TECHNICAL UNIVERSITY OF MUNICH

TUM Data Innovation Lab

"Grid Load Peak Prediction"

Authors	Constantin Gahr, Ayishetu Haruna and Chris Löchel
Mentor(s)	Eva Bammann (sonnen e-Services GmbH)
	Dr. Stefan König (sonnen e-Services GmbH)
Project Lead	Dr. Ricardo Acevedo Cabra (Department of Mathematics)
Supervisor	Prof. Dr. Massimo Fornasier (Department of Mathematics)

Abstract

Renewable energy sources play an important role for the generation of electrical energy. However, their often unpredictable nature additionally strains the electrical grid and could potentially cause outages. For this reason, the German government rewards companies who discharge power at times of high demand and thereby stabilize the grid. In general, this can only happen once a year, namely in the 15 minute time slot of the highest demand.

Sonnen, as a company in the energy market, could potentially profit from these so called "avoided grid fees". Their batteries, which can discharge into the electrical grid, are perfectly suited for the task and could distribute their charge at the peak load. Since it is unknown when these peaks will happen, we were tasked with developing suitable models which can predict the peak with a high accuracy.

We developed three fundamentally different models. First, we created several statistical models where we assign each 15 minutes time slot the probability of it being the peak. Next, we implemented a support vector machine and a neural network which use real-time weather data to predict the peak load. Thirdly, we set up an ARIMA model which predicts the power consumption for the whole year using live data. These models are then evaluated based on their accuracies and their financial profitability.

We finished this project with a web application show-casing our results. We present our predictions in a user friendly way and provide an easy to use code base, ready to be applied on this years data.

Contents

1 Introduction

Chris, Constantin

1.1 sonnen's Business Model

sonnen was founded in 2010 by Christoph Ostermann and Torsten Stiefenhofer and is located in the small German village Wildpoldsried. The company wants to create a world in which "everyone is able to cover their energy needs with decentralized and clean energy sources" and aims to "emancipate our world from the dependence on fossil fuels and anonymous energy corporations" (see [**1**]).

Their main product is the sonnenBatterie, an energy storage system used to store renewable energy typically produced by solar panels. As such, their common customers are medium sized households and small companies.

Normally, this solar power is generated during the day while the power consumption peaks in the morning and evening (see figure 1). Therefore, there is a mismatch between production and consumption which makes self-sufficient energy production using only solar panels unsustainable. Instead, these homes sell their generate power to the grid and buy it back in the evening - for a much higher price.

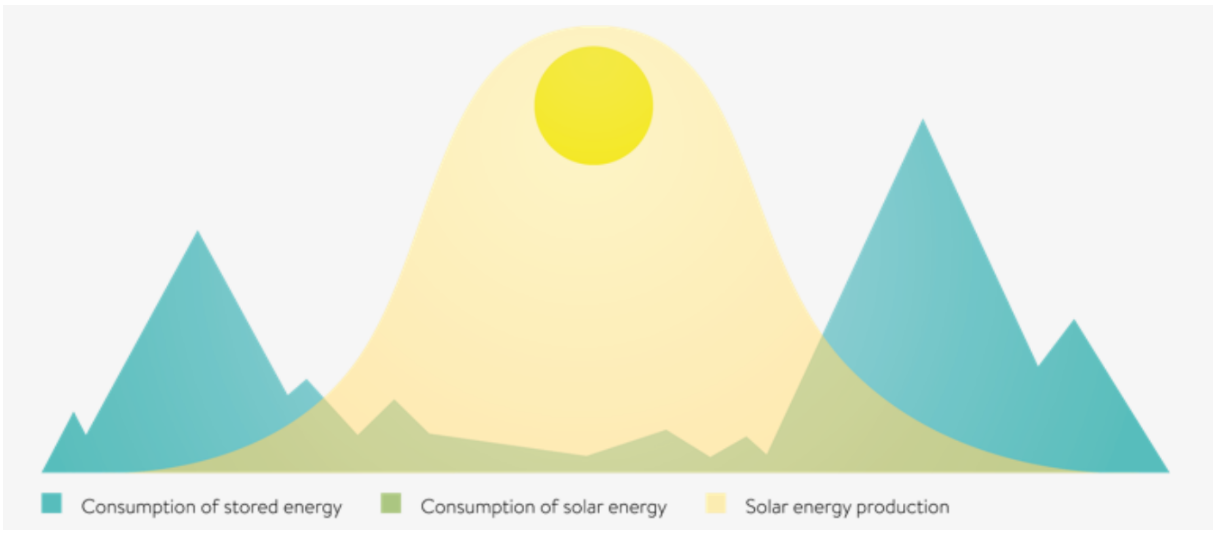


Figure 1: Average daily energy consumption and production (using solar panels).

sonnen solves this problem. Their battery stores the power generated and discharges whenever the customers need it, let it be for cooking, washing or just watching TV. This way, they allow their costumers to use the maximal amount of energy generated by their solar panels.

sonnen even goes one step further. They combine several of their batteries together and create a so-called virtual power plant. As such, they act as a power plant discharging energy to the electrical grid, if needed, but also buying energy if the price is low. This creates interesting opportunities, some realized, others planned.

Right now, costumers can use the sonnenCommunity (see [**2**]) and the sonnenFlat which provides customers with as much energy as they need for a fixed monthly price. In return, sonnen can use these batteries and distribute the power, let it be for charging electrical vehicles (see [**3**]) or stabilizing the grid.

1.2 The German Energy Sector and Power Grid

In 2017, on average 15.9% of the energy produced in Germany came from renewable energy sources like wind and solar (see [**4**]). Unfortunately, renewable power sources put additional burden on the already strained electrical grid.

Traditionally, power was generated by big power plants using fossil or nuclear energy sources. It was then distributed on the national level by the transmission system operators (TSOs) and further spread on the local level using the distribution system operators (DSOs) (compare figure 2, see [**5**]).

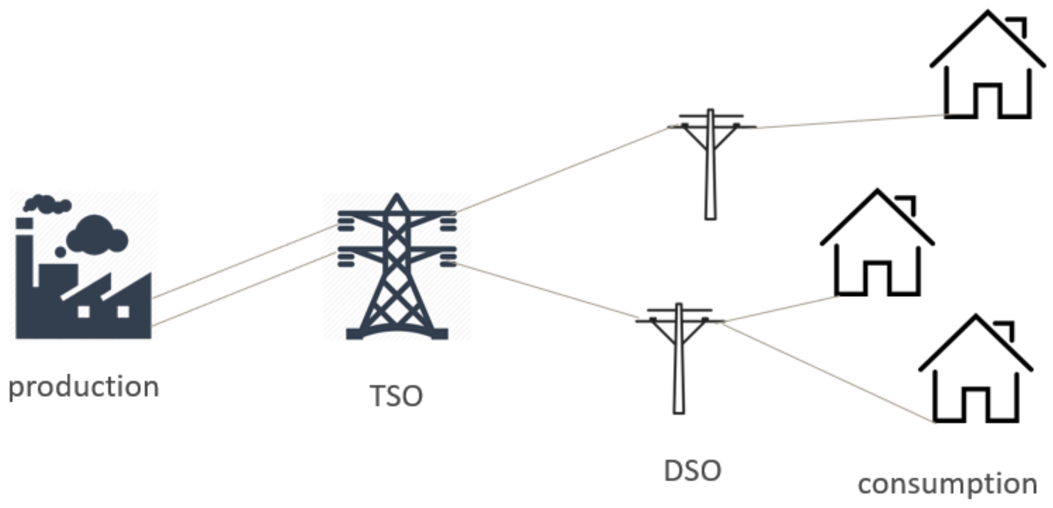


Figure 2: The traditional German power grid

However, with the invention of better, green energy sources and the goal of being nuclear power free by 2022, this changed. Energy produced by large generators is usually fed into the high voltage transmission system. In order to reach the consumers which are connected to the low voltage distribution system, energy must pass a transformer. In times of high consumption, the large quantity of energy that flows from the transmission to the distribution system may lead to an overcharge of the existing transformer.

At this point, sonnen's virtual power plants enters the game. They do not only generate energy but are able to store it as well. As such, they can absorb power in times of high energy production and redistribute it in times of high demand. Additionally, the DSOs reward such behavior and provide a financial incentive for stabilizing the electrical grid. They are called "avoided grid fees" (see [**7**], [**8**]). This means discharging electrical energy into the grid when the load reaches its yearly peak one can generate high revenues.

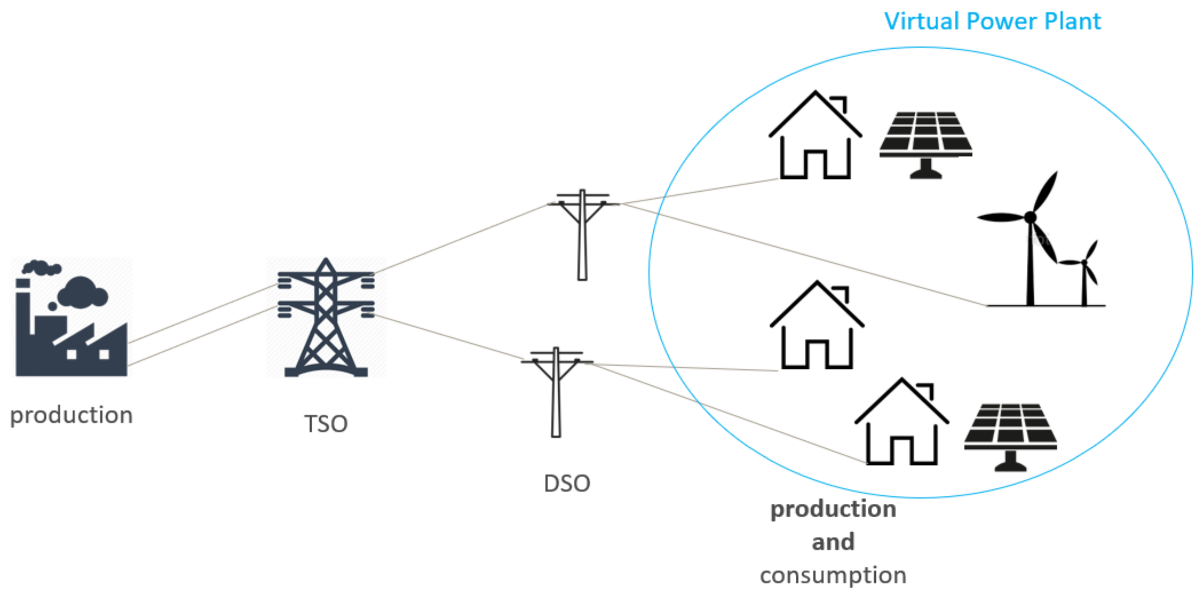


Figure 3: The German power grid with renewable energy sources

1.3 Problem definition and goals of the project

The goal of our project was to predict these peaks. Using different statistical and machine learning techniques we are now able to identify the peaks with a high accuracy. In chapter 2, we first discuss which data we used, where we got it from and which steps were necessary to ensure the correctness of the data. In chapter 3, we introduce the employed methods, provide their mathematical background and evaluate their results. We continue with our web application in chapter 4. Here, we show how Sonnen can use our algorithms and provide an exemplary implementation. Lastly, in chapter 5, we conclude with an overview over our results, what we achieved and which steps should be taken next.

2 Statistical Data Exploration

In a first step, we acquire the power measurements we want to use in our models (section 2.1). After having loaded the various files into our data base, we cleaned the data and verified the data (section 2.2). Next, we analyzed the data which is explained in detail in section 2.3.
Additionally, we also wanted to use weather data. The process of acquiring, cleaning and parsing this data into the database is described in section 2.4.
The whole process of getting, cleaning and using the data is describe in figure 4.

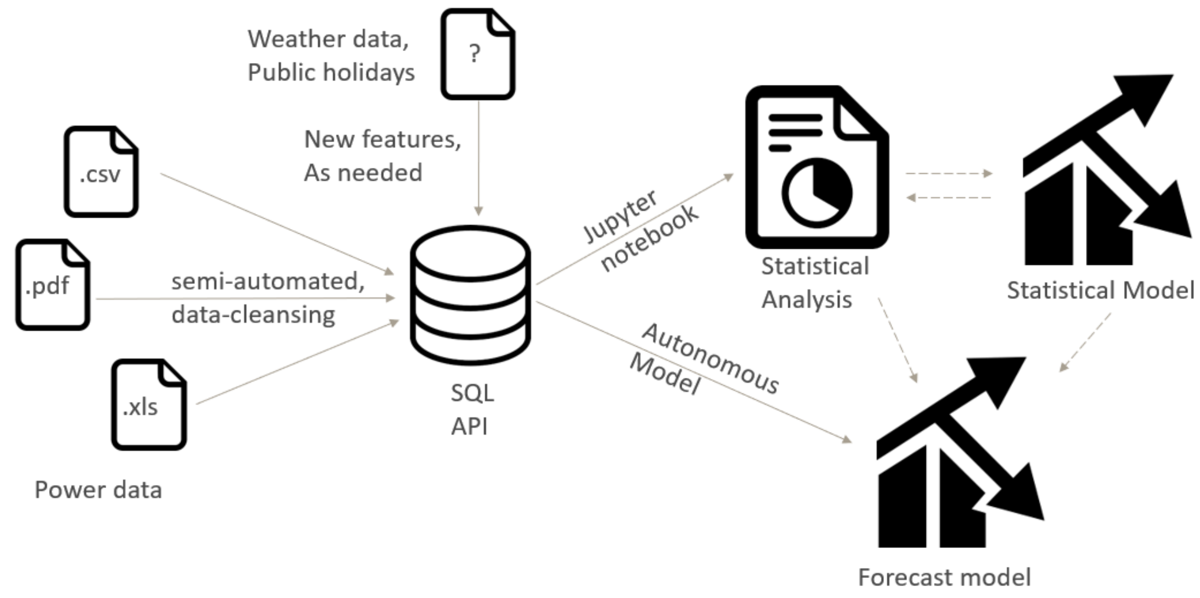


Figure 4: Data exploration setup

In the end, we developed several models with are going to be described in detail in chapter 3. They are a statistical model (section 3.1), a support vector machine and neural network (both in section 3.2) and finally a different approach using the ARIMA model in section 3.3.

2.1 Data Acquisition

Chris

We started by downloading the electrical grid load over the year. By law, every DSO in Germany has to publish these online (§17 Stromnetzzugangsverordnung, [**9**]). Unfortunately for us, the law doesn't include a mandatory format. As a consequence, we encountered various different file formats, namely csv-files, xls-files and even pdf-documents. We then parsed this data into our database. In order to do so, we wrote a python script that allowed us to automate this process up to a certain point. Nevertheless, it needed to be adapted to every single data set since the files often had no common structure.

Unfortunately, we weren't able to use the data provided by "Netze BW GmbH". They published their data as copy protected pdf-files which aren't machine readable.

In the end we found power measurements for around 70% of Sonnens most important DSOs (the ones with the most batteries), where around 25% of the missing measurements came from Netze BW GmbH.

2.2 Data Cleansing

Constantin

As soon as we had the data in our data base, there were several more steps in order to cleanse the data so that it could be used for our analyses.

First, we got rid of incorrect or implausible measurements. For this purpose, all measurements containing NaN, -1 or no value at all were removed.

Next, we converted all dates into wintertime[1]. Here, we encountered a strange issue were the measurements where not in a monotonous ordering. For example, we encountered the following data (with unconverted timestamps):

Date	Time	Power [MW]
01.05.	23:30	1241346
01.05.	23:45	1202197
01.05.	00:00	1167392
01.05.	00:15	1137825
01.05.	00:30	1109169
01.05.	00:45	1084797
02.05.	01:00	1059780

As you can see, the date should have changed to the second of May while it clearly didn't. We guess this was caused by some mixing of summer and winter time on the DSO side. We fixed this problem while converting to wintertime and obtained data which is in a monotonous order.

Lastly, we introduced a new column to our data. In order to make the DSOs comparable, we wanted to consider the power usage relative to the maximum power usage per year rather than absolute values. For this purpose, we scaled all the power values such that the minimum gets mapped to 0 and the maximum is mapped to 1.

2.3 Statistical Data Exploration

Chris, Constantin

As soon as we had prepared our data, we were interested in how it is distributed. For this purpose, we defined plots that seemed interesting with the goal of identifying factors that influence the energy consumption. In a first step, we decided to look at the distribution of maxima within a day, within a week and over the months of the year. In the following, each of these plots will be presented in a subsection.

Here, we briefly want to comment on a very interesting phenomenon we saw when first working with the data. At the beginning of our work, we identified two different groups.

[1]We made the conscious decision to use wintertime so that there is no ambiguity about which time we are talking about. Also, depending on the DSO, different timezones were used (i.e. only wintertime or winter and summer time mixed). This conversion then forced us to really check if the dates are all correct.

