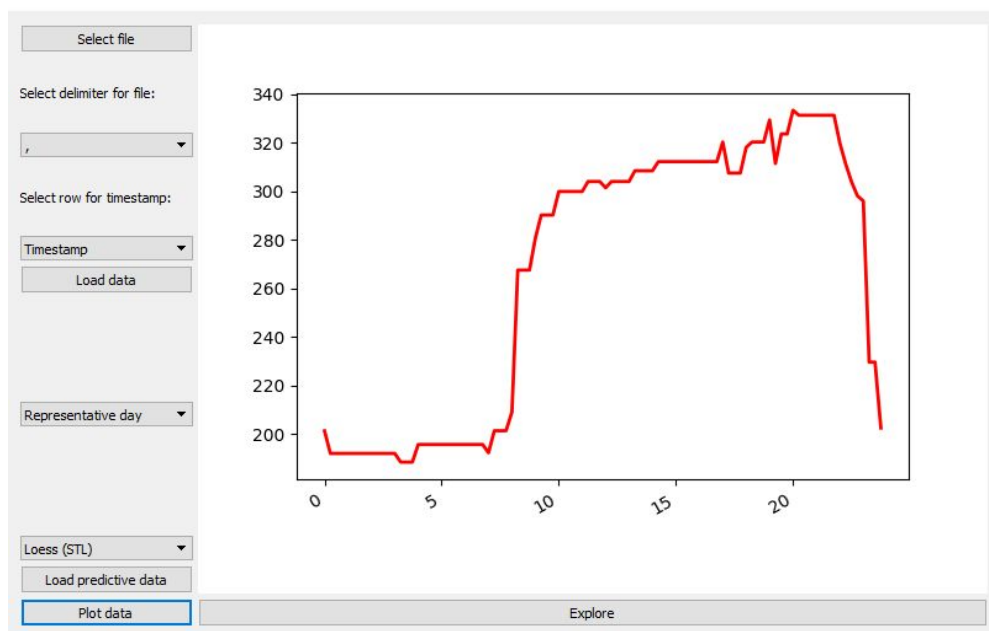


Determining a Representative Day for yearly smart building data



UNIVERSITY OF
SOUTHERN DENMARK



Niels Frederik Norberg

Lasse Pedersen

ninor15@student.sdu.dk

lasse15@student.sdu.dk

I Index

I Index	1
II Abstract	3
III Reading guide	3
IV Acknowledgements	3
1 Introduction	4
2 Research	5
2.1 Background	5
2.1.1 Data format	5
2.1.2 Representative day	5
2.1.3 Time series data	5
2.1.4 Regression and regressors	6
2.1.5 Related work	6
2.1.5.1 Synthetic Data	6
2.1.5.2 Big Building Data	6
2.1.6 Importance	7
2.2 Simple models	8
2.2.1 Linear Ridge Regression	8
2.2.2 Boosting Decision Tree Regression	8
2.2.3 Nearest Neighbors Regression	9
2.3 Advanced implementations	9
2.3.1 Seasonal Decomposition	9
2.4 Machine Learning	10
2.4.1 Artificial neural networks	10
3 Experimentation	11
3.1 Data sources	11
3.2 Initial experimentation	11
3.2.1 Linear and Ridge Regression	12
3.2.2 Polynomial Regression	14
3.2.3 Boosting Decision Tree Regression	16
3.2.4 Nearest Neighbors Regression	18
3.3 Advanced experimentation	21
3.3.1 Noise removal using seasonal decomposition	21
3.3.2 Pipelines	22
4 User interface	23
4.1 Qt	23
4.2 PyQt	23
4.2.1 Integrating with Matplotlib	23
4.2.2 Issues with PyQt	24
5 Documentation	25

5.1 How to use	25
5.1.1 Prerequisites	25
5.1.2 To run the software	25
5.1.3 Using the software	26
5.1.3.1 Graphical User Interface	27
5.2 Process of representative day model generation	28
6 Discussion	29
6.1 Challenges	29
6.2 Modelling	29
6.3 Decisions	31
6.4 Usage	31
6.5 Future improvement	32
7 Conclusion	33

II Abstract

This study in determining a representative day was founded in a fascination of smart buildings and interest in modelling data from such smart buildings. One of the current major issues on a global scale is high energy consumption and wasted energy. To combat these issues this project set out to experiment with using data modelling to find patterns in building data. The purpose of modelling the building data, is to use it to find areas in which reducing energy consumption is a possibility. Additionally the models can also be used to find irregularities in the energy consumption, that is to find places where the data does not seem to match the expectation.

Most of the experimentation was done through trial and error, as it also served as a method of learning more about statistical modelling and time series data.

The project resulted in several different models of varying complexity and accuracy, capable of modelling the expected power consumption throughout any given day, based on the data from a building. The results of the project could quite easily be implemented in industries with interest in data modelling, as well as be used for increasing energy efficiency, by using only the necessary energy for any given task, by locating where energy is wasted.

III Reading guide

This paper is meant to be read as a whole. To achieve this, the paper should be read in the order that it is presented, without skipping chapters. Due to the nature of the paper, there will be multiple references to different technologies, concepts and methods throughout it, these will be explained when encountered.

The report is written with the expectation that the reader has basic understanding of statistics. If the reader wishes to use the non-gui implementations, a basic understanding of Python is also expected.

IV Acknowledgements

We would like to express gratitude towards our supervisor Sanja Lazarova-Molnar, for her support and guidance throughout the project.

We would like to also express gratitude towards our co-supervisor Elena Markoska, also for support and guidance throughout the project as well as being very helpful when discussing our implementation.

1 Introduction

The purpose of this report is to investigate how data from smart buildings can be used to improve the systems in these buildings. As such the paper will elaborate on how data can be used to predict a standard or representative day, based on historical data from the building. The prediction of a given date can be done using multiple different statistical models, some of which will be presented. This predicted day can then be used to investigate anomalies in the building, such as sensor failure, building performance regressions or usage anomalies.

For most cases the predicted day will still rely on some real-time or predicted data, such as the expected outside temperature for the predicted day, as well as its variation over the given day. A more sophisticated model would probably be able to predict based on date, though that prediction would not vary from year to year with the current data sets.

The problem being addressed by the project it was very clearly a regression problem. As the major types of problems in statistical modelling are classification and regression problems, determining the type of problem was very important. Classification problems are characterized by trying to identify which set of categories a new observation belongs in. Regression problems are characterized by trying to analyze the relationship between variables. Given the two characterizations it is clear that the problem of predicting behavior of data is a regression problem, as it relates to analyzing relationships and correlations in the data and not in identifying where observations belong.

The reasons for working with the topics of the project stem from a fascination of smart buildings as well as an interest in data modelling. Smart buildings are, as the name may suggest, the buildings of the future and with the buildings becoming smarter and possibly more connected to the internet, maybe related to the Internet of Things, it is bound to be interesting. Doing a project with something that is bound to change the way many day to day things are perceived and done within next couple of decades definitely play a big part in the choice of topic. The interest in data modelling was initially more closely entwined with the idea of machine learning, but progressed during the project into a more general interest in data modelling and the challenges faced when working with data.

2 Research

As part of the project's goal to identify a representative day from the data, multiple different regression methods were explored. This was done because it was unknown what pattern the data would take on, and therefore what method would give the best result. Below is a short summary of some of the regression methods and how they work.

2.1 Background

Many concepts will be used throughout the report which might be unfamiliar to the reader. Some of these will be explained when they are used, but those that are used throughout the report can be found below. Additionally background information concerning other similar projects can also be found in this sub-chapter.

2.1.1 Data format

All the data used in the project is in the CSV format. CSV stands for "comma-separated values"¹ and is a simple data format commonly used for datasets. If the reader wishes to explore the raw data most spreadsheet editors can import CSV files or they can simply be read using a text editor. Be forewarned that some of the larger datasets included may cause slowdowns in some of the most common text editors.

2.1.2 Representative day

The entire goal of the project was to create a model which could, based on building data, generate a representative day, for that specific building. A representative day is in this case a model which, when given the mean temperature of the building at any given time throughout the day, can predict how the day would look. Additionally the representative day model also has to be able to receive time and temperature data outside of this range, and based on this data predict how a future day should look, in terms of power consumption throughout the day.

Representative days do not necessarily have to predict power consumption, but can predict any isolated type of data in the set.

2.1.3 Time series data

Time series data is data which is ordered by a timestamp, with each data point having its own timestamp. Often time series data is used when describing repeated measurement over a period of time, such as power usage for buildings.

¹ https://en.wikipedia.org/wiki/Comma-separated_values

2.1.4 Regression and regressors

Regression is the statistical technique used for estimating and modelling the relationships between different variables. The most simple type of regression is linear regression, which attempts to find the best possible straight line to describe the data. It is however very important to select the right regression method for the given problem, as not all kinds of data can be modeled in the same way, e.g. the data processed in this project cannot be accurately represented as a straight line, but other regression methods yield quite accurate representations.

A regressor is simply the algorithm used to perform the regression.

2.1.5 Related work

The following section will cover other projects which has worked with similar data or a similar objective.

