Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Application of AI-Based Methods for Electrical Load Forecasting in Power Systems

Master's Final Degree Project

Dovydas Morkūnas

Project author

Prof. Saulius Gudžius

Supervisor

Lect. Jonas Vaičys

Scientific consultant

Kaunas, 2023

Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Application of AI-Based Methods for Electrical Load Forecasting in Power Systems

Master's Final Degree Project

Electrical power engineering (6211EX010)

Dovydas Morkūnas
Project author

Prof. Saulius Gudžius
Supervisor

Lect. Jonas Vaičys
Scientific consultant

Doc. Gytis Svinkūnas
Reviewer

Kaunas, 2023

Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Dovydas Morkūnas

Application of AI-Based Methods for Electrical Load Forecasting in Power Systems

Declaration of Academic Integrity

I confirm the following:

1. I have prepared the final degree project independently and honestly without any violations of the copyrights or other rights of others, following the provisions of the Law on Copyrights and Related Rights of the Republic of Lithuania, the Regulations on the Management and Transfer of Intellectual Property of Kaunas University of Technology (hereinafter – University) and the ethical requirements stipulated by the Code of Academic Ethics of the University;

2. All the data and research results provided in the final degree project are correct and obtained legally; none of the parts of this project are plagiarised from any printed or electronic sources; all the quotations and references provided in the text of the final degree project are indicated in the list of references;

3. I have not paid anyone any monetary funds for the final degree project or the parts thereof unless required by the law;

4. I understand that in the case of any discovery of the fact of dishonesty or violation of any rights of others, the academic penalties will be imposed on me under the procedure applied at the University; I will be expelled from the University and my final degree project can be submitted to the Office of the Ombudsperson for Academic Ethics and Procedures in the examination of a possible violation of academic ethics.

Dovydas Morkūnas

Confirmed electronically

Morkūnas, Dovydas. Application of AI-Based Methods for Electrical Load Forecasting in Power Systems. Master's Final Degree Project / supervisor prof. dr. Saulius Gudžius, scientific consultant lect. Jonas Vaičys; Faculty of Electrical and Electronics Engineering, Kaunas University of Technology.

Study field and area (study field group): electrical engineering, engineering science.

Keywords: artificial intelligence, clustering, forecasting model, electrical load forecasting.

Kaunas, 2023. 55.

Summary

This project focuses on application of AI-based methods for forecasting electrical load. Hourly electrical load data from the year 2020 of 176 users connected to the Lithuanian electrical power distribution grid was used to create forecasting models. Dynamic time warping and Euclidean distance matrices, together with k-means and hierarchal clustering methods were used for clustering. A filter was implemented to filter out large irregularities. Forecasting model structure was created, aiming to forecast 24 hours into the future. A linear regression model was applied. TensorFlow was used for creation of forecasting models using temporal convolutional networks and feed-forward neural networks. Statistical model evaluation metrics were calculated to determine performance of the models. Results indicate that AI-based load forecasting models did not outperform linear regression models. The best performing model was found to be a linear model using DTW distance matrix and k-means clustering (R2 = 0,5438). The worst performing model was found to be a linear model using Euclidean distance matrix and k-means clustering (R2 = 0,3601).

Morkūnas, Dovydas. Dirbtiniu intelektu grįstų metodų taikymas apkrovų prognozavimo uždaviniuose elektros energetikos sistemose. Magistro baigiamasis projektas / vadovas prof. dr. Saulius Gudžius, mokslinis konsultantas lekt. Jonas Vaičys; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): elektros inžinerija, inžinerijos mokslai.

Reikšminiai žodžiai: dirbtinis intelektas, klasterizavimas, prognozavimo modelis, apkrovos prognozavimas.

Kaunas, 2023. 55 p.

Santrauka

Šiame projekte pagrindinis dėmesys skiriamas dirbtiniu intelektu pagrįstų metodų taikymui apkrovų prognozavimui. Prognozavimo modelių sudarymui naudoti 176 vartotojų, prijungtų prie Lietuvos elektros skirstomojo tinklo, 2020 metų valandinės apkrovos duomenys. Buvo atliktas klasterizavimas naudojant dinaminio laiko deformacijos ir Euklido atstumų matricas, kartu su k-vidurkių ir hierarchiniu klasterizavimo metodais. Staigiems apkrovos šuoliams išfiltruoti buvo panaudotas filtras. Sukurta prognozavimo modelio struktūra, siekiama prognozuoti apkrovą 24 valandas į ateitį. Buvo pritaikytas tiesinis regresinis modelis. TensorFlow buvo naudojamas prognozavimo modelių kūrimui naudojant temporalinius konvoliucinius tinklus ir grįžtamuosius neuroninius tinklus. Modelių tikslumui nustatyti buvo apskaičiuotos statistinės modelių vertinimo metrikos. Rezultatai rodo, kad dirbtiniu intelektu grįsti apkrovos prognozavimo modeliai nepralenkė tiesinės regresijos modelio. Geriausiai veikiantis modelis buvo tiesinis modelis, naudojant DTW atstumo matricą ir k-vidurkių klasterizavimą (R2 = 0,5438). Prasčiausiai veikiantis modelis buvo tiesinis modelis, naudojant Euklido atstumo matricą ir k-vidurkių klasterizavimą (R2 = 0,3601).

Table of contents

List of figures

List of tables

List of abbreviations and terms

Abbreviations:

ADN – Active distribution network;

ADMM – Alternating direction method of multipliers;

AI – Artificial intelligence;

ANN – Artificial neural network;

BPNN – Back-propagation neural network;

CNN – Convolutional neural network;

DDPG – Deep deterministic policy gradient;

DDQN – Double deep Q-network;

DER – Distributed energy resources;

DL – Deep learning;

DQN – Deep Q-network;

DR – Demand response;

DRL – Deep reinforcement learning;

DTW – Dynamic time warping;

ESS – Energy storage system;

EV – Electric vehicle;

FL – Fuzzy logic;

FNN - Feed-forward neural network;

GRU – Gated recurrent unit;

HESS – Hybrid energy storage system;

HVAC – Heating ventilation and air conditioning;

IoT – Internet of Things;

LSTM – Long short-term memory;

MAPE – Mean absolute percentage error;

MLP – Multilayer perceptron;

ML – Machine learning;

PEV – Plug-in electric vehicle;

PV – Photovoltaics;

RL – Reinforcement learning;

RNN – Recurrent neural network;

RMSE – Root mean square error;

SG – Smart grid;

TCN – Temporal convolutional network;

Introduction

Energy plays an important role in the world. A consistent increase in global energy use has been observed in recent years. To adapt, we need to focus on increasing clean energy generation and replacing other forms of electricity sources. The challenge is finding ways to produce and use clean energy on a large scale. Looking into the future, renewable resources like wind, solar, and hydro power will be the main sources of energy [1].

Not only generation sources have to change. It is crucial that consumer habits and awareness also change. The rising popularity of renewable energy in the global power grid makes it crucial to enhance the precision of its predictions, which is important for the planning, managing, and running power systems. However, this is difficult due to the unpredictable and inconsistent nature of renewable energy data [2].

The power grid is gradually adopting a connected, smart grid (SG) system based on the Internet of Things (IoT). This offers numerous advantages, but it also gives rise to new challenges that we did not have previously. The large amounts of data produced by the SG require new techniques for data analysis. Machine learning (ML) approaches are the key solutions needed for efficient data management and extraction [3].

So far, many strategies have been created to enhance the prediction accuracy of renewable energy. These include physical models, statistical approaches, artificial intelligence (AI) methods, and combinations of these techniques. Deep learning (DL), a type of ML with the potential to find non-linear patterns and consistent subtle structures within data, has been regularly mentioned in various studies [2].

Various ML-based and AI-based methods and models are being used in electrical power systems, such as reinforcement learning (RL), recurrent neural networks (RNN), temporal convolutional networks (TCN), and feed-forward neural networks (FNN). Application of these methods will be discussed in this project.

In power systems, opportunities to improve the accuracy and efficiency of electrical load forecasting lead to reduced costs, improved stability, and improved customer satisfaction.

The aim of this project is to investigate the application of AI-based methods for electrical load forecasting in power systems, and to apply AI-based methods to create forecasting models to forecast electrical load.

1. Literature review of AI-based methods for electrical load forecasting in power systems

The electrical power system is a critical infrastructure that requires high levels of stability and reliability. The volume and complexity of data involved in managing the power grid is growing rapidly. ML and AI are the tools to help us to keep up with the growth. As AI improves rapidly, understanding and utilizing its capabilities is becoming increasingly important. Through an in-depth review of current literature, practical applications, and case studies, this project aims to show how AI-based methods can be used to improve the current power system, mainly focusing on load forecasting.

1.1. Artificial intelligence in electric power systems

The change in our energy grid mainly involves a shift from traditional, centralized energy sources to distributed energy resources (DERs), which are less harmful to the environment. To overcome the difficulties presented by the unpredictable nature of renewable energy, we need innovative strategies. SG technologies like modern metering systems, energy storage systems (ESS), and home energy management systems are being installed worldwide to adjust to this change [4].

SG has emerged as a significant innovation in the electrical power industry over the last ten years. New advanced technologies are constantly being introduced, forcing engineers to consider how they can use them to enhance the efficiency of the power system. Examples of these new technologies include AI techniques such as ML, Artificial Neural Networks (ANN), DL, RL, and Deep-Reinforcement Learning (DRL) [5].

DERs have been gaining attention due to having many advantages. They are affordable and can be constructed quickly, compared to larger centralized power plants. This makes DERs attractive in evolving and liberalized electricity markets. In certain cases, DERs can eliminate the need to strengthen the grid, decreasing the expenses of building and maintaining high-voltage power lines. Also, utilizing DERs minimizes the need for central dispatch and decreases energy losses by having consumption closer to generation sources. DERs also enhance microgrid resilience, ensuring local supply during grid disturbances. A typical microgrid consists of DERs, electrical loads, and an ESS. DERs consist of renewable energy sources, like wind turbines or solar photovoltaics (PV), and often include a backup power generator [4].

As the population increases, living standards rise, and people use more power-hungry devices in their homes, the amount of energy used by homes has increased significantly in the recent years. It's important to smartly manage how homes use power to help them save on electric bills and reduce the load on the power grid during peak hours. The demand side can help manage energy by helping to balance demand and supply [6].

The concept of AI originated in 1956. At its core, AI is a developing field that merges various areas such as cybernetics, informatics, computer science, mathematical logic, and neurophysiology to stretch and extend human intelligence. “The ultimate goal of AI is to enable computers to think and act as capably as humans and eventually, to enhance human intelligence. AI is particularly apt at sorting through data, finding patterns, and making predictions” [1].

The most critical issue that comes from the generation of large amounts of data in SG is identifying effective methods to analyze and extract meaningful information. Without finding patterns from the collected data, the data serves little to no purpose. ML is the necessary tool to do the job of data analysis of the large amounts of data produced in an IoT-based power system. It serves as a crucial component in the SG systems, which is mainly driven by data collection, analysis, and decision making. ML methods offer a practical means to analyze this data and by doing that helps to make correct decisions to operate the power grid, helping the SG to function for what it was designed [3].

In recent years, Google's DeepMind developed AlphaGo, which mastered the game of Go, even defeating the world's leading human players. More recently, it advanced to AlphaGo Zero and AlphaZero [7] showcasing the big progress of DRL, a category of ML. The innovative ML algorithms like DRL have achieved major breakthroughs due to considerable increase of computational power, sophisticated learning models, and large amounts of data. "These algorithms will provide significant opportunities and motivate vigorous development in the Smart energy and electric power system field represented by SG and electricity internet (EI)" [1].

The conventional approaches used in analyzing, controlling, and making decisions in electrical power systems mainly rely on physical modeling and numerical computations. Within the power systems, utilizing AI technologies are considered to be a big opportunity to significantly increase cost-effectiveness and reduce costs. "The AI techniques, such as expert systems (ESs), ML, fuzzy logic (FL), ANNs, and RL are the few examples of cutting-edge technologies by which large volume of collected information being processed, and deliver the solution to the complex problems associated with SG" [5].

Thomas et al. [8] studied a hybrid energy management system consisting of a university building, 30 EVs used for work related trips, 150 kWh, 20 kW ESS and 50 kWp solar power plant. The aim of the study was to apply a new algorithm to an energy management system in order to reduce the total cost of electricity consumption. To achieve the goal, clustering algorithms with a stochastic approach to generation and consumption volatility were used. The applied algorithm allowed to achieve almost 17 times lower costs compared to the previously used deterministic method.

1.2. Reinforcement learning in electric power systems

When considering supervised learning and unsupervised learning, RL represents a form of active learning. RL can be considered to be a dynamic learning process that adapts and evolves based on the context. "RL emphasizes taking action based on state to achieve a maximum expected reward, as shown in Figure 1, where RL interacts with its environment via a trial-and-error mechanism and learns the optimal strategy by maximizing the cumulative reward" [1].

In real operational scenarios, power systems usually have numerous unpredictable factors. Since these factors change continually, the system's dynamic response cannot be identical for every error. Supervised learning based controllers need large amounts of training data and often do not deliver optimal control decisions. "As stated above, when using the RL method, the system only needs to respond to the evaluated information of the current control effect; consequently, it has high real-time control capability and robustness, which has resulted in widespread RL use in power system security and stability control" [1].

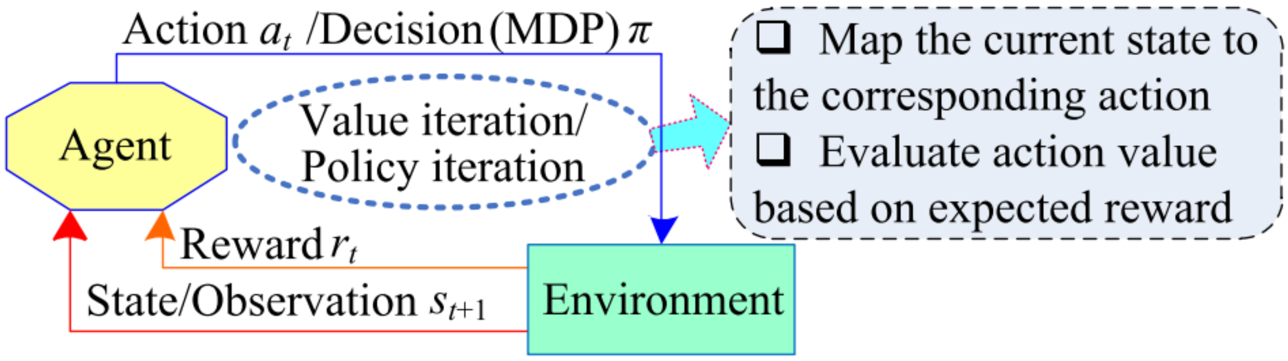

Action a_t /Decision (MDP) π
Agent
Value iteration/ Policy iteration
❑ Map the current state to the corresponding action
❑ Evaluate action value based on expected reward
Reward r_t
State/Observation s_{t+1}
Environment

Figure 1. The basic principle of RL [1]

Based on RL, Ernst et al [9] were the first in creating a two-part system for maintaining stability in power systems, which used both an online and offline mode. These two online and offline modes of RL control are part of this structure. The online mode involves the agent continuously interacting with and learning from the real environment in an attempt to find the best strategy. "This learning applies to cases where system modeling is difficult or when special operating conditions cannot be reproduced in the simulation model" [1].

Vázquez et al. [10] have found that there was a significant increase in the number of studies that involved RL after 2012, and a year after the publication of the paper "Playing Atari with Deep Reinforcement Learning" [11] there was a spike in 2015. The quantity of published works has risen significantly involving application of RL techniques to electric vehicles (EVs), which was barely researched before 2013. "After 2013, there was also a significant increase of publications in the areas of RL applied to smart appliances, and distributed generation (DG) combined with energy storage systems (i.e. solar PV and batteries)" [10].

In the last few years, several researchers have tried to combine RL and DL, creating new DRL techniques. The primary objective of this was to improve the stability and security of power systems. [1].

RL has been used in managing a wide range of energy systems. These include EVs, heating, ventilation, and air conditioning (HVAC), smart household devices, and battery systems. "The future of demand response (DR) greatly depends on its ability to prevent consumer discomfort and integrate human feedback into the control loop. RL is a potentially model-free algorithm that can adapt to its environment, as well as to human preferences by directly integrating user feedback into its control logic" [10].

Overall, the field of RL is rapidly advancing, and the development of new innovative algorithms for various applications may provide solutions to challenges that were previously difficult to address [10].

1.2.1. Reinforcement learning in energy management systems

The growing need for power, together with its crucial role in today's world, highlights the necessity for smart home energy systems that can decrease energy consumption if needed [12].

“Superior energy management strategies can help to extend the battery service lifetime and reduce the system lifetime cycle cost by controlling the battery charge/discharge rate appropriately” [13].

Nakabi et al. [4] have studied applications of multiple DRL algorithms aimed to improve microgrid’s the power management system. They have proposed “a novel microgrid model that consists of a wind turbine generator, an energy storage system, a set of thermostatically controlled loads, a set of price-responsive loads, and a connection to the main grid”. “The proposed energy management system is designed to coordinate among the different flexible sources by defining the priority resources, direct demand control signals, and electricity prices. Seven DRL algorithms were implemented and are empirically compared in the paper” [4]. The results clearly indicate that DRL methods vary significantly in how well they can find the best solutions. By tweaking the used asynchronous advantage actor-critic approach, introducing an experience replay function and a semi-deterministic training phase, these modifications have led the researchers closer to getting optimal results.

Liu et al. [14] have proposed “a home energy management optimization strategy based on deep Q-network learning (DQN) and double deep Q-network learning (DDQN) to perform scheduling of home energy appliances”. “The applied algorithms are model-free and can help the customers reduce electricity consumption by taking a series of actions in response to a dynamic environment. In the test, the DDQN is more appropriate for minimizing the cost in a home energy management system (HEMS) compared to DQN” [14].

A Hybrid Energy Storage System (HESS) offers a viable solution to the challenges of single energy storage systems. It efficiently meets the demands for both high power and energy requirements, specifically for plug-in hybrid electric vehicles (HEVs) [13].

Xiong et al. [15] studied “RL-based real-time power management for hybrid energy storage system in the plug-in hybrid electric vehicle. Power allocation is a crucial issue for hybrid energy storage system (HESS)”. In the study, real-time power-management strategy based on RL is raised to obtain the best power distribution between the battery and the ultracapacitor. “A comparison between the RL-based online power management and the rule-based power management shows that the RL-based online power management strategy can lessen the energy loss effectively and the relative decrease of the total energy loss can reach 16.8%”. “The results indicate that not only can the RL-based real-time power management strategy limit the maximum discharge current and reduce the charging frequency of the battery pack, but also can decrease the energy loss and optimize the system efficiency” [15].

1.2.2. Reinforcement learning in demand response

With the progress of SG technologies, DR has significantly contributed in enhancing grid reliability and minimizing cost for consumers [16]. “Buildings, as major energy consumers, can provide great untapped DR resources for grid services” [17].

Incentive-based DR programs can be used to encourage users to reduce electricity demand during peak periods by introducing rewards [18].

A study by Wen et al. [18] proposed an incentive-based DR program that used modified DL and RL. “A modified DL model based on RNNs was first proposed to identify the future uncertainties of environment by forecasting day-ahead wholesale electricity price, photovoltaic (PV) power output, and power load”. RL was used to find the optimal incentive rates during different time periods to maximize the profits for both energy service providers (ESPs) and energy users. “The results showed that the proposed modified DL model can achieve more accurate forecasting results compared with some other methods. A short-term DR program was developed for peak electricity demand period, and the experimental results show that peak electricity demand can be reduced by 17%” [18].

Paper by Lu et al. [16] proposes a multi-agent DRL based DR method for energy management of discrete manufacturing systems. “In this regard, the industrial manufacturing system is initially formulated as a partially-observable Markov game. Then, a multi-agent deep deterministic policy gradient algorithm is adopted to obtain the optimal schedule for different machines”. The simulation demonstrated that the introduced DR algorithm has the capability to reduce electricity expenses and maintain production tasks, in comparison to a benchmark of not using DR.

“The existing literature that is targeted toward optimal DR management problems can be predominantly divided into two categories; their key features, advantages, and limitations”. The summary is given in Table 1 [19].

Table 1. Summary of the existing methods used in optimal DR management [19]

Category	Key features	Modeling Method	Advantages or Limitations
Model-based	Relies on full knowledge of distributed energy resource’s (DER’s) operating model and parameters Relies on an accurate forecast of exogenous parameters Unable to deal with the multi-source uncertainties effectively and efficiently	Deterministic	Unable to deal with uncertainties
		SP	Unable to accurately estimate the probability distribution of uncertain parameters Computationally inefficient
		RO	Leads to overly conservative solutions
Model-free	Requires no full system identification and no a priori knowledge of the system Employs data-driven and machine learning approaches to learn a generalizable DR strategy Computationally efficient at deployment	RL	Unable to deal with problems with high-dimensional continuous states and/or action spaces
		DRL	Capable of handling high-dimensional continuous states and action spaces Effective learning of fine-grained control policies

In RL, an agent is trained to develop a policy that is nearly optimal by interacting repeatedly with a black-box environment, meaning that it lacks the complete description of a system and prior knowledge of the environment. “Furthermore, an RL agent can harness the increasing influx of data collected from Internet of Things sensors, and thus enables successive data processing and interpretation to train a representation of the DR management strategies that are generalizable and cope with the environmental uncertainties” [19]. As a result, a properly trained RL model has the capability to quickly generate real-time decisions for DR management, completing assigned energy management tasks, with response times measured in milliseconds, while also being computationally effective.

In residential DR management problems, the conventional Q-learning (QL) approach has gained popularity and is widely used because of its simplicity [19].

Zhang et al. [17] proposed a solution using RL to cost-effectively integrate optimal decision-making problems. “Beside RL’s ability to solve sequential optimal decision-making problems, its adaptability to easy-to-obtain building models and the off-line learning feature are likely to reduce the controller's implementation cost" [17].

Lately, there has been an increasing number of studies aiming to combine RL and DL. "Deep RL (DRL) techniques promise effective learning of more sophisticated and fine-grained control policies than those achieved by traditional RL methods founded upon look-up tables or shallow regression models. In this regard, the deep Q network (DQN) method constitutes the most popular approach" [19].

Mathew et al. [12] presented a DRL model for DR, in which the virtual agent learns to do the task in a manner similar to how humans learn to do it. "The proposed approach outperformed the state-of-the-art mixed integer linear programming for load peak reduction. The proposed model was analyzed with loads from five different residential consumers. The method increased monthly savings of each consumer by reducing their electricity bill drastically along with minimizing the peak load of the grid" [12].

Ye et al. [19] created a novel real-time strategy for managing DR in a residential home, utilizing the twin delayed deep deterministic policy gradient (TD3) learning method. "This approach was model-free, and thus does not rely on knowledge of the distribution of uncertainties or the operating models and parameters of the DERs. The proposed method was applied to the energy management problem for a household with a portfolio of the most prominent types of DERs". "Case studies involving a real-world scenario were used to validate the superior performance of the proposed method in reducing the household's energy costs while coping with the multi-source uncertainties through comprehensive comparisons with the state-of-the-art DRL methods" [19].

DR applications that utilize monetary incentives play an important role in helping to achieve a better balance between supply and demand within the power system, and by doing so enhancing its overall reliability [18].

1.2.3. Deep reinforcement learning

DL is a fundamental ML method, just like supervised and unsupervised learning. Various RL methods include the Q-learning, SARSA (State-Action-Reward-State-Action), DQN (Deep Q Net), and DDPG (Deep Deterministic Policy Gradients). The combination of RL with DL has led to an advanced technique known as DRL. This approach is mostly used in unknown environments and can be applied to numerous tasks that require deep understanding of complex situations, raw inputs, and strategic control. The DRL development was introduced in 2013, by Google's Deep Mind [5].