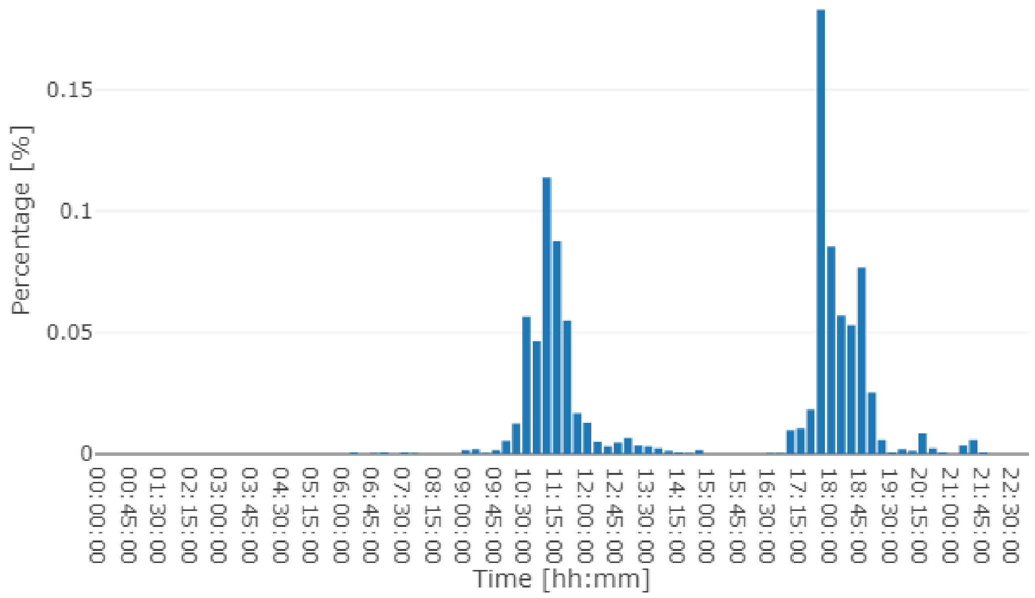


Figure 5: Distribution of maxima within a day

The first one had their maximum at around 18 o'clock and the second one, consisting of measurements from a DSO located in Berlin, had their maximum around noon. We had trouble explaining why the DSOs behave so differently and presented our work in Milestone Meeting 1. While investigating it further, we checked the data we downloaded and found out that there were two different data sets on the website of the DSO from Berlin which had almost the same name, one being called "Lastverlauf" and the other one "Jahreshöchstlast und Lastverlauf". It turned out that we downloaded the wrong one and the phenomena disappeared as soon as we corrected the data.

2.3.1 Distribution of maxima within a day

In figure 5, you can see the distribution of maxima within a day as a bar plot. On the x-axis, we see the time in 15-minute intervals and on the y-axis the percentage of maxima within a day that occurred at that time. We can see that the maximal power usage occurs around 12:00 o'clock and around 18:00 o'clock. However, while only looking at the blue bar plot, we don't know how high this power usage is and whether it is high enough to be interesting for us. That's why we decided to only look at values that are high enough in a next step.

To investigate the relationship between the 12-o'clock and the 18-o'clock maxima further, we generated a bar plot of the highest 95% of the power usages of the year and of

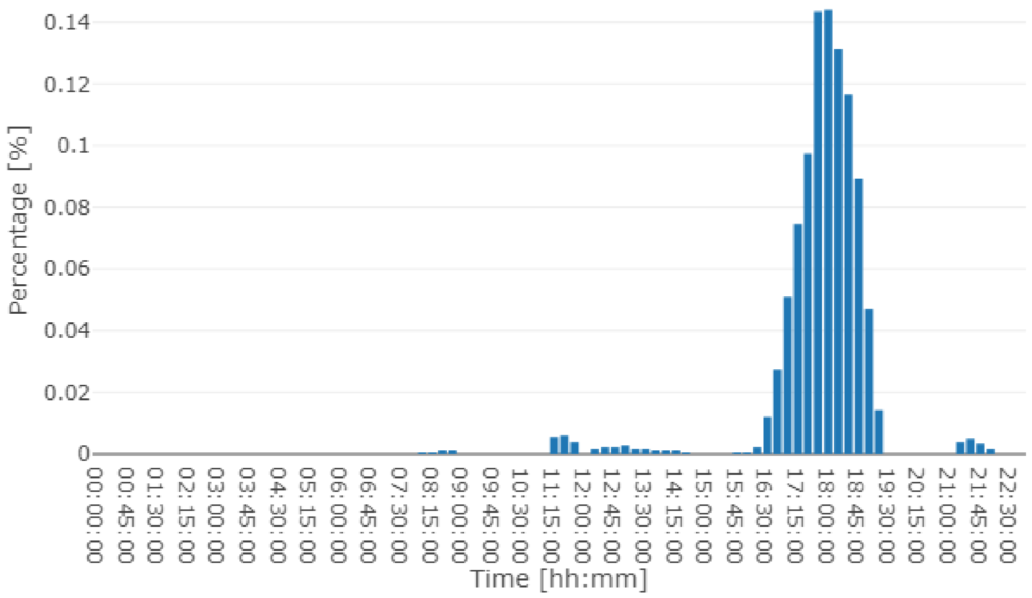


Figure 6: Distribution $\tilde{\rho}_{T,\sigma}$ of the highest 95% within a day

the highest 100% of the power usages of the year which you can see in figure 6 and figure 7, respectively. It is easy to notice that the peak around 18-o'clock is much stronger than the one around noon. Nevertheless, we see in figure 7 that still 16% of the 15-minute time slots with the highest energy consumption happen around noon (here 13% at 11:30 o'clock and 3% at 13:30 o'clock where we have to keep in mind that we converted the mix of summer- and wintertime into wintertime only).

2.3.2 Distribution of maxima within a week

In a next step, we wanted to find out on which days maxima typically occur. For this purpose, we first generated a bar plot that shows how many maxima of the week occurred at which day (see figure 7). You can easily notice that roughly 21% of the maxima within a week happen at Wednesdays, followed by 19% on Tuesday and 18% on Thursdays. In contrast to that, the maximum power usage happens on the weekend with a probability of roughly 15%. Again, we can only see the number of maxima within a week but not the respective value compared to the overall maximum. In order to analyze this, we again have to look at the distribution of the highest 95% of the values and the distribution of the peak loads of the year.

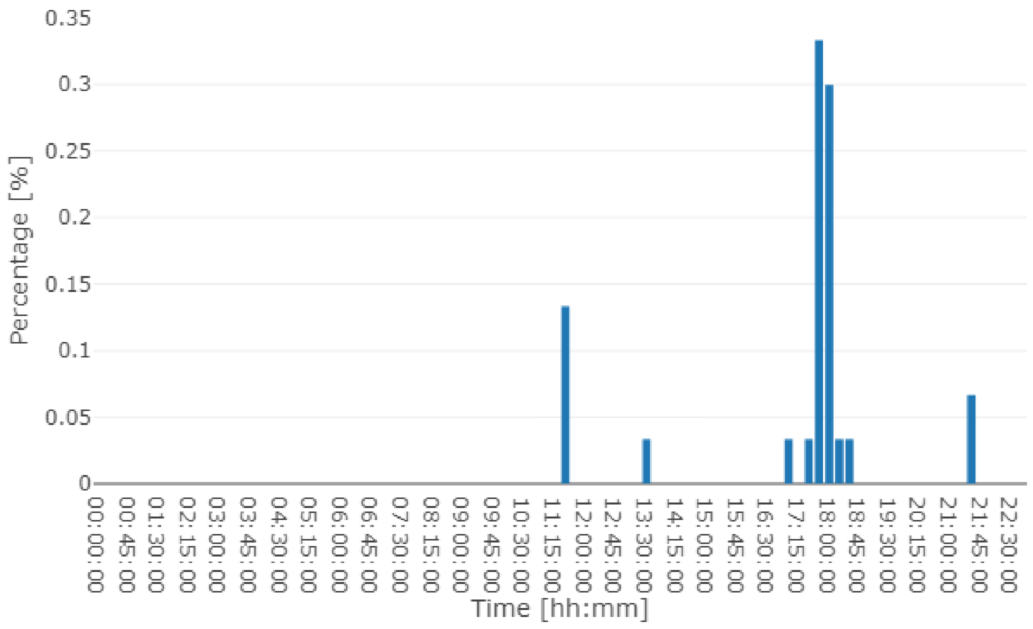


Figure 7: Distribution of the highest peak load within a day

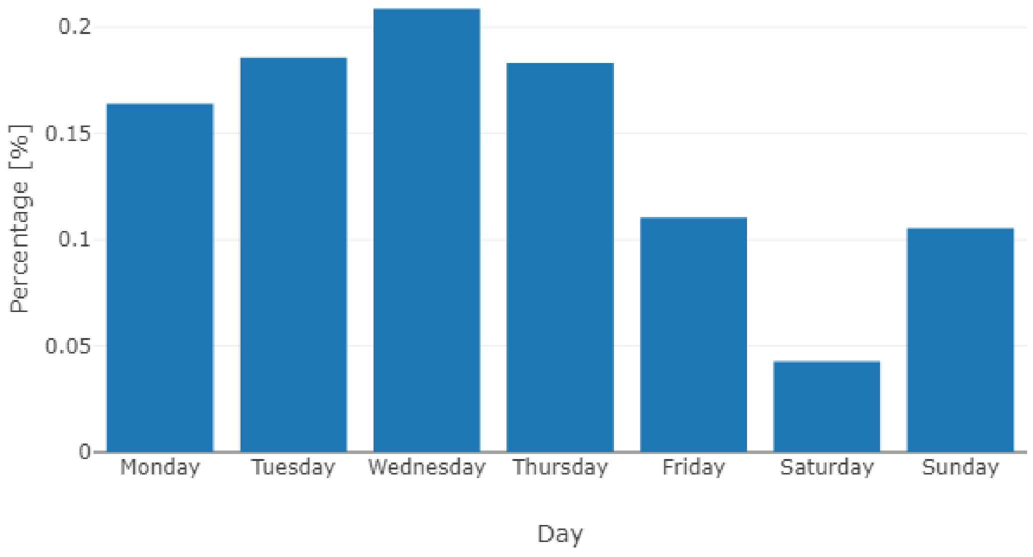


Figure 8: Distribution of maxima within a week

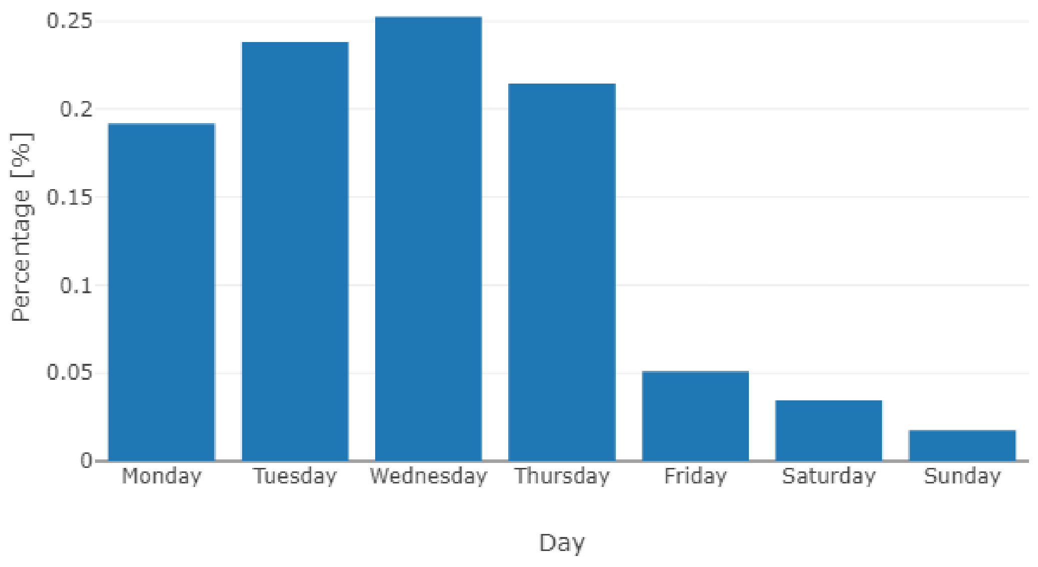


Figure 9: Distribution $\tilde{\rho}_{D,\sigma}$ of the highest 95% within a week

To investigate the distribution within a week further, we plotted the distribution of the highest 95% and the highest values within the year of all data. You can see these distributions in figure 8 and 9, respectively. It is very interesting that the highest 95% occur within the week with probability 95%. Neverthless, if you look at the distribution of the highest peak load within a week, you can see that 27% still happened on the weekend (17% on Sundays and 10% on Saturdays). That means that there are almost no high values measured on the weekend but if there is an outlier, its load is so big that it can be a peak.

2.3.3 Distribution of maxima over the different months

Last but not least, we were interested in the distribution of the maxima over the different months. In this case, it is sufficient to only look at the distribution of the highest 95% and the overall maxima since the maximum within a year is already the quantity we are looking at. Considering figure 11 with the highest 95%, you can see that most of the maxima occur in the winter time, namely between October and February. Nevertheless, there are still some values appearing in April and June. When comparing these observations with figure 12 displaying the highest values only, we see that 77% of the highest peak loads happen in January. Besides, there are 13% occurring in February, 7% in December and 3% in April.

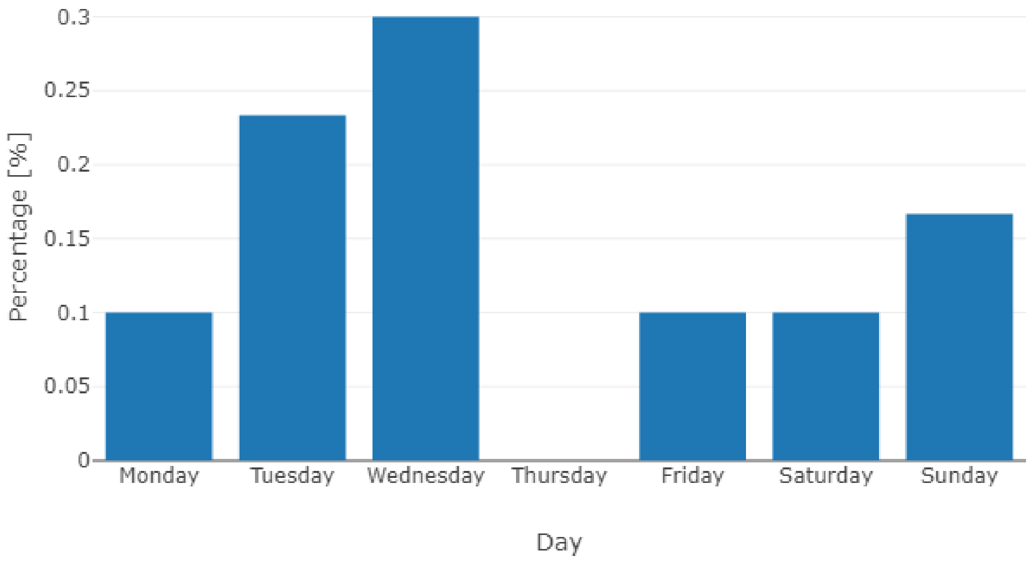


Figure 10: Distribution of the highest peak load within a week

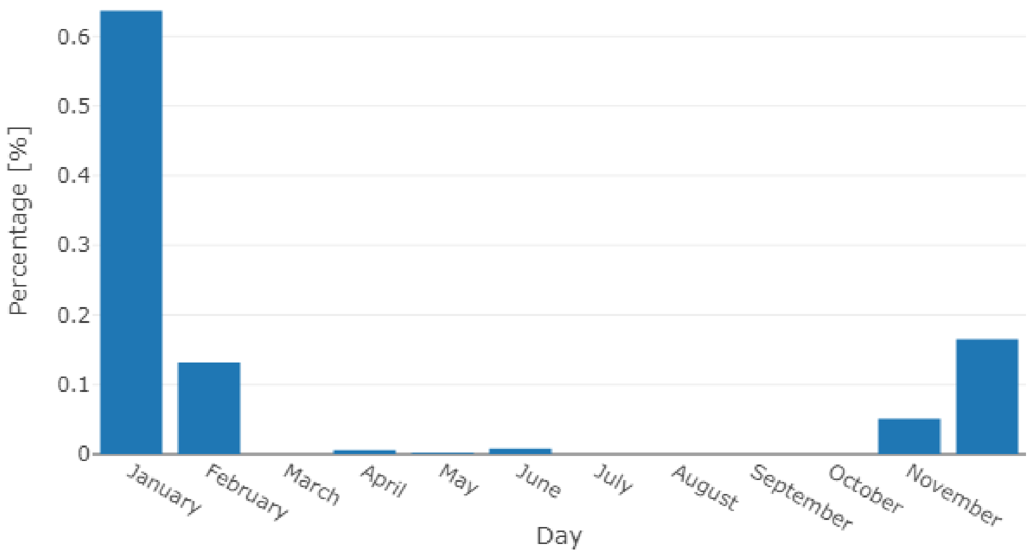


Figure 11: Distribution $\tilde{\rho}_{M,\sigma}$ of the highest 95% over the different months

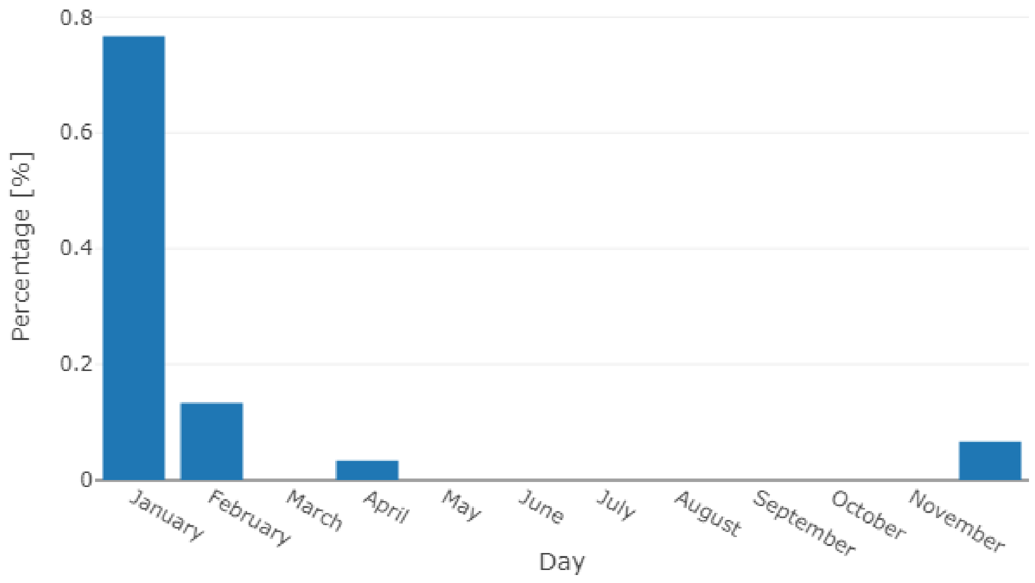


Figure 12: Distribution of absolute maxima over the different months

2.3.4 Conclusions

The statistical data analysis is a very powerful tool to give us a first intuition as to when the grid load peak occurs. We have seen that it happens with high probability either around 18:00 o'clock or at around 12:00 o'clock. In a next step, we saw that the chances are very high that it happens on a weekday, most probably on Tuesday or Wednesday. Furthermore, most of the peak loads occur during the winter time with a high probability that they happen in January. These results are very interesting and can be used for setting up our prediction model.

