Report on Forecasting Day-ahead Electricity Prices with a Multi-layered Neural Network Model

Carlos Gil, r0885289, and Mariana Adão da Fonseca, r0867610
Smart Distribution Systems [H00P3a] in 2021-2022
EIT-KIC Master in Energy, Katholieke Universiteit Leuven

CONTENTS

I Introduction 1

II Problem Definition 1

III Applied Methodology 1
III-A Data Setup for Model . . . 1
III-A1 Training Set (70% of the given data) . . . 1
III-A2 Validation Set (30% of the given data) . . . 1
III-A3 Test set (1) . . . 1
III-A4 Test set (2) . . . 1
III-B Feature Selection . . . 1
III-C Neural Network Architecture Selection . . . 2
III-D Input (Test) Data . . . 2

IV Proposed Solution Models 3

V Final Prediction and Observations 3

Appendix A: Further Information on the Architectures of the Proposed Neural Network Models 4
A-A Model 1 - (50 input layer; 50 hidden layer; 24 output layer) . . . 4
A-B Model 2 - (60 input layer; 48 hidden layer; 24 output layer) . . . 4
A-C Model 3 - (75 input layer; 150 hidden layer; 24 output layer) . . . 4

Appendix B: Supporting Files 4

References 4

LIST OF FIGURES

1 Visualisation of the Cross-validation Method [6] . . . 1
2 Correlation Data of BELPEX, Solar and wind output . . . 2
3 Example of a Feed-forward Neural Network with one hidden layer (with 3 neurons) [3] . . . 2
4 Prices data for input test data. . . . 2
5 Solar data for input test data. . . . 3
6 Wind data for input test data. . . . 3
7 behaviour of Loss (mean sqared error) for the different models tested . . . 3
8 Comparison of the three models using a common test data . . . 4
9 Day-ahead price prediction for the three weeks . . . 4
10 Model 1 Architecture Summary . . . 4
11 Model 2 Architecture Summary . . . 4
12 Model 3 Architecture Summary . . . 4

LIST OF TABLES

I Solution Models: number of Neurons per Layer . . . 3

Report on Forecasting Day-ahead Electricity Prices with a Multi-layered Neural Network Model

***Preface* - This report fulfils the requirements for Course H00P3a Smart Distribution Systems within the Department of Electrical Engineering (ESAT) in Katholieke Universiteit Leuven. The course aims to consolidate a deep insight and understanding of smart distribution grids and the tools for designing and implementing an architecture facilitating coordination, monitoring, and control of the grid.**

I. Introduction

This report intends to describe the application of machine learning to forecast day-ahead electricity prices for provided renewable energy sources. This report will also discuss the reasoning behind the proposed neural network architectures and discuss the influence of the selected features on the program's performance. As an endpoint, this report will conclude with the final plots and performance metrics results of the proposed neural network architectures, along with some future work observations. The coded and trained models and the predictions file obtained from the proposed solutions are also attached.

II. Problem Definition

This report aims to solve, as accurately as possible, the problem of predicting the Spot price of electricity on an hourly basis of the Belgian day-ahead wholesale electricity market, also known as BELPEX exchange [7], for solar and wind for 3 (random unknown) days in 2018.
The data-set available for training is the hourly prices of the BELPEX market for 2016 and 2017 and generation values[1] in MW for solar and wind in 15-minute time frames, for a whole year, also for 2016 and 2017.

III. Applied Methodology

A. Data Setup for Model

The problem at hand has the goal of predicting prices; therefore, we are handling a value prediction problem and not a categorization problem, which alternatively aims to classify a given dataset in two or more categories (Training, Validation and test data). We want to estimate how accurately our model will perform in practice for this problem. As a result, we must perform dataset splits to train our neural network model and test it. This report applies the cross-validation method, a re-sampling method that uses different portions of the data to test and train a model on different iterations [3]. We first split the given dataset into 2 — Train and Test - where we kept aside three random weeks of data for the test set. And then split the remaining data into Training and Validation sets, where 70% of the data is for training. The split ratio and set types are described below:

[1]the values given by the course lecturers did come with designated units, therefore, after evaluating the values, we assumed the data referred to generation in MW.

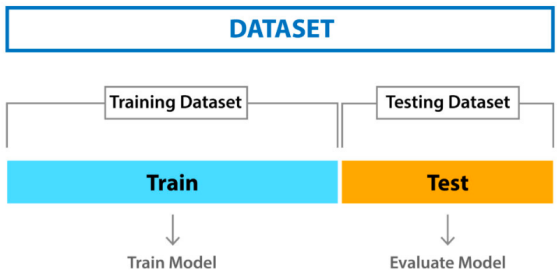

Fig. 1. Visualisation of the Cross-validation Method [6]

1) Training Set (70% of the given data): The actual dataset we use to train the model. The model sees and learns from this data [3].

