Technical University of Crete
School of Electrical and Computer Engineering

Development of a Competitive Autonomous Agent for Smart Grid Energy Markets

Diploma Thesis

Mastorakis Antonios

Thesis Committee:
Advisor: Georgios Chalkiadakis, Professor ECE TUC
Fotios Kanellos, Associate Professor ECE TUC
Vasilis Samoladas, Associate Professor ECE TUC

Chania
May, 2022

Acknowledgments

I would like to especially thank my supervisor Professor Georgios Chalkiadakis as well as Stavros Orfanoudakis for their consistent support and guidance during the running of this project. This thesis is dedicated to my parents, for their endless love, support and encouragement.

Abstract

Modern electricity markets require real-time sales and purchases, for which many factors must be taken into account to keep pace with market growth rates. In this context, the international Power Trading Agent (PowerTAC) competition simulates a realistic platform for electricity deals and sales, equivalent to real stock exchanges energy such as Nord Pool and EEX. On this platform, various intelligent software agents – brokers (developed by research teams around the globe) compete with each other, with the main purpose of obtaining the maximum possible profit. Each team creates its agent and through various strategies in both retail and wholesale markets is trying to achieve a better combination of buying and selling. Another important part of the competition is that agents aim to obtain a disproportionately high share of the market, resulting in financial losses due to the obligation payment of huge fees to the regulatory authorities. Against this background, this thesis focused on the improvement of an already existing agent, TUC TAC. This particular agent was created by a research team at the Technical University of Crete in 2020, and managed to finish first in PowerTAC that year. Two main changes were carried out to achieve the objective of improving TUC TAC. The first is the addition of a Predictor (forecast factor) for the competition's wholesale market. By predicting future wholesale market prices, TUC TAC will be able increase its overall profits. We tackled this problem via classical machine learning methods, including Neural Networks. The second major change is the optimization of a Monte Carlo Tree Search algorithm that was already used by TUC TAC for bidding in the wholesale market's double auctions, via (a) adding the new predictor but also (b) via regulating better the parameters of the algorithm so that selling the same energy amount in the retail market will lead to smaller fines due to the fewer losses in the broker's energy balance sheet. We conducted extensive simulation experiments to test our modifications and evaluate various versions of our agent in environments of fluctuating difficulty. Our experiments verify the effectiveness of the TUC TAC agent enhancements provided in this thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter describes the motivation behind our study, how we conducted it, and the major contributions we made. It also provides an overview of the remaining parts of this thesis.

1.1 Motivation

The energy market has seen significant structural and dynamic changes as a result of globalization, resulting in a regulated and competitive market environment [1]. Most traditional power grids are turning into technology for preferable solutions, gradually evolving into advanced systems known as smart grids. Smart grids provide for more effective energy use, improved market-level communication, and real-time balancing of energy supply and demand. Reduced fossil fuel use is also one of the key goals of Smart Grid systems. This is crucial for the environment not only because future fossil fuel supplies will eventually run out and we will need an alternative source of energy, but also because of the coal emissions that are produced due to the burning of fossil fuels [2].

To make this new market feasible, researchers need a tool that will enable them to explore in new and creative ways. Power Trading Agent Competition (PowerTAC) is an agent-based competitive simulation that models a modern energy market, like the European and North American wholesale energy markets. Brokers provide energy services to clients through tariff contracts and are required to serve those customers by trading in a wholesale market. The PowerTAC simulation platform is equipped with and offers most functionalities a smart power grid can have, including electric vehicles, renewable energy, etc, resulting in highly realistic simulations. The team from the school of Electrical and Computer Engineering at the Technical University of Crete developed TUC TAC, a competitive broker that participated in the PowerTAC 2020 and PowerTAC 2021 competition [3].

One primary objective in this thesis was to observe data and decide what to keep and what to discard. The PowerTAC offers a pretty realistic environment with a lot of different variables that also exist in the real world. The ultimate goal of our project was to develop an optimal predictor that uses the environment's data and feeds it into machine learning/deep learning algorithms to forecast the future energy prices of the market, so our agent can also improve the overall performance of the existing TUC-TAC agent.

1.2 Thesis Contributions

The major goal of this thesis is to enhance TUC TAC's 2020 overall performance by optimizing its wholesale market trade and we achieve that with two major improvements.[1] The first one is the creation of a predictor for the clearing prices, that will enable the broker to make informed choices when buying energy at a discount in a day-ahead market. Numerous strategies were used to achieve that goal. At first, developing the predictor was approached as a regression problem, starting with linear regression and then with polynomial regression. Machine Learning algorithms such as LSTM and RNN were also examined. The second improvement that we made is the modification of the Monte Carlo Tree Search algorithm in the Wholesale Market of our agent, which is responsible for buying large amounts of energy. This MCTS algorithm is responsible for consulting the agent on when is the most profitable time to make a transaction and we aim to sell the same amounts of kWh in the retail market but with less total charges and kWh imbalance.[2] In this way, our agent aims to maximize profits and minimize the losses. The final outcome was more than satisfying, as we managed to improve the efficiency of our broker by adding one prediction method and by enhancing the MCTS algorithm.

The methods proposed in this thesis, particularly the MCTS extension we implemented, contributed to achieving second place in this year's PowerTAC finals. Compared to last year's TUC TAC agent, TUC TAC 2022 extended the MCTS algorithm to effectively reduce the penalties that had to be paid due to the created energy imbalance. Moreover, as planned, our method achieved lower average costs per kWh bought in the continuous double auctions of the competition's wholesale market.

1.3 Overview of the Thesis

The work described in this thesis represents a module that is responsible for trading electrical energy efficiently, in a continuous double auction representing the modern energy wholesale markets of Europe and North America. The main modifications involve creating an effective predictor to forecast future clearing prices and make profitable transactions while alternating the wholesale market strategy of the agent to make more profitable transactions. The remaining sections of this thesis are arranged as follows: Chapter 2 provides some background information on areas essential to our project, namely Machine Learning, Deep Learning as well as papers related to these fields. Following, in Chapter 3 we provide a thorough explanation of the competition's guidelines and elements and we cite some successful agents that have won the PowerTAC in previous years. Next, we represent in Chapter 4 the major development of our agent along with each method that was used for our predictor, how we employed them and how we created the datasets. Chapter 5 contains our experimental results with comparisons between them and a discussion about the overall performance of TUC-TAC. Finally, in Chapter 6 we review our project focusing on what we achieved and what could be improved in the future.

[1] TUC TAC 2021 was essentially identical with TUC TAC 2020 baring minimal tweaks.

[2] When the broker's balance is negative, the broker is charged interest on a daily basis and when the broker's balance is positive, the broker is paid a daily interest. The balance is updated daily (once every 24 hours) [1]

Chapter 2

Background

Predictive analytics is the foundation of this thesis, therefore our background studies consisted mainly of predicting method-based papers, which we studied in order to understand better the way that we should approach our problem. Moreover, terms such as Machine Learning and Deep Learning are going to be spoken about and explained, since these were the main tools that helped us come to a result.

2.1 Machine Learning

Machine learning is an area of computer science and artificial intelligence (AI) that focuses on utilizing data and algorithms learn with continuously increasing accuracy. Machine learning is a crucial part of the rapidly expanding discipline of data science. Algorithms are trained to generate classifications or predictions using statistical approaches. These insights drive decision-making within applications and enterprises, with the goal of influencing important growth key performance indicator. KPIs are a quantifiable measure of performance over time for a specific objective and provide targets for teams to shoot for, milestones to gauge progress, and insights that help people make better decisions. There are three main aspects to machine learning [4]:

- **A Decision Process**: Machine learning algorithms are used to produce predictions or classifications in general. The algorithm will generate an estimate about a pattern in the data based on some input data, which can be labeled or unlabeled.
- **An Error Function**: The model's prediction is evaluated using an error function. If there are known examples, an error function can be used to compare the model's accuracy.
- **An Model Optimization Process**: If the model can match the data points in the training set better, weights are adjusted to reduce the distance between the known example and the model prediction. The algorithm will repeat this assess and optimize method, updating weights on its own until a particular level of accuracy is achieved.

Machine learning is divided into three primary categories [4]:

- **Supervised learning**: Supervised learning, often known as supervised machine learning, is the process of using labeled datasets to train algorithms that reliably classify data or predict outcomes. The weights are changed when new data is added to the model until it is well fitted. This occurs during the cross validation phase, which ensures that the model does not overfit or underfit the data. Supervised learning can be used to solve a variety of real-world problems at scale, such as spam classification in a separate folder from your email. In supervised learning, neural networks, naive Bayes, linear regression, logistic regression, random forest, support vector machine (SVM), and other methods are employed [5].

- **Unsupervised Learning**: Supervised learning, often known as supervised machine learning, is the process of using labeled datasets to train algorithms that can consistently classify data or predict outcomes. The weights are changed when new data is added to the model until it is well suited. This occurs during the cross validation process, which is used to ensure that the model does not overfit or underfit. Organizations can utilize supervised learning to scale up a variety of real-world challenges, such as spam classification in a separate folder from email. In supervised learning, techniques such as neural networks, naive Bayes, linear regression, logistic regression, random forest, support vector machine (SVM), and others are utilized.

- **Semi-supervised learning**: Semi-supervised learning is a suitable compromise between supervised and unsupervised learning. During training, it uses a smaller labeled data set to assist categorization and feature extraction from a larger, unlabeled data set. When there isn't enough labeled data to train a supervised learning algorithm, semi-supervised learning can help (or not being able to afford to label enough data) [5].

- **Reinforcement learning**: Reinforcement learning is a paradigm for learning that develops the ability to maximize sequential decisions, under uncertainty, using trial and error. To put it more specifically, it aims to develop the optimum method for making repeated sequential decisions over time in a dynamic system under uncertainty. Reinforcement Learning agents commonly interact with a simulator of the relevant stochastic dynamic system, also known as an environment, aiming to develop an optimal policy, a method for making repeated, sequential decisions over time in a dynamic system. The goal of reinforcement learning is to discover the best course of action to take in various stages of a dynamic system. To put it otherwise, Reinforcement Learning agents aim to control a Markov Decision Process with unknown dynamics. Reinforcement learning works in a mathematical framework that consists of a state space, an action space, and a reward signal [5].

2.2 Deep Learning

Deep learning is a subset of machine learning, which employs learning the neural network with three or more layers. These neural networks make an effort to mimic how the human brain operates so that it may "learn" from vast volumes of data. While a neural network with a single layer may still produce approximations, more hidden layers can assist to improve and optimize for accuracy [4].

Deep neural networks are made up of several layers of connected neurons, each of which improves upon the prediction or classification made by the one underneath it. This method is called forward propagation. A deep neural network's visible layers are its input and output layers. The deep learning model ingests the data for processing in the input layer, and the final prediction or classification is performed in the output layer.

Gradient descent [5] is an iterative first-order optimisation algorithm used to find a local minimum/maximum of a given function. This method is commonly used in machine learning (ML) and deep learning (DL) to minimise a cost/loss function. In machine learning, a gradient is a derivative of a function that has more than one input variable. Known as the slope of a function in mathematical terms, the gradient simply measures the change in all weights with regard to the change in error.

Backpropagation [5] is an automatic differentation and neural network training algorithm that employs techniques like gradient descent to calculate prediction errors before changing the function's weights and biases by iteratively going back through the layers in an effort to train the model. A neural network can generate predictions and make necessary corrections for any faults thanks to forward propagation and backpropagation working together. The algorithm continuously improves in accuracy over time.

2.3 Related Work

We now briefly list some works on time series predictions, as these are relevant to our work in this thesis.

As we mentioned above, Neural Networks are the key component of every time series forecasting. Mohammed and Mahmud et al. [6] enumerate a wide list of Deep Learning-Based Time Series Previous Work that used to a great degree Artificial Neural Networks, such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Deep Autoencoders (AE), Restricted Boltzmann Machines (RBM) and Deep Belief Networks (DBN).

Moreover, Ismail and Gunady et al.[7] describe a time series classification problem that they approached with saliency methods and came to the conclusion that they fail to produce high-quality interpretations when applied to multivariate time series data, however, they can produce accurate maps when multivariate time series are represented as either images or univariate time series.

Using a wavelet de-noising-based backpropagation neural network, Wang et al.[8] suggested a unique method for predicting stock prices. Using monthly closing price data from the Shanghai Composite Index from 1993 to 2009, they demonstrate noticeably better performance when compared to a traditional multi-layer perceptron. They credit wavelet-based de-noising and preprocessing for the increased performance.

In Lago et al.[9], a new modeling framework is proposed to forecast electricity prices. The topic of deep learning algorithms has various prediction models that have already been suggested for this job. The Deep Neural Network (DNN) model, the Long-Short-

Term Memory (LSTM) model, and the Gated Recurrent Unit (GRU) model are three of the four suggested DL forecasts that are demonstrated to have statistically considerably higher prediction accuracy than all other models.

Fisher and Krauss et al.[10] modified LSTM and evaluated its effectiveness against memory-free algorithms like random forest, logistic regression classifier, and deep neural network. They did that to assess LSTM's capacity in financial market predictions. LSTM outperformed other algorithms, however during the 2008 financial crisis, random forests outperformed LSTM. Overall, they have successfully shown that an LSTM network is capable of effectively extracting useful information from noisy financial time series data.

More related work on PowerTAC and prediction methods will be provided in Chapter 3.

Chapter 3

PowerTAC: The Power Trading Agent Competition

After our discussion in the previous Chapter about prediction techniques and their theoretical background, we now present the PowerTAC platform and its components. Ketter and Collins [2] aspired to creating this innovative, realistic, economically competitive simulation of the future energy markets that includes a number of smart grid elements. By building competing agents and comparing their individual tactics against one another, this simulator allows academics to experiment with retail and wholesale market decision-making, as well as better comprehend the behavior of future consumer models.

3.1 Competition Overview

PowerTAC is a power trading agent competition that began in 2012 and is conducted every year since then. Competing teams create trading agents that act as self-interested "brokers" and aggregate energy supply and demand with the purpose of earning a profit. Brokers buy and sell energy through contracts with retail customers such as households,owners of electric vehicles or small and medium enterprises. They can also trade energy in a wholesale market that models a real-world market such as the European or North American wholesale energy markets.As the competition founders mention, this simulation is designed to model the energy trading environment mainly from an economic and not an especially technical viewpoint. So broker agents must operate effectively by scheduling and carrying out tasks over a range of timelines in three markets: a customer market, a wholesale market, and a market for balancing resources.

3.1.1 Simulation Time

For a simulation to work, the game time needs to be discrete, thus some discrete time-blocks are created.These are referred to as "time-slots," and each one represents an hour of simulated time while only taking up about 5 milliseconds of actual time. There are at least 1440 time-slots in each game (two months of simulated time and at least 2 hours of real-time per game). As a result, there is always a time-slot in a PowerTAC game that is active, as well as a set of future times lots that the brokers can reserve and trade energy for. In order to avoid financial gain, the brokers' primary goal is to balance supply and demand in each of the future time periods.

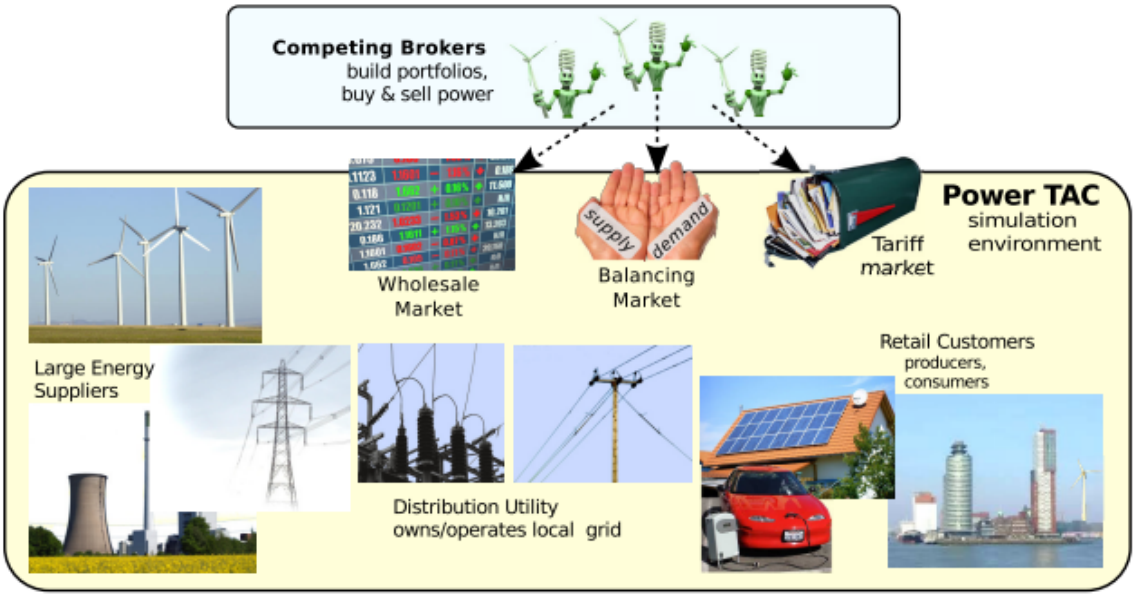

Figure 3.1: Basic elements of PowerTAC scenario[1]

3.1.2 Brokers

Broker-agents are the real-life analogy to energy retailers and have similar actions in their activity book.In each time-slot, every agent can decide and perform any of the following actions:

- Adjust tariff prices, if tariff terms allow it.
- Design and submit new tariffs for publication to customers.
- Change tariff terms for existing customers by replacing a superseded tariff with a new one.
- Offer controllable capacities for real-time balancing, to the extent allowed by tariff terms.
- Create asks and bids to sell or procure energy for future time slots. See Section 5 for details.

3.1.3 Weather Reports

Another significant feature of PowerTAC is the weather reports. In each time-slot,a weather report for the current time is sent to the brokers along with a weather forecast about the next 24 time-slots. Given that the weather has a direct impact on customer models, the agents use this information to make predictions. In particular, weather data is derived from real-world weather reports, thus throughout the competition, agents aren't aware of the game's location for obvious reasons.

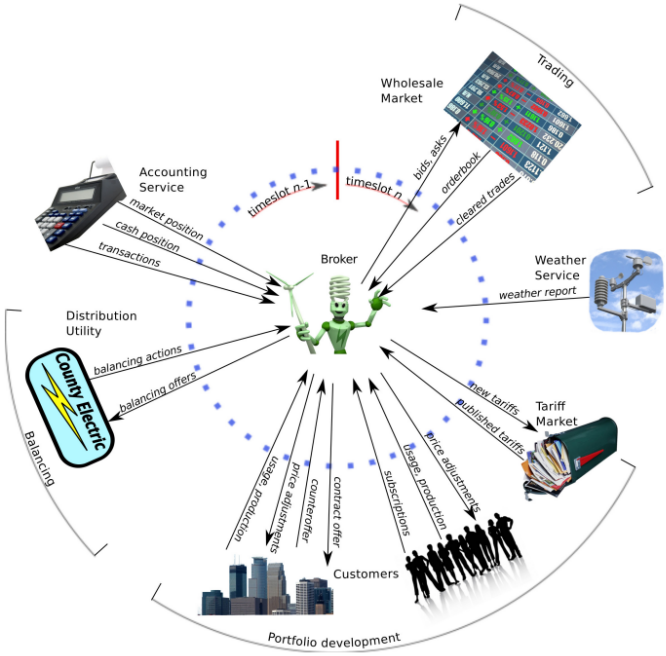

Figure 3.2: Basic actions of an agent broker in a timeslot.