2.4 Weather Dataset

Ayishetu, Chris

In order to further incorporate feature integration to facilitate the model creation, weather data is explored next. This data is made available on the Climate Data Center website of the Deutscher Wetterdienst[2]. The datetime, station and area filters provided facilitates acquiring weather data for specific DSO stations, time range and location. The dataset is downloaded only for the corresponding DSOs and years in which we have power data available. The metadata of this dataset features hourly station observations of air temperature, at 2 meters above ground, in degree celsius.

2.4.1 Weather Data Cleansing

The python library Pandas, ideal for handling files and parsing the data (for instance date) into the desired format, is utilized in this process. After reading in the data, we convert the date values into datetime. We maintain only the columns needed; such as DSO ID, datetime and temperature. The "txt" files are then converted into "csv" for easier handling.

We then run a check for missing time interval readings. Having identified some missing values in the temperature readings, we treat these by using interpolation to fill in the missing values. The interpolation method selected for this purpose is "time". Time interpolation is chosen as this works on daily and higher resolution data to interpolate given length of interval. Following the decision we made to handle ambiguities with regards to what time we are talking about, we then convert the weather dates into wintertime.

2.4.2 Weather Data Exploration

An initial visualization of the weather data is done using a scatter plot for the temperature values. A common pattern is observed throughout the whole year across all DSOs. The plots show low temperature values from November till February and/or March the following year. This corresponds to autumn and winter. Temperatures begin to rise again between March and/or April till September; corresponding to spring and summer.

[2]The Deutscher Wetterdienst (DWD) is a federal authority in the business area of the Federal Ministry of Transport and Digital Infrastructure responsible for the fulfilment of the meteorological requirements of all economic and social areas in Germany.

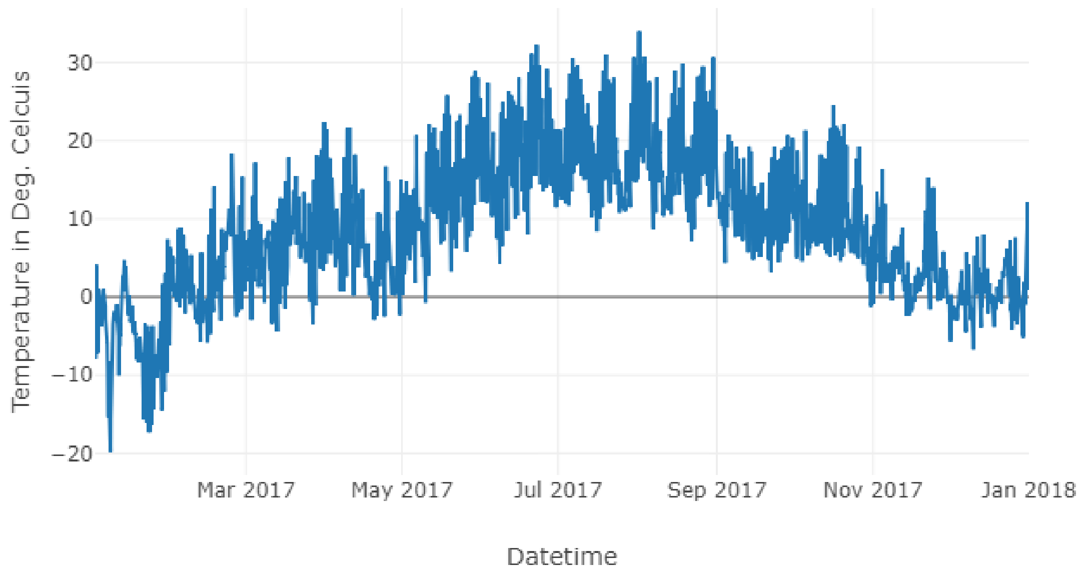


Figure 13: Temperature distribution over the year

2.5 Correlation Between Power and Temperature

A further analysis of the weather data is carried out by exploring the correlation between temperature and power. To ensure consistency, hourly temperature observations are resampled into 15-minute time slots to match the power data. The values for the new 15-minute time slots are then created using interpolation as done above for the missing data. The power data is then merged with the temperature data, using datetime as the common index or unique id, for the available DSOs.

2.5.1 Correlation for Entire Dataset

We first run a correlation function on the entire dataset, where we group by DSO and this produces the following results. In this instance, we observe the correlation values to all be negative. This implies that power and temperature are negatively correlated. We also observe that although there exists a negative correlation, this is not a strong correlation for most of the DSOs. To have a strong correlation, we should except values closer to 1 in absolute value. However, we obtained values between -0.7 and 0.

DSO ID	1	3	5	10	11	12	13	14
Correlation	-0.482	-0.145	-0.270	-0.661	-0.433	-0.115	-0.464	-0.044
DSO ID	17	18	20	21	71	72	81	9
Correlation	-0.138	-0.195	-0.158	-0.161	-0.277	-0.531	-0.154	-0.362

Table 1: Correlation between power and temperature for entire dataset

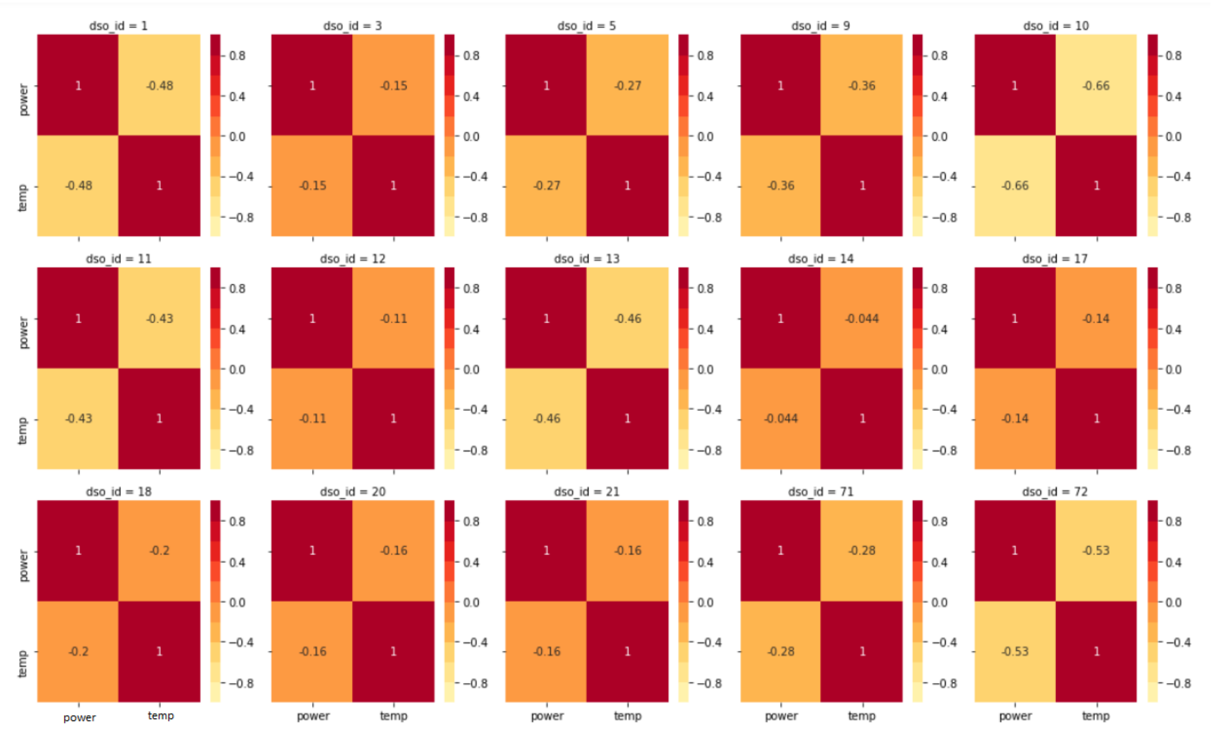


Figure 14: Correlation between power and temperature

2.5.2 Correlation for Highest Peak Values

We then run a correlation on the highest peak values since this portion of the data is our main focus. Based on the analysis performed on the power data to identify the highest peak values, we select scaled power values greater than 0.95. This produces the following results. Here, we still observe negative correlation between power and temperature for most of the DSOs. Although these correlations are not strong, we also observe a positive correlation for DSOs 1 and 9.

DSO ID	1	3	5	10	11	12	13	14
Correlation	0.075	-0.031	-0.553	N/A	-0.400	-0.039	-0.422	-0.434
DSO ID	17	18	20	21	71	72	81	9
Correlation	-0.429	-0.437	-0.267	-0.460	-0.219	-0.337	-0.078	0.600

Table 2: Correlation between power and temperature for entire dataset

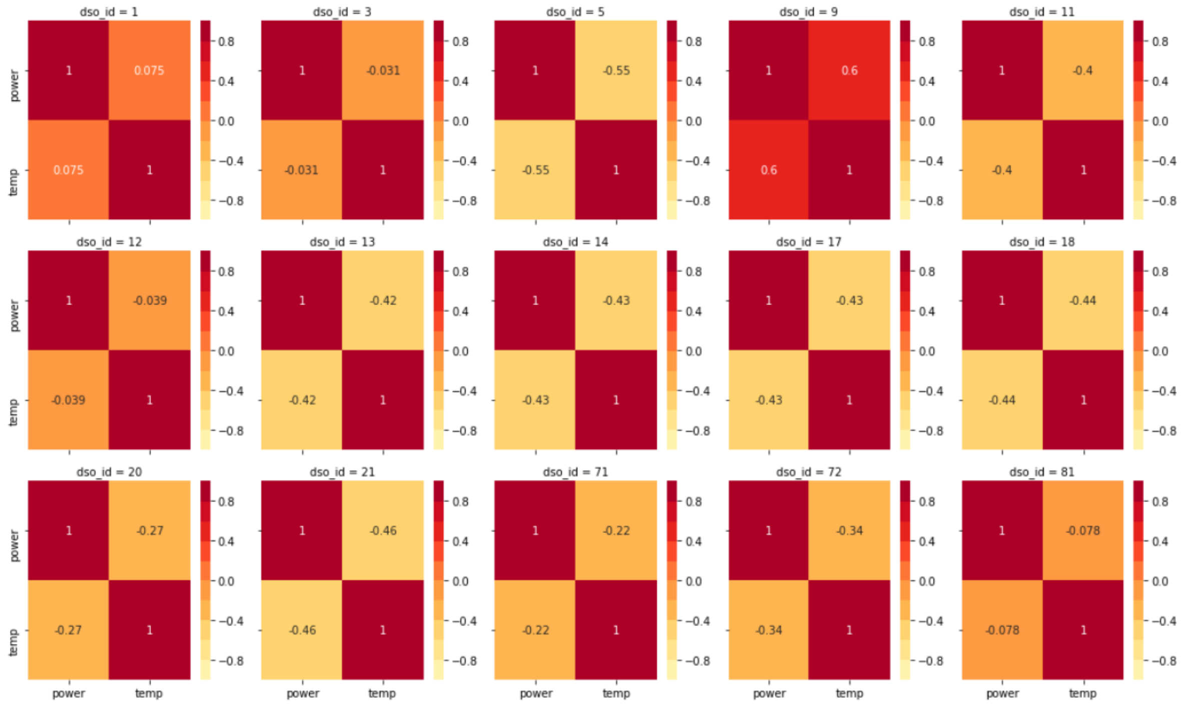


Figure 15: Correlation between power and temperature for highest peak data

2.5.3 Conclusion

Following the correlation run, we can see that the correlations are negative as expected for most of the DSOs. This result is to be expected, since when temperatures are low, we expect that more power is consumed. However, we do not see a very strong correlation here for most of the DSOs. The strength of the correlation being low implies that other factors or features besides temperature may contribute to the behavior of the power values; specifically with respect to how low or high the power consumed is.

3 Data Modeling

Chris, Constantin

The main task of the project is the prediction of the 15 minute time interval in the year when the grid load is maximal (i.e. the power consumption is highest). If we were able to identify it (or several candidates), we could discharge the batteries and earn the fee for stabilizing the electrical grid.

Naturally, the power consumption is maximal only once a year. This means that we only have one measurement per year which totals to 30 measurements for all locations. 30 data points are by far not enough to do any meaningful machine learning or statistical predictions and poses the biggest challenge of this project.

We can solve it in two different ways. In the simplest case, we just try to predict everything and hope that we will hit the peaks. In general, this will not work well since the peaks are artifacts happening rarely. We expect that we are only able to model measurements with medium power usage (compare figure 16) and that we will miss the peaks.

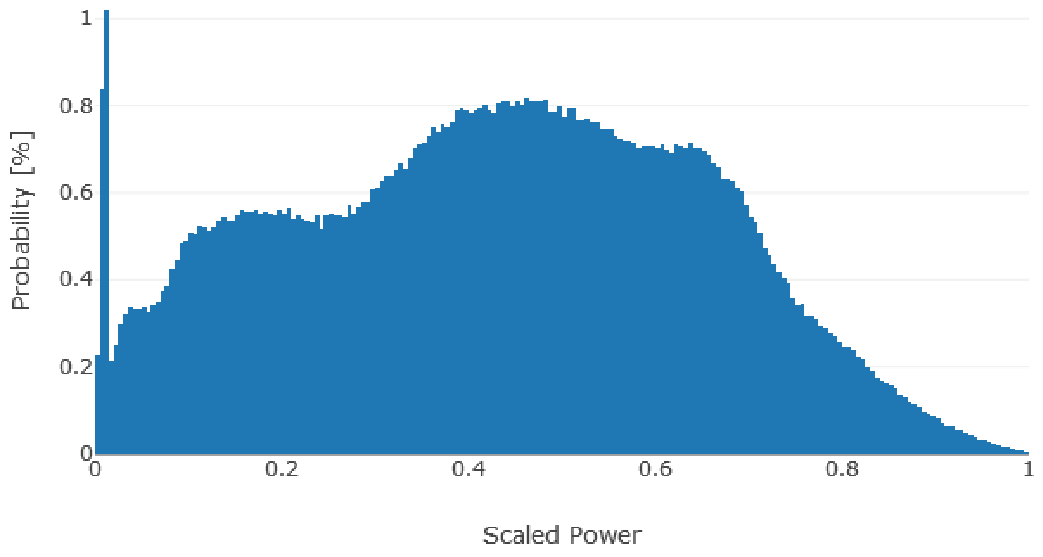


Figure 16: The distribution of power values.

However, the power consumption over the year possesses a certain continuity. If we know the power consumption now then it cannot change too much in 15 minutes. Such a model, namely the ARIMA model, is discussed in section 3.3.

For the other case, we try to enlarge our dataset. By choosing a parameter σ and trying to predict not only the peak but the power values higher than σ, we can increase our data set up to the factor of 100 (around 2000 sample for $\sigma = 0.95$). While the resulting data set is still comparatively small (compare figure 16), depending on the threshold σ, we still have enough data to work with.

This methodology is used in the statistical model (3.1), the support vector machine model and the neural network (both in 3.2).

3.1 Statistical Model

Con-
stantin

3.1.1 Overview

The statistical model is motivated by the results of the data exploration:

- In the winter, power consumption is higher than in the summer (see figure 11).
- In general, the peak load is around 18:00 o'clock (see figures 6 and 7).
- The power consumption is higher on weekdays (see figure 9).
- ...

We formalize these observations in form of a statistical model. We assign each 15 minute time slot of the year the likelihood of the maximum happening there, based on historic data. This idea guides the development of the following three statistical models. The first one, our initial idea, formulates the dependence between time, day of the month and month. The second one formulates the dependence between time, day of the week and week of the year. The last model enhances the second one by relaxing some assumptions we made.

