Handling Noisy and Untidy Data in the Steel Industry: A Practical Demonstration

T. Transfeld

IU Akademie

28.04.2024

Table of Contents

Table of Figures

Index of Tables

1. Introduction

When raw data comes in, it can rarely be analyzed immediately. Before analysis can be attempted, the datasets need to be cleaned. This is one of the most time consuming tasks in data analysis.[1]

Data preprocessing entails activities that order the information on a structural level, but also looks at the data itself to check for missing values, outliers and making the decision if and how to treat these issues. Well preprocessed data is crucial for the success for everything that follows.

The aim of this paper is demonstrating common methods of assessing and tidying noise in real world data. I use the dataset “Steel Industry Dataset”[2] found several times on Kaggle, with the original source being DAEWOO Steel Co. Ltd in Gwangyang, South Korea, stored at Korea Electric Power Corporation (pccs.kepco.go.kr)[3] which is no longer active. It was chosen as it represents a current problem: Energy consumption and carbon dioxide emissions in the steel industry. The steel industry in Germany is one of the largest single contributors of carbon dioxide emissions[4], so even a small reduction will have a huge impact. While technological changes are the only key to a future with net zero emissions, the change is slow. Data analysis could help find means to make small reductions in the short term. Analyzing the circadian rhythm of energy usage may allow predictions about energy usage on certain days over the course of the seasons, to see if substituting conventional energy with solar power is possible.

As indicated on Kaggle, the dataset has no missing values, and is labelled as clean, but there seem to be issues with the values of the *Date_Time* variable. I found it interesting to look into this, because time issues in software are generally a bit complicated. There are a lot of irregularities, such as leap years, leap seconds, exceptions from regular leap years, and this is just the beginning. Furthermore the dataset needs to be evaluated for outliers and the claim that there are no missing values needs to be verified.

I will base my work on the Data Workflow described by Rattenbury et al[5]. I will use additional resources during the different tasks. The Data Workflow consists of three stages, the raw data stage, the refined data stage and the production data stage. The focus will lie on reaching the refined data stage. I will first describe the dataset, then complete the metadata, check the structure, the granularity and accuracy of the values and finally fix issues that were turned up in the process. The refined data can then be used for ad hoc reporting. Each step will start with profiling[6], examining individual values, summaries across multiple values in the form of tables or visualizations, with the goal to understand the dataset.

2. Exploration of Data Quality Issues.

2.1 The Structure, Meta-Data and Granularity of the Dataset

This first step entails the description and exploration of the dataset’s structure. This will very likely turn up quality issues that will need to be addressed before moving to the next step.

1 Dasu T., Johnson T., 2003. 23
2 Kaggle (n. d.)
3 Sathishkumar et al, 2023
4 Klimaschutz Industrie (n. d.)
5 Rattenbury et al, 2017, Chapter 2. A Data Workflow Framework
6 Rattenbury et al, 2017, Chapter 4. Overview of Profiling

When exploring the structure of the table, I will use Wickham's concept of tidy data for making quality judgements and help for making decisions on how to proceed. Wickham[7] distinguishes between tasks of data cleaning such as checking for outliers, parsing dates, imputation of missing values and tidying, which means "structuring datasets to facilitate analysis". Wickham defines the attributes of tidy data as follows:

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

Table 1 contains the variables contained in the table, the units or formats the data is in, as well as a description of each variable. The dataset comes as a csv file with a comma as delimiter. It contains 11 variables. Of these variables, four are time related and six are collected data regarding usage of energy and emissions. The last variable characterizes the load of the steel plant with three qualifiers. There are no reference values for each of the load types.

