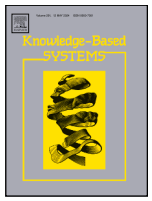
Knowledge-Based Systems 329 (2025) 114300

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



AutoEnergy: An automated feature engineering algorithm for energy consumption forecasting with AutoML

Nasser Alkhulaifi [a,*], Alexander L. Bowler [b], Direnc Pekaslan [c], Nicholas J. Watson [b], Isaac Triguero [a,d,e]

[a] Computational Optimisation and Learning (COL) Lab, School of Computer Science, University of Nottingham, Nottingham, NG8 1BB, UK
[b] School of Food Science and Nutrition, Faculty of Environment, University of Leeds, Leeds, LS2 9JT, UK
[c] Lab for Uncertainty in Data and Decision Making (LUCID), School of Computer Science, University of Nottingham Nottingham, NG8 1BB, UK
[d] Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain
[e] DaSCI Andalusian Institute in Data Science and Computational Intelligence, Granada, Spain

ARTICLE INFO

Keywords:
Automated feature engineering
Automated machine learning
Automl
Energy consumption forecasting
Power consumption prediction

ABSTRACT

Feature engineering (FE) plays a crucial role in Machine Learning pipelines, yet it remains a time-consuming process requiring heavy domain expertise. While Automated Machine Learning (AutoML) has automated model selection and hyperparameter tuning, it often overlooks FE, which is particularly needed in specialised domains such as Energy Consumption Forecasting (ECF). To address this limitation, we introduce AutoEnergy, a novel, domain-aware FE algorithm tailored for ECF. AutoEnergy automatically generates interpretable features from timestamps and past consumption values through rule-based transformations, integrating them with AutoML for fully automated ECF modelling while reducing human intervention. The performance of AutoEnergy was evaluated using eighteen diverse real-world energy consumption datasets spanning residential, commercial, industrial, and grid power domains. Through extensive benchmarking against baseline AutoML without FE and established FE methods, namely TSFresh (with TSEfficient and TSMinimal configurations) and FeatureTools (FT), AutoEnergy demonstrated significant improvements in both predictive accuracy and computational efficiency. AutoEnergy achieved forecasting error reductions of 19.52 % to 84.72 % compared to benchmarking methods, with strong performance on smaller datasets and statistical validation via Friedman and Wilcoxon tests. AutoEnergy demonstrated notable computational efficiency by running 1.31 and 4.41 times faster than FT and TSEff, respectively. Although 1.58 times slower than TSMin, AutoEnergy achieved 82.38 % lower forecasting errors. Integrating AutoEnergy with the state-of-the-art Tabular Prior Data Fitted Network (TabPFN) resulted in significant forecasting error reductions across test sets. These findings highlight AutoEnergy's potential to improve AutoML performance while reducing reliance on domain expertise for FE, paving the way for fully automated ML pipelines in ECF applications.

1. Introduction and background

Feature Engineering (FE) is a crucial step in developing machine learning (ML) pipelines, as it transforms raw data into informative features that enhance model performance [1,2]. This is because raw data are often not ideal in their original form for algorithms to learn effectively [3,4]. Furthermore, real-world datasets are often small or limited due to data collection limitations, privacy issues, or resource constraints [5,6]. In such scenarios, FE could compensate for the limited data by extracting informative features. Moreover, FE may improve computational efficiency by eliminating noisy, redundant, and irrelevant data [7,8]. Finally, FE may improve not only predictive accuracy but also explainability [9].

Despite these advantages, FE remains a time-consuming process that is prone to human error while relying heavily on domain expertise and iterative experimentation [7,10]. Deep learning algorithms [11,12], while capable of automatically learning useful representations from raw data, lack interpretability and typically require large datasets to perform well, a condition that is often infeasible in real-world scenarios [5,13]. This has led to a growing interest in automated FE methods [14].

Modern Automated Machine Learning (AutoML) frameworks [15] such as AutoGluon [16], H2O [17], and FLAML [18] offer streamlined

* Corresponding author.
E-mail addresses: Nasser.Alkhulaifi@nottingham.ac.uk (N. Alkhulaifi), a.l.bowler@leeds.ac.uk (A.L. Bowler), direnc.pekaslan1@nottingham.ac.uk (D. Pekaslan), n.j.watson@leeds.ac.uk (N.J. Watson), triguero@decsai.ugr.es (I. Triguero).

https://doi.org/10.1016/j.knosys.2025.114300
Received 19 February 2025; Received in revised form 15 August 2025; Accepted 17 August 2025
Available online 25 August 2025

solutions for ML pipeline development through automated model selection and hyperparameter tuning. Nevertheless, these solutions often assume that data preparation and feature generation have been completed and the data are ready for training. As a result, tasks such as FE and the integration of domain knowledge are largely left to human practitioners [14].

Furthermore, many general-purpose AutoML systems are proposed for broad applicability across ML tasks, but their focus on generalisation often limits their effectiveness in specialised domains [19]. Such limitations become particularly evident in domains that require interpretable features, such as energy consumption forecasting (ECF) of time series data, where understanding the factors influencing energy consumption is important for well-informed decisions. This challenge may be attributed to the complex nature of power usage patterns, which can involve various linear and nonlinear relationships, fluctuating behaviours, and potential dependencies on temporal and environmental factors [20,21]. Therefore, domain-relevant FE could be a beneficial approach for AutoML to excel in modelling ECF problems.

1.1. Feature engineering in supervised learning for energy consumption forecasting

To better understand the FE process and its objectives in supervised learning contexts, it can be formally defined as follows: given a dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$ with N instances, the objective is to find a feature transformation function $\phi : X \rightarrow X'$ that improves the performance of a learning algorithm A when trained on the transformed dataset $D' = \{(\phi(x_i), y_i)\}_{i=1}^{N}$. Here, x_i represents the input features, y_i denotes the corresponding target values, X is the original feature space, and X' is the transformed feature space [14].

In ECF, implementing FE begins with identifying key raw features, which vary across studies. Yet, historical data and weather information are consistently identified as the most significant ones [22]. Historical data offers insights into consumption patterns, while weather data reflects environmental factors, such as outdoor temperature that affect energy usage. Some studies; however, have improved forecasting models by engineering features from raw data, such as time-based and periodic features, to capture energy temporal patterns more comprehensively [23,24].

Nonetheless, manually extracting these features often requires both energy domain expertise and data science skills [7,8]. Energy domain experts identify the key factors influencing energy consumption, while data scientists validate these insights through data analysis and extract the relevant features from the raw data. Consequently, there is a pressing need to automate FE in ECF problems to streamline the process, reduce human bias, and enhance model performance without requiring extensive domain-specific knowledge for each new data.

1.1.1. Automated feature engineering

Automated FE encompasses a range of approaches aimed at reducing manual intervention in the feature generation process by leveraging algorithmic solutions [25]. The concept of *automated* FE, as proposed by [8], may involve two key steps: first, generating a comprehensive search space of possible feature processing operations, and second, employing optimisation techniques to identify the most effective FE strategy. These operations, which include transformations such as aggregation functions and arithmetic operations, can be combined to create a set of new informative features. Although the literature lacks explicit and comprehensive categorisations of automated FE methods relevant to ECF problems, we may broadly classify the current landscape into the following categories:

- Traditional Statistical Methods: these methods apply statistical and mathematical transformations to create new features through dimensionality reduction or statistical computations. For example, Principal Component Analysis (PCA) [26] and wavelet decomposition [27] are widely employed in ECF to improve performance and automate FE. However, these methods present significant challenges when applied to ECF problems. While PCA effectively reduces data dimensionality, it often obscures the intuitive relationship between the transformed principal components and the original data features, complicating interpretation. Similarly, wavelet decomposition generates coefficients that are more complex and less interpretable than the raw data, requiring specialised domain knowledge for proper interpretation. These interpretability challenges are particularly critical in ECF, where understanding the factors influencing energy use is essential for developing effective policies and solutions.
- Learning-Based Methods: these methods primarily utilise artificial neural networks to learn and automatically generate new features from raw data. A prominent example is Autoencoders, which have been studied in ECF [11,28,29]. However, Autoencoders require large datasets and produce abstract features with unclear ties to the original data, hindering explainability. While post hoc methods such as LIME [30] which is limited to local interpretability and SHAP [31] offer insights, their reliability is debated [32,33]. LIME's explanations, for example, depend heavily on parameter choices and may exclude important features [34], further emphasising the need for inherently interpretable features for ECF.
- Hybrid Semi-Automated Methods: the CAAFE method [14] is an example of a context-aware semi-automated FE approach designed for tabular data. This method integrates human expertise (i.e., domain knowledge) with large language models (LLMs) to streamline the FE process. While some steps in this method are automated, human intervention is still necessary to guide the process and make critical decisions. Such methods face two key challenges: computational limitations when handling datasets with a large number of attributes, and the potential for LLM hallucinations during feature generation.
- Heuristic-Based Methods: these search-based algorithms rely on predefined rules to generate new features. For example TSfresh (TS) [35], a widely used automated FE method in the literature, applies various time series characterisation techniques to compute features without the need for manual intervention. This algorithm characterises time series data in terms of data point distribution, correlation properties, stationarity, entropy, and nonlinear time series analysis. However, this method can be computationally expensive and risk overfitting due to the large number of generated features [36,37]. Another widely recognised search-based FE method is Featuretools (FT) which was developed based on the Deep Feature Synthesis algorithm [38]. This algorithm applies a series of transformation functions to create new features. These functions include mathematical operations such as summing, averaging, and counting functions. Although it has shown promising results in domains such as education and e-commerce, particularly in predicting student dropout, project excitement, and repeat buyer behaviour, its application in domains such as ECF remains unexplored.

Nevertheless, the literature lacks clarity regarding what *automated* truly means in the context of FE in supervised learning. That said, this work focuses on heuristic-based methods as benchmarks for comparison with the proposed method for the following four key reasons: (a) they represent established FE techniques directly applicable to ECF problems without additional training or human intervention, (b) the proposed method aligns with the heuristic-based approaches by employing rule-based criteria for FE, (c) these methods generally maintain traceable FE processes supporting interpretable energy consumption analysis and (d) they are potentially more readily integrated into AutoML frameworks, thus enabling more practical automated pipelines.

1.2. Research novelty and contributions

Given the aforementioned limitations, this work proposes `AutoEnergy`, a domain-aware automated FE method that constitutes a

novel synthesis to: (a) improve AutoML performance through domain-specific FE optimised for ECF; (b) minimise the human intervention and domain expertise required by automating the time-consuming, manual FE process; and (c) maintain interpretability through a heuristic search design, generating human-readable and traceable features. The key novelty of the proposed FE algorithm lies not in the individual components (i.e., feature types), but in its fully automated, expert-free, domain-aware feature extraction, optimised selection, and strategic integration tailored to ECF problems, enabling end-to-end automated ECF modelling when integrated with AutoML. The method automatically generates features by applying a series of FE functions to the dataset's timestamps and target variables (i.e. past consumption values). To assess the proposed method on AutoML performance, a comprehensive set of eighteen real-world datasets, representing various energy consumption patterns in different domains (e.g., residential and commercial buildings, wind turbines, industrial settings, food storage facilities, and grid power consumption), was utilised [1]. Additionally, the proposed method was systematically compared with well-established FE methods in the literature, namely TS and FT, assessing both (a) predictive accuracy and (b) computational efficiency. This thoroughly evaluated the method's overall effectiveness and practical applicability in real-world ECF tasks. While our previous exploratory study [39] laid the foundation for this paper, the novelty and contributions of the current work are as follows:

- An improved automated FE method tailored for ECF is introduced with a higher degree of automation in feature extraction and optimised selection compared to our previous work, as introduced and explained in Section 2. It considerably improves AutoML performance, as demonstrated by comparisons with and without the proposed FE method, as shown in Subsection 4.1. Additionally, the impact of the proposed FE method on the state-of-the-art Tabular Prior Data Fitted Network (TabPFN) algorithm [40] was also examined. [2]
- Compared to our previous investigation, this work includes comprehensive benchmarking against existing FE methods. The proposed FE method achieves a superior reduction in forecasting errors and computational efficiency for ECF problems compared to the benchmarking methods, as shown in Subsection 4.2.
- Compared to our previous study, this work incorporates a wider range of energy consumption datasets, including new energy systems and settings such as steel manufacturing environments and food and drinks cold facilities. This allows for a more extensive evaluation of the proposed method's applicability and effectiveness across different settings and environments.

The structure of the remaining sections of this paper is as follows: Section 2 outlines the proposed method for automating FE for ECF problems, Section 3 provides details of the experimental design, including the datasets used, benchmarking methods, and evaluation criteria. Analysis and discussion of findings are presented in Section 4. To conclude, Section 5 summarises key insights.

2. AutoEnergy: an automated end-to-end feature engineering algorithm

This section outlines the problem under investigation in Subsection 2.1 and subsequently details the proposed method to address it in Subsection 2.2.

2.1. Problem definition

Given a dataset D consisting of N instances, where each instance is represented by a tuple (t_i, y_i), with t_i denoting a timestamp and y_i the corresponding target variable, we define the dataset as follows:

$$D = \{(t_1, y_1), (t_2, y_2), \ldots, (t_N, y_N)\} \tag{1}$$

The aim is to develop an algorithm that implements an automated FE process to enhance the performance of a predictive model $\mathcal{M}$ within an AutoML framework.

