Time Series Forecasting of Energy Behavior in Solar Panel Prosumers

Contents

Executive Summary

This report delves into the “Enefit - Predicting Energy Behavior of Prosumers” dataset, aiming to uncover the key drivers of electricity production and consumption behavior of solar panel prosumers and develop efficient prediction models. To achieve this goal, the dataset was thoroughly analyzed using Python and the CRISP-DM data mining method was used to provide a structured process and clear guidance for the research (Wirth & Hipp, 2000). During the research process, multiple models were tested, including linear statistical models (ARIMA and SARIMAX), machine learning models (XGBoost), ANN (LSTM), and multiple fusion models (XGBoost+LSTM) to find the best prediction method.

The results show that in terms of electricity production by solar panel prosumers, the intensity of solar radiation received on the ground is the main influencing factor, while cloud cover has a smaller impact on production; in terms of electricity consumption, the electricity consumption in the previous hour is the key factor, while the impact of electricity price is relatively small. Finally, the performance of the prediction model showed that XGBoost had the highest accuracy in predicting both electricity production and consumption.

Based on the final predictive model, feasible, data-driven strategies can be provided to energy companies, such as optimizing solar resource scheduling, adjusting grid load distribution, and formulating more flexible electricity price mechanisms, thereby achieving efficient management of energy supply and demand and maximizing profits.

Introduction

In recent years, with the global transition to clean energy, distributed photovoltaic (PV) systems have expanded rapidly and become an important pillar in the field of renewable energy. According to the "Fit for 55" plan proposed by the European Union, by 2030, renewable energy is expected to account for 45% of total final energy consumption (Qiu, 2023), fully demonstrating its core position in the future energy structure. In addition, by the end of 2024, the global photovoltaic installed capacity has reached more than 22TW (IEA-PVPS, 2025). The rapid growth in this field further highlights the potential and market demand of photovoltaic technology. However, photovoltaic power generation is highly dependent on weather conditions and consumption patterns, and its volatility and uncertainty pose severe challenges to grid planning and coordination.

In this context, this study uses the "Enefit - Predicting Energy Behavior of Producers and Consumers" dataset provided by the Kaggle platform, focusing on the behavioral

analysis and prediction of distributed energy prosumers, in order to provide data-based decision support to major participants in the energy market and help achieve sustainable energy development goals.

Relevant research

In order to meet the current demand for efficient prediction of the behavior of distributed photovoltaic system prosumers, Poplawski et al. (2023) proposed a prediction framework based on machine learning, which used LASSO and random forest (RF) models to predict the energy balance of photovoltaic power generation, and introduced weather data and recursive multi-step strategies, thereby significantly improving the prediction accuracy. The following will analyze the inspiration and shortcomings of this study in combination with key points and limitations.

Key points

- **Weather data and recursive strategy improve prediction accuracy**
  Poplawski's study effectively improved the prediction ability of the model by introducing weather data (such as average temperature and maximum light intensity) as exogenous variables and combining it with a recursive multi-step prediction strategy. This provides important inspiration for how to improve the prediction model in this study.

- **Feature Optimization and Multi-step Forecast Design**
  In Poplawski's study, the LASSO model screened out the most critical features through regularization methods, reduced redundant interference, and thus improved the model performance. In addition, the comparative analysis of direct multi-step prediction and recursive multi-step prediction strategies provides a basis for the selection of methods in the multi-step prediction task of this study.

Limitations

- **The model is difficult to capture complex dynamic characteristics**
  LASSO and random forests perform well in dealing with linear features and some nonlinear features, but they are insufficient in modeling complex dynamic characteristics and long-term dependencies in time series (Roth, 2024). This study attempts to make up for this shortcoming by introducing XGBoost and LSTM models.

- **Limitations of time dynamics and feature interactions**
  LASSO and random forests have limited ability to capture time dynamics and feature interactions, so it is difficult to fully model the complexity in time series.

To avoid this problem, this study explored hybrid methods (a combination of LSTM and XGBoost) to try to improve the robustness and predictive performance of the model.

Data exploration

Data Processing

- **Data Source**
  The dataset used in this study comes from the "Enefit - Predict Energy Behavior of Prosumers" competition on the Kaggle platform. It contains multiple CSV files, covering the hourly electricity production and consumption of prosumers (train.csv), electricity prices (electricity_prices.csv), historical meteorological data (historical_weather.csv), and weather forecasts (forecast_weather.csv). These structured data provide target variables and external explanatory variables for building a robust prediction model (The images and URL of these original datasets can be seen in Appendix 1).

- **Data Integration**
  By using the datetime field as the primary key, the tables are left-joined to complete the data integration of each table. In the exploration phase, the data is aggregated by month to observe trends; in the modeling phase, the hourly granularity is retained to ensure that each record has complete production, consumption, meteorological and electricity price information to build a standard time series data framework.

Data Cleaning

- **Missing value processing**
  Different missing value processing strategies are used for different types of variables:
  - **Core variables (electricity production and consumption):** linear interpolation is used to maintain the continuity of the time series.
  - **Auxiliary features (meteorology, electricity prices, etc.):** median imputation is used in combination, and the selection is flexible depending on the degree of missingness.

  This strategy not only retains the trend of the time series, but also avoids the introduction of abnormal fluctuations.

- **Noise reduction**

  In order to reduce the short-term volatility in meteorological data (such as temperature, dew point, solar radiation, etc.), "Time-windowed mean smoothing" is applied to smooth them. Taking temperature as an example, the Time Series Line Chart (Figure 1) shows the changes in the data range of temperature data before and after noise reduction. It can be seen that after noise reduction, its standard deviation is reduced by about 20%, and the value range is reduced by more than 89%, which greatly improves the modelability of the data. In addition, some redundant or weakly correlated variables are deleted to simplify the feature space and improve modeling efficiency.

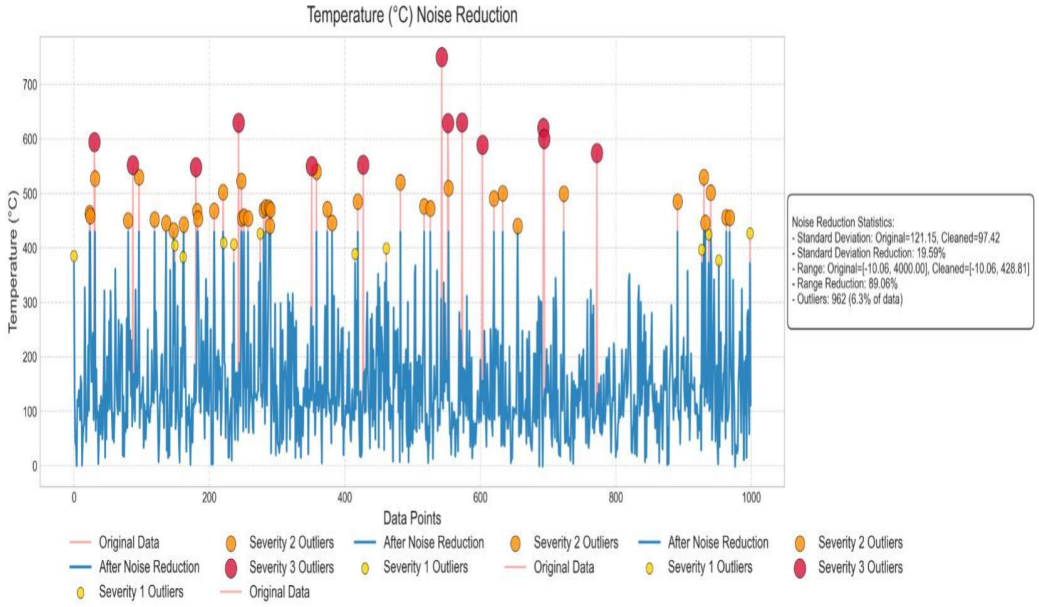


Figure 1: Temperature variable noise reduction before and after

- **Identification and processing of outliers**

  The “Interquartile range method (IQR)” is used in combination with domain knowledge to identify outliers, and different strategies are adopted according to the severity:

  - Mild outliers: truncated to 1.5 times the IQR range;
  - Moderate outliers: truncated to 2 times the IQR range;
  - Severe outliers: replaced with the median;
  - For core variables such as production and consumption, extreme values are retained to retain the true fluctuation trend and only marked.

  The following electricity price box plot (Figure 2) shows the effect of outlier identification and the processing of the above methods.

