1 2 9 0

UNIVERSIDADE D
COIMBRA

João Gabriel Marques Carvalho

ELECTRICITY CONSUMPTION FORECAST MODEL FOR THE DEEC BASED ON MACHINE LEARNING TOOLS

Dissertação no âmbito do Mestrado Integrado em Engenharia Eletrotécnica e de computadores
Ramo de energia

Orientada pelo
Professo Doutor Tony Richard de Oliveira de Almeida

February de 2020

UNIVERSITY OF COIMBRA
FACULTY OF SCIENCES AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

INTEGRATED MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

Electricity consumption forecast model for the DEEC based on machine learning tools

João Gabriel Marques Carvalho

Supervisor:
Prof. Dr. Tony Richard de Oliveira de Almeida

Jury:
Prof. Dr. Humberto Manuel Matos Jorge
Prof. Dr. Nuno Miguel Mendonça da Silva Gonçalves
Prof. Dr. Tony Richard de Oliveira de Almeida

Dissertation submitted to the Electrical and Computer Engineering Department of the Faculty of Science and Technology of the University of Coimbra in partial fulfillment of the requirements for the Degree of Master of Science.

Coimbra, February of 2020

To my mother who never gave up fighting for me.

“However difficult life may seem, there is always something you can do and succeed at.”

Stephen Hawking

Acknowledgements

I would first like to thank my thesis advisor Prof. Dr. Tony Richard de Oliveira de Almeida of DEEC, at University of Coimbra, who supported and advised all the way through to the successful completion of this dissertation.

To my colleagues and friends for the friendship and moral support.

To my dogs for it love and grace.

And finally, it is imperative to express my most deep gratitude to my parents and sister, for their profound and continuous support, encouragement and strength, that they gave me throughout all these years of study and through the process of researching and elaborating this dissertation. This achievement would not have been possible without them.

Thank you,
João Carvalho

Abstract

In this thesis, the design of a machine learning neural network capable of making energy predictions is presented. With the increase in energy consumption, tools for the prediction of energy consumption are gaining great importance and their implementation is required. This concern is the main goal of the presented work.

We strive to explain the history of machine learning, what machine learning is and how it works. It is also sought to explain the mathematical background and use of neural networks and what tools have been developed nowadays to create machine learning solutions. Machine learning is a computer program that can perform trained tasks in a similar way as the human mind. The neural network (ANN) is one of the most used and important machine learning solution through which pivotal data can be obtained.

For predicting the energy consumption at the Department of Electrical and Computer Engineering (DEEC) of the University of Coimbra, a neural network was trained using real data from the overall consumption of the DEEC towers.

Phyton was the language used and the supervised learning regression algorithm utilized. With this prediction, we finally compare our data with real data, so that we may analyze it. The data used in the training of the neural network goes from 2015/July/10 to 2017/December/31, a total of 906 days. For each day of the year, there is a maximum of 3 values, which is considered a small sample, but the only one available

The final comparison between real and predicted data was only done for the month of January 2018. From the data achieved, predictions were made, but with a certain level of discrepancy, that is explained with the low amount of data available. In the future, one of the things that should be considered is to enlarge the training datasets, considering a larger amount of input variables.

The main goal proposed for this thesis was successfully obtained. With all the presented research it was strived to create text that would allow being a steppingstone in the creation of better solutions. This is an extraordinary field that in the future will be able to elevate our knowledge to a completely different level.

Keywords: **Artificial neural network, multilayer perceptron, Feed Forward, Backpropagation, Prediction**

Resumo

Nesta tese apresentaremos o trabalho sobre a criação de uma rede neuronal de aprendizagem automática, capaz de realizar previsões energéticas. Com o aumento do consumo energético, devem desenvolvidas ferramentas capazes de prever o consumo. Esta necessidade levou à pesquisa deste tema.

Procura-se explicar a história da aprendizagem automática, o que é a aprendizagem automática e como é que esta funciona. Também se procura explicar os seus antecedentes matemáticos, a utilização de redes neuronais e que ferramentas foram atualmente desenvolvidas; de forma a criar soluções de aprendizagem automática.

A aprendizagem automática consiste num programa informático, que após treino é capaz de desempenhar tarefas de forma similar à mente humana. A rede neuronal (ANN) é uma das mais importantes ferramentas de aprendizagem automática, através da qual se pode obter informação fundamental.

Para prever o consumo de energia no Departamento de Engenharia Eletrotécnica e de Computadores (DEEC) da Universidade de Coimbra, uma rede neural foi treinada usando dados reais do consumo total das torres do DEEC.

Phyton foi a linguagem utilizada e recorreu-se ao logaritmo de regressão de aprendizagem supervisionada. Com esta previsão, comparam-se os dados obtidos com os dados reais, o que permite a sua análise. Os dados usados no treino da rede neuronal vão de 2015/julho/10 a 2017/dezembro/31, num total de 906 dias. Por cada dia do ano existe um máximo de 3 valores, considerando-se assim uma amostra pequena.

A comparação final entre os dados reais e os dados previstos foi somente realizada no mês de janeiro de 2018.

A partir dos dados obtidos realizaram-se previsões, apesar de um certo nível de discrepância; justificada pela pequena quantidade de dados disponíveis. No futuro, deve-se aumentar os dados de treino de forma a obter um maior número de variáveis de *entrada*.

O principal objetivo proposto nesta tese foi atingido com sucesso. Com toda a pesquisa apresentada, buscou-se criar informação que permitisse ser um marco na criação de melhores soluções. Este é um campo extraordinário que no futuro permitirá elevar os nossos conhecimentos a outros níveis.

Palavras-chave: **Rede neuronal artificial, Perceptron de multicamada, Alimentação avante, Retro propagação, Previsão**

Table of Contents

Acronyms and Symbols

Abbreviation	Meaning
ANN	Artificial Neural Networks
BP	Back-Propagation
DEEC	Departamento de Engenharia Eletrotécnica e de Computadores
FCTUC	Faculdade de Ciências e Tecnologia da Universidade de Coimbra
FFNN	Feed Forward Neural Network
Lux (LX)	Illuminance
MLP	Multilayer perceptron’s
SKlearn	Scikit-learn

List of figures

List of tables

1 Introduction

Index

1.1 Motivation and overall goals

Every year, energy consumption grows world widely. This growth in energy consumption increases the need for a better planning of energy use, which also includes the better planning of energy distribution and energy consumption measurement [1]. Therefore, the same situation is observed in our country, Portugal.

Energy is now regarded as one of the strategic elements of society. More than just an asset, it represents a significant value to the services it provides directly and indirectly. It is ubiquitous in developed societies and it follows the same trend in developing societies [2].

With the increase of electric consumption, the necessity to predict how much electrical power is going to be consumed day by day is pivotal. This task has many variables to content with.

For example, for a building like the Department of Electrical and Computer Engineering, an educational/research institution, time of day, temperature, wind, humidity, season, class periods, weekends, total of staff/students/visitors, and many more variables can be considered as direct factors of the department energy demand.

However, just utilizing the variables *day*, *month*, *weekend*s, *seasons*, *temperature* and *humidity* we can strive for a machine learning solution that allow us to determine how much electricity is going to be used in any day of the year. The machine learning solution that we are going to utilize is an *Artificial Neural Network*.

Artificial neural network (ANN) is an emerging discipline, a branch of artificial intelligence, which has been developing rapidly in recent years. An ANN is a complex network formed by many processing units through the way of connection, that, based in intuitive human thinking, combines distributed storage of information together, resulting in sudden novel ideas or solutions to problems [3]. As such, this thesis has the aim of developing a machine learning program to predict how much electricity will the Department of Electrical and Computer Engineering of Coimbra's University (DEEC) consume each day of the year.

With the variables referred above (day, month, weekend, season, temperature and humidity) as input neurons, and the energy consumption, from the DEEC building towers, (R, S, T and B) as output neurons, an ANN its trained with two goals: (i) To compare the values predicted with the values trained; with the final objective to visualize the error in the predictions; and (ii) to make queries about predictions outside the data utilized to train; as we can observe from fig 1.1

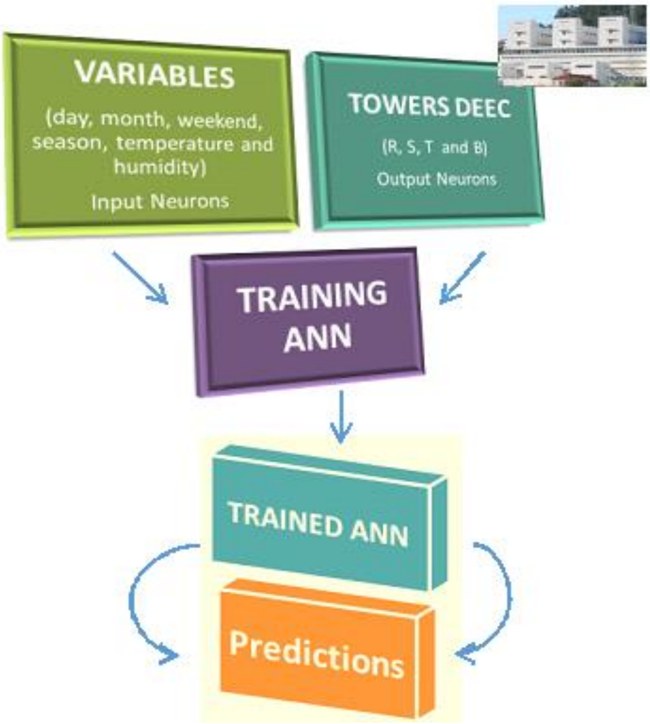

Fig.1.1 Representative Workflow study – Basic representation of how an artificial neural network can predict values

1.2 Related work

The domains in machine learning are vast and expanding every year. Today, many fields of modern science use machine learning solutions to tackle problems, such as computer-aided disease diagnosis, bioinformatics, computer vision [4], and many others like energy consumption prediction. All of them have the same logic in preparation and execution, what changes is its complexity and difficulty in execution.

In disease diagnosis, there are programs with the main purpose to find the tissues of interest, and then measure and analyze whether these tissues produce lesions. Also, programs that perform organ detection from a given complex dataset with abnormalities, automatic detection of lacunas of presumed vascular origin, and even programs that detect cerebral microbleeds from MRI images [4].

Bioinformatics deals with computational and mathematical approaches for understanding and processing biological data [5]. Computer vision has a dual goal. From the biological science point of view, computer vision aims to come up with computational models of the human visual system. From the engineering point of view, computer vision aims to build autonomous systems, which could perform some of the tasks which the human visual system can perform (and even surpass it in many cases) [6].

1.3 Contributions

In summary, the contributions of this work result in a trained neural network with 6 input neurons, and 4 output neurons, represented in Fig.1.1.

With real values and a backpropagation algorithm a neural network is trained to make predictions of energy consumption in select parts of the DEEC building. This provides the basics to the expansion of the input neuron to other variables to make the prevision closer as possible to the reality.

1.4 Structure of Dissertation

This dissertation is divided into 4 chapters.

- The current chapter provides the motivations, overall goals, related research and expected contributions of this work.

- Chapter 2 will strive to explain what machine learning is, how does the computer learn, what are artificial neural networks and what tools are used to create machine learning solutions
- In Chapter 3, will explain the implementation and results, first we explain how the program works and analyze the results by comparing the predictions with the real results.

- Finally, in Chapter 4 we will draw conclusions and propose future work.

2 Machine learning

Index

Machine learning (ML), also often *data mining*, *computational intelligence*, or *pattern recognition* [5], is a computing science that evolved from pattern recognition and from the theory of computational learning [7].

Nowadays, machine learning is a major success factor in the ongoing digital transformation across all industries. With machine learning, startups and behemoths alike can make new products that can learn to perform their intended task better, faster and more intelligently than humans ever could [7]. They work using historical data to train a program so that it can learn what it as to do.

There are several tools nowadays available for the programing, training and development of machine learning algorithms. These tools can perform the same tasks but depending the engineer background some tools may be better suited than others.

2.1 What is Machine learning

Machine learning is not a new area. It has existed since the 1970s, when the first related algorithms were developed [7] and is becoming more and more used in all sorts of fields nowadays [8].

The expansion in computing power has allowed us to use machine learning, to tackle more complex problems. The increase of collected/stored data has granted the ability to apply machine learning solutions to an increasing expansive range of domains, such as [7, 12, 14]:

- Security heuristics that distill attack patterns to protect, for instance, ports or networks;
- Image analysis to identify distinct forms and shapes, such as for medical analyses or face and fingerprint recognition;
- Deep learning to generate rules for data analytics and big data handing, just as the ones used in marketing and sales promotions;
- Object recognition and predictions from combined video streams and multisensory fusion for autonomous driving;
- Pattern recognition to analyze code for weaknesses like criticality and code smells (which is any characteristic in the source code of a program that possibly indicates a deeper problem);
- Mean sea level pressure, wind speed, and relative humidity can be predicted by utilizing artificial neural networks;
- The machine learning methods designed for high penetration level of photovoltaic (PV) power prediction included Artificial Neural Networks (ANNs), Support Vector Regression (SVR) and Regression Trees (RT);

Overall, the general idea behind most machine learning technics is complex, but essentially the same. A computer learns to perform a task by studying training set of examples, so then it can perform the same task, with new data, that it has not been encountered before, and therefore, give us an expected data [7].

2.2 How does the computer learn?

Training ANNs is generally performed by applying a learning model/strategy to a cost function [14]. There are two types of learning strategies that gives to the computer the ability to learn and train. Those strategies are *Supervised Learning* and *Unsupervised Learning* [7].

Supervised Learning happens when a machine learning program utilizes real live input and output datasets for training. This type of learning is very much like giving students a problem and a way to solve it, so that, when a similar problem appears, they can figure out its solution [7].

On the other hand, *Unsupervised Learning* consist in training a machine learning program with only input data, expecting that the computer will figure out what the solution is. This type of learning is like giving a student a set of patterns and asking him/her to figure out the underlying motifs that generated the patterns. [7]

Within these two types of learning strategies there are sub-types known as *applications*, and in each application, *algorithms*, exemplified in figure 2.1.

