Forecasting Electricity Load and Wind Generation: A Comparative Analysis of Machine Learning Models Enhanced by Bayesian Optimization under Different Sampling

Zhen Wang

A Thesis in

The Department

of

Supply Chain and Business Technology Management

Presented in Partial Fulfillment of the

Requirements for the Degree of

Master of Science (Business Analytics and

Technology Management)

at Concordia University

Montréal, Québec, Canada

September 2024

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Zhen Wang

Entitled: Forecasting Electricity Load and Wind Generation: A Comparative Analysis of Machine Learning Models Enhanced by Bayesian Optimization under Different Sampling

and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Business Analytics and Technology Management)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

________________________________ Chair

Dr. Mahdi Mirhoseini

________________________________ Examiner

Dr. Rustam Vahidov

________________________________ Examiner

Dr. Dongliang Sheng

________________________________ Supervisor

Dr. Salim Lahmiri

Approved by ____

Chair of Department or Graduate Program Director

October 24, 2024 ____

Dean of Faculty

Abstract

Forecasting Electricity Load and Wind Generation: A Comparative Analysis of Machine Learning Models Enhanced by Bayesian Optimization under Different Sampling

Zhen Wang

The study explores the application of advanced machine learning techniques to forecast electricity load and wind generation data, focusing on the optimization and comparative analysis of various models. Given the critical importance of accurate energy forecasting in managing power grids and integrating renewable energy sources, this research seeks to enhance forecasting precision through the application of Bayesian optimization for hyperparameter tuning across multiple models. Utilizing time-series data, this study systematically evaluates the performance of several predictive models. Each model's parameters were meticulously optimized using Bayesian techniques to identify the most effective configurations for handling the complex dynamics of energy data. The research methodology involved a comparison within single datasets to identify the best model. Subsequently, the best-performing models were further analyzed across different datasets to validate their robustness and generalizability. The primary evaluation metric is the Root Mean Squared Error (RMSE), complemented by additional metrics to provide a comprehensive assessment of model accuracy and effectiveness. Key findings demonstrate that while some models excel in capturing overall trends, challenges remain in addressing the volatility and variability inherent in the data. The insights derived from this study not only advance the field of energy forecasting but also offer practical implications for energy policymakers and stakeholders in optimizing grid performance and renewable energy integration.

Keywords: time series, machine learning, Bayesian optimization

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Salim Lahmiri, for his invaluable guidance and support throughout my research. Special thanks to the thesis committee: Dr. Mahdi Mirhoseini, Dr. Rustam Vahidov and Dr. Dongliang Sheng for their valuable feedback and time. I also extend my thanks to Concordia University and the Department of Supply Chain and Business Technology Management for providing the resources necessary for completing this thesis. Lastly, A special thank you to my family for their endless love, patience, and encouragement. I also appreciate the friendship and support of my colleagues and friends throughout this journey. Thank you all.

Contents

List of Tables

List of Figures

1. Introduction

1.1 Background and Motivation

Electricity load and wind generation forecasting are critical components of modern energy systems. These forecasting processes serve as the foundation of energy system management, ensuring that the balance between supply and demand is precisely maintained (Pawar & TarunKumar, 2020). This balance is crucial for multiple reasons, including the optimization of resource allocation, the maintenance of grid stability, and the promotion of sustainable energy practices. The ability to predict electricity load and wind generation with high precision allows for a more efficient and reliable grid operation, reducing the risk of blackouts and ensuring that energy is available when and where it's needed most (Jones, 2017).

The integration of renewable energy sources, such as wind power, into the energy mix introduces new challenges and complexities in forecasting. Renewable sources like wind are inherently more influenced by environmental conditions, which can fluctuate widely and unpredictably (Katzenstein, 2010). While the inherent unpredictability of renewable energy sources poses a significant challenge, the application of time series data in forecasting models provides a strategic solution to address the problem. Therefore, the challenge lies not only in managing the unpredictability but also in effectively using time series data to create accurate models that could explain the temporal variability (Prema & Rao, 2015).

Time series forecasting models are designed to leverage historical data to predict future data. The complexity arises in selecting and optimizing models that can effectively interpret the patterns and relationships within the data, accommodating both the seasonal trends and the shorter-term fluctuations driven by environmental conditions. Forecasting models must be capable of dynamically adjusting to new data and patterns as they emerge. This necessitates the development of models that continually refine their predictions based on the latest available data. Such models include advanced machine learning and deep learning approaches, which can learn complex relationships within the data and improve their accuracy over time.

The motivation to build forecasting models is driven by the need to support the growing reliance on renewable energy. As the world moves towards a more sustainable energy future, the role of accurate forecasting becomes increasingly important. It enables energy systems to integrate renewable sources more effectively, reducing reliance on fossil fuels and lowering carbon emissions (Kariniotakis, 2017). Furthermore, accurate forecasting supports economic efficiency by optimizing the use of energy resources, reducing waste, and potentially lowering the cost for consumers. By improving the accuracy of electricity load and wind generation forecasts, stakeholders can make more informed decisions, paving the way for a more resilient, efficient, and sustainable energy future.

1.2 Research Context

In the context of the European energy landscape, particularly through the lens of Austria, Germany, and the Netherlands, the research looks into the dynamics of electricity load and wind generation forecasting. These countries represent a cross-section of Europe's energy ecosystem, each with its unique integration of renewable energy sources and demand patterns influenced by diverse environmental, economic, and societal factors.

The research focuses on the utilization of time series data as a pivotal strategy in addressing the complexities inherent in forecasting the variability of electricity load and wind generation. The nuanced understanding of time series data enables a more accurate analysis of patterns over time, essential for managing the unpredictability of wind energy and the fluctuating demand for electricity. By dissecting the forecasting process into separate considerations for electricity load and wind generation, the study aims to tailor predictive models to the specific characteristics and challenges of each domain, thereby enhancing accuracy and reliability.

1.3 Research Objectives

The objective of this thesis is to conduct a comprehensive comparative analysis of various predictive modeling techniques for forecasting electricity load and wind generation in three European countries. This research seeks to shed light on the relative strengths and weaknesses of a broad array of predictive modeling techniques when applied to the specific challenges and data characteristics found in these regions. Specifically, the research aims to achieve the following objectives:

The first objective is to implement and train multiple models. At the base of this research is the practical application of a diverse set of predictive modeling techniques. The models selected for this study are: Convolutional Neural Networks (CNN), Temporal Convolutional Neural Network (TCNN), Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (BiLSTM), Gated Recurrent Unit (GRU), Bidirectional Gated Recurrent Unit (BiGRU), Deep Neural Network (DNN), K-Nearest Neighbors (KNN), and Decision Trees. The models represent a blend of classical machine learning algorithms and more recent deep learning approaches. Each model brings unique strengths to the table, such as CNN's ability to capture spatial-temporal patterns, LSTM's advantage in handling sequential data, and GRU's simplified architecture that retains the ability to manage long-term dependencies while keeping computational efficiency. Implementing and training the models involves not just the application of the algorithms but also the delicately tuning of hyperparameters.

The second objective is to evaluate the performance of models. The effectiveness of each model will be assessed through a combination of metrics and statistical methods. Accuracy metrics such as Root Mean Squared Error (RMSE) will be employed to quantify the forecasting precision of each model. Beyond accuracy, the evaluation will also consider factors such as training time.

The third objective is to compare across different time intervals. A key aspect of this research is the comparison of model performance across different forecasting horizons, specifically 15-minute and 60-minute intervals. This analysis is critical because the optimal model for short-term forecasting may not necessarily perform best over longer intervals, due to the different dynamics and uncertainties involved.

The last objective is to identify the most effective predictive model. The culmination of this research is the identification of the predictive model or models that offer the best balance of forecasting accuracy and computational efficiency for electricity load and wind generation in the European context. This involves not just a straightforward ranking of models based on performance metrics but also consideration of factors such as model scalability, ease of implementation, and computational resource requirements.

1.4 Significance of the Research

The outcomes of this research hold significant implications for the energy sector, particularly in terms of enhancing the reliability, efficiency, and sustainability of energy systems in Europe and beyond. Furthermore, the findings contribute to extension of the current knowledge in predictive modeling and time series analysis. By conducting the comparative analysis, the research also shows how the predictability of electricity load and wind generation changes over different time scales and how each model adapts to these changes. And therefore, to provide a clear recommendation that can guide utilities, grid operators, and policymakers in selecting and deploying forecasting models that best meet their needs in their specific context.

1.5 Structure of the Thesis

This thesis is organized into several chapters, each addressing specific aspects of the research study. Chapter 1 introduces the research topic, outlining the background, motivation, objectives, and significance of the study. Chapter 2 presents a comprehensive review of literature related to electricity load and wind generation forecasting, as well as predictive modeling techniques. Chapter 3 describes the methodology employed in data collection, preprocessing, model implementation, and evaluation. Chapter 4 presents the results and findings of the analysis, including a comparative assessment of forecasting models. Chapter 5 discusses the implications of the findings, identifies limitations of the study, and suggests avenues for future research. Finally, Chapter 6 offers concluding remarks and summarizes the key contributions of the research.

2. Literature Review

2.1 Electricity Load Forecasting

The primary research question revolves around how to improve the accuracy of electricity load forecasting using advanced modeling methods. Researchers have integrated machine learning models with traditional forecasting methods to enhance prediction accuracy. For instance, Kaytez (2020) presents a comprehensive study focusing on forecasting Turkey's net electricity consumption using a hybrid model. This model integrates the Autoregressive Integrated Moving Average (ARIMA) method with the Least Squares Support Vector Machine (LS-SVM) to enhance forecasting accuracy. Specifically, the ARIMA model is utilized for its effectiveness in understanding and predicting future points in a series, while the LS-SVM approach is applied for its proficiency in handling non-linear data patterns. This combination aims to address the complex dynamics of electricity consumption patterns in Turkey, a market characterized by its rapid development and increasing energy demand. The main findings demonstrate the hybrid model's superior performance over traditional models like multiple linear regression and single ARIMA models, as well as the official forecasts. Kaytez also highlights the model's ability to offer more realistic and reliable electricity consumption forecasts, which are crucial for strategic planning in energy investments and policy making. Gao et al. (2021) present a hybrid machine learning model for forecasting residential electricity consumption in China, combining an Extreme Learning Machine (ELM) optimized by the Jaya algorithm with online search data as predictors. This innovative approach significantly enhances forecasting accuracy by leveraging the predictive power of internet search trends related to electricity use. The model's effectiveness is validated through comparative analysis with traditional models and standalone ELM, showing superior performance in accuracy metrics. Hadjout et al. (2022) introduce a deep learning-based ensemble model for forecasting monthly electricity consumption in Algeria, incorporating Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Temporal Convolutional Networks (TCN) techniques. This approach aims to leverage the strengths of each method to enhance prediction accuracy. By evaluating the model using 14 years of data from nearly 2000 clients in Bejaia, the study demonstrates its effective performance over traditional forecasting models, offering valuable insights for energy management and planning in the Algerian market. Similarly, Albuquerque et al. (2022) conduct a comprehensive analysis to forecast Brazilian power electricity consumption using machine learning models, specifically Random Forest and Lasso Lars, and compares their performance to benchmark specifications such as ARIMA and Random Walk. The study reveals that machine learning method significantly outperform traditional models in accuracy for short to medium-term forecasting. These models effectively capture the complexities of trend and seasonality in power electricity consumption data, attributing their success to the inclusion of a broad set of explanatory variables, including weather and calendar effects, which are crucial for enhancing forecast accuracy, especially in the context of the Brazilian electricity market. Saranj and Zolfaghari (2022) adopt a hybrid model that combines Adaptive Wavelet Transform (AWT) with Long Short-Term Memory (LSTM) networks and ARIMAX-GARCH models. This approach is designed to enhance prediction accuracy by addressing the multi-frequency characteristics of electricity consumption data. The study tests the model using data from the UK market, considering various exogenous variables such as climate conditions and calendar dates. The hybrid model demonstrates superior performance over traditional forecasting models and alternative filtering methods, indicating its effectiveness in improving the predictive power for short-term electricity consumption forecasting.

Over time, the shift has been towards combining different data sources and leveraging the strengths of both statistical and machine learning models. The results consistently indicate a significant improvement in forecasting accuracy, providing crucial insights for energy management and planning. Nonetheless, challenges remain in handling the volatility of electricity consumption, integrating diverse data types, and ensuring model scalability. There's an ongoing need for models that can adapt to changing consumption patterns and incorporate real-time data effectively.

Table 1 Summary of Studies on Electricity Forecasting

Study	Features	Models	Sample	Findings
Kaytez (2020)	Economic variables; Industry variables; Electricity consumption	MLR; Single ARIMA; ARIMA-LSSVM hybrid approach	Net electricity consumption data of Turkey from 1970 to 2017	ARIMA-LSSVM hybrid approach with best RMSE of 0.904
Gao et al. (2021)	Online search volume of keywords	SARIMA; SVR; BPNN; ELM; Jaya-ELM;	108 samples of monthly residential electricity consumption data of China from 2011 to 2019	Jaya-ELM with best MAE of 2.00 and MAPE of 2.41%
Hadjout et al. (2022)	Electricity consumption	LSTM; GRU; TCNN; Ensemble of LSTM, GRU, and TCNN; SARIMA	168,432 samples (used) of monthly electricity consumption data of Bejaia, Algeria from 2006 to 2019	Ensemble with best MAE of 1649.65 and MAPE of 3.04%
Albuquerque et al. (2022)	Time variables; Weather variables; Historical price; Economic variables; Electricity consumption	Random Forest; Lasso Lars; Lars; ARIMA; Elastic Net	546 samples of daily power electricity consumption data of Brazil from February 1, 2017 to July 31, 2018	Random Forest with best MAE and MAPE across different time horizons
Saranj and Zolfaghari (2022)	Time variables; Weather variables; Electricity consumption	A new hybrid model of "AWT-LSTM-ARIMAX-GARCH-RF"	56,927 samples of half-hourly electricity consumption data of the UK from January 1, 2018 to March 31, 2021	The hybrid model with best RMSE of 1.249 and MAPE of 0.869%

2.2 Wind Generation Forecasting

The wind forecasting market is relatively new compared to the electricity forecasting market, primarily because wind energy's integration into the energy mix has gained significant momentum only in the last few decades. While electricity load and demand forecasting has been a critical component of energy management for a longer period, driven by the need to balance supply and demand in grids primarily powered by fossil fuels, the rise of renewable energy sources has introduced new complexities. Wind energy's variable and unpredictable nature necessitates advanced forecasting techniques to optimize its integration, leading to the development of a specialized wind forecasting market.

Academically, the focus has been on optimizing wind speed forecasting models to accommodate the variability and unpredictability of wind speeds. Studies like Huang et al. (2023) have explored an advanced wind speed forecasting model using a genetic algorithm optimized Long Short-Term Memory network, and further enhances it through ensemble learning with a Differential Evolution based No Negative Constraint Theory. This hybrid approach targets improving wind speed prediction accuracy at two wind farms in Inner Mongolia, China, and Sotavento, Galicia, Spain. The model demonstrates superior forecasting performance compared to traditional and single model approaches. Yaghoubirad et al. (2023) evaluates LSTM, GRU, CNN, and CNN-LSTM models for multistep ahead wind speed and power generation forecasting in Zabol City, Iran, over 6 months, 1 year, and 5 years. GRU was found to be the most accurate, particularly when utilizing multivariate data. Li et al. (2023) introduces a combined forecasting system for optimal wind speed prediction, incorporating data preprocessing, artificial intelligence algorithms, and multi-objective optimization. The findings demonstrate that the proposed integrated forecasting system enhances both the prediction capacity and accuracy of wind speed forecasts. In addition to addressing the limitations of individual models, the developed system selects an appropriate distribution for interval estimation.

These methodologies aim to improve forecast accuracy and reliability by considering the nonlinear and non-stationary characteristics of wind speed data. There's a noticeable trend towards leveraging ensemble learning and advanced optimization techniques to refine forecasting models. Among the research, key challenges include the need for more efficient data preprocessing techniques and the integration of environmental and geographical data into forecasting models. Additionally, there's a gap in developing models that can seamlessly adapt to new data and predict extreme wind events accurately.

Table 2 Summary of Studies on Wind Forecasting

Study	Features	Models	Sample	Findings
Huang et al. (2023)	Wind speed	ANN; LSTM; EvLSTM; EnEvLSTM	1,150 samples of 10-minute and one-hour interval wind speed data of Inner Mongolia, China and 10-minute interval wind speed data of	EnEvLSTM with best MAE, RMSE and MAPE across different interval and experiments

			Sotavento, Galicia from March 3rd, 2017 to March 12th, 2017	
Yaghoubirad et al. (2023)	Weather variables	LSTM; GRU; CNN; CNN-LSTM	6-month, 1-year and 5-year time horizon wind speed data of Zabol City, Iran from 1984 to 2021	GRU with the best MAE and MAPE for both univariate and multivariate models
Li et al. (2023)	Wind speed	ARIMA; GRNN; BPNN; TCNN; LSTM; Elman Neural Network; A combined system based on VMD-BEGA-LSTM, MODOA, MMED-AT and AFSA-ACO	Three sets of data, each consisting of 2592 samples of 10-minute interval wind data of Penglai, China	The proposed model with the best MAE, RMSE and MAPE across different sites

2.3 Critique of the literature and main contributions of the study

Both electricity and wind forecasting are evolving towards more integrated and adaptive forecasting models that incorporate a blend of both traditional methods and advanced computational techniques to improve accuracy and reliability. Despite the considerable advancements that have been achieved in recent years, there are still opportunities for further research, particularly in enhancing real-time data integration, model adaptability, and forecasting under varying conditions.

The electric power generation industry is undergoing significant transformations driven by a complex mix of factors. Historically, fossil fuels such as coal, oil, and natural gas have dominated power generation due to their high energy density and established infrastructure. However, these sources face increasing scrutiny due to their environmental impact, including greenhouse gas emissions and air pollution. As a result, many countries are shifting towards cleaner energy sources to address climate change and sustainability concerns. New energy generation technologies, particularly renewable sources like wind and solar power, are becoming increasingly prominent. Wind power offers a promising solution to reduce reliance on fossil fuels, providing a cleaner and more sustainable alternative. The integration of wind power into the existing energy mix presents both opportunities and challenges. On one hand, it can help reduce carbon emissions and diversify the energy supply. On the other hand, it introduces variability and intermittency issues that need to be addressed to ensure grid stability and reliability.

Many prior studies have been constrained by geographic and market-specific limitations, not taking into full account how different energy mixes affect the effectiveness of forecasting models. There is a notable lack of comparative analysis to evaluate model performance in different context, which is crucial for understanding and improving model generalizability and adaptability in the global energy transition. The limitations of previous studies in handling multiple energy sources simultaneously further underscore the significance of this research.

While much of the existing research has focused specifically on either electricity load forecasting or wind generation forecasting, this study innovatively applies a unified model architecture across both markets. This approach addresses the gap in the previous works, where models are typically specialized for single forecasting tasks. By employing the same set of models for both electricity load and wind generation forecasts, this research demonstrates the versatility and adaptability of these models across different forecasting contexts. To further enhance the generalizability and performance of the models, this study utilizes Bayesian optimization for hyperparameter tuning. Traditional methods of hyperparameter selection often rely on manual adjustments or fixed parameters, which may not be optimal for different markets. By applying Bayesian optimization, the research automatically computes and adjusts hyperparameters tailored to each specific market. This methodology allows for a more precise and context-sensitive model calibration, improving forecasting accuracy and performance across both electricity load and wind generation forecasts.

The application of these techniques suggests potential for further research into the use of universal model architectures and advanced optimization methods in other forecasting areas. Additionally, practitioners can benefit from the insights gained into how models can be generalized and optimized for different forecasting tasks, leading to more efficient and accurate forecasting solutions. As the energy landscape continues to evolve, incorporating advanced forecasting models will be crucial for navigating the complexities of integrating renewable energy sources and ensuring a sustainable and reliable power supply.

To summarize, the research contributes to:

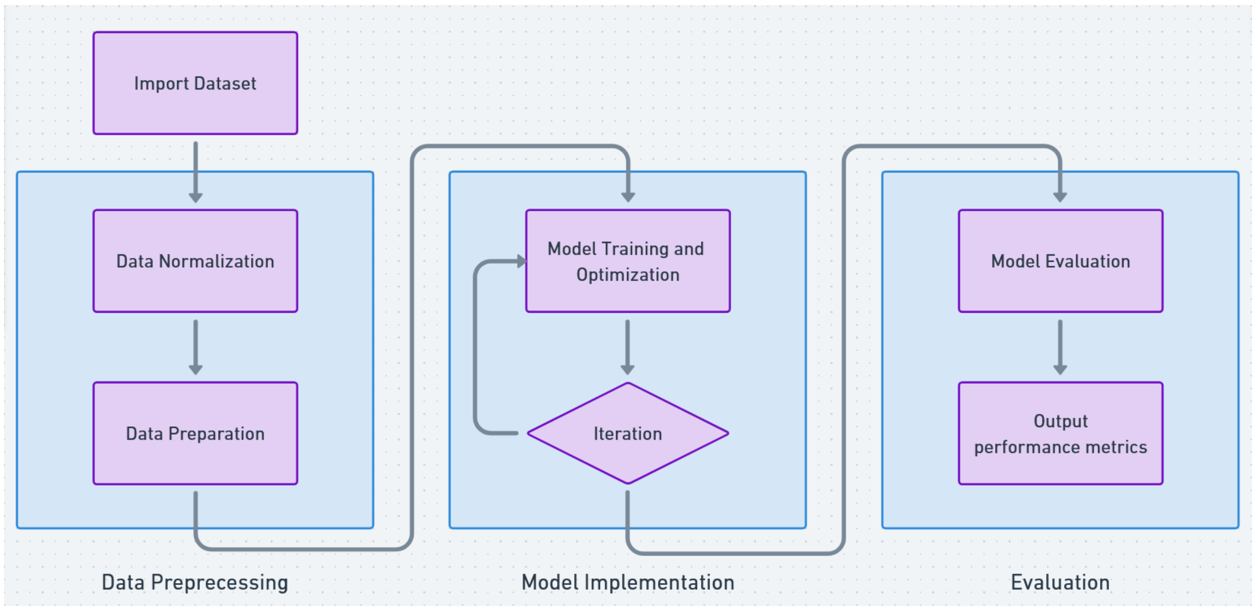
- Considering the task of data forecasting applied to two major energy problems: electricity load and wind generation.
- Implementation and comparison of different optimized deep learning models.
- Use of Bayesian optimization to fine tune the deep learning models.
- Comparison of deep learning models against standard data analytics models.
- Utilization of quarterly-hour and hourly frequency sampling for electricity load and wind generation data.
- Cross-Domain Analysis: Compared models across different energy markets and expanded the applicability of models, addressing the gap in understanding model performance in diverse contexts.

3. Methodology

3.1 Research Design

This research adopts a structured approach to forecast time series data on electricity load and wind generation through the deployment of machine learning models. The following flowchart illustrates the coding process. To maintain consistency, the raw time series datasets are first segmented by different markets, countries and time intervals. Then, the 12 datasets will be imported into the pre-configured code and executed one by one. No modification to the code will be made during this phase, maintaining uniformity in the execution for all datasets. The subsequent sections will provide a detailed description of the stages in the flowchart.

Figure 1 Flowchart of the Experiments



3.2 Data Collection and Preprocessing

The original dataset (Open Power System Data, 2020) is sourced from the Open Power System Data platform, which provides preprocessed and continuous data from the European Network of Transmission System Operators for Electricity (ENTSO-E) Transparency platform. This ensures high reliability and suitability for time-series analysis. It contains various types of time series data relevant for power system modeling, including electricity load as well as wind power generation. The data is aggregated based on country, control area or bidding zone, covering the European Union and some neighboring countries. All variables are available in hourly resolution and some are available in higher resolution, including half-hourly and quarter-hourly formats, spanning from 2015 to mid-2020.

The datasets are first divided into two domains – electricity load and wind generation. And for each domain, the datasets are separated by different countries – Austria (AT), Germany (DE), and Netherlands (NT). Finally, the datasets are further differentiated by fifteen-minute and sixty-minute intervals. At this stage, the datasets used to train our models have undergone an initial differentiation step. Following this, we will perform additional preprocessing to prepare the dataset for direct use in model training section. This preprocessing process is standardized to streamline subsequent steps, as we will need to complete and evaluate a total of 12 datasets for training and comparison purposes. The streamlined procedure ensures consistency and accuracy across all datasets, making it easier to compare results. Additionally, this approach enhances efficiency and reduces the potential for errors. This convenience is particularly important in the context of time series data, which may frequently become available. By implementing a streamlined procedure, we ensure that the addition or updating of data can be managed quickly and efficiently, without compromising the integrity of our analysis.

We leverage Python to encapsulate these processes into reusable, modular components. By defining functions for each step of the data preprocessing, we can easily call these functions as needed, enhancing code reusability and maintainability. This approach exemplifies the procedural programming paradigm, which is centered around the idea of organizing code into procedures or routines that are called upon to perform tasks. Utilizing Python's procedural capabilities ensures that our data processing is not only efficient but also adaptable. As new datasets are introduced or existing datasets are updated, the predefined functions can be easily adjusted or extended without disrupting the overall workflow.

Building upon our standardized, streamlined preprocessing procedures, we have developed specific functions within our Python code to manage and manipulate our datasets effectively. A key function we've defined is called "prepare_data". This function is designed to organize the imported raw data, by allocating 80% of the data for the training set and 20% for the test set. Unlike conventional random sampling or stratification technique, our approach specifically segments the first 80% of the sequential data for training due to the inherent temporal dependencies characteristic of time series data. This method ensures that the temporal features and sequential trends of the data are maintained, which is crucial for the reliability of our time series analysis. Following this data segmentation, we utilize a window of thirty observations to predict the subsequent observation. Consequently, we reshape the data into the format (n, 30, 1), where 'n' represents the number of such 30-observation windows in the dataset. Once the data is segmented and reshaped according to our specifications, the function will finalize by outputting the prepared datasets: training and testing data sets. This restructuring is critical as it aligns with our model's input requirements, allowing us to effectively train our predictive models on time series data while maximizing the use of available information for accurate forecasting.

Since the function have been ready to be deployed, the next step in our workflow is to import the raw data and perform an initial cleanup. As previously noted, the raw dataset comprises observations recorded at both 15-minute and 60-minute intervals, spanning from 2015 to approximately 2020. The dataset contains approximately 200,000 and 40,000 observations, respectively. However, given the limitations of our research infrastructure, handling such large datasets could be overly demanding. To mitigate this, we strategically select only the most recent 10,000 observations from each dataset to use in our model training and testing processes. This selection reduces computational load and storage requirements, thereby alleviating the strain on our research facility. Simultaneously, we ensure the integrity of our data by dropping any observations that contain null values.

Following the cleanup, we proceed to normalize our data using the “MinMaxScaler” function from the “scikit-learn” library. The function is a feature scaling technique used to normalize data. It transforms features by scaling each feature to a given range, typically 0 and 1. This is achieved by subtracting the minimum value of the feature and then dividing by the range. The formula is as follows:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

This normalization step is straightforward yet critical for the performance of machine learning models.

