Electricity Price Forecasting

Margaret Toohey

Capstone Project Report

Udacity Machine Learning Nanodegree, Nov 2020

Definition

Overview

Electricity price forecasting (EPF) is concerned with the prediction of the price of wholesale electricity, for a region or energy market. The energy market is a highly deregulated and controlled market, and many of the markets across Europe, the USA and Australia trade under strict market rules using spot & derivative contracts. For this reason, in order for participants in the markets to successfully win those contracts, accurate EPF is crucial. [1]. According to Weron et al, accurate price prediction is fundamental for energy companies' decision making mechanisms. [2]

Factors Effecting the Electricity Market

Like any commodity, accurate price prediction is crucial for energy companies and suppliers to be able to bid into and participate in the energy market. Unlike other commodities, the stability of power systems is bound by a constant balance between production and consumption, and excess electricity on the system cannot be stored. Another unique feature of the electricity market is the high variability in the demand, based on weather (temperature, wind-speed, sunlight), time of the day (on-peak vs. off-peak) and more long term seasonal effects (winter, summer, weekends, holidays). [1] The variability caused by these factors leads to a market unlike other commodities markets, exhibiting both seasonal trends and unanticipated price spikes or drops.

Another, more recent, factor in the EPF is the amount of Renewable Energy Sources (RES) that are available to the market. RES (e.g. Wind [3], solar and hydro [4] etc. electricity sources) have an impact on the price of electricity. Generally the generation of RES leads to a decrease in the wholesale price of electricity [5] since the RES marginal costs are close to zero. The drawback, however, is that the amount of RES available to the power system is not constant, it is variable over a number of factors, such as weather, and therefore not as stable an energy source as traditional fossil fuel or nuclear generation.

Statistical Methods

Traditionally, statistical methods have been used for EPF, methods such as time series and regression models have been used with good effect, but due to the complexity of the markets, predictions from these models do not give accurate long term EPF. Advanced statistical methods such as bootstrapping, distribution-based probabilistic forecasts and quantile regression averaging. The bootstrap method was developed by Efron et al in 1979

[6] which solved the problem of estimating the sampling distribution of some random variable on the basis of observed data. This method has been successfully applied to EPF for short term predictions on high density forecasts, but begins to suffer in accuracy after longer prediction time periods. [7]. Distribution-based forecasts can be used not only to predict the price of electricity, but for forecasting wind power, like in Wu et al. [8] Quantile regression averaging (QRA) is a method by which quantile regression is applied to the point forecasts of a number of forecasting models, introduced by Weron et al [1] particularly for the application of EPF. QRA and has been extended by various methods to automate the model selection, such as using Principle Component Analysis (PCA) [9] for use prediction spot prices of electricity. This approach can also be used to investigate the effect of various exogenous variables on the market, such as the impact of (RES) on electricity price. [10]. The main issue with statistical methods for EPF is the accuracy over longer term forecasts. [11]

Machine Learning Methods

The recent computational advancements have led to an increase in popularity in Machine Learning (ML) methods. Artificial neural networks (ANN) have been used for short term EPF [12], [13]. Deep neural networks (DNN) have been used frequently for prediction of day ahead electricity prices [14] [11]. Hybrid models have also been used, such as in [15] when the authors combined a wavelet transform, ARMA and kernel based machine learning for day-ahead EPF. Radial basis function neural networks (RBFN) have also been used in combination with hybrid methods for day-ahead EPF. [16] Recurrent neural networks (RNN) are natural choices of algorithm for the EPF problem due to their temporal nature. Unlike other feedforward neural networks, RNN maintain an internal state, and so makes them applicable to prediction of time series and temporal data. RNN have been used to model many electricity markets, such as the Turkish market [17] and the European Power Exchange [18]. As forecasting is essentially a regression problem, linear models have also had success at day-ahead price modelling. Linear models are a form of supervised ML problem, where a real valued variable is predicted, given a set of input features. The input features can be high dimensional vectors, leading to parameter rich regression models which are suitable for forecasting. The method of Least Absolute Shrinkage and Selection Operator (LASSO) has been used for forecasting and has given accurate results. [19], [20], [21]. LASSO regression uses shrinkage to shrink data towards a centralized point (for instance, the mean) [22] and LASSO is better suited to sparse models with fewer features. Since the day-ahead price of electricity is linearly variable on its own previous value and also stochastically variable on exogenous variables, traditionally auto-regressive (AR) models have be used, such as the Auto-regressive Moving Average (ARMA) and the Auto-regressive Integrated moving Average (ARIMA). Another AR method for time series models is the Linear Estimated Auto-regressive model (LEAR), which is based on the LASSO method but with AR.

As indicated in the review paper of Lago et al [11], a common issue throughout publications on ML techniques for EPF is the testing periods are too short to yield statistically significant results, and in some cases only used test datasets of one week periods. The issue with short time periods like this is that there is a strong seasonal component to electricity prices, and also “special days” such as holidays etc. are not accounted for in these timelines.

