Energy Data Science - Report 2

Kacper Aleksander 244693EV

Tallinn University of Technology
16th of December, 2024

Contents

1 Introduction

I worked on time-series data that revolved around electricity prices and demand. The purpose of this task was to analyze those data, create forecasting models for the target variable `demand`, and compare their precision.

Predicting energy demand is important in the energy sector, as it can help plan ahead and match energy production more closely to the demand, so that the grid frequency is less variable - and less variability is preferred.

2 Analysis

2.1 Reading and understanding the data

The data consisted of the following features:

- `time` - timestamp
- `temp` - air temperature [in °C]
- `dwpt` - dew point [in °C]
- `rhum` - relative humidity [in %]
- `snow` - snow depth [in mm]
- `wdir` - wind direction [in degrees]
- `wspd` - average wind speed
- `wpgt` - peak wind gust [in km/h]
- `pres` - sea-level air pressure [in hPa]
- `price` - electricity price in Estonia on that hour [in EUR/kWh]
- `demand` - electricity demand [in kWh]

The data spanned from 01.09.2021 to 24.08.2022 and was divided into train and test sets. There were missing values in features `snow` and `demand`.

2.2 Cleaning the data

I noticed that missing values in `snow` corresponded to the lack of snow, so I filled all the missing values (`NaN`) of `snow` with zeros.

To fill a small amount of missing `demand` data, that I assumed was missing completely at random (MCAR), I used linear interpolation based on time.

2.3 Exploring the data

While exploring the data, I focused on two main plots - the pairplot (Figure 1), which is a scatterplot between all of the features, and on the correlation matrix (Figure 2) - which reflects how strongly the variables correlate with each other.

The pairplot served as a basis for looking for interesting patterns. There was a strong linear correlation between the `temp` and `dwpt` features, which later

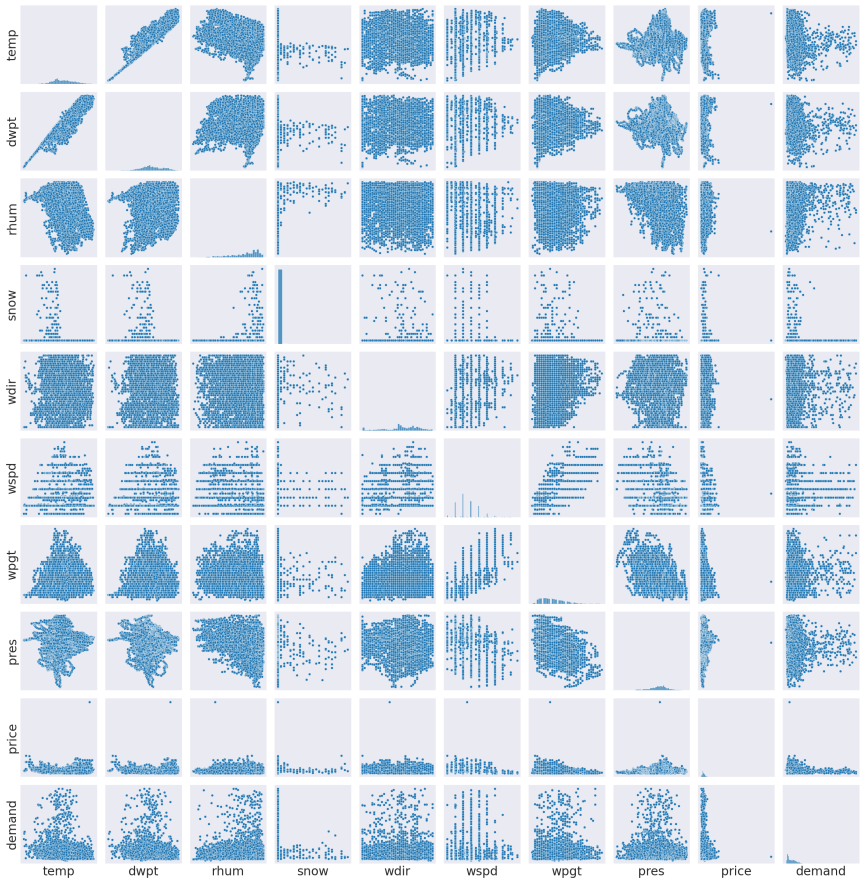

Figure 1: Pairplot (multiple scatterplots)

became a basis for removing the latter. I focused my attention mostly on what features correlate with `demand`, which I was to predict the values of.

