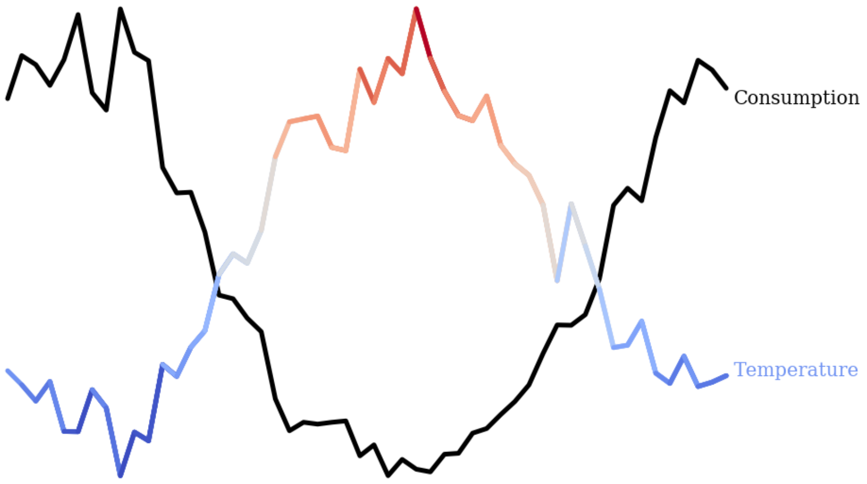


Semester Project,
Business Intelligence and Analyzing Big Data

The intermingling of European oil prices, elspot prices, local weather, and electrical energy consumption in Agder, Norway

Arild Madshaven*

Faculty of Business and Economics
University of Lausanne UNIL
Lausanne, Switzerland

June 12, 2020

*arild.madshaven@epfl.ch

Abstract

The purpose of this project was to determine if any relationships could be pinpointed between oil prices, el prices, and power consumption among customers in the Agder county of Norway. Data was fetched from various sourced and gathered in a data warehouse structure. Further on, the data was cleansed and analyzed through standard descriptive statistics, and two statistical algorithms: the Granger causality test and the apriori algorithm. Key findings indicated that individual differences could be observed on a customer basis. These findings could prove valuable for companies accountable for power transmittance, and the knowledge of whether or not customers pay attention to prices could possibly benefit companies involved in power sales and trading. Some customers were identified as fixed on prices, whereas more customers were fixed on weather dynamics. Relationships were identified between oil and el prices, and weather was identified to highly influence el prices.

Contents

0 Introduction

Humans fixed on targets are notoriously prone to tunnel vision. The project topic was initially loosely defined to maximize the odds of unexpected results, if arising, to catch the author's attention. However, if one were to define a phrase describing the underlying question at project initiation, it would be akin to *"Do Norwegians regulate their electrical energy consumption after shifting elspot prices?"*.

Times are a-changing, and novel smartphone apps like Tibber [13] allow its users to smart-charge their electric cars during hourly windows of lower prices. Growing up in Norway, the author did not traditionally sense a fixation on elspot prices among countrymen, but the introduction of smart tools do undeniably ease the work of monitoring the sudden shifts and windows of opportunities in the market and could thus produce *smart customers*. Identifying atypical customers could be of paramount importance to companies concerned with electrical energy distribution and grid maintenance. For instance, customers whose consumption time series are typically characterized by infrequent sudden peaks followed by flat zeros, pose a disproportionate threat to the stability of the grids, and could be bargained with to free capacity and thus avoid the costly process of upgrading the electrical distribution network.

Nevertheless, the purpose of this semester project was primarily to gain familiarity with big data acquisition, wrangling, and mining, through the use of SQL and key statistical methods. In some sense, achieving valuable results was considered secondary to picking up valuable knowledge along the way.

0.1 Key algorithms

Two key algorithms were explored to mine for data patterns. The Granger causality test [4] and the apriori algorithm for frequent itemset mining [2]. The former defines *Granger causality* as *time series B's ability to forecast time series A*, in which case B *Granger causes* A. This approach pointed towards a causal relationship between the temperature and some customers' consumption, and for some customers the elspot prices reportedly Granger caused their consumption. Undoubtedly, temperature Granger caused elspot prices, and to the author's surprise, the method also claimed elspot prices Granger caused oil prices.

The apriori algorithm on the other hand looks for frequent itemsets for which rules on the form *if A then probably also B* can be derived. Unfortunately, this method relies heavily on the quality of preprocessing (i.e. bagging) when dealing with numerical features. The algorithm revealed that prices are typically high when temperatures are low, and some typical price responses to shifting temperatures. It also suggested that oil and el prices are linked, or, at least, influenced by common hidden actors.

1 ETL Process

The project workflow, especially the process of **E**xtracting, **T**ransforming and **L**oading data is briefly illustrated in fig. 1. The project in its entirety was carried out using Jupyter notebooks [9] with Python 3 [10] kernel and MySQL's Connector/Python [6] to communicate with MySQL servers from a Python environment. Following subsections describe the different operations in closer detail.

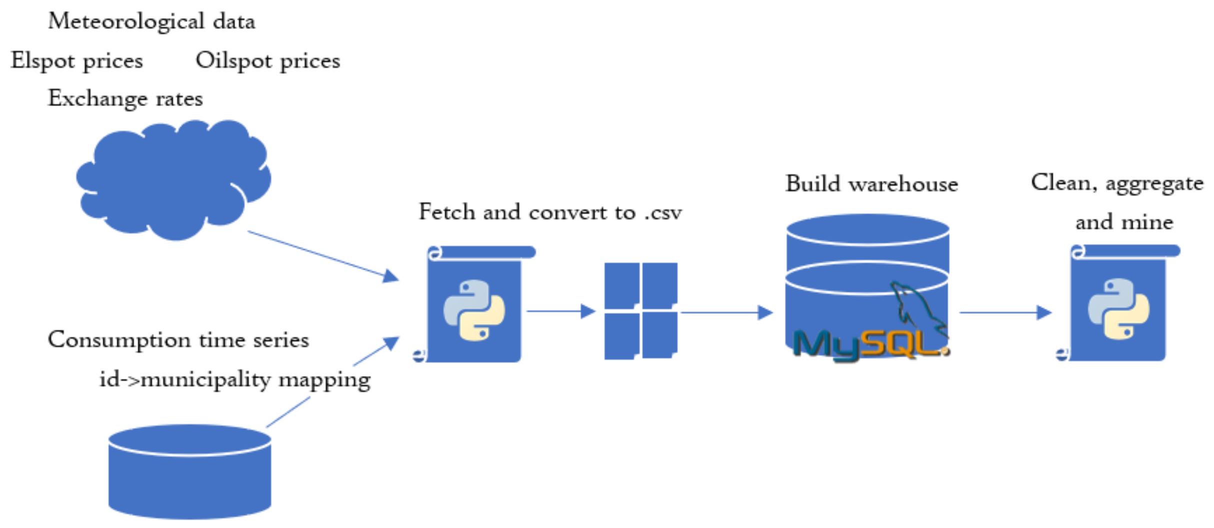


Figure 1: Project ETL process overview.

