Predicting the energy consumption of a commercial location using Machine Learning

Robert Luijendijk Jens de Waard

June 2016

Coach: Matthijs Spaan Client: Paul de Graaf

Preface

This report discusses the development of an application for benchmarking various machine learning models. This project was executed as our Bachelor Project.

We would like to thank Matthijs Spaan for coaching us through the project, and Paul de Graaf and the entire team at Peeeks for providing us with a fun and engaging place to work.

Summary

The supply and demand side of the energy grid must be equal at all times. In the past, it was usually up to the supply side to meet the demand by turning coal and gas stations on or off. Now that the supply side is switching to sustainable and less predictable energy sources, it falls to the demand side to manage their energy need.

By utilizing the variable energy need of consumers such as cold storage or ice halls, Peeeks can reduce the energy costs of their costumers. In order to succesfully reduce their costs, Peeeks needs to predict their energy use for the day in blocks of 15 minutes.

Peeeks uses machine learning to obtain these forecasts. In order to be able to tell which machine learning model produced the best results, Peeeks needed a framework for testing and benchmarking the various models.

Because our stint at Peeeks lasted only 10 weeks, we needed to develop an application that could easily be further developed by the developers at the company. To this end we analysed our code with Pylint. We also placed emphasis on the feedback from SIG. We considered a well-designed, extensible application to be more important than a multitude of incomprehensible functionalities.

The application comes with a suite of tests. These tests help show the intent of the code and, as we extended the application, ensured that we did not inadvertently break parts of the program.

Contents

Preface **i**

Summary **iii**

1 Introduction **1**

I Assignment and Requirements **3**

2 Background Information **5**
- 2.1 Dutch electricity network 5
- 2.2 The electricity market 5
- 2.3 Peeeks 6

3 Assignment **7**
- 3.1 Current situation 7
- 3.2 Application 7

4 Requirements **9**
- 4.1 Compare all algorithms 9
- 4.2 New datasets 9
- 4.3 Forecast windows 9
- 4.4 Different data 10
- 4.5 Metrics 10
- 4.6 Data Output 10

II Design **11**

5 Class Diagram **13**

6 Sequence Diagrams **15**
- 6.1 Printing a list of classes 15
- 6.2 Calling the program through the CLI 16

7 Metrics 17
7.1 Mean Error 17
7.2 Mean Absolute Error 18
7.3 Mean Squared Error 18
7.4 Mean Absolute Percentage Error 18
7.5 R2 score 19

8 What Wasn't Done 21
8.1 Database 21
8.2 Graphical User Interface 21

III Implementation 23

9 Program Structure 25
9.1 Revised class diagram 25

10 Using the Program 27
10.1 Installing algorithms 27
10.2 The Command Line Interface 28
10.2.1 Getting a list of installed algorithms 28
10.2.2 Running a benchmark 29
10.2.3 Adding new data 30

11 Provided Machine Learning Models 31
11.1 Ordinary Least Squares 31
11.2 Ridge Regression 31
11.3 Lasso 32
11.4 Gaussian Process 33
11.5 Support Vector Regression 34

12 Testing and Quality Assurance 35

13 Requirements Evaluation 37
13.1 Compare all algorithms 37
13.2 New datasets 37
13.3 Forecast windows 37
13.4 Different data 38
13.5 Metrics 38

14 SIG Feedback Evaluation 39
14.1 Evaluation 39
14.1.1 Code Duplication 39
14.1.2 Unit Interfacing 40

15 Libraries and Tools 41
15.1 NumPy 41

15.2 Pandas 41
15.3 scikit-learn 42
15.4 Matplotlib 42
15.5 Pylint 42
15.6 PyUnit 42

16 Conclusion **43**

17 Recommendations **45**
17.1 Graphical User Interface 45
17.2 Different datasets 45
17.3 Parameter optimalization 46

IV Appendices **47**

A Initial Project Description **49**

B SIG Feedback 1 **51**

C Info Sheet **53**
C.1 Members of the project team 54
C.2 Client 54
C.3 Coach 54
C.4 Contact 54
C.5 Location 54

D Research Report **55**

Chapter 1

Introduction

Machine learning is a powerful tool in forecasting values and making sense out of large quantities of data. It is a broad field and over the years many models have been developed, some more different than others. This means that not every model is as suitable for solving a given problem. Finding the right model can often be challenging.

Peeeks uses machine learning to try and forecast the energy consumption of their assets. After using a Gaussian Process for their prototype, they are now looking at other models to see if those might perform better. This report is about the development of a framework aimed at comparing such machine learning models.

The report consists of three parts. The first part describes our assignment and the requirements put forth by the client. The second details our approach and design of the application. Finally, the third part has contains our work and recommendations for how the application can be further developed.

Part 1 consists of chapters 2 through 4. Chapter 2 contains more information about Peeeks. It explains the market they operate in and some information to help understand their business model. Chapter 3 is about the assigment. What problem did we try to solve and why is there a problem at all. Chapter 4 lists the requirements put forth by Peeeks.

Part 2 is Design. Through several UML diagrams we describe how we intended to write a program that was able to fulfill all the requirements from 4. Chapters 5 and 6 contain class and sequence diagrams respectively. Chapter 7 explains a few of the metrics that we planned to use to compare machine learning algorithms. We also touch on some designs one might reasonable expect, and why they're missing, in chapter 8.

Finally, the third part of the report is Implementation. In this chapter, we contrast the design of the program with what was actually implemented. To retain this symmetry, chapter 9 starts of with an updated class diagram reflecting the state of the finished product. It highlights the changes and offers some insights on why they were necessary. Chapter 10 is able to serve as a manual of sorts for the program. The key features are explained and their usage is shown.

The program would not be able to function without installed algorithms. We provide a few different machine learning models, which are explained in chapter 11. Each of these algorithms differ slightly and each has a different application.

Chapter 13 touches on each of the requirements laid forth in chapter 4 and if we feel they were implemented satisfactory. The program is built upon several open source libraries, which are explained in chapter 15. We explain what purpose the library fills and which parts we used for our project.

Part I

Assignment and Requirements

Chapter 2

Background Information

In this chapter we will explain how the electricity market in the Netherlands functions and how Peeeks relates to it. The background information shall cover the Dutch electricity network and the basics in power exchange.

2.1 Dutch electricity network

It is clear that society is highly reliant on a functioning electrical grid. The electricity grid in the Netherlands (and, in fact, most of the world) is designed to operate on a frequency of 50 Hz, called the utility frequency[7]. Appliances designed to operate on this frequency will not function efficiently or safely if used on another frequency. This means that the network needs to maintain this frequency at all times.

The frequency changes depending on the amount of power supplied to and subtracted from the network. Effectively, this means that supply must always equal demand. The party responsible for ensuring this is called the Transmission System Operator (TSO). In the case of the Netherlands, the TSO is TenneT.

2.2 The electricity market

There are several ways the balance of the grid is influenced. The first of these is the APX, the Amsterdam Power Exchange[3]. The APX is a Day-Ahead auction, which means that trading takes place on one day for the delivery of electricity the next day. It is also a blind auction. This means that neither buyers nor sellers can see the offers that are being made. After all offers are made, the supply and demand is compared and the APX calculates the price per hour.

Each party connected to the grid (also called a Balance Responsible Party, or BRP for short) has to notify TenneT of their intended energy usage or supply[8]. This is called an E-Programme, and it details the predicted energy consumption or generation per 15 minutes. TenneT verifies these E-Programmes and make sure they are consistent.

The E-Programmes are predictions and this means that the BRPs often deviate from them. This results in an imbalance on the grid. Because these imbalances lead to a change in utility frequency, TenneT estabilishes a price for deviating from the E-Programme[10]. This price can be positive or negative, in which case BRPs are payed for consuming power. Through this imbalance market, TenneT seeks to restore balance to the grid.

Aside from these markets, TenneT also has contracts with suppliers for 'emergency power'[9]. When the grid is very imbalanced, they are called upon to supply power or reduce consumption as a final resort.

2.3 Peeeks

Peeeks seeks to utilize the 'flex' in consumers to capitilize on the imbalance market[6]. They intend to work together with these partners, who have flexibility in their power usage. With Peeeks managing their installations to move power consumption to when the price is low, or negative, they can reduce the cost of electricity by 10 to 40%.

This has the side effect of enabling the grid to utilize more sustainable energy. Traditionally, balance was mostly attained by lowering production. Wind and solar powerplants are less predictable than fossil fuels in energy production. If it is especially windy, the windmills will produce a lot of power, but this can drop suddenly if the weather changes.

Chapter 3

Assignment

3.1 Current situation

Peeeks is a BRP and, as explained in chapter 2, needs to send TenneT an e-programma every day. To this end, they need to be able to predict the energy consumption of their assets. While developing their original forecasting application, they settled on a Gaussian process to model the energy consumption. An implementation of a Gaussian process was provided by sk-learn, and it appeared to work reasonably well.

3.2 Application

Peeeks decided to see if other machine learning models would be able to better predict the energy consumption of their assets. Other models might be computed faster, or give beter results. Rather than checking a lot of different algorithms by hand, they wanted a framework that would help test and evaluate these algorithms.