Table 1: Collection of the dataset's variables[8]

Variable	**Unit/Format**	**Description of the variable, Constraints if known**
date	DD/MM/YYYY HH:MM	Timestamp indicating when the energy consumption data was recorded.
Usage_kWh	kWh	The continuous energy consumption in kilowatt-hour (kWh)
Lagging_Current_Reactive.Power_kVarh	kVarh	Continuous kVarh for lagging current reactive power.
Leading_Current_Reactive_Power_kVarh	kVarh	The continuous measurement of the leading current reactive power in kVarh.
CO2(tCO2)	ppm	Continuous ppm Com: Carbon dioxide emissions in parts per million
Lagging_Current_Power_Factor	%	Power factor in %
Leading_Current_Power_Factor	%	Power factor in %
NSM	S	Continuous variable indicating the seconds elapsed since midnight
WeekStatus	str	Weekend or Weekday
Day_Of_Week	str	Monday, Tuesday, Wednesday, Friday, Saturday, Sunday
Load_Type	str	Light_Load, Medium_Load, Maximum_Load.

Looking at each variable with Wickham's attributes in mind, we can see that overall the principles are observed relatively well already: Almost all variables have their own column. The variable *Date_Time* could be divided into two columns. Each observation forms a row, with data recorded every 15 minutes. This qualifies as fine granularity, which may be changed depending on the question. There is only one observational unit that describes different aspects of energy usage

7 Wickham, 2014, p 4
8 Kanagarathinam et al, 2024, Section 3.1 Data

over time, so only one table is needed. Since a lot of the information can be deduced from other values present in the table, reducing its complexity is an option.

WeekdayStatus and *Day_Of_Week* as well as *NSM* are candidates for this kind of deletion, as this information can be inferred from the timestamp. Doing so will avoid possibly erroneous entries from the columns. It is to be noted, that in this dataset Mondays, Tuesdays, Wednesdays, Thursdays and Fridays are weekdays, whereas Saturdays and Sundays are considered weekend days. This fact is relevant for later analyses, as definitions of working days and weekends are not universal. Before dropping the columns, they are analyzed for errors to assess general cleanliness of the data.

We currently have only one table that is a coherent observational unit, but it may be necessary to form another, that is more coarse and resamples the data to show a sum of daily energy use.

From just looking at the data collected in Table 1 and the csv file, the following issues are noted:

- The *date* needs to be parsed as datetime object, to be able to take any further action.
- *Lagging_Current_Reactive.Power_kVarh* needs to be renamed to *Lagging_Current_Reactive_Power_kVarh* for consistency within the dataset.
- The metadata lacks a definition for the *Load_Type* label. It is a qualitative label with the possible cases *Light_Load*, *Medium_Load*, *Maximum_Load*, with no definition of constraints for each label. Load characterizes the energy demand of industrial plants. It is unclear if this label relates to an entire day or the measuring interval. This requires analysis of the data at hand. If the labels will be used they should be encoded as numbers for easier interaction. This is an example of an action of semantic profiling.[9]
- Overall lack of defined constraints.[10]

2.2 Checking for Accuracy: Duplicates, Missing Values and Mislabelled Data

a) In General

For all text variables *pandas.unique()* was used to test if other values then the expected are in the list. For all numeric values, *pandas.isnull()* was used to check if NaN, NaT or None appear in the file. The tests did not turn up any irregularities.

b) The *date* Column

The first issue is the name of the column. It should be renamed to *Date_Time*, to be more specific about what the variable contains.

There are several issues that can arise with timestamps in timed series, such as missing timestamps, duplicate timestamps and formatting issues. Having clean timestamps is important as it is the axis that every other key relies on and that makes the different values comparable over time.

The next step is parsing the data objects as dates. This step is crucial for all further processing, and has the potential to turn up formatting issues, when they exist. This is also an

9 Rattenbury et al, 2017, Chapter 4. Individual Value Profiling: Semantic Profiling
10 Rahm & Do, 2000, p 5

example of syntactic profiling[11]: Values in this column need to be dates: parsing the content assures that they are dates in the correct format. Parsing the dates as date time objects is important because Python is then able to understand the objects as date, allowing operations that are unique to dates to be performed. No formatting issues arose during the parsing of the timestamps.

The data is collected every day, every 15 minutes for an entire year. 2018 was not a leap year, so we should expect to find 35040 data entries. Counting the rows with Python yields 35041, where the first row contains the keys. While the number of rows corresponds to the expected number, we still don't know yet if there are missing and duplicate timestamps or issues with ordering.

