FI-SHAP: Explanation of Time Series Forecasting and Improvement of Feature Engineering Based on Boosting Algorithm

Yuyi Zhang[1], Ovanes Petrosian[1,2](✉), Jing Liu[1], Ruimin Ma[1], and Kirill Krinkin[2]

[1] Saint-Petersburg State University, 198504 St. Petersburg, Russia
st088518@student.spbu.ru, petrosian.ovanes@yandex.ru

[2] Saint Petersburg Electrotechnical University "LETI", 197376 St. Petersburg, Russia

Abstract. Boosting Algorithm (BA) is state-of-the-art in major competitions, especially in the M4 and M5 time series forecasting competitions. However, the use of BA requires tedious feature engineering work with blindness and randomness, which results in a serious waste of time. In this work, we try to guide the initial feature engineering operations in virtue of the explanation results of the SHAP technique, and meanwhile, the traditional Feature Importance (FI) method is also taken into account. Previous BA explanation works have rarely focused on forecasting, so the contribution of this work is (1) to develop a BA explanation framework-"FI-SHAP", which focuses on time series forecasting, (2) to improve the efficiency of feature engineering. At the same time, to measure explainability performance, (3) we also establish a new practical evaluation framework that attempts to remove development barriers in the field of explainable AI.

Keywords: Feature engineering · Time series forecasting · Explainable AI

1 Introduction

In the M4 [1] and M5 [2] competitions, Boosting Algorithm showed excellent time series forecasting performance, especially LightGBM, which won the championship in M5 and is a recent representative of BA. Strictly speaking, time series forecasting has only two columns: time and forecast target. The corresponding statistical models [3,4] belonging to the white box model and the LSTM [5] and RNN [6] belonging to the black box model are frequently used. However, for energy time series forecasting and other realistic forecasting requirements, there are often obvious external factors, including other time-series features and non-time series features. These factors lead to noise in the time series of the forecast target, thereby reducing forecasting performance. Therefore, ML models that adapt to multi-feature input such as Boosting Algorithm are naturally considered by a large number of researchers in time series forecasting. Whether it is XGBoost [7], LightGBM [8], or hybrid methods based on them, they have been widely developed and applied.

The feature engineering is extremely important for time series forecasting based on boosting algorithms. Time series forecasting can be summarized as an autoregressive [9] problem, therefore, the construction of lag variables is exceedingly critical [10,11]. The construction problem of the lag variables includes which features should be constructed for which lag features and the respective number of lag periods. This makes it necessary for us to know the feature importance of the model in the prediction process, so as to enrich the information of those important features, in this work, that is, to construct lag features that tend to be important features. Both the XAI [12–15] method and the Feature Importance [16–18] function can obtain the importance of features.

The development of the XAI technique used in this work is based on a "post-hoc" approach, which is based on the idea of perturbing the input data, including removing or changing features and recording the performance changes of the black-box model. Therefore, the feature contribution will be higher if its removal or change has a significant impact on the performance of the black-box model. This is why the use of these XAI methods is not limited by the variety of black-box models. The BA model is unusual from other black-box models in that it can output an explanation result - Feature Importance (FI), which is a technique widely used in feature engineering, including feature selection and extraction. In addition, we built an explanation framework that combines FI and SHAP - FI-SHAP, and finally used FI, SHAP, and FI-SHAP and other explanation methods to guide the construction of lag features, so as to improve the forecasting performance. The results show that the hybrid method - FI-SHAP can significantly improve the forecasting performance compared to the other two methods.

In Sect. 2, existing related research results are described and compared with the method we created. Section 3 explains in detail the methodology used in this work, including the description of FI-SHAP, the improvement method, and the evaluation framework. The time series forecast results, explanation results, and evaluation results of the explanation results are shown in Sect. 4. On this basis, the conclusions of this work are organized in Sect. 5.

2 Background

What feature engineering needs to achieve is to enrich the information of the data set, so that the prediction model can learn more "knowledge" and show better performance [19]. Existing feature engineering methods in time series forecasting tasks fall into two categories: exogenous and endogenous. The exogenous scheme is to add features that may affect the time series; the endogenous scheme is to enrich the information of the dataset by extracting the hidden features of the original features. In this work, only the endogenous scheme is discussed.

Feature engineering does not have a fixed execution route. It often requires practitioners to design features based on their professional knowledge, so it is extremely dependent on human experience. This is unreliable and time consuming. To alleviate this problem, a large number of automatic feature engineering methods have been developed, such as [20–24]. These mainstream frameworks use multiple feature groups contained in the original dataset to discover new relevant features, and they all focus more on classification tasks. Even though these methods can be partially used in time series forecasting, they do not reflect the temporal features. In contrast, our feature engineering method based on XAI is specially designed for time series data and can reflect temporal features.