2.2. AutoEnergy: the proposed feature engineering method

The proposed FE method, as illustrated Fig. 1, applies a set of FE functions $\{\text{GeneratedFeatures}_j\}_{j=1}^{M}$ which process the timestamp t_i and the target variable y_i in D to generate a series of feature subsets $\mathbf{F}'_{i,j}$. The complete feature vector $\mathbf{F}'_i$ for each instance i is then created by concatenating these feature subsets:

$$\mathbf{F}'_i = \bigoplus_{j=1}^{M} \text{GeneratedFeatures}_j(t_i, y_i), \quad \forall i \in \{1, \ldots, N\} \tag{2}$$

where $\bigoplus$ denotes the concatenation operation, combining all feature subsets $\mathbf{F}'_{i,j}$ generated by the functions into a single feature vector for each instance.

Following this, the predictive model $\mathcal{M}$ is trained using these complete feature vectors $\mathbf{F}'_i$ along with their corresponding target variable y_i, in an AutoML framework:

$$\mathcal{M} = \text{AutoML}\left(\{(\mathbf{F}'_i, y_i)\}_{i=1}^{N}\right) \tag{3}$$

Although many of these features exist in the literature, the proposed algorithm is fully automated, expert-free, domain-aware feature extraction, optimised selection, and strategic integration tailored to ECF problems, enabling end-to-end automated ECF modelling when integrated with AutoML. In particular, for lag and rolling-window statistical features, as described in the following subsections:

2.2.1. Temporal, sine and cosine transform-derived features

These features exploit temporal data to identify patterns influenced by time, such as distinguishing between on-peak vs off-peak hours and weekdays vs weekends, as shown in Fig. 1. In the first function, namely F_{TIME}, of Algorithm 1, time-based features are extracted from the timestamp t_i such as the hour of the day as outlined in steps 5–8. These time-related features provide insights into temporal patterns and trends in the data. The second function, namely F_{CYCLICAL}, of Algorithm 1, applies Fourier-based transformations, which has been used in time series feature encoding [4], to temporal variables through steps 9–22, where 'hour of day' ranges from 0 to 23, and 'day of week' ranges from 0 (Monday) to 6 (Sunday) to represent the daily and weekly seasonal cycle. This transformation is justified by the following key principles:

- The Fourier series theorem establishes that periodic patterns, such as energy consumption's daily and weekly seasonality, can be approximated through a combination of sinusoidal components. This mathematical foundation serves as an approach for capturing cyclical consumption patterns across various temporal frequencies.
- Sine/cosine transformations map temporal features onto a unit circle (where hour 0 and hour 23 become adjacent points), thus preserving the natural circular topology of time, see Fig. 2. This mathematical property ensures that every time point has a unique, continuous representation while maintaining the cyclical relationship between adjacent time periods. More importantly, this transformation is bijective and information-preserving: each hour maps to distinct coordinates with no averaging or smoothing of features, as demonstrated by the uniform Euclidean distances (0.26) between all adjacent hours in Fig. 2 (d). The continuous appearance of the sine/cosine functions reflects their inherent mathematical properties rather than any

[1] All results in this work are reproducible as the code and datasets used are publicly available on GitHub. See https://github.com/Nasser-Alkhulaifi/AutoEnergy

[2] TabPFN is a foundation model that leverages transformer-based meta-learning, designed specifically for tabular data problems and recognised as a state-of-the-art AutoML framework.

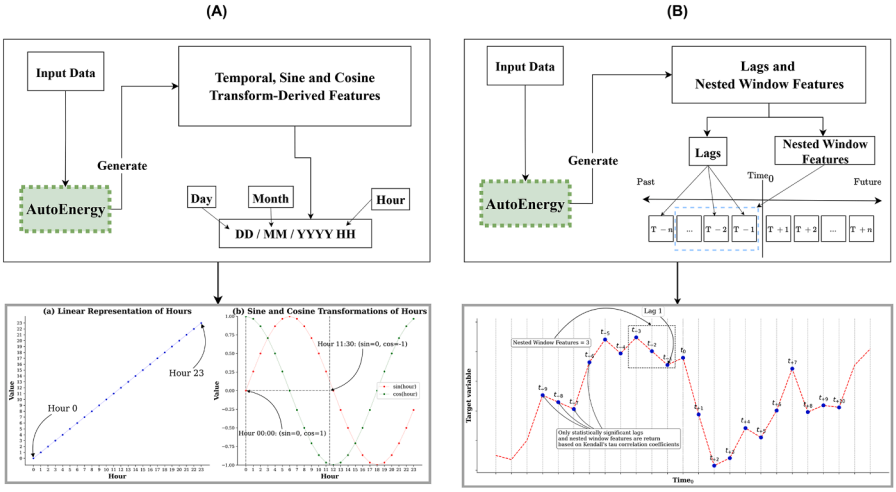


Fig. 1. The proposed Algorithm 1 for generating temporal, sine, and cosine transform-derived features is illustrated in subfigure (A), while the proposed Algorithm 2 for generating lag and nested window features is illustrated in subfigure (B).

Algorithm 1 Temporal, sine and cosine transform-derived features.

Input: DataFrame D with t (timestamps) and y (target variable)
Output: DataFrame with temporal and cyclical features

$D' \leftarrow D$
$Features_{\text{time}} \leftarrow F_{\text{time}}(D, t)$
$Features_{\text{cyclical}} \leftarrow F_{\text{cyclical}}(Features_{\text{time}})$
$D' \leftarrow D'$ append $Features_{\text{time}}$, $Features_{\text{cyclical}}$
function $F_{\text{TIME}}(D, t)$
    $Features_{\text{time}} \leftarrow$ Extract time-based features from t
    return $Features_{\text{time}}$
end function
function $F_{\text{CYCLICAL}}(Features_{\text{time}})$
    $Features_{\text{cyclical}} \leftarrow$ empty list
    for feature f in $Features_{\text{time}}$ **do**
        if f is 'hour of day' **then**
            $f_{\sin} \leftarrow \sin\left(\frac{2\pi f}{24}\right)$
            $f_{\cos} \leftarrow \cos\left(\frac{2\pi f}{24}\right)$
        else if f is 'day of week' **then**
            $f_{\sin} \leftarrow \sin\left(\frac{2\pi f}{7}\right)$
            $f_{\cos} \leftarrow \cos\left(\frac{2\pi f}{7}\right)$
        end if
        Append $f_{\sin}, f_{\cos}$ to $Features_{\text{cyclical}}$
    end for
    return $Features_{\text{cyclical}}$
end function

loss of temporal resolution; each of the 24 hours retains its unique identity in the transformed space. Unlike linear encoding of hours (0–23), which may create a misleading maximum distance between hour 23 and hour 0, the circular transformation ensures these temporally adjacent hours maintain their true neighbouring relationship. This continuous representation aligns with the physical reality of energy usage patterns, where consumption often changes gradually due to thermal inertia and operational behaviours unless disrupted by sudden events.

2.2.2. Lags and nested window features

The proposed algorithm utilises an automated approach to identify statistically significant lags and computes rolling statistics across nested window sizes using Kendall's tau correlation. These features capture temporal dependencies and multi-scale statistical characteristics in the energy time series data, as shown in Algorithm 2, and Fig. 1 and as follows:

A: In the first function, namely $F_{\text{LAGS}}(y)$ of Algorithm 2, and as outlined in steps 10–13, the algorithm employs Kendall's tau correlation to automatically identify statistically significant lags of y_i (i.e. the target variable). This process is inspired by the FRESH (Feature Extraction and Scalable Hypothesis Testing) algorithm [41], ensuring that only useful lags are retained. Kendall's tau is a rank-based correlation statistic to assess the strength and direction of association between two variables, where its p-value tests the null hypothesis that there is no association between the current and lag values. It is defined as:

$$\tau = \frac{(C - D)}{\sqrt{(C + D + T) \cdot (C + D + U)}} \tag{4}$$

where C is the number of concordant pairs, D is the number of discordant pairs, T is the number of ties only in the first variable, and U is the number of ties only in the second variable. This approach is justified by the following:

Algorithm 2 Lags and nested window features.

Input: DataFrame D with target variable y
Output: DataFrame with lag and window features

$D' \leftarrow D$
$top\_lags \leftarrow F_{\text{lags}}(y)$ ▷ Find significant lags
$Features_{\text{lags}} \leftarrow$ Create features using top_lags
$top\_windows \leftarrow F_{\text{stats}}(y)$ ▷ Find optimal window sizes
$Features_{\text{stats}} \leftarrow$ empty list
for $window\_size$ in $top\_windows$ **do**
    Compute rolling statistics of y over $window\_size$
    Add computed statistics to $Features_{\text{stats}}$
end for
function $F_{\text{LAGS}}(y)$
    Compute statistical significance for each lag
    return top 10 significant lags
end function
function $F_{\text{STATS}}(y)$
    $max\_window\_size \leftarrow \lfloor \text{length}(y)/3 \rfloor$
    Evaluate rolling windows up to max_window_size
    return top 10 significant window sizes
end function

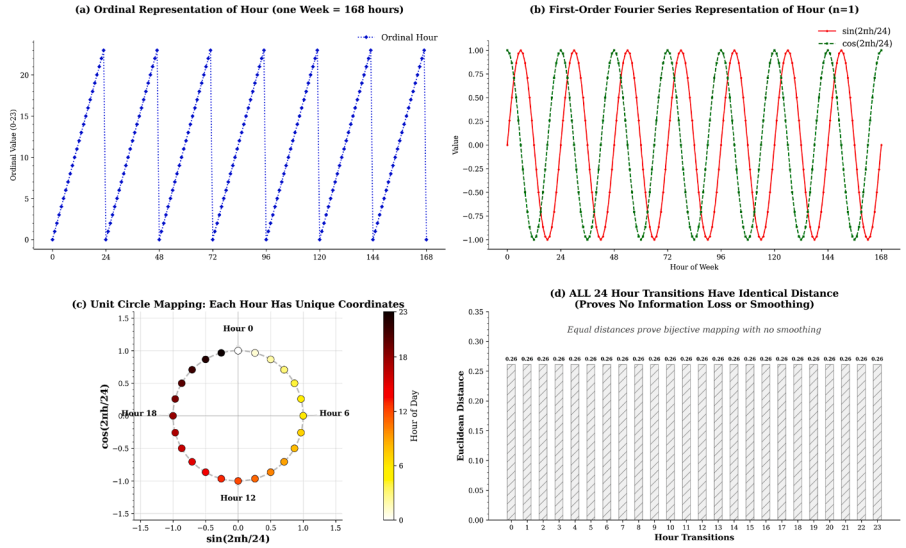


Fig. 2. Ordinal versus Fourier-based cyclical encoding of temporal features demonstrating information preservation without smoothing. (a) Ordinal representation of hours (0–23) over one week (168 hours) showing a discontinuous pattern with artificial jumps between hour 23 and 0. (b) First-order Fourier series transformation using $\sin(2\pi h/24)$ and $\cos(2\pi h/24)$ providing continuous cyclical features. (c) Unit circle mapping where each hour maps to unique coordinates, proving a bijective (one-to-one) transformation with no information loss. (d) Euclidean distances between all 24 consecutive hour pairs after transformation, showing identical distance of 0.26 for every transition, mathematically confirming that the Fourier encoding preserves all temporal information without averaging or smoothing effects. The apparent continuity reflects the inherent mathematical properties of trigonometric functions, not any loss of temporal resolution.

- Non-parametric robustness: Kendall's tau assumes no specific distribution, which is suitable for energy data that may not follow normal distributions due to irregular consumption patterns. It demonstrates superior robustness to outliers compared to Pearson correlation [42], where the latter is sensitive to such anomalies and consumption spikes, which is critical for energy data that frequently contain these irregularities.
- Statistical significance: hypothesis testing on Kendall's tau coefficients with a p-value threshold of 0.05 ensures that only lags exhibiting statistically significant correlations with the target variable y_i are retained. Lags with ($p < .05$) indicate less than a 5 % probability that the observed correlation is due to chance. This statistical filter ensures that the selected lags preserve genuine temporal dependencies and provides an additional layer of validation for the chosen lags.
- Computational efficiency: to handle large datasets effectively and avoid highly correlated lags (i.e., reduce multicollinearity), only the top ten lags with the lowest p-values among the significant lags are generated.

B: In the second function, namely $F_{\text{STATS}}(y)$ of Algorithm 2, and as outlined in steps 14–18, the algorithm incorporates a nested rolling window approach to compute statistical features at multiple time scales. It evaluates a range of window sizes up to one-third of the series length. However, it is worth acknowledging that this cap is empirical and justified by the following:

- Sensitivity analysis: an experiment was conducted across multiple window size thresholds (one-quarter, one-third, one-half, and the full sequence length) to systematically evaluate the impact of the maximum nested-window length on model performance and FE processing time. Although this approach is not theoretically optimal, it remains empirically grounded, as it preserves the automated element of FE while offering a practical trade-off balance between forecasting accuracy and processing efficiency.
- Computational efficiency and multi-scale pattern capture: limiting the maximum window to one-third of the series length attempts to balance capturing short-term fluctuations, medium-term variations, and longer-term trends while avoiding redundant historical information and containing computational complexity. It prevents the creation of excessively large windows that a) produce substantial overlap between consecutive rolling statistics, leading to multicollinearity; b) increase memory requirements and computational overhead without proportional gains in predictive value; and c) risk over-fitting by incorporating overly broad temporal contexts that may not reflect underlying energy-consumption patterns.
- Statistical significance: the algorithm computes Kendall's tau correlations for the rolling-window statistics and retains window sizes with $p < 0.05$. This statistical filter further ensures that, irrespective of the maximum window-size threshold, only relationships that are statistically significant are preserved, providing an additional layer of validation for the selected window sizes.