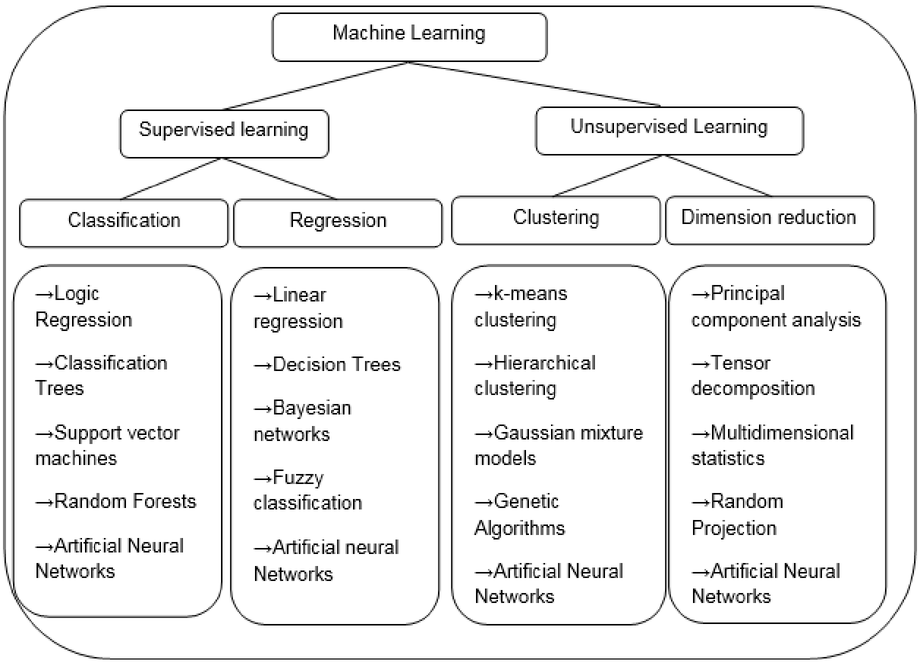

Fig. 2.1. Machine-learning approaches – In machine learning a computer learns how to perform a task by first being trained, in the figure we can see the types of learning that exist, its subtypes and what type of algorithms there are in existence [2].

Within *supervised learning strategies* we can divide all applications in *classification* algorithms and *regression* algorithms.

Classification algorithms take inputs from a dataset and the class of each piece of data, so that the computer may learn to classify new data. For example, presenting the image of a number figure and let the program to identify the number value.

For these classification type problems, we can utilize logic regression, classification trees, support vector machines, random forests, and artificial neural networks (ANNs) as tools to solve such problems.

Regression algorithms are used when the objective is to predict a value of an entity, for example, predicting what is going to be the consumption of electricity in a certain building. To solve regression type problems, *regression* algorithms include Linear regression, decision trees, Bayesian networks, Fuzzy classification, and ANNs.

It is important to clarify that sometimes supervised learning must deal with *under* or *overfitting* issues.

An underfitting model easily captures the complex patterns in data such as linear and logistic regression; while an overfitting model is more complex, like decision trees [23]. *Underfitting* happens when a model it is incapable of capturing the underlying pattern of data, usually models with high bias and low variance. This happens when the amount of available data to build an accurate model is too small or it is used a nonlinear data in order to create a linear model [23].

Overfitting, on other hand, happens when the model captures the noise along with the underlying pattern in data, models with low bias and high variance. This situation occurs when the model it is trained many times over noisy dataset [23].

Within unsupervised learning algorithms we can divide all applications in *clustering* algorithms and *dimensionality reduction* algorithms.

Clustering algorithms take inputs from a dataset covering various dimensions and divide them into clusters satisfying certain criteria. To solve clustering type problems, clustering algorithms include hierarchical clustering, Gaussian mixture models, genetic algorithms (in which the computer learns the best way for a task through artificial selection), and ANNs as tools to solve them.

A dimensionality reduction algorithm takes the initial dataset that covers various dimensions and project the data to fewer dimensions. These fewer dimensions will then try to better capture the data´s fundamental aspects. To solve dimensionality reduction problems, we have principal component analysis, tensor decomposition, multidimensional statistics, random projection, and ANNs as tools to achieve a resolution [7].

ANN is a major player in the machine learning world for its ability to utilize any type of learning algorithm, as so will be used in this dissertation to help conquering its goals.

2.3 What are Artificial Neural Networks

Neural network analysis (NNA) was proposed nearly 50 years ago by *Warren McCulloch* and *Walter Pitts* [9], but it is only in the last 20 years that software applications have been developed to handle practical problems from mathematics, engineering, medicine, economics, meteorology, psychology, neurology and many others fields. ANNs is one of the most widely used solutions for energy prediction problem [15].

Some of the most important fields are voice recognition, analysis of electromyography and other medical signatures, identification of military targets, and identification of explosives in passengers' suitcases. They are also used in weather trends forecasting, prediction of mineral exploration sites, electrical and thermal load prediction, adaptive and robotic control, and many others.

ANNs are a great tool because they can build predictive models from past data collected by sensors, making it great for process control, so they can be used as an alternative method in engineering analysis. They are a mathematical and computer mimic of the human brain [10] [15]. The way the network is trained, it requires no detailed information about a system. Instead, it learns by analyzing the relationship between the input data, controlled and uncontrolled variables and the output data. From there, after the creation of a network of connections between neurons, neural networks can predict and work for the job that it was created for [10].

In this chapter, it will be discussed the most used neural network configuration (Fig. 2.2), known as multilayer perceptron's, together with the concept of basic backpropagation training.

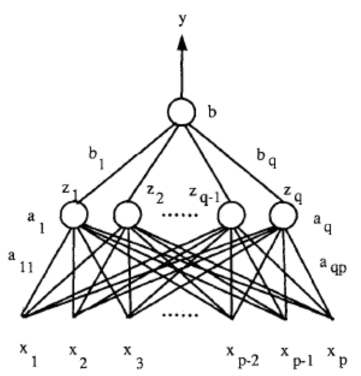

Fig. 2.2 Neural network – Example of neural network with x_p inputs and y output [16]

2.3.1 Multilayer Perceptron's

According to Wang et al. (2005), the multilayer perceptron is the most popular type of neural network in work today. They belong to the family of feedforward neural networks, a type of neural network capable of approximating generic classes of functions, including continuous and integrable functions [11].

Multilayer perceptron has a minimum of 3 layers of nodes named, *input layer*, *hidden layer* and *output layer*. In the *input layer,* it exists the input nodes that are connected to all the nodes in the *hidden layer* (Fig 2.3), and the hidden nodes to the output nodes. Every single node is called a *neuron*. Every neuron relates to the next layer neurons, and every connection has a corresponding *weight*, *bias* and *activation function* [11]

The idea of *weight*, also known as synaptic weight, is a foundational concept in artificial neural networks. The w*eight* represents a factor by which any values passing into the neuron are multiplied [21]. To each connection, it is assigned a w*eight* that represents its relative importance [11]. A set of weighted inputs allows each artificial neuron or node in the system to produce related outputs [21].

Activation function, also called *transfer function,* is the function in an artificial neuron that delivers an output based on its input [11]. This activation function can be linear, discrete, or some other continuous distribution function. The *activation function* is the backbone of the multilayer perceptron training, a function with differentiable properties. The ideal function is a *sigmoid function*, the function generally used in most feed forward neural network applications [20].

In order to train a network is used an algorithm technique called *Backward Propagation.* Backpropagation was derived by multiple researchers in the early 60's and was rediscovered and popularized by Rumelhart & McClelland (1986) [20]. It is currently the most common approach to training feed forward ANNs [20]. In its training phase, the values for the weights, activation functions and bias are determined by utilizing historic data, so that the neural network learns to give the right answer [20,22].

Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. In any model, prediction is pivotal to detect and understand de prediction errors in order to minimize and avoid mistakes that can put at stake the accuracy of the built models. Model with high *bias* pays very little attention to the training data and oversimplifies the model and it always leads to high error on training and test data. An ideal model with good balance presents a low *bias* and a low *variance*. It is important elucidate the notion of *variance*, which is the variability of model prediction for a given data point or a value which tells us spread of our data. In an ideal model, as referred earlier, a low bias and variance avoid the dangers of over fitting and under fitting. This kind of model displays a low *total error* despite the *irreducible error* that can never be changed. *Irreducible error* is the error that cannot be reduced even in good models; it is a measure of the amount of noise in our data that is never reduced [22].

To sum up the basics of al process it is important to know some fundamental steps:

1. Based on historic data the network calculates what it assumes to be the outputs.
2. The resultant outputs from the network are compared with the expected outputs
3. Weights, biases and error of each neuron are adjusted, as many times as needed in order to improve the network results.

At last, this constant adjustment flow shows the network's learning ability, the main core of all experiment [20].

In the next sub-chapters, we will explain in more detail all the system.

2.3.2 Multilayer Perceptron's Structure

As mentioned before, a multilayer perceptron structure consists of an *input layer*, one or more *hidden layers*, and an *output layer,* as shown in fig. 2.3. In this figure, we can see a very basic neural network, with two neurons in the *input layer*, two *hidden layers* with four neurons each, and two neurons in the *output layer* with its connections.

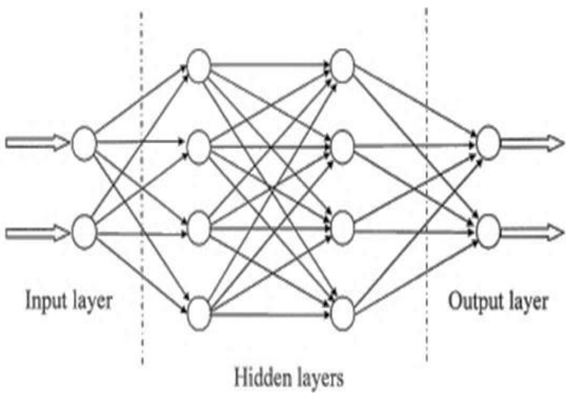

Fig. 2.3 Schematic of a fully connected multilayer perceptron's neural network with two inputs and two outputs and layer representation [23]

As we can clearly see in figure 2.4, every single neuron is connected to every neuron in next layer.

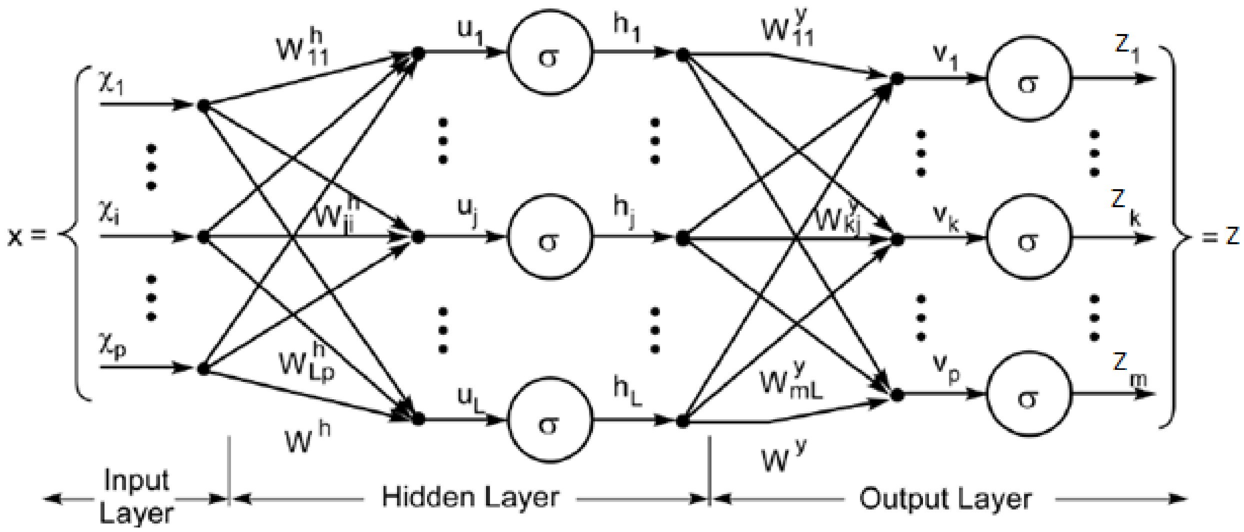

Fig 2.4 Multilayer Perceptron´s Structure with named neuron, inputs and outputs adapted from [24]

Each arrow has a certain weight, w_{ij}^{l}, which represents how much the connection from j^{th} neuron of the *l*-1th layer can influence the i^{th} neuron of the l^{th} layer (Fig.2.5). x_i represents the input of the multilayer perceptron and z_i^l the output of the i^{th} neuron of the l layer of the multilayer perceptron.

Every connection between neurons also includes a w_{i0}^{l} that is an extra weigh parameter that represents the bias for i^{th} neuron of l^{th} layer. As such, w of multilayer perceptron includes $\mathrm{w_{ij}^{l}}$, $\mathrm{j} = 0, 1, \ldots, \mathrm{N_{l-1}}, \mathrm{i} = 1, 2, \ldots, \mathrm{N_l}$, l=2,3,...,L, that is, equation (1) represents all the weights and bias of all the connections to the first neuron of the second layer [11].

$$w = [\, w_{10}^{2}\; w_{11}^{2}\;\; w_{12}^{2} \ldots ,\; w_{N_L N_{L-1}}^{2} \,]^{T} \tag{1}$$

2.3.3 Information processed by a neuron

In a neural network, every neuron has a function. The neurons in the *input layer* inserts data from a dataset to the network, the *output layer* neurons give an answer to said data in concordance with all previous neurons and the rest of the neurons in the *hidden layer* process inputs from the former layer of neurons and gives output data to the next layer.

In Fig. 2.5, it can be seen an illustration of the way in which a neuron in a multilayer perceptron processes information.

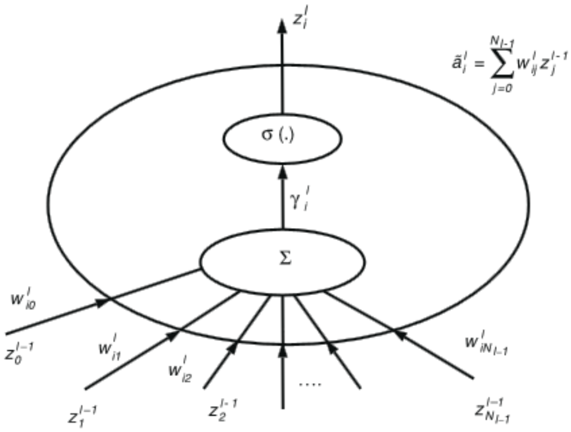

Fig. 2.5 Information processing by i^{th} of the l^{th} layer - In this figure it is seen a representation of an individual neuron from its inputs from other neurons to its output

For example, let's use the first neuron of the second layer, presented in Fig. 2.4. It receives stimuli from the neurons in the input layer, that is x_1 to x_p . At first, each input is multiplied by the corresponding weight. Then, the corresponding result from each connection is added to produce a weighted sum γ. This weighted sum is then passed through a neuro activation function, σ (), to produce the output for that neuron. This output, z_1^2, becomes the input stimulus for the neurons in the next layer from that neuron.[11].