3.1.2 Mathematical Background

Statistical Model I We observe that each 15 minute interval in the year can be uniquely specified using time, day (of the month) and month (t, d, m). We then calculate the probability that the interval (t, d, m) contains the peak load.
We start by counting how often each tuple (t, d, m) contained a peak load in the previous years and call the resulting matrix $\tilde{A}$. Rescaling $\tilde{A}$

$$A := \frac{\tilde{A}}{\sum_{t,d,m} \tilde{A}(t, d, m)}$$

yields a matrix A. A contains the relative occurrences of peaks per date which we can interpret as a probability distribution.
We use this result to approximate the probability distribution we initially wanted to calculate

$$P(\mathcal{P}(t, d, m) = 1) \approx A(t, d, m)$$

where $\mathcal{P}(t, d, m)$ denotes the power consumption at time (t, d, m).

This model has several flaws. By design, the matrix A has $35712 = 96 \cdot 31 \cdot 12$ entries (number of quarter hours per day, number of days in a month and number of months). For $\sigma = 1$ we would have 30 measurements filling this matrix. Therefore, A contains mostly zero entries and cannot be used to predict the peaks of new data.

Nonetheless, the idea is good, we just need another way to model $P(\mathcal{P}(t, d, m) = 1)$. Instead of predicting the peaks, we can try to predict when the load is high, i.e. higher than some threshold σ

$$P(\mathcal{P}(t, d, m) \geq \sigma)$$

Next we define the following random variables

$$\begin{aligned}
T_\sigma &:= \text{“The power at time } t \text{ is higher than } \sigma\text{”}\\
\tilde{D}_\sigma &:= \text{“The power at day (of month) } d \text{ is higher than } \sigma\text{”}\\
M_\sigma &:= \text{“The power in the month } m \text{ is higher than } \sigma\text{”}
\end{aligned}$$

with corresponding distributions $\rho_{T,\sigma}, \rho_{\tilde{D},\sigma}$ and $\rho_{M,\sigma}$. These distributions can be approximated with the distributions $\tilde{\rho}_{T,\sigma}, \tilde{\rho}_{\tilde{D},\sigma}$ and $\tilde{\rho}_{M,\sigma}$ calculated from the data (see figures 6 and 11).

Assuming independence, we can then calculate

$$\begin{aligned}
P(\mathcal{P}(t,d,m) \geq \sigma) &= P(\{T_\sigma = t\} \cap \{\tilde{D}_\sigma = d\} \cap \{M_\sigma = m\})\\
&= \rho_{T,\sigma}(t) \cdot \rho_{\tilde{D},\sigma}(d) \cdot \rho_{M,\sigma}(m)\\
&\approx \tilde{\rho}_{T,\sigma}(t) \cdot \tilde{\rho}_{\tilde{D},\sigma}(d) \cdot \tilde{\rho}_{M,\sigma}(m).
\end{aligned}$$

However, the Chi-Square test[3] for independence showed that day and month are not independent.[4] For this reason we use conditional probabilities

$$\begin{aligned}
P(\mathcal{P}(dt) \geq \sigma) &= P(\{T_\sigma = t\} \cap \{\tilde{D}_\sigma = d\} \cap \{M_\sigma = m\})\\
&= P(T_\sigma = t) \cdot P(M_\sigma = m) \cdot P(\tilde{D}_\sigma = d | M_\sigma = m)\\
&= \rho_{T,\sigma}(t) \cdot \rho_{\tilde{D}|M,\sigma}(d) \cdot \rho_{M,\sigma}(m)\\
&\approx \tilde{\rho}_{T,\sigma}(t) \cdot \tilde{\rho}_{\tilde{D}|M,\sigma}(d) \cdot \tilde{\rho}_{M,\sigma}(m)
\end{aligned}$$

Here, the conditional distributions $\tilde{\rho}_{\tilde{D}|M,\sigma}$ can easily be computed from the data.

Statistical Model II By construction, statistical model I can only model monthly behavior and misses the correlation between day of the week and peak load (see figure 9). For this reason, we created the statistical model II. Here we identify each 15 minute window of the year with the tupel time, weekday and week of the year (t, d, w). Then we proceed as in model I. We define the random variables

$$\begin{aligned}
T_\sigma &:= \text{“The power at time } t \text{ is higher than } \sigma\text{”}\\
D_\sigma &:= \text{“The power at day (of the week) } d \text{ is higher than } \sigma\text{”}\\
W_\sigma &:= \text{“The power in week } w \text{ is higher than } \sigma\text{”}
\end{aligned}$$

with corresponding distributions $\rho_{T,\sigma}, \rho_{D,\sigma}$ and $\rho_{W,\sigma}$. These are approximated by $\tilde{\rho}_{T,\sigma}, \tilde{\rho}_{D,\sigma}$ (see figure 9) and $\tilde{\rho}_{M,\sigma}$.

[3]The Chi-Square test requires that the test matrix has no zero entries. We achieve this by using three categories (January, February to November, December) as months.

[4]For $\sigma = 0.95$ and null hypothesis H_0: “day and month are independent” the test returns a p-value of $p \approx 10^{-100}$.

Again, assuming independence, we approximate

$$\begin{aligned} P(\mathcal{P}(dt) \geq \sigma) &= P(\{T_\sigma = t\} \cap \{D_\sigma = d\} \cap \{W_\sigma = w\}) \\ &= \rho_{T,\sigma}(t) \cdot \rho_{D,\sigma}(d) \cdot \rho_{W,\sigma}(w) \\ &\approx \tilde{\rho}_{T,\sigma}(t) \cdot \tilde{\rho}_{D,\sigma}(d) \cdot \tilde{\rho}_{W,\sigma}(w). \end{aligned}$$

Similarly to model I, weekday and week are not independent[5].

We decided that we don't fix this by introducing the conditional distribution $\rho_{D|W,\sigma}$ where each day in the week gets its own distribution. The reason is that we explicitly introduced weekdays to model the intra-weekly behavior assuming all weeks are basically the same. Instead we fix this fault in the statistical model III.

Statistical Model III The previous two statistical models have the flaw that we assume independence while in fact the data is not independent. In model I this was easily fixed, in model II we decided against an easy fix in favor of a good intuition of the model. This third and final statistical model aims to fix this while maintaining a good understanding of the model.
We have shown that the distributions $\rho_{D,\sigma}$ and $\rho_{W,\sigma}$ of D_σ and W_σ are not independent. Now we try to replace them with new distributions which are independent with respect to our data. We achieve this by minimizing the difference

$$\begin{aligned} &||P(\{D_\sigma = d\} \cap \{W_\sigma = w\}) - P(\{D_\sigma = d\}) \cdot P(\{W_\sigma = w\})||_2 = \\ =&||P(\{D_\sigma = d\} \cap \{W_\sigma = w\}) - \rho_{D,\sigma}(d) \cdot \rho_{W,\sigma}(w)||_2. \end{aligned}$$

The difference consists of the product of the distributions $\rho_{D,\sigma}(d)$ and $\rho_{W,\sigma}(w)$ subtracted from the joint probability of having a peak at day d and week w. If it was 0, we would have independence.
We approximate the joint distribution with a matrix $A_{w,d}$ containing the counts of how often the power at day d and week w is high (A is a rescaled version of the matrix in figure 17).
Rewriting using this approximation yields

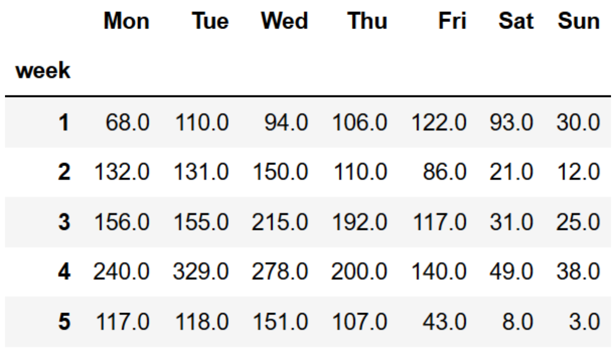
$$\sum_{d,w} (A_{w,d} - \rho_{D,\sigma}(d) \cdot \rho_{W,\sigma}(w))^2.$$

By minimizing this expression

$$\min_{\rho_{D,\sigma}, \rho_{W,\sigma}} \sum_{d,w} (A_{w,d} - \rho_{D,\sigma}(d) \cdot \rho_{W,\sigma}(w))^2,$$

we obtain two distributions $\rho_{D,\sigma}$ and $\rho_{W,\sigma}$ which are as independent as possible with regards to our data.

[5] For $\sigma = 0.95$ the Chi-Square test for independence with null hypothesis H_0: "weekday and week are independent" returns a p-value of $p \approx 10^{-18}$. We used the categories (Mon, Tue, Wed, Thu, Fri) for the days and (1, 2, 3, 4, 5, 6, 7 to 48, 49, 50, 51, 52) for the weeks.



week	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	68.0	110.0	94.0	106.0	122.0	93.0	30.0
2	132.0	131.0	150.0	110.0	86.0	21.0	12.0
3	156.0	155.0	215.0	192.0	117.0	31.0	25.0
4	240.0	329.0	278.0	200.0	140.0	49.0	38.0
5	117.0	118.0	151.0	107.0	43.0	8.0	3.0

Figure 17: Count of 15 minutes intervals with load higher than 95% of the peak load grouped by day and week

Since $\rho_{D,\sigma}$ and $\rho_{W,\sigma}$ are probability distributions, we can replace them with vectors x and y only containing non-negative values which sum up to 1. The result is the following final problem

$$\begin{aligned}
\underset{x,y}{\text{minimize}} & \sum_{d,w}(A_{w,d} - x_d y_w)^2 \\
\text{subject to } & x_d \geq 0 \quad \forall d \\
& y_w \geq 0 \quad \forall w \\
& \sum_d x_d = 1 \\
& \sum_w y_w = 1.
\end{aligned}$$

The solution of this minimization yields the optimal choice for $\rho_{D,\sigma}$ and $\rho_{W,\sigma}$. Unfortunately, the problem itself is not convex. Therefore, common minimization algorithms fail to find the best solution. The same is true in our case. Depending on the initial distributions for $\tilde{\rho}_{T,\sigma}$ and $\tilde{\rho}_{D,\sigma}$, we get different solutions. In any case we don't get better than the distributions used in Statistical Model II and therefore cannot find solutions which are "as independent as possible".

In the end, we decided that we will replace the distribution $\rho_{D,\sigma}$ with the conditional distribution $\rho_{D|W,\sigma}$ where each week gets its own distribution of weekdays calculated from the data.

3.1.3 Results

In this last part we want to evaluate the performance of the statistical models. This will happen in two steps. We start with a theoretical analysis and show that these models should perform well. In the second part we show that indeed they perform very well.

Theoretical Analysis By definition, the model always returns a probability distribution where each 15 minutes interval of the year gets its own probability. Using this probability we can calculate the cumulative probability of hitting the peak load after N

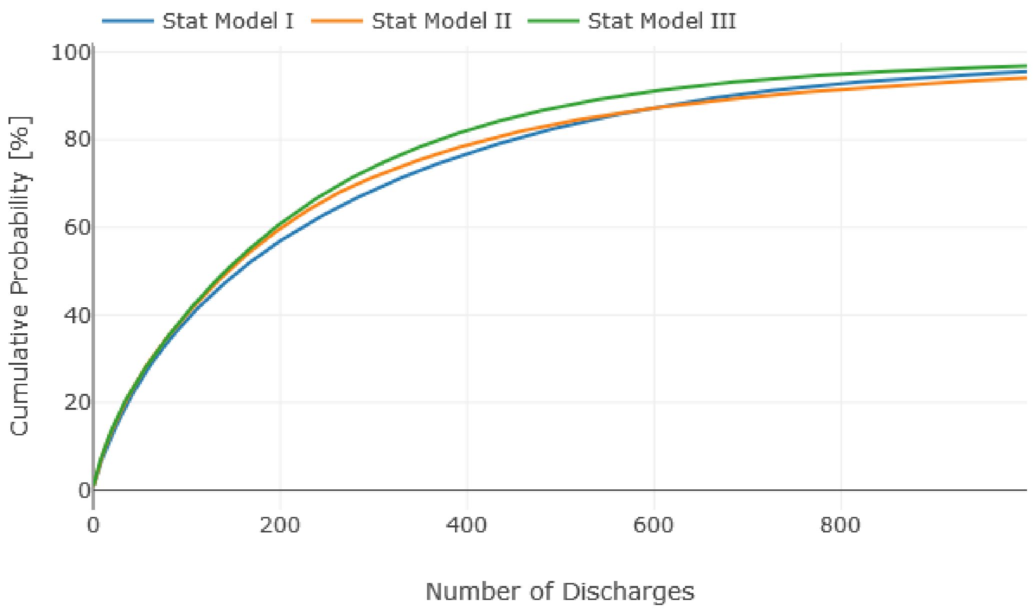


Figure 18: The cumulative probability of hitting the peak load for $\sigma = 0.95$.

discharges of the batteries. As can be seen in figure 18, Statistical Model III seems to perform slightly better than the other two models for a high number of discharges.
However, this graph doesn't show us whether these probabilities are good enough, i.e. if this model would be profitable. In reality, each discharge costs money (about 25 ct/kW) while discharging at the peak load gains money (about 90 €/kW). Assuming now that at each location we have batteries with a combined power of 1 MW we can calculate the expected gain. Figure 19 shows this graph. As you can see, we expect that the maximum profitability is reached around 75 discharges. There, we expect a gain of around 11 000 € per location.

Practical Analysis The theoretical analysis has to be taken with a grain of salt since all these models assume independence which is not necessarily given. For this reason, we evaluate how these models perform on the real data.
We evaluated these models using 10-fold cross-validation. As you can see in figure 20, for large σ, model I and III don't generalize to new data and fail to identify the maximum (more then 1 000 discharges needed to identify 50% of the peaks correctly). For $\sigma = 0.99$ and $\sigma = 0.95$, the models reach peak performance. Further decreasing σ reduces the accuracy.
For $\sigma = 0.99$, we now have a great result. Using around 100 discharges allows model II and III to identify more than 50% od the peak loads in the year. Using the cost and gain from above (250 €/MW for discharging a battery, 90 000 €/MW for discharging at the peak load), we can now calculate the effective gain using the statistical model with

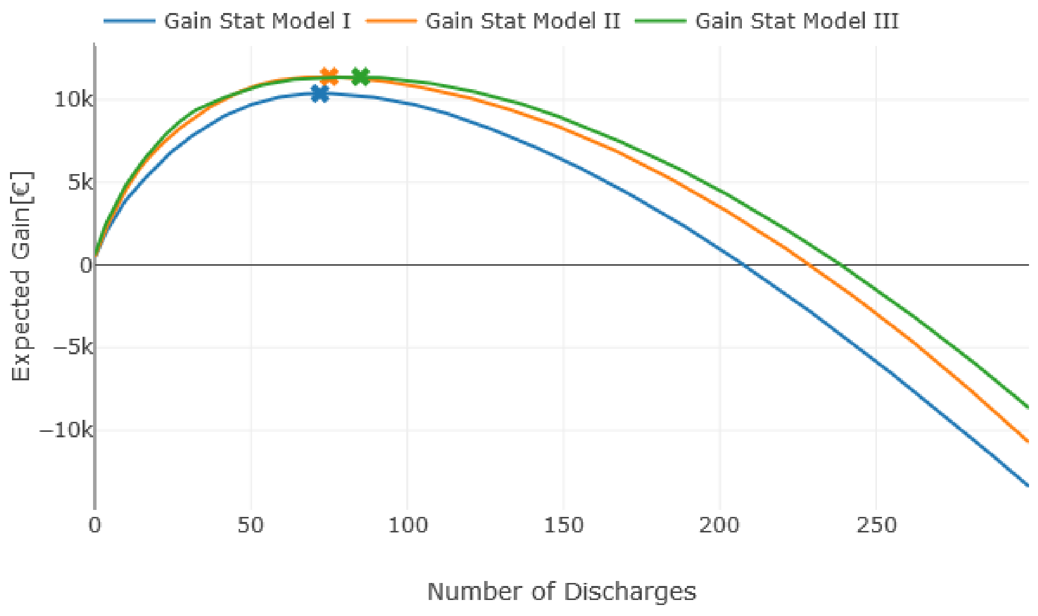


Figure 19: The expected gain for $\sigma = 0.95$. The maximal gains are marked with a cross.

$\sigma = 0.99$ and 100 discharges:

$$0.5 \cdot 90\,000\,€ - 100 \cdot 250\,€ = \underline{\underline{20\,000\,€}}$$

Applying this algorithm would have resulted in an average revenue of 20 000 € per MW and location.