2) Validation Set (30% of the given data): The validation set serves to evaluate the model frequently by fine-tuning the model hyper-parameters, as in input weights and biases. Hence the model occasionally sees this data but never "Learns" from it. By using this set, we can update higher level hyper-parameters. So the validation set affects a model, but only indirectly [3].

3) Test set (1): (3 random weeks time-frame of the preserved data, of 2016 and 2017). The sample of data was used to provide an unbiased evaluation of the performance of a final model fit on the training dataset. It is only used once the model is wholly trained (using the train and validation sets). The test set is what is used to evaluate competing models [3].

4) Test set (2): (3 weeks time-frame of data separately given for the 2018 years) Same characteristics as Test set (1).

B. Feature Selection

For this problem, this report proposes working on a data structure within a 2-week horizon, hence a 14-day time span. The aim is to predict the prices of days 8 to 14 - using the given prices from days 1-7 and the solar and wind data aligned with days 2-8. Note that there is a superposition in the dataset.
We did not include more features in the model because we realized that the auto-correlation of the data is relatively small when navigating through all the prices data points of the two-year horizon.In other words, if one takes a dataset of a random day, it is unlikely to be related to another day of the year. So it is not necessary to include another set of data.
Furthermore, data correlation is considerable between the datasets of Belpex and solar and wind data, nevertheless, the correlation is poor between solar and wind. This in turn means that as long as the output values of solar and wind are high, there will be a tendency of the prices to be low, and it is expected to see that in the price prediction.

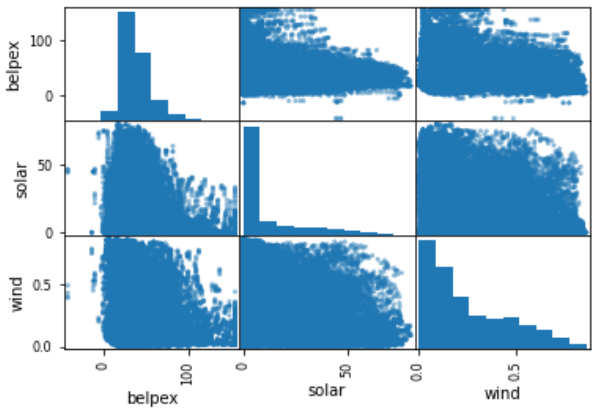

Fig. 2. Correlation Data of BELPEX, Solar and wind output

Considering the above, the inputs of the system will be solar generation, wind generation, and electricity prices, aiming to predict the next day/week (in our model) using data from the given dataset of wind and solar of the previous week and the previous week's prices.

C. Neural Network Architecture Selection

The constructed artificial neural network in this report is of type feedforward (aka. multiple-input), in which nodes' connections do not form a loop [3]. In other words, since all information flows in a forward manner, from input to output, the output of a given layer does not influence that same layer. In particular, we applied a multiple-input model with one single output. Because of the selected architecture, we cannot use the Sequential API [2] included in Keras[2]. Instead, the Functional API was used; also a framework to create neural networks.
This API operates in a way that loads the dataset points in a 7x24 shape as the standard time series. Hence, this corresponds to one week multiplied by 24 hours for different batches of weeks. This means that all the datasets are weeks of the year divided into seven days and 24 hours of data for each day. We split the 2-year dataset using shift/lag - thus creating around 52 samples of 7 days and 24 prices a day.
It is understood that there are three inputs of different data types (solar generation, wind generation and electricity prices), so all the datasets are split in that shape to be consistent with the topology of the neural network. After this, the three inputs are concatenated to make a single input for the neural network's first layer. We experimented with different topologies regarding the number of neurons, but the base topology was established into three layers (input, hidden, and output), similar to the figure's 3 example.