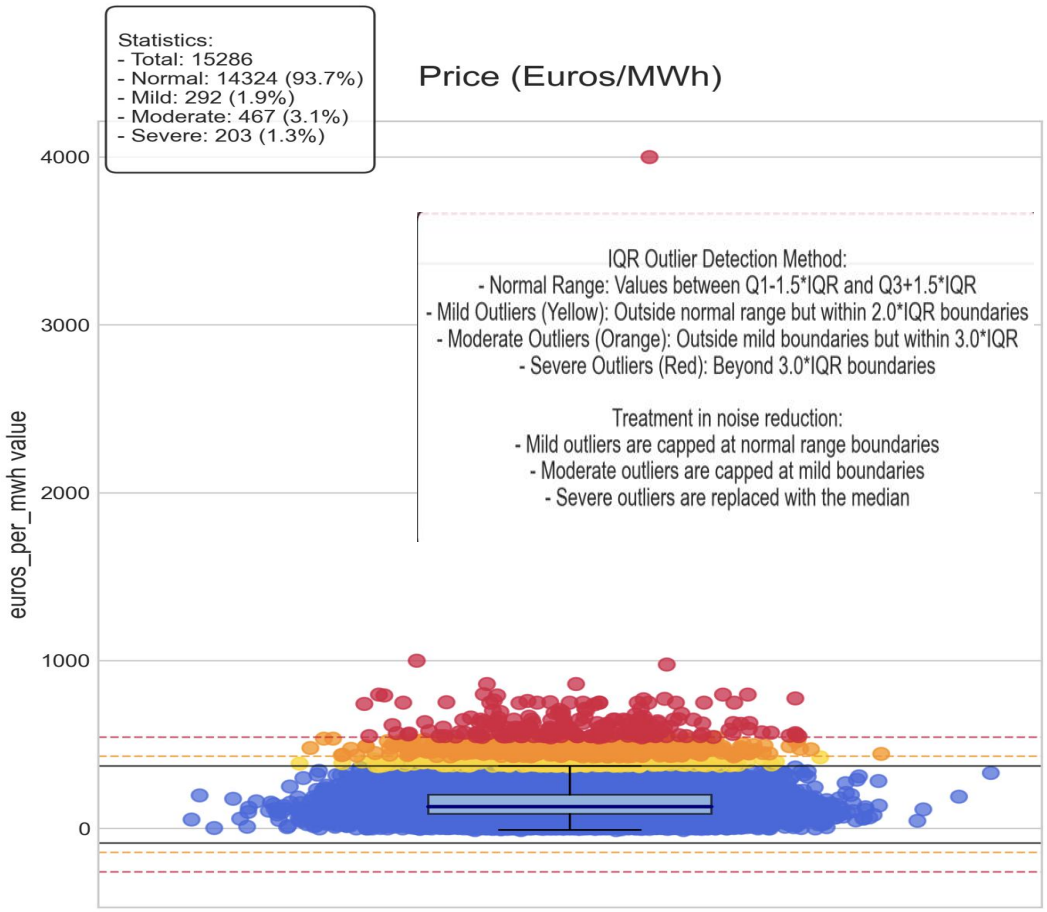


Figure 2: Box diagram for detecting electricity prices

Data Transformation

- **Standardization**

  Due to the different dimensions of each feature (such as temperature in °C and radiation in W/m²), RobustScaler is used in this study to standardize all numerical variables to resist the influence of outliers and maintain the data center trend. Taking power production data as an example, according to the following two line graphs (Figure 3), it can be seen that the distribution of standardized power production data is more concentrated and smoother than that after logarithmic transformation, showing the characteristics of normal distribution.

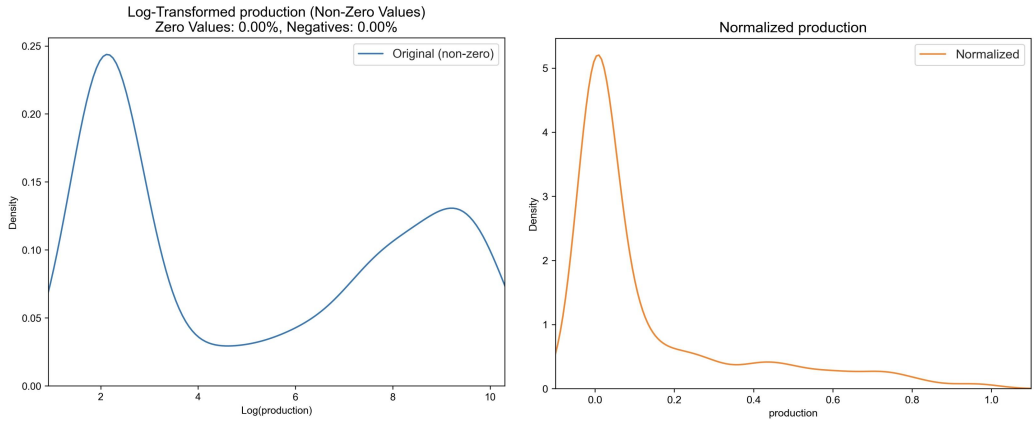


Figure 3: Density plots of electrical quantity variables before and after standardization

- **Feature Construction**

  In order to enhance the modeling ability of time characteristics and weather effects, the following derivative variables are constructed (These variables are explained in more detail in Appendix 2):

  Lag feature:
  lag_prod_1h: production in the previous hour;
  lag_cons_1h: consumption in the previous hour.

  Meteorological derivative features:
  dewpoint_dep: the difference between temperature and dew point temperature, reflecting humidity changes;
  diffuse_ratio: the ratio of diffuse radiation to total radiation;
  is_rainy: whether it rains (binary variable).

  These variables can extract new features from weather data and effectively capture short-term dependencies in time series, thereby improving the model's ability to perceive behavioral patterns.

Feature Extraction

- **Feature Importance Evaluation**

  In order to improve the prediction performance of the model and reduce redundant features, this study first used XGBoost and Permutation Importance methods to evaluate the importance of features. Specifically, Permutation Importance measures the contribution of a single feature to the prediction result by scrambling the value of the feature and observing the change in model

performance, thereby providing a more robust feature influence ranking (Altmann et al., 2010).

After completing the feature importance analysis, two feature screening strategies were designed and compared:

- Top 20 important features method: based on the feature importance ranking, the top 20 features are selected;
- Cumulative contribution 90% method: all feature sets with cumulative importance reaching 90% are selected.

From the performance of different screening strategies in the following figure (Figure 4), it can be seen that the second strategy performs better in MAE and RMSE and has higher stability, so this strategy is used for feature screening.

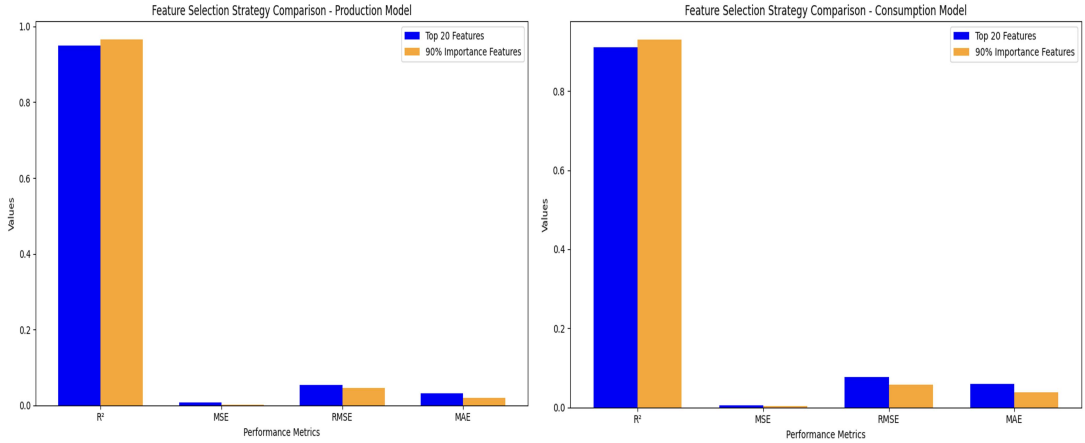


Figure 4: The performance of two variable selection strategies (production model and consumption model)

- **Feature Selection**

  By using the cumulative contribution 90% method to screen the feature importance, feature subsets for predicting power consumption and power production were constructed, and on this basis, the corresponding correlation coefficient heat maps (Figure 5 and Figure 6) were drawn to further analyze the linear relationship between the selected features and the target variable.

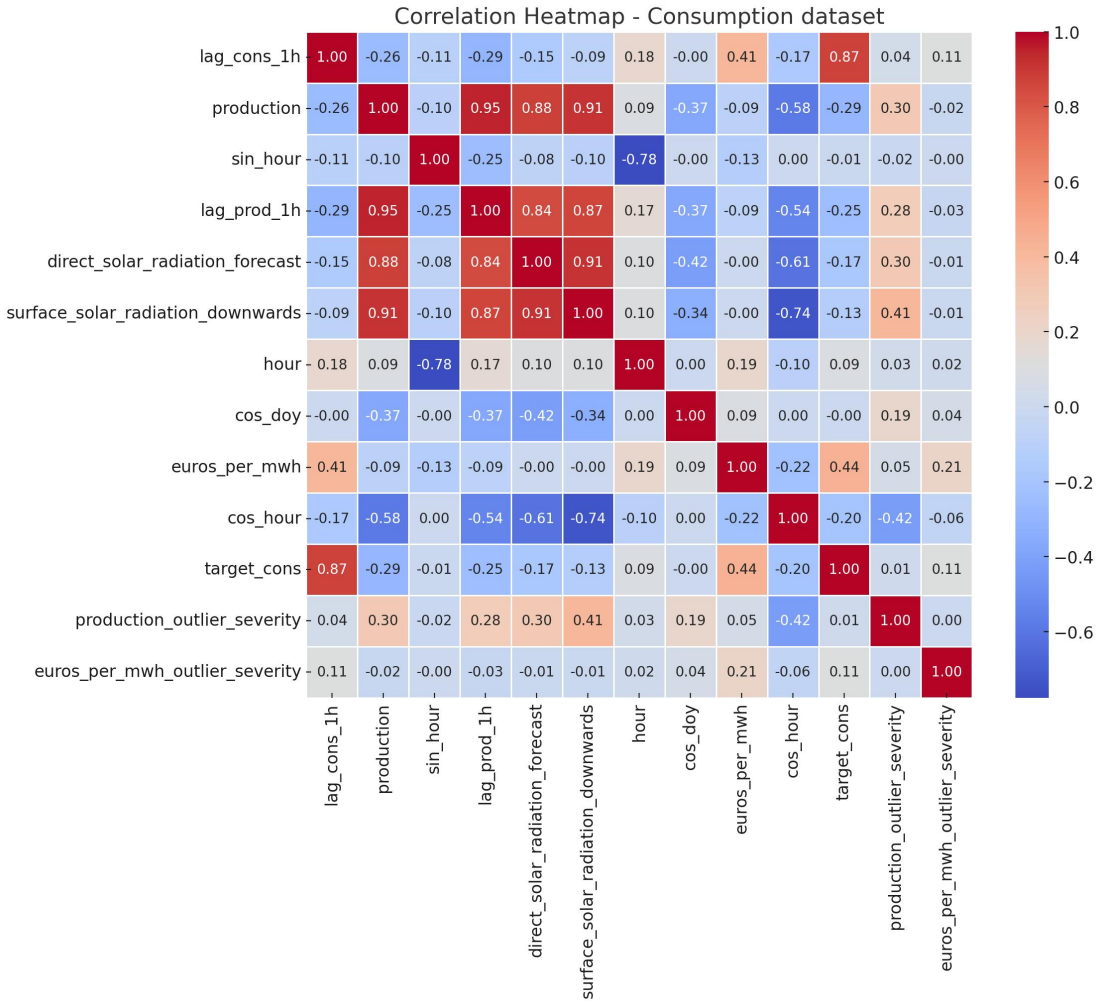


Figure 5

Figure 5 reveals that there is a strong positive correlation between the consumption and production in the previous hour and the target predicted power consumption. This indicates that these features significantly affect power consumption. In addition, features related to solar radiation have a moderate positive correlation with the target variable. On the other hand, features such as electricity price and time of day have a weak correlation with the target variable.