After normalizing the data, we call the previously defined “prepare_data” function to split the data into designated training and test sets. This marks the completion of data preparation and preprocessing phases, setting the stage for model training.

3.3 Model Implementation

In this section, with the data adequately preprocessed, we transition to the model implementation part of the research. This section involves applying various machine learning algorithms to the prepared datasets and tuning model parameters.

We define every model and set the parameter search space before model optimization and training phase to maintain clarity and consistency. Here is a brief overview of each model implemented in this study, detailing their specific parameters.

3.3.1 Convolutional Neural Network

The Convolutional Neural Network (LeCun et al., 2015) is known for their ability to perform automatic feature extraction. CNNs utilize layers of convolutional filters that apply over the input data, enabling the model to learn increasingly complex features at each layer. This eliminates the need for manual feature selection, reducing the risk of human bias and error. In time series analysis, CNNs can identify and learn important features from raw data sequences without pre-defined assumptions. Once a feature is learned, a CNN can recognize it in any part of the data sequence.

To implement the CNN model, we define a “create_cnn_model” function, designed to encapsulate the architecture of the CNN tailored for time series forecasting. The model begins with a Conv1D layer since we have sequential data with one spatial dimension. The layer uses a specified number of filters and a kernel size to scan through the input data. The “relu” activation function is employed to introduce non-linearity, enabling the model to learn more complex patterns.

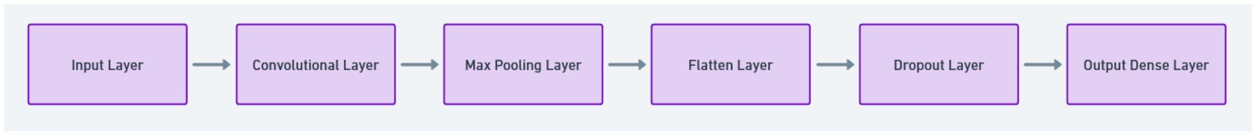
Mathematically, this convolution operation computes the inner product between a subsequence of the input data X (with length k, the kernel size) and the filter w:

$$C_i = \sum_{i=1}^{k} w_i \cdot X_i \tag{2}$$

Following the convolutional layer, a MaxPooling1D layer reduces the dimensionality of the data, which helps in reducing computation as well as controlling overfitting by abstracting the features extracted by the convolutional layers. Then, a Flatten layer is used to transform the pooled feature map into a single linear vector and a Dropout layer is used to prevent overfitting. The architecture concludes with a Dense layer that uses a "linear" activation function. Finally, the model is compiled with the "adam" optimizer with a learning rate of 0.001, and the loss function is set to mean squared error.

After defining the function, we set up the parameter search space for optimization targeting the CNN model's filter, kernel size, pool size, and dropout rate. Filter determines the number of filters in the convolutional layers, ranging from 16 to 256. Kernel size defines the size of the window to be used in each convolutional filter, set between 2 and 8. Pool size dictates the size of the pooling window, ranging from 2 to 4. Dropout rate controls the proportion of neurons to drop out during training, set between 0.0 and 0.5. With the definition of the function and the search space, we have completed the architecture setup for the CNN model as shown in Figure 2.

Figure 2 Architecture of the CNN



3.3.2 Temporal Convolutional Neural Network

The Temporal Convolutional Neural Network (Bai et al., 2018) is a variant of the traditional convolutional neural network specifically designed to handle sequence prediction problems more effectively, particularly in time series forecasting. TCNNs use causal convolutions, which ensure that the output at a given time is only determined by past inputs. This is a significant advantage for making predictions in real-time scenarios, where future data cannot be assumed available.

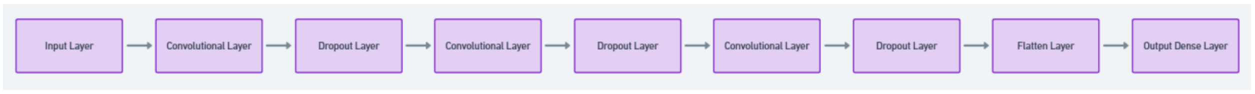
Similarly with CNN, we define a "create_tcnn_model" function, which includes the architecture of our TCNN. But as a relatively newer model architecture compared to CNN, there are fewer ready-to-use libraries available for TCNN and we will have to build TCNN up based on the code of CNN. The model will include several Conv1D layers with a causal padding option, ensuring that the convolution output for a specific timestamp can only depend on current and past data. Mathematically, for a given input sequence X, a dilated causal convolution at time step t with a kernel size k, dilation rate d, and filter weights w is defined as:

$$y(t) = \sum_{i=0}^{k-1} w_i \cdot x_{t-d \cdot i} \tag{3}$$

In our model, the dilation rate starts at 1 and increases with each subsequent layer. For instance, when d=1 for the first layer, it becomes 2 for the second layer, 3 for the third layer, and so on, expanding the receptive field exponentially. Each convolutional layer is followed by an activation layer set to "relu", introducing non-linearity into the model. This is coupled with a dropout layer to prevent overfitting by randomly omitting a proportion of the feature detectors during training. After, this is followed by a Flatten layer, and a Dense layer with a "linear" activation function. Finally, the model is compiled using the "adam" optimizer, with a learning rate of 0.001 and mean squared error as the loss function.

The parameter space for TCNN tuning includes the settings of the number of convolutional layers, filter, kernel size, dilation rate, and dropout rate. Specifically, the number of convolutional layers varies from 2 to 4, filters are adjusted between 16 and 256, kernel size extends from 2 to 8, dilation rate is set from 1 to 2, and dropout rate spans from 0.0 to 0.5. We have now completed the preparation of our second model, TCNN, as shown in Figure 3.

Figure 3 Architecture of the TCNN



3.3.3 Long Short-Term Memory Network

The Long Short-Term Memory (Hochreiter, 1997) network is a type of recurrent neural network (RNN) specifically designed to address the shortcomings of traditional RNNs, particularly in handling long-term dependencies. Traditional RNNs often struggle with the vanishing gradient problem, where information from earlier input data is lost as it propagates through the network. LSTMs address this issue with their unique architecture of gates that regulate the flow of information. LSTMs contain memory cells that can maintain information in memory for long periods of time. Each LSTM memory cell has three gates: the forget gate, the input gate, and the output gate, which manage the flow of information. For each time step t, the memory cell stat C_t is updated based on the current input X_t and the previous hidden state H_{t-1}, as described as follows:

1. Forget Gate: Decides which part of the previous cell state to retain or forget.

$$f_t = ReLU(X_t U_f + H_{t-1} W_f) \quad (4)$$

2. Input Gate: Decides how much new information should be added to the cell state.

$$i_t = ReLU(X_t U_i + H_{t-1} W_i) \quad (5)$$

3. New Candidate Memory: Generates the candidate new information for the memory cell.

$$N_t = tanh(X_t U_c + H_{t-1} W_c) \quad (6)$$

4. Cell State Update: The cell state is updated based on the forget and input gate outputs.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot N_t \tag{7}$$

5. Output Gate: Controls the information flow from the cell to the next hidden state.

$$O_t = ReLU(X_t U_o + H_{t-1} W_o) \tag{8}$$

6. Hidden State Update: The hidden state is passed to the next LSTM layer or the output layer.

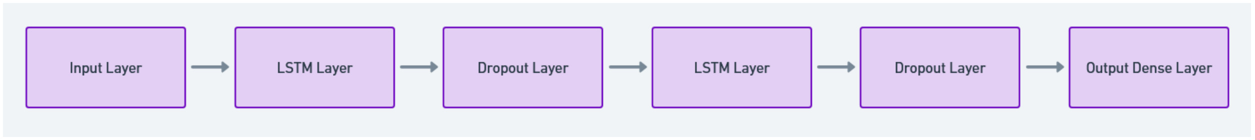
$$H_t = O_t \cdot \tanh(C_t) \tag{9}$$

This feature of LSTM is particularly useful in applications like anomaly detection in time series data. Nevertheless, LSTMs are computationally intensive compared to simpler RNNs or other deep learning models. This is due to the complex architecture of LSTM cells. This complexity can lead to longer training times and higher computational costs, especially with large datasets or deep networks.

In our code, the architecture of LSTM is defined in the “create_lstm_model” function. The model begins with a first LSTM layer. This first LSTM layer processes input sequences and ensures that the output includes sequences, necessary for chaining multiple LSTM layer. Following the first LSTM layer, a dropout layer is introduced to prevent overfitting. Then a subsequent LSTM layer, receives sequences from the previous layer, further refining the model’s ability to learn from the data. The output from this layer does not return sequences, preparing for a final dense output layer. A dense layer with a “linear” activation function produces the final output of the model. At last, the model is compiled using the “adam” optimizer, with a learning rate of 0.001 and mean squared error as the loss function.

In the parameter search space, the number of neurons for the first and second LSTM layers is individually defined, allowing the model to capture complexities in the data at different levels of abstraction. Specifically, the number of neurons for the first LSTM ranges from 16 to 128, while the second spans from 8 to 64. Additionally, the "Dropout rate" is adjustable within a range from 0.0 to 0.5. The architecture of the LSTM could be summarized as shown in Figure 4.

Figure 4 Architecture of the LSTM



3.3.4 Bidirectional Long Short-Term Memory Network

The Bidirectional Long Short-Term Memory (BiLSTM) network is an enhancement of the LSTM model that involves training two LSTMs on the input sequence: one with the input in natural order and the other in reverse order. This approach allows the model to gather information from both past and future states simultaneously. The outputs of the forward and backward layers at each time step t are concatenated to form a single output vector. Mathematically, the hidden state at time step t in the BiLSMT is:

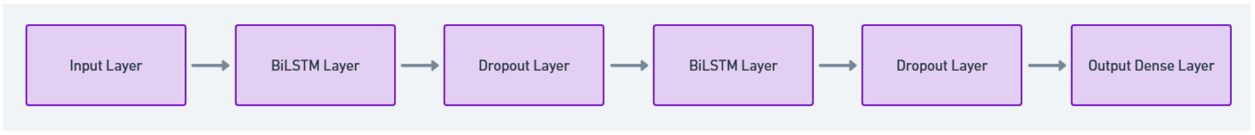
$$H_t^{BiLSTM} = [H_t^{forward}, H_t^{backward}] \tag{10}$$

By processing data from both directions, BiLSTMs capture a richer understanding of the context and thus capture dependencies and features that may be missed by single-direction LSTMs. Like LSTM, BiLSTM requires longer training times and greater demands on computational resources.

The overall architecture of our BiLSTM model is similar to our LSTM model. Leveraging the robust capabilities of the "Keras" library, the definition of the "create_bilstm_model" function closely mirrors the structure used in the "create_lstm_model" function. The first BiLSTM layer is a Bidirectional wrapper applied to an LSTM layer with hyper-tunable number of neurons in each direction. This configuration allows the layer to process the input sequence both forwards and backwards. The second BiLSTM layer similarly utilizes the Bidirectional wrapper on an LSTM layer with another hyper-tunable number of neurons. Dropout layers are interspersed between BiLSTM layers to prevent overfitting. The final layer is a Dense layer with a "linear" activation function. Lastly, the model is compiled with the "adam" optimizer with a learning rate of 0.001, and the loss function is mean squared error.

The parameter search space designated for the BiLSTM model is similar to that of the LSTM as well. Specifically, the number of units in the first BiLSTM layer is set between 16 and 128, while the second can range from 8 to 64. Additionally, the dropout rate is configurable from 0.0 to 0.5. The architecture of the BiLSTM could be summarized as shown in Figure 5.

Figure 5 Architecture of the BiLSTM



3.3.5 Gated Recurrent Unit

The Gated Recurrent Unit (Cho, 2014) is a type of RNN similar to the LSTM network but designed to be simpler and more efficient in certain scenarios. GRUs have only two gates, update and reset. The following formulas describe the key operations within each GRU unit:

1. Update Gate: The update gate controls how much of the previous hidden state should be carried forward to the current time step.

$$z_t = ReLU(X_t U_z + H_{t-1} W_z) \tag{11}$$

2. Reset Gate: The reset gate decides how much of the previous hidden state should be forgotten when computing the candidate hidden state. And it controls the contribution of the previous hidden state to the candidate hidden state computation.

$$r_t = ReLU(X_t U_r + H_{t-1} W_r) \tag{12}$$

3. Candidate Hidden State: The candidate hidden state is computed based on the reset gate's influence on the previous hidden state. The reset gate allows the model to determine how much of the previous hidden state to use in computing the new candidate hidden state.

$$\tilde{h}_t = tanh(X_t U_h + (r_t \cdot H_{t-1}) W_h) \tag{13}$$

4. Hidden State Update: The final hidden state is update by combining the previous hidden state and the candidate hidden state using the update gate. This formula balances the influence of past and new information, controlled by the update gate.

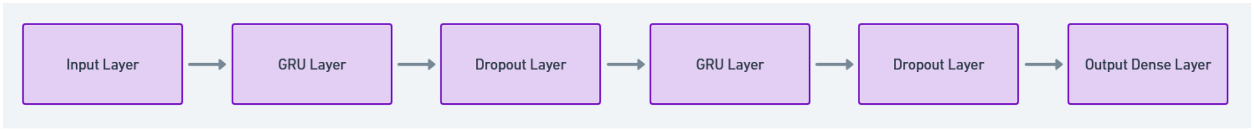
$$h_t = (1 - z_t) \cdot H_{t-1} + z_t \cdot \tilde{h}_t \tag{14}$$

GRUs require fewer parameters than LSTMs because they lack an output gate, making them faster to train while still capturing dependencies in sequence data effectively. This reduction in complexity often leads to faster training times without a significant decrease in performance, especially in tasks where long-term dependencies are less critical. The simpler architecture of GRUs, while beneficial in terms of reduced complexity, also means they offer less control over what the network chooses to remember or forget compared to LSTMs. But overall, GRUs could provide a good balance between computational efficiency and model performance.

When defining "create_gru_model" function, which also benefit from the scikit-learn library, the overall process is similar to that of the LSTM in the previous section. A first GRU layer processes the input sequence and provides a temporal dimension output that is necessary for chaining to the next recurrent layer. A second GRU layer further refines the ability of the model to learn from the temporal data, processing the sequence output from the previous GRU layer. Interspersed between the GRU layers, dropout layers help mitigate the risk of overfitting. A dense layer with a single neuron concludes the model. At last, the model is compiled with the "adam" optimizer, with a learning rate of 0.001 and mean squared error as the loss function.

The hyperparameter search space defined for the GRU model is identical to that of the LSTM. Notably, the number of neurons in the initial GRU layer varies between 16 and 128, and for the subsequent layer, it ranges from 8 to 64. The dropout rate is adjustable, with values spanning from 0.0 to 0.5. The architecture of the GRU is as shown in Figure 6.

Figure 6 Architecture of the GRU



3.3.6 Bidirectional Gated Recurrent Unit

The Bidirectional Gated Recurrent Unit (BiGRU) model extends the traditional GRU architecture. BiGRUs process sequences both forwards and backwards, enabling the model to access past and future context. At each time step t, the hidden states from both directions are concatenated:

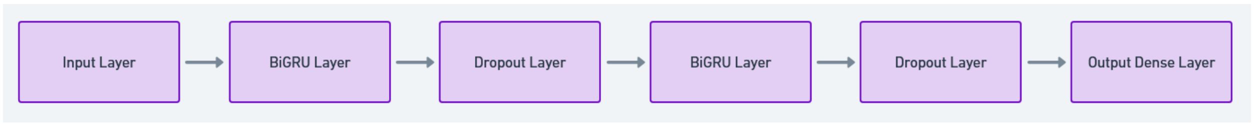
$$H_t^{BiGRU} = [H_t^{forward}, H_t^{backward}] \tag{15}$$

Compared to BiLSTMs, BiGRUs typically have fewer parameters due to the simpler gating mechanisms of GRUs. Despite being more efficient than BiLSTMs, BiGRUs still require more computational resources than their unidirectional counterparts due to processing sequences twice.

The definition process for the "create_bigru_model" function aligns closely with the other three RNN models previously discussed. The model begins with a Bidirectional wrapper applied to a GRU layer. After each BiGRU layer, Dropout layers help reduce overfitting. The second BiGRU layer adds further depth to the model's ability to process information bidirectionally. A Dense layer with "linear" activation serves as the output layer. The model is compiled using the "adam" optimizer with a learning rate of 0.001 and the loss function is mean squared error.

The parameter search space for the BiGRU model is consistent with that of other RNN models previously discussed. Specifically, it allows for the number of neurons in the first BiGRU layer to range from 16 to 128 and in the second BiGRU layer from 8 to 64. Additionally, the dropout rate is adjustable from 0.0 to 0.5. The architecture of the BiGRU is as shown in Figure 7.

Figure 7 Architecture of the BiGRU



3.3.7 Deep Neural Network

The Deep Neural Network (DNN) is a standard, straightforward multilayer perceptron (MLP) consisting of fully connected layers. Each layer performs a linear transformation of the input followed by a non-linear activation function. Mathematically, the operation in a Dense layer is described as:

$$y_i = f(\sum_{j=1}^{n} w_{ij}x_j + b_i) \tag{16}$$

Where:

- y_i is the output of the i-th neuron,
- w_{ij} is the weight associated with the connection between the j-th input and the i-th neuron,
- x_j is the j-th input to the neuron,
- b_i is the bias term,
- $f(\cdot)$ is the activation function.

In our case, each hidden layer uses the Rectified Linear Unit (ReLU) activation function, which allows the model to capture more complex patterns in the data by only passing positive values through the layers.

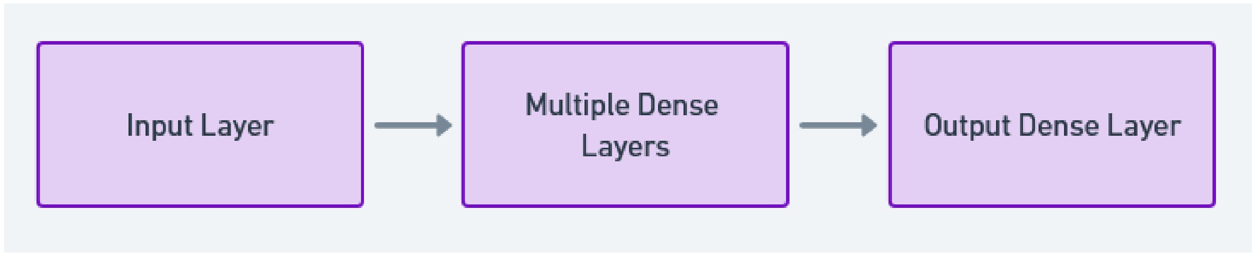
$$ReLU(x) = \max(0, x) \tag{17}$$

This type of neural network is fundamental in deep learning, offering a baseline architecture from which more complex models evolve. With multiple hidden layers, DNNs can learn complex patterns and relationships in the data. And DNNs are inherently scalable, capable of handling large datasets and expanding in complexity as needed by adding more layers or neurons. Unlike RNNs or CNNs, DNNs treat input features as independent and identically distributed. They fail to recognize the context or sequence in the input data, which is often crucial for more complex tasks.

The structure of the DNN is defined to be flexible, allowing adjustments in the number of layers and units to suit different complexity requirements and dataset characteristics. The model starts with a Dense layer, processing the input data, with the activation function of 'relu'. Subsequently, a loop constructs additional layers as part of the optimization process to determine the optimal number of layers for the model. Each subsequent layer uses the same number of units and activation function, stacking up the network's capability to learn more complex patterns. The model ends with a final Dense layer with one neuron and a "linear" activation function. Finally, the model is compiled with the "adam" optimizer, with a learning rate of 0.001 and the loss function of mean squared error.

The parameter search space for a DNN model is relatively simple, primarily involving the adjustment of the model's depth, which is set to vary between 1 and 3 layers. The parameter "units" determines the number of neurons per layer, with a range extending from 8 to 256. The architecture of the DNN is as shown in Figure 8.

Figure 8 Architecture of the DNN



3.3.8 k-Nearest Neighbors

The k-Nearest Neighbors (Fix, 1985) algorithm is a simple, versatile, and widely used machine learning method based on feature similarity. Unlike many other machine learning algorithms that require a training phase to learn a model, kNN makes predictions using the entire dataset as the “model”. Nonetheless, one of the major drawbacks of kNN is its computational inefficiency, especially with large datasets. Every prediction requires a distance calculation to all training points, which can be computationally expensive and slow.

In kNN regression, the model finds k-nearest neighbors to a given input point using a distance metric, and then makes a prediction by averaging the target values of these neighbors. Mathematically, the predicted value for an input is:

$$\hat{y}(x) = \frac{1}{k}\sum_{i=1}^{k} y_i \tag{18}$$

The code for kNN is much simpler than the previously mentioned deep learning models. We initialize the “KNeighborsRegressor” from the scikit-learn library to build a basic kNN model and then we define the parameter search space. We set the range for “k” from 3 to 10. This parameter determines the number of nearest neighbors to be considered when making predictions.

We then configure the “weights” parameter, which presents two categorical options: “uniform” and “distance”. Under the 'uniform' setting, each neighbor contributes equally to the final prediction. Conversely, the 'distance' option assigns greater importance to nearer neighbors. The third parameter “p” is the power parameter for the Minkowski metric.

- When p=1, the distance metric used is Manhattan distance:

$$d(x, x') = \sum_{i=1}^{n} |x_i - x_i'| \tag{19}$$

- When p=2, it is the Euclidean distance:

$$d(x, x') = \sqrt{\sum_{i=1}^{n} (x_i - x_i')^2} \tag{20}$$

The parameter space for the kNN model has been established, and we will leave it to the optimization process to decide the optimal parameters.

3.3.9 Decision Tree

The Decision Tree (Breiman, 2017) is a non-parametric supervised learning method used for both classification and regression tasks. Decision trees are able to manage complex, non-linear relationships. A decision tree builds a tree-like structure by splitting the data at each node based on the feature that increases the homogeneity of the target variable within each region. In regression trees, the goal is to minimize the variance of the target variable within each region after splitting the data. Mathematically, for a given split at node t, the split criterion is based on reducing the Mean Square Error. The MSE at a node is calculated as:

$$MSE_t = \frac{1}{N_t} \sum_{i=1}^{N_t} (y_i - \bar{y}_t)^2 \tag{21}$$

The decision tree evaluates every possible split and selects the one that maximizes the reduction in MSE, which is also referred to as variance reduction:

$$\Delta MSE = MSE_{parent} - \left(\frac{N_{left}}{N_{parent}} MSE_{left} + \frac{N_{right}}{N_{parent}} MSE_{right}\right) \tag{22}$$

The tree continues to split until one of the stopping criteria is met: the maximum number of levels in tree, minimum samples of a node to perform a split, and minimum number of samples a leaf node must have. Once the tree has reached a leaf node, it predicts the output by averaging the target values of the samples in that node:

$$\hat{y}(x) = \frac{1}{N_{leaf}} \sum_{i=1}^{N_{leaf}} y_i \tag{23}$$

In contrast to various neural network models, decision trees have inherent limitations in extrapolating beyond the scope of the training data. This characteristic presents a significant challenge in time-series forecasting, where it is frequently necessary to predict future values that extend past the range observed within the training dataset.

When setting up the base model, we utilize the “DecisionTreeRegressor” from the scikit-learn library. Correspondingly, we establish the parameter search space to facilitate the exploration and optimization of model parameters. We set the maximum depth parameter range between 3 and 20. A deeper tree is capable of capturing more intricate patterns within the dataset. However, it also runs the risk of learning noise rather than the underlying data trends. Conversely, setting a shallower tree enhances the model's generalizability but may not capture enough complexity from the data. We set the range for “min_samples_split” between 2 and 20. This parameter specifies the minimum number of samples required to split an internal node within the decision tree. Higher values restrict the tree’s complexity while lower values facilitate a more liberal growth of the tree. The last parameter “min_samples_leaf” is set within a range from 1 to 10. This parameter determines the minimum number of samples required in a leaf node. A lower threshold in this range may capture highly specific patterns, and conversely a higher value promotes better generalization.

3.4 Model Optimization and Training

Upon the completion of defining all required models, we proceed to the model optimization and training. Bayesian optimization (Snoek et al., 2012) is specifically designed to identify an optimal set of parameters that achieve minimal values for the loss function. This technique utilizes a probabilistic model to predict the performance of the models given different hyperparameter settings and iteratively updates the model based on the outcomes of the evaluations. This method is effective compared to random or grid search approaches, particularly under conditions where model performance evaluations are computationally demanding or when navigating a vast hyperparameter space.

The optimization process can be described in the following steps:

1. Gaussian Process Surrogate Model: In Bayesian Optimization, a Gaussian Process (GP) is commonly used as a surrogate model for the objective function $f(x)$. Given the current evaluations of the function at different hyperparameter values, the GP predicts the function value at any new point x by providing a mean prediction $\mu(x)$ and uncertainty $\sigma(x)$. The GP regression model can be expressed as:

   $$f(x) \sim GP(\mu(x), k(x, x')) \tag{24}$$

   Where $k(x, x')$ is the kernel function representing the correlation between different points in the search space.

2. Acquisition Function: The acquisition function $\alpha(x)$ selects the next set of hyperparameters to evaluate by balancing high uncertainty and low mean prediction. Common acquisition functions used include:

   - Expected Improvement (EI):

     $$\alpha_{EI}(x) = E[max(0, f(x) - f(x^+))] \tag{25}$$

     Where $f(x^+)$ is the best observed objective function value.

   - Upper Confidence Bound (UCB):

     $$\alpha_{UCB}(x) = \mu(x) + \kappa\sigma(x) \tag{26}$$

     Where $\mu(x)$ is the predicted mean, $\sigma(x)$ is the uncertainty, and κ is a parameter that controls the balance between exploration and exploitation.

3. Objective Function Evaluation: Once the next set of hyperparameters is selected using the acquisition function, the model is trained using those hyperparameters, and the validation performance is evaluated.
4. Cross-Validation: Typically, we will apply cross-validation to evaluate models, such as k-fold cross-validation. In k-fold cross-validation, the data is shuffled and split into 'k' subsets, where each subset gets a turn at being the testing set while the others are used for training. However, this method is unsuitable for time series data. Time series observations are chronologically ordered and shuffling them disrupts their inherent temporal nature. The sequence of events is crucial in time series analysis as the value at a given time point is often dependent on previous values. Randomly splitting the data into folds for cross-validation can lead to scenarios where the model is inadvertently trained on future data before learning from past data, leading to unrealistic performance estimates and potential lookahead biases. Given the limitations of traditional cross-validation methods for time series data, we choose the "TimeSeriesSplit" method provided by the scikit-learn library for our optimization process. "TimeSeriesSplit" ensures that the training set always contains observations that occur prior to those in the test set.
5. Optimization Process: The process continues for a specified number of iterations, refining the hyperparameters to optimize the objective function. Mathematically, at each iteration, Bayesian Optimization selects hyperparameters by optimizing the function:

$$x_{next} = \arg\max \alpha(x|\mu(x), \sigma(x)) \tag{27}$$

   Where $\alpha(x)$ is the acquisition function, $\mu(x)$ is the predicted mean function from the GP, and $\sigma(x)$ is the predicted uncertainty.