In order to optimize the utilization of renewable energy, an algorithm AccCap-DRL, based on DRL was proposed by Liu et al. [20]. "AccCap-DRL partitions a distribution into segments by time intervals, employs Wasserstein generative adversarial network (WGAN) to describe distributions of renewable energy data, and employs DDPG to obtain approximate policies for renewable energy accommodation in different scenarios". "Simulation results from real power generation and users' demand data show high effectiveness of the proposed algorithm, and high efficiency of evaluating accommodation capability" [20].

Wang et al. [21] proposed a DR method "to reduce the long-term charging cost of single plug-in electric vehicles (PEV) while overcoming obstacles such as the stochastic nature of the user's driving behaviour, traffic condition, energy usage, and energy price". "The problem is formulated as a Markov Decision Process (MDP) with an unknown transition probability matrix and solved using DRL techniques. The proposed method does not require any initial data on the PEV driver's behaviour and shows improvement on learning speed when compared to a pure model-free RL method". "A combination of model-based and model-free learning methods called Dyna-Q RL is utilized in their strategy. A value approximation method using deep neural networks is employed for estimating the long-term expected reward of all state-action pairs". A combination of historical price data and a long short-term memory (LSTM) network was employed to forecast future prices. The simulation results show the effectiveness of this method and its capacity to identify of an optimal strategy, while preventing the depletion of the state of charge (SOC) during trips, as compared to current EV charging systems [21].

The DRL community has achieved multiple independent enhancements to the DQN algorithm. However, it remains uncertain which of these advancements have the potential to be effectively integrated. Paper by Hessel et al. [22] examines "six extensions to the DQN algorithm and empirically studies their combination". Their experiments show that "the combination provides state-of-the-art performance on the Atari 2600 benchmark, both in terms of data efficiency and final performance".

Kolodziejczyk et al. [23] proposed a DRL-driven method to reduce the cost associated with real-time energy procurement in a storage-integrated PV system implemented within a microgrid. A combination of the Q-learning algorithm and a dense deep neural network is used to create an optimal decision policy. "The algorithm incorporates enhancements that were found to improve learning speed and stability by prior work, such as experience replay, target network, and increasing discount factor. Extensive simulation results performed on real data confirm that their approach is effective and outperforms rule-based heuristics".

Hua et al. [24] explored the energy management issue in the field of energy Internet (EI) using a multidisciplinary approach. "In this paper, a new energy regulation issue is considered based on the operational principles of EI. Multiple targets are considered along with constraints. Then, the practical energy management problem is formulated as a constrained optimal control problem". "Notably, no explicit mathematical model for power of renewable power generation devices and loads is utilized. To obtain the desired control scheme, a model-free DRL algorithm is applied". The results indicate that the suggested approach outperforms the comparison with the optimal power flow solution.

1.2.4. Reinforcement learning problems

Vázquez et al. [10] review indicates that while many studies consider human comfort and satisfaction, the majority of them focus on individual systems operating under demand-independent electricity prices and a stable environment. "However, when electricity prices are modelled as demand-dependent variables, there is a risk of shifting the peak demand rather than shaving it". They have identified that applications of RL need to be further explored if multi-agent systems want to be used to participate in DR programs under demand-dependent electricity prices.

More RL algorithms have to be tested in real-world applications, including applications of HVAC control and EVs, rather than only using simulated environments to test the algorithms. "One of the reasons that might be preventing building owners and building energy managers from implementing RL controllers in actual commercial or residential buildings is the lack of experimental results in physical systems proving its capabilities and more importantly, its reliability" [10].

It should be noted that the complexity of the problem in question should determine which RL algorithms should be used. It is not always necessary to use complex methods, as simpler algorithms may often be enough for solving a given problem [10].

Vázquez et al. [10] suggest that, “in addition to defining the basic information for RL, i.e., the states, actions, rewards, and the type of action-selection algorithm used in the RL problem, authors should clearly state whether”:
1. The states, actions and rewards are deterministic or stochastic.
2. The transition probabilities of the environment are stationary (whether they remain constant during the simulation or experiment), and which state transitions, if any, are non-stationary.
3. Agents act independently, simultaneously, or sequentially on the environment.
4. Electricity prices are demand dependent or independent.
5. The control algorithm is model-free or model-based, off-policy or on-policy.
6. Any predictions are used as states, and how accurate they are.
7. The algorithm was tested in a physical system or in a simulated environment.
8. Actual human feedback was used.

“Following this basic framework will improve the understanding of the diverse problems being tackled, increase the reproducibility of the results, and highlight for which type of problems certain RL algorithms perform better than others” [10].

1.3. Machine learning

ML is a concept that describes how a system learns and makes forecasts from the given data. It uses different types of algorithms to examine data, following specific rules to make predictions and decisions based on the data. The process of machine learning involves designing and programming specific algorithms to achieve performance [3].

In electric power systems, ML covers multiple areas of use. ML is involved in estimating load, cost, energy production, forecasting optimal schedules, assisting in identifying faults, adaptive control, determining appropriate system values, and detecting potential unauthorized access during a data breach [3].

ML is a tool for managing large amounts of data and finding valuable insights. It can be used in tasks like pattern recognition in demand and generation, generation forecasting or control applications. A large number of methods have been already discussed in prior studies, and more novel techniques are being developed to improve performance in certain use cases [3].

In the era of big data, ML is crucial for finding useful insights from the data. For those with access to big data and sufficient computational resources, ML has become a significantly important tool. The current rise of AI can be largely attributed to advancements in ML, especially the significant progress made in DRL. DL is a branch of ML, and ML itself is a fundamental area of research in AI. ML has been used as a solution to address bottlenecks related to finding insights about data [1].

Cheng et al. [1] identified that the following bottlenecks will be encountered in the next steps of ML development:
1. Data bottleneck. “ML requires massive effective samples for training. Ideally, we strive to develop good models with a small amount of data, even in the big data era”.

2. Generalization bottleneck. Most of current ML methods still cannot achieve ideal generalization performance or trained models in some practical problems. Generalization performance degradation occurs when a trained model is used in a changing environment. Furthermore, the process of training a new model is time-consuming; thus, we need to use limited set of data to gain control to adapt to different circumstances [1].
3. Energy consumption bottleneck.
4. Interpretability bottleneck. "Most of the current ML systems are black boxes, meaning that we can use them to make predictions - even very accurate predictions - but it is difficult to explain how such predictions are made. This will cause black box-like models to be difficult or impossible to use in certain high-risk application scenarios with high security and robustness requirements such as power network dispatching" [1].
5. Reliability bottleneck. "Issues such as data privacy and ownership will prevent us from effectively sharing ML experiences".

1.3.1. Deep learning

In the areas of AI, DL and DRL have gained popularity in the last few years, the use of DL in AI applications has been attracting a lot of interest. This approach has become particularly important in a wide range of research fields. Traditional ML methods have limitations when processing raw data. However, DL allows the processing of such raw data without the need for feature extraction, unlike traditional methods. DL techniques signify a new phase in AI development, as they overcome these challenges of dealing with raw data [5].

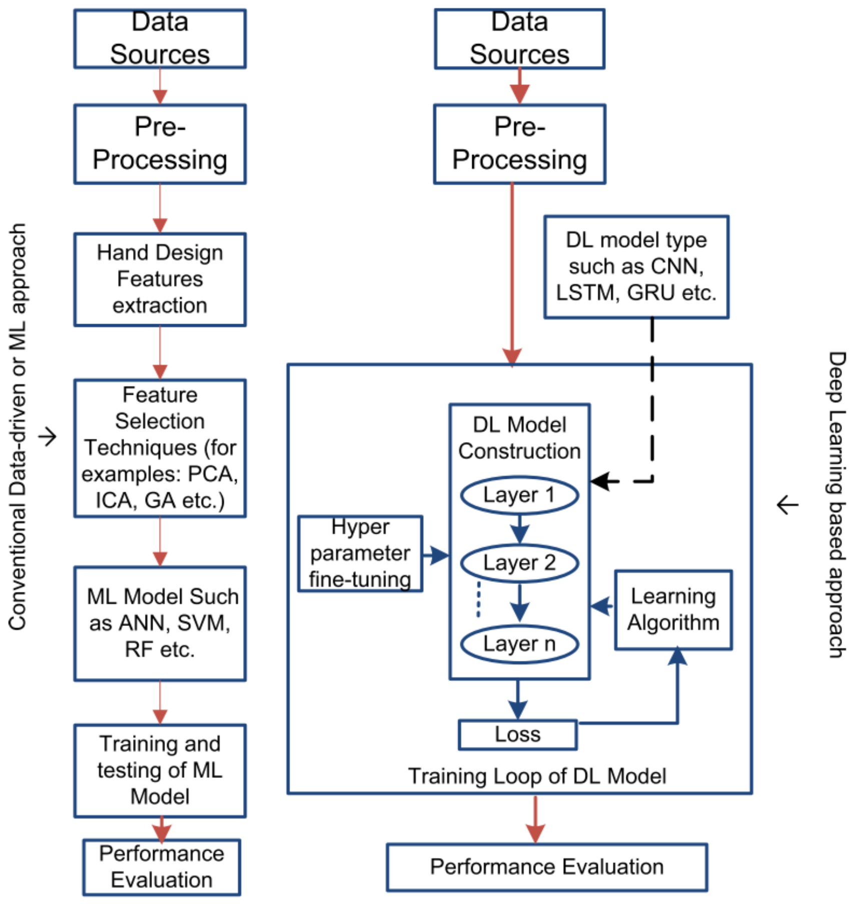

Conventional Data-driven or ML approach
Data Sources
Pre-Processing
Hand Design Features extraction
Feature Selection Techniques (for examples: PCA, ICA, GA etc.)
ML Model Such as ANN, SVM, RF etc.
Training and testing of ML Model
Performance Evaluation
Deep Learning based approach
Data Sources
Pre-Processing
DL model type such as CNN, LSTM, GRU etc.
DL Model Construction
Layer 1
Layer 2
Layer n
Hyper parameter fine-tuning
Learning Algorithm
Loss
Training Loop of DL Model
Performance Evaluation

Figure 2. Conventional ML vs DL [5]

"In contrast to conventional ML, the DL uses multiple layers for the extraction of a higher-level feature vector progressively from the raw input data". Figure 2 shows general differences between conventional ML and DL [5].

DL has a wide range of uses in the field of electrical engineering. It's being applied in areas like improved monitoring and diagnostics for power systems, adoptive protection schemes, and the management of distributed power systems. DL also eases the possibility of island operation in microgrids, supports advanced forecasting, and plays a role in electric vehicles. Also, it's used in monitoring power quality and it's applicable to almost every aspect of the electrical utility industry [5].

DL methods can be classified as [5]: "supervised, semi-supervised, and unsupervised. Furthermore, there is one more class of DL approach named RL or DRL". Figure 3 shows the available DL architectures in the literature.

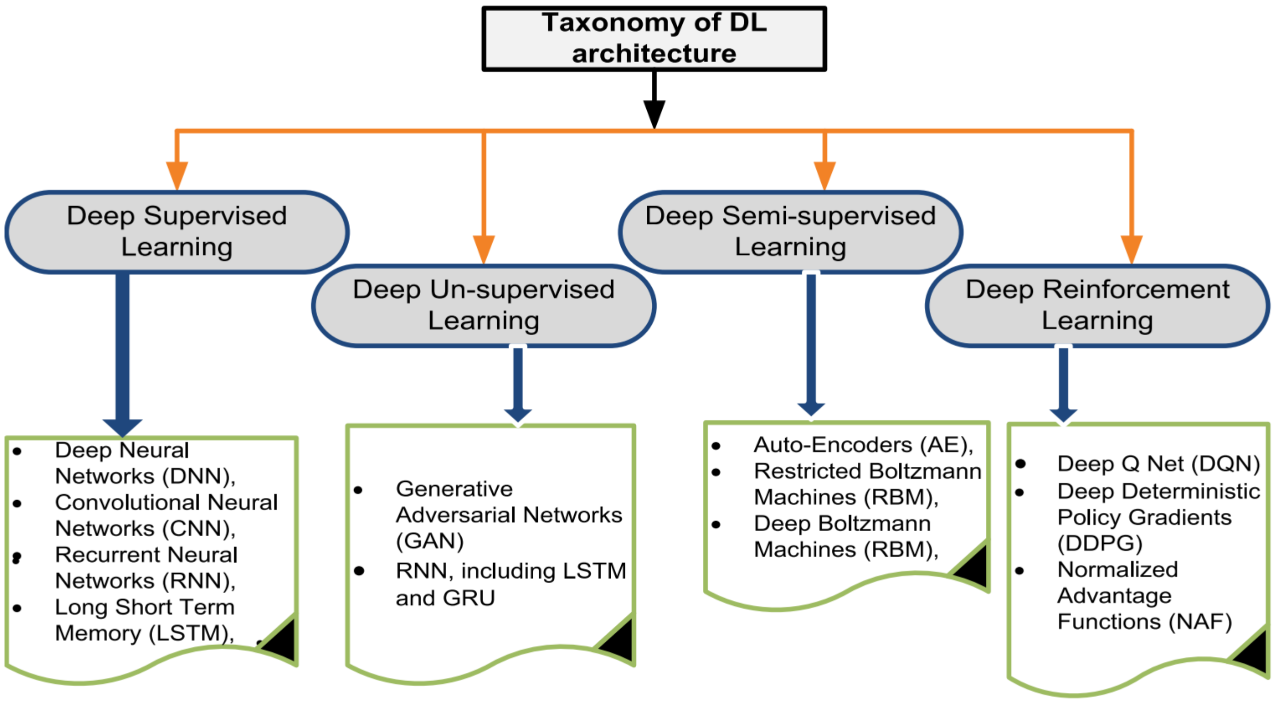

Taxonomy of DL architecture
Deep Supervised Learning
Deep Un-supervised Learning
Deep Semi-supervised Learning
Deep Reinforcement Learning
Deep Neural Networks (DNN),
Convolutional Neural Networks (CNN),
Recurrent Neural Networks (RNN),
Long Short Term Memory (LSTM),
Generative Adversarial Networks (GAN)
RNN, including LSTM and GRU
Auto-Encoders (AE),
Restricted Boltzmann Machines (RBM),
Deep Boltzmann Machines (RBM),
Deep Q Net (DQN)
Deep Deterministic Policy Gradients (DDPG)
Normalized Advantage Functions (NAF)

Figure 3. Taxonomy of DL architecture available in literature [5]

Afrasiabi et al. [25] introduced a multi-agent day-ahead microgrid energy management framework. "The objective is to minimize energy loss and operation cost. To forecast market prices, wind generation, solar generation, and load demand, a DL-based approach is designed based on a combination of convolutional neural networks (CNNs) and gated recurrent unit (GRU)". "To preserve the information privacy of agents, the alternating direction method of multipliers (ADMM) is utilized to find the optimal operating point of microgrid distributedly. To enhance the convergence performance of the distributed algorithm, an accelerated ADMM is presented based on the concept of over-relaxation". The proposed framework was tested on a realistic test system. In some cases, the convolutional neural network – gated recurrent unit (CNN-GRU) method managed to achieve at least 50% more accuracy than multiple other methods such as 2D-CNN, GRU, and long short-term memory (LSTM) [25].

1.4. Recurrent neural networks in electric power systems

RNN belongs to a category of ANNs, where the connections between nodes form a directed graph. [2]. “It models the temporal dynamic behaviors exhibited in time series data via the use of feedback connections to recall the neural states at previous time steps”. RNNs, unlike FNNs, can utilize their internal neural states to handle time series sequences. This makes them well-suited for forecasting renewable energy [2].

A typical structure of RNN is shown in Figure 4.

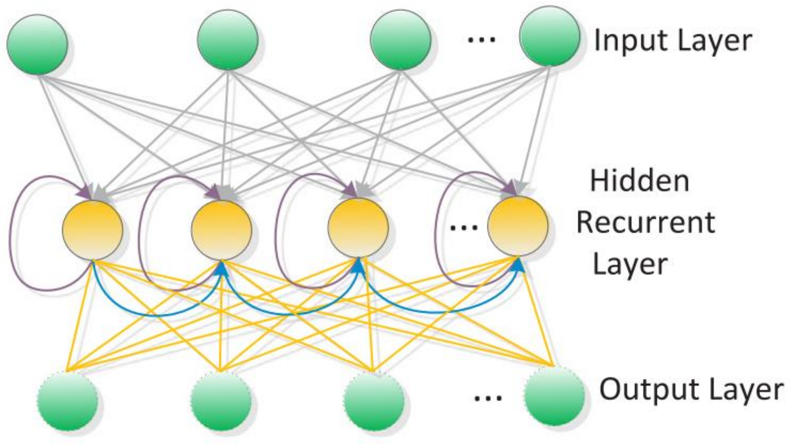


Figure 4. A typical structure of an RNN [2].

With smart meters and sensors becoming common in power systems, they produce large quantity of data, but this data is often not being used fully. Most current methods for predicting short-term power load struggle to effectively deal with this complex big data [26].

As the power system continues to improve, the importance of load forecasting keeps growing. “It is not only an important part of power system dispatching, operation, and planning, but also the foundation of economic operation and safe operation of power system. However, power load data generally have the characteristics of complexity and nonlinearity” [27]. Currently most forecasting methods are unable to simultaneously account for both the time series patterns and non-linear characteristics of data.

1.4.1. Long short-term memory

Understanding long-range dependencies in RNNs is a big challenge because it has the issues of gradient vanishing or exploding. “Gradient vanishing in RNN refers to the problems that the norm of the gradient for long-term components decreasing exponentially fast to zero, limiting the model’s ability to learn long-term temporal correlations, while the gradient exploding refers to the opposite event” [28].

In order to overcome these issues, Hochreiter and Schmidhuber [29] introduced the LSTM architecture in 1997. The LSTM architecture nowadays is widely used for forecasting load and electrical power generation.

Rafi et al. [30] proposed a new technique to forecast short-term electrical load. The method is based on integrating both CNN and LSTM networks. The results show that the used method resulted in higher precision and accuracy in short-term load forecasting. “The proposed CNN-LSTM model can deal with the long sequence time series electric load data and predict the future load demand over a considerable period. It was suggested that a highly precise load forecasting framework can be developed in the future by using GRU integrated with CNN network” [30].

Zheng et al. [31] proposed a LSTM RNN-based equivalent model to accurately represent the active distribution networks (ADNs), “motivated by the similarities between power system differential algebraic equations and the forward calculation flows of RNNs”. It reveals that “the proposed LSTM RNN-based equivalent model can be considered as one of the useful tools to represent the dynamic behaviors of ADNs. Moreover, the proposed LSTM RNN-based equivalent model can be also adopted for power stability control and voltage stability assessment in the analysis of electric power systems” [31].

Kaur et al. [32] presented a management scheme using DL for smart grid energy. They have presented an RNN-LSTM based data analytics method. “During the analysis, various external factors related to the environment are also considered namely visibility, temperature, wind speed, wind bearing, pressure, cloud cover etc. Using the proposed scheme energy consumptions patterns are predicted for testing and training data sets with minimum errors”. “Based upon these predictions DR management is performed and the gap between energy demand and supply can be minimized”. Results show that the value of root mean square error (RMSE) resulting after using the proposed scheme indicate that the proposed scheme is effective [32].

Patyn et al. [33] compared neural architectures for DR using RL to control a heat pump. The incentive was “to shift loads in a day-ahead market in order to minimize daily electricity costs. The simulation considered a multilayer perceptron (MLP), a CNN and a LSTM neural network to model the environment dynamics”. “All architectures outperform a trivial thermostat controller and shift loads successfully after 20-25 days. For this particular setup, there is no significant difference between the MLP and the LSTM, while they do outperform the CNN model. The MLP is preferred as it requires far less computation time” [33].

Wang et al. [34] suggested a day-ahead PV power forecasting model “assembled by fusing DL modeling and time correlation principles under a partial daily pattern prediction (PDPP) framework to tackle the deficiencies of conventional AI modeling methods such as overfitting problem and insufficient generalization ability to complex nonlinear modeling”. The findings of the simulation demonstrate that the suggested forecasting approach, using time correlation modification (TCM), is more accurate compared to the LSTM-RNN model [34].

1.4.2. Gated recurrent unit

Gated recurrent unit (GRU) was proposed by Cho et al. [35] “to make each recurrent unit to adaptively capture dependencies of different time scales. Similarly to the LSTM unit, the GRU has gating units that modulate the flow of information inside the unit, however, without having a separate memory cell” [36].

The LSTM and GRU are different in a couple of significant ways. One of them is how much memory content is available to other parts of the network. In LSTM, an output gate regulates this. GRU doesn't limit the access to its memory content [36].

Another difference is where the input gate (corresponding reset gate in GRU) is located. The GRU manages the flow of data from the previous activation when calculating the new one. It doesn't separately adjust how much of the new activation gets included. Instead, the update is performed using update gate [36].

Xiuyun et al. [27] proposed a short term load forecasting model by using the GRU model of DL. The simulation and forecast is made using load data of a region in China, Heilongjiang province. “Comparing the prediction results with the traditional ones, it is proved that the error of GRU model is lower and the prediction effect is better” [27].

Kuan et al. [37] proposed a multilayered self-normalizing GRU model for short-term electricity load forecasting. The model introduced a scaled exponential linear unit (SELU) activation function to the hidden states of GRU, overcoming the vanishing gradient problem. Comparison results demonstrate that the proposed multilayered self-normalizing GRU can effectively forecast the electricity loads over a long horizon [37].

Zheng et al. [38] proposed a short-term load forecasting method for residential community based on GRU neural network. Influence of temperature, humidity, rainfall, and wind speed on the load was analyzed. Numerical simulation results show that compared with LSTM and traditional RNN, GRU neural network model has better performance, convergence speed and robustness. It was proven to be a more effective residential community short-term load forecasting method [38].

Wu et al. [26] proposed a GRU-CNN hybrid neural network model which combines the GRU and CNN, aiming at improving the accuracy of short term load forecasting. “The proposed model was tested in a real-world experiment, and the mean absolute percentage error (MAPE) and the RMSE of the GRU-CNN model are the lowest among back-propagation neural network (BPNN), GRU, and CNN forecasting methods”. The proposed GRU-CNN model was able to achieve more accurate short-term load forecasting by fully use the data [26].

1.5. Temporal convolutional network

TCN was initially proposed by Lea et al. [39] “for action segmentation that learns a hierarchy of intermediate feature representations, which contrasts with the traditional low-versus high-level paradigm”. Authors claim that this model produced superior results on multiple datasets, and has the advantage of being trained faster compared to other models.

Yin et al. [40] proposed a “multi-temporal-spatial-scale method to process the load data by reducing the load data noise error and enhancing the time series characteristics. The proposed approach can learn the nonlinear feature and time series characteristics of load data simultaneously”. To forecast power consumption for a city in China for the upcoming day and week, a forecasting model was designed. This model uses historical load data from 7 days, 21 days, 99 days, and 199 days as training inputs. Compared with 22 other short-term forecasting models, the simulation results using the multi-temporal-spatial-scale TCN network produced the most accurate results under multiple forecasting performance metrics [40].

Tang et al. [41] proposed a forecasting model for predicting short-term energy consumption, utilizing TCN and temporal attention mechanism. The model was able to effectively capture the complex relationship between weather data and load. “Fuzzy c-means combined with dynamic time warping

(DTW) is developed to conduct cluster analysis on load data to classify loads with high similar characteristics into the same cluster". "Experimental results show that compared with multiple forecasting models, the method put forward can obtain lower MAPE and RMSE. Overall, the approach developed in this article effectively improves accuracy and offers stable prediction results" [41].

1.6. Feed-forward neural networks

Taylor et al. [42] presented a FNN for the hourly load forecasting. The load of the next 24 hours was forecasted using 24 independent networks. "Based on simulation results, the FNN approach has reduced the MAPE of hourly load forecast by more than 30% in comparison to the previously used conventional time series model. The FNN method also reduced the daily energy forecast MAPE that was 40% less than that from conventional time series model".

