Department of Mathematics
and Computer Science

University of Southern Denmark

Master Thesis in Data Science

Exploring climate data's relevance to predict Energy consumption

Author
Mathias Østergaard Hansen
Exam nr: 453228
mathh17@student.sdu.dk

Supervisors
Vaidotas Characiejus(SDU)
Jeppe Miki Busk Olsen(Energinet)

May 6, 2022

Introduction

Energinet and DMI are both organisations working with a lot of data. Fortunately both organisations are generous enough to share most of their data publicly.

Data Collection and Preprocessing

Observational data from DMI

In 2020 DMI began uploading complete data sets of their weather data through an API-service. There are four data sets available through their API, the two that could be relevant to the project would be "climate data" and "Meteorological Observation", metObs. Climate data is the latest release for the API and it does not contain a lot of data, there are a lot of holes in the data set and very few stations with data on for relevant parameters. The good thing about the climate data set is that is quality checked and rinsed for outliers. The metObs data set is a dataset of a wide range of meteorological observations recorded by DMI, all the way back to 1953 actually. The data set is mostly complete with very few gaps but the downside to this dataset is that the data set is not quality checked and requires the user to handle erroneous data. There are ways to deal with outliers and in the context of this project it is easier to deal with outliers rather than dealing with a mostly incomplete data set, as the climate data set is still very incomplete. DMI records the weather data from its weather stations placed around all of Denmark. DMI describes the occurrence of outliers usually comes from malfunctioning sensors, wear and tear, and extremely rarely vandalism. In the case of this project the two interesting parameters from the metObs data set are the temperature mean and radiation mean, which is the mean of the hour. These two variables are means over an hour of data. Which means the possible outliers will be watered down before being reported. In this project, each recorded observation will be added together with a set of observations from other weather stations and then have another mean calculated from those values. Thus impact of an outlier from one reading will be very small, therefore no further action will be taking against outliers. Later on when the Climate dataset is fully developed this would be the preferred data set.
When the data set is downloaded through the API, the data is structured station by station, then each station contains a dataframe of all the recorded observations. For the purpose of predicting the consumption for the areas DK1 and DK2 the weather stations will be divided into three groups, DK1,DK2 and neither. The stations that get placed in the neither category will be excluded. The criteria for excluding a station is if the station is placed either at sea or on the island of Bornholm. The reason for this is that Bornholm is only responsible for a minimal amount of the overall consumption of DK2 which is the area it would placed in, and the weather on Bornholm might differ significantly from the rest of DK2.

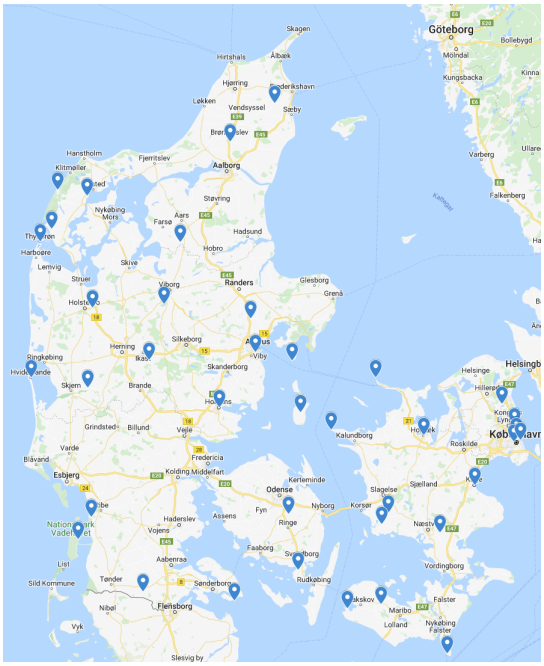


Fig. 1: Map of weather stations in DK1 and DK2

This was done by taking the most eastern, western, southern and northern coordinates from both DK1 and DK2. The returned stations are then checked manually to see if it is in within the right category, and if it is not the data from the station is removed. This resulted in complete data from 25 stations for DK1, and 15 DK2, Figure 1. All the data from each station in each area are then added together to get a mean for each feature for the given price area. This will the data that is used to train and tune the models. The reason for choosing the features time, temperature and radiation comes from advice and experience from Energinet. From their experience it would be the two interesting features from the observations. Time and temperature is obvious, given the amount of electricity used for heating, but the radiation might be a surprise to some. The reason that there Energinet expect to see a correlation between radiation and the electricity consumption is because of the solar panels that an increasingly amount of danish homes have installed. These panels produce electricity that is added directly to the danish electrical grid. This means Energinet will have to produce less electricity from the power plants because the solar produced electricity is not consumed by the single consumer but it is fed into the general grid.