Feature engineering methods for time series forecasting include [25–27]. They essentially add lag features on the basis of the above methods, that is, introduce autoregressive features, so as to ensure that enough time features are involved in training. In addition, feature selection is performed by constructing a large number of features (including autoregressive features) in advance, and calculating the feature importance of the model after forecasting, such as [28–30]. However, this process has a certain degree of blindness and randomness. However, our XAI-based feature engineering method is able to quantitatively calculate the order of the required lags, thereby eliminating this blindness and randomness.

3 Feature Engineering Based on XAI Method

3.1 Construction of Lag Features

For feature engineering in time series forecasting, the construction of lag features is extremely critical. Firstly, time series forecasting, theoretically, is an autoregressive task, that is, using its own historical data to predict future data, so the construction of lag features is indispensable for time series forecasting tasks. Secondly, the more lag features are not the better, and too many lag features will cause the decline of the prediction performance. This is because the autoregressive process will cause the accumulation of errors, which means that the more lag features, the more errors will accumulate. Therefore constructing a suitable number of lag features is especially critical in time series forecasting. Automatic feature engineering developed by us will focus on the construction of lag features in time series forecasting.

3.2 XAI Methods

The popularity of machine learning and deep learning has led to a wider focus on explainable artificial intelligence (XAI). Machine learning models and deep learning models are generally regarded as “black boxes” with internally unknown characteristics. Therefore, when these models are applied, it is very important to gain human’s trust, clarify the specific meaning of their errors, and the reliability of their predictions.

The Feature importance in the boosting model is an important part of feature engineering. On the one hand, it has perfect mathematical theoretical knowledge [31]; on the other hand, it is applicable to almost all tree models and is extremely convenient. Feature importance is essentially Information Gain, which is used for feature selection when the decision tree is split, and its calculation is based on

Shannon entropy. Features with larger information gain are considered important features. The calculation process of the information gain is defined as:

Expected information (Shannon entropy):

$$Info(X) = -\sum_{i=1}^{n} P(x_i) log_2 P(x_i) \tag{1}$$

Information Gain:

$$Gain = Info(X) - \sum_{i=1}^{n} \frac{|X^i|}{X} Info(X^i) \tag{2}$$

X is a random variable, P is the probability of all cases.

As mentioned above, in order to construct the right amount of lag features, we must know the importance of the features in advance. In addition to the FI function that comes with the boosting model, XAI is also a way worth trying.

The explanation method achieves the purpose of explanation by constructing a simpler and easier-to-understand model and allowing it to continuously approximate the model that needs to be explained. This kind of method is called Post-Hoc, which is different from Intrinsic, which integrates interpretable functions into the black-box model. The former rarely relies on the architecture of the black-box model and can be widely used in models that have been trained. Among them, Generalized Additive Models [35], Bayes Rule List [34], and Neural Additive Model [36] belong to Intrinsic, and their operation is closely related to the black box model. SHAP [37,39,40] and LIME [38] are two exceedingly popular model-agnostic (Post-Hoc) explanations. In this work, we only consider SHAP due to its complete code repository and rigorous mathematical theory.

The creation of SHAP is based on the Shapley value, which treats each feature as a "player" to build a system where "single-player (single feature)" and "alliance (feature combination)" participate in the "game (black-box model)". SHAP actually attributes the output value to the shapely value of each feature. In other words, it calculates the Shapley value of each feature and based on this, measures the impact of the feature on the final output value. For linear models with independent features, the sum of the contributions of all the features of the sample is equal to the predicted value minus the average predicted value, but this is obviously not true for Boosting algorithms. Therefore, for the features of the Boost model, the Shapley value needs to be calculated for all possible feature combinations (including different orders) and then weighted and summed, which is defined as:

$$\phi(val) = \sum_{S \subseteq \{x_1 \cdots x_p\} \setminus \{x_j\}} \frac{|S|!(p-|S|-1)!}{p!} (val(S \cup \{x_j\}) - val(S)) \tag{3}$$

where S is a subset of the features used in the model, x is the vector of feature values of the sample to be explained, p is the number of features, and $val(S)$ refers to the model output value under the feature combination S. We can quantitatively construct lag variables based on the explanation results of XAI, enabling automatic feature engineering. The overall process is shown in Fig. 1.