Every date is associated with a weekday and a weekday status. Using Python, I tested, if the association of a date with a weekday and a weekday status was correct. Every date is associated with the right weekday. Mondays, Tuesdays, Wednesdays, Thursdays and Fridays are considered weekdays, Saturdays and Sundays weekends. For the weekday status, consistency throughout the dataset is more important than the exact definition of what days are considered weekdays, as the distinction between weekends and weekdays is dependent on local context.

c) Continuity of Timestamps

I tested if the timestamps are continuously logged every 15 minutes, by comparing the difference of every timestamp and the one before it. During this analysis, I noticed that the time stamp with 00:00 time appears at the last moment of the day, not at the first. Otherwise this procedure revealed that data was logged continuously, no record was left out. This satisfies another limitation in syntactic profiling: Every timestamp required is present and none outside of it.

d) Sequence of Timestamps

It is noticeable that the timestamps with 00:00 appear at the last moment of the day in the list, not at the first. While this could be to a wrong ordering of data, this type of sorting error is unlikely. Further arguments for this are that the second time indicator in the dataset, *NSM* counts 0. This is another indicator, that the midnight timestamps are at the beginning of a day, not at the end.

According to ISO8601, 00:00 refers to the beginning of a day and 24:00 to the end of the day,[12] although both refer to the same moment in time. This results in clocks having either 24:00 or 0:00 as midnight, not both, a possible source of this issue. When working with Python and Pandas, the ordering of the timestamp has wide technical and statistical implications:

Currently, Python takes this timestamp as the first timestamp of the day.[13] This means that all data collected at this timestamp will be associated with data taken roughly 24 hours earlier, leading to data points plotted in the wrong order.

Figure 1 shows a plot of the energy usage over the span of 25 days. In several places more than one line can be seen, where there should only be one. This derives from Python reading a value that is interpreted as the beginning of the day at a place in the column where it is at the end of this day. The values are plotted in the sequence of how they appear in the table. This results in lines going back and forth, because this sequence is not chronological. This plotting issue is not

11 Rattenbury et al, 2017, Chapter 4. Individual Value Profiling: Syntactic Profiling
12 https://www.iso.org/iso-8601-date-and-time-format.html
13 The Python Software Foundation (n.d) date time module

just an aesthetic problem, but illustrates how the values may be treated when doing any operation on them, without taking care of this sequencing issue.

For example: When summing up energy consumption over time periods, the order of timestamps is negligible. But when interpreting the rhythmic quality of the data, it is likely to cause problems: Datapoints may be interpreted wrongly as outliers. Eg. if the factory was shut down from May 5, 20:00 – May 6, 4:00, while working at full capacity before and after this range. The timestamp May 5, 0:00 shows a low energy consumption, the timestamp May 6, 0:00 has an energy consumption equivalent to full load. When evaluating with Python and Pandas, the timestamp May 5, 0:00 will show low energy consumption at the beginning of May 5, and not at the end of the day, while there will be peak energy consumption at the beginning of May 6, even though the factory was shut down for 4 hours already.