After extensive literature review, the group recommend using DNN or LEAR as the optimal ML approaches for EPF, and recommended using at least 4 years of data from the same market for training, and then two years for testing. This recommendation formed the basis of the problem statement for this project. Initially I investigated using a deep learning model (DeepAR) for the task and evaluating using the DNN of Lago et al, however upon reading the DeepAR best practices [23], DeepAR performs well when the input feature space is large, and they recommended using a standard forecasting algorithm for a smaller set of input

features. For this task, I decided to use the Linear Learner [24] algorithm from Amazon SageMaker, and use the LEAR model from Lago et al for benchmarking.

Personal Motivation

There were a number of reasons I chose the domain of EPF for the capstone project. I have always had an interest in renewable and sustainable energy sources, and personally I feel strongly that society as a whole should be maximizing the RES in the energy market. As seen from the literature, RES plays a large role in the day-ahead price. An increase in EPF accuracy could lead more market participants to use RES, and will also encourage government bodies to promote investment in renewable schemes [25].

From a pragmatic perspective, as a software engineer, I wanted to use the capstone project to work on an area I wouldn't normally get to study, like financial models and market predictions, so combining these reasons led me to choose Electricity Price Forecasting as the topic of my project. In addition, I wanted to further my understanding of the Linear Learner algorithm and apply it to a regression problem, as it is a different context to the classification problem given in the project.

Problem Statement

As mentioned above, after extensive literature review, Lago et al recommended using LEAR for the problem of linear modelling of electricity price forecasting [11], and also recommended using at least 4 years of data for training and testing.

EPF (like any forecasting problem) is a time-series, prediction problem. Given historical data on how a variable changes over a time-series, the goal of a forecasting problem is to accurately predict the value of the variable for the next time step. Given the historical data, we can frame this prediction problem as a supervised machine learning problem, giving a number of time series as an input, and the variable of interest as the label.

The goal of this project is to use 6 years of historical electricity price data, for the Scandinavian region from the NordPool energy group, train a Linear Learner model, which could be used by energy companies for day-ahead price forecasting and hence successful bidding into an energy market.

Metrics

Initially, when creating a project proposal, I planned on using the Mean Average Percentage Error (MAPE) and the relative Mean Absolute Error (rMAE) and the Root Mean Squared Error (RMSE) to calculate the accuracy of my model, since MAPE, rMAE and RMSE are commonly used to quantify the error of forecasting models [26]. However, upon review of my project proposal, the reviewer recommended some reading material about best practice for forecasting problems [27] in which the author states that the MAE is the correct metric for seasonal time series data, rather than rMAE, so this is the metric used for evaluation of the project.

Mean Absolute Percentage Error – MAPE

The MAPE of a series is average value of the relative differences between the predicted and actual values. It can also be used as a loss function for regression problems, but in our case we will be using it to evaluate the accuracy of our forecasted values for the day-ahead price of electricity. We will multiply the calculated MAPE by 100% so it can be reported as a percentage.

$$\text{MAPE} = 100T^{-1}\sum_{t=1}^{T}|y_t - \hat{y}_{t|t-1}|/|y_t|$$

Mean Absolute Error – MAE

The MAE of a time series or forecast is calculated by taking the sum of the errors between predicted and actual data, as a fraction of the total time in the series or forecast. Intuitively it is the arithmetic average of the absolute errors, and is used to measure the forecast error of time series.

$$\text{MAE} = T^{-1}\sum_{t=1}^{T}|y_t - \hat{y}_{t|t-1}|$$

Root Mean Squared Error - RMSE

The root mean square error is calculated by taking the square root of the sum of the square of the difference between the predicted value and the actual value, see below. RMSE is appropriate for calculating errors between the prediction and the actual value, rather than comparing across different datasets, so it is suitable for the problem described here. RMSE is used for a wide variety of accuracy estimation, not just in forecasting.

$$\text{RMSE} = \sqrt{T^{-1}\sum_{t=1}^{T}(y_t - \hat{y}_{t|t-1})^2}$$

Analysis

Data Exploration

The data used in this project is from the openly available historical price data from the NordPool group at https://www.nordpoolgroup.com/historical-market-data/. This dataset contains:

- The day-ahead price of electricity – unit Euro/MWh
- The day-ahead load forecast (the total amount of power being consumed from the electrical power grid) – unit MWh
- The day-ahead wind generation forecast – unit MWh

The data is give in hourly increments, starting in January 2013 (the first data point is 2013-01-01 00:00:00) up until Dec 2018 (2018-12-24 23:00:00) and there are no data points missing from the dataset.

Data Visualization

The three variables (day-ahead price, day-ahead load forecast and day-ahead wind generation) are plot as a function of time in Figure 1. There are a number of observations to be made about the data. It can be seen from the graph that the data in each series is of vastly different scale. In fact, the values of the load data (Exogenous 1) and the wind

generation (Exogenous 2) are so much larger than the price that it's hardly visible on the plot.

