Comparative study of the state of the art of machine learning-based models for load forecasting

AIMAN ABDUL WAHAB, University of Twente, The Netherlands

A reliable energy forecasting system is crucial in the energy sector. Multiple classical machine learning models have been investigated in terms of accuracy. This paper aims to shed light on two hybrid models: CNN-XGBoost and LSTM-XGBoost for short-term load forecasting (STLF). The Convolutional Neural Network (CNN) and Long Short-term Memory (LSTM) models can be used to extract relevant time series patterns which can then be passed on to XGBoost and perform better forecasting. The model is evaluated on the load energy data collected from a company residing in a business park in the Netherlands. Temperature and weather data were also considered. The two gradient boosting models were compared, and XGBoost was chosen for constructing a hybrid model because it performed better than CatBoost. The hybrid models were compared and performed better than XGBoost in most benchmarks. Between the two hybrid models, CNN-XGBoost outperformed LSTM-XGBoost by a small margin. Additionally, the paper explores an interpretability tool, Maximum Mean Discrepancy (MMD-Critic) with limited research in the time series domain and is used for understanding black-model decision-making. The model performed worse when attempting to forecast the criticism date with a MAE = 7.23 compared to the preceding two and three weeks with scores of 5.85 and 6.01. The paper demonstrated that hybrid models are more reliable than decision tree models and additionally presented a tool for understanding the weak links of a hybrid model.

Additional Key Words and Phrases: CatBoost, CNN, LSTM, STLF, Gradient Boosting Decision Tree, XGBoost, MMD-Critic, CNN-XGBoost, LSTM-XGBoost, Criticism

1 INTRODUCTION

There is an ongoing shift towards decentralized energy systems. Large power plants are being used to send energy only in one-way transmission, from producers to consumers. Energy transmission is becoming more decentralized thanks to renewable energy and smart grids. Businesses are less reliant on fixed energy contracts. They use energy more efficiently at a lower cost and generate profit by selling excess energy produced via the grid. Despite the advantages, the shift to a two-way energy transmission is causing grid congestion. Without proper measures to ensure a balance of demand and supply of energy passing through the transmission lines, various problems arise, such as price fluctuations and blackouts. Therefore, a stable energy delivery is needed.

Given the significant costs involved in improving the transport capacity within power grids, computational intelligent models can mitigate congestion. The role of the 'model' is to perform energy forecasting such that it can give insights to businesses on peak demand to prepare accordingly.

Extensive research is being done regarding which models are most suited for load forecasting. Models range from time-series methods, classic machine learning models, deep learning models, and hybrid models. Researchers extensively cite the inadequacy of time-series models, such as Autoregressive Integrated Moving Average (ARIMA) in addressing high data dimensions and complex seasonality [1–3].

Alquthami et al. compared the various machine learning and deep learning algorithms for STLF in terms of accuracy and proposed an Enhanced Decision Tree Classifier (EDTC) which integrates fitting function, loss function, and gradient boosting. Deep learning models require a large amount of data to prevent overfitting, which may cause them to get outperformed by decision tree-based models. Other research papers also cited them as underperforming due to lack of locality and higher computational costs [4, 5].

In another paper by Fan et al., Support Vector Machine (SVM) performed slightly better than XGBoost with a Δ RMSE (Root Mean Square Error) of 0.199 in terms of accuracy but at the expense of being 90 times more computationally expensive. Thus, XGBoost and boosting models are recommended for their accuracy and performance [3, 4].

Xianjin and Zhonghua proposed a hybrid model by stacking Catboost and Load Short Term Memory (LSTM), passing the combiner data to a Rigid Regression for the prediction [2]. Features included time features (year, month, weekday, etc.), weather features, and holiday features that are then passed directly to Catboost with no encoding methods. The authors performed a comparison with regular CatBoost, LSTM, and GBDT. The stacking had a Mean Absolute Error (MAE) of 0.9232, a Mean Absolute Percentage Error (MAPE) of 2.36%, and a Root Mean Square Error (RMSE) of 1.1727, different from CatBoost alone, which had 1.1103, 2.77%, and 1.4758, respectively but surpassed GBDT.

Kim et al. predicted solar radiation using a CNN-CatBoost Hybrid model, one of one layer and another of two layers. CNN was adapted to handle time-series data from weather observations such as temperature, humidity, and cloud volume. This adaption allowed CNN to extract significant features from the sequential data important for predicting solar radiation levels. The extractions were then passed to CatBoost to perform the forecasts. The study measured solar radiation across 18 locations and for specific time intervals of sunrise and sunset. The models were compared across different stations and the one-layer hybrid model had an MAE of 0.1040 and an RMSE of 0.2106. The two-layer hybrid model performed better, with a 0.1027 MAE and 0.2104 RMSE. Both performed better than CatBoost and CNN individually. [6]

Shwartz-Zic and Armon delved into hybrid model comparisons of different forms such as an assembly of a gradient boost with another gradient boost, one deep learning model with another deep learning model, and a deep learning with a gradient boost model. The authors compared different papers, trained and preprocessed the hybrid models the same way as in the respective papers, and tested them across various datasets. The models included XGBoost,

TScIT 37, July 8, 2022, Enschede, The Netherlands

Dynamic Neural Field (DNF-NET), 1D-CNN, and Deep Ensemble with and without XGBoost. The results showed that a deep learning network with a gradient-boosting model gave the smallest MSE [4].

While the papers show promising results when doing hybrid models, they do not adequately address the loss of interpretability. Models should be intepretable so that stakeholders can understand how the predictions get selected. There also has not been significant research done on different hybrid model comparisons.