Appendix A contains the project description that was posted on BepSys. It also contains some background information that we covered in chapter 2 and a couple of requirements that are further elaborated on in chapter 4.

A large part of the project was the extensibility of the program. It should be easy to add new algorithms and datasets to the framework. To summarize, the goal of the project is to **construct an extensible application that will compare multiple algorithms on a dataset and display the outcome.**

Chapter 4

Requirements

The product manager provided us with a few requirements that the program needed to meet. These requirements evolved during the course of the project and are listed below.

4.1 Compare all algorithms

The program should be able to run all installed algorithms on a dataset. The results of this run should be aggregated and shown.

4.2 New datasets

Datasets lie at the heart of the application. It should be easy to add new datasets to the framework, with minimal effort on the part of the user.

We can assume that all datasets are presented in the same way. They are all pickled pandas dataframes with a DateTimeIndex and a single column containing the energy consumption of an asset.

4.3 Forecast windows

We cannot assume the forecast window, the amount of days between the last day in the dataset and the day to be predicted, is always a single day. For example, if today is wednesday, we need to forecast the consumption for thursday. We may have all the data up to tuesday, or only until last sunday. This changes the forecast window from 1 to 3.

4.4 Different data

While they currently predict the energy consumption solely on past consumption, Peeeks plans on eventually incorporated other types of data into the forecast. The framework will need to be able to handle multiple kinds of data, preferably without rewriting the code. Combining different types of data may lead to a more accurate forecast. For example, it is not unreasonable to assume a warm day will lead to increased energy consumption of an ice skating rink.

4.5 Metrics

To find out how well the algorithms are performing on the dataset, the program needs to use metrics like the mean squared error, mean average percentage error, mean absolute error and mean error. The application needs to return these metrics at the end of a run from the training of the algorithms on the datasets.

4.6 Data Output

The application needs to write the results of a benchmark to a comma separated value (.csv) file. This will allow a user to easily import the results into a spreadsheet program.

Part II

Design

Chapter 5

Class Diagram

At the beginning of the project we created a class diagram, shown in figure 5.1. This helped visualize the complexity and scope of the project. Describing the relations between the components of the application also helped decrease coupling between them. We did not include every method and attribute of every class. For example, we felt getters and setters would dilute the clarity of the diagram.

The CommandLineInterface class would be the entrypoint of the program. A Graphical User Interface was not planned, but if implemented, would replace this class. The CommandLineInterface class delegated the parsing of the arguments to the ArgParser class, which defines which arguments are required, which are optional, and what the default values are.

We planned on creating a plugin-like system for adding algorithms and datasets. The user of the program should easily be able to add new data and algorithms by simply placing Python modules in a specified folder. The ClassList class is responsible for scanning a folder and returning a list of classes present in the Python modules in that folder.

These classes could then be used by Trainer to train the algorithm on the specified dataset.

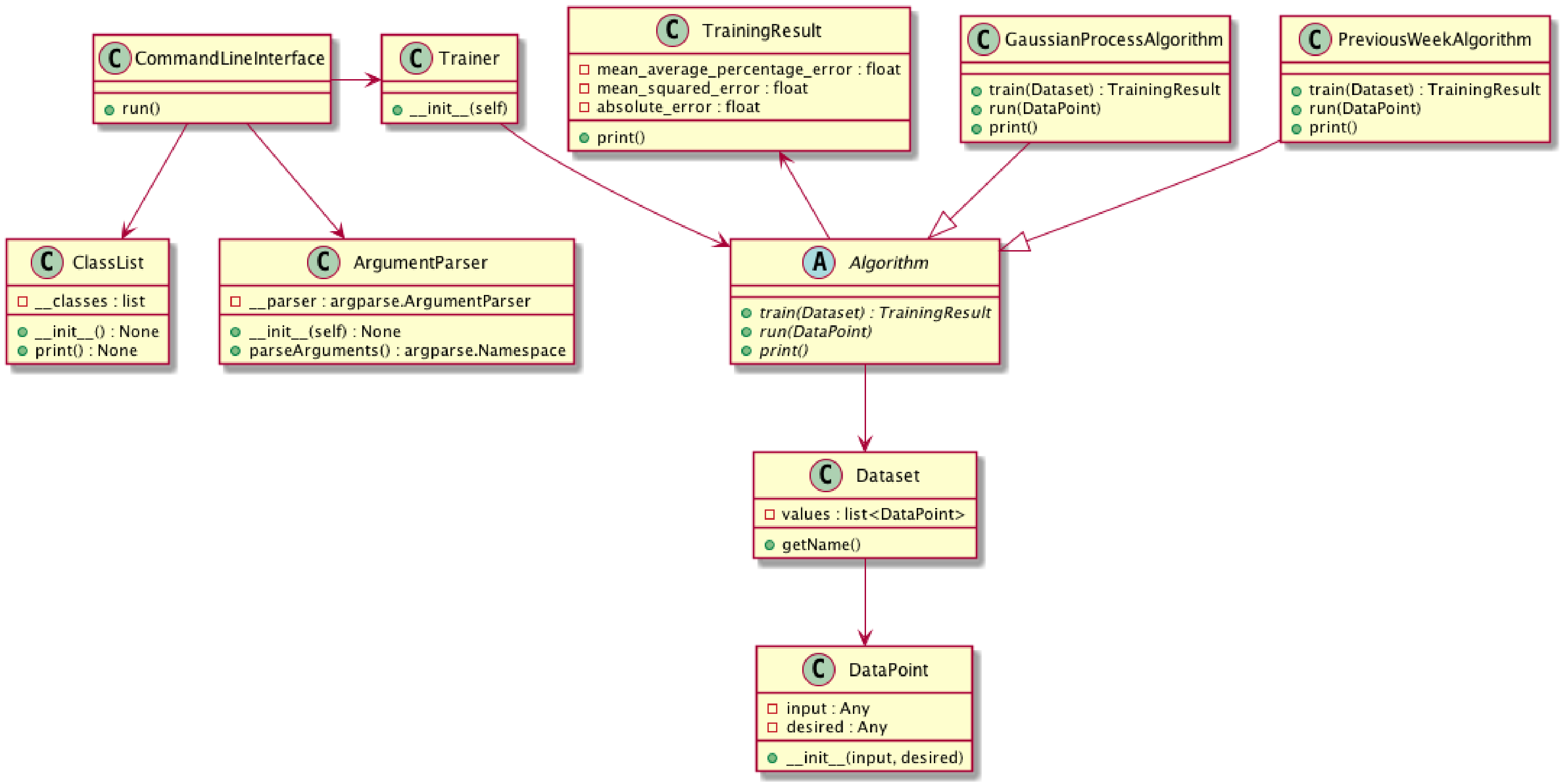


Figure 5.1: The initial class diagram

Chapter 6

Sequence Diagrams

To illustrate our thoughts of how the program would work, we drew up some sequence diagrams. They show the flow of execution through the program and the sequence in which the different components interact which each other[1].

Creating sequence diagrams before actually writing code allows us to see if the program is unneccessarily convoluted. As with class diagrams, time spent designing the system will save time in the development phase.

6.1 Printing a list of classes

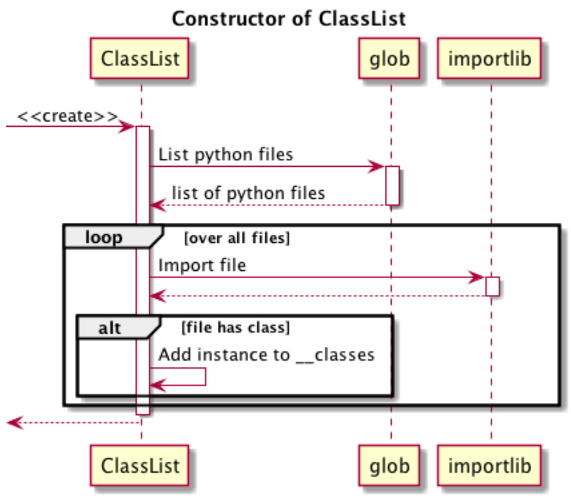


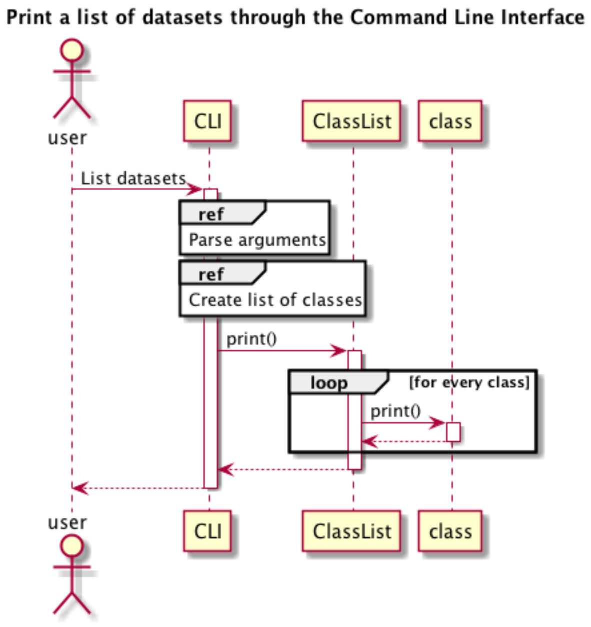
Figure 6.1: A Sequence diagram for printing a list of classes

A large part of the functionality of our program hinged on being able to scan a folder for new algorithms and datasets, so this was one of the first things we designed. Figure 6.1 shows how we planned on writing the program.