3.1.4 Wholesale Market

The wholesale market functions as a short-term spot market for buying and selling energy commitments in specific time-slots, where each time-slot represents a simulated hour.There are always 24 ongoing auctions since agents can participate in auctions to trade energy at any time during the simulation.These auctions, which are periodic double auctions, are comparable to those held in wholesale energy markets in Europe or North America.Each simulation starts with 14 days' worth of preliminary information (also known as "bootstrap data"), which includes customer information, wholesale market information, and weather information depending on the default broker.Brokers can submit bids (orders to buy energy) and asks (orders to sell energy),represented by a quantity and an optional limit price. By matching buy and sell orders, the simulation clears the bids and establishes the daily clearing price for each auction. The market does not clear if the minimum ask price is higher than the highest bid price.

3.1.5 Trading and time slots available for trade

Brokers can submit orders to the wholesale market for delivery between one and 24 hours in the future.The time slots that are open for trading are marked as "enabled," and brokers are informed when a time slot's status changes.Orders placed for time slots that are not enabled (either deactivated or not yet enabled) are cancelled. The market continuously gathers submitted orders and those that are taken into consideration for clearing are exactly those that have come since the beginning of the last clearing. Each order is represented by 4 variables (b, s, e, p). A broker b, a time slot s, an amount of energy e in megawatt-hours, and optionally a limit price per megawatt-hour p. Quantities of energy and prices are viewed as suggested debits (negative values) and credits (positive values) to the broker's accounts for energy and cash.Limit orders are orders that include a limit price p, whereas market orders are orders that do not include a limit price.

3.1.6 Market clearing

The wholesale market clears the orderbook for each of the enabled time slots when the simulation clock advances to a new time slot. Each broker receives an updated list of the available time slots at the same time. This reduces the amount of time that the set of enabled time slots from the perspective of the broker and the set of enabled time slots from the perspective of the market are different.Demand and supply curves are built using bids and asks in the clearing process to establish the clearing price for each enabled time slot. The supply and demand curves' point of junction is the clearing price.It is important to remember that asks have negative energy and positive cash, whereas bids propose positive energy and negative cash. Also, market orders(those without a price) are ranked first, just as if they had the highest or lowest asking prices.

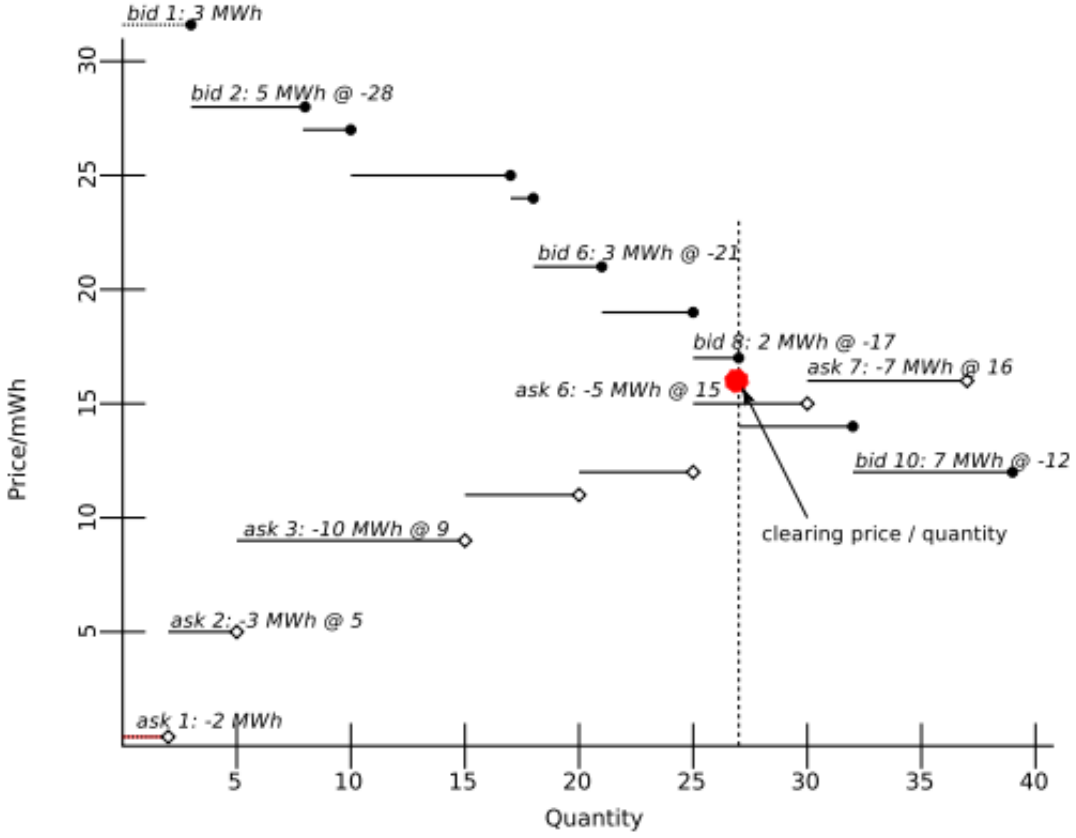

Figure 3.3: Market clearing example.

In the example above,we have asks arranged by growing price and bids sorted by decreasing (negative) price.There is no price specified in either the bid 1 or the ask 1; these are unrestricted "market orders" and are always given priority. Bids 1-8 are all matched by lower-priced asks, and asks 1-6 are all matched by higher-priced bids, although only the first 2 MWh of ask 6 is matched.Ask 7 and bids 9-10 cannot be matched.The clearing price, which is the average of the prices in ask 6 and bid 8, is 16, and the cleared volume is 27 MWh.

3.1.7 Balancing Markets

The Balancing Market's only responsibility is to exercise capacity controls on behalf of the agents in order to balance supply and demand in each time-slot. In this approach, the balancing utility arrives and imposes a penalty on the broker by charging the broker for the lost energy at a considerably higher price.

3.1.8 Distribution utility

The Distribution Utility (DU) is primarily responsible for 3 different operations.As its name implies, the first one distributes electricity to each client while also charging each broker distribution costs related to the energy sent through the grid.Additionally, DU is in charge of charging the Transmission Capacity Fees (TSF). A broker should pay these fees in exchange for their clients' participation in demand peaks. This implies that each broker will be required to pay for a percentage of the excess energy during times of peak demand. After 168 time-slots, TSF bill the three demand peaks with the highest prices (1 week of simulated time). In the current PowerTAC competition these fees are the main problem an agent faces when it tries to dominate in the retail market. The Distribution Utility's final duty is to post some default tariffs for the situations where no other broker has published any tariffs.

3.2 Successful PowerTAC Agents

PowerTAC has been held for 10 years as we mentioned before. Having such a long history of competitive activity suggests that there are already many innovative agent techniques that are being used. In this section, we decided to cite some PowerTAC agents implementations that have won the competition and are worth to be mentioned.

TUC TAC 2020
TUC TAC 2020 is an autonomous agent developed by a team led by professor Georgios Chalkiadakis and his students at the Technical University of Crete [3]. Its main strategy is based on the principle that acquiring half of the market share will give TUC TAC half of the total profits, but also only half of the inevitable transmission capacity fees. From the beginning, it became obvious that greedy strategies were not the right way to work, so the team turned to other methods such as decision trees combined with heuristic and non-heuristic algorithms. The basic principle of this technique came from the Lemonade Stand Game tournament [11]. The core objective of TUC TAC's strategy, which is to take half the available market share and leave the other half to competitors, is pretty similar to that one. By so doing, TUC TAC anticipates always earning the most money while splitting all commissions with the other agents.

The Wholesale Market module is TUC TAC's second and equally significant component. In the wholesale market's double auctions, its primary duties are to purchase and sell energy. However, for it to be successful, the best bids must be identified so that consumers do not have to use balancing utility to obtain their energy, in which case TUC-TAC would be paid more for every KWh that was not reserved by the agent. For that reason, TUC TAC implements a Monte Carlo Tree Search algorithm for bidding in the double auctions of the wholesale market, adapting it to this setting. The basic algorithm consists of four

stages: Selection, Expansion, Simulation, and Backpropagation. Monte Carlo Tree Search is an adaptable technique that can be used in a wide range of decision-making situations.

Mertacor
Another agent developed by a Greek team, and which won the PowerTAC 2019 competition, is Mertacor2019. Despite the agent continuously ranking among the best players in the competition, the team is not concentrating on the consumption predictor module. Instead, they calculated consumers' consumption with a simple classification method. Mertacor2020 finished second in PowerTAC 2020, and used Q-Learning techniques in order to maximize the profits from the retail market.

3.2.1 Predictors in PowerTAC agents

The predictor module of the PowerTAC agent is the focus of this thesis, as already mentioned. The brokers must make trades in multiple markets and to be successful, brokers must make many good predictions about future supply, demand, and prices. Clearing price prediction is an important part of the broker's wholesale market strategy because it helps the broker to make intelligent decisions when purchasing energy at low cost in a day-ahead market [12]. Below we present some of the most note-worthy agents that applied different kinds of prediction techniques in order to be effective.

TUC TAC 2020
TUC TAC had a predictor that estimated the net demand of the customers for the next 24 timeslots, based on the given weather forecast and the past net demand values. There was an attempt to improve this predictor in the thesis of Stefanos Kontos [13]. To this purpose, data was collected from the 2019 PowerTAC final games and the variables that were taken into consideration for forecasting were time slot, temperature, wind speed, wind direction, cloud cover, and net usage. At first, the predictor was approached as a regression problem, using linear regression and kernel regression but the experimental results were not so positive, suggesting that this algorithm was not appropriate for the problem. Consequently, Deep Learning methods were examined, such as Feed Forward Neural Networks, Recurrent Neural Networks, and k Nearest Neighbors. This approach was also not so desirable but had a more positive outcome since the experimental results of the Deep Learning Methods were better compared to the results of the Machine Learning techniques. Even if the project's goal wasn't entirely met, valuable insights and intuitions might still be drawn from the process. The research indicates that Neural Networks are superior to the other regression techniques [13].

AgentUDE17
From 2016 to 2019, Agent UDE [14] continuously finished in the top 3 spots and more specifically won the tournament in 2017 and 2018. Their forecasting methods relied basically on linear regression, with the main focus being on customer type and capabilities to predict future demands. In the wholesale market, AgentUDE predicts market trends regardless of weather conditions by tracking historical market data [15].

Crocodile Agent
Crocodile Agent [16] is another note-worthy agent, as it finishes repeatedly in the third place from 2018 until 2021. This agent implemented the Erev-Roth reinforced learning

method [17] for its bidding tactics in order to determine the best course of action for lowering the cost function. The main sub-module consists of (i) bidding strategies module, (ii) reward module and (iii) weighted randomizer module. Bidding strategies are picked based on the reward module during the optimization cycle, and the likelihood of their selection is determined by a weighted randomizer drawn from the probability distribution.

Vidyut Vanika

Vidyut Vanika [18] is 2021's year winner and has used many predictors to accomplish that achievement. First of all, it has a Customer Usage Predictor that uses a Neural Network with two hidden layers of size 7 each, and 10 epochs of training over the training data. The input data includes the weather report, time of day (0–23), and day of the week (1–7). During prediction, the weather forecast is used in place of the weather report to predict the usage for the next 24 hours. Moreover, Vidyut Vanika uses a Limit Price Predictor based on the previous work of Urieli and Stone 2014 [19] on Markov Decision Process based wholesale bidding strategy.

Chapter 4

Our Approach

This project's main goal was to develop a predictor module and improve the Monte Carlo Tree Search Algorithm. Numerous strategies were used to achieve the prediction, with the main ones being Regression and Deep Learning algorithms. In this chapter, we explain these different methods, why we chose them and we also provide a theoretical background of our Monte Carlo Tree Search implementations, as well as the modifications that we have made.

4.1 Predicting Methods

In this section we cite the predictive techniques that we utilized in order to find the most optimal and appropriate one for our agent. Four main methods and their theoretical background will be presented below.

4.1.1 Linear Regression

Linear regression is a typical method of predictive analysis [20]. It predicts the future of a target by using linear relationships between a dependent variable (target) and one or more independent variables (predictors). The prediction is based on the premise that the target and predictors have a dependent or causal relationship Y = aX +b, where a is the intercept and b is the slope of the line. Based on a given predictor variable, this equation can be used to predict the value of a target variable(s). Linear-regression models are straightforward and provide a basic mathematical method for generating predictions and can be used in a variety of corporate and academic study.

In our linear predictor, we used the values of 'Execution Price' and the values of 'Timeslots' in order to train the data-set and make predictions. We took 1200 values from each variable for training and the rest 253 for testing.

4.1.2 Polynomial Regression

Another frequently used algorithm for predictive analysis is polynomial regression, which is a form of linear regression, as is consists of multiple linear regressions, that estimates the relationship as a n^{th} degree polynomial.
The general equation of polynomial regression is written in the following form:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + ... + +\beta_n x^n \tag{4.1}$$

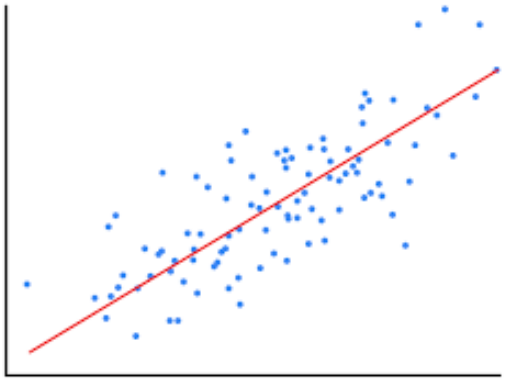

Figure 4.1: Linear Regression example.

- β_1 = linear effect parameter.
- β_2 = quadratic effect parameter.
- β_o = a constant parameter, which is determined according to the polynomial function when x = O.

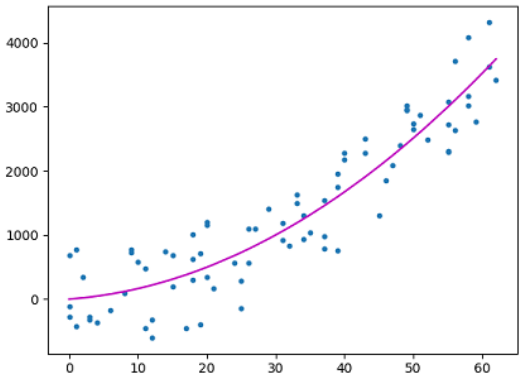

Figure 4.2: Polynomial Regression example.

Linear regression requires the relation between the dependent variable and the independent variable to be linear but in this regression technique, the best fit line is not a straight line,it is rather a curve that fits into the data points.

Any climatic data is expected to behave in a nonlinear manner, so it will be too difficult to visualize or predict the data using the linear regression model. Polynomial regression is expected to have a lower error rate.

In our polynomial predictor, we worked the same way we did with the Linear one. We used the values of 'Execution Price' and the values of 'Timeslots' in order to train the data-set and make predictions. We took 1200 values from each variable for training and the rest 253 for testing.

4.1.3 RNN and LSTM

Recurrent neural networks (RNNs) [21] are effective learning models that allow temporal information to be conveyed through several time steps by incorporating recurrent connections on the hidden layers. As a result, they may be able to model time-dependent aspects in sequential series. Theoretically, RNN can handle any length of sequence data. However, as more layers using certain activation functions are added to neural networks, the gradients of the loss function approaches zero, making the network hard to train. This is called the gradient vanishing problem [5] and it affects RNNs.

To tackle the vanishing gradient problem, gated recurrent neural networks (GRNNs), such as the long short-term memory (LSTM) and gated recurrent unit (GRU), use carefully constructed recurrent units with set unit weight [22]. The most popular and successful GRNN model is long short-term memory (LSTM). It adds the memory cell, a computational unit that substitutes typical nodes in the network's hidden layer, allowing it to overcome the training issues that plagued earlier recurrent networks. The usual LSTM architecture has three gates and a hidden state [22].

RNN algorithm
Recurrent neural network (RNN) [21] is an extension of conventional feedforward neural network with dynamic input. An RNN is characterized by introducing recurrent connections on the hidden layers,which makes modeling sequence data possible. The RNN uses index t to represent the different positions in the input sequence and suppose a hidden state h_t to represent the state of the system at time t. At time t, the RNN accepts input x_t, and the recurrent hidden state h_t is activated by the former hidden state h_{t-1} and current input x_t, and updated in real time by a nonlinear activation function:

$$h_t = tan(h)(W[x_t, h_{t-1}] + b) = tan(h)([W_x x_t + W_h h_{t-1}] + b) \tag{4.2}$$

where W_x is the weight matrix for the input x_t, W_h is the weight matrix for the recurrent input h_{t-1}, and the b term is the bias vector. By fitting the parameters W and b, the sequence data can be learned well.

Typical LSTM model
The RNN can be considered as a neural network that passes through time, which means the length of time sequence is the depth of RNN network. Once length of time sequence is very long, the problems of vanishing and exploding gradients occur when back propagating errors across many time steps. The LSTM model is proposed to overcome the problem of vanishing gradient from RNN by introducing gating mechanism in recurrent units to control how information flows. In addition to three multiplicative gates (forget gate f_t, input gate i_t, output gate o_t) and hidden state h_t, the recurrent unit includes a memory state S_t, in which S_t is used to help maintaining long-term memory. The gates are sigmoid units that take activation from the current input x_t as well as from the hidden layer at the previous time step. At time t, new memory state S_t is formed by its self-connected recurrent edge S_{t-1} and G_t, the Ground-truth Trajectory. The associated equations are as follows:

$$f_t = \sigma(W_f \bullet [x_t, h_{t-1}] + b_f)$$
$$i_t = \sigma(W_i \bullet [x_t, h_{t-1}] + b_i)$$
$$o_t = \sigma(W_o \bullet [x_t, h_{t-1}] + b_o)$$
$$G_t = \sigma(W_g \bullet [x_t, h_{t-1}] + b_g)$$
$$S_t = f_t * S_{t-1} + i_t * G_t$$
$$h_t = o_t * tan(h)(S_t)$$

where • denotes matrix multiplication and * is pointwise multiplication, σ is a sigmoid function, W_f, W_i,W_o,W_g are weight matrices of f_t,i_t,o_t,G_t and b_f,b_i,b_o,b_g are the corresponding bias terms [23].

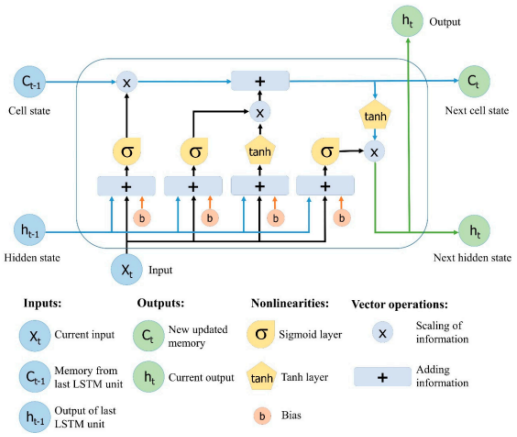

Figure 4.3: Typical LSTM model.