2.1.5.1 Synthetic Data

Synthetic data has some similarities to the work that has been done as part of this project as it regards data that is not obtained via measurements but is created from measured data. Synthetic data however is useful in significantly different cases such as dealing with sensitive data where creating the synthetic data from the actual measurements helps in ensuring data anonymity. The main relation is that the methodologies used are at the very least somewhat similar. In many cases when creating synthetic data it may however be more of a classification problem than a regression problem if it is not strictly numerical data.²

2.1.5.2 Big Building Data

Big Building Data is a platform from the Smart Living Lab in Switzerland. Their work is mostly regarding automation of smart buildings and how to optimize the buildings based on usage. The projects being conducted by teams as part of Big Building Data are encouraged to share historical data and processing to avoid each team developing their own gathering and processing methods. Their main goals of relying on data to back the automation of modern smart buildings could result in models similar to ones developed as part of this project. As such the work done in this project could potentially be helpful to Big Building Data or vice versa.³

² <http://www.ijstr.org/final-print/mar2017/A-Review-Of-Synthetic-Data-Generation-Methods-For-Privacy-Preserving-Data-Publishing.pdf>

³ <https://www.sciencedirect.com/science/article/pii/S1876610217329582>

2.1.6 Importance

The use for building models which can analyse energy consumption has steadily been growing in importance across the world, since analysis of energy consumption would help building owners in reducing general energy consumption, along with reducing the energy waste. This both serves as a cost-saving measure for buildings, as well as a way to reduce global pollution.

The United States is faced with the problem that the energy productivity is very low, meaning that the output and efficiency of the energy used is very low. The United States is also one of the single largest consumers of energy worldwide and one of the biggest contributors to the increase of CO₂ in the world, even compared other highly developed countries, such as in Europe. As such it is important for the United States to reduce the waste of energy, by only using as much energy as necessary for any given task.⁴

In Denmark and Europe measures are also being taken to reduce CO₂ emissions and improve energy efficiency. The European Energy Efficiency Directive is currently in effect and is aiming for a reduction in energy consumption for the entire European Union of 20% by 2020 and 30% by 2030 compared to 2012.⁵

Currently Europe is lacking behind compared to the expected progress. The progress between 2012 and 2016 was on track and based on these results the commission decided to increase the goal to the 30% by 2030. In 2017 however the results were less than ideal, showing setbacks in some aspects. These setbacks from 2017 lead the European Commission to conclude that in order to reach the set goals, especially the 2020 goal, additional measures will have to be enacted.⁶

In the danish industry the interest in energy efficiency is on the rise. In the industry one of the main factors for the interest is the possibility of monetary savings yielded by the decrease in unnecessary energy consumption. In addition to the monetary savings the public relations improvement often yielded by "going green" has value for the danish companies and industry.⁷ Even without the direct advantages of engaging in supporting the energy efficiency directive, the industry might have it in their best interests to take part in order to avoid sanctions and regulations from the state or the European Union.

In order to facilitate a reduction in energy consumption and increase in energy efficiency it is important to know how the energy is being used and where there are possibilities of improvements. It is also important to have a metric which can be used to compared improvements to what the expected usage would have been without the improvement. This is the goal for the modelling to be done in the project with the hopes of creating useful software to help with these processes.

⁴ <https://www.mckinsey.com/business-functions/sustainability-and-resource-productivity/our-insights/wasted-energy-how-us-can-reach-its-energy-potential>

⁵ <https://ec.europa.eu/energy/en/topics/energy-efficiency/energy-efficiency-directive>

⁶ <http://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1511978095545&uri=COM:2017:687:FIN>

⁷ <https://di.dk/virksomhed/miljoe/energi/energieffektivisering/pages/energieffektivisering.aspx>

2.2 Simple models

The following sections will cover the simple models researched as part of the project. The ones described are linear ridge, boosting decision tree, and nearest neighbors regression. In addition to these simple linear regression and polynomial regression was also researched, but are not described due to their simplicity.

2.2.1 Linear Ridge Regression

Ridge Regression is a regression method that is useful when the data has a possibility of suffering from multicollinearity. Multicollinearity, or collinearity, occurs when there is a seemingly linear relationship between independent variables. Collinearity may lead to inaccurate regression coefficients, which in turn reduces the validity of the model. There are several possible causes for collinearity, one possible cause being that the data was not collected independently, another possible cause being how the model is defined and the choice of model, and lastly outliers with extreme values. When the collinearity is not otherwise avoidable ridge regression can be used to try to accommodate the colinearity.

Ridge regression introduces a small degree of bias to the regression, in order to reduce the standard errors, which should result in a more reliable model.

The first step in ridge regression is standardizing all variables, both dependent and independent. All calculations in ridge regression is based on the standardized values. Ridge regression proceeds like regular linear regression, until the variance in the data becomes too large, at which point a small value is added to the diagonal elements of the correlation matrix. This addition to the matrix is added to attempt to improve the final result of the regression.⁸

2.2.2 Boosting Decision Tree Regression

Boosting is the process of improving weak algorithms by using weighting to help the weak algorithms use the proper values for estimation and by combining the results from several weak algorithms to gain higher total accuracy. Adaptive boosting, AdaBoost, is done by fitting the regressor on the data and then fitting copies of the regressor with the same data, but adjusting the weighting of instances based on the error of the previous regression. The regression will as such become increasingly accurate, but also increasingly complex.

An example of a relatively weak regression algorithm the decision tree. The decision tree regression can be effectively boosted with AdaBoost. Decision tree regression works by applying a greedy search looking for the best fit at each stage from the top to the bottom of the tree.⁹

⁸ https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Ridge_Regression.pdf

⁹ <http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html>

2.2.3 Nearest Neighbors Regression

Nearest neighbors regression is based on pattern recognition and similarity measurements of distance functions. For the regression a number K of neighbors is used to find the average of the values of the neighbors.

To use nearest neighbor regression the dataset must be properly examined to find the optimal value for K . A low value for K will usually result in a lot of noise and overfitting unless it is a rather small dataset, larger values for K will reduce the noise but may result in underfitting given a too large K . The size of K must be somewhat proportional to the dataset.¹⁰

2.3 Advanced implementations

In addition to the regression methods described above advanced methods were also investigated, mainly for their viability. One of these methods were seasonal decomposition, which was found to most likely be viable and as such was investigated further and in the end implemented.

2.3.1 Seasonal Decomposition

Seasonal decomposition is a series of procedures used to describe the trend and seasonal factors in time series data. By decomposing data it is possible to estimate the seasonal effects and create seasonally adjusted values by removing the seasonal effects. This serves to create data that more accurately will represent actual trends and behavior described by the data. There are two structures for basic decomposition: additive and multiplicative.¹¹

The additive structure is useful when the seasonal variation is relatively constant. The multiplicative structure is useful when the seasonal variation increases over time. An important thing to note is that the multiplicative structure does not work if the data contains values of 0.

To create a seasonal decomposition is done by completing a set of steps:

1. Estimate the trend. This is usually done with smoothing procedures such as moving averages or polynomial regression.
2. Remove the trend from the data as follows:
 - a. Additive: $Observation - Trend$
 - b. Multiplicative: $Observation \div Trend$
3. Estimate the seasonal factors. This is most easily done by averaging the values of the data with the removed trend for the desired season.
4. Determine the residual data:
 - a. Additive: $Observation - Trend - Seasonal$
 - b. Multiplicative: $Observation \div (Trend \times Seasonal)$

¹⁰ <http://scikit-learn.org/stable/modules/neighbors.html>

¹¹ <https://newonlinecourses.science.psu.edu/stat510/node/69/>

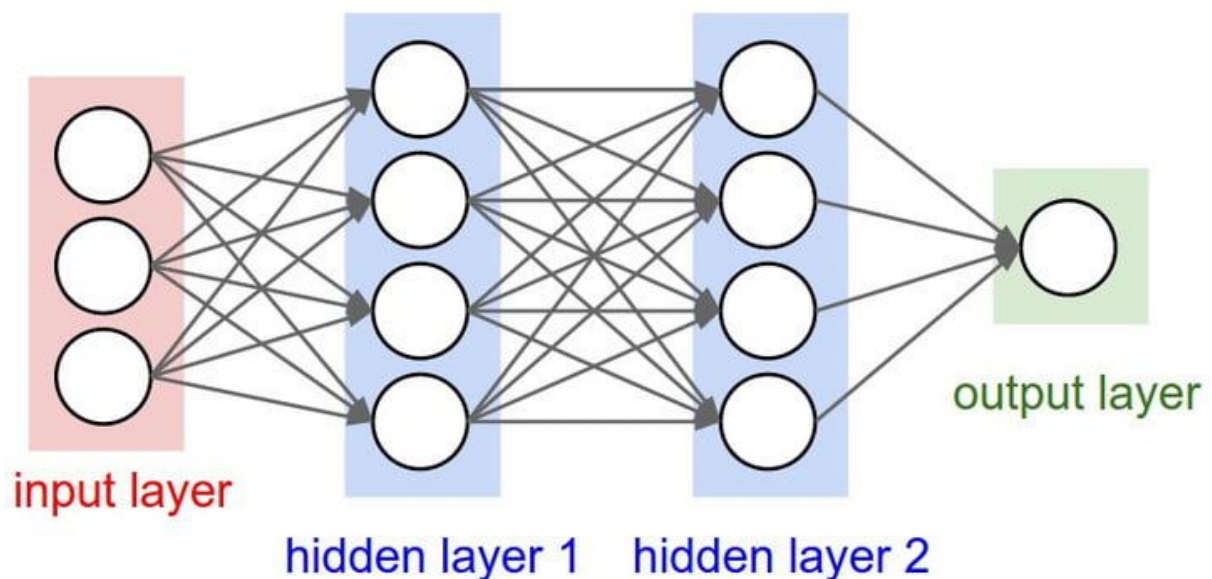
A seasonal decomposition can be done using local regression (LOESS) as well. When using LOESS the decomposition is essentially done the same way as the additive structure, the only difference being that LOESS is used for calculating the trend from which everything else is calculated. LOESS tends to create a smoother trendline due to working localized subsets of values and is less susceptible to having issues due to outliers.