Kim et al. proposed an MMD-critic model that can be used to increase interpretability or explain the decision-making of a black-box model. MMD-critic uses two main concepts: prototypes and criticism. Prototypes are data instances that are 'representative' of the data points. They reside close to the date distribution. Criticism represents instances of data points that differ from the data distribution. 'Maximum mean discrepancy' (MMD) informs how well the selected prototypes are and the goal is to choose the prototypes that output the smallest MMD result. While 'Witness' is for criticism, the greater the witness output deviates from 0, the more it represents a difference from the typical dataset. A reflection of the criticism is encouraged for these instances. This paper aims to shed light on this tool [7].

The scope of this paper is to identify the better machine learning model between CatBoost and XGBoost while also assessing whether constructing it with a neural network increases the accuracy. Another aim of this paper is to successfully use the MMD-critic model for time series data to highlight patterns the hybrid model failed to capture.

Thus my research question (RQ) is formed:

***RQ**: How can traditional machine learning and deep learning models be effectively combined to enhance predictive accuracy and interpretability in load forecasting?*

The question tackles multiple themes, thus it is broken down into two sub-questions:

SQ1: To what extent do hybrid models outperform regular gradient boosting models in regards to accuracy?

SQ2: In what ways can the MMD-critic tool be used to effectively identify weak links in a hybrid model?

By the end of this research, we expect to have contributed in two ways. First, we will compare multiple hybrid models, thus expanding beyond the current scope of classical model comparisons. Second, we will investigate an Explainable AI (XAI) tool not much researched and apply it to the hybrid models.

Section 2 details the methodologies which include an introduction to our dataset, analysis performed on the data, data preparation, model tuning, and an overview of benchmarks and metrics used to compare the models. **Section 3** provides a brief overview of the model architectures used in this paper, **Section 4** provides an overview and a separate methodology for the MMD-Critic tool, and **Section 5** presents the results and discussions arising from it.

2 METHODOLOGIES

Load data has been collected for a business park located in The Netherlands. The park included multiple companies, such as utility, electric, energy, and logistics. Smart meter data from one company was provided and used for data analysis and forecasting. Data scientists widely use the CRISP-DM framework, which consists of business and data understanding, data preparation, modeling, evaluation, and development. The CRIPSM-DM structure is adopted in this paper.

2.1 Business and data understanding

2.1.1 Data understanding and description. The inspection focuses solely on the instantaneous power factor column in the time series aggregated data of 15 minutes for multiple fields from 1 January 2023 until 31 May 2024. The instantaneous power column is called 'obis_9_7_0_mean' and is referred to as load energy in this paper. More description on this field, and other fields not used in this paper can be found in the IEC 62056 protocol on the description of OBIS code [8]. The weather station, Deelan, provided us with weather data, including wind (DD, FH, FF, FX), temperature (T, T10N, TD), pressure (P), sun rays (SQ, Q), precipitation (DR), moisture (RH), visibility (VV), cloudiness (N), and humidity (U). The weather data chosen to be incorporated into the dataframe are T and FH. T is the temperature in Celsius measured at the height of 1.50m, and FH is the hourly average wind speed measured in 0.1 m/s unit. The full description of each abbreviated term is described in the Koninklijk Nederlands Meteorologisch Instituut (KNMI) platform [9]. All features were merged in a dataframe and tools such as pandas, TensorFlow, and NumPy were used.

2.1.2 Data analysis. Box plots of monthly and weekday instantaneous power were plotted and analyzed. Only the weekday box plot is shown in Figure 1. Energy consumption is higher during the autumn and winter months, from October to March, and has a lower value in spring and summer. The mode in June is higher than the precedent and subsequent months. Working days have more energy consumption than non-working days. The highest is on Tuesday and Thursday and the lowest is on Sunday as shown in Figure 1. The dataset also contains a large number of outliers.

Figure 2 compares the load electricity for the years 2023 and 2024. There was a sudden spike in usage starting from September, which continued throughout the following year. Overall, there is a sizeable difference between all months when comparing 2023 with 2024 for the analyzed company.

Figure 3 displays the hourly box plot, it can be seen that from 00:00 until 17:00 energy remains below 5 kW and then starts increasing, with a peak at 8 AM. It goes back low after 16:00.

Some days had a few missing values (around 2-3 for the 15-minute aggregated data) and were replaced with the preceding value. When entire days had missing data, the values were replaced with the mean value for that month.

All features were plotted against the target feature, the features that provided no apparent trend were removed from the dataframe. A correlation matrix was plotted using seaborn. Features that had a very low correlation ($|x| < 0.1$) and features that had a high correlation with each other ($|x| > 0.5$) were removed due to providing little to no information. Only features highly correlated with the target variable were maintained.

It was also investigated how the usage changes during holiday seasons such as Christmas Day, and New Year's Day, and not all of