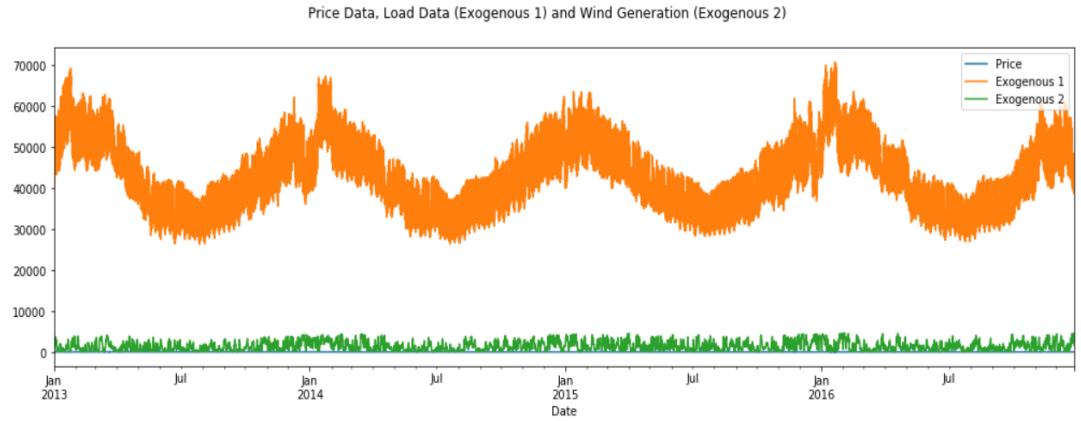

Figure 1: Time series representing the price (blue), load (orange) and wind generation (green) of the NordPool group from Jan 2013 to Dec 2018.

In order to better understand the input and output data, observations will be discussed on each time-series individually.

Day-Ahead Price

The day-ahead price forecast (in Euro per MWh) for the Nordic regions, between Jan 2013 and Dec 2018 can be seen in Figure 2. The first thing we note is the data is almost always greater than zero, and there is rarely price spikes. Another observation of the price is how different the scale of the data is to the exogenous variables, with values between €0 and €100/MWh for most of the series.

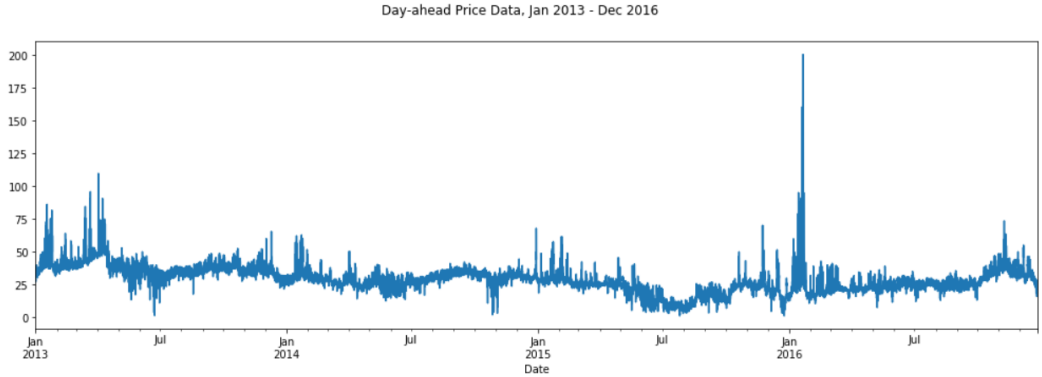

Figure 2: The price of electricity in the NordPool group market, from Jan 2013 to Dec 2018

Day-Ahead Load Forecast

The day-ahead load forecast for the same region can be seen in Figure 3. The load is the overall energy demand on a power system, in MWh. Figure 3 clearly shows a seasonal trend in the load of the NordPool region, where demand is generally lower during the summer months than in the winter months. This trend is expected, as summers in the Nordic regions aren't very warm, but the winters are quite cold, so energy consumption would increase in the winter. There also appears to be a higher frequency trend, within the seasonal trend, likely due to the off-peak/on-peak nature of energy demand. Presumably this correlates to morning and evening times having a higher energy consumption than during the night, for

example. The scale of this data is much larger than the price data, where most values fall between 20,000 and 70,000 MWh over the time series.

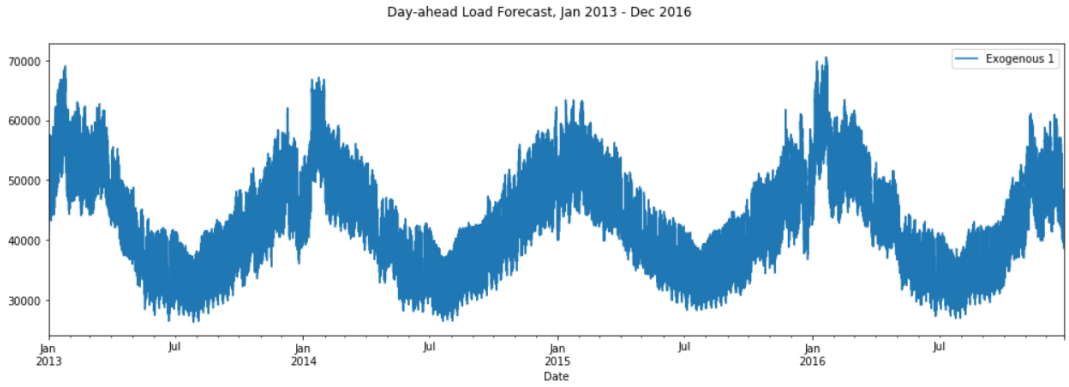

Figure 3: Day-ahead Load on the NordPool system, in MWh, from Jan 2013 to Dec 2018 in hourly increments