2.4 Machine Learning

Machine learning a method of allowing computers to 'learn' by applying statistical techniques for progressively improving the results. Machine learning uses mathematical optimization with the statistical approach to continually improve recognition of patterns and fluctuations in data.

2.4.1 Artificial neural networks

Connectionist systems more commonly known as artificial neural networks are widely used in machine learning because of the structure and the ability to not be initially programmed to be very task specific, but having the artificial neurons 'learning' by testing the network and subsequently weighting the neurons and connections adjusting the system. Artificial neural networks usually comprise of several hidden layers of artificial neurons that communicate with each other in order to produce the best possible result.



A simplified drawing of an artificial neural network¹²

If the goal was to implement statistical regression modelling with artificial neurons one of the hidden layers might be model selection, another might parametric selection and the last possibly even several layers might be combining several models to gain the best combined model of the data.

¹² <https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/>

3 Experimentation

The following sections will cover the source of data used for experimentation in the project, as well as the implementations of the models used for the experiments. The initial experimentation section covers the simpler implementations and the advanced cover the later implementations.

3.1 Data sources

To conduct a statistical analysis on anything data is needed. Originally the plan was to obtain data from a local building, which had smart-building capabilities, and to use this data to both create and test the model. Unfortunately the data from that building was not available for the project to use and as such an alternative data source had to be found.

The final data source used was therefore a dataset made available by OpenEI¹³, which is an organisation which specializes in gathering data about energy usage in buildings. The dataset chosen was a set comprising of 11 buildings, each of which had power usage and temperature available for an entire year¹⁴. This dataset is also included in the data folder along with the source code.

Very little information is available about the type of building and its location, as the data is anonymized, but based on the data itself it can be inferred that the building is probably located in the United States of America, as the data was made available by the US department of energy. The title of the buildings also describe them as either retail or office buildings, and the temperature variation shows that the buildings are located in an area with a hotter climate.

3.2 Initial experimentation

Initially multiple different regression methods were tested, with each version either featuring a different pre-processing of data or a different model used to predict based on the data. Some of those versions will be presented in this chapter, with a focus on what advantages and disadvantage each implementation offered.

It is of note that all of the simple implementations in this subchapter only use a single value as the predictor, that is either time or outside temperature is used but not both at once.

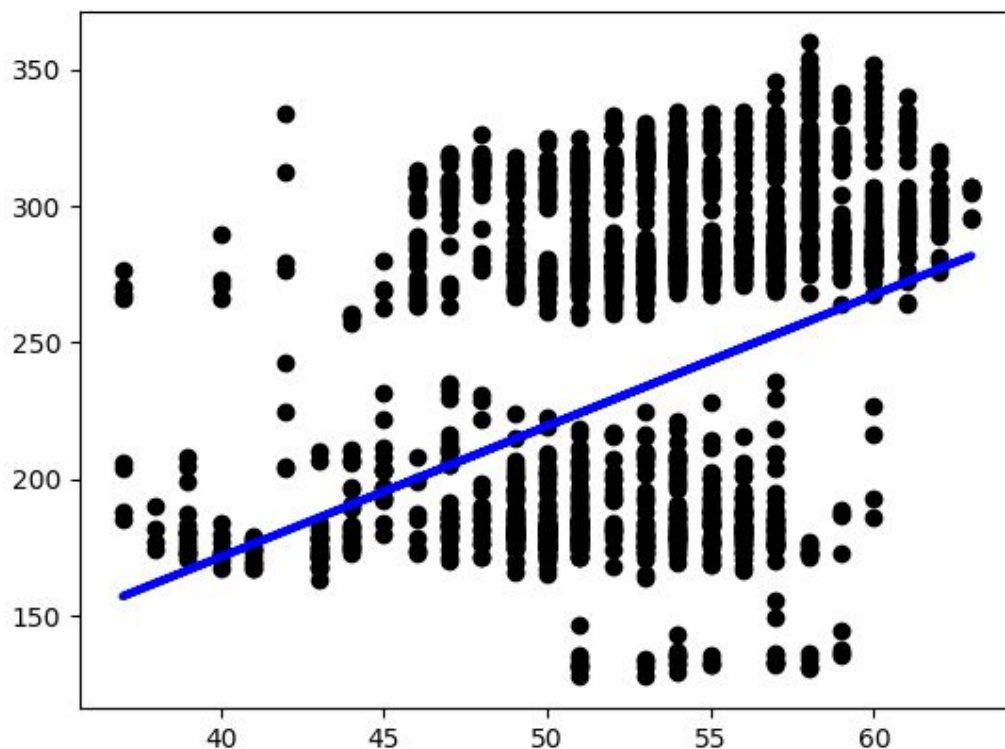
In the beginning and most of the simpler model implementations the loading and formatting of the data was done manually by loading the file with the NumPy package, stripping the date from the timestamp and formatting the data into arrays usable in modelling.

¹³ https://openei.org/wiki/Main_Page

¹⁴ <https://openei.org/datasets/dataset/consumption-outdoor-air-temperature-11-commercial-buildings>

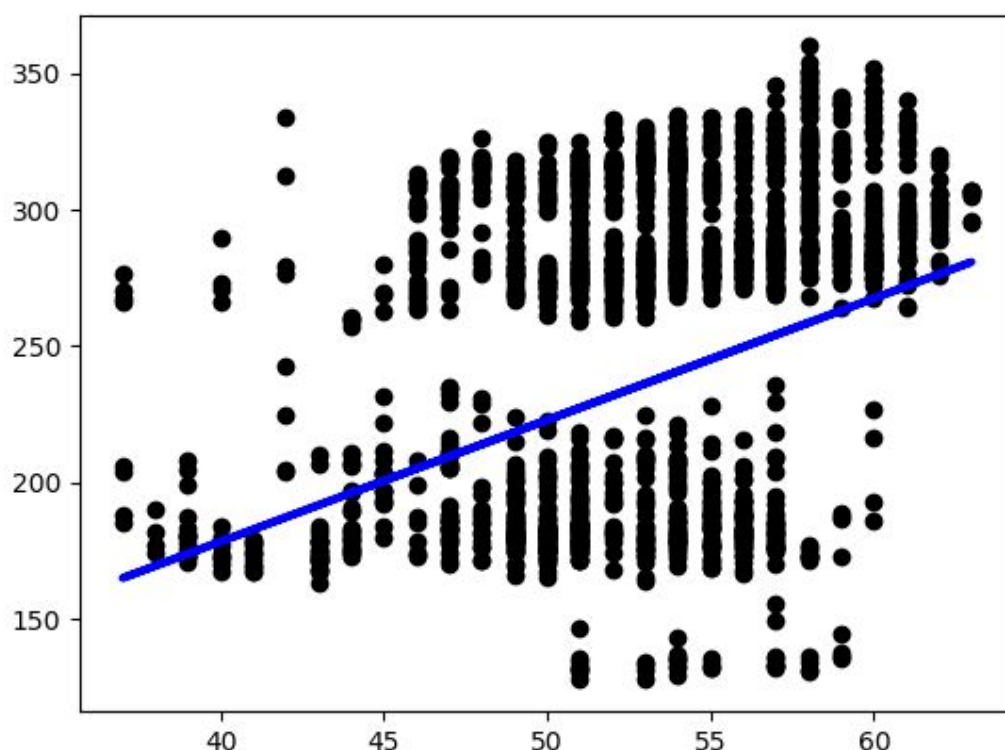
3.2.1 Linear and Ridge Regression

One of the first experiments was done with linear regression, comparing the outside air temperature to the power consumption. When performing this regression a positive trend was found. This trend suggests a positive relationship between the temperature and the power consumption, furthermore suggesting that the power consumption increases with the temperature. The issue with the linear regression is that it does not take the time of day into account and while it might be hot at night, the energy consumption will remain rather low as the building is not in use. This is only true for our specific data and not necessarily generally for buildings, but it is seemingly a general issue when using linear regression.

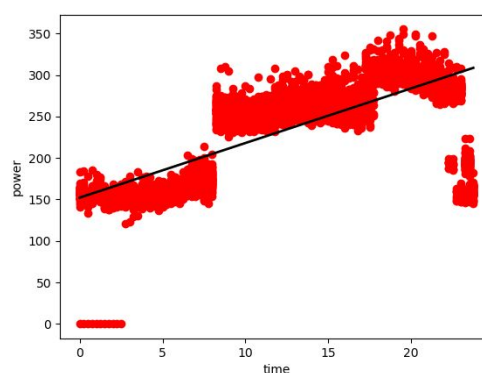
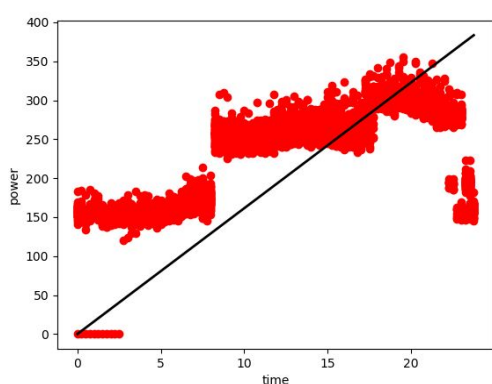


That results of linear regression are quite poor is also very clear in the graph. The R^2 score for the linear regression is 0.0061 which is very low, but slightly better than a straight horizontal line at the mean value.