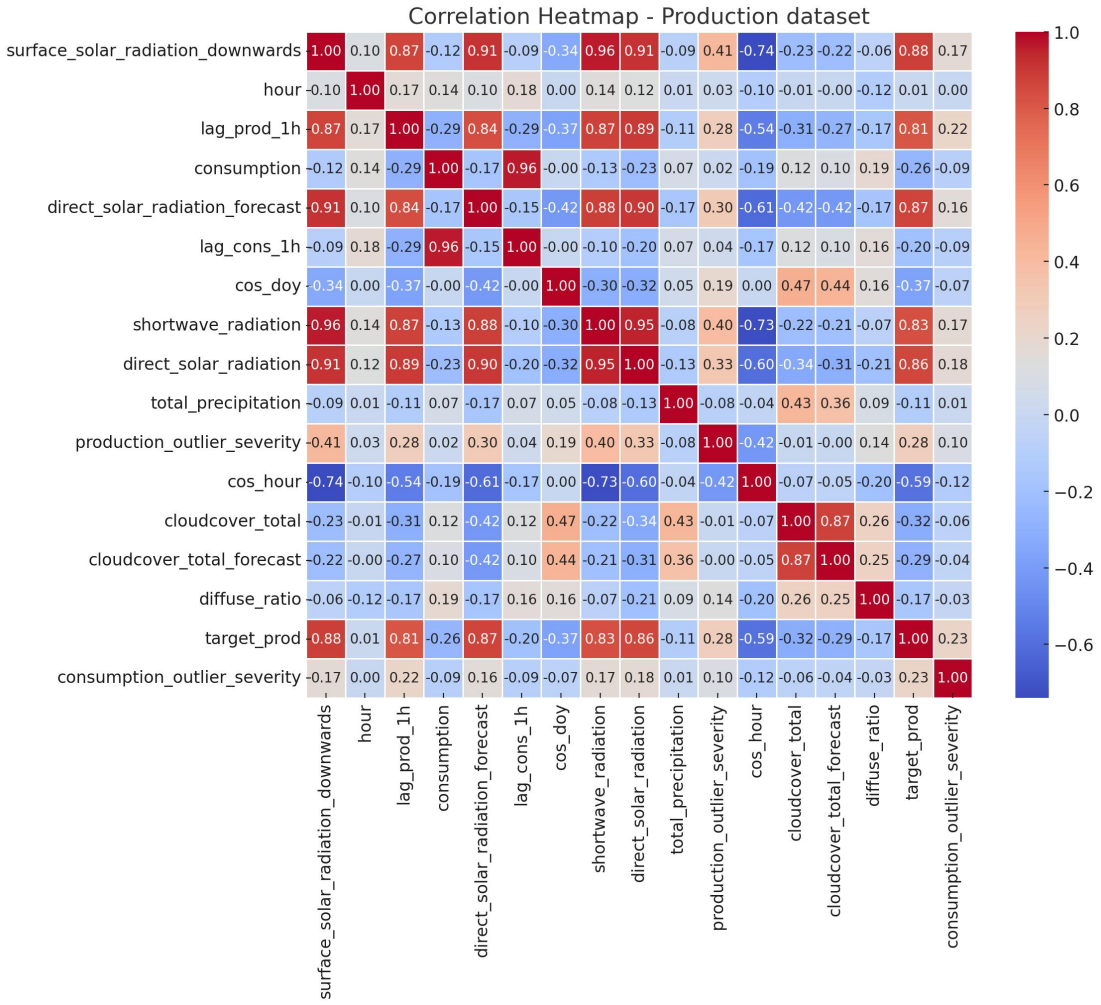


Figure 6

Figure 6 reveals that there is a strong positive correlation between the power production in the previous hour and the surface solar radiation and the target predicted power production. This indicates that these features significantly affect the power production of the photovoltaic system. In addition, shortwave radiation and direct solar radiation also have a moderate positive correlation with the target variable. On the other hand, cloud cover and cosine time features have a weak correlation with the target variable, indicating that these features have limited direct impact on power production.

In general, the consumption forecasting model mainly relies on lag variables and time series cycle characteristics, while the production forecasting model is highly dependent on meteorological factors and solar radiation related variables, which also reflects the difference in the impact mechanism of user behavior and natural environment (The specific features selected by the two subsets can be seen in Appendix 3)。

Modelling

ARIMA and SARIMA models (baseline models)

In order to establish a performance benchmark for energy production and consumption forecasting, this study first used the ARIMA and SARIMA models.

- **Model Introduction**
  These two models play an important role in time series analysis, especially for capturing trends and cyclical characteristics (Wang, Li & Lim, 2019). At the same time, as can be seen from the following figures (Figure 7 and Figure 8), after the ADF Test, the series becomes more stable through the first-order difference processing, which fully meets their modeling requirements.

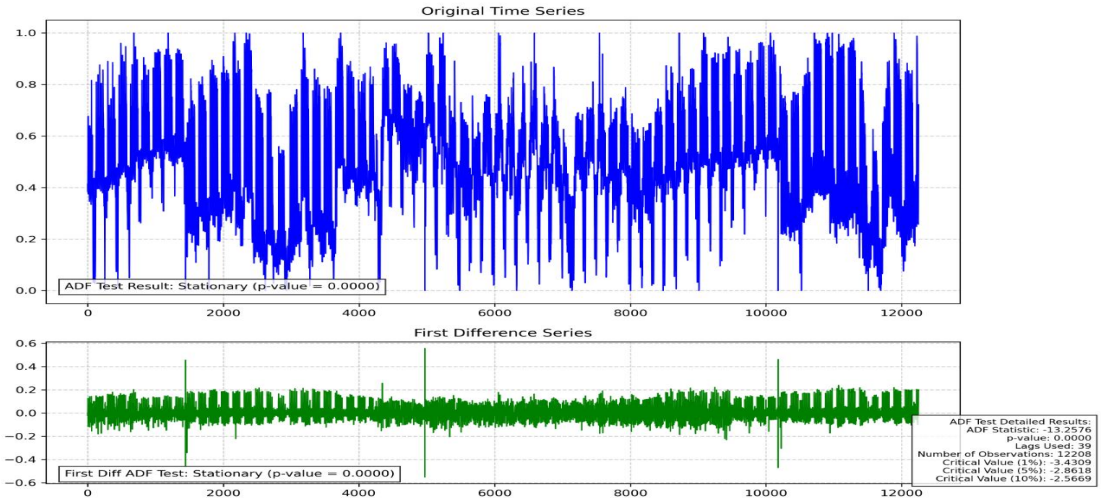


Figure 7: ADF Test and Time Series Visualization of consumption

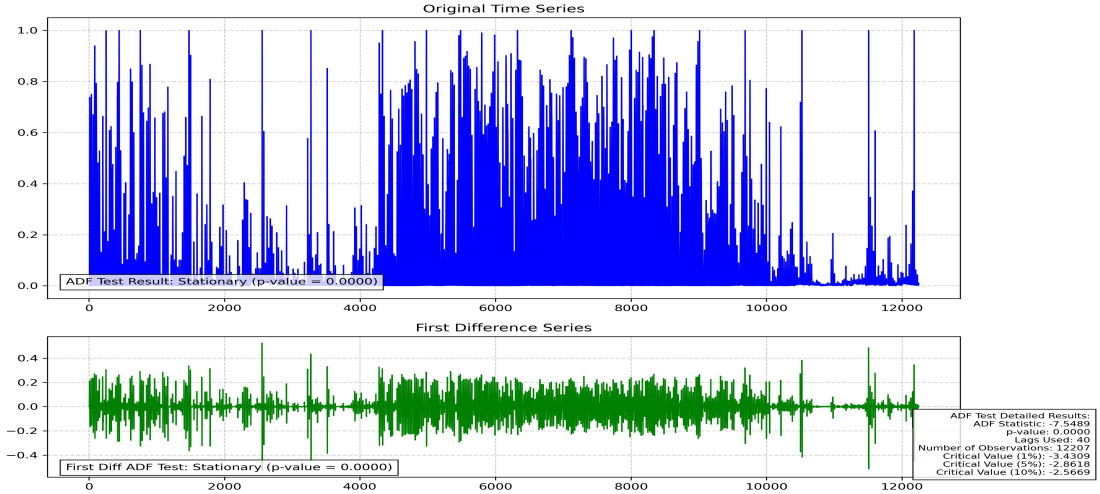


Figure 8: ADF Test and Time Series Visualization of consumption

- ARIMA (Autoregressive Integrated Moving Average): Applicable to time series data without significant seasonal characteristics, capable of capturing trend changes.

- SARIMA (Seasonal ARIMA): Adds seasonal parameters to the ARIMA model, which can effectively model daily 24-hour periodic fluctuations.

- **Model performance**

  From the forecast results (Figure 9), ARIMA and SARIMA have certain advantages in capturing the trend and seasonal characteristics of energy production and consumption, but at the same time, it can be observed that the deviation between the actual value and the predicted value is large, especially in the time period with significant nonlinear relationship, the forecast error is more obvious.