The correlation matrix showed strong correlations between some of the features, which was valuable information.

2.4 Analyzing distributions of varirables

To analyze the distribution of each variable, I used histograms (Figure 3). Many of those distributions had irregular shapes, far away from the normal distribution. The variables also had different scales. Before forecasting, it is important to make variables more regularly distributed and bring them to the same scale, to ensure that no feature is over- or underrepresented.

2.5 Transformations on variables

I transformed `price` with a natural logarithm. Afterwards, I standardized all of the features. Distributions after these transformations can be seen in Figure

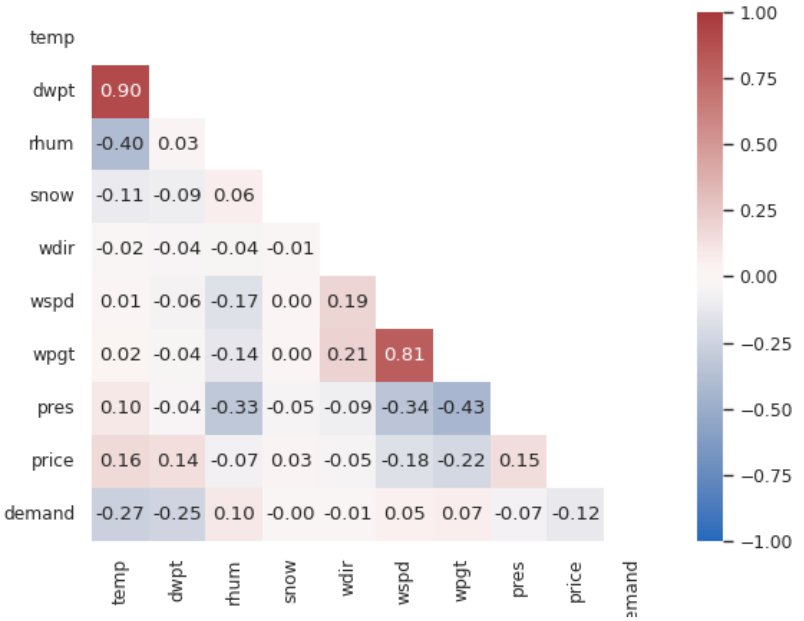

Figure 2: Correlation matrix

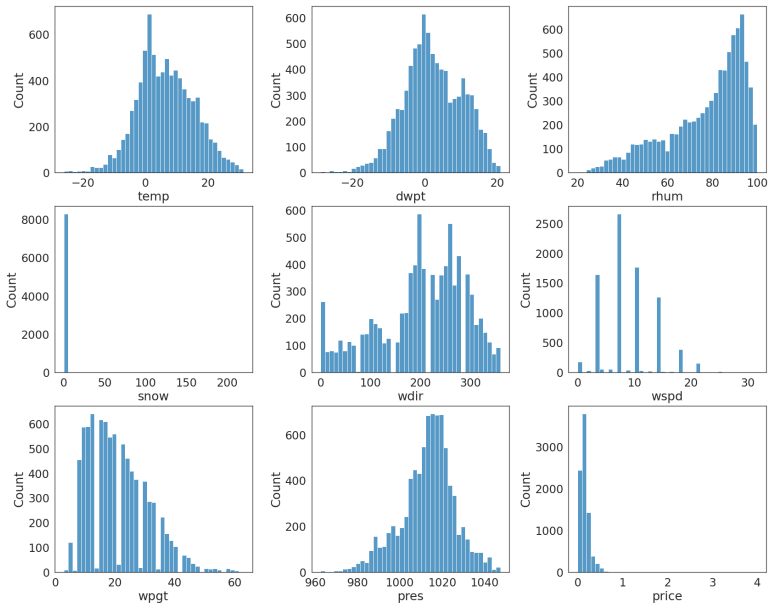

Figure 3: Distributions of features (histograms)