And since the Bayesian optimization from "skopt" library supports "callback" parameters during the fitting procedure, we incorporate commands to record the duration of each iteration as well as the optimal parameters identified at the end of each iteration. Furthermore, we track and compile the total time to complete the optimization process for each model. This tracks a thorough record of the effects of parameter tuning throughout the optimization timeline, thus improving our understanding of the computational efficiency and effectiveness of the model tuning phase.

Having established the foundational elements, we proceed to optimize our models. We utilize "BayesianSearchCV" from the "skopt" library, an advanced method that employs Bayesian optimization techniques for hyperparameter tuning. By defining each model and its corresponding parameter search space beforehand, we can seamlessly integrate "BayesianSearchCV" into our workflow. For the cross-validation method, we specify "TimeSeriesSplit" to ensure the temporal sequence of the dataset is considered. For other parameters, we limit the number of iterations to 30 to balance between thoroughness in the search and computational efficiency. We use Mean Squared Error (MSE) as the scoring metric and set the random state to 42 to ensure the reproducibility of our results. After obtaining the optimal parameters through the fitting of the Bayesian optimizer, the next step is to retrain the model utilizing these hyper-tuned parameters. Finally, the best model retrained with the optimized parameters will be saved for future use.

At the end of this phase, we will have robust models that can effectively predict outcomes based on the trends and patterns identified in the training data.

3.5 Evaluation Metrics

Now that we have the best models using the optimal parameters, the next step is to evaluate these models. This evaluation is crucial for determining which model yields the most accurate forecasts for each specific dataset. During the data preparation phase, the dataset was carefully partitioned, reserving 20% as a test set. This subset, which the models have never seen during before, can reflect the models' ability to predict the new, unseen data. For a comprehensive evaluation, four key metrics have been selected: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Deviation (MAD), and Mean Absolute Scaled Error (MASE).

Root Mean Squared Error is a widely used measure of the differences between values predicted by a model and the values actually observed. It squares the errors before averaging them, which penalizes larger errors.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{28}$$

Mean Absolute Error measures the average magnitude of the errors in a set of predictions, without considering their direction. It’s the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{29}$$

Similar to MAE, Mean Absolute Deviation measures the average absolute deviation of data points relative to a dataset’s mean. It is a measure of dispersion.

$$MAD = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{y}_i - \mu_y\right| \tag{30}$$

Mean Absolute Scaled Error measures the accuracy of forecasts relative to a naïve benchmark prediction, typically the value from the previous period. It is especially useful because it is scale-independent and can handle data with zeros. A MASE less than one indicates that the model performs better than a naïve baseline model. It is particularly suitable for time-series data because it normalizes the error based on the historical data variability.

$$MASE = \frac{\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|}{\frac{1}{n-1}\sum_{i=2}^{n}|y_i - y_{i-1}|} \tag{31}$$

We will compare the performance of each model within individual markets to identify which model predicts the most accurate in a specific market context. Subsequently, this analysis will be extended to compare the best-performing models from each market against each other across different markets. We can thus identify the most effective model for a particular market and also assess the generalizability and robustness of the model across different market environments. Moreover, we will have valuable insights into the relative strengths and weaknesses of the models, which will help make informed decisions about model selection and deployment in different market situations.

4. Results

4.1 Descriptive Analysis of Data

We will begin our analysis with the dataset used in the study. To recapitulate, the dataset comprises 12 distinct subsets, derived from a factorial combination of three variables: three countries (Austria, Germany, and Netherlands), two market types (electricity load and wind generation), and two observation intervals (15 and 60 minutes). The Table 3 below provides a detailed description of the data, demonstrating key statistical measures such as the mean, standard deviation, minimum, and maximum values, along with the 25th, 50th (median), and 75th percentiles.

Table 3 Description of the datasets

Market	**Country**	**Interval**	**Mean**	**Standard Deviation**	**Min**	**25%**	**50%**	**75%**	**Max**
Load	**Austria**	15 minutes	6332	1137	4000	5322	6271	7406	8994
		60 minutes	6872	1385	4004	5745	6829	7815	10779
	Germany	15 minutes	51306	9015	32669	43549	50702	59754	69079
		60 minutes	53825	9842	31923	45786	53398	61629	75551
	Netherlands	15 minutes	11878	1468	7412	10758	11835	12813	16196
		60 minutes	12474	1942	7451	10967	12172	13782	17782
Wind	**Austria**	15 minutes	616	671	4	112	348	912	2792
		60 minutes	828	792	4	165	548	1336	2969
	Germany	15 minutes	9103	7240	126	3663	6861	12433	43702
		60 minutes	14301	10497	136	5852	11637	20744	46064
	Netherlands	15 minutes	581	529	0	145	420	877	2097
		60 minutes	822	637	0	261	676	1339	2199

In all three countries, the data indicate that electrical load exceeds wind generation by a large margin. This observation underscores a significant reliance on other traditional energy within these countries. Such a pattern highlights the current energy dynamics where traditional power remains predominant over renewable wind energy sources within these national grids. From the Table 3 we can also find that Germany exhibits the highest levels of electricity load and wind generation among the datasets on average. Additionally, the Netherlands demonstrates a higher average electricity load compared to Austria, although their average wind generation levels are approximately equivalent. The standard deviation also reveals significant fluctuations in every dataset. Generally, these variations suggest a high level of volatility inherent in both markets. Notably, the variability in wind generation is more pronounced than that in electricity load. This highly variability in wind generation can be attributed to the inherently unpredictability of wind speeds and patterns.

In addition to the above, it is crucial to mention a particular aspect of the dataset from the Netherlands that requires careful consideration. The minimum value recorded in the Netherlands wind data is 0. The presence of 0 values poses a challenge. Ignoring these records could disrupt the temporal continuity of the time series while imputing these zeros with substitute values introduces the dilemma of choosing an appropriate fill value. Therefore, we decide to retain these zero values in the dataset. This decision, however, impacts the selection of appropriate performance metrics for model evaluation. Specifically, it precludes the use of Mean Absolute Percentage Error (MAPE) as an evaluation metric. MAPE is typically calculated by dividing the absolute error of the prediction by the actual values, which becomes undefined when actual values are zero. To address this, some implementations, including Python's scikit-learn library, modify the MAPE calculation by incorporating a small positive number in the denominator to prevent division by zero. While this adjustment ensures that the metric is always defined, it can lead to disproportionately high percentage errors when actual values are close to zero, severely skewing the overall MAPE. Given these considerations, we exclude MAPE from our evaluation criteria. Instead, we selected other metrics such as RMSE, MAE, MAD and MASE, which do not suffer from the same limitations and provide a more reliable assessment of model performance in the presence of zero values.

Following the statistical summary, we will have a preliminary inspection of the visual graphs. The visualizations will offer further insights into the data, helping to identify any underlying patterns, trends, or anomalies that may not be immediately apparent from the statistical metrics alone.

Figure 9 and Figure 10 presented below illustrate the time series visualizations of the datasets. These figures are designed to present the temporal patterns and fluctuations observed within the data over the 60-minute intervals. This is because data collected at 15-minute intervals essentially provides a higher-resolution view of the same phenomena captured in the 60-minute data, whereas we now only observe the general trend, and therefore focus on the latter in the trend visualization to improve clarity.

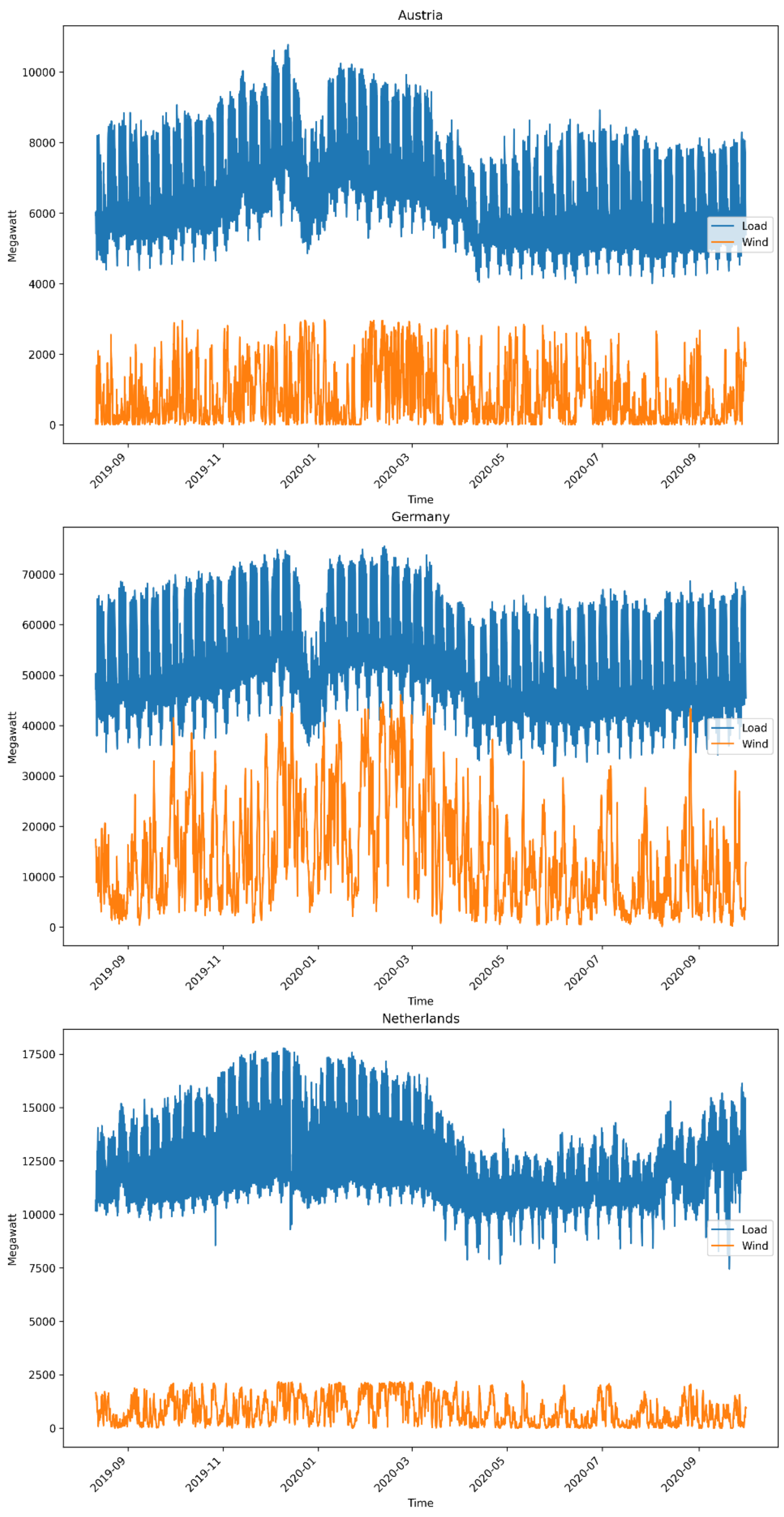


Figure 9 Time plot for 10000 observations

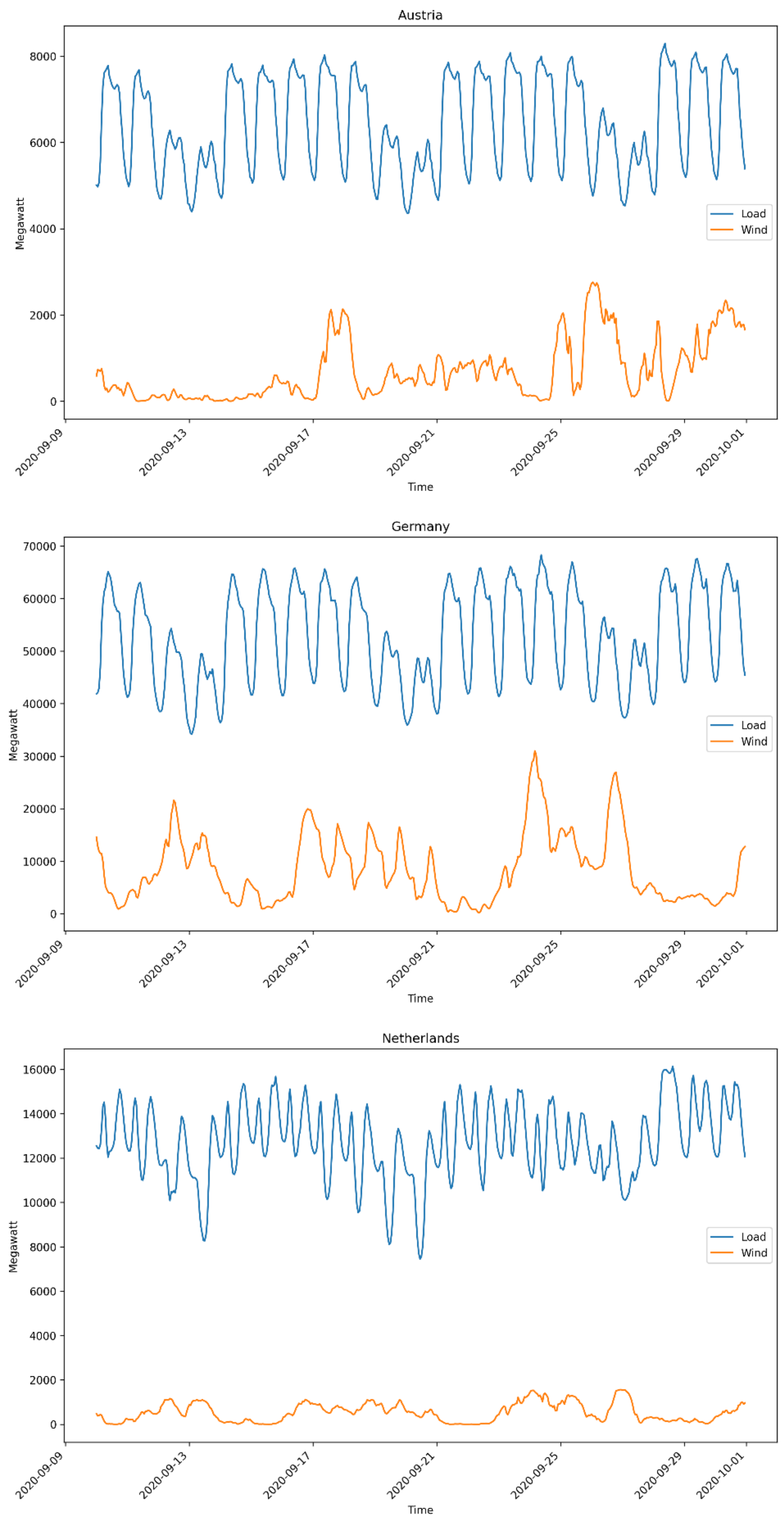


Figure 10 Time plot for 504 observations

Figure 9 presents a comprehensive visualization based on the entire dataset, encompassing 10,000 observations. The coverage confirms the initial statistical observation that the electric load surpasses the wind generation by a considerable amount across the dataset. Furthermore, it is evident that electric load patterns exhibit greater regularity when compared to the wind generation.

To facilitate a more detailed inspection of the data, Figure 10 reduces the scope to 504 observations, equivalent to 504 hours or approximately 21 days. This temporal contraction significantly enhances the visibility of underlying patterns without sacrificing the broader temporal context. The strategic sampling allows patterns to be discerned more readily by the naked eye. Within this more focused timeframe, the electricity load trends in Austria and Germany reveal discernible cycles. There is a conspicuous daily cycle, marked by fluctuations every 24 hours, and a more pronounced weekly cycle evident every seven days. These periodic trends are less apparent in the Netherlands data. In contrast, the wind generation displays an expected lack of discernible pattern, aligning with the variable nature of wind which does not follow strict cyclical patterns like those observed in electricity data.

4.2 Model Performance Evaluation

Upon collecting all evaluation results, we systematically aggregate and analyze them. Our primary objective is to identify the best-performing model for each dataset, primarily considering the RMSE as the principal metric here. Secondary criteria, including additional metrics and the computational efficiency reflected by the time taken under optimization process, are also taken into account. The tables and figures below present the evaluation results that we have compiled.

Table 4 Evaluation Results for Austria Load (15-Minute Intervals)

Model	**RMSE**	**MAE**	**MAD**	**MASE**	**Optimization Time Used (Minutes)**
CNN	164.22	128.31	1008.30	1.93	5.07
TCNN	101.02	78.38	988.29	1.18	271.32
LSTM	202.28	170.68	995.37	2.57	41.08
BiLSTM	129.36	99.06	1014.16	1.49	83.43
GRU	90.06	72.37	998.05	1.09	40.46
BiGRU	122.81	98.70	992.23	1.49	76.80
DNN	74.22	57.15	1002.09	0.86	3.92
kNN	123.96	97.71	1005.88	1.47	0.33
Tree	63.07	46.77	1005.82	0.71	1.17

Table 4 presents the evaluation results of electrical loads for 15-minute intervals for Austria. The Decision Tree model shows the lowest RMSE, MAE and MASE values at 63.07, 46.77 and 0.71 respectively. The Tree also demonstrates considerable efficiency in terms of optimization time, requiring only 1.17 minutes. While this duration is slightly longer than that for the kNN, it is substantially quicker compared to the neural network models. The DNN also shows commendable performance among deep learning models with a low optimization time of 3.92 minutes. Yet overall, the decision tree proves to be the most effective model for this dataset. And as illustrated in Figure 11 and Figure 12 below, each model approximately captures the underlying trend of the data. Figure 13 and Figure 14 depict plots that compare the predicted values from the models with the actual observations for the last 24 hours of the dataset. The last 96 observations are included here due to the 15-minute intervals. In the later section, which addresses the data from 60-minute intervals, comprises only 24 observations. Due to the higher resolution, as depicted in Figure 14, both the actual and predicted value curves lack smoothness, which may contribute to the higher error.

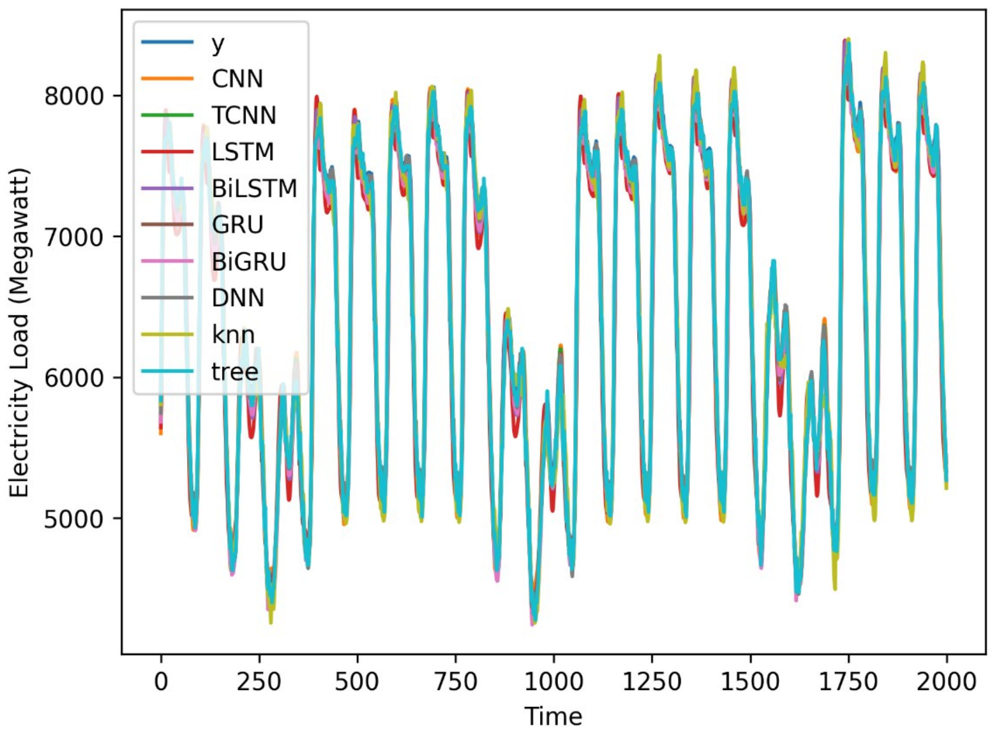


Figure 11 All Results for Austria Load (15-minute intervals)

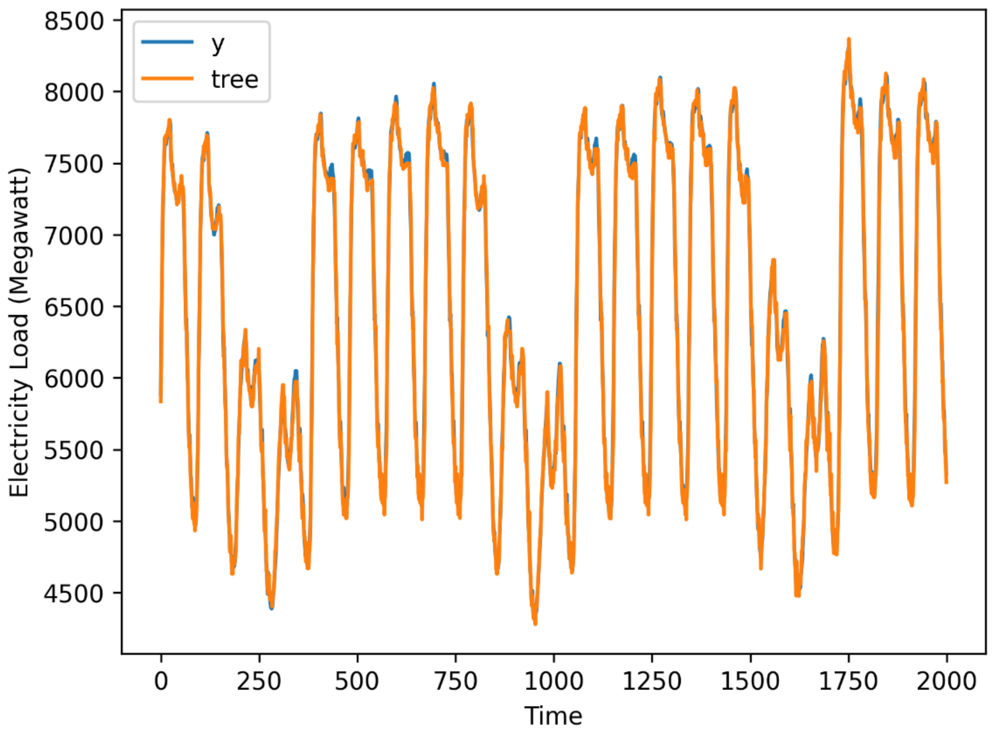


Figure 12 Best vs. Actual Results for Austria Load (15-minute intervals)

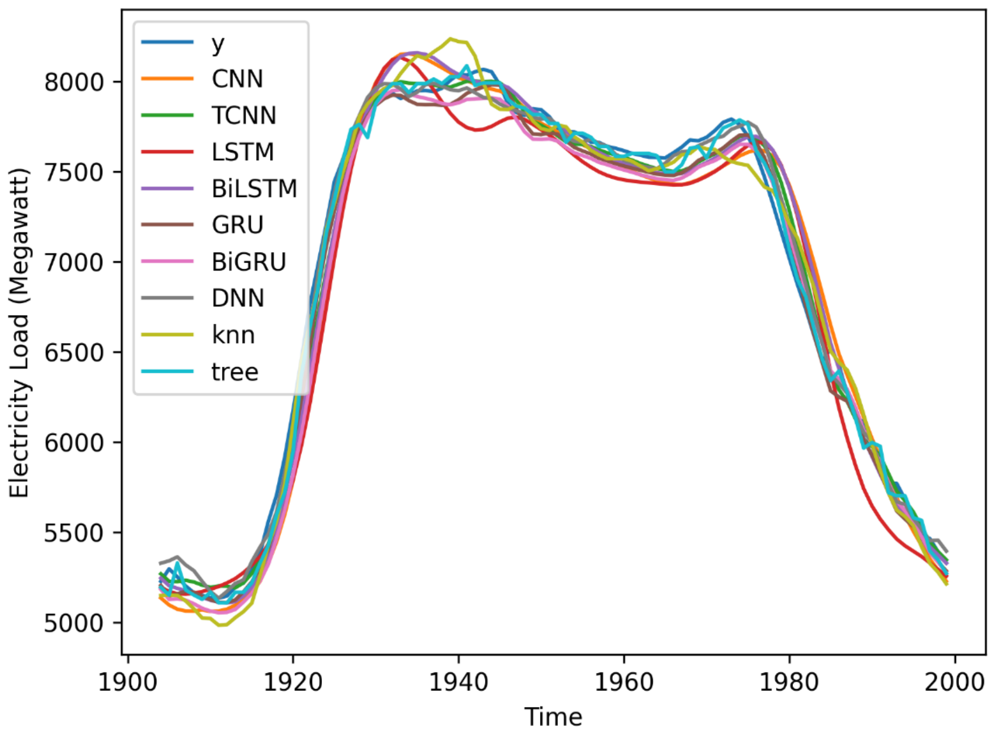


Figure 13 All Results Last 24-hour for Austria Load (15-minute intervals)

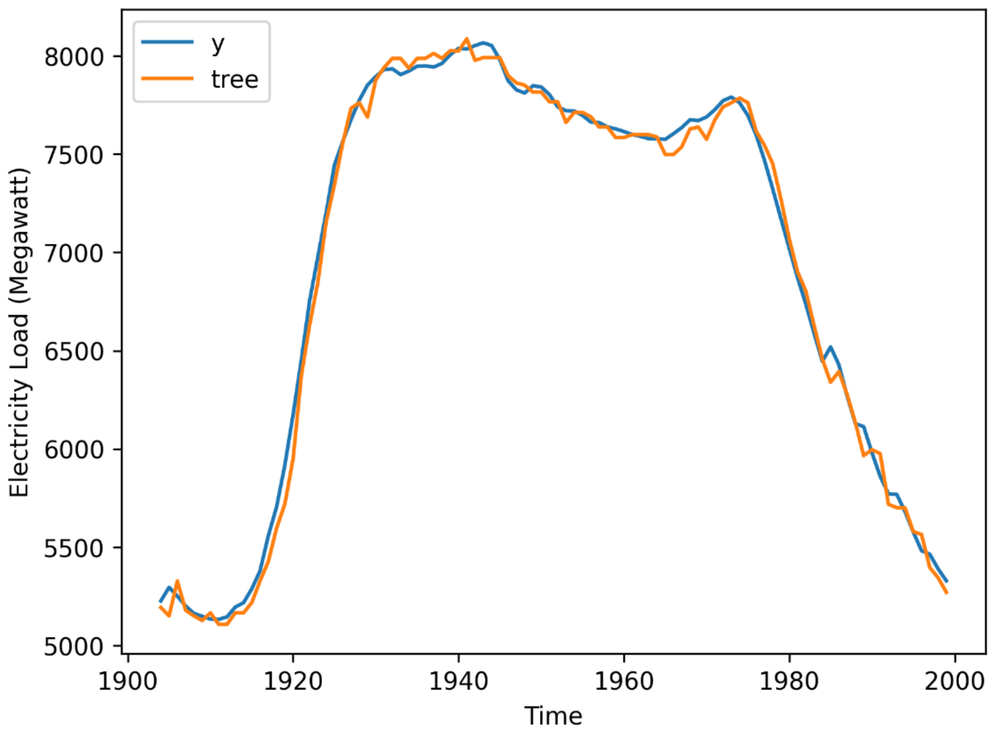


Figure 14 Best vs. Actual Results Last 24-hour for Austria Load (15-minute intervals)

Table 5 Evaluation Results for Germany Load (15-Minute Intervals)

Model	RMSE	MAE	MAD	MASE	Optimization Time Used (Minutes)
CNN	1531.60	1201.88	7952.66	2.26	5.23
TCNN	1301.13	995.12	8124.17	1.87	201.07
LSTM	1458.86	1108.28	8274.61	2.09	43.08
BiLSTM	1151.74	859.83	7948.68	1.62	82.22
GRU	710.93	526.08	8057.85	0.99	35.77
BiGRU	1169.70	970.33	8224.91	1.83	81.47
DNN	964.24	795.62	7974.43	1.50	3.69
kNN	1078.55	765.21	7974.71	1.44	0.34
Tree	476.06	351.20	8113.38	0.66	0.25

Table 5 presents the evaluation results of electrical loads for 15-minute intervals for Germany. The results also identify the Decision Tree as the most proficient model in terms of RMSE, MAE, MASE and optimization time. The GRU ranks second with RMSE of 710.93 and MAE of 526.08. However, its optimization time takes about 36 minutes which is much higher than the 0.25 minutes of the Tree. Thus, the Decision Tree is identified as the best model for this dataset. Figure 15 and Figure 17 illustrate that while all models generally capture the overarching trend of the data, some are not closely matching the curve of the true values. However, Figure 16 demonstrates that the predicted values from the Decision Tree align closely with the curve of the true values.