Talaat et al. [43] proposed a model for predicting electricity load mid-term to short-term, which is capable of forecasting loads for different hours and days throughout the month. It used a hybrid model, combining a multilayer FNN and the grasshopper optimization algorithm with the aim to enhance the accuracy of forecasted load. The proposed model, utilizing the combination of a multilayer FNN and the grasshopper optimization algorithm resulted in a reliable model with high effectiveness, precision and accurate results.

1.7. Limitations and gaps in the current literature

After conducting literature review, the main limitations, and gaps in literature on AI-based methods for electrical load forecasting in power systems have been found to be the following:

- Training AI or ML models requires massive amounts of training data to reach desired performance. The goal is to develop good models with a small amount of data, but this is challenging.
- Most of the current AI and ML models are "black boxes". It's difficult to understand how these systems make predictions. This can limit their use in applications with strict security and reliability requirements, like power system dispatching.
- Most models are evaluated only through simulations. To evaluate their effectiveness in real-world scenarios, more real-world applications should be implemented.
- There is a lack of comparisons between different AI-based methods. Studies could provide more insights into strengths and weaknesses of each used model.
- More discussions about computational efficiency and scalability should be performed, especially for analyzing massive systems with large amounts of data.

2. Methodology

The first step to create any forecasting model is to analyze the data.

Data used in this project consists of hourly electricity consumption and production data from a total of 176 customers connected to the Lithuanian electrical power distribution grid, from the year 2020. Number of days: 366, number of hours: 8784.

The data consists of 3 types of customers: 68 industrial, 64 commercial, and 44 residential users.

Not all users have generation capacity. Out of all the users, 14 industrial users, 5 commercial users and 21 residential users have generation capacity. All other users are consumers only.

Geographic locations of the users are given in Figure 5.

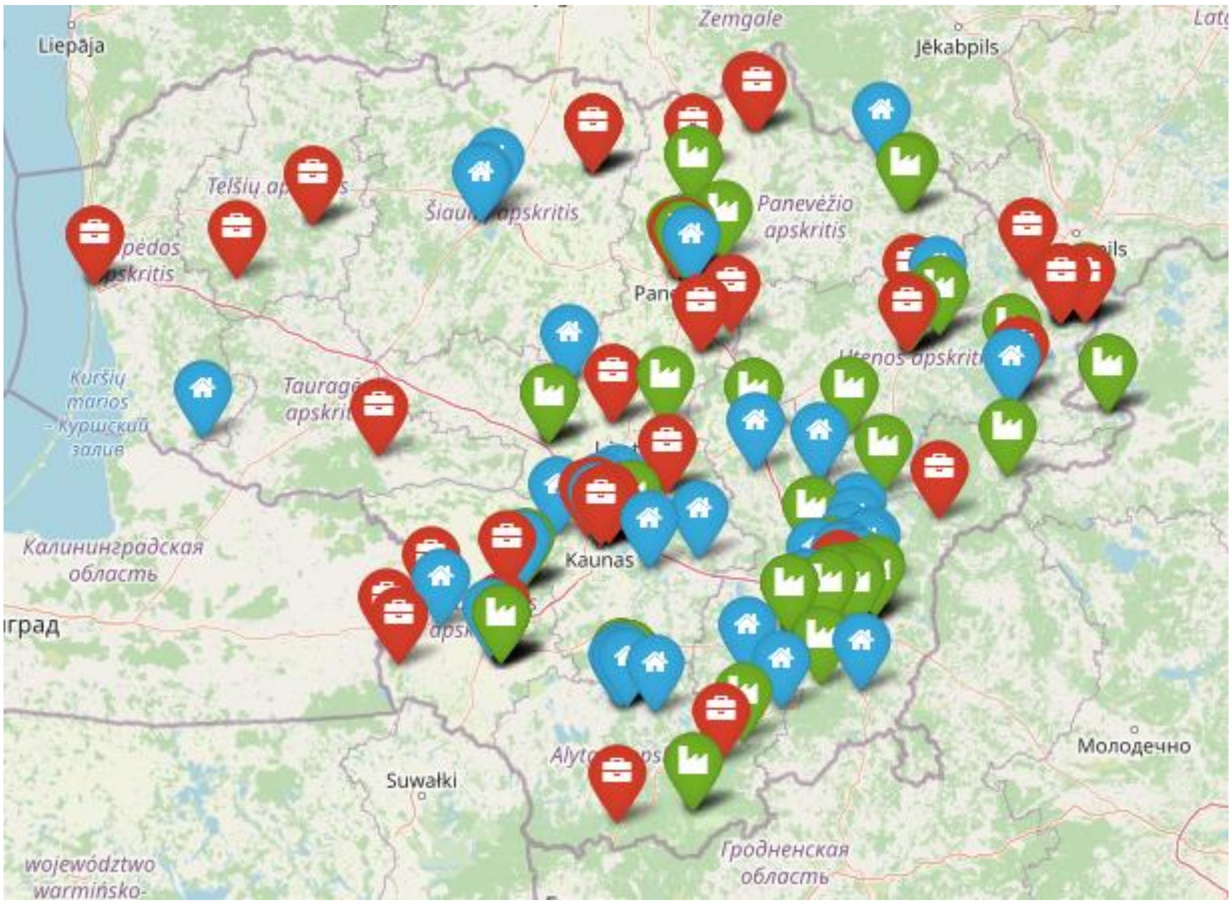


Figure 5. Map containing commercial (red), industrial (green) and residential (blue) users.

For visual representation of consumption and production plots of all users see Appendix 1.

The data is not entirely clean, some users are missing data completely for periods, and there are significant spikes in the data. Despite not having perfectly clean data, the data will be prepared in a way that allows to create forecasting models. Having clean and reliable data is very important, data influences every step, from training to prediction. Inaccurate data can lead to poor model performance.

Two different methods of calculating distance matrices will be used. A distance matrix is a matrix that contains distances taken between the elements of a dataset.

The two types of distance matrices used: Dynamic Time Warping (DTW) and Euclidean distances.

DTW calculates the optimal alignment between two given (time-dependent) sequences, the objective of DTW is to compare two sequences [44]. DTW is very suitable in situations where data may be shifted in time relative to each other. If two load graphs are similar, but shifted from each other even by an hour, the Euclidean distance will not necessarily conclude that the two graphs are similar, but DTW will conclude that they are similar.

Euclidean distance [45] is a distance between two points in a straight line in a Euclidean space. Euclidean distance matrix compares the hourly values of two time series with each other and thus describes the similarity of the two time series.

These distance matrices are chosen because they are suitable for time series data. They provide different perspectives on similarity or dissimilarity between time series.

After calculating the distance matrices, two clustering methods will be applied to gather users with similar usage patterns into smaller groups: K-means and hierarchical. Clustering helps to categorize data into smaller sets that are similar inside the cluster, and dissimilar between clusters. For each model, each cluster will be trained and tested individually.

The aim of the K-means clustering algorithm is to divide a number of points in a number of dimensions into a number of clusters so that the within-cluster sum of squares is minimized. “The algorithm seeks "local" optimal, solutions such that no movement of a point from one cluster to another will reduce the within-cluster sum of squares” [46].

In the hierarchical clustering, to group data, the elements, and their distances relative to each other are measured in order to decide which elements belong to a group. “This can be a similarity, although on many occasions a dissimilarity measurement, or a stronger distance, is used” [47].

After clustering, a simple filter will be applied to filter out big spikes in data.

After filtering, clustering will be performed again with filtered data.

After filtering, a linear regression model and two AI-based methods will be used for forecasting: TCN and FNN. A forecasting model will be created for each of the distance matrix methods, for each of the clustering methods, and for each of the forecasting methods, resulting in 12 models in total.

The first model to be applied will be the Linear Regression model. The reason for using linear regression is its simplicity and interpretability. Despite being simple, it often gives good results in many forecasting scenarios.

The second model used will be the TCN. A TCN is a type of neural network designed to handle sequence data.

The main principle of a neural network is its ability to learn and improve its predictions over time. It does this using an algorithm known as backpropagation. Backpropagation is the process of adjusting the weights of a neural network based on the error rate.

The third model used will be the FNN. FNNs are one of the simpler types of ANN. In a FNN, the information moves only forward from the input layer, passing through the hidden layers, and going to the output layer. There are no cycles or loops in FNN.

Both TCN and FNN will be created using Keras by TensorFlow [48], an open-source machine learning platform.

A model for linear regression is structured to forecast 24 hours into the future from the data of the last 24 hours.

A more complex forecasting structure was created for AI-based models. To forecast 24 hours into the future, the structure for AI-based models is based on data windowing, and is given in Figure 6.

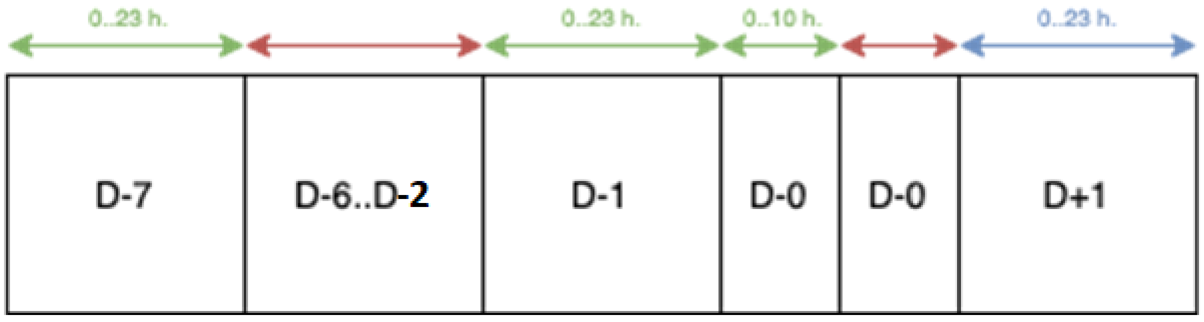


Figure 6. Forecasting model structure for AI-based models.

Here D-0 is current day, D+1 is the day after, D-1 is the day before, and D-7 is the day 1 week before the current day.

Used data:

- data of the day 1 week before (i.e., if today is Tuesday, last Tuesday’s data will be used),
- data of the day before (i.e., if today is Tuesday, last Monday’s data will be used),
- data of the current day until 11 am.

11 am was chosen because 11 am (GMT+2) is the last time to submit the offers in the Nordpool day-ahead market.

According to the structure of the mode, new datasets will be created for each of the 4 clustering methods used, and each of the forecasting methods, resulting in 8 datasets. Each dataset contains training data, and data to be predicted.

All trained models utilize k-fold cross-validation. Cross-validation is a method used to evaluate the model's performance on an independent data set. It works by splitting dataset into multiple pieces, using these different pieces of data for training the model and for testing.

In this project, a 4-fold cross-validation will be used. This means the data will be divided into four parts, with each part being used as a test set while the remaining parts are used for training, rotating until each part has been used as a test set.

The goal of cross-validation is to avoid overfitting, which occurs when the model is too complex and captures noise in the data together with the pattern.

Lastly, to evaluate the performance of each model, various metrics will be compared. These include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Square Error (RMSE), Coefficient of Determination (R2), and Adjusted R2.

These metrics offer different insights into the performance of the models. MAE, MSE, and RMSE calculates the discrepancy between predicted and actual values. R2 and Adjusted R2 calculates the explanatory capabilities of the models.

3. Application of AI-based models for forecasting

To apply AI-based models for forecasting, steps described in the methodology will be performed.

3.1. Distance matrices

To process the data, distance matrices were created from the available time series for each data point (8784 x 176). Two methods of constructing distance matrices were used: Euclidean distance, and DTW. Before calculating distance matrices, data normalization was done by dividing each value by the maximum value of the dataset.

Resulting distance matrices show how similar two time series are to each other. Having a total of 176 users means having a distance matrix of 176 x 176.

3.2. Clustering

After having the distance matrices calculated by both methods, the data is clustered by two different methods: K-mean and hierarchal.

After specifying the number of clusters at the beginning, random points are selected, and the centers of the primary clusters are changed through iterations until the solution converges and the clusters with the smallest within-cluster sum of squares are found.

To determine the optimal number of clusters for K-means clustering, the elbow method is commonly used. The elbow method works by calculating the sum of the square distance between points in a cluster. Naturally, the more clusters you have, the smaller the sum of the square distance between points. Optimal number of clusters is chosen at a point where more clusters no longer give significant improvement.

Visual representation is also used to visually distinguish the distances between clusters. It groups more similar points closer to each other. Just like for K-means clustering, hierarchal clustering utilizes visualization, called a dendrogram. The height of each line represents the distance between groups of points.

3.2.1. Clustering results

The clustering results are obtained after applying methods specified in the previous chapter and are given in Figures 7-10. Because clustering was done using normalized data, clusters contain a mix of industrial, commercial, and residential users.

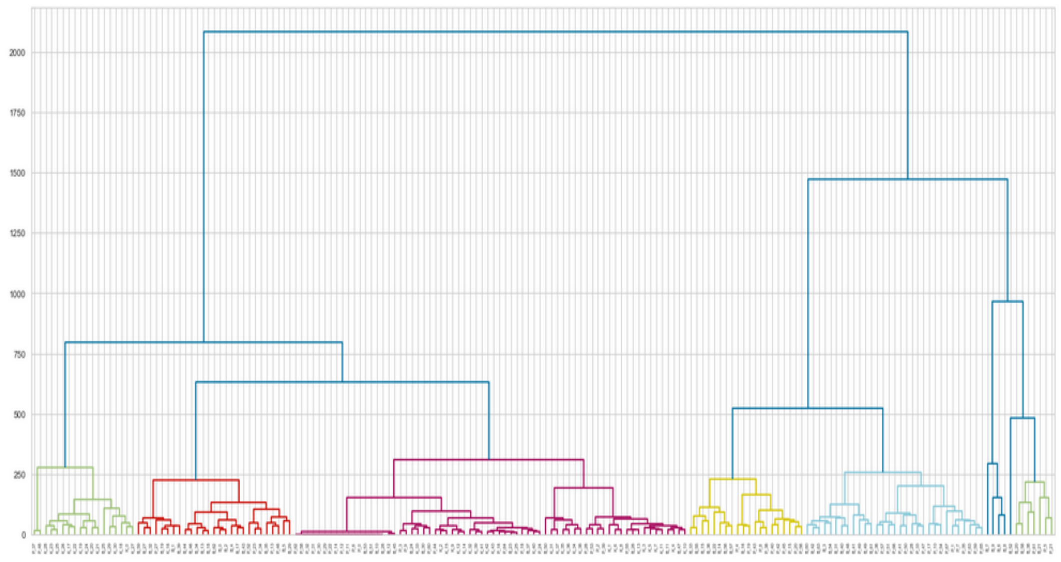


Figure 7. Euclidean distance matrix, Hierarchal clustering dendrogram.

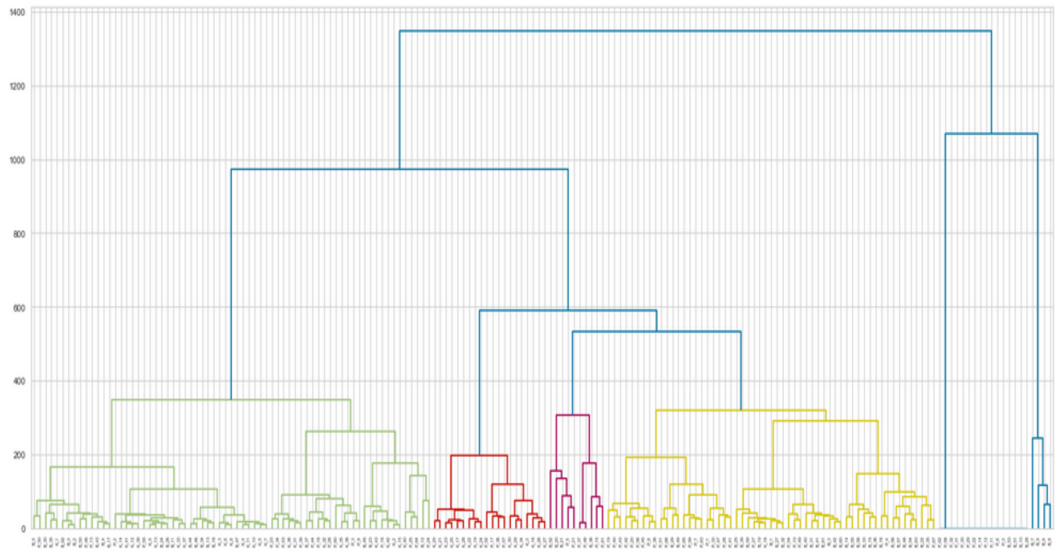


Figure 8. DTW distance matrix, Hierarchal clustering dendrogram.

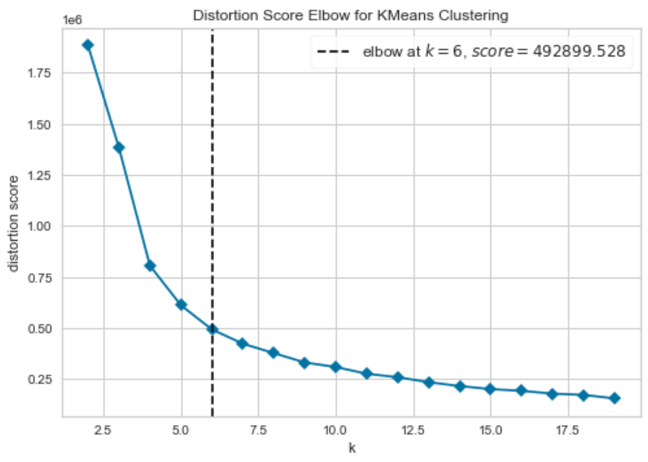


Figure 9. DTW distance matrix, K-Means clustering elbow plot.

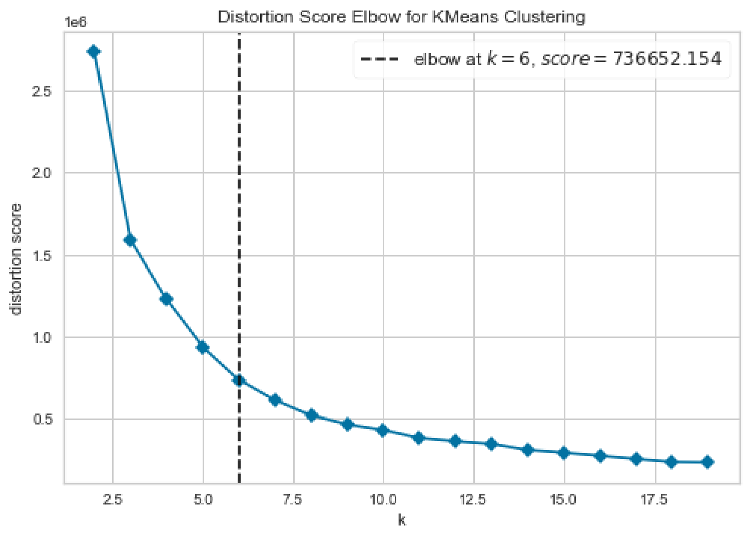


Figure 10. Euclidean distance matrix, K-Means clustering elbow plot.

In the DTW distance matrix with Hierarchal clustering, 7 clusters have been chosen as the optimal solution. In other cases, 6 clusters are proposed and were chosen as the most optimal solution.

Since the main purpose of clustering is to divide similar users into several similar groups, these clusters will be used throughout further analysis to analyze similar users.

Further on, visual results of clustering are presented. The plots show the average load values of the users belonging to the cluster. Each cluster is represented by a different colour. Power output is given on the vertical axis in relative, normalized units.

To make the plots more readable, the horizontal axis of average power of each cluster plot shows two weeks of the year, starting from hour 2000.

3.2.1.1. DTW Hierarchal

DTW distance matrix with hierarchal clustering is divided into 6 clusters. Similarities between users in each cluster can be seen by similarities the plots in Figure 11.

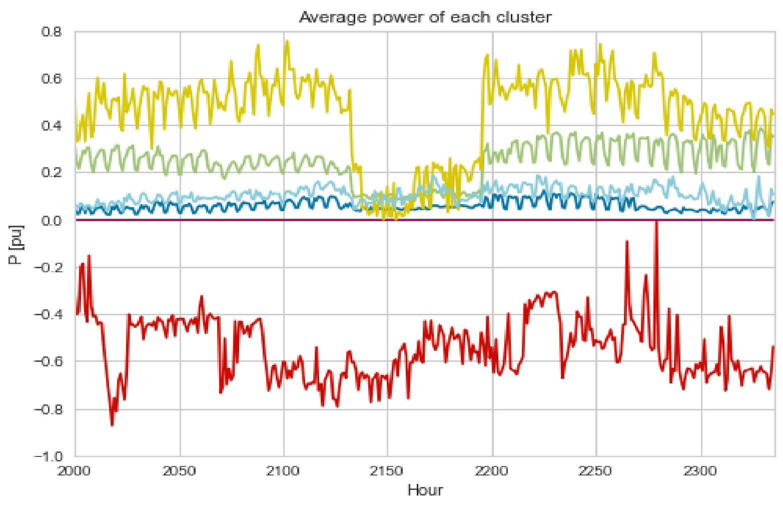


Figure 11. DTW Hierarchal clustering. Average power [pu] of each cluster, hour 2000-2336.

This clustering method manages to separate incorrect data where all values are 0, this cluster is plotted in a straight horizontal line. Users in cluster indicated by red color produce more power than they consume.

In Figure 12, the graphs show all the customers belonging to the cluster. Some power output similarities can be seen clearly.

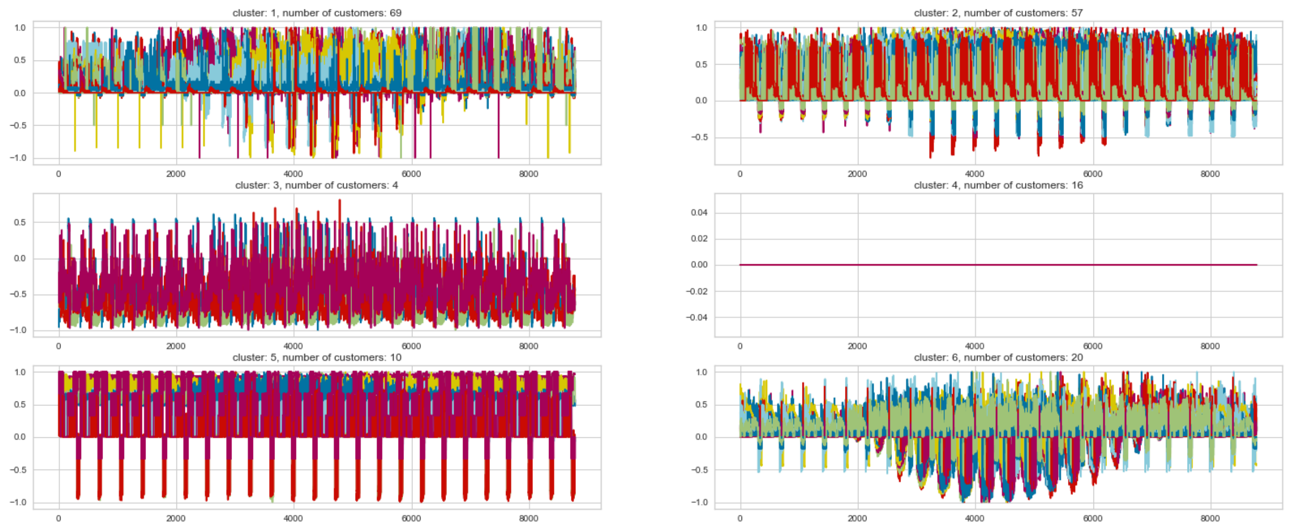


Figure 12. DTW Hierarchal clustering. Power output of all clusters.