1.1 Raw sources

Six sources of data were used as foundation for this project. 20 consumption time series labelled with id related to name of municipality of origin, donated by Agder Energi Nett [1], the company responsible for transmitting energy within Agder county in Norway. Time series of elspot prices downloaded from Nordpool group [7] who run the leading power market in Europe. Brent oilspot prices from the U.S. Energy Information Administration [14]. USD/NOK conversion rates from the DNB group [3] to make use of oilspot prices (USD) as compared to elspot prices (NOK). Historical temperature data from the Norwegian Meteorological Institute fetched through its Frost API [5]. Lastly, information linking municipality names to their respective weather stations, fetched from the Norwegian Climate Service Center [8].

1.1.1 Consumption time series

The consumption time series were 20 **.excel** files named **Belastning (0)**, **Belastning (1)**, ..., **Belastning (19)** where the suffix revealed its location of origin (see table 1), and its content as displayed in table 2. *Belastning* directly translates to *load*. These files were slightly preprocessed through a few lines of Python code: The timestamp was inconsistent across tables (day first or month first) and the suffix was fetched from the file name and added as a table column. The tables were then exported to **.csv** files to ease the upcoming SQL work.

Table 1: Table linking location id from the consumption time series suffixes to their respective municipality of origin.

location_id	municipality
0	Grimstad
1	Arendal
2	Birkenes
3	Bygland
4	Bykle
...	...

Table 2: Raw consumption load sample time series. Reports are hourly, stretching at most from January 2018 to February 2020. *Tid* translates to *time*, *Tilsynelatende energi* to *apparent energy*, and *Merkeytelse* indicates the load capacity of the transformer.

Tid (CET)	Tilsynelatende energi (kVAh)	Merkeytelse (kVAh)	Merkeytelse + 20% (kVAh)
01-01-18 00:00	27	100	120
01-01-18 01:00	27.5	100	120
01-01-18 02:00	27	100	120
01-01-18 03:00	23	100	120
01-01-18 04:00	23	100	120
...	...	...	...

1.1.2 Elspot prices

The elspot prices tables were downloaded as three **.csv** files where each contained hourly reports through a calendar year. The files were subject to minor timestamp cleansing through a few lines of Python code; the first two columns of table 3 were merged into a single timestamp column.

Table 3: Raw elspot sample time series. Prices in NOK/MVAh.

Date	Hours	FI	DK1	DK2	Kr.sand	...
01-01-18	00 - 01	258,01	213,62	258,01	258,01	...
01-01-18	01 - 02	258,99	258,99	258,99	258,99	...
01-01-18	02 - 03	255,75	255,75	255,75	255,75	...
01-01-18	03 - 04	242,04	242,04	242,04	242,04	...
01-01-18	04 - 05	242,43	242,43	242,43	242,43	...
...	...	...	...	...	...	...

1.1.3 Brent oilspot prices

An open source platform providing hourly historical crude oil prices was not identified during this project. Instead, daily averages ranging from 1987 to 2020 were fetched as a single **.csv** file as displayed in table table 4. One minor step of preprocessing was done to remove the first lines of the file, containing data descriptions.

Table 4: Crude oil prices sample time series. Notice that no prices are updated during weekends.

Day	Dollars per barrel
04/28/2020	15.60
04/27/2020	15.17
04/24/2020	15.87
04/23/2020	15.06
04/22/2020	13.77
...	...

1.1.4 Exchange rates

This particular data was fetched from the DNB group [3] as three **.csv** files, one for each year from 2018 to 2020. A table view is presented in table 5.

Table 5: Exchange rates showing the cost in NOK for various currencies, fetched from the DNB group [3]. Notice that some prices indicate the cost of one, whilst others (such as SEK and DKK) indicate the cost of one hundred.

Dato	USD	EUR	SEK	DKK	...
31.12.2018	8.6911	9.9448	97.11	133.19	
28.12.2018	8.7592	10.0359	97.5	134.4	
27.12.2018	8.7641	9.974	96.52	133.63	
26.12.2018	8.6506	9.9209	96.66	132.86	
25.12.2018	8.7033	9.9326	95.89	133.06	
...	...	...	...	...	

1.1.5 Weather data

The Norwegian Meteorological Institute provides free access to historical weather data through the Frost API [5]. A user was created to obtain an access token. Further on the data collection was quite straight forward in Python:

```
endpoint = 'https://frost.met.no/observations/v0.jsonld'
parameters = {
    'sources': '<WEATHER_STATION_ID>',
    'elements': 'max(air_temperature PT1H)',
    'referencetime': '2018-01-01/2020-03-01',
}
r = requests.get(endpoint, parameters, auth=(client_id,''))
js = r.json()
```

The resulting **.json** files were further converted to **.csv** to ease the upcoming SQL work. Reading the **.json** data directly from MySQL was attempted but proved much less efficient. One file was outputted per weather station, as displayed in table 6

A wide selection of weather data was accessible for *some* weather stations, such as cloud cover, downpour, and humidity. However, for many a station only a few selected measures were offered. A compromise, for this assessment at least, was to only extract the *maximum temperature per hour*, a measure that was reported by most stations. For those locations where the nearest station could not provide this measure, the second nearest station was used instead. As consumption data was provided with *municipality name* resolution (due to anonymity) the *nearest* station could not be unambiguously decided anyway. This measure offered the additional advantage to implicitly aggregate the reports from certain stations operating on various time resolutions *finer* than hourly, i.e. several measures per hour.

Weather stations

To facilitate a successful combination of weather data and consumptions, a table linking weather station id to city name had to be established. This resource was not found except for in an interactive

Table 6: Weather data sample table as fetched through the Frost API.

sourceId	referenceTime	elementId	value	unit	timeOffset
SN36200:0	2018-01-01 00:00:00+00:00	air_temperature	6.5	degC	PT0H
SN36200:0	2018-01-01 01:00:00+00:00	air_temperature	6.4	degC	PT0H
SN36200:0	2018-01-01 02:00:00+00:00	air_temperature	6.3	degC	PT0H
SN36200:0	2018-01-01 03:00:00+00:00	air_temperature	6.3	degC	PT0H
SN36200:0	2018-01-01 04:00:00+00:00	air_temperature	6.3	degC	PT0H
...	...	...	...	...	...

map provided by the Norwegian Climate Service Center [8]. Table table 7 was manually created by scrolling the map and searching for city names.