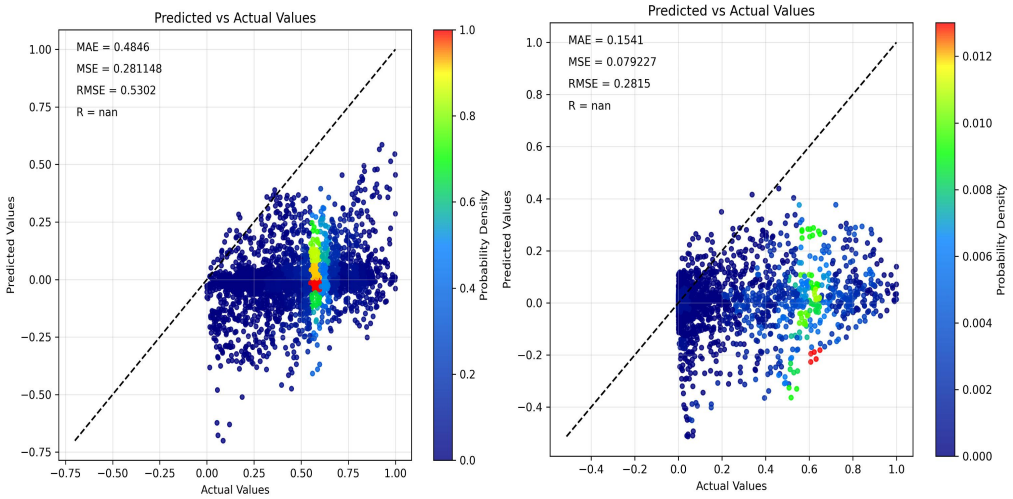


Figure 9: SARIMAX Prediction vs Actual Consumption & Production

XGBoost and LSTM models

To make up for the shortcomings of ARIMA and SARIMA models, this study introduced XGBoost and LSTM models.

- **Model introduction**

XGBoost

- Principle: A machine learning model based on gradient boosting, which is good at handling complex nonlinear features and feature interactions (Chen & Guestrin, 2016).
- Applicability: In energy consumption and production forecasting, XGBoost can efficiently capture the complex relationship between variables such as weather and electricity consumption.

LSTM

- Principle: A recurrent neural network that captures long-term and short-term dependencies in time series through a gating mechanism (Staudemeyer & Morris, 2019).
- Applicability: LSTM can handle time dependencies in energy production and consumption, such as the impact of the behavior of the previous hour on the current moment (The configuration process of LSTM in the experiment can be seen in Appendix 4).

These two models can effectively solve the limitations of ARIMA and SARIMA in nonlinear modeling and time dependency processing.

- **Model performance**

XGBoost: It excels in nonlinear modeling, can capture complex feature interactions, and outperforms the baseline model in both consumption and production prediction (Figure 10).

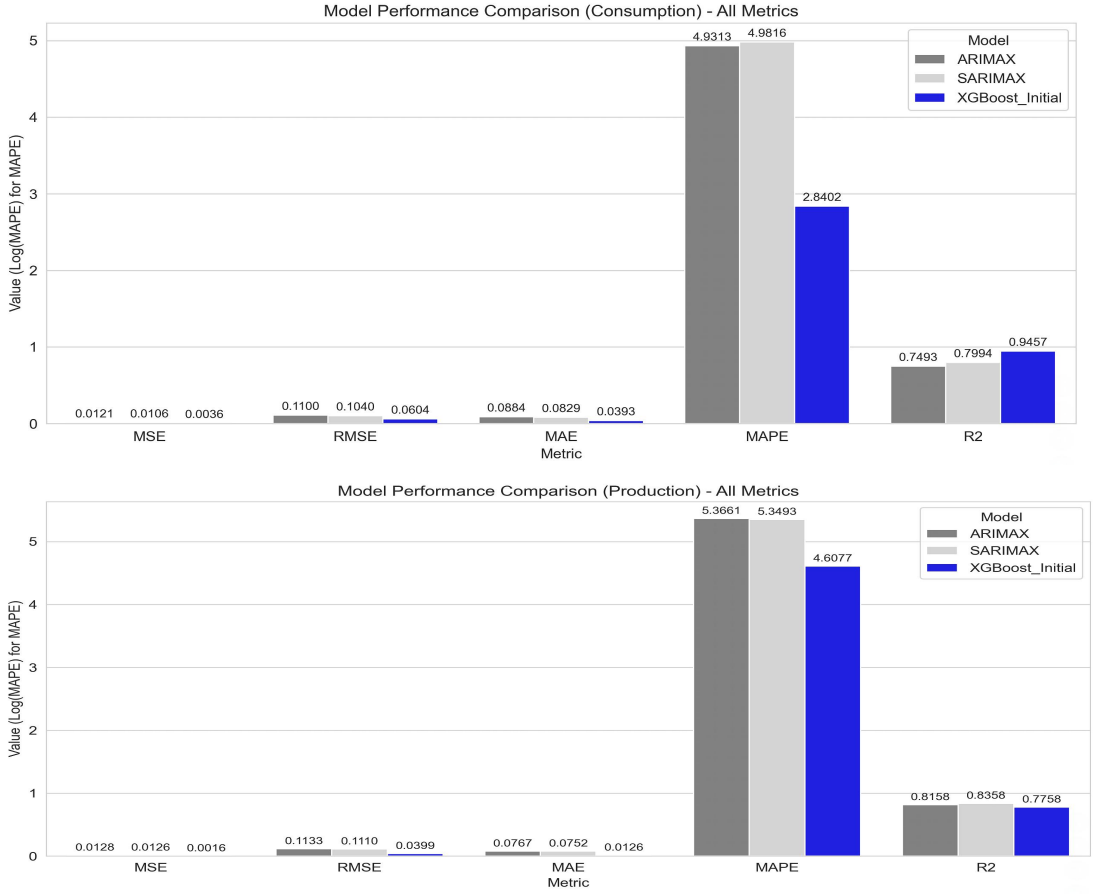


Figure 10: XGBoost Model Performance Comparison (Consumption & Production)

LSTM: has an advantage in time-dependent modeling, especially in capturing dynamic changes in energy production and consumption (Figure 11).

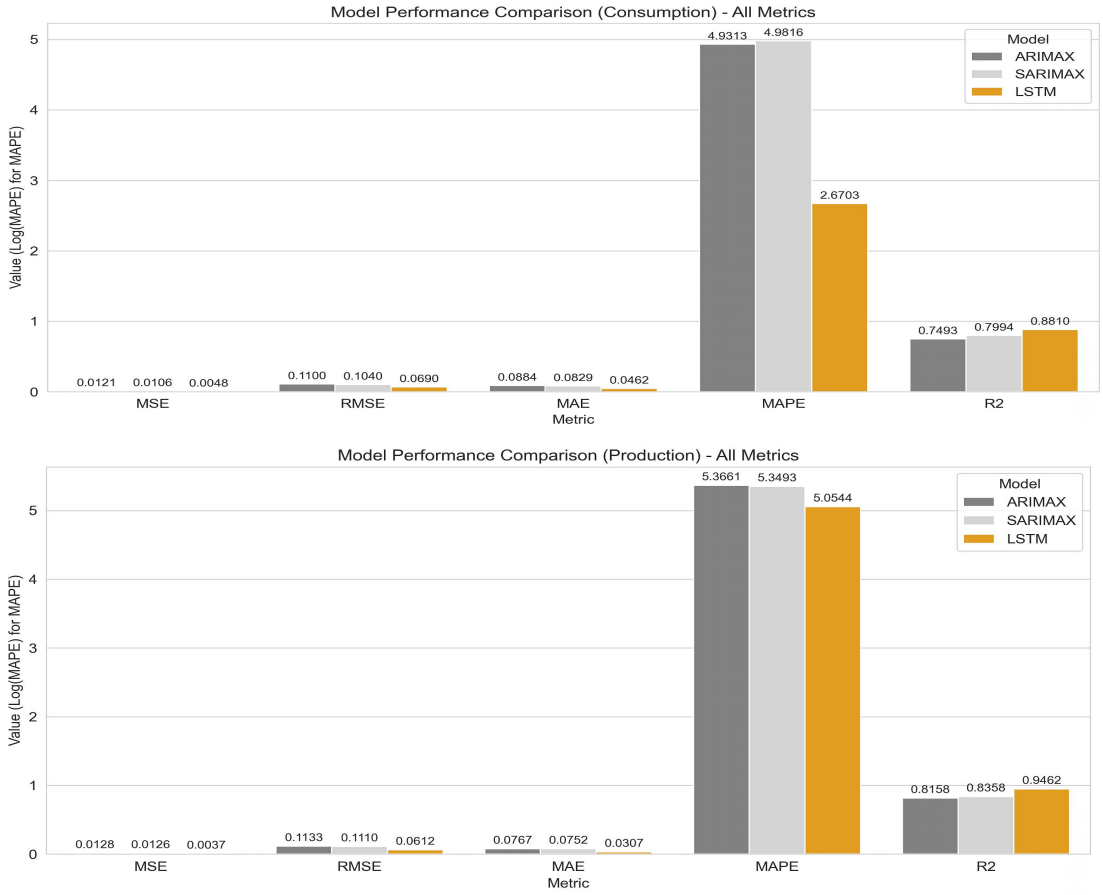


Figure 11: LSTM Model Performance Comparison (Consumption & Production)

Preliminary results show that both XGBoost and LSTM significantly outperform the baseline model, especially in key performance indicators such as RMSE, MAE, and R^2.

Parameter optimization of XGBoost and LSTM models

- **Parameter tuning process**

  In order to further improve the performance of the model, this study optimized the hyperparameters of XGBoost and LSTM:

  XGBoost parameter tuning:

  - Key parameters: number of trees (n_estimators), maximum tree depth (max_depth), learning rate (learning_rate), etc.