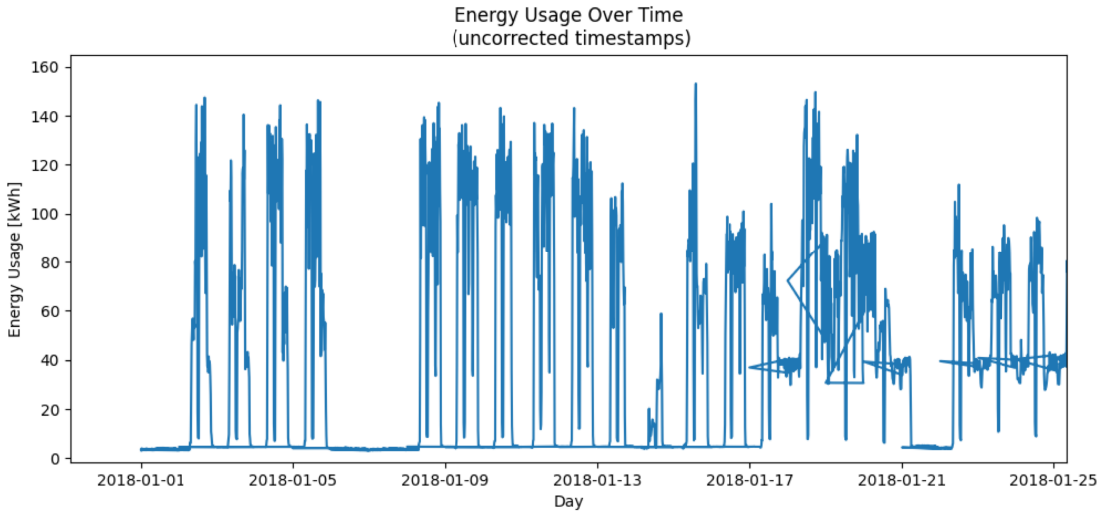

Figure 1: Energy usage over time, plotted with uncorrected timestamps

Looking at Figure 1 makes it obvious that this is not a problem of ordering of the timestamps. If we would just reorder them so that the 00:00 timestamps are at the beginning of the day, we would create energy spikes where they do not make sense.

In order to fix this issue, the simplest method seems to be to change the date of the timestamp, so it will be associated with the following day.

parsed_dates		*corrected_dates*
21.July 00:00 (last moment of the day)	→	22. July 00:00 (first moment of the day)

Plotting the data with the corrected timestamps leads to a graph as expected, shown in Figure 2: The parallel, back and forth lines are gone.

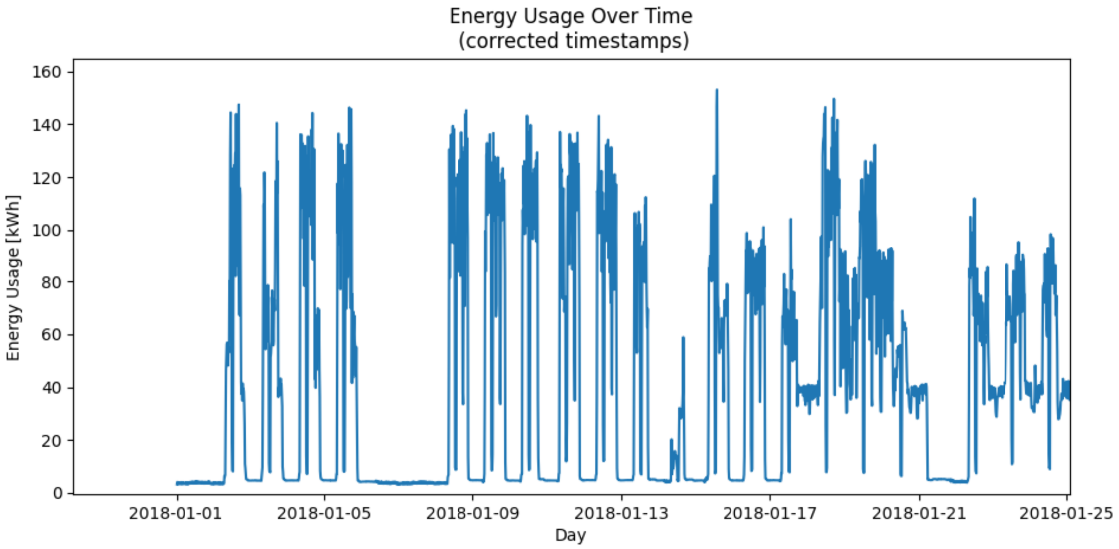

Figure 2: Energy usage over time, plotted with corrected timestamps

While this seems simple and the right thing to do, it does have implications down the line: *Day_Of_Week* and *WeekdayStatus* need to be corrected in the respective lines, if the columns are not removed.

e) Structuring

Structuring are actions that change the form or schema of a dataset.[14] Since the classification into weekdays or weekend days can easily be derived from the timestamp during analysis, the columns *Day_Of_Week* and *WeekdayStatus* are candidates for deletion. The benefits of deletion are that the dataset is more lean, and potentially faulty datapoints may be removed. In the light of the change of the midnight timestamp, this is an excellent option to avoid having to change too many records down the line. The downside is that original information, such as what day is considered a weekday, is lost. This needs to be accounted for in documentation. An alternative is practiced by Kanagarathinam et al.[15] in an investigation for sustainable energy management, based on the same dataset: The scientists parse weekdays as integers from 0-6 and the weekday status as 0 or 1. This would conserve the the original information, while still making the data more comfortable to interact with.