2.3.4 Activation Functions

Activation functions define the neuron output. In classification problems de neuron output has binary values (zero or one) and in regression problems de neuron output comprehends the interval [0,1].

The most used hidden neuron activation function is the **sigmoid function** $\sigma(\gamma)$ given by the equation (2):

$$\sigma(\gamma) = \frac{1}{(1+e^{-\gamma})} \tag{2}$$

As shown in fig.2.6, equation (2), the sigmoid function is a smooth switch function having property of (3)

$$\sigma(\gamma) \rightarrow \begin{cases} 1 \ as \ \gamma \rightarrow +\infty \\ 0 \ as \ \gamma \rightarrow -\infty \end{cases} \tag{3}$$

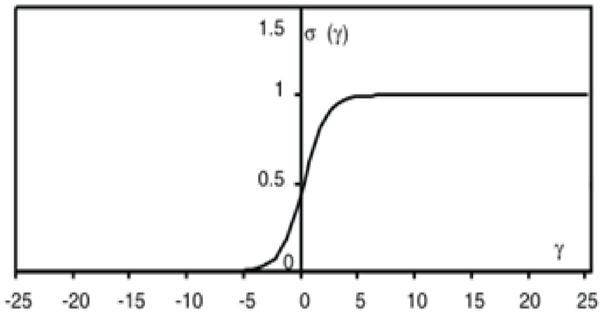

Fig. 2.6 Sigmoid function – Figure of a Standard logistic sigmoid function

But the sigmoid function is not the only one used today for we have also several others like. The **Arc-tangent function** shown in figure 2.7 and given by the equation (4) below:

$$\sigma(\gamma) = \frac{2}{(\pi)}\arctan(\gamma) \tag{4}$$

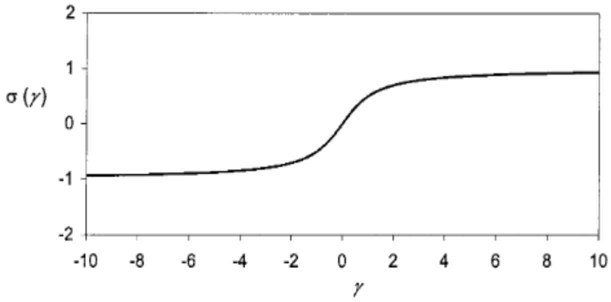

Fig. 2.7 Arc-tangent function- Principal values of the $\frac{2}{(\pi)}\arctan(\gamma)$ function

The **Hyperbolic-tangent function** shown in Figure 2.8 and given by the equation (5):

$$\sigma(\gamma) = \frac{(e^{\gamma} - e^{-\gamma})}{(e^{\gamma} + e^{-\gamma})} \tag{5}$$

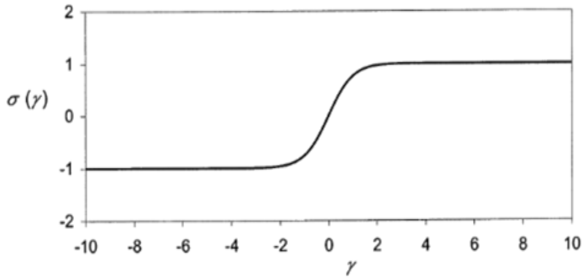

Fig. 2.8 Hyperbolic-tangent function - Principal values of the $\frac{(e^{\gamma} - e^{-\gamma})}{(e^{\gamma} + e^{-\gamma})}$ function

All these logistic functions and many more are bounded, continuous, monotonic, and continuously differentiable. But the activation functions for output neurons can either be *logistic* functions (e.g., sigmoid), or *simple linear* functions that compute the weighted sum of the stimuli. The **linear activation function**, equation (6), is defined as:

$$\sigma(\gamma) = \gamma = \sum_{j=0}^{N_{L-1}} w_{ij}^{L} z_j^{L-1} \tag{6}$$

The use of linear activation functions in the output neurons can help to improve the numerical conditioning of the neural network training process back propagation [11].

2.3.5 Effect of Bias

The weighted sum is expressed by the equation (7):

$$\gamma_i^l = w_{i1}^l z_1^l + w_{i2}^l z_2^{l-1} + \cdots + w_{iN_{l-1}}^l z_1^l \tag{7}$$

and is zero, if all the previous hidden layer neuron responses (outputs) z_1^{l-1}, $z_2^{l-1}, \dots, z_{N_{l-1}}^{l-1}$ are zero. In order to create a bias, we assume a fictitious neuron whose output is $z_0^{l-1} = 1$ and add a weight parameter w_{i0}^{l-1} called bias.

The weighted sum can then be written as equation (8)

$$\gamma_i^l = \sum_{j=0}^{N_{L-1}} w_{ij}^{L} z_j^{L-1} \tag{8}$$

The effect of adding the bias is that the weighted sum is equal to the bias when all the previous hidden layer neuron responses are zero, equation (9), that is,

$$\gamma_i^l = w_{i0}^l, \text{ if } z_1^{l-1} = z_2^{l-1} = \cdots = z_{N_{l-1}}^{l-1} = 0 \tag{9}$$

The parameter w_{i0}^l is the bias value for i^{th} neuron in l^{th} layer as shown in Fig.2.5 [11].

2.3.6 Neural Network Feedforward

A Feed Forward Neural Network (FFNN) is defined as a simple type of neural network in which the information flow is in the forward direction from the input towards the hidden and output nodes [14].

A FFNN is used to calculate the outputs *y* from multilayer perceptron's neural networks by using inputs x_i and weights w. This is achieved by first feeding to the external inputs into the neurons of the first hidden layer, as observed in Fig. 2.3 and 2.4, until finally the information reaches the output layer of the neural network.

The computation is given by the equations (10) and (11),

$$z_i^1 = x_i \quad, i = 1,2, \dots, N_i \quad, \text{n}=N_1 \tag{10}$$

$$z_i^l = \sigma\left(\sum_{j=0}^{N_{L-1}} w_{ij}^{L} z_j^{L-1}\right), \text{ i=1,2, …, } N_l \text{ , l=2,3,…,L} \tag{11}$$

The outputs of the neural network are extracted from the output neurons as equation (12)

$$y_i = z_i^L, \text{ i=1,2,…, } N_L \text{ },m = N_L \tag{12}$$

During feedforward computation, the neural network weights w are fixed [11].

2.3.7 Number of neurons in the hidden layer

The universal approximation theorem states that a neural network with a single hidden layer with a finite number of neurons can approximate virtually any nonlinear function [11]. However, it does not specify the number of hidden neurons necessary for a given problem complexity. The precise number of hidden neurons remains an open question. Although there is no clear-cut answer, the number of hidden neurons depends largely on the degree of nonlinearity and the dimensionality of the original problem. In other words, highly nonlinear problems need more neurons and smoother problems need fewer neurons. At this point, stands out a new problem that must be addressed.

On one hand, too many hidden neurons may lead to overlearning, which happens when the neural network loses its flexibility in calculating numbers that do not exist in the training dataset. On the other hand, too few hidden layer neurons do not give enough freedom to the neural network to accurately learn the behavior of the problem.

Therefore, to solve this issue is fundamental to ask the question “How big the network really needs to be?”.

There are three possible solutions.

The first one is *user experience*, where the individual practical skills are by far the most important quality. The second one is a *trial and error process*, where the user tenacity will succor a good network. And the third and last solution is an *adaptive process* or *optimization process* that adds/deletes neurons as needed, during training [11].

2.3.8 Number of hidden layers

As stated above, for the existence of multilayer perceptron, at least 3 layers of neurons are necessary: the *input layer*, the *hidden layer* and the *output layer*. In practice, neural networks with one or two hidden layers are commonly used. Intuitively, four-layer perceptron’s perform better in modeling nonlinear problems where exist certain localized behavioral components that repeat in other regions of the problem [11].

2.3.9 Training algorithm

Next, the backpropagation algorithm, that is one of the most used training algorithms nowadays, will be briefly explained [11].

2.3.9.1 Back Propagation algorithm

As we have seen until now the main objective in all neural networks is to find the optimal set of weights (*w*) so that the output parameters *(y = (x, w))* are as close as possible to the expected values. For this, an algorithm is utilized through a process called training.

The training data are pairs of (x_k, d_k), k = 1, 2, …, P, where d_k is the desired outputs of the neural model for inputs x_k, and *P* is total number of training samples. During training, the neural network performance is evaluated by computing the difference between actual neural outputs and desired outputs for all the training samples. This difference, also known as the error, equation (12), is quantified by

$$E = \frac{1}{2} \sum_{k \epsilon T_r} \sum_{j=1}^{m} (y_j(x_k, w) \; - \; d_{jk})^2 \tag{12}$$

where d_{jk} is the j^{th} element of d_k, y_j(x_k,*w*) is the j^{th} neural network output for input x_k, and T_r is an index set of training data. The weight parameters *w* is adjusted during training, such that this error is minimized.[11]

2.3.9.2 Training Process

The first step in training is to initialize the weight parameters w being suggested small random values.

During training, *w* is updated along the negative direction of the gradient of E, as

$w = w - \; \eta \; \frac{\partial \, E}{\partial w}$, until E becomes small enough.

Here, the parameter η is called the learning rate. If we use just one training samples at a time to update *w*, the per-sample error functionE_k , equation (13) given by

$$E_k = \frac{1}{2} \sum_{j=1}^{m} (y_j(x_k, w) \; - \; d_{jk})^2 \tag{13}$$

Is used and *w* is updated as equation (14). [11]

$$w = w - \; \eta \; \frac{\partial \, E_k}{\partial w}. \tag{14}$$

2.3.9.3 Error Back Propagation

Using the definition of E_k, equation (13), the derivate of E_k with respect to the weight parameters of the l^{th} layer can be computed by simple differentiation as equation (15) and (16),

$$\frac{\partial E_k}{\partial w_{ij}^l} = \frac{\partial E_k}{\partial z_i^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \tag{15}$$

And

$$\frac{\partial z_i^l}{\partial w_{ij}^l} = \frac{\partial \sigma}{\partial \gamma_i^l} = z_i^{l-1} \tag{16}$$

The gradient $\frac{\partial E_k}{\partial z_i^l}$ can be initialized at the output layer, equation (17), as

$$\frac{\partial E_k}{\partial z_i^l} = (y_i(x_k, w) \;-\; d_{ik}) \tag{17}$$

using the error between neural network outputs and desired outputs (training data). Subsequent derivatives $\frac{\partial E_k}{\partial z_i^l}$ are computed by back propagating this error form *l*+1th layer to *l*th layer (Fig. 2.9). [11]

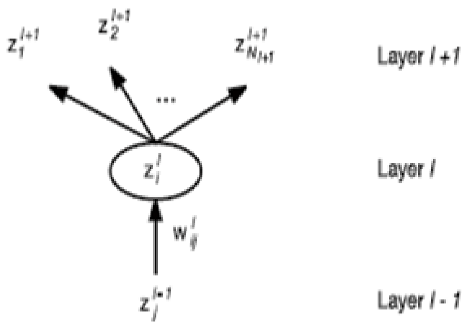

Fig. 2.9 Neuron Relations - Relationship between i^{th} neuron of l^{th} layer, with neurons of layer *l* – 1 and *l* + 1

2.4 Tools commonly used for creating machine learning solutions

With the advent and rise of machine learning popularity many tools have been developed for an easier learning and development experience as such, most are open source. In Table 1 we compare some of the most popular machine learning tools, each one with different characteristics, such as: type of license, supported languages, variety of machine-learning models and software maturity, for example. [7]

Table 1. Comparison between tools commonly used to create neural network solutions

	Tool				
	Python	R	Spark	Matlab	TensorFlow
License	Open source	Open source	Open source	Proprietary	Open source
Distributed	No	No	Yes	No	No
Visualization	Yes	Yes	No	Yes	No
Neural nets	Yes	Yes	MLP classifier	Yes	Yes
Supported languages	Python	R	Scala, Java, Python, and R	Matlab	Python and C++
Variety of machine-learning models	High	High	Medium	High	Low
Suitability as a general-purpose tool	High	Medium	Medium	High	Low
Maturity	High	Very High	Medium	Very High	Low

For numerical and statistical problems, the communities are divided between two programs. Some prefer **R** and others prefer **Python**. However, an absolute division cannot be considered. The machine learning field is very wide and there is no single tool perfect for everyone. It is then best for a software engineer to become acquainted with many different tools and learn which one is the most appropriate for a given situation. [7]

Having said that, why this division?

The division happened really because of the background of its users. R is popular with people with a somewhat stronger statistical background, for it has a superb collection of machine-learning and statistical-inference libraries. Chances are if a fancy algorithm is found somewhere and we want to try it on data, an implementation in R exists for it.

R also boasts a ggplot2 visualization library, which is a tool to produce excellent graphs.

On the other hand, Python is popular with users with a computer science background.

Python was not made specifically for machine learning or statistics, but it has an extensive library for numerical computing (NumPy), scientific computing (SciPy), statistics (StatsModels), machine learning (scikit-learn) [7]. These are largely wrappers of C code, so we get Python's convenience with C's speed.

Although there are fewer machine-learning libraries for Python than there are for R, many programmers find working with Python easier for they might already know the language or find it easier to learn than R. The users may also find Python convenient for preprocessing data, reading it from various sources, cleaning it and bringing it to the required formats.

For visualization, Python relies on matplotlib. We can do pretty much everything on matplotlib, but we might discover we have to put in some effort. The seaborn library is built on top of it, letting you produce elegant visualizations with little code. In general, R and Python work when the dataset fits in the computer's main memory [7]. If that's not possible, a distributed platform must be used.

The most well-known is **Hadoop**, but Hadoop has the problem of being difficult to run even simple machine learning algorithms. So, many people prefer to work at the higher level of abstraction that **Spark** offers.

Spark leverages Hadoop but looks like a scripting environment, we can interact with it using Scala, Java, Python, or R. Spark also has a machine-learning library that implements key algorithms, so for many purposes you don't need to implement anything yourself [7].

H2O is a relatively new entrant in the machine-learning scene. It is a platform for descriptive and predictive analytics that uses Hadoop and Spark. We can also use it with R and Python. It implements supervised and unsupervised-learning algorithms and a Web interface through which you can organize your workflow.

A promising development is the **Julia** programming language for technical computing, which aims at top performance. Because Julia is new, it does not have nearly as many libraries as Python or R. Yet, thanks to its impressive speed, its popularity might grow.