[2]an open-source software library that provides a Python interface for artificial neural networks

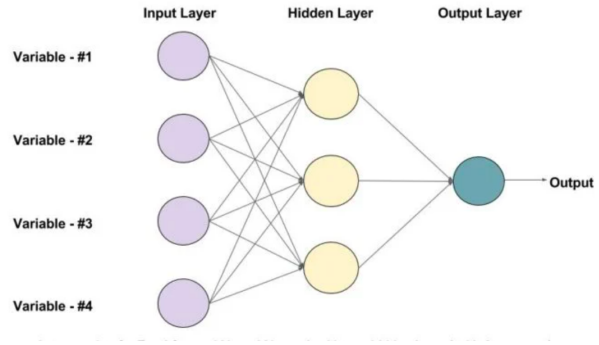

Fig. 3. Example of a Feed-forward Neural Network with one hidden layer (with 3 neurons) [3]

As the problem consists of a price prediction, we want to estimate values. Thus, we used the ReLU function for the input and hidden layer and the Linear function for the output layer. We did not use another activation function as most are designed for categorization, while the chosen ones are mainly used for value predictions (source). We tried to invert the applied function's order/combination (e.g. "Linear,ReLU ,ReLU"), but the results were worse than the combination "ReLU ,ReLU ,Linear", so we preferred the first option. In further sections the details of the neural networks are discussed, as well as the results.

D. Input (Test) Data

The input data or test data is meant to be not known by the model created in order to confirm the accuracy of the model. In this case, data from 3 weeks was provided for BELPEX (electricity prices), solar output and wind output. For prices, 7 days were provided and for solar and wind 8 days were provided, nevertheless, given the construction of the model, it was decided to use only 7 days instead to fit the standard shape of the input. Data was normalised in order to fit with the training/validation dataset. For the case of solar, the values were divided by the total installed capacity of PV by 2018 and multiplied by 100%, the value used was 4254 MW **??**. For wind, a similar operation was performed, using the total installed capacity of 3268 MW**??**. The input data given and normalised are shown in the figures 4, 12 and 6.

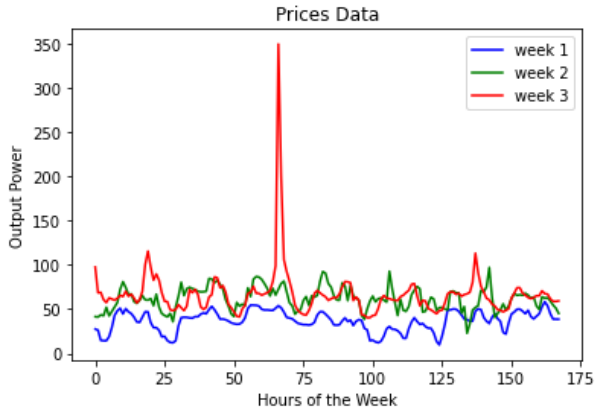

Fig. 4. Prices data for input test data.

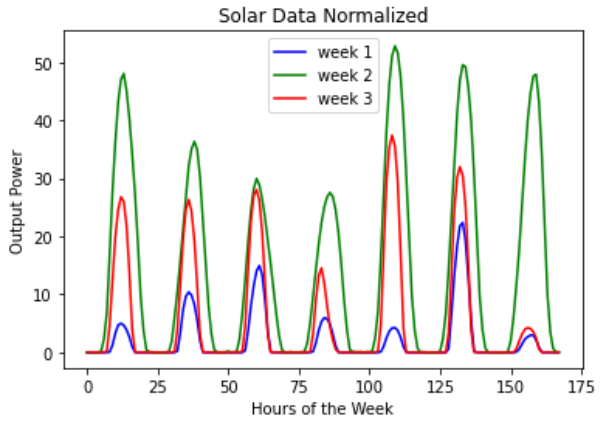

Fig. 5. Solar data for input test data.

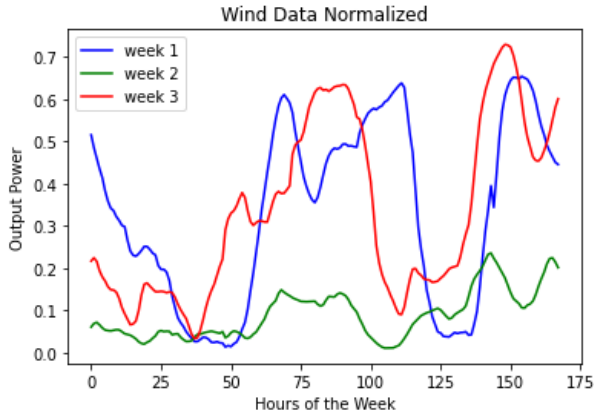

Fig. 6. Wind data for input test data.

For the training data, we assured that outliers were eliminated. Reflecting on the trends and seasonality of the dataset, it would be to expect that data regarding solar and wind would reveal some seasonality as these change depending if it's winter or summer. However, the effect will not be seen as the time window output is small. Since the model responds to inputs of the previous week, we assume that the influence of seasonality will not have a relevant influence on the output results, which justifies the selection of only 3 inputs.