3.2.1.2. DTW K-means

DTW distance matrix with K-means clustering is also divided into 6 clusters. This clustering method also manages to separate incorrect data where all values are 0, though it also includes one more user with sporadic power output statistics, which can be seen in Figure 13.

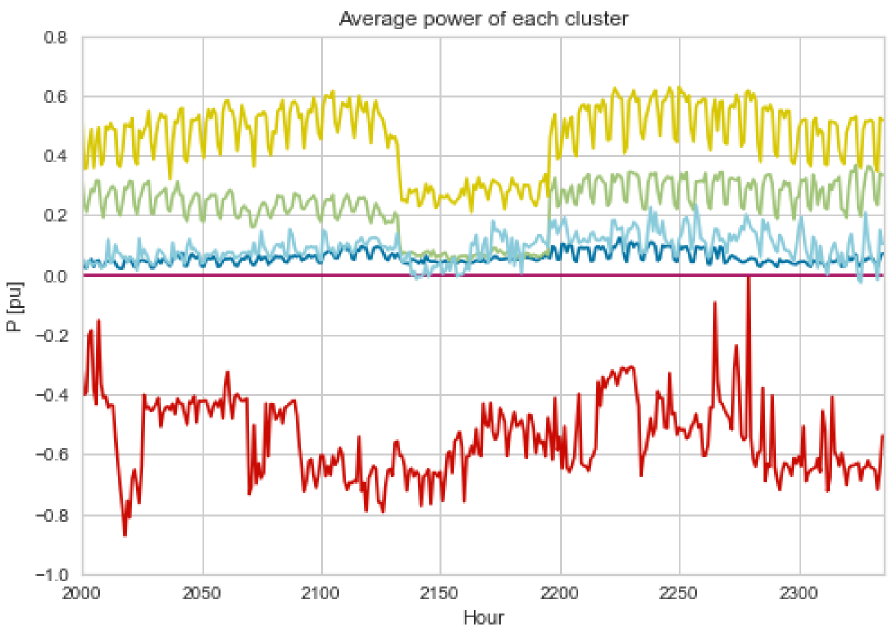


Figure 13. DTW k-means clustering. Average power [pu] of each cluster, hour 2000-2336.

In Figure 14, the graphs show all the customers belonging to the cluster. Some power output similarities can be seen clearly.

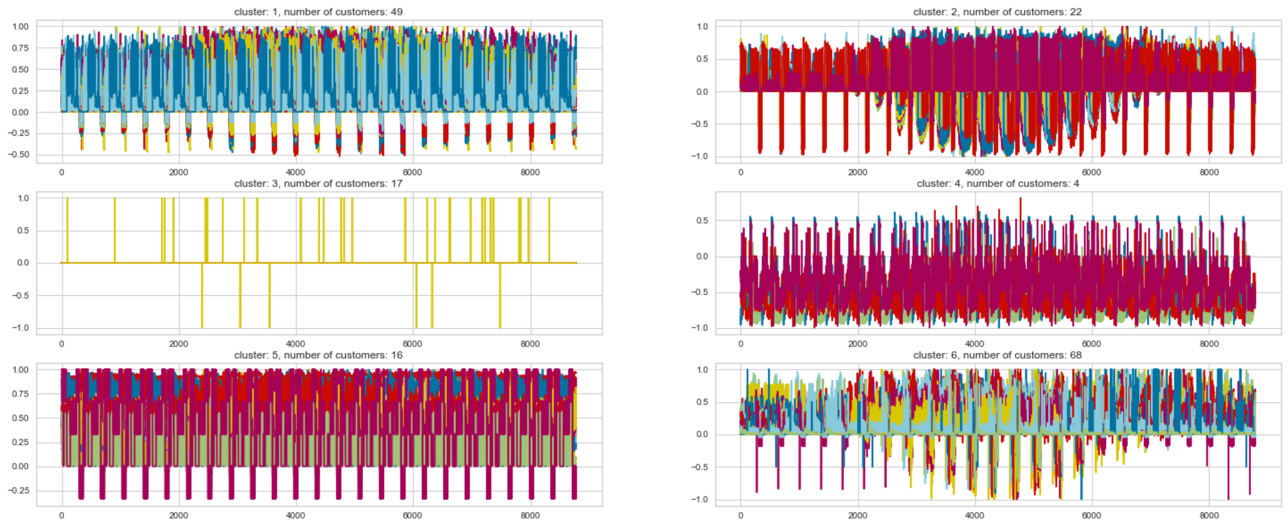


Figure 14. DTW k-means clustering. Power output of all clusters.

3.2.1.3. Euclidean Hierarchal

Euclidean distance matrix with hierarchal clustering is divided into 7 clusters. Unlike the previous two methods, this clustering method does not manage to separate incorrect data where all values are 0. Average power of each cluster can be seen in Figure 15.

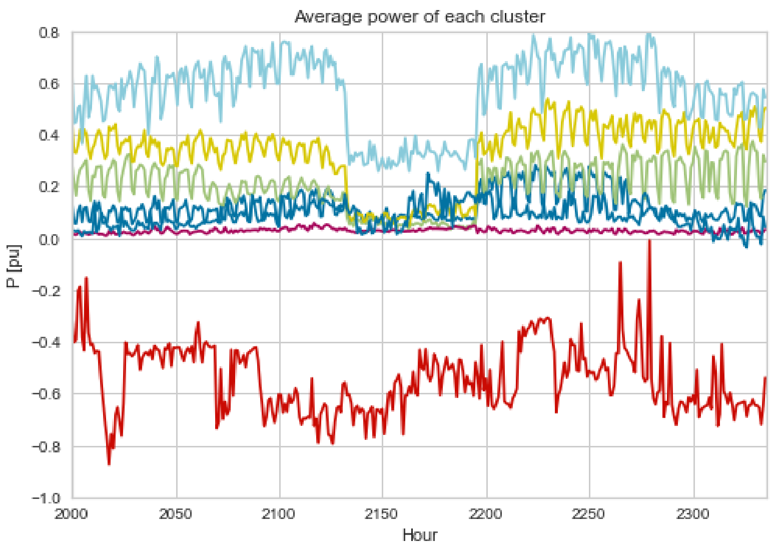


Figure 15. Euclidean Hierarchal clustering. Average power [pu] of each cluster, hour 2000-2336.

In Figure 16, the graphs show all the customers belonging to the cluster. Some power output similarities can be seen clearly. Customers with constant load of 0 have not been separated into the same cluster.

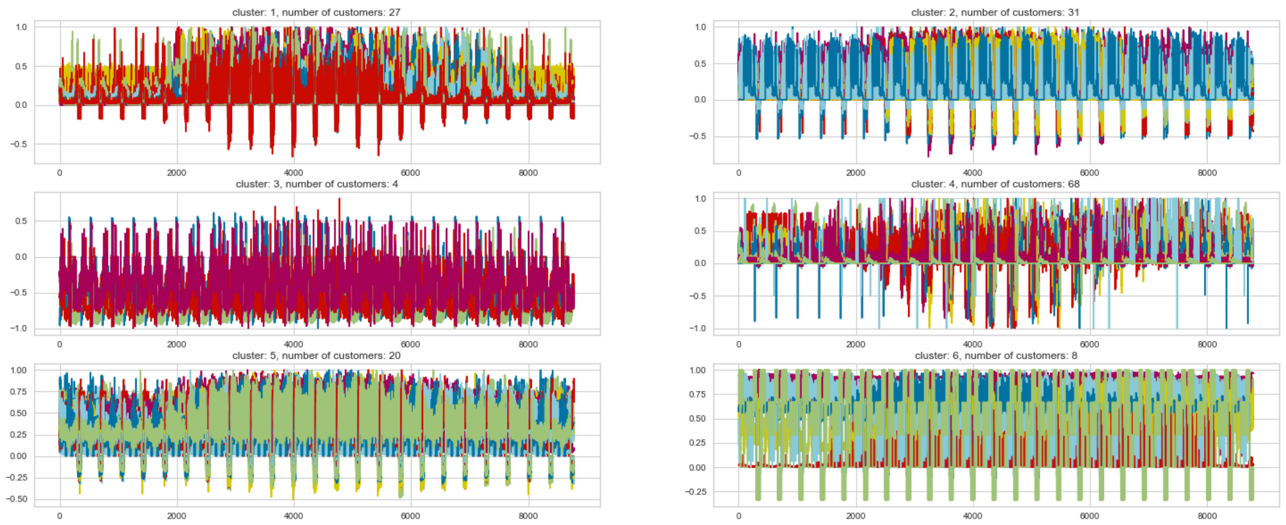


Figure 16. Euclidean Hierarchal clustering. Power output of all clusters.

3.2.1.4. Euclidean K-means

Euclidean distance matrix with K-means clustering is divided into 6 clusters. Unlike the first two methods, this clustering method also does not manage to separate incorrect data into one cluster where all values are 0. Average power of each cluster can be seen in Figure 17.

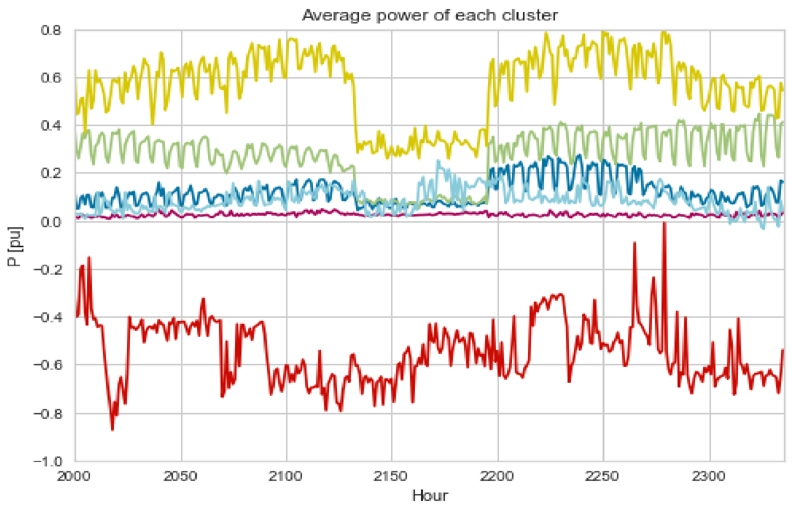


Figure 17. Euclidean K-means clustering. Average power [pu] of each cluster, hour 2000-2336.

In Figure 18, the graphs show all the customers belonging to the cluster. Some power output similarities can be seen clearly. Customers with constant load of 0 have not been separated into one cluster.

The Euclidean distance matrix with both K-means and hierarchal clustering methods fails to distinguish users with constant consumption of 0, i.e., bad data. Total number of customers with 0 consumption data: 16.

Clustering can help to distinguish bad data without having to manually search for it.

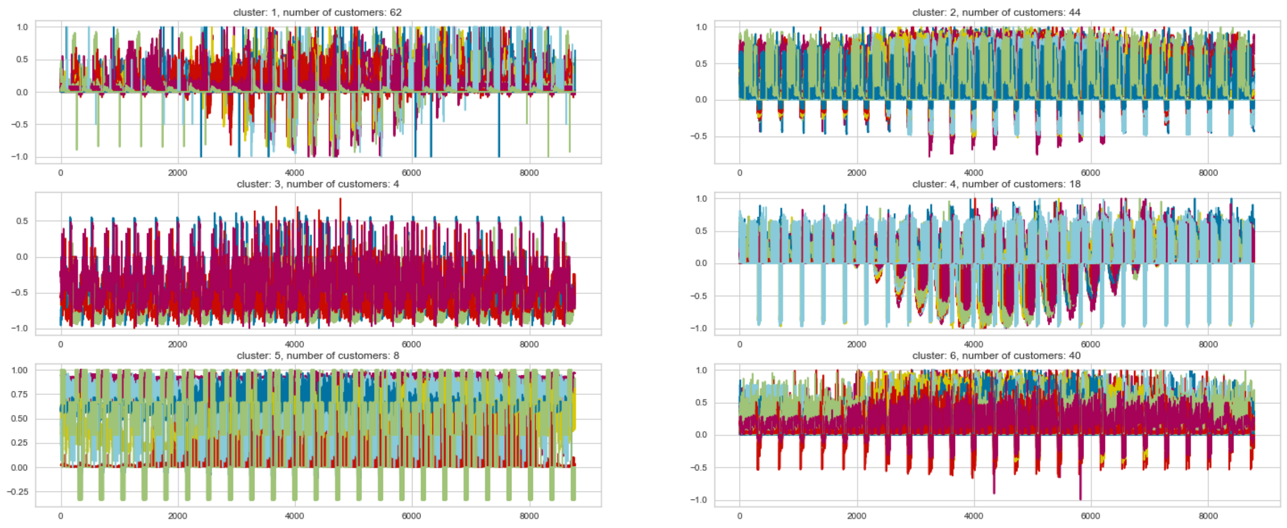


Figure 18. Euclidean K-means clustering. Power output of all clusters.

The resulting plots clearly indicate how consumers with different power output characteristics are clustered together.

3.2.2. Filtering

It has been found that some users have very irregular consumption behaviors. For creating a forecasting model, this is not good.

For example, in Figure 19 a plot of normalized load of a residential user ID No. 23 is given.

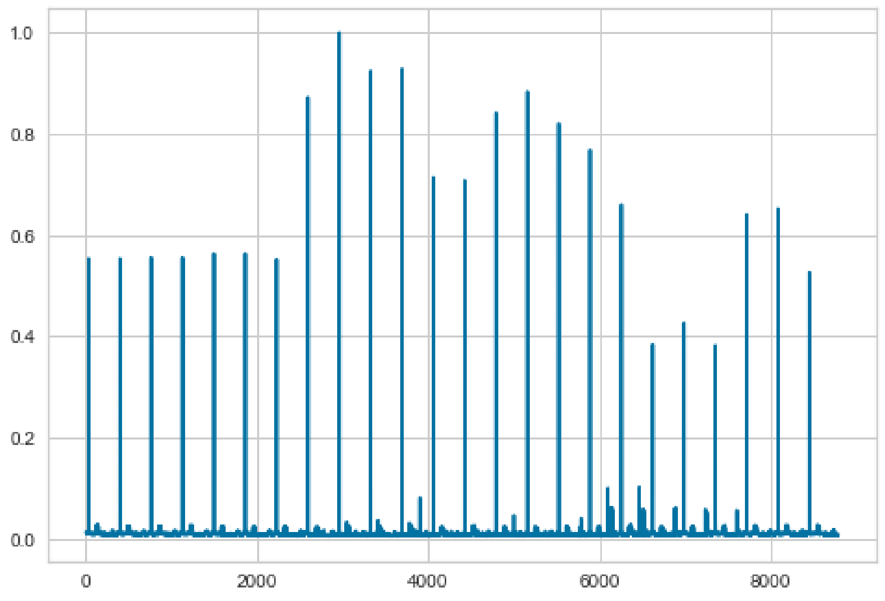


Figure 19. Plot of normalized load of residential user No. 23.

In the graph 24 peaks are visible. The reason for that is unknown, but this may indicate that load data was entered every 2 weeks throughout the year. It was decided to create a filter that would filter out these inconsistencies.

The filter was designed to check each value of the time timeseries and compare it to 1 value before it and 1 value after it. It is achieved by using Formulas (1) and (2):

If *n*/*n*-1 > *x* and *n*/*n*+1 > *x*, then:

$$n = ((n-1)+(n+1))/2, \tag{1}$$

If: *n* – value of the current timestep; *n*-1 – value one timestep before n; *n*+1 – value one timestep after *n*; *x* – filter constant to filter out values over *x*.

If *n*/*n*-1 < -*x* and *n*/*n*+1 < -*x*, then:

$$n = ((n-1)+(n+1))/2, \tag{2}$$

If: *n* – value of the current timestep; *n*-1 – value one timestep before *n*; *n*+1 – value one timestep after *n*; *x* – filter constant to filter out values over *x*.

This filter was applied to each timestep of each user, with filter constant *x* = 0.5.

The resulting plot of residential user ID No. 23 is given in Figure 20. Peaks over 0.5 pu have been removed.

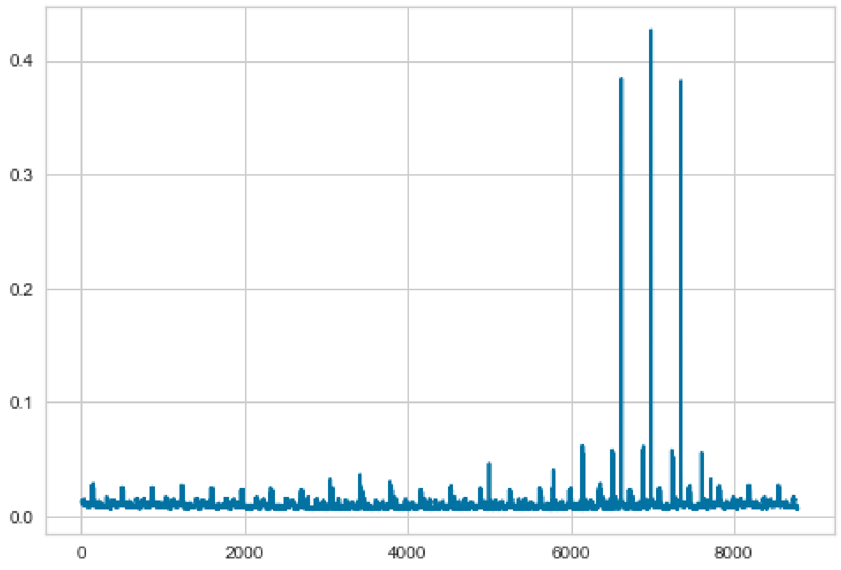


Figure 20. Plot of normalized load of residential user No. 23 after filtering.

3.2.3. Clustering results after filtering

The following Figures 21-25 show that the largest peaks have been removed:

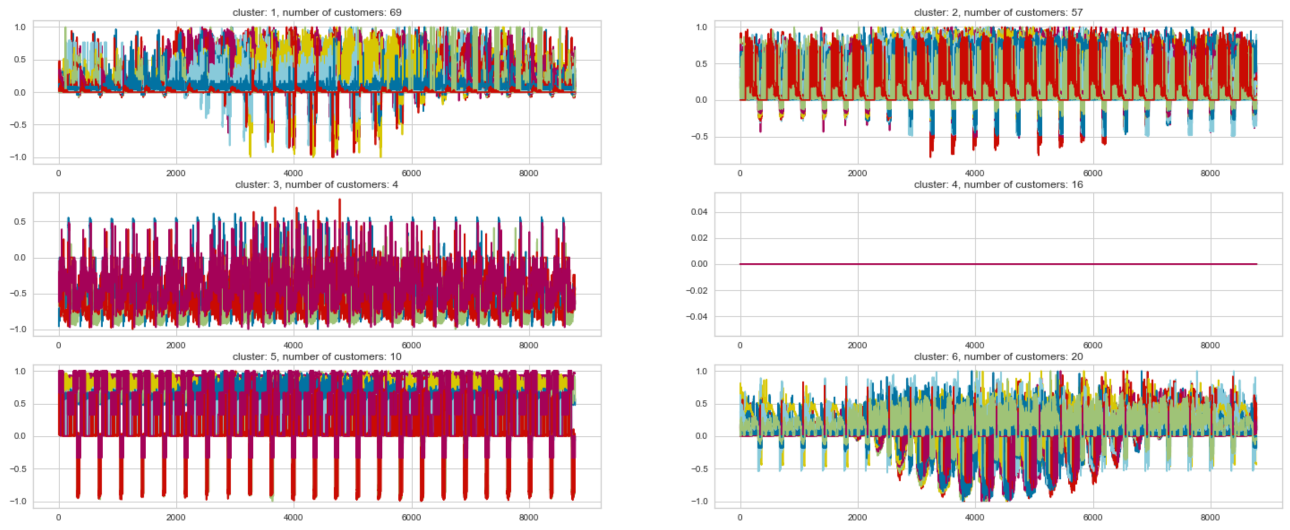


Figure 21. DTW Hierarchal clustering. Power output of all clusters after filtering.

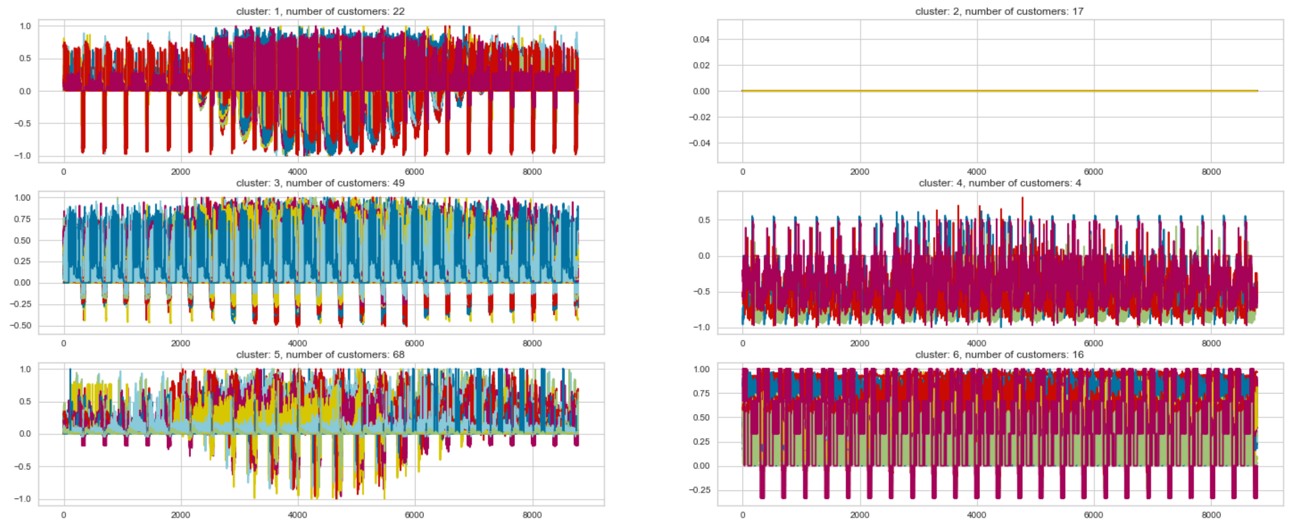


Figure 22. DTW K-means clustering. Power output of all clusters after filtering.

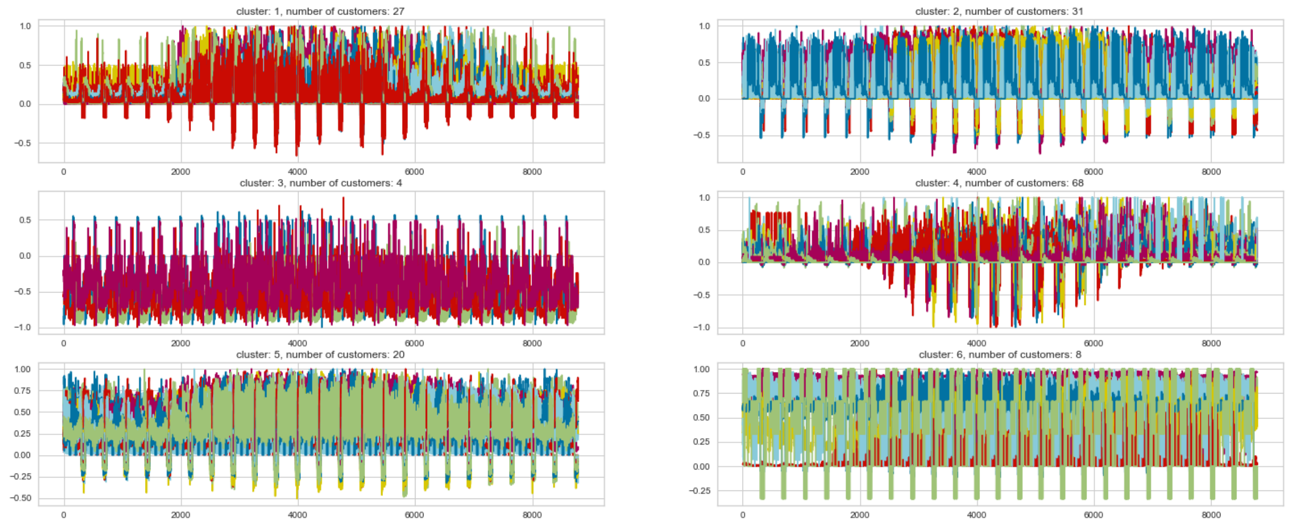


Figure 23. Euclidean Hierarchal clustering. Power output of all clusters after filtering.