Strong commercial players include **Matlab** and **SAS**, both having a distinguished history. Matlab has long offered solid tools for numerical computation, to which it has added machine-learning algorithms and implementations. For engineers familiar with Matlab, it might be a natural fit. SAS is a software suite for advanced statistical analysis; it also has added machine-learning capabilities and is popular for business intelligence tasks [7].

In this dissertation, Python is utilized to construct the machine learning solution to achieve its goal because of its ease of language and learning curve.

3 Data and program analysis

Index

Summary:

How does our machine learning solution works?

What does it do?

Did we achieve our objective?

In this chapter all these questions will be answered and explained.

3.1 Program Implementation

3.1.1 Program Implementation

Before starting to explain the program implementation, it is necessary to analyze what is expected from the program.

It is expected that using a machine learning solution we can make predictions of the energy consumption of towers R, S, T, B and the overall consumption in DEEC exemplified in figure 3.1.

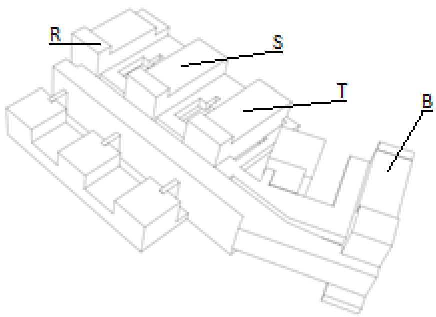

Figure 3.1 - Diagram of DEEC - In this figure it is seen the tower location as well as all the possible location in the department

What type of learning strategy is to be used to achieve the solution to the problem?

The answer to this question is very important, because it is the very core of the machine learning solution. As it was explained in section 2.2, machine learning is divided in two types of learning strategies: supervised learning and unsupervised learning the decision of what type of learning strategies used is completely depended on the data in the training phase and what problem it is expected to solve.

So, does the data expected to be used in this solution have both input and output values?

Yes, the data used has both input and output values. From this answer alone a conclusion can be taken: the program will use a supervised learning algorithm, meaning that the training strategy will be a classification type algorithm or a regression type algorithm.

Which one then?

Classification types are used when a program is expected to identify what it is, for example what number is in a picture. On the other hand. regression types are used when the program is expected to predict a value. With this in mind, it is identified that the current task of this dissertation will need a regression type algorithm since predicting values is exactly the essence of the problem to solve.

Therefore, the program to implement will use a supervised learning strategy utilizing a regression algorithm. Once the learning strategy is defined, the range of alternatives regarding possible algorithms for implementing the learning machine is then reduced.to a smaller amount. In figure 2.1, we can see some of the languages that can be utilized for the expected effect but as stated above artificial neural networks are the one that it is used.

From here, it is possible to start developing the program, because without the understanding of all this, many versions can be made and all of them wrong because this simple exercise was not made.

So, Python language was selected as the preferred tool for implementing the required solution regarding the aim of this dissertation. It's ease of use, plus the support of a large community, in particularly to those that are new in the field, were relevant in this choice.

3.1.2 Training set

The developed solution consists of two code files, the "Neural network" file (Appendix A) and the "Test" file (Appendix B), and the required training sets.

The training sets are the values used to train the neural network and, as stated above, they have input and output values. In appendix D, it is possible to see all the values in the training of the neural network for 2016 of the datasheet.

The data is presented in the datasheet and it is organized as follows:

- Number of month for example July is 7
- Number of day
- If it is weekend it is 1 if not is 0
- Season spring being number 1, summer number 2, autumn number 3 and winter number 4
- Humidity in percentage
- Temperature in ºC.

The input values used were chosen for their practicality: month, day, weekend (Saturday/Sunday), season, temperature and humidity are all the input values used in this work.

All input values used have their way of influencing how much energy the department consumes. For example, it is expected that, during a weekend the energy consumption will go down because there are no classes in session. If the temperature is low, the energy consumption will go up because, eventually, heaters are turn on. The same can be said about humidity: high humidity will make consumption also go up because it can be raining or the humidity is making a hot tropical day. Same applies to the season of the year: It can be expected that in winter the energy consumption will be higher because of the lower temperatures.

The output values used are the energy consumption of towers R, S, T, B and overall consumption of the building. In the R tower of the DEEC (Fig.3.1), there are several rooms that are mainly laboratories with electrical motors, and, for this, this tower has the biggest power consumption of all four towers. The S tower has also laboratories, but these do not have high electrical consumption equipment's as they are mainly computer labs and minimal electronic labs. The T tower has mostly lecture rooms, and B tower is where it is located the study rooms. Finally, the overall consumption is all the consumption of whole department. It is important to clearly state that the overall consumption is the consumption of all the towers and the rest of all locations.

3.1.3 Main program

The program is divided into 2 files, Appendix A is all the code from file *Neural_Network.py* and it is half of the developed code.

This part of the code has the objective of creating, training and then saving the neural network that is going to be used to make predictions.

The libraries used, listed in Figure 3.2 and 3.8, are *pytorch*, *numpy*, *pandas* and *Scikit-learn* (*sklearn*). These libraries are used in the developed software in tasks that range from the creation of the neural network to its utilization.

```
import torch.nn as nn
import numpy as np
import pandas as pd
from sklearn import preprocessing
```

Figure 3.2 Library of "Neural network" file – Code for the library of the neural network file

Pytorch is a *Python* implementation of the Torch machine learning framework that has enjoyed a broad uptake at *Twitter*, *Carnegie Mellon University*, *Salesforce* and *Facebook* [17].

Numpy is the base data structure used for data and model parameters [18]. ***Pandas*** is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive [19].

Scikit-learn is a Python module that integrates a wide range of state-of-the-art machine learning algorithms, for medium-scale supervised and unsupervised problems [19].

So, by using *pandas*, the excel file with the training set is uploaded and saved into the variable name "dataset". Afterwards, this variable is divided into 11 other variables so then they can easily be handled to create the array that is used in the training of the neural network.

At first, it may seem that it would be faster to simply make the array from the variable "dataset". However, this variable has all the training data, even outputs that are not used in certain neural networks (for example, using output data for tower B is counterproductive if it is being created a neural network for tower R). So, it is faster to simply change the output used in the array making faster creation and testing of neural networks. This part of the code can be analyzed in figure 3.3.

```
dataset = pd.read_excel('datasheet.xlsx', skiprows=1)
ano = dataset['ano']
mes = dataset['mês']
dia = dataset['dia']
sabdom = dataset['sab/dom']
estacao = dataset['estacao']
temp = dataset['temp']
humidade = dataset['humidade']
torrer = dataset['torrer']
torres = dataset['torres']
torret = dataset['torret']
torreb = dataset['torreb']
consumo = dataset['consumo']
z = np.array(np.column_stack((mes, dia, sabdom, estacao, temp,
humidade, torreb)))
```

Figure 3.3 Data Processing – Data processing part of the "neural network" file

After defining the required dataset for training the neural network, it is necessary to normalize the data into values between 0 and 1. This scaling is one of the main parts of an ANN learning process. If the inputs are not between (0,1) or (-1,1) the program cannot equally distribute the importance of each input, thus naturally large values become dominant making the ANN training ineffective. So, in Figure 3.4 the array in scaled using a minmax scaler algorithm (eq 18) to achieve that objective.

$$x' = \frac{x - min\,(x)}{max(x) - min\,(x)} \tag{18}$$

Finally, the data is divided into inputs and outputs, and converted into torch tensors, which is a container that can house data in *N* dimensions, so that they can be used in the training of the neural network.

Having the data ready to be used the neural network need now to be created, so that it can be trained, and predictions found.

```
Scaler = preprocessing.MinMaxScaler()

scale = Scaler.fit_transform(z)

X_train = scale[:, [0, 1, 2, 3, 4, 5]]
Y_train = scale[:, [6]]

"Transforming data from numpy to torch format"

xtrain = torch.Tensor(X_train)
ytrain = torch.Tensor(Y_train)
```

Figure 3.4 Scaling – Code for the scaling part of the “neural network” file

```
model = nn.Sequential(nn.Linear(6, 1000),
                      nn.Linear(1000, 1000, bias=True),
                      nn.Linear(1000, 1000, bias=True),
                      nn.Linear(1000, 1000, bias=True),
                      nn.Linear(1000, 1000, bias=True),
                      nn.Linear(1000, 1000, bias=True),
                      nn.Linear(1000, 1), nn.Sigmoid())
```

Figure 3.5 neural network creation – Code for the neural network creation of the “neural network” file

In Figure 3.5, it is presented the code to create the neural network. Having the name “model” it is created with six input neurons, five hidden layers and one output neuron. The arguments of nn.linear define the inputs and outputs of each layer as such it is necessary to always be careful that the number of outputs from one layer is the same to the number of inputs in the next layer so that no error is generated. The number of neurons used in each hidden layer is 1000 making figure 3.6 the final architecture of the neural network.

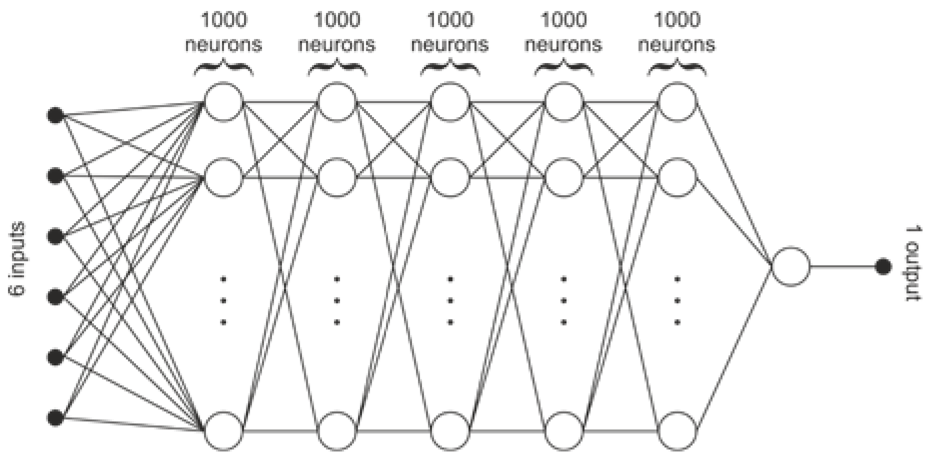

Figure 3.6 Representation of neural network architecture – In figure it is possible to observe a representation of the neural network created having 6 inputs neurons, 5 hidden layers with 1000 neuron in each layer and 1 output neuron

The number of neurons in each hidden layer was chosen because of the good results that it gave in testing but also by trial and error, one of the ways to find the number of neurons in a layer as explained in section 2.3.7.

Finally, the activation function used is the sigmoid function, as defined by the last argumento of "nn.sequencial". It is important to note that, even with only on "nn.sigmoid", this one works in all the layers.

After preparing the training data and creating the neural network,it is possible to start training the ANN.

```
criterion = torch.nn.MSELoss()  #"Mean Squared Error Loss"
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)

for epoch in range(2000):
    # Forward Propagation
    y_pred = model(xtrain)
    # Compute and print loss
    loss = criterion(y_pred, ytrain)
    # Zero the gradients
    optimizer.zero_grad()
    # perform a backward pass (back propagation)
    loss.backward()
    # Update the parameters
    optimizer.step()
```

Figure 3.7 Training neural network – Training neural network segment of the "neural network" file

In Figure 3.7, it is listed the code used to train the neural network. This code makes the previous created neural network run for 2000 epoch (cycles) in which first the data is forward through the neural network newly created, to then calculate the error and finally through backpropagation training adjust the values in every connection. The number of epochs chosen is again, like before in the number of neurons in each layer, a case of trial and error were the objective is to find a value that gives values of quality and importance.

Finally, the parameterization of the values in the dataset is saved in a file (Scale.txt) and the trained neural network is also saved in a file (Neural Network.txt), both files will be paramount in the next part of the code (Figure 3.8).

```
torch.save(model, "Neural network.txt")
torch.save(Scaler, "Scale.txt")
```

Figure 3.8 Saving data – Code segment from “neural network” file were the neural network and the scale is saved

To recap, the first half of the code creates a neural network 7 layer deep; it then trains that neural network with prepared data to finally save it so that predictions can be made.

```
import torch
import numpy as np
import pandas as pd
from sklearn import preprocessing
```

Figure 3.9 Library of the “Test” file – Code segment were the libraries for the test file is presented

Now the second half of the code, is used to make the prediction. It is very similar to the first half, both in libraries (Figure 3.9) and data preparations (Figure 3.10) but in this part of the program the values in “Scale.txt” are used to parametrize the input data. This makes it so that the data used in the prediction are in the same scale as the data used in training. Of course, this brings with it a certain error, for if the input data has an even smaller minimum or a higher maximum this will make those values default to zero and one respectively. It is important to mention that if the error does exist the next time the neural network is trained with the new data that error will be mitigated. Another difference in the preparation of the data is that there is no output data to prepare since that is what it is wished to be found.

```
model = torch.load("Neural network.txt")
Scale = torch.load("Scale.txt")



datatest = pd.read_excel('testedatasheet.xlsx', skiprows=1)
anoteste = datatest['ano']
mesteste = datatest['mês']
diateste = datatest['dia']
sabdomteste = datatest['sab/dom']
estacaoteste = datatest['estacao']
tempteste = datatest['temp']
humidadeteste = datatest['humidade']

xtest = np.array(np.column_stack((mesteste, diateste, sabdomteste,
estacaoteste, tempteste, humidadeteste, humidadeteste)))
```

Figure 3.10 Load and Data processing – Code segment for the loading and processing of data for the “test” file

Ergo, after preparing the input data, it is forwarded through the neural network to obtain predicted values (Figure3.11), however the predictions made are values between zero and one. At this moment, the file “Scale.txt” comes into play again, in order to convert the predicted values into real data. This is achieved by making a matrix with both the input and output values, to then make the conversion (Figure 3.11), finally that data is saved in an excel file (Figure 3.12) to then be analyzed and studied in how effective the solution using a machine neural network in python is.

```
"Processes the data into numbers between 0 and 1"

xtestscale = Scale.transform(xtest)
X_test = xtestscale[:, [0, 1, 2, 3, 4, 5]]

"Transforming data from numpy to torch format"

Xtest = torch.Tensor(X_test)



"Testing neural network"
# Forward Propagation

data = model(Xtest)

y_output = data.detach().numpy()
matrix_no_scale = np.column_stack((Xtest, y_output))
output = Scale.inverse_transform(matrix_no_scale)
```

Figure 3.11 Data transformation – Code segment were the data is transformed to usable values

```
"convert your array into a dataframe"
df = pd.DataFrame(output)
"save to xlsx file"
filepath = 'output.xlsx'
df.to_excel(filepath, index=False)
```

Figure 3.12 Data saving – Code segment for the "test file" were the predict output values are saved

In conclusion when the user wants to make a prediction it is only necessary the insertion of data into the excel file "testedatasheet", run the neural network previously trained and view the predicted data in the output file.