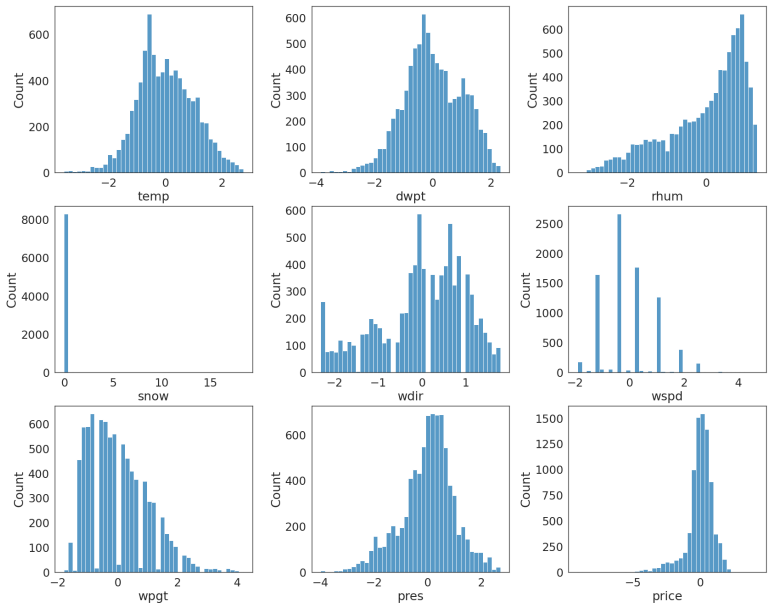

Figure 4: Distributions of features (histograms) - after transformations

4.

2.6 Designing new features

I decided to design five new time-related features and examine if any of them would be useful. These included:

- `hour` - hour of the day in UTC
- `dayofweek` - 1 being Monday, 7 being Sunday
- `is_weekend` - 1 if it is Saturday or Sunday, 0 if not
- `lag_24` - the value of demand 24 hours ago
- `lag_168` - the value of demand 168 hours ago (a week ago)

To check their usefulness, I plotted a correlation matrix (Figure 5) once again.

2.7 Ranking features by relevance

Based on the new correlation matrix, I decided to pick 3 most important features:

1. `temp` - highly correlated with demand

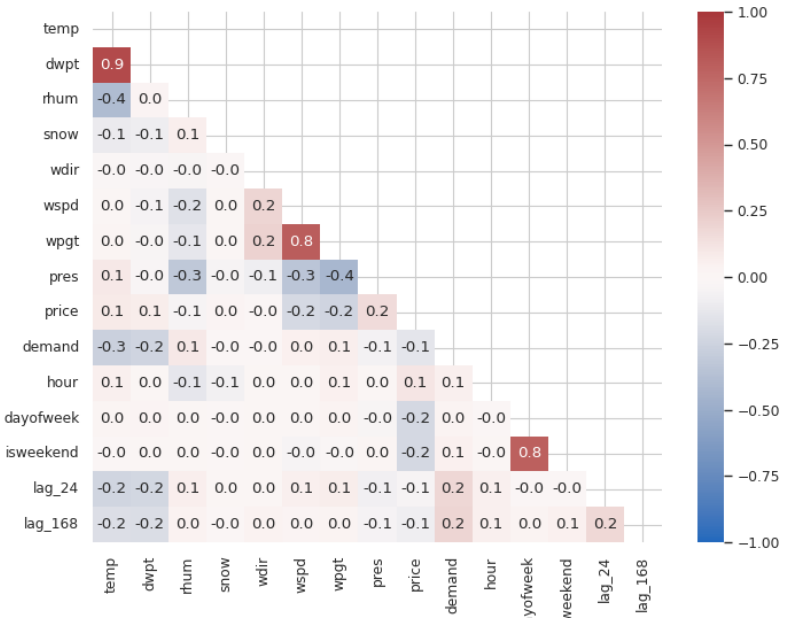

Figure 5: Correlation matrix with new features

2. `price` - reasonably correlated with demand, low correlations with other features, and by general knowledge, price can have an impact on demand

3. `pres` - reasonably correlated with demand

I didn't choose `dwpt`, even though it was strongly correlated with `demand`, because it was also strongly correlated with `temp`, which I had already included.

3 Modeling

3.1 Seasonality and stationarity analysis

Before I started creating ARMA-family models, I checked whether the data I have was stationary, and whether it had seasonality. I did the latter it by plotting `demand` against hours (Figure 6) and days of week (Figure 7). The Figure 6 showed clearly, that there was daily seasonality. The weekly seasonality was not as striking, so I decided to ignore this aspect.

I performed the Augmented Dickey Fuller test to assess stationarity, and based on the results, I assumed the data were stationary.

The last step of seasonality analysis was to plot ACF and PACF graphs (Figure 8). It confirmed the initial seasonality assumption and showed that values that have one hour difference are highly correlated, which was valuable information for forecasting.