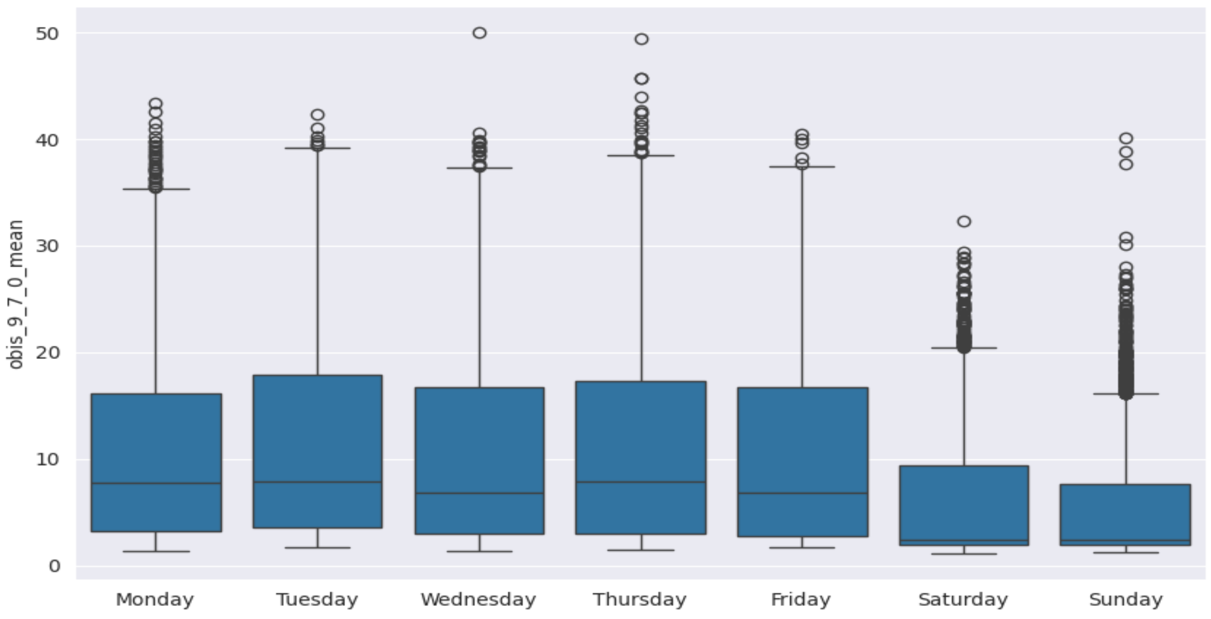

Fig. 1. Weekday box plots of load energy for the year 2023

them had an apparent change in usage when they were compared to the typical days. The correlation between holidays and load usage was also very low (below 0.1) thus the column was omitted.

The witnessed seasonality: energy spikes on specific months, days, times, and years were added as extra columns. Many columns incorporated from the weather data were dropped due to low correlation with the target variable. The ones kept were temperature and wind speed. The feature 'month' had a 0.41 correlation with load usage, the 'year' had a 0.23 correlation and the 'weekday' column was replaced with 'business day', a boolean that is set to '1' for working days. This presented a 0.23 correlation. Regarding the seasonality for periods, a boolean value 'peak time' was incorporated, with '1' signifying 'high' set from 06:00 to 17:00. This feature had a 0.41 correlation with load energy.

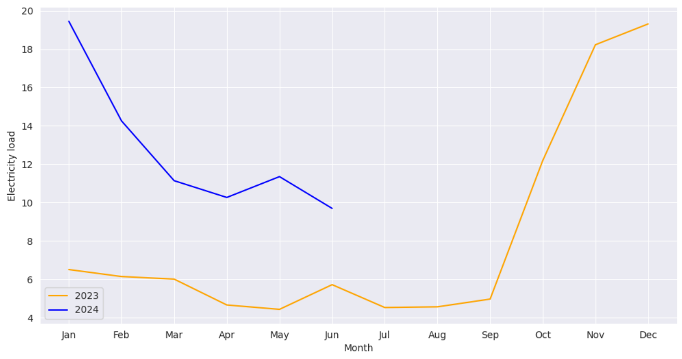

Fig. 2. Yearly electricity load forecast comparison between years 2023 and 2024

Lag values, which represent past observations of a variable, were also added to the dataframe. Different types of lags were tested such as lags of previous days, previous weeks, and previous months.

To summarize, the following columns were added: lag values, business days, temperature, wind speed, and peak time.

2.1.3 Data Preparation. Two types of data preparations were experimented with: 2D (Dimensional) data only for decision trees and 3D data for all models.

First, investigations were done on whether the incorporation of smaller or greater past units presented a better model accuracy. Different lags were considered such as three monthly, seven weekly, seven daily, and seven hourly lags. The lags were tested by being added as extra columns. The initial rows had to be discarded depending on the chosen lag size. For example, three hourly lags would mean that the first 15x4x3 = 180 columns would be dropped. The number fifteen is because of the data being 15 minutes aggregated. Six, nine, eleven, and fifteen months training instances were performed for forecasting horizons of three days, one day, and one hour. The lags with the lowest MAE were chosen as additional columns for the hybrid model.

The standard data structure for training neural network models such as LSTM and CNN is 3D-shaped. More on expected data preprocessing requirements for neural network models can be found on the tensorflow website [10]. The data structure (batch size, features) is transformed into (batch size, time offset, features). Time offset represents preceeding points the model will look at in making the next prediction. This is also known as a rolling window. The rolling window size chosen was 8, as the lag correlations deteriorated past this point which were observed in a partial autocorrelation functions (PACF) plot. A PACF shows the correlation of time series data for different lags. Since our data frame is of 15-minute intervals, an eight-rolling window size represents a 2-hour rolling period.