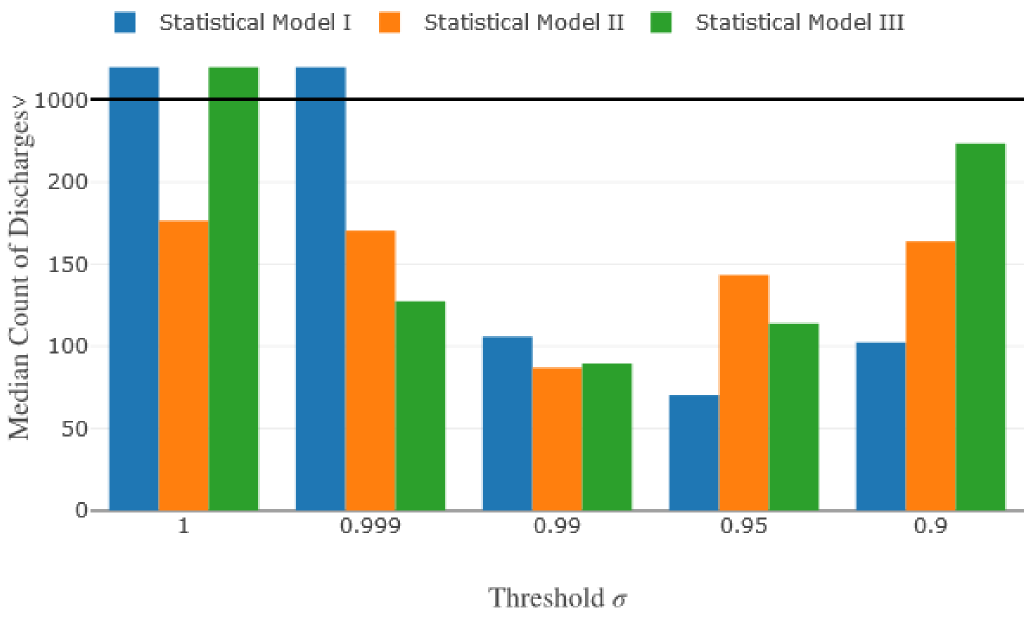


Figure 20: The median count of discharges need for hitting the peak load.

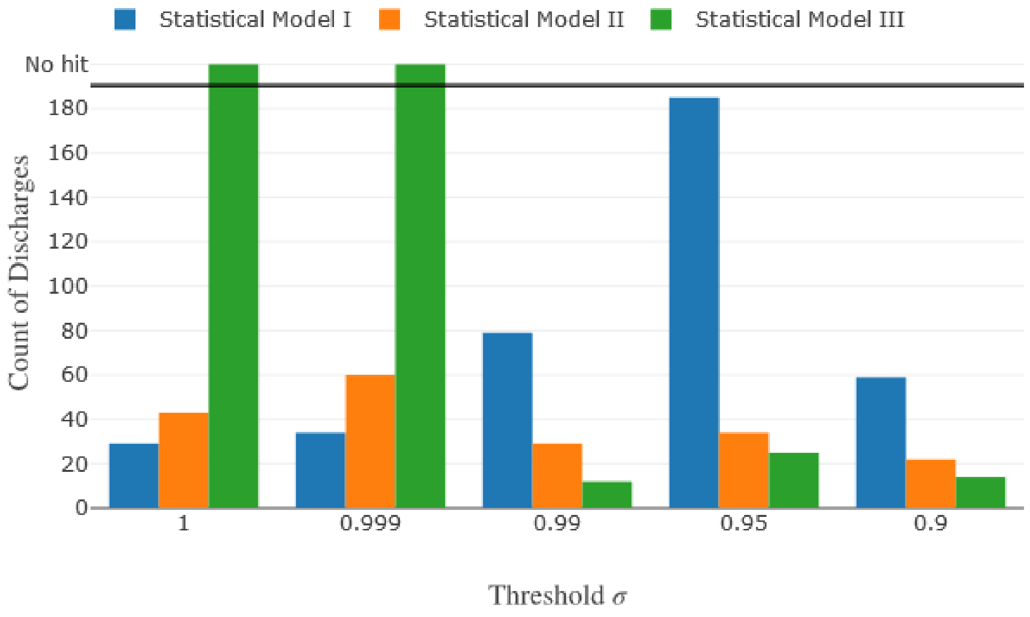


Figure 21: The number of discharges needed to hit the peak load in Berlin 2015.

3.2 Support Vector Machine and Neural Network

Constantin

3.2.1 Overview

In this chapter, we discuss how we apply support vector machines and neural networks to predict the peak load.
In general, we try to determine whether the next 15 minute interval has a high or a low power consumption. As before, high means higher then some threshold σ.

3.2.2 Mathematical Background

Data Preparation We start with preparing the data for both models. In the SQL database, the data is available in form of a list containing date, temperature and power level. In this form, the time, day and week of a measurement is indirectly available and cannot be understood by our models.
Therefore, we split the date information into several important aspects, namely month, day of month, day of week, week of year, hour and minute. These are then one-hot encoded which allows both models to learn the periodicity of the problem (i.e. "the power consumption is high at 18:00").
The only exemption is the temperature data. Since the temperature is continuous, in contrast to the aforementioned features, we don't change it at all.
As a result, our data now has around 50 different features.

Additionally, the data set available is highly unbalanced. As we have already discussed, depending on the choice of σ, we have around 2 000 high power and around 2 000 000 low power measurements (for $\sigma = 0.95$). Therefore, in the training step of the models we randomly sub-sample the low power values and get a training data set with ratio 5:1 (low to high power data). Additionally, we weight the low power and high power class in a ration of 1:5, achieving similar sized data.

Support Vector Machine The support vector machine tries to split the data into two parts separated by a (hyper-)plane. In the simplest case, one might say: "if the temperature is below 0 °C, we conclude that the power value is high and discharge our battery".
In practical application, a support vector machine is much more potent and can use the so called kernels which generalize "normal" planes to ones which aren't necessarily straight (for example balls).
In our case, we use a Gaussian kernel with γ equal to

$$\gamma = \frac{1}{\text{“number of features”} \cdot \text{“standard deviation of measurements”}}.$$

and the penalty $C = 1$.[6]

[6]Unfortunately there wasn't enough time to properly train these parameters. For this reason this model has to be understood as a proof of concept and not as a finished product.

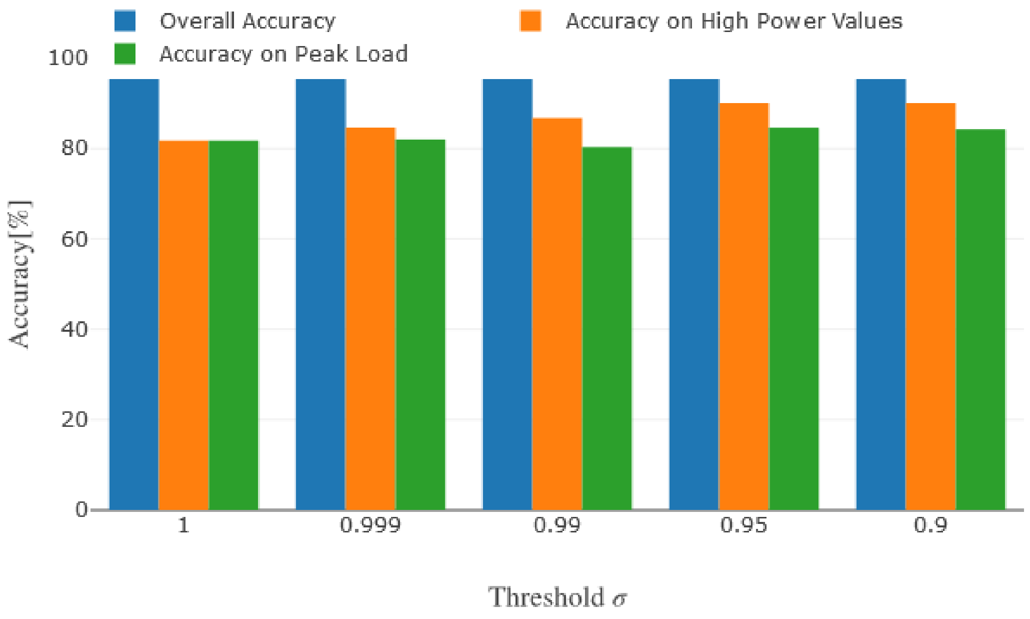


Figure 22: The accuracy of the SVM evaluated for different σ.

Neural Network Additionally to the support vector machine, we implement a neural network to predict whether the power usage at a certain time and temperature is high. We use a 3 layer feed forward neural network. The first layer has 48 inputs (temperature and one-hot encoded date information) and 12 nodes. The second layer has 8 nodes and the last third layer has one output node.
In the first two layers, we use the "relu" activate function f

$$f(x) = \max(0, x).$$

In the last layer, we use a sigmoid function where we interpret values larger then 0.5 as 1 and smaller values as 0. The neural net is then trained using the stochastic gradient descend with the binary cross entropy as error function.

3.2.3 Results

In the end, both methods perform very well, without any training of their respective hyper-parameters. As one can see in figure 22 and figure 23, both support vector machine and neural network have a very high overall accuracy. However, this was to be expected since most measurements are zero anyways and therefore should be predicted easily.
Thus, this number is not really representative of the actual performance of our models. After all, we are only interested in the peaks. For this reason, we additionally analyze the accuracy on the high power values and the accuracy on the peak loads. In general, both models are able to identify times with power values higher than σ with a high accuracy, slightly increasing for smaller σ.

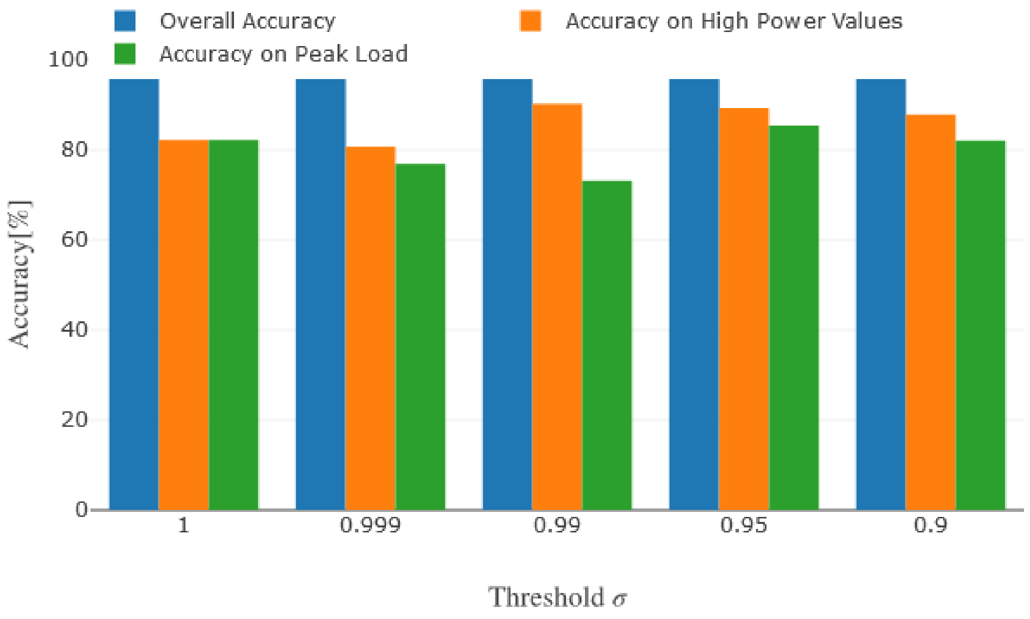


Figure 23: The accuracy of the neural network evaluated for different σ.

The accuracy of the support vector machine for identifying the actual peaks is comparable as well, in general above the 80% mark.
The neural network performs slightly worse and identifies the actual peak loads only with an accuracy of 75% for $\sigma = 0.99$.

In our use-case, we not only need a high accuracy but also a sufficiently small number of discharges. After all, one has to pay per discharge. As expected, the number of predicted high power values increases with smaller σ (figure 24 and 25). This dependence is nearly exponential which is why a threshold not much larger then $\sigma = 0.99$ is a must. Otherwise, the discharges become to expensive.

We conclude this chapter with an estimate of the actual gain we could have expected. As before, a discharge costs 250 € per MW and results in 90 000 € per MW if one hits the peak.
For $\sigma = 0.99$ this results in a gain per location and MW power of

$$90\,000\,€ * 0.8 - 100 \cdot 250\,€ = 47\,000\,€$$

using the support vector machine and 100 discharges (which are estimate very conservatively, see figure 24).
For the neural network we achieve a comparable performence with a gain of

$$90\,000\,€ * 0.7 - 100 \cdot 250\,€ = 38\,000\,€$$

per location and MW power. Again we use 100 discharges.

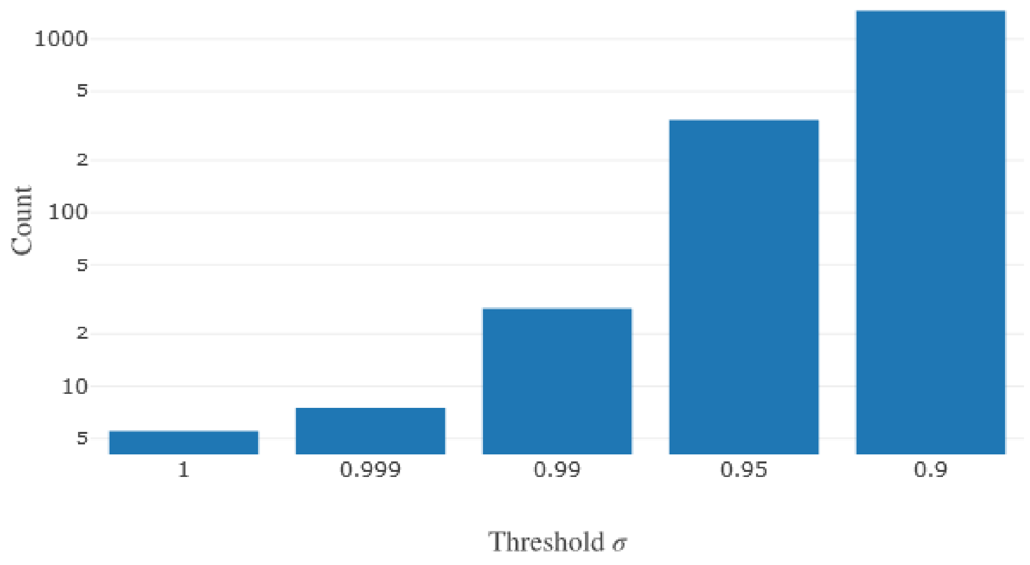


Figure 24: The number of predicted peak loads of the SVM per threshold σ.

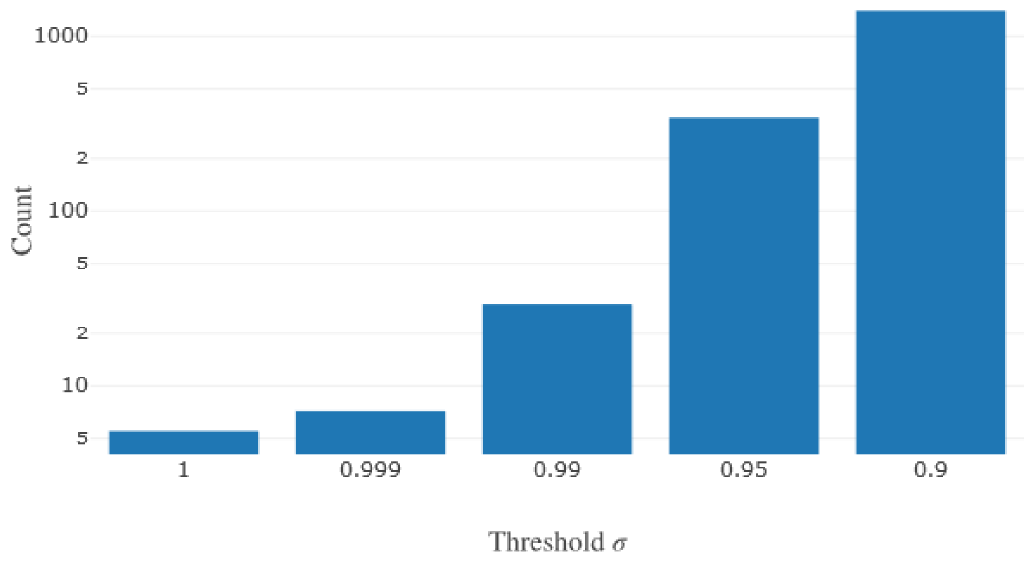


Figure 25: The number of predicted peak loads of the neural network per threshold σ.

3.3 ARIMA Model

Chris

3.3.1 Overview

One way to predict a time series is the so-called Autoregressive Integrated Moving Average (ARIMA) model (see [**10**]). It can be used for stationary time series data in order to predict future points in time. The ARIMA model tries to predict the whole time series. Before implementing it, we had the thought that it might fit the time series well but miss the peaks. Having this in mind, we present first the mathematical background of the ARIMA model and afterwards results of our implementation.

3.3.2 Mathematical Background

The ARIMA model is a combination of an Autoregressive (AR) model (see [**11**]) and a Moving Average (MA) model (see [**12**]).

The Autoregressive model specifies that the output variable depends linearly on its own previous values and on a stochastic term:

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t,$$

where ϕ_1,...,ϕ_p are the model parameters, p is called the order of the AR model, c is a constant and ϵ_t is white noise.

The Moving Average model specifies that the output variable depends linearly on the current and various past values of a stochastic term:

$$X_t = \mu + \epsilon_t + \theta_1\epsilon_{t-1} + ... + \theta_q\epsilon_{t-q},$$

where μ is the mean of the series, $\theta_1, ..., \theta_q$ are the parameters of the model, the value of q is called the order of the MA model and the $\epsilon_t, ..., \epsilon_{t-q}$ are white noise error terms.