When a new ClassList instance is instantiated, it is passed as a path to a folder. It then asks glob, a built-in Python library, for a list of Python files in that directory. Those files, or ”modules”, are then imported. The script checks for classes in those files and adds them to a list it maintains.

6.2 Calling the program through the CLI

Figure 6.2: A sequence diagram for the command line interface



The sequence diagram in figure 6.2 illustrates how we intended the program to be called. The sequence diagram shows the program being asked to print a list of datasets. After using the built-in python library argparse to recognize this command, it uses ClassList to get a list of dataset classes. It calls the function print on every class in this list, resulting in the requested output.

Chapter 7

Metrics

There are several performance metrics that can be used to determine the effectiveness of an algorithm. Our program shows several metrics after running a benchmark. This chapter will give a few examples of such metrics, show how they are calculated and how they relate to each other.

7.1 Mean Error

The mean error (ME) is one of the simplest accuracy determination of an algorithm that has performed on a dataset. The formula is as follows:

$$ME = \frac{1}{n} \sum_{i=0}^{n} (\hat{X}_i - X_i) \tag{7.1}$$

where $\hat{X}_i$ is the predicted value, X_i is the actual value and n the number of samples. This metric weights all errors equally. Unfortunately, errors with opposite signs cancel each other out. This may not be undesirable in every situation. However, underestimating and overestimating the energy consumption will not make the cost of energy cancel out, making this metric unsuitable for this specific application.

7.2 Mean Absolute Error

The Mean Absolute Error (MAE) overcomes this problem with the ME by taking the absolute difference between forecast and the actual value.

$$MAE = \frac{1}{n}\sum_{i=0}^{n}|\hat{X}_i - X_i| \tag{7.2}$$

Positve and negative errors no longer cancel each other out. The MAE treats all errors equally, whether they are larger or small, positive or negative. However, not all situations consider an error of 4 to be twice as bad as an error of 2.

7.3 Mean Squared Error

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(\hat{X}_i - X_i)^2 \tag{7.3}$$

The Mean Squared Error, as the name implies, squares the error. Like with the Mean Absolute Error, this means that positive errors won't cancel out negative errors. Additionally, the Mean Squared Error rates higher errors more severely. This is especially useful for situations where the cost of an error does not scale linearly with that error. For example, if TenneT were to fine higher deviations from the E-Programme more severly.

7.4 Mean Absolute Percentage Error

The Mean Absolute Percentage Error, or MAPE, is one of the most widely used performance metrics. All earlier mentioned metrics show the performance in an absolute number. However, this number is hard to interpret without information about the dataset. An error of 10 without context tells you very little.

The MAPE seeks to solve this problem by showing the error as a percentage of the actual value.

$$MAPE = \frac{1}{n}\sum_{i=0}^{n}|\frac{X_i - \hat{X}_i}{X_i}| \tag{7.4}$$

The metric is not perfect however. Because of the division by X_i it is undefined where $X_i = 0$. It is also not symmetric, as negative errors get a higher penalty than positive errors. Several metrics that overcome this have been proposed in literature.

7.5 R2 score

The r2 score is also know as the coefficient of determination. It often takes a value between 0 and 1, but when non-linear functies are fitted it can also take on negative values. These values can be arbitrarily large.

$$[label = eq:r2]R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{7.5}$$

$$SS_{tot} = \sum_i (y_i - \bar{y})^2 \tag{7.6}$$

$$SS_{res} = \sum_i e_i^2 = \sum_i (y_i - f_i)^2 \tag{7.7}$$

The R2 score is calculated as shown in equation **??**. Here, SS_{res} is the residual sum of squares, or the error between the predicted and actual values squared. SS_{tot} is called the explained sum of squares, which is the sum of the distances between every value and the mean of those values squared.

Chapter 8

What Wasn't Done

The previous chapters did not contain some designs that one might reasonably expect. There were no designs for a database schema or mockups for graphical user interfaces. In this chapter, we will explain our decision to leave these out.

8.1 Database

We decided against using a database to save the results of the program. While some benefit could be had from saving old benchmark results, the frequency with which algorithms and especially datasets would change would mean the old results would quickly become obsolete.

Saving the results in a SQL database would quickly increase the complexity of the program with little benefits. Having to setup a MySQL server would also introduce more dependencies. We briefly considered a built-in solution like SQL, but this was also not worth the additional complexity of the code.

8.2 Graphical User Interface

In one of the first meetings with the product owner, we discovered that they preferred functionality over a graphical user interface (GUI). The end users of the program would be the development team at Peeeks and they are all proficient at using the command line. We designated a GUI as a Could Have and did not create a design.

Part III

Implementation

Chapter 9

Program Structure

The previous part described how the program was designed on the onset of the project. While developing the program, we learned a lot and revised this design. This chapter details the final result and highlights the most interesting changes.

9.1 Revised class diagram

Figure 9.1 shows the class diagram for the finished product. There are more classes than in the original diagram in figure 5.1. This is mainly because we added additional functionality and is not so much the result of updated design.

The TrainingResult object was replaced by BaseErrorReport and Combined-ErrorReport. These classes group the benchmarking results of an algorithm. Additionally, CombinedErrorReport keeps track of the best performing algorithm so far.

The Trainer class was not implemented, as there are only a few lines of code required to train an algorithm on a dataset, and these were placed in the Algorithm class.

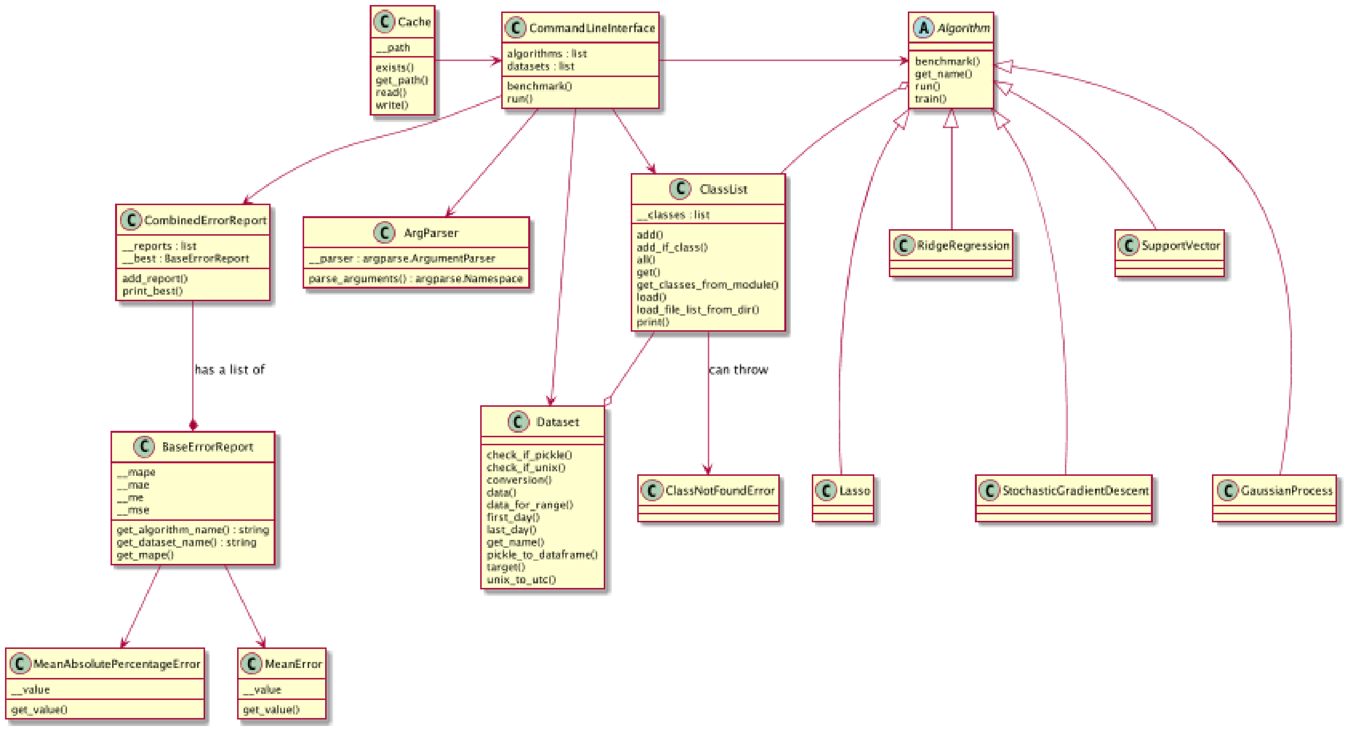


Figure 9.1: Class diagram of the final product

Chapter 10

Using the Program

The goal of the project was to create a program that can aid in finding the best algorithm for predicting energy usage. This chapter will explain how to use the program to this end.

10.1 Installing algorithms

As was required, it is easy to add new algorithms and datasets to the system. The program comes with an `algorithms/` folder that already has six algorithms in it. All these algorithms are subclasses of the abstract `Algorithm` class. Adding new algorithms is as easy as subclassing that class and placing the subclass in the `algorithms/` folder.

