Domain Background

At the corporate level, electricity price forecasts have become a fundamental input to energy companies' decision-making mechanisms. Price forecasts, from a few hours to a few months ahead, have become of particular interest to power portfolio managers. A generator, utility company or large industrial consumer who is able to forecast the volatile wholesale prices with a reasonable level of accuracy can adjust its bidding strategy and its own production or consumption schedule in order to reduce the risk or maximize the profits in day-ahead trading.

Germany and France are the two main electricity markets in Central Europe. Both markets are large producers of electricity that export to all the countries around them, and this is because they usually have the lowest prices of the markets of the central-west part of the continent. But Germany and France have a very different energy mix. While France produces three quarters of its electricity from nuclear energy, Germany covers its production largely with coal. The dependence on coal is one of the reasons why Germany wanted to be a reference in the revolution of renewable energies and right now is the first country in terms of expansion and implementation of renewable energy for the production of electricity. An aggressive policy of energy transition placed it as the European leader in renewable energy, but this fact leads to new challenges.

High renewables penetration is a challenge for the electricity market. Grid operators are obliged to feed-in renewable electricity independent of the market price. However, the spot electricity price is not independent from renewable generation as variable renewable power production is negatively correlated with the electricity price. Whenever large volumes of intermittent renewable electricity are fed into the power grid, the electricity price tends to decline. As renewable installations have almost zero operational generation cost, they are certainly dispatched to meet demand. More expensive conventional power plants are crowded out, and the electricity price declines.

On the other hand, intermittent renewable power not only influences price level, but also price increases its volatility being a challenge to accurately forecast.

Problem Statement

In this project I propose to develop a Wholesale Energy Price Estimator based on Machine Learning tools and use it to forecast the day-ahead spot price for Germany, trying to overcome the above mentioned issues derived from the high renewables penetration, namely decline trend of price level as renewable capacity increases over the years (meaning non-stationary behavior) and high volatility (leading to high peaks and the necessity of dense granularity of input data).

Solution Statement. Algorithms and Techniques

There are several time series forecasting models, being the most common classical methods:

- Exponential smoothing model (ETS)
- Autoregressive models (ARIMA, SARIMA,..)
- Dynamic Linear Model (GLM)

These models can deal with relatively simple trends and seasonal patterns such as quarterly and monthly data. However, higher frequency time series often exhibit more complicated seasonal patterns. For example, daily data may have a weekly pattern as well as an annual pattern. Hourly data usually has three types of seasonality: a daily pattern, a weekly pattern, and an annual pattern. Such multiple seasonal patterns are becoming more common with modern high frequency data recording. Most of the methods we have listed before are unable to deal with these seasonal complexities.

To deal with such complex series, we will use Neural Networks models (RNN) that are based on simple mathematical models of the brain. They allow complex nonlinear relationships between the response variable and its predictors.

So we propose to solve the stated problem by developing a Wholesale Energy Price Estimator based on a Long Short-Term Memory Recurrent Neural Network (LSTM RNN) as implemented in the DeepAR algorithm under the SageMaker product on Amazon Web Services (AWS). DeepAR is a supervised learning algorithm for forecasting scalar time series (one-dimensional) to produce both point and probabilistic forecasts.

LSTM type of recurrent neural network are designed to overcome problems of basic RNNs, so the neuronal network can learn long-term dependencies. Specifically, it tackles vanishing and exploding gradients –the phenomenon where, when you back-propagate through time too many time steps, the gradients either vanish (go to zero) or explode (get very large) because it becomes a product of numbers all greater or all less than one.

We will use that model to forecast the Germany day-ahead energy spot price for the first week of 2019 with hourly resolution. We will use historical price data from the previous 4 years (2015-2018) to train the model and the last week of those years to validate/evaluate the model and tuning its hyperparameters.

Datasets and Inputs

We will use data provided for the Open Power System Data (https://open-power-system-data.org/). A platform that aims to enhance the efficiency and quality of centralized data provision by collecting, checking, processing, aggregating, documenting and publishing data required by most energy modelers.

The current phase of this project runs from January 2018 until December 2020 and is carried out by Neon *Neue Energieökonomik*, Technical University of Berlin, DIW Berlin and ETH Zürich.