Table 7: Table linking municipality name to its nearest weather station.

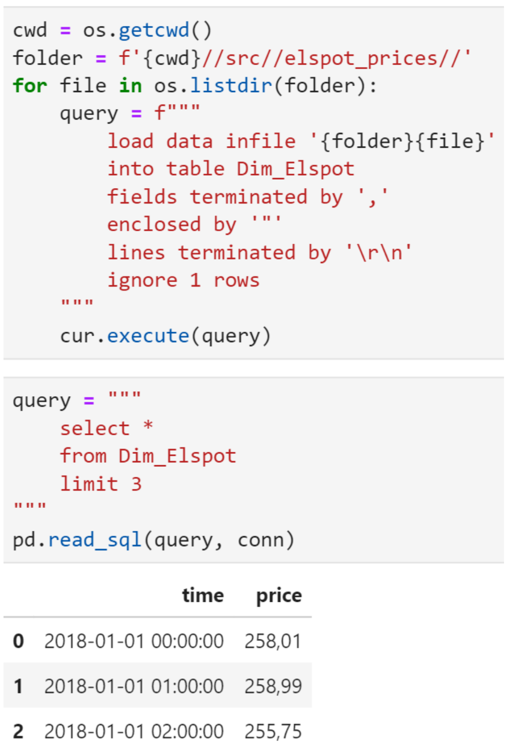
municipality	nearest_station
Arendal	SN36200
Birkenes	SN39040
Bygland	SN39750
Bykle	SN40880
Evje	SN39750
...	...

1.2 Building the Warehouse

The work of building the SQL warehouse was carried out by using MySQL servers and its Python Connector:

```
import mysql.connector
conn = mysql.connector.connect(user=MYSQL_USER_NAME, password=PASSWORD)
cur = conn.cursor()
```

After establishing a connection to the local MySQL database, queries could easily be passed and executed from the Jupyter environment. Simple features (such as loops) from the Python environment were exploited to efficiently run multiple queries (see fig. 2).



```python
cwd = os.getcwd()
folder = f'{cwd}//src//elspot_prices//'
for file in os.listdir(folder):
    query = f"""
        load data infile '{folder}{file}'
        into table Dim_Elspot
        fields terminated by ','
        enclosed by '"'
        lines terminated by '\r\n'
        ignore 1 rows
    """
    cur.execute(query)
```

```python
query = """
    select *
    from Dim_Elspot
    limit 3
"""
pd.read_sql(query, conn)
```

```
                  time   price
0  2018-01-01 00:00:00  258,01
1  2018-01-01 01:00:00  258,99
2  2018-01-01 02:00:00  255,75
```

Figure 2: Example query execution in Jupyter using MySQL's Python Connector.

All fetched .csv files were loaded into tables, and the warehouse structure was chosen in accordance with the idea of a star schema [11] (see fig. 3). In this manner, our main data of interest, that is - the consumption, is easily accessible in the Fact table, whilst aspects of the data, such as weather, municipality, and transformer capacity, are within an arms reach in dimension tables.

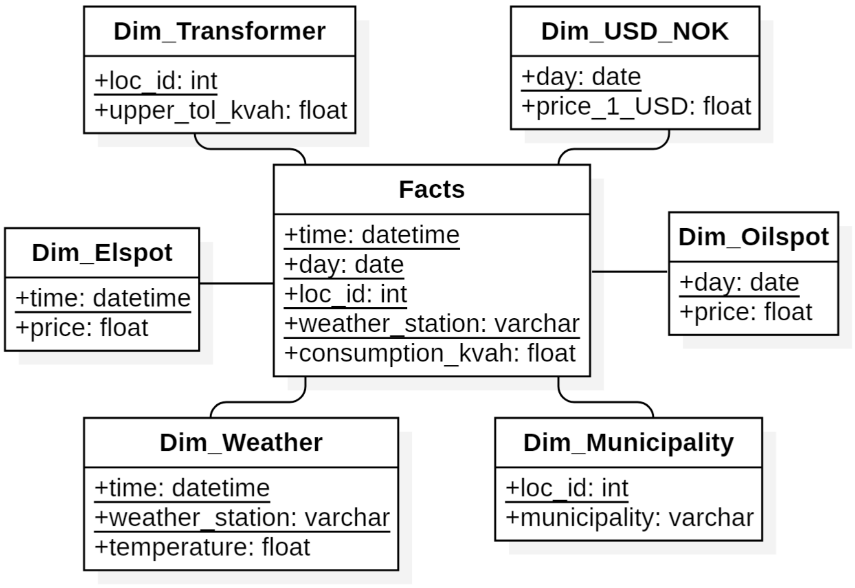


Figure 3: The warehouse star schema. Illustration built with StarUML [12].

Selected tables relevant for further cleansing and processing in Python were outputted as **.csv** files. Specifically, Dim_Oilspot, Dim_USD_NOK, Dim_Elspot, Dim_Weather, and Facts. This step is not necessary per se, but it was considered convenient during prototyping in order to ensure the warehouse could not be accidentally altered after construction.

2 Data Cleansing

The different data sources required various treatments for their missing or flawed instances. Maximizing data integrity is instrumental to data science and achieving intelligible results, and missing time series values may render large chunks of data useless. A brief description of the techniques utilized to filter and repair broken data is therefore offered in this section, preceding further analyses.

2.1 Consumption time series

The twenty consumption time series contained hourly values spanning over two years. Discrepancies on such datasets are expected, and some were identified.

It was in this study considered appropriate to **disregard** zero-measurements and measurements outside the range of four standard deviations from the mean. This assumption may deserve some criticism since zero-measurements could indicate a broken transformer rather than a broken sensor. However, it was outside the scope of this project to analyze extreme outliers related to over-consumption and transformer life expectancy.

The resulting time series were right joined with a complete hourly date range spanning from its first to last timestamp entry, most containing somewhere between 50 and 150 missing hourly values.

The industry standard for handling missing consumption data considers last week's data at the same day and hour. However, it was discovered that typically 80-90% of missing values occurred either alone or in groups of two. A comparison of last week-interpolation and linear interpolation over a span of four months can be observed in fig. 4. By naïvely using last week's value one might produce odd series when there are strong ongoing trends or atypical days involved. On the flip side - where holes were small in length, a linear interpolation typically yielded a much more sensible series. These observations led to the conclusion that holes of length one or two were linearly interpolated, and larger holes were filled with either last week or next week's data. This method served to fill all existing holes in the consumption time series.