An algorithm requires only two methods to be implemented. `get_name` should return a string that uniquely identifies that algorithm. The other, `get_process` should return a object representing the way the algorithm predicts future values. If no other methods are overriden, this object should have fit() and predict() methods. All machine learning algorithms provided by Scikit Learn provide these methods.

The abstract class also has `train`, `run` and `benchmark` methods. Overriding these allows for greater flexibility in the algorithms that can be installed. Listing 10.1 gives an example of how to add a new algorithm. A file containing this class need only be placed in the `algorithms/` folder to be found by the system. Note that the example does not contain the `benchmark` method. A real algorithm should either implement `get_process` to return an object implementing `fit` and `predict` or provide an implementation of `benchmark` that returns a `BaseErrorReport`.

Listing 10.1: 'Example algorithm'

```
class PreviousDaysAlgorithm(Algorithm):
  def train(self, data, target):
    pass

  def run(self, datapoints):
    return datapoints.sum() / datapoints.length

  def get_name():
    return 'previous_two_days'
```

10.2 The Command Line Interface

The entry point of the application is the CommandLineInterface.py script. This script is intended to be called from the command line. Incorrectly calling the script, or calling it with the -h option, will result in a help message being printed to the console.

Listing 10.2: Calling the program without arguments

```
$ python3 CommandLineInterface.py
usage: CommandLineInterface.py [-h] {list,benchmark,
    forecast} ...
CommandLineInterface.py: error: the following arguments
    are required: command
```

Listing 10.3: Program help message

```
$ python3 CommandLineInterface.py -h
usage: CommandLineInterface.py [-h] {list,benchmark,
    forecast} ...

positional arguments:
  {list,benchmark,forecast}

optional arguments:
  -h, --help            show this help message and exit
```

10.2.1 Getting a list of installed algorithms

The program is capable of scanning the `algorithms/` and `data/` folders for algorithms and datasets respectively. Calling the CLI with the list command will output a list of all installed classes. This is shown in listing 10.4.

Listing 10.4: Listing installed algorithms

```
$ python3 CommandLineInterface.py list --type algorithms
gaussian_process
lasso
ridge
stochastic_gradient
support_vector
```

10.2.2 Running a benchmark

When invoked with the benchmark command, the program is able to run algorithms on datasets and see how they perform. It is possible to run a specific algorithm, or run all of them in succession. Listing **??** shows how the program can be used to run all algorithms on the provided Uithof dataset.

```
$ python3 CommandLineInterface.py benchmark all uithof --
    start 2015-01-01 --end 2015-02-28
Benchmarking gaussian_process on uithof
Benchmarking lasso on uithof
Benchmarking ols on uithof
Benchmarking ridge on uithof
Benchmarking stochastic_gradient on uithof
Benchmarking support_vector on uithof
/---------- Report for uithof ----------\
| Algorithm:            gaussian_process |
| Mean Squared Error:          275.3335  |
| Mean Absolute Error:          13.0929  |
| R2 score:                       -0.71  |
|                                        |
\----------------------------------------/
/---------- Report for uithof ----------\
| Algorithm:                       lasso |
| Mean Squared Error:          424.3764  |
| Mean Absolute Error:          14.3846  |
| R2 score:                        0.78  |
|                                        |
\----------------------------------------/
/---------- Report for uithof ----------\
| Algorithm:                         ols |
| Mean Squared Error:          106.3274  |
| Mean Absolute Error:           7.3702  |
| R2 score:                        0.78  |
|                                        |
\----------------------------------------/
/---------- Report for uithof ----------\
| Algorithm:                        ridge  |
| Mean Squared Error:             78.3651  |
| Mean Absolute Error:             6.6729  |
| R2 score:                          0.51  |
|                                          |
\------------------------------------------/
/----------- Report for uithof -----------\
| Algorithm:        stochastic_gradient  |
| Mean Squared Error:            132.2321  |
| Mean Absolute Error:             8.1622  |
| R2 score:                          0.72  |
|                                          |
\------------------------------------------/
/----------- Report for uithof -----------\
| Algorithm:              support_vector  |
| Mean Squared Error:             96.6133  |
| Mean Absolute Error:             8.1439  |
| R2 score:                          0.40  |
|                                          |
\------------------------------------------/

Algorithm lasso performs best on dataset uithof with an
    R2 score of 0.78
```

10.2.3 Adding new data

One of the largest requirement of the program was the ability to easily add new data. The `data/` folder is scanned for datasets. There are few requirements for these datasets. They should be pickled [1]Pandas dataframes. The dataframes should have a DateTimeIndex and a single column containing the energy usage data.

[1]A pickle is a Python object that has been serialized to a byte stream.

Chapter 11

Provided Machine Learning Models

We provide six different algorithms with the system. These are the ordinary least squared method, the ridge regression method, the lasso method, the Gaussian Process, the stochastic gradient descent and a support vector algorithm. All the implementations are provided by Sci-kit Learn. This chapter will shed some light on how a few of these different methods function.

11.1 Ordinary Least Squares

Ordinary Least Squares, or OLS, is a simple method for estimating parameters. It seeks to fit a straight line through the data by minimizing the vertical distance between the line and the datapoints. Figure 11.1 shows such a line produced by OLS. The x axis is the mean of the energy consumption of the previous two days (per PTU) and the y axis is the corresponding energy consumption of the next day.

11.2 Ridge Regression

The Ridge Regression estimator is defined as:

$$\hat{\beta} = argmin\{\sum_{i}(y_i - x_i\beta)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\} \tag{11.1}$$

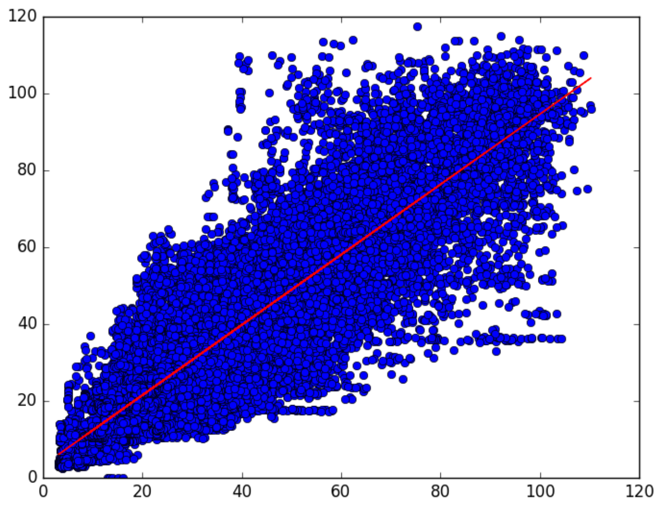


Figure 11.1: A line fitted to the data using OLS

Like OLS, ridge regression is a linear regression model. Unlike OLS, ridge regression can always find a solution, even when the amount of equations is less than the amount of parameters. Figure 11.2 shows a line fitted to the dataset using Ridge regression.

11.3 Lasso

Lasso is short for 'least absolute shrinkage and selection operator'. It was developed by Robert Tibshirani to solve some problems of ordinary least squares estimates[11]. OLS estimates are often lacking in prediction accuracy, meaning they have a large variance. Lasso seeks to combine the best features of subset selection and ridge regression by shrinking some coefficients and setting some to 0. Setting coefficients to zero indicates that they are unlikely to influence the output much. This reduces the number of relevant variables, simplifying the model.

The lasso estimate is defined as[1]:

$$\hat{\beta} = argmin\{\sum_{i=1}^{N}(y_i - \sum_j \beta_j x_{ij})^2\} \tag{11.2}$$

[1]This estimate is a simplified version of the one shown in the paper by Tibshirani.

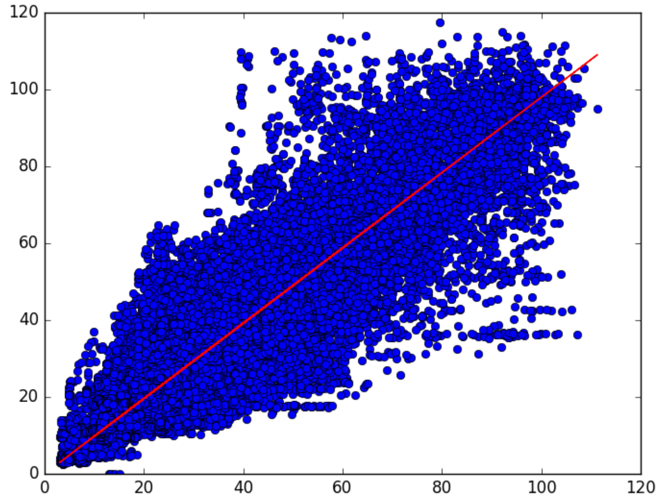


Figure 11.2: Ridge regression

$\hat{\beta}$ is a vector of coefficients for each input variable x_j, with i being the index of the current training sample. There is the added constraint that $\sum_j |\beta_j| \leq t$, where t is a parameter that controls that shrinkage of the coefficients.

In order to utilize lasso it is required that the input is normalized so that the mean is 0 and the variance is 1. This was done using the sklearn preprocessing package. This package provides a StandardScaler class. Fitting this scaler on one of the input days and apply it to all inputs and the output produced the desired results.