Given a time series X_t, where t is an integer and X_t is a real number, an ARMA(p',q) model is given by

$$X_t - \alpha_1 X_{t-1} - ... - \alpha_{p'} X_{t-p'} = \epsilon_t + \theta_1\epsilon_{t-1} + ... + \theta_q\epsilon_{t-q}$$

or equivalently

$$\left(1 - \sum_{i=1}^{p'} \alpha_i L^i\right) X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \epsilon_t.$$

Here, L is the shift operator, the α_i are the parameters of the MA part of the model, q is the order of the MA part of the model, the θ_i are the parameters of the AR part of the model, p' is the order of the AR part of the model and the ϵ_i are white niose terms, exactly as above.
If the polynomial on the left side has a unique root of multiplicity d, it can be written as

$$\left(1 - \sum_{i=1}^{p'} \alpha_i L^i\right) = \left(1 - \sum_{i=1}^{p'-d} \phi_i L^i\right) (1 - L)^d.$$

An ARIMA(p,d,q) process expresses this polynomial factoriaztion property with $p := p'-d$ and is given by

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right) (1-L)^d X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \epsilon_t.$$

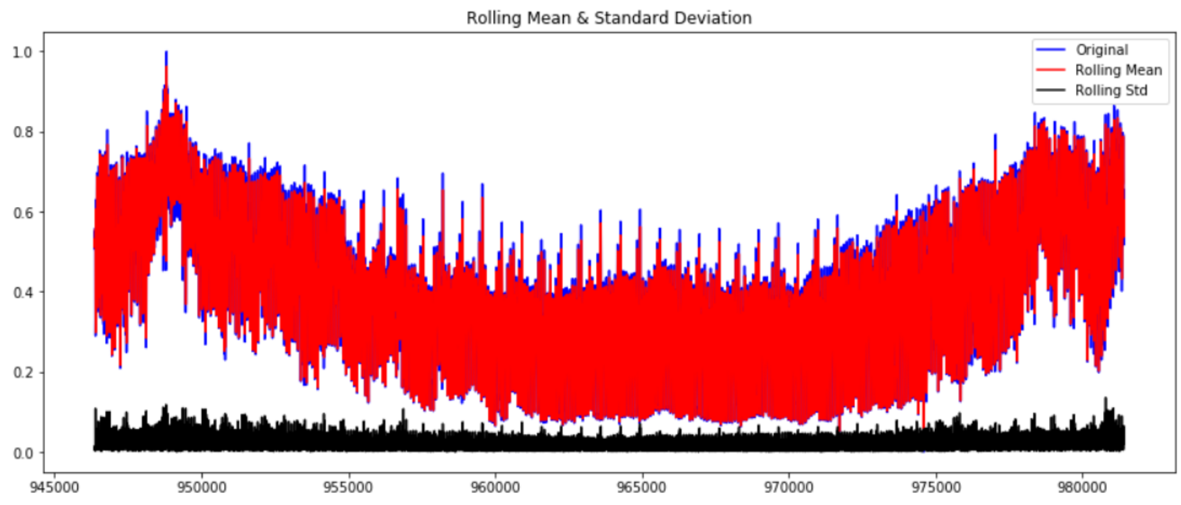
The ARIMA model is integrated into Python as the function statsmodels.tsa.arima_model-.ARIMA(p,d,q). The challenge while implementing this model is to find a meaningful stationary representation of the data. Furthermore, the number of AR parameters p, differences d and MA parameters q need to be chosen. Then, the ARIMA model can be applied on a training data set, e.g. a previous year. Afterwards, we can predict values for the upcoming year and compare them with the actual values that occurred in order to find out how good the ARIMA model performed.

3.3.3 Results

In the following, the implementation of the ARIMA model will be explained using data of the DSO "Avacon Netz GmbH" for the years 2014 and 2015 in the region of Sachsen-Anhalt. While creating the model, we followed the step-by-step explanation of [**13**]. Nevertheless, the author of article [**13**] trains his model on the same data as he tests it on. Therefore, we had to adapt the algorithm so that everything is correct. For doing so, we investigated article [**14**]. After loading all the needed Python packages and the data from the database, we applied the Dicky-Fuller test to the training data, i.e. data from the year 2014, in order to see whether it is stationary or not.

In figure 26, you can see the results of the test. The null-hypothesis of the Dicky Fuller test is that the time series is not stationary. If the value of the test statistic is lower than the critical value (here, we plotted the critical value for $\alpha = 1\%, \alpha = 2\%$ and $\alpha = 5\%$), the null-hypothesis can be rejected and we can assume that the time series is stationary. In this case, the test tells us that the time series is not stationary since the value of the test statistic is bigger than the critical value for $\alpha = 5\%$. Optically, this makes sense since the rolling mean (in red) and the standard deviation (in black) clearly change over time but rolling mean and standard deviation of a stationary process do not vary significantly over time.

Because of that, we need to think of another parameter we can apply the ARIMA model prediction to. After various considerations, we decided to split the data into the into two parts: the average of the last two values and the difference between the actual value and this average. The new value, called moving_avg_training_diff, will be the parameter we apply the ARIMA model to. The disadvantage of this method is that we need live data in order to do the predictions since we need to build the average of the current values. We will comment in the results section as to how one could get live data. In order to apply the ARIMA model to the moving_avg_training_diff time series, we first test



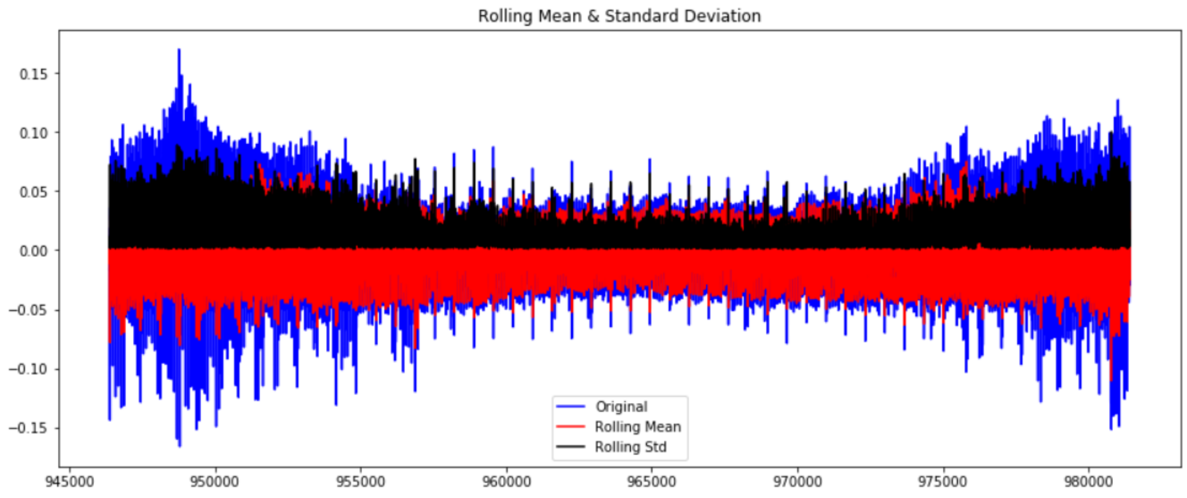
```
Results of Dickey-Fuller Test:
Test Statistic                 -1.154911e+01
p-value                         3.482386e-21
#Lags Used                      5.200000e+01
Number of Observations Used     3.498700e+04
Critical Value (1%)            -3.430537e+00
Critical Value (5%)            -2.861623e+00
Critical Value (10%)           -2.566814e+00
dtype: float64
```

Figure 26: Dicky-Fuller test for the original time series

whether it is stationary. In figure 27, you can see the results of the Dicky-Fuller test. The moving_avg_training_diff time series is printed in blue, the rolling mean in red and the standard deviation in black. Clearly, the value of the test statistic is below the 1% critical value, so we can reject the null-hypothesis and assume from now on that this time series is stationary.

In order to apply the ARIMA model, we now need to identify the right parameters p, the order of the AR model, and q, the order of the MA model. In order to calculate q, one plots the so-called Autocorrelation function. This function is a measure of the correlation between the time series and a lagged version of itself. The Autocorrelation function of our modified time series moving_avg_training_diff can be seen in figure 28. On the x-axis, we have the number of 15-minute time slots. The two dotted, grey lines on either side of 0 are confidence intervals which can be used to determine q. We choose q as the value on the x-axis where the Autocorrelation function crosses the upper grey line for the first time. In our case, this results in $q = 8$.

In order to determine p, we plot the Partial Autocorrelation function, see figure 29. This function measures the correlation between a time series with a lagged version of itself but after eliminating the variations already explained by the previous comparisons. For example, at leg 3, the function will explain the correlation but remove the affects already



```
Results of Dickey-Fuller Test:
Test Statistic                  -35.329808
p-value                           0.000000
#Lags Used                       52.000000
Number of Observations Used   34985.000000
Critical Value (1%)              -3.430537
Critical Value (5%)              -2.861623
Critical Value (10%)             -2.566814
dtype: float64
```

Figure 27: Dicky-Fuller test for the moving_avg_diff time series

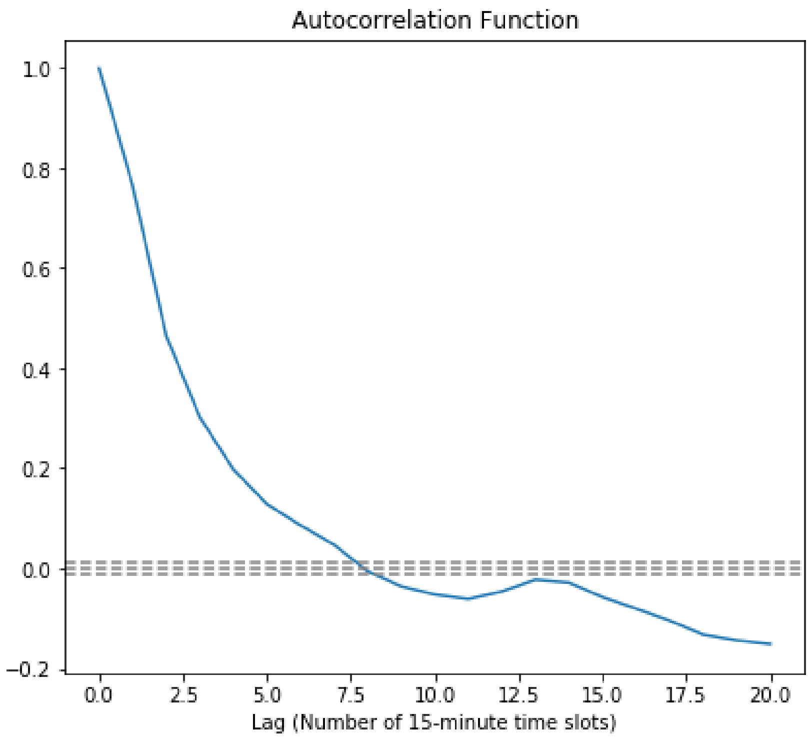


Figure 28: Autocorrelation Function

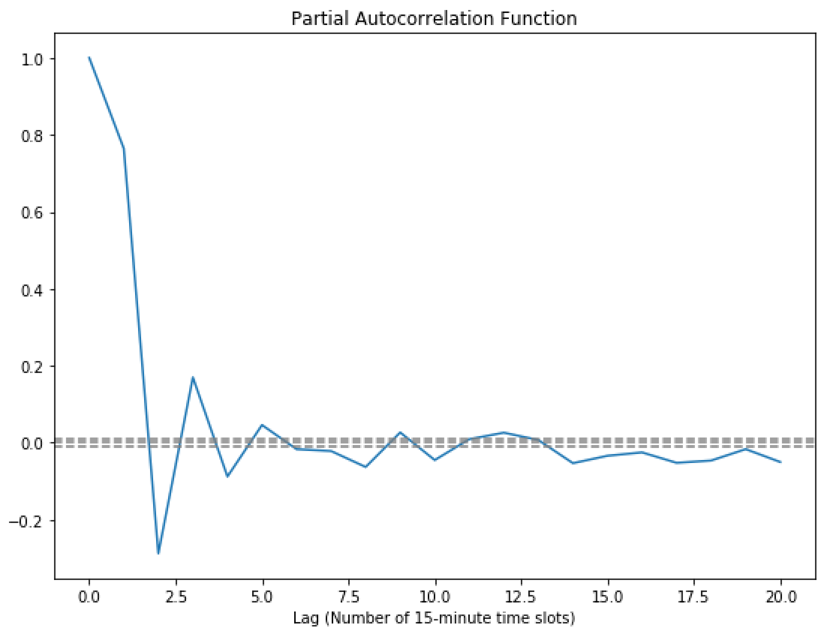


Figure 29: Partial Autocorrelation Function

explained by legs 1 and 2. Again, we see the number of 15-minute time slots on the x-axis. Exactly as before, p is the first intersection of the Partial Autocorrealtion function and the upper grey line representing the confidence interval. Here, we can see that $p = 2$.

After having identified the values for p and q, we can choose d, the degree of differencing, meaning the number of times the data had past values subtracted. In literature, d is typically chosen as 1 and we do so, too. Next, we can apply the ARIMA model to the moving_avg_training_diff time series. You can see the results in figure 30. In blue, we see the training time series and in red the trained estimation. The results do not seem very promising. The prediction models the overall trend but especially the outliers, in which we are particularly interested, are not captured well. Also trying various other parameters for the number of 15-minute time slots used for calculating the rolling mean, or d, did not enhance the prediction.

Nevertheless, let's analyze the results. In order to calculate the prediction for our test time series, we simply added the results of the ARIMA prediction to the moving average of the last two values of the test time series. Like this, we got the series shown in orange in figure 31. It matches the real time series, shown in blue, quite well. The mean squared error between the prediction and the actual values is 0.126%. In order to see what is really going on in detail, we zoomed in to three days in January and analyzed what is happening. You can see the results in figure 32. In general, we seem to underestimate the values. This wouldn't be a problem though because if we underestimate all values, we could simply take the 15-minute time slots of the highest x% of the prediction in order

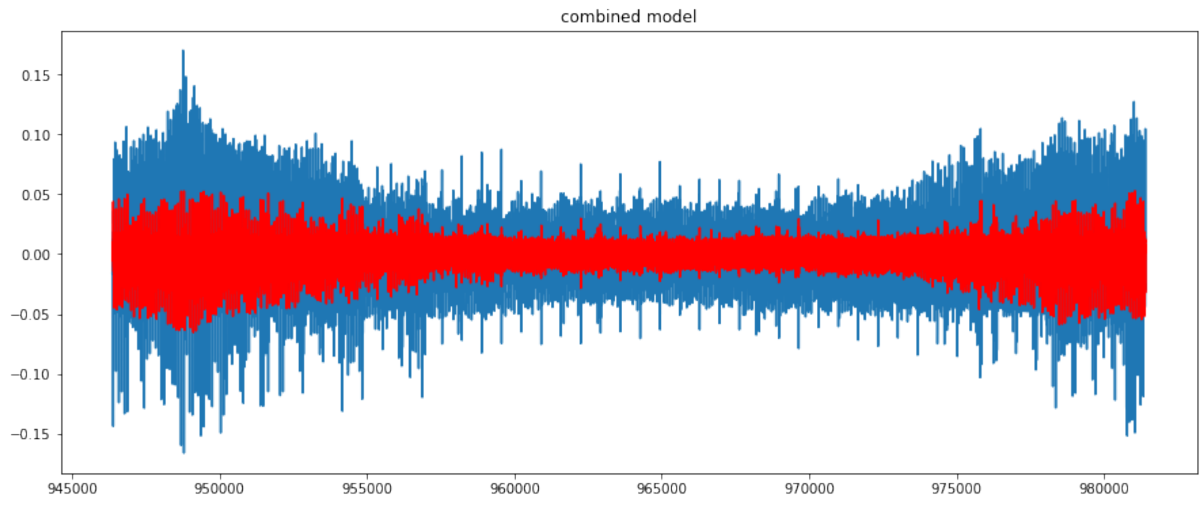


Figure 30: ARIMA prediction

to estimate the 15-minute time slots of the highest x% of the real series. But if we do so and compare the results with each other, we see an interesting phenomenon. For the real data, 42 time slots have a value higher than 90% of the highest value of the year. In our prediction, 69 values are higher than 90% of the highest predicted value. But if we compare the time slots we get with each other, only 29 are the same. This means that our prediction actually misses more than 13 15-minute time intervals. This can be explained when looking at figure 32 a bit closer. We see that in general, the orange curve is shifted just a bit to the right compared to the blue one. This means that it has higher values when the average of the previous values is high, so clearly, we are predicting high values when there has just been high values. But this means that we're missing high values that happen when the predecessor have not been high. This is definitely not optimal.