Electricity Consumption Data from Energinet

Just like DMI, Energinet's data is available through an API. From their web portal: energidataservice.dk there are a wide range of data sets available. The relevant data set for this project is the one called *"Production and Consumption*

- *Settlement"*. This is a data set of the over all electricity consumption divided into the two price areas DK1 and DK2, and fortunately this data set is quality checked and 99.9% correct. This data set contains many features, of which the price area, time, gross consumption and solar power self consumption will be the ones relevant in this case. The feature, solar power self consumption, will not be a part of the final data set, but the data from this feature will be deducted from the gross consumption feature. the reason for this is because Energinet wants to know how much electricity they need the power plants to produce, and that is exclusive of this production from the solar panels.

Meteorological Forecast Data

When the models have been trained on the observed data from the MetObs dataset, the models should be able to use similar forecast data from DMI. Energinet receives forecast data from DMI in different time resolutions, for this project the data set will be the hourly forecast data which comes in every 3rd hour 54 hours ahead. The way this is structured is that for the first hour, Energinet receives data from 1,4,7,10 etc. hours before. For the second hour it will be 2,5,8,11 etc. and for the third hour it will be 3,6,9,12. For the purpose of predicting hourly consumption in 48-hour intervals this data will have to be restructured to be 1,2,3,4 hours from one time step. This was done by unloading predictions into categories for each step ahead. So all the forecasts for 1 hour ahead was in one dataframe. This was stored in one dictionary. This allowed for one hour to be pulled from each category and put back into one big dataframe. This results in for one time step the next 48 hours of forecast data will be the next 48 rows of the dataframe. Then index 49 will be three hours after hour one and then the next 48 hours for that hour will be stored. This allows for predictions being done for single time steps 48 hours ahead, for each available time step. For the purpose of this project, the time horizon that the models will try to make forecasts for is the year of 2019. Thus the model will be tested on a complete seasonal year, and it should yield enough predictions to give insights to the performance of the models. An interesting topic to explore was how precise the forecast data were in relation to the actual observed values. Also, how much does the time ahead the values is predicted affect the precision of the forecast? The forecasted temperature seems to fit the observed values rather well. However, the forecasted radiation values seems to deviate quite a bit from the observed values. looking at the plot it seems to underestimate the radiation quite a bit. In order to get an understanding of how precise these forecasts are the R^2 value was calculated for both 1 and 48 hour ahead forecasts.

R^2 values for forecast data		
	1 hour ahead	48 hours ahead
temperature	0.942	0.938
Radiation	0.705	0.674

Table 1: R^2 values for the year 2019 for 1 and 48 hours ahead forecasts.

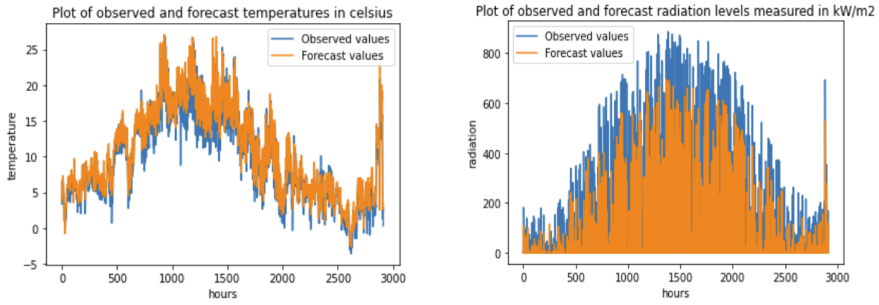


Fig. 2: plot of observed and forecast data for the temperature and radiation for forecasts 48 hours ahead only.

Looking at the R^2 values for temperature DMI is able to adequately forecast the temperature over the course of a year. Both for 1 and 48 hours ahead, the reduction in the R^2 value going from 1 hour to 48 hours is minimal and does not appear to be of any concern. However, the R^2 values for radiation suggests the same as the plots. There are significant errors in the forecast of sunlight, when compared to temperature. In terms of data quality this might turn out to be quite the factor for eventual error in the predictions. Since you want the training data to be as similar to the actual data as possible.

Data Preprocessing and Exploration

Developing machine learning models require a deep understanding of the available data. In order to get a overview of the correlations and relations between the data from DMI and Energinet the data will be explored thoroughly and visualized. There are some initial assumptions about the data that will be investigated during this process:

- There is a weekly pattern of power consumption where there is a decline during weekends and holidays.
- There is a correlations between the time of day and consumption.
- There is a negative correlation between sun radiation and consumption.
- There is a correlation between temperature and consumption.

These assumptions will be either confirmed or denied through the exploration of the data, or literature from relevant studies. These assumption comes from my general idea of how the electrical consumption behaves and Energinet's anecdotal experiences. First the time of the observations will have to be organized in a manner that allows to fully investigate it's impact on consumption. For the purpose of this project where there are behavioural patterns for electricity during the week and holidays, a binary feature will be added to the data set, if it is a holiday or not. This feature should capture the decline in consumption that evidently happens on weekends and holidays [9]. Looking at the behaviour