11.4 Gaussian Process

Before we started this project, Peeeks predicted the energy consumption of their assets using a Gaussian Process. Gaussian Processes see a lot of use with regards to timeseries and finance-related fields. We can define the Gaussian Process as follows:

Definition 11.4.1. For any set S, a **Gaussian Process** on S is a set of random variables $(Z_t : t \in S)$ so that $\forall n \in \mathbb{N}, \forall t_1 \dots t_n \in S$, $(Z_1 \dots Z_n)$ has a multivariate Gaussian distribution.

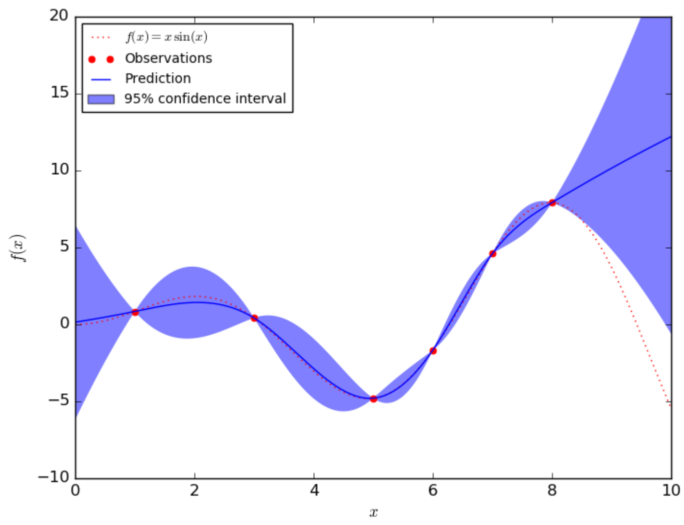


Figure 11.3: An example of a fitted Gaussian Process

11.5 Support Vector Regression

Support Vector Regression differs from OLS, Ridge Regression and Lasso in that the curve it generates doesn't need to be a straight line. While you can capture more complex relationships between input and output, it has the drawback of being more computationally intensive.

Support Vector Machine, which lie at the basis of both Support Vector Regression and Support Vector Classification, are able to use a 'kernel' to transform the input into a higher dimension. In this new input space, it can try to fit a straight line through the data. This is called a hyperplane. When the hyperplane is then mapped back to the original input space, it appears as a curved line.

Chapter 12

Testing and Quality Assurance

Testing is an important part of the development process of the application, because testing gives some guarantee for a stable application. Testing ensures that parts of the program will function as they should do. If they do not, it is noticed in the development phase of the program and not at the end. For testing we use PyUnit, this is a framework for unit tests and mocking.

From the start of the project we embraced the method of Test Driven Development. This means that the programmer wil write his tests first and then the implementation. The idea is that you write tests according to the specifications and then write code until the tests pass. This way code you will write will be in line with your initial idea. At first this went relatively well, but in the end we could not maintain this method. We do acknowledge the importance of writing many tests, which is why the most important aspects of the application do contain the tests.

Chapter 13

Requirements Evaluation

13.1 Compare all algorithms

The first of the requirements was the ability to run all algorithms on a dataset and compare them. By running the command shown in listing 13.1, the program will run all algorithms on the uithof dataset and display the algorithm with the highest R2 score. We consider this requirement fulfilled.

Listing 13.1: "Comparing algorithms"

```
$ python CommandLineInterface benchmark all uithof
```

13.2 New datasets

It should be easy to add new datasets to the framework, with minimal effort on the part of the user.

So long as all the pickled dataframes follow the same structure, a unix timestamp as an index with a single column containing the measured value, the program will treat them all the same. It will only be necessary to place them in the `data/` folder and call the application with the proper name.

13.3 Forecast windows

While it is often the case that Peeeks has the consumption data up to the previous day, this is not always the case. That is why it was required to have a variable amount of days between the last datapoint and the forecasted day.

We've implemented a **gap** command line option that indicated this forecast window. It can be used together with the **date** option to control which day and which data is used for forecast. They are optional arguments with default values 1 and tomorrow. When no other values are provided the program uses the consumption data up to yesterday 23:45 to forecast the consumption per PTU for tomorrow.

We believe this provides enough flexibility so that Peeeks can still forecast, even when the consumption data is still missing.

13.4 Different data

Peeeks plans on eventually using data other than past consumption for forecasting. Data like weather conditions could have an impact on the energy consumption of an ice skating rink.

The application can currently only handle datasets with the structure described above. However, it is possible to append other kinds of data to the energy consumption to incorporate it into the forecast. This requires a bit more programming, but would not take much time. Peeeks has already experimented with including the day of the week in the forecast.

We did not get around to incorporating this feature in the program ourselves, but Peeeks should be able to do this with relative ease.

13.5 Metrics

To determine if the algorithm gives a good forecast of the dataset it needs to be trained. After the training, there should be data of how the algorithm has performed on the training dataset. A clear and simple way to do this is making use of metrics. These give an indication of how the algorithm has performed on the dataset. Due to the difference in the datasets and the algorithms there is no single best metric, therefore we list them all and highlight the metric that performed best on the dataset. The requirement for this solution is the neccesary knowledge of the metrics used, the people that are going to use the application know this and have the required knowledge appriorατly asses the metrics given by the application.

Chapter 14

SIG Feedback Evaluation

Our code was submitted to review by the Software Improvement Group (SIG). Their mission is to improve software quality worldwide by consulting software development companies. Software submitted to SIG is graded between 1 and 5 stars.

All BEP groups have to submit their code twice. After the first submission, SIG gives some feedback on how the code can be made more maintainable. This chapter discusses the feedback we received and how we dealt with it.

The feedback from SIG is shown in appendix B.

14.1 Evaluation

In our feedback from SIG we got 4 stars out of 5, indicating that our code has above average maintainability. The aspects of our code where we scored low are code duplication and unit interfacing.

14.1.1 Code Duplication

Code duplication means that several classes or modules have the same code. This is undesirable, as changes to the code must be made in several different locations and they will quickly go out of sync.

SIG noted that we had some code duplication regarding the Algorithm abstract class and its implementations. This duplication was not picked up by Pylints code duplication detection and we had not noticed it yet. We refactored the abstract class and managed to reduce our codebase with nearly 300 lines of code.

14.1.2 Unit Interfacing

This is about the parameters of the code. In their feedback it became clear that we used too much default parameters. This is generally undesirable because it is a sign of lack of abstraction and leads to code that is hard to understand.

After evaluation their feedback with our code, we decided to let the code as it is. We did this because it is far easier for the client to not have to change the parameters all the time.

Chapter 15

Libraries and Tools

In our project we made use of several existing libraries and other technologies. Pythons extensive ecosystem is the source of its reputation as the language of choice for rapid prototyping. This chapter describes the dependencies of the framework and how they are used.

15.1 NumPy

NumPy is "the fundamental package for scientific computing with Python"[4]. It is a continuation of the Numeric and Numarray projects, which were eventually merged to form NumPy.

NumPys biggest contribution to the project is the ndarray object. The ndarray represents "a multidimensional, homogeneous array of fixed-size items"[5]. These arrays are fast and efficient and have support for operations on all individual elements. This proved especially useful when computing the performance of an algorithm on a dataset.

15.2 Pandas

Pandas is an open source data analysis framework that provides high performance data structures. It aims to be a building block for practical 'real world' data analysis with Python[2].

Pandas provides a dataframe object. Like the ndarray discussed previously, this is datastructure scales well. The data is provided to the program through a serialized dataframe. Pandas enables us to efficiently filter and restructure this data. It is then converted into a ndarray for computation.

15.3 scikit-learn

scikit-learn, or sklearn, is a Python framework that is all about machine learning. It provides a lot of different machine learning models and is the driving force behind the algorithms that we have provided. It also comes with a variety of metrics that we use to evaluate the algorithms.

Scikit-learn is dependent on NumPy, therefore it is even more essential to the application. We use scikit-learn to get all the available regression algorithms that it contains.

15.4 Matplotlib

Matplotlib is a library for creating 2D plots in the style of Matlab, with support voor 3D plots as well. Users can plot a wide variety of graphs, from simple lines to scatterplots, boxplots and histograms. All the various plots that can be seen in this report were plotted using this library.

15.5 Pylint

Pylint is a code quality checker for Python. It checks for a number of things, including PEP8 compliancy, line lengths and name convention. It comes shipped with Pyreverse, a tool that we used to generate UML diagrams describing our code. Using Pylint helped keep our code maintainable and easy to read.

15.6 PyUnit

PyUnit is a library for testing Python code. It comes bundled with Python itself. We used PyUnit to write unit and integation tests for our code. This allowed us to catch regression errors as they happened and ensured a higher code quality.

Chapter 16

Conclusion

Over the past 10 weeks we developed an application for benchmarking and comparing machine learning algorithms for predicting energy consumption. This chapter contains our final remarks on how we feel the product turned out.

The application has been implemented with all of the requirements that are vital for Peeeks to work with the program. The application can be called a succes in that aspect.

At the moment the program can run algorithms on the given dataset and determine, via graphical output of metrics, if the chosen algorithm is suitable to use for predictions. The application can also give predictions with, or without time intervals.