For the retained window sizes, the algorithm computes the following rolling statistics:

- Rolling mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{5}$$

- Rolling standard deviation:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{6}$$

- Rolling maximum:

$$\max = \max(x_1, x_2, \ldots, x_n) \tag{7}$$

- Rolling minimum:

$$\min = \min(x_1, x_2, \ldots, x_n) \tag{8}$$

- Rolling kurtosis:

$$\text{Kurt} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4}{(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2)^2} - 3 \tag{9}$$

- Rolling skewness:

$$\text{Skew} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2)^{3/2}} \tag{10}$$

Table 1
Notations and definitions for all mathematical symbols, variables, and notation used throughout Section 2.

Symbol	Definition
D	Dataset composed of N instances
N	Total number of instances in the dataset
M	Number of feature engineering functions
t_i	Timestamp associated with the i-th instance
y_i	Target variable associated with the i-th instance
$\mathbf{F}'_{i,j}$	Feature subset generated from t_i using the j-th FE function
$\mathbf{F}'_i$	Complete feature vector for the i-th instance
$\bigoplus$	Concatenation operation
$\mathcal{M}$	Predictive model trained within an AutoML framework
F_{TIME}	Temporal feature extraction function
F_{CYCLICAL}	Cyclical feature extraction function
F_{LAGS}	Lag feature extraction function
F_{STATS}	Rolling window statistical feature extraction function
τ	Kendall's tau correlation coefficient
C, D, T, U	Concordant pairs, discordant pairs, ties in first/second variable
n	Size of the rolling window
x_i	Values within the current rolling window
$\bar{x}$	Rolling mean
s	Rolling standard deviation

where n is the size of the rolling window, and x_i are the values within the current window. By calculating these statistics for each retained window size, the algorithm captures different aspects of temporal dynamics in the energy data.

2.3. Considerations

The design of the proposed FE method adheres to the following considerations:

- Computational efficiency: the algorithm handles large datasets by limiting the number of generated lags and window sizes based on statistical significance.
- Avoidance of data leakage: features at time t_i are computed using only data available up to t_i, preventing look-ahead bias.
- Interpretability: the generated features have the potential to aid in model explainability. For instance, lag features capture direct historical dependencies, while rolling statistics quantify concepts such as trend (mean), volatility (standard deviation), extremes (max/min), and distribution shape (skewness/kurtosis) over specific time windows, enabling domain experts to understand how each feature contributes to the model's predictions. In other words, rolling means can reveal baseline consumption patterns, standard deviations can identify periods of irregular usage, and lag features can capture recurring behaviours such as daily routines or equipment cycling patterns, allowing energy engineers to understand the factors driving consumption in their systems. Table 1 shows the notations and definitions for all mathematical symbols, variables, and notations used.

3. Experimental design

This section outlines the experimental design adopted in this work, including the dataset used in Subsection 3.1, benchmarking methods in Subsection 3.2, detailed experimental procedure in Subsection 3.3, and lastly the evaluation criteria and statistical tests in Subsection 3.4.

3.1. Datasets

Due to the limited availability of standardised benchmark datasets for ECF using AutoML, this study evaluated the proposed FE method using real-world datasets from related research and repositories. Eighteen energy datasets were employed, spanning a range of energy domains, including residential buildings (e.g., home appliances), industrial and manufacturing facilities (e.g., steel factory), food and drink cold storage facilities, urban or regional energy use, and renewable energy sources (e.g., wind turbines). Table 2 and Fig. 3 provide further details about the datasets used. These datasets provide a comprehensive representation of different energy systems, encompassing both univariate and multivariate data with varying sample sizes and temporal resolutions.

3.2. Benchmarking feature engineering methods

To demonstrate the effectiveness of the proposed FE method, this study compares key FE methods from the literature, namely TS and FT, as discussed in Subsection 1.1.1, with the proposed method. These FE methods were selected as they represent leading open-source solutions that have been successfully applied to a variety of forecasting problems, thus providing a robust benchmark across different FE approaches. While energy-specific FE approaches exist in the literature [53], these are predominantly manual, expert-driven methodologies lacking automated elements to minimise domain knowledge reliance, streamline ML model development, and improve AutoML performance. Therefore, TS and FT serve as the most comparable automated FE baselines available for systematic comparison, as they share AutoEnergy's automated nature while offering extensive documentation and accessible implementations that ensure reproducible and methodologically sound comparative evaluation. This combination of automation capabilities and implementation accessibility makes them the most appropriate benchmarks for rigorous evaluation against the proposed FE method. It is noteworthy that the TS method offers three primary configurations for FE: a) MinimalFCParameters (TSMin), which includes a limited number of features suitable for quick tests; b) EfficientFCParameters (TSEff), which comprises all features generated by the TS method except those marked with the *"high_comp_cost"* attributes[3]; and c) ComprehensiveFCParameters (TSComp), which includes all generated features by the TS method.

In this work, AutoGluon [16] was selected as the AutoML framework due to its superior performance in previous research [39], where it outperformed other AutoML methods. This selection is justified by: A) logical scientific progression that builds upon previously established findings and maintains focus alignment on comprehensive evaluation of different FE methods against the proposed FE method within the same AutoML framework; B) methodological complexity arising from using multiple AutoML frameworks with different architectures, optimisation strategies, and ensemble approaches, which may introduce confounding variables that obscure the true impact of automated FE contributions (i.e., results performance variations may reflect differences between the AutoML frameworks rather than the contribution of the FE methods); and C) computational feasibility challenges associated with evaluating multiple AutoML frameworks across eighteen diverse datasets with multiple FE pipelines. Nevertheless, TabPFN [40], a state-of-the-art AutoML framework, was incorporated as an additional experiment within the study design to further assess the robustness and potential generalisability of the proposed FE method. In this experiment, all FE methods and models were trained using Python 3.11.[4]

3.3. Experimental procedure

The experimental procedure and steps conducted in this work, as shown in Fig. 4, are as follows:

Stage One: Partitioning each dataset into two segments: 80 % was allocated to training the AutoGluon models (i.e., train dataset), while the remaining 20 % was reserved for assessing model performance

[3] Features with high computational costs, see: https://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature_extraction.html#tsfresh.feature_extraction.settings.ComprehensiveFCParameters.

[4] All computational experiments were performed on a system featuring an x86_64 architecture, 64 GB of RAM, and dual Quadro RTX 5000 GPUs.

Table 2
Overview of the eighteen energy datasets used in this study. Column *Dataset* lists the dataset name. Column *Description* summarises the energy system and domain. Column *Type* indicates whether the dataset is univariate (Uni) or multivariate (Multi(k), where k is the number of additional features, e.g., weather variables). Column *N* gives the total number of samples. Column *Resolution* shows the sampling interval (m = minutes, h = hours). Column *Total Duration (days)* gives the total time span covered by the dataset in whole days. Column *Ref.* cites the source.

Dataset	Description	Type	N	Resolution	Total Duration (days)	Ref.
AEP	Power consumption by American Electric Power	Uni	121,269	1h	5053	[43]
Appliances	Energy consumption data from home appliances	Multi (27)	19,735	10m	138	[44]
CAISO_Elec	Electricity load by California ISO	Uni	26,304	1h	1096	[45]
COMED	Energy usage by Commonwealth Edison	Uni	57,735	1h	2406	[43]
DEOK	Energy consumption by Duke Energy OH/KY	Uni	57,735	1h	2406	[43]
EKPC	Energy usage by East Kentucky Power	Uni	45,330	1h	1889	[43]
FDCS_1	Energy usage by Food and Drinks Cold Storage	Multi (9)	1944	1h	81	[46]
FDCS_2	Energy usage by Food and Drinks Cold Storage	Multi (9)	2472	1h	103	[46]
FE	Power consumption by FirstEnergy	Uni	62,870	1h	2620	[43]
NI	Northern Illinois Hub energy usage	Uni	58,450	1h	2436	[43]
PJME	Energy use in PJM East Region	Uni	145,362	1h	6057	[43]
PJMW	Power consumption in PJM West	Uni	143,202	1h	5967	[43]
Solar_Home	Ausgrid solar home electricity	Uni	17,568	30m	366	[47]
Steel	Energy usage by AEWOO Steel Co.	Multi (8)	8760	1h	365	[48]
TCity	Power usage in Tetouan	Multi (3)	52,416	10m	364	[49]
UNICON	La Trobe University electricity consumption	Multi (4)	8663	1h	361	[50]
Victoria	Electricity demand of 5 AU states	Multi (1)	20,352	1h	848	[51]
WindT	Wind turbine SCADA systems	Multi (2)	50,530	10m	351	[52]

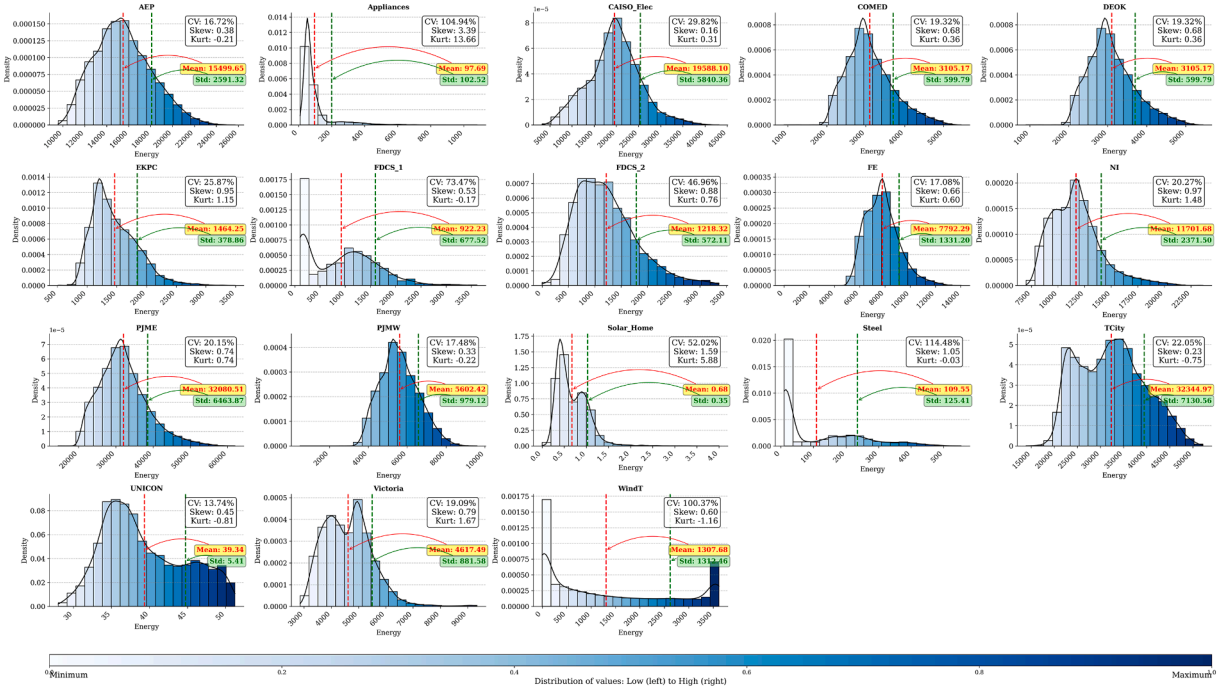


Fig. 3. The energy datasets used in this study are depicted in histograms of the target variable with colour-coded bars representing normalised bin positions. Red and green dashed lines indicate the mean and standard deviation, respectively. Annotated statistics include coefficient of variation (CV: relative variability), skewness (distribution asymmetry), and kurtosis (tailedness). The colour gradient in the histogram bars represents the distribution of values from low (left) to high (right). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

on newly, previously unseen data (i.e., test dataset). This train-test validation method was selected due to its simplicity and computational efficiency, particularly given the extensive experimental design involving 18 datasets, each subjected to five FE methods. Moreover, in AutoGluon, cross-validation is inherently integrated, eliminating the need for a separate validation set, as models are trained on multiple folds of data, with each instance evaluated against the hold-out fold that was not used during training to generate out-of-fold predictions, which are then used to calculate the final cross-validation score [16]. It is also important to note that the data were not shuffled, as the energy consumption data exhibit temporal patterns, making it essential to preserve the chronological order of the timestamps.