3.2 Experimental data analysis

The training data, seen in Fig C1 to C7 in Appendix C, is data used to train the neural network This data goes from 2015/July/10 to 2017/December/31 which corresponds to 906 days, and in each day there are six input values and one output value for each tower and overall consumption. This data was obtained by using the values registered in the department database and the reason why is only used this amount is because it was all the available data at the time.

The amount of data used is a small sample in comparison with other similar works. For example, in a similar energy consumption problem, the data used by the authors comprise consumptions from over 426.305 homes in Bexar County, Texas (TX) with four years of monthly consumption [1].

This indicates that there might be some discrepancy between the real-life data and the predicted data that can be correlated to the small amount of training data but also to the impossibility of having all the possible input variables correlated to energy consumption. However, this is not a total obstacle to try to obtain some predictions from the data available.

As it can be observed in Figure 3.13, 3.14, 3.15, 3.16 and 3.17, these are the comparison between both real and predicted data for the month of January 2018. This data was obtained by inputting into the trained neural network already observed data, having the neural network save the output data and the input data into an excel file where it is analyzed and compared with the real data.

It is possible to easily verify that the program learned the cycle of weekend and weekday; as expected with the large amount of sample data in our training set but the values predicted have an expected discrepancy between real and predicted with some having a bigger discrepancy than others (fig 3.15, 3.16, 3.17).

When analyzing the figures in question (Figure 3.13, 3,14, 3.15, 3.16, 3.17) all generally present the same evolution, each of them with a major or minor discrepancy. Analyzing Figure 3.13 and 3.14 we see that the raises and falls of the predicted and real values are parallel with a similar average. However, observing the obtained data from figure 3.16, 3.17 and 3.18 the same does not occur. There is a larger discrepancy in each of the last three figures (figure 3.16, 3.17, 3.18). The situation in question can be explained from the large instability in the training sets (Figures C1, C2, C3, C4, C5 in Appendix C).

In the training values used to obtain the predictions presented in figures 3.13 and 3.14 (figures C1 and C2, Appendix C), it is easily noticed when a class is in section or not.

From the data available for towers T and B (figures C3 and C4, Appendix C), it is possible to observe abnormal behaviors on the power consumption. For Tower T, there are presented some extremely low values of power consumption, and for Tower B it is noticeable a large period of unchanged power consumption, which is indicative of an eventual data corruption.

This makes the neural network have a harder time to predict consumption and unfortunately brings problems to the prediction. Even though all the discrepancies and the corruption of data in Fig.C4, it is possible to collect data from the predicted numbers since they are inside the range of values expected.

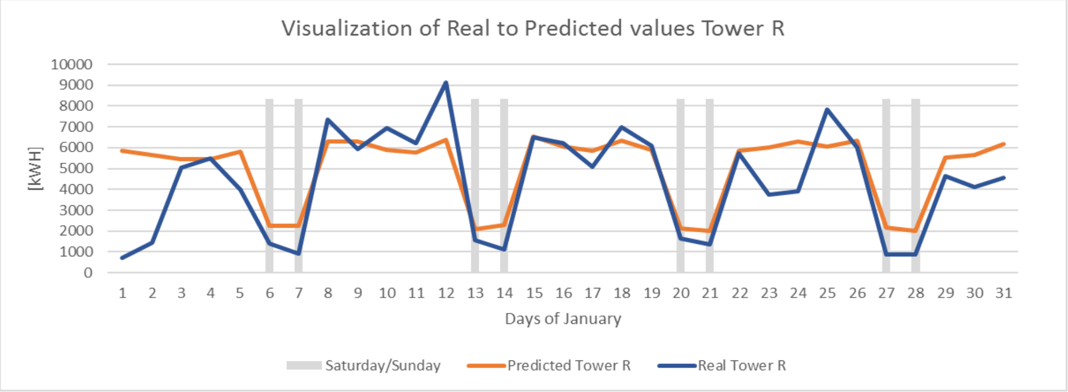

Figure 3.13- Visualization of Real to Predicted values in Tower R for January – In this graph it is observed and compared the real and predicted values of Tower R energy consumption. It is also observed that the raises and falls of both are parallel and that the predicted values already can predict and behavior of weekdays and weekends. Both average values are very close to each other.

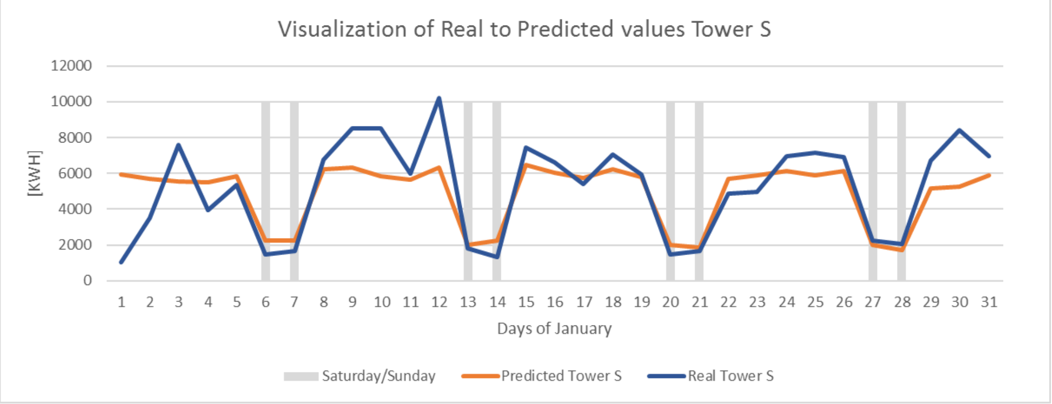

Figure 3.14- Visualization of Real to Predicted values in Tower S for January – In this figure it is observed and compared the real and predicted values of Tower S energy consumption. It observed that the raises and falls of both are parallel and that the predicted values already can predict the behavior of weekdays and weekends. Both average values are very close to each other.

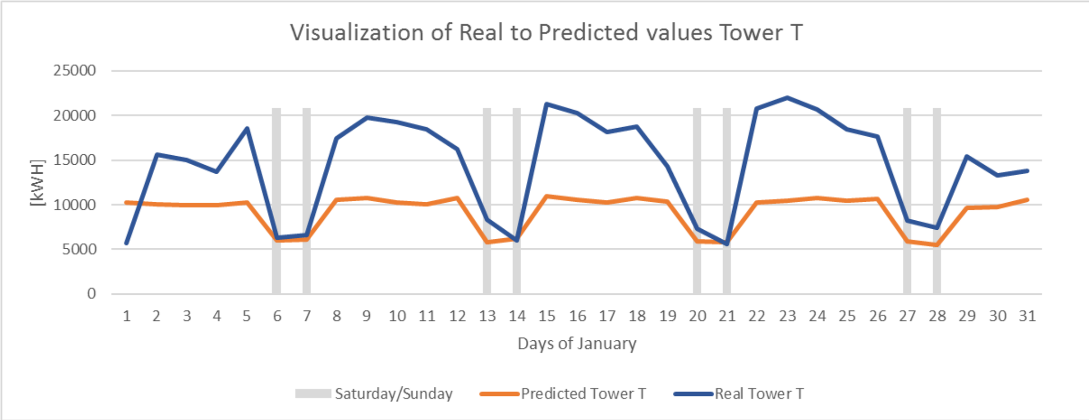

Figure 3.15- Visualization of Real to Predicted values in Tower T for January – In this figure it is observed and compared the real and predicted values of Tower S energy consumption. It is observed a large discrepancy in the values. It is also observed that the predicted values know when the days of the week and weekend are. The rises and lows of both graphs also parallel.

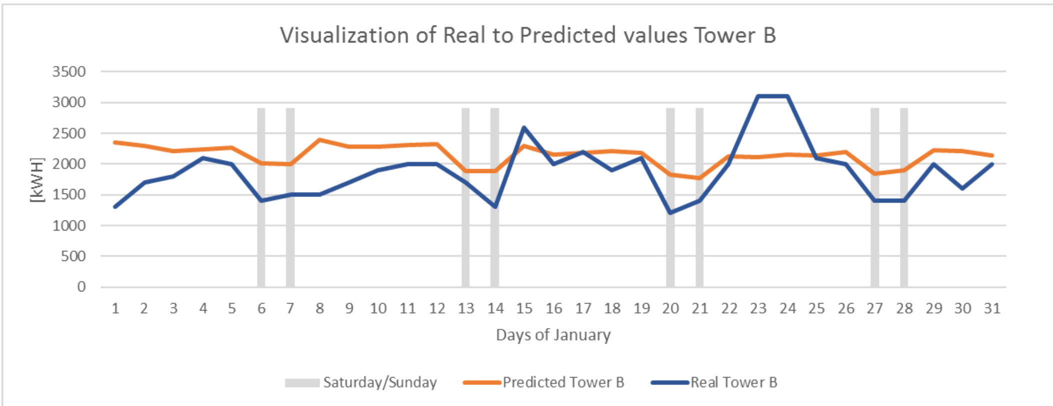

Figure 3.16- Visualization of Real to Predicted values in Tower B for January – In this figure it is observed and compared the real and predicted values of Tower B energy consumption. It is observed discrepancies in its performance.

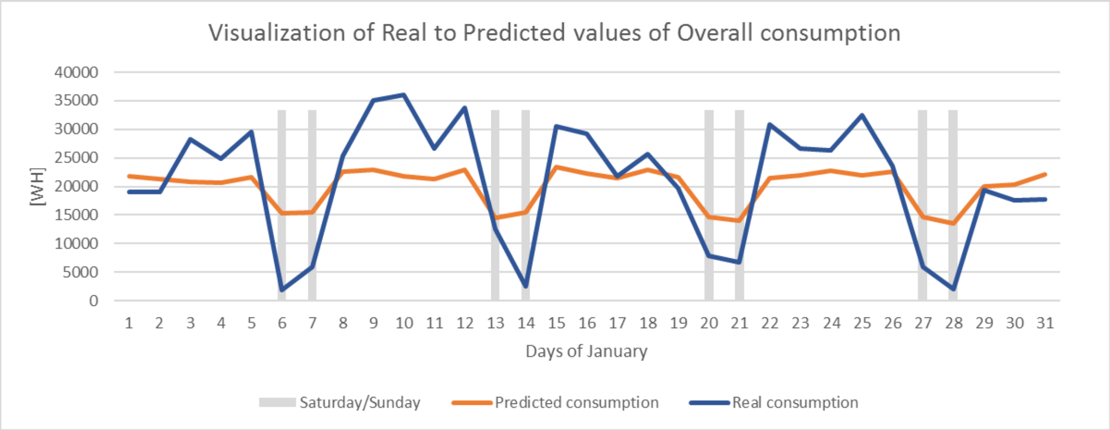

Figure 3.17 – Visualization of Real to Predicted values of Overall consumption for January – In this figure it is observed and compared the real and predicted values of the overall consumption of DEEC. It is observed discrepancies in its performance, but it is also possible view similarities between both lines in the figure.

It is important to point out that the predicted values are quite good for the amount of data used. As stated above, there are only 906 entries. This is about two and a half years of data with each entry having 6 inputs and the output expected in the prediction. The month tested above is also a time period with more erratic energy consumption, since it is timeframe where there is less students in the department and no classes, and it corresponds to exam season. January is also one of the months that has less training values since the training sets start in the middle of July. As such there is a lot of variables that the program is for now unable to cope with but with more data in the training set and an increased number of input neurons this inability will disappear in time.

The same can be observed in the month of February 2018 (Fig 3.18, 3.19, 3.20, 3.21, 3.22) in it large discrepancies are observed between the real and predicted values. These discrepancies are again explained with the low amount of values, the months used are some of the months with the most erratic energy consumption in the year but also with observed data corruption that create even bigger discrepancies reflecting on the predictions presented in the figures below.

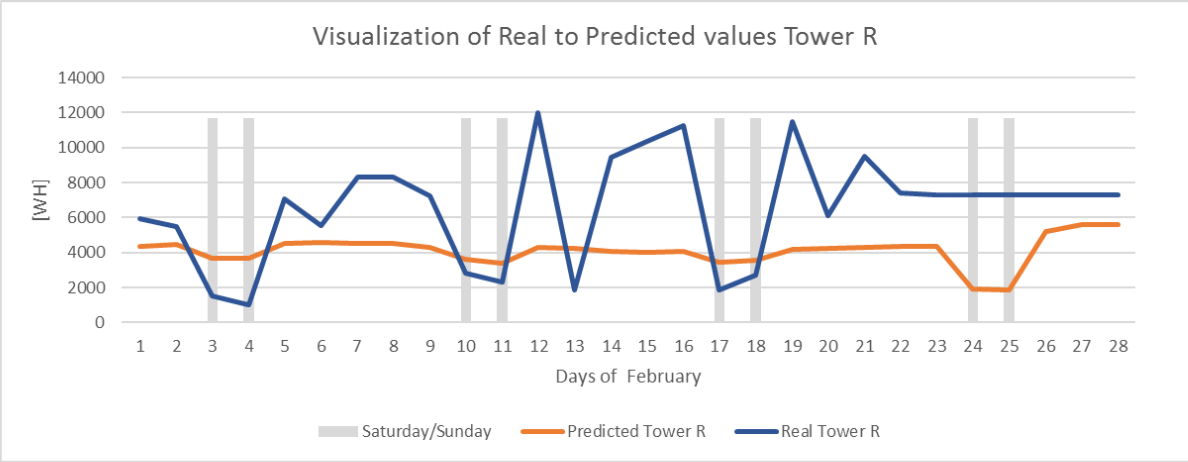

Figure 3.18- Visualization of Real to Predicted values in Tower R for February – In this graph it is observed and compared the real and predicted values of Tower R energy consumption. It is also observed that the raises and falls for the weekends are parallel, that there is data corruption in the real values and It is observed discrepancies in its performance.