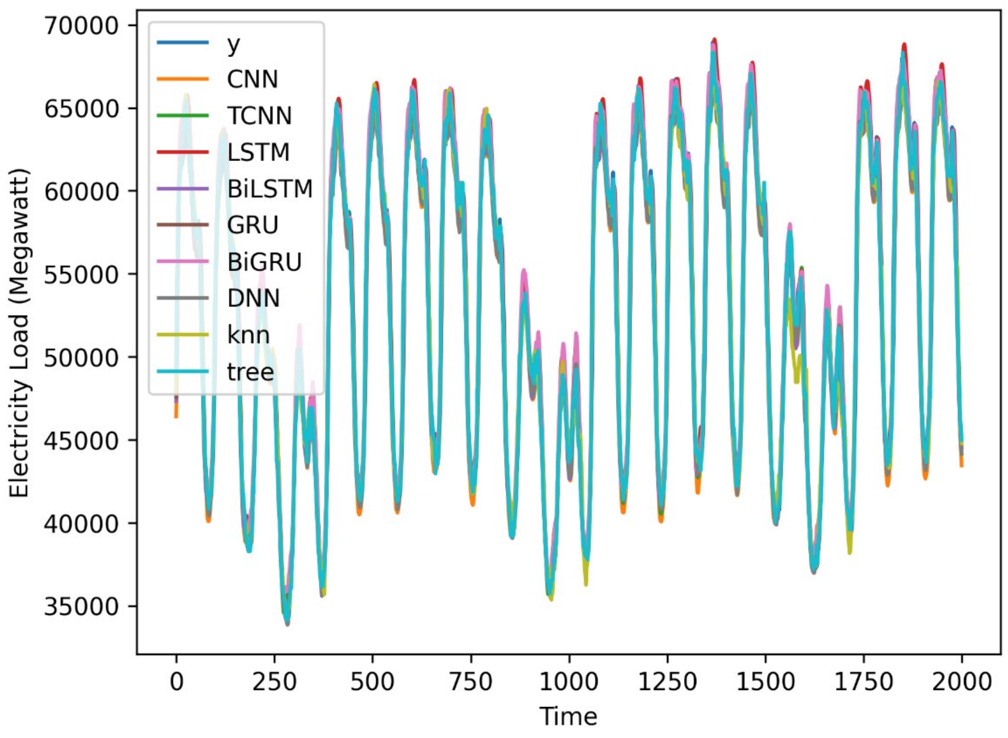


Figure 15 All Results for Germany Load (15-minute intervals)

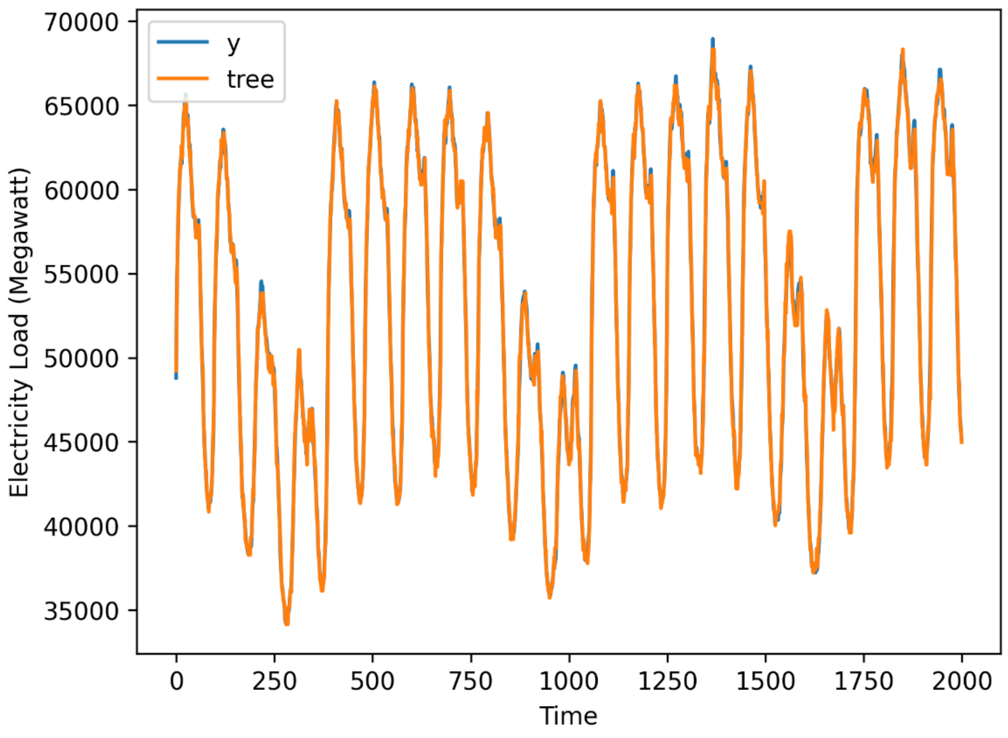


Figure 16 Best vs. Actual Results for Germany Load (15-minute intervals)

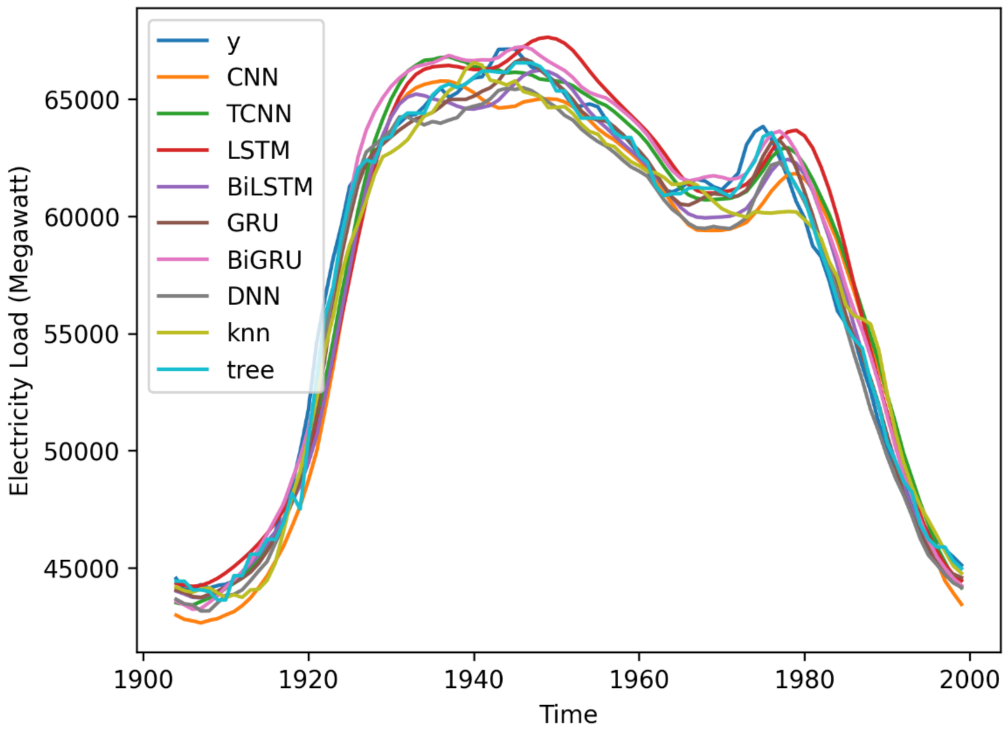


Figure 17 All Results Last 24-hour for Germany Load (15-minute intervals)

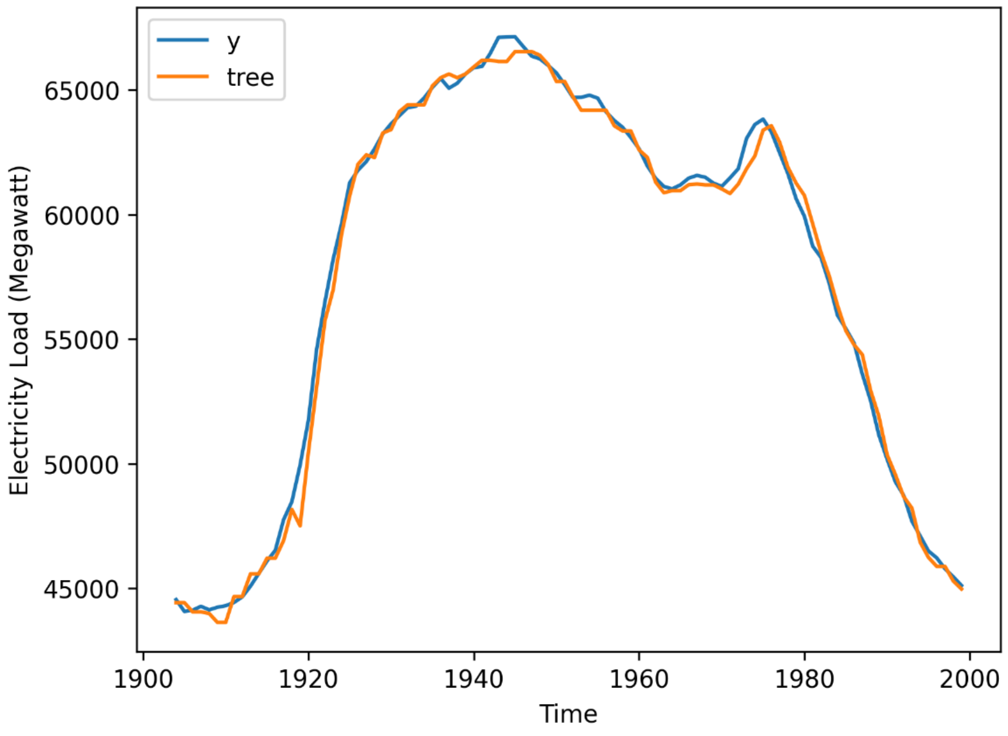


Figure 18 Best vs. Actual Results Last 24-hour for Germany Load (15-minute intervals)

Table 6 Evaluation Results for Netherlands Load (15-Minute Intervals)

Model	**RMSE**	**MAE**	**MAD**	**MASE**	**Optimization Time Used (Minutes)**
CNN	579.66	454.99	1279.64	3.47	6.24
TCNN	531.33	458.01	1353.38	3.50	199.55
LSTM	603.68	491.99	1343.66	3.76	48.10
BiLSTM	288.31	220.70	1337.44	1.69	96.50
GRU	387.51	302.98	1288.35	2.31	54.89
BiGRU	392.22	324.74	1309.96	2.48	92.64
DNN	320.27	265.89	1335.56	2.03	4.04
kNN	528.58	393.56	1221.29	3.01	0.57
Tree	166.91	107.21	1302.92	0.82	0.36

Table 6 presents the evaluation results of electrical loads for 15-minute intervals for Netherlands. Once again, the Decision Tree shows exceptional performance with the lowest RMSE, MAE and MASE. Moreover, it takes only 0.36 minutes to optimize. However, the MASE values for other models in this market exceed 1. Their performance does not surpass that of a simple naïve benchmark. In Figure 19 through Figure 22 below, the majority of models adhere to the general trend but demonstrate a limited fit to the curve of the true values. This discrepancy is partly due to the fluctuations within the data, which are discernible roughly in the plots. In particular, the error is most pronounced when encountering new maximum and minimum values. However, the predictive curves generated by some neural network models tend to more closely adhere to the true value curves, especially in these extreme cases.

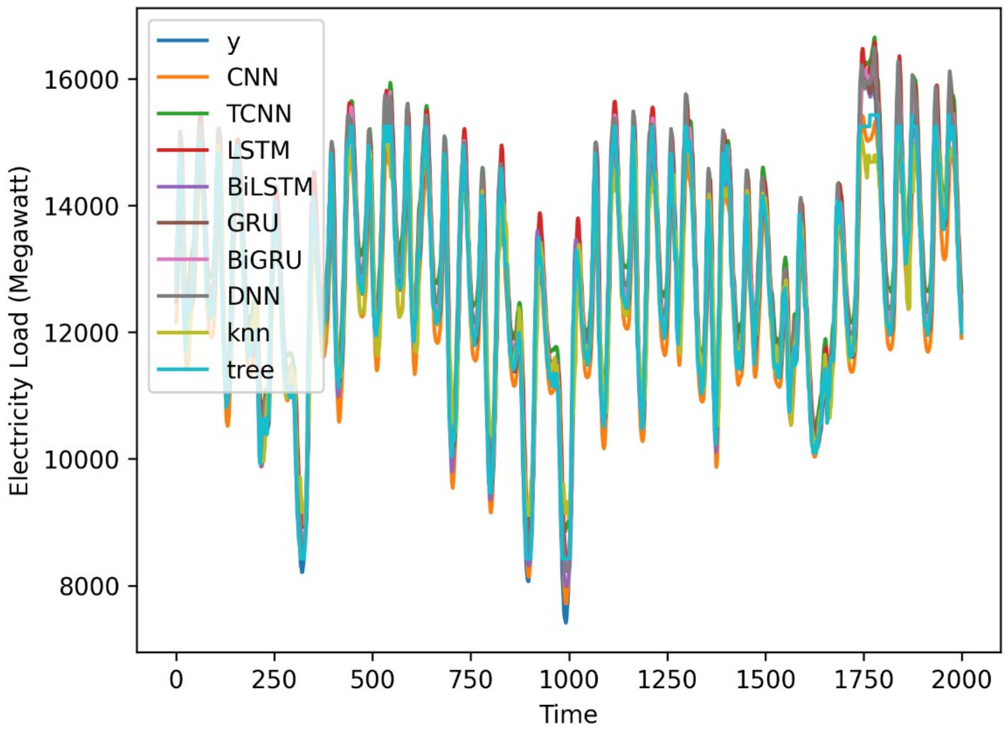


Figure 19 All Results for Netherlands Load (15-minute intervals)

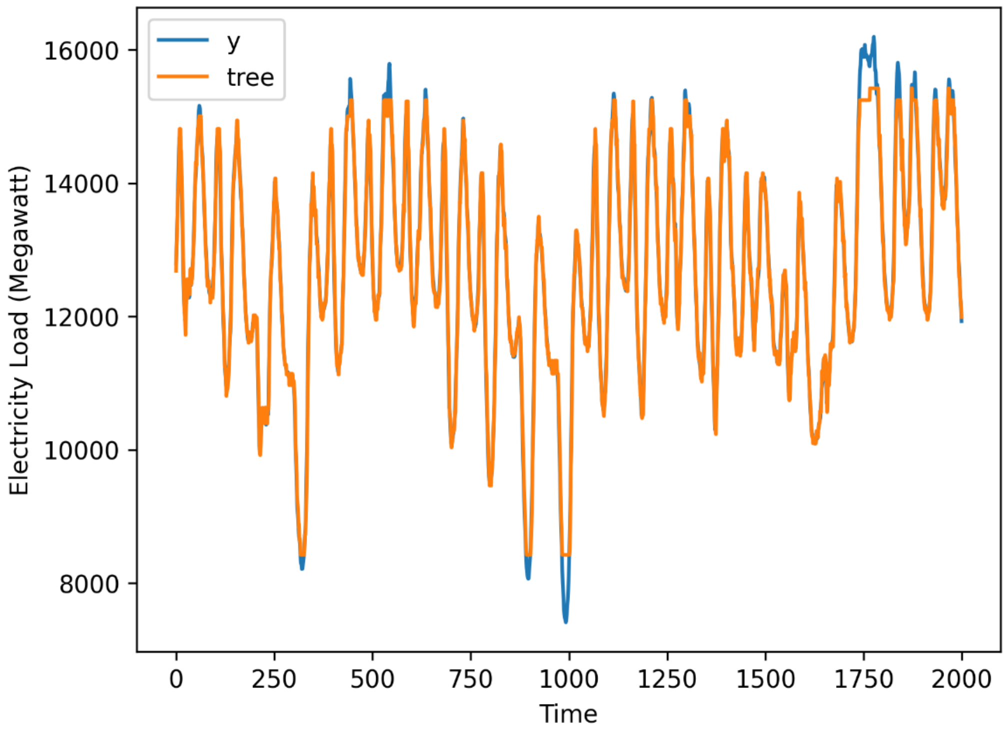


Figure 20 Best vs. Actual Results for Netherlands Load (15-minute intervals)

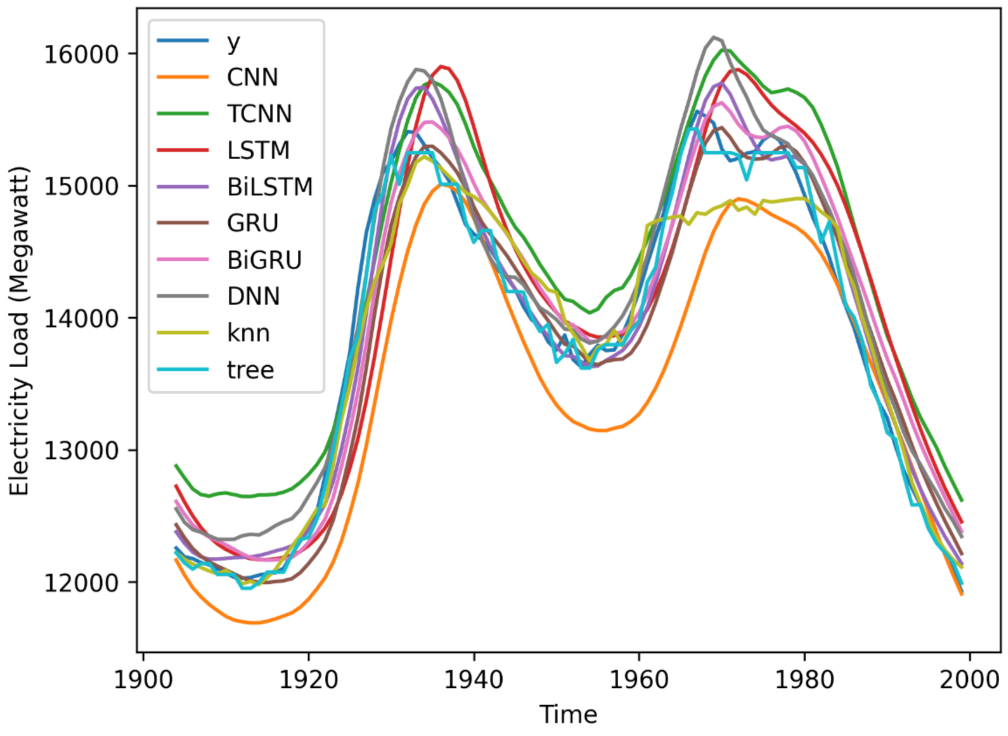


Figure 21 All Results Last 24-hour for Netherlands Load (15-minute intervals)

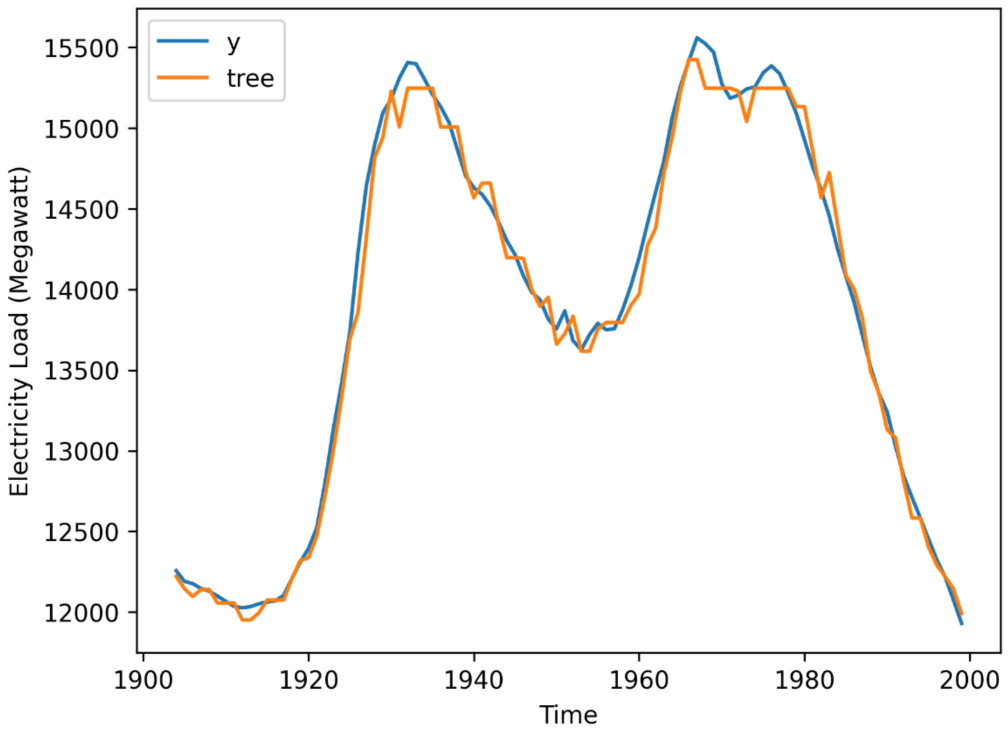


Figure 22 Best vs. Actual Results Last 24-hour for Netherlands Load (15-minute intervals)

Table 7 Evaluation Results for Austria Load (60-Minute Intervals)

Model	RMSE	MAE	MAD	MASE	Optimization Time Used (Minutes)
CNN	320.62	250.44	937.87	1.07	8.40
TCNN	461.24	437.99	1064.05	1.86	290.55
LSTM	602.14	513.50	760.96	2.19	60.64
BiLSTM	319.49	257.52	895.50	1.10	120.59
GRU	254.98	189.56	941.03	0.81	50.20
BiGRU	215.09	175.17	972.80	0.75	108.37
DNN	119.98	90.09	1004.36	0.38	4.60
kNN	160.70	124.65	1002.79	0.53	0.41
Tree	136.50	99.64	999.60	0.42	0.35

Table 7 presents the results of 60-minute intervals of electrical loads for Austria. For 60-minute interval in the same market, although the Decision Tree still performs strongly, DNN takes the first place with better RMSE, MAE and MASE. In terms of time used, the Decision Tree is still in first place, and DNN, although longer than the decision tree, is acceptable compared to the other neural network models. Figure 23 through Figure 26 present graphical comparisons between the actual values and the predicted values at 60-minute intervals. These plots are noticeably tighter than those depicted for the 15-minute intervals. It can be interpreted that the plots for the 15-minute intervals represent a higher sampled version of the latter portion of the 60-minute interval plots.