Using linear ridge regression instead of simple linear regression yielded slightly better results with the R^2 score being 0.0359. The ridge regression is without a doubt better, as it does use more factors when calculating, but the results were still quite inaccurate. One of the main reasons for the inaccurate results of the ridge regression, is the fact that the data does not actually show signs of collinearity, an issue that was not discovered until later.



The linear ridge regression was also attempted on the data concerning the time and power consumption. This yielded at first odd looking results due to the `fit_intercept` parameter being set to false, which only works properly when the data is already centered, as the `fit_intercept` parameter determines whether SciKit should center the data, c.f. SciKit Learn documentation¹⁵. Having the `fit_intercept` set to false the R^2 score came to -2.169 , as the fit is actually worse than a straight line at mean value. The model with `fit_intercept=False` is shown below to the left.

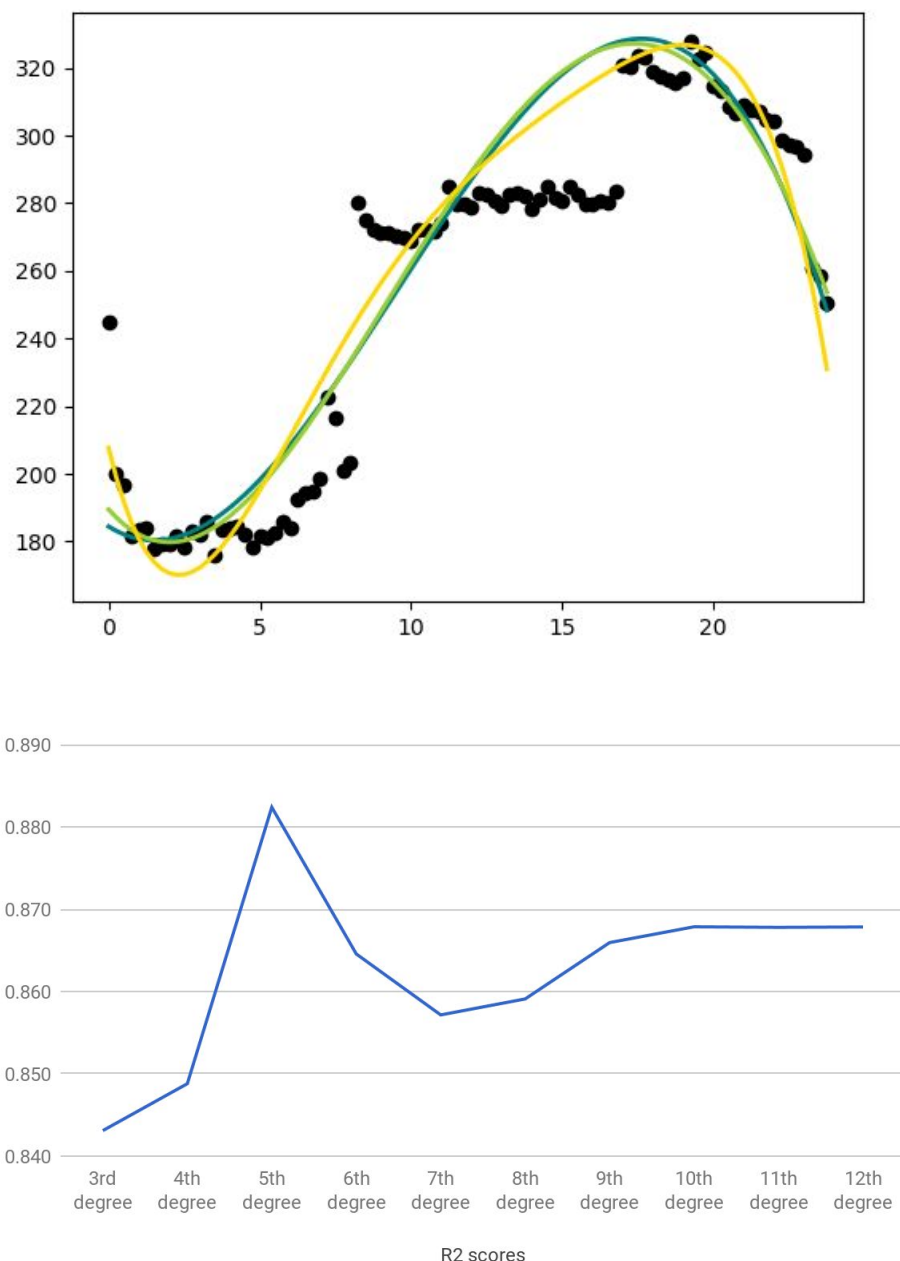


Using the `fit_intercept` to automatically center the values improved the model accuracy significantly, but still did not yield very accurate results, with an R^2 score of 0.484 . The model using `fit_intercept=True` is shown above to the right.

¹⁵ http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

3.2.2 Polynomial Regression

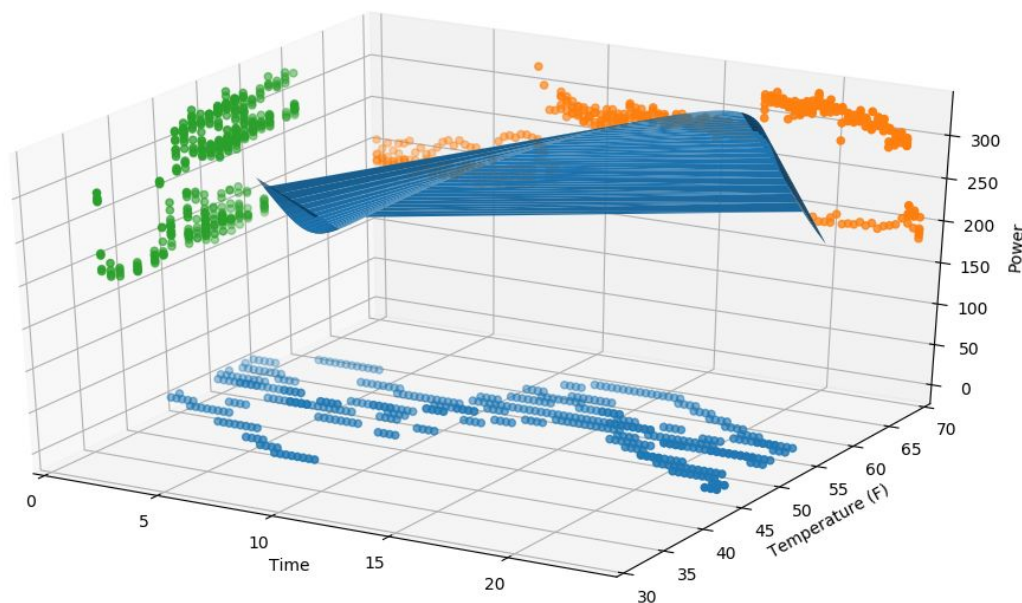
Polynomial regression is implemented using averaged data, to simplify data processing. Polynomial regression is a process of creating a n^{th} -degree polynomial function to describe the data. When creating polynomial functions the degree of the polynomial must be defined. The degree defines how many independent parts (constants) are used to describe it. For larger n -values (degrees) of a polynomial more complex data can be modelled. The simple implementation uses the degrees of three, four, and five to attempt modelling the averaged data. Further experimenting with the model of degrees of the polynomial have also been tried, but the best results were found with the fifth degree polynomial.



It is however important to note, that when the polynomial regression was performed on the building 1 data, using the SciKit Learn *PolynomialFeatures* package, a warning stating that the matrix is not conditioned well for polynomial regression was issued, when using degrees of sixth or higher. Additionally when increasing the degree of the

polynomial, it might lead to overfitting or finding relationships that are not actually present in the real world scenarios the data is describing.

The polynomial regression was further experimented with for modelling when introducing the third dimension. For the three dimensional implementation of polynomial regression all of the variables in the data was used, not just averages. To plot and model in three dimensions, a triangulated surface (trisurf) plot was created, using polynomial regression to model the relationship between the time and temperature on the y-axis and the relationship between the time and power consumption on the z-axis.



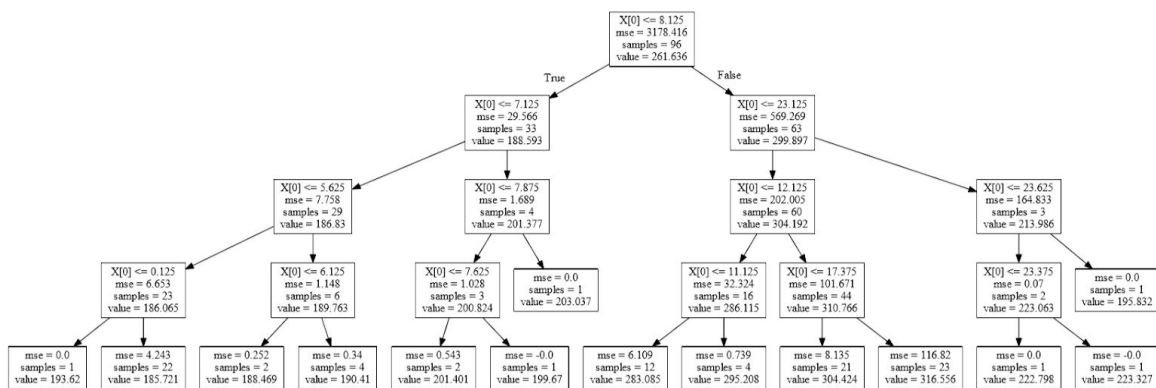
By creating a trisurf plot and model using the combined regressions, it is possible to the entire output space of the model, with the given data, making the three dimensional model potentially the most general. The trisurf is created by triangulating and creating surface areas between the points of the two separate polynomials and combining them into a single plot.