IV. Proposed Solution Models

For the problem of this report, we explored three different solution models. The models were compared under the same loss model, the Mean square error (MSE), and follow the same metrics for comparison and optimization, which is accuracy. Since they are based on the same framework of comparison, the difference between them is the number of neurons in each layer. Note that we are inputting one week-long, 24-hour time-base data and we expect to output a prediction for 7 days, after this, the first day is retrieved as final result. Regarding different combinations for the number of neurons for each layer and we aimed to create a quasi-sequential network (two sequential layers) but using different inputs **??**. The structure of the models are shown in the table I:

For more information on the proposed solution models consult appendix A.

	Layer		
Solution Model Proposed	Input	Hidden	Output
Model 1	50	50	24
Model 2	60	48	24
Model 3	75	150	24

TABLE I. Solution Models: number of Neurons per Layer

V. Final Prediction and Observations

The assessment criteria for this report is the performance metric MSE of the training set, the mean square error. Looking at the results of the 3 models, we compare the 3 proposed solution models based on the performance metric for the dataset "Test set (1)" and "Test set (2)".

It appears that the best combination we could find is model 3, as it shows a performance metric of mean square error at 15.6. Hence, Model 3 showed the best performance of the alternatives: for a determined number of iterations - number of batches - these values are the best price prediction set we have found for a determined testing data set of the initial two year's data.

In figure 7, the model's loss value can be verified after 1200 iterations (epochs). A quick convergence on the three models is observed at approximately 200 iterations.

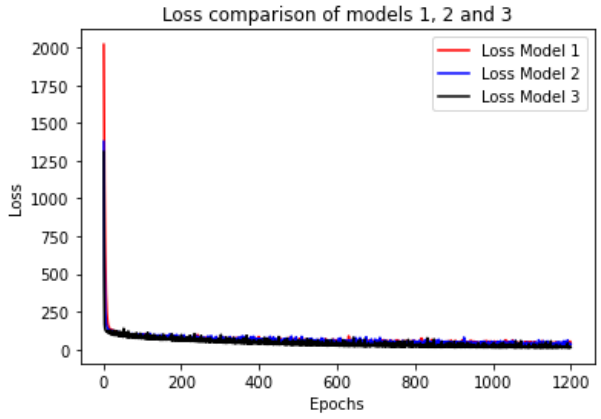

Fig. 7. behaviour of Loss (mean sqared error) for the different models tested

On the other hand, taking a random separated test data from the initial dataset (see previous sections), the model was tested in order to check how accurate compared to the actual data. In figure 8, the predicted values can be compared with the actual data.

Regarding the provided test data for 2018 (see figures figures 4), it can be observed that the prices of week 3 (input data or test data) show a peak prices during the 3rd day (around 350 EUR/MWh) that can be associated with the low input of solar at that time (between 9pm and 1am) and possibly a high gas price at that moment that triggered a high cost of electricity (given the merit order on the whole sale market). However, given that the outliers of the training set were eliminated, no peaks are expected in the output, in other words, the predicted prices will most likely look smooth throughout the day. The estimated prices can be observed in the figure 9

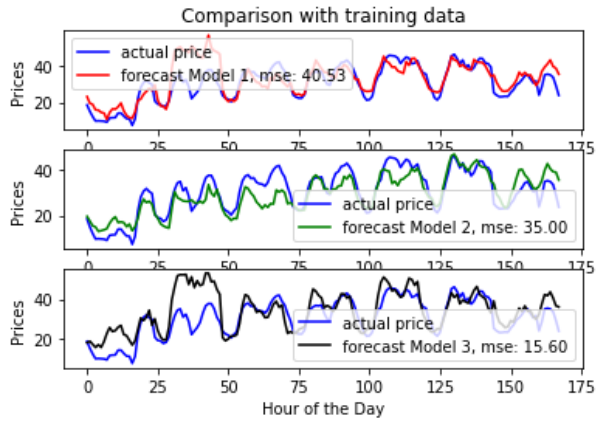

Fig. 8. Comparison of the three models using a common test data

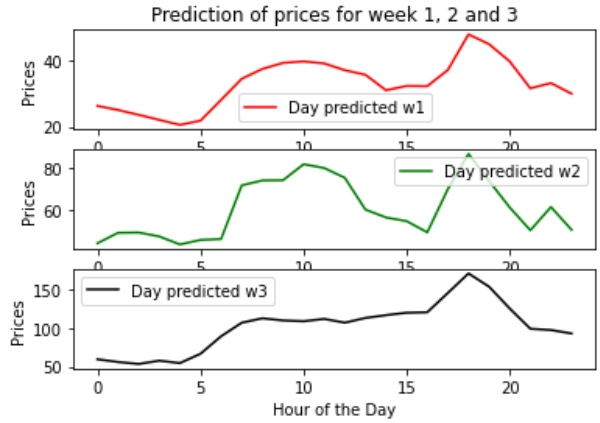

Fig. 9. Day-ahead price prediction for the three weeks

APPENDIX A
FURTHER INFORMATION ON THE ARCHITECTURES OF THE PROPOSED NEURAL NETWORK MODELS

A. Model 1 - (50 input layer; 50 hidden layer; 24 output layer)

Model: "neural_network_1"