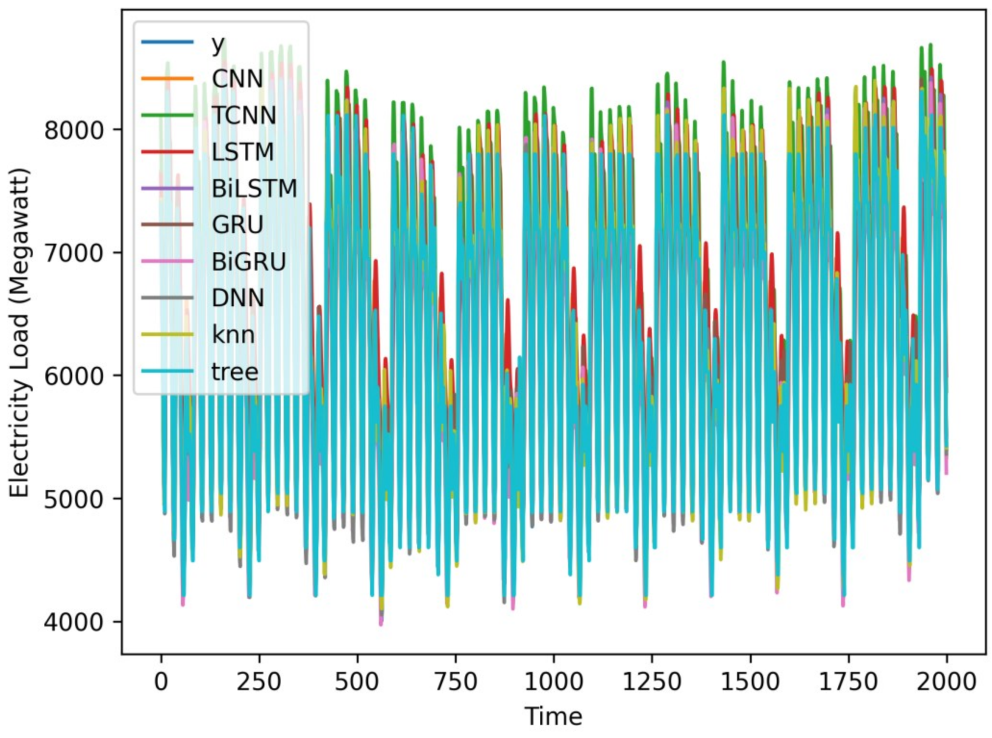


Figure 23 All Results plot Austria Load (60-minute intervals)

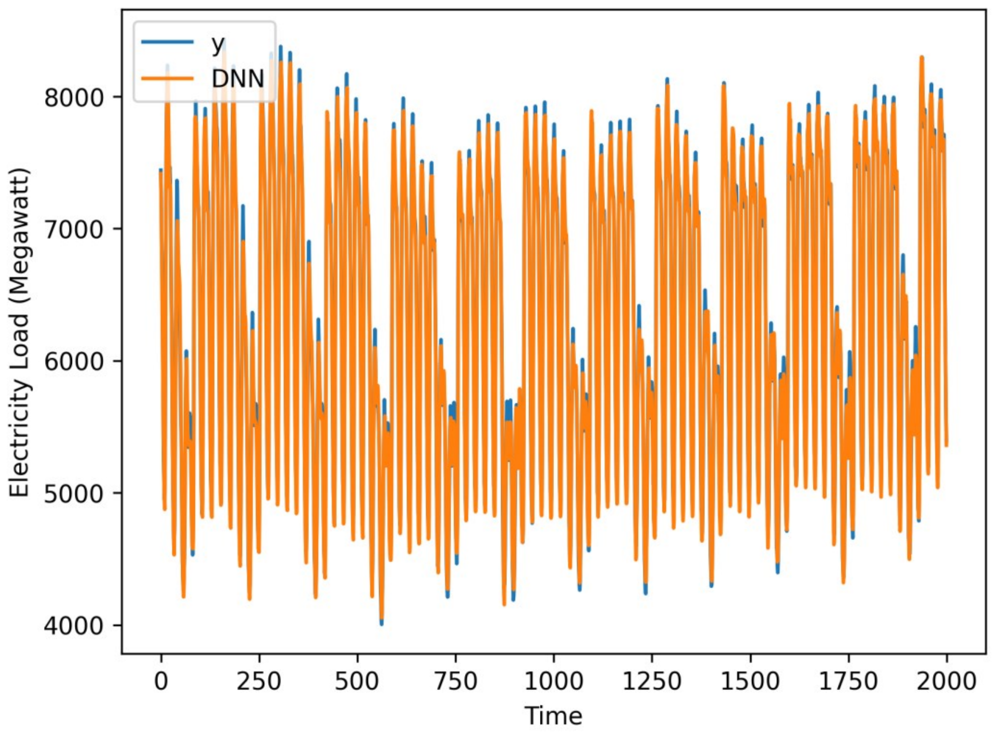


Figure 24 Best vs. Actual Results for Austria Load (60-minute intervals)

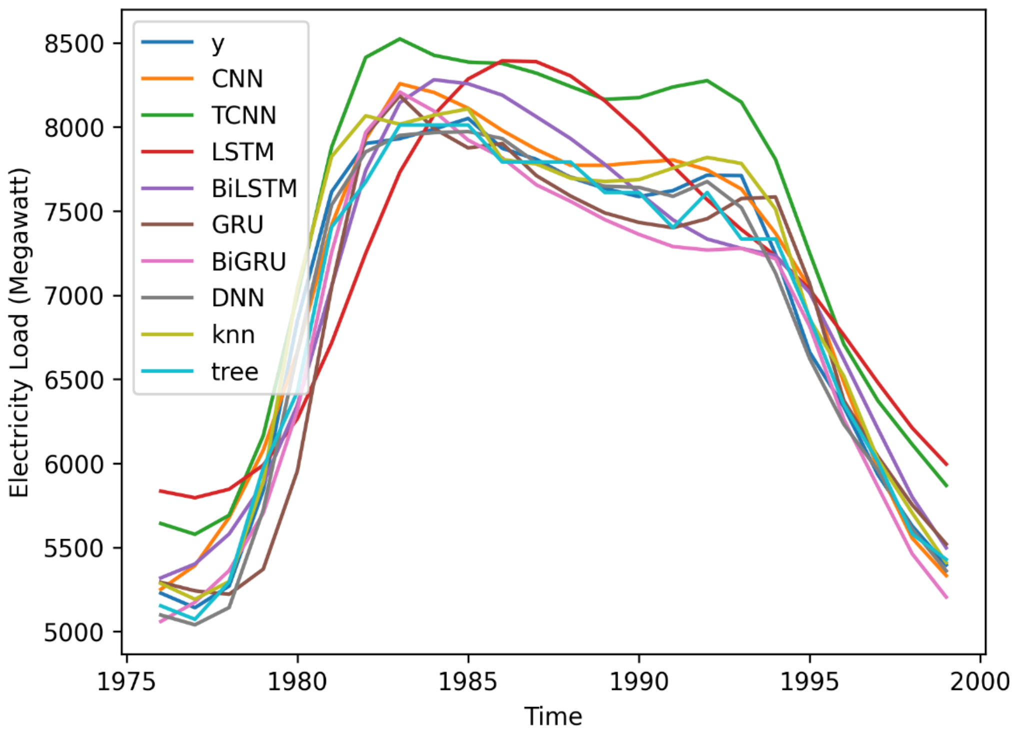


Figure 25 All Results Last 24-hour for Austria Load (60-minute intervals)

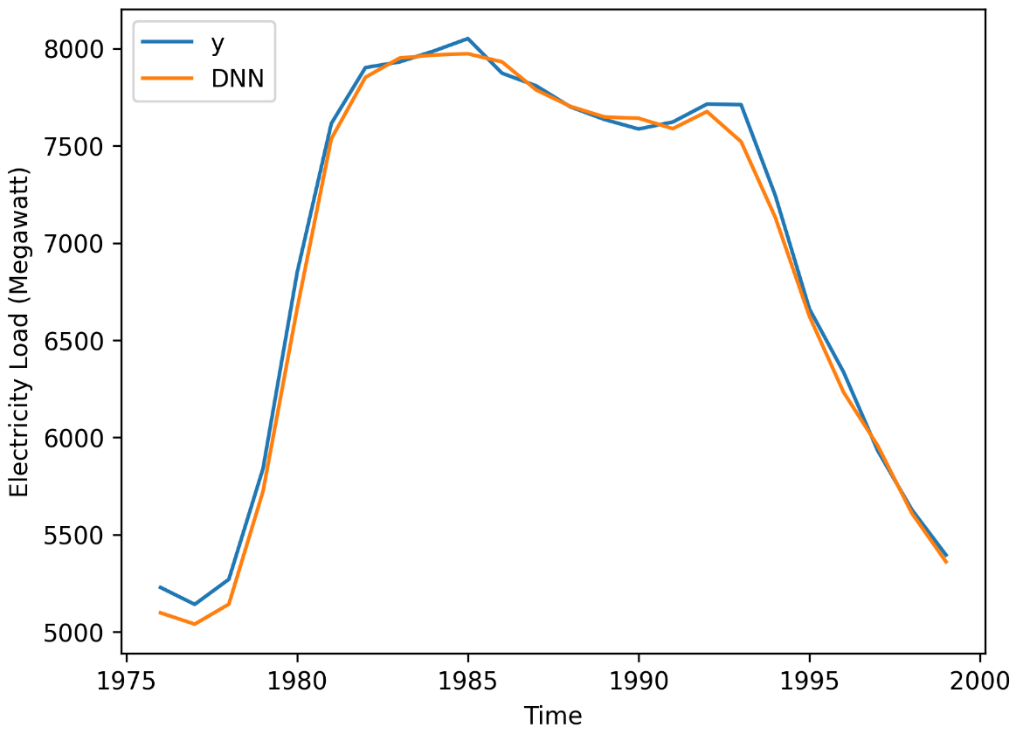


Figure 26 Best vs. Actual Results Last 24-hour for Austria Load (60-minute intervals)

Table 8 Evaluation Results for Germany Load (60-Minute Intervals)

Model	RMSE	MAE	MAD	MASE	Optimization Time Used (Minutes)
CNN	1903.04	1254.28	7585.96	0.69	5.21
TCNN	1083.49	784.54	7699.85	0.43	205.13
LSTM	2806.71	2061.80	7153.53	1.13	42.81
BiLSTM	1890.08	1429.04	7639.69	0.78	81.20
GRU	1506.70	1161.04	7580.95	0.64	35.93
BiGRU	1436.11	1090.38	7771.06	0.60	68.11
DNN	809.76	609.66	7905.96	0.33	3.29
kNN	983.67	749.60	7915.07	0.41	0.36
Tree	960.88	721.90	7928.43	0.40	0.21

Table 8 presents the results of 60-minute intervals of electrical loads for Germany. DNN exhibits outstanding performance with the lowest RMSE, MAE and MASE. And all models demonstrate improved performance to varying degrees, on the 60-minute dataset compared to 15-minute interval. The visualizations in Figure 27 to Figure 30 further confirm the consistency and orderliness of the observed data trends. Such findings are consistent with Germany's reputation as a highly developed and industrially advanced country.

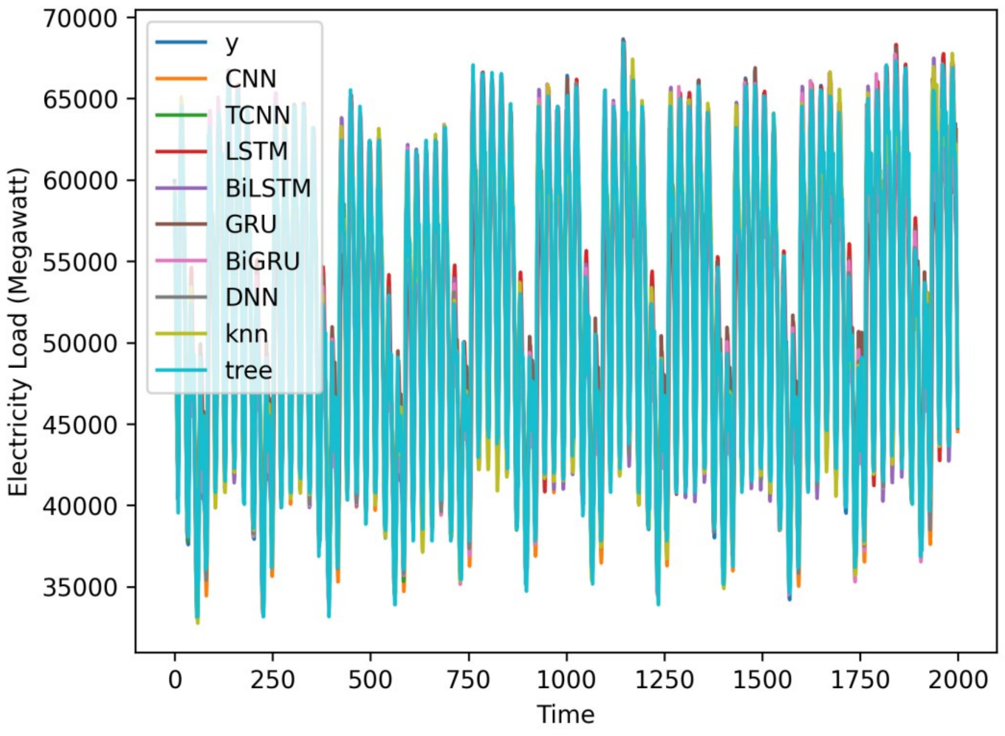


Figure 27 All Results for Germany Load (60-minute intervals)

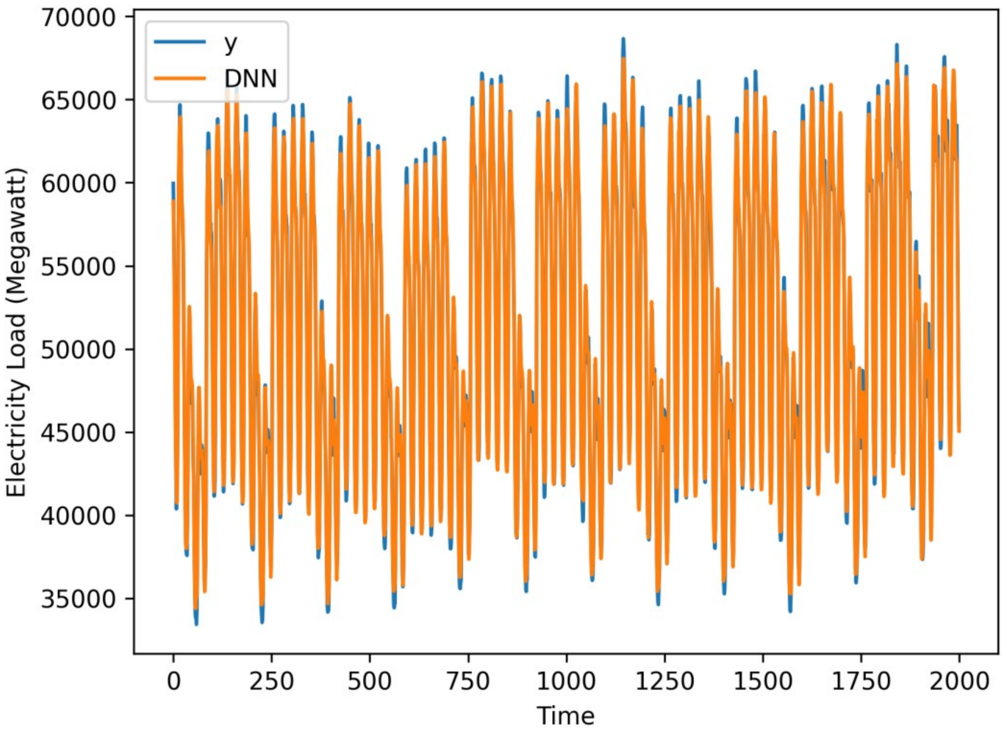


Figure 28 Best vs. Actual Results for Germany Load (60-minute intervals)

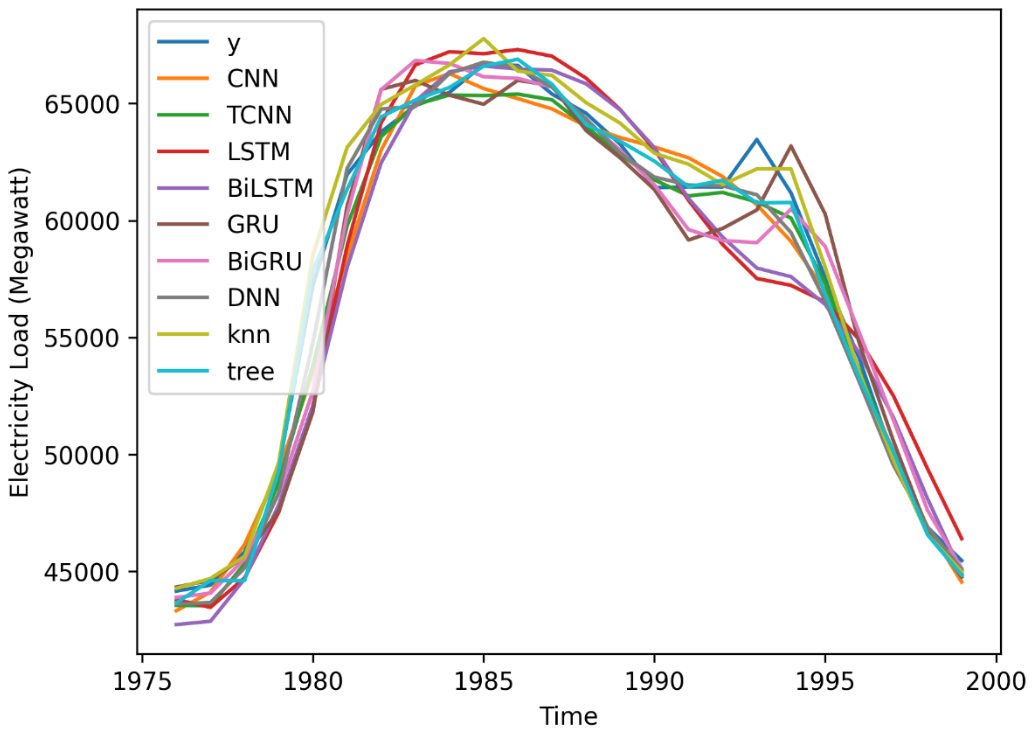


Figure 29 All Results Last 24-hour for Germany Load (60-minute intervals)

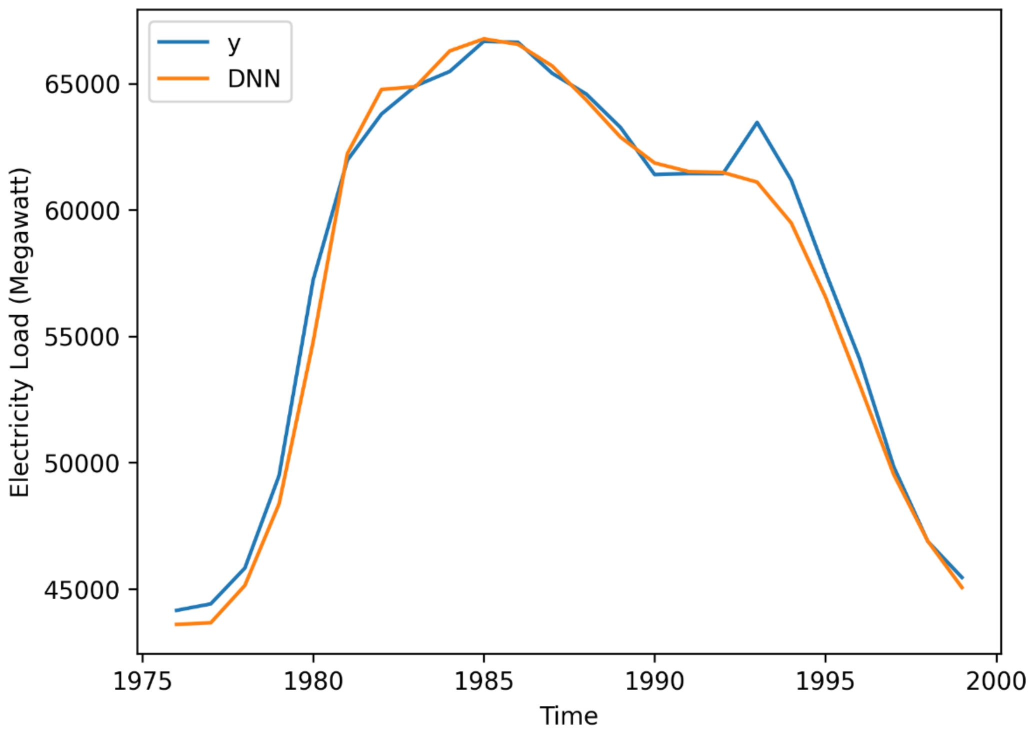


Figure 30 Best vs. Actual Results Last 24-hour for Germany Load (60-minute intervals)

Table 9 Evaluation Results for Netherlands Load (60-Minute Intervals)

Model	RMSE	MAE	MAD	MASE	Optimization Time Used (Minutes)
CNN	531.04	380.77	1103.86	1.05	6.51
TCNN	358.62	256.75	1168.39	0.71	209.14
LSTM	1161.89	930.44	693.73	2.58	53.13
BiLSTM	824.52	683.11	1015.22	1.89	112.32
GRU	657.00	553.35	1177.89	1.53	54.52
BiGRU	441.83	338.57	1206.93	0.94	95.14
DNN	335.28	246.16	1197.70	0.68	5.15
kNN	665.96	482.78	1194.83	1.34	0.52
Tree	375.28	269.35	1219.41	0.75	0.41

Table 9 presents the results of 60-minute intervals of electrical loads for Netherlands. DNN retains its preeminent position. And the Decision Tree performs robustly. Notably, TCNN stands out as a significant spot, securing the second place in terms of error metrics. However, the optimization time for the TCNN is a considerable drawback. TCNN takes approximately 209 minutes to optimize for this dataset alone. The complexity and irregularity in Netherlands dataset are further exemplified in the visualizations in Figure 31 to Figure 34. We are nearly unable to derive meaningful information from the comparison plots that encompass all 2000 predictions. Closer examination of the plots of the final 24-hour data reveals a pronounced discrepancy between the predicted values and the actual values curves, indicating a generally poor fit across most models.

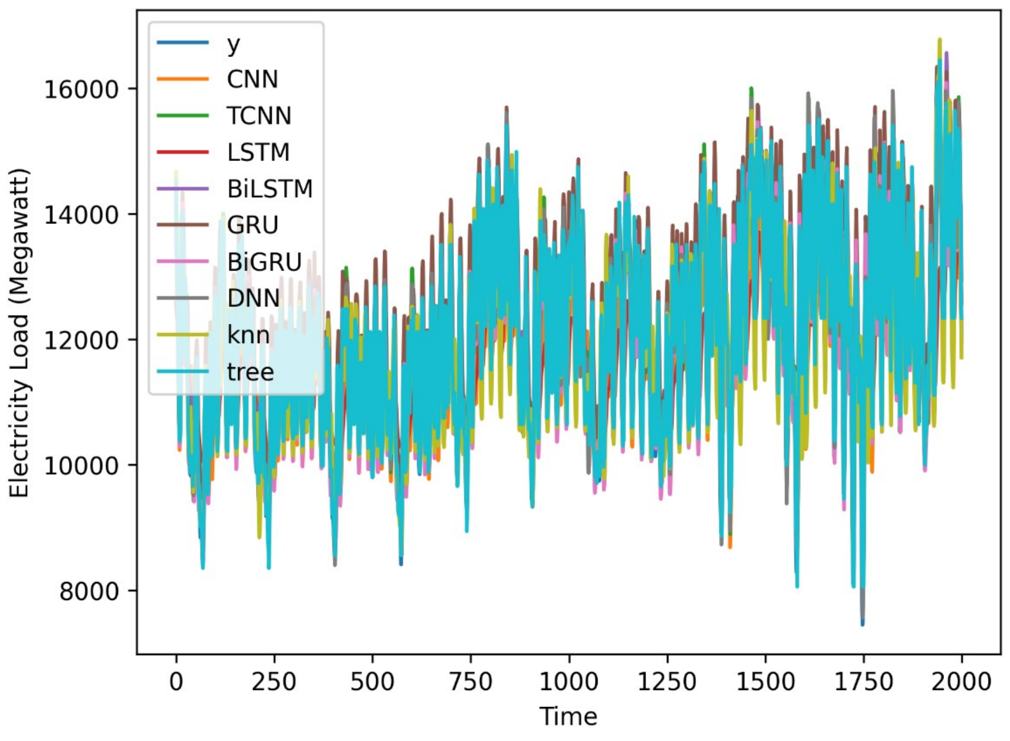


Figure 31 All Results for Netherlands Load (60-minute intervals)

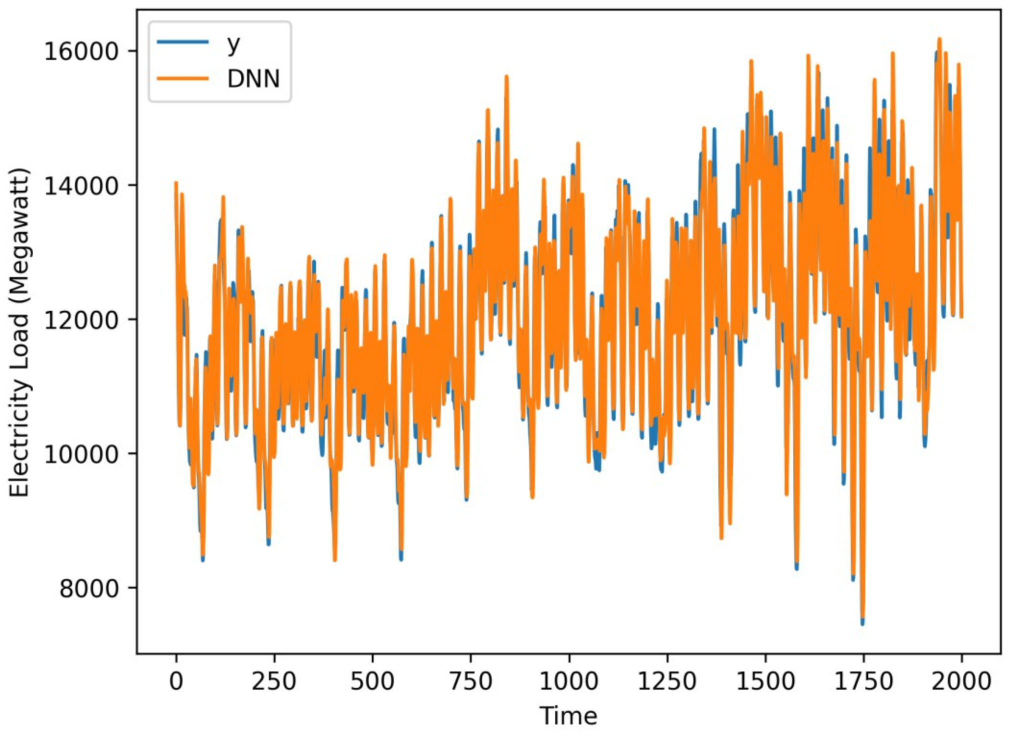


Figure 32 Best vs. Actual Results for Netherlands Load (60-minute intervals)

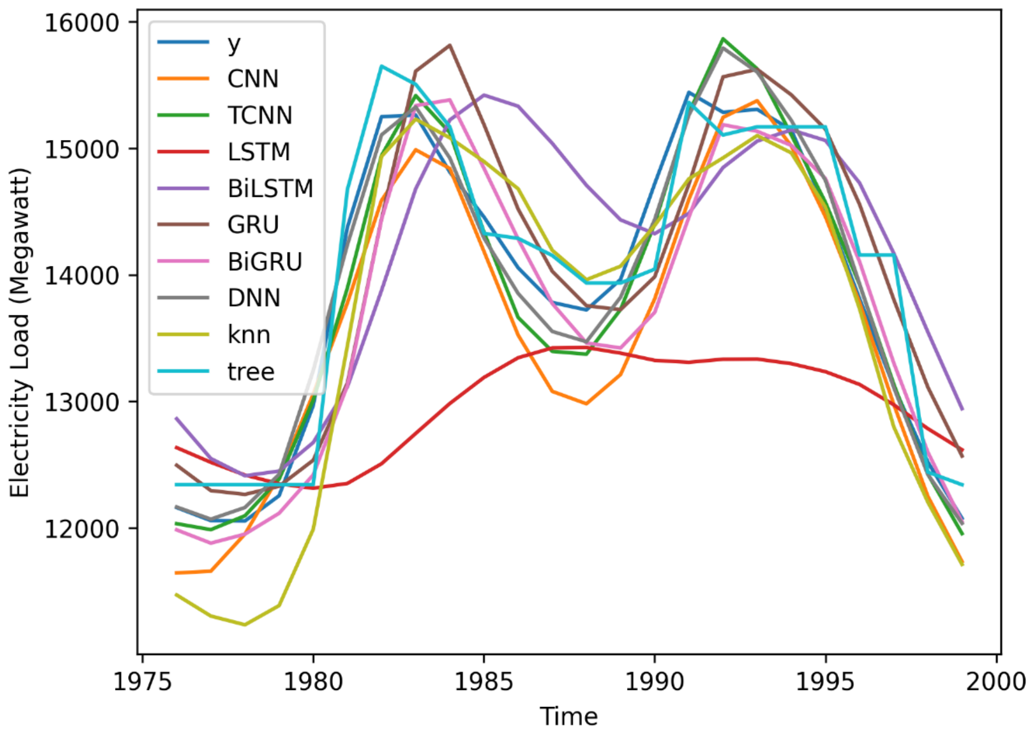


Figure 33 All Results Last 24-hour for Netherlands Load (60-minute intervals)

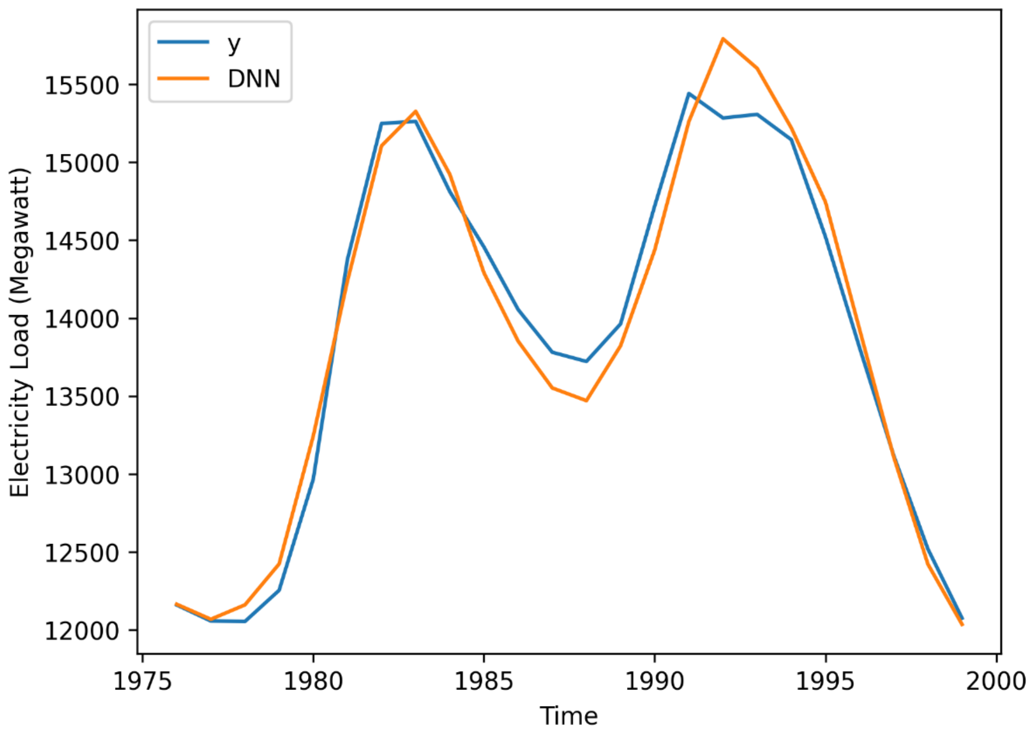


Figure 34 Best vs. Actual Results Last 24-hour for Netherlands Load (60-minute intervals)

Table 10 Evaluation Results for Austria Wind (15-Minute Intervals)

Model	RMSE	MAE	MAD	MASE	Optimization Time Used (Minutes)
CNN	86.71	55.61	543.39	1.83	6.34
TCNN	49.05	31.03	543.45	1.02	275.37
LSTM	137.67	96.55	530.15	3.17	43.83
BiLSTM	96.90	72.30	538.13	2.38	84.39
GRU	76.27	51.79	537.82	1.70	35.70
BiGRU	72.11	51.04	558.54	1.68	76.18
DNN	131.41	94.59	522.23	3.11	3.47
kNN	134.96	87.94	532.94	2.89	0.36
Tree	47.61	30.28	548.70	1.00	0.21

For wind generation, Table 10 presents the evaluation results of 15-minute intervals for Austria. Returning to the 15-minute intervals, the Decision Tree once again reclaims its position as the top-performing model. It is worth noting that the metrics of TCNN are slightly lower than that of Decision Tree. However, the time used for TCNN is still a significant drawback, approximately 1000 times that of Decision Tree. The volatility of wind generation is further exemplified in Figure 35 through Figure 38. The increased frequency of data points from 15-minute intervals presents significant challenges for model predictions.