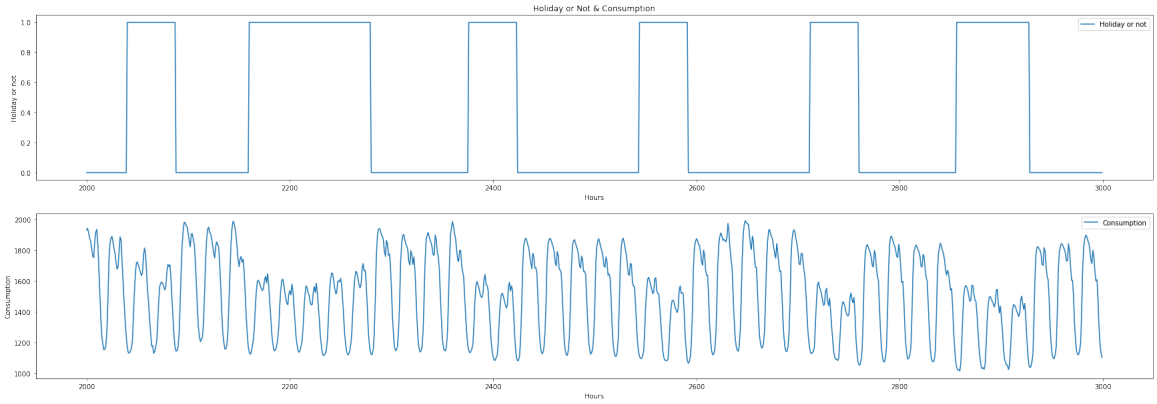


Fig. 3: Graphs over consumption and holiday feature over a period of time, specifically around Easter

of consumption and weekends or holidays it appears that there is a correlation between weekends and consumption, Figure 3. The consumption appears to decrease during the weekends, which confirms the results put forth by Ziel. This pattern possibly comes from a large part of the industrial sector closes down over weekends and public holidays. Ziel also compares the effect of public holidays like easter over a seven year period, and it appears that the impact of holidays is rather stable. This representations of days of the week seems to capture the behaviour of the electricity consumption in a sufficient manner.

Since most of the danish population is sleeping during it is expected that the electricity consumption is reduced during the night, but exactly how is the behavioural pattern of the consumption in relation to the hour of the day?

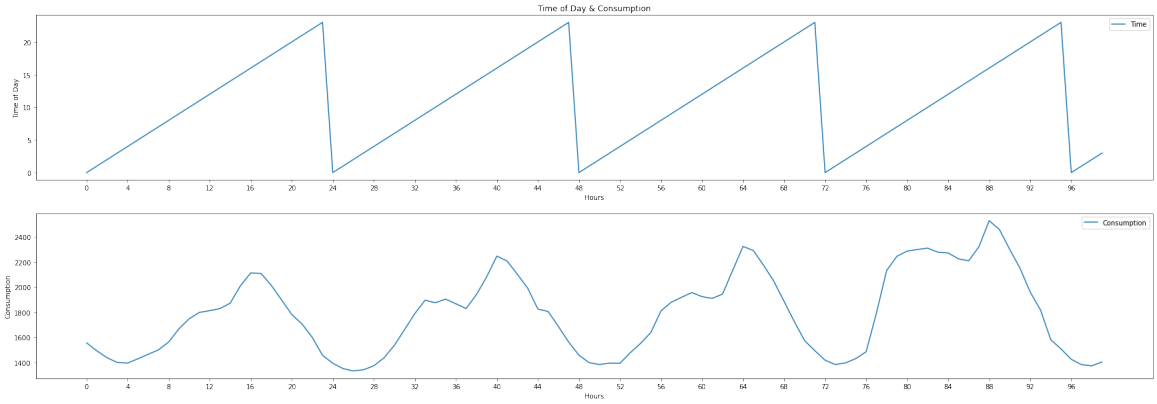


Fig. 4: Graphs over consumption and time of the day over a period of time

As expected there is a clear correlation between the time of day and electricity consumption. Something that seems apparent from the graph, Figure 4, is that there are daily peaks after 4pm, where people return from work. This makes

sense since people generally will be preparing dinner, and using their appliances to wash their clothes etc.

During the last 10 years solar panels have become increasingly more popular in Denmark, in September 2019 106,461 soloar panels to be exact, Bolius. As stated earlier, this is not a form of consumption that Energinet needs to predict, therefore the consumption is deducted from the gross consumption. By now, with so many solar panels installed the effect from these solar panels is assumed be noticeable on the gross consumption.