2.2 Elspot prices

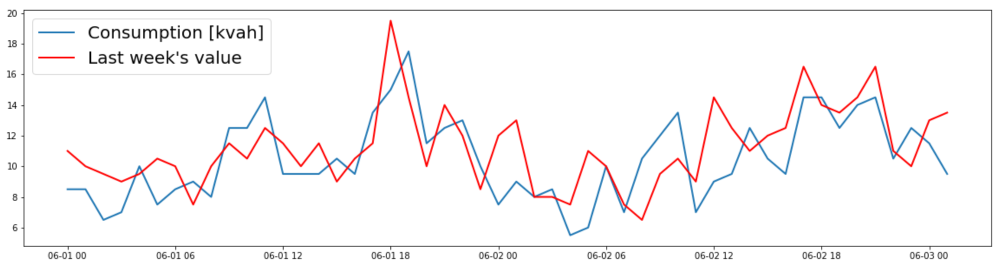
The elspot dataset was remarkably consistent. Three rows, that is, three hours over a span ranging from 2018 to 2020, were missing. These were linearly interpolated. Faulty data was not detected elsewhere in this particular data.

2.3 Brent oilspot prices

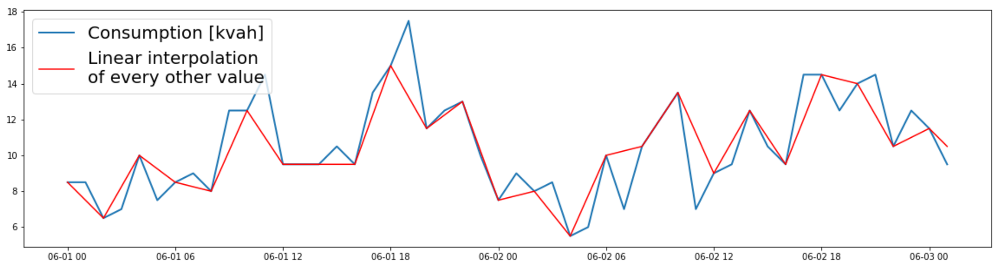
Prices for oil were not updated during weekends, but were elsewise very consistent. Missing values were thus set to inherit the most recent existing value.

2.4 Exchange rates

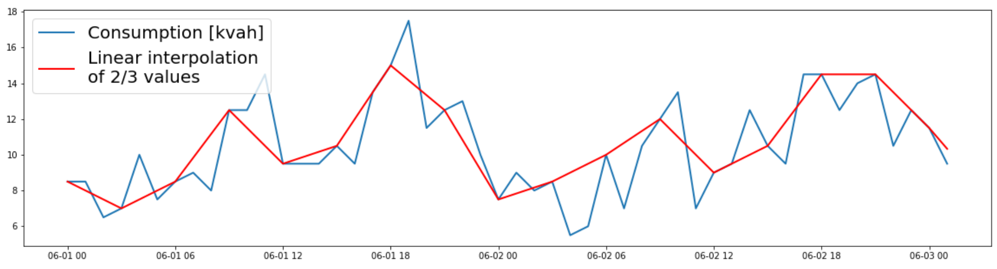
The same tendency applied to the exchange rate data as to the oilspot prices. Weekend values were missing and were set to inherit the most recent existing data.



(a) Traditional interpolation. Root Mean Squared Error: 2.95.



(b) Linear interpolation of every other value. Root Mean Squared Error: 1.39.



(c) Linear interpolation of two out of three values. Root Mean Squared Error: 1.80.

Figure 4: A brief test of different methods of interpolation. A reference complete time series of four months was chosen for validation. Note that these plots offer a zoom-in on a period of two days.

2.5 Weather data

This section requires more elaboration due to the more intricate nature of the missing values.

Half the weather time series were impeccable whilst a few suffered from mild or severe losses. For one station entire months were missing. To best compensate for this was exploited the fact that all weather reports were from the same county in Norway and thus bore resemblance to one another. The following algorithm was suggested to fill the missing values:

```
Identify other location with most similar characteristics in terms of mean and std
Z—transform other's time series based on other's characteristics
Detransform other's values using local characteristics and use to interpolate local
```

Six months of complete data from the most severely damaged time series were used for validation. It should be mentioned that a few additional approaches - such as a median or mean interpolation based on all stations were explored. However, the aforementioned algorithm yielded the best results

- a root mean square error of about 2.6 compared to 3.3 when using mean values. An illustration of the Z-transformed fit can be observed in fig. 5.

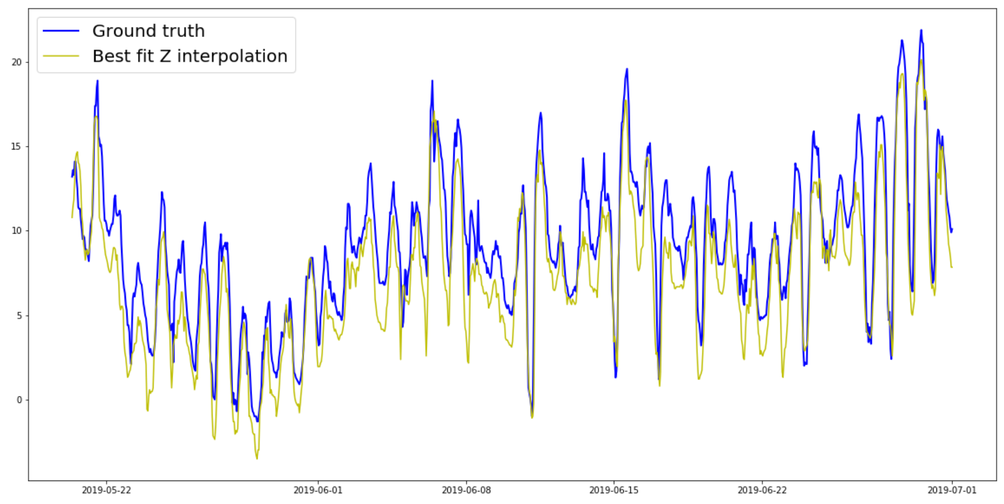


Figure 5: A temperature time series snippet estimated by identifying the overall most similar other time series, Z-transformed. Estimated root mean squared error was 2.6 over a period of 6 months for this particular series.

3 Data Analysis

This section concerns the core of the project, the analysis. The analysis hierarchy starts with descriptive analysis in the form of a few general graphs and summaries, and a sample data cube. Further on the results from statistical analysis using Granger causality tests and frequent itemset mining will be presented.