Stage Two: This stage involves two steps. In the first step, AutoGluon models were trained without FE (No.Feat). This serves as a baseline scenario to assess AutoGluon's forecasting capabilities with minimal input, utilising the *TimeSeriesPredictor*. This class, as shown in Fig. 4, includes different forecasting methods, including statistical-based, neural network-based, hybrid, and ensemble methods. This deliberately simplified configuration is designed to replicate AutoML's performance under conditions that simulate: (a) a worst-case scenario (i.e., the absence of domain-specific FE knowledge), and (b) an initial testing phase where

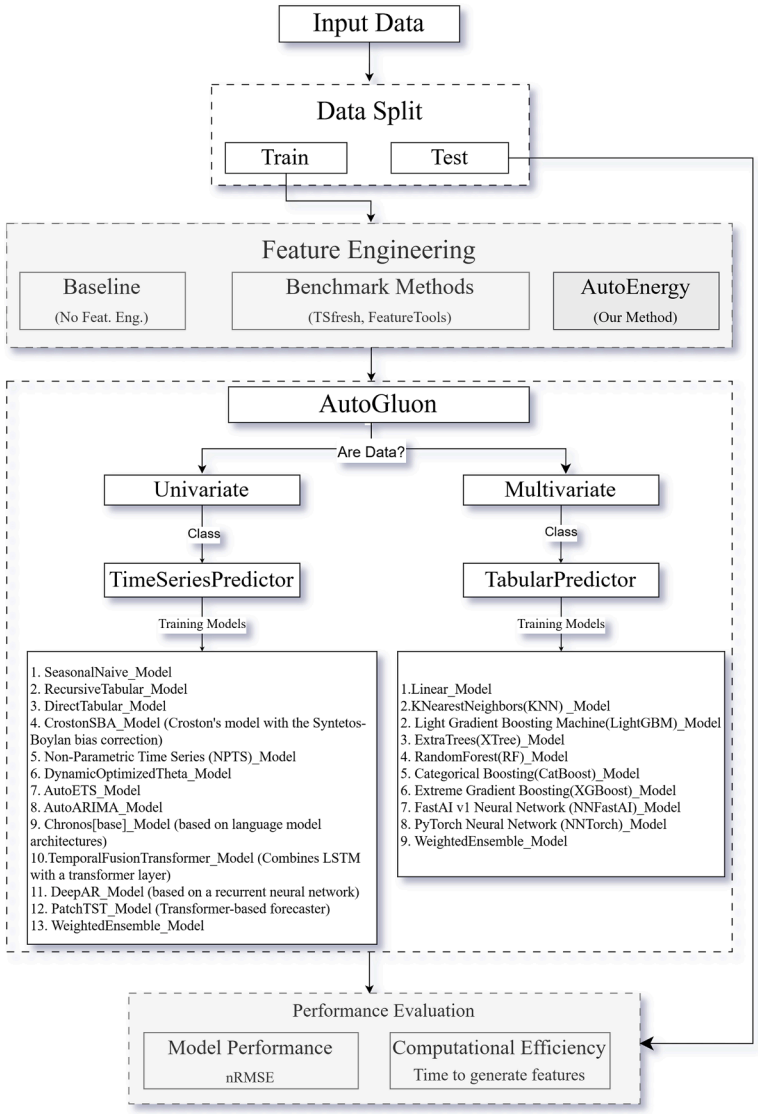


Fig. 4. Experimental design. See Subsection 3.3 for a detailed explanation of the experimental setup.

the model learns with limited data. In the second step, both benchmarking FE methods, as described in Subsection 3.2, and the proposed FE method as described in Subsection 2.2, were applied to all datasets as a preprocessing step before training the AutoGluon models with the *TabularPredictor* class.

Stage Three: In this final stage of the experimental procedure, the performance of all trained AutoGluon models was evaluated using the test sets and the evaluation metrics outlined in Subsection 3.4.

Special Case TabPFN: While TabPFN is highly effective [40], it operates within specific architectural constraints, exhibiting superior performance exclusively on datasets containing up to 10,000 samples and 500 features. Given these limitations, the experimental procedure (see Fig. 5) comparing TabPFN with versus without the proposed FE algorithm was confined to eight datasets that satisfied these specifications, as shown in Table 4.

3.4. Evaluation metrics and statistical tests

In this work, the Normalised Root Mean Squared Error (nRMSE) is used to evaluate and compare the proposed FE method versus the benchmarking methods described in Subsection 3.2. This quantitative metric, as depicted in Eq. 11, offers a standardised measure of error magnitude. Its normalised nature facilitates quantitative performance comparisons across diverse datasets, regardless of the varying scales of their respective y_i values.

$$nRMSE = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}}{y_{\max} - y_{\min}} \tag{11}$$

where y_i is the observed or actual values in the dataset; $\hat{y}_i$ is the values predicted by the model; n: the total number of observations in the dataset, $y_{\max}$ and $y_{\min}$ are The highest and lowest observed values in the dataset, respectively.

In addition to nRMSE, the study evaluates the computational efficiency of each FE method by measuring their processing times. This assessment is crucial for understanding the practical applicability of these methods in real-world scenarios, where computational resources may be limited, rapid processing is critical, or when dealing with large datasets. By comparing the execution times across different FE methods, the study provides valuable insights into their scalability and performance trade-offs. In other words, this evaluation helps identify methods that achieve low forecasting errors while maintaining computational cost, thereby offering a comprehensive evaluation of each method's overall effectiveness and efficiency.

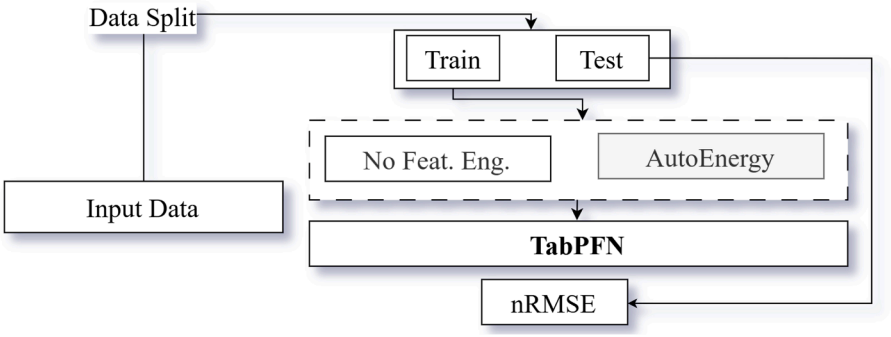


Fig. 5. Experimental design with TabPFN. See "Special Case TabPFN" in Subsection 3.3 for a detailed explanation of this particular case.

To support the experimental findings statistically, non-parametric hypothesis tests were employed to identify significant differences between the methods [54]. The Friedman Aligned-ranks test [55] is employed to examine the presence of statistically significant differences among the FE methods, with the significance level set at $\alpha = 0.05$. Following this, the Bonferroni post hoc procedure is applied to determine which specific FE methods have significant differences in the one-versus-many $(1 * n)$ comparisons performed, where one method (i.e., AutoEnergy) is compared against n other FE methods. Furthermore, the Wilcoxon Signed-Rank test [56,57] was employed with $\alpha = 0.05$ to further investigate potential differences between pairs of FE methods that were not identified as significant by the previous tests, ensuring a comprehensive statistical analysis.[5]

4. Results and analysis

This section presents an analysis and discussion of the experimental results. Subsection 4.1 discusses the overall impact of AutoEnergy on the performance of AutoGluon and TabPFN. Subsequently, Subsection 4.2 provides a comparative evaluation of the proposed FE method against benchmark approaches, considering both forecasting errors and processing time. This is followed by Subsection 4.3, which discusses computational efficiency and sensitivity analysis, and finally, Subsection 4.4 presents the feature importance analysis and statistical testing.

4.1. Impact of AutoEnergy on AutoML predictivity

The performance comparison between AutoEnergy and benchmark FE methods across eighteen test sets is presented in Table 3 and Fig. 6. Despite the diversity in dataset characteristics across different energy settings and environments (detailed in Table 2 and Fig. 3), AutoEnergy consistently enhanced AutoGluon's performance compared to the baseline (i.e., without FE). The results show an average nRMSE reduction of 83.22 %, with improvements (i.e., reduction in forecasting errors) ranging from 28.22 % for the Appliances dataset to 98.39 % for the PJME dataset compared to the baseline.

This improvement is complemented by a 53.69 % enhancement in prediction stability, as indicated by the lower standard deviation (0.0358 vs 0.0773). This shows the contribution of the proposed FE method to improving AutoGluon's predictive accuracy. Moreover, the results in Table 4 indicate that automated FE can further enhance the performance of state-of-the-art AutoML methods such as TabPFN [40]. In six of the eight datasets that satisfy TabPFN's constraints, integrating AutoEnergy with TabPFN yielded the lowest forecasting errors, outperforming FT, TSEff, and TSMin FE methods. The only exceptions were the Steel and Victoria datasets, where TSMin and FT achieved lower errors, respectively. When considered alongside the AutoGluon results in Table 3 and Fig. 6, these findings indicate that AutoEnergy can potentially generalise across diverse AutoML frameworks, which differ in architecture, model-selection mechanisms, hyper-parameter optimisation strategies, and ensemble approaches.

The consistent improvements across diverse energy systems may suggest that underlying consumption patterns share common characteristics that AutoEnergy's proposed functions, explained in Subsection 2.2, can effectively capture and transform raw energy data into useful predictive features for training AutoML models. Such findings demonstrate AutoEnergy's effectiveness in automating the FE process for ECF problems without requiring domain expertise. Meanwhile, the limited effectiveness of AutoML frameworks in specialised domains such as ECF can be attributed to their general-purpose design, which prioritises broad applicability across ML tasks at the expense of domain-specific optimisations that FE methods, such as AutoEnergy, can provide.

4.2. Comparison with benchmark methods

The proposed AutoEnergy FE method demonstrated an overall superior performance across test sets compared to the benchmarking methods. This is evident by achieving a mean nRMSE of 0.0338 compared to FT (0.042), TSMin (0.1918), and TSEff (0.1535), as presented in Table 3. This represents average reductions in forecasting errors of 19.52 %, 82.38 %, and 77.98 %, respectively, over these methods. In terms of computational efficiency (i.e., time needed for FE), AutoEnergy achieved a mean processing time of 471.94 seconds, which was 1.31x faster (23.72 % improvement) than FT (618.68s) and 4.41x faster (77.31 % improvement) than TSEff (2079.81s), though 1.58x slower (57.78 % slower) than TSMin (299.12s). While the latter showed faster processing time, AutoEnergy reduced forecasting errors by 82.38 % on average compared to TSMin.

As shown in Table 3, processing time varies substantially across datasets, primarily driven by dataset size (i.e., the number of samples). Table 2 provides an overview of the characteristics of the eighteen energy datasets, including their sample sizes. Notably, larger datasets require significantly more intensive computations during automated FE. This relationship is evident in specific examples: high-sample datasets such as PJME (N = 145,362; AutoEnergy = 2,061s; FT = 2,492.4s; TSEff = 4,663.4s) exhibit considerably longer processing times due to the computational scaling of automated FE operations, whereas smaller datasets such as FDCS_1 (N = 1,944; AutoEnergy = 2s; FT = 14.6s; TSEff = 2s) are processed much faster. Importantly, this pattern is not unique to AutoEnergy but is consistently observed across all benchmarking methods. This observation is further supported by Fig. 7, which shows that all FE methods display strong positive Pearson correlations ($r = 0.93$–0.97) between dataset size and processing time. This systematic relationship confirms that the computational overhead is intrinsic to automated FE, rather than a limitation specific to AutoEnergy.

[5] For more information on Machine Learning statistical tests, visit SCI2S's website on Statistical Inference in Computational Intelligence and Data Mining https://sci2s.ugr.es/sicidm

Table 3
Results obtained on all the test sets (N = 18). nRMSE values and processing times in seconds (lower is better) for the proposed AutoEnergy method compared to other FE methods. Bold values highlight the lowest nRMSE and fastest processing time for each dataset. nRMSE gives more weight to larger errors due to the squaring operation, making it more sensitive to outliers. It is normalised by the range (max-min) of the target variable, allowing for fair comparison of forecasting error across datasets with different scales. In Appendices A and A.1 provides additional results of RMSE, MAE, MAPE and R^2. Note that the TScomp configuration is not reported, as it proved computationally infeasible in our experiments (runs crashed after several hours of running due to excessive memory use). We include this note to highlight the practical computational limits of exhaustive FE approaches (e.g., TScomp) in real-world energy forecasting and to underscore the importance of considering computational feasibility and practicality in future work.