At the moment the application can directly be taken in use. The users of this program all have technical backgrounds, thus they should have no problem of using it. However the program can benefit from new functions and upgraded UI. This is something to look at in the future, because for now it does everything it should do.

We are both very happy with the result and had a great time working on the application. We both think the program can have a positive effect on the actions of Peeeks.

Chapter 17

Recommendations

In our conclusion it was stated that the project had been a succes, although there was some room for improvement. In this part we will further elaborate upon our view of the application in the future.

17.1 Graphical User Interface

What was already stated in the conclusion that a GUI was not neccesary, however if the program will be used by more people, it is for professionality and a clear view of the application it is neccesary that there must be a GUI. With the addition of a GUI it will also be easier to implement new features, because of clear overview that the application can give with a GUI. The options will be seen in an instant, instead of searching in the command line what the functions actually are.

17.2 Different datasets

In the state that the application is right now, it can only accept one kind of dataset. The dataset must be structured in a specific way to be handled by the program, this is not a problem at the moment. This is because most of the datasets that come from the electrical companies are in the format that our program requires. In the future Peeeks may use other variables in their equation to reduce the cost of electricity. The datasets that they will use are most likely not in the format that our application can handle.

17.3 Parameter optimalization

At this moment the parameters are set, and can only be manually changed in the code. Without this feature the algorithm cannot optimaly adjust itself to the dataset. It would be completely finished if the application would find out itself what the parameters are. This way the user would not have to do the arduous task of constantly changing the parameters to get the result he wants.

Part IV

Appendices

Appendix A

Initial Project Description

The following is the project as descriped by Peeeks on BepSys.

> The national electricity grid needs to be balanced at all times. This means that generation of energy must match the demand at all times. As the supply of renewable electricity fluctuates due to factors beyond our control, such as changes in weather conditions, Peeeks wants to shift the energy usage of installations with thermal buffers such as cold stores, ice skating rinks etc. to moments when renewable energy availability is highest. In order to optimise our portfolio of assets we need to be able to accurately forecast the energy needs of the various assets in the condition that we do not actively control the asset. We do this by utilizing the power of machine learning.
> The project would be to design a framework that allows us to find the optimal machine learning algorithm that predicts the energy need of an asset for the next day. To make sure we use the best possible algorithm, we want a team to develop a framework that allows us to test different algorithms. It is important that this framework is capable of testing different algorithms on available datasets in an unbiased way. So the participating students will have to think about the fairness of tests and be able to think of solutions on the spot.
> When developing this framework it is important that it is modular in its design, so that new algorithms can be tested with minimal effort. Included in this assignment is the challenge to beat our current forecasting algorithm by utilizing new algorithms or improving ours. The framework should be capable not only of allowing new algorithms, but also different types of data. Typical data that will be used when training the algorithm is the historical electrical usage of the asset combined with weather data. The options are however limitless, so if we (or you!) think of a new relevant data source, it

should be possible to incorporate this data in the framework without a lot of work.

Appendix B

SIG Feedback 1

De code van het systeem scoort 4 sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code bovengemiddeld onderhoudbaar is. De hoogste score is niet behaald door lagere scores voor Duplication en Unit Interfacing.

Voor Duplicatie wordt er gekeken naar het percentage van de code welke redundant is, oftewel de code die meerdere keren in het systeem voorkomt en in principe verwijderd zou kunnen worden. Vanuit het oogpunt van onderhoudbaarheid is het wenselijk om een laag percentage redundantie te hebben omdat aanpassingen aan deze stukken code doorgaans op meerdere plaatsen moet gebeuren.

Bij jullie zit de duplicatie vooral in de bestanden in de directory ”algorithms”. De inhoud van deze bestanden geeft de indruk dat het hier om een library gaat, maar uit de code is niet te bepalen of jullie die library-code wel/niet hebben aangepast. In het algemeen is het beter om code die door een externe partij is ontwikkeld apart te plaatsen. De naamgevingsconventie verschilt per programmeertaal, maar als je het in een directory ”lib” plaatst is er sprake van een harde grens tussen jullie eigen code en library-code. Op die manier voorkom je ook dat mensen per ongeluk een bestand dat bij een library hoort aanpassen.

Voor Unit Interfacing wordt er gekeken naar het percentage code in units met een bovengemiddeld aantal parameters. Doorgaans duidt een bovengemiddeld aantal parameters op een gebrek aan abstractie. Daarnaast leidt een groot aantal parameters nogal eens tot verwarring in het aanroepen van de methode en in de meeste gevallen ook tot langere en complexere methoden.

In jullie geval zijn er een aantal methodes die default parameters bevatten. Dat is inderdaad een handige manier om meer flexibiliteit in het gedrag aan te brengen, maar als je meer dan één default parameter hebt maakt het de code wel moeilijk(er) leesbaar. Daarnaast is het gebruik van default parameters niet zinvol op het moment dat er in de praktijk nooit verschillende waardes voor die parameters worden gebruikt. In zo'n situatie kun je de flexibiliteit beter wegnemen om de code zo simpel mogelijk te houden.

Het is goed om te zien dat jullie naast productiecode ook unit tests hebben geschreven. De verhouding tussen de hoeveelheid testcode en de hoeveelheid productiecode is met 1 op 10 nog wel voor verbetering vatbaar. Hopelijk lukt het jullie om die verhouding tijdens het vervolg van het project nog wat op te trekken.

Appendix C

Info Sheet

Name of the the program: Predem

Name of the client organization: Peeeks

Date of the final presentation: 24-06-2016

Description: The client has need for acurate predictions of power usage for their assets. There is historic data for all of their assets, they want to use it to predict the power usage. They need an application that can quickly predict the power usage for an asset.

Challenge: The project required knowledge about Machine Learning. Machine Learning is a course at the bachelor of the TU Delft however, this is not mandatory and not comprehensive enough for this project. Another challenge for us was that we both had never worked with Python before.

Research: Our research involved getting to know how what kind of machine learning methods there were and how the different algorithms worked.

Process: We involved the Scrum methodology in our process, in this process the team, but also the client could issue ideas and features. For the implementation side of the project we used Test Driven Development.

Product: Our product is an application that will be accessed via the command-line. The results will show metrics in how the algorithms have performed on the dataset. All of our core features and the requirements have been implemented.

Outlook: The program will be used by Peeeks and has been built with the idea that it could be extended if needed.

C.1 Members of the project team

Name: Jens de Waard
Interests: Security, Machine learning
Contribution and Roles: Developer and Scrum Master

Name: Robert Luijendijk
Interests: Machine learning and User interaction
Contribution and Roles: Developer and Product Owner

C.2 Client

Name: Paul de Graaf
Affiliation: Peeeks

C.3 Coach

Name: Matthijs Spaan
Affiliation: Algorithmics Group at EEMCS TU Delft

C.4 Contact

Jens de Waard: jensdewaard@gmail.com
Robert Luijendijk: luijendijk12@gmail.com

C.5 Location

The final report can be found at: http://repository.tudelft.nl

Appendix D

Research Report

BEP - Research Report

Robert Luijendijk
4161467

Jens de Waard
4009215

April 2016

Preface

This document is the research report for the Energy Consumption Prediction assignment for the Bachelor Project course, which is the last project based course of the Bachelor of Science programme at the TU Delft. The assignment was issued by Peeeks, a start-up located in Delft. This research report contains all background information and preliminary reading we thought was necessary to successfully complete the assignment.

We would like to thank Matthijs Spaan, our coach from the TU Delft, for guiding us through the process of writing a research report. We would also like to thank Paul de Graaf and Jaap Beekers for anwering all our questions regarding the technical details of forecasting the energy market.

Contents

Preface **i**

1 Introduction **1**

2 The electrical network and the energy market **2**
2.1 Introduction . . . 2
2.2 The Dutch electric grid . . . 2
2.3 The imbalance market . . . 3
2.4 The role of Peeeks . . . 3

3 Machine Learning **4**
3.1 Introduction . . . 4
3.2 Learning methods . . . 4
3.2.1 Supervised learning . . . 4
3.2.2 Unsupervised learning . . . 4
3.2.3 Semi-supervised learning . . . 5
3.2.4 Reinforcement learning . . . 5
3.3 Models . . . 5
3.3.1 Neural Networks . . . 5
3.3.2 Deep Learning . . . 5
3.3.3 Rule System . . . 6
3.3.4 Regression . . . 6
3.3.5 Bayesian . . . 6
3.3.6 Decision Tree . . . 6
3.3.7 Instance Based . . . 7
3.3.8 Clustering . . . 7
3.4 Peeeks' current algorithm . . . 7

4 Testing Machine Learning Algorithms **8**
4.1 Model validation . . . 8
4.2 Metrics for errors . . . 8
4.3 Application to the project . . . 9

5 Python **10**

5.1 Introduction 10
5.2 NumPy 10
5.3 Pandas 10
5.4 Scikit-learn 11
5.5 Matplotlib 11
5.6 SciPy Library 11
5.7 IPython 11

6 Conclusion **12**

A Definitions **1**

B Machine learning framework **2**
B.1 Input and output data 2
B.2 Application entrypoint 2

Chapter 1

Introduction

The load on the electrical grid is not constant throughout the day. It ramps up during the morning as people get up and go to work and ramps back down again when they go to sleep. As power can't be stored on the electrical grid, energy production has to be increased or lowered to compensate for the change in demand. Traditionally, this was done on the production side of the electrical grid. Coal and gas power plants supply a predictable amount of energy to the grid and can be turned off and on with relative ease.