3.1 Descriptive analysis

The raw consumption time series constitute a data frame of 330K lines, and 5 feature columns containing metadata such as temperature and el prices. It can be quite daunting to approach such amounts of data directly, so a few visualizations and summaries can help as we start out.

First off it is useful to verify that the data set is rather balanced. fig. 6 shows the inter distribution of rows across months and customers. The length of the time series vary, but the number of rows is quite evenly distributed across months, but with notably more data for January. These aspects should be taken into consideration as we interpret future results.

Secondly it could be useful to gain insight into the dynamics of the time series. fig. 7a was produced by normalizing the mean consumption and experienced temperature of every customer per month of the year, and fig. 7b by grouping by week for one single (typical) customer. The same was done for el prices, but across the entire time span from 2018 to 2020 such that the relative price was equal for all customers at any timestamp. Temperature is represented by color, prices by shape sizes, and consumption by level.

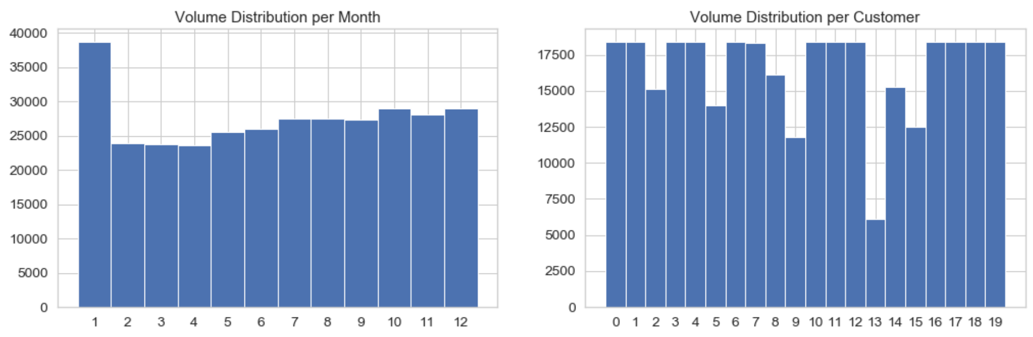
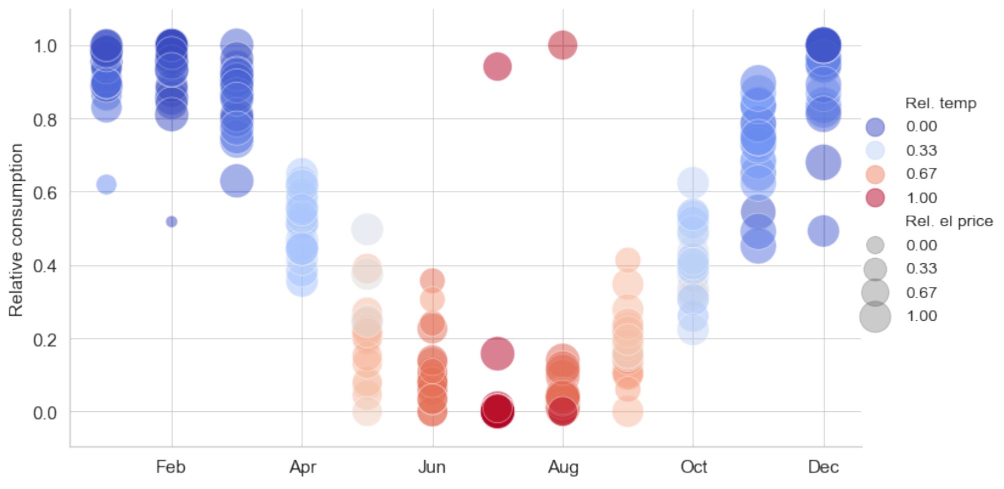
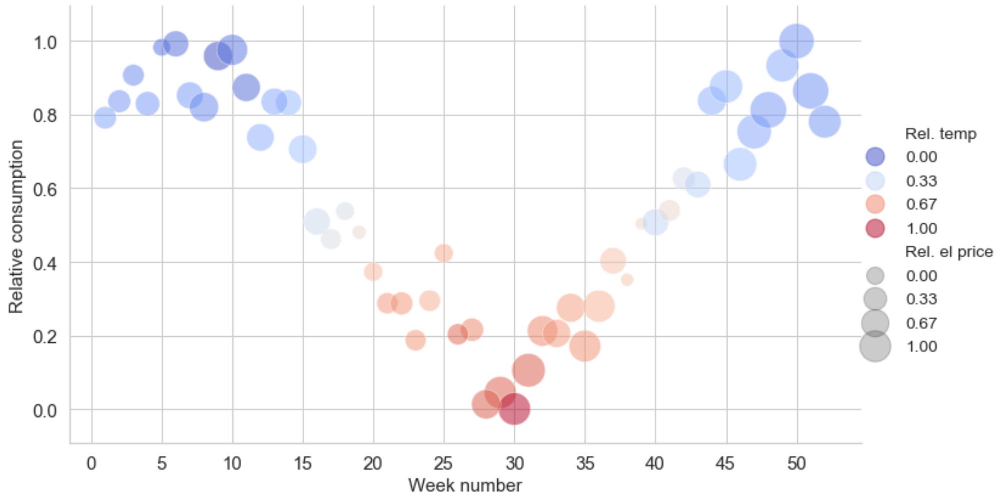


Figure 6: Distribution of rows across months and locations.



(a) Mean relative consumption, temperature and el prices per month of the year. Each circle represents one month, as experienced by one customer. All customers are present.



(b) Mean relative consumption, temperature and el prices per week of the year as experienced by customer at location 10.

Figure 7

Not surprisingly, it can be noted that power consumption increases during winter, and it looks as though prices are relatively high during the last period of the year. Some customers seem to be atypical, having very high consumption during summer. These customers could be cabins used during summer, or perhaps an industrial customer with more production during July and August.