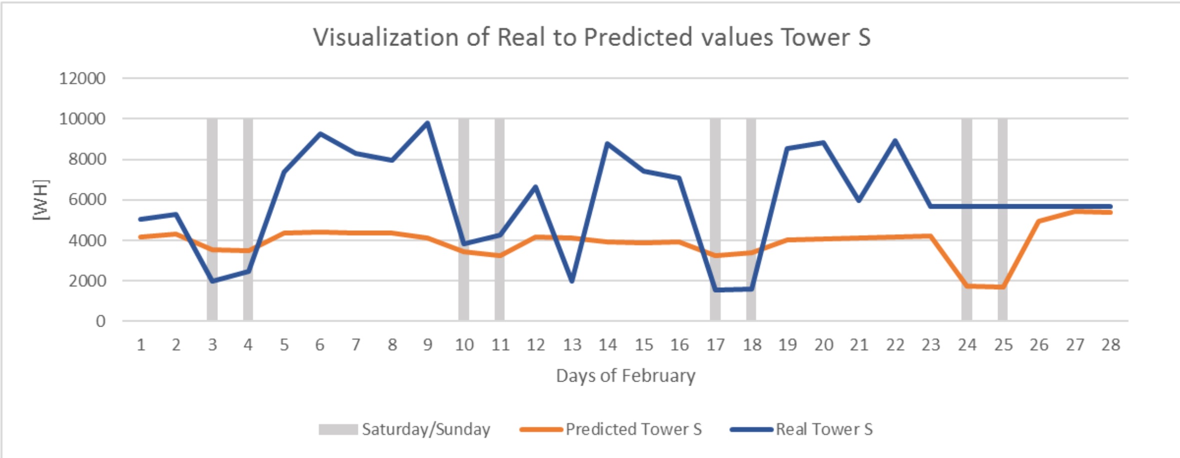

Figure 3.19- Visualization of Real to Predicted values in Tower S for February – In this graph it is observed and compared the real and predicted values of Tower S energy consumption. It is also observed that the raises and falls for the weekends are parallel, that there is data corruption in the real values, it is observed discrepancies in its performance.

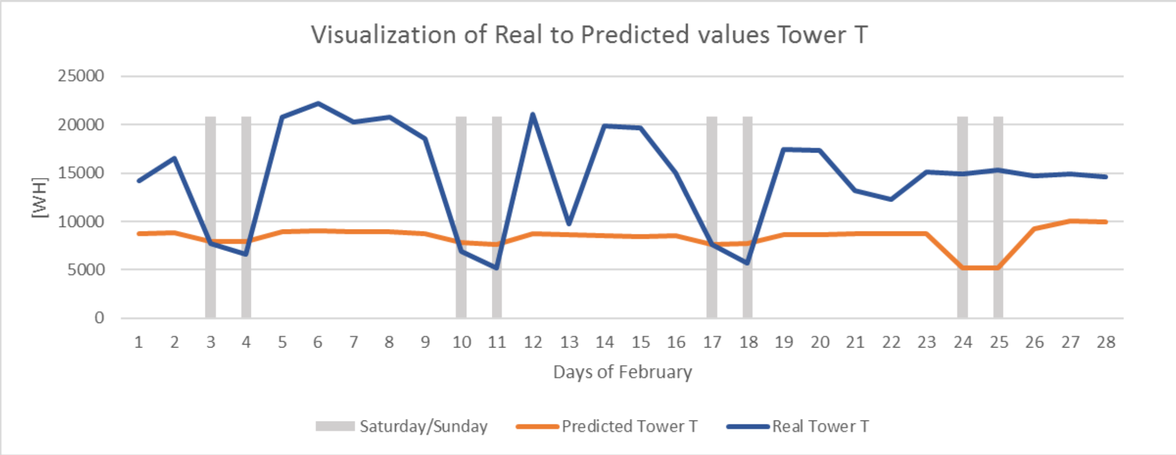

Figure 3.20- Visualization of Real to Predicted values in Tower T for February – In this graph it is observed and compared the real and predicted values of Tower T energy consumption. It is observed a large discrepancy in the values. It is also observed that the predicted values know when the days of the week and weekend are.

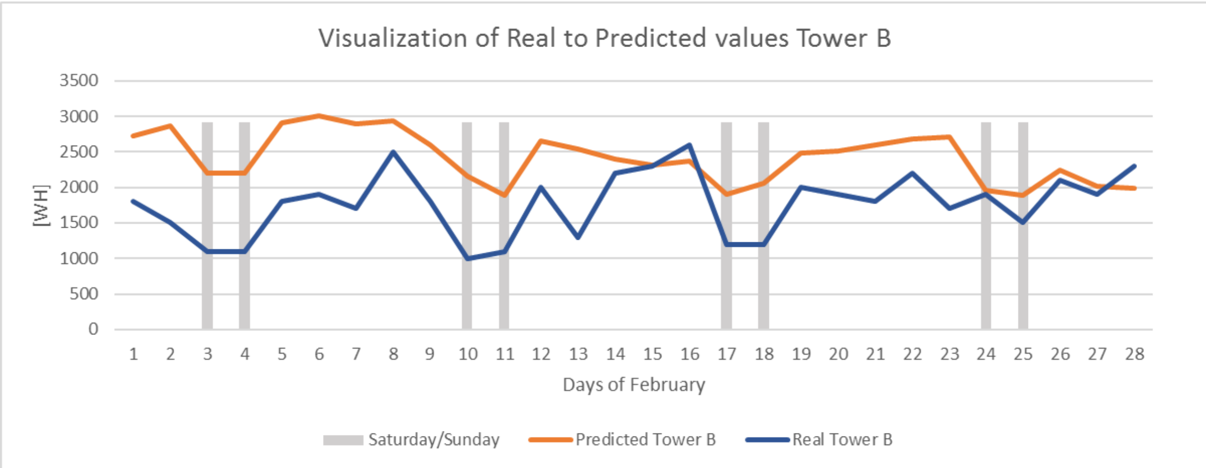

Figure 3.21- Visualization of Real to Predicted values in Tower B for February – In this figure it is observed and compared the real and predicted values of Tower B energy consumption. It is observed discrepancies in its performance.

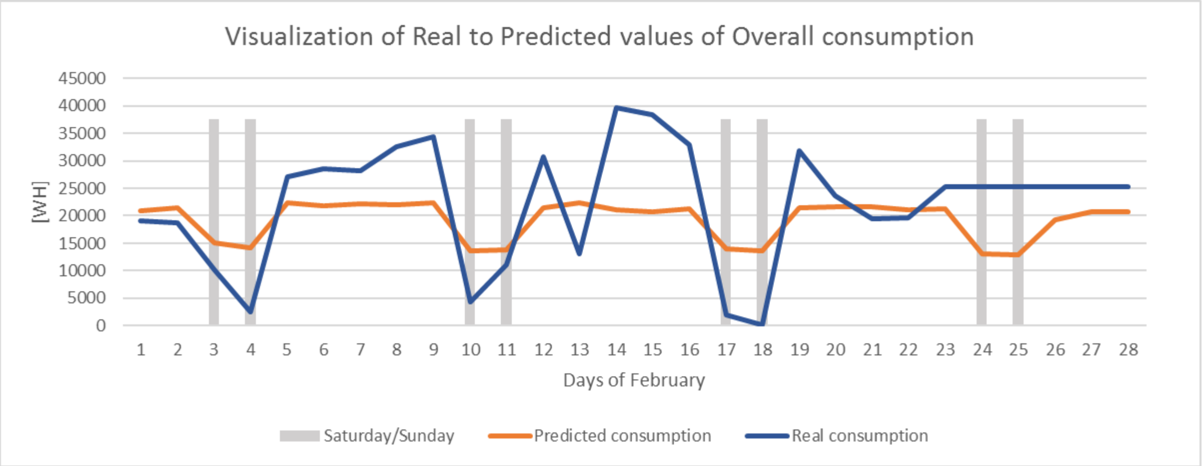

Figure 3.22- Visualization of Real to Predicted values in Overall consumption for February – In this figure it is observed and compared the real and predicted values of the overall consumption of DEEC. It is observed discrepancies in its performance, it is also possible to view similarities between both lines in the figure and data corruption in the real values.

4 Conclusion and Future work

Throughout this dissertation it has been analyzed and explored several aspects of the broad scientific study that is *Machine learning*. It has been explored what machine learning is, what tools are most commonly used today, how does a machine learn and what type of algorithm exists. The Artificial Neural Networks (ANN) was our preferred machine learning algorithm from its mathematical standpoint. With the main purpose to create code that allows us to predict how much electricity DEEC consumes in the respective towers (R, S, T and B) and overall.

Using the variables *day*, *month*, weekend (*Saturday/Sunday)*, *season*, *humidity*, *temperature* as input layers and then passing them through 5 hidden layers with 1000 neurons each previously trained with already known values predictions were achieved.

The main goal proposed at the beginning of this dissertation was successfully obtained despite some level of discrepancies in the predicted values of January and February. The detected discrepancies seem to be the result of the low amount of training data that was provided and some observed data corruption in the real values being fully expected for those discrepancies to disappear with larger amounts of training data.

So, for future work one of the things that must be made is enlarge the training datasets with a larger amount of input variables. Some of those possibly can be the *number of students in the department*, *number of classes*, *average department lux* (lx), *elevator uses* and at last but pivotal, “enlarge the already available number of input values”, since the larger the amount data available to train, the better the neural network can predict and even ignore certain errors in the collection process.

We can also enlarge the amount of output values by introducing other levels of the department (DEEC), not only the department towers and its overall consumption.

Machine learning is an area of study both fascinating and the future of all technology.

It is an area that soon will be able to achieve extraordinary things. From giving a boost to health professionals, in the detection of diseases; fully and automated machines and factories capable to do many tasks without human help; intelligent and independent exploration machines to fully autonomous artificial intelligence, as many others.

References

1. BERRIEL, Rodrigo F., LOPES, Andre Teixeira, RODRIGUES, Alexandre, VAREJAO, Flavio Miguel and OLIVEIRA-SANTOS, Thiago, 2017. Monthly energy consumption forecast: A deep learning approach. *Proceedings of the International Joint Conference on Neural Networks*. 2017. Vol. 2017-May, p. 4283–4290. DOI 10.1109/IJCNN.2017.7966398.

2. FONSECA, Susana, TURISMO, Mestre and LOCAIS, Identidades, 2008. A eficiência energética do ponto de vista dos cidadãos. *VI Congresso Português de Sociologia*. 2008.

3. ZHANG, Hong, 2011. A preliminary study on artificial neural network. *Proceedings - 2011 6th IEEE Joint International Information Technology and Artificial Intelligence Conference, ITAIC 2011*. 2011. Vol. 2, p. 336–338. DOI 10.1109/ITAIC.2011.6030344.

4. LIU, Jin, PAN, Yi, LI, Min, CHEN, Ziyue, TANG, Lu, LU, Chengqian and WANG, Jianxin, 2018. Applications of deep learning to MRI images: A survey. *Big Data Mining and Analytics*. 2018. Vol. 1, no. 1, p. 1–18. DOI 10.26599/bdma.2018.9020001.

5. CHICCO, Davide, 2017. Ten quick tips for machine learning in computational biology. *BioData Mining*. 2017. Vol. 10, no. 1, p. 1–17. DOI 10.1186/s13040-017-0155-3.

6. HUANG, T S, 1997. Computer Vision: Evolution and Promise. *Report*. 1997.

7. BOYARSHINOV, Victor, 2005. Machine Learning *Machine Learning*. *Computer*. 2005. Vol. 2005, no. April.

8. DUBOSSON, Fabien, BROMURI, Stefano and SCHUMACHER, Michael, 2016. A python framework for exhaustive machine learning algorithms and features evaluations. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*. 2016. Vol. 2016-May, p. 987–993. DOI 10.1109/AINA.2016.160. © 2016 IEEE.

9. OF, Bulletin and BIOPHYSICS, Mathematical, 1943. IDEAS IMMANENT IN NERVOUS ACTIVITY I . Introduction neuron m a y be excited by impulses a r r i v i n g at a sufficient n u m b e r of neighboring synapses within the period of latent addition , which lasts less than one q u a r t e r of a millisecond . O. . 1943. Vol. 5, p. 115–133.

10. HIRSHBERG, A. and ADAR, R., 1997. Artificial neural networks in medicine. *Israel Journal of Medical Sciences*. 1997. Vol. 33, no. 10, p. 700–702.

11. WANG, Rong Xiu and YUAN, Xiang Hui, 2005. Accuracy and track velocity of coarse grating displacement measurement system. *Bandaoti Guangdian/Semiconductor Optoelectronics* [online]. 2005. Vol. 26, no. SUPPL., p. 152–155. Available from: https://www.ieee.cz/knihovna/Zhang/Zhang100-ch03.pdf?fbclid=IwAR3-quybtazmyz5hnxqkV5j7FLX0XQjfiwcSi_k3AV2oQ2ETH5ccE6N7v5c

12. BALA, Kanchan, CHOUBEY, Dilip Kumar and PAUL, Sanchita, 2017. Soft computing and data mining techniques for thunderstorms and lightning prediction: A survey. *Proceedings of the International Conference on Electronics, Communication and Aerospace Technology, ICECA 2017*. 2017. Vol. 2017-January, p. 42–46. DOI 10.1109/ICECA.2017.8203729. © 2017 IEEE.

13. YANLIN, He, ZHIQIANG, Geng, YONGMING, Han, XU, Yuan and QUNXIONG, Zhu, 2018. A novel nonlinear virtual sample generation approach integrating extreme learning machine with noise injection for enhancing energy modeling and analysis on small data: Application to petrochemical industries. *2018 5th International Conference on Control, Decision and Information Technologies, CoDIT 2018*. 2018. P. 134–139. DOI 10.1109/CoDIT.2018.8394788.

14. THEOCHARIDES, Spyros, MAKRIDES, George, GEORGE, E and KYPRIANOU, Andreas, 2018. System Power Output Prediction. *2018 IEEE International Energy Conference (ENERGYCON)*. 2018. P. 1–6.

15. AMASYALI, Mehmet Fatih and BILGIN, Metin, 2015. Comparison of machine learning methods for the sequence labelling applications. . 2015. P. 503–506. DOI 10.1109/siu.2015.7129870.

16. GUO, Heng, 1992. Network Feature Extraction. . 1992. Vol. 3, no. 6, p. 68.

17. https://www.infoworld.com/article/3159120/facebook-brings-gpu-powered-machine-learning-to-python.html?fbclid=IwAR3UwTFWezMGWu3BVyEiHwsBeU1G86iat82vf7XkirF5urExQOeqHNkf_q8

18. https://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html?fbclid=IwAR1E4mCG1Yy3qeXK6BdFMLhktn1ZTfviYY_k7TLYX1-SGuiewflhxkwnn2A

19.

20. CHRISTIAN W. DAWSON & ROBERT WILBY (1998) An artificial neural network approach to rainfall-runoff modelling, Hydrological Sciences Journal, 43:1, 47-66, DOI:10.1080/02626669809492102

21. https://www.techopedia.com/definition/33274/weight-neural-networks (15/de setembro)

22. *Schmidhuber, Jürgen (2015-01-01). "Deep learning in neural networks: An overview". Neural Networks.* ***61****: 85–117. arXiv:1404.7828. doi:10.1016/j.neunet.2014.09.003. PMID 25462637.*

23. https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229 (15 de SET)

24. Kalogirou, Soteris. (2006). Artificial neural networks in energy applications in buildings. International Journal of Low-carbon Technologies. 1. 201-216. 10.1093/ijlct/1.3.201.