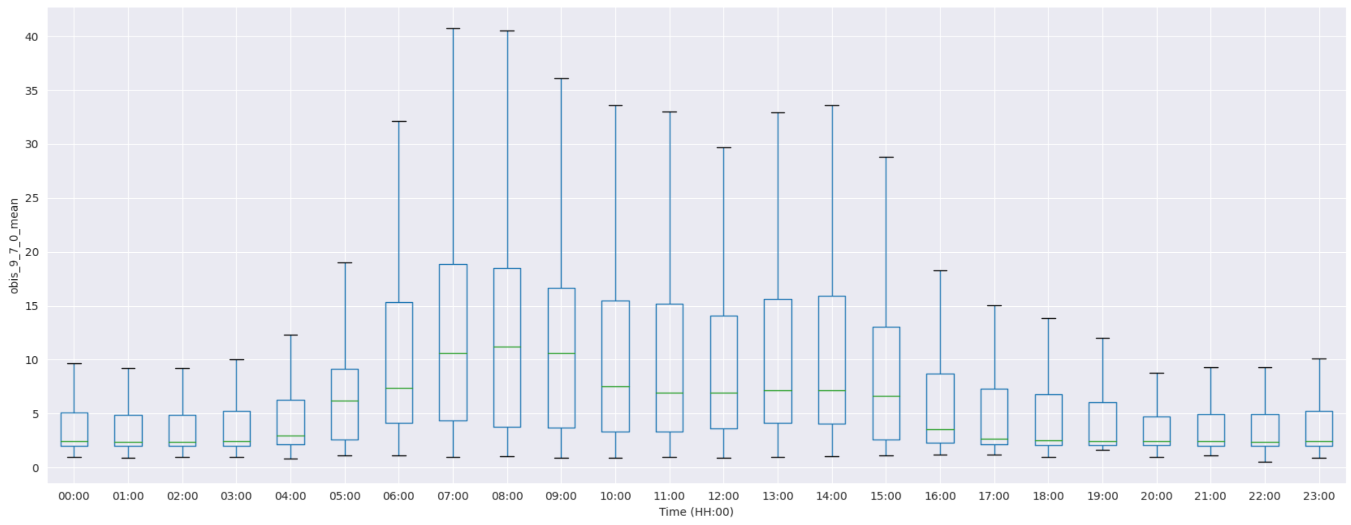

Fig. 3. 2023 hourly box plot for load energy

2.1.4 Modelling. The Catboost and XGBoost models were hypertuned and then forecasted for three days, one day, and one hour and were compared on the seven hourly, seven weekly, and three monthly lags. Grid search was performed on XGBoost and CatBoost for the parameters subsample, n_estimates, max_depth, learning_-rate, colsample_bytree, and colsample_bylevel and on CatBoost for the parameters learning_rate, depth, l2_leaf_reg, grow_policy, and iterations. For neural network models in hybrid models, tuning was performed by manually testing different learning rates. The training sizes varied between six, nine, eleven, and fifteen months, the validation size was of fixed three weeks, and the testing sizes were the forecasting horizons.

After the better of the two models is selected, it is used in construction with two neural network models CNN and LSTM. The flow chart is shown in Figure 4. First, the lags are added as columns and the NaN values are discarded. Rolling window is then applied to data with window size eight. If only the base model is constructed, it is flattened, hypertuned, and trained. The neural network models follow a similar procedure but flattening is not required. For the hybrid implementation, the neural network model is first created separately in a similar process and then the results are passed on to a boosted tree model. The architecture of the hybrid models is described in section 3.0.4.

Because the rolling window uses preceding points and because our data is 15-minute aggregated, only one 15-minute forecasting is possible. To go beyond this, predictions were looped and used as past units until the desired time was reached.

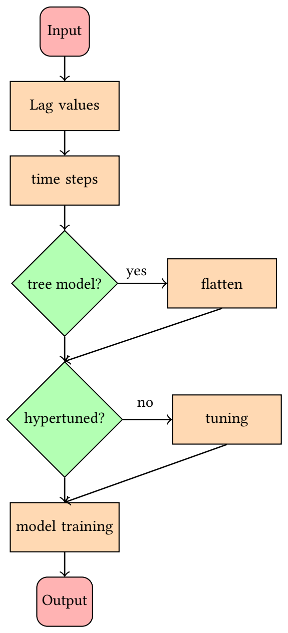

Fig. 4. Flowcharts illustrating the data preparation process

2.1.5 Evaluation. This section details the metrics used for model comparisons.

Mean Absolute Error: Calculates the absolute difference between the model's predicted value and actual value [11].

Mean Squared Error (MSE) & Root Mean Squared Error (RMSE): MSE computes the average squared difference between predicted values and actual values, while RMSE is the square root of MSE.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (1) \qquad \text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (2)$$

where y_i represents the actual value at observation i and $\hat{y}_i$ represents the predicted value.

For all comparisons, MAE was mostly used. However, RMSE is sometimes presented in the results as well.

3 ARCHITECTURAL DESCRIPTION OF VARIOUS MODELS

This section presents a brief overview of the models used in this paper, it is not intended to be detailed. For a more nuanced description, other resources should be consulted.

3.0.1 Gradient Boosting, XGBoost, and CatBoost. XGBoost is based on the concepts of decision tree ensembles and gradient boosting addressed below.

A decision tree is used for decision-making and predictions. The entire dataset is represented by the root node and then splits recursively left and right until an outcome is made. The ensemble consists of combining multiple decision trees to result in a better prediction score.

The model starts with a weak prediction and becomes better as it adds new trees that learn from the previous model by minimizing the 'objective' function, which includes loss and regularization. Loss consists of metrics such as MSE, MAE, or Huber loss, while regularization helps with overfitting. When using the MSE, the objective function is derived as follows:

$$\text{obj}^{(t)} = \sum_{j=1}^{T}[(\sum_{i\epsilon I_j} g_i)w_j + \frac{1}{2}(\sum_{i\epsilon I_j} h_i + \lambda)\omega_j^2] + \gamma T \quad (1)$$

where T is the number of leaves in the tree, g_i and h_i are first and second order derivatives of loss function, ω_j represents the weight of the leaf, and λ and γ are regularization parameters to penalize for overfitting.