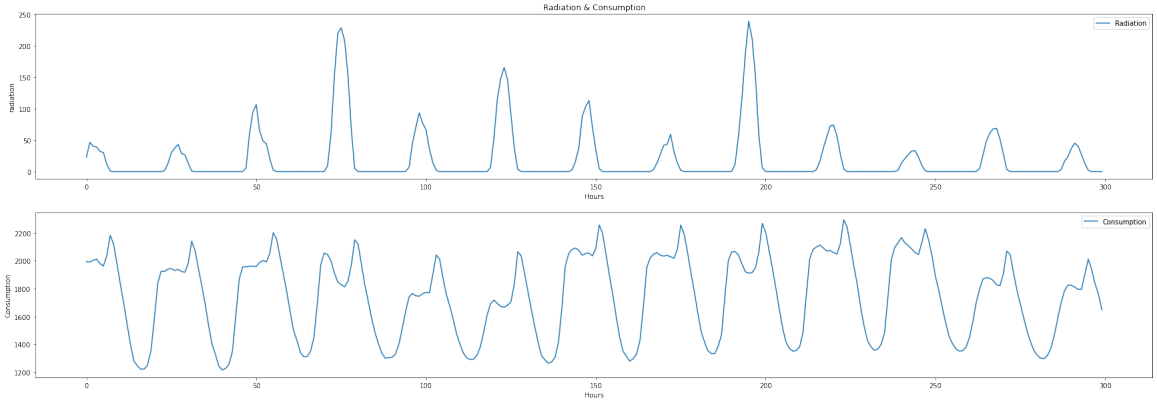


Fig. 5: Graphs over consumption and radiation over a period of time

Inspecting the graph, Figure 5, it is possible to see a noticeable dip in consumption on the days with high radiation. Around 75 hours, and 200 there are small dips in the consumption, which could be caused by the increased radiation. There are also other possible reasons this could happen. It might be because people are more willing to spend time outside when the sun is shining, and that they just simply are not using their appliances. Nevertheless, in the purpose of this project, the causation is not the main factor, but the correlation is.

Since such a big amount of the residential electricity consumption is used for heating up houses, it is expected to see some form of correlation between temperature and electricity. In the danish heating industry, the companies convert the temperature to degree-days to best capture the energy required to heat up a house, [2]. Degree-days is a unit for measuring the difference between the outside temperature and 17 degrees Celsius. This gives a degree-day of 5 if the outside temperature is 12°and -2 if the outside temperature is 19°. The reason the heating industry uses this unit rather than just using temperature, is because this best captures the impact the cold has on the energy required to keep a house at the mean temperature. By advice from Energinet, this will also be the unit used for this project, since Energinet also uses degree-days when measuring temperature in relation to energy consumption.

There appears to be a relatively good correlation between degree-days and the electricity consumption, Figure 6. As the degree-days gets closer to zero it does seem like the correlation begins to stagnate, which fits the narrative

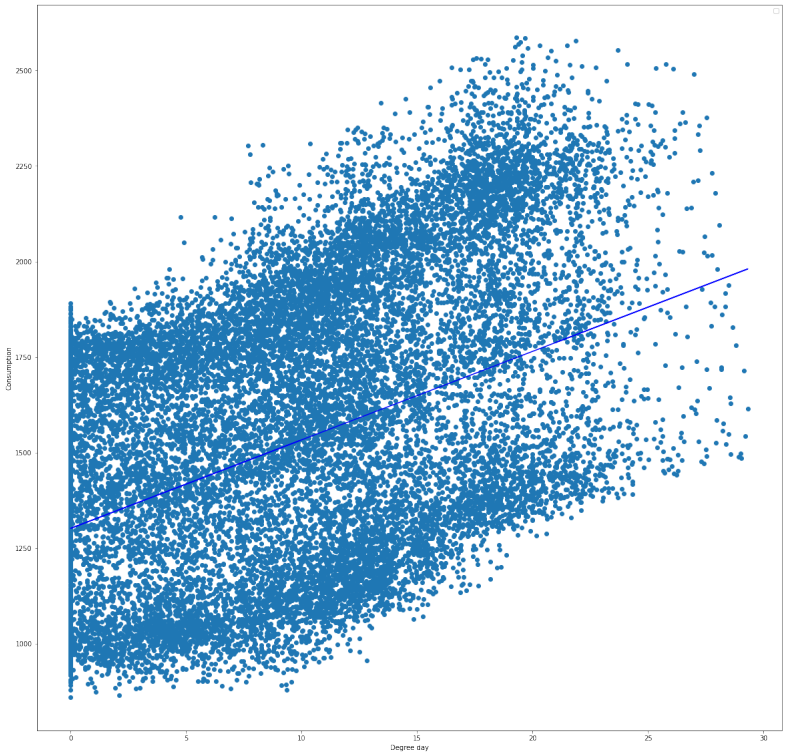


Fig. 6: Plot of the consumption in relation to degree-days with a tendency line

stated earlier, that after a certain point the temperature has no effect on the consumption.