  - Optimization method: Combine grid search and random search to gradually select the best performance parameter combination.

  LSTM parameter tuning:

- Key parameters: number of network layers (Layers), number of hidden units (Units), Dropout rate, learning rate, etc.
- Optimization method: Use Bayesian optimization and cross-validation to find the optimal parameter configuration.

- **Parameter adjustment results**

After adjusting the parameters, the performance of XGBoost and LSTM has been significantly improved:

XGBoost: It is more accurate in nonlinear feature modeling, significantly reducing RMSE and MAE, and improving R^2 (Figure 12).

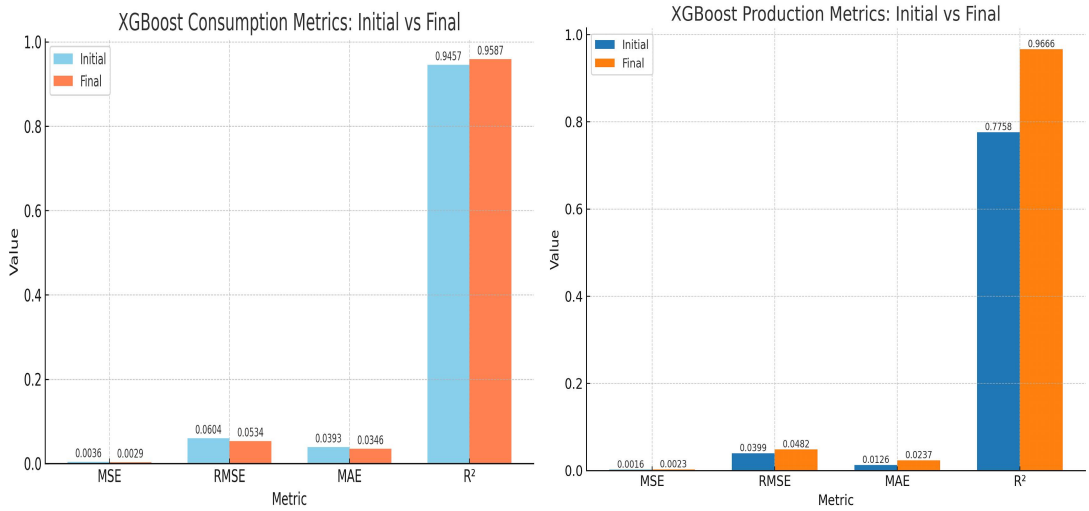


Figure 12: XGBoost Metrics Before vs After Turning (Consumption & Production)

LSTM: By optimizing the Dropout rate and the number of hidden units, the prediction results after parameter adjustment are much better than the initial model (Figure 13).

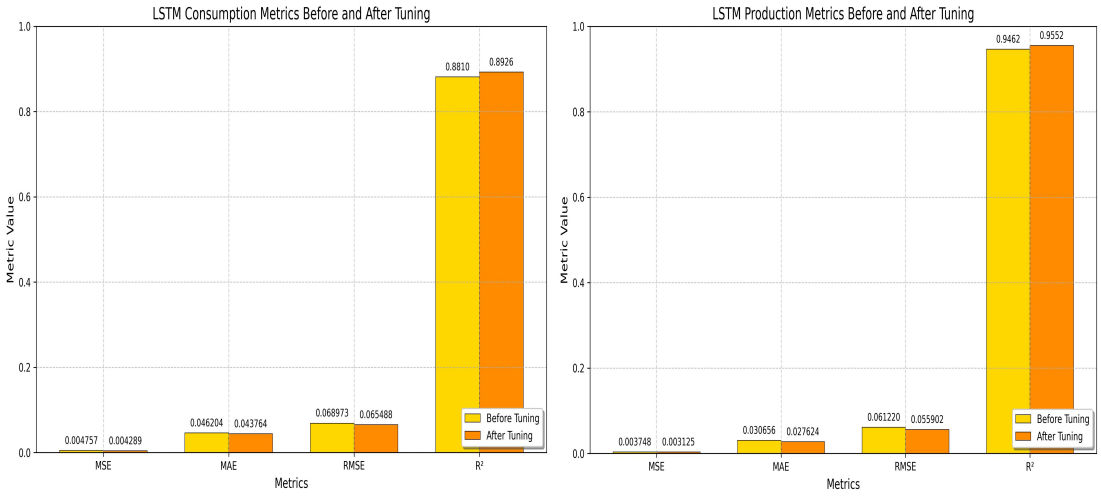


Figure 13: LSTM Metrics Before vs After Turning (Consumption & Production)

In summary, the prediction results of the optimized XGBoost and LSTM models both show the significant ability of the models in prediction, and the trend of the predicted values and the actual values on the test set is highly consistent, especially in capturing periodic fluctuations (The corresponding predicted line graph can be seen in Appendix 5).

Model fusion

- **Model fusion method**

  In order to try to obtain better prediction results,this study used three different fusion methods to combine the XGBoost model with the LSTM model:

  1. Exponential 0.6 weighted fusion: Based on a linear combination of fixed weights (0.6), the prediction results of XGBoost and LSTM are fused.
  2. Automatic optimization weighted fusion: The fusion weights of XGBoost and LSTM are dynamically adjusted through the automatic optimization algorithm to minimize the error.
  3. Deep learning fusion: By building a simple deep neural network, the prediction results of XGBoost and LSTM are used as input features to further learn and optimize the final output.

- **Model performance**

  From the performance of the three fusion models, they show different advantages in different performance indicators:

  - Exponential 0.6 weighted fusion: simple and efficient, stable performance in most scenarios.
  - Automatic optimization weighted fusion: Dynamically adjust the weights, and the performance is slightly improved compared with exponential weighted fusion.
  - Deep learning fusion: It can capture complex feature interactions and perform better in some cases, but it has higher computational complexity.

Results interpretation

- **Electricity consumption prediction**

  Figure 14 shows the performance indicators of the XGBoost model, LSTM model and three fusion models on the electricity consumption prediction test set. It can be observed from the figure:

- The XGBoost model outperforms other models in all evaluation indicators, showing lower errors and higher prediction accuracy.
- Although the fusion model has improved in some indicators, the overall performance is still slightly inferior to the single XGBoost model.

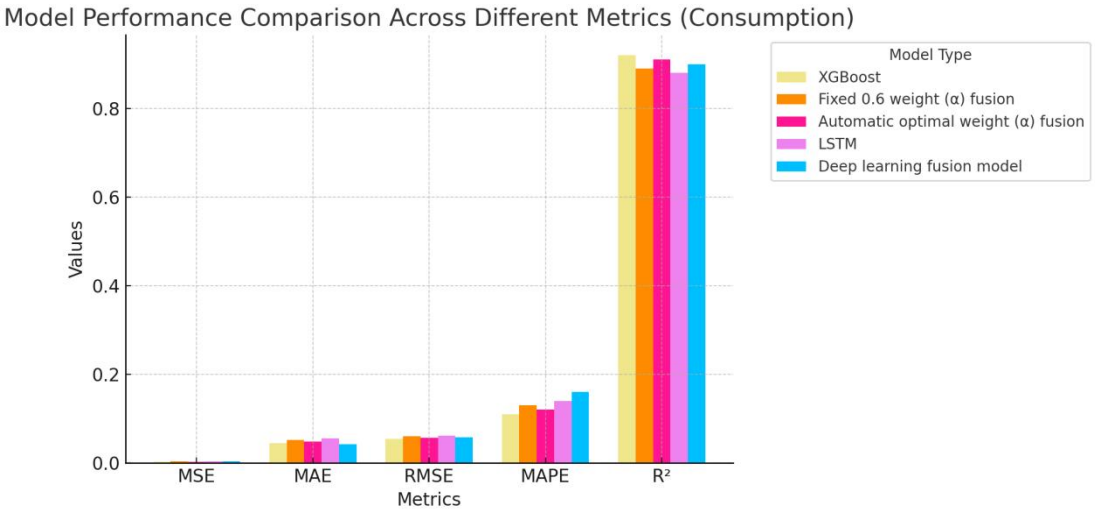


Figure 14

- **Electricity production prediction**

  Figure 15 shows the performance indicators of the XGBoost model, LSTM model and three fusion models on the electricity production prediction test set. It can be seen from the results:

  - The XGBoost model still performs best in all evaluation indicators and is significantly better than other models.
  - The fusion model slightly improves the performance of LSTM in some cases, but cannot surpass the XGBoost model overall.

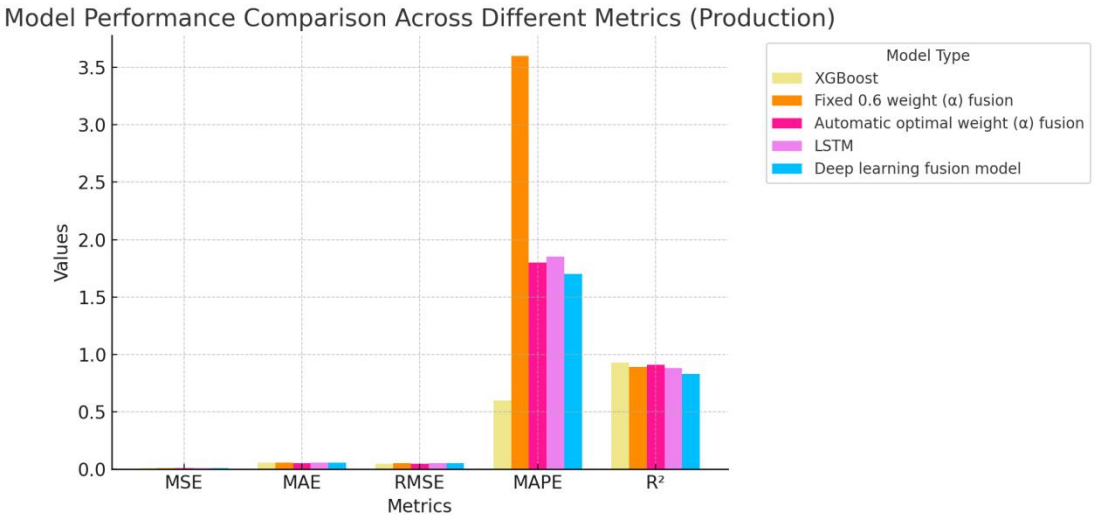


Figure 15

Combining the results of electricity consumption and electricity production prediction, the XGBoost model outperforms other models in all evaluation indicators and shows

excellent prediction ability. Therefore, the XGBoost model was finally selected as the main tool for predicting the energy behavior of solar panel prosumers (The relevant code of the XGBoost model can be seen in Appendix 6 and 7).