Nevertheless, we thought of a way to get candidates for the maximal value using this model and assuming that we actually have live data. The algorithm we developed looks as follows:

```
**Initialize:** Set $i = 0$, $current\_max = 0.6 * previous\_year\_max$, $values = [\,]$ and choose the threshold $T$ (e.g. $T = 0.9$)
**for** *i in range(first index of Prediction, last index of Prediction):* **do**
    **if** *Prediction[i]>current_max * T* **then**
        $current\_max = max(current\_max, Prediction[i])$;
        $values.append(i)$;
    **end**
**end**
```

Algorithm 1: Using the ARIMA model to predict the peak load

In the initialization phase, we set the *current_max* to 0.6 times the maximum of the previous year. This was done in order to not start with 0 because otherwise, we collect many

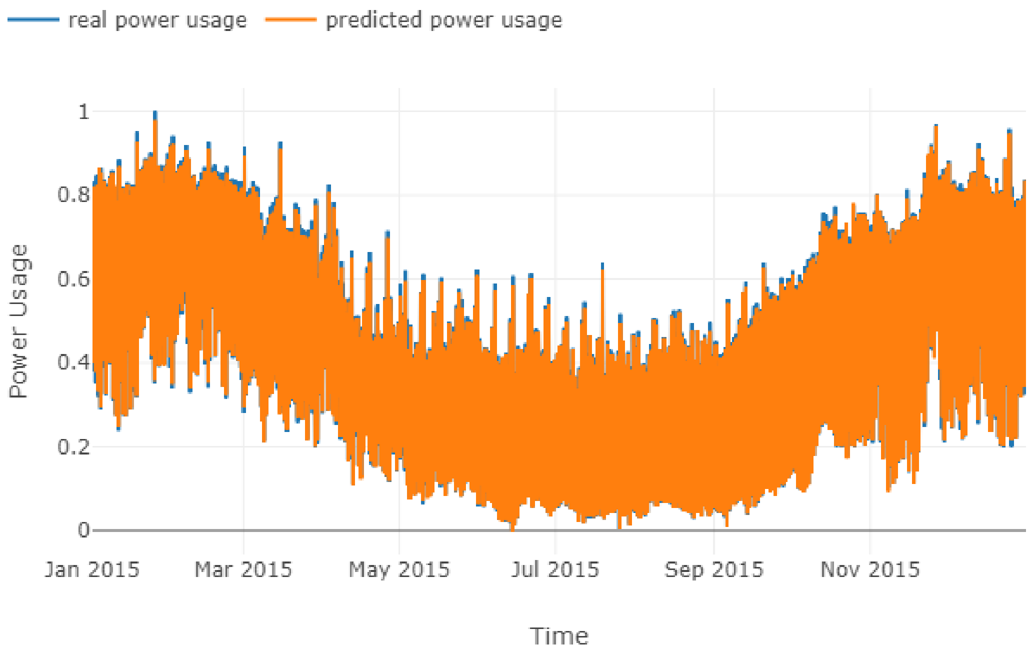


Figure 31: Results

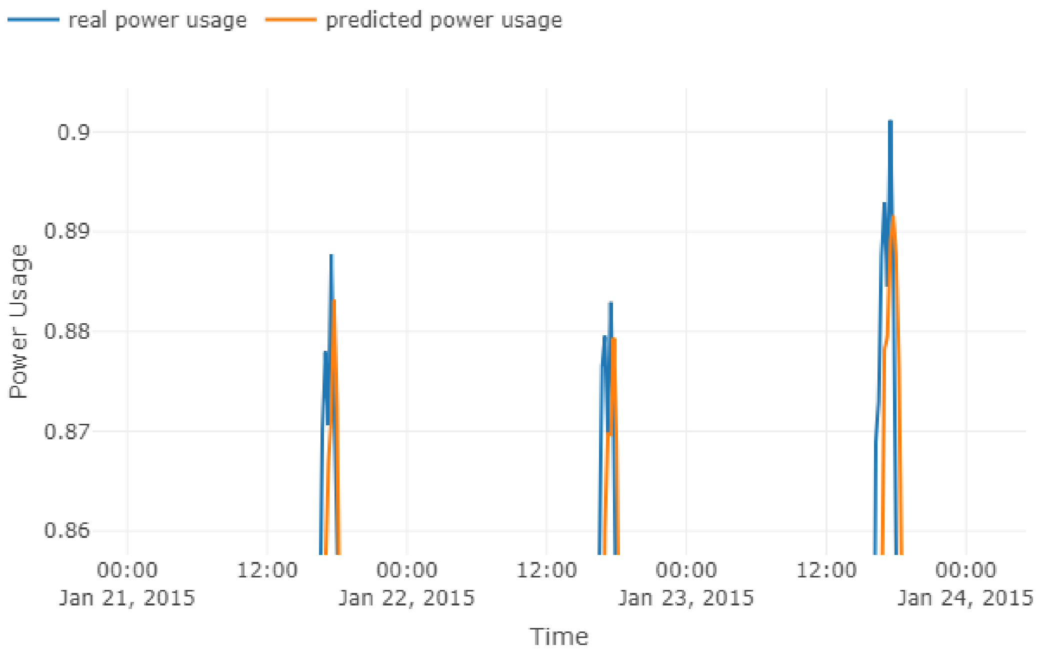


Figure 32: Results, zoomed in

values at the beginning of the year that do not play a role for the peak. Nevertheless, we didn't want to set *current_max* too high in the initialization step in order to be sure that it is not higher than the peak of the new year. When performing this algorithm, we get a couple of candidates for the peak. The number of these candidates obviously varies with the choice of T. In our example, if we choose $T = 0.9$, we get 270 candidates and actually hit the maximum. This results in total cost of $270 * 250€/MW = 67.500€/MW$ as the payment for 15 minutes of discharge is 250€/MW and earnings of 90.000€/MW as this is the value we get for hitting the maximum. In total, we have a profit of 22.500€/MW in this example. If we set $T = 1$, we only get 23 values but still hit the maximum. This results in a profit of 90.000€/MW - 23*250€/MW = 84.250€/MW. In figure 33, you can see the results for all DSOs which have published more than one year of data, so that we can apply the ARIMA model and the average over all of them. The DSO with ID 72 corresponds to the DSO we've been applying the ARIMA model to here. We see that we hit the maximum with $T = 1$ in three out of four cases. For a smaller T, we always hit the maximum. Furthermore, you can also see the gain we would have made when applying our algorithm for the different values of T. We plotted the gain on the y-axis and the average number of 15-minute time-slots we fed electricity into the grid on the x-axis in figure 34. The gain is biggest if we choose $T = 0.99$, so this would be our best choice for future years.

At the end of the evaluation of the ARIMA model, let us shortly comment on the availability of live data. By law, DSOs are only obliged to publish the power data of the past year at the beginning of the new one. Therefore, we do not have DSO power data available in real time. Nevertheless, TSOs need to publish their data in real time. Therefore, one could download this data and investigate whether there is a correlation between TSO and DSO data. If so, one could apply the ARIMA model for TSO data, get the time slots that are predicted to be a maximum and then feed in electricity into the grid at these times. Another way of getting live data would be taking battery data into account. If this data was correlated to DSO data, we could again use it to predict peaks. Unfortunately, we did not have enough time left at the end of our project in order to investigate the topic of getting live data properly. Instead, we concentrated on finishing our various models and leave it for a future project to find ways of getting live data that can be used as an approximation of DSO power data.

DSO ID		T				
		1	0.99	0.95	0.9	0.85
	# of feed-ins	17	21	167	579	1194
71	maximum hit?	1	1	1	1	1
	gain/loss	85750	84750	48250	-54750	-208500
	# of feed-ins	23	33	76	270	600
72	maximum hit?	1	1	1	1	1
	gain/loss	84250	81750	71000	22500	-60000
	# of feed-ins	38	61	162	404	794
12	maximum hit?	1	1	1	1	1
	gain/loss	80500	74750	49500	-11000	-108500
	# of feed-ins	40	71	356	914	1945
14	maximum hit?	0	1	1	1	1
	gain/loss	-10000	72250	1000	-138500	-396250
	# of feed-ins	**29,5**	**46,5**	**190,25**	**541,75**	**1133,25**
Average	**maximum hit?**	**0,75**	**1**	**1**	**1**	**1**
	gain/loss	**60125**	**78375**	**42437,5**	**-45437,5**	**-193312,5**

Figure 33: Results of applying algorithm 1 to all DSOs with more than 2 years of data

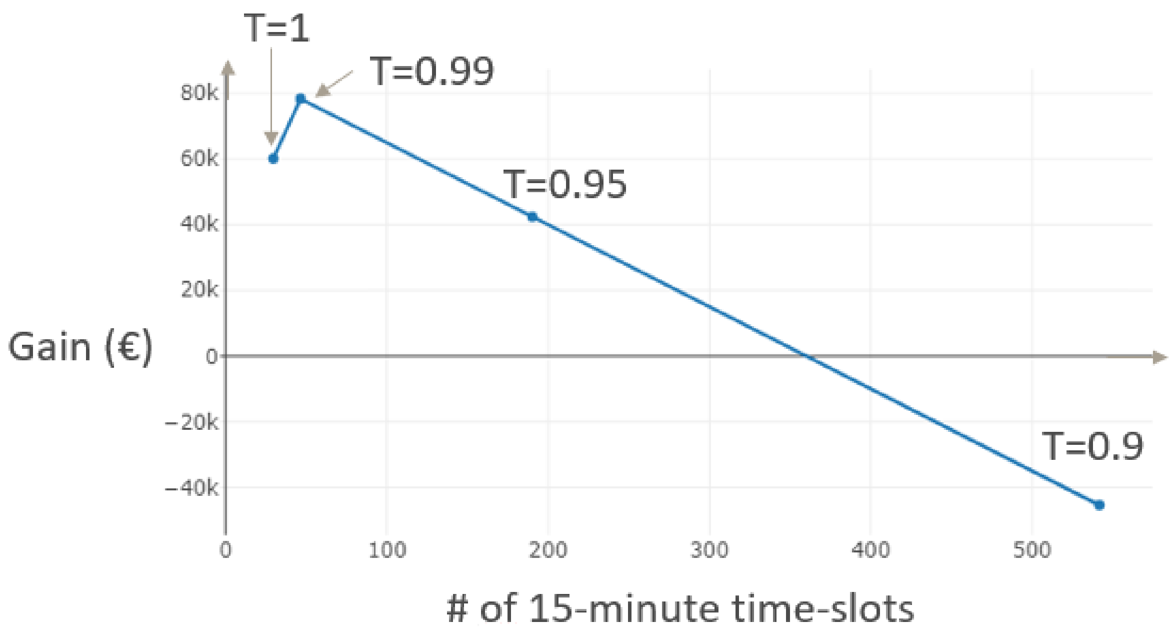


Figure 34: Average gain of the ARIMA model used as specified in algorithm 1

4 Web Application

Ayishetu

4.1 Overview

The goal of the web application is to provide sonnen with a user interface for peak load predictions. This serves the purpose of displaying peak load predictions from the model in a simpler way to the intended user.

4.2 Architecture

The architecture of the application utilizes Flask, a lightweight python framework that provides tools, libraries and technologies for building a web application. In our application, flight serves the purpose of handling the backend processes such as connecting to the database, our prediction models and serving information to the frontend. A python framework works best for our purposes since our models are created using python and python libraries. The frontend of the application is built using HTML, Javascript and CSS. Bootstrap and Jinja are used as the templating structure to serve views to the frontend of the application.

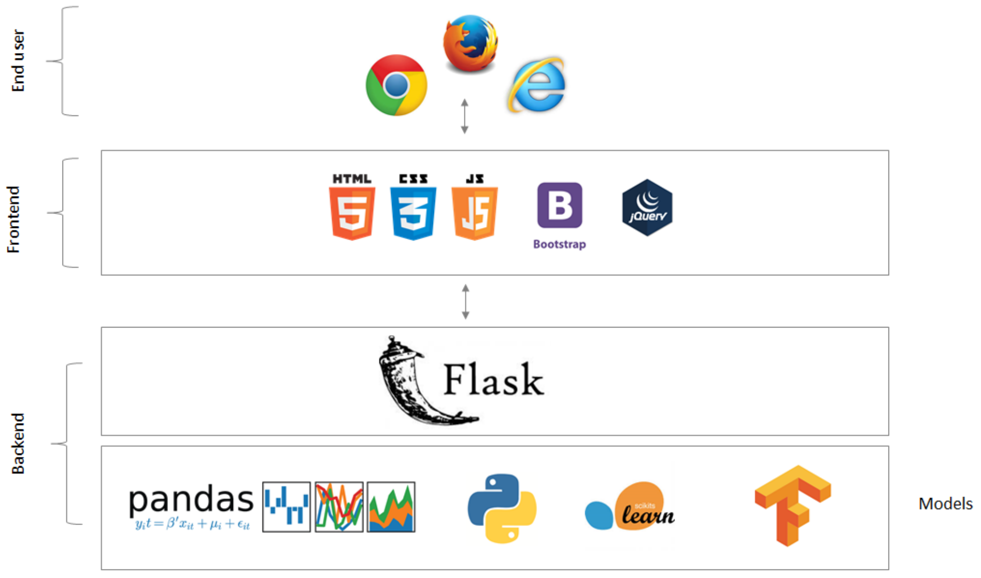


Figure 35: Application Architecture

4.3 Application Process Workflow

In the backend, Flask plays the main role of building the backend logic, routing or mapping URLs to specific functions, templating and handling requests. The dataset is pre-processed and saved as csv format as well as in a database. Jupyter Notebook handles this pre-processing, model creation and model saving. The saved prediction models are then loaded inside Flask during initialization. This saves time by ensuring the model is not re-run every time a prediction request is made. Peak load predictions can then be run on our dataset to produce prediction results. The prediction results are returned from Flask's routing functions using the Flask's template rendering libraries.

The frontend handles sending requests to the backend and serving template views. Ajax is employed to send requests without requiring page reloads. This process sends requests to the backend with inputs and this returns a JSON string of the prediction results based on the input string.

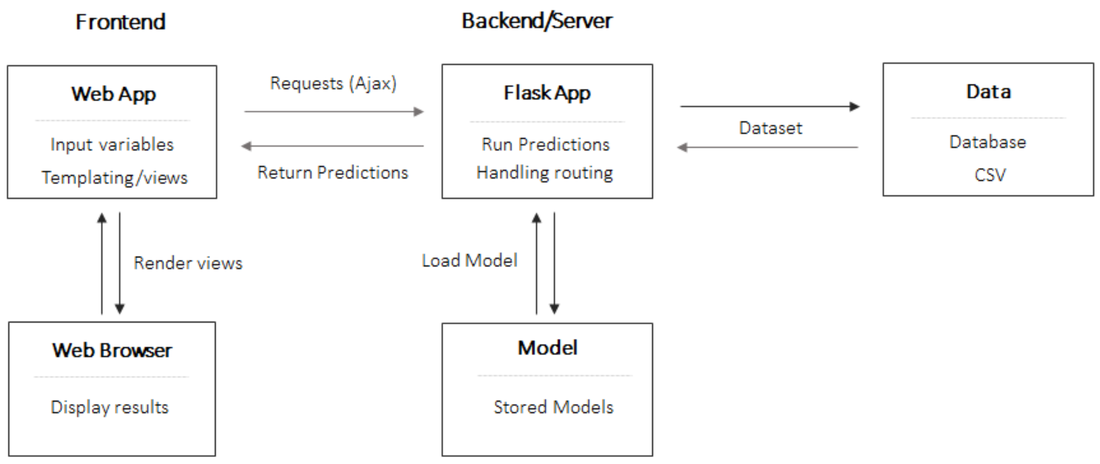
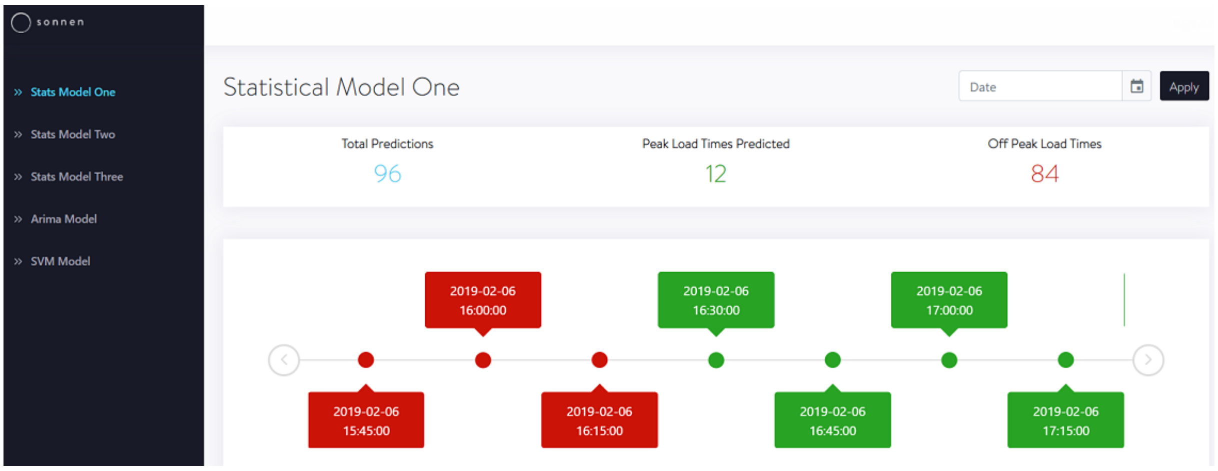


Figure 36: Application Process Workflow

4.4 Model Prediction Functions

The results from this prediction are saved as a dataframe in Flask and a function wrapped around it to return prediction values based on a specific date or time input. The statistical models work by predicting the peak load values for the entire year. The Arima model works by predicting the possibility of a peak for the next 15-minute slot based on the predictions for the previous two.

An ajax call sends a request with date and time as input (which the user can select) to the prediction function. The request to the backend then returns the predictions for this specific date. These are then displayed to the user. Peaks are displayed for 15-minute time slots using green to indicate predicted peak slots hence advising sonnen to discharge a battery. Red is used to indicate time slots that are not peak times.