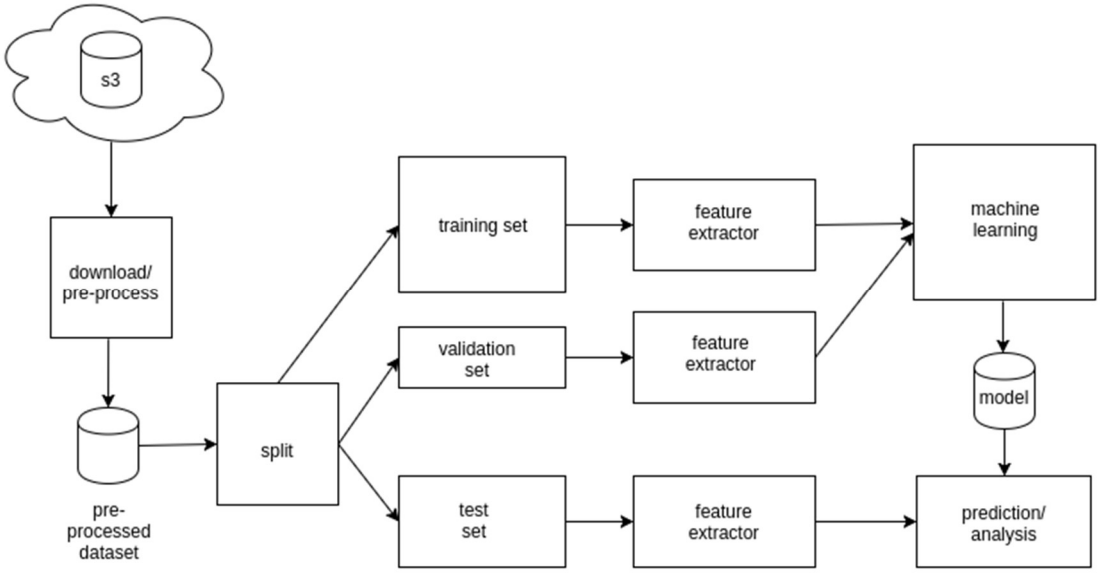
Among the different open datasets provided in the data platform we will use the package named “Time Series” (https://doi.org/10.25832/time_series/2019-06-05). This data package contains different kinds of time series data relevant for power system modelling, namely electricity consumption or load (MWh) for 37

European countries as well as wind and solar power generation (MWh), installed capacities (MW) and prices (€/MWh) for a growing subset of countries. The time series become available at different points in time depending on the sources. The data has been downloaded from the sources, resampled and merged in a large CSV file with hourly resolution and are free to download.

Process Overview

Once we obtained the public data from the Open Power System Data platform, and as the data came in 1h time resolution as we wanted we did not have to aggregate the data within a given period. We prepared training, test and validation sets. We then upload the prepared dataset to our S3 bucket. This data location is then used with SageMaker provided DeepAR algorithm, to predict future values of Germany electricity spot prices

Following diagram provides a high level overview of the project architecture:



Evaluation Metrics

If we specify optional test channel data, the DeepAR algorithm evaluates the trained model with different accuracy metrics. The algorithm calculates the root mean square error (RMSE) over the test data as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{nT}\sum_{i,t}\left(\hat{y}_{i,t} - y_{i,t}\right)^2}$$

$y_{i,t}$ is the true value of time series i at the time t. $\hat{y}i_{,t}$ is the mean prediction. The sum is over all n time series in the test set and over the last T time points for each time series, where T corresponds to the forecast horizon (prediction length).

In addition, the algorithm evaluates the accuracy of the forecast distribution using weighted quantile loss. For a quantile in the range [0, 1], the weighted quantile loss is defined as follows:

$$\text{wQuantileLoss}[\tau] = 2\frac{\sum_{i,t} Q_{i,t}^{(\tau)}}{\sum_{i,t} |y_{i,t}|}, \quad \text{with} \quad Q_{i,t}^{(\tau)} = \begin{cases} (1-\tau)|q_{i,t}^{(\tau)} - y_{i,t}| & \text{if } q_{i,t}^{(\tau)} > y_{i,t} \\ \tau|q_{i,t}^{(\tau)} - y_{i,t}| & \text{otherwise} \end{cases}$$

$q_{i,t}(\tau)$ is the τ-quantile of the distribution that the model predicts.

We use both error measures (RMSE and the average of the prescribed quantile losses) to evaluate the model so that they will be included as part of the log output. Additionally, RMSE will be used to compare our model with a benchmark.

Data exploration

The database has 393 columns and 125593 rows, covering from 2015-01-01 00:00:00 to 2019-05-01 01:00:00 with 1h sampling frequency.