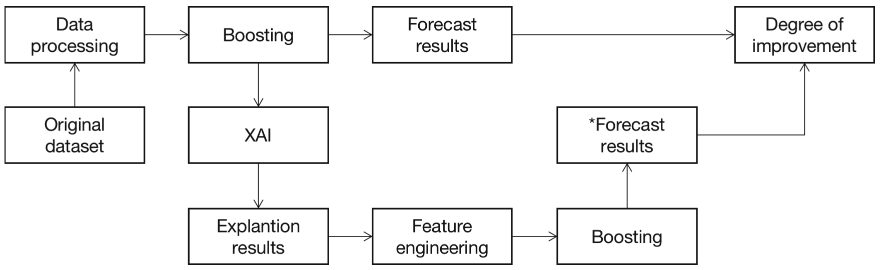


Fig. 1. The process of feature engineering that XAI uses for time series forecasting. In theory, the XAI part can be replaced with other methods that can output feature importance (contribution). The interpretation result is the feature importance (contribution), which is also the basis for feature engineering processing in our method framework.

4 Hybrid Explanation Method: FI-SHAP

4.1 Description

Actually, since a lot of resources are used in feature engineering when machine learning is performing tasks, the explanation of Boosting Algorithm has been considered as early as its development. Feature Importance (FI) [13], as an integrated attribute of Boosting Algorithm itself, has been applied to feature selection for a long time. However, as an explanation of Boosting Algorithm, FI has many shortcomings that cannot be ignored, including:

- FI cannot reflect whether the influence of features on the forecast results is positive or negative.
- FI cannot reflect the interrelationship between features and target variables. In essence, this means that the interpretation is not ideal.

Different from the popular Model-Agnostic Approximations in the Explainable AI field, FI is an attribute of Boosting Algorithm itself. Therefore, FI should be paid attention to in the framework of constructing Boosting Algorithm explanation, even if it has extremely poor performance for user-oriented explanation.

A hybrid explanation method combining FI and XAI was constructed, with the purpose of trying to combine feature engineering and Explainable AI to improve the forecast performance of Boosting Algorithm in time series forecasting. Especially for energy and other time series that are susceptible to external interference. Taking the two Explainable AI methods mentioned above: SHAP and LIME as examples, the new hybrid explanation method is defined as:

$$\varphi(val) = mean\phi(val)_j \times \frac{FI(x_j)}{\sum_{i=1}^{p} FI(x_i)} \tag{4}$$

$\phi(val)_j$ represents the contribution of j feature to the forecast result under the explanation framework of SHAP.

FI-SHAP combines traditional feature engineering with the emerging Explainable AI. On the one hand, it enhances XAI's specificity for Boosting Algorithm, and on the other hand, it provides users with more comprehensive explanation results.

4.2 Improvement of Forecasting Performance

As mentioned above, the core of our automatic feature engineering is to construct a suitable amount of lag features to achieve the purpose of feature engineering improvement. Initially, we used the features of the original data to forecast directly using the boosting model, i.e. without feature engineering. Explain the model after the model is trained to obtain the importance of the features, and calculate the weight of each feature, which is defined by the following formula:

$$Weight = \frac{F_j}{\sum_{j=1}^{n} F_j} \tag{5}$$

F_j represents the explanation value of each feature.

For different explanation methods, the representation of F_j is also different. For SHAP, F is $\phi(val)$ (Eq. 3); for FI, F is Gain (Eq. 2), and for our FI-SHAP, F is $\varphi(val)$ (Eq. 4).

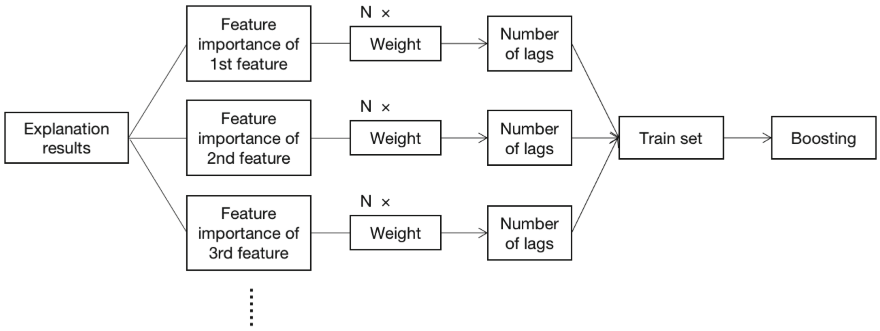

Explanation results
Feature importance of 1st feature
Feature importance of 2nd feature
Feature importance of 3rd feature
N ×
Weight
Number of lags
Train set
Boosting

Fig. 2. The process of feature engineering works by explanation results. N is the total number of features, which can be set by the user according to actual needs.