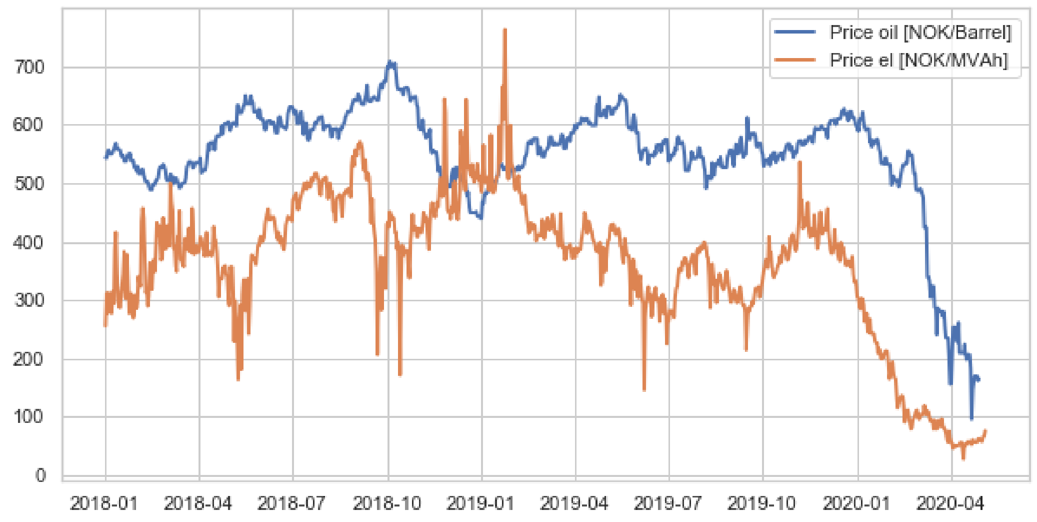


Figure 8: Price development in oil and el prices.

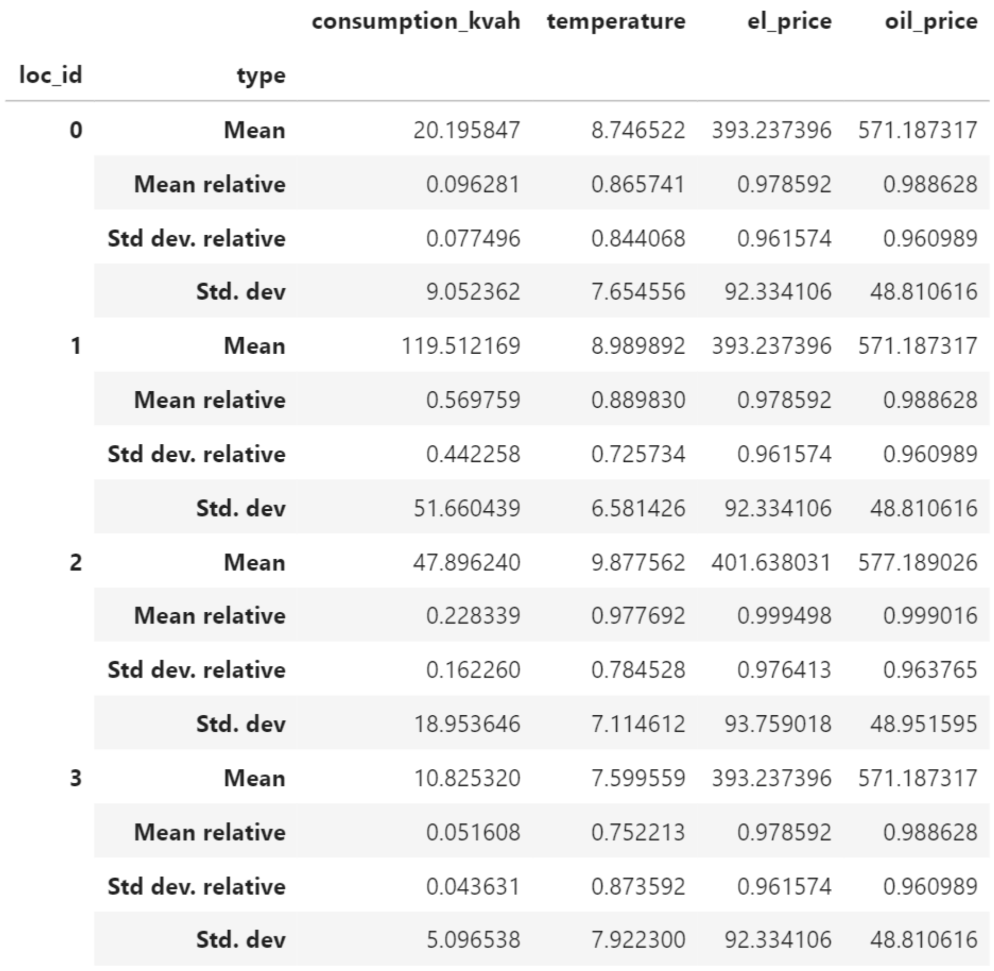
Very important to further interpretation is the knowledge of the recent development in oil and el prices, as can be observed in fig. 8. There are **multiple** reasons for this, such as politics and price war, most of which are not modelled in this project.

Data cube
A sample data cube was assembled to present an insight to statistical characteristics of the different customers. Part of the cube is presented in fig. 9. This cube can serve to illustrate how consistently, and to what degree compared to others, the different customers consume power and how their local weather is. The experienced oil and el prices are naturally very similar to one another. The **relative** rows are calculated as $mean/max(mean)$ or $std/max(std)$. I.e. how the customer compared to the customer with the highest mean or std.

3.2 Statistical Analysis: The Granger causality test

The Granger causality test aims to determine if one time series has the predictive power to forecast another. In other words, if forecasting future values of X can be more precisely achieved by exploiting past values of Y as compared to only using past values of X, then Y is said to *Granger cause* X. In practice, this is carried out by a series of f-tests.

Resolution plays a big role when utilizing this method. For instance, short-term changes in temperature may not affect el prices, but perhaps long-term changes do. The tests were therefore carried out with different resolutions, and with some varying results. On the technical note, it is important that the time series are stationary. This was assured by using Dickey-Fuller tests and derivatives.



loc_id	type	consumption_kvah	temperature	el_price	oil_price
0	Mean	20.195847	8.746522	393.237396	571.187317
	Mean relative	0.096281	0.865741	0.978592	0.988628
	Std dev. relative	0.077496	0.844068	0.961574	0.960989
	Std. dev	9.052362	7.654556	92.334106	48.810616
1	Mean	119.512169	8.989892	393.237396	571.187317
	Mean relative	0.569759	0.889830	0.978592	0.988628
	Std dev. relative	0.442258	0.725734	0.961574	0.960989
	Std. dev	51.660439	6.581426	92.334106	48.810616
2	Mean	47.896240	9.877562	401.638031	577.189026
	Mean relative	0.228339	0.977692	0.999498	0.999016
	Std dev. relative	0.162260	0.784528	0.976413	0.963765
	Std. dev	18.953646	7.114612	93.759018	48.951595
3	Mean	10.825320	7.599559	393.237396	571.187317
	Mean relative	0.051608	0.752213	0.978592	0.988628
	Std dev. relative	0.043631	0.873592	0.961574	0.960989
	Std. dev	5.096538	7.922300	92.334106	48.810616

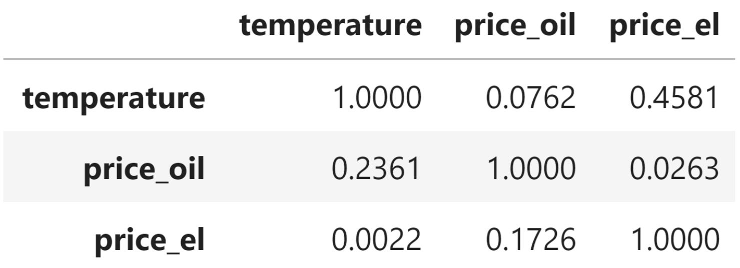
Figure 9: Statistical data cube showing mean values and standard deviations for a subset of the customers (loc_id) and how they relate to the other customers. A high relative mean value means the customer has a mean that is close to the maximum mean across all customers, and vice versa for relative standard deviation.