The regularization feature addressed in this section is not part of the traditional gradient boosting but instead is implemented in 'optimized' models such as XGBoost. Other elements incorporated in XGBoost consist of pruning and parallel tree boosting. The information has been retrieved from the official documentation on XGBoost [12].

CatBoost is built on gradient boosting and a categorical feature preprocessing is by default integrated. Except for this, it also uses 'symmetric leaf nodes' for faster tree convergence [13].

3.0.2 LSTM. Both LSTM and RNN use backpropagation in their architecture. Recurrent Neural Network (RNN) predicts new values based on previous ones via a feedback loop. The feedback loop may cause vanishing gradients.

The LSTM architecture is an enhanced implementation of RNN to solve the gradient vanishing problem. It does this by using two gates: long-term and short-term memory. It uses a combination of tanh and sigmoid as its activation functions. LSTM has three gates: 'Forget gate', 'Input gate', and 'Output gate'. Forget gate determines the proportion of long-term memory to retain. The input gate updates the long-term memory information, and the output gate updates the short-term memory information. For more details, refer to "Long Short-Term Memory" by Hochreiter et al. [14].

3.0.3 CNN. CNN is another popular neural network model, vastly used in image processing. The architecture is as follows: The data is first passed through a convolution layer with filters used to detect specific features, such as trends or patterns. The filters are improved while applying backpropagation several times. The dot product of the filter and input is called a feature map, where each cell represents a group of similar data patterns. The feature map passes through an activation layer and a pooling that further reduces the data. Finally, the output is flattened and passed on to the dense layer.

3.0.4 Hybrid Models. Figures 5 and 6 showcase the architectures of LSTM-XGBoost and CNN-XGBoost.

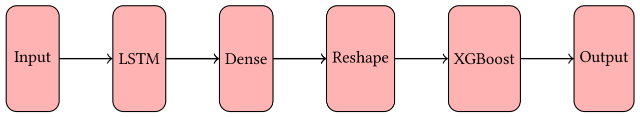

Fig. 5. Flowchart of the LSTM-XGBoost architecture

First, the Input of shape (batch size, time offset, features) is trained on LSTM, then passes through a dense layer with activation layer 'ReLu' and reshaped to 2D before being trained on XGBoost.

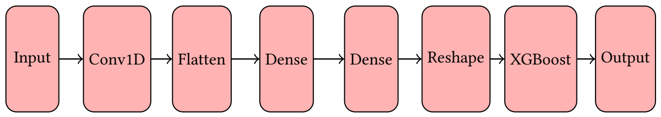

Fig. 6. Flowchart of the CNN-XGBoost architecture

A similar process is done for the CNN-XGBoost hybrid model where the input is passed through a convolutional layer, flattened, densed twice, first with the activation layer 'ReLu' and then a second time with the 'linear' activation layer, is reshaped, and then passed to XGBoost.

4 PROTOTYPES & CRTICISIM

Models may have data entries where they fail to generalize. Prototypes and criticism will help find those cases. In the book *Interpretable machine learning*, Molnar provides the reader with a pseudo-algorithm on the usage of MMD-critic for black-box model intepretability [15]. However, careful consideration is needed when selecting the prototypes and criticism for time-series data as time sequential order is assumed. Thus, an adjusted approach tailored to time-series data is used when finding the prototypes and criticism using the MMD-critic tool.

First, train the machine learning model as usual. The prototypes and criticism are found as stated below:

- Loop until the end of the forecasting horizon, which also depends on the time-series data interval:
  $$\text{Loop} = \frac{\text{Forecast horizon}}{\text{time-series aggregated interval}}$$
  where the forecast horizon and time-series aggregated interval have the same time units. For example, 24-hour forecasting on 15-minute data aggregation requires 24*60/15 = 96 iterations. While looping, perform the following:
  - aggregate the trained dataset to contain only rows of the same time period. This is a crucial step, otherwise, the prototypes selected could be a of a different time causing the time sequence order to be distorted. To illustrate an

example, find the prototype for '00:15' by grouping the trained set size by this time. Prototypes are found in the sequence 00:00, 00:15, ..., 23:45.
  – find the prototype and criticism using the MMD-critic function by passing the aggregated dataset and the number of prototypes and criticisms to be found for the current time horizon. In this paper, only one was considered for both prototype and criticism.
- Perform forecasting on the machine learning model using the computed prototypes and criticism.

The prototype and criticism MAE are measured using different window sizes and for a one-day forecasting horizon. First, the MAE of one-day forecasts is computed for the prototype and criticism data. The next experiment involves selecting a date of 'criticism' and the model attempts to forecast that day. The date is selected by choosing data points residing closer to the criticism instances. The MAE scores are compared with those of 2 and 3 weeks prior.

5 RESULTS AND DISCUSSION

This section aims to find the best model with the appropriate feature engineering techniques. It starts by finding the most significant lag periods by testing them on hourly, weekly, and monthly periods for the two decision trees. The models: CatBoost and XGBoost are compared and the better-performing one is used for constructing the hybrid model. The boosted tree is then compared to the hybrid model to determine whether the results improved. The section ends with an extensive analysis regarding time periods the CNN-XGBoost model fails to forecast accurately which is achieved through the usage of prototypes and criticism.