Day-Ahead Wind Generation Forecast

Figure 4 shows the forecasted wind generation, in MWh. As expected, the wind generation is noisier than the load data, and there is a slight seasonal trend visible in the data, with more wind generation over the winter months than the summer. The scale of this data is also significantly smaller than the load forecast, with most values falling between 0 and 400 MWh.

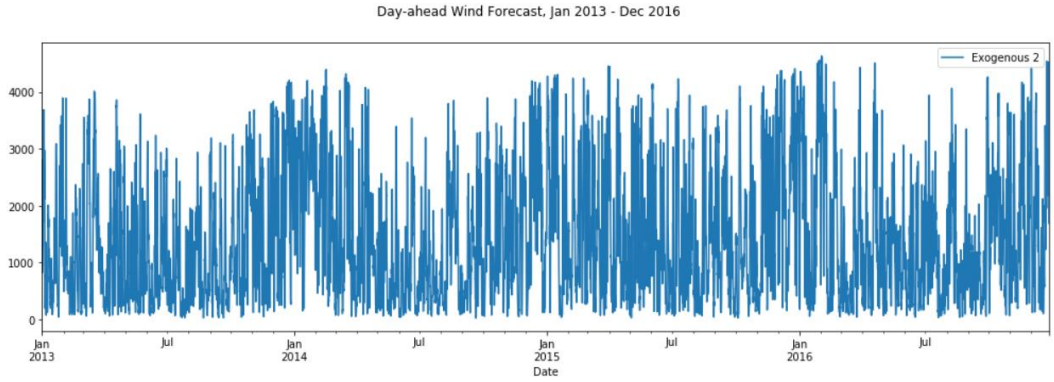

Figure 4: Day-ahead Wind Generation forecast, in MWh, for the Nordic region, from Jan 2013 to Dec 2018, in hourly increments.

Algorithms and Techniques

The model chosen to predict the price of electricity is the Linear Learner model from Amazon SageMaker. Linear Learner is a supervised learning algorithm for classification or regression ML problems, including forecasting scalar time series [24]. The algorithm learns a linear function, given a set of input features and a training label, and then maps those inputs to an approximation of the label. It can take, as an input, a high dimensional vector, and gives a scaler output, or a classification. In our case, we have multiple time series as an input and we want to forecast the day-ahead price of electricity, which is a scalar quantity, so Linear Learner is a suitable choice of model.

Benchmark

The model used to benchmark the Linear Learner model is the open source LEAR model in the EPF toolbox, from Lago et al [28], which has been specifically created for benchmarking linear models. The LEAR model provided is specifically tailored to electricity price forecasting, and the features and hyper-parameters have already been optimized for the task. According to the research group, LEAR is the most accurate linear model, so is arguable the gold standard by which to compare linear models to. This package also contains a method for evaluating the accuracy of the predictions from the trained model, and this includes a method for MAPE, MAE and RMSE. This allows a direct comparison to the accuracy of my model.

Methodology

Data Preprocessing

[Note: The data preprocessing & implementation outlined here were done in a Jupyter notebook, running a python3 kernel on a SageMaker notebook instance, filename: Electricity_price_forecasting-0-0-1.ipynb]

Training Data:

Lago et al recommend a training set of 4 years data, so the test dataset contained the values from 01 Jan 2013, at 00:00:00 until 26 Dec 2016 at 23:00:00, in hourly increments.

Test Data:

The test data is the final two years of data in the set, and contains the values from 27 Dec 2016 at 00:00:00 until 27 Dec 2018 at 23:00:00, in hourly increments.

Scaling

As mentioned above, the scales of the data on each feature column are vastly different. Unscaled data could lead to the model weighting higher values more than lower ones, despite difference in units. In order to overcome this, we can scale our data by a scaling factor, ensuring the whole dataset is standardized across all units and scales. The EPF toolbox provides a method for data scaling, with a number of scaling options:

- Normalized to the interval [0,1]
- Normalized to the interval [-1, 1]
- Standardized to follow a normal distribution
- Normalized on the median value of the data
- Scale the data based on the asinh transformation.

The scaling factor I chose was to normalize the data on the interval [0, 1], and the scaled data is shown in Figure 5.