3.2.3 Boosting Decision Tree Regression

Boosting Decision Tree Regression (BDTR) is done using SciKit Learns built-in tree packages, as well as its ensemble packages. The tree package is used for the Decision tree regressor itself, while the ensemble package is used for the Adaboost regressor. The Adaboost then uses the decision tree regressor as its base transformer.¹⁶

In the specific code included a max decision tree depth of 4 is used, which means that the maximum amount of “sub decisions” that the tree is allowed to create is 4, from the base node. Adaboost is then applied to the tree regressor to attempt to make the final regressor more resistant to small changes in the input set.

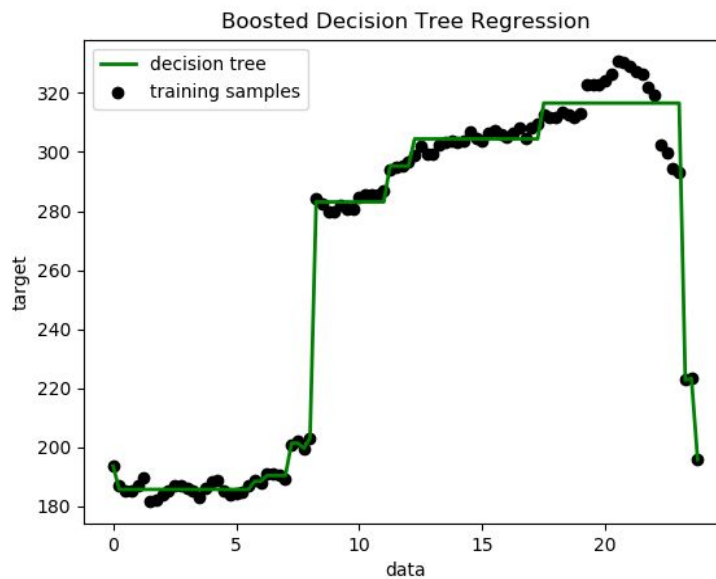
When using decision tree it is possible to export the tree created, by fitting the data, as a visual representation. This was done below for building 1, with a max depth of 4. The max depth was chosen via experimentation, as larger values tended to overfit the model.



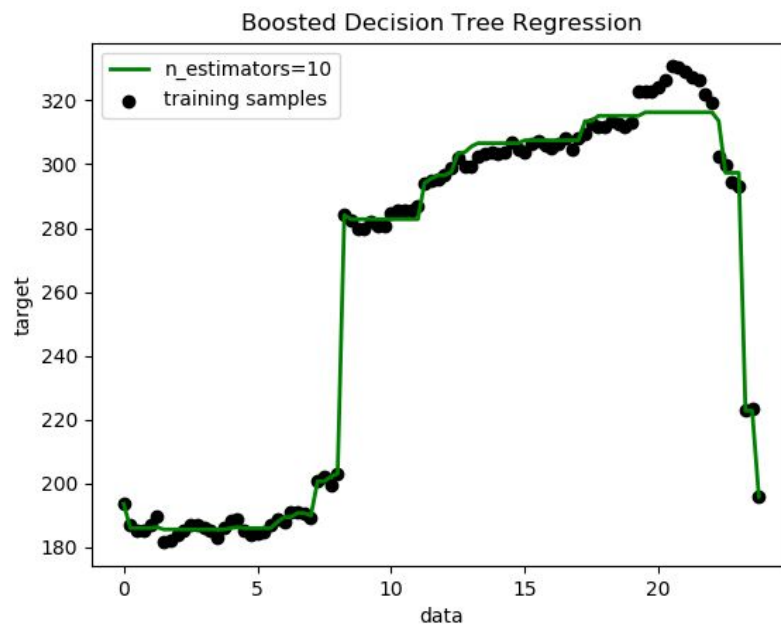
See appendix A for a larger version (decision-tree.pdf)

Using the above tree the following prediction was made, by fitting the model with the average temperature for any given time, and predicting based on a subset of data that was not included in the training data.

¹⁶ <http://scikit-learn.org/stable/modules/tree.html>



Adaboost was then applied to the model, with an estimator max count of 10, which resulted in the following model:



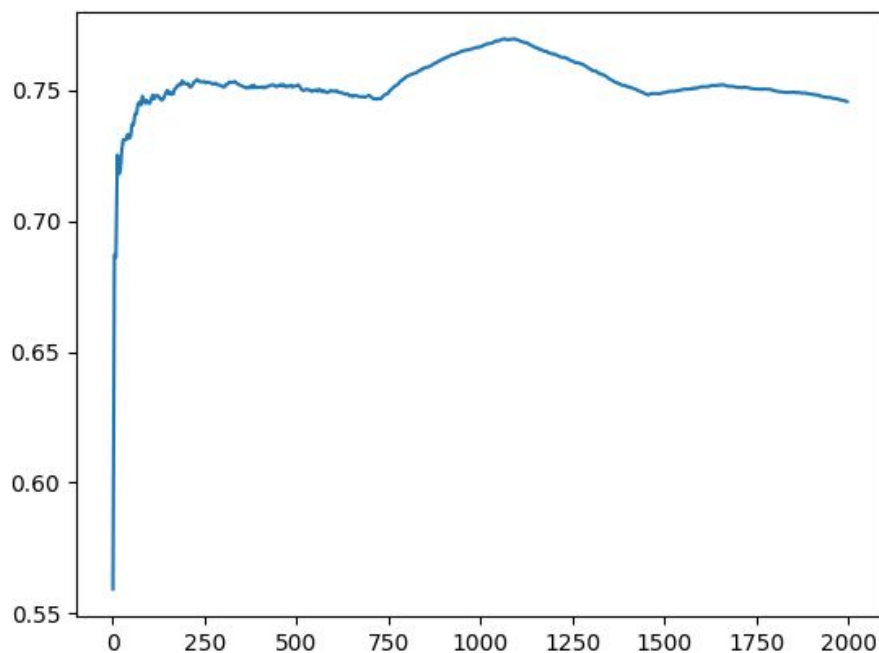
The new model shows significant improvements when looking at the finer details, while still preserving the same overall shape.

Decisions trees allow this to happen and can therefore be useful. Thanks to the decision tree graphs it is also possible to analyse the decision tree, which makes it possible to explore why the final prediction output is what it is, and why sometimes the prediction is incorrect. It should be noted that it is not possible to explore the ensemble made by Adaboost in the same way that it is possible to explore the decision tree.

3.2.4 Nearest Neighbors Regression

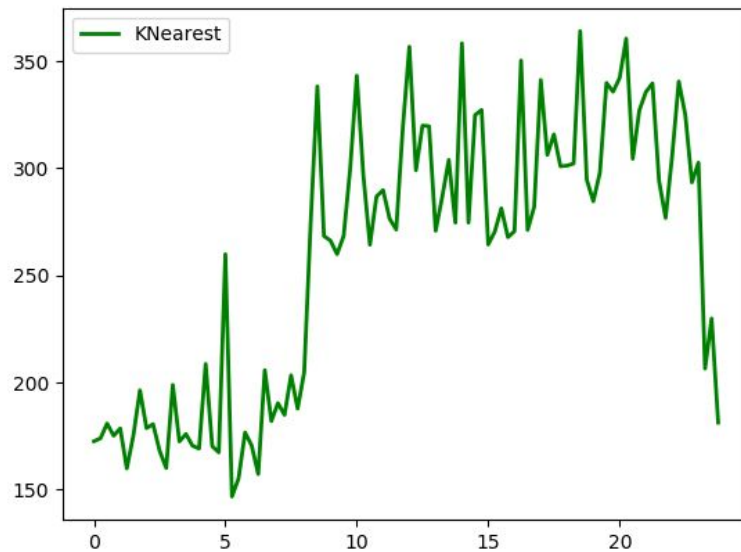
The nearest neighbors regression is done with the SciKit Learn package `KNeighborsRegressor`. The regression takes some parameters though most of them are only needed in very specific circumstances where certain data points are to be weighted more heavily than others or specific distance algorithms are to be used. For this project most of the parameters were left at the default values and only the number of neighbors were actively changed and passed in an attempt to create accurate models.¹⁷

The nearest neighbors implementation also includes a method for finding the best number of neighbors for the given dataset. This is done by using the SciKit Learn package `r2_score` which calculates the R^2 score or coefficient of determination a value usually between 0 and 1 describing how well the regression fits the testing data, 1 being perfect. The R^2 value can technically be negative, but will only do so when the model is worse than a horizontal line. For the simple implementation used the testing data is simply a small section of the data.