Dataset	FE Methods									
	AutoEnergy		FT		TSMin		TSEff		No.Feat	
	nRMSE	Time	nRMSE	Time	nRMSE	Time	nRMSE	Time	nRMSE	Time
AEP	**0.0096**	1585.90	0.0113	1576.84	0.1659	**636.50**	0.1737	4280.70	0.1856	0.00
Appliances	0.0870	**44.33**	**0.0824**	382.34	0.1831	177.06	0.1529	1411.01	0.1212	0.00
CAISO_Elec	**0.0163**	**88.30**	0.0232	211.91	0.2495	211.06	0.1074	1468.03	0.2036	0.00
COMED	**0.0150**	372.41	0.0151	558.55	0.1535	**352.04**	0.1579	2506.55	0.1905	0.00
DEOK	**0.0150**	**360.22**	0.0151	561.21	0.1535	361.33	0.1579	2502.54	0.1905	0.00
EKPC	**0.0135**	**226.76**	0.0155	416.53	0.1931	295.27	0.2208	2089.80	0.1499	0.00
FDCS_1	**0.0861**	**2.03**	0.0914	14.65	0.1893	22.80	0.0996	51.06	0.2711	0.00
FDCS_2	**0.1173**	**2.79**	0.1266	18.62	0.1944	25.21	0.1786	66.12	0.2571	0.00
FE	**0.0097**	427.77	0.0119	625.56	0.1719	**262.22**	0.1766	2035.33	0.1513	0.00
NI	**0.0070**	**376.61**	0.0083	564.64	0.1732	425.00	0.1693	3111.96	0.2074	0.00
PJME	**0.0059**	2061.00	0.0068	2492.46	0.2104	**666.59**	0.1979	4663.42	0.3675	0.00
PJMW	**0.0100**	2205.06	0.0117	2321.66	0.1834	**676.22**	0.1784	4304.59	0.1682	0.00
Solar_Home	**0.0808**	**39.66**	0.0860	133.49	0.2065	131.43	0.1673	853.15	0.1507	0.00
Steel	**0.0104**	**13.95**	0.0122	62.22	0.0122	49.94	0.0132	266.38	0.2040	0.00
TCity	**0.0123**	**320.00**	0.0922	506.37	0.2478	439.00	0.1759	3276.16	0.2760	0.00
UNICON	**0.0408**	**13.23**	0.0421	61.29	0.3090	50.78	0.2137	263.87	0.2875	0.00
Victoria	**0.0100**	**55.47**	0.0110	158.34	0.2911	170.75	0.1138	1088.43	0.1877	0.00
WindT	**0.0614**	**299.46**	0.0930	469.73	0.1647	431.06	0.1084	3197.56	0.4115	0.00
Mean	**0.0338**	471.94	0.0420	618.68	0.1918	**299.12**	0.1535	2079.81	0.2212	0.0
Std. Dev.	**0.0358**	705.75	0.0404	744.73	0.0633	**216.15**	0.0495	1507.69	0.0773	0.0

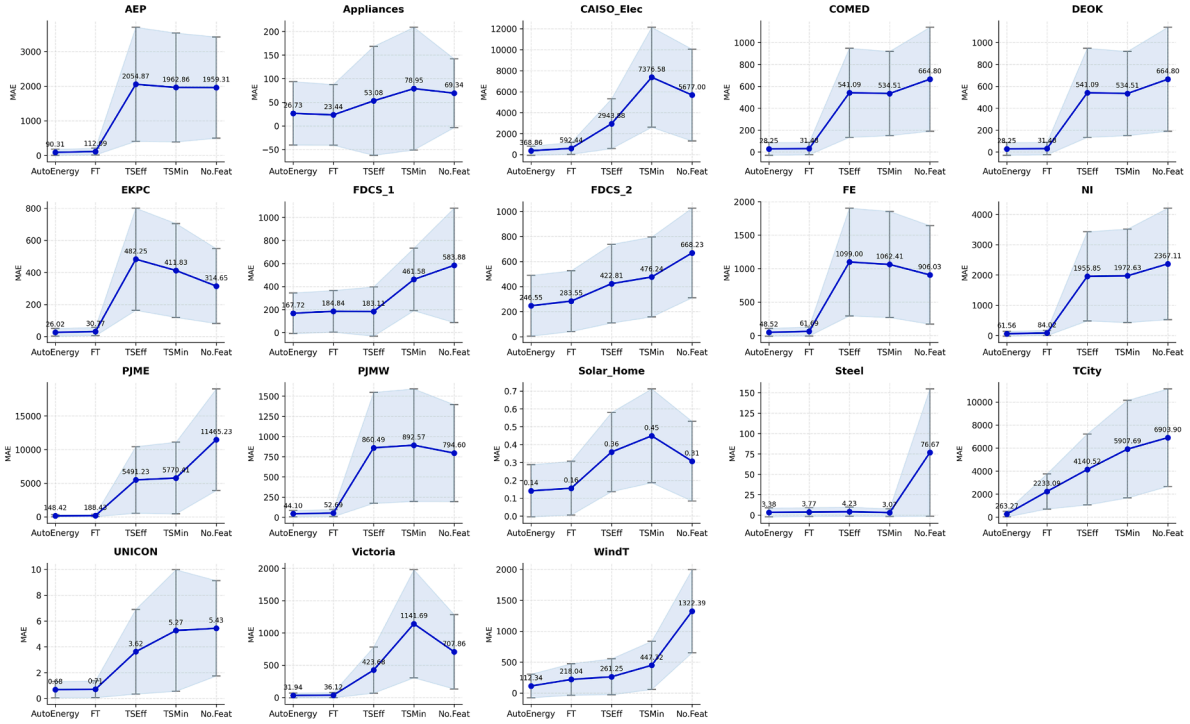


Fig. 6. Error analysis across test datasets. Each subplot shows the mean absolute error (MAE), computed as MAE = $\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$, between true and predicted values. The black line represents the MAE (lower is better), while the shaded areas and error bars indicate error variability. MAE provides an average magnitude of prediction error, treating all deviations equally and offering insight into prediction consistency that complements the scale-sensitive nRMSE in Table 3.

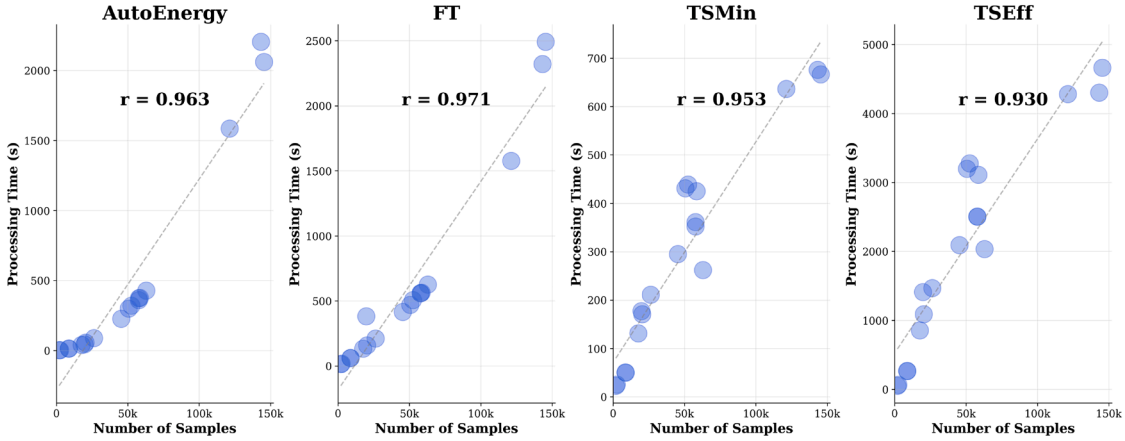


Fig. 7. Pearson correlation between dataset size (i.e., number of samples) and processing time for each method across all eighteen energy datasets. Markers represent individual datasets.

Table 4
nRMSE for TabPFN without FE versus with FE methods across datasets that satisfy the TabPFN constraints ($N \leq 10{,}000$, $D \leq 500$). Lower values are better. Boldface indicates the best results. In Appendices A and A.2 provides additional results of RMSE, MAE, MAPE and R^2.

Dataset	TabPFN	AutoEnergy	FT	TSEff	TSMin
Appliances	0.1412	**0.0710**	0.0712	0.1153	0.1497
FDCS_1	0.1549	**0.0718**	0.0778	0.0984	0.1822
FDCS_2	0.1504	**0.1128**	0.1278	0.1514	0.1620
Steel	0.0080	0.0081	0.0082	0.0135	**0.0076**
TCity	0.2691	**0.0184**	0.0212	0.0701	0.2299
UNICON	0.2226	**0.0391**	0.0422	0.2026	0.1963
Victoria	0.1496	0.0158	**0.0146**	0.0994	0.1453
WindT	0.1792	**0.0841**	0.0974	0.1595	0.1269

AutoEnergy's performance advantage is less pronounced on datasets with skewed distributions that also include exogenous features (Appliances with 27 features and Steel with 8 features, see Table 2). These additional features, such as weather-related, already capture useful predictive information, thereby reducing the relative improvement that AutoEnergy's temporal FE can provide. Furthermore, the skewed distributions in these datasets may suggest that consumption patterns are dominated by extreme values or specific operational modes, where exogenous variables are likely stronger predictors than temporal patterns. When datasets already contain informative variables that directly influence consumption, the marginal benefit of AutoEnergy is likely to be diminished, though AutoEnergy maintains competitive performance, achieving the lowest nRMSE for the Steel dataset while remaining competitive for the Appliances dataset, where FT performs best.

Table 5 reveals that capping the nested window length at one-third of each series offers a favourable accuracy-cost trade-off for ECF across the eighteen datasets, as evidenced by its lowest average nRMSE of 0.1758 and second-fastest processing time overall. It achieves the minimum error in 6 of 18 datasets, versus 1, 4, and 7 for the quarter, full-length and half caps, respectively, yet the latter was considerably slower in processing time compared to the one-third cap. Although such findings remain data-dependent, with some datasets showing sub-optimal results under the one-third constraint, this configuration consistently delivers near-optimal performance across diverse energy datasets representing different energy systems, offering a balance between forecasting accuracy and computational efficiency for automated FE in ECF tasks; it thus minimises the need for expert input and manual intervention, enabling fully automated, end-to-end ECF systems.

4.3. Computational efficiency and sensitivity analysis

While computational efficiency claims are often dataset-dependent, AutoEnergy shows promising results in scalability relative to other existing well-established FE methods. For instance, for the two datasets exceeding 140,000 samples (PJME = 145,362; PJMW = 143,202 - see Table 2 for dataset characteristics), AutoEnergy maintained computational feasibility across those cases. More precisely, it achieved faster processing times (2,061s and 2,205s, respectively) compared to FT (2,492s and 2,322s) and significantly outperformed TSEff (4,663s and 4,305s). However, more rigorous scalability evaluation on very large datasets (e.g., millions of samples) and streaming data scenarios is therefore recommended for future work to fully establish the method's scalability characteristics. It is worth noting, though, that automated FE methods such as AutoEnergy are particularly beneficial for smaller datasets, where raw data often lacks sufficient informative patterns, whereas very large datasets may inherently contain rich relationships that could reduce the relative necessity for extensive FE.

As real-world datasets are often small due to cost and privacy concerns [5,6], AutoEnergy's performance on small datasets is particularly noteworthy. For the three smallest datasets (FDCS_1: 1,944 samples, FDCS_2: 2,472 samples, and UNICON: 8,663 samples), AutoEnergy reduces nRMSE error to an average of 0.0814, showing significant improvements compared to baseline (0.2719, 70.06 % lower error), TSMin (0.2309, 64.75 % lower error), TSEff (0.1640, 50.36 % lower error), and FT (0.0867, 6.11 % lower error). Regarding the processing time for these datasets, AutoEnergy generates features in 6.02 seconds on average, which is 21.1x faster than TSEff (127.02s), 5.5x faster than TSMin (32.93s), and 5.2x faster than FT (31.52s). Although this observation is data-dependent, AutoEnergy shows better performance on small datasets, improving both forecasting accuracy and computational efficiency compared to the benchmarking methods. This superior performance could be attributed to AutoEnergy being specifically designed for ECF problems, compared to the benchmarking FE methods.

4.4. Analysis of feature importance and statistical testing

The importance and distribution of the top 20 features across the examined energy datasets, computed by AutoGluon, are shown in Fig. 8, where feature importance is calculated using permutation importance [16]. It measures the decrease in model performance when the values of a specific feature are randomly shuffled, thereby quantifying each feature's contribution to predictive accuracy in a model-agnostic

Table 5
Sensitivity analysis results (nRMSE and processing time) for the time series fraction constraint defining the candidate range for nested window statistical feature generation, where only features with statistically significant Kendall's tau correlation within this range are retained to generate the final model input features. The examined lengths are: quarter (1_4), one-third (1_3), half (1_2), and full sequence length (1). This experiment evaluates the impact of varying maximum rolling window fractions on predictive model performance and efficiency, using solely the nested window features as inputs to isolate their standalone contributions and inform optimal parameter selection. See Subsubsection 2.2.2 for more details.