5.1 Metrics comparison between XGBoost & CatBoost

Model	Forecast	Type	Lag time	MAE
XGBoost	1 day	Monthly	1, 2, 3	6.23
	1 hour	Monthly	1, 2, 3	5.18
	1 day	Weekly	1, 2, 3	3.84
	1 hour	Weekly	1, 2, 3	**3.59**
	1 day	Daily	1, 2, ..., 8	**3.64**
	1 hour	Daily	1, 2, ..., 8	**3.70**
	1 day	Hourly	1, 2, ..., 8	**2.87**
	1 hour	Houry	1, 2, ..., 8	**2.50**
CatBoost	1 day	Monthly	1, 2, 3	**5.84**
	1 hour	Monthly	1, 2, 3	**4.89**
	1 day	Weekly	1, 2, 3	**3.73**
	1 hour	Weekly	1, 2, 3	4.20
	1 day	Daily	1, 2, ..., 8	3.92
	1 hour	Daily	1, 2, ..., 8	3.90
	1 day	Hourly	1, 2, ..., 8	2.95
	1 hour	Hourly	1, 2, ..., 8	2.53

Table 1. XGBoost and CatBoost results for 6 months training size

Table 1 showcases the results of XGBoost and Catboost for 6-month training and different lag intervals. Monthly lags presented the poorest performance with MAE values of 6.23 and 5.84 for 1-day forecasting for XGBoost and Catboost respectively. The MAE scores decreased as the lag time shortened indicating a stronger correlation between nearby data points. The smallest MAE values were for 1-hour forecast and hourly past data points with MAE scores of 2.50 for XGBoost and 2.53 for CatBoost, this represents a $|\Delta MAE| = 2.68$ for XGBoost and $|\Delta MAE| = 2.36$ for CatBoost when subtracted from monthly lags of one-hour forecasting.

XGBoost performed better than CatBoost in 5 out of 8 cases and was selected as the gradient tree to be incorporated in the hybrid model. For daily lags and one-day forecasting, XGBoost had an MAE of 3.64 and CatBoost 3.73, a difference of 0.09. Additionally, even though smaller lag periods represent better results, it is unrealistic to have hourly lag values when performing 1-day forecasting, thus the addition of daily lag values into the data frame was chosen instead.

5.2 Metrics comparison XGBoost & Hybrid models

Tables 2 and 3 summarize the results of the models for different horizons and training sizes.

Note: The data for XGBoost was preprocessed differently in this section than in the previous one. For three-day forecasting, predicted lag values should have been used instead of the actual values for lag days one and two.

As can be seen, in most cases, the hybrid models outperformed XGBoost. The only exception was for fifteen months of training size when performing 3-day and 1-hour forecasting.

CNN-XGBoost was the better hybrid model of the two, outperforming LSTM-XGBoost seven times. The architecture of CNN was also slightly different, previously shown in Figures 5 and 6 where CNN had an additional Dense layer with 'ReLU' as the activation unit. This feature incorporation might have given this hybrid model an advantage over the other.

While the hybrid models performed better overall, many instances were minor improvements. The smallest positive difference was of $|\Delta MAE| = 0.01$. In most cases, the difference was around 0.05-0.35. This does come with higher computational costs. Training a neural network model is known to take more training time as compared to decision tree models [16]. In this case, it took even longer as it involved the cost of training the neural network and the gradient boosting model. The accuracy and computational cost trade-off should be considered when deciding which model to use.

In some cases, longer forecasting intervals outperform the shorter ones such as the column with an 11-month training size. The model performed poorly on 1-hour forecasting, the CNN-XGBoost had a value of MAE 12.45 and performed much better during a three-day forecasting horizon with a value of MAE 6.89. The predictions are performed every 15 minutes and the difference is quite large indicating that the other points severely 'compensated' for the poor performance. This can be seen again for the nine-month training size column where 1-hour forecasting is 0.91 for CNN-XGBoost but 12.94 for 1-day forecasting. This issue is addressed in the next subsection.

5.3 Metric comparison between Prototypes & Criticism

96 prototype and criticism data instances were calculated for different window sizes. These points represent a full 24-hour day. A CNN-XGBoost hybrid model was trained on 11-month data and

Training Size	Model	Forecasting Horizon	RMSE	MAE
6 months	CNN-XGBoost	3 days	5.02	**2.90**
	LSTM-XGBoost		5.18	2.96
	XGBoost		5.05	2.91
	CNN-XGBoost	1 day	0.92	**0.44**
	LSTM-XGBoost		0.93	0.48
	XGBoost		0.94	0.45
	CNN-XGBoost	1 hour	0.07	**0.06**
	LSTM-XGBoost		0.13	0.13
	XGBoost		0.1	0.09
9 months	CNN-XGBoost	3 days	12.49	**9.22**
	LSTM-XGBoost		13.07	9.90
	XGBoost		12.6	9.29
	CNN-XGBoost	1 day	16.22	12.94
	LSTM-XGBoost		14.72	**11.53**
	XGBoost		16.66	13.44
	CNN-XGBoost	1 hour	1.15	**0.91**
	LSTM-XGBoost		1.41	1.25
	XGBoost		1.40	1.23

Table 2. Comparison between XGBoost and hybrid models for 6 and 9 months training size

Training Size	Model	Forecasting Horizon	RMSE	MAE
11 months	CNN-XGBoost	3 days	8.63	6.89
	LSTM-XGBoost		8.81	7.02
	XGBoost		8.45	**6.76**
	CNN-XGBoost	1 day	9.02	**7.34**
	LSTM-XGBoost		9.75	7.98
	XGBoost		9.26	7.65
	CNN-XGBoost	1 hour	13.01	12.45
	LSTM-XGBoost		12.48	**11.95**
	XGBoost		13.78	13.77
15 months	CNN-XGBoost	3 days	13.15	9.02
	LSTM-XGBoost		12.60	8.73
	XGBoost		12.65	**8.67**
	CNN-XGBoost	1 day	11.87	7.82
	LSTM-XGBoost		11.42	**7.63**
	XGBoost		11.66	7.7
	CNN-XGBoost	1 hour	1.76	1.73
	LSTM-XGBoost		1.05	1.02
	XGBoost		0.34	**0.23**

Table 3. Comparison between XGBoost and hybrid models for 11 and 15 months training size

then forecasts were performed on the prototypes and criticism instances. The algorithmic approach for finding the instances was explained in section 4. Table 4 displays the MAE and RMSE scores when forecasting using the prototype and criticism data instances.