Power production is shifting to more sustainable methods like solar and wind power. However, the power produced by these plants is less predictable. If a day is more or less windy than forecast, wind and solar power output will change accordingly. To increase the grids capacity for sustainable power sources, the demand side of the equation should be flexible.

Tennet is the distribution network operator of the electrical grid in the Netherlands. It is their responsibility to ensure there is an equal supply and demand of power on the grid. They also calculate the cost (or benefit) of supplying or substracting power from the grid.

Peeeks aims to reduce the electrical costs of the clients in their portfolio by 10 to 40% by successfully predicting their energy needs and the energy market. These predictions can be difficult to make due. The algorithms used to make these predictions can be improved through machine learning. Peeeks is looking for a framework to assess and compare these algorithms.

Chapter 2

The electrical network and the energy market

2.1 Introduction

This chapter details the electrical network of the Netherlands and how power is bartered on the energy exchange. The third sections explains how Peeeks plans to use these aspects to their advantage.

2.2 The Dutch electric grid

The power grid is an important piece of infrastructure and has been for the past sixty years[**?**]. It is important that the supply meets the demand, so that everybody has the power they need. However, storing power in the grid is difficult and impractical to do[**?**]. This means that the supply must be roughly equal the demand 24 hours a day for seven days a week.

Electricity consists of charged particles, which attract and repulse each other. This is observed as an oscillation, which has a frequency. Should the balance between supply and demand be lost, this frequency will increase or decrease. The Dutch electrical grid is designed to operate at a frequency of 50 Hz[**?**]. Deviations from this frequency can cause damage to the grid and appliances connected to it.

A transmission system operator such as TenneT is tasked with keeping the balance between supply and demand. Parties can notify TenneT about their intended power usage or power generation for the next day. These E-Programmes are binding and deviations from the program are financially regulated by TenneT on the imbalance market.

2.3 The imbalance market

In reality, there will always be some form of imbalance on the grid. This could be caused because the weather is worse than predicted and, for example, less people go to an attraction park and the water ride won't be used.

When imbalance occurs, TenneT will need to rectify this as soon as possible. They do so by placing tarifs on the supply and substracting of unplanned energy during a 15 minute timeslot. For example, should supply exceed demand at a given time, they may decide to pay a Balance Responsible Party (BRP, a party that supplies or substracts energy to or from the network) to substract more energy than stated in their E-Programme. Or, when there is less supply than expected, it may be costly to substract more than originally planned.

The prices for causing imbalance during a given PTU are calculated at the end of that PTU. There are seperate prices for supplying and substracting energy. This means that the cost for a PTU is the amount of energy supplied times the price for substracting, minus the amount of energy substracted times the price for supplying, or:

$$P_{PTU} = B * P_{substract} - A * P_{supply} \tag{2.1}$$

The prices $P_{substract}$ and P_{supply} are determined by TenneT using a system of rules. These rules take a variety of variables in account, including whether there was a surplus or shortage of energy and the prices of energy TenneT has to pay to suppliers[**?**].

2.4 The role of Peeeks

Peeeks controls the power of companies with large cooling or heating systems. They will use these systems as a power storage for when the price is low. During low prices, the cooling systems are engaged, lowering the temperature in the cold storage facilities. Likewise, when power is expensive, the temperature will be allowed to rise. By predicting the imbalance market and modeling the temperatures of the cooling and heating systems, Peeeks aims to provide a 10 to 40% decrease in costs for their customers.

Chapter 3

Machine Learning

3.1 Introduction

Machine learning is the "field of study that gives computers the ability to learn without being explicitly programmed"[?]. Through machine learning, computers can learn from data and gain the ability to predict future values. In this chapter, we will cover various methods of machine learning.

3.2 Learning methods

There are 4 main types of learning methods, these can be used to teach the algorithm what it must do[?]. These methods differ slightly from eachother, and which is used is often depended on the algorithm you want to create, or what kind of data you want to adress.

3.2.1 Supervised learning

Supervised learning works with data that is already labeled. The algorithm is able to infer a function by being trained on a dataset. Supervised learning gets its name from the vector with desired values, called the supervisory signal.

3.2.2 Unsupervised learning

Supervised learning required knowledge of the desired output. In cases when there is no supervisory signal available, unsupervised learning can be used. This means unsupervised learning can be used to show a hidden structure in the data.

3.2.3 Semi-supervised learning

Semi-supervised learning is used when an algorithm needs to go through data which is both labeled and unlabeled. The addition of even a small amount of labeled data can significantly improve the accuracy of the resulting algorithm. Since the generation of a fully labeled dataset may be costly, semi-supervised learning is a good middle road.

3.2.4 Reinforcement learning

Using this learning type, the machine is exposed to self learning and it will teaching itself by trial and error. Reinforcement learning differs from supervised learning in that there are no pairs of correct input/output combinations provided (like with unsupervised learning) but that it is not entirely unguided like unsupervised learning. It requires some form of function to compute the reward for a decision[**?**].

3.3 Models

There are many different models suitable for machine learning, we will discuss a few of these.

3.3.1 Neural Networks

Neural networks are modeled after the neurons in the brain. The idea is that every node, or neuron, communicates with it's neighbours to exchange information. The rate at which it does that can be adjusted. Neural networks are used primeraly for classification. Neural networks are primarily used for classification, but they use it also in speech recognition[**?**].

3.3.2 Deep Learning

An algorithm for deep learning is a very complex algorithm with many different hidden layers and complex. Deep learning is an extension of neural networks, however most of the time it is seen separate from the classical neural networks, because of the high complexity that comes with deep learning. One difference between neural networks and deep learning is that deep learning algorithms learn through both unsupervised learning and supervised learning. Deep learning is mostly used in the area of facial recognition and image detection[**?**].

3.3.3 Rule System

The rule system algorithms are used for finding correlations between multiple points of data. Often this is data about products of, for example, a supermarket. With these data the algorithm can find general rules, an example that can be found is: if people buy ham and butter, they almost always will also buy bread. This can be useful for supermarkets in their strategy of where to put their products[**?**].

3.3.4 Regression

Regression in mostly used for prediction and forecasting. This prediction is usually applied to data which is classified, for example the prediction of power usage in the Dutch electricity network. But regression is not only used for prediction, it is also used for causal relationship detection.

3.3.5 Bayesian

Bayesian algorithms are based on Bayes theorem, which describes a probability of an event, based on the conditions that might be related to the event. The formula for this is:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{3.1}$$

Figure 3.1: Bayesian theorem

$P(c|x)$ = The posterior probability of c given x
$P(x|c$ = The posterior probability of x given c
$P(c)$ = The chance that c occurs
$P(x)$ = The chance that x occurs

Bayesian algorithms are used for the classification of data that is unlabeled[**?**].

3.3.6 Decision Tree

The decision tree algorithm is an algorithm that is based on the shape of trees, the branches diverge in 2, sometimes more, sections. This makes it an excellent tool for calculating the chance of object occuring for example. It is no surprise that this algorithm is primarily used in classification of objects[**?**].

3.3.7 Instance Based

Instance based learning is a kind of lazy learning, this means that the algorithm will only look at the dataset that it is given, and assumes that as it's world view. It will compare the given dataset to the datasets that have been stored in its memory. It is called instance based because it creates the hypthoses directly from its instance, thus the dataset.

3.3.8 Clustering

Clustering is a task of sorting all the objects with, for example, the same trait with eachother. This technique is largely used in unclassified datasets, because it also has the ability to learn the traits. After that he sorts them with eachother[**?**].

3.4 Peeeks' current algorithm

For its current forecasting Peeeks uses a gaussian process algorithm. This algorithm is included with Sci-kit Learn and worked rather well, so they decided to stick with it. The algorithm itself is used as a forecast algorithm. It will take the data from the previous two days to predict that usage of the next. In other words, the energy usage of Day 0 and Day 1 are used to predict the usage of Day 3.

The algorithm itself is a generalization of a normal distribution theorem, or Bell curve. This states what value is most likely to occur, it does this with the normal distribution, which in turn is connected to the probability densitiy function.

$$f(x|\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.2}$$

Figure 3.2: Probability density function

μ = The mean of the distribution.
σ = The standard deviation.
σ^2 = The variance.

For a forecast algorithm this is good because of the answer from the Gaussian process will most likely be the correct one, or the prediction is very close, which is also acceptable[**?**].

Chapter 4

Testing Machine Learning Algorithms

4.1 Model validation

After developing an algorithm, it has to be tested and trained. This is usally done by dividing the dataset into two partitions, a training set and a test set.

In conventional validation, the dataset is partitioned into two sets, 70% for training and 30% for testing. This type of validation has a few drawbacks, namely that the error does not properly represent the model of the performance[**?**].

A more useful type of validation is k-fold cross validation. This consists of dividing the dataset in k equal partitions. The set is then trained with $k-1$ partitions and tested on the last. This process is repeated k times, called folds. The results of the folds are then combined.