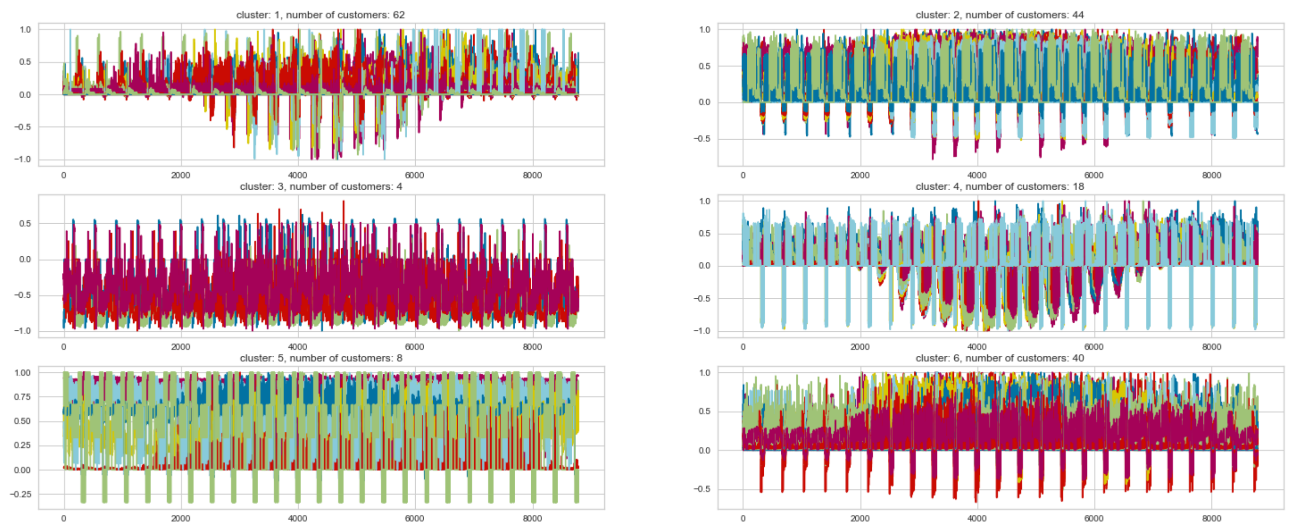


Figure 24. Euclidean K-means clustering. Power output of all clusters after filtering.

After applying filtering, some very irregular consumption behaviors have been removed.

3.3. Forecasting model

A linear regression model and two AI-based methods were used for forecasting: TCN and FNN.

A forecasting model is created according to the model structure indicated in the methodology for each of the distance matrix methods, for each of the clustering methods, and for each of the forecasting methods, resulting in 12 models in total.

3.3.1. Linear regression model

Linear regression is a basic statistical model which fits a linear equation to observed data [49]. The equation of a linear model is according to Formula (3):

$$Y = aX + b, \tag{3}$$

If: Y – dependent variable (in this case, power output); X – independent variable (time); a – slope of the line; b – y-intercept.

When having more than one independent variable, the Formula (4) of multiple linear regression is used:

$$Y = b0 + b1X1 + b2X2 + .. + bn * Xn + \varepsilon, \tag{4}$$

If: Y – dependent variable; $X1$, $X2$,.., Xn are the independent variables; $b0$, $b1$,.., bn are the coefficients of the independent variables; ε – error term.

$b0$, $b1$,.., bn coefficients determine how much each X contributes to the value of Y. ε is a constant for anything we cannot account for.

The goal of linear regression is to find the best estimates for the coefficients that minimize the differences between the predicted Y values and the actual Y values. A common method to estimate these coefficients is the method of least squares. It minimizes the sum of the squared residuals according to Formula (5):

$$S = \Sigma(Yi - (b0 + b1X1i + b2X2i + .. + bn * Xni))^2, \tag{5}$$

If: Yi – the actual value of Y for observation i; $(b0 + b1X1i + b2X2i + .. + bn * Xni)$ – the predicted value of Y for observation i.

Linear regression model was applied to forecast 24 hours into the future from the last 24 hours. 4 different clustering methods were trained independently. Each cluster was trained and tested individually.

Before plotting the average model predictions, clusters containing values with only 0s were removed.

Plots provided in Figures 25-28 represent average results of actual and predicted data for each method. To make the results easier to read, the horizontal axis shows the first 72 hours of the trained model. Vertical axis represents average load (positive values) or generation (negative values) of the users in the cluster. Orange curve represents the actual values, blue curve represents the predicted values.

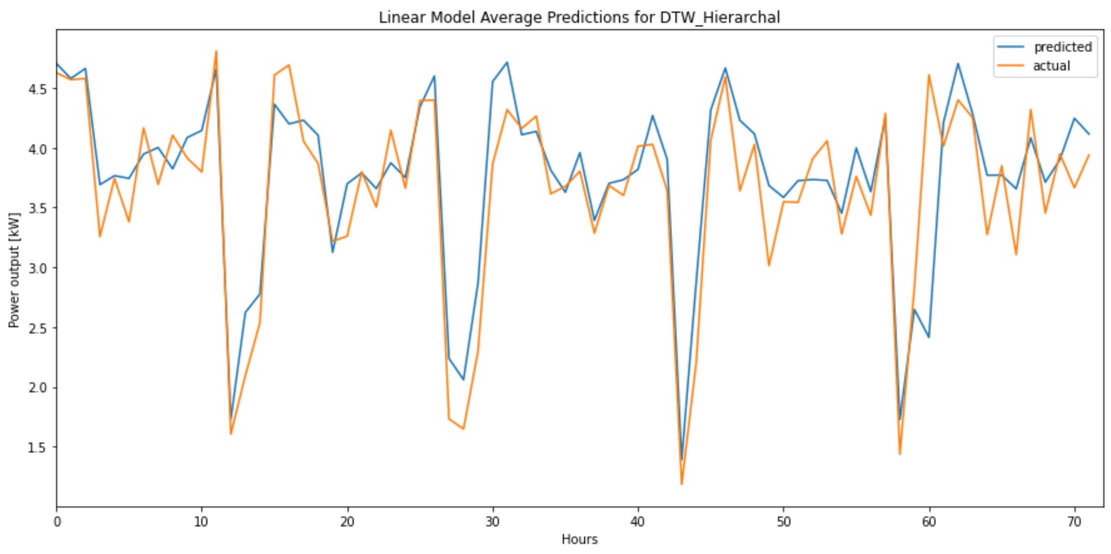


Figure 25. Linear DTW Hierarchal actual and predicted power output.

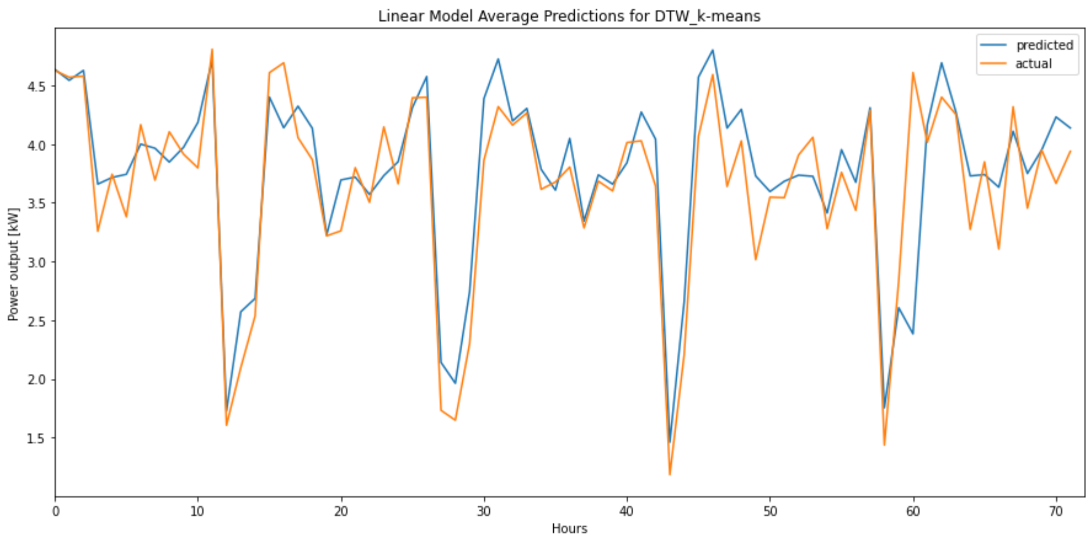


Figure 26. Linear DTW K-means actual and predicted power output.

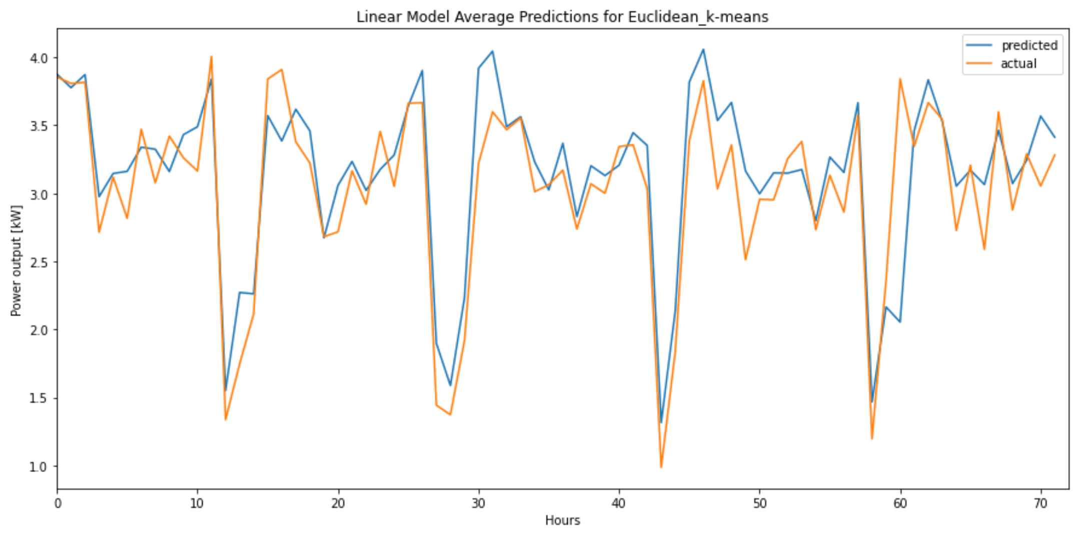


Figure 27. Linear Euclidean K-means actual and predicted power output.

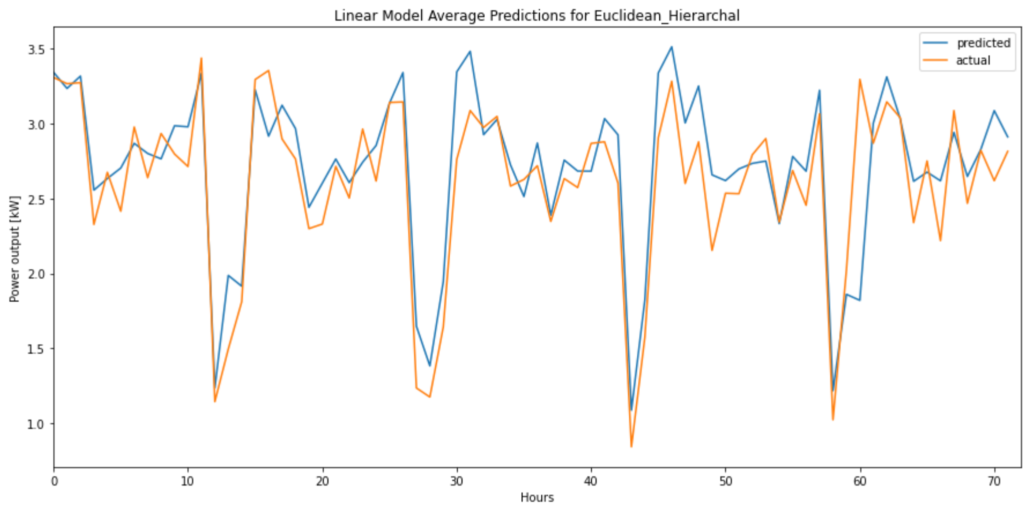


Figure 28. Linear Euclidean Hierarchal actual and predicted power output.

In Figures 25-28, it can be seen that a simple linear regression model performs quite well.

3.3.2. Temporal convolutional network models

TCN has the ability for each output to be computed as a function of previous inputs within a fixed-size window. When applied to an input sequence x[t], the output sequence y[t] is calculated according to Formula (6) and is applied at each time step in the sequence:

$$y[t] = \sum_{i=1}^{k} (f(i) * x[t - i + 1]), \tag{6}$$

If: $y[t]$ – the output at time t; $x[t - i + 1]$ – the input at time t-i+1; $f(i)$ – the weight (learned by the model) for the i-th input; k – the size of the window.

For training TCN models, 4 different clustering methods were trained independently. Each cluster was trained and tested individually.

The TCN model contains a densely-connected neural network layer of 1000 neurons. The activation function used is Rectified Linear Unit (ReLU), a popular activation function in neural networks.

Before plotting the average model predictions, clusters containing values of only 0s were removed.

Plots provided in Figures 29-32 represent average results of actual and predicted data for each method. To make the results easier to read, the horizontal axis shows the first 72 hours of the trained model. Vertical axis represents average load (positive values) or generation (negative values) of the users in the cluster. Orange curve represents the actual values, blue curve represents the predicted values.

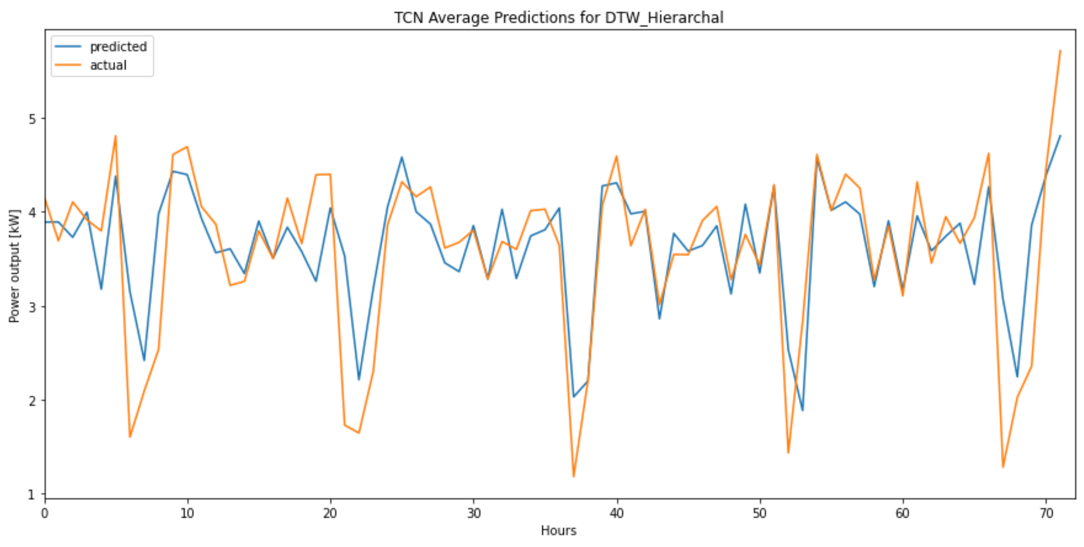


Figure 29. TCN DTW Hierarchal actual and predicted power output

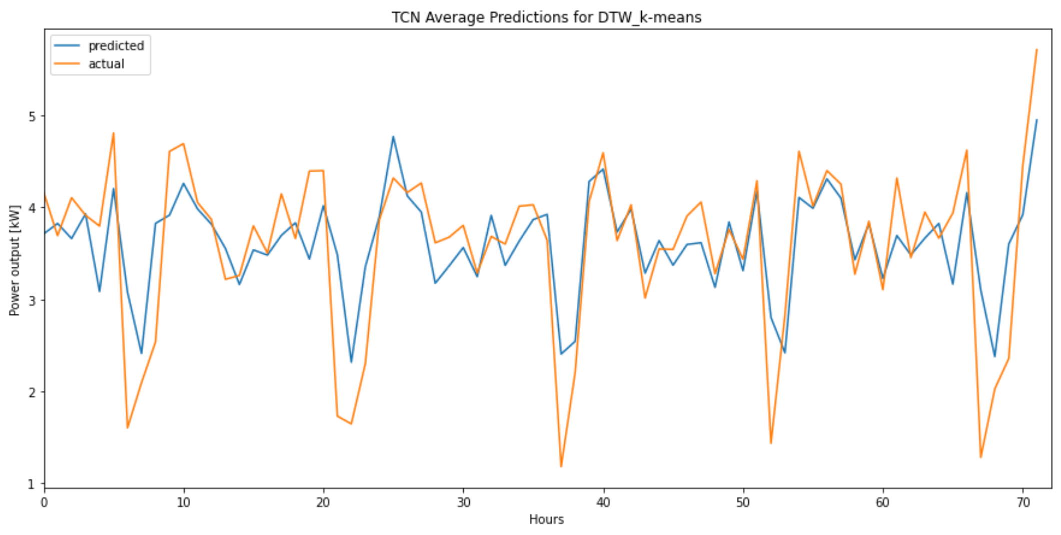


Figure 30. TCN DTW K-means actual and predicted power output.

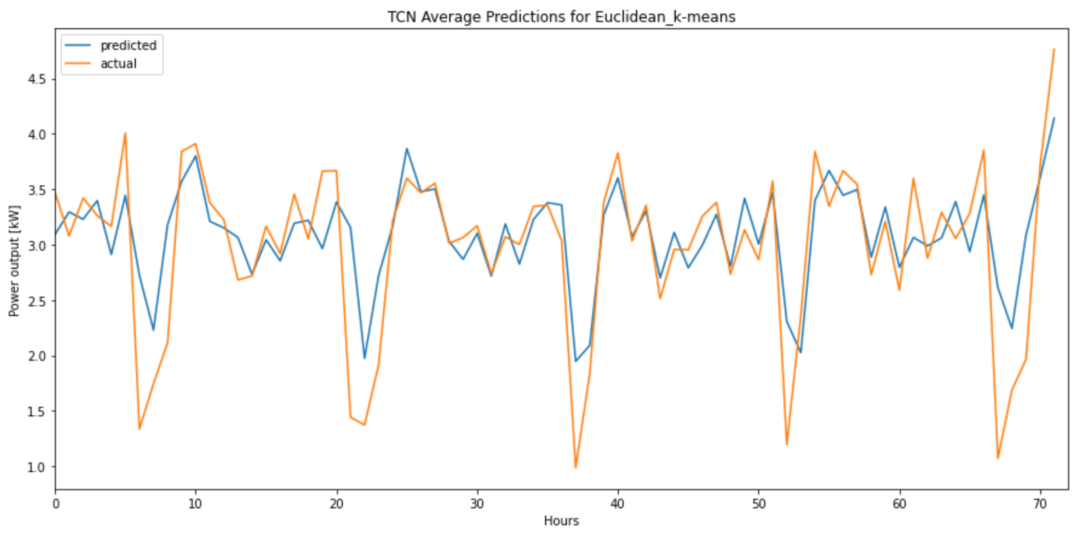


Figure 31. TCN Euclidean K-means actual and predicted power output.

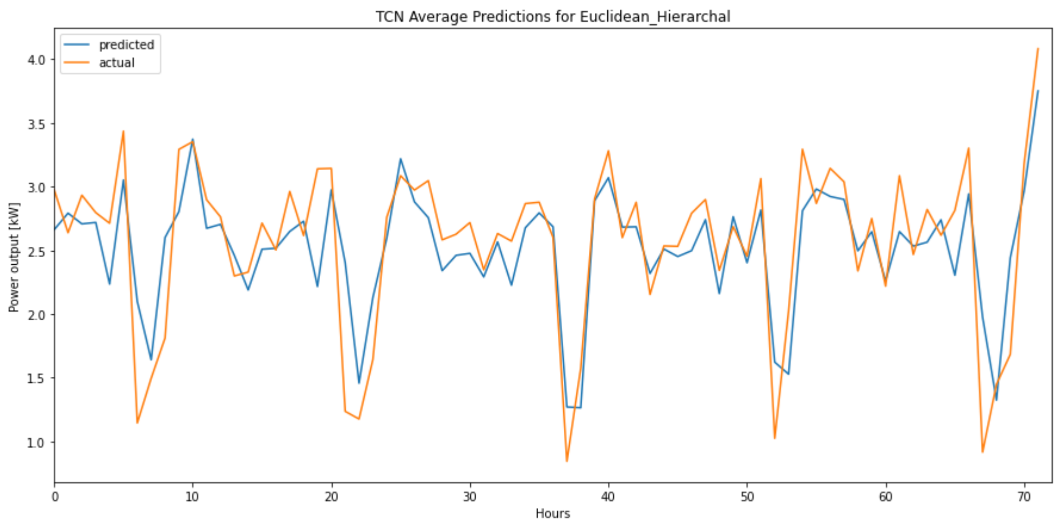


Figure 32. TCN Euclidean Hierarchal actual and predicted power output.

In Figures 29-32, it can be seen that predicted values of all TCN models are very similar. For model loss plots see Appendix 2. Model loss plots indicate that as the number of epochs increases, training and testing loss keeps decreasing. With each epoch the rate of change of loss gets smaller.

3.3.3. Feed-forward neural network models

In FNNs, connections do not form a cycle. The information moves from the input layer, through the hidden layers, to the output layer. The FNN consists of several layers of nodes or neurons, including the input layer, one or more hidden layers, and the output layer.

The FNN consists of several layers of neurons, including the input layer, hidden layers, and the output layer. Each neuron in a layer is connected with all neurons in the next layer.

For a simple FNN with 1 hidden layer, the output of the hidden layer will be calculated according to Formula (7):

$$h = f(W * x + b), \tag{7}$$

If: $W * x$ – the value of the weights and input; b – bias vector; $f()$ – the activation function (in this case ReLU).

The output layer also has its weights, bias, and can have a different activation function ($g()$). The final output of the FNN can be calculated as Formula (8):

$$o = g(V * h + c), \tag{8}$$

If: o – the final output; V – weights; h – output of the hidden layer; c – bias; $g()$ – activation function.

During model training, weights and biases are being adjusted by defining a loss function and using an optimization algorithm like gradient descent. Adjusting weights and biases to minimize the loss function is done through backpropagation.

The basic formula used in backpropagation to adjust the weights using gradient descent is given in Formula (9):

$$W_{new} = W_{old} - \eta * \frac{\partial_{Loss}}{\partial W_{old}}, \tag{9}$$

If: W_{new} – the updated weight; W_{old} – the old weight; η – learning rate; ∂_{Loss} – partial derivative of the loss; ∂W_{old} – partial derivative of the old weight.

The process is repeated until the loss function can no longer be decreased significantly.

In the used model, FNN includes an input layer of 59 neurons, hidden layers of 3000 and 1000 neurons, and an output layer of 24 neurons. Just like for the TCN, 4 different clustering methods were trained independently. Each cluster was trained and tested individually.

Before plotting the average model predictions, clusters containing only 0's were removed.

Plots provided in Figures 33-36 represent average results of actual and predicted data for each method. To make the results easier to read, the horizontal axis shows the first 72 hours of the trained model. Vertical axis represents average load (positive values) or generation (negative values) of the users in the cluster. Orange curve represents the actual values, blue curve represents the predicted values.