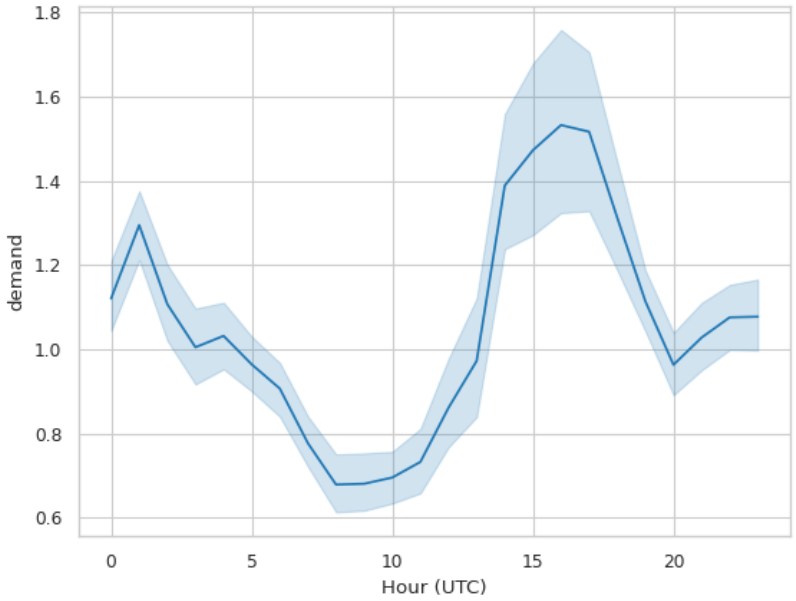

Figure 6: Lineplot - demand against hours

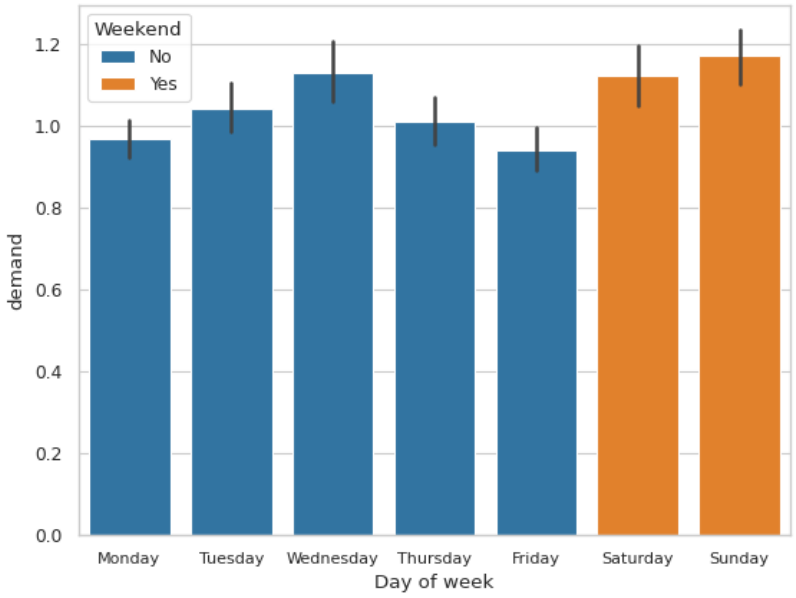

Figure 7: Barplot - demand against days of week

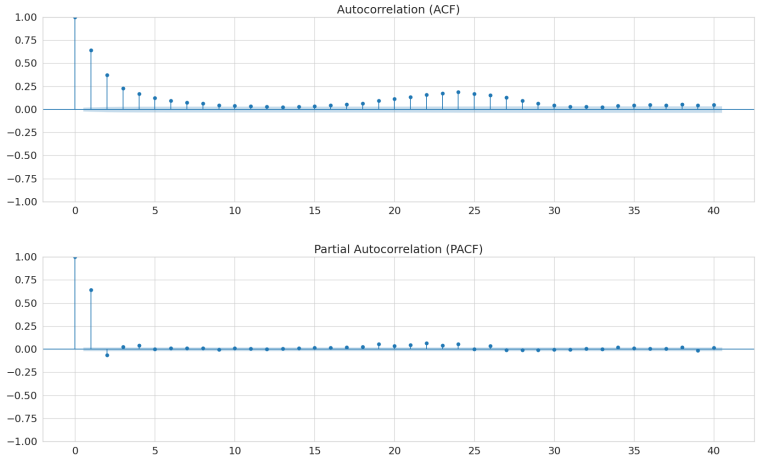

Figure 8: ACF and PACF graphs

3.2 Model description

I created three models to predict `demand` values:

1. SARIMA model - an ARMA-family model, taking seasonality into account
2. SARIMAX model - similar to SARIMA, but with exogenous variables: `temp`, `lag_168` and `price`
3. Naive 7 day lag model - predicted value to be the same as the value 7 days ago

Coefficients for the first two models were fitted automatically using the Python package `pmdarima`.