Related to weekdays and weekend-days is also the question of public holidays: A more useful distinction is workdays and non-workdays. Public holidays are often not considered workdays. Following this, the energy consumption of a steel plant may be differ between public holidays on weekdays and ordinary weekdays. To be able to interpret this correctly, having this information at hand enriches the dataset. A column is added that encodes workdays as 1 an non-workdays as 0, by using a source of public holidays[16]. This action combines intrarecord structuring and enrichment[17].

14 Rattenbury et al, 2017, Ch. 5 Section: Overview of Structuring
15 Kanagarathinam et al., 2024, Section 3.3 Data Preprocessing
16 Public Holidays (n. d.)
17 Rattenbury et al, 2017, Ch. 6 Transformation: Enriching

2.3 Checking for Accuracy: Exploratory Data Analysis

In order to clean out noise within the data, one needs to explore the values. A first step is looking at the basic statistical characteristics of the data set. Following that, we will visually check the data, its correlations and characterize the data for outliers.

This step is entails syntactic profiling[18]. We don't know the constraints of many integer based variables in the dataset, but we can examine the dataset and determine constraints.

a) Overview of the Characteristics

To get an overview over the characteristics of the dataset, Pandas has the *describe()* function. The output delivers a count, mean value, minimum value, maximum value, the 25, 50 and 75 percentile, as well as the standard deviation. Table 2 shows the output listed, the numbers have been rounded.

Table 2: Listed output from pandas.describe()

	Usage_kWh	Lagging_Current_Reactive_Power_kVarh	Leading_Current_Reactive_Power_kVarh	CO2(tCO2)	Lagging_Current_Power_Factor	Leading_Current_Power_Factor	NSM
count	35040	35040	35040	35040	35040	35040	35040
mean	27.39	13.04	3.87	0.012	80.58	84.37	42750.00
min	0.00 | 2.45	0.00	0.00	0.000	0.00 | 36.94	0.00 | 12.50	0.00
25%	3.20	2.30	0.00	0.000	63.32	99.70	21375.00
50%	4.57	5.00	0.00	0.000	87.96	100.00	42750.00
75%	51.24	22.64	2.09	0.020	99.02	100.00	64125.00
max	157.18	96.91	27.76	0.070	100.00	100.00	85500.00
std	33.44	16.31	7.42	0.016	18.92	30.46	24940.53

b) Detecting Missing Values from the Overview

The fact that every column has a minimum of 0 strikes my curiosity. While this value is expected for some values, it seems curious for others, for example *Usage_kWh.* Zeros are a way how missing values can be disguised, if they are not labelled as NA or NAN[19]. Looking at the plots as well as relying on common knowledge, it is obvious that a steel plant will never have an energy use of 0 kWh. It will have a low baseline, because even at minimal operation, it will consume energy. Looking up the rows for which *Usage_kWh* is 0, we indeed find a row where all values except the ones that are derived from the date are 0. While this is statistically insignificant in the larger picture, it does seem to be a missing value, that contains zeros instead of non-values. This is something to look out for.