		Business days		Non-business days	
Window size		RMSE	MAE	RMSE	MAE
11 Months	Prototype (grey rows)	5.46	4.80	3.58	3.57
	Criticism (white rows)	5.48	4.81	4.30	4.03
9 Months		5.60	4.76	3.23	3.23
		5.89	5.07	3.72	3.65
6 Months		5.48	4.67	3.21	3.20
		5.66	4.84	3.37	3.35
1 Month		5.21	4.45	3.27	3.27
		5.56	4.88	3.31	3.30
1 Week		4.94	4.27	3.24	3.23
		5.36	4.60	3.25	3.25

Table 4. Prototypes and criticism for 11-month training data for CNN-XGBoost at different aggregated time intervals. The prototype values are represented in grey and the criticism in white. The predicted prototype and criticism results were compared with the days 16 January 2023 (business day) and 22 January 2023 (non-business day).

The following list of observations can be made.

- The window difference between prototype and criticism is smaller for non-business days. This means there is less data discrepancy on weekends.
- For business days and 11-month aggregation, the difference between prototype and criticism is small, $\Delta RMSE = 0.02$ and $\Delta MAE = 0.01$. This narrow difference, and the fact that the metrics scores are higher compared to the cells in non-business days indicates that the data has high discrepancy and proper representative points can not be found.
- For business days, the prototype-criticism difference widens as you shrink the window size and is quite different even for 1-week data aggregation, $\Delta RMSE = 0.42$ and $\Delta MAE = 0.33$. This indicates that criticism dates can properly be found.
- For non-business days the 11-month aggregation should be considered as the gap between prototype and criticism is high, $\Delta RMSE = 0.72$ and $\Delta MAE = 0.46$. A 9-month aggregation has a $\Delta RMSE = 0.49$ and $\Delta MAE = 0.42$ and the gap gets narrower on smaller window sizes. This indicates that the dates after the six month period have more discrepancy. This was already witnessed in Figure 2 when a noticeable spike occurred in September.

In a short summary, to analyze on business-day data, the training data should be narrowed down to at least nine months before analyzing weak point links. This is different for non-business days where a larger window size is encouraged.

The prototypes and criticism values are observed for 6-month aggregation, Figure 7 displays a graph of prototype, criticism, and 1 August 2023 which was arbitrarily selected. As expected, the prototype and date values are close to each other, while the criticism points stand out. Extreme deviations are observed at 14:45, where the criticism instance is 27.09kW, while the prototype and for the selected date August 1, 2023, are 11.53kW and 7.44kW respectively. The second highest criticism peak is for the hour 18:15 with the value of 18.89kW compared to 4.28kW for the prototype and 2.06kW for August. After observing the data of criticism, the 6-month data frame was grouped by midnight and filtered by load energy values greater than 10kW. This resulted in a single output, the DateTime 10 January 2023 with a load value of 12.51kW. The second largest value was 7.29kW in the seventh month, much later than when this deviation first occured. All other entries for 10 January also present relatively high values in comparison to the typical values for that month. The data should be carefully investigated whether it was an error, a one-time occurrence, or an uncaptured trend by the model.

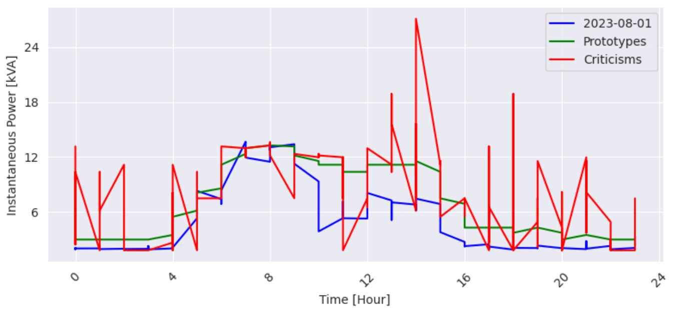

Fig. 7. 24-hour plot of prototypes, criticism, and the date 1 August 2023 for 6-month data training.

Lastly, we will investigate how poorly a model performs when forecasting on days of criticism.

The date 24 July 2023 is used in this example, the reason is that it has values aligning closer to the criticism instances. For example, at 2 AM when the value for this particular date is 1.86kW, the criticism instance is 1.80kW, and the prototype is 2.98kW. This is just one example of suspicion, the data was analyzed for multiple rows and compared to the criticism value. For the previous week, the data was missing and thus not used in the comparison. Standard training, validation, and forecasting were performed for the dates 24 January 2023, 10 January 2023, and 3 January 2023, the results are presented in Table 5.

Metrics	2023-07-24	2023-07-10	2023-07-03	$\lvert\Delta_{2-3}\rvert$	Δ_{2-4}	$\lvert\Delta_{3-4}\rvert$
RMSE	7.99	5.85	6.01	2.14	1.98	0.16
MAE	7.21	4.62	4.81	2.59	2.40	0.19

Table 5. 1-day forecasts RMSE and MAE computation for the date of criticism '2023-07-24' and the preceding 2 and 3 weeks

As observed, it performed significantly worse on forecasting the 24th as compared to the preceding two and three weeks. Forecasting for the criticism date had an RMSE = 7.99 and an MAE = 7.21, unlike the preceeding two and three weeks with RMSE and MAE scores of 5.85, 4.62, and 6.01, 4.81 respectively. The error difference between the criticism date and the preceding weeks was also much higher than that of the preceding two and three weeks, $\Delta_{2-3} = 2.14$, $\Delta_{2-4} = 1.98$, and $\Delta_{3-4} = 0.16$ where 2 represents the date July 24th, 3 represents the date July 10th, and 4 the date July 3rd. Thus, the selected date represents a valid criticism and the model might have failed to train on such patterns.