Conclusion

This study analyzed the electricity production and consumption behavior of distributed photovoltaic systems using the "Enefit - Predicting Energy Behavior of Prosumers" dataset. Among all the tested models, XGBoost demonstrated outstanding predictive performance, particularly in capturing nonlinear relationships and feature interactions. Additionally, efforts were made to further improve prediction accuracy through parameter optimization and model fusion, providing reliable data-driven insights for energy management. Key findings indicate that electricity production is primarily influenced by solar radiation, while consumption depends on prior consumption and production patterns.

- **Recommendations for Energy Companies**:
  1. Adopt XGBoost-based predictive tools to optimize solar resource allocation and grid load distribution.
  2. Implement dynamic energy pricing based on predictive insights to enhance profitability and consumer satisfaction.
  3. Invest in advanced data collection infrastructure, such as real-time smart meters, to improve prediction precision.

- **Future Model Optimization**:
  1. Integrate high-frequency, real-time data for more granular forecasting.
  2. Explore Transformer-based architectures to better capture long-term temporal dependencies.
  3. Apply transfer learning to adapt models across diverse geographical regions.

(Word Count: 2968)

References

Altmann, A., Toloşi, L., Sander, O. & Lengauer, T., 2010. Permutation importance: A corrected feature importance measure. *Bioinformatics*, 26(10), pp. 1340-1347.

Chen, T. & Guestrin, C., 2016. Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785-794.

IEA-PVPS, 2025. Overview of the Global Photovoltaic Market (2025 Release). Available at: https://iea-pvps.org/wp-content/uploads/2025/04/Task-1-Snapshot-2025-Chinese-Summary.pdf [Accessed 25 May 2025].

Poplawski, T., Dudzik, S. & Szeląg, P., 2023. Forecasting of energy balance in prosumer micro-installations using machine learning models. *Energies*, 16(18), p. 6726.

Qiu, L., 2023. Adjustment directions, impacts, and implications of European energy security policy. *China Energy Media Research Institute*. Available at: https://www.desn.com.cn/news/show-1604789.html [Accessed 25 May 2025].

Roth, V., 2004. The generalized LASSO. *IEEE Transactions on Neural Networks*, 15(1), pp. 16-28.

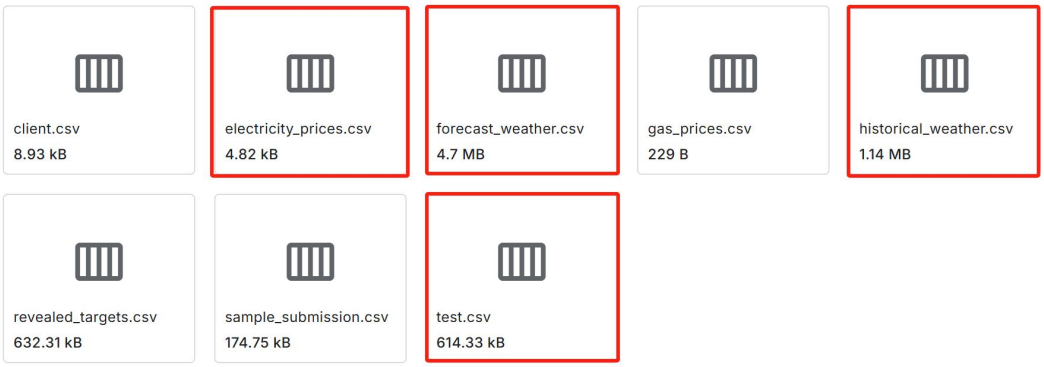
Staudemeyer, R.C. & Morris, E.R., 2019. Understanding LSTM--a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*. Available at: http://arxiv.org/abs/1909.09586 [Accessed 25 May 2025].

Wang, S., Li, C. & Lim, A., 2019. Why are the ARIMA and SARIMA not sufficient. *arXiv preprint arXiv:1904.07632*. Available at: http://arxiv.org/abs/1904.07632 [Accessed 25 May 2025].

Wirth, R. & Hipp, J., 2000. CRISP-DM: Towards a standard process model for data mining. *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, 1, pp. 29-39.

Appendices

Appendix 1. The images and URL of these original datasets



4 datasets selected from the original 8 datasets on kaggle

Data Source:

https://www.kaggle.com/competitions/predict-energy-behavior-of-prosumers

Appendix 2. New Feature

New Feature	Explanation
dewpoint_dep	**=[dewpoint-temperature]** Deviation related to dew point temperature, an indicator reflecting changes in air humidity.
diffuse_ratio	**=[direct_solar_radiation] ÷ [shortwave_radiation]** The diffusion ratio reflects the scattered part of solar radiation.
is_rainy	Binary notation for whether it is raining (1 indicates rain, 0 indicates no rain)
lag_prod_1h	The electricity production volume of the previous hour is used to capture the historical effect in the time series
lag_cons_1h	The electricity consumption in the previous hour reflects the consumption situation of the previous hour

The explanation of new feature

Appendix 3. The specific features selected by the two subsets

Feature	Explanation
lag_cons_1h	The electricity consumption in the previous hour reflects the consumption situation of the previous hour.
production	It represents the quantity of electricity production or the output of electricity.
lag_prod_1h	The electricity production volume of the previous hour is used to capture the historical effect in the time series.
sin_hour	The sinusoidal transformation value of an hour, used to capture periodic trends.
direct_solar_radiation_forecast	Prediction of future direct solar radiation.
surface_solar_radiation_downwards	The downward surface solar radiation reflects the solar radiation received by the ground.
hour	It represents the number of hours in a day and is usually used to capture seasonal changes.
cos_doy	The cosine transform value of the hour,Used for capturing periodic changes.
euros_per_mwh	The euro price per megawatt-hour of electricity repre s the electricity price.
cos_hour	The cosine transform value of the hour,is used to cap periodic changes.

Variables of the consumption model under the strategy of accumulating 90

Feature	Explanation
surface_solar_radiation_downwards	The downward surface solar radiation reflects the solar radiation received by the ground.
consumption	The electricity consumption in the previous hour reflects the consumption situation of the previous hour.
lag_prod_1h	The electricity production volume of the previous hour is used to capture the historical effect in the time series.
direct_solar_radiation_forecast	Prediction of future direct solar radiation: The prediction of future solar radiation based on meteorological models.
direct_solar_radiation	The direct solar radiation that accumulates to the planar surface perpendicular to the sun within one hour.
shortwave_radiation	Short-wave radiation, a component of solar radiation energy, affects temperature and power production.
cos_doy	The cosine transform value of the day of the year helps capture seasonal fluctuations.
lag_cons_1h	The electricity consumption in the previous hour reflects the consumption situation of the previous hour.
direct_solar_radiation	Direct solar radiation, affects power production.
total_precipitation	Total precipitation. Precipitation can affect temperature and electricity demand.
production_outlier_severity	The severity of production outliers measures whether production data deviates from the normal range.
cos_hour	The cosine transform value of the hou,is used to capture periodic changes.
cloudcover_total	Total cloud cover indicates the degree of cloud coverage in the sky and affects the amount of solar radiation.
cloudcover_total_forecast	Future cloud cover prediction, the degree of future cloud coverage predicted based on meteorological models.
diffuse_ratio	The diffusion ratio reflects the scattered part of solar radiation.

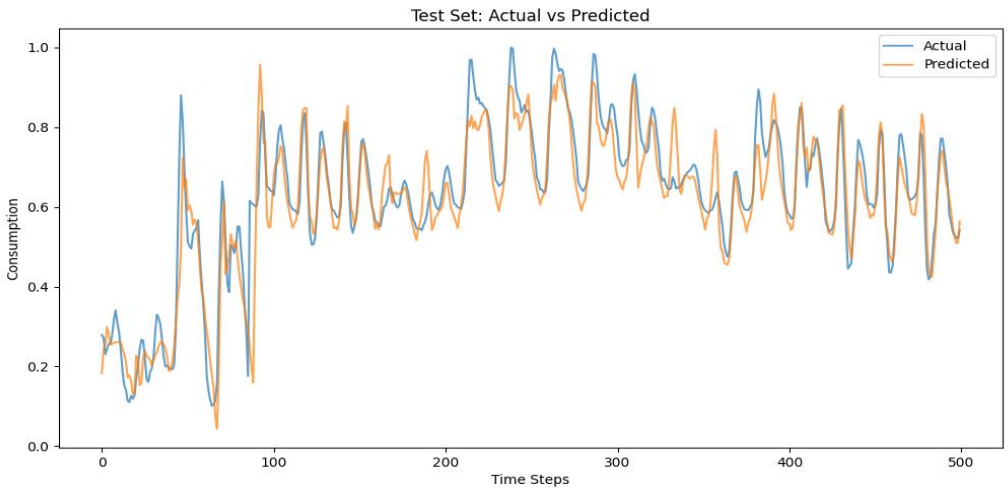
Variables of the production model under the strategy of accumulating 90