Calculate the weight of each continuous feature through the explanation results of SHAP and Feature Importance (FI) [32,33] output, and set the total number of features to be constructed, so that lag features can be constructed according to the weight (Fig. 2). After obtaining the weights, we also need to set the total number N of features to be constructed, and construct lag features for the continuous features in the original dataset according to the result of N*weight. However, in fact, we do not know the exact value of N, so the automatic feature engineering we provide takes an iterative approach to construct lag features. For example, when the user sets N to 50, our framework will start from 1 to 50, and select the best performing result. Therefore, theoretically, if the computing power of the computer allows, the value of N can be set larger. However, based on actual performance, we recommend setting the value of N not to exceed 200. See this link for details: https://github.com/Zhangyuyi-0825/FI-SHAP.

4.3 Evaluation Framework

In order to effectively measure the improvement effect of these explanation methods on feature engineering, we use the "average lag explanation" as the baseline of the experiment. In the "average lag explanation", the explained value of all features is artificially set to 1 (or any value, as long as the explained value of all features is guaranteed to be equal), that is, the weight of each feature is equal. So the effect it has is that each feature needs to build the same number of lag features. For example, when N = 30 and the original number of features is 5, the "average lag explanation" is to construct 6 lag features for each feature. Shap might assign 20 lag features to the first feature, 15 features to the second, and no lag features to features with low explanatory values. Finally, compare the effect of these methods on the improvement of forecasting performance, which also represents the improvement effect of these methods on feature engineering.

5 Simulation Results

5.1 Description of Data Sets

The energy time series data used in this work comes from Kaggle, which was collected from two solar power plants located in India with a time period of 34 d (every 15 min). The data set consists of two parts, the first part is the power generation data set, which is generated by the inverter, including direct current, alternating current, daily yield, and total yield. The second part is the data collected by the sensors, including temperature and solar radiation. The inverters are named *sourcekey*, and each power plant has 22 inverters, for a total of 44 inverters. These inverters all generate power generation data at the same time, resulting in a dataset with a total of 136,476 rows and 7 columns (within 34 d).

We adopted a multi-threaded processing scheme, that is, according to different inverter ids (*sourcekey*), the data table is divided into 44 data sets with about 3000 rows and 7 columns (within 34 d). We use both raw boosting models and boosting models with feature engineering on these datasets to highlight the best performing explanation methods.

5.2 Forecasting Results

We use XGBoost with LightGBM to separately make forecasting on 44 small datasets from two solar power plants. The forecasting results are shown in Fig. 3.

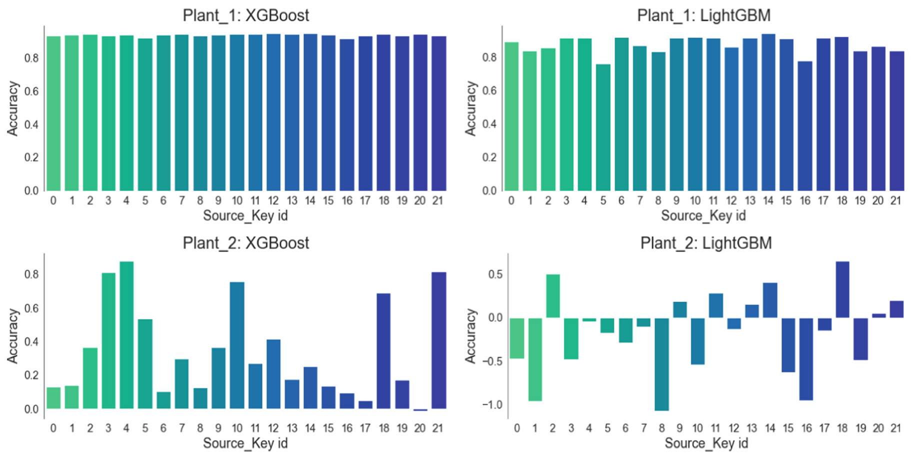


Fig. 3. Raw forecasting results (no feature engineering). The results show that the data set of power plant 1 is inherently of high quality, while the data set of power plant 2 contains a large number of anomalies, and even LightGBM almost fails for it.