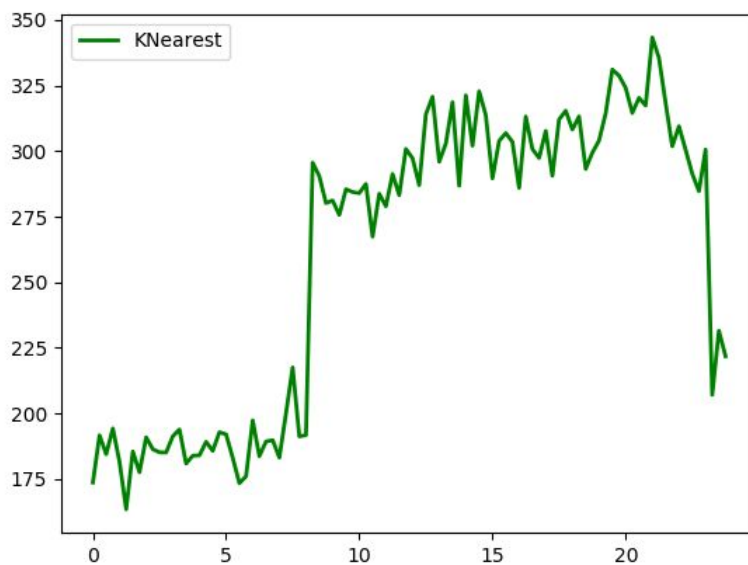


The y-axis describes the R^2 score for the given K value shown on the x-axis. The testing could possibly be improved with cross validation, running the test with every section of the dataset and averaging the scores.

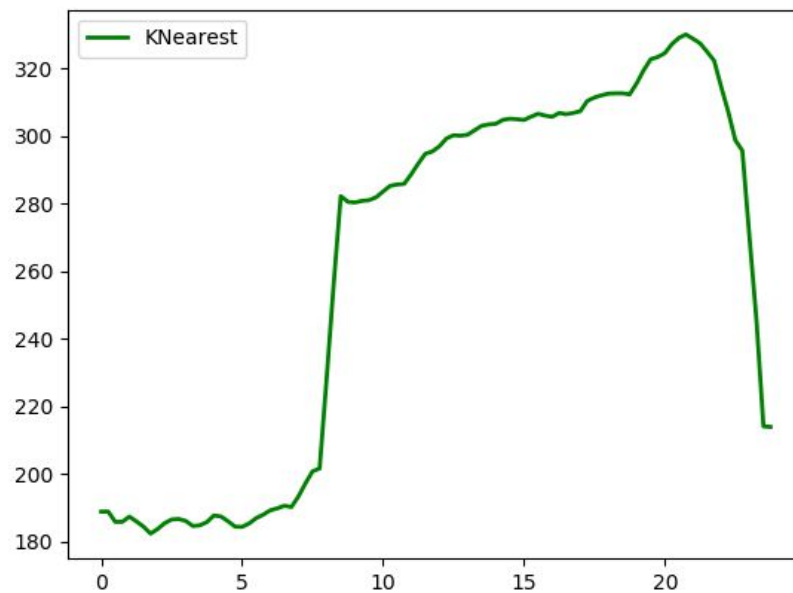
¹⁷ <http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html>



The image above shows the nearest neighbors regression run as greedily as possible with the number of neighbors (K) that is used for the calculation being 1. It is severely overfitted and shows no usable trend in the data as it is simply too erratic.



The image above shows the nearest neighbors regression run with a more reasonable number of neighbors used for calculations. In the graph above the number of neighbors used is 10, and the trend is more easily distinguishable than before. It is still overfitted due to the size of the dataset compared to the size of K and fluctuates too much to be usable as a general representative day.



In the last example shown above, the number of neighbors calculated for is 1080. The 1080 neighbors was the number approximated to yield the best results after doing the R^2 analysis. This model is quite accurate and although it still fluctuates more than the boosted decision trees, the model is a clear improvement. The downside of using such a large number of neighbors for calculations is that the execution time goes up, this is not a huge downside as the distance calculations used in nearest neighbors regression is quite cheap, but it is worth mentioning as it could prove troublesome for even larger datasets.

3.3 Advanced experimentation

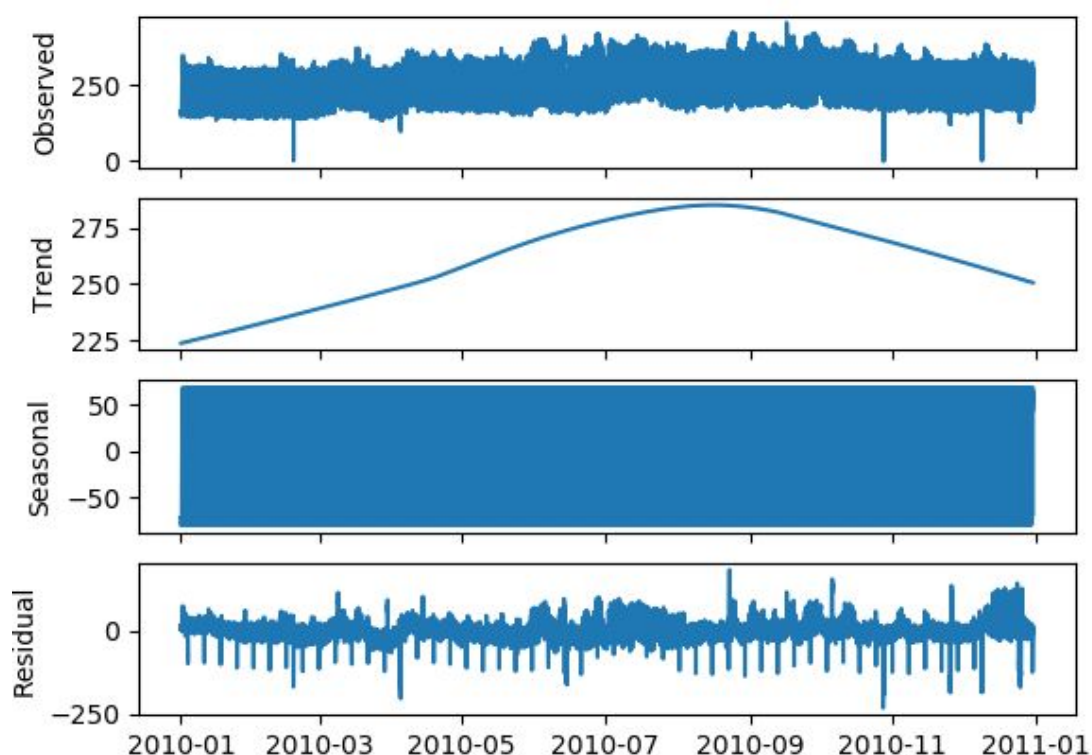
The advanced experimentation was mainly done by incorporating seasonal decomposition into the generation of the representative day, this is covered in the following sections.

Loading data was also an aspect improved upon in these implementations, as they use the pandas package, since pandas have the capabilities to read comma separated value files (.csv) and parse the data automatically.

3.3.1 Noise removal using seasonal decomposition

As previously explained seasonal decomposition splits the data into 3 parts, the trend, the seasonal and a residual. This feature of seasonal decomposition was used to filter the data before using it to train the developed model. This filtering was done by only training the model on the trend and seasonality that was extracted, thereby ignoring the residuals which contain the noise. Unfortunately the residuals can also contain data which is not actually noise.¹⁸

When decomposing a dataset it is possible to plot the data after decomposing it but before using it, an example of a decomposed power usage can be seen below. This decomposition was done using Loess method on the “building 1” dataset.



The data in the “Seasonal” section is unfortunately almost impossible to read visually, as there was too many densely packed data points.

¹⁸ http://www.statsmodels.org/dev/generated/statsmodels.tsa.seasonal.seasonal_decompose.html

In the implementation this works well, but there are multiple future improvements that can be made. One of such improvements is the fact that the decomposition right now relies on the input data having a set amount of data points per day, in this case 96. This means that any dataset which has more or fewer data points per day will not get decomposed properly, which in turn can lead to either more noise being included in the training data or to more of the relevant data being left out of the data. Determining the amount of data points per day should therefore in future version be done as part of the decomposition, based on the timestamps provided.

Another improvement is that multiple passes of seasonal decomposition can be made, to extract multiple seasonals with differing time spans, since currently the model assumes that the only seasonal pattern is a daily pattern. A weekly or even monthly pattern could also be extracted and used as part of training the model. The need for this can for example be seen by looking at the data for building 1 (See above figure) after decomposition, which has a clear weekly reduction in power usage at the end of each week.

3.3.2 Pipelines

To further improve the precision of the model a combination of models were used, chaining them together in a so-called “pipeline”. A pipeline is a collection of regressors and transformers in SciKit Learn¹⁹. By using a pipeline SciKit Learn ensures that all the models used are trained on the same data and in the same order.

In this case the regressors in the pipeline were a polynomial regressor and a decision tree regressor, with the decision tree being the final output from the pipeline. The main advantage of using a pipeline like this in our case was that it enabled us to eliminate odd values by using the polynomial regressor as a “smoothing” function, since most extreme variations would be eliminated by it.

¹⁹ <http://scikit-learn.org/stable/modules/pipeline.html>

4 User interface

The original plan for the project was to implement a web-interface for the model, which the user would then be able to access from anywhere with an internet connection. This proved to be troublesome in Python, and especially troublesome when trying to incorporate the plotting library used (Matplotlib). Therefore Qt was instead chosen as the GUI framework.

4.1 Qt

Qt is a C++ framework which focuses on cross-platform development. A large part of this framework is the GUI sub-framework, which is also a large part of Qt's popularity. Qt was chosen for the project because it is freely available and because there is a Python binding available for it, allowing us to interact with the Qt framework through Python code instead of through C++ code. Using Qt still requires C++ compilation/building tools installed, cf. the Documentation.²⁰

4.2 PyQt

PyQt is a community developed binding for Qt in Python, allowing the project to interact with Qt as if it were just a normal Python library. In the project PyQt5 was used, because it is the latest stable release of PyQt. PyQt5 is equivalent to Qt5, but as a Python binding for easy implementation.²¹

4.2.1 Integrating with Matplotlib

As part of the project it was important that the graphs that was shown to the user were not just static images, but something the user could interact with, even after the graph was shown to the user, and without having to re-generate it for each user interaction.