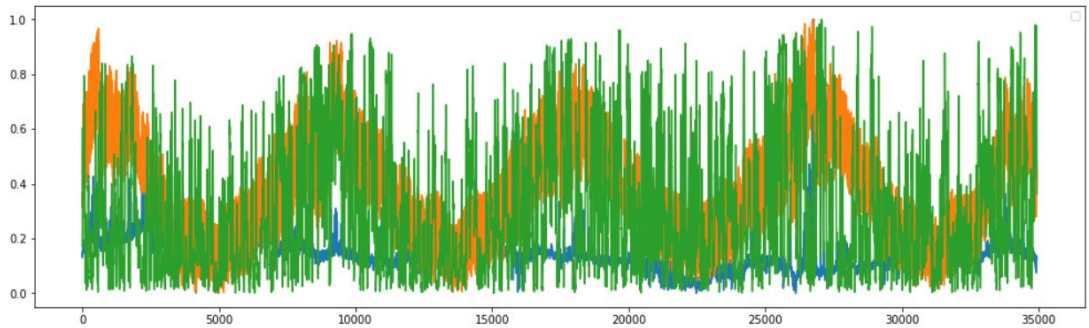

Figure 5: Price, load and wind generation data after scaling to values between 0 and 1.

For this projects, a scaling transform was fit on the training input features, and then applied to the test input features. Another scaler object was fit on the training labels.

Implementation

Training

After splitting and scaling our data appropriately, the data is ready to be trained on, by doing the following:

1. Convert the training data to recordIO-wrapped protobuf format, as SageMaker algorithms require. This was done using the sagemaker.amazon.common and io packages.
2. Upload the converted data to S3. The learner algorithm runs on a compute instance in the cloud, so the data must be uploaded to S3 and the location saved, to be passed to the estimator during training.
3. Get the Image URI for the linear learner algorithm. This is essentially a Docker container, with all the dependencies needed for the algorithm to run.
4. Create an estimator object. When creating an estimator, we pass in the image URI, our IAM role, some specifics about the compute instance we want training to be run on and the path we want the model artifacts to be stored. For this project, the instance type used for training was an m1.c4.xlarge.
5. Set the hyper-parameters, which were set as follows:

    - Feature_dim = 2
    - Mini_batch_size = 100
    - Predictor_type = ‘regressor’
    - Epochs=20
    - Loss=’absolute_loss’

6. Fit the model with the training data, by calling the fit method on the estimator object we created in step 4, and passing in the location to our training data in S3.

Deployment

Deployment refers to the way one can access the model created from training, and pass in test data in order to get a prediction from the model.

The steps in this implementation were as follows:

1. Create an endpoint, for the trained model to be deployed to.

2. Specify the serializer and deserializer the endpoint should use, when passing in or out data from the model. This means test data doesn't need to be converted to recordIO-wrapped format like during training, which is more convenient for testing.
3. Pass the test data into the endpoint.
4. In our case, since the data was scaled before training, the output data needs to be inversely scaled before comparison with the actual data.

Results

The predicted values vs the actual values are shown in Figure 6.

The predicted day-ahead price under-represented some of the price spikes in the actual data, and generally had a lot less noise than the actual data. This lead to a poor MAPE and large RMSE, as seen below:

- MAPE = 22.9%
- RMSE = 10.78

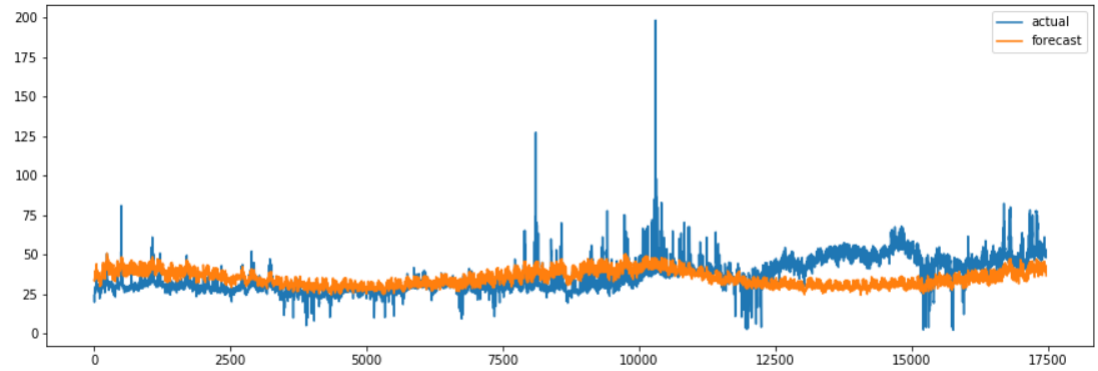

Figure 6: Predicted values, using the model described above compared to the actual values for day-ahead price.

Refinement

[Note: The data processing outlined here is in Jupyter notebook, filename Electricity_price_forecasting.ipynb, running on a python3 kernel, on a SageMaker notebook instance]

As seen from Figure 6, the accuracy of the model was not very good, so I then made some adjustments to the implementation.

Feature Engineering

The main issue with the predicted values in the version outlined above, is that they didn't track any of the spikes or dips in the real data. This was likely an artifact of not enough features in the input data.