Appendix 4. The configuration process of LSTM in the experiment

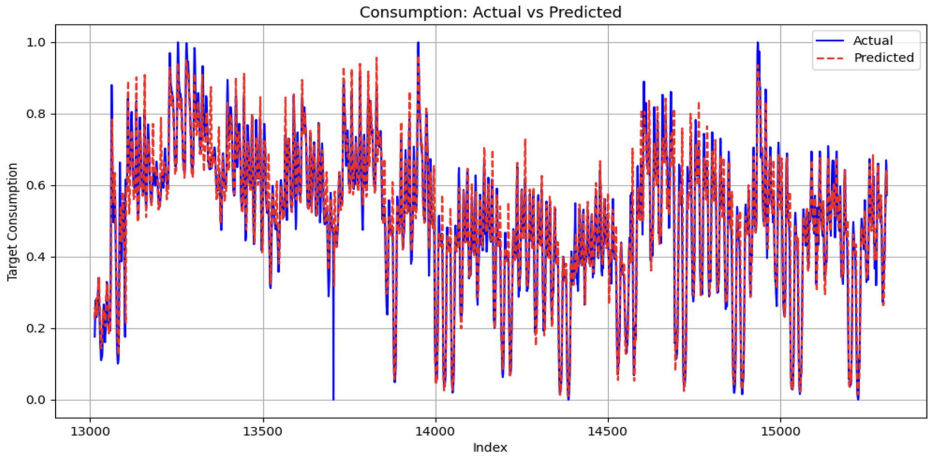
Config_ID	Layers	Units	Dropout_Rate	Use_Attention	Learning_Rate
1	1	64	0.2	FALSE	0.001
2	2	32	0.2	FALSE	0.001
3	2	64	0.2	FALSE	0.001
4	2	128	0.2	FALSE	0.001
5	2	64	0.3	FALSE	0.001
6	2	64	0.5	FALSE	0.001
7	2	64	0.2	FALSE	0.0001
8	2	64	0.2	TRUE	0.001
9	3	64	0.2	FALSE	0.001

Experimental Setup: LSTM Configurations

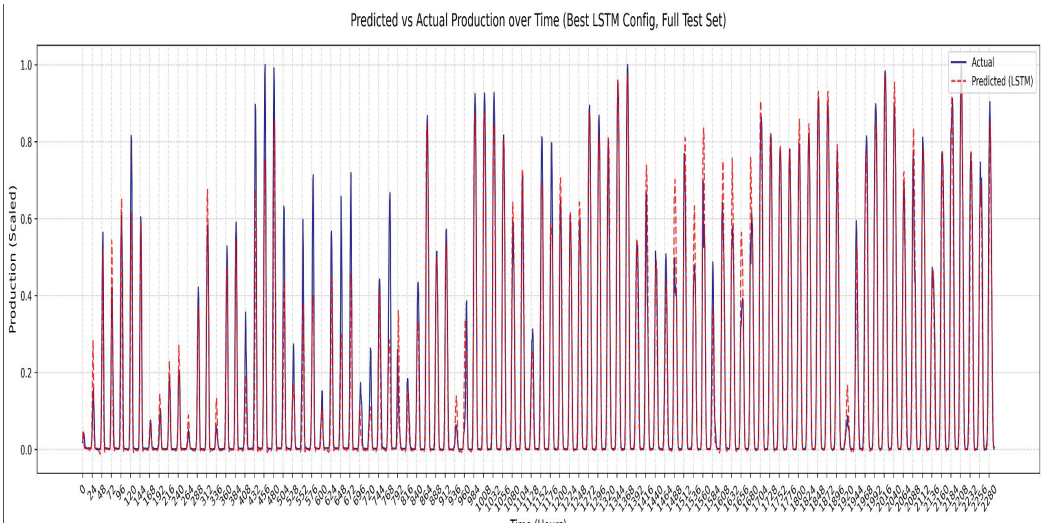
Appendix 5. The corresponding predicted line graph



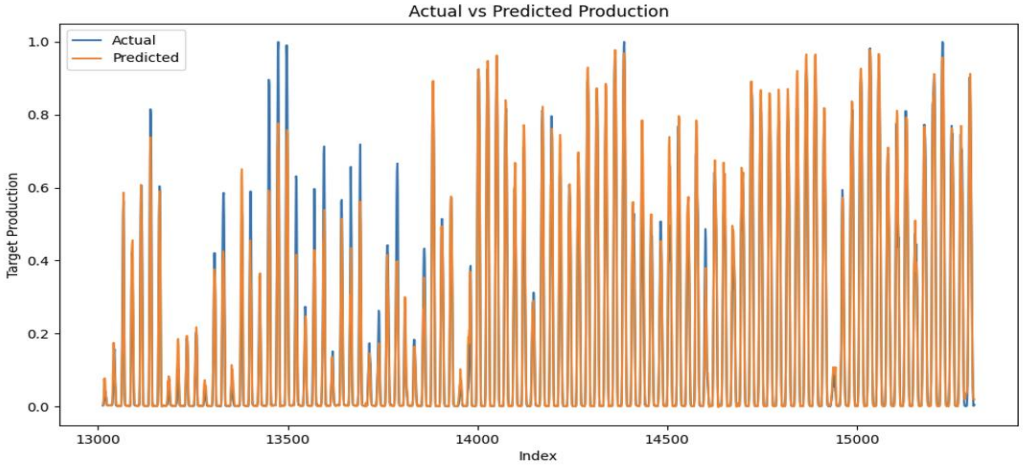
LSTM Actual vs Predicted-Consumption



LSTM Actual vs Predicted-Production



XGBoost Actual vs Predicted-Consumption



XGBoost Actual vs Predicted-Production

Appendix 6. The optimized XGBoost-Consumption model code

```python
from sklearn.model_selection import TimeSeriesSplit, train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import xgboost as xgb
import pandas as pd
import numpy as np
import datetime
import os
import optuna

# 创建基于时间戳的结果目录
timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
result_dir = f'results_{timestamp}'
os.makedirs(result_dir, exist_ok=True)

# 加载消费数据集
df_cons = pd.read_csv('/Users/hujiahui/Desktop/xgboost/final_data_for_consumption.csv')

# 提取特征和目标变量
X_cons = df_cons.drop( labels: ['target_cons'], axis=1).select_dtypes(include=['number'])
y_cons = df_cons['target_cons']

# 按时间顺序划分训练集（70%）、验证集（15%）、测试集（15%）
train_size = int(len(X_cons) * 0.7)
val_size = int(len(X_cons) * 0.15)

X_train_cons = X_cons[:train_size]
X_val_cons = X_cons[train_size:train_size + val_size]
X_test_cons = X_cons[train_size + val_size:]

y_train_cons = y_cons[:train_size]
y_val_cons = y_cons[train_size:train_size + val_size]
y_test_cons = y_cons[train_size + val_size:]

# 训练初始模型
model_cons = xgb.XGBRegressor(n_estimators=100, random_state=42)
```

```python
model_cons.fit(X_train_cons, y_train_cons)

# 定义单模型评估函数（包含 MAPE）
def evaluate_single_model(model, X_test, y_test, name, return_metrics=False):
    y_pred = model.predict(xgb.DMatrix(X_test) if isinstance(model, xgb.Booster) else X_test)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    # 更新 MAPE 计算，防止出现由于极小值引发的数据爆炸
    # 首先，确保 y_test 的值是 NumPy 数组，以便进行一致的数值操作
    y_test_np_array = y_test.to_numpy() if isinstance(y_test, pd.Series) else np.asarray(y_test)

    # 找到 y_test_np_array 中绝对值大于 1e-6 的元素的索引
    non_zero_idx = np.where(np.abs(y_test_np_array) > 1e-6)[0]

    if len(non_zero_idx) > 0:
        # 使用这些索引过滤 y_test_np_array 和 y_pred
        # y_pred 已经是 NumPy 数组
        y_test_filtered = y_test_np_array[non_zero_idx]
        y_pred_filtered = y_pred[non_zero_idx]

        # 计算 MAPE
        mape = np.mean(np.abs((y_test_filtered - y_pred_filtered) / y_test_filtered)) * 100
    else:
        # 如果所有 y_test 值都过小，则 MAPE 未定义或设为 NaN
        mape = np.nan

    print(
        f"模型 {name} 性能: \n  MSE: {mse:.4f}\n  RMSE: {rmse:.4f}\n  MAE: {mae:.4f}\n  R²: {r2:.4f}\n  MAPE: {mape:.4f}%\n")
    return [mse, rmse, mae, r2, mape] if return_metrics else None

# 评估初始模型（在验证集上）
metrics_initial_cons = evaluate_single_model(model_cons, X_val_cons, y_val_cons, name: "电力消费（初始模型）", return_metrics=True)
```

```python
# 超参数调优（5 轮）
tuning_rounds = 5

# 定义五折时间序列交叉验证
tscv = TimeSeriesSplit(n_splits=5)

# 消费模型调优目标函数（基于交叉验证）
def objective_cons(trial):
    params = {
        'n_estimators': trial.suggest_int('n_estimators', 1700, 1800),
        'max_depth': trial.suggest_int('max_depth', 4, 5),
        'learning_rate': trial.suggest_float('learning_rate', 0.0125, 0.013),
        'subsample': trial.suggest_float('subsample', 0.97, 0.98),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.67, 0.68),
        'gamma': trial.suggest_float('gamma', 0.0003, 0.0004),
        'min_child_weight': trial.suggest_int('min_child_weight', 7, 8),
        'alpha': trial.suggest_float('alpha', 0.42, 0.45),
        'lambda': trial.suggest_float('lambda', 0.42, 0.44),
        'objective': 'reg:squarederror'
    }

    # 在训练集上进行五折交叉验证
    fold_r2_scores = []
    for train_idx, val_idx in tscv.split(X_train_cons):
        X_train_cv, X_val_cv = X_train_cons.iloc[train_idx], X_train_cons.iloc[val_idx]
        y_train_cv, y_val_cv = y_train_cons.iloc[train_idx], y_train_cons.iloc[val_idx]

        model = xgb.XGBRegressor(random_state=42, **params)
        model.fit(X_train_cv, y_train_cv)
        y_pred = model.predict(X_val_cv)

        r2 = r2_score(y_val_cv, y_pred)
        fold_r2_scores.append(r2)

    # 返回交叉验证的平均 R²
    return np.mean(fold_r2_scores)

# 多轮贝叶斯优化 - 消费模型
best_r2_cons, best_params_cons = -float('inf'), None
for r in range(1, tuning_rounds + 1):
    print(f"--- 第 {r} 轮贝叶斯优化 (Consumption) ---")
    study = optuna.create_study(direction='maximize')
    study.optimize(objective_cons, n_trials=50)
    best_params = study.best_params
    print(f"第 {r} 轮最佳参数 (Consumption):", best_params)

    # 在整个训练集上训练模型，并在验证集上评估
    model = xgb.XGBRegressor(random_state=42, **best_params)
    model.fit(X_train_cons, y_train_cons)
    metrics = evaluate_single_model(model, X_val_cons, y_val_cons, name: f"第 {r} 轮电力消费", return_metrics=True)

    if metrics[3] > best_r2_cons:
        best_r2_cons, best_params_cons = metrics[3], best_params
print(f"整体最佳参数 (Consumption): {best_params_cons}\n整体最佳 R² (Consumption): {best_r2_cons}")

# 训练最终模型
dtrain_cons = xgb.DMatrix(X_train_cons, label=y_train_cons)
dtest_cons = xgb.DMatrix(X_test_cons, label=y_test_cons)
params_cons = best_params_cons.copy()
num_round_cons = params_cons.pop('n_estimators')
booster_cons = xgb.train(params_cons, dtrain_cons, num_boost_round=num_round_cons, evals=[(dtest_cons, 'eval')],
                         early_stopping_rounds=300, verbose_eval=False)
metrics_final_cons = evaluate_single_model(booster_cons, X_test_cons, y_test_cons, name: "最终电力消费模型", return_metrics=True)

# 保存最终模型为 JSON 文件
booster_cons.save_model(os.path.join(result_dir, f'best_model_cons_90plus_{timestamp}.json'))

# 将性能指标保存到 Excel 文件（包含 MAPE）
metrics_df = pd.DataFrame({
    'Metric': ['MSE', 'RMSE', 'MAE', 'R²', 'MAPE'],
    'Consumption_Initial': metrics_initial_cons,
    'Consumption_Final': metrics_final_cons
})
metrics_df.to_excel(os.path.join(result_dir, f'model_performance_cons_90plus_{timestamp}.xlsx'), index=False)
print(f"消费模型性能指标已保存为 '{result_dir}/model_performance_cons_90plus_{timestamp}.xlsx'")
print("消费模型训练完成，结果已保存至：", result_dir)
```