4 Forecasting

4.1 Forecasting information

For the training data, I used the provided training set, but only the last two weeks (336 records) to decrease model training time. As test data, I used the provided testing set, which consisted of one week (168 records).

For the performance metric, I used mean absolute error (MAE).

I used a rolling approach for forecasting - I forecasted one day at a time, and then updated the model with real (test) data.

Below I present the results of my work.

4.2 Results

Mean absolute error scores:

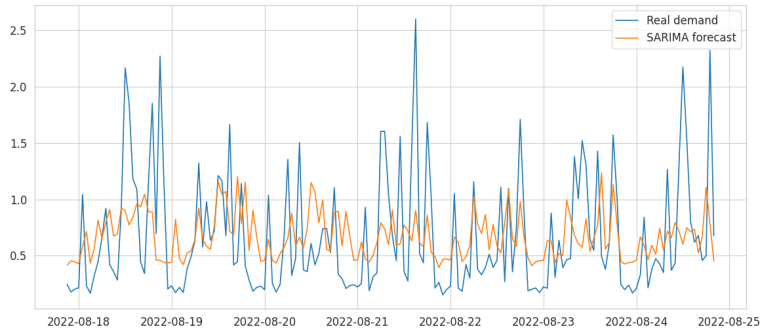

Figure 9: SARIMA forecast against real data

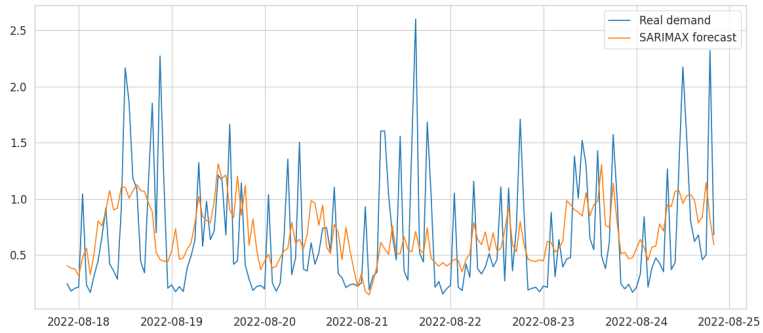

Figure 10: SARIMAX forecast against real data

- SARIMA - 0.38051
- SARIMAX - 0.36929
- Naive 7 day lag - 0.37416

4.3 Comment on results

All three models achieved similar scores. When looking at the figures, it seems like ARMA-family models (Figure 9 and 10) are more averaged-out - they have quite low highs and high lows. On the other hand, the naive 7 day lag forecast (Figure 11) looks more realistic - as this is exactly the data from 7 days ago.

The SARIMAX model - the most complex one - is the winner, but only by a small margin. The second best model, the naive 7 day lag, performed only a little worse, but its advantage is its simplicity, and the execution time is minimal. The SARIMA model performed the worst out of the three - it was expected that it would perform worse than SARIMAX, which had exogenous variables as an addition.

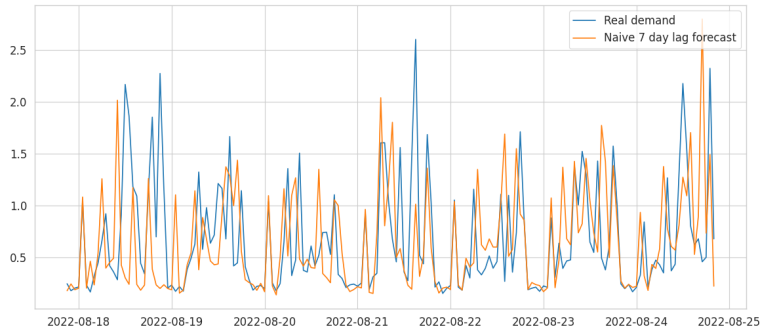

Figure 11: Naive 7 day lag forecast against real data

The forecast might have yielded better results if I tried other forecasting methods, such as random forest or neural networks.