4.2 Sockets

In a network, a socket is a communication's connection point that may be named and addressed. Socket programming demonstrates how to create communication channels between remote and local processes using socket APIs. Socket-using processes can be on the same system or on other systems on different networks. Sockets can be used in both single-user and network applications. Sockets make it possible to share data between processes on the same system or across a network, transfer work to the most efficient machine, and gain quick access to centralized data. TCP/network IP's standard is socket application program interfaces (APIs). Socket APIs are supported by a wide range of operating systems. Multiple transport and networking protocols are supported by i5/OS sockets.Thread safety is provided via socket system and socket network functions. In this thesis, we used sockets to connect the Python predictors to our Java-based agent TUC TAC.

4.3 The Wholesale Market Module

The second basic part of this thesis was the improvement of the Wholesale Market Module of 2021 TUC-TAC. Its primary duties are to purchase and sell energy during the Wholesale's Market double auction. Finding the finest bids is necessary for it to be effective, so

that customers won't have to turn to Balancing Utility to obtain their energy. The Balancing utility will charge more for every single KWh that was not reserved by TUC-TAC if this module fails to obtain the energy that the subscribing consumers need, leading to many fines. The aforementioned goal is achieved by implementing the Monte Carlo Tree Search algorithm.

4.3.1 Monte Carlo Tree Search

Monte Carlo tree search is a probabilistic search algorithm that has a unique decision making technique and it is very efficient in solving the game tree,a graph representing all possible game states within a game. It is based on random sampling of game states, it does not need to brute force its way out of each possibility. Additionally, it is not necessary for us to create effective heuristics or evaluations. The main stages of the Monte Carlo Tree Search consists of four stages:

- **Selection**: The method chooses a child node in this initial step by starting with a root node and choosing the node with the highest win rate. Additionally, we want to guarantee that each node has an equal chance. The idea is to keep selecting optimal child nodes until we reach the leaf node of the tree. The algorithm achieves that by using the UCT (Upper Confidence Bound) formula:

$$\frac{w_i}{n_i} + c\sqrt{\frac{\ln t}{n_i}} \tag{4.3}$$

  Where w_i is the number of wins after the i-th move, n_i is the number of simulations after the i^{th} move, c is the exploration parameter and t is the total number of simulations for the parent node.

- **Expansion**: When UCT is no longer able to be used to locate the successor node, the game tree is expanded by the addition of all potential states from the leaf node.

- **Simulation**: After Expansion, the algorithm randomly selects a child node, simulating a randomized game from that node until it reaches the game's final state.

- **Backpropagation**: When the game has ended, the algorithm assesses the current situation to determine who has won. All visited nodes have their visit scores increased as it moves up to the root. It also updates win score for each node if the player for that position has won the playout.

Monte Carlo Tree Search algorithm keeps repeating these four phases until some fixed number of iterations or some fixed amount of time. With this method, we estimate each node's winning score based on random moves. Therefore, the estimate gets more trustworthy when more iterations are executed. The algorithm estimations will be less precise at the beginning of a search and get better over time. A graphic representation of the algorithm is shown at figure 4.4.

4.3.2 Monte Carlo Tree Search in TUC TAC

The main algorithm that was implemented in this module was a variant of the Monte Carlo Tree Search method used by TUC TAC 2020 which in turn was a variant of the

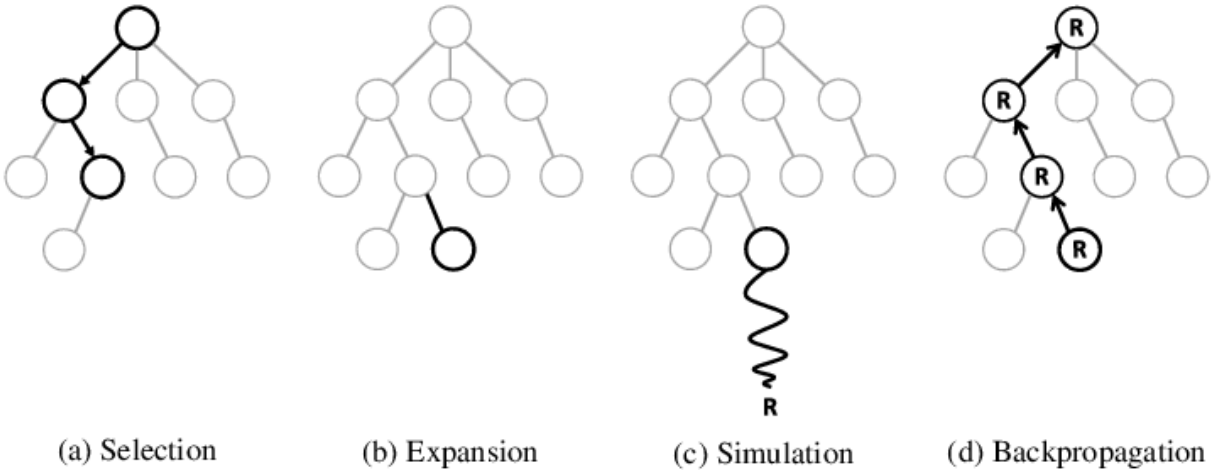

Figure 4.4: Phases of Monte Carlo Tree Search Algorithm.

method developed by Chowdhury[24]. In general, Monte Carlo Tree Search (MCTS) is a search method used in artificial intelligence. It is a probabilistic and heuristic-driven search method that blends traditional tree search implementations with principles from machine learning and reinforcement learning.
The double auction of the wholesale market, in the PowerTAC competition is a complex action-space that necessitates quick and exact actions in order to be profitable.In order to quickly navigate through massive decision trees and determine the optimal course of action, MCTS algorithm was chosen.
We can see the pseudo-code for our bidding technique in Algorithm 1. The root node and its children are initially generated. The values for each child are created by adding the anticipated limit price and a standard observed variance. Each child represents a distinct bid in the double auction. The limit price predictor that existed in a previous TUC TAC implementation is now replaced by an effective predicting algorithm. The MCTS algorithm then proceeds to iterate over the tree by growing the nodes and simulating a few auction results. Here we added the action of "HALF BID" that did not exist in the previous implementation of TUC TAC 2020. In this action, our broker instead of bidding for the whole price, make half the amount of the bid. By adding this action, we make our agent more competitive as he enters more bids than before and therefore buys amounts of energy at a better price. All nodes that were a part of the current path are updated after each iteration, and we preserve data on visit count and average unit cost so that we can later calculate the UCT value for each node. Furthermore, we fine-tuned the parameters of the algorithm to work better with the new predictor and tested it in various games to see the results.

Algorithm 1 Calculate Bids for wholesale Market using MCTS

```
energyToBuy = neededMWH(t)
for i < NUMBER_OF_ITERATIONS do
    curNode = root
    curNode.GenerateKids()
    while energyToBuy > 0 do
        if curNode.hasUnexploredKids() then
            curNode ≥ GetRandomUnvisitedChild()
            while energyToBuy > 0 do
                if action = BID then
                    limitPrice = predictions[t]
                    clearingPrice = GetRandomGaussianNumber()
                    if limitPrice > clearingPrice then
                        Csim = energyToBuy · clearingPrice
                    end if
                else if action = HALFBID then
                    limitPrice = predictions[t]
                    clearingPrice = GetRandomGaussianNumber()
                    if limitPrice > clearingPrice then
                        Csim = energyToBuy · clearingPrice/2
                    end if
                end if
            end while
            break
        else
            curNode ≥ GetBestUCTChild()
            energyToBuy ≥ Simulate(curNode)
        end if
    end while
    C_avg = C_sim/energyToBuy
    Bacpropagate(C_avg)
end for
bid = GetBestRootChild().bid
BidInAuction(bid, t)
```

Chapter 5

Experimental Results

In this section, we present the results originating from the experiments that we deducted. The results concern both the predictive methods and the Monte Carlo Tree Search algorithm. The type of algorithm utilized will be used to show and explain every experiment's outcome. The following step is to compare the outcomes of each algorithm. Tables about the errors of every predictor are presented in order to demonstrate which technique fits our problem better. Consequently, we present the results from the Monte Carlo Tree Search Algorithm modification, compared to other agents. In that way, we will be able to verify weather our modified MCTS version had a positive impact on our agent's performance.

5.1 Preparation and Early Development

5.1.1 Dataset Construction

The PowerTAC competition is a very realistic simulator that uses real life weather data from selected cities around the globe. More specifically, the simulation is made up of many costumer models that vary in key areas. There are three basic groups of clients based on the power type: consumers, who only buy energy from our broker, producers, who only sell energy to our broker, and prosumers that are a hybrid of the first two.

In addition to the data from the customer model, the simulator depicts us how much energy each client uses throughout each time window. Moreover, weather forecasts and weather reports are offered for each time slot, and we also store this information because we view it as important information for building models. The features that we used for our predictors are the following:

- **Timeslot**: A timeslot instance describes an interval of time (slot) for which power may be traded in the wholesale market.
- **Hour**: The current timeslot's hour.
- **Day**: The current timeslot's day.
- **Month**: The current timeslot's month.
- **Year**: The current timeslot's year.

- **Temperature**: The current timeslot's temperature.
- **Wind Speed**: The current timeslot's wind speed.
- **Wind Direction**: The current timeslot's wind direction.
- **Cloud Cover**: The current timeslot's cloud cover.
- **Execution Price**: The current timeslot's clearing price of the trade.
- **Execution MWh**: The current timeslot's traded quantity in MWh of the specified product.

5.1.2 Dataset Collection

In order to have solid results we test the effectiveness of our predictors using data from the 2021 PowerTAC finals, while also dividing them into two main categories based on their difficulty level. The first one is the "hard" games, where due to extreme weather phenomena there was a huge net demand, leading to a rise in the price of kWh. The other category is the "easy" games, where the weather is good in general so the price of kWh fluctuates to normal levels. The results were saved in a ".state" file, so I created a parser in Python that extracted all the information that we needed and created 10 JSON files, 5 from each category.

5.2 Predictor Results

5.2.1 Easy Games

In this section, we present the results of our predictors for the "easy" datasets.

Predictor 1 (Linear Regression) Results

In Linear Regression we have illustrated 4 plots from 4 different days. The results are shown below:

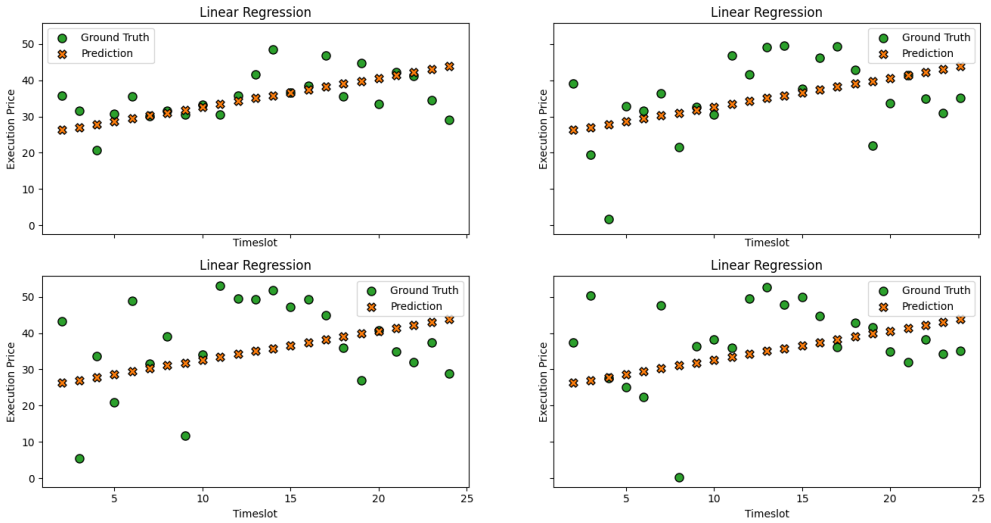

Figure 5.1: Linear Regression with the first "easy" dataset. The four smaller plots compare the ground truth and the prediction results from different days. The green dot shows the ground truth while the x depicts the linear model's predictions.

In figure 5.1 we observe our first try for prediction. Besides the first graph which has better results compared to the other three ones, a great deviation between the predicted values and the ground truth exists.

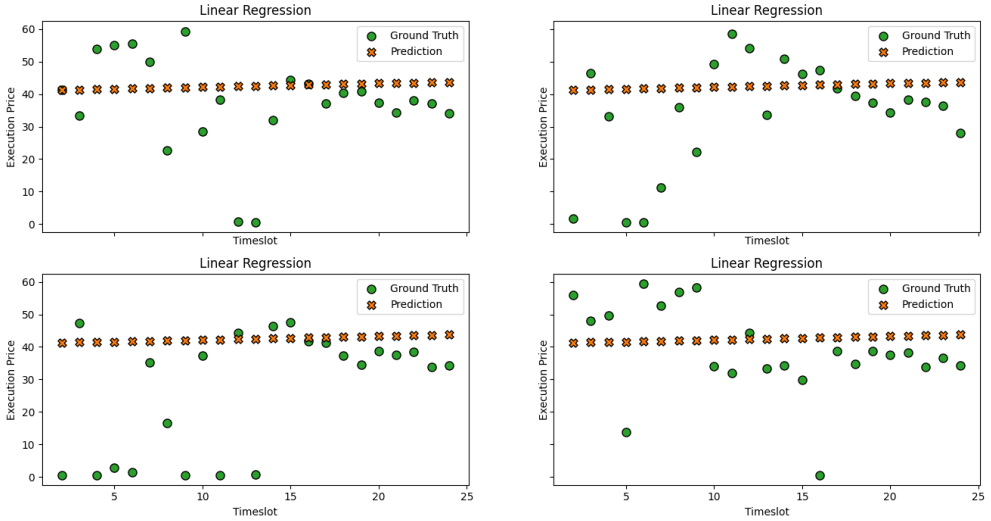

Figure 5.2: Linear Regression with the second "easy" dataset. The four smaller plots compare the ground truth and the prediction results from different days. The green dot shows the ground truth while the x depicts the linear model's predictions.

We can see a significant difference between the projected values and the actual values in figure 5.2. Linear regression predicts a straight line at approximately 40 euros, while the ground truths have a range from 0 to 60 euros.

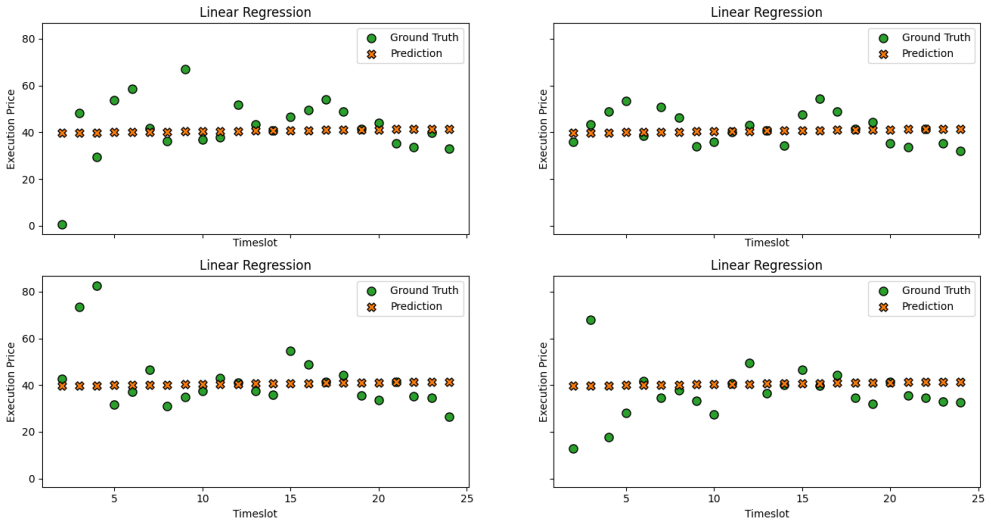

Figure 5.3: Linear Regression with the third "easy" dataset. The four smaller plots compare the ground truth and the prediction results from different days. The green dot shows the ground truth while the x depicts the linear model's predictions.

In figure 5.3 we notice that the two upper graphs show better results compared to the other two graphs, as the prediction line and the ground truth line are more compatible. However, there is still a significant difference between the anticipated values and the actual ones.

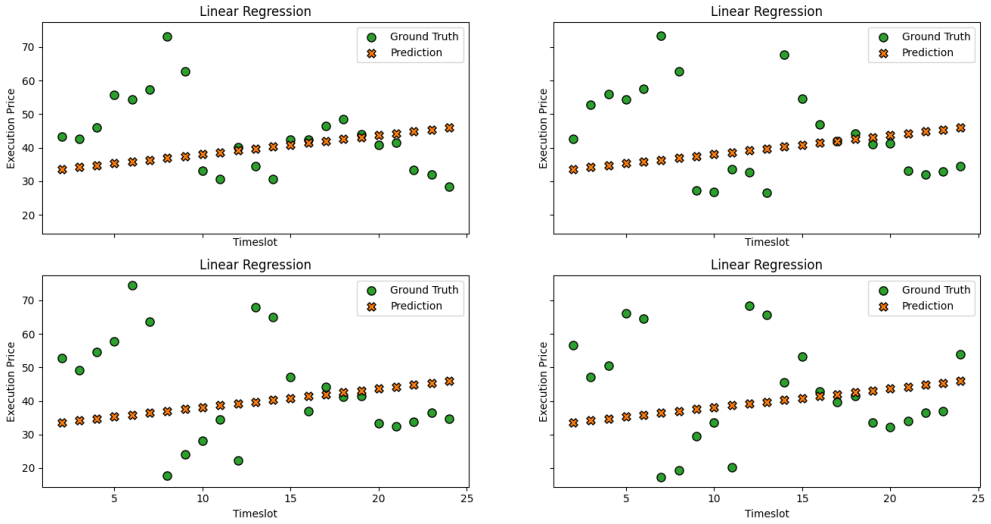

Figure 5.4: Linear Regression with the fourth "easy" dataset. The four smaller plots compare the ground truth and the prediction results from different days. The green dot shows the ground truth while the x depicts the linear model's predictions.

As we can observe from the image 5.4 also, linear regression is not a very effective way of prediction, as the predicted values are not coinciding with the actual ones. Our prediction line ranges from 30 to 45 euros, while the ground truth line ranges from 20 to 75 euros.