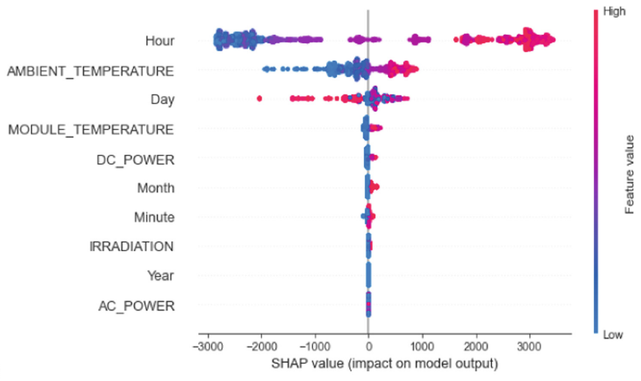
It should be noted that LightGBM has significantly outperformed XGBoost in time series competitions in recent years, because the scale of the datasets used in the competition is large. The basis of LightGBM is still XGBoost, and the addition of algorithms such as histogram enables LightGBM to train faster than XGBoost and to ensure accuracy. However, on small datasets, the advantages of LightGBM disappear accordingly. This is also the reason why LightGBM's performance is poor compared to XGBoost in this work.

From such these data sets, we test the improvement performance of these explanation methods using the power plant 1 dataset and the repair performance of these explanation methods using the power plant 2 dataset.

5.3 Explanation Results

Compared with Feature Importance, the explanation results (Fig. 4) output by SHAP can show both the positive and negative effects of the feature, and can provide users with more comprehensive explanation information. Here, we only show the explanation results of a data set produced by an inverter as an example.

It can be seen from the results that the impact of "Hour" is both positive and negative, and the overall effect is relatively balanced. However, the impact of "AMBIENT TEMPERATURE" on the forecasting results is biased negatively, that is, within a certain range, the rise in temperature will have a certain positive impact on the power generation, and other features are also explained according to the same logic. For Feature Importance, such an interpretation effect cannot be achieved, and FI can only show the ranking of feature importance, as shown in Fig. 5.



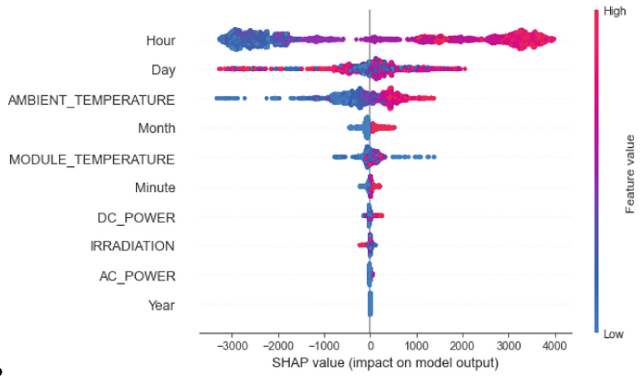


Fig. 4. SHAP explanation (Plant 1, Source Key id: 0): a: XGBoost; b: LightGBM

5.4 Improvement Results

We create different kinds of lag features based on different explanation results, thereby enriching feature engineering to improve the performance of forecasting models. The improvement effect of the explanation method is tested by the data set of Power Plant 1, and the repair effect of the explanation method is tested by the data set of Power Plant 2.

The final improvement results are shown in Table 1 and Table 2, Table 1 is the upgrade of Power Plant 1, and Table 2 is the repair of Power Plant 2.

According to the results, in the dataset of higher quality Power Plant 1, all explanation methods show improved effect. On the whole, the improvement effect of FI-SHAP is the best, especially in LightGBM. The second is SHAP, and FI has a general effect on improving high-quality data. In the low-quality Power Plant 2 dataset, synthetically, almost all explanation methods have insignificant repair effects. On the one hand, it means that only the construction of autoregressive features is not enough to achieve good performance. On the other hand, the results also show that for small datasets, the adaptability of LightGBM is not as good as that of XGBoost.