Feature importance

After having visually inspected all the relevant features available, a final step was taken to see the importance of these features. As it is visible from the graphs and plots above, the correlations are not equal. Therefore, to get a better understanding of the information a machine learning model is able to extract from these features, the SHAP values from the features will be calculated. By calculation a predictions SHAP values, SHapley Additive exPlanations, you can break down the impact the values of each feature has on the prediction. This is extremely useful for checking for redundant features in your data set. Shapley values are based on a solution concept from game theory used to determine the contribution of each player, and the name shapley comes from the mathematician, Lloyd Shapley who introduced the theory in 1951. In 2017 this theory was used in machine learning, [7], with a python package that is able calculates these values based on your predicter, SHAP. In order to calculate the SHAP values a very basic predicter will be implemented, in this case a XGBoost predicter. XGBoost is a boosting algorithm which boosts many small decision tree, for calculation

the SHAP values on a prediction like this, the "tree interpreter" method will be used from the SHAP package. The values are calculated by the formula::

$$\phi_i = \sum_{S \subseteq F\{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)] \tag{1}$$

SKAL SKRIVES FÆRDIG

Using the simple XGBoost model the SHAP values calculated for the features can be presented in a summary plot:

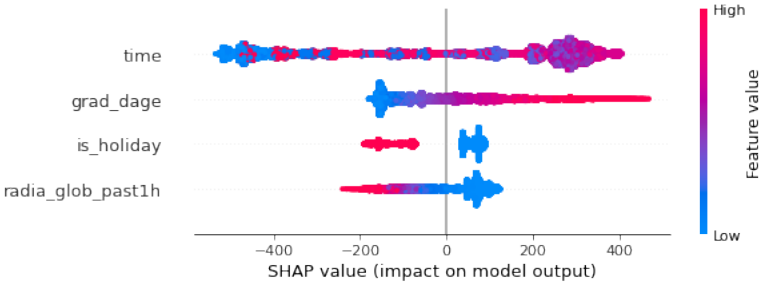


Fig. 7: Summary plot of the SHAP values, for each feature

Fig. 8: Plot of the SHAP values for temperature

Degree-days have a nice gradient coloring throughout the whole plot. It seems to be very clear to the model that a high value degree-day means an increase in consumption. the impact of appears to be more consistent for degree-days than the time feature. The degree-day feature stops abruptly for low values, this is at the 17 degrees, where the degree-days are capped and goes to zero. In order to check whether or not the model is able to gather some information from higher temperatures i made the same plot, but with temperature in celsius, Figure 8, and it appears that the temperature has the same impact as degree-days. For the is_holiday feature all values that indicates a weekend or a holiday seem to produce a negative deviation for the consumption, which is in line with the correlation found in the plot.

The radiation plot shows that the impact of the radiation actually causes a negative deviation, and this confirms that the model is actually able tick up on the small dips that appeared to be present on sunny days.

However, the time feature seems to be more difficult to decipher. Since the SHAP values represents data in a continuous manner, representing the time feature is not very good, since it is cyclical and not continuous. This means whenever the time passes midnight, the feature value goes from the highest possible value to the lowest, but the consumption is not cyclical. This explains the mix of the peak values throughout the plot. Looking at the right side of the plot it would appear that the model is able to pick up the daily peaks in the data, that happens around 16 when people return home from work.

After inspecting both the data correlations and their impact on data models, there does not seem to be any redundant features. The final data set that will be used for training looks like this

Training Dataset				
Time	Degree-days	Radiation	holiday	Consumption in MWh

Table 2: Features in the data set that will be used for training the model

Literature Review

Predicting energy consumption is not a new subject of study in the field of machine learning. There are wast amounts of papers, reports and projects on the subject. Through these academic papers almost every regressional model have been implemented to predict the energy consumption in some form. With the increase in popularity for the subject of machine learning and especially neural networks in the last 20 years, many new methods of predicting this consumption have been developed. In the report [5] it is made clear how big of an increase forecasting energy consumption has seen, and that is just from 2005-2015. In the figure from the report, the increase is depicted in a diagram showing the growth from 1994-2014, which appears to be exponential. This report indicates that there is a big interest in the subject. In the report they state that the reason for the importance of being able to predict the electricity consumption has many factors. Distribution of electricity is a big factor, since you do not want to overproduce, or under produce. In Denmark overproducing is handled by exporting excess power to neighbouring countries. This happens quite frequently because Denmark is able to produce large amounts of electricity by wind. Therefore correctly distributing and allocation is of big importance.

To get some perspective on state of the art models and what is considered best practice on this topic, relevant literature will have to be researched and reviewed. Review papers will be the initial way of gaining a broad perspective on relevant models and implementations. In the report, [1], a wide variety of papers on the subject of Artificial Nerual Networks(ANN's) used to forecast electricity demand are reviewed. In total they review 56 papers, and interestingly enough 46 of the papers are on Short Term Load Forecasting(STLF). It appears from the paper that the most popular load forecasting method is short term, in terms of ANN's. From the applicability of ANN's and the way they use data to make predictions this makes sense. Because ANN's is very reliant on the quality and precision of the data available to the model, so making long term load forecasting(LTLF) or even medium term would seem difficult without any real world data. STLF seems to be predicted from weather forecasts, which becomes more and more incorrect the further in the future the forecasts are done. In terms of best performing models the paper states that usually the best performing models are models that are able to capture recurrent patters, such as recurrent neural networks(RNN). Which is in line with they claim that using the previous energy consumption