Layer (type)	Output Shape	Param #	Connected to
solar_input (InputLayer)	[(None, 7, 24)]	0	
wind_input (InputLayer)	[(None, 7, 24)]	0	
plweek_input (InputLayer)	[(None, 7, 24)]	0	
concatenate_12 (Concatenate)	(None, 7, 72)	0	solar_input[0][0] wind_input[0][0] plweek_input[0][0]
input_layer (Dense)	(None, 7, 50)	3650	concatenate_12[0][0]
layer_2 (Dense)	(None, 7, 50)	2550	input_layer[0][0]
output_layer (Dense)	(None, 7, 24)	1224	layer_2[0][0]

Total params: 7,424
Trainable params: 7,424
Non-trainable params: 0

Fig. 10. Model 1 Architecture Summary

B. Model 2 - (60 input layer; 48 hidden layer; 24 output layer)

Model: "neural_network_2"

Layer (type)	Output Shape	Param #	Connected to
solar_input (InputLayer)	[(None, 7, 24)]	0	
wind_input (InputLayer)	[(None, 7, 24)]	0	
plweek_input (InputLayer)	[(None, 7, 24)]	0	
concatenate_13 (Concatenate)	(None, 7, 72)	0	solar_input[0][0] wind_input[0][0] plweek_input[0][0]
input_layer (Dense)	(None, 7, 60)	4380	concatenate_13[0][0]
layer_2 (Dense)	(None, 7, 48)	2928	input_layer[0][0]
output_layer (Dense)	(None, 7, 24)	1176	layer_2[0][0]

Total params: 8,484
Trainable params: 8,484
Non-trainable params: 0

Fig. 11. Model 2 Architecture Summary

C. Model 3 - (75 input layer; 150 hidden layer; 24 output layer)

Model: "neural_network_3"

Layer (type)	Output Shape	Param #	Connected to
solar_input (InputLayer)	[(None, 7, 24)]	0	
wind_input (InputLayer)	[(None, 7, 24)]	0	
plweek_input (InputLayer)	[(None, 7, 24)]	0	
concatenate_14 (Concatenate)	(None, 7, 72)	0	solar_input[0][0] wind_input[0][0] plweek_input[0][0]
input_layer (Dense)	(None, 7, 72)	5256	concatenate_14[0][0]
layer_2 (Dense)	(None, 7, 150)	10950	input_layer[0][0]
output_layer (Dense)	(None, 7, 24)	3624	layer_2[0][0]

Total params: 19,830
Trainable params: 19,830
Non-trainable params: 0

Fig. 12. Model 3 Architecture Summary

APPENDIX B
SUPPORTING FILES

The code, the trained models and the predictions file obtained from the proposed solution models can be found attached to this document as files of type *.py* in a compressed *.zip*, and in *predictions.csv*

REFERENCES

[1] Team, K. Keras documentation: Keras layers API. (keras.io,2021), https://keras.io/api/layers/

[2] Chollet, F. Deep Learning with Python. (Manning, Cop,2018)

[3] Gupta, V. Understanding Feed-forward Neural Networks. (LearnOpenCV,2017,10), https://learnopencv.com/understanding-feedforward-neural-networks/

[4] Wilkin, B. Strategic PV Analysis and Outreach. (IEA International Energy Agency,2018), https://iea-pvps.org/wp-content/uploads/2020/01/NSR_Belgium_2018.pdf

[5] Jaganmohan, M. Belgium: installed wind power capacity 2021. (Statista,2022,5), https://www.statista.com/statistics/1130448/total-wind-power-in-belgium/

[6] Goyal, C. Importance of Cross Validation: Are Evaluation Metrics enough?. (Analytics Vidhya,2021,5), https://www.analyticsvidhya.com/blog/2021/05/importance-of-cross-validation-are-evaluation-metrics-enough/

[7] Yuso Yuso — Wat zijn Belpex prijzen?. (yuso.be,2020,4), https://yuso.be/blog/showonhomepage/wat-zijn-belpex-prijzen-2/