Looking at a plot, fixing this problem is simple with a process called Imputation[20]. This means that the values are replaced with an estimation. I chose the average value of the neighbors, which seems fitting in a dataset with high fluctuations. Alternatives are filling the values with means or median values, marking them as inaccurate, or removing the entire record. In a dataset where fluctuation in the values is high, means or overall averages are not the best option, as they might be regional outliers. Marking the row as inaccurate makes little sense, when all values are 0, as this is not helpful in a later analysis. Also removing an entire row in a timed series seems like a

18 Rattenbury et al, 2017, Chapter 4. Individual Value Profiling: Syntactic Profiling
19 Katzil & Jarmul, 2016, Chapter
20 Rattenbury et al, 2017, Chapter 2: Designing Refined Data

worse choice over filling the gaps with an estimation that fits. In Table 2, the values that were changed through this action are added in red. It is to be noted, that only the minimum values changed, the remaining statistical characteristics did not change.

c) Detecting Outliers: Energy Usage

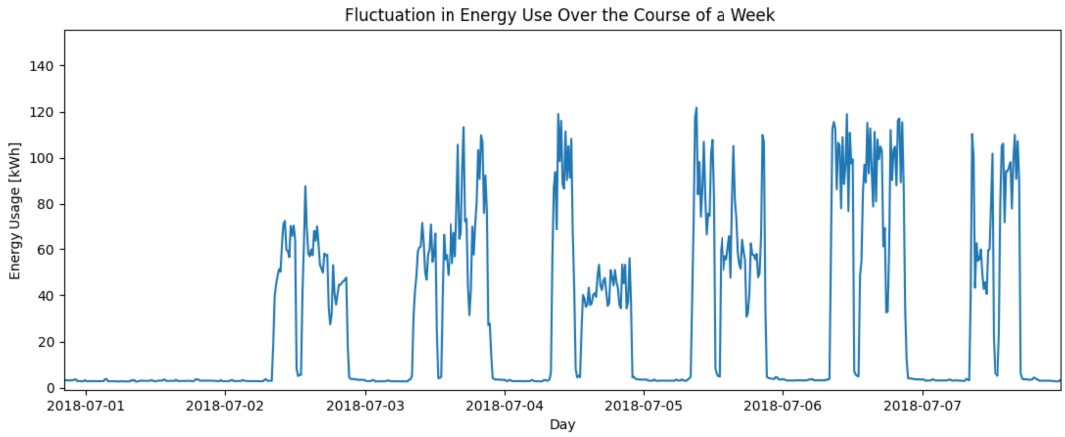

Figure 3: Fluctuation in energy use over the course of a week

Looking at the fluctuation in energy use depicted in Figure 3, it becomes clear that the daily and weekly rhythm of energy consumption will make it hard to detect outliers. A single spike of 100 kWh on a Sunday may be an outlier, the same energy use on a Wednesday morning is not. Due to assumed fluctuations in demand, occasionally Saturdays and in rare cases Sundays have energy uses similar to weekdays. Sometimes weekdays have energy uses like Sundays. A first step is to create a report that distinguishes between energy usage on working-hours and non-working hours, like in Table 3.

Table 3: Characterization of energy use of working-hours and off-hours

	Usage_kWh	
	Weekdays, working hours	Weekdays, off-hours
mean	54.18	5.91
min	2.48	2.45
25%	33.05	2.95
50%	54.65	3.38
75%	73.21	3.92
max	157.18	107.89
std	33.56	11.39
IQR	40.16	0.97
Threshold: IQR * 1.5	60.24	1.46
Outlier below	IQR:← <-27.19	IQR:← <1.5
	Means: <-46,5	Means: <28,26
Outlier above	IQR:← >133.45	IQR:← >5.38
	Means: >154,86	Means: >40,08

To characterize this dataset for outliers, I will use the Interquartile Range and 3 standard deviations around the mean. Calculating the upper and lower bounds for both shows that relying on standard deviation is completely unsuitable for this dataset, as it would qualify the baseline energy usage as outlier due to the naturally occurring fluctuations.

Using the IQR Method, the lower bounds for working days is -27.19 kWh, which does not make sense in the context of the data, so we choose 1.5 kWh as lower bound. The upper bound is 133.45 kWh on working days and 5.38 kWh during off hours. Keep in mind that more energy usage during off hours does not necessarily mean that the data is faulty. A lot of times this means that the steel plant will just work during hours that are usually off.