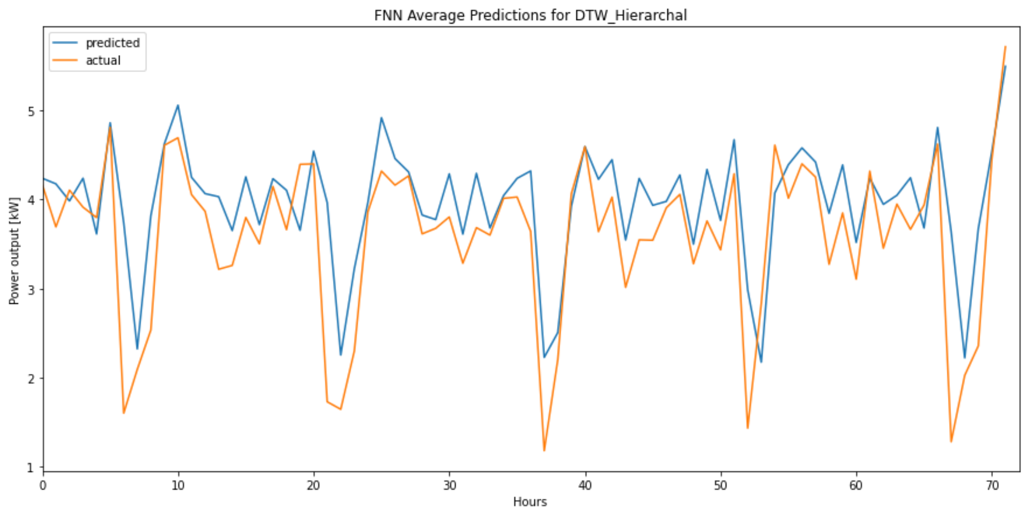


Figure 33. FNN DTW Hierarchal actual and predicted power output.

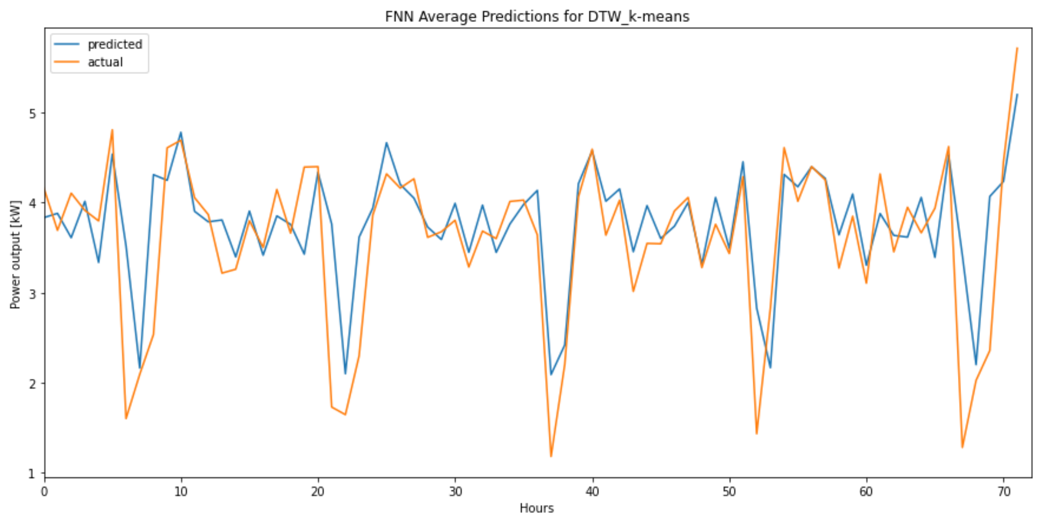


Figure 34. FNN DTW K-means actual and predicted power output.

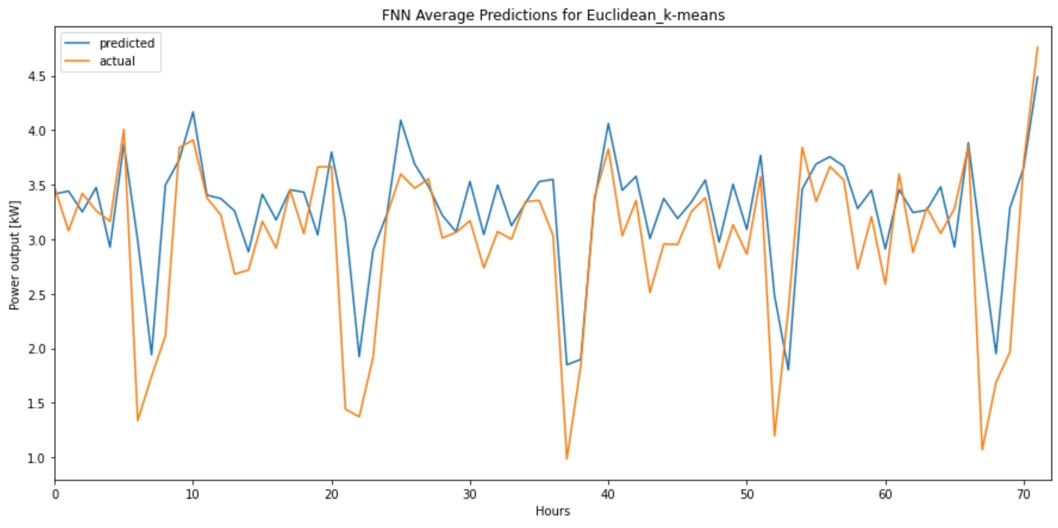


Figure 35. FNN Euclidean K-means actual and predicted power output.

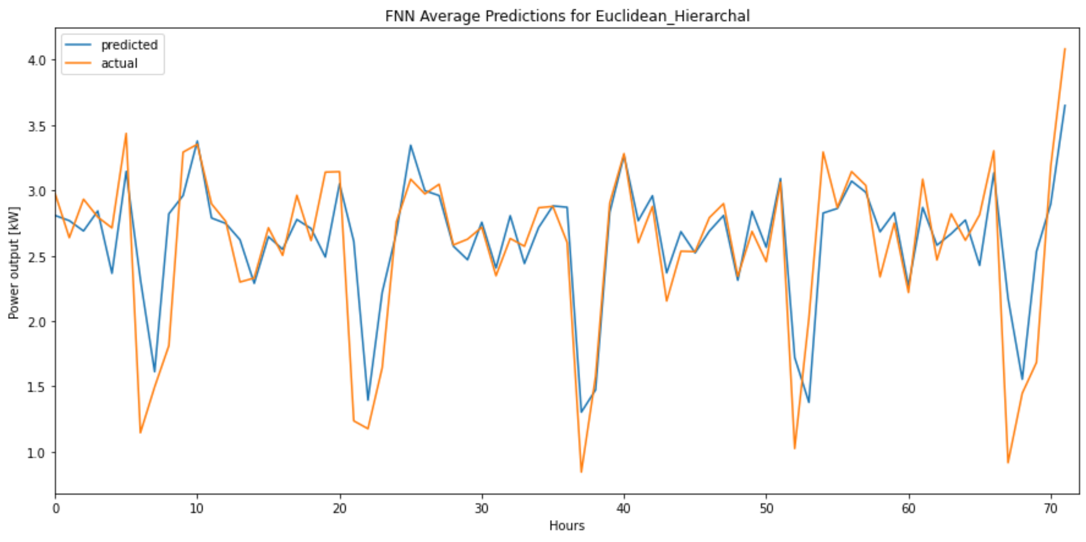


Figure 36. FNN Euclidean Hierarchal actual and predicted power output.

In Figures 33-36, it can be seen that predicted values of all FNN models are very similar, just like in the linear and TCN models. For model loss plots see Appendix 2.

3.4. Evaluation metrics

Various evaluation metrics are commonly used to evaluate performance of ML and AI-based forecasting models.

Metrics commonly used to assess performance of the model:

– Mean Absolute Error (MAE),
– Mean Squared Error (MSE),
– Root Mean Square Error (RMSE),
– Coefficient of Determination (R2),
– Adjusted R2.

For MAE, MSE, and RMSE, lower values indicate better performance. For R2 and Adjusted R2 higher values indicate better performance.

MAE is calculated according to Formula (10):

$$\mathrm{MAE} = (1/n) \sum |Y_{actual} - Y_{pred}|, \tag{10}$$

If: n – the number of observations; Y_{actual} – the actual value; Y_{pred} – the predicted value.

MSE is calculated according to Formula (11):

$$\mathrm{MSE} = (1/n) \sum (Y_{actual} - Y_{pred})^2, \tag{11}$$

If: n – the number of observations; Y_{actual} – the actual value; Y_{pred} – the predicted value.

RMSE is calculated according to Formula (12):

$$\mathrm{RMSE} = \sqrt{\mathrm{MSE}}, \tag{12}$$

R2 is calculated according to Formula (13):

$$\mathrm{R2} = 1 - \left(\frac{SS_{res}}{SS_{tot}}\right), \tag{13}$$

If: SS_{res} – the sum of squares of residuals; SS_{tot} – the total sum of squares.

Adjusted R2 is calculated according to Formula (14):

$$\text{Adjusted R2} = 1 - \frac{(1-\mathrm{R2})(n-1)}{n-p-1}, \tag{14}$$

If: n – the number of observations; p – the number of predictors, R2 – coefficient of determination.

All these metrics were calculated for each of the forecasting models to evaluate the created models.

3.4.1. Comparison of forecasting models

Evaluation metrics were calculated for each of the forecasting models, and for each of the individual cluster. The mean evaluation metrics for each of the used forecasting models are given in Table 2.

Table 2. Mean evaluation metrics of all models

	MAE	MSE	RMSE	R2	Adjusted R2
Linear DTW Hierarchal	1,0760	3,8296	1,5189	0,5049	0,3256
Linear DTW K-means	1,1237	4,3486	1,6197	0,5438	0,3787
Linear Euclidean K-means	0,9460	2,4717	1,3362	0,3601	0,1286
Linear Euclidean Hierarchal	0,9111	2,2094	1,2905	0,4004	0,1834
TCN DTW Hierarchal	1,1654	4,2545	1,6932	0,5025	0,3182
TCN DTW K-means	0,7479	1,5263	1,0232	0,5322	0,3588
TCN Euclidean K-means	1,0697	3,4703	1,5354	0,4691	0,2724
TCN Euclidean Hierarchal	0,9074	2,1196	1,3217	0,5027	0,3183
FNN DTW Hierarchal	1,1691	4,5451	1,7129	0,5313	0,3576
FNN DTW K-means	1,1330	3,5533	1,6267	0,4873	0,2972
FNN Euclidean K-means	1,0275	3,5939	1,5493	0,4577	0,2567
FNN Euclidean Hierarchal	0,8560	2,0860	1,3061	0,5186	0,3402

The standard deviation of metrics for each of the used forecasting models are given in Table 3. In both Tables 2 and 3, the darker the green colour, the better the performance of the model in the corresponding metric compared to others.

Table 3. Standard deviation metrics of all models

	MAE	MSE	RMSE	R2	Adjusted R2
Linear DTW Hierarchal	0,3066	2,0820	0,4679	0,0772	0,1051
Linear DTW K-means	0,5124	3,0002	0,7917	0,0992	0,1350
Linear Euclidean K-means	0,2124	0,9674	0,3236	0,0692	0,0943
Linear Euclidean Hierarchal	0,1251	0,6222	0,1894	0,0551	0,0751
TCN DTW Hierarchal	0,8290	6,0753	1,2566	0,2124	0,2910
TCN DTW K-means	0,5475	1,4199	0,7574	0,2870	0,3934
TCN Euclidean K-means	0,7506	4,5401	1,1145	0,1640	0,2248
TCN Euclidean Hierarchal	0,3846	1,7073	0,6129	0,1905	0,2610
FNN DTW Hierarchal	0,9000	6,9113	1,3648	0,1405	0,1925
FNN DTW K-means	0,6671	4,3015	1,0310	0,1567	0,2148
FNN Euclidean K-means	0,6796	4,8642	1,1466	0,1807	0,2478
FNN Euclidean Hierarchal	0,3424	1,6394	0,6041	0,1713	0,2347

Higher standard deviation values show that the data is widely spread (less reliable). Lower standard deviation values show that the data are closer together.

In Figures 37-41, a comparison of mean values of metrics across all trained models is given. Standard deviation is included as error bars.

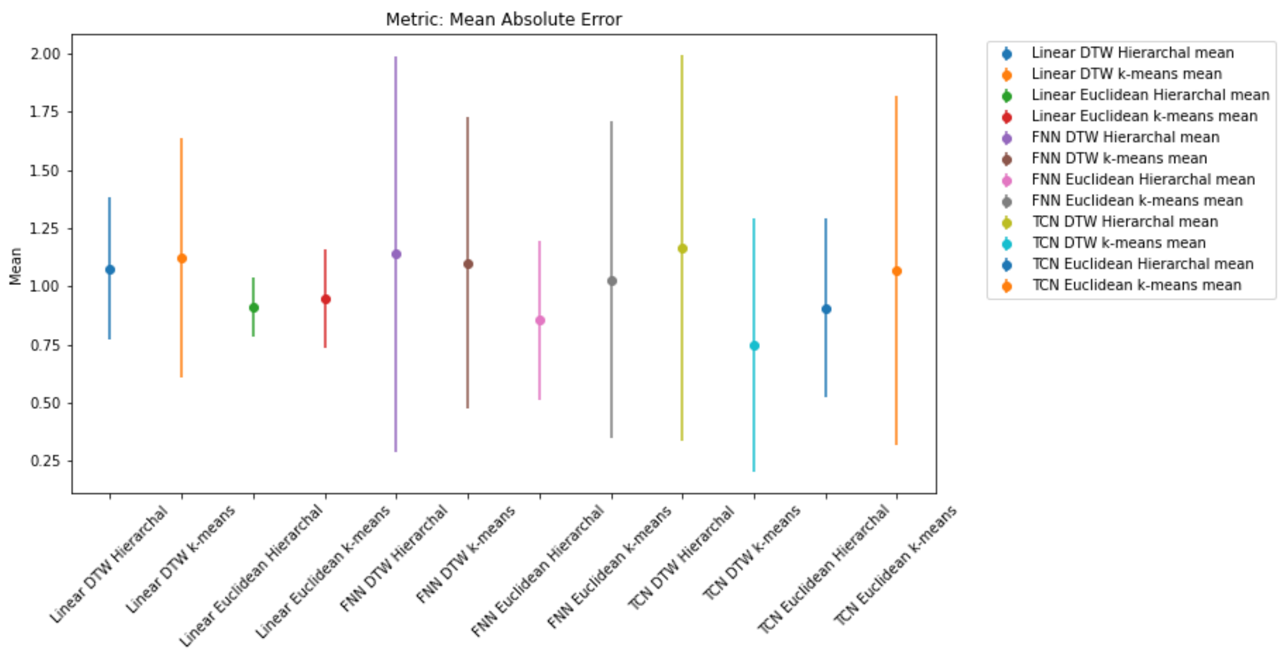


Figure 37. Mean Absolute Error across all models.

MAE measures the average error between predicted and actual data. Lower values indicate that the model performs better. In Figure 37, TCN DTW k-means model performed the best (MAE = 0,7479). TCN DTW Hierarchal method performed the worst (MAE = 1,1654), and had the largest standard deviation (σ = 0,8289).

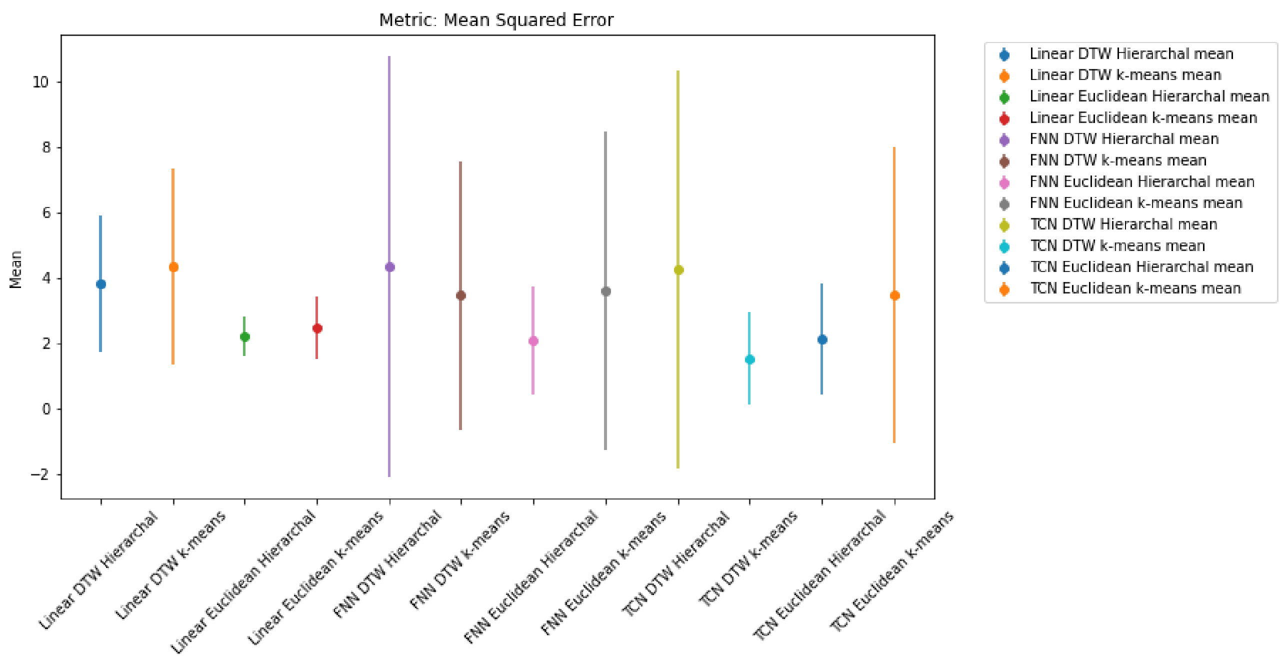


Figure 38. Mean Squared Error across all models.

MSE measures the average squared error between predicted and actual values. Lower values indicate that the model performs better. In Figure 38, TCN DTW k-means model performed the best (MSE = 1,5263). FNN DTW Hierarchal method performed the worst (MSE = 4,545), and also had the largest standard deviation (σ = 6,9113).

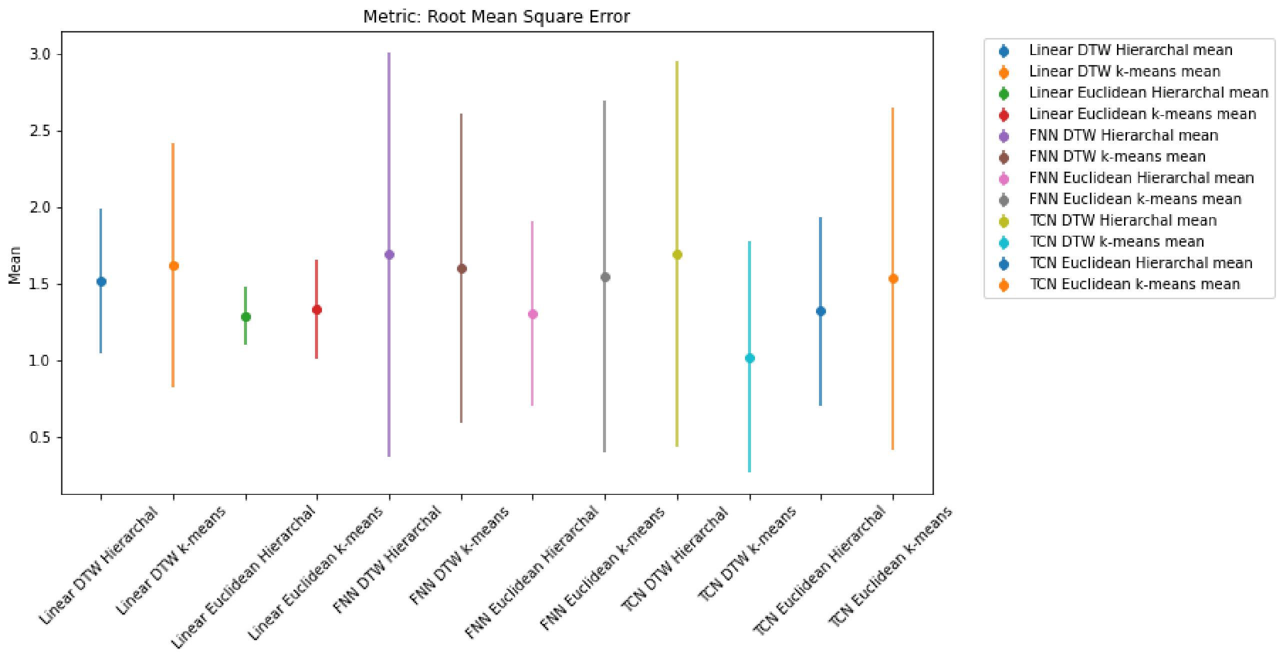


Figure 39. Root Mean Squared Error across all models.

RMSE measures the root average squared error between predicted and actual values. Lower values indicate that the model performs better. In Figure 39, TCN DTW k-means model also performed the best (RMSE = 1,0232). FNN DTW Hierarchal method performed the worst (RMSE = 1,7129), and also had the largest standard deviation (σ = 1,3648).

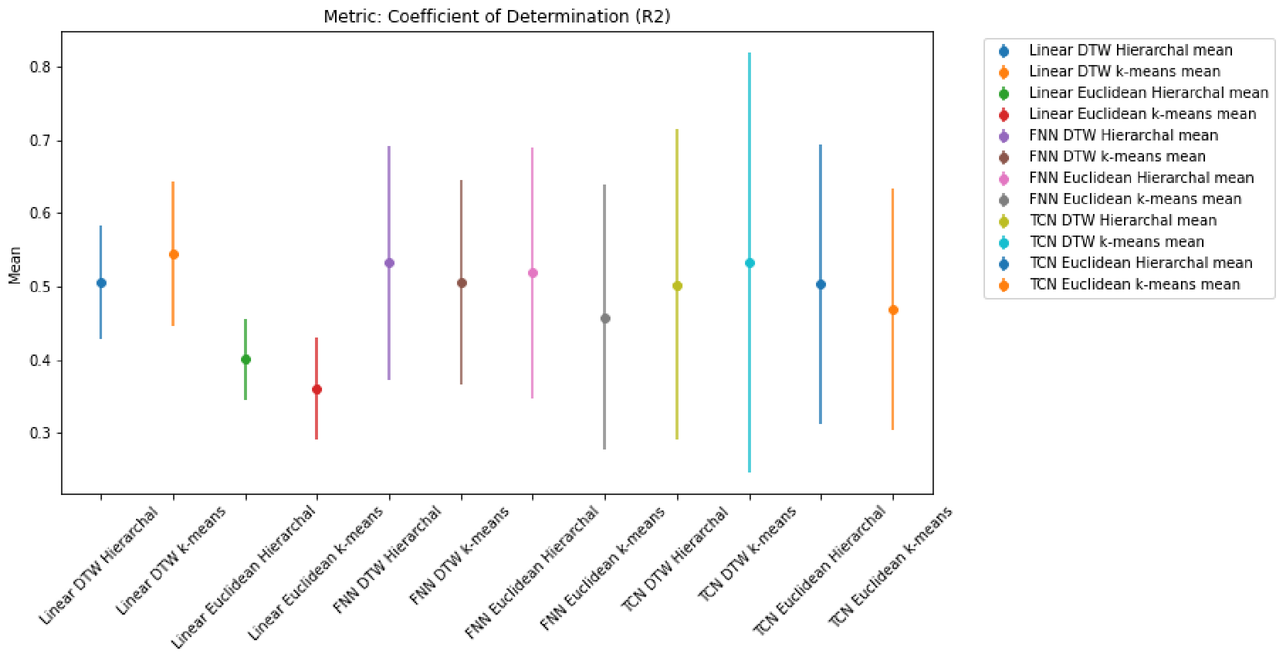


Figure 40. Coefficient of Determination (R2) across all models

Coefficient of determination measures how much of the model‘s variability is explained by the model. Higher values generally mean that the model performs better. In Figure 40, linear DTW k-means model has the highest mean value (R2 = 0,5438). Linear Euclidean k-means performed the worst across mean values (R2 = 0,3602).