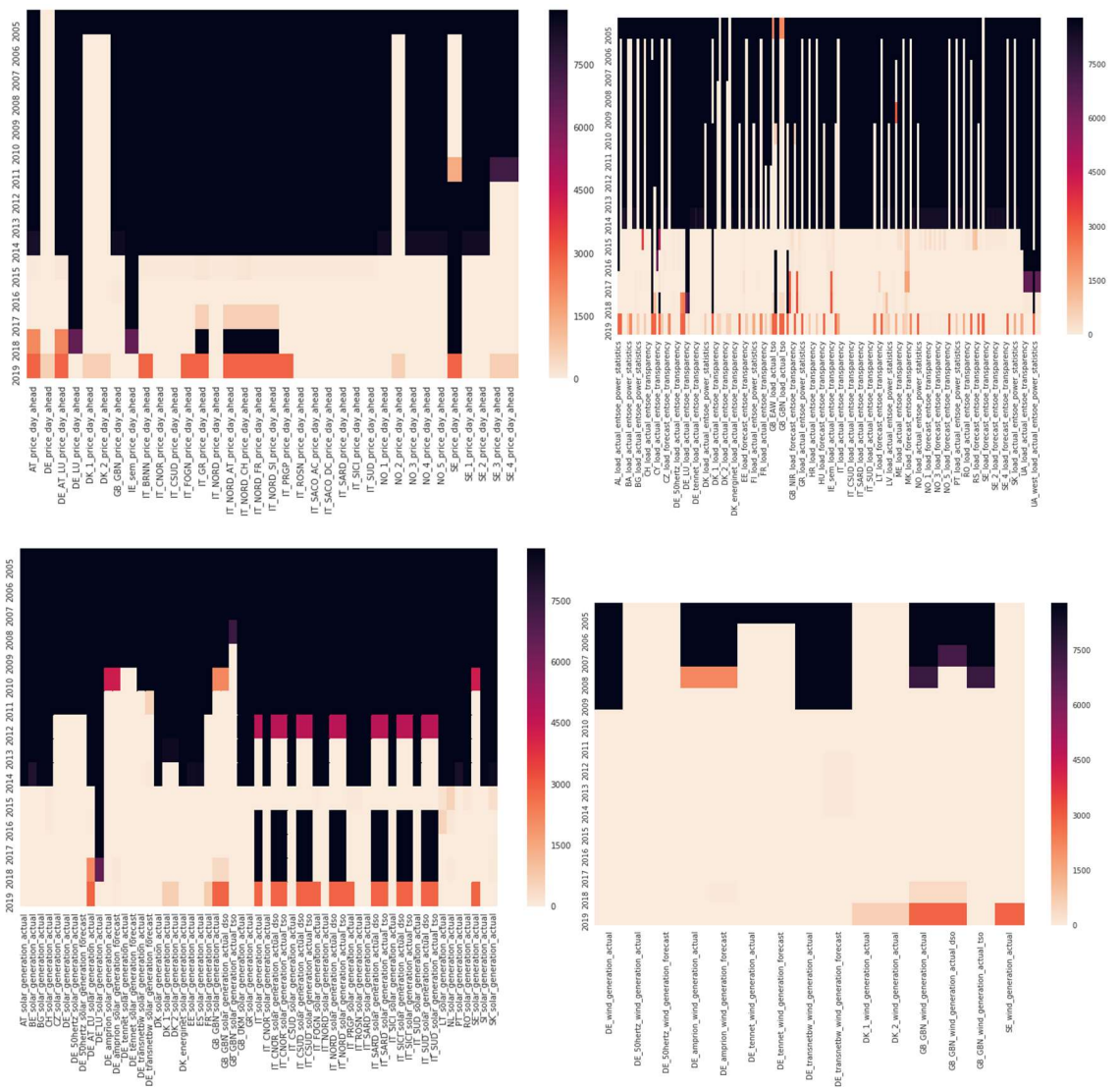
A preliminary task we had to face at the very beginning was to select which country to work on. To do so we filtered all the columns containing energy prices data, obtaining 36 columns corresponding to 8 different countries (Austria, Deutschland, Denmark, Great Britain, Ireland, Italy, Norway and Sweden) with some of them containing more than one price time series corresponding to the different electric price areas in that country[1].

Then we found out how many missing values (NaNs according to the database info) there were in those price columns and how these NaNs were distributed along the years.

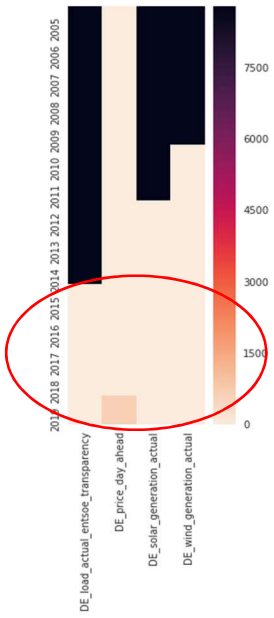
Repeating the above procedure but filtering now successively by “load”, “solar_generation” and “wind_generation” we found out how these features are populated for the different years.

The following images show how many data we have for the whole range of years in the 4 magnitudes of interest (price, load, solar generation and wind generation) meaning black color a complete lack of data and light pink full data presence for the year.

[1] An electricity price area is a zone throughout which the electricity is traded at the same spot price on a power exchange market. Electricity price areas are decided by transmission system operator and can be a whole country (like Austria, Germany, Ireland or UK), or parts of it (like Italy or Nordic countries).



In view of that we chose Germany, and the period 2015-2018, like a good option for this project:



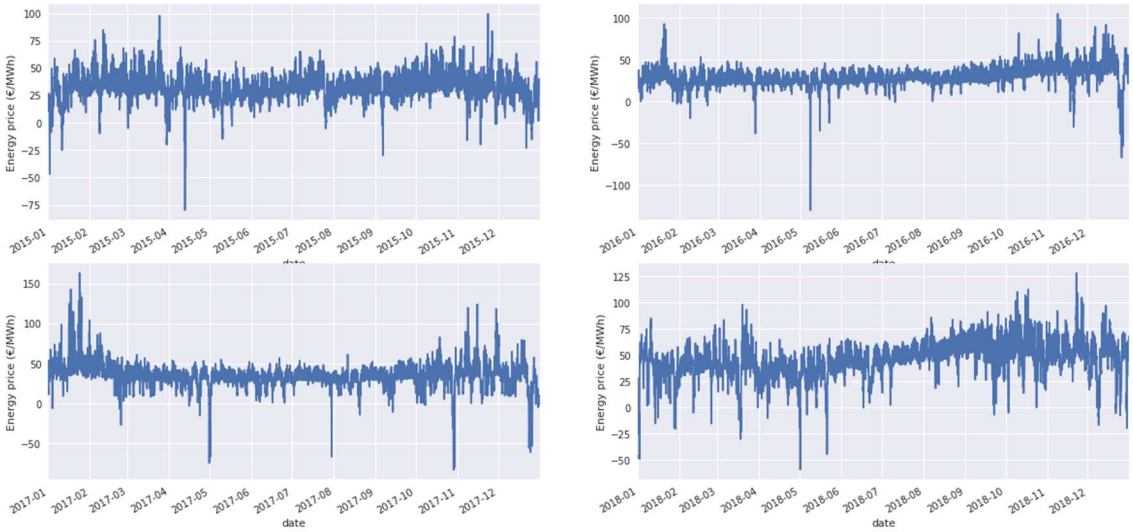
Having chosen Germany for our project, we will keep only 5 columns hereafter:

```
['cet_cest_timestamp',
 'DE_load_actual_entsoe_transparency',
 'DE_price_day_ahead',
 'DE_solar_generation_actual',
 'DE_wind_generation_actual']
```

Exploratory Visualization

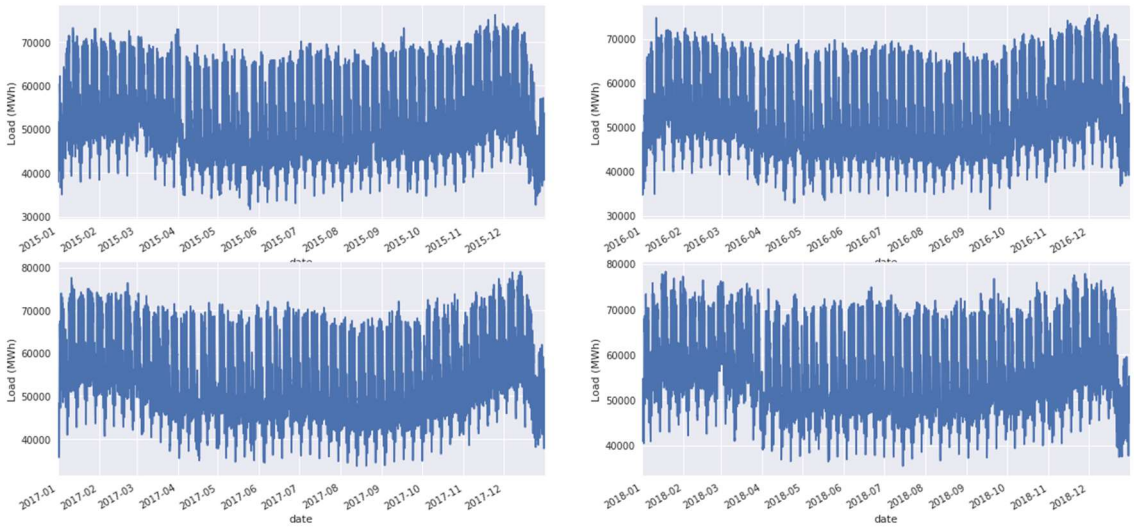
We took advantage of the function to split the whole multiyear time series in several 1-year slices to create as many time series as there are complete years.

The target time series ("DE_price_day_ahead") for the selected 4 years (2015-2018) were:



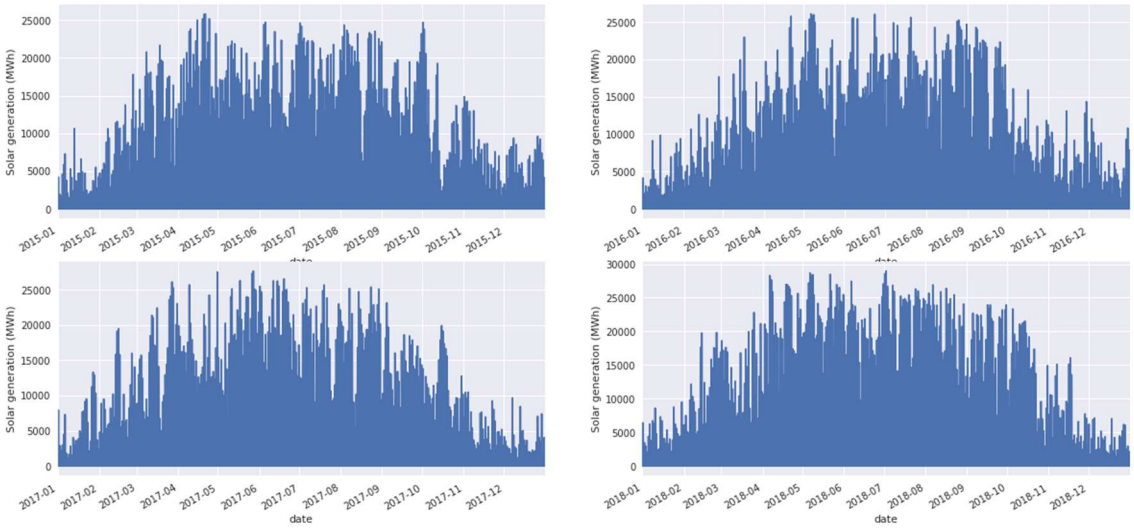
The first thing that strikes us is the strong volatility and the presence of negative spikes prices. Such negative prices are not the norm in Germany, but they are far from rare, over Christmas and New Year specially, when electricity consumption is low due to nation-wide holidays, and when high winds and non-flexible conventional electricity sources can provide for excess power production, electricity prices in Germany have frequently turned negative.