As mentioned in the Data Visualization section, both inputs to the model have trends over different timescales. This means that the time of the day, week or year may have a role in the forecasted price of electricity, so it makes sense to extract these features and pass them into the model, as separate time series. The features extracted were as follows:

1. The day of the week, as a number (Monday = 0, Sunday = 6) – This allows the algorithm learn the weekly trends of the data, such as weekends and week days.
2. The week ordinal of the year (0 – 51) – This allows the model learn about the seasonal trends of the data.
3. The month, as a number (0 – 11) – This allows the model learn about the seasonal trends of the data.
4. The hour of the date-time (00 – 23) – This will allow the model learn some of the off-peak, on-peak characteristics.
5. In addition to the various date and time features, 7 days of lag features for the price data was included in the input data.

After feature extraction, the total dataset looked like below.

	Price	Exogenous 1	Exogenous 2	weekday	Week	Month	Hour	Day	lag1	lag2	lag3	lag4	lag5	lag6	lag7
Date															
2013-01-03 14:00:00	36.42	55768.0	3686.0	3	1	1	14	3	36.40	36.51	36.60	36.71	36.52	36.38	35.82
2013-01-03 15:00:00	36.58	56413.0	3679.0	3	1	1	15	3	36.42	36.40	36.51	36.60	36.71	36.52	36.38
2013-01-03 16:00:00	36.94	57285.0	3644.0	3	1	1	16	3	36.58	36.42	36.40	36.51	36.60	36.71	36.52
2013-01-03 17:00:00	36.94	57558.0	3553.0	3	1	1	17	3	36.94	36.58	36.42	36.40	36.51	36.60	36.71
2013-01-03 18:00:00	36.81	56793.0	3440.0	3	1	1	18	3	36.94	36.94	36.58	36.42	36.40	36.51	36.60

Figure 7: Screengrab of the head() of the dataset after feature extraction.

Similarly to the initial version, the training data was 4 years and the test data the final 2 years of the dataset. However unlike the initial version, the training data was further split into training and validation data, to be passed to the estimator.

Scaling

As per the initial version, the data was scaled using the EPT toolbox DataScaler object. Since the predicted forecast for the initial version lacked the variability of the actual data, I decided to use a scaling method that kept more variance than the normalization technique used initially. For this version, I used the scaling method of standardizing the data to follow a normal distribution, and the data after scaling can be seen in Figure 8.

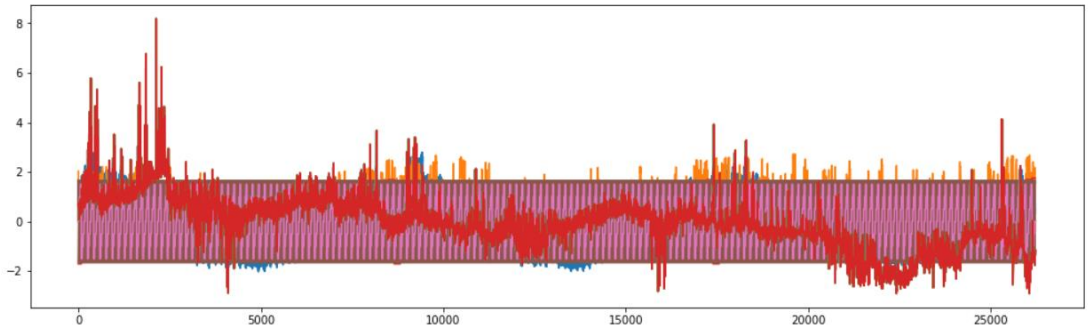

Figure 8: All the input features shown in the table in Figure 7, after scaling to follow a normal distribution. Note, the label data is not included in this as a separate scaler was fit for the inputs and output.

Hyper-parameter Tuning

The last refinement made to the model was to use the SageMaker hyper-parameter tuner to create a model, based on the best training job. The learning_rate and the mini_batch_size parameters were passed to the hyper-parameter tuner, along with bounding values. The hyper-parameter then iterated over number of different models, until the combination of parameters that minimized the objective loss was found. This training job was then attached to the estimator before deployment.

The rest of the steps were the same as in the initial version, outlined above.

Results

Model Evaluation and Validation

The results of the predictions from the test data are shown in Figure 9. The refinements to the model made a huge improvement to the accuracy of the forecast. The accuracy metrics, over the test dataset are:

- MAPE = 2.9%
- MAE = 1.01
- RMSE = 2.33

It can be seen from the graph that the price spikes and dips were accurately predicted with this model, so adding the date and time features was a crucial step. After feature engineering, a total of 14 input features were used for training. This allowed the timing and seasonal trends of the electricity demands be included in the final model, which lead to a far more accurate model for this data than the initial model.

Including a hyper-parameter tuning step also improved the model, and allowed the best learning rate and mini batch number for the data at hand.