Table 1. Forecast quality (R^2) of power plants 1

Source key	0	1	2	3	4	5	6	7	8	9	10
XGBoost	0.934	0.940	0.942	0.934	0.938	0.920	0.938	0.941	0.935	0.940	0.941
Avg.	0.934	0.941	0.947	0.941	0.939	0.923	0.942	0.946	0.940	0.944	0.944
FI	0.938	0.953	**0.953**	0.942	**0.944**	0.927	0.938	0.951	0.951	0.942	0.944
SHAP	**0.940**	0.952	0.952	**0.944**	**0.944**	0.932	**0.943**	**0.952**	0.950	**0.943**	**0.945**
FI-SHAP	**0.940**	**0.954**	0.951	**0.944**	0.942	**0.933**	0.941	**0.952**	**0.953**	**0.943**	**0.945**
Source key	11	12	13	14	15	16	17	18	19	20	21
XGBoost	0.941	0.945	0.940	0.945	0.936	0.916	0.934	0.942	0.934	0.944	0.932
Avg.	0.944	0.948	0.943	0.946	0.941	0.920	0.942	0.944	0.937	0.948	0.941
FI	**0.946**	0.953	0.943	0.948	0.942	0.926	0.938	0.946	0.947	0.953	0.949
SHAP	0.945	0.953	0.943	0.948	0.942	0.934	**0.939**	0.946	0.950	0.953	0.950
FI-SHAP	0.944	**0.954**	0.943	0.948	0.942	**0.935**	**0.939**	0.946	**0.951**	0.953	**0.951**
Source key	0	1	2	3	4	5	6	7	8	9	10
LightGBM	0.892	0.836	0.856	0.914	0.914	0.760	0.919	0.867	0.831	0.913	0.920
Avg.	0.911	0.885	0.889	0.930	0.934	0.764	0.933	0.888	0.877	0.932	0.934
FI	0.928	0.889	0.893	0.940	0.936	0.774	**0.936**	0.895	0.879	0.935	0.940
SHAP	0.925	0.889	0.899	**0.944**	**0.939**	0.774	0.935	0.893	0.878	0.941	0.941
FI-SHAP	**0.931**	**0.892**	**0.900**	**0.944**	0.938	**0.859**	0.934	**0.904**	**0.888**	**0.942**	**0.945**
Source key	11	12	13	14	15	16	17	18	19	20	21
LightGBM	0.912	0.857	0.914	0.939	0.909	0.779	0.914	0.923	0.838	0.864	0.838
Avg.	0.924	0.890	0.922	0.948	0.929	0.793	0.929	0.935	0.889	0.893	0.882
FI	0.935	0.891	0.936	0.947	0.926	0.827	0.937	0.940	0.881	0.887	0.880
SHAP	0.937	0.894	**0.942**	**0.953**	**0.932**	0.903	0.939	**0.944**	0.883	0.890	0.888
FI-SHAP	**0.939**	**0.904**	0.938	**0.953**	**0.932**	**0.915**	**0.942**	0.943	**0.890**	**0.894**	**0.893**

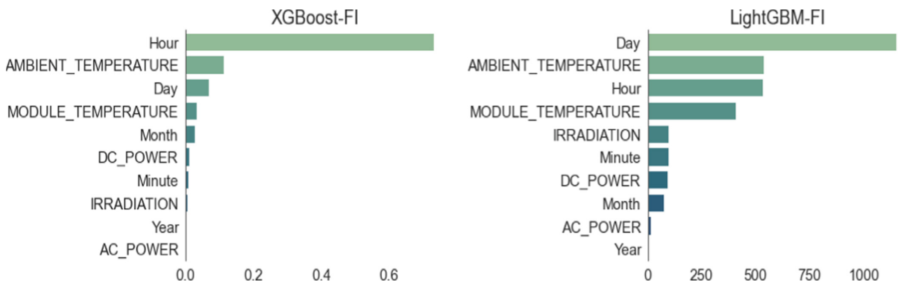


Fig. 5. Feature importance (Plant 1, Source Key id: 0)

The performance improvement brought by more lag features is still not negligible, because when we are doing actual feature engineering, there are still many ways to participate, as described earlier. However, in this work, we only focus on the construction of lag features, in order to study time series forecasting tasks more professionally. In the repair of low-quality data, it can be clearly seen that the repair effect of FI-SHAP is more obvious for XGBoost, and the repair effect of SHAP is more obvious for LightGBM.