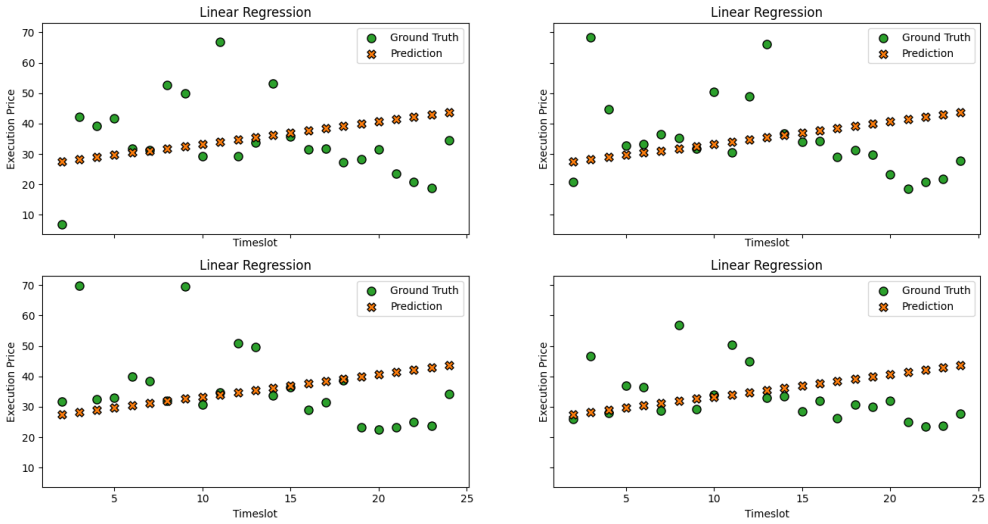

Figure 5.5: Linear Regression with the fifth "easy" dataset. The four smaller plots compare the ground truth and the prediction results from different days. The green dot shows the ground truth while the x depicts the linear model's predictions.

Figure 5.5 illustrates a great deviation between the predicted values and the ground truth. A significant difference occurs between the projected values and the actual values, with the exception of the final graph, which performs better than the other three.

Linear Regression is not a very effective predictive method and this happens for multiple reasons. Firstly, it demands the relationship between our dependent and predictor variables to be linear, however, in our case and in general, the behavior of any climate data is set to be a nonlinear way. In addition, linear regression assumes there is no multicollinearity between the variables and this makes the model unstable. For the reasons above-mentioned we had to advance to a more functional method for our predictions.

Predictor 2 (Polynomial Regression) Results

We now present results for each figure has 3 subplots, one for each exponent of our polynomial experiments. The first one has exponent n = 4, the second one n = 6, and the third one n = 8. We present them in this way, to ease their comparisons and decide which exponent fits better every time.

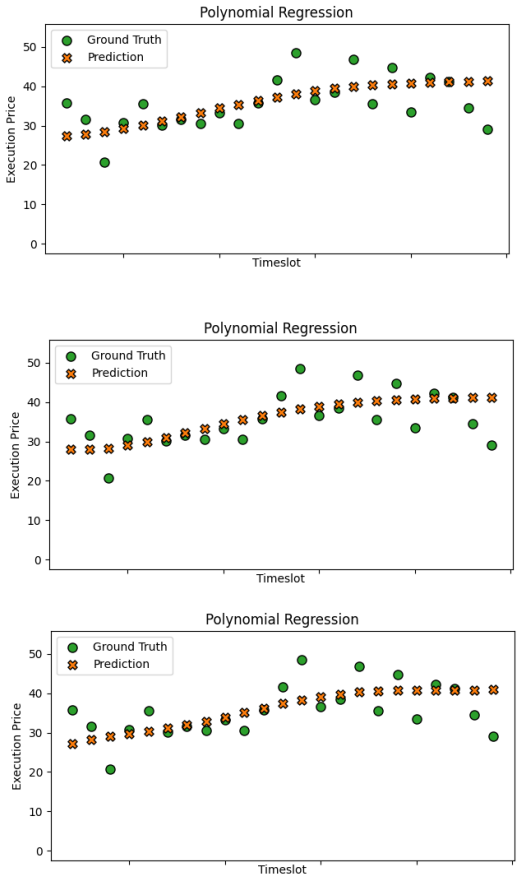

Figure 5.6: Polynomial Regression with the first "easy" dataset. The three smaller plots compare the ground truth and the prediction results for the three different exponents. The green dot shows the ground truth while the x depicts the polynomial model's predictions.

In figure 5.6 we can see that we have better results compared to the linear regression predictor, however, the result is not the desirable one, as the actual values differ from our predictions. For instance, the error of the predicted values is around 10 euros. Between the three exponents, the most efficient one is n = 8, with a slight lead against the other two.

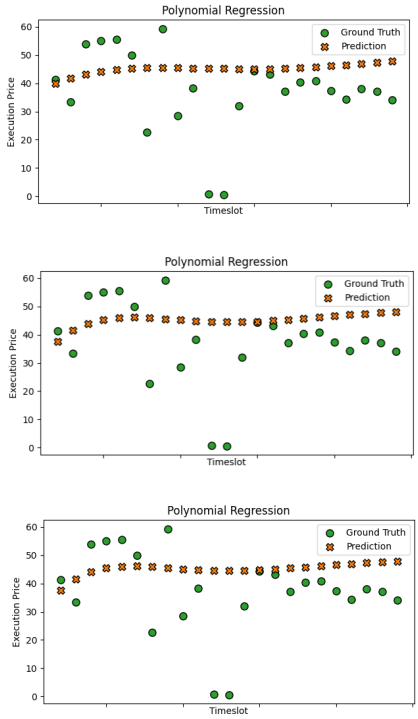

Figure 5.7: Polynomial Regression with the second "easy" dataset. The three smaller plots compare the ground truth and the prediction results for the three different exponents. The green dot shows the ground truth while the x depicts the polynomial model's predictions.

In figure 5.7 we observe that we have worse results than the linear regression predictor ones. The error of the predicted values is around 20 euros. Between the three exponents, the most efficient one is n = 8, with a slight lead against the other two.

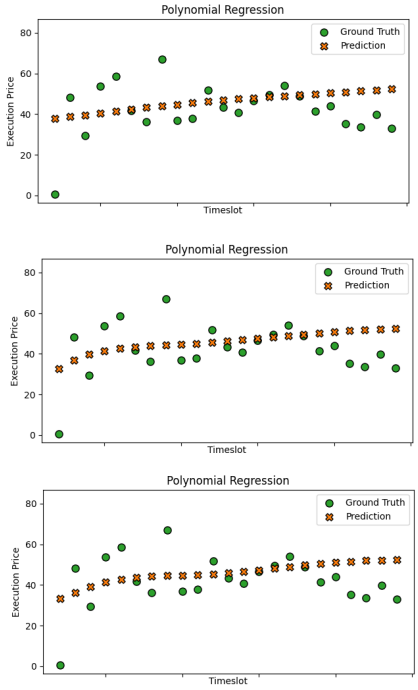

Figure 5.8: Polynomial Regression with the third "easy" dataset. The three smaller plots compare the ground truth and the prediction results for the three different exponents. The green dot shows the ground truth while the x depicts the polynomial model's predictions.

In figure 5.8 we can see that we have again worse results compared to the linear regression predictor, as the ground truth values differ from our predictions. For instance, here the error of the predicted values is around 40 euros. Between the three exponents, the most efficient one is n = 4.

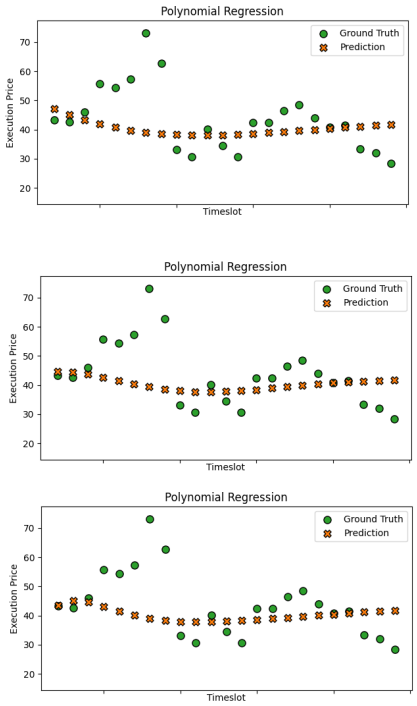

Figure 5.9: Polynomial Regression with the fourth "easy" dataset. The three smaller plots compare the ground truth and the prediction results for the three different exponents. The green dot shows the ground truth while the x depicts the polynomial model's predictions.

As we can see in Figure 5.9, the predicted values have smaller differences with the ground truth compared to the linear case. The error of the predicted values is around 40 euros. Between the three exponents, the most efficient one is n = 4, with a slight lead against the others.

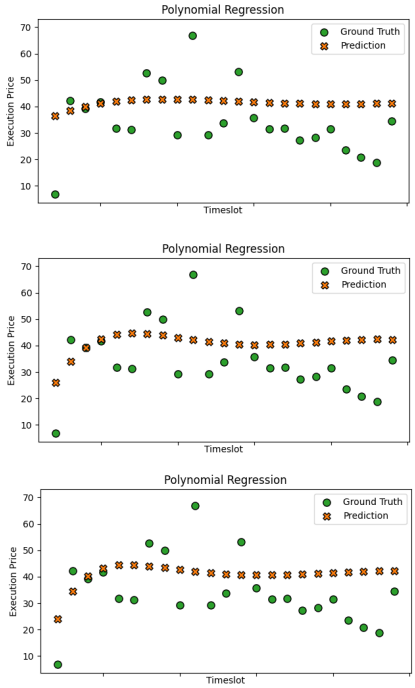

Figure 5.10: Polynomial Regression with the fifth "easy" dataset. The three smaller plots compare the ground truth and the prediction results for the three different exponents. The green dot shows the ground truth while the x depicts the polynomial model's predictions.

In figure 5.10 we can see that we have almost the same results as the linear regression predictor. Still, the actual values differ from our predictions, as the error of the predicted values is around 30 euros. Between the three exponents, the most efficient one is n = 4.

It seems that polynomial regression cannot deal with big data. The polynomial regression's inability to handle extremely sensitive data, such as weather, should be noted as a significant downside. Original data curves cannot even be kept up with by high orders. As a result, the polynomial model will not consider many samplers (points), which will have an impact on how the expected weather results are projected [25].

Predictor 3 (LSTM 1) Results

In order to implement the predictor we constructed a Recurrent NN using the LSTM algorithm with specific architecture:

- 4x Hidden Layers with 32 Neurons each
- Dropout rate: 0.2
- Batch size: 16
- Epochs: 12
- Loss function: Mean Square Error (MSE)
- Optimizer: Adam

We present 5 different plots, one for each "easy" dataset. They present 24 timeslots of the game, thus we have 24 predicted values and 24 ground truth points.

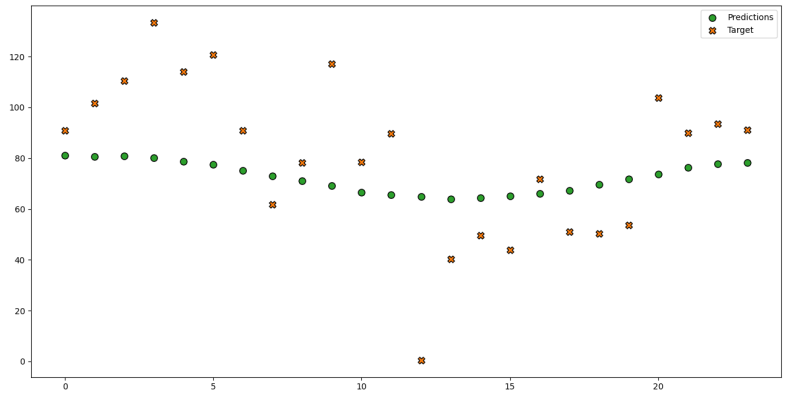

Figure 5.11: LSTM 1 with the first "easy" dataset. The green dot shows the prediction while the x depicts the model's ground truth.

In figure 5.11 we can observe a deviation between the predicted values and the actual ones. The predicted line ranges from 70 to 80 euros, while the ground truth line ranges from 0 to 130 euros.

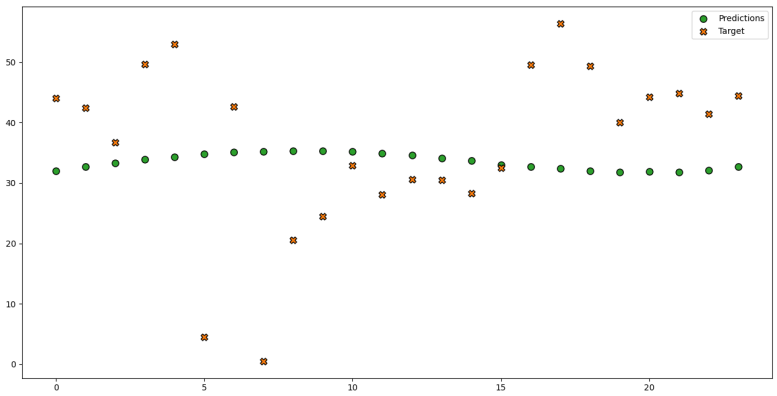

Figure 5.12: LSTM 1 with the second "easy" dataset. The green dot shows the prediction while the x depicts the model's ground truth.

In figure 5.12 the results are obviously better than the previous dataset. Here the line of the predicted values ranges from 30 to 35 euros and the line of the real values varies from 0 to 60 euros.

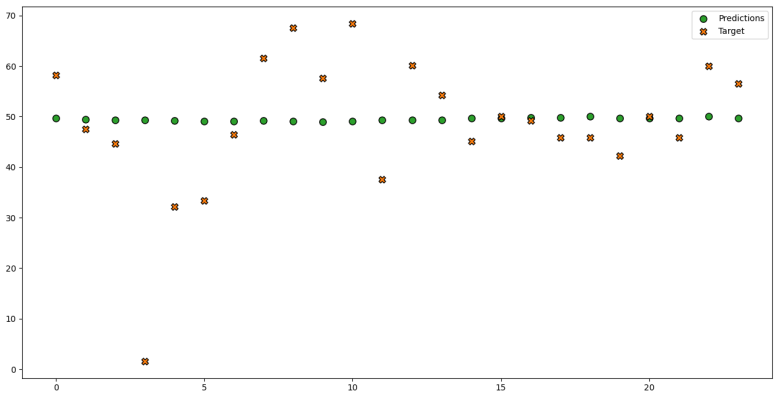

Figure 5.13: LSTM 1 with the third "easy" dataset. The green dot shows the prediction while the x depicts the model's ground truth.

Figure 5.13 depicts the least effective prediction of our first LSTM method. The deviation between the two lines is great and the result is not the desired one. The error of the predicted values ,in this figure, is around 50 euros.

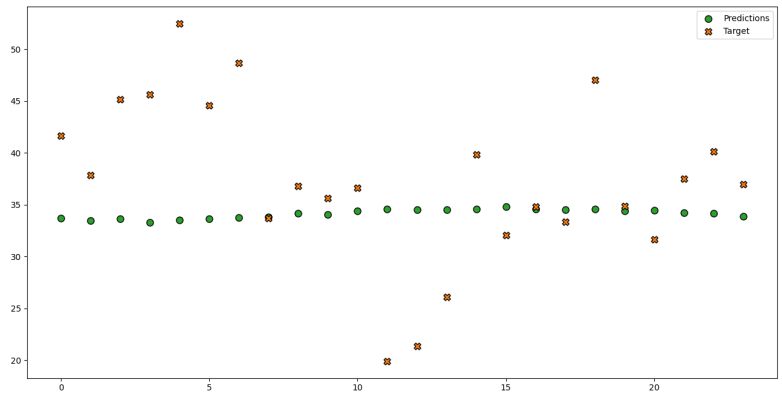

Figure 5.14: LSTM 1 with the fourth "easy" dataset. The green dot shows the prediction while the x depicts the model's ground truth.

In figure 5.14 we see the best prediction so far for the first LSTM model. We can notice that the actual values and the predicted ones are quite close, with some even matching exactly with their corresponding values.

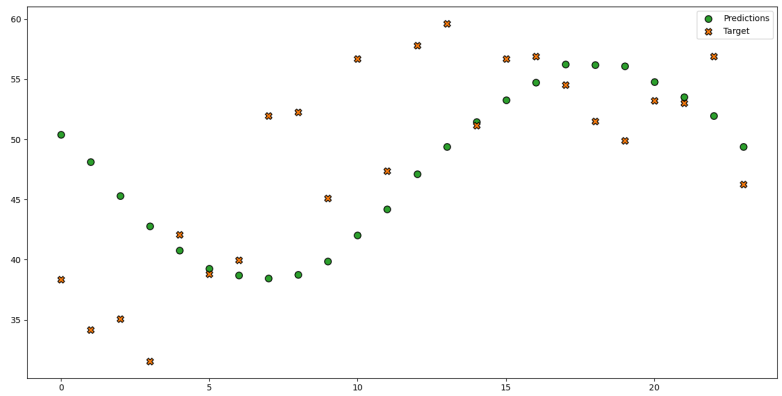

Figure 5.15: LSTM 1 with the fifth "easy" dataset. The green dot shows the prediction while the x depicts the model's ground truth.

Figure 5.15 also represents a descent try of prediction as there is not much divergence between the predicted values and the ground-truth ones. The error of the predicted values is again around 50 euros.

Our first LSTM predictor is not effective at all, as we understand by looking the figures above. The best prediction was made with the fourth dataset, however, our predicted values had great divergence from the ground truth. Linear Regression and Polynomial Regression seem to have better results than this method, however, we have to see the metrics of each one and compare them for the most accurate conclusion.