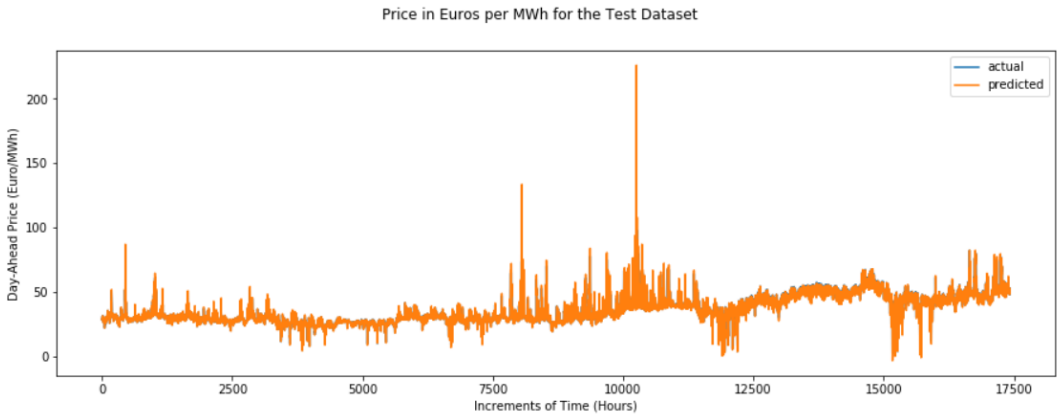

Figure 9: Predictions (orange) and actual data (blue) after using feature extraction, scaling and hyperparameter tuning.

As the exogenous variables for any electricity market are specific to the region, this model could only be used to forecast day-ahead price for the NordPool market. However initial results indicate that the model is reasonably robust against price spikes and drops, given the correct input features.

Justification

[Note: The benchmark model was implemented in a Jupyter notebook (Filename: LEAR_Benchmark_model.ipynb) using a python3 kernel on a SageMaker notebook instance]

The benchmark model used for comparison was the LEAR model from Lago et al [11] [28] which has been specifically designed as a bench mark for evaluation of linear models. The

LEAR model is implemented in the EPF Toolbox, which is an open source evaluation toolbox available in Python.

The dataset used is the same as was used for my model, with the first 4 years used for training and the last 2 years used for testing. The authors describe the input features for the LEAR model in the review paper [11] , namely:

- Historical day ahead prices of the previous three days and one week ago
- The day-ahead forecasts of the two variables of interest (load and wind generation)
- Historical day ahead forecasts of the variables of interest, the previous day, and one week ago
- A dummy variable representing the day of the week.

This results in a total of 247 input features for the LEAR model, which is significantly more than the 14 used in my model.

The results of the benchmark model, over the test dataset, are shown below:

- MAPE = 0.06 %
- MAE = 1.30
- RMSE = 3.48

The biggest difference in accuracy between my model and the LEAR model is the MAPE, where the value for my model was 2.9%, the LEAR model was only 0.06%. In the other two metrics, my model was slightly better (1.01 MAE vs 1.3 and 2.33 RMSE vs 3.48).

Timing

As a short note on timing, I want to point out here that the LEAR model and evaluation took ~ 12 hours to run, since it was running on the local SageMaker notebook instance. The linear learner was running on an m1.c4.xlarge instance and training and evaluation was approximately 6 - 9 minutes.

Conclusion

Reflection

While the LEAR model out-performed mine in terms of MAPE, the MAE and RMSE were comparable, which indicates my model was a good predictor of day-ahead electricity prices. The ability to run the linear learner algorithm on a more powerful compute instance means it was a much quicker algorithm to train and deploy, and hyper-parameter tuning didn't take long (~ 9 mins). The LEAR model took about x120 the time to train and evaluate. In my opinion, the MAPE accuracy sacrificed with the linear learner model is more than compensated for with the speed of training and deploying.

Improvement

I think the main area for improvement in this model would be to extract more features from the exogenous data, like they have done in the LEAR model. I only used lag values for the price data, but could have included lag values for the wind generation and load also. Also, the lag features I added were only for the last week, this could have been done differently, for example over different time periods, like with the LEAR model.

If there were more features extracted, the DeepAR model would be a potentially good choice for forecasting price, as that algorithm performs well with larger feature sets [23]. Future work could include using the DeepAR algorithm for training and then comparing predictions to the linear learner and LEAR models.

Bibliography

[1] R. Weron, "Electricity price forecasting: A review of the state-of-the-art with a look into the future,," *International Journal of Forecasting,* vol. 30, no. 4, pp. 1030-1081, 2014.

[2] R. W. Jakub Nowotarski, "Recent advances in electricity price forecasting: A review of probabilistic forecasting," *Renewable and Sustainable Energy Reviews,* vol. 81, pp. 1548-1568, 2018.

[3] G. B. B.-M. H. Carlo Brancucci Martinez-Anido, "The impact of wind power on electricity prices," *Renewable Energy,* vol. 94, pp. 474-487, 2016.

[4] L. P. M. P. Angelica Gianfreda, "The Impact of RES in the Italian Day–Ahead and Balancing Markets," *The Energy Journal,* vol. 37, pp. 161-184, 2016.

[5] F. y. Luigi Grossi, "Robust forecasting of electricity prices: Simulations, models and the impact of renewable sources," *Technological Forecasting and Social Change,* vol. 141, pp. 305-318, 2019.

[6] B. Efron, "Bootstrap Methods: Another Look at the Jackknife," *Annual Statistics,* pp. 1-26, 1979.