Statistical Model One

Total Predictions	Peak Load Times Predicted	Off Peak Load Times
96	12	84

2019-02-06 15:45:00
2019-02-06 16:00:00
2019-02-06 16:15:00
2019-02-06 16:30:00
2019-02-06 16:45:00
2019-02-06 17:00:00
2019-02-06 17:15:00

Figure 37: Sample Application View

4.5 Possible Improvement

The idea goal will be to have the application take in live data, run the models and then return the predictions. However there exists the limitations of access to live data. Possible improvements also include extension of the input capabilities for selection of prediction results.

5 Summary

Statistical Model The statistical model (see chapter 3.1) is the baseline of our project. It is easy to understand, performs very well and it has been tested thoroughly. As such, we provide a finished product, ready to be used by Sonnen.

Despite its simplicity it provides a very solid baseline with an implementation which could be used immediately. It is a complete product which predicts the peaks and can be used easily only needing access to the database.

In average we expect a gain of 20 000 € per location and MW.

Support Vector Machine and Neural Network The support vector machine and the neural network (see chapter 3.2) we implemented provide predictions of the peak load using real time weather data and hence are able to create exceptional results. While they are not ready for production yet, they are a proof of concept showing that they are the right tool for the prediction of the peak loads.

Here, the next steps to be taken are the optimization of the parameters of the models, the correct kernel and the threshold σ for the support vector machine and number of layers and nodes for the neural network. Additionally one should incorporate live data like battery power levels or TSO data. This should greatly improve the accuracy.

For the support vector machine we expect a gain of at least 47 000 € per MW and location and for the neural network we expect a gain of at least 38 000 € per MW and location.

ARIMA Model The ARIMA model (see chapter 3.3) tackles the problem from the opposite direction. Here, we don't try to predict the peaks of the year. Instead, we predict the power consumption of the next 15 minutes based on the power consumption of the previous two 15 minute time slots. Using live data, we are then able to identify the peak loads with a high accuracy.

Unfortunately, there is no life data available yet and the ARIMA model is more of a theoretical solution than a finished product. Nonetheless, the idea already arouse interest for other applications, like the prediction of the charge of Sonnen's batteries.

Here, we expect a gain of 78 000 € per location and MW (for $T = 0.99$). However, this number has to be taken with a grain of salt since we could only use a small subset of our data to calculate this number because we need two years of data to apply the ARIMA model.

Web Application All these models are incorporated in our web application (see chapter 4). It show cases how one could apply our results in practices and provides a mock up for sonnen. As such we not only provide a theoretic implementation but also an actual program which can be applied easily in practice.

Bibliography

[1] sonnen GmbH, *sonnen's webpage*, https://sonnengroup.com/, Accessed: 01.11.2018

[2] Grey Cells Energy, *sonnen's "Community": Aggregating Domextic Battery Storage*, https://greycellsenergy.com/examples/sonnens-community-aggregating-domestic-battery-storage/y, Accessed: 01.11.2018

[3] Andy Colthorpe, *sonnen adds EVs to virtual power plant community*, https://www.energy-storage.news/news/sonnen-adds-evs-to-virtual-power-plant-community, Accessed: 01.11.2018

[4] Federal Ministry for Economic Affairs and Energy, *Development of Renewable Energy Sources in Germany 2017*, published in 2018

[5] Glowacki Law Firm, *Distribution System Operator (DSOs)*, https://www.emissions-euets.com/internal-electricity-market-glossary/623-distribution-system-operators-dsos, Accessed: 01.11.2018

[6] Next Kraftwerke, *Was ist ein virtuelles Kraftwerk?*, https://www.next-kraftwerke.de/wissen/virtuelles-kraftwerk, Accessed: 01.11.2018

[7] Verband kommunaler Unternehmen, *Vermiedene Netznutzungsentgelte (vNNE) für dezentrale Erzeugung im Zuge der Energiewende*, https://www.vku.de/fileadmin/user_upload/Verbandsseite/Themen/Energiewende/150825_7_VKU-PP_vNNE_final.pdf, Accessed: 01.11.2018

[8] German Federal Ministry for Economic Affairs and Energy, *What exactly are grid fees?*, https://www.bmwi-energiewende.de/EWD/Redaktion/EN/Newsletter/2017/13/Meldung/direkt-account.html, Accessed: 01.11.2018

[9] German Federal Ministry of Justice and Consumer Protection, *Verordnung über den Zugang zu Elektrizitätsversorgungsnetzen (Stromnetzzugangsveroerdnung - StromNZV) §17 Veröffentlichungspflichten der Betreiber von Elektrizitätsversorgungsnetzen*, https://www.gesetze-im-internet.de/stromnzv/__17.html, Accessed: 01.11.2018

[10] Robert Nau (Duke University), *ARIMA models for timeseries forecasting*, http://people.duke.edu/ rnau/411arim.htm, Accessed: 03.12.2018

[11] Paul Bourke, *Auto-regression Analysis (AR)*, http://paulbourke.net/miscellaneous/ar/, Accessed: 04.02.2019

[12] Pennsylvania State University, *Lecture STAT 510: Applied Time Series Analysis*, https://newonlinecourses.science.psu.edu/stat510/node/48/, Accessed: 03.12.2018

[13] Aarshay Jain, *A comprehensive beginner's guide to create a Time Series Forecast*,

https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/, Accessed: 15.12.2018

[14] Jason Brownlee, *How to Create an ARIMA Model for Time Series Forecasting in Python*, https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/, Accessed: 08.01.2019

Appendix

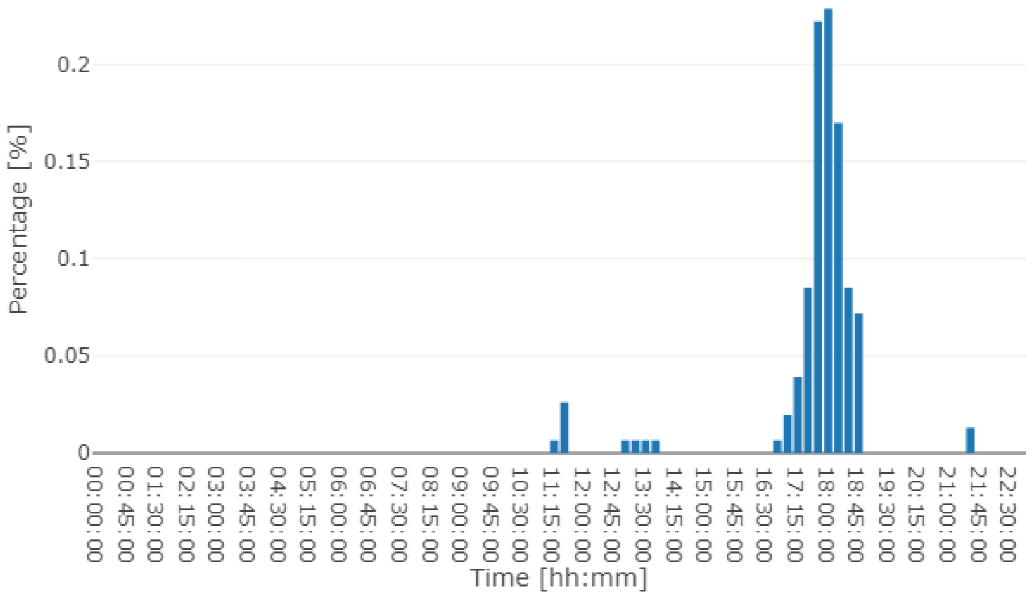


Figure 38: Distribution of the highest 99% within a day

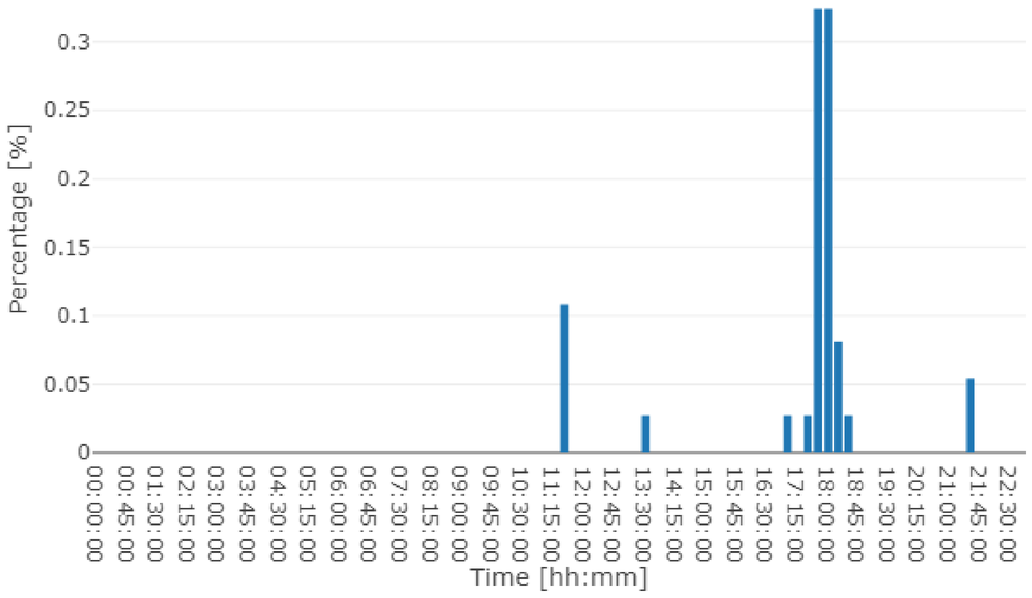


Figure 39: Distribution of the highest 99.9% within a day

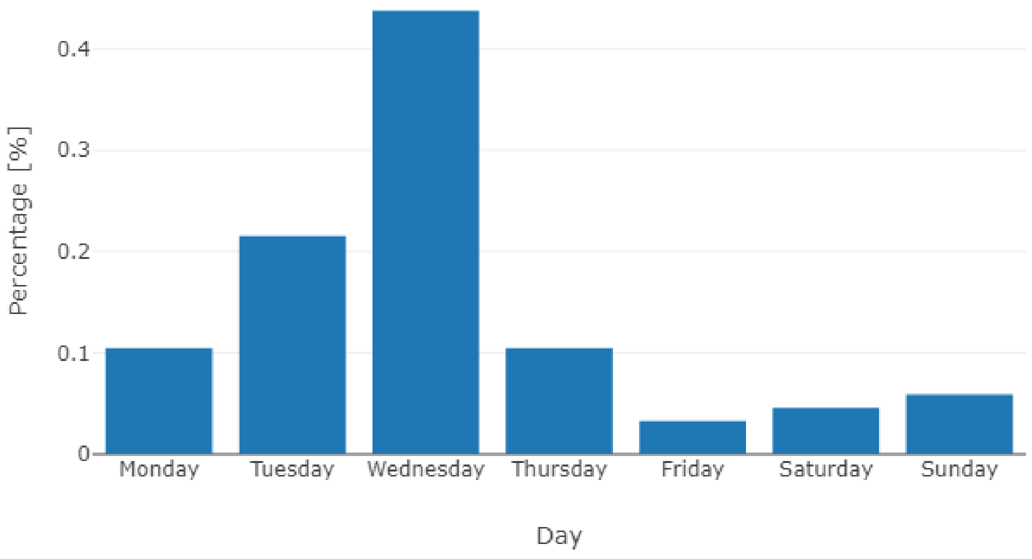


Figure 40: Distribution of the highest 99% within a week

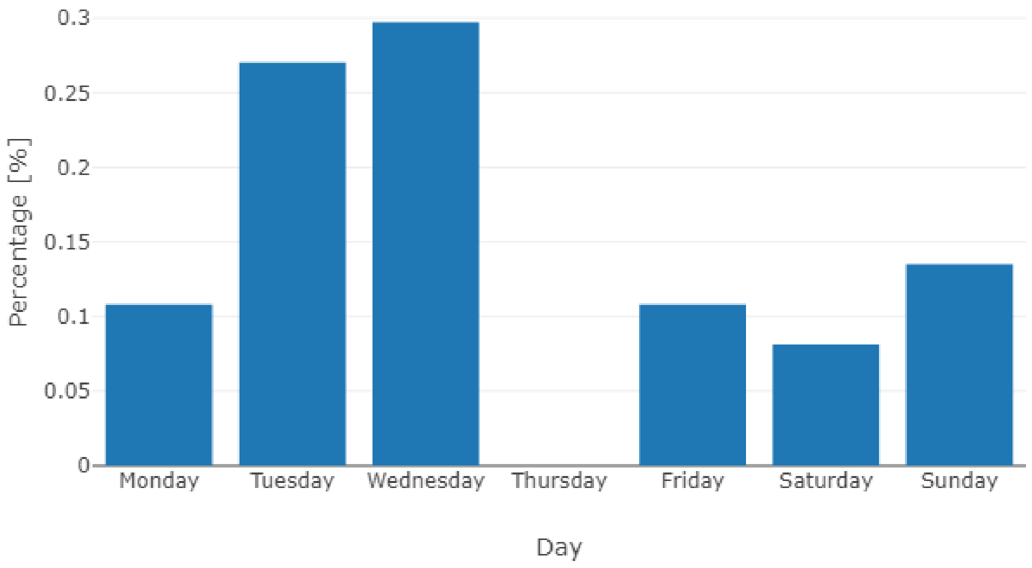


Figure 41: Distribution of the highest 99.9% within a week

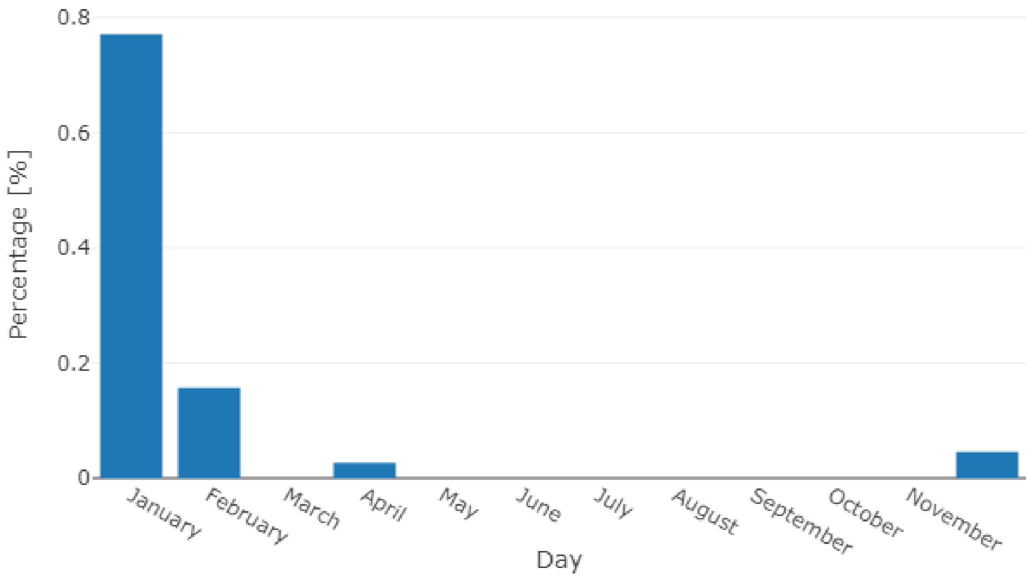


Figure 42: Distribution of the highest 99% within a year

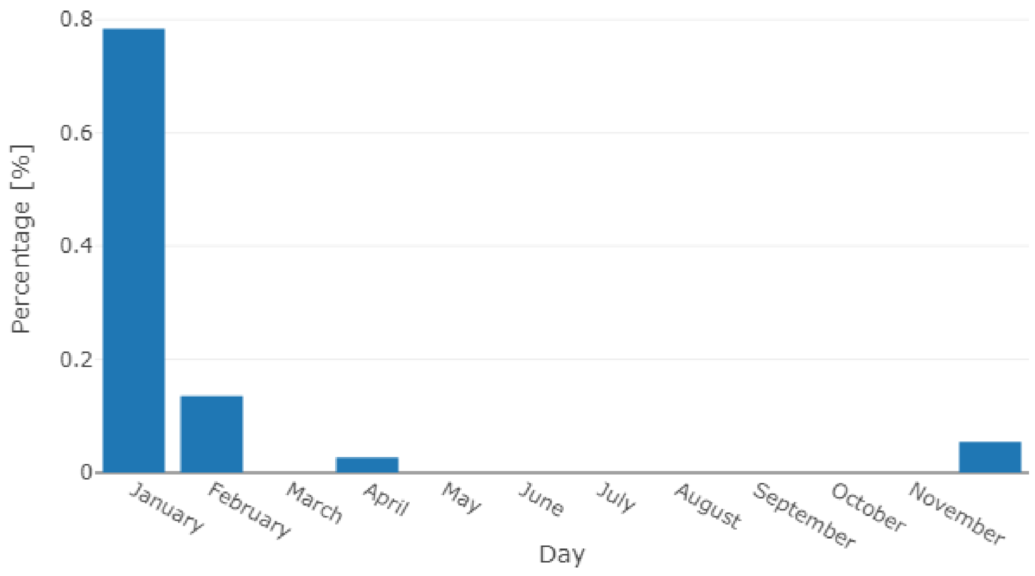


Figure 43: Distribution of the highest 99.9% within a year