Predictor 4 (LSTM 2) Results

For the second LSTM method, the models learn to predict 24 hours into the future, given 24 hours of the past. The blue line depicts the 24 inputs that we gave our predictor, the green labels are the future actual values for the next 24 hours and the orange marks are our 24 predictions. The main difference between the second is that the LSTM 1 model predicts the entire output sequence in a single step, while LSTM 2 decomposes the prediction into individual time steps. In that way, the output of each model can be fed back into itself at each stage, and predictions can be generated based on the results of the step before, as in Generating Sequences With Recurrent Neural Networks [26]. This type of model has the advantage that it can be set up to produce output with a varying length. LSTM's 2 architecture is :

- Hidden Layers with 32 Neurons each
- Batch size: 32
- Epochs: 20
- Loss function: Mean Square Error (MSE)
- Optimizer: Adam

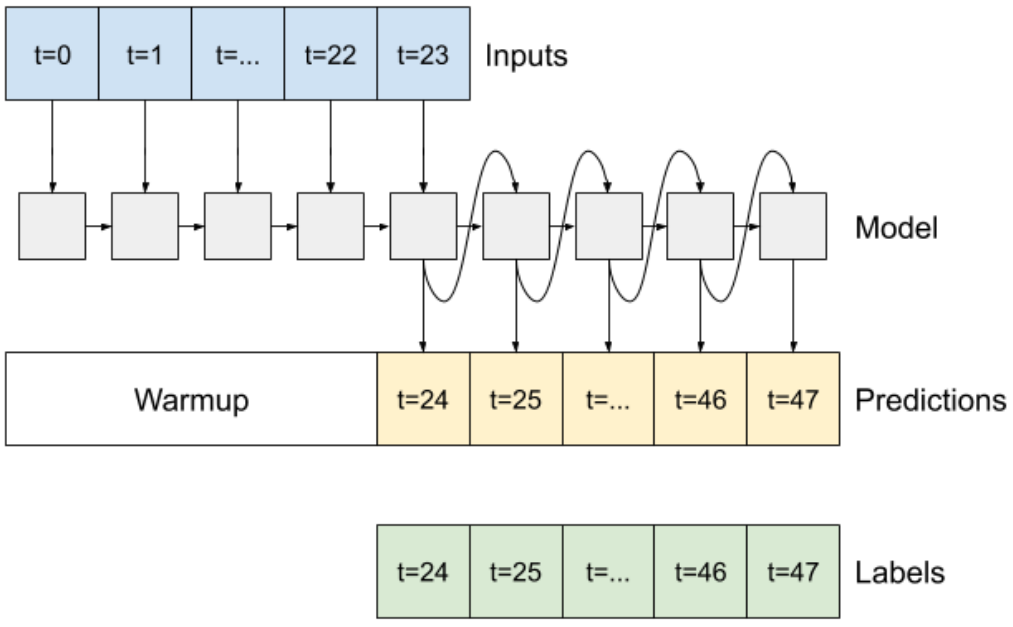

Figure 5.16: Auto-regressive feedback loop of the LSTM 2 method. Ideas were modified from this tutorial[27]

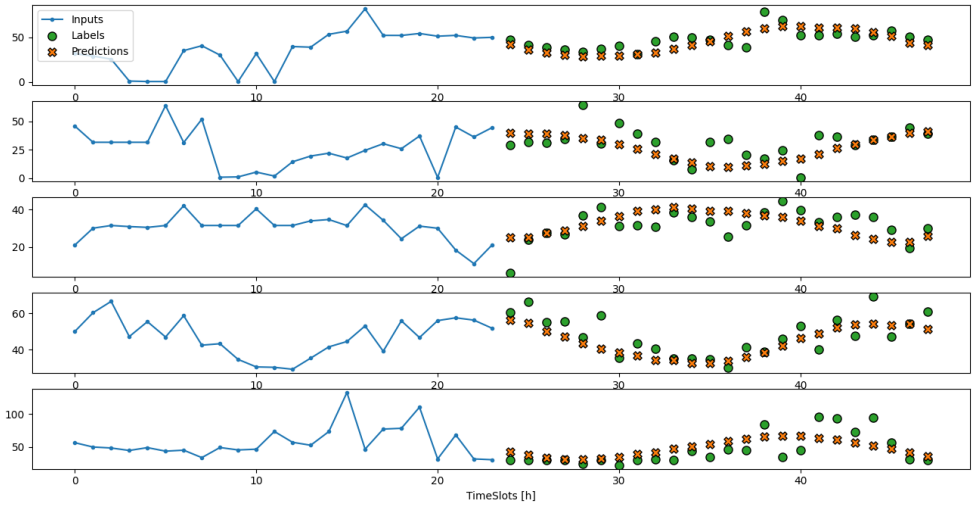

Figure 5.17: LSTM 2 with the first "easy" dataset. The green dot shows the ground truth, the x depicts the model's predictions and the blue line represents our input from the past 24 hours. The four smaller plots compare the ground truth and the prediction results from different days.

In figure 5.17 we observe a great result as the 24 ground truths and our predictions coincide and the error of the predicted values is less than 5 euros.

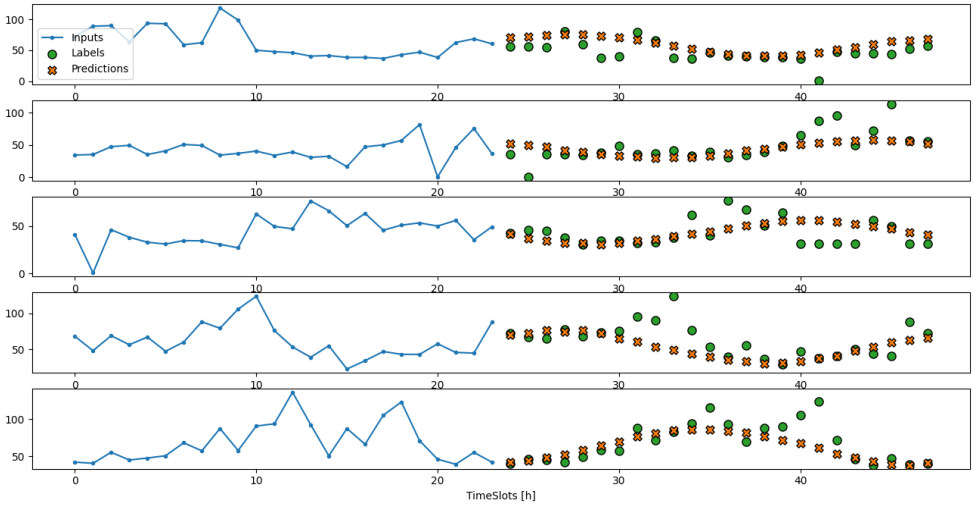

Figure 5.18: LSTM 2 with the second "easy" dataset. The green dot shows the ground truth, the x depicts the model's predictions and the blue line represents our input from the past 24 hours. The four smaller plots compare the ground truth and the prediction results from different days.

The 24 ground truths and our predictions are in perfect agreement, and the inaccuracy of the predicted values is less than 5 euros, as shown in figure 5.18.

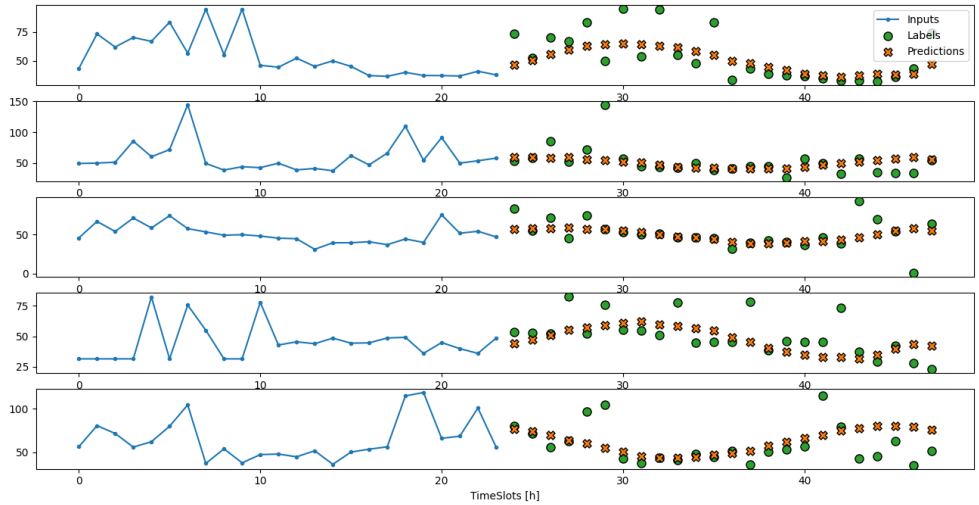

Figure 5.19: LSTM 2 with the third "easy" dataset. The green dot shows the ground truth, the x depicts the model's predictions and the blue line represents our input from the past 24 hours. The four smaller plots compare the ground truth and the prediction results from different days.

Another very satisfying execution of our predictor is shown in figure 5.19. The predictions and the actual values are very close to each other and the prediction error is approximately 4 euros.

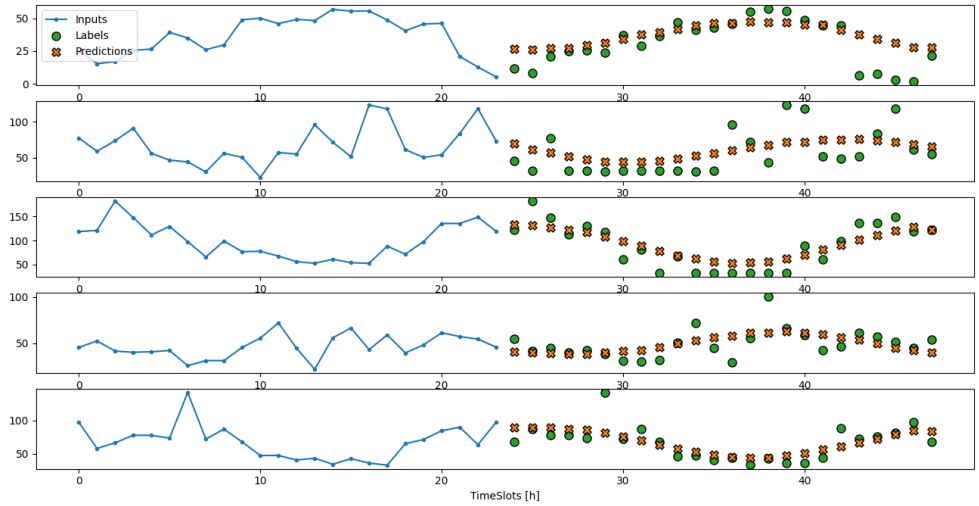

Figure 5.20: LSTM 2 with the fourth "easy" dataset. The green dot shows the ground truth, the x depicts the model's predictions and the blue line represents our input from the past 24 hours. The four smaller plots compare the ground truth and the prediction results from different days.

We can see a fantastic result in figure 5.20, where the 24 ground truths and our predictions

match up perfectly and the predicted values are off by less than 5 euros.

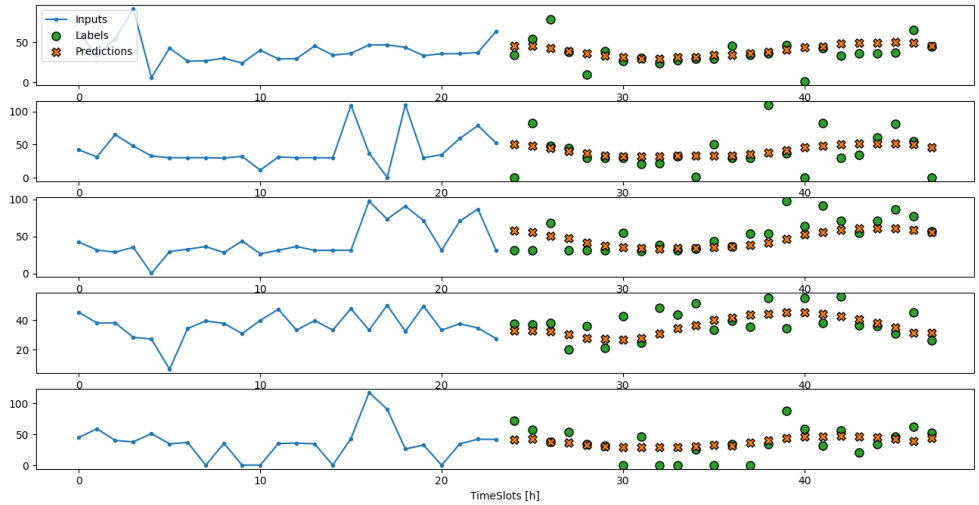

Figure 5.21: LSTM 2 with the fifth "easy" dataset. The green dot shows the ground truth, the x depicts the model's predictions and the blue line represents our input from the past 24 hours. The four smaller plots compare the ground truth and the prediction results from different days.

Figure 5.21 represents the best performance of our agent, since the predicted error is below 3 euros.

The figures above prove that our second LSTM predictor is a very accurate one, since ground truths and predicted values are almost in the same position. The deviation between them is very low and we have finally come to a desirable result at least for the "easy" games in our settings.

5.2.2 Hard Games

In this section we present the results of the predictors for the "hard" datasets.

Predictor 1 (Linear Regression) Results

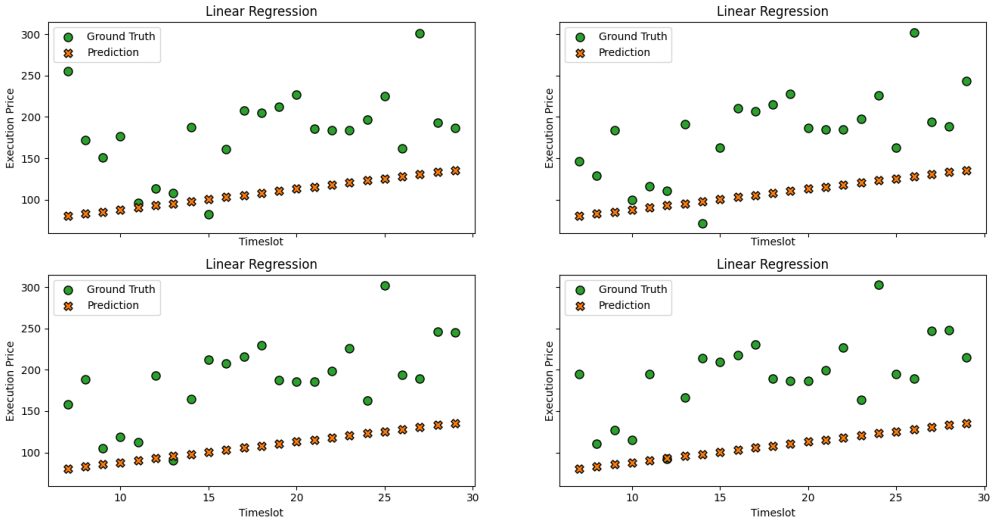

Figure 5.22: Linear Regression with the first "hard" dataset. The four smaller plots compare the ground truth and the prediction results from different days. The green dot shows the ground truth while the x depicts the linear model's predictions.

Figure 5.22 illustrates a great deviation between the predicted values and the ground truth. A significant difference occurs between the projected values and the actual values, meaning that in the first hard dataset our linear predictor did not perform as we expected.

Here in figure 5.23, we perceive the best outcome of our linear predictor with the hard datasets. There exist a deviation between the ground truth and our predictions but it is quite small compared to the other datasets.

Here at figure 5.24 we notice the second best try of our linear predictor. The deviation between the predicted values and the real ones is small compared to the other datasets. The error of the predicted values is around 40 euros.

Figure 5.25 also represents a quite big divergence between the predicted values and the actual ones. The first line lies at 200 euros while the other ranges from 150 to 325 euros.

Figure 5.26 shows the greatest deviation of our try with the linear predictor. The predicted values and the actual ones do not coincide at all, meaning that our predictor is not effective.

By watching the graphs above, we conclude that Linear Regression is not a compelling method for predictions when it comes to "hard" datasets. Actual values have great di-

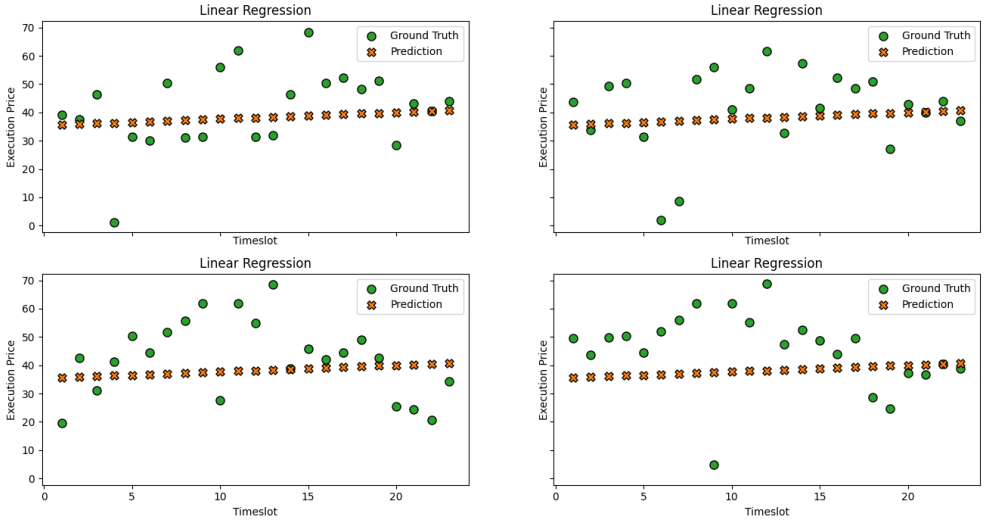

Figure 5.23: Linear Regression with the second "hard" dataset. The four smaller plots compare the ground truth and the prediction results from different days. The green dot shows the ground truth while the x depicts the linear model's predictions.

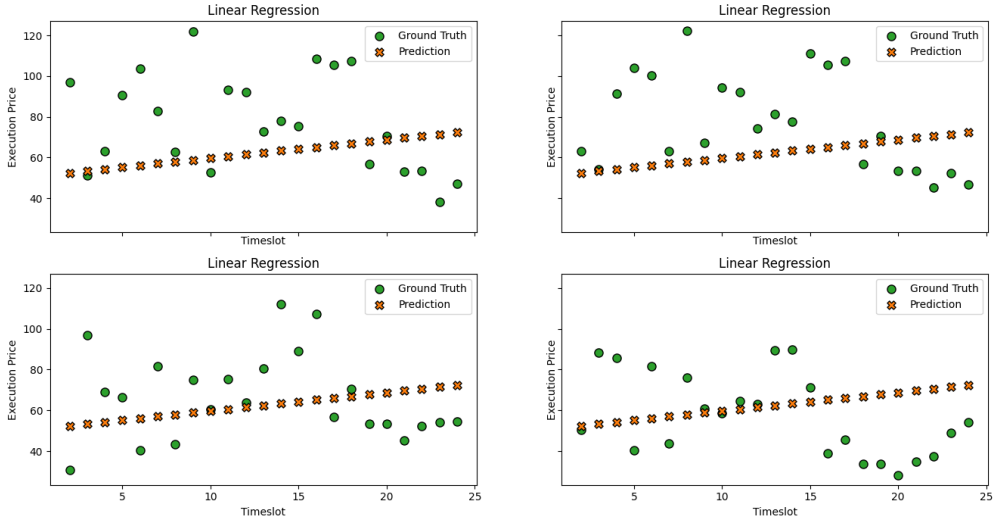

Figure 5.24: Linear Regression with the third "hard" dataset. The four smaller plots compare the ground truth and the prediction results from different days. The green dot shows the ground truth while the x depicts the linear model's predictions.

vergence from the predicted ones, showcasing the incompetence of this method for this problem.

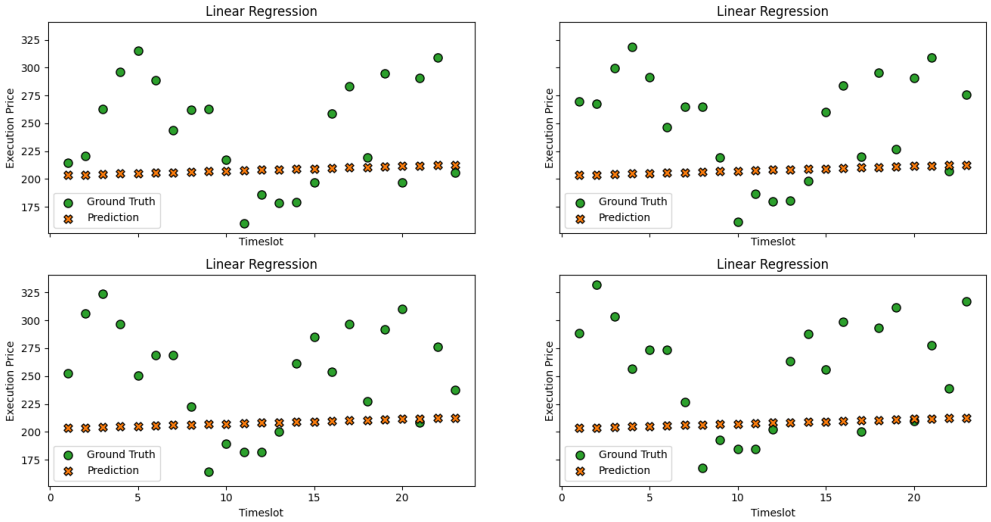

Figure 5.25: Linear Regression with the fourth "hard" dataset. The four smaller plots compare the ground truth and the prediction results from different days. The green dot shows the ground truth while the x depicts the linear model's predictions.