6 CONCLUSION

The aim of this research was to investigate the effectiveness of different machine learning models in time-series analysis. It included an in-depth comparison between two popular gradient boosting algorithms CatBoost and XGBoost exploring their performance for different lags. The accuracy improved as the window span decreased indicating stronger correlations with nearer entries. XGBoost provided better results as compared to CatBoost. Besides that, it was effectively shown that accuracy levels improve when incorporating them in hybrid models. This was determined by extensive analysis of the base results, XGBoost, with the other two models LSTM-XGBoost and CNN-XGBoost by comparing them in different categories, where the hybrid models surpassed most of them. CNN-XGBoost was also shown to be one of the better hybrid models. The results raised suspicion that all models had difficulties in capturing specific trends. This suspicion was confirmed when analyzing the prototypes and criticism for business and non-business days. The analysis demonstrated that the model was able to capture non-business day trends better than business day ones. Additionally, the model showed a better ability to generalize on smaller windows for business days. There was a noticeable difference in accuracy when forecasting a day aligning with the 'prototype' as compared to the 'criticism'. The tool achieved its purpose in making the hybrid model more comprehensible. It also showcased the points of criticism which could potentially help stakeholders better tune their models to capture the missing trends.

Both sub questions have been thoroughly answered. Subquestion 1 was answered by showing that CNN-XGBoost resulted in minor improvements in most cases. Subquestion 2 was answered by providing an example on how prototypes and criticism helped catch weak forecasting patterns by the model.

6.1 Future work

Simplistic architectures were chosen when constructing the neural network models, the accuracy difference between our base model and the hybrid one could be improved by considering a more complex structure. While trade-offs regarding accuracy and training time were addressed, more investigations can be made in determining whether the hybrid model is more suited when taking computational costs in consideration. The MMD-critic model can be further investigated for its effectiveness in helping stakeholders improve a model's accuracy with an example on how the accuracy was increased after incorporating additional tuning due to this tool.

A APPENDIX

During the preparation of this work the author used Grammarly and Overleaf in order to assist in highlighting spelling mistakes and unclear sentences. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the content of the work.

B REFERENCES

[1] Chenrui Zhang, Zhonghua Chen, and Jing Zhou. Research on short-term load forecasting using k-means clustering and catboost integrating time series features. *2020 39th Chinese Control Conference (CCC)*, pages 6099–6104, 2020.
[2] Xianjin Yang and Zhonghua Chen. A hybrid short-term load forecasting model based on catboost and lstm. In *2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP)*, pages 328–332, 2021.
[3] Junliang Fan, Xiukang Wang, Lifeng Wu, Hanmi Zhou, Fucang Zhang, Xiang Yu, Xianghui Lu, and Youzhen Xiang. Comparison of support vector machine and extreme gradient boosting for predicting daily global solar radiation using

temperature and precipitation in humid subtropical climates: A case study in china. *Energy Conversion and Management*, 164:102–111, 2018.

[4] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need, 2021.

[5] Razak Olu-Ajayi, Hafiz Alaka, Ismail Sulaimon, Funlade Sunmola, and Saheed Ajayi. Building energy consumption prediction for residential buildings using deep learning and other machine learning techniques. *Journal of Building Engineering*, 45:103406, 2022.

[6] Hyojeoung Kim, Sujin Park, Hee-Jun Park, Heung-Gu Son, and Sahm Kim. Solar radiation forecasting based on the hybrid cnn-catboost model. *IEEE Access*, 11:13492–13500, 2023.

[7] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[8] Ing. Petr NÁVRAT. Iec62056 obis codes, 2024. Accessed: 2024-06-30.

[9] KNMI. Netherlands seismic and acoustic network, 1993. Royal Netherlands Meteorological Institute (KNMI). Other/Seismic Network. Digital Object Identifier (DOI): 10.21944/e970fd34-23b9-3411-b366-e4f72877d2c5.

[10] TensorFlow. Time series forecasting. https://www.tensorflow.org/tutorials/structured_data/time_series. Last updated: 2023-06-29, Accessed: 2024-06-29.

[11] Anand Singh Rajawat, Omair Mohammed, Rabindra Nath Shaw, and Ankush Ghosh. Chapter six - renewable energy system for industrial internet of things model using fusion-ai. In Rabindra Nath Shaw, Ankush Ghosh, Saad Mekhilef, and Valentina Emilia Balas, editors, *Applications of AI and IOT in Renewable Energy*, pages 107–128. Academic Press, 2022.

[12] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[13] Anna Veronika Dorogush, Andrey Gulin, Gleb Gusev, Nikita Kazeev, Liudmila Ostroumova Prokhorenkova, and Aleksandr Vorobev. Fighting biases with dynamic boosting. *CoRR*, abs/1706.09516, 2017.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[15] Christoph Molnar. *Interpretable Machine Learning*. Lulu.com, 2020.

[16] L.O. Hall, X. Liu, K.W. Bowyer, and R. Banfield. Why are neural networks sometimes much more accurate than decision trees: an analysis on a bio-informatics problem. In *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483)*, volume 3, pages 2851–2856 vol.3, 2003.