4.2 Metrics for errors

After training and testing the model, there are several indicators that show how well it performed. None of these metrics are necessarily 'better' than the others, but each have their own uses.

$$E = \sum_{i=1}^{n} (\hat{X}_i - X_i) \tag{4.1}$$

Figure 4.1: Calculating the error

The **error** of an algorithm gives a simple indicator of how much the predictions missed the mark. However negative and positive errors cancel out, so this may not be a reliable indicator of performance.

$$AE = \sum_{i=1}^{n} |\hat{X}_i - X_i| \tag{4.2}$$

Figure 4.2: Calculating the absolute error

The **absolute error** is useful for situations where negative and positive errors should not be canceling each other out. By taking the absolute value of the error, they add up instead, i.e. errors of 1 and -1 add up to an error of 2, not 0.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (\hat{X}_i - X_i)^2 \tag{4.3}$$

Figure 4.3: Calculating the mean squared error

The **mean squared error** is the average difference between the predicted value $\hat{X}_i$ and the actual value X_i squared. This metric improves upon the absolute error by adding more gravitas to larger errors.

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} |\frac{X_i - \hat{X}_i}{X_i}| \tag{4.4}$$

Figure 4.4: Calculating the mean absolute percentage error

The **mean absolute percentage error** expresses the accurary of the algorithm as a percentage. This gives some context to the absoute error. An error of 5 on a prediction of 1000 is not as bad as an error of 5 on a prediction of 5.

4.3 Application to the project

The above information is relevant to the developing of an application for training and testing machine learning algorithms. As the objective of the application is to aid in deciding which algorithm to use for forecasting, it should be able to calculate and show all the above metrics.

To train the algorithms, it will use k-fold cross validation to reduce the size of needed datasets.

Chapter 5

Python

5.1 Introduction

The application will be developed with Python, as Peeeks already uses this programming language for their other projects. Additionally, Python has a rich ecosystem of data analysis modules. Together, these modules are called SciPy. This chapter contains an overview of relevant modules, their uses and how they relate to machine learning and predicting the energy market.

5.2 NumPy

NumPy is short for Numerical Python and is one of the most used Python modules. It is the foundational package for scientific python and is a dependency for most of the other modules in this chapter. NumPy contains classes for constructing N-dimensional arrays and functions for manipulating them. The datastructures provided by NumPy are much more efficient than the conventional Python arrays. This allows for faster algorithms and means that the programming languages C, C++ and Fortan can handle these arrays without converting them first[**?**]. NumPy will form the basis of the computational intelligence aspect of the application.

5.3 Pandas

The Python Data Analysis Library is a powerful data analysis environment that provides several powerful datastructures. The most important of these is the dataframe. This is a two dimensional table with rows and columns. Pandas makes use of NumPy and as a result it is a fast and reliable module[?]. The datasets that are used by Peeeks come in the form of datagrams, and as such the use of Pandas is a requirement.

5.4 Scikit-learn

Scikit-learn is a collection of tools for machine learning. It is built on NumPy, SciPy and Matplotlib. Scikit-learn offers numerous machine learning methods such as classification and regression[?]. This module will form the basis for developing and comparing machine learning algorithms.

5.5 Matplotlib

Matplotlib is a module witht the intention to add Matlab plotting functionality to the SciPy stack. With plots the performance of the machine learning algorithms can be visually presented to the user.

5.6 SciPy Library

The SciPy library, also referred to as SciPy, contains a number of domain-specific functions. It contains modules for optimization, linear algebra, integration, FFT, signal processing and more. While SciPy is a very powerful library, the machine learning goals of the application would best by served by using scikit-learn directly.

5.7 IPython

IPython is a powerful interactive shell and a part of the SciPy stack[?]. However, the application will be developed using the PyCharm IDE by Jetbrains. PyCharm is a powerful IDE and there will not be a need for an interactive shell.

Chapter 6

Conclusion

Companies trading on the APX and imbalance markets benefit greatly from being able to predict these markets. Sending in wrong E-programmes can be very expensive. If companies are better able to predict their energy needs, they are better able to adapt to the energy prices and reduce energy costs. In addition, a more flexible energy market has a greater capacity for renewable energy sources, as their power output is less consistent.

Machine learning can provide a reliable method of predicting the amount of energy used by companies. The field of machine learning encompasses many fields, not all of which are suitable for predicting continuous values. Gaussian Process Regression is the current technique employed by Peeeks. While this type of regression shows a lot of promise, it does not take domain specific knowledge into account and as such there is room for improvment.

To be able to tell of an certain algorithm performs better than another, there is need for object metrics. There are various error rates that can be calculated from the performance of an algorithm. Not all these type of error calculate provide the same indication of performance, and not all are suitable for algorithms predicting energy usage.

Python is currently used by the developers at Peeeks, and it is the language that will be used to develop the framework. Python is increasingly used for data science and there are a lot of modules available. Stacks such as SciKit Learn and Numpy make Python an excellent language for the purposes of this assignment.

Bibliography

[1] Association analysis: Basic concepts and algorithms.

[2] Jason Brownlee. A tour of machine learning algorithms.

[3] Bruce Dunn, Haresh Kamath, and Jean-Marie Tarascon. Electrical energy storage for the grid: A battery of choices. *Science*, 334(6058):928–935, 2011.

[4] Patrick Hall. How does deep learning work and how is it different from normal neural networks and/or svm? Accessed 29-04-2016.

[5] Ibrar Hussain. Clustering in machine learning.

[6] IPython. Ipython interactive computing.

[7] Mark Lee. Reinforcement learning.

[8] Tom M Mitchell. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45:995, 1997.

[9] Micael Negnevitsky. *Artificial Intelligence, A guide to intelligent systems*, volume 2. 2005.

[10] NumPy. About numpy.

[11] Pandas. Python data analysis library - pandas.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[13] Carl E Rasmussen and Christopher KI Williams. Gaussian processes for machine learning (adaptive computation and machine learning). 2005. *ISBN 026218253X*.

[14] Sunil Ray. 6 easy steps to learn naive bayes algorithm (with code in python).

[15] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, July 1959.

[16] Giovanni Seni and John Elder. *Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions.*

[17] TenneT. About electricity.

[18] TenneT. Onbelansprijssystematiek.

[19] Wenye Wang, Yi Xu, and Mohit Khanna. A survey on the communication architectures in smart grid. *Computer Networks*, 55(15):3604 – 3629, 2011.

Appendix A

Definitions

The aim of our Bachelor Project is to develop an application that can aid in the development of machine learning algoritms.

BRP Balance Responsible Party. A party that either adds or substracts power from the power grid.

E-Programme A forecast of the amount of power a BRP thinks it will add or substract from the grid, divided up into PTU.

PTU Programme Time Unit. A block of 15 minutes.

APX Amsterdam Power Exchange. This is the main market for electricity. Companies can buy or sell their power in time slots of an hour, for the next day.

Appendix B

Machine learning framework

The information in this research report is intended to aid in the development of a framework for developing machine learning algorithms to minimize power costs. This chapter contains some directions on how to achieve this.

B.1 Input and output data

The purpose of the application is measuring and comparing the performance of machine learning algorithms. The user of the application will be able to load an implementation of a algorithm into the program. This algorithm will be trained and tested on a dataset that will also be provided by the user. The result and error rate will then be displayed.

B.2 Application entrypoint

Peeeks has several developers of their own and does not list a graphical user interface as a hard requirement. The main functionality of the application, measuring the performance of a prediction algorithm on a provided dataset, will be exposed through a python script, e.g.:

```
python peeeks-measure <algorithm> <dataset>
```

Bibliography

[1] Donald Bell. *UML Basics: The sequence diagram.* 2004. URL: http://www.ibm.com/developerworks/rational/library/3101.html (visited on 05/31/2016).

[2] Pandas Developers. *pandas.* 2016. URL: http://pandas.pydata.org/pandas-docs/stable/ (visited on 05/31/2016).

[3] APX Group. *Auction.* 2016. URL: https://www.apxgroup.com/trading-clearing/auction/ (visited on 06/14/2016).

[4] NumPy. *NumPy.* 2016. URL: http://www.numpy.org/ (visited on 05/31/2016).

[5] NumPy. *Numpy ndarray Documentation.* 2016. URL: http://docs.scipy.org/doc/numpy-1.10.0/reference/generated/numpy.ndarray.html (visited on 05/30/2016).

[6] Peeeks. *Bespaar.* 2016. URL: http://www.peeekspower.com/index.php/bespaar/ (visited on 06/14/2016).

[7] TenneT. *About electricity.* 2016. URL: http://www.tennet.eu/nl/en/about-tennet/about-electricity.html (visited on 06/14/2016).

[8] TenneT. *Balance Responsibility.* 2016. URL: http://www.tennet.eu/nl/customers/services/systemservices/balance-responsibility.html (visited on 06/14/2016).

[9] TenneT. *Brochure on Emergency Power.* May 2012.

[10] TenneT. *The Imbalance Pricing System.* Oct. 2005.

[11] Robert Tibshirani. “Regression Shrinkage and Selection via the Lasso”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288. ISSN: 00359246. URL: http://www.jstor.org/stable/2346178.