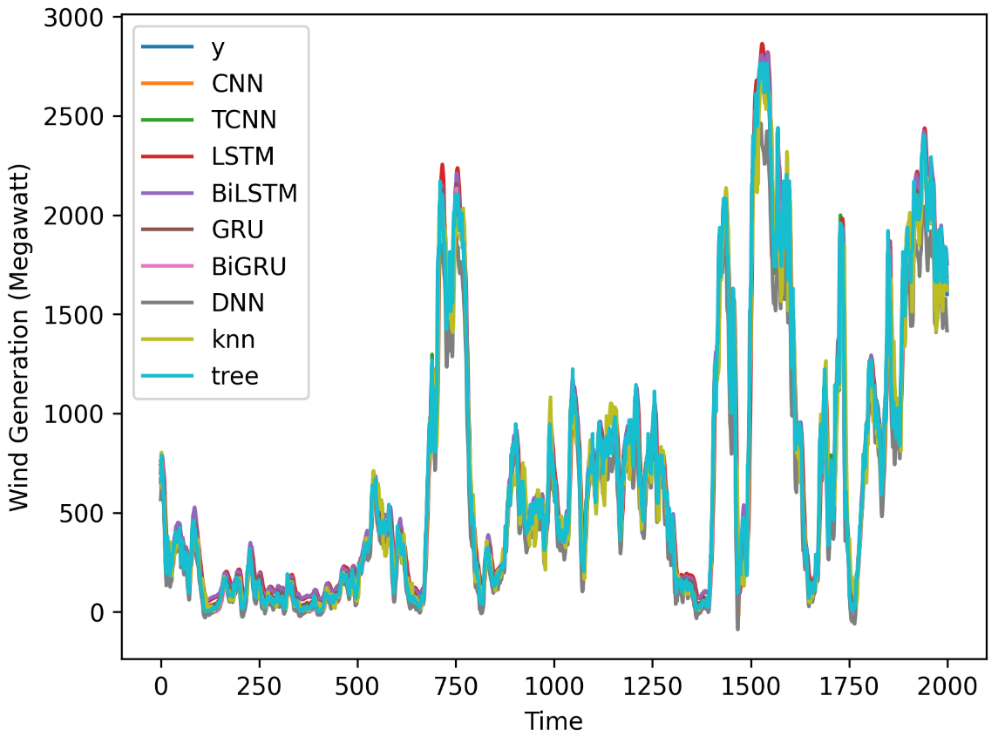


Figure 35 All Results for Austria Wind (15-minute intervals)

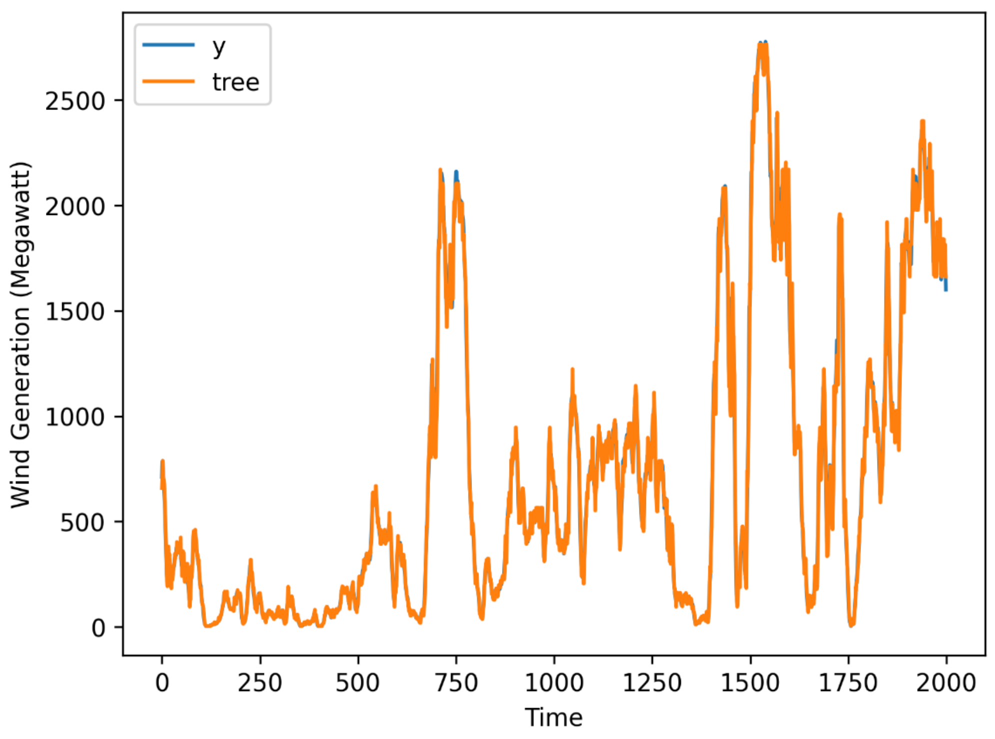


Figure 36 Best vs. Actual Results for Austria Wind (15-minute intervals)

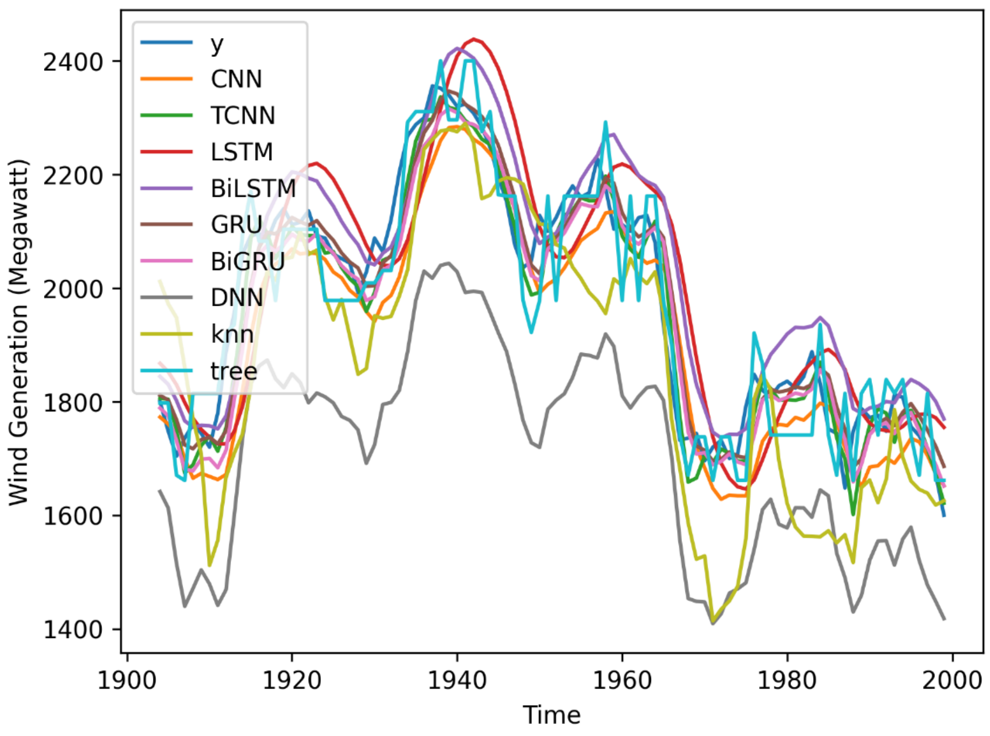


Figure 37 All Results Last 24-hour for Austria Wind (15-minute intervals)

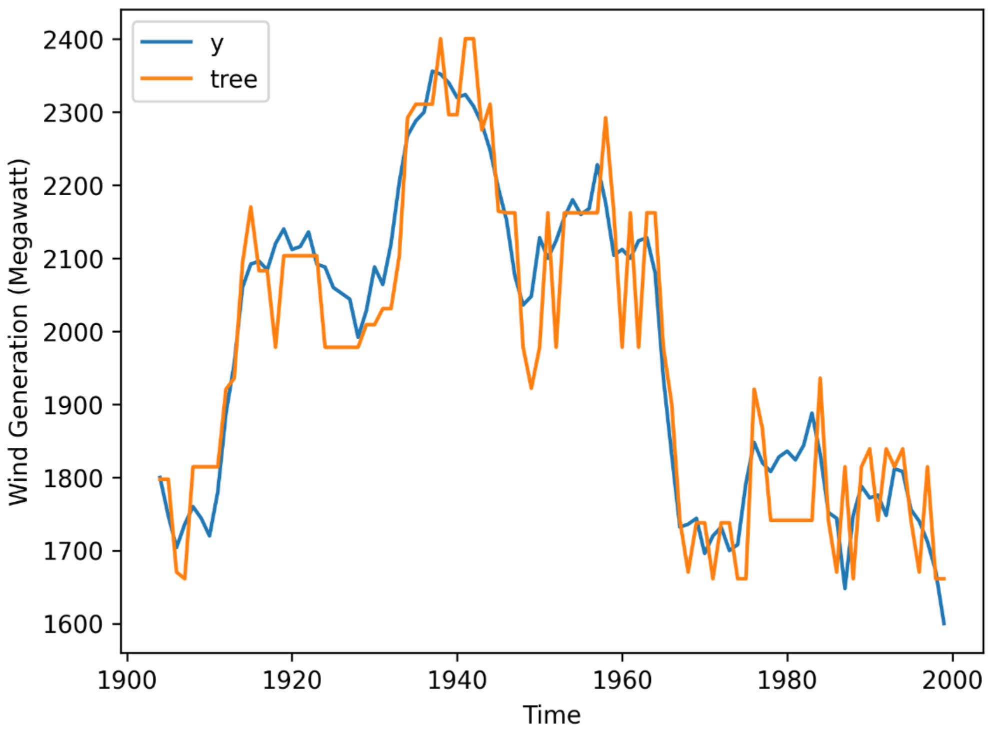


Figure 38 Best vs. Actual Results Last 24-hour for Austria Wind (15-minute intervals)

Table 11 Evaluation Results for Germany Wind (15-Minute Intervals)

Model	RMSE	MAE	MAD	MASE	Optimization Time Used (Minutes)
CNN	1089.05	799.40	5043.56	3.75	4.21
TCNN	582.33	414.58	5352.63	1.94	196.70
LSTM	997.84	704.16	5075.81	3.30	41.34
BiLSTM	869.45	666.12	5286.35	3.12	79.75
GRU	687.55	474.10	5100.22	2.22	35.19
BiGRU	686.10	536.30	5208.85	2.51	77.77
DNN	450.76	321.39	5200.90	1.51	3.17
kNN	889.23	638.81	5103.64	2.99	0.37
Tree	253.29	178.07	5223.61	0.83	0.22

Table 11 presents the evaluation results of 15-minute intervals for Germany. As shown in the time series plots presented in preliminary analyses, the wind market exhibits significant randomness, which impacts the performance of forecasting models. Only Decision Tree stands out with MASE less than 1. Although the metrics are not initially promising, the models generally conform to the actual trend as depicted from Figure 39 to Figure 42. In Figure 42, the prediction curve generated by the Decision Tree demonstrates a closer alignment with the true value curve.

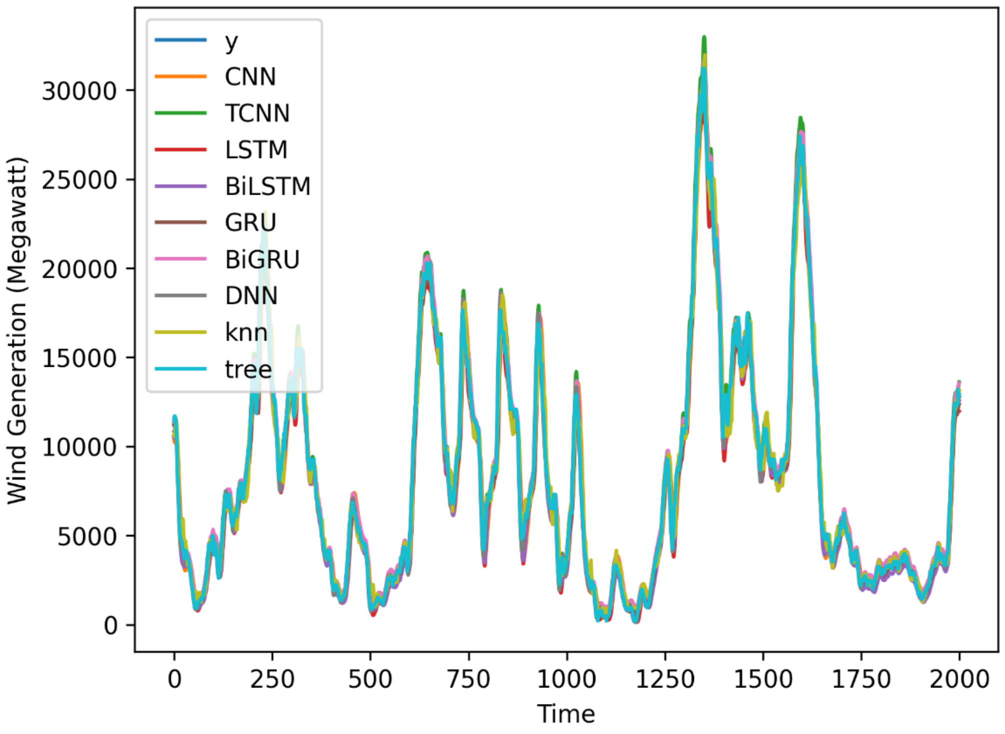


Figure 39 All Results for Germany Wind (15-minute intervals)

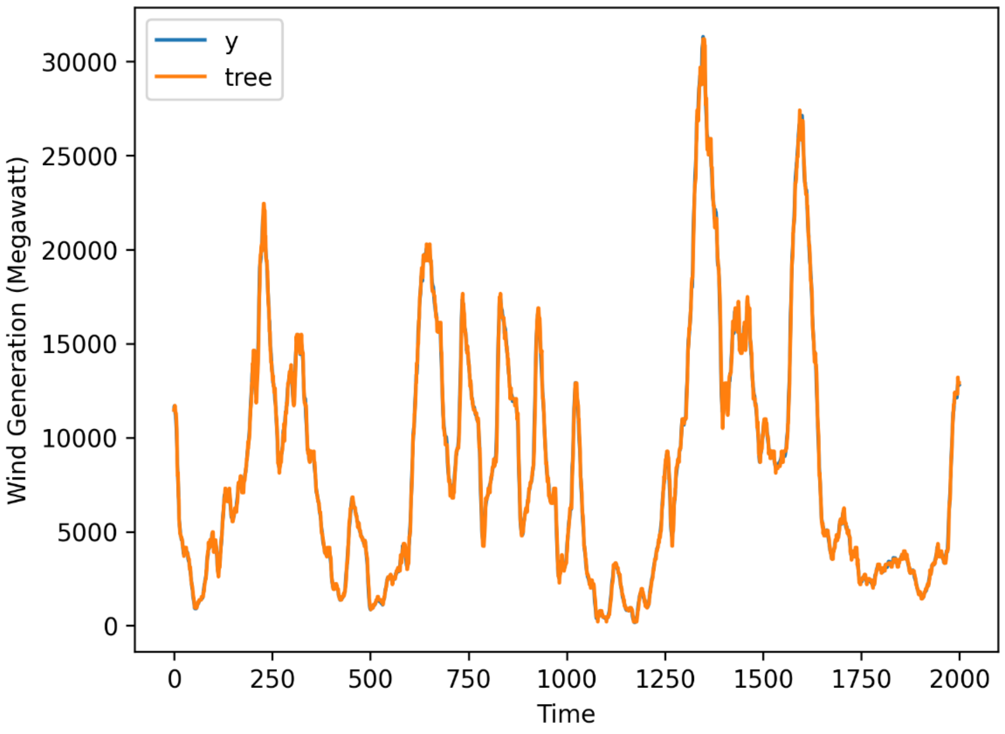


Figure 40 Best vs. Actual Results for Germany Wind (15-minute intervals)

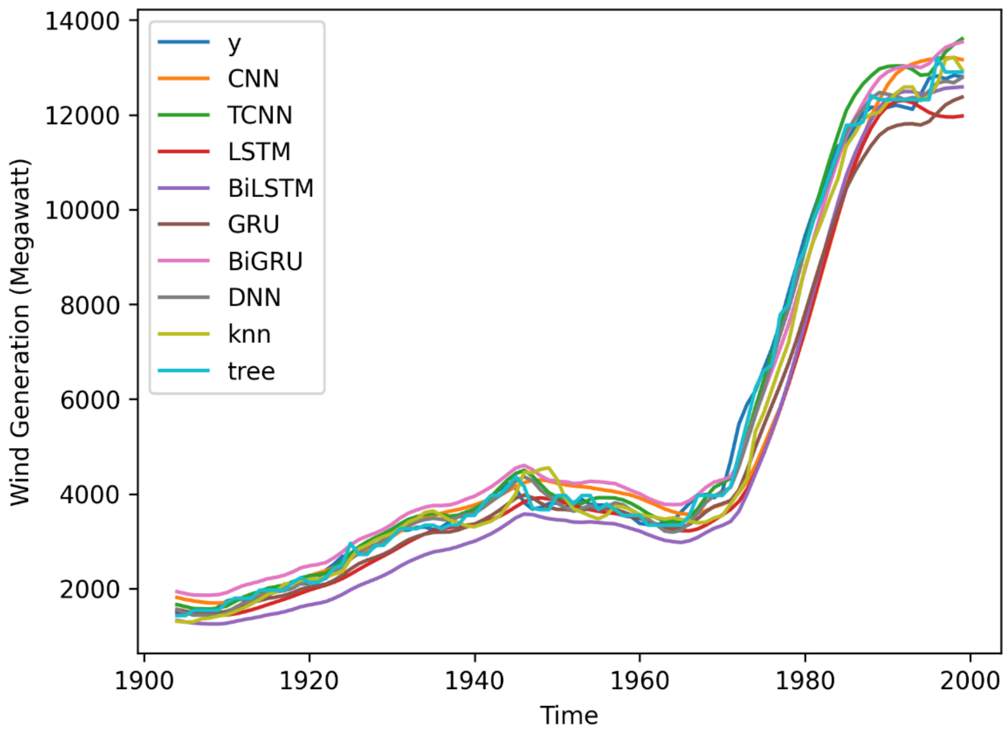


Figure 41 All Results Last 24-hour for Germany Wind (15-minute intervals)

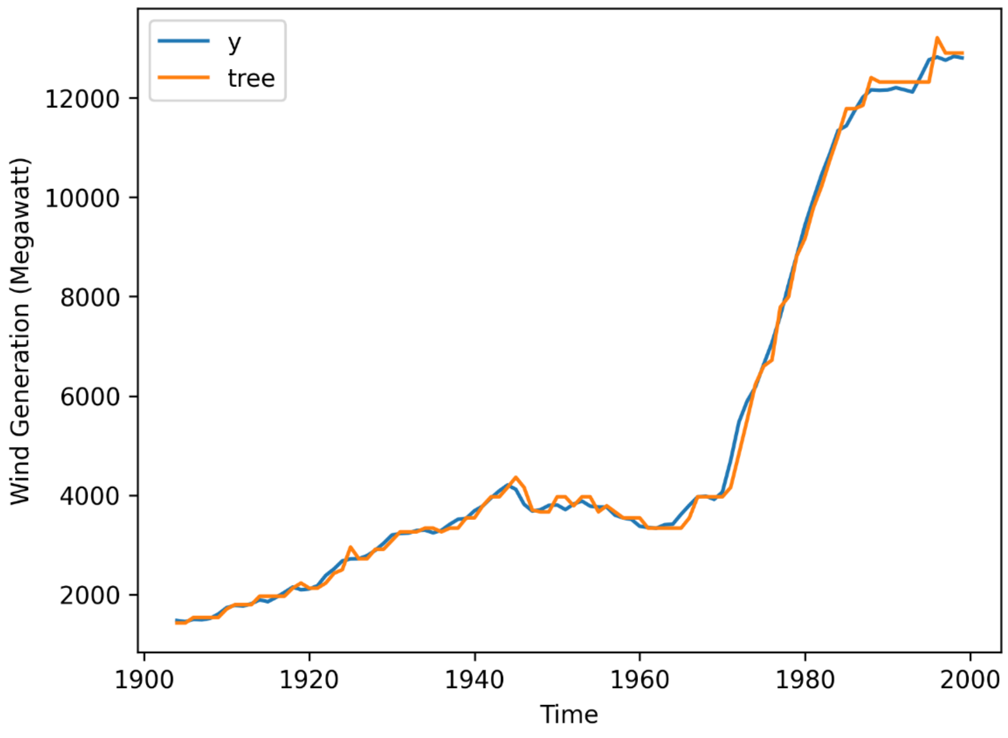


Figure 42 Best vs. Actual Results Last 24-hour for Germany Wind (15-minute intervals)

Table 12 Evaluation Results for Netherlands Wind (15-Minute Intervals)

Model	**RMSE**	**MAE**	**MAD**	**MASE**	**Optimization Time Used (Minutes)**
CNN	81.34	58.35	355.78	2.64	4.26
TCNN	45.00	32.83	343.11	1.49	199.61
LSTM	69.86	49.43	342.20	2.24	38.54
BiLSTM	64.25	46.10	342.83	2.09	73.46
GRU	51.78	37.13	359.19	1.68	31.66
BiGRU	53.72	37.32	357.55	1.69	69.67
DNN	45.17	31.63	343.48	1.43	4.82
kNN	74.19	51.00	347.47	2.31	0.36
Tree	37.06	24.90	357.32	1.13	0.28

Table 12 presents the evaluation results of 15-minute intervals for Netherlands. In this dataset, none of the models can make satisfactory predictions. Among the models, the Decision Tree, typically the better model, exhibits only modest result. In Figure 43 and Figure 44, the plotted data points appear to closely align with the actual value curve, suggesting a relatively accurate model performance at first glance. However, a more detailed examination of Figure 45 and Figure 46 reveals a different scenario. These figures show that the data points show a jagged pattern and deviate at a noticeable distance from the true value curve, indicating that the model's predictions are not as closely aligned with the actual outcomes as initially perceived.