Dataset	Maximum Length							
	1/4		1/3		1/2		1	
	nRMSE	Time	nRMSE	Time	nRMSE	Time	nRMSE	Time
AEP	0.2241	1498.01	0.2229	1634.39	**0.2227**	1816.42	0.3124	2110.72
Appliances	0.2492	40.28	0.2447	44.35	0.2966	50.82	**0.2429**	63.48
CAISO_Elec	0.2002	85.50	0.1776	92.99	**0.1669**	104.12	0.1849	125.12
COMED	0.1519	367.54	0.1513	388.01	**0.1498**	439.48	0.1826	517.67
DEOK	0.1508	359.59	0.1501	384.05	**0.1465**	432.18	0.1811	511.69
EKPC	0.1625	224.61	**0.1560**	243.71	0.1735	279.53	0.2243	326.94
FDCS_1	0.1723	1.73	**0.1626**	1.85	0.1729	2.10	0.1690	2.68
FDCS_2	0.1558	2.32	0.1545	2.46	**0.1507**	2.79	0.1509	3.55
FE	0.2009	432.82	0.2180	460.15	**0.1674**	524.25	0.1708	617.90
NI	0.1906	378.06	0.1977	412.28	0.2182	464.61	**0.1812**	537.66
PJME	0.2289	2030.82	0.2289	2173.77	0.2289	2427.23	**0.1883**	2808.38
PJMW	0.1876	1826.87	**0.1863**	1987.15	0.1871	2219.01	0.1887	2570.72
Solar_Home	0.1766	38.03	0.1892	41.23	0.2003	46.85	**0.1660**	57.38
Steel	0.0117	13.54	**0.0112**	14.60	0.0121	16.54	0.0122	20.39
TCity	0.2574	275.11	**0.2078**	297.74	0.2349	332.94	0.2088	389.78
UNICON	0.2458	12.90	0.2458	13.90	**0.2133**	15.73	0.2594	19.56
Victoria	**0.1377**	53.73	0.1395	57.58	0.1711	65.52	0.1680	79.12
WindT	0.1294	247.24	**0.1217**	267.58	0.1348	300.13	0.1483	355.55

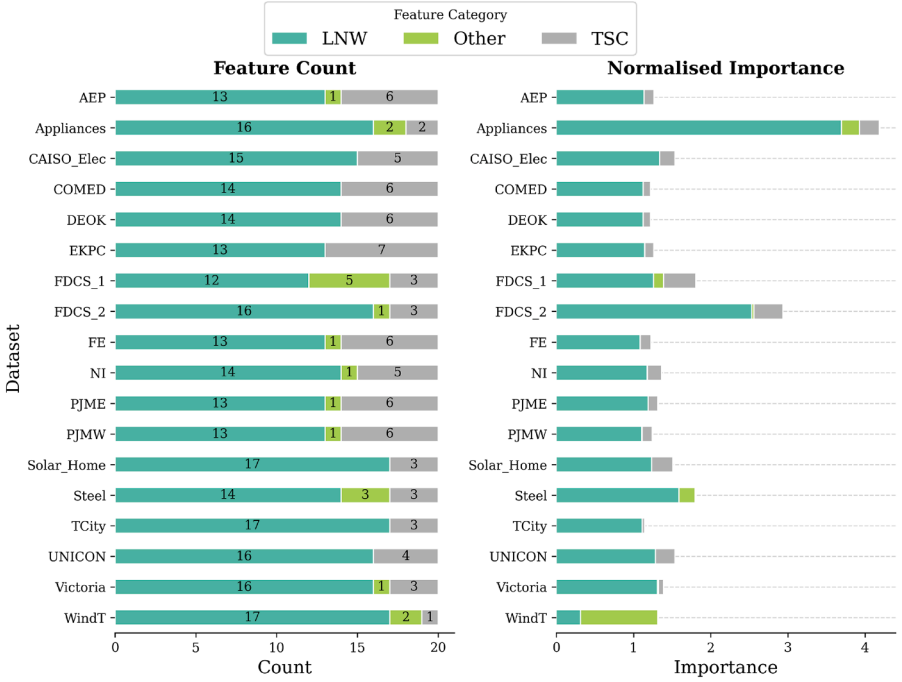
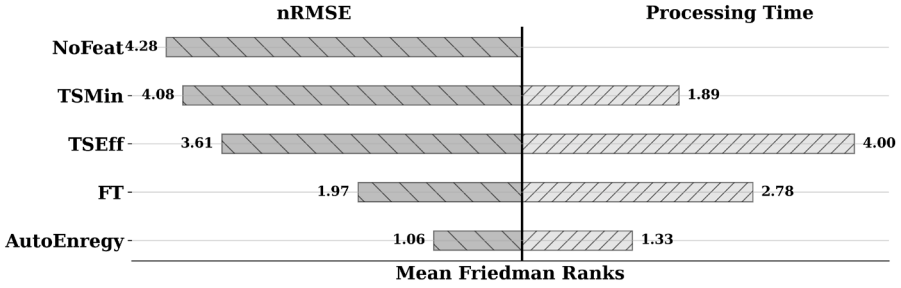


Fig. 8. Comparison of the top 20 most important features across datasets, grouped into three categories: Temporal, Sine and Cosine Transform-Derived Features (TSC - generated using Algorithm 1), Lags and Nested Window Features (LNW - generated using Algorithm 2), and Other (e.g., weather features). The left panel displays the feature count with numerical annotations, while the right panel shows the cumulative normalised importance (as a proportion of total importance) of these categories. This visualisation emphasises the relative significance of each feature category within the top 20 features of each dataset. LNW features dominate the importance analysis, likely due to their ability to capture essential temporal dependencies in energy consumption patterns. Specifically, lag features model historical influences, such as thermal inertia [58], where past usage affects near-future demand due to gradual changes. Meanwhile, the effectiveness of nested window features suggests that energy consumption operates over multiple temporal scales: short-term windows capture immediate operational fluctuations, while longer-term windows reflect underlying trends and seasonal variations. Together, these characteristics enhance the predictive power of LNW features for ECF problems. Detailed feature importance analysis can be found in the supplementary material.



Related-Samples Friedman's Two-Way Analysis of Variance by Ranks Summary

Metric	Test Statistic	Degree Of Freedom	Asymptotic Sig.(2-sided test)
nRMSE	57.883	4	<.001
Processing Time	43.867	3	<.001

Fig. 9. Comparisons between the proposed FE algorithm and other FE methods using the Friedman Average Ranking (lower is better).

manner [6]. The results reveal that features generated by AutoEnergy are the most predictive, even though eight of the eighteen datasets have additional features such as weather attributes. In particular, statistically significant lags and nested window features, as explained in Subsection 2.2.2, appear to be the dominant predictors contributing to model performance. This may suggest that lag features effectively capture historical influences, such as thermal inertia in buildings [58], where past consumption directly affects immediate future demand due to gradual changes in heating or cooling systems, revealing that energy behaviours may exhibit persistence and path-dependency rather than abrupt shifts. Meanwhile, the effectiveness of nested window features may indicate that energy consumption operates across multiple temporal scales (i.e., short-term windows capture immediate operational fluctuations while longer-term windows encode baseline trends and seasonal adjustments). Because these temporally explicit predictors may map directly to operational concepts such as "daily average demand," practitioners can trace a model's output back to concrete, intuitive drivers, thereby markedly enhancing interpretability. In spite of that, feature importance is often dataset-dependent, so this observation may vary across different datasets and settings. Further feature-importance scores and details, including eighteen tables (one for each dataset), are provided in the supplementary material.

Beyond feature-importance rankings, the engineering of temporal, cyclical, lag, and nested features by AutoEnergy, as explained in Subsection 2.2, enhances interpretability and transparency in several useful ways. First, sinusoidal time encodings (sine/cosine of hour-of-day and day-of-week) map directly to familiar operational cycles; their effects, therefore, can be read as "daily/weekly seasonality" rather than opaque latent factors. Second, lagged and windowed statistics (e.g., load at $t - 24h$ or the previous 24-hour mean) preserve units and time scales, making the direction and magnitude of their influence predictable (higher recent demand plausibly raises near-term forecasts), which may support counterfactual "what-if" reasoning and analysis. Third, AutoEnergy's design is a rule-based (heuristic) search algorithm that applies explicit, auditable transformations (temporal shifts, rolling aggregates, and trigonometric projections), keeping provenance clear and making each generated feature human-readable, traceable, and reproducible, thereby strengthening the interpretability advantage of the method. Finally, because the generated features are emitted as standard tabular columns with fixed names and units, they can be integrated directly into AutoML frameworks that provide built-in feature-importance tools (e.g., AutoGluon), enabling non-experts to see how engineered features influence forecasts and thereby preserving interpretability in fully automated, end-to-end ECF workflows.

[6] For more information, see https://auto.gluon.ai/dev/api/autogluon.tabular.TabularPredictor.feature_importance.html and https://auto.gluon.ai/dev/api/autogluon.timeseries.TimeSeriesPredictor.feature_importance.html.

To evaluate the statistical significance of performance differences, both the Friedman and the Wilcoxon tests were conducted as detailed in Subsection 3.4. Friedman's test revealed significant differences among the methods ($p < 0.001$), with AutoEnergy achieving the best mean rank for nRMSE (1.06), followed by FT (1.97), TSEff (3.61), TSMin (4.08), and NoFeat (4.28), as presented in Fig. 9. Post hoc analysis using Bonferroni correction demonstrated that AutoEnergy significantly outperformed TSEff, TSMin, and NoFeat in terms of nRMSE (all $p < 0.001$), as shown in Table 6. While the Friedman test with Bonferroni adjustment showed no significant difference between AutoEnergy and FT ($p = 0.820$), the pairwise Wilcoxon test, which specifically examines the direct comparison between these two methods, revealed a statistically significant improvement ($p = 0.001$) as depicted in Table 7. The discrepancy may stem from the Friedman test's omnibus design and conservative Bonferroni adjustment: the original AutoEnergy-FT comparison was non-significant pre-adjustment ($p = .082$), and the correction further attenuated this effect ($p = .820$), whereas the Wilcoxon test, focused solely on this pairwise comparison without multiple-testing penalties, thereby detected a statistically significant difference ($p = .001$) that the more conservative post hoc analysis may have overlooked.

In terms of computational efficiency, as presented in Fig. 9, Friedman's test ranked AutoEnergy first (1.33), with post hoc tests showing significant improvements over FT ($p = 0.005$) and TSEff ($p < 0.001$), while the difference with TSMin was not statistically significant ($p = 1.000$), a finding also confirmed by the Wilcoxon test ($p = 0.446$). Although TSMin achieved better processing times, AutoEnergy's superiority in accuracy, demonstrated earlier through its substantial reduction in forecasting errors, remained noticeable. These results provided statistical evidence supporting AutoEnergy's favourable balance between accuracy and computational efficiency for ECF problems compared to the benchmarking methods.

Table 6
Friedman test results for nRMSE and processing time comparisons.

Comparison	Test Statistic	Std. Error	Std. Test Statistic	Sig.	Adj. Sig.[a]
nRMSE					
AutoEnergy-FT	-0.917	0.527	-1.739	0.082	0.820
AutoEnergy-TSEff	-2.556	0.527	-4.849	<0.001	0.000
AutoEnergy-TSMin	-3.028	0.527	-5.745	<0.001	0.000
AutoEnergy-NoFeat	-3.222	0.527	-6.114	<0.001	0.000
Processing Time					
AutoEnergy-TSMin	-0.556	0.430	-1.291	0.197	1.000
AutoEnergy-FT	-1.444	0.430	-3.357	<0.001	0.005
AutoEnergy-TSEff	-2.667	0.430	-6.197	<0.001	0.000

Note: Each row tests the null hypothesis that distributions are the same.
[a]Bonferroni-adjusted significance values.

Table 7
Wilcoxon Signed Ranks test results comparing AutoEnergy with other FE methods for nRMSE and processing time.

Metric	Comparison	Z	Asymp. Sig. (2-tailed)
nRMSE			
	FT - AutoEnergy	-3.202[b]	0.001
	TSEff - AutoEnergy	-3.724[b]	<0.001
	TSMin - AutoEnergy	-3.724[b]	<0.001
	NoFeat - AutoEnergy	-3.724[b]	<0.001
Processing Time			
	FT - AutoEnergy	-3.680[b]	<0.001
	TSEff - AutoEnergy	-3.724[b]	<0.001
	TSMin - AutoEnergy	-0.762[b]	0.446

[b] Based on negative ranks.

5. Conclusion

This work proposed AutoEnergy, an automated FE method designed specifically for ECF to improve AutoML performance. This algorithm automatically generates interpretable features from timestamps and historical energy consumption values while reducing reliance on domain expertise for FE. Through comprehensive evaluation across eighteen diverse real-world energy datasets, encompassing residential buildings, wind turbines, industrial settings, and grid power consumption, AutoEnergy demonstrated significant enhancement of AutoML's predictive performance compared to existing FE methods.

The experimental results revealed that AutoEnergy achieved superior forecasting accuracy with a mean nRMSE of 0.0338, representing substantial reductions in forecasting errors of 19.52 %, 82.38 %, 77.98 %, and 84.72 % compared to FT, TSMin, TSEff, and baseline approaches, respectively. These improvements were statistically validated through both the Friedman and the Wilcoxon tests. In terms of computational efficiency, AutoEnergy demonstrated better processing times, averaging 471.94 seconds across the test sets, performing 1.31x and 4.41x faster than FT and TSEff, respectively. Notably, the method exhibited exceptional performance on small datasets, which is particularly relevant given the common constraints of data availability in real-world energy applications.

The superior performance of AutoEnergy can be attributed to its domain-specific design, where the proposed functions detailed in Section 2 generate interpretable features that effectively capture underlying consumption patterns across diverse energy systems, leading to improved AutoGluon performance. This was further validated through its integration with the state-of-the-art TabPFN algorithm, where AutoEnergy achieved forecasting error reductions ranging from 2.1 % to 92.8 % across different datasets compared to using TabPFN without AutoEnergy. These findings highlight that while general-purpose AutoML frameworks prioritise broad applicability, domain-specific FE methods can significantly enhance performance for specialised tasks such as ECF, offering an effective balance between accuracy and computational efficiency.