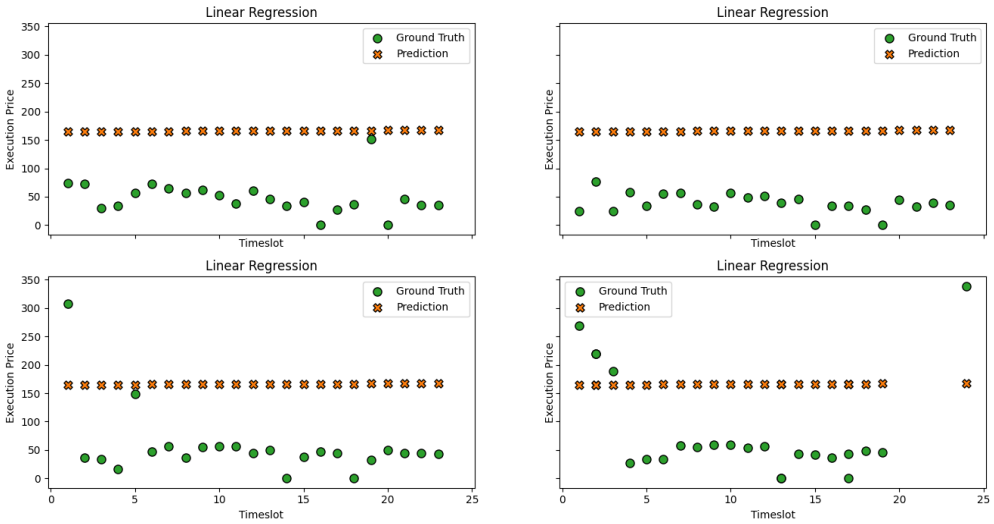

Figure 5.26: Linear Regression with the fifth "hard" dataset. The four smaller plots compare the ground truth and the prediction results from different days. The green dot shows the ground truth while the x depicts the linear model's predictions.

Predictor 2 (Polynomial Regression) Results

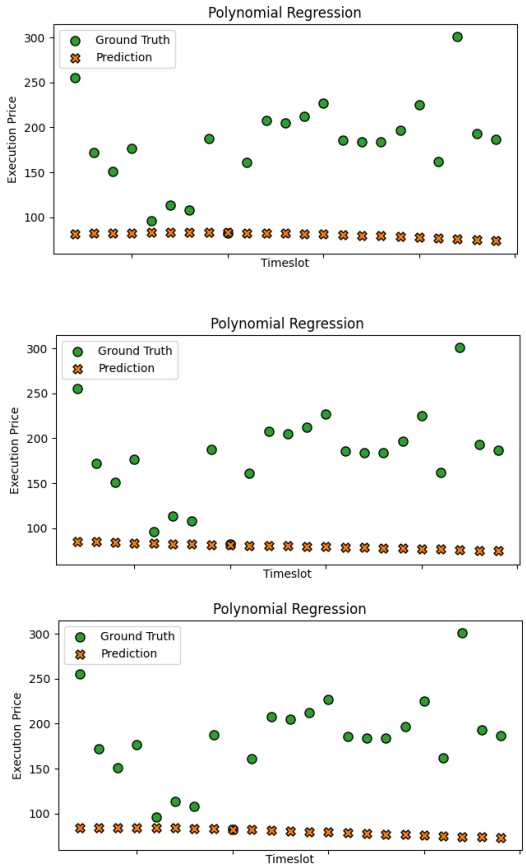

Figure 5.27: Polynomial Regression with the first "hard" dataset. The three smaller plots compare the ground truth and the prediction results for the three different exponents. The green dot shows the ground truth while the x depicts the polynomial model's predictions.

In figure 5.27 we notice that we have unsuccessful results compared to the linear regression predictor. The line of the predicted values and the line of the actual ones deviate a lot, as the first one lies on 100 euros, while the second one ranges from 100 to 300 euros. Between the three exponents, the most efficient one is n = 4, with a slight lead against the others.

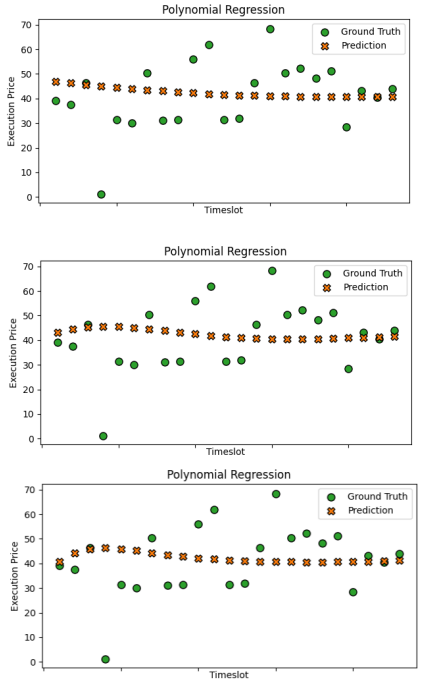

Figure 5.28: Polynomial Regression with the second "hard" dataset. The three smaller plots compare the ground truth and the prediction results for the three different exponents. The green dot shows the ground truth while the x depicts the polynomial model's predictions.

In figure 5.28 we observe the best result of our polynomial regression predictor. The predicted values and the real ones come to a very satisfying deviation and between the three exponents, the most efficient one is n = 6. The error of the predicted values is around 50 euros.

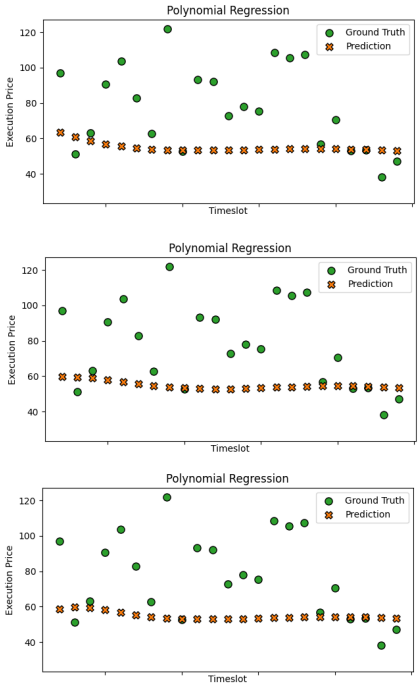

Figure 5.29: Polynomial Regression with the third "hard" dataset. The three smaller plots compare the ground truth and the prediction results for the three different exponents. The green dot shows the ground truth while the x depicts the polynomial model's predictions.

In figure 5.29 we notice that the results are quite good compared to the other datasets, using polynomial regression. The predicted line and the actual line do not deviate a lot compared to the linear case. Between the three exponents, the most efficient one is n = 8.

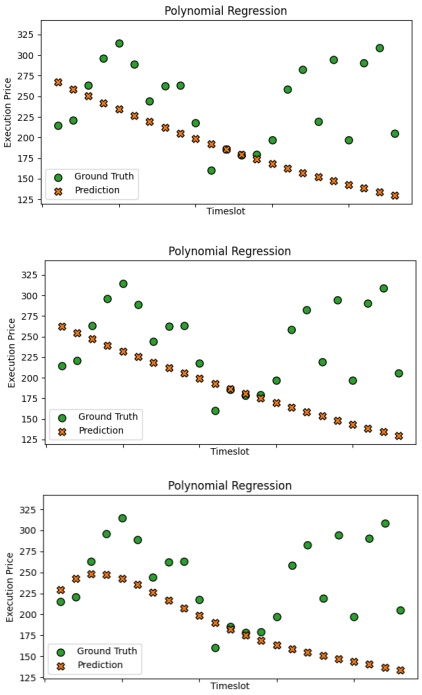

Figure 5.30: Polynomial Regression with the fourth "hard" dataset. The three smaller plots compare the ground truth and the prediction results for the three different exponents. The green dot shows the ground truth while the x depicts the polynomial model's predictions.

In figure 5.30 we observe that we have not desirable results, as the predicted line and the actual line do not coincide. For instance, the error of the predicted values is around 150 euros. Between the three exponents, the most efficient one is n = 6.

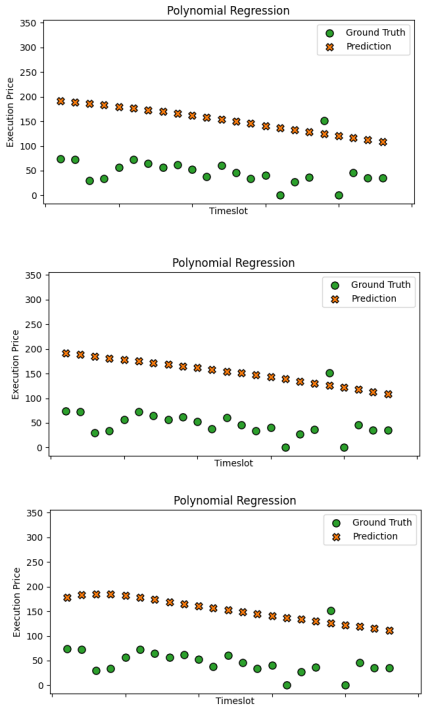

Figure 5.31: Polynomial Regression with the fifth "hard" dataset. The three smaller plots compare the ground truth and the prediction results for the three different exponents. The green dot shows the ground truth while the x depicts the polynomial model's predictions.

In figure 5.31 we observe that we have the worst results of the polynomial regression predictor. The deviation between the real values and the predicted values is huge. Between the three exponents, the most efficient one is n = 8, with a slight lead against the other two.

Polynomial Regression here does not give us a desirable outcome, as we detect that predicted values deviate to a great extent from ground truths. One of the main reasons that deviation happens is overfitting. When a model fits too closely to the training dataset, it is considered overfitted. The model captures noise in the data and not just the underlying trends. This has the effect of making the model perform better on a training dataset than on a testing dataset. Overfitting typically happens when a model has too many features or is overly complex.

Predictor 3 (LSTM) Results

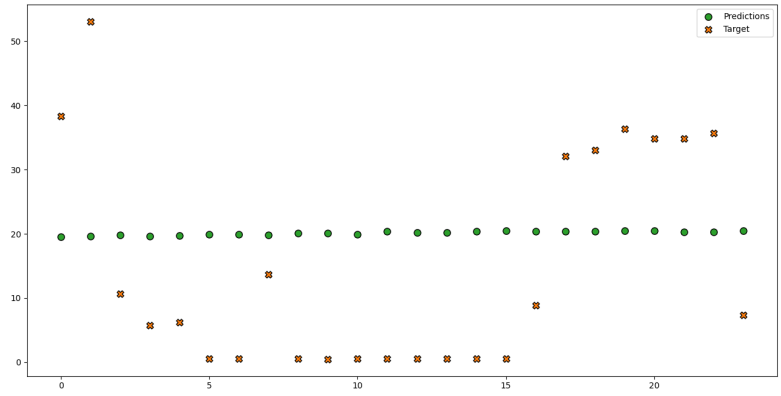

Figure 5.32: LSTM 1 with the first "hard" dataset. The green dot shows the prediction while the x depicts the model's ground truth.

In figure 5.32 we observe the results of our first LSTM predictor using the first hard dataset. A deviation between the predicted values and the actual ones exists and the error of the predicted values is around 30 euros.

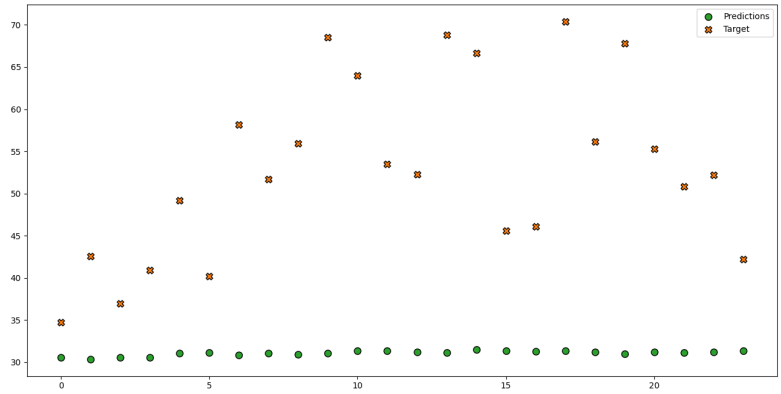

Figure 5.33: LSTM 1 with the second "hard" dataset. The green dot shows the prediction while the x depicts the model's ground truth.

We can see a significant difference between the anticipated and real values in figure 5.32. The line of the predicted values and the line of the actual ones deviate, as the first one lies on 30 euros, while the second one ranges from 300 to 70 euros.

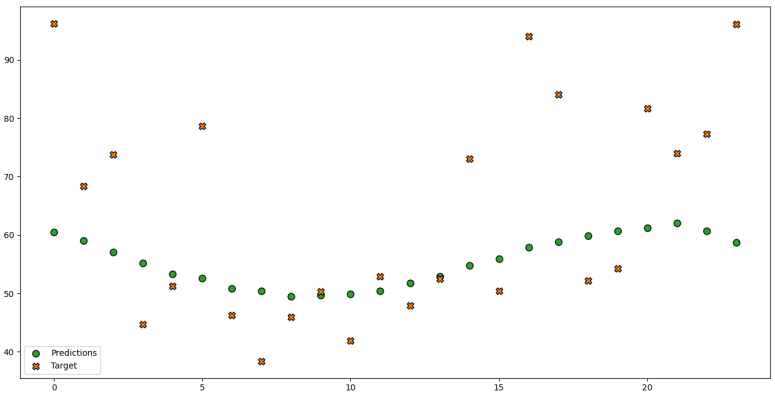

Figure 5.34: LSTM 1 with the third "hard" dataset. The green dot shows the prediction while the x depicts the model's ground truth.

In figure 5.3 we can observe a divergence between the predicted values and the actual ones. The predicted line ranges from 50 to 60 euros, while the ground truth line ranges from 40 to 90 euros. The error of the predicted values is around 35 euro.

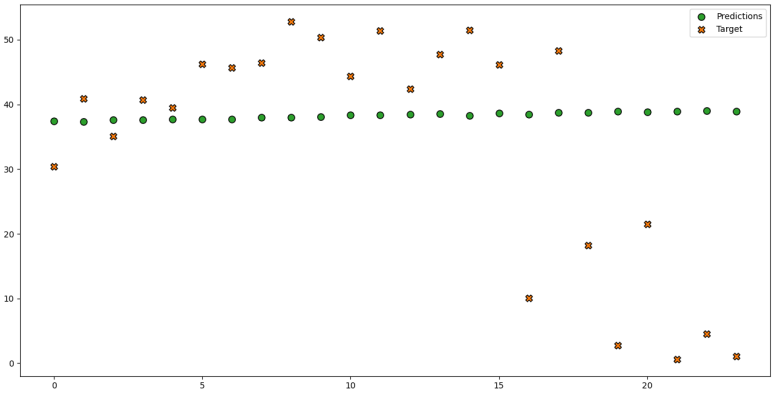

Figure 5.35: LSTM 1 with the fourth "hard" dataset. The green dot shows the prediction while the x depicts the model's ground truth.

In figure 5.35 we can observe a deviation between the predicted values and the actual ones. The predicted line lies at 40 euros, while the ground truth line ranges from 0 to 50 euros.

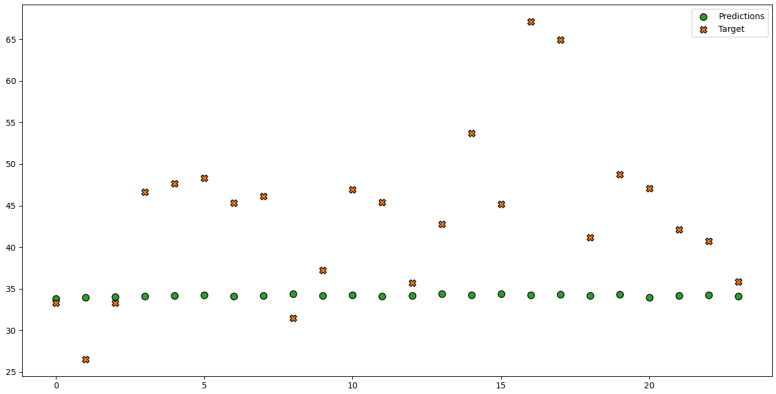

Figure 5.36: LSTM 1 with the fifth "hard" dataset. The green dot shows the prediction while the x depicts the model's ground truth.

In figure 5.36 we see the best prediction so far for the second LSTM model. We can notice that the actual values and the predicted ones are quite close, with some even matching exactly with their corresponding values. The error of the predicted values is around 30 euros.

The figures above indicate that, contrary to the "easy" datasets, our LSTM 1 predictor gives us more accurate results. It seems to work better than both Linear and Polynomial Regression and the metrics will confirm this statement.

Predictor 4 (LSTM) Results

Now we will present the experimental results from our best working predictor since it is the most accurate of all the other methods that we experimented with. The blue line depicts prices from 24 hours of the past, the green dots are the actual values and the orange marks are our predictions.

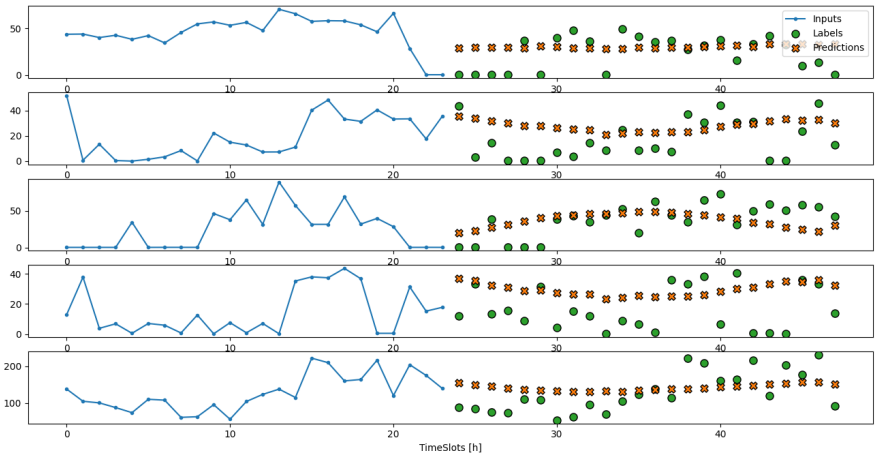

Figure 5.37: LSTM 2 with the first "hard" dataset. The green dot shows the ground truth, the x depicts the model's predictions and the blue line represents our input from the past 24 hours. The four smaller plots compare the ground truth and the prediction results from different days.

In figure 5.37 we observe a great result as the 24 ground truths and our predictions coincide and the error of the predicted values is less than 5 euros.