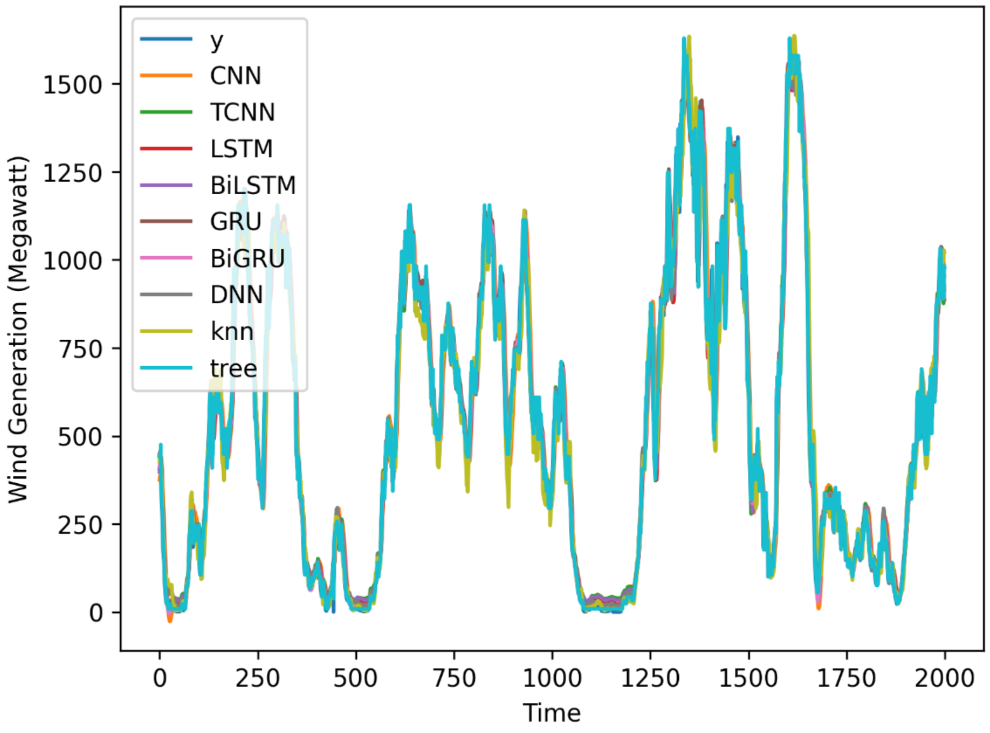


Figure 43 All Results for Netherlands Wind (15-minute intervals)

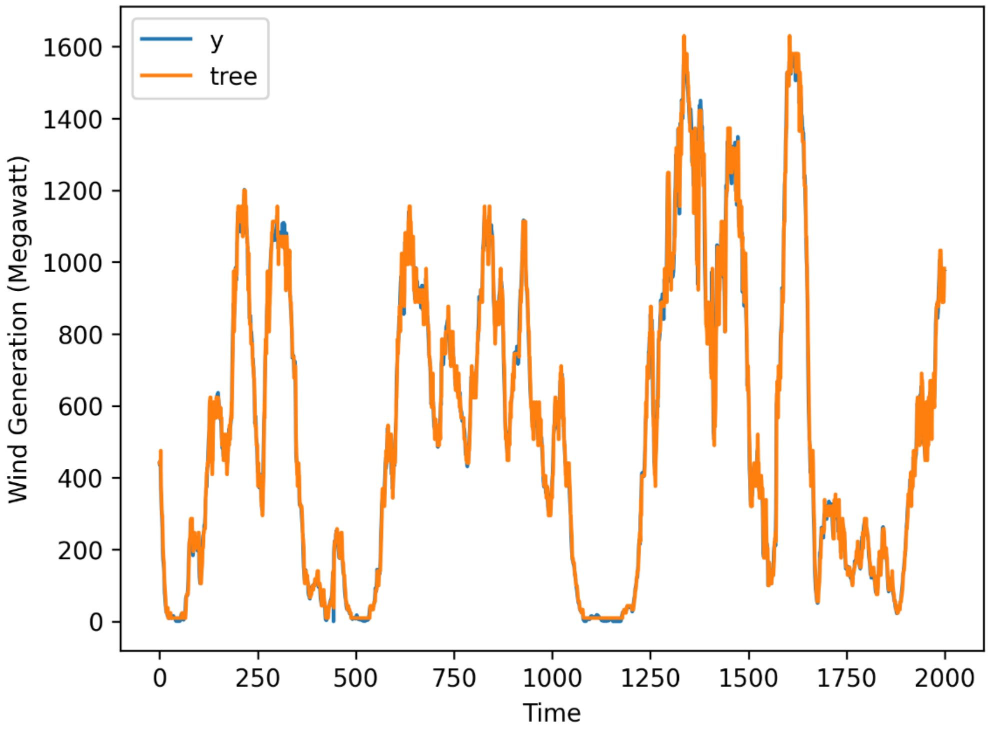


Figure 44 Best vs. Actual Results for Netherlands Wind (15-minute intervals)

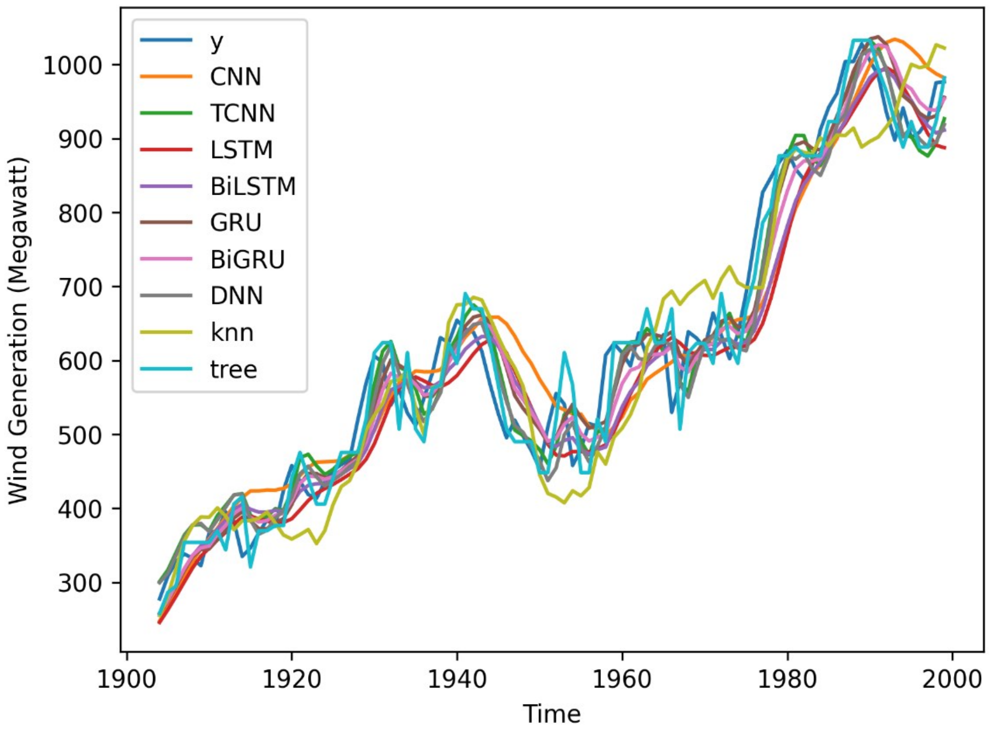


Figure 45 All Results Last 24-hour for Netherlands Wind (15-minute intervals)

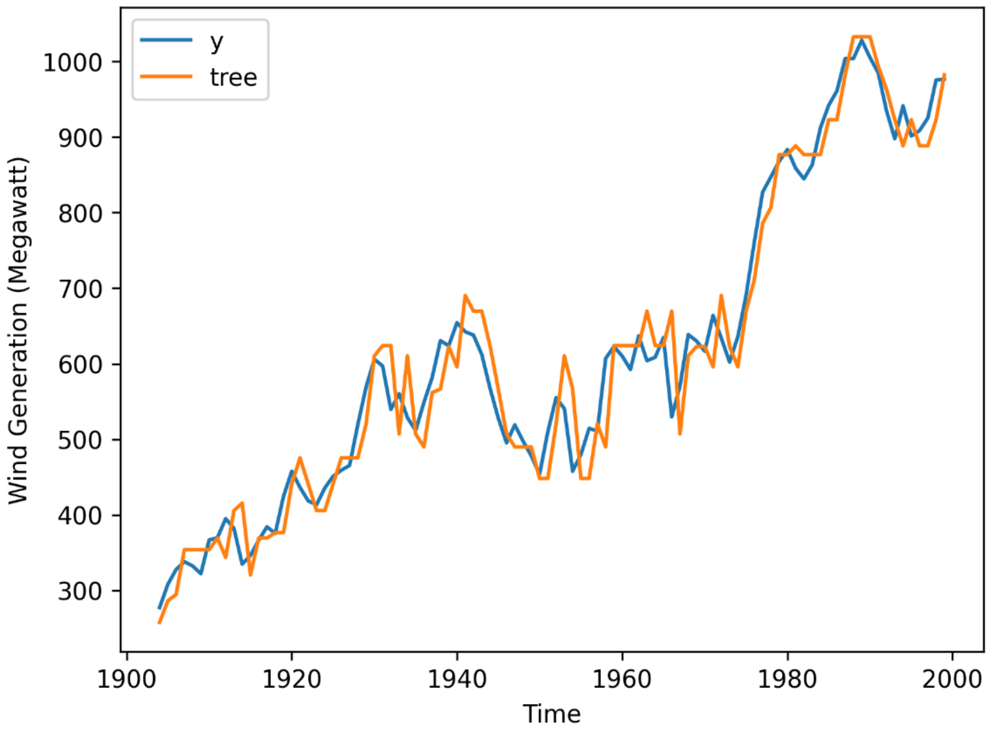


Figure 46 Best vs. Actual Results Last 24-hour for Netherlands Wind (15-minute intervals)

Table 13 Evaluation Results for Austria Wind (60-Minute Intervals)

Model	**RMSE**	**MAE**	**MAD**	**MASE**	**Optimization Time Used (Minutes)**
CNN	204.43	139.14	445.32	1.46	4.90
TCNN	153.52	99.65	447.48	1.05	289.37
LSTM	236.93	182.37	420.69	1.92	38.86
BiLSTM	234.10	178.87	478.34	1.88	90.91
GRU	169.11	112.71	470.58	1.19	48.15
BiGRU	156.80	98.55	472.54	1.04	94.36
DNN	166.92	110.01	459.52	1.16	3.22
kNN	254.46	176.06	426.64	1.85	0.38
Tree	155.56	100.03	470.63	1.05	0.25

The next three tables show the wind market at 60-minute intervals. Table 13 presents the results of Austria. Among the models, TCNN, Decision Tree and BiGRU are relatively superior. However, while these models outperform their counterparts, they all have a MASE greater than 1. In Figure 47 through Figure 50, the plots have a pronounced level of clutter. The visual congestion in these figures makes it difficult to discern whether the models align closely with the actual data trends.

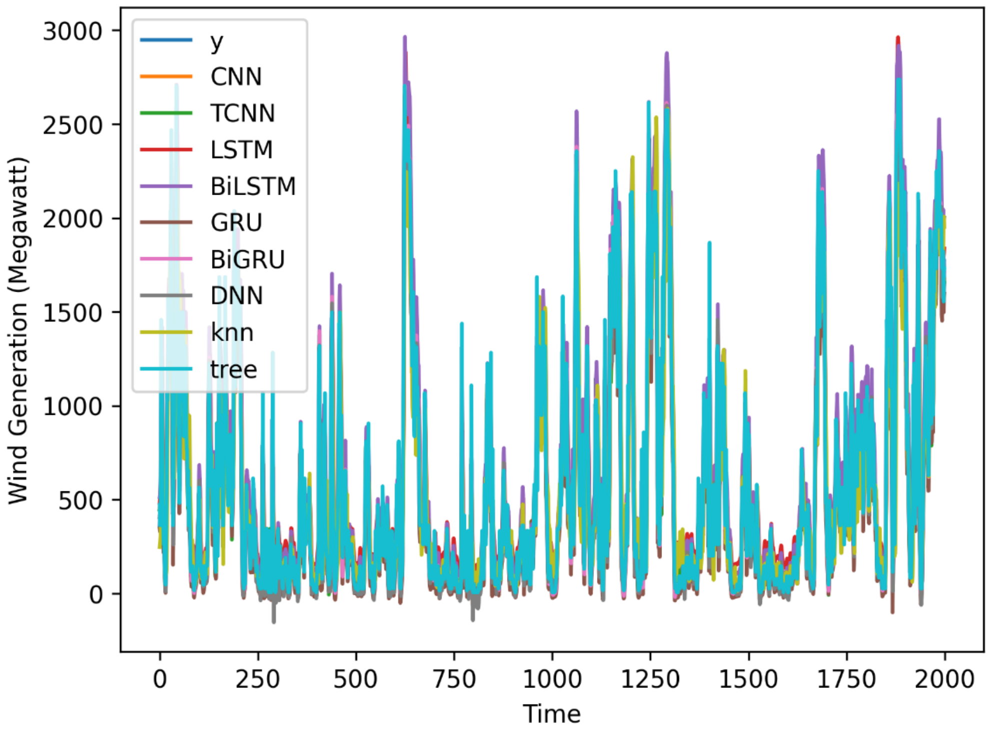


Figure 47 All Results for Austria Wind (60-minute intervals)

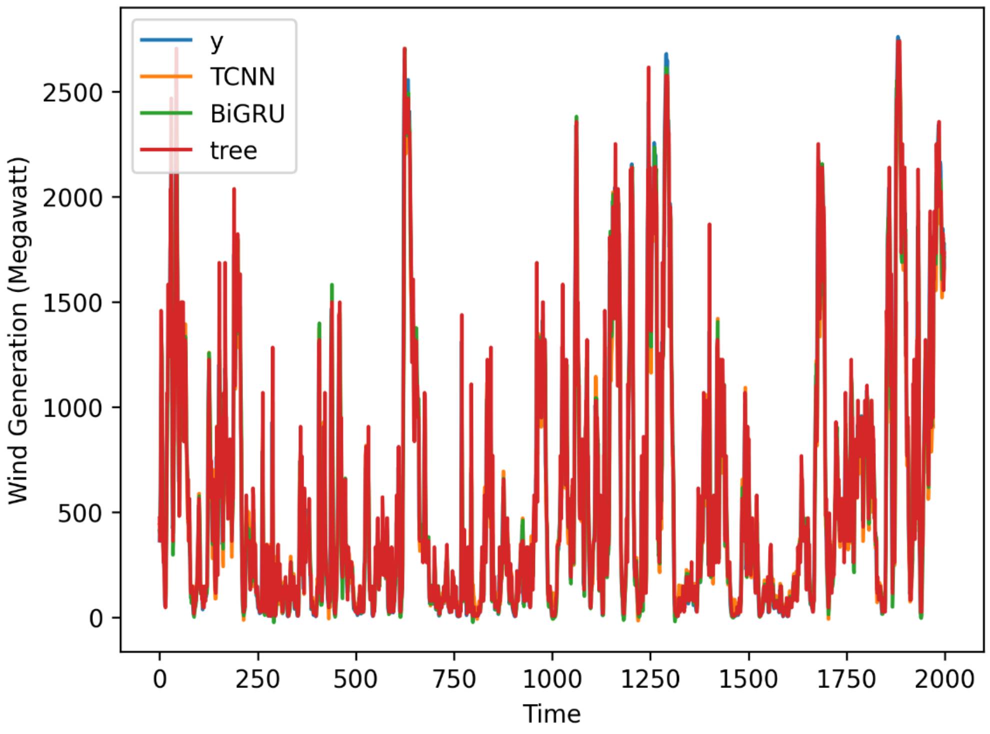


Figure 48 Best vs. Actual Results for Austria Wind (60-minute intervals)

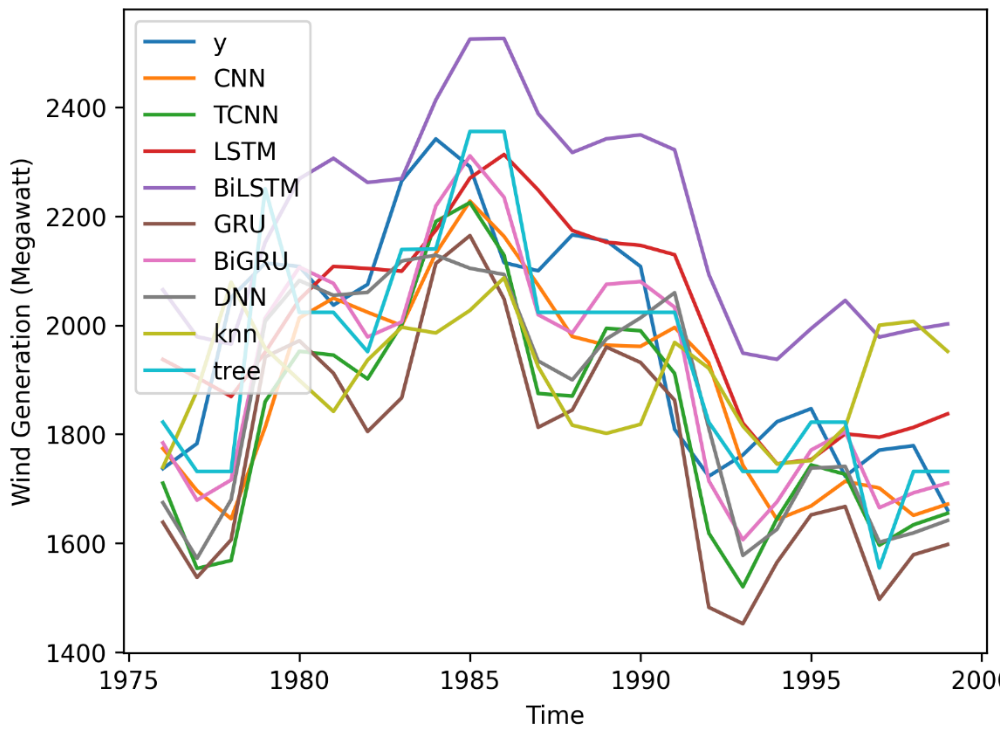


Figure 49 All Results Last 24-hour for Austria Wind (60-minute intervals)

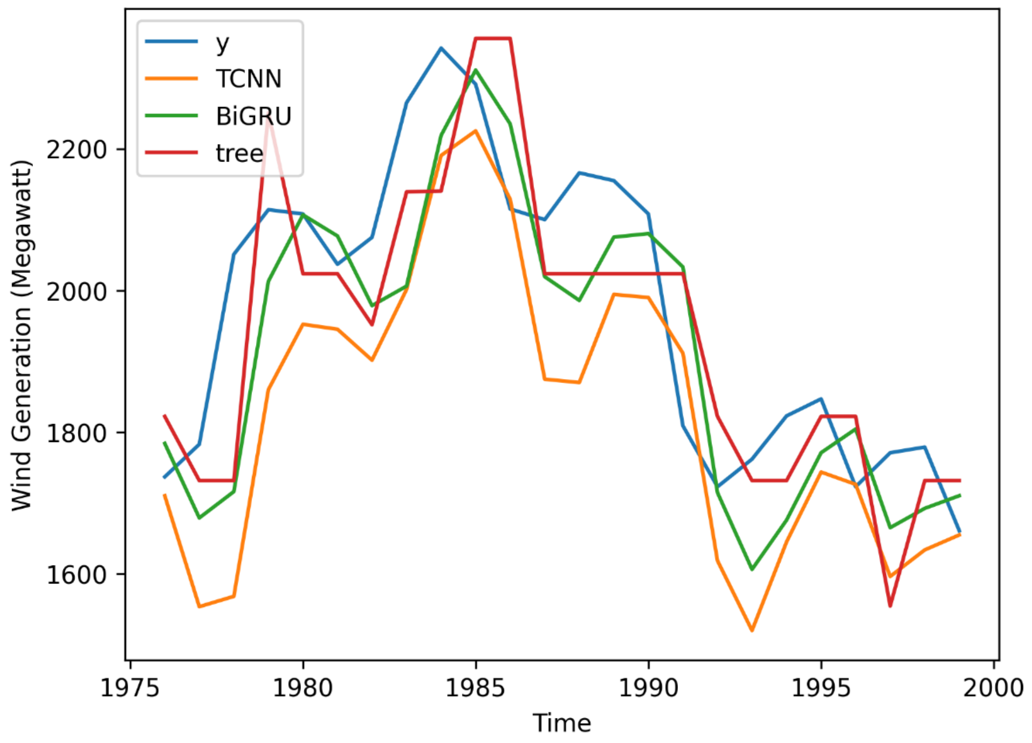


Figure 50 Best vs. Actual Results Last 24-hour for Austria Wind (60-minute intervals)

Table 14 Evaluation Results for Germany Wind (60-Minute Intervals)

Model	**RMSE**	**MAE**	**MAD**	**MASE**	**Optimization Time Used (Minutes)**
CNN	1885.93	1401.90	5281.51	1.98	7.02
TCNN	2399.06	1945.11	5223.12	2.74	202.02
LSTM	2416.48	1885.91	4572.87	2.66	51.41
BiLSTM	2344.06	1949.61	4925.83	2.75	83.73
GRU	1350.46	1018.58	5252.57	1.44	34.19
BiGRU	1358.89	1002.56	5191.29	1.41	77.57
DNN	1008.61	758.25	5248.66	1.07	3.67
kNN	2228.24	1623.88	4911.05	2.29	0.35
Tree	765.73	554.87	5332.17	0.78	0.26

Table 14 presents the results of 60-minute intervals for Germany. As with the corresponding 15-minute market, the Decision Tree becomes the only model with a MASE less than 1. DNN has a MASE slightly above 1, which narrowly misses the benchmark. The scenarios depicted in Figure 51 through Figure 54 are similar trends to those observed at 15-minute intervals. When examining these models on at the last 24-hour observations, there is a misalignment from the true value curve.

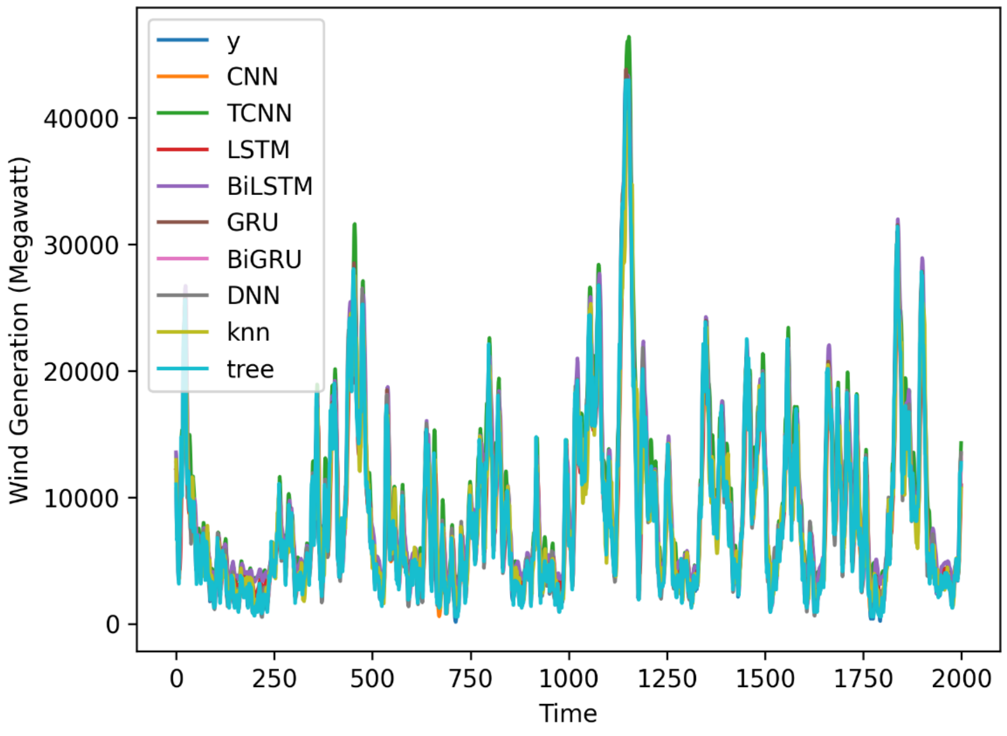


Figure 51 All Results for Germany Wind (60-minute intervals)

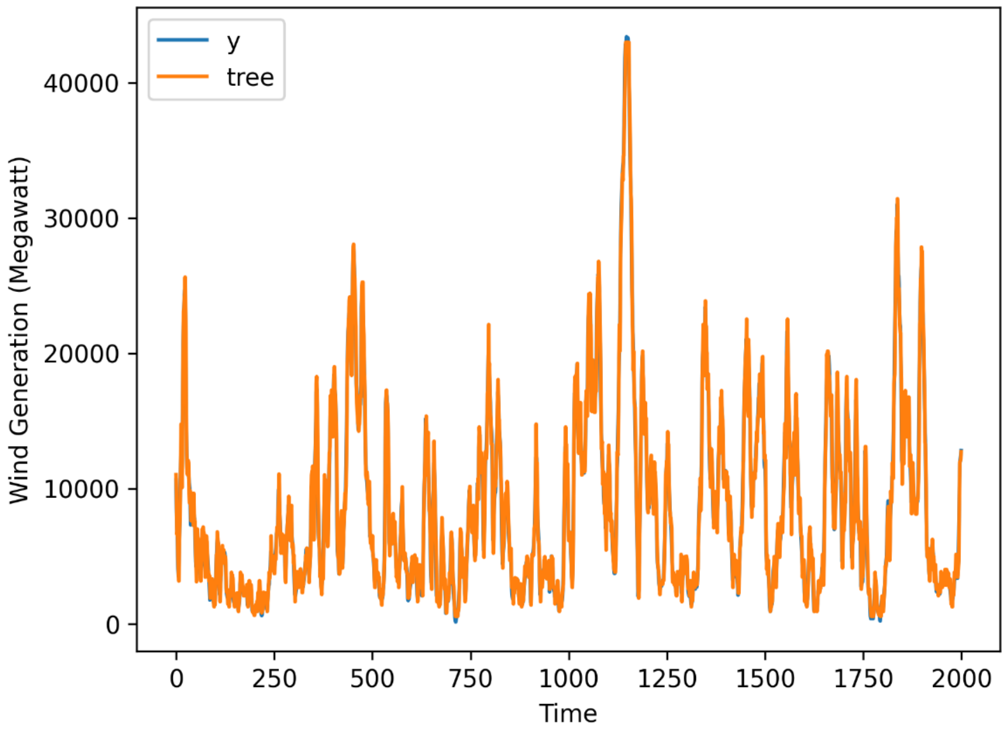


Figure 52 Best vs. Actual Results for Germany Wind (60-minute intervals)

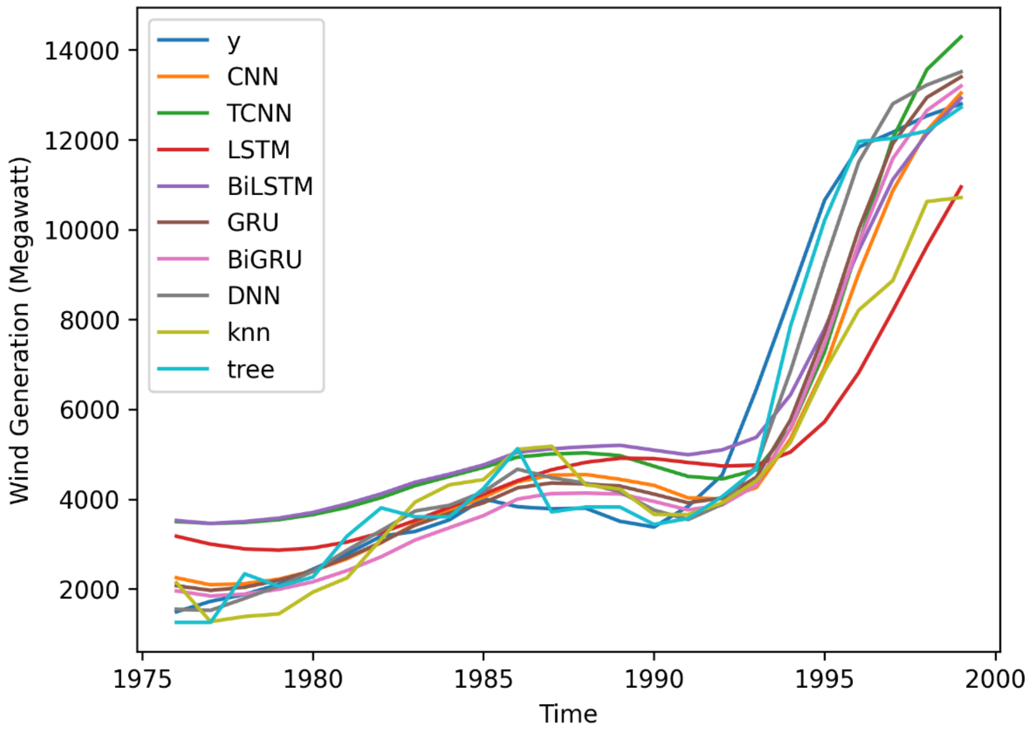


Figure 53 All Results Last 24-hour for Germany Wind (60-minute intervals)