Fortunately Matplotlib actually includes a standard implementation for showing graphs in Qt, without having to export the graphs to an external format, allowing us to manipulate the graphs as Qt elements. Additionally it also includes a navigation tool which allows the user to interact with the graph using the Matplotlib navigation bar.

As such integration with Matplotlib turned out to be pretty straightforward when using Qt, compared to integration when using a web solution.

²⁰ <https://www.qt.io/>

²¹ <https://riverbankcomputing.com/software/pyqt/intro>

4.2.2 Issues with PyQt

There were some problems with PyQt which had to be worked around doing the development process.

One of the problems when interfacing with Qt through PyQt is that PyQt silently catches any exceptions thrown by the program, and instead of re-throwing or otherwise showing them to the user/developer of the program it simply exits the program with an error code. This error code is essentially meaningless and results in increased debugging time, as the valuable debug information otherwise contained in the exception is thrown away. It is of note that this issue seemed to mainly appear on Windows machines, as exceptions and stack traces were still properly shown on Linux machines.

To mitigate this issue a combination of the debugging breakpoints and test files, which did not use Qt at all, were used, though this was not optimal, as it required re-running the program and reloading the data each time.

Another issue was that the original plan to use the Qt designer to create the actual user interface (UI) turned out to be infeasible, as the designer had problems both in starting up and producing unusable UI files. As such the entire UI was designed manually in the code, using a simple automatic layout.

5 Documentation

The included software is a collection of experiments made throughout both the research and experimentation phase of the project. Also included is the final product, which is a GUI program, primarily made to show how the model works and how it could be used.

The project has been managed with version control via Github²². The most accurate and most user friendly implementation is the version with the graphical user interface, found in the "src/gui-program" folder, which is also the only implementation that is capable of loading different datasets during runtime. The other implementations are found in the "src/main" folder and are implemented as single run applications with all parameters defined in the implementation file. The parameters of the model in these implementations can only be changed in the source code and the same goes for the dataset used in the model.

5.1 How to use

Most of the included software requires the user to have some programming knowledge, to be effectively used, with the exception of the GUI program.

5.1.1 Prerequisites

- Python version 3.5+
- Python Pip
- Project files (Available from Github)

5.1.2 To run the software

To run the software included in the project some requirements needs to be fulfilled:

1. Install Visual C++ Build Tools [[Download](#)]
 - When installing check the optional boxes for C++/CLI support and VC++ 2015.3 v14.00 (v140) toolset for desktop
2. Install required pip packages using `pip install -r requirements.txt`
3. Run the appropriate python file
 - Some python scripts require the input file to be in a specific location

²² <https://github.com/fotoply/PowerConsumptionPredictor>

5.1.3 Using the software

How using the software works depends on which part of the software is to be used.

If the model and part of the software to be used is one of the simpler implementations without the graphical user interface everything is handled in the code. These can be run as is, but contain no interactivity except for viewing the graph created by the model. Editing the files is rather simple as everything happens in the `run()` method:

- Changing the data file:
 - Edit the path for the `loadDataFromCSV(**path**)`
- Changing modelling parameters:
 - Edit the parameters of the regressor
 - **Setting additional parameters and removing existing parameters is not recommended!**
- Add or remove a scatter plot of the data
 - To add, add the `plt.scatter(x_values, y_values)` to the code before the `plt.show()`
 - To remove, add a `#` in front of the `plt.scatter()` line

5.1.3.1 Graphical User Interface

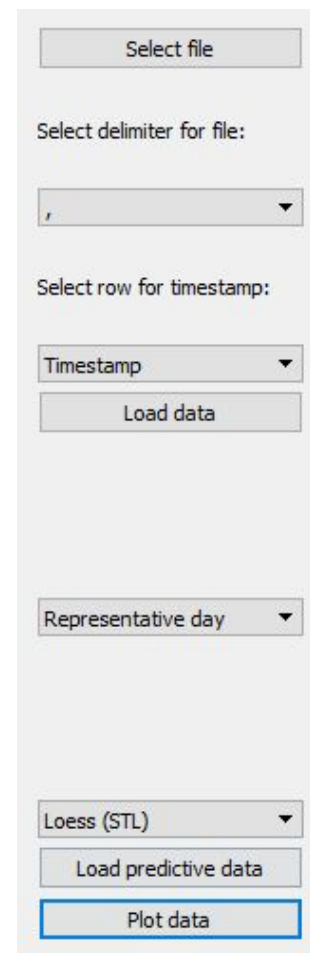
Using the graphical user interface (GUI) is significantly easier compared to modifying the code directly. Some preliminary experiments are not available in the GUI. The plots will be shown to the right of the menu shown.

To load a dataset simply click the [**Select file**] button and choose the file. Then select the correct delimiter and the correct column for the timestamp and click the [**Load data**] button. To model the data choose a model type from the dropdown menu and click the [**Plot data**] button.

For seasonal decomposition choose what to decompose for, Outside Air Temperature (OAT) or Power Usage (kW/6 kW) are the ones normally available in the provided datasets, and choose the method of decomposition. Additive and STL (Loess) decomposition works on any dataset, while multiplicative requires that the dataset does not contain any values that are zero.

For generating the representative day choose the method of decomposition, representative day as the model and then press the [**Plot data**] button. After the model has been created it is possible to load test data by clicking the [**Load predictive data**] button. This results in the model being applied to the series of outside air temperature data points and will result in a graph showing the expected behavior based on the data. If the actual values are available, denoted by the *-actual* file in the same folder, the actual data will be plotted as a scatterplot alongside the graph of the model. Additionally, several metrics will be calculated and shown in a small pop-up dialog.

To inspect any of the generated graphs more closely click the [**Explore**] button. This will open the graph in a separate window, which has zooming, saving and navigation of the graph, among other features.



The image shows a vertical menu of controls for the GUI. At the top is a button labeled "Select file". Below it is a label "Select delimiter for file:" followed by a dropdown menu showing a comma character. Next is a label "Select row for timestamp:" followed by a dropdown menu showing "Timestamp". Below that is a button labeled "Load data". Further down is a dropdown menu labeled "Representative day". Below that is a dropdown menu labeled "Loess (STL)". Below that is a button labeled "Load predictive data". At the bottom is a button labeled "Plot data", which is highlighted with a blue border.

5.2 Process of representative day model generation

The final process used to generate the representative day is a multi-step process which incorporates the experience that was acquired in the experimentation. One of the requirements for the model was that it had to be reasonably fast to train, either by allowing partial training, so that new data could be fitted continuously to the model or by simply having a low enough fitting time that fitting once per day would not cause issues even with large datasets.

First the program verifies that the user actually selected a file for the dataset. Then each of the data columns in the dataset is decomposed. A data column is any column which is not the timestamp column. From the decomposed data the trend and the seasonality is recombined, creating a simplified year, and the residuals are discarded. Additionally the mean of each column, throughout a single day, is calculated and saved. For each of the data columns the time of day, date and month is also extracted and saved as separate columns in each data table. This was done to isolate each part of the date, so that the model could properly account for each separately. It also allowed the model to exclude the year from its calculations, as a representative day is expected to not depend on the year, except if changes are made to the building itself.

The program then checks that there is exactly two data columns in the dataset, as the current model is only configured to handle two columns, but it could easily be expanded. It was not expanded during the project due to a lack of additional columns in the datasets to test the expanded model with. If there are only two data columns it proceeds to the training process.

For the training process a pipeline is created. The pipeline uses a polynomial regression first and a decision tree second, with the polynomial regression serving as a smoother for the incoming data. This pipeline is then fitted with the data extracted from the seasonal decomposition, where it assumes that the first data table is the y values and the second data column is the x values. It also drops any row which contains an NA value, as this indicates a row with missing values.

Then to generate the visuals shown to the user the first data table's mean values, which was extracted earlier, is used to predict a single day on this model. This model is then saved to the computer's memory and the general prediction is shown to the user.

6 Discussion

The overall purpose of this project was to be able to generate a representative day based on outside air temperature for use in data comparison. The software developed has takes a series of parameters of the outside air temperature as well a timestamp of when it was recorded. By statistically modelling historical data the model is capable of modelling the expected power consumption of the building for any given series of temperatures over the course of a day. The following sections will discuss the challenges faced as well as the models, and the decisions made during the project.

6.1 Challenges

From the very beginning of the project the plan was to work with time series data in order to create the models. Working with time series data proved to require quite substantial research just to get sufficient understanding of how time series data is structured and how to actually work with the different parametric values stored in the data.

The solution was programmed using the Python programming language. Python was not a language that the team was very familiar with beforehand. The previous experience was mostly in use with small scripting applications for very simple purposes such as tracking the time spent in specific applications and simple servers. Due to the unfamiliarity of python, a significant amount of time was spent at the beginning of the project on just getting used to python. Additionally the libraries that were used for modelling were new and required time to understand and use properly; especially Pandas and SciKit Learn that are integral in the modelling and Matplotlib that is used for plotting data. In an effort to make the final models more easily usable for anyone a graphical user interface was developed using PyQt. It should have been possible to create the user interface easily through a graphical designer, but this proved to be more troublesome than helpful and in the end the user interface was built by coding it. The interface is designed to be practical.

6.2 Modelling

In the beginning of the project the models that were researched and experimented with were very simple starting with linear and polynomial regression methods. This was done mainly to gain experience with modelling in Python and to learn how to handle the different variables available in the dataset. Both the linear and the polynomial models are not very accurate when dealing with time series data as they tend to have no way to account for the trend in the data, which naturally shifts the data as time changes.