The first feature time series ("DE_load_actual_entsoe_transparency") for the selected 4 years (2015-2018) were:



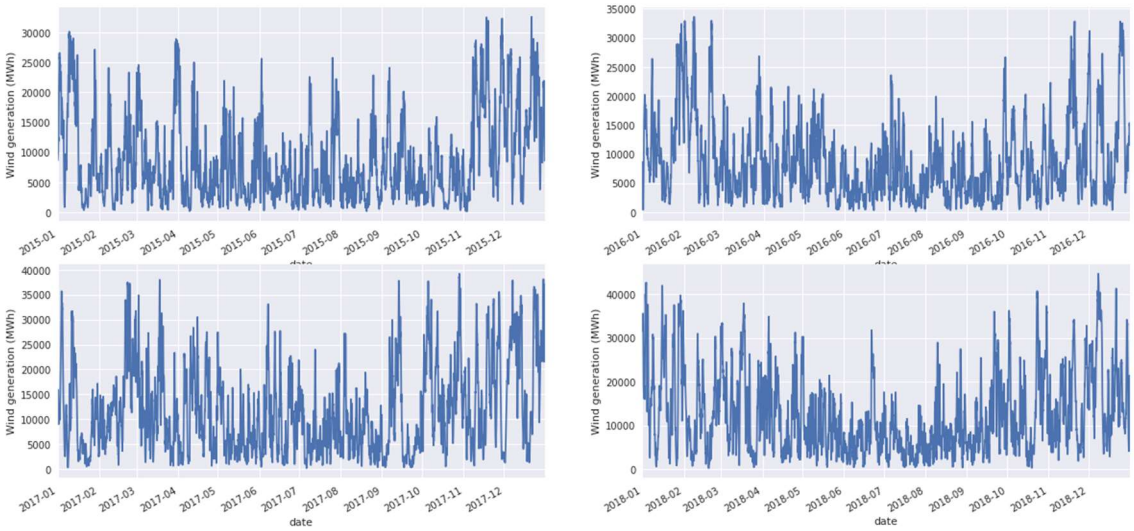
We can notice a strong daily pattern (day load higher than night) and a weaker seasonal pattern (winter load higher than summer) and some odd days specially during the last days of each year.

The second feature time series (“DE_solar_generation_actual”) for the selected 4 years (2015-2018) were:



It shows a clear daily pattern (day-night cycles) and yearly pattern (summer-winter).

The third feature time series (“DE_wind_generation_actual”) for the selected 4 years (2015-2018) were:



Unlike load or solar, the wind generation does not exhibit a clear pattern, but it can be noted higher generation in winter than in summer.

Some statistics are:

Mean	Price (€/MWh)	Load (MWh)	Solar (MWh)	Wind (MWh)
2015	31.6	54557	3984	8843
2016	29	54768	3934	8766
2017	34.2	56309	4096	11718
2018	44	58060	4707	12394

Std	Price (€/MWh)	Load (MWh)	Solar (MWh)	Wind (MWh)
2015	12.7	9983	6043	7158
2016	12.5	9788	6032	6846
2017	17.6	10263	6314	8811
2018	17.6	9894	7165	9048

Note the constant increase in generation due to the increasing wind installed capacity over the years. Solar generation is far to be a near Gaussian process so Standard deviation does not make any sense in that case.

Data Preprocessing

First preprocessing task was to split the time column “cet_cest_timestamp” into 4 different columns (Year, Month, Day and Hour) and construct with them a DateTime object (ex: 2015-01-01 00:00:00) to index the data frame with, as it is a requirement of the DeepAR algorithm to receive DateTime indexed series.

	Year	Month	Day	Hour
0	2005	01	01	00:00:00
1	2005	01	01	01:00:00
2	2005	01	01	02:00:00
3	2005	01	01	03:00:00
4	2005	01	01	04:00:00

Additionally, we had to deal with the missing values. According the DeepAR documentation the latest release of DeepAR now handles missing values in the target time series directly within the model. This makes forecasting with time series that contain missing values easier (no imputation during pre-processing is required) and more accurate (by utilizing the RNN model instead of relying on coarse-grained external imputation techniques). However, missing values are not supported in the time features, so we did have to impute them.

Then we replaced NaN with "NaN" in the target time series ("DE_price_day_ahead") so that DeepAR can handle with them and imputed with mean value in the three dynamic feature time series ("DE_load_actual_entsoe_transparency", "DE_solar_generation_actual" and "DE_wind_generation_actual").