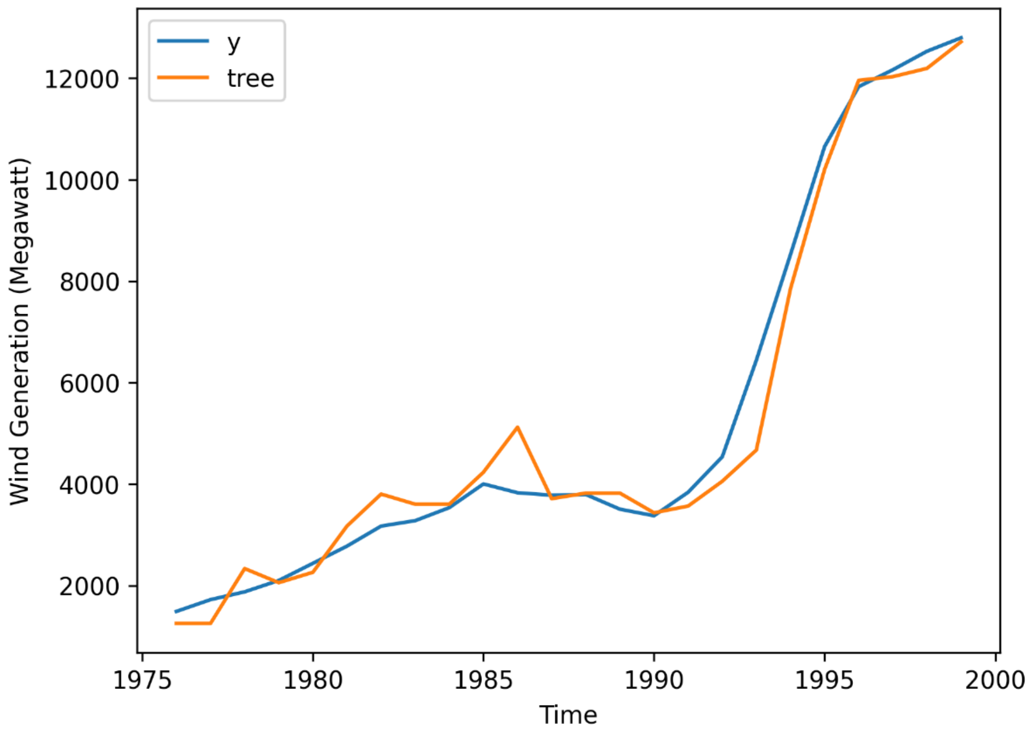


Figure 54 Best vs. Actual Results Last 24-hour for Germany Wind (60-minute intervals)

Table 15 Evaluation Results for Netherlands Wind (60-Minute Intervals)

Model	RMSE	MAE	MAD	MASE	Optimization Time Used (Minutes)
CNN	124.25	90.14	365.67	1.64	5.22
TCNN	83.59	60.10	397.16	1.09	148.45
LSTM	145.28	112.25	353.87	2.04	37.73
BiLSTM	104.73	76.66	367.15	1.40	88.66
GRU	90.09	65.88	378.76	1.20	38.16
BiGRU	87.53	62.84	378.54	1.14	75.44
DNN	92.51	66.62	378.68	1.21	3.68
kNN	147.00	112.01	357.58	2.04	0.34
Tree	78.82	55.65	377.81	1.01	0.27

Lastly, Table 15 presents the results of 60-minute intervals for Netherlands. Decision Tree again performs best, with an RMSE of 78.82. However, its MASE is slightly above the benchmark at 1.01. TCNN closely follows, with an RMSE of 83.59 and a MASE of 1.09. In Figure 55 through Figure 58, the models demonstrate a capacity to approximate the general trend of the dataset. However, they do not adequately capture the finer details of the true value curve.

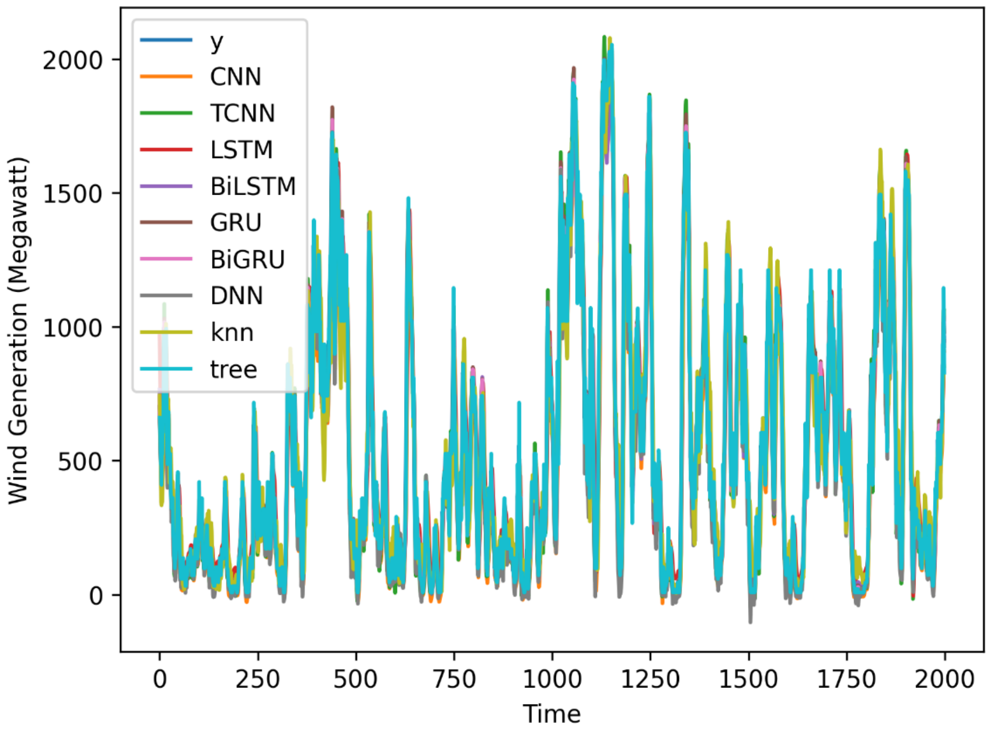


Figure 55 All Results for Netherlands Wind (60-minute intervals)

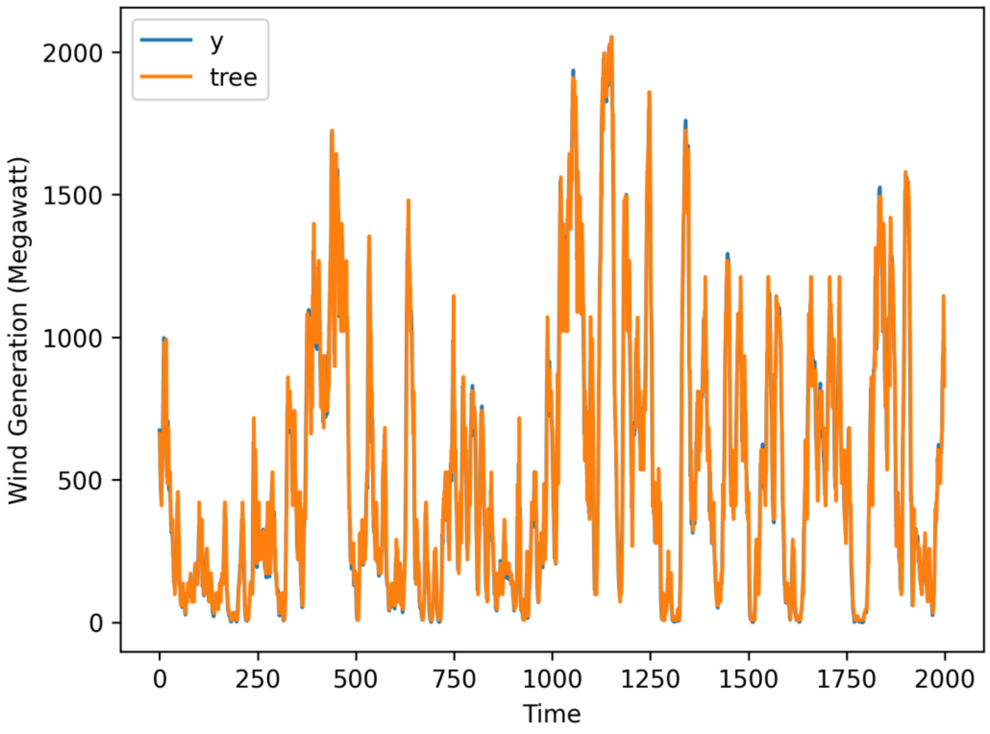


Figure 56 Best vs. Actual Results for Netherlands Wind (60-minute intervals)

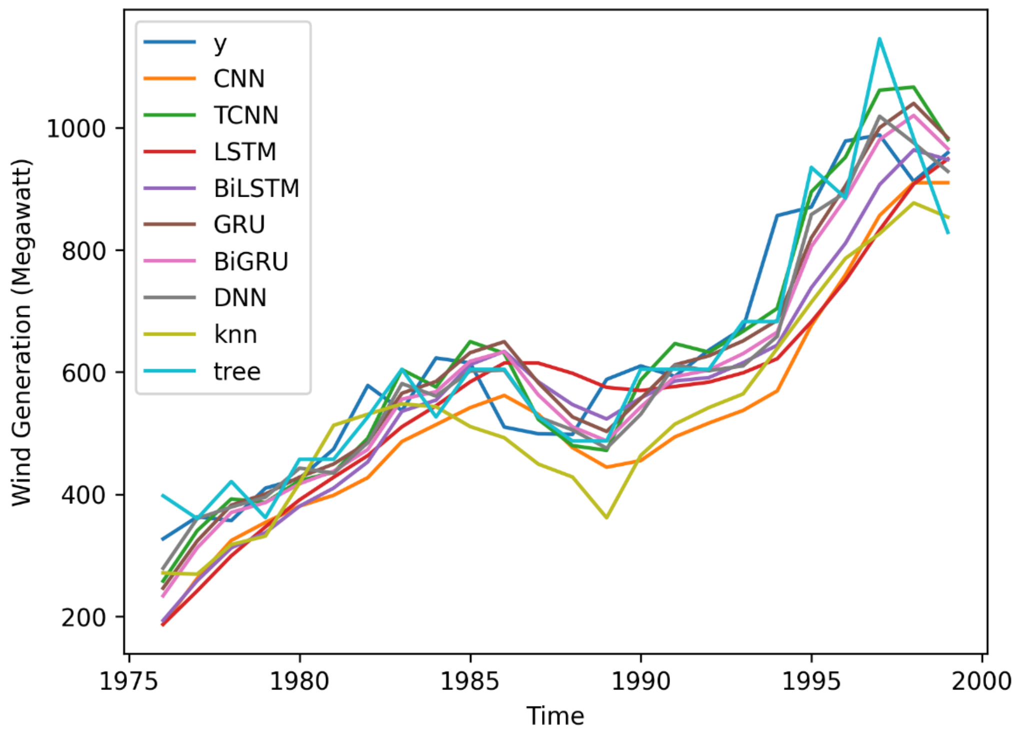


Figure 57 All Results for Netherlands Wind (60-minute intervals)

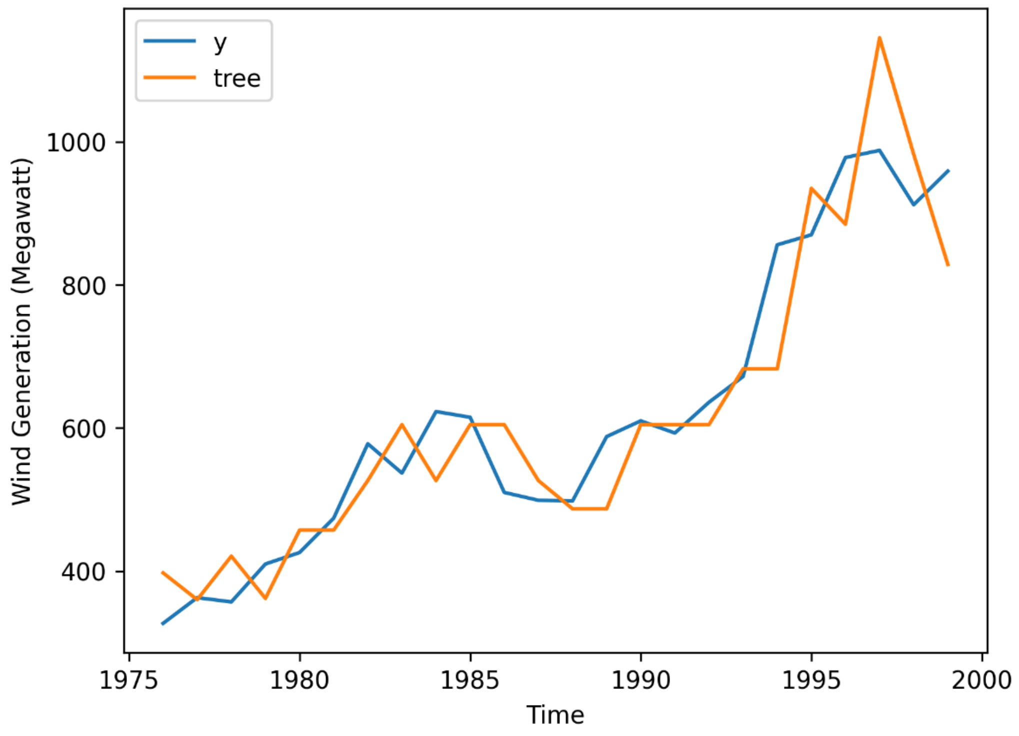


Figure 58 Best vs. Actual Results Last 24-hour for Netherlands Wind (60-minute intervals)

After comparing the performance within each respective market, we identified the most effective model for each individual market based on the established metrics. Then, these best performing models are selected for a side-by-side comparative analysis. The four tables below present the comparison.

Table 16 Comparison of Load (15-minute intervals)

Country	Model	RMSE	MAE	MAD	MASE	Optimization Time Used (Minutes)
Austria	**Tree**	63.07	46.77	1005.82	0.71	1.17
Germany	**Tree**	476.06	351.20	8113.38	0.66	0.25
Netherlands	**Tree**	166.91	107.21	1302.92	0.82	0.36

Table 16 presents the electricity load datasets at 15-minute intervals for all three countries. Decision Trees are the best models, yielding satisfactory predictions across all three datasets.

Table 17 Comparison of Load (60-minute intervals)

Country	Model	RMSE	MAE	MAD	MASE	Optimization Time Used (Minutes)
Austria	**DNN**	119.98	90.09	1004.36	0.38	4.60
Germany	**DNN**	809.76	609.66	7905.96	0.33	3.29
Netherlands	**DNN**	335.28	246.16	1197.70	0.68	5.15

Table 17 presents the electricity load datasets at 60-minute intervals. DNNs are the best models in terms of errors. And the computational time required for training is acceptable as well.

Table 18 Comparison of Wind (15-minute intervals)

Country	Model	RMSE	MAE	MAD	MASE	Optimization Time Used (Minutes)
Austria	**Tree**	47.61	30.28	548.70	1.00	0.21
Germany	**Tree**	253.29	178.07	5223.61	0.83	0.22
Netherlands	**Tree**	37.06	24.90	357.32	1.13	0.28

Table 18 shows the wind generation datasets at 15-minute intervals. The performance of the models generally fell short of expectations, except for the Germany dataset, where the models demonstrated relatively better predictive accuracy. Nevertheless, the Decision Tree consistently achieved the highest ranking in respective datasets.

Table 19 Comparison of Wind (60-minute intervals)

Country	**Model**	**RMSE**	**MAE**	**MAD**	**MASE**	**Optimization Time Used (Minutes)**
Austria	**TCNN**	153.52	99.65	447.48	1.05	289.37
Germany	**Tree**	765.73	554.87	5332.17	0.78	0.26
Netherlands	**Tree**	78.82	55.65	377.81	1.01	0.27

Table 19 demonstrates the wind generation datasets at 60-minute intervals. The datasets are not promising like the previous ones. Decision Tree are the best models for Germany and Netherlands, while TCNN is the best for Austria.

5. Discussion

5.1 Interpretation of Findings

Focusing on the datasets, an initial review indicates that traditional energy sources remain the predominant contributors to electricity load across all three countries analyzed. Concurrently, wind power constitutes a relatively minor segment of the total energy mix, its contribution significantly influenced by the stochastic characteristics of wind availability. The findings also underscore the reliance on conventional energy sources and highlight the challenges that renewable energy faces in increasing its market share and improving its reliability. Furthermore, the results suggest an ongoing need to balance the utilization of traditional energy and the integration and expansion of renewable energy to meet both current energy demands and future sustainability goals. As previously mentioned, the electricity load patterns in the three countries are considerably regular, which has helped the models make relatively good predictions. On the contrary, wind power presents more irregular patterns which complicate the models to generate reliable forecasts. It also highlights the need for sophisticated modeling techniques that can accommodate the considerable unpredictability of wind energy sources.

In terms of experimental results, the performance of the neural network models did not fully meet expectations, with only the DNN demonstrating notable proficiency within the 60-minute interval for electricity load datasets. While other neural network models successfully captured general trends, the quantitative metrics suggest their overall performance was merely average.

Several potential reasons could explain this. First, neural networks, particularly deep learning models, tend to be computationally intensive. In this research, each iteration of the optimization process took several minutes, and optimization was limited to thirty iterations. This constraint likely prevented the models from converging to the optimal parameter combinations, especially considering the complexity of neural networks. Thirty iterations may simply be insufficient for the models to explore the full range of parameter space, potentially limiting their ability to achieve superior results. Additionally, some models appeared to reach the upper bounds of the defined parameter space, suggesting that the parameter space itself may have been too restrictive, hindering the discovery of more effective configurations.

Another key factor may be the window size of thirty observations used for prediction. For the 60-minute intervals, this represents just over a day's worth of data, while for the 15-minute intervals, it corresponds to roughly seven and a half hours. These window lengths may not provide enough historical context for the models to fully capture long-term patterns, especially in datasets where consumption patterns or wind generation vary on longer timescales.

The nature of the data also presents inherent challenges. The electricity load data includes unpredictable peaks, and wind generation data is known for its inherent volatility and non-stationary characteristics. Decision tree may be better suited to handle these challenges due to their ability to segment the data and fit complex, non-linear relationships without needing as much fine-tuning or extensive optimization. In contrast, neural networks often require more data preprocessing, careful tuning of hyperparameters, and larger datasets to perform well. Moreover, decision trees are less sensitive to issues like multicollinearity and can often produce satisfactory results even when the underlying data is noisy or irregular, as is the case with wind generation and electricity load datasets.

Furthermore, decision tree models tend to be more robust in cases of overfitting when compared to deep learning models, which are prone to overfitting when working with smaller training sets or insufficiently varied data. In this context, the decision tree models might have achieved better generalization and performed more effectively on the test data.

In conclusion, the observed performance differences between the decision tree and neural network models can be attributed to several factors: the limited optimization iterations, restricted parameter spaces, potentially insufficient window lengths for neural networks, and the intrinsic complexities of the datasets. These factors collectively exacerbate the difficulty for neural networks to achieve high performance in these specific forecasting tasks, giving decision tree models a comparative advantage.

5.2 Implications

The research underscores the critical role of accurate forecasts in electricity load and wind generation for enhancing grid efficiency and supporting governmental operations. Leveraging advanced machine learning techniques and Bayesian optimization, organizations can make informed decisions to enhance efficiencies and operational capacities. The ability to forecast electricity load and wind generation is essential for effective energy management and extends beyond mere operational concerns. Accurate predictions help prevent power outages, thereby fostering social stability and enhancing grid profitability. Thus, the development of precise and dependable models is crucial for enabling informed decision-making in power consumption management. The broader economic implications are significant. Failures in power supply often cause social instability. By providing accurate forecasts of power load, organizations can proactively address potential energy crises.

The study provides a thorough exploration of models for predicting electricity load and wind generation. The detailed analysis of these prediction models can be employed to evaluate the status of power grids, offering stakeholders crucial data to enhance their management of electrical power and other related operations. It delivers essential insights into effective prediction models for electricity load and wind generation, which are instrumental for organizations managing power grids in optimizing power management and relevant processes. The research comprises instructive and technical content beneficial for academics and professionals within the industry.

5.3 Limitation and Future Research

The first notable limitation encountered in our study pertains to the computational resources utilized. The field of energy forecasting deals with extensive datasets, and in our initial dataset compilation, we had over a million observations in total. However, due to the constraints posed by our computing infrastructure, we have to select only the most recent observations for detailed analysis. This limitation was identified during our initial trials where training a single neural network model required more than 24 hours. Consequently, a decision was made to reduce the dataset size to manage computational demands more effectively. Furthermore, there are significant opportunities for enhancing the optimization of model parameters and the iterative settings used in our experiments. These aspects of our methodology were particularly influenced by the limitations of our current devices. With an upgrade in computational resources, there would be substantial scope to extend both the parameter search space and the number of iterations, potentially leading to improvements in model performance and accuracy. Future improvements in computational capacity could unlock new possibilities for more comprehensive data analysis and refined model tuning, thereby enhancing the robustness and applicability of the models.

While the current study provides substantial insights into energy forecasting, there remains a possibility that the models may not fully capture the complicated dynamics of real-world energy systems. Future research could focus on developing and testing more sophisticated models that incorporate a broader range of variables and interaction effects. As the energy sector continues to evolve, the integration of renewable energy sources becomes increasingly critical. Research could explore forecasting models that specifically address the variability and unpredictability associated with renewable energy sources, such as solar and wind power. There is an opportunity to expand the geographical scope of the research to include cross-regional studies that examine the applicability of forecasting models in different climatic and socio-economic contexts.

6. Conclusion

The research has systematically explored the efficacy of various predictive models in forecasting electricity load and wind generation, contributing significantly to the understanding and improvement of forecasting practices in the energy sector. The analysis reveals that among the various models tested, the Decision Tree consistently outperforms others in forecasting accuracy for almost every dataset. DNN is the most effective for predicting electricity loads at 60-minutes interval. The results of DNN implies the potential of advanced machine learning techniques in enhancing traditional energy forecasting methods.

For wind generation, there are significant challenges due to the inherent irregularity and unpredictability of wind patterns. As a result, almost every model evaluated in this study underperforms relative to expectations. The stochastic nature of wind makes it complex for models, and this difficulty is reflected in the consistently moderate performance. Despite advancements in predictive modeling, the capricious characteristics of wind continue to impede the ability of models to achieve optimal accuracy in wind generation forecasts. It underscores the need for research and development in this field to enhance the predictive capabilities of models dealing with such volatile energy sources.

The study's contributions extend beyond the empirical findings. Methodologically, it advances the application of complex neural network architectures in the energy sector, providing a blueprint for future research and development in energy forecasting. Practically, it offers actionable insights for energy policymakers and professionals on selecting and implementing forecasting models that can handle the complexities of modern energy systems. These findings not only pave the way for more reliable energy forecasting models but also contribute to the broader discourse on integrating machine learning into energy management systems, thereby supporting more sustainable and efficient energy use globally. The research highlights the potential for these technologies to transform energy operations, emphasizing the critical role of accurate forecasting in the transition to more adaptive and resilient energy systems.

The research also contributes to the theoretical underpinnings of energy forecasting by exploring and validating various predictive models for electricity load and wind generation. By comparing the efficacy of different models, this research has also contributed to theoretical discussions on the suitability of various forecasting methodologies under specific conditions encountered within the energy sector. Furthermore, the use of Bayesian optimization for hyperparameter tuning in complex models has demonstrated a robust framework for improving model performance, setting a precedent for future research to build upon.

In conclusion, the research fills a crucial gap in our understanding of electricity load and wind generation forecasting and offers a multi-dimensional benefit to the field. The continued exploration of these themes is essential for the evolution of energy management strategies and the development of policies that support the informed and efficient use of resources.

References

Albuquerque, P. C., Cajueiro, D. O., & Rossi, M. D. (2022). Machine learning models for forecasting power electricity consumption using a high dimensional dataset. *Expert Systems with Applications*, *187*, 115917.

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Breiman, L. (2017). *Classification and regression trees*. Routledge.

Cho, K. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Fix, E. (1985). *Discriminatory analysis: nonparametric discrimination, consistency properties* (Vol. 1). USAF school of Aviation Medicine.

Gao, F., Chi, H., & Shao, X. (2021). Forecasting residential electricity consumption using a hybrid machine learning model with online search data. *Applied Energy*, *300*, 117393.

Hadjout, D., Torres, J., Troncoso, A., Sebaa, A., & Martínez-Álvarez, F. (2022). Electricity consumption forecasting based on ensemble deep learning with application to the Algerian market. *Energy*, *243*, 123060.

Hochreiter, S. (1997). Long Short-term Memory. *Neural Computation MIT-Press*.

Huang, C., Karimi, H. R., Mei, P., Yang, D., & Shi, Q. (2023). Evolving long short-term memory neural network for wind speed forecasting. *Information Sciences*, *632*, 390-410.

Jones, L. E. (2017). *Renewable energy integration: practical management of variability, uncertainty, and flexibility in power grids*. Academic press.

Kariniotakis, G. (2017). *Renewable energy forecasting: from models to applications*. Woodhead Publishing.

Katzenstein, W. (2010). *Wind power variability, its cost, and effect on power plant emissions*. Carnegie Mellon University.

Kaytez, F. (2020). A hybrid approach based on autoregressive integrated moving average and least-square support vector machine for long-term forecasting of net electricity consumption. *Energy*, *197*, 117200.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.

Li, J., Wang, J., & Li, Z. (2023). A novel combined forecasting system based on advanced optimization algorithm-A study on optimal interval prediction of wind speed. *Energy*, *264*, 126179.

Open Power System Data. (2020). *Data package: Time series* (Version 2020-10-06). https://doi.org/10.25832/time_series/2020-10-06

Pawar, P., & TarunKumar, M. (2020). An IoT based Intelligent Smart Energy Management System with accurate forecasting and load strategy for renewable generation. *Measurement*, *152*, 107187.

Prema, V., & Rao, K. U. (2015). Development of statistical time series models for solar power prediction. *Renewable Energy*, *83*, 100-109.

Saranj, A., & Zolfaghari, M. (2022). The electricity consumption forecast: Adopting a hybrid approach by deep learning and ARIMAX-GARCH models. *Energy Reports*, *8*, 7657-7679.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, *25*.

Yaghoubirad, M., Azizi, N., Farajollahi, M., & Ahmadi, A. (2023). Deep learning-based multistep ahead wind speed and power generation forecasting using direct method. *Energy Conversion and Management*, *281*, 116760.