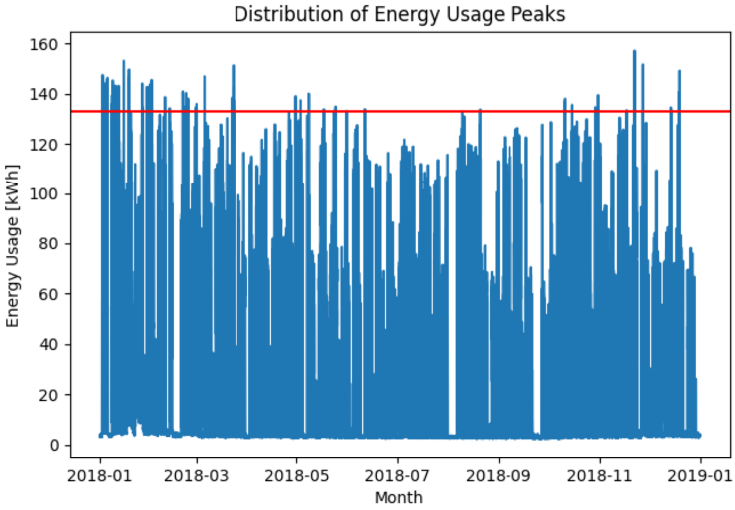

Figure 4: Distribution of energy usage peaks over the year

When selecting datapoints with a usage of over 133.45 kWh, 118 datapoints are returned. Checking in a plot (Figure 4), one can observe that most of these peaks happen before April and after October. This is an example of set based profiling, and answers the question about the "shape and extend of the distribution of values"[21] in a dataset.

Additionally, the peaks always appear in waves of higher peaks, so the energy consumption is likely due to an increase in product demand or need for heating in the winter. It looks unlikely that there are a significant number of real outliers.

When locating outliers below the lower bound in the original dataset, only one datapoint shows up: The zeroed missing values. In the corrected dataset, no outlier in the lower bound shows up.

A more fine grained detection of outliers is possible, but this would require more information such as if the factory was producing goods that day or over the night, or if it was off, and how much energy is spent on production vs. other things, like heating.

d) Detecting Outliers: Carbon Dioxide

The Carbon dioxide emissions will be high if a lot of energy is used and low if little is used. Testing for correlation yields a correlation factor of 0.99, an almost perfect positive correlation. A visual analysis shows that in the first week, CO_2 values are not recorded in accordance with this correlation. This is likely a measuring error. An alternative to spotting this in the plot is testing if the correlation factor is the same for shorter time spans. This would reveal that for the first working day this correlation is off. It can be corrected by substituting the values calculated with the help of the

21 Rattenbury et al, 2017, Chapter 4. Set-Based Profiling

correlation factor. Autocorrelation[22] can be used to check if all workdays have a similar curve of energy consumption.

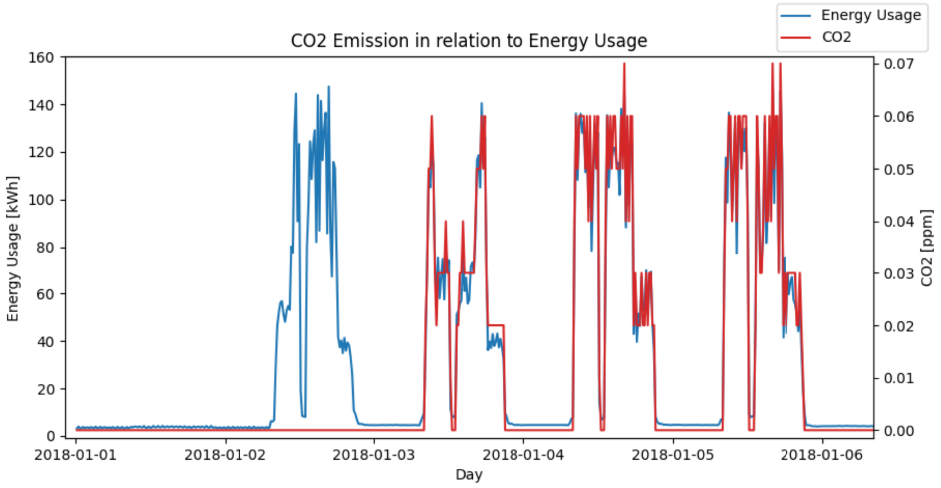

Figure 5: Error in measuring CO_2 emissions in the first week of January 2018

While in this dimension, this error is probably not statistically significant, it is important to implement a check and fix for it, especially if this type of data will be used continuously.