Implementation

For machine learning tasks like classification or regression we typically create train/test data by randomly splitting examples into different sets. For forecasting it is mandatory to do this split in time rather than by individual data points.

In this project we reserved the last week of each of the time series for evaluation purposes and use only the first part as training data.

We chose 7x24 h as prediction length and the same for context length.

In order to use the train/test data for DeepAr however, you will need to do some preprocessing to have the data formatted following DeepAR Input/Output interface that according documentation should be JSON format that represents each time series as a JSON object (like dictionary). Finally, we split the formatted data into training and test channels and hosted both on our S3 bucket, to be fed into an estimator at training phase.

Following are the necessary inputs when submitting a training job:

- Uniquely identifiable job name
- SageMaker algorithm image path where the DeepAR training code is available
- URLs of the Amazon S3 bucket where you have the training and test data stored
- URL of the S3 bucket where you want to store the output of the job (upon training completion SageMaker archives the model artifacts and makes

those available as a tar-file named model.tar.gz at the specified location on S3

Next we need to set the hyperparameters for the training job. For example, frequency of the time series used, number of data points the model will look at in the past and number of predicted data points. The other hyperparameters concern the model to train (number of layers, number of nodes per layer, likelihood function) and the training options (number of epochs, batch size, learning rate...).

Once we have trained a model we use the deploy method on our trained Estimator to create a hosted model definition, configure the endpoint, and deploy the model to the endpoint - all in one step.

Refinement

In a first step we trained the model by using just the time data of previous years without using any additional feature that we can rationally expect to know forehand when we face this problem. Such features (covariates) could be load and renewables production as both of them are usually forecasted and published by TSOs (Transmission System Operators) which are in charge of take off the energy, bring it to market and balance the in-feed.

So, in a second step we will improve the model by taking into account the load, and renewable energy delivered (both Wind and Solar) as additional time series features.

One of the top characteristic for DeepAR is the ability to supply custom time-varying features to the model. When forecasting a time series, users often have other, explanatory time series that can provide useful information about the target time series and help to explain is variance.

(https://aws.amazon.com/es/blogs/machine-learning/amazon-sagemaker-deepar-now-supports-missing-values-categorical-and-time-series-features-and-generalized-frequencies/)

Using this time-varying information can ultimately improve accuracy by “explaining” effects that otherwise would have been attributed to noise.

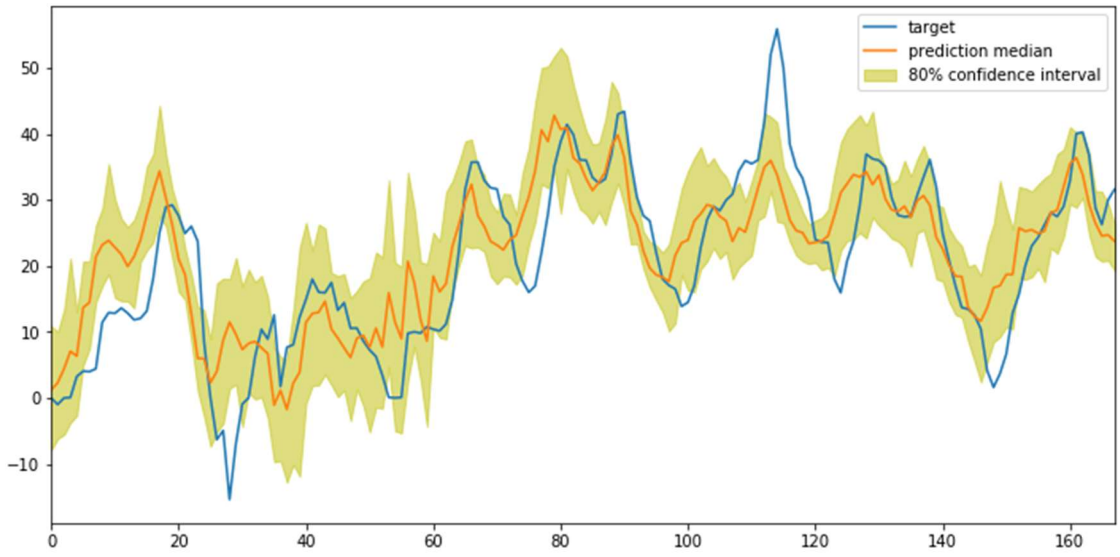
Benchmark Model

Additionally, we developed as well a simple statistical model to compare our model with. This simple model was used as a threshold to determine if our machine learning model has succeeded or not. This model assigned to each day-time of the forecast period (first week of 2019) a predicted value calculated as the mean for the 4 previous years (2015-2018) of the X period moving average around the corresponding day-time for each year. We assigned different values to X from 1 to 10 in order to use as benchmark the best possible simple model.