for forecasting in the future is a better parameter than the weather. Especially the Long Short Term Memory,LSTM, is able to achieve some very good results. An interesting note is that the paper states that with the LSTM the predictions tend to become more accurate the more consumers the data is aggregated over. The paper also reviews the performance metrics used in the 56 papers. The most popular metric for performance is MAPE which stands for Mean Average Percentage Error. This metric results in a percentage rather than an actual value for the error. This allows for comparison between the different implementations and models. The review paper, [6], comes to similar conclusions as the previous paper when applied to larger scale predictions. However in one of the papers that is reviewed, the extreme gradient boosting algorithm, XGBoost, shows the best results. But in that specific paper it seems to be a regular Deep Neural Network (DNN) rather than a form of RNN. In the review paper, [4], they state the use of boosting algorithms as being "underexploited". The modern boosting algorithms, such as XGBoost, LightGBM and CatBoost, have shown state-of-the-art prediction performances. The paper goes into great length to explain the ease of implementation of these modern boosting algorithms. Especially the XGBoost algorithm is generally perceived as having a very good out-of-the-box performance. This is in line with one of the conclusions from [5] in which they state:*"As a conclusion, developing simple methods with acceptable accuracy is an attractive area of future studies."* Since the computational resources needed for these very deep neural networks are significantly higher than the newer machine learning algorithms, such as gradient boosting algorithms.

From the review papers, the state-of-the-art methods of load forecasting seems to be using a form of RNN, more specifically LSTM's seems to be very popular. Boosting algorithms also seem to be able to achieve acceptable accuracies. LSTM's and neural networks in general have many parameters and a big part of setting up a neural network is to tune these parameters, to get an understanding of how this is done some papers of actual implementations will be reviewed also.

IKKE HELT FÆRDIGT

Methodology

Now that a general understanding of what is perceived to be some of the best practices for modeling load forecasting, it is necessary to understand the chosen methods on a deeper level. In this section the models: Gradient Boosting, specifically XGBoost and the RNN model LSTM will be explained in order to gain an in-depth understanding of the models.

Extreme Gradient Boosting

Gradient Boosting

The general idea behind machine learning models is to construct one model that is able to make predictions. However, boosting algorithms aims to train a sequence of weak learners that as a whole is able to make predictions, also called

ensemble learning. Each model is trained to compensate for the weaknesses of the model before. In the relevance of gradient boosting, the weak learners consists of small decision trees, which are sequentially improved upon based on the residual errors of the previous decision tree. The first tree in the sequence is usually leaf which predicts the average of the target values, for regression problems, and after that the boosting algorithm will gradiently reduce the error by adding more trees. Decision trees consists of a certain amount of binary statements, also called splits, these splits is used to either make a prediction or there will be a new split until they reach a prediction, Figure 9. Gradient boosting algorithms utilizes these trees as the weak learners they train sequentially. The sequential connection of these trees means for each tree, that tree is fitted to the residual errors of the previous tree, which it then tries to minimize. In the case of regressional implementations of gradient boosting algorithms, the residual error is calculated by a loss function. Since gradient boosting is a supervised learning algorithm, it would be easy for these trees to fit itself to the training data and thus very quickly reduce the residual errors to zero. In order for this not to happen, each trees contribution is scaled by the learning rate. The learning rate ensures that each tree contributes the same amount. Friedman who was the one to first implement gradient boosting states that lower values for the learning rate typically yields better results [3]. There is however, a trade-off in terms of implementations. Having a low learning rate require more trees in the model and that increases the computational requirements for the model. Each tree then contributes to the overall prediction with a weight that is added to the rest of the trees weights. This means the model aggregates the weights of all the small trees, this value is then the deviance from the average value that the prediction is. When implementing gradient boosting algorithms the depth of the trees will have to specified as it is one of the hyperparameters for this form of boosting algorithm. The trees calculated by gradient boosting algorithms have a fixed depth, and allows for the individual trees to be quite deep. However increasing the depth of the trees, also increases the computational recourses required by the model quite significantly.

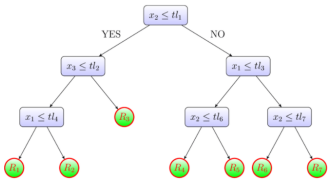


Fig. 9: Example of decision tree

XGBoosting

The algorithm XGBoost is based upon the main principles of gradient boosting, but there are some very important differences. One of the main differences are the way the decision trees are calculated within the algorithm. The initial leaf in the first tree is always 0.5 instead of the average that the normal gradient boosting algorithm used. The splits in decision trees are based upon a value called information gain, and in regular decision trees this information gain is based upon the reduction in entropy. However, in XGBoosting the splits for the decision trees are calculated based on a value called structure score, or similarity score. This score is a representation of the similarities in the leaf of the split,