Table 2. Forecast quality (R^2) of power plants 2

Source key	0	1	2	3	4	5	6	7	8	9	10
XGBoost	0.132	0.141	0.365	0.810	0.878	0.535	0.102	0.297	0.124	0.367	0.755
Avg.	0.494	0.159	0.372	0.869	0.904	**0.556**	0.153	0.327	0.174	0.449	0.821
FI	**0.569**	0.209	**0.465**	0.934	0.924	0.544	0.192	0.375	0.227	**0.554**	0.889
SHAP	0.568	**0.309**	0.384	0.946	**0.943**	0.538	**0.216**	0.394	0.362	0.398	0.890
FI-SHAP	0.558	0.252	**0.497**	**0.951**	0.937	0.536	0.212	**0.433**	**0.375**	0.401	**0.896**
Source key	11	12	13	14	15	16	17	18	19	20	21
XGBoost	0.271	0.415	0.176	0.251	0.134	0.093	0.051	0.689	0.173	−0.011	0.813
Avg.	**0.318**	0.598	0.321	0.311	0.347	0.281	0.324	0.788	0.216	0.061	0.822
FI	0.301	0.608	0.456	**0.357**	0.449	0.343	0.360	0.938	0.247	0.337	0.826
SHAP	0.303	0.656	0.462	0.316	0.445	0.305	0.352	0.947	**0.258**	0.073	0.827
FI-SHAP	0.302	**0.682**	**0.541**	0.355	**0.465**	**0.471**	**0.368**	**0.948**	0.249	**0.296**	**0.828**
Source key	0	1	2	3	4	5	6	7	8	9	10
LightGBM	−0.472	−0.962	0.502	−0.486	−0.045	−0.176	−0.292	−0.107	−1.073	0.188	−0.546
Avg.	−0.042	−0.314	0.576	0.654	0.405	0.080	0.004	0.318	−0.586	**0.398**	0.191
FI	**0.278**	−0.061	0.570	0.714	0.884	**0.205**	**0.169**	**0.396**	−0.179	0.367	0.500
SHAP	0.248	**0.206**	**0.597**	**0.770**	**0.903**	0.047	**0.022**	0.324	−0.008	0.309	0.615
FI-SHAP	0.253	0.074	0.585	0.688	0.889	0.135	0.117	0.356	0.022	0.324	**0.670**
Source key	11	12	13	14	15	16	17	18	19	20	21
LightGBM	0.288	−0.134	0.155	0.407	−0.628	−0.955	−0.148	0.653	−0.487	0.049	0.202
Avg.	**0.327**	0.110	0.213	0.442	−0.051	−0.210	0.013	0.708	−0.288	0.193	0.258
FI	0.317	0.168	0.177	0.429	0.127	0.012	0.162	0.729	−0.224	0.177	0.325
SHAP	0.307	0.107	**0.503**	**0.477**	0.000	**0.026**	**0.190**	**0.901**	−0.290	**0.224**	0.313
FI-SHAP	0.290	**0.188**	0.403	0.475	**0.142**	−0.025	0.161	0.884	−**0.222**	0.188	**0.345**

6 Conclusion

In this work, we confirm that the construction of lagged features can improve the performance of forecasting models in time series forecasting tasks, but for lower quality data, lagged features are not enough. Simulation results also show that our constructed automatic lag feature method: FI-SHAP's improvement effect is the most stable in higher quality data. In lower quality data, FI-SHAP still has a significant repairing effect on the forecasting of XGBoost. Synthetically, XGBboost outperforms LightGBM for small datasets and adapts better to poorer quality data. Most of the existing feature engineering methods focus on classification tasks, while feature engineering methods for time series forecasting also pay little attention to lag feature construction. In this work, reasonable lag feature construction is proven to be critical.

Acknowledgment. This work was supported by the Ministry of Science and Higher Education of the Russian Federation by the Agreement №075-15-2020-933 dated 13.11.2020 on the provision of a grant in the form of subsidies from the federal budget for the implementation of state support for the establishment and development of the world-class scientific center "Pavlov center" "Integrative physiology for medicine, high-tech healthcare, and stress-resilience" technologies.

References

1. Makridakis, S., Spiliotis, E., Assimakopoulos, V.: The M4 competition: results, findings, conclusion and way forward. Int. J. Forecast. **34**(4), 802–808 (2018)
2. Makridakis, S., Spiliotis, E., Assimakopoulos, V.: The M5 competition: background, organization, and implementation. Int. J. Forecast. (2021)
3. Seber, G.A.F., Lee, A.J.: Linear Regression Analysis. Wiley (2012)
4. Fattah, J., Ezzine, L., Aman, Z., et al.: Forecasting of demand using ARIMA model. Int. J. Eng. Bus. Manag. **10**, 1847979018808673 (2018)
5. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
6. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014)
7. Chen, T., Guestrin, C.: Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794 (2016)
8. Ke, G., Meng, Q., Finley, T., et al.: Lightgbm: a highly efficient gradient boosting decision tree. Adv. Neural Inf. Process. Syst. **30**, 3146–3154 (2017)
9. Dickey, D.A., Fuller, W.A.: Distribution of the estimators for autoregressive time series with a unit root. J. Am. Statist. Assoc. **74**(366a), 427–431 (1979)
10. Gao, R., Duru, O., Yuen, K.F.: High-dimensional lag structure optimization of fuzzy time series. Exp. Syst. Appl. **173**, 114698 (2021)
11. ZhiYuan, C., Khoa, L.D.V., Boon, L.S.: A hybrid model of differential evolution with neural network on lag time selection for agricultural price time series forecasting/. In: International Visual Informatics Conference, pp. 155–167. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70010-6_15
12. Gunning, D., Aha, D.: DARPA's explainable artificial intelligence (XAI) program. AI Magazine **40**(2), 44–58 (2019)
13. Adadi, A., Berrada, M.: Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). IEEE Access **6**, 52138–52160 (2018)
14. Tjoa, E., Guan, C.: A survey on explainable artificial intelligence (XAI): toward medical XAI. IEEE Trans. Neural Netw. Learn. Syst. (2020)
15. Das, A., Rad, P.: Opportunities and challenges in explainable artificial intelligence (XAI): a survey. arXiv preprint arXiv:2006.11371 (2020)
16. Zhou, Z., Hooker, G.: Unbiased measurement of feature importance in tree-based methods. ACM Trans. Knowl. Discov. Data **15**(2), 1–21 (2021)
17. Lundberg, S.M., Lee, S.I.: Consistent feature attribution for tree ensembles. arXiv preprint arXiv:1706.06060 (2017)
18. Xie, Z., Fang, G.Q., Huang, Y.H., et al.: FIST: a feature-importance sampling and tree-based method for automatic design flow parameter tuning. In: 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 19–25. IEEE (2020)
19. Guyon, I., Elisseeff, A.: An introduction to feature extraction. In: Feature Extraction, pp. 1–25. Springer, Heidelberg (2006). https://doi.org/10.1007/978-3-540-35488-8_1
20. Kanter, J.M., Veeramachaneni, K.: Deep feature synthesis: towards automating data science endeavors. In: 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 1–10. IEEE (2015)
21. Katz, G., Shin, E.C.R., Song, D.: Explorekit: automatic feature generation and selection. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 979–984. IEEE (2016)