25. Faghfouri, Aram & Frish, Michael. (2011). Robust discrimination of human footsteps using seismic signals. Proc SPIE. 10.1117/12.882726.

Appendix A – “Neural Network” file

```
import torch.nn as nn
import numpy as np
import pandas as pd
from sklearn import preprocessing

"Data Processing"

dataset = pd.read_excel('datasheet.xlsx', skiprows=1)
ano = dataset['ano']
mes = dataset['mês']
dia = dataset['dia']
sabdom = dataset['sab/dom']
estacao = dataset['estacao']
temp = dataset['temp']
humidade = dataset['humidade']
torrer = dataset['torrer']
torres = dataset['torres']
torret = dataset['torret']
torreb = dataset['torreb']
consumo = dataset['consumo']
z = np.array(np.column_stack((mes, dia, sabdom, estacao, temp, humidade, torreb)))

"Processes the data into numbers between 0 and 1"

Scaler = preprocessing.MinMaxScaler()

scale = Scaler.fit_transform(z)

X_train = scale[:, [0, 1, 2, 3, 4, 5]]
Y_train = scale[:, [6]]
```

```
"Transforming data from numpy to torch format"

xtrain = torch.Tensor(X_train)
ytrain = torch.Tensor(Y_train)

"Creating the neural network with 6 input neurons, 4 hidden layer and 1 output neuron"

model = nn.Sequential(nn.Linear(6, 1000),
                      nn.Linear(1000, 1000, bias=True),
                      nn.Linear(1000, 1000, bias=True),
                      nn.Linear(1000, 1000, bias=True),
                      nn.Linear(1000, 1000, bias=True),
                      nn.Linear(1000, 1000, bias=True),
                      nn.Linear(1000, 1), nn.Sigmoid())

"Training neural network"

criterion = torch.nn.MSELoss()  #"Mean Squared Error Loss"
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)

for epoch in range(2000):
    # Forward Propagation
    y_pred = model(xtrain)
    # Compute and print loss
    loss = criterion(y_pred, ytrain)
    # Zero the gradients
    optimizer.zero_grad()
    # perform a backward pass (back propagation)
    loss.backward()
    # Update the parameters
    optimizer.step()
```

```
"Saving neural network and Scaler"

torch.save(model, "Neural network.txt")
torch.save(Scaler, "Scale.txt")
```

Appendix B – “Test” file

```
import torch
import numpy as np
import pandas as pd
from sklearn import preprocessing

"Load neural network"

model = torch.load("Neural network.txt")
Scale = torch.load("Scale.txt")

"Data Processing"

datatest = pd.read_excel('testedatasheet.xlsx', skiprows=1)
anoteste = datatest['ano']
mesteste = datatest['mês']
diateste = datatest['dia']
sabdomteste = datatest['sab/dom']
estacaoteste = datatest['estacao']
tempteste = datatest['temp']
humidadeteste = datatest['humidade']

xtest = np.array(np.column_stack((mesteste, diateste, sabdomteste,
estacaoteste, tempteste, humidadeteste, humidadeteste)))

"Processes the data into numbers between 0 and 1"

xtestscale = Scale.transform(xtest)
X_test = xtestscale[:, [0, 1, 2, 3, 4, 5]]


"Transforming data from numpy to torch format"

Xtest = torch.Tensor(X_test)
```

```
"Testing neural network”
# Forward Propagation

data = model(Xtest)

y_output = data.detach().numpy()
matrix_no_scale = np.column_stack((Xtest, y_output))
output = Scale.inverse_transform(matrix_no_scale)

"convert your array into a dataframe"
df = pd.DataFrame(output)

"save to xlsx file"
filepath = 'output.xlsx'
df.to_excel(filepath, index=False)
```

Appendix C

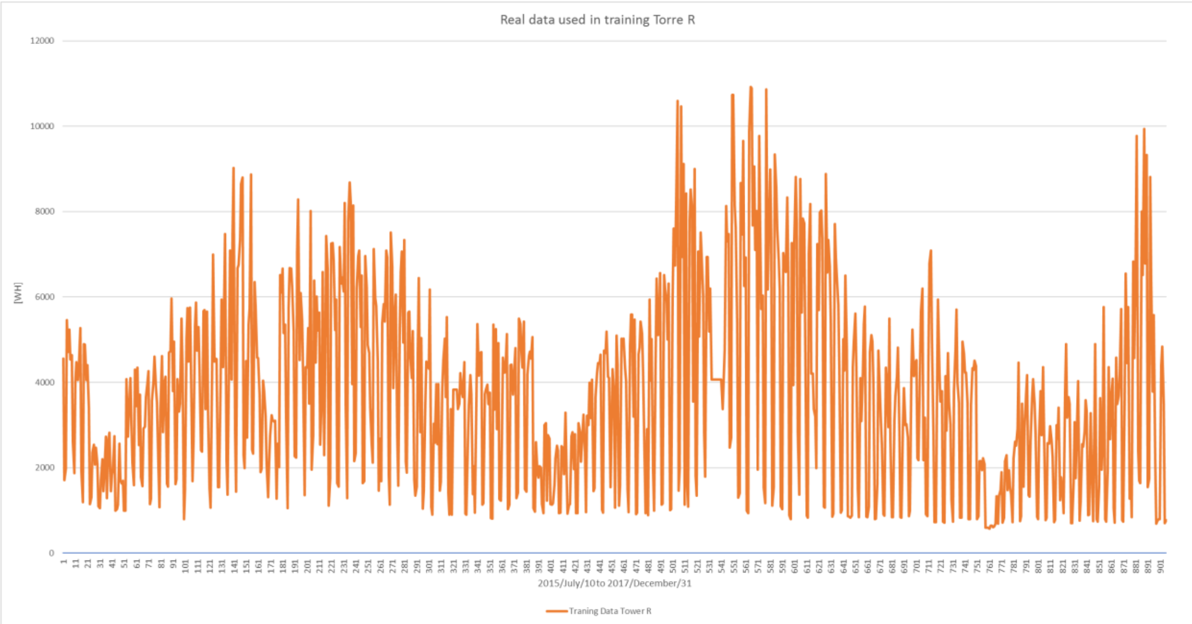

Fig. C1 Real data used in training Tower R – Real data of the energy consumption of tower R used for the training of neural network

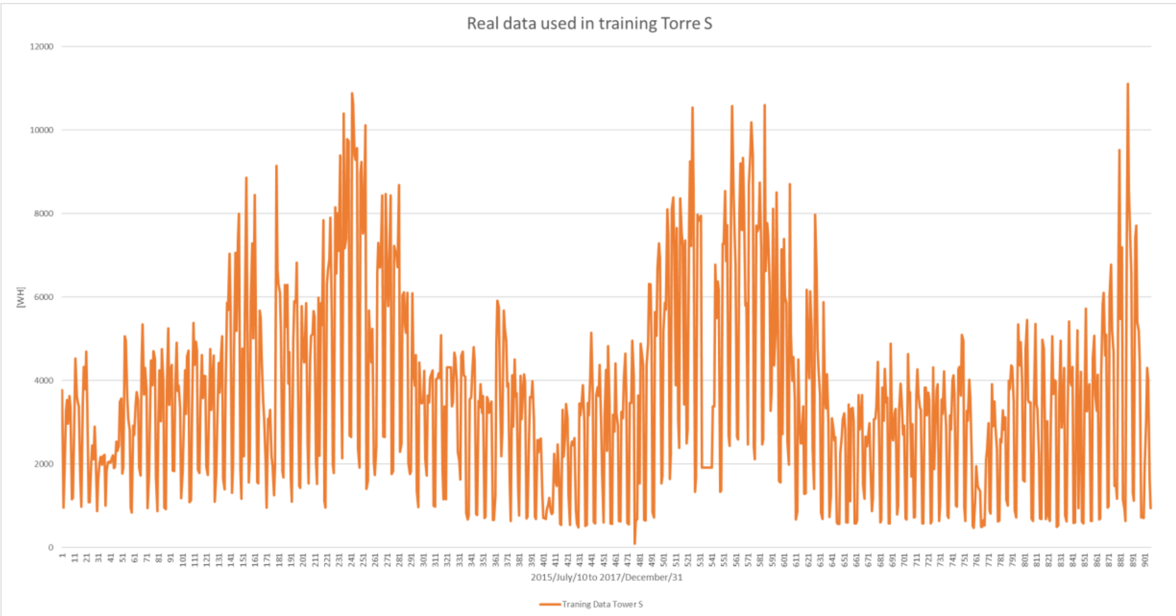

Fig. C2 Real data used in training Tower S – Real data of the energy consumption of tower S used for the training of neural network

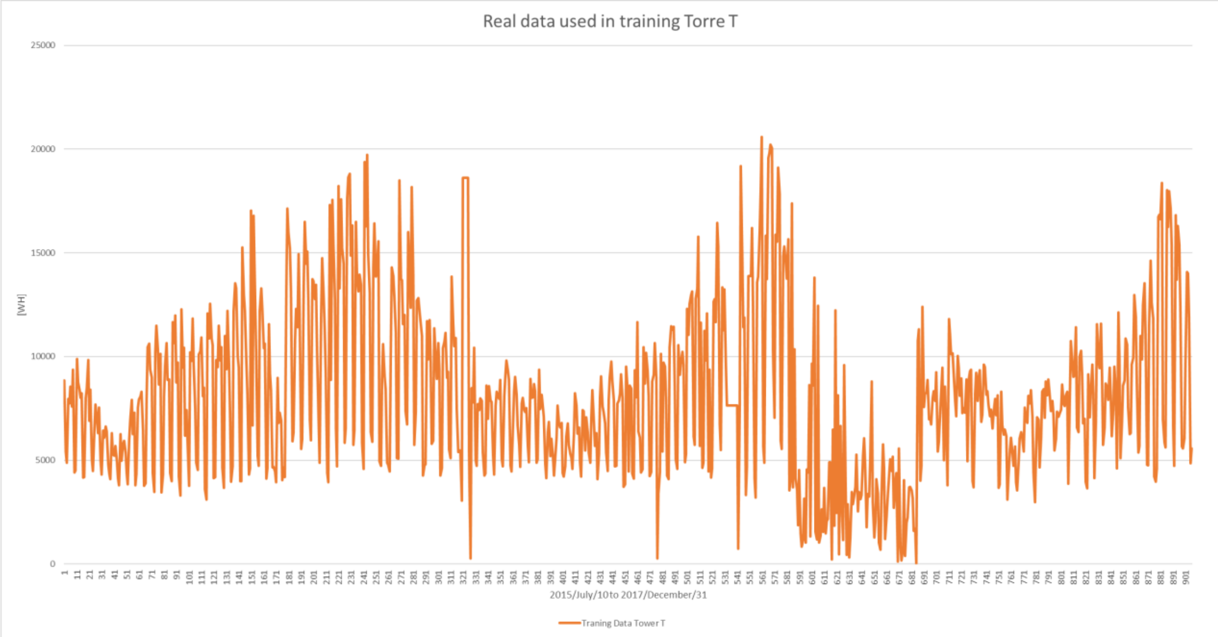

Fig C3 Real data used in training Tower T – Real data of the energy consumption of tower T used for the training of neural network

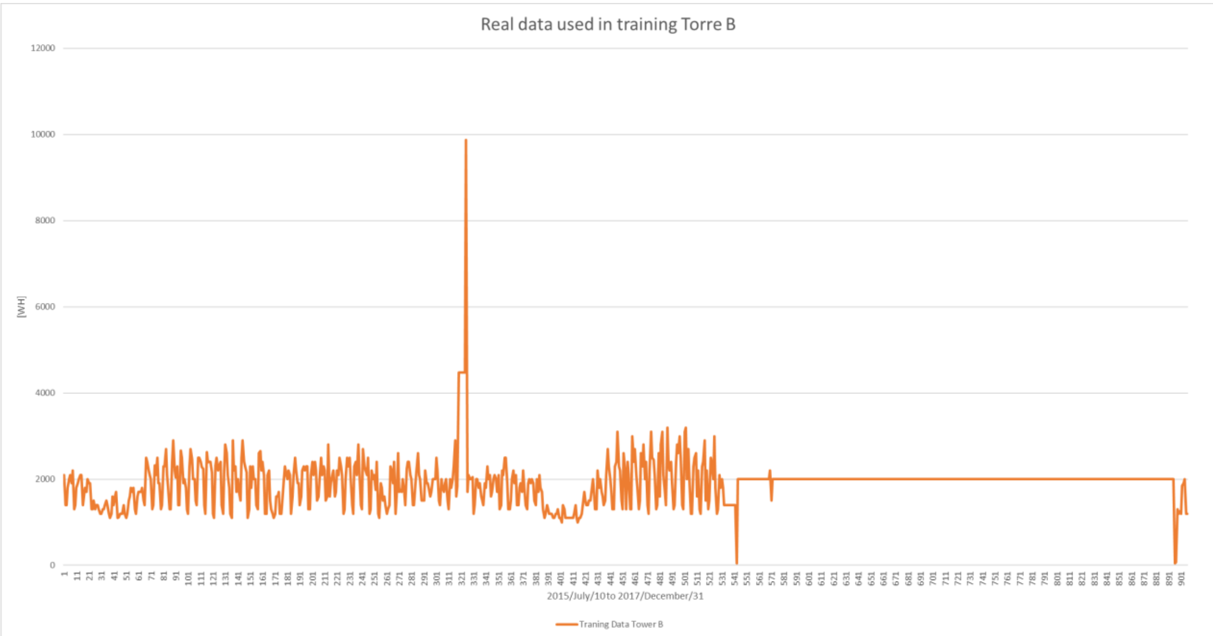

Fig C4 Real data used in training Tower B – Real data of the energy consumption of tower B used for the training of neural network