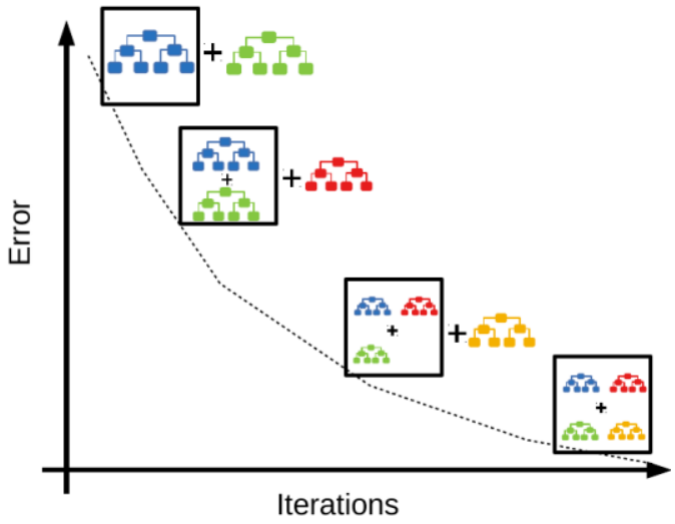


Fig. 10: Illustration of how the gradient boosting algorithm sequentially adds trees to reduce error

where a small value means a strong similarity and it is calculated like this:

$$Similarity\ score = \frac{Sum\ of\ Residuals^2}{Number\ of\ Residuals + \lambda} \tag{2}$$

Lambda is a regularization parameter and will have to be specified during implementations. The purpose of lambda is to reduce the predictions sensitivity to individual observations, thus helping the model from overfitting. The information gain of a split is then calculated by:

$$L_{split} = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma \tag{3}$$

The method of updating the gain of the split in order to find the split are not the same for all models. When the algorithm has found the best split, the loss function used to calculate the residual errors are squared error algorithm:

$$\sum_{i=1}^{n} L(y_i, p_i) = \frac{1}{2}(y_n - p_n)^2 \tag{4}$$

However in order to reducing overfitting the following regularization has been added to the model:

$$\sum_{i=1}^{n} L(y_i, p_i) + \gamma T + \frac{1}{2}\lambda ||w||^2 \tag{5}$$

Where the w stands for the output value. That is calculated with the following formula:

$$Optimal\ value = -\frac{1}{2}\sum_{j=1}^{T} \frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T \tag{6}$$

In theory this would have to be done on all values in the data set. But doing so for very large data sets will be too computationally heavy. Luckily the XGBoost algorithm have multiple ways of updating the gain during training. An exact greedy algorithm and a approximate algorithm for split finding:

Algorithm 1: Exact Greedy Algorithm for Split Finding

Input: I, instance set of current node
Input: d, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i$, $H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ ***to*** m **do**
  $G_L \leftarrow 0$, $H_L \leftarrow 0$
  for j *in sorted*$(I,$ *by* $\mathbf{x}_{jk})$ **do**
    $G_L \leftarrow G_L + g_j$, $H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split with max score

Algorithm 1, the exact greedy algorithm iterates over all possible splits and thus comes to the best possible split. But from the XGBoost documentation it is advised to only use this algorithm for smaller dataset since it becomes quite computationally heavy. in the paper this is Therefore the approximate algorithm was developed to increase XGBoost scaling capabilities for bigger data sets. In the implementation that is done by XGBoost the model is able to decide which algorithm to use in the specific situation. Even though this helps reducing the computational requirements for finding the best split. XGBoost goes to an even greater extend by implementing column blocks for parallel learning. This lets the algorithm split up the dataset so multiple threads can store and work on calculating statistical properties. In the paper describing XGBoost, [8], they state that at larger data sets the cache-aware algorithm is able to run twice as fast as the naive.

Long Short Term Memory

LSTM's are a form of Neural Network, NN, in the subcategory of NN's called Recurrent Neural Networks, RNN. It is therefor crucial to understand both NN's and RNN's to fully utilize the power of LSTM's.

Neural Networks

Neural networks are a set of algorithms that together tries to generalize over a set of training data enough to make predictions for a given feature. A NN in its simplest form consist of 3 types of layers, a input layer which contains

the input data for the algorithms to generalize over. A Output layer which will contain a representation of the target feature in a value between 0 and 1. In the middle of these two layers are what is called the hidden layers. The reason the middle layers are called hidden layers is because the value these layers contain are usually not relevant for the user to know. In a implementation of a NN, there