3.2.1 Results

The customers differ. For some customers, that is, at location **3, 4, 10, 11, and 18** it was suggested that temperature Granger causes consumption when dealing with monthly aggregates. At location **7, 12, and 14** it was suggested that el prices Granger causes consumption.

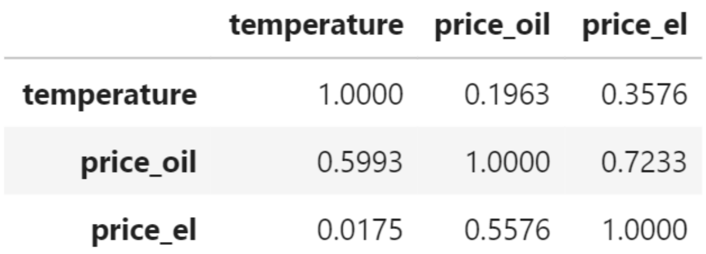
As for the relationship between prices and weather, it was found by monthly resampling that temperatures definitely Granger cause el prices. fig. 10a shows the results from these tests. One peculiar notion from this figure, is that the hypothesis *oil price does not cause temperature* is almost rejected. One might speculate if this is due to the short period for which data is collected and the dramatic oil price drop during the colder months of 2020. However, it was found when resampling by periods of weeks (fig. 10b that this tendency fell short whilst the temperature/price_el relationship remained.

Another result from these tests was that el prices seemingly Granger causes oil prices when dealing with monthly aggregates, whilst the tendency disappeared when resampling by weeks. It could be that local demands do not drive the oil prices, but that long-term tendencies do.



	temperature	price_oil	price_el
temperature	1.0000	0.0762	0.4581
price_oil	0.2361	1.0000	0.0263
price_el	0.0022	0.1726	1.0000

(a) Monthly resampling. From this table we can derive that temperature Granger causes el prices, and that el prices Granger cause oil prices.



	temperature	price_oil	price_el
temperature	1.0000	0.1963	0.3576
price_oil	0.5993	1.0000	0.7233
price_el	0.0175	0.5576	1.0000

(b) Weekly resampling. From this table we can derive that temperature Granger causes el prices.

Figure 10: Results from Granger causality tests. The null hypothesis is *column x does not cause row y*. A p-value of less than 0.05 means we reject the null hypothesis.

3.3 The apriori algorithm for frequent itemset mining

The apriori algorithm does, in all shortness, process transaction sequences and derives from those frequently occurring itemsets. One famous example springs from the domain of grocery shopping, where a relationship between beers and diapers was allegedly found.

The algorithm produces rules on the form:

Items: {'beer', 'diapers'}
Support: 0.5000
BODY -> HEAD[Confidence, Lift]

{'beer'} -> {'diapers'}[1.00, 1.33]

There are a few terms which demand explanation:

1. **Items** is the *itemset*, a set of items that seem to co-occur across the sequence of transactions.
2. **Support** (n_rows containing itemset / n_rows total) indicates the relative frequency of the itemset. A support threshold should be set such that derived rules have substantial support.
3. **BODY and HEAD** describe the suggested relationship within the itemset. It is suggested that the presence of BODY makes the presence of HEAD more likely. Keep in mind that this relationship does not imply causality. Neither does is equal correlation; the presence of HEAD is not expected to affect the presence of BODY.
4. **Confidence** (n_transactions containing BODY and HEAD / n_transactions containing BODY) reveals how often the rule holds water. I.e. when BODY, how often also HEAD?

5. **Lift** (Confidence / Support(HEAD)) reveals how *interesting* the rule is. If HEAD is frequently found regardless of BODY, the rule is considered not interesting. A Lift threshold should therefore be set (and higher than 1) to filter out non-interesting rules.

Since the nature of the apriori algorithm is to mine for frequent itemset, one needs to present it a sequence of transactions. For this particular task we're dealing with numerical time series, so how one chooses to generate the transactions will affect the end results to a high degree. Two main approaches were attempted:

How extreme is the value compared to the mean?
For this approach it was generated for each time step transactions describing each feature by how many standard deviations it was above or below the mean. A series of results were found using different resolutions and thresholds. The entire set of results can be found in the source code[1], and a few selected strong rules are presented here:

Items: {'p3std_price', 'm1std_temperature'}
Support: 0.0049
BODY -> HEAD[Confidence, Lift]

{'p3std_price'} -> {'m1std_temperature'}[0.87, 5.26]

Meaning the presence of prices above the 99.8 percentile makes it likely that temperatures are in the (2.3, 15.9) percentile. *p3std_price* means *el prices are more than 3 standard deviations above mean* and *m1st_ temperature* means *temperatures are between 1 and 2 standard deviations below the mean.* Understanding the syntax, the following next rule

Items: {'m3std_price', 'm0std_temperature'}
Support: 0.0035
BODY -> HEAD[Confidence, Lift]

{'m3std_price'} -> {'m0std_temperature'}[0.96, 1.50]

indicates that very low prices are typically accompanied by normal temperatures. The support for these two rules were 93 and 66 rows, respectively, and median temperatures across the county were used.

