production. 2023;409:137130.