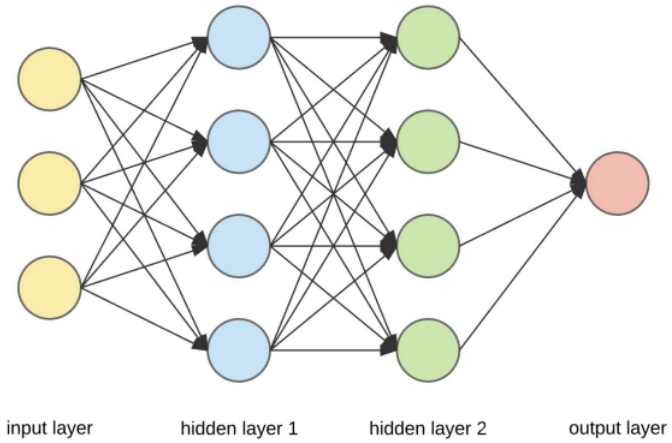


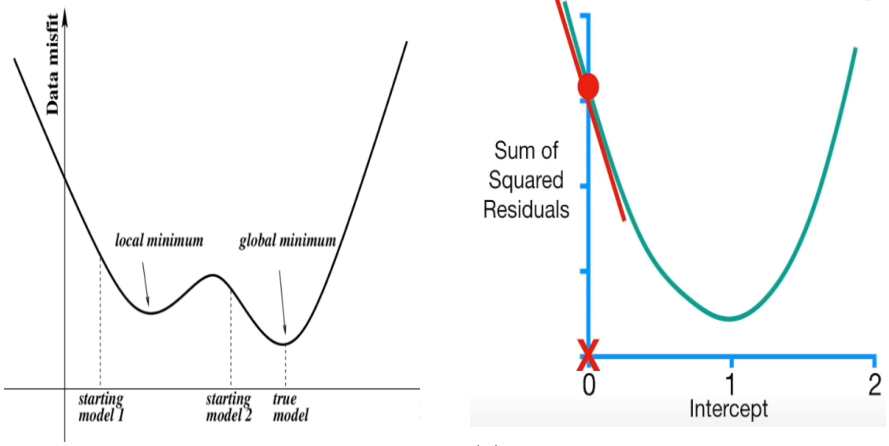
Fig. 11: Diagram of a simple neural network

can be many hidden layers, but usually there will be one input and one output layer. The job for these hidden layers are to capture particular patterns for the data. Each node in the layers will try and capture some part of information from the data sent from the previous layer. The change in the information between the layers happens because each node is connected to the nodes in the previous layer, and these connections contains weights and biases. When we talk about training a neural network or that the network is learning, what is meant is that these weights and biases are tweaked and tuned to better capture the input data for the right output.

Gradient Descent

When a model's weights and biases are updated it is done through backpropagation. What is meant by updating the weights and biases for a model is that they are updated in an effort to reduce the Sum of the Squared Residuals(SSR). Reaching the lowest point of SSR's is called reaching the global minimum, this is where the model is as precise as it can possible be. However, finding the global minimum cannot be guaranteed, the reason for this is the way the SSR are minimized during training. In order to decrease the SSR in a timely manner, the SSR is minimized by gradiently calculating the descent of the slope of the SSR. This is called gradient descent and is the way the SSR is minimized so that when the model updates the weights and biases the new values should reduce the SSR.

This might seem simple for the model, the issue is that there might be multiple local minimum and the model might not be able to move past local minimum. With the initialization of the model being random this is the reason there is no



(a) Depiction of local and global minimum (b) Depiction of gradient descent slope and the steps along the X-axis

Fig. 12: Caption for this figure with two images

guarantee for finding the global minimum. But with gradient descent the model will usually find a local minimum.

Recurrent Neural Networks A problem that regular feed-forward neural networks suffers from are the lack of memory between predictions. FFN's are not able to transfer knowledge from one prediction to the next. In terms of predicting on sequential data where the previous prediction are related to the next this is a downfall. This is where Recurrent Neural Networks(RNN) solves that issue. RNN's contain a "hidden state" which contains knowledge from previous time steps, this hidden state then feeds that information into the next prediction and is updated with new knowledge again.

Long Short Term Memory

Implementation

Results

Discussion

Conclusion

References

[1] José Juan Pazos-Arias Antón Román-Portabales Martín López-Nores. *Systematic Review of Electricity Demand Forecast Using ANN-Based Machine Learning Algorithms*. 2021. DOI: https://doi.org/10.3390/s21134544.

[2] *Energihåndbogen*. 2019. URL: https://evu.dk/wp-content/uploads/2019/06/Graddage.pdf.

[3] Jerome H. Friedman. "GREEDY FUNCTION APPROXIMATION: A GRADIENT BOOSTING MACHINE". In: 1999, pp. 1189–1232.

[4] Georgia Papacharalampous Hristos Tyralis. *Boosting algorithms in energy research: a systematic review*. 2021. DOI: https://doi.org/10.1007/s00521-021-05995-8.

[5] Gary R Weckman Iman Ghalehkhondabi Ehsan Ardjmand. *An overview of energy demand forecasting methods published in 2005–2015*. 2016. DOI: https://doi.org/10.1007/s12667-016-0203-y.

[6] Radu Zmeureanu Jason Runge. *A Review of Deep Learning Techniques for Forecasting Energy use in Buildings*. 2021. DOI: https://doi.org/10.3390/en14030608.

[7] Su-In Lee Scott M. Lundberg. *A Unified Approach to Interpreting Model Predictions*. 2017. URL: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[8] Carlos Guestrin Tianqi Chen. *XGBoost: A Scalable Tree Boosting System*. 2016. DOI: https://doi.org/10.48550/arXiv.1603.02754.

[9] Florian Ziel. *Modeling public holidays in load forecasting: a German case study*. 2018. DOI: https://doi.org/10.1007/s40565-018-0385-5.