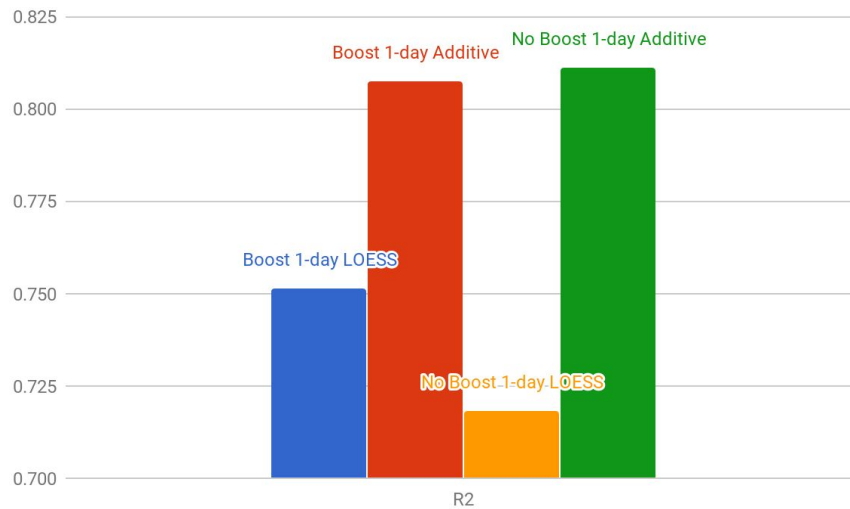
The models used over the course became increasingly complex and as such harder to implement properly, as well as computationally more expensive. The second phase of model experimentation used the Nearest Neighbors, Ridge, AdaBoost and Gaussian regression methods. These methods proved to be significantly more accurate and were therefore the ones that were expanded upon in the project.

The ridge regression method had only slightly more accurate results compared to the regular linear regression though it did fit slightly better. It was also unable to accurately predict extreme variation in the data. The gaussian regression was the slowest method tested in the entire project and would not complete processing on the computers we had available, if using more than 1000 samples. Using the gaussian regression it was however possible to obtain the p-value for the prediction, which made it possible to graphically display an area around the graph showing the confidence interval. Note that even though displaying the confidence interval is possible it was not very accurate due to the low sample size. The AdaBoost was used with decision tree regression and proved to return seemingly accurate models. One of the key features that made AdaBoost especially accurate was its ability to handle sudden variations, since it does not rely on sine curves. The initial implementation however has a few issues, since that it uses mean values instead of the entirety of the dataset.

The Nearest Neighbors regression was one of the methods that were initially brushed over very quickly as it seemed to overfit every time. Toward the end of the project additional analysis with the method was conducted, effectively proving that the number of neighbors taken into account when experimenting earlier was simply too low given the size of the dataset. This analysis was done by calculating R^2 value for every number of neighbors between 1 and 2000, on the data for building 1.

The actual modelling of the representative day with the more advanced implementations was done with pipelines, as mentioned in the experimentation section, with a polynomial function for smoothing and a decision tree for the regression. Fitting the pipelines with the decomposed data yielded very accurate models that managed to estimate the behavior very well. It is of note that some features, such as decreased consumption in weekends were not correctly modelled in most case. To prevent this multiple passes of decomposition would be necessary.

The pipelines could instead of the decision tree regressor have used the AdaBoost regressor, but this has proved to yield mostly insignificant improvements in most cases, in terms of model accuracy. The boosting generally improves the regression by LOESS the most, but still usually results in poorer results than the additive method. Depending on which metrics are used, and how the metrics are weighted, boosting might be better, but as always there are other factors to take into account as well. When using AdaBoost with the pipelines the regression takes a very long time, noticeably and significantly longer time to a point where the potential improvement in the model is not necessarily large enough for the extra time spent. Below is the boosting analysis for the R^2 metric for building 1 shown as an example, please refer to Appendix A for the rest of the analysis data.



R² analysis showing a histogram of the values (Higher is better)

6.3 Decisions

The decision to create the solution using Python was made in collaboration with the project supervisors. Python was as mentioned not something that was necessarily preferable, but as the supervisor mainly used Python and had previously used the libraries needed it was the optimal choice, as otherwise the supervisor would not be able to help with the code itself.. The main alternative to using Python would be using the R programming language. R is a programming language and environment specifically designed for statistical analysis. In review using R for many of the tasks might have been a better choice as it is built specifically for the purpose whereas Python relies on importing packages such as Pandas, but the downside would be that it would not have been possible to create a user interface that anyone could potentially use and understand.

In the early beginnings of the project there were aspirations to work with machine learning using TensorFlow. This initial idea was to work with the models first in a strictly statistical context and later expand upon the project by implementing the models with TensorFlow. This would allow the project the ability to continuously work with new parameters and improve the models with artificial neural networks. This was dropped early on due to the complexity of working with the models even from a purely statistical view, which was estimated to only get more complicated if machine learning was involved. This decision turned out to have been the sensible choice, as challenges faced later in the project would likely have been too difficult to handle if the project had been focused on machine learning and not just statistical modelling.

6.4 Usage

The main use for the software developed is for data comparison as it is useful to be able to compare the raw data from a smart building to calculated predictions based on historical data. An example of use for the software could be a company with a smart building in which improvements are scheduled. Data from the building before the improvements could be used to train the models which could then subsequently be

used to compare data gathered after the improvements to quantify whether or not the improvements actually yielded the desired outcomes.

The current implementations deal with only power consumption and outside air temperature, but could easily be modified to take other factors into account based on what kind of data is being gathered and the purpose of the comparison. The models themselves as well as the implementation do not rely on the specific features of the data, but the number of factors will need to be implemented correctly when modifying the models for other uses than the one dealt with in this project.

6.5 Future improvement

Future implementations of the solution could be improved by adding the ability to account for holidays, closing days and special event days by using a calendar. Another improvement would be the ability to give specific recommendations for the building based on trends observed in the data, in an automatic fashion. Giving recommendations will need to be specifically designed for the building types the models are being trained for, as well as having additional knowledge about the goal and what parameters are possible to tweak in order to help with the desired goal.

Increasing the number of different types of data gathered could help in improving the functionality greatly. To attain additional data types more sensors will be needed for the buildings the models are to be applied to, but the data such as occupancy status or when alarms are engaged and disengaged would prove very valuable in finding areas of potential savings.

Improvements to the model can also be gained by performing more in depth pre-processing of the data. The models currently include very little pre-processing. The simpler models include no pre-processing, except for averaging the data in certain circumstances. The final process for modelling uses the pandas method `dropna()` to remove rows in which one or more values are not available (NA), as well as using seasonal decomposition. Improving the pre-processing could include some outlier detection and possibly removal to ensure the generality of the model. Another aspect of pre-processing could be in normalization to take irregularities in the data into account.

Another major improvement that might make the solution better could be implementing the models in a machine learning context. Implementing it using machine learning would result in highly increased ability for the models to improve over time by introducing layers of artificial neural networks, which would be able to run continuously, helping in handling issues and other anomalies in real time data. Although introducing machine learning into the solution would yield benefits there are also challenges that will need to be addressed as a result. One of the main challenges is that for models, such as the ones used for the solution, the machine learning would largely be supervised by the new measurements which can result in the estimations following the trend of the data instead of modelling based on the historical data.

7 Conclusion

The project resulted in successful generation of a representative day. The process of becoming able to create the final models was one of trial and error, where some steps might have been redundant in retrospect, but as part of a process it surely resulted in greater understanding of statistical modelling, as well as a greater understanding of the dataset.

Significant amounts of time was spent on experimenting with different models that were not actually used for the representative day generation in the end. One of the prime examples was experimenting with ridge regression as the data is not show signs of collinearity, and without collinearity ridge regression cannot effectively be used. Had the research into ridge regression been more thorough, the actual experiment would not have been needed to find the issues. But it is still important to stress that many of the experiments, while unused, still served to explore the data and to explore which models might not work.

The main uses for the solution developed in the project is as previously mentioned fault detection, but it might also be useful for testing whether changes to a building, to improve its efficiency, actually served that purpose, by comparing projected data to actual data.. Another use for the results of the project is the use of the actual research into statistical modelling. The trial and error methods used for finding the models for the problem processed in the project is inherently inefficient, but the knowledge gained by it is not wasted. It is also impossible to avoid this trial and error method when the amount of pre-project knowledge on statistical modelling was low.

Whether the models created during the project can be actualized in helping the industry decrease the wasted energy is hard to say for a project spanning four months. Were the models to be tested in practice for a couple of years, data might show if it is actually helpful in modelling the energy consumption in order to understand it better. The alternative in regards to e.g. the European directive is to simply implement power saving measures, but the actual effect will be very hard to measure without reference data. Additionally modelling is sure to find the areas in which power savings will be most easily had, which is definitely a positive for whoever is trying to implement it.

Related to the project are the question of optimal models, and the model developed with pipelines show a lot of promise for practical use. The pipelines model also has the best results by comparison, but whether it is actually the best model has not been fully explored in this project, nor has it necessarily been fully optimized. The scope of this project was never to explore statistical modelling to find the best method, but another project might be able to use the data and headway created in this project to build on and possibly explore the optimization of models and determining the best model.

Appendix A

All elements in Appendix A is in the included zip file ("Appendix.zip"). For images see the "Images" subfolder and for code see the "Project" subfolder. For additional analysis data, see the "Analysis" subfolder.