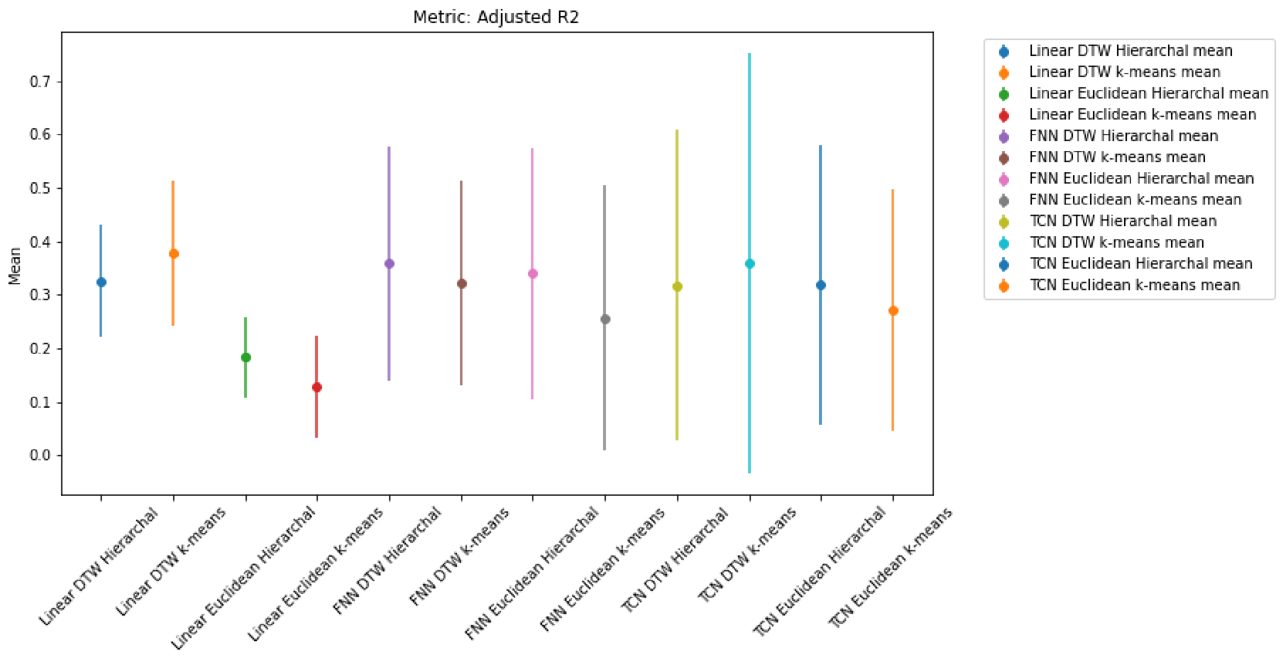


Figure 41. Adjusted Coefficient of Determination across all models

Adjusted coefficient of determination also measures how much of the model‘s variability is explained by the model. Unlike R2, adjusted R2 accounts for the increasing number of variables added to the model. If adding more values does not increase model‘s performance more than adding random noise, Adjusted R2 will be lower. Higher values generally mean that the model performs better. In Figure 41, linear DTW k-means model has the highest mean value (Adj. R2 = 0,3787). Linear Euclidean k-means performed the worst across mean values (Adj. R2 = 0,1286).

For summary statistics of each metric across all models see Appendix 3.

From the calculated metrics, the model with the highest coefficient of determination was achieved using linear regression, DTW distance matrix and k-means clustering method (R2 = 0,5438), though error metrics of this model are not the best compared to the others. The second-best performing model is TCN DTW K-means (R2 = 0,5322). It‘s error metrics are the best among all of the models, but it also has the largest standard deviation, which makes it less reliable.

In linear models, choosing the right distance matrix seem to have the largest impact on model’s performance. Even though linear models using Euclidean distance matrix did not perform as well in R2 and adj. R2 metrics, linear models with Euclidean distance matrix had the lowest standard deviation across all models.

Different model structures were used for linear and AI-based models. This may have led to better performance of linear models. Before choosing model structures, multiple other options were applied. Using the model structure created for the linear model on an AI-based model resulted in inferior results. Using the model structure created for the AI-based model on a linear model resulted in inferior results as well. Model structures indicated in the methodology provided the best results out of attempted combinations.

Model structure for AI-based models contained a 13-hour gap (from D-0 11 am to D-1 0 pm) between the last value used for training, and the first forecasted value. The linear model structure did not contain a gap between training data and forecasted data.

Even though the AI-based model forecasting structure contained a 13-hour gap, forecasting accuracy was similar to the basic linear regression models. TCN DTW k-means model performed slightly better than other models, but it also had one of the highest standard deviations across all metrics.

Another reason for the difference between model’s performance might be that the quality of the data used in this project is quite poor. Many users were missing data completely for periods of time, the data had big spikes (see Appendix 1).

Conclusions

1. In the last few years, more and more applications of AI have been introduced aiming to aid in various fields of electrical power systems, namely electrical load forecasting. This project focused on applying and comparing linear regression with AI-based forecasting methods for electrical load forecasting.
2. Linear regression, TCN and FNN were used to create load forecasting models to forecast load of 176 users. Results indicate that AI-based load forecasting models did not outperform linear regression models.
3. A linear model using DTW distance matrix and k-means clustering had the highest forecasting performance (R2 = 0,5438), (Adj. R2 = 0,3738), but it also had one of the worst mean error metrics (MAE = 1,1237), (MSE = 4,3486), (RMSE = 1,6197). The second-best model was TCN DTW K-means (R2 = 0,5322), (Adj. R2 = 0,3588). The latter also had the best mean error metrics (MAE = 0,7479), (MSE = 1,5263), (RMSE = 1,0232).
4. The worst performing model was the linear model using Euclidean distance matrix and k-means clustering (R2 = 0,3601), (Adj. R2 = 0,1286), (MAE = 0,9460), (MSE = 2,4717), (RMSE = 1,3362).
5. To further improve the accuracy of models, multiple steps could be taken: using a different approach preparing the data, structuring the data differently, adjusting the forecasting model structure, applying other forecasting methods, and conducting search for optimal model parameters.
6. To get a clearer understanding of performance of applied models, more forecasting methods could be applied for comparison: non-linear regression models, autoregressive–moving-average models, other AI-based models like RL or RNN.
7. Using complex AI-based models for load forecasting do not necessarily lead to better results. More attention should be paid to improving quality of the data used to train AI-based load forecasting models.

List of references

1. CHENG, L. - YU, T. *A new generation of AI: A review and perspective on machine learning technologies applied to smart energy and electric power systems*. In *International Journal of Energy Research*. 2019. Vol. 43, no. 6, p. 1928–1973.

2. WANG, H. et al. *A review of deep learning for renewable energy forecasting*. In *Energy Conversion and Management*. 2019. Vol. 198, p. 111799.

3. HOSSAIN, E. et al. *Application of Big Data and Machine Learning in Smart Grid, and Associated Security Concerns: A Review*. In *IEEE Access*. 2019. Vol. 7, p. 13960–13988.

4. NAKABI, T.A. - TOIVANEN, P. *Deep reinforcement learning for energy management in a microgrid with flexible demand*. In *Sustainable Energy, Grids and Networks* [online]. 2021. Vol. 25, p. 100413. Available from: <https://doi.org/10.1016/j.segan.2020.100413>.

5. MISHRA, M. et al. *Deep learning in electrical utility industry: A comprehensive review of a decade of research*. In *Engineering Applications of Artificial Intelligence* [online]. 2020. Vol. 96, no. December 2019, p. 104000. Available from: <https://doi.org/10.1016/j.engappai.2020.104000>.

6. CHUNG, H. et al. *Distributed deep reinforcement learning for intelligent load scheduling in desidential smart grids*. In. 2021. Vol. 17, no. 4, p. 2752–2763.

7. SILVER, D. et al. *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play*. In *Science*. 2018. Vol. 362, no. 6419, p. 1140–1144.

8. THOMAS, D. et al. *Optimal operation of an energy management system for a grid-connected smart building considering photovoltaics' uncertainty and stochastic electric vehicles' driving schedule*. In *Applied Energy* [online]. 2018. Vol. 210, p. 1188–1206. Available from: <https://doi.org/10.1016/j.apenergy.2017.07.035>.

9. ERNST, D. et al. *Power Systems Stability Control: Reinforcement Learning Framework*. In *IEEE Transactions on Power Systems*. 2004. Vol. 19, no. 1, p. 427–435.

10. VÁZQUEZ-CANTELI, J.R. - NAGY, Z. *Reinforcement learning for demand response: A review of algorithms and modeling techniques*. In *Applied Energy*. 2019. Vol. 235, p. 1072–1089.

11. MNIH, V. et al. *Playing Atari with Deep Reinforcement Learning*. In [online]. 2013. Available from: <http://arxiv.org/abs/1312.5602>.

12. MATHEW, A. *Intelligent residential energy management system using deep reinforcement learning*. In. 2018. Vol. 3, no. 2, p. 2–4.

13. XIONG, R. et al. *Towards a smarter hybrid energy storage system based on battery and ultracapacitor - A critical review on topology and energy management*. In *Journal of Cleaner Production*. 2018. Vol. 202, p. 1228–1240.

14. LIU, Y. et al. *Optimization strategy based on deep reinforcement learning for home energy management*. In *CSEE Journal of Power and Energy Systems*. 2020. Vol. 6, no. 3, p. 572–582.

15. XIONG, R. et al. *Reinforcement learning-based real-time power management for hybrid energy storage system in the plug-in hybrid electric vehicle*. In *Applied Energy*. 2018. Vol. 211, p. 538–548.

16. LU, R. et al. *Multi-agent deep reinforcement learning based demand response for discrete manufacturing systems energy management*. In *Applied Energy* [online]. 2020. Vol. 276, no. June, p. 115473. Available from: <https://doi.org/10.1016/j.apenergy.2020.115473>.

17. ZHANG, X. et al. *An Edge-Cloud Integrated Solution for Buildings Demand Response Using Reinforcement Learning*. In *IEEE Transactions on Smart Grid*. 2021. Vol. 12, no. 1, p. 420–431.

18. WEN, L. et al. *Modified deep learning and reinforcement learning for an incentive-based demand response model*. In *Energy* [online]. 2020. Vol. 205, p. 118019. Available from: <https://doi.org/10.1016/j.energy.2020.118019>.

19. YE, Y. et al. *Real-Time Autonomous Residential Demand Response Management Based on Twin Delayed Deep Deterministic Policy Gradient Learning*. In *Energies*. 2021. Vol. 14, no. 3, p. 531.

20. LIU, Y. et al. *Evaluating smart grid renewable energy accommodation capability with uncertain generation using deep reinforcement learning*. In *Future Generation Computer Systems* [online].

2020. Vol. 110, p. 647–657. Available from: <https://doi.org/10.1016/j.future.2019.09.036>.

21. WANG, F. et al. *Autonomous PEV Charging Scheduling Using Dyna-Q Reinforcement Learning*. In *IEEE Transactions on Vehicular Technology*. 2020. Vol. 69, no. 11, p. 12609–12620.

22. HESSEL, M. et al. *Rainbow: Combining Improvements in Deep Reinforcement Learning*. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018* [online]. 2017. p. 3215–3222. Available from: <http://arxiv.org/abs/1710.02298>.

23. KOLODZIEJCZYK, W. et al. *Real-time energy purchase optimization for a storage-integrated photovoltaic system by deep reinforcement learning*. In *Control Engineering Practice* [online]. 2021. Vol. 106, no. December 2019, p. 104598. Available from: <https://doi.org/10.1016/j.conengprac.2020.104598>.

24. HUA, H. et al. *Optimal energy management strategies for energy Internet via deep reinforcement learning approach*. In *Applied Energy*. 2019. Vol. 239, p. 598–609.

25. AFRASIABI, M. et al. *Multi-agent microgrid energy management based on deep learning forecaster*. In *Energy*. 2019. Vol. 186, p. 115873.

26. WU, L. et al. *A Short-Term Load Forecasting Method Based on GRU-CNN Hybrid Neural Network Model*. In *Mathematical Problems in Engineering*. 2020. Vol. 2020.

27. XIUYUN, G. et al. *Short-term Load Forecasting Model of GRU Network Based on Deep Learning Framework*. In *2nd IEEE Conference on Energy Internet and Energy System Integration, EI2 2018 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., 2018.

28. KONG, W. et al. *Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network*. In *IEEE Transactions on Smart Grid*. 2019. Vol. 10, no. 1, p. 841–851.

29. HOCHREITER, S. - SCHMIDHUBER, J. *Long Short-Term Memory*. In *Neural Computation* [online]. 1997. Vol. 9, no. 8, p. 1735–1780. Available from: <http://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf>.

30. RAFI, S.H. et al. *A Short-Term Load Forecasting Method Using Integrated CNN and LSTM Network*. In *IEEE Access*. 2021.

31. ZHENG, C. et al. *A novel equivalent model of active distribution networks based on LSTM*. In *IEEE Transactions on Neural Networks and Learning Systems*. 2019. Vol. 30, no. 9, p. 2611–2624.

32. KAUR, D. et al. *Smart grid energy management using RNN-LSTM: A deep learning-based approach*. In *2019 IEEE Global Communications Conference, GLOBECOM 2019 - Proceedings*. 2019. p. 0–5.

33. PATYN, C. et al. *Comparing neural architectures for demand response through model-free reinforcement learning for heat pump control*. In *2018 IEEE International Energy Conference, ENERGYCON 2018*. 2018. p. 1–6.

34. WANG, F. et al. *A day-ahead PV power forecasting method based on LSTM-RNN model and time correlation modification under partial daily pattern prediction framework*. In *Energy Conversion and Management* [online]. 2020. Vol. 212, no. February, p. 112766. Available from: <https://doi.org/10.1016/j.enconman.2020.112766>.

35. CHO, K. et al. *Learning phrase representations using RNN encoder-decoder for statistical machine translation*. In *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference* [online]. Association for Computational Linguistics (ACL), 2014. p. 1724–1734. Available from: <https://arxiv.org/abs/1406.1078v3>.

36. CHUNG, J. et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. In [online]. 2014. Available from: <http://arxiv.org/abs/1412.3555>.

37. KUAN, L. et al. *Short-term electricity load forecasting method based on multilayered self-normalizing GRU network*. In *2017 IEEE Conference on Energy Internet and Energy System Integration, EI2 2017 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., 2017. p. 1–5.

38. ZHENG, J. et al. *Short-term Power Load Forecasting of Residential Community Based on GRU Neural Network*. In *2018 International Conference on Power System Technology, POWERCON 2018*

- *Proceedings*. Institute of Electrical and Electronics Engineers Inc., 2019. p. 4862–4868.

39. LEA, C. et al. *Temporal convolutional networks: A unified approach to action segmentation*. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2016. Vol. 9915 LNCS, p. 47–54.

40. YIN, L. - XIE, J. *Multi-temporal-spatial-scale temporal convolution network for short-term load forecasting of power systems*. In *Applied Energy* [online]. 2021. Vol. 283, no. November 2020. Available from: <https://doi.org/10.1016/j.apenergy.2020.116328>.

41. TANG, X. et al. *Short-Term Load Forecasting Using Channel and Temporal Attention Based Temporal Convolutional Network*. In *Electric Power Systems Research*. 2022. Vol. 205, p. 107761.

42. TAYLOR, J.H. - SHARIF, S.S. *Short-Term Load Forecasting by Feed-forward Neural Networks*. In [online]. 2000. Available from: <https://www.researchgate.net/publication/228554625>.

43. TALAAT, M. et al. *Load forecasting based on grasshopper optimization and a multilayer feed-forward neural network using regressive approach*. In *Energy*. 2020. Vol. 196, p. 117087.

44. MÜLLER, M. *Dynamic Time Warping*. In *Information Retrieval for Music and Motion* [online]. 2007. p. 69–84. Available from: <https://doi.org/10.1007/978-3-540-74048-3_4>.

45. DANIELSSON, P.E. *Euclidean distance mapping*. In *Computer Graphics and Image Processing*. 1980. Vol. 14, no. 3, p. 227–248.

46. HARTIGAN, J.A. - WONG, M.A. *Algorithm AS 136: A K-Means Clustering Algorithm*. In *Applied Statistics*. 1979. Vol. 28, no. 1, p. 100.

47. MURTAGH, F. - CONTRERAS, P. *Algorithms for hierarchical clustering: an overview, II*. In *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* [online]. 2017. Vol. 7, no. 6, p. e1219. Available from: <https://onlinelibrary.wiley.com/doi/full/10.1002/widm.1219>.

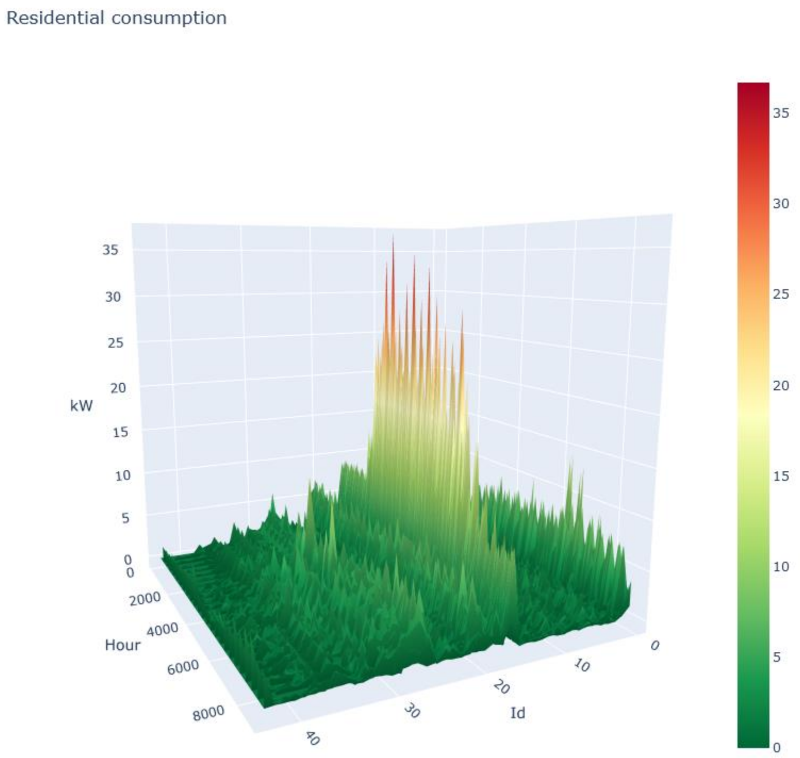
48. MARTIN, A. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. In [online]. 2005. p. 19. Available from: <https://www.tensorflow.org/>.

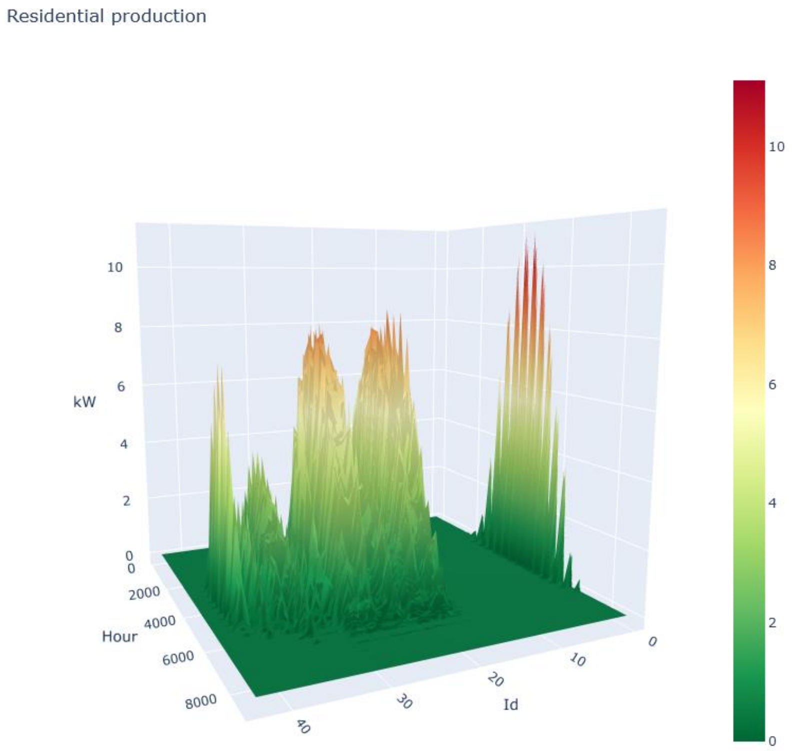
49. SU, X. et al. *Linear regression*. In *Wiley Interdisciplinary Reviews: Computational Statistics* [online]. 2012. Vol. 4, no. 3, p. 275–294. Available from: <https://onlinelibrary.wiley.com/doi/full/10.1002/wics.1198>.

Appendices

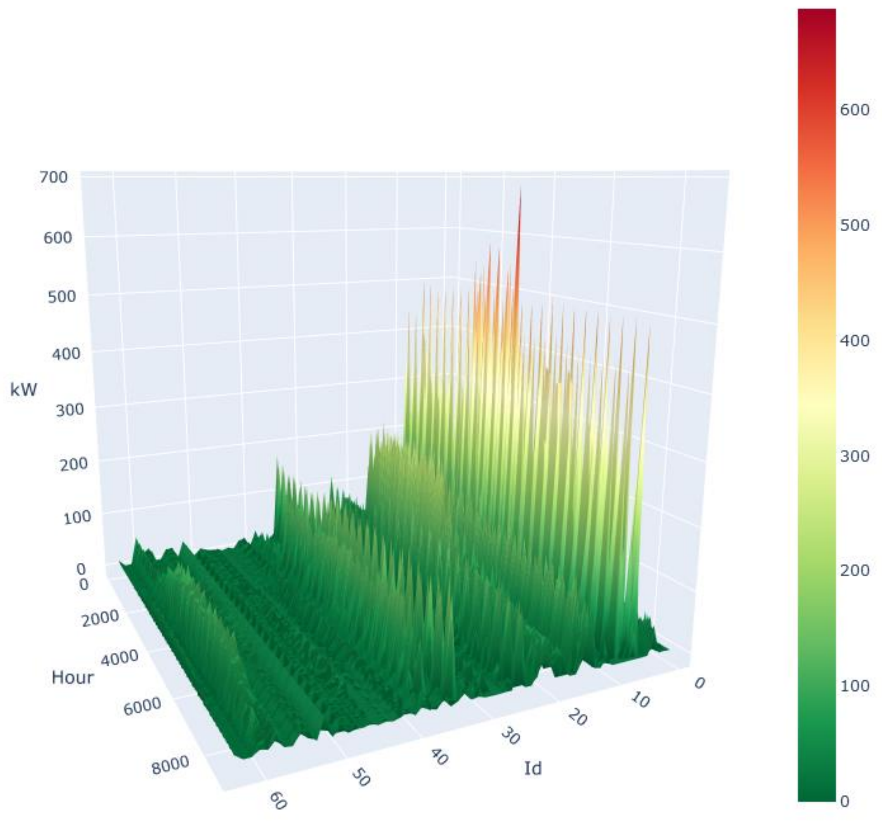
Appendix 1. Consumption and production plots of residential, industrial, and commercial users

The axis labeled “kW” represents the power output of the user. The axis labeled “Hour” represents each hour of the year from 0 to 8784. The axis labeled “Id” represents user ID number.