[7] R. S. Florian Ziel, "Probabilistic mid- and long-term electricity price forecasting," *Renewable and Sustainable Energy reviews,* pp. 251-266, 2018.

[8] B. Z. H. L. Z. L. Y. C. X. M. Junli Wu, "Statistical distribution for wind power forecast error and its application to determine optimal size of energy storage system," *Int. Journal of Electrical Power and Energy Systems,* pp. 100-107, 2014.

[9] J. N. R. W. Katarzyna Maciejowska, "Probabilistic forecasting of electricity spot prices using Factor Quantile Regression Averaging," *International Journal of Forecasting,* pp. 957-965, 2016.

[10] l. Maciejowska, "Assessing the impact of renewable energy sources on the electricity price level and variability – A quantile regression approach," *Energy Economica,* p. 85, 2020.

[11] G. M. B. D. S. R. W. Jesus Lagoa, "Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark," *Renewable and Sustainable Energy Reviews,* 2020.

[12] S. H. S. T. Nitin Singh, "A PSO-Based ANN Model for Short-Term Electricity Price Forecasting," *Ambient Communications and Computer Systems,* pp. 553-563, 2018.

[13] G. X. Yi Da, "An improved PSO-based ANN with simulated annealing technique," *Neurocomputing,* vol. 63, pp. 527-533, 2005.

[14] S. J. P. T. H. A. S. N. P. R. Radhakrishnan Angamuthu Chinnathambi, "Deep Neural Networks (DNN) for Day-Ahead Electricity Price Markets," in *IEEE Electrical Power and Energy Conference*, Toronto, 2018.

[15] L. C. L. L. Zhang Yang, "Electricity price forecasting by a hybrid model, combining wavelet transform, ARMA and kernel-based extreme learning machine methods,," *Applied Energy,* vol. 190, pp. 291-305, 2017.

[16] M. M. A. N. S. M. H. H. Javad Olamaee, "Day-ahead price forecasting based on hybrid prediction model," *Complexity,* vol. 21, no. 52, pp. 156-164, 2016.

[17] I. O. O. T. Umut Ugurlu, "Electricity Price Forecasting Using Recurrent Neural Networks," *Energies,* vol. 11, no. 5, p. 1255, 2018.

[18] Y. W. J. M. Q. J. Yiyan Chen, "BRIM: An Accurate Electricity Spot Price Prediction Scheme-Based Bidirectional Recurrent Neural Network and Integrated Market," *Energies,* vol. 12, no. 12, p. 2241, 2019.

[19] J. N. R. W. B. Uniejewski, "Automated variable selection and shrinkage for day-ahead electricity price forecasting," *Energies,* vol. 9, no. 8, p. 621, 2016.

[20] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society,* vol. 10, pp. 267-288, 1996.

[21] R. W. F. Ziel, "Day-ahead electricity price forecasting with high-dimensional structures: Univariate vs. multivariate modeling frameworks," *Energy Economics,* vol. 70, pp. 396-420, 2018.

[22] S. Glen, "Lasso Regression: Simple Definition," StatisticsHowTo.com, 2020. [Online]. Available: https://www.statisticshowto.com/lasso-regression/. [Accessed 23 11 2020].

[23] "Best Practices for Using the DeepAR Algorithm," AWS, 2020. [Online]. Available: https://docs.aws.amazon.com/sagemaker/latest/dg/deepar.html#deepar_best_practices. [Accessed 22 11 2020].

[24] AWS, "Linear Learner," AWS, [Online]. Available: https://docs.aws.amazon.com/sagemaker/latest/dg/linear-learner.html. [Accessed 22 11 2020].

[25] G. Wilkins, "Renewable Energy Support Mechanisms – An Irish Perspective," 22 07 2020. [Online]. Available: https://www.capspire.com/renewable-energy-support-mechanisms-an-irish-perspective-by-graham-wilkins/. [Accessed 11 21 2020].

[26] Y. W. Shuman Luo, "A two-stage supervised learning approach for electricity price forecasting by leveraging different data sources,," *Applied Energy,* vol. 42, pp. 1497-1512, 2019.

[27] R. J. Hyndman, Forecasting: Principles & Practice, Western Australia: UWA, 2014.

[28] J. Lago, "https://epftoolbox.readthedocs.io/en/latest/index.html," Read the Docs, 2020. [Online]. Available: https://epftoolbox.readthedocs.io/en/latest/index.html. [Accessed 22 11 2020].

[29] F. N. Luigi Grossi, "Robust forecasting of electricity prices: Simulations, models and the impact of renewable sources,," *Technological Forecasting and Social Change,* pp. 305-318, 2019.

[30] J. Circle, "Quantile Loss Function for Machine Learning," Evergreen Innovations, 2020. [Online]. Available: https://www.evergreeninnovations.co/blog-quantile-loss-function-for-machine-learning. [Accessed 21 11 2020].

[31] *https://www.nordpoolgroup.com/historical-market-data/,* 2020.

[32] "DeepAR Forecasting Algorithm," AWS, 2020. [Online]. Available: https://docs.aws.amazon.com/sagemaker/latest/dg/deepar.html. [Accessed 22 11 2020].