While AutoEnergy demonstrates superior performance in ECF applications, several limitations should be acknowledged. First, the algorithm's FE process is specifically tailored to energy time series data, which may limit its generalisability to other domains without substantial modifications. Future work may examine AutoEnergy in other application domains or energy-like domains such as pollution and air quality forecasting. Second, while AutoEnergy has demonstrated strong performance on datasets of up to approximately 145k samples (e.g., the PJME and PJMW datasets), an evaluation of its scalability on much larger datasets and real-time streaming scenarios remains necessary to confirm its computational efficiency under industrial-scale workloads. Third, while the empirical sensitivity analysis provides evidence supporting the one-third sequence-length constraint for nested window features, this parameter remains heuristically determined rather than theoretically optimised, suggesting that future research should explore better optimisation methods that could dynamically adjust the maximum window length based on underlying temporal patterns and series characteristics. Finally, despite the comprehensive experimental evaluation encompassing five distinct pipelines across eighteen diverse datasets and comparison against well-established benchmarking methods, it remains impractical to exhaustively compare AutoEnergy against all existing FE methods, particularly given the rapid evolution of AutoML techniques and the diverse approaches to different types of data and domains. That said, AutoEnergy can be seen as a step towards end-to-end, fully automated ML models for ECF and similar forecasting problems.

CRediT authorship contribution statement

Nasser Alkhulaifi: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Alexander L. Bowler:** Writing – review & editing, Supervision, Conceptualization; **Direnc Pekaslan:** Writing – review & editing, Supervision; **Nicholas J. Watson:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization; **Isaac Triguero:** Writing – review & editing, Validation, Supervision, Methodology, Investigation, Conceptualization.

Data availability

I have shared the link to my data/code at the Attach File step

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported in part by U.K. Engineering and Physical Sciences Research Council (EPSRC) for the University of Nottingham under Grant EP/S023305/1; in part by the EPSRC Centre for Doctoral Training in Horizon: Creating Our Lives in Data; in part by the TSI-100927-2023-1 Project, funded by the Recovery, Transformation and Resilience Plan from the European Union Next Generation through the Ministry for Digital Transformation and the Civil Service. The work of Isaac Triguero was supported by the María Zambrano Fellowship funded by the Spanish Ministry of Universities and Next Generation Funds from the European Union.

Supplementary material

Supplementary material associated with this article can be found in the online version at 10.1016/j.knosys.2025.114300.

Appendix A. Additional results of four additional metrics (RMSE, MAE, MAPE, and R^2).

The following metrics are commonly used to evaluate the predictive performance of regression models:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \tag{A.1}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2} \tag{A.2}$$

$$\mathrm{MAPE} = \frac{100\,\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{A.3}$$

Table A.8
RMSE and MAE results for various feature engineering methods using AutoGluon across all test sets. Lower is better. Bold indicates best results. These results are complementary to those presented in Table 3.

Dataset	FE Methods									
	AutoEnergy		FT		TSMin		TSEff		No.Feat	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
AEP	**125.88**	**90.310**	148.90	112.09	2515.3	1962.9	2633.5	2054.9	2445.8	1959.3
Appliances	72.205	26.727	**68.366**	**23.436**	151.98	78.951	126.89	53.078	100.58	69.336
CAISO_Elec	**573.73**	**368.86**	818.03	592.44	8785.4	7376.6	3782.1	2943.9	7171.6	5677.0
COMED	**64.451**	**28.250**	64.819	31.482	658.33	534.51	676.95	541.09	816.74	664.80
DEOK	**65.096**	**28.533**	65.468	31.797	664.91	539.85	683.72	546.50	824.91	671.45
EKPC	**35.211**	**26.018**	40.448	30.771	505.42	411.83	578.16	482.25	391.87	314.65
FDCS_1	**243.44**	**167.72**	258.66	184.84	535.56	461.58	281.64	183.11	766.88	583.88
FDCS_2	**345.89**	**246.55**	373.22	283.55	573.15	476.24	526.69	422.81	758.16	668.23
FE	**75.202**	**48.521**	92.156	61.690	1326.0	1062.4	1362.6	1099.0	1167.6	906.03
NI	**101.35**	**61.563**	119.97	84.023	2503.8	1972.6	2448.3	1955.9	2998.8	2367.1
PJME	**220.78**	**148.42**	253.55	188.43	7858.0	5770.4	7392.1	5491.2	13,728	11,465
PJMW	**59.313**	**44.096**	69.084	52.694	1132.3	892.57	1101.2	860.49	995.97	794.60
Solar_Home	**0.20367**	**0.14167**	0.21669	0.15633	0.52050	0.44938	0.42167	0.35878	0.37987	0.30742
Steel	6.5323	3.3761	6.5576	3.7719	**5.5918**	**3.0670**	7.0883	4.2280	109.38	76.673
TCity	**360.15**	**263.27**	2704.0	2233.1	7269.4	5907.7	5161.6	4140.5	8099.1	6903.9
UNICON	**0.93407**	**0.68319**	0.96273	0.70705	7.0677	5.2662	4.8874	3.6194	6.5745	5.4347
Victoria	**48.557**	**31.940**	53.593	36.121	1415.5	1141.7	553.51	423.68	912.61	707.86
WindT	**221.48**	**112.34**	335.26	218.04	593.74	447.32	390.85	261.25	1483.6	1322.4
Mean	**145.54**	**94.280**	304.04	231.60	2027.5	1613.4	1539.2	1192.4	2376.1	1952.9

Table A.9
MAPE(%) and R^2 Results for various feature engineering methods using AutoGluon across all test sets. Lower MAPE and higher R^2 are better. Bold indicates best results. These results are complementary to those presented in Table 3.

Dataset	FE Methods									
	AutoEnergy		FT		TSMin		TSEff		No.Feat	
	MAPE%	R2	MAPE%	R2	MAPE%	R2	MAPE%	R2	MAPE%	R2
AEP	**0.62000 %**	**0.99730**	0.77000 %	0.99630	12.940 %	0.040100	13.440 %	-0.052300	13.660 %	-0.0014000
Appliances	18.030 %	0.37090	**14.350 %**	**0.43610**	82.030 %	-1.7868	46.070 %	-0.94280	90.740 %	-0.22070
CAISO_Elec	**1.9900 %**	**0.99010**	3.0200 %	0.97990	32.960 %	-1.3214	14.770 %	0.56980	31.570 %	-0.54690
COMED	**1.0500 %**	**0.98920**	1.1500 %	0.98910	18.270 %	-0.15890	17.920 %	-0.22540	21.640 %	-0.72930
DEOK	**1.0605 %**	**0.97931**	1.1615 %	0.97921	18.453 %	-0.15731	18.099 %	-0.22315	21.856 %	-0.72201
EKPC	**1.7400 %**	**0.99180**	2.0600 %	0.98920	30.440 %	-0.72100	36.180 %	-1.2520	21.660 %	-0.010600
FDCS_1	21.760 %	**0.82470**	**20.810 %**	0.80210	137.90 %	0.15150	25.350 %	0.76530	168.05 %	-0.73980
FDCS_2	**27.620 %**	**0.60140**	32.510 %	0.53590	64.640 %	-0.094500	52.570 %	0.075700	98.190 %	-0.91520
FE	**0.63000 %**	**0.99660**	0.83000 %	0.99490	14.310 %	-0.052800	14.820 %	-0.11170	11.640 %	0.17650
NI	**0.52000 %**	**0.99820**	0.71000 %	0.99740	17.430 %	-0.14990	17.560 %	-0.099500	20.330 %	-0.61260
PJME	**0.47000 %**	**0.99880**	0.61000 %	0.99850	17.390 %	-0.44900	16.380 %	-0.28230	41.430 %	-3.4767
PJMW	**0.80000 %**	**0.99640**	0.97000 %	0.99520	15.810 %	-0.18400	15.040 %	-0.11970	14.520 %	-0.0035000
Solar_Home	**25.740 %**	**0.59270**	28.950 %	0.53890	108.63 %	-1.6604	86.690 %	-0.74600	60.000 %	-0.41700
Steel	6.0600 %	0.99730	6.4200 %	0.99730	**5.3300 %**	**0.99800**	8.7300 %	0.99690	220.76 %	0.25290
TCity	**0.93000 %**	**0.99660**	7.1100 %	0.80770	18.630 %	-0.38970	13.660 %	0.29930	24.050 %	-0.72510
UNICON	**1.7300 %**	**0.96560**	1.7900 %	0.96350	12.200 %	-0.96770	8.5500 %	0.059100	13.670 %	-0.70260
Victoria	**0.70000 %**	**0.99680**	0.78000 %	0.99610	22.790 %	-1.7180	8.7500 %	0.58440	15.040 %	-0.12980
WindT	**127.32 %**	**0.97280**	514.93 %	0.93760	145.91 %	0.80430	144.05 %	0.91520	7235.3 %	-0.22200
Mean	**13.264 %**	**0.90369**	35.496 %	0.88582	43.104 %	-0.43439	31.025 %	0.011589	451.33 %	-0.54184

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n} \left(y_i - \bar{y}\right)^2} \tag{A.4}$$

where y_i denotes the actual value, $\hat{y}_i$ denotes the predicted value, $\bar{y}$ is the mean of the actual values, and n is the total number of observations. MAE (Mean Absolute Error) measures the average magnitude of prediction errors without considering their direction. RMSE (Root Mean Squared Error) penalises larger errors more heavily, making it sensitive to outliers. MAPE (Mean Absolute Percentage Error) expresses errors as a percentage of actual values, providing scale-independent interpretability; however is undefined when any $y_i = 0$. R^2 (Coefficient of Determination) indicates the proportion of variance in the dependent variable explained by the model, with values closer to 1 representing a better fit.

A.1. Additional results across all test sets with AutoGluon.

Tables A.8 and A.9 present comprehensive evaluation results using four additional metrics (RMSE, MAE, MAPE, and R^2) across all 18 datasets, providing multiple perspectives on FE methods' performance with AutoGluon. These results are complementary to those presented in Table 3.

A.2. Additional results across all test sets with TabPFN.

Tables A.10 and A.11 present a comprehensive evaluation using four additional metrics (RMSE, MAE, MAPE, and R^2) across all 18 datasets, providing multiple perspectives on FE methods' performance with TabPFN. These results are complementary to those presented in Table 4.

Table A.10
RMSE and MAE results for various feature engineering methods using TabPFN Across all test sets. Bold indicates the best per row and metric. These results are complementary to those presented in Table 4.

Dataset	Methods									
	TabPFN		TabPFN_AutoEnergy		TabPFN_FT		TabPFN_TSEff		TabPFN_TSMin	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
Appliances	117.18	102.10	**58.891**	24.478	59.070	**24.179**	95.739	74.172	124.23	111.26
FDCS_1	444.96	321.52	**206.34**	**127.91**	223.44	153.03	282.83	200.28	523.57	414.20
FDCS_2	443.46	343.62	**332.63**	**238.43**	376.95	277.53	446.51	348.85	477.77	373.33
Steel	4.2997	1.8133	4.3478	1.9157	4.3778	2.1459	7.2560	4.1050	**4.0600**	**1.7699**
TCity	6936.3	5547.4	**473.57**	**191.57**	545.77	379.33	1807.3	1343.5	5925.3	4732.0
UNICON	5.0920	4.1715	**0.89409**	**0.62719**	0.96541	0.67900	4.6337	3.7580	4.4883	3.7482
Victoria	709.85	588.43	74.830	**28.601**	**69.195**	39.349	471.69	364.28	689.46	548.88
WindT	645.94	346.10	**303.05**	**150.93**	351.15	153.10	574.87	405.29	457.21	334.79
Mean	1163.4	906.89	**181.82**	**95.557**	203.86	128.67	461.36	343.02	1025.8	814.99

Table A.11
MAPE(%) and R^2 results for various feature engineering methods using TabPFN across all test sets. Bold indicates the best per row and metric. hese results are complementary to those presented in Table 4.

Dataset	Methods									
	TabPFN		TabPFN_AutoEnergy		TabPFN_FT		TabPFN_TSEff		TabPFN_TSMin	
	MAPE%	R2	MAPE%	R2	MAPE%	R2	MAPE%	R2	MAPE%	R2
Appliances	146.00 %	-0.57350	19.610 %	**0.60250**	**18.830 %**	0.60010	97.080 %	-0.050400	161.76 %	-0.76870
FDCS_1	88.250 %	0.50360	**17.600 %**	**0.89330**	30.150 %	0.87480	43.360 %	0.79940	179.56 %	0.31270
FDCS_2	46.260 %	0.34480	**28.400 %**	**0.63130**	37.090 %	0.52660	46.650 %	0.33570	52.540 %	0.23940
Steel	**1.7000 %**	0.99880	2.1500 %	0.99880	2.6400 %	0.99880	5.4200 %	0.99670	1.7900 %	**0.99900**
TCity	19.460 %	-0.21430	**0.70000 %**	**0.99430**	1.4100 %	0.99250	4.2800 %	0.91760	17.220 %	0.11390
UNICON	10.700 %	-0.021300	**1.5800 %**	**0.96850**	1.7100 %	0.96330	9.4400 %	0.15420	9.6000 %	0.20650
Victoria	12.420 %	0.34320	**0.60000 %**	0.99270	0.84000 %	**0.99380**	7.4000 %	0.71000	11.260 %	0.38040
WindT	247.27 %	0.77000	**91.090 %**	**0.94940**	156.86 %	0.93200	234.74 %	0.81780	165.61 %	0.88480
Mean	71.507 %	0.26891	**20.216 %**	**0.87885**	31.191 %	0.86024	56.046 %	0.58513	74.917 %	0.29600

References

[1] P. Domingos, A few useful things to know about machine learning, Commun. ACM 55(10) (2012) 78-87. https://doi.org/10.1145/2347736.2347755.