The 24 ground truths and our predictions are in perfect agreement, and the inaccuracy of the predicted values is less than 5 euros, as shown in figure 5.38.

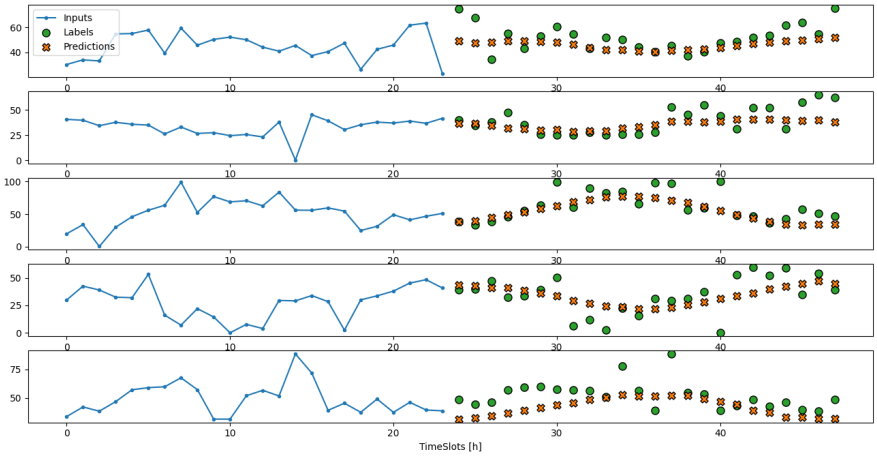

Figure 5.38: LSTM 2 with the second "hard" dataset. The green dot shows the ground truth, the x depicts the model's predictions and the blue line represents our input from the past 24 hours. The four smaller plots compare the ground truth and the prediction results from different days.

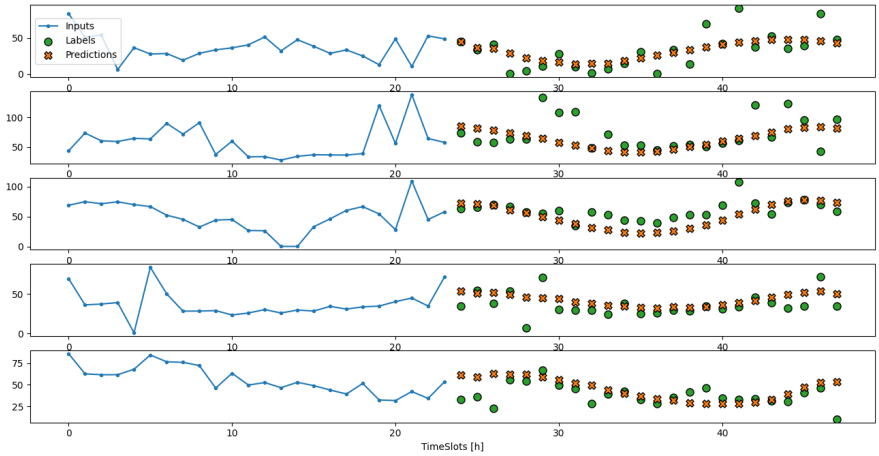

Figure 5.39: LSTM 2 with the third "hard" dataset. The green dot shows the ground truth, the x depicts the model's predictions and the blue line represents our input from the past 24 hours. The four smaller plots compare the ground truth and the prediction results from different days.

Figure 5.39 represents the best performance of our agent, since the predicted error is below 3 euros.

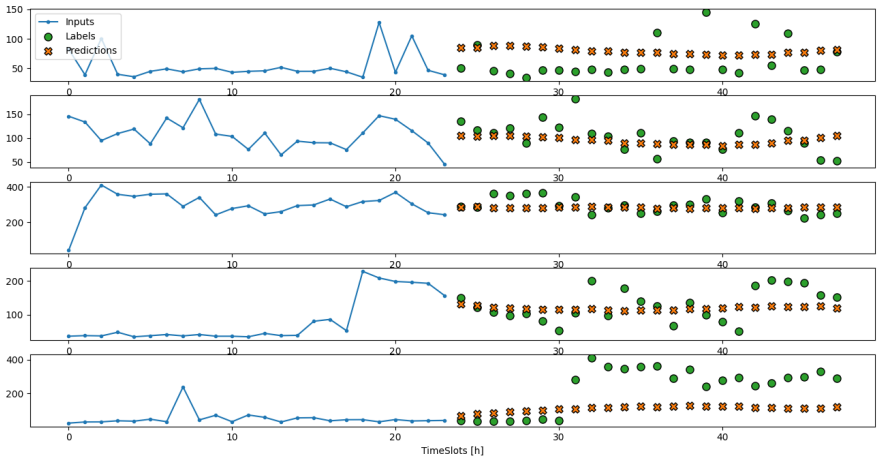

Figure 5.40: LSTM 2 with the fourth "hard" dataset. The green dot shows the ground truth, the x depicts the model's predictions and the blue line represents our input from the past 24 hours. The four smaller plots compare the ground truth and the prediction results from different days.

We can see a fantastic result in figure 5.40, where the 24 ground truths and our predictions match up perfectly and the predicted values are off by less than 5 euros.

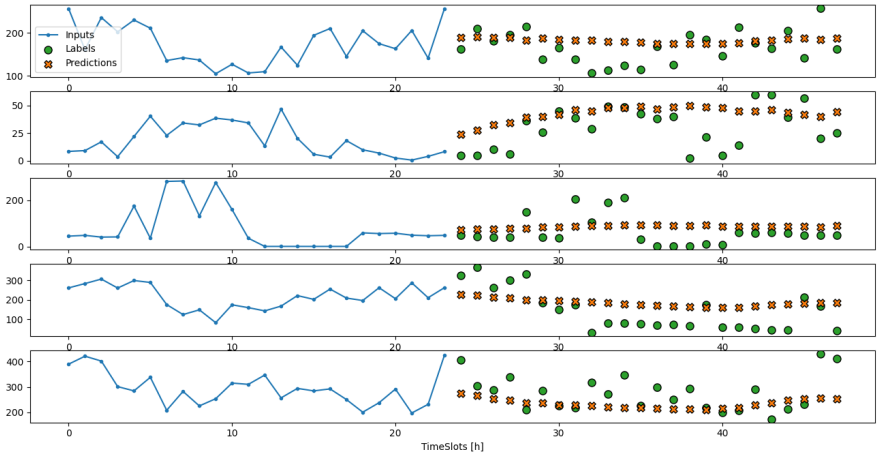

Figure 5.41: LSTM 2 with the fifth "hard" dataset. The green dot shows the ground truth, the x depicts the model's predictions and the blue line represents our input from the past 24 hours. The four smaller plots compare the ground truth and the prediction results from different days.

In figure 5.41 the 24 ground facts and our predictions are in perfect agreement, and the inaccuracy of the predicted values is approximately below 3 euros.

As we expected, LSTM 2 is again the most efficient method for predictions. We can clearly see in the aforementioned figures, our predicted values are very close to the actual ones, meaning that our predictor works ideally with both "easy" and "hard" datasets.

5.2.3 Observed Errors

A coherent way to evaluate every predictor's efficiency is to calculate two metrics, the Mean Squared Error (MSE) and the Root Mean Squared Error (RMSE), related to the deviation of our predictions from the actual results. MSE is a metric that tells us the average squared difference between the predicted values and the actual values in a dataset. The lower the MSE, the better a model fits a dataset.

$$MSE = \frac{1}{n} \quad \Sigma(y_i^{\text{real}} - y_i^{\text{pred}})^2$$

Correspondingly RMSE is a metric that tells us the square root of the average squared difference between the predicted values and the actual values in a dataset. The lower the RMSE, the better a model fits a dataset.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n} \quad \Sigma(y_i^{\text{real}} - y_i^{\text{pred}})^2}$$

"Easy" Games Errors

MSE	Linear	Poly(4)	Poly(6)	Poly(8)	LSTM 1	LSTM 2
dataset 1	143.02	139.38	139.02	139.42	786.11	0.4288
dataset 2	294.85	337.25	335.20	335.05	236.88	0.3488
dataset 3	137.05	198.25	200.71	202.53	2200.45	0.4086
dataset 4	269.91	228.31	230.93	229.84	155.42	0.3147
dataset 5	193.79	177.99	189.27	188.32	167.10	0.2864

Table 5.1: MSE for the "easy" datasets

Table 5.1 shows the Mean Squared Error of every prediction method that we have implemented. We can easily detect that LSTM 2 is the most accurate method, as we have already seen from the graphs, as it has by far the lowest MSE compared to the other methods. LSTM 1 appears to have the greatest MSE, therefore it is the least effective method. To assess the performance of the other two methods, we have to calculate the averages because no clear answer can be deduced from these results.

RMSE	Linear	Poly(4)	Poly(6)	Poly(8)	LSTM 1	LSTM 2
dataset 1	11.95	11.8	11.79	11.8	28.03	0.6531
dataset 2	11.70	18.36	18.30	18.30	15.39	0.5905
dataset 3	27.46	14.08	14.16	14.23	46.90	0.6392
dataset 4	16.42	15.11	15.19	15.16	12.46	0.5609
dataset 5	13.92	13.34	13.75	13.72	12.92	0.5351

Table 5.2: RMSE for the "easy" datasets

Table 5.2 shows the Root Mean Squared Error of our predictors. It is obvious again LSTM 2 has the lowest RMSE. Therefore this corroborates the findings of Table 5.2 and section 5.2.1 is the most preferable method in order to predict future values and correspondingly LSTM 1 is the least accurate one.

"Hard" Games Errors

MSE	Linear	Poly(4)	Poly(6)	Poly(8)	LSTM 1	LSTM 2
dataset 1	7,877	12,864	12,985	13,019	439.97	0.2840
dataset 2	226.21	171.86	169.09	170.67	321.87	0.3075
dataset 3	754.55	751.73	750.96	748.47	331.96	0.2606
dataset 4	4,430	9,034	8,989	9,131	346.13	0.9221
dataset 5	12,395	13,927	13,871	13,840	308.78	0.3741

Table 5.3: MSE for the "hard" datasets

Here at 5.3 we have the Mean Squared Error of every prediction method for the "hard" datasets. LSTM 2 is the most accurate method, no matter the difficulty of the dataset, as it has again the lowest MSE compared to the other methods. Contrary to the "easy" datasets, LSTM 1 here is very accurate and it comes second to our ranking list. Linear and Polynomial Regression are not desirable at all with these datasets, as the MSE's of both methods are massive.

RMSE	Linear	Poly(4)	Poly(6)	Poly(8)	LSTM 1	LSTM 2
dataset 1	88.75	113.42	113.95	114.10	20.97	0.5329
dataset 2	15.04	13.10	13.00	13.06	17.94	0.5545
dataset 3	27.46	27.41	27.40	27.35	18.21	0.5104
dataset 4	66.56	95.04	94.81	95.54	18.60	0.9602
dataset 5	111.33	118.01	117.77	117.64	17.57	0.6116

Table 5.4: RMSE for the "hard" datasets

Table 5.4 shows us the Root Mean Squared Error of our predictors. LSTM 2 has the lowest RMSE, therefore this verifies it as the most preferable method for predicting future values while Linear and Polynomial Regression are the least accurate ones.

Averages

MSE	Linear	Poly(4)	Poly(6)	Poly(8)	LSTM 1	LSTM 2
Easy	207.724	216.236	219.026	219.032	709.192	0.3574
Hard	5,136	7,349	7,352	7,381	349.67	0.4296

Table 5.5: Average MSE for the "hard" datasets

RMSE	Linear	Poly(4)	Poly(6)	Poly(8)	LSTM 1	LSTM 2
Easy	16.29	14.53	14.74	14.64	23.14	0.59576
Hard	41.82	73.39	73.386	73.538	18.658	0.63392

Table 5.6: Average RMSE for the "hard" datasets

In Tables 5.5 and 5.6 we have calculated the average MSE and RMSE of every predictor for both easy and hard datasets. We can deduce by the results that all predictors except

from LSTM 1 performed much better with the easy datasets as the divergence between the numbers is huge. The same, but with a lower order of magnitude, applies to the RMSE metrics. All except for the first LSTM method were accurate as they presented lower error rates.

The results from both easy and hard datasets indicate that our model works better with the LSTM 2 predictor. It is the most accurate method according to our experimental results, as it has the lowest error rates and in its graphs, ground truths and prediction values in many cases almost coincide.

5.3 Wholesale Market Results

As discussed previously, we have implemented some improvements in the Monte Carlo Tree Search Algorithm of 2020's TUC-TAC and we tested its effectiveness by comparing it with multiple other agent variations. The first one is the Sample Broker, the default agent that PowerTAC gives us to start with. Sample Broker has a very simple implementation in its wholesale market and it does not include an MCTS algorithm. Furthermore, we conducted experiments with TUC-TAC 2020 (LSTM2), which is the TUC-TAC 2020 agent that has incorporated an LSTM2 predictor for more accurate results. To have a solid comparison of the results between the agents we also had to include a version of the TUC-TAC agent without the new predictor, so we can see that there have been actual improvements to the Monte Carlo Tree Search Algorithm. For this particular reason, we also tested the agent TUC-TAC 2020 (new MCTS), which has the old prediction method but the improved MCTS algorithm. Sample Broker is the basic agent that we discussed before, TUC-TAC 2020 is the first version of our agent that has a simple predictor and an MCTS implementation but not the improved one. Then we have TUC-TAC 2020 (LSTM2), which retains the optimal predictor and the old version of MCTS. TUC-TAC 2020 (new MCTS) consists of the old prediction method, but the improved MCTS algorithm. Lastly, we have TUC-TAC 2022 which uses the LSTM2 prediction method and the improved Monte Carlo Tree Search algorithm. The metrics that we took into consideration for comparing these three agents are the following:

- **Total Retail Balance**: The earnings our agent made from selling energy to consumers through tariffs. We want this value to be as high as possible.
- **Total Retail kWh**: The amount of kWh our agent sold to consumers. We want this value to be as high as possible.
- **Total Wholesale Balance**: The amount our agent paid or earned from buying or selling energy in wholesale's market double auction. We want this value to be as high as possible.
- **Total Wholesale kWh**: The amount of kWh our agent bought/sold in wholesale's market double auction. We want this value to be as high as possible.
- **Total Imbalance Penalty**: A penalty fee that we have to pay if we are not able to buy all the energy that we need and we "owe energy to our clients. We want this value to be as low as possible.
- **Total kWh Imbalance**: The total amount of kWh that had to be bought but were not. We want this value to be as low as possible.

- **Total Average Selling Price**: The average price our agent sells energy (per kWh). We want this value to be as high as possible.

In this section, we present the results from 10 Games, 5 with the "easy" datasets and 5 with the "hard" ones, for 1500 timeslots each. For each dataset, we have plotted 4 different images, from 4 different days. Highlighted numbers represent the best-performing agent in a specific evaluation metric category.

5.3.1 Easy Games

Game 1					
Metrics (Total)	Sample Broker	TUC-TAC 2020	TUC-TAC 2020 LSTM2	TUC-TAC 2020 new MCTS	TUC-TAC 2022
Retail Balance	**1,161,082**	684,251	712,233	569,046	627,775
Retail kWh	15,319,242	17,331,504	17,714,986	**18,014,346**	17,368,024
Wholesale Balance	**261,806**	251,013	227,310	189,789	181,647
Wholesale kWh	**56,211,71**	50,305,171	41,136,075	54,723,331	49,169,192
Imbalance Penalty	3,228,110	3,404,894	3,549,583	3,354,726	**2,179,554**
kWh Imbalance	10,089,637	15,623,575	16,835,411	15,517,659	**8,779,922**
Avg Selling Price	22.99	30.84	28.66	33.33	**39.48**

Table 5.7: Wholesale Market Results for Game 1

In Table 5.7 we can notice that TUC TAC 2022 performs better at all the categories that we aimed to optimize. The average sale per kWh has risen, while the kWh imbalance and the total charge have declined. As we can see from the data above, the inclusion of the LSTM 2 predictor as well as the integration of the Monte Carlo Tree Search Algorithm has helped with all the areas that we intended to improve.

Game 2					
Metrics (Total)	Sample Broker	TUC-TAC 2020	TUC-TAC 2020 LSTM2	TUC-TAC 2020 new MCTS	TUC-TAC 2022
Retail Balance	**1,106,423**	601,448	678,724	623,781	576,504
Retail kWh	14,623,269	18,918,778	**19,092,581**	18,519,559	17,832,905
Wholesale Balance	**300,337**	293,313	268,789	197,775	233,503
Wholesale kWh	**60,834,628**	54,137,711	43,565,037	58,483,312	52,377,660
Imbalance Penalty	4,299,771	4,759,642	4,930,928	4,304,485	**3,314,501**
kWh Imbalance	15,608,123	23,780,158	25,283,043	21,300,039	**14,879,724**
Avg Selling Price	25.46	31.09	29.54	33.44	**42.75**

Table 5.8: Wholesale Market Results for Game 2

Table 5.8 shows that TUC TAC 2022 performs better in every area where we tried to improve it. While the kWh imbalance and the overall charge have decreased, the average sale per kWh has increased. The incorporation of the LSTM 2 predictor along with the integration of the Monte Carlo Tree Search Algorithm has assisted with all the categories

that we aimed to improve as we can clearly see from the numbers above.

Game 3					
Metrics (Total)	Sample Broker	TUC-TAC 2020	TUC-TAC 2020 LSTM2	TUC-TAC 2020 new MCTS	TUC-TAC 2022
Retail Balance	**1,018,929**	712,464	641,607	606,284	642,429
Retail kWh	13,418,713	16,148,729	**16,446,728**	16,076,478	15,621,780
Wholesale Balance	296,762	**299,344**	238,494	206,626	212,977
Wholesale kWh	51,886,476	45,137,603	39,378,367	**56,856,636**	50,613,704
Imbalance Penalty	6,537,826	7,894,708	5,452,547	4,260,570	**3,144,003**
kWh Imbalance	21,278,426	31,273,478	23,488,415	17,413,045	**13,288,216**
Avg Selling Price	24.57	31.84	29.59	38.22	**46.38**

Table 5.9: Wholesale Market Results for Game 3

TUC TAC 2022 performs better in every area where we attempted to make improvements, as seen in Table 5.9. The average sale per kWh has increased while the kWh imbalance and total charge have reduced. As we can see from the data above, the inclusion of the LSTM 2 predictor as well as the integration of the Monte Carlo Tree Search Algorithm has helped with all the areas that we intended to improve.

Game 4					
Metrics (Total)	Sample Broker	TUC-TAC 2020	TUC-TAC 2020 LSTM2	TUC-TAC 2020 new MCTS	TUC-TAC 2022
Retail Balance	**1,259,756**	759,530	664,093	696,765	675,111
Retail kWh	16,645,615	19,716,246	**19,939,754**	19,608,000	18,988,996
Wholesale Balance	**220,217**	196,754	200,956	139,062	178,562
Wholesale kWh	**62,250,876**	52,932,244	43,561,435	54,451,878	48,570,076
Imbalance Penalty	1,153,622	2,659,475	2,343,957	2,156,683	**1,415,395**
kWh Imbalance	414,088	11,666,555	9,446,477	7,634,207	**2,879,205**
Avg Selling Price	21.21	28.81	28.58	28.36	**34.05**

Table 5.10: Wholesale Market Results for Game 4

We can see from Table 5.10 that TUC TAC 2022 performs better in every area where we tried to enhance it. kWh imbalance and total charge have decreased, but the average sale per kWh has increased. As we can see from the data above, the LSTM 2 predictor and Monte Carlo Tree Search Algorithm have helped in all the areas where we wanted to make improvements.