3. Evaluation of Data Treatment

Overall, the dataset has been very clean, and while digging deeper is always possible, it is not always useful and can end up being very time consuming. The most value was added by making the timestamps clearer followed by the zeroed missing values, as well as dropping certain columns and building the structure to parse the values. Adding public holidays is a nice to have, but not strictly necessary to answer the hypothetical question if the energy demand could be covered by solar energy.

Looking for outliers can be a bottomless pitch, with the highest danger of overwriting actually correct data with false values. At the same time, it does not always add more value to the dataset.

4. Conclusion

Although this dataset was labelled as cleaned, I was still able to find numerous issues that were worth changing to improve the dataset's usability. While in terms of outliers, duplicates and missing values, this dataset was relatively clean and nothing significant was turned up, the work to verify that the data is clean needed to be made. In the real world, data is rarely this clean.

I noticed that data preprocessing requires immense knowledge of the context of the data and the data itself as well as a deep understanding of the question that is to be answered by analyzing the cleaned data. This encompasses knowledge of the physical properties of data, knowledge of how the different values should correlate and information about the tools the data was collected with. A lot of this contextual information is not part of the practice dataset, so I resolved to operating on assumptions, where research did not bring me further. I want to mention

22 Chatfield, 2019, p 22

the importance of documenting the assumptions that were made and the decisions that derived from them.

The overlap of the data preprocessing with the actual analysis also becomes clear. The statistical methods that are needed for making predictions from a data set are the same that are needed for making sure the data set is clean. But it is not enough to just apply statistical methods. For superficially similar cleanliness issues, different means of repairs require knowledge of the characteristics of dataset as a whole, the individual value in its row, as well as knowledge of the repairing methods and their effect on the dataset. This requires weighing the drawbacks and benefits of actions and to be able to make informed decisions.

5. References

Chatfield, C. (2019). *The Analysis of Time Series: An Introduction*. Chapman and Hall/CRC.

Dasu, T., & Johnson, T. (2003). *Exploratory Data Mining and Data Cleaning*. John Wiley & Sons.

Kaggle. (n.d.). Steel Industry Datasets. Retrieved April 11, 2024, from https://www.kaggle.com/datasets/ayushparwal2026/steel-industry-datasets/data

Kanagarathinam, K., Dharmaprakash, R., & Sathya, S. (2024). Predictive Modeling of Energy Consumption in the Steel Industry Using CatBoost Regression: A Data-Driven Approach for Sustainable Energy Management. *International Journal of Robotics and Control Systems*, 4, 33-49.

Klimaschutz Industrie. (n.d.). Stahlindustrie. Retrieved April 16, 2024, from https://www.klimaschutz-industrie.de/themen/branchen/stahlindustrie/

Kazil, J., & Jarmul, K. (2016). *Data Wrangling with Python*. O'Reilly Media, Inc.

Python Software Foundation. (n.d.). Datetime Module. Retrieved April 15, 2024, from https://docs.python.org/3/library/datetime.html

Rattenbury, T., Hellerstein, J. M., Heer, J., Kandel, S., & Carreras, C. (2017). *Principles of Data Wrangling*. O'Reilly Media, Inc.

Rahm, E., & Do, H. (2000). Data Cleaning: Problems and Current Approaches. *IEEE Data Engineering Bulletin*, 23, 3-13.

Sathishkumar, V. E., Shin, C., & Cho, Y. (2023). Steel Industry Energy Consumption. UCI Machine Learning Repository. https://doi.org/10.24432/C52G8C

Wickham, H. (2014). Tidy Data. *Journal of Statistical Software*, 59(10), 1-23.

Public Holidays. (2018). 2018 Dates. Retrieved April 18, 2024, from https://publicholidays.co.kr/2018-dates/