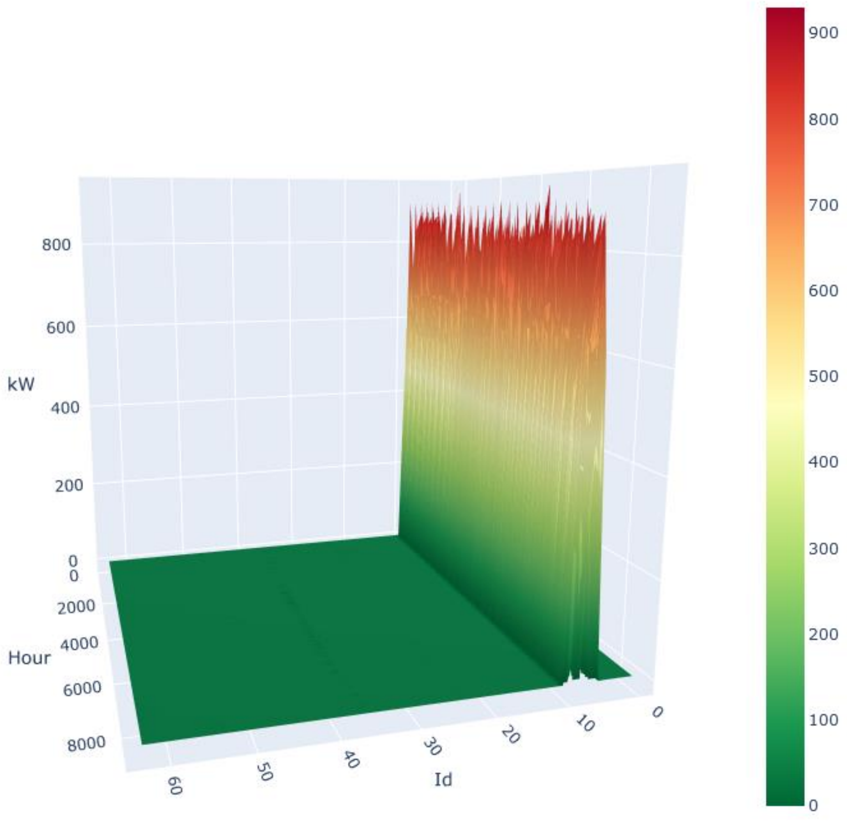


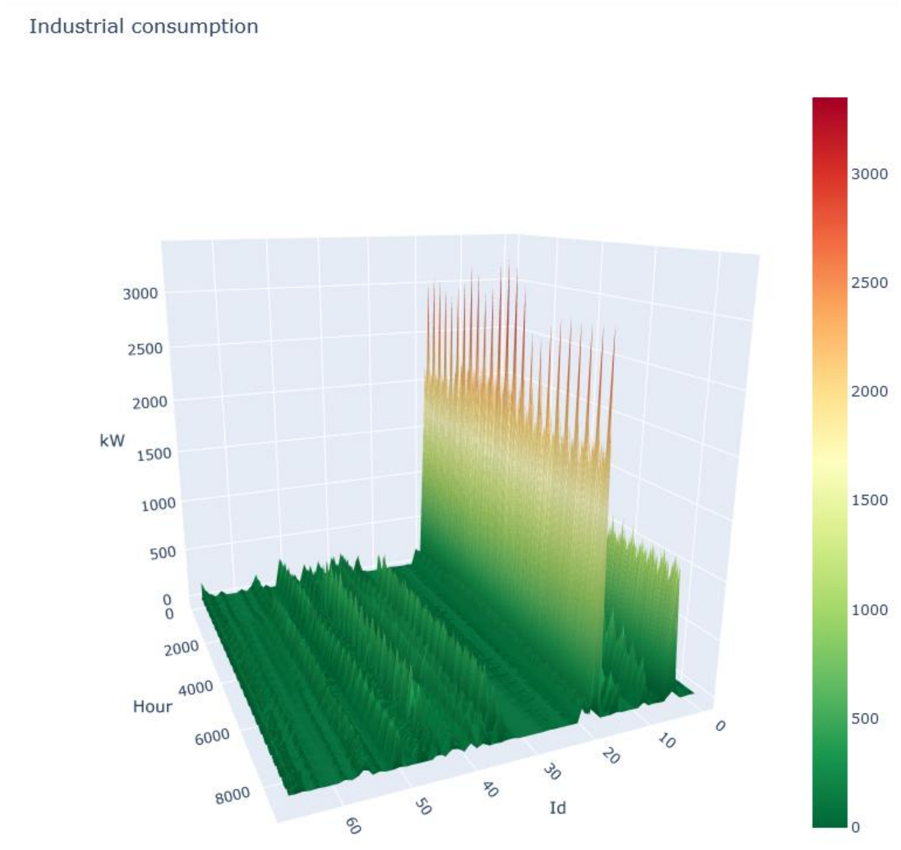


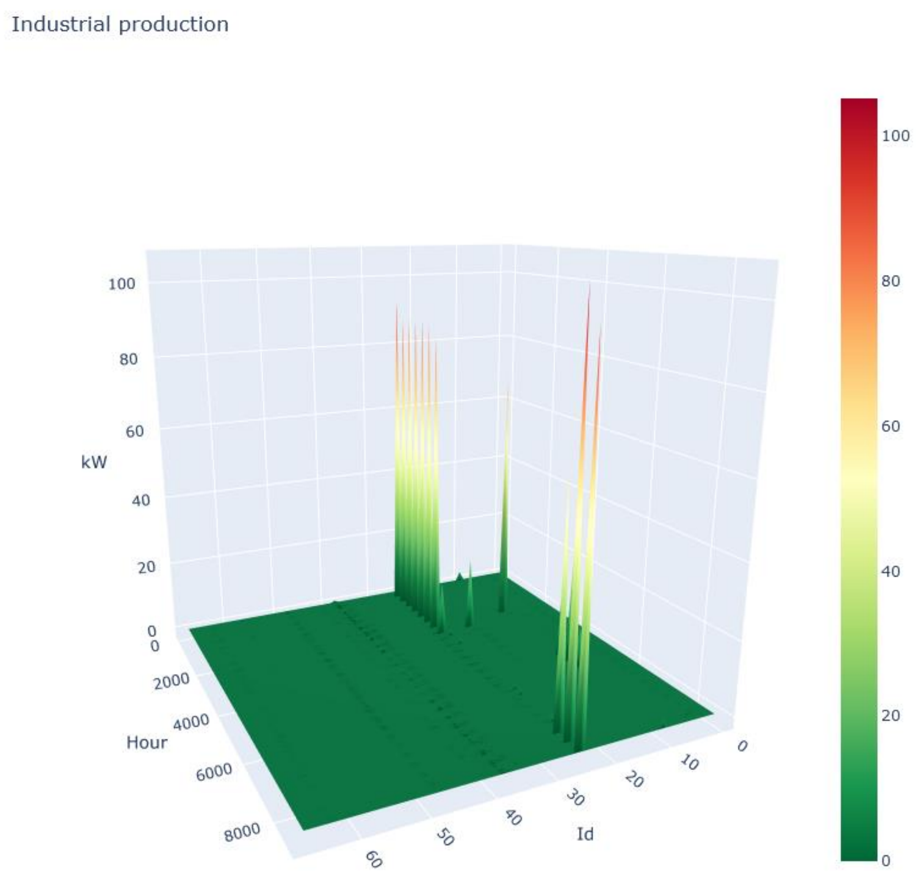
Commercial consumption



Commercial production

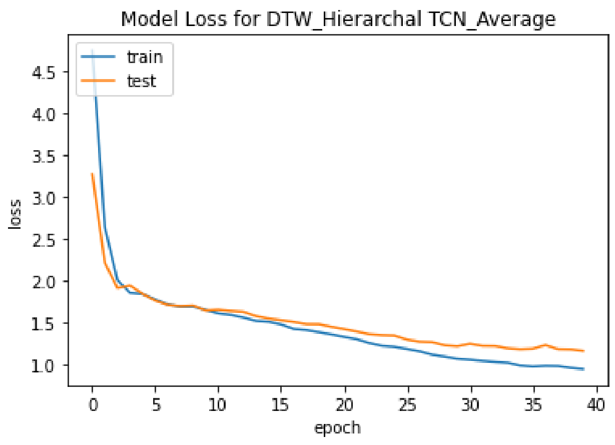


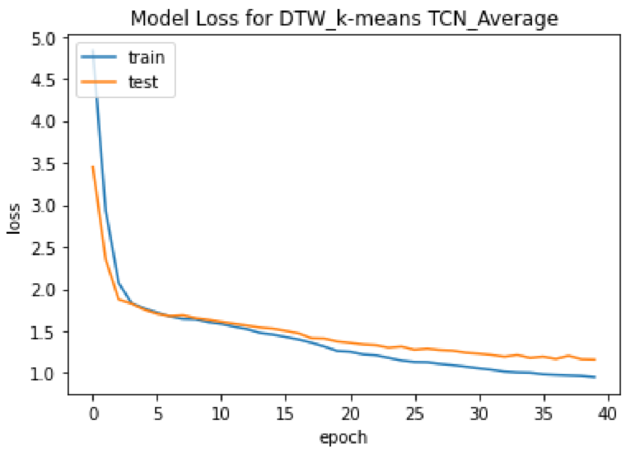

Industrial consumption
kW
0
500
1000
1500
2000
2500
3000
Hour
0
2000
4000
6000
8000
Id
0
10
20
30
40
50
60
0
500
1000
1500
2000
2500
3000

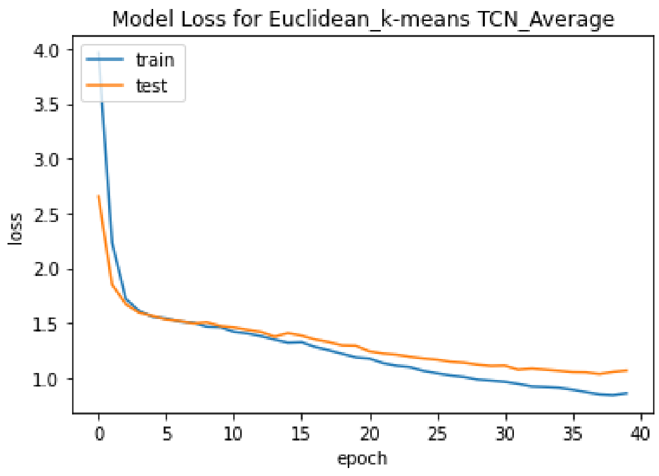

Industrial production
kW
0
20
40
60
80
100
Hour
0
2000
4000
6000
8000
Id
0
10
20
30
40
50
60
0
20
40
60
80
100

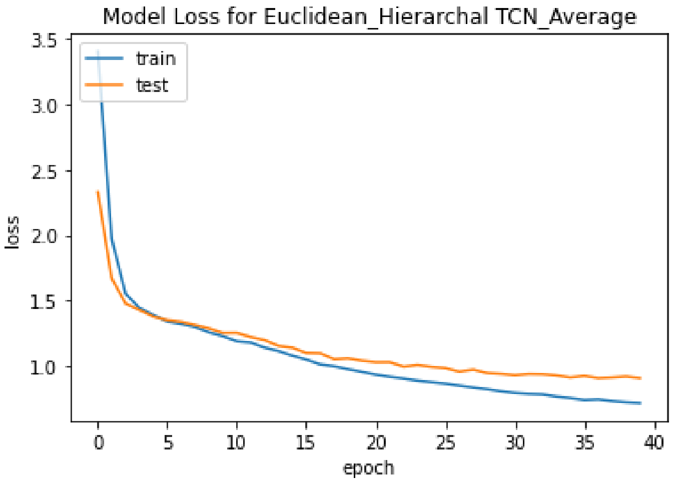
Appendix 2. Testing and training loss of AI-based models

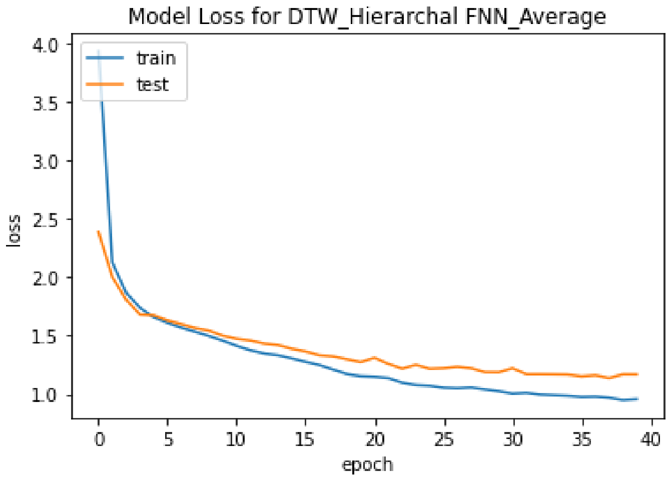
Testing and training loss for each of the TCN and FNN models:

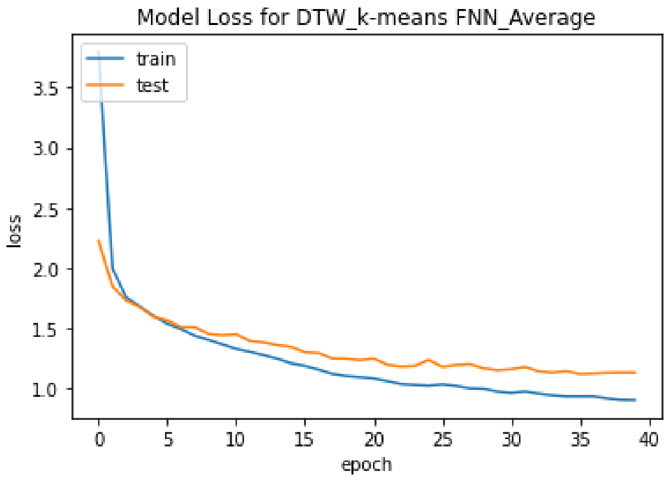


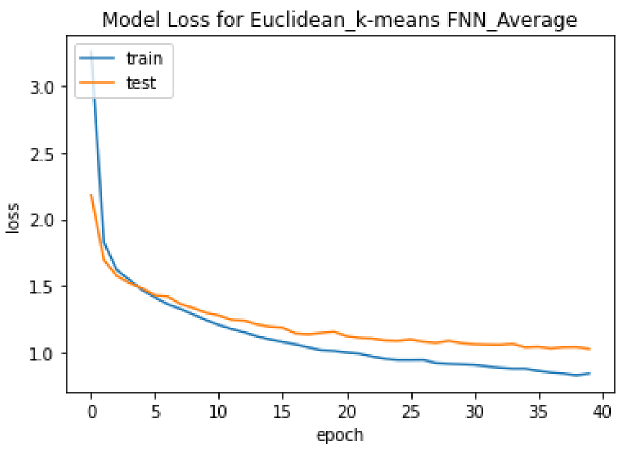




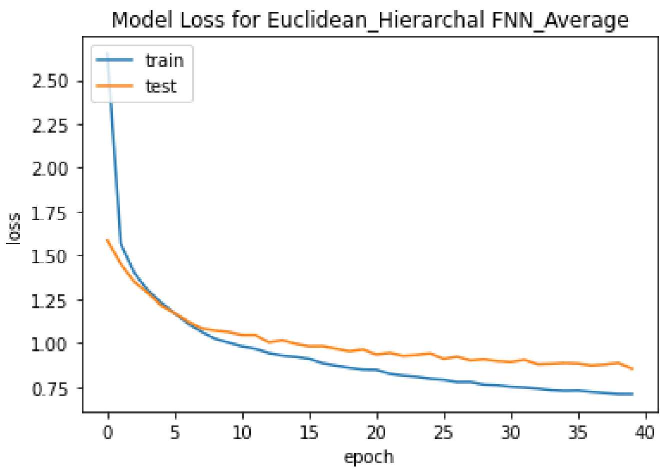

Model Loss for Euclidean_Hierarchal TCN_Average
train
test
loss
epoch


Model Loss for DTW_Hierarchal FNN_Average
train
test
loss
epoch


Model Loss for DTW_k-means FNN_Average
train
test
loss
epoch



Model Loss for Euclidean_k-means FNN_Average
train
test
loss
epoch



Model Loss for Euclidean_Hierarchal FNN_Average
train
test
loss
epoch

Appendix 3. Evaluation metrics of each model

Summary statistics for each metric across all models:

Linear DTW Hierarchal:
```
            MAE       MSE      RMSE        R2  AdjustedR2
count  5.000000  5.000000  5.000000  5.000000    5.000000
mean   1.076042  3.829575  1.518928  0.504892    0.325631
std    0.306647  2.082002  0.467923  0.077157    0.105077
min    0.847109  1.401463  1.122342  0.385942    0.163601
25%    0.873910  3.044616  1.241663  0.472091    0.281026
50%    0.906438  3.083215  1.277281  0.531696    0.362119
75%    1.184421  4.706691  1.690153  0.561752    0.403122
max    1.568333  6.911889  2.263201  0.572980    0.418288
```

Linear DTW K-means:
```
            MAE       MSE      RMSE        R2  AdjustedR2
count  5.000000  5.000000  5.000000  5.000000    5.000000
mean   1.123685  4.348629  1.619661  0.543799    0.378653
std    0.512425  3.000211  0.791740  0.099151    0.135035
min    0.826583  2.632061  1.175498  0.455255    0.258066
25%    0.885973  2.840323  1.239425  0.466489    0.273369
50%    0.906463  2.914813  1.268029  0.535487    0.367275
75%    0.963245  3.688332  1.385978  0.559222    0.399707
max    2.036163  9.667618  3.029376  0.702543    0.594847
```

Linear Euclidean K-means:
```
            MAE       MSE      RMSE        R2  AdjustedR2
count  6.000000  6.000000  6.000000  6.000000    6.000000
mean   0.945991  2.471704  1.336231  0.360148    0.128593
std    0.212396  0.967382  0.323592  0.069212    0.094297
min    0.756013  1.672721  1.059322  0.294394    0.039073
25%    0.818579  1.801402  1.131043  0.307893    0.057378
50%    0.893598  2.185552  1.255503  0.336730    0.096633
75%    0.967761  2.695002  1.372289  0.409294    0.195621
max    1.346682  4.248232  1.946281  0.461270    0.266306
```

Linear Euclidean Hierarchal:
```
            MAE       MSE      RMSE        R2  AdjustedR2
count  7.000000  7.000000  7.000000  7.000000    7.000000
mean   0.911100  2.209432  1.290514  0.400384    0.183372
std    0.125140  0.622248  0.189389  0.055087    0.075082
min    0.822484  1.627872  1.164591  0.336606    0.096506
25%    0.842188  1.904053  1.181614  0.361405    0.130251
50%    0.873252  1.995921  1.226540  0.395952    0.177271
75%    0.906986  2.264972  1.288249  0.429785    0.223422
max    1.183616  3.504181  1.702742  0.487749    0.302479
```

TCN DTW Hierarchal:
```
            MAE        MSE      RMSE        R2  AdjustedR2
count  5.000000   5.000000  5.000000  5.000000    5.000000
mean   1.165449   4.254525  1.693216  0.502509    0.318171
std    0.828957   6.075285  1.256602  0.212364    0.291023
min    0.383389   0.270544  0.513795  0.127498   -0.195733
25%    0.774966   1.189742  1.079800  0.557229    0.393139
50%    1.005833   1.984809  1.397806  0.567922    0.407764
75%    1.099481   2.839116  1.669348  0.628097    0.490357
max    2.563575  14.988413  3.805334  0.631797    0.495330
```

TCN DTW K-means:
```
            MAE       MSE      RMSE        R2  AdjustedR2
```

```
count  5.000000  5.000000  5.000000  5.000000    5.000000
mean   0.747941  1.526299  1.023190  0.532185    0.358793
std    0.547504  1.419914  0.757377  0.287021    0.393419
min    0.000000  0.000000  0.000000  0.245877   -0.033767
25%    0.352536  0.233853  0.479224  0.368781    0.134957
50%    0.980893  1.866968  1.358227  0.492822    0.304837
75%    1.117676  2.102717  1.438798  0.553447    0.387940
max    1.288598  3.427958  1.839698  1.000000    1.000000
```

TCN Euclidean K-means:

```
            MAE        MSE      RMSE        R2  AdjustedR2
count  6.000000   6.000000  6.000000  6.000000    6.000000
mean   1.069693   3.470283  1.535380  0.469144    0.272421
std    0.750630   4.540097  1.114547  0.164046    0.224843
min    0.347562   0.230483  0.475172  0.264142   -0.008635
25%    0.553959   0.578828  0.754525  0.350462    0.109801
50%    0.935155   1.906836  1.283458  0.458180    0.257416
75%    1.273112   3.829608  1.927558  0.619493    0.478468
max    2.387626  12.170856  3.449719  0.646541    0.515580
```

TCN Euclidean Hierarchal:

```
            MAE       MSE      RMSE        R2  AdjustedR2
count  7.000000  7.000000  7.000000  7.000000    7.000000
mean   0.907402  2.119617  1.321692  0.502712    0.318343
std    0.384560  1.707332  0.612924  0.190458    0.261015
min    0.353000  0.234531  0.480518  0.243831   -0.036569
25%    0.643118  0.751187  0.856737  0.335526    0.089281
50%    0.896336  2.078551  1.422353  0.618444    0.477017
75%    1.193176  2.983486  1.711719  0.649390    0.519368
max    1.429890  5.054890  2.212063  0.686878    0.570653
```

FNN DTW Hierarchal:

```
            MAE        MSE      RMSE        R2  AdjustedR2
count  5.000000   5.000000  5.000000  5.000000    5.000000
mean   1.169140   4.545055  1.712901  0.531320    0.357561
std    0.900010   6.911305  1.364788  0.140493    0.192541
min    0.333058   0.214762  0.459015  0.313300    0.058742
25%    0.816732   1.364431  1.156773  0.491870    0.303609
50%    0.992032   1.965142  1.389736  0.575667    0.418310
75%    0.999574   2.357614  1.516676  0.586959    0.433735
max    2.704303  16.823324  4.042307  0.688806    0.573411
```

FNN DTW K-means:

```
            MAE        MSE      RMSE        R2  AdjustedR2
count  5.000000   5.000000  5.000000  5.000000    5.000000
mean   1.133048   3.553301  1.626655  0.487316    0.297248
std    0.667103   4.301533  1.030965  0.156701    0.214828
min    0.351027   0.239943  0.484931  0.232079   -0.052675
25%    0.985866   1.824454  1.340426  0.450282    0.246486
50%    0.987602   1.893922  1.365166  0.539751    0.369154
75%    1.146781   2.730582  1.643543  0.597416    0.448153
max    2.193965  11.077601  3.299208  0.617054    0.475120
```

FNN Euclidean K-means:

```
            MAE        MSE      RMSE        R2  AdjustedR2
count  6.000000   6.000000  6.000000  6.000000    6.000000
```

```
mean   1.027501   3.593903  1.549332  0.457675   0.256690
std    0.679599   4.864151  1.146603  0.180748   0.247779
min    0.349489   0.232227  0.476075  0.256183  -0.019512
25%    0.553264   0.593456  0.761058  0.319609   0.067349
50%    0.899340   1.784095  1.256762  0.430296   0.219220
75%    1.271910   3.868923  1.932946  0.607662   0.462320
max    2.182830  13.002556  3.552228  0.679779   0.561132
```

FNN Euclidean Hierarchal:

```
            MAE       MSE       RMSE        R2  AdjustedR2
count  7.000000  7.000000  7.000000  7.000000    7.000000
mean   0.856028  2.086006  1.306064  0.518617    0.340224
std    0.342427  1.639414  0.604064  0.171287    0.234736
min    0.345998  0.229848  0.474773  0.261216   -0.012580
25%    0.600594  0.687415  0.819642  0.393580    0.168939
50%    0.952138  2.439164  1.531731  0.557681    0.393607
75%    1.111980  2.882984  1.681157  0.661972    0.536696
max    1.268910  4.792231  2.134346  0.700316    0.589270
```