When aggregating on days, it was found, with 35 days of support:

Items: {'m2std_price_el', 'm3std_price_oil'}
Support: 0.0409
BODY -> HEAD[Confidence, Lift]

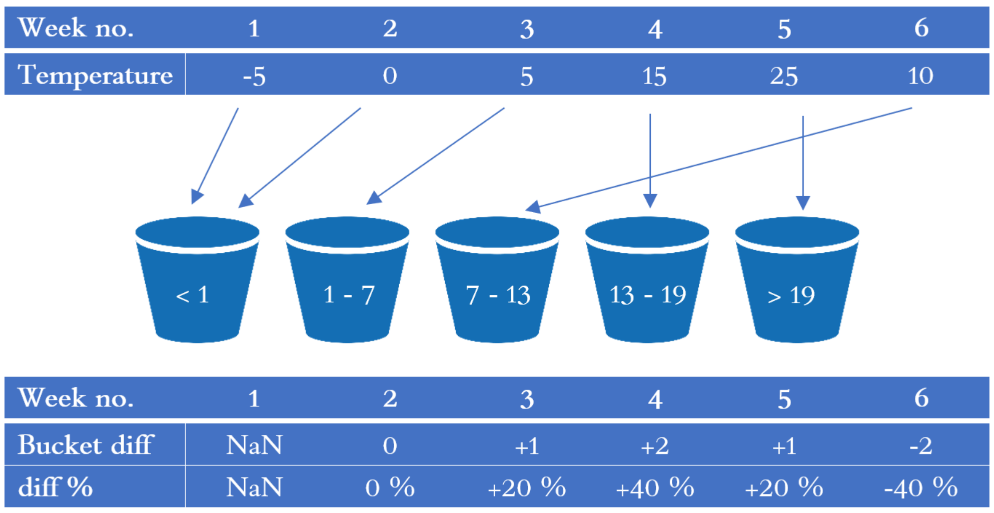
{'m3std_price_oil'} -> {'m2std_price_el'}[1.00, 10.96]

This result should probably be analyzed in the light of fig. 8, showing the dramatic start to 2020. When oil prices hit rock bottom, the el prices were also extremely low.

However, one clear disadvantage with this approach was its vulnerability to outliers. Customers who were seemingly very obsessed with prices turned out to have failing transformers when

[1]https://github.com/arilmad/BI_project/blob/master/miner.ipynb

prices were low. Discoveries like these drove the search for a new approach.



Week no.	1	2	3	4	5	6
Temperature	-5	0	5	15	25	10

Week no.	1	2	3	4	5	6
Bucket diff	NaN	0	+1	+2	+1	-2
diff %	NaN	0 %	+20 %	+40 %	+20 %	-40 %

Figure 11: Bucketing method used to track the time series dynamics over time. The values at each time step are first sorted into buckets, and difference between the current and the previous bucket indicates to what degree the value is increasing or decreasing.

How are the local dynamics?

If we allowed for aggregation of data, for instance weekly or monthly averages, it could be useful to examine how the averages climbed or fell from one time step to the next. The bucketing approach visualized in fig. 11 yielded a series of interesting rules, among which were:

Items: {'p10_temperature', 'm40_el_price'}
Support: 0.0400
BODY -> HEAD[Confidence, Lift]

{'m40_el_price'} -> {'p10_temperature'}[0.95, 3.84]

Here, *m40_el_price* should be read as *el prices drop by 40%*. This does not mean that el prices are reduced by half, but that the price moves down 40% of the buckets are described in fig. 11. This particular rule was backed by 18 aggregated months of data, and seems to suggest that rapidly decreasing el prices typically increase your chance of observing slightly increasing temperatures. Furthermore:

Items: {'p40_temperature', 'm30_consumption_kvah', 'm20_el_price'}
Support: 0.0111
BODY -> HEAD[Confidence, Lift]

{'m30_consumption_kvah', 'm20_el_price'} -> {'p40_temperature'}[0.83, 26.79]

{'p40_temperature', 'm30_consumption_kvah'} -> {'m20_el_price'}[1.00, 6.16]

suggest increasing consumption, decreasing prices and increasing temperatures as a frequent itemset. These rules fall very much in line with results from the descriptive analysis. The lift of 26.79 indicates that temperatures very rarely increase to this degree, and that the presence of m30 consumption and m20 el prices thus is a very good predictor of this particular temperature shift.

4 Discussion

The initial descriptive analyses may present easily accessible information about the different customers, and how they compare to one another. Some customers were characterized by modest and smooth consumption time series, others consumed more and with larger degrees of variation. Already at this point may customers eligible for negotiation be identified if the power supplier needs to free up capacity.

The Granger causality tests suggested that some customers modify their consumption based on prices, but more common was a fixation on weather and consumption. These sorts of tests can be utilized for other parameters, for instance to identify customers whose consumption time series are highly correlated to special hours of the day, or special times of the year. Such information can prove very valuable for the aforementioned negotiations, and probably also for applications outside my scope of knowledge, as these kinds of tests were efficiently run and provided information on an individual level.

The apriori algorithm is very dependent on the quality of the preprocessing. For instance - making changes to the number of buckets could yield no rules. Picking the correct number of buckets was a true conundrum: more buckets reduced the impact of unlucky thresholding, but also reduced generalizability. However, amongst the rules that recently appeared were ones relating higher temperatures to lower prices, and vice versa. The algorithm also identified a relationship between the oil and el prices, which was also produced by the Granger causality test.

The data for which these relationships have been derived is by no means comprehensive enough for substantial confidence to establish. However, some relationships seemed to be intuitively acceptable, such as those connecting consumption and weather, whilst others were counterintuitive, suggesting oil prices affect temperatures. More research should definitively be carried out if one were to convert these results into profitable knowledge, but the project might give hints about which approaches deserve further research - and which are probably best left untouched.

References

[1] *Agder Energi Nett AS.* https://aenett.no. Accessed: 2020-05-03.

[2] *Apriori Algorithm.* https://en.wikipedia.org/wiki/Apriori_algorithm. Accessed: 2020-05-03.

[3] *DNB Historical Exchange rates.* https://www.dnb.no/en//business/markets/foreign-exchange/New/Historical/this-year.html. Accessed: 2020-05-04.

[4] *Granger Causality.* https://en.wikipedia.org/wiki/Granger_causality. Accessed: 2020-05-03.

[5] *MET Norway - Frost API.* https://frost.met.no/index.html. Accessed: 2020-05-03.

[6] *MySQL Connector/Python.* https://dev.mysql.com/doc/connector-python/en/. Accessed: 2020-05-27.

[7] *Nordpool Group.* https://www.nordpoolgroup.com/. Accessed: 2020-05-03.

[8] *Norwegian Climate Service Center.* https://seklima.met.no/observations/. Accessed: 2020-05-03.

[9] *Project Jupyter.* https://jupyter.org/. Accessed: 2020-05-27.

[10] *Python Programming Launguage.* https://www.python.org/. Accessed: 2020-05-27.

[11] *Star Schema.* https://en.wikipedia.org/wiki/Star_schema. Accessed: 2020-05-29.

[12] *StarUML.* http://staruml.io/. Accessed: 2020-05-29.

[13] *Tibber.* https://tibber.com/en. Accessed: 2020-05-03.

[14] *U.S. Energy Information Administration.* https://www.eia.gov/opendata/qb.php?sdid=PET.RBRTE.D. Accessed: 2020-05-04.