22. Kaul, A., Maheshwary, S., Pudi, V.: Autolearn-automated feature generation and selection. In: 2017 IEEE International Conference on Data Mining (ICDM), pp. 217–226. IEEE (2017)
23. Khurana, U., Turaga, D., Samulowitz, H., et al.: Cognito: automated feature engineering for supervised learning. In: 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), pp. 1304–1307. IEEE (2016)
24. Lam, H.T., Thiebaut, J.M., Sinn, M., et al.: One button machine for automating feature engineering in relational databases. arXiv preprint arXiv:1706.00327 (2017)
25. Cerqueira, V., Moniz, N., Soares, C.: Vest: Automatic feature engineering for forecasting. Mach. Learn. 1–23 (2021)
26. Li, L., Ou, Y., Wu, Y., et al.: Research on feature engineering for time series data mining. In: 2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC), pp. 431–435. IEEE (2018)
27. Zdravevski, E., Lameski, P., Mingov, R., et al.: Robust histogram-based feature engineering of time series data. In: 2015 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 381–388. IEEE (2015)
28. Selvam, S.K., Rajendran, C.: tofee-tree: automatic feature engineering framework for modeling trend-cycle in time series forecasting. Neural Comput. Appl. 1–20 (2021)
29. Punmiya, R., Choe, S.: Energy theft detection using gradient boosting theft detector with feature engineering-based preprocessing. IEEE Trans. Smart Grid **10**(2), 2326–2329 (2019)
30. Hu, Y., An, W., Subramanian, R., Zhao, N., Gu, Y., Wu, W.: Faster clinical time series classification with filter based feature engineering tree boosting methods. In: Shaban-Nejad, A., Michalowski, M., Buckeridge, D.L. (eds.) Explainable AI in Healthcare and Medicine. SCI, vol. 914, pp. 247–260. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-53352-6_23
31. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**(3), 379–423 (1948)
32. Pan, F., Converse, T., Ahn, D., et al.: Feature selection for ranking using boosted trees. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, pp. 2025–2028 (2009)
33. Chen, T., Wang, X., Chu, Y., et al.: T4SE-XGB: interpretable sequence-based prediction of type IV secreted effectors using eXtreme gradient boosting algorithm. Front. Microbiol. **11** (2020)
34. Letham, B., Rudin, C., McCormick, T.H., et al.: Interpretable classifiers using rules and Bayesian analysis: building a better stroke prediction model. Ann. Appl. Statist. **9**(3), 1350–1371 (2015)
35. Caruana, R., Lou, Y., Gehrke, J., et al.: Intelligible models for healthcare: predicting pneumonia risk and hospital 30-day readmission. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1721–1730 (2015)
36. Agarwal, R., Frosst, N., Zhang, X., et al.: Neural additive models: interpretable machine learning with neural nets. arXiv preprint arXiv:2004.13912 (2020)
37. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 4768–4777 (2017)
38. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should i trust you? Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144 (2016)

39. Zou, J., Xu, F., Petrosian, O.: Explainable AI: using Shapley value to explain the anomaly detection system based on machine learning approaches. Manag. Process. Sustain. **7**(1), 355–360 (2020)
40. Lundberg, S.M., Erion, G.G., Lee, S.I.: Consistent individualized feature attribution for tree ensembles. arXiv preprint arXiv:1802.03888 (2018)