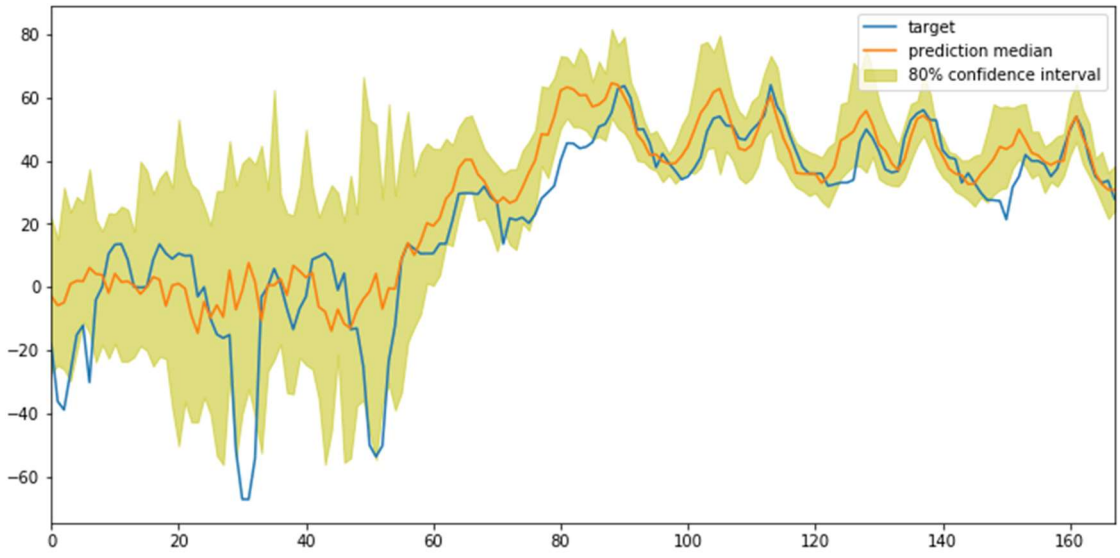
Model Evaluation and Validation

We evaluated the model on a test set of data. In the chosen dataset we can find the real energy price for the test period of each year to be used as benchmark so that we could evaluate the goodness-of-fit of the model to the real data.

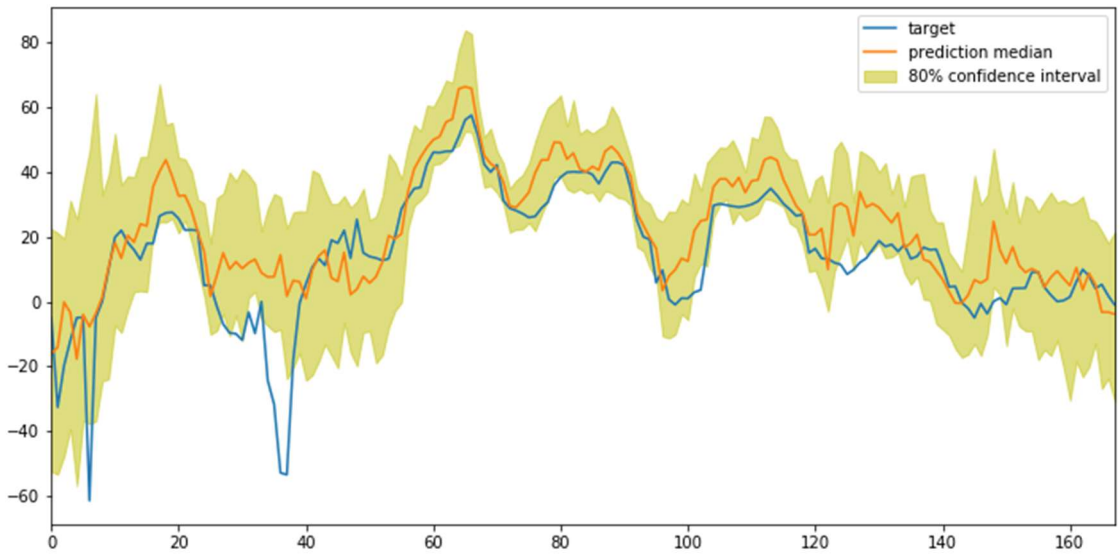
Test results for the last week of 2015:



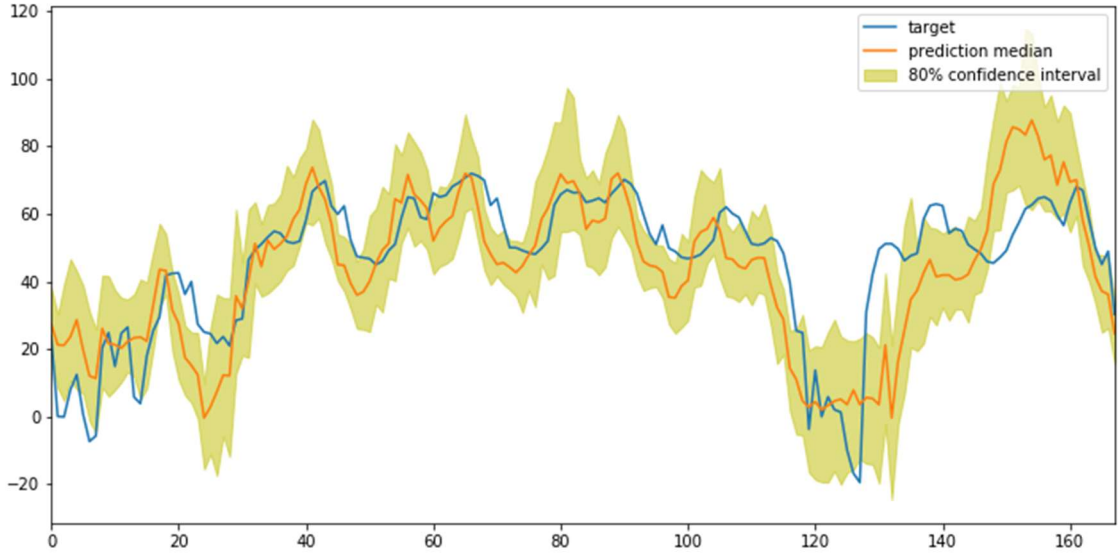
Test results for the last week of 2016:



Test results for the last week of 2017:



Test results for the last week of 2018:

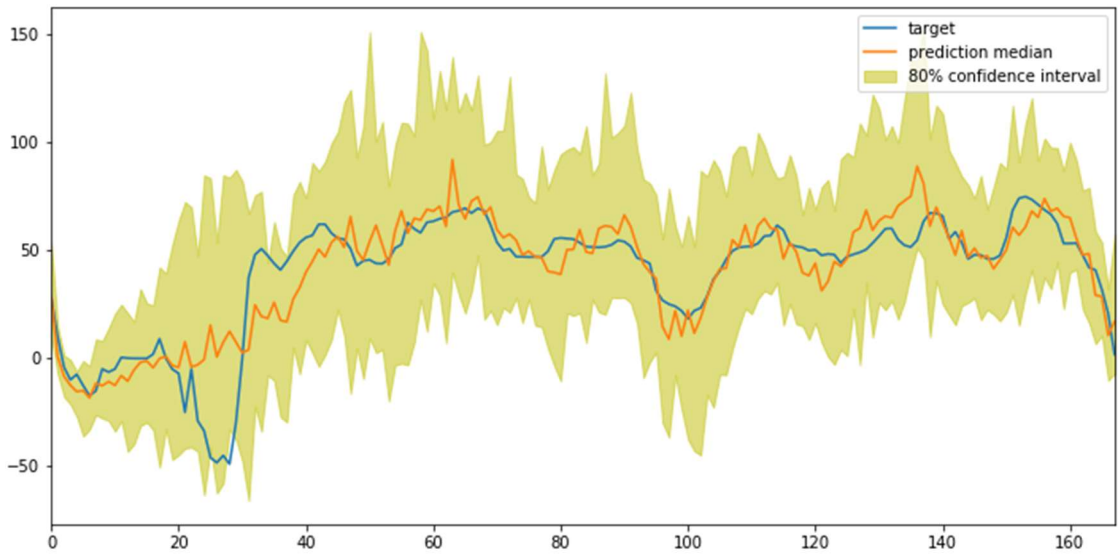


The algorithm calculated the root mean square error (RMSE) and the average of the weighted quantile losses (wQuantileLoss) over the entire set of test, given:

$RMSE_{test}$ = 13.18 €/MWh

$wQuantileLoss_{test}$ = 0.25

We used as well our already trained model to forecast the first week of 2019:



Getting an error $RMSE_{2019}$ = 14.44 €/MWh. Slightly higher than the test, but the first week in Germany is challenging as January, 1 is day off and this fact seems to be not properly captured by the model.

Justification

In this project we developed 3 different forecasting tools, all based on DeepAR:

- Model_1: Was trained by using just the time data of previous years without using any additional feature

- Model_2: Was trained taking into account the load, and renewable energy delivered (both wind and solar) as additional time series features, and using the same hyperparameters used on the Model_1 in order to compare both and quantifying the improving

- Model_3: Is the same as Model_2 but optimizing the hyperparameters to get a more powerful estimator

- Model_benchmark: Was developed as a moving average estimator in order to serve as a threshold to determine if our machine learning models have succeeded

Using RMSE as metric, we computed the error when forecasting the first week of 2019, getting the following results:

	RMSE
Model_1	29.88
Model_2	17.84
Model_3	14.44
Model_benchmark	19.43

Model_1 failed to beat the benchmark as the volatility of the energy prices is really important, but when introducing additional time features the model clearly improves the benchmark even without a fine tuning of its hyperparameters.

By adjusting the hyperparameters a sound improvement can be done, but the model still fails in capturing the strong negative spikes. Strong decline in power prices is especially noticeable when power demand is low. This indicated amongst other things that a high share of inflexible must-run capacities (e.g. power plants with balancing power obligations or combined-heat-and-power plants running heat- and not power-driven) is not able to change its own power generation at times of high renewable energy feed-in into the German grid. Those must-run power plants rather incur negative power prices than reducing the own power generation partially or completely. So there are some external factors that are not taken into account into the model and make hard to accurately predict energy prices in events of low demand and high renewables production, but the model is quite good for the rest of the hours.