[2] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35(8) (2013) 1798-1828. https://doi.org/10.1109/TPAMI.2013.50.

[3] Z. Liu, D. Zhang, H. Liu, Z. Dong, W. Jia, J. Tan, Visible-hidden hybrid automatic feature engineering via multi-agent reinforcement learning, Knowl. Based Syst. 299 (2024) 111941. https://doi.org/10.1016/j.knosys.2024.111941.

[4] T. Verdonck, B. Baesens, M. Óskarsdóttir, S. vanden Broucke, Special issue on feature engineering editorial, Mach. Learn. 113(7) (2021) 3917-3928. https://doi.org/10.1007/s10994-021-06042-2.

[5] L. Grinsztajn, E. Oyallon, G. Varoquaux, Why do tree-based models still outperform deep learning on typical tabular data?, Adv. Neural Inf. Process. Syst. 35 (2022) 507–520.

[6] N. Hollmann, S. Müller, K. Eggensperger, F. Hutter, TabPFN: a transformer that solves small tabular classification problems in a second, 2022. https://doi.org/10.48550/ARXIV.2207.01848

[7] Z. Wang, L. Xia, H. Yuan, R.S. Srinivasan, X. Song, Principles, research status, and prospects of feature engineering for data-driven building energy prediction: a comprehensive review, J. Build. Eng. 58 (2022) 105028. https://doi.org/10.1016/j.jobe.2022.105028.

[8] A. Mumuni, F. Mumuni, Automated data processing and feature engineering for deep learning and big data applications: a survey, J. Inf. Intell. (2024). https://doi.org/10.1016/j.jiixd.2024.01.002.

[9] A. Gosiewska, A. Kozak, P. Biecek, Simpler is better: lifting interpretability-performance trade-off via automated feature engineering, Decis. Support Syst. 150 (2021) 113556. https://doi.org/10.1016/j.dss.2021.113556.

[10] Y. Wu, X. Xi, J. He, AFGSL: automatic feature generation based on graph structure learning, Knowl. Based Syst. 238 (2022) 107835. https://doi.org/10.1016/j.knosys.2021.107835.

[11] C. Fan, Y. Sun, Y. Zhao, M. Song, J. Wang, Deep learning-based feature engineering methods for improved building energy prediction, Appl. Energy 240 (2019) 35-45. https://doi.org/10.1016/j.apenergy.2019.02.052.

[12] S. Dong, P. Wang, K. Abbas, A survey on deep learning and its applications, Comput. Sci. Rev. 40 (2021) 100379. https://doi.org/10.1016/j.cosrev.2021.100379.

[13] H. Wang, Z. Lei, X. Zhang, B. Zhou, J. Peng, A review of deep learning for renewable energy forecasting, Energy Convers. Manage. 198 (2019) 111799. https://doi.org/10.1016/j.enconman.2019.111799.

[14] N. Hollmann, S. Müller, F. Hutter, Large language models for automated data science: introducing CAAFE for context-aware automated feature engineering, Adv. Neural Inf. Process. Syst. 36 (2024). https://dl.acm.org/doi/abs/10.5555/3666122.3668060.

[15] F. Hutter, L. Kotthoff, J. Vanschoren, Automated Machine Learning: Methods, Systems, Challenges, Springer International Publishing, 2019. https://doi.org/10.1007/978-3-030-05318-5

[16] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, A. Smola, AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data, 2020. https://arxiv.org/abs/2003.06505.

[17] E. LeDell, S. Poirier, H2O autoML: scalable automatic machine learning, in: Proceedings of the AutoML Workshop at ICML, volume 2020, ICML San Diego, CA, USA, 2020. https://www.automl.org/wp-content/uploads/2020/07/AutoML_2020_paper_61.pdf.

[18] C. Wang, Q. Wu, M. Weimer, E. Zhu, FLAML: A Fast and Lightweight AutoML Library, 2019. https://arxiv.org/abs/1911.04706.

[19] X. He, K. Zhao, X. Chu, AutoML: a survey of the state-of-the-art, Knowl. Based Syst. 212 (2021) 106622. https://doi.org/10.1016/j.knosys.2020.106622.

[20] L. Zhang, J. Wen, Y. Li, J. Chen, Y. Ye, Y. Fu, W. Livingood, A review of machine learning in building load prediction, Appl. Energy 285 (2021) 116452. https://doi.org/10.1016/j.apenergy.2021.116452.

[21] P. Manandhar, H. Rafiq, E. Rodriguez-Ubinas, Current status, challenges, and prospects of data-driven urban energy modeling: a review of machine learning methods, Energy Rep. 9 (2023) 2757-2776. https://doi.org/10.1016/j.egyr.2023.01.094.

[22] Y. Sun, F. Haghighat, B.C.M. Fung, A review of the-state-of-the-art in data-driven approaches for building energy prediction, Energy Build. 221 (2020) 110022. https://doi.org/10.1016/j.enbuild.2020.110022.

[23] C. Fan, F. Xiao, Y. Zhao, A short-term building cooling load prediction method using deep learning algorithms, Appl. Energy 195 (2017) 222-233. https://doi.org/10.1016/j.apenergy.2017.03.064.

[24] J. Moon, S. Park, S. Rho, E. Hwang, A comparative analysis of artificial neural network architectures for building energy consumption forecasting, Int. J. Distrib. Sens. Netw. 15(9) (2019) 155014771987761. https://doi.org/10.1177/1550147719877616.

[25] K. Wang, P. Wang, C. Xu, Toward efficient automated feature engineering, in: 2023 IEEE 39th International Conference on Data Engineering (ICDE), IEEE, 2023, p. 1625-1637. https://doi.org/10.1109/ICDE55515.2023.00128.

[26] K. Li, C. Hu, G. Liu, W. Xue, Building's electricity consumption prediction using optimized artificial neural networks and principal component analysis, Energy Build. 108 (2015) 106-113. https://doi.org/10.1016/j.enbuild.2015.09.002.

[27] L. Peng, L. Wang, D. Xia, Q. Gao, Effective energy consumption forecasting using empirical wavelet transform and long short-term memory, Energy 238 (2022) 121756. https://doi.org/10.1016/j.energy.2021.121756.

[28] B. Yu, J. Li, C. Liu, B. Sun, A novel short-term electrical load forecasting framework with intelligent feature engineering, Appl. Energy 327 (2022) 120089. https://doi.org/10.1016/j.apenergy.2022.120089.
[29] Y. Moon, Y. Lee, Y. Hwang, J. Jeong, Long short-term memory autoencoder and extreme gradient boosting-based factory energy management framework for power consumption forecasting, Energies 17(15) (2024) 3666. https://doi.org/10.3390/en17153666.
[30] M.T. Ribeiro, S. Singh, C. Guestrin, "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, KDD '16, ACM, 2016, p. 1135-1144. https://doi.org/10.1145/2939672.2939778.
[31] S. Lundberg, S.-I. Lee, A Unified Approach to Interpreting Model Predictions, 2017. https://arxiv.org/abs/1705.07874.
[32] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nature Mach. Intell. 1(5) (2019) 206-215. https://doi.org/10.1038/s42256-019-0048-x.
[33] D. Slack, S. Hilgard, E. Jia, S. Singh, H. Lakkaraju, Fooling LIME and SHAP: adversarial attacks on post hoc explanation methods, in: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society, AIES '20, ACM, 2020, p. 180-186. https://doi.org/10.1145/3375627.3375830.
[34] D. Garreau, U. Luxburg, Explaining the explainer: a first theoretical analysis of LIME, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 1287–1296. https://proceedings.mlr.press/v108/garreau20a.html.
[35] M. Christ, N. Braun, J. Neuffer, A.W. Kempa-Liehr, Time series feature extraction on basis of scalable hypothesis tests (tsfresh - a python package), Neurocomputing 307 (2018) 72-77. https://doi.org/10.1016/j.neucom.2018.03.067.
[36] S. Yang, K.M. Wilson, T. Roady, J. Kuo, M.G. Lenné, Evaluating driver features for cognitive distraction detection and validation in manual and level 2 automated driving, Human Factors 64(4) (2020) 746-759. https://doi.org/10.1177/0018720820964149.
[37] S.M. Moghadam, T. Yeung, J. Choisne, A comparison of machine learning models' accuracy in predicting lower-limb joints' kinematics, kinetics, and muscle forces from wearable sensors, Sci. Rep. 13(1) (2023). https://doi.org/10.1038/s41598-023-31906-z.
[38] J.M. Kanter, K. Veeramachaneni, Deep feature synthesis: towards automating data science endeavors, in: 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), IEEE, 2015, p. 1-10. https://doi.org/10.1109/DSAA.2015.7344858.
[39] N. Alkhulaifi, A.L. Bowler, D. Pekaslan, I. Triguero, N.J. Watson, Exploring automated feature engineering for energy consumption forecasting with autoML, in: 2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2024, p. 2993-2998. https://doi.org/10.1109/SMC54092.2024.10831959.
[40] N. Hollmann, S. Müller, L. Purucker, A. Krishnakumar, M. Körfer, S.B. Hoo, R.T. Schirrmeister, F. Hutter, Accurate predictions on small data with a tabular foundation model, Nature 637(8045) (2025) 319-326. https://doi.org/10.1038/s41586-024-08328-6.
[41] M. Christ, A.W. Kempa-Liehr, M. Feindt, Distributed and parallel time series feature extraction for industrial big data applications, 2016. https://arxiv.org/abs/1610.07717.
[42] C. Croux, C. Dehon, Influence functions of the spearman and kendall correlation measures, Stat. Methods Appl. 19(4) (2010) 497-515. https://doi.org/10.1007/s10260-010-0142-z.
[43] R. Mulla, Hourly Energy Consumption, 2018. https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption.
[44] L.M. Candanedo, V. Feldheim, D. Deramaix, Data driven prediction models of energy use of appliances in a low-energy house, Energy Build. 140 (2017) 81-97. https://doi.org/10.1016/j.enbuild.2017.01.083.
[45] S. Mayes, N. Klein, K.T. Sanders, Using neural networks to forecast marginal emissions factors: a CAISO case study, J. Clean Prod. 434 (2024) 139895. https://doi.org/10.1016/j.jclepro.2023.139895.
[46] N. Alkhulaifi, A.L. Bowler, D. Pekaslan, G. Serdaroglu, S. Closs, N.J. Watson, I. Triguero, Machine learning pipeline for energy and environmental prediction in cold storage facilities, IEEE Access 12 (2024) 153935-153951. https://doi.org/10.1109/ACCESS.2024.3482572.
[47] Ausgrid, Solar home electricity data, 2022. Accessed: 05 June 2024, https://www.ausgrid.com.au/Industry/Our-Research/Data-to-share/Solar-home-electricity-data.
[48] V.E. Sathishkumar, C. Shin, Y. Cho, Efficient energy consumption prediction model for a data analytic-enabled industry building in a smart city, Build. Res. Inf. 49(1) (2020) 127-143. https://doi.org/10.1080/09613218.2020.1809983.
[49] A. Salam, A. El Hibaoui, Energy consumption prediction model with deep inception residual network inspiration and LSTM, Math. Comput. Simul. 190 (2021) 97-109. https://doi.org/10.1016/j.matcom.2021.05.006.
[50] H. Moraliyage, N. Mills, P. Rathnayake, D. De Silva, A. Jennings, UNICON: an open dataset of electricity, gas and water consumption in a large multi-campus university setting, in: 2022 15th International Conference on Human System Interaction (HSI), IEEE, 2022. https://doi.org/10.1109/HSI55341.2022.9869498.
[51] M. O'Hara-Wild, R. Hyndman, E. Wang, R. Godahewa, tsibbledata: Diverse datasets for "tsibble", 2019. https://doi.org/10.32614/CRAN.package.tsibbledata.
[52] U. Singh, M. Rizwan, Scada system dataset exploration and machine learning based forecast for wind turbines, Res. Eng. 16 (2022) 100640. https://doi.org/10.1016/j.rineng.2022.100640.
[53] C. Zhang, L. Cao, A. Romagnoli, On the feature engineering of building energy data mining, Sustain. Cities Soc. 39 (2018) 508-518. https://doi.org/10.1016/j.scs.2018.02.016.
[54] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures: 3rd ed., Chapman and Hall/CRC, 2003. https://doi.org/10.1201/9781420036268.
[55] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180(10) (2010) 2044-2064. https://doi.org/10.1016/j.ins.2009.12.010.
[56] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30. https://www.jmlr.org/papers/volume7/demsar06a/demsar06a.pdf.
[57] S. Garcia, F. Herrera, An extension on" statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, J. Mach. Learn. Res. 9(12) (2008). https://www.jmlr.org/papers/volume9/garcia08a/garcia08a.pdf.
[58] M. Martínez Comesaña, L. Febrero-Garrido, F. Troncoso-Pastoriza, J. Martínez-Torres, Prediction of building's thermal performance using LSTM and MLP neural networks, Appl. Sci. 10(21) (2020) 7439. https://doi.org/10.3390/app10217439.