Appendix 7. The optimized XGBoost-Production model code

```python
from sklearn.model_selection import TimeSeriesSplit, train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import xgboost as xgb
import pandas as pd
import numpy as np
import datetime
import os
import optuna

# 创建基于时间戳的结果目录
timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
result_dir = f'results_{timestamp}'
os.makedirs(result_dir, exist_ok=True)

# 加载生产数据集
df_prod = pd.read_csv('/Users/hujiahui/Desktop/xgboost/final_data_for_production.csv')

# 提取特征和目标变量
X_prod = df_prod.drop( labels: ['target_prod'], axis=1).select_dtypes(include=['number'])
y_prod = df_prod['target_prod']

# 按时间顺序划分训练集（70%）、验证集（15%）、测试集（15%）
train_size = int(len(X_prod) * 0.7)
val_size = int(len(X_prod) * 0.15)

X_train_prod = X_prod[:train_size]
X_val_prod = X_prod[train_size:train_size + val_size]
X_test_prod = X_prod[train_size + val_size:]

y_train_prod = y_prod[:train_size]
y_val_prod = y_prod[train_size:train_size + val_size]
y_test_prod = y_prod[train_size + val_size:]

# 训练初始模型
model_prod = xgb.XGBRegressor(n_estimators=100, random_state=42)
```

```python
model_prod.fit(X_train_prod, y_train_prod)



# 定义单模型评估函数（包含 MAPE）
def evaluate_single_model(model, X_test, y_test, name, return_metrics=False):
    y_pred = model.predict(xgb.DMatrix(X_test) if isinstance(model, xgb.Booster) else X_test)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    # 更新 MAPE 计算，防止出现由于极小值引发的数据爆炸
    # 首先，确保 y_test 的值是 NumPy 数组，以便进行一致的数值操作
    y_test_np_array = y_test.to_numpy() if isinstance(y_test, pd.Series) else np.asarray(y_test)

    # 找到 y_test_np_array 中绝对值大于 1e-6 的元素的索引
    non_zero_idx = np.where(np.abs(y_test_np_array) > 1e-6)[0]

    if len(non_zero_idx) > 0:
        # 使用这些索引过滤 y_test_np_array 和 y_pred
        # y_pred 已经是 NumPy 数组
        y_test_filtered = y_test_np_array[non_zero_idx]
        y_pred_filtered = y_pred[non_zero_idx]

        # 计算 MAPE
        mape = np.mean(np.abs((y_test_filtered - y_pred_filtered) / y_test_filtered)) * 100
    else:
        # 如果所有 y_test 值都过小，则 MAPE 未定义或设为 NaN
        mape = np.nan

    print(
        f"模型 {name} 性能: \n  MSE: {mse:.4f}\n  RMSE: {rmse:.4f}\n  MAE: {mae:.4f}\n  R²: {r2:.4f}\n  MAPE: {mape:.4f}%\n")
    return [mse, rmse, mae, r2, mape] if return_metrics else None

# 评估初始模型（在验证集上）
metrics_initial_prod = evaluate_single_model(model_prod, X_val_prod, y_val_prod, name: "电力生产（初始模型）", return_metrics=True)
```

```python
# 超参数调优（5 轮）
tuning_rounds = 5

# 定义五折时间序列交叉验证
tscv = TimeSeriesSplit(n_splits=5)

# 生产模型调优目标函数（基于交叉验证）
def objective_prod(trial):
    params = {
        'n_estimators': trial.suggest_int('n_estimators', 500, 1500),
        'max_depth': trial.suggest_int('max_depth', 6, 8),
        'learning_rate': trial.suggest_float('learning_rate', 0.005, 0.02),
        'subsample': trial.suggest_float('subsample', 0.9, 1.0),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.85, 0.95),
        'gamma': trial.suggest_float('gamma', 0, 0.001),
        'min_child_weight': trial.suggest_int('min_child_weight', 5, 9),
        'alpha': trial.suggest_float('alpha', 0.3, 0.5),
        'lambda': trial.suggest_float('lambda', 0.4, 0.6),
        'objective': 'reg:squarederror'
    }

    # 在训练集上进行五折交叉验证
    fold_r2_scores = []
    for train_idx, val_idx in tscv.split(X_train_prod):
        X_train_cv, X_val_cv = X_train_prod.iloc[train_idx], X_train_prod.iloc[val_idx]
        y_train_cv, y_val_cv = y_train_prod.iloc[train_idx], y_train_prod.iloc[val_idx]

        model = xgb.XGBRegressor(random_state=42, **params)
        model.fit(X_train_cv, y_train_cv)
        y_pred = model.predict(X_val_cv)

        r2 = r2_score(y_val_cv, y_pred)
        fold_r2_scores.append(r2)

    # 返回交叉验证的平均 R²
```

```python
    return np.mean(fold_r2_scores)

# 多轮贝叶斯优化 - 生产模型
best_r2_prod, best_params_prod = -float('inf'), None
for r in range(1, tuning_rounds + 1):
    print(f"--- 第 {r} 轮贝叶斯优化 (Production) ---")
    study = optuna.create_study(direction='maximize')
    study.optimize(objective_prod, n_trials=50)
    best_params = study.best_params
    print(f"第 {r} 轮最佳参数 (Production):", best_params)

    # 在整个训练集上训练模型，并在验证集上评估
    model = xgb.XGBRegressor(random_state=42, **best_params)
    model.fit(X_train_prod, y_train_prod)
    metrics = evaluate_single_model(model, X_val_prod, y_val_prod, name: f"第 {r} 轮电力生产", return_metrics=True)

    if metrics[3] > best_r2_prod:
        best_r2_prod, best_params_prod = metrics[3], best_params
print(f"整体最佳参数 (Production): {best_params_prod}\n整体最佳 R² (Production): {best_r2_prod}")

# 训练最终模型
dtrain_prod = xgb.DMatrix(X_train_prod, label=y_train_prod)
dtest_prod = xgb.DMatrix(X_test_prod, label=y_test_prod)
params_prod = best_params_prod.copy()
num_round_prod = params_prod.pop('n_estimators')
booster_prod = xgb.train(params_prod, dtrain_prod, num_boost_round=num_round_prod, evals=[(dtest_prod, 'eval')],
                         early_stopping_rounds=300, verbose_eval=False)
metrics_final_prod = evaluate_single_model(booster_prod, X_test_prod, y_test_prod, name: "最终电力生产模型", return_metrics=True)

# 保存最终模型为 JSON 文件
booster_prod.save_model(os.path.join(result_dir, f'best_model_prod_90plus_{timestamp}.json'))

# 将性能指标保存到 Excel 文件（包含 MAPE）
metrics_df = pd.DataFrame({
    'Metric': ['MSE', 'RMSE', 'MAE', 'R²', 'MAPE'],
```

```python
    'Production_Initial': metrics_initial_prod,
    'Production_Final': metrics_final_prod
})
metrics_df.to_excel(os.path.join(result_dir, f'model_performance_prod_90plus_{timestamp}.xlsx'), index=False)
print(f"生产模型性能指标已保存为 '{result_dir}/model_performance_prod_90plus_{timestamp}.xlsx'")
print("生产模型训练完成，结果已保存至：", result_dir)
```