Game 5					
Metrics (Total)	Sample Broker	TUC-TAC 2020	TUC-TAC 2020 LSTM2	TUC-TAC 2020 new MCTS	TUC-TAC 2022
Retail Balance	**1,045,170**	570,335	597,333	798,332	653,401
Retail kWh	13,805,294	17,586,837	16,887,005	**19,775,841**	16,085,445
Wholesale Balance	283,567	293,045	298,991	**336,467**	209,644
Wholesale kWh	**65,480,217**	53,623,015	44,565,535	47,569,245	51,982,762
Imbalance Penalty	1,454,595	4,185,356	3,726,703	6,107,284	**2,429,584**
kWh Imbalance	1,793,111	19,052,757	16,237,154	27,833,945	**9,157,177**
Avg Selling Price	22.42	30.40	29.31	31.69	**34.87**

Table 5.11: Wholesale Market Results for Game 5

In Table 5.11 we can notice that TUC TAC 2022 performs better at all the categories that we aimed to optimize. The average sale per kWh has risen, while the kWh imbalance and the total charge have declined. As we can see from the data above, the inclusion of the LSTM 2 predictor as well as the integration of the Monte Carlo Tree Search Algorithm has helped with all the areas that we intended to improve.

Average					
Metrics (Total)	Sample Broker	TUC-TAC 2020	TUC-TAC 2020 LSTM2	TUC-TAC 2020 new MCTS	TUC-TAC 2022
Retail Balance	**1,118,272**	665,605	668,492	658,841	635,044
Retail kWh	14,762,426	14,392,418	18,016,210	**18,398,844**	17,179,430
Wholesale Balance	**271,737**	266,693	246,908	213,944	203,266
Wholesale kWh	**59,332,793**	51,227,148	42,441,290	54,416,880	50,542,679
Imbalance Penalty	3,334,784	4,580,815	4,000,743	4,036,750	**2,496,607**
kWh Imbalance	9,836,677	20,279,304	18,258,100	17,939,780	**9,796,849**
Avg Selling Price	23.33	30.59	29.13	33	**39.50**

Table 5.12: Average Wholesale Market Results for Easy Games

Table 5.12 shows the averages of every agent from the 5 easy dataset games.

From the tables above we can notice that TUC-TAC 2022 is the best performing agent of all four. It sells energy at a higher price than Sample Broker, TUC-TAC 2020, TUC-TAC 2020 (LSTM2), and TUC-TAC 2020 (new MCTS) do and it has decreased the Total Charge compared to the other agents, meaning that our agent satisfies our client's needs and does not owe energy to them. This also leads to a lower kWh imbalance and we can also conclude that by looking at the Tables. This makes sense not only because it has an improved version of the MCTS algorithm but also because it has the LSTM2 predictor implemented, making its buying and selling actions much more profitable. TUC-TAC 2020 (LSTM2) does not work as expected since it has a higher amount of Total Charge and kWh imbalance compared to TUC-TAC 2020. TUC-TAC 2020 (LSTM2) also underperforms compared to TUC-TAC 2020 (new MCTS) as we can see, since the second one sells energy at a higher price and has a lower Total Charge and kWh Imbalance. This means that our improvement has made a difference and combined with the right prediction method can lead to a desirable result.

5.3.2 Hard Games

Game 6					
Metrics (Total)	Sample Broker	TUC-TAC 2020	TUC-TAC 2020 LSTM2	TUC-TAC 2020 new MCTS	TUC-TAC 2022
Retail Balance	**1,385,510**	962,072	1,031,247	856,236	925,307
Retail kWh	18,302,992	22,016,266	21,860,860	21,113,038	**22,431,176**
Wholesale Balance	**245,823**	161,626	202,862	183,074	210,602
Wholesale kWh	47,407,940	50,874,266	39,482,092	**57,017,501**	51,141,310
Imbalance Penalty	5,103,532	4,127,445	4,843,460	**1,994,133**	2,241,592
kWh Imbalance	11,292,632	8,223,403	12,333,559	**969,432**	2,048,637
Avg Selling Price	24.15	29.09	28.80	39.12	**46.1**

Table 5.13: Wholesale Market Results for Game 6

In Table 5.13 we notice different results compared to the other tables. Here, TUC TAC 2020 new MCTS performs better, as it has less Total Charge and kWh Imbalance compared to the other agents. However, TUC TAC 2022 has again achieved to rise the average sale of the kWh.

Game 7					
Metrics (Total)	Sample Broker	TUC-TAC 2020	TUC-TAC 2020 LSTM2	TUC-TAC 2020 new MCTS	TUC-TAC 2022
Retail Balance	**1,069,176**	718,684	670,174	726,264	632,113
Retail kWh	14,124,094	**18,275,306**	17,309,926	17,066,679	16,231,280
Wholesale Balance	240,934	**245,248**	158,373	146,221	178,721
Wholesale kWh	**61,701,900**	53,490,969	48,661,490	58,976,059	53,950,690
Imbalance Penalty	3,204,016	4,359,164	3,202,315	2,740,855	**2,115,002**
kWh Imbalance	9,548,288	19,450,364	13,328,501	13,662,065	**8,380,753**
Avg Selling Price	21.80	30.99	28.81	32.16	**39.35**

Table 5.14: Wholesale Market Results for Game 7

Table 5.14 shows that TUC TAC 2022 performs better in every area where we tried to improve it. While the kWh imbalance and the overall charge have decreased, the average sale per kWh has increased. The incorporation of the LSTM 2 predictor along with the integration of the Monte Carlo Tree Search Algorithm has assisted with all the categories that we aimed to improve as we can clearly see from the numbers above.

Game 8					
Metrics (Total)	Sample Broker	TUC-TAC 2020	TUC-TAC 2020 LSTM2	TUC-TAC 2020 new MCTS	TUC-TAC 2022
Retail Balance	**1,302,000**	798,332	670,863	689,551	616,236
Retail kWh	17,195,805	19,775,841	20,236,817	**20,523,285**	18,984,145
Wholesale Balance	**401,902**	336,467	302,034	278,056	260,470
Wholesale kWh	49,284,366	47,569,245	35,215,605	**58,715,303**	49,470,299
Imbalance Penalty	7,389,105	6,107,284	6,219,380	4,798,049	**3,185,106**
kWh Imbalance	22,560,475	27,833,945	25,709,010	19,318,039	**11,710,998**
Avg Selling Price	27.59	31.69	29.74	38.33	**43.35**

Table 5.15: Wholesale Market Results for Game 8

In Table 5.15 we can notice that TUC TAC 2022 performs better at all the categories that we aimed to optimize. The average sale per kWh has risen, while the kWh imbalance and the total charge have declined. As we can see from the data above, the inclusion of the LSTM 2 predictor as well as the integration of the Monte Carlo Tree Search Algorithm has helped with all the areas that we intended to improve.

Game 9					
Metrics (Total)	Sample Broker	TUC-TAC 2020	TUC-TAC 2020 LSTM2	TUC-TAC 2020 new MCTS	TUC-TAC 2022
Retail Balance	**1,379,664**	984,276	833,737	997,119	831,435
Retail kWh	18,252,226	**26,152,775**	22,263,002	23,395,772	22,825,668
Wholesale Balance	224,228	207,034	65,190	**428,057**	340,495
Wholesale kWh	35,345,213	37,086,738	11,122,206	**52,280,195**	46,459,272
Imbalance Penalty	12,789,434	12,306,536	9,485,572	6,364,038	**4,617,453**
kWh Imbalance	23,871,158	22,666,119	12,431,486	5,064,049	**3,167,264**
Avg Selling Price	24.26	30.8	29.03	71.51	**72.71**

Table 5.16: Wholesale Market Results for Game 9

TUC TAC 2022 performs better in every area where we attempted to make improvements, as seen in Table 5.16. The average sale per kWh has increased while the kWh imbalance and total charge have reduced. As we can see from the data above, the inclusion of the LSTM 2 predictor as well as the integration of the Monte Carlo Tree Search Algorithm has helped with all the areas that we intended to improve.

Game 10					
Metrics (Total)	Sample Broker	TUC-TAC 2020	TUC-TAC 2020 LSTM2	TUC-TAC 2020 new MCTS	TUC-TAC 2022
Retail Balance	1,365,339	**1,464,437**	1,420,079	864,916	834,367
Retail kWh	18,054,485	24,632,508	**24,726,180**	20,957,228	22,002,045
Wholesale Balance	266,538	303,487	248,553	362,956	**380,487**
Wholesale kWh	46,423,086	41,573,614	34,698,296	**56,852,168**	50,071,344
Imbalance Penalty	10,209,012	11,357,822	9,299,839	**3,784,663**	4,442,230
kWh Imbalance	17,318,530	21,605,974	19,777,612	**37,864**	2,894,616
Avg Selling Price	22.75	31.46	29.16	59.56	**71.16**

Table 5.17: Wholesale Market Results for Game 10

We see distinct outcomes in Table 5.17 in comparison to the other tables. Since TUC TAC 2020 new MCTS has lower Total Charge and kWh Imbalance compared to the other agents, it performs better in this situation. However, TUC TAC 2022 has once more succeeded in increasing the average sale of kWh.

Average					
Metrics (Total)	Sample Broker	TUC-TAC 2020	TUC-TAC 2020 LSTM2	TUC-TAC 2020 new MCTS	TUC-TAC 2022
Retail Balance	**1,300,337**	985,560	925,220	826,817	767,892
Retail kWh	17,185,920	**22,170,539**	21,279,357	20,611,200	20,494,863
Wholesale Balance	275,885	250,772	195,402	**279,673**	274,155
Wholesale kWh	48,034,101	37,556,366	33,835,937	**56,768,245**	50,218,583
Imbalance Penalty	7,739,019	7,651,650	6,610,113	3,936,347	**3,320,277**
kWh Imbalance	16,918,261	19,955,961	16,716,033	7,810,290	**5,640,454**
Avg Selling Price	24.11	30.80	29.11	48.36	**54.53**

Table 5.18: Average Wholesale Market Results for Hard Games

In Table 5.18 we can see the averages of every agent from the 5 hard dataset games.

In these experiments, the tables show us that we get better results than what we expected. We can notice again that TUC-TAC 2020 (new_MCTS) carries out almost all experiments better than TUC-TAC 2020 and TUC-TAC 2020 (LSTM2) in all aspects compared. It sells energy at a higher price, whereas is charged less and has a lower kWh imbalance. Running the games with the hard datasets makes it more difficult for Sample Broker and TUC-TAC 2020 to buy and sell energy, in opposition to TUC-TAC 2022 which sells energy at a very high price, almost two times the value of TUC-TAC's 2020. Total Charge and Total kWh Imbalance have been reduced a lot compared to the other two brokers and in general, TUC-TAC 2022 is performing better. Again we can see that agent TUC-TAC 2020 (new_MCTS) performs better than TUC-TAC 2020, meaning that our changes were useful and made an actual improvement to the broker. On the contrary, TUC-TAC 2020 (LSTM2) performs better with the hard datasets, as it has less Total Charge and less kWh imbalance compared to TUC-TAC 2020.

In general, from the experiments conducted we can tell with absolute certainty that TUC-TAC 2022, which consists of the LSTM2 predictor and the improved version of the Monte Carlo Tree Search algorithm, is the optimal agent compared to the other four brokers. We have to mention that our work on MCTS was efficient since TUC-TAC 2020 (new_MCTS) performed better than TUC-TAC 2020 and TUC-TAC 2020 (LSTM2) in all of the categories of comparison that we tested it. The addition of the prediction method was the combination that we needed to have even better results than before.

Chapter 6

Conclusions

Our main target was to improve the performance of agent TUC TAC 2020 with the actions that were mentioned above and to participate in PowerTAC 2022. The problem was approached with Machine Learning Techniques mostly and we can say that the results were desirable ones. Our current agent performs better, as it sells energy more profitably and pays far fewer penalties.
As far as it concerns the prediction methods, we concluded that Linear and Polynomial Regression were not suitable for our problem, in contrast to the LSTM method that fitted perfectly and helped us complete our work. In Wholesale Market, many variations of the Monte Carlo Tree Search Algorithm were tested, from changing the strategy to alternating the algorithm's parameters and we came to an end with the current implementation.

6.1 Future Work

Despite the fact that a variety of techniques were used for this project, there is still an opportunity for enhancements and additions. To begin with, the wholesale market module can become even better than it currently is and bring fewer penalty fees and kWh imbalance. Our current numbers are already low compared to TUC TAC 2020, but they can be reduced further. This can be achieved by improving our current Monte Carlo Tree Search Algorithm by adding more iterations or using a better bidding strategy.

Moreover, besides the clearing price predictor, a net demand predictor can be added to our module. This will result in having fewer Transmission Capacity Fees and will make our agent more efficient, as we will be informed about future high risk demand peaks. More tariff types can also lead to a significant upgrade of our agent, as we will have increased profits compared to that of the other competitors.

We could also experiment with other artificial intelligence methods that are either tested by successful PowerTAC brokers or are celebrated in related research fields. As it concerns the predictive methods, Gated recurrent units (GRU), a gating mechanism in recurrent neural networks similar to LSTM, could be a very accurate method for our forecast predictions. Moreover, Belief Networks, which describe the database structure using a tree format, could also be used to provide valid prediction results. Regarding the bidding strategies of our agent, besides the Monte Carlo Tree Search Algorithm, we could im-

plement other methods such as the Cournot quantity model or Nash equilibrium for the bids that our agent has to make at double auction [28]. Finally, we could also experiment with an interruptible demand auction model for congestion relief in hour-ahead bilateral contracts dominating the electricity market. [28].

Bibliography

[1] W. Ketter, J. Collins, and M. de Weerdt, "The 2020 power trading agent competition," *Environmental Economics eJournal*, 2020.

[2] W. Ketter, J. Collins, and P. P. Reddy, "Power tac: A competitive economic simulation of the smart grid," *Energy Economics*, vol. 39, pp. 262–270, 2013.

[3] S. Orfanoudakis, S. Kontos, C. Akasiadis, and G. Chalkiadakis, "Aiming for half gets you to the top: Winning powertac 2020," in *In Proc. of the 18th European Conference on Multi-Agent Systems (EUMAS-2021), pages 144-159, A Virtual Conference - Online, June 2021.*, 2021.

[4] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Applied Sciences*, 2019.

[5] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *IEEE Transactions on Neural Networks*, vol. 16, 2005.

[6] B. Lim and S. Zohren, "Time-series forecasting with deep learning: A survey," *Philosophical Transactions of the Royal Society A*, vol. 379, 2021.

[7] A. A. Ismail, M. K. Gunady, H. C. Bravo, and S. Feizi, "Benchmarking deep learning interpretability in time series predictions," *ArXiv*, vol. abs/2010.13924, 2020.

[8] J. Wang, J. Wang, Z. G. Zhang, and S.-P. Guo, "Forecasting stock indices with back propagation neural network," *Expert Syst. Appl.*, vol. 38, pp. 14 346–14 355, 2011.

[9] J. Lago, F. D. Ridder, and B. D. Schutter, "Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms," *Applied Energy*, 2018.

[10] T. G. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *Eur. J. Oper. Res.*, vol. 270, pp. 654–669, 2018.

[11] A. M. Sykulski, A. C. Chapman, E. M. de Cote, and N. R. Jennings, "Ea2: The winning strategy for the inaugural lemonade stand game tournament," in *ECAI*, 2010.

[12] M. M. P. Chowdhury, "Predicting prices in the power tac wholesale energy market," in *AAAI*, 2016.

[13] K. Stefanos, *"energy consumption prediction via machine learning algorithms", senior undergraduate diploma thesis, school of ece, technical university of crete*, 2021.

[14] S. Özdemir and R. Unland, "Agentude 17 : Imbalance management of a retailer agent to exploit balancing market incentives in a smart grid ecosystem," 2018.

[15] ——, "Wholesale bidding approaches of an autonomous trading agent in electricity markets," in *SmartER Europe*, 2016.

[16] D. Grgić, H. Vdovic, J. Babic, and V. Podobnik, "Crocodileagent 2018: Robust agent-based mechanisms for power trading in competitive environments," *Comput. Sci. Inf. Syst.*, vol. 16, pp. 105–129, 2019.

[17] J. Babic and V. Podobnik, "Adaptive bidding for electricity wholesale markets in a smart grid," 2014.

[18] S. Ghosh, E. Subramanian, S. P. Bhat, S. Gujar, and P. Paruchuri, "Vidyutvanika: A reinforcement learning based broker agent for a power trading competition," in *AAAI*, 2019.

[19] D. Urieli and P. Stone, "An mdp-based winning approach to autonomous power trading: Formalization and empirical analysis," in *AAAI Workshop: AI for Smart Grids and Smart Buildings*, 2016.

[20] A. K. Laha, "Linear regression for predictive analytics," *Advances in Analytics and Applications*, 2018.

[21] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *ArXiv*, vol. abs/1808.03314, 2018.

[22] N. K. Manaswi, "Rnn and lstm," in *Deep Learning with Applications Using Python : Chatbots and Face, Object, and Speech Recognition With TensorFlow and Keras*. Berkeley, CA: Apress, 2018, pp. 115–126, ISBN: 978-1-4842-3516-4. DOI: 10.1007/978-1-4842-3516-4_9. [Online]. Available: https://doi.org/10.1007/978-1-4842-3516-4_9.

[23] Y. Zhang, X. Hao, and Y. Liu, "Simplifying long short-term memory for fast training and time series prediction," *Journal of Physics: Conference Series*, 2019.

[24] M. M. P. Chowdhury, C. Kiekintveld, S. Tran, and W. G. S. Yeoh, "Bidding in periodic double auctions using heuristics and dynamic monte carlo tree search," in *IJCAI*, 2018.

[25] O. Weslati, S. Bouaziz, and M. M. Serbaji, "The efficiency of polynomial regression algorithms and pearson correlation (r) in visualizing and forecasting weather change scenarios," in *Recent Advances in Polynomials*, K. Shah, Ed., Rijeka: IntechOpen, 2022, ch. 7. DOI: 10.5772/intechopen.102726. [Online]. Available: https://doi.org/10.5772/intechopen.102726.

[26] A. Graves, "Generating sequences with recurrent neural networks," *ArXiv*, vol. abs/1308.0850, 2013.

[27] Tensorflow, *Time series forecasting*, https://www.tensorflow.org/tutorials/structured_data/time_series#advanced_autoregressive_model.

[28] M. Prabavathi and R. Gnanadass, "Energy bidding strategies for restructured electricity market," *International Journal of Electrical Power Energy Systems*, vol. 64, pp. 956–966, 2015, ISSN: 0142-0615. DOI: https://doi.org/10.1016/j.ijepes.2014.08.018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0142061514005444.