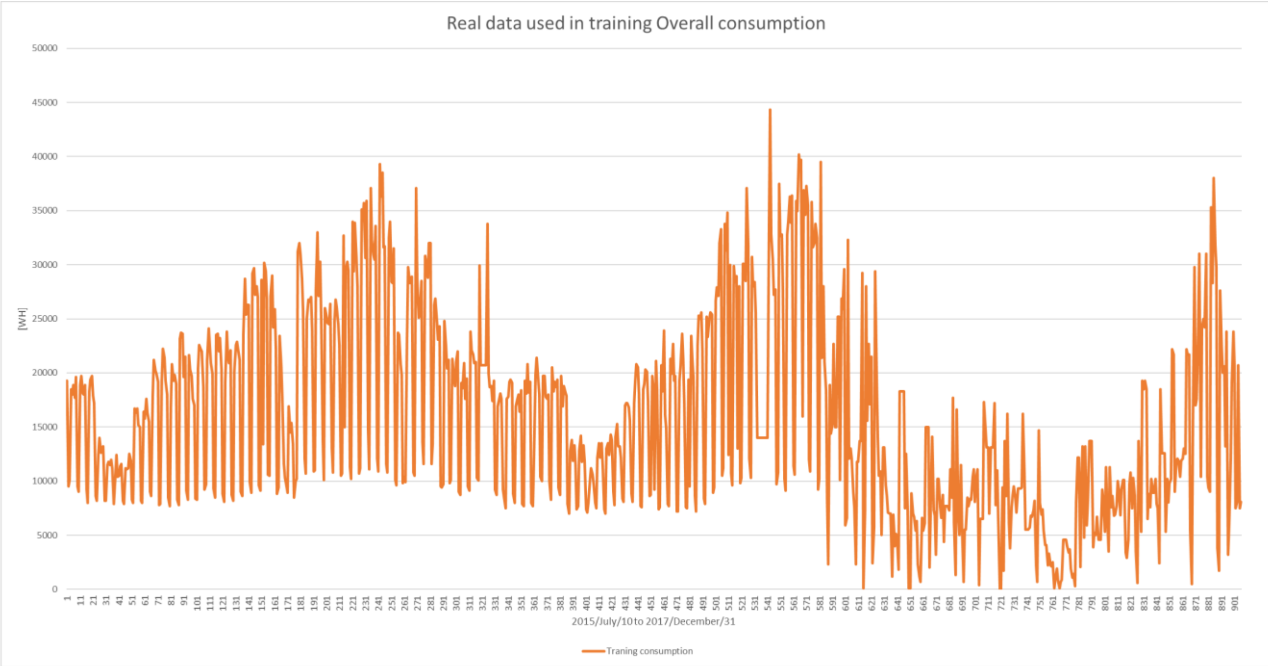

Fig C5 Real data used in training overall consumption – Real data of the energy consumption of DEEC used for the training of neural network

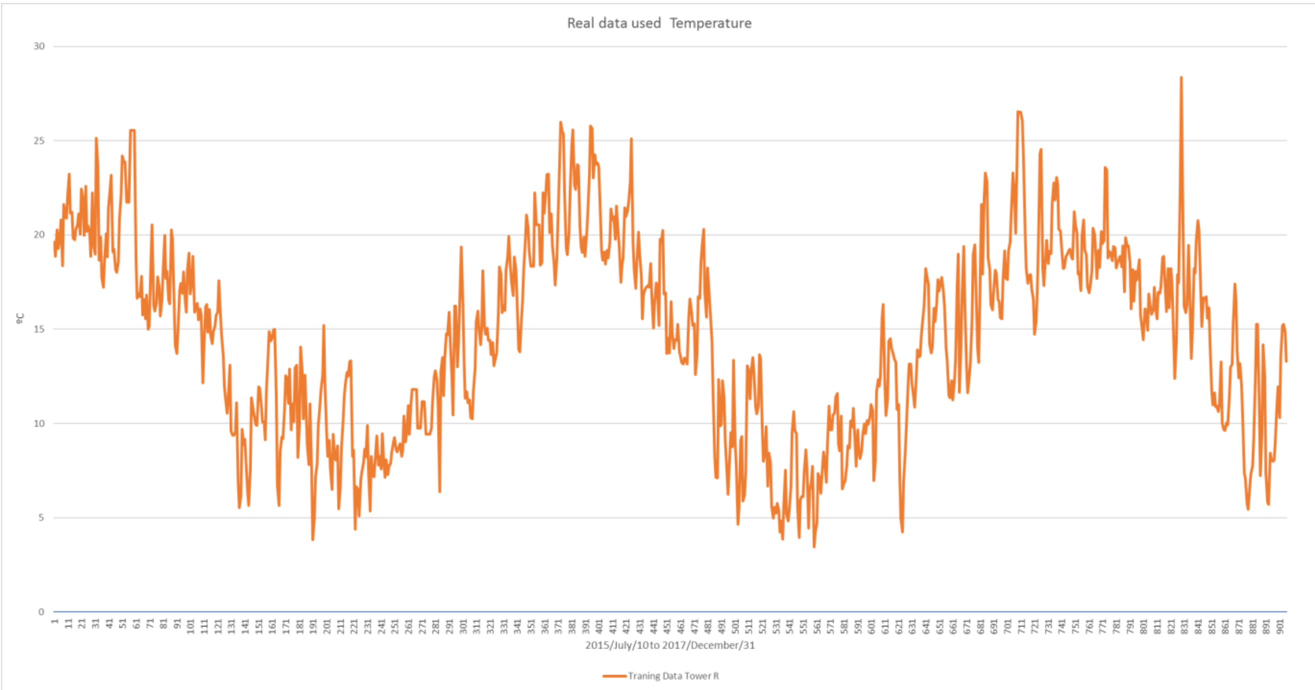

Fig C6 Real data used Temperature – Real data of temperature used in the training of the neural network

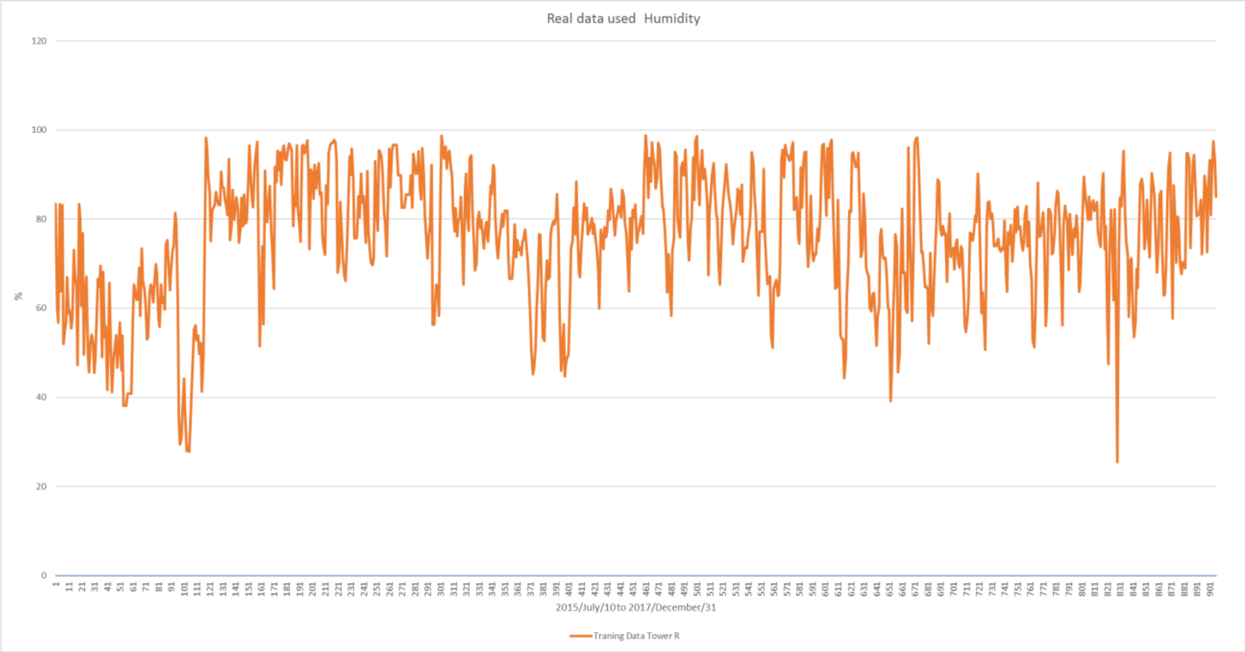

Fig C7 Real data used Humidity – Real data of humidity used in the training of the neural network

Appendix D – Example of Datasheet

Ano	Month	Day	Sat/Sun	Season	Temp	Humidity	Tower R	Tower S	Tower T	Tower B	Overall
2016	1	1	0	1	11,04167	94,94792	1270	1960	4040	1200	8500
2016	1	2	1	1	10,09375	87,70833	2560	1250	4660	1500	10000
2016	1	3	1	1	12,9375	95,55208	2010	1960	4180	2000	10200
2016	1	4	0	1	13,10417	96,5	6520	9150	13220	2300	31200
2016	1	5	0	1	8,197917	93,30208	6100	6660	17140	2100	32000
2016	1	6	0	1	9,9375	93,27083	6670	6310	16000	2000	31000
2016	1	7	0	1	14,05208	95,57292	5160	6100	15220	2200	28700
2016	1	8	0	1	12,71875	96,95833	5350	5170	13160	2100	25800
2016	1	9	1	1	10,25	96,17708	3150	1830	5910	1200	12100
2016	1	10	1	1	12,57292	95,28125	1050	1680	6330	1600	10700
2016	1	11	0	1	10,76042	78,4375	5770	6290	10820	2100	25000
2016	1	12	0	1	8,90625	85,13542	6680	5290	12310	2500	26800
2016	1	13	0	1	7,8125	82,92708	6670	6290	11420	2200	26600
2016	1	14	0	1	11,04167	96,54167	6230	3920	14930	1900	27000
2016	1	15	0	1	7,520833	82,35417	5210	4680	12090	1900	23900
2016	1	16	1	1	3,823529	78,4	2270	1670	5540	1400	10900
2016	1	17	1	1	4,96875	74,98958	2240	1100	6020	1600	11000
2016	1	18	0	1	7,177083	96,33333	7070	4950	13460	2200	27700
2016	1	19	0	1	7,885417	96,57292	8290	5900	16490	2300	33000
2016	1	20	0	1	9,875	94,6875	4520	5870	14460	2200	27100
2016	1	21	0	1	10,95833	96,04167	6090	6830	15060	2300	30300
2016	1	22	0	1	11,79167	97,69792	5450	4520	12710	2300	25000
2016	1	23	1	1	12,5625	90,96875	4080	1470	6930	1300	13800
2016	1	24	1	1	15,19792	73,23958	1360	1430	5970	1300	10100
2016	1	25	0	1	12,19792	91,125	4360	5780	13740	2100	26000
2016	1	26	0	1	9,635417	90,32292	4350	4790	13640	2400	25200
2016	1	27	0	1	8,260417	84,60417	5270	4430	12770	2100	24600
2016	1	28	0	1	9,104167	92,19792	3500	5120	13460	2400	24500
2016	1	29	0	1	7,572917	86,83333	8020	5850	10310	2200	26400
2016	1	30	1	1	6,489583	90,14583	1950	2470	7050	1500	13000
2016	1	31	1	1	9,416667	92,47917	2750	1540	4870	1600	10800
2016	2	1	0	1	8,208333	85,54167	6380	4480	11920	2000	24800
2016	2	2	0	1	8,041667	86	4460	5070	14750	2500	26800
2016	2	3	0	1	8,8125	76,26042	6010	5110	12860	2100	26100
2016	2	4	0	1	5,479167	73,70833	5210	5670	11320	2300	24500
2016	2	5	0	1	6,635417	72	5640	5520	9210	2200	22600
2016	2	6	1	1	8,697917	87,61458	2540	2060	4380	1500	10500
2016	2	7	1	1	10,13542	83,32292	3600	1520	3940	1600	10700
2016	2	8	0	1	11,4375	94,82292	6590	5980	17320	2800	32700
2016	2	9	0	1	12,0625	96,61458	2300	2200	8880	1600	15000

2016	2	10	0	1	12,71875	97,05208	4290	5850	17560	1900	29600
2016	2	11	0	1	12,51579	97,04211	7430	5330	15380	2100	30300
2016	2	12	0	1	13,29167	97,78125	6660	7840	12780	2200	29500
2016	2	13	1	1	13,32292	97,03125	1115	1120	6510	1600	11300
2016	2	14	1	1	8,270833	92,9375	1750	955	4700	1700	10200
2016	2	15	0	1	8,59375	67,96875	7250	6290	18230	2200	34000
2016	2	16	0	1	4,395833	70,10417	7260	6640	13280	2100	29400
2016	2	17	0	1	6,635135	83,89189	6850	6910	17570	2500	33900
2016	2	18	0	1	6,458333	76,8125	5230	7900	15240	2300	30700
2016	2	19	0	1	5,104651	71,0814	5940	5500	14460	2100	28000
2016	2	20	1	1	7,026316	67,15789	1640	1990	5850	1200	10700
2016	2	21	1	1	7,4375	66,11458	1560	1780	6520	1300	11200
2016	2	22	0	1	7,9375	74,15625	7170	8150	17660	2000	35100
2016	2	23	0	1	8,625	81,76042	6310	6570	18630	2500	34100
2016	2	24	0	1	8,229167	94,04167	6460	8010	18810	2300	35700
2016	2	25	0	1	9,905263	89,94737	6130	7110	14860	2500	30600
2016	2	26	0	1	7,5	95,82292	8200	9390	16310	2000	35900
2016	2	27	1	1	5,354167	88,95833	3450	4090	5740	1400	14700
2016	2	28	1	1	8,260417	75,76042	1290	2140	6430	1200	11100
2016	2	29	0	1	7,239583	75,69792	7790	10400	16500	2300	37100
2016	3	1	0	1	7,1875	75,82292	8680	7170	13950	2400	32300
2016	3	2	0	1	8,5	85,14583	8220	7410	13140	2100	30900
2016	3	3	0	1	9,34375	80,44792	3960	9790	13950	2800	30500
2016	3	4	0	1	7,78125	90,13542	8140	9740	13460	2200	33600
2016	3	5	1	1	8,25	83,78125	2160	2680	5830	1400	12100
2016	3	6	1	1	7,59375	85,09375	2340	2650	4570	1300	10900
2016	3	7	0	1	9,452632	79,37895	6210	10890	19390	2700	39300
2016	3	8	0	1	7,810526	74,73684	6960	10650	16280	2300	36300
2016	3	9	0	1	7,145833	90,70833	7090	9410	19720	2200	38500
2016	3	10	0	1	8,094737	77,54737	5290	9290	14920	2100	31600
2016	3	11	0	1	7,290323	72,41935	6500	9570	13100	2500	31700
2016	3	12	1	1	7,71875	70,22917	1640	2370	6360	1200	11600
2016	3	13	1	1	7,864583	69,67708	1690	1910	5880	1300	10800
2016	3	14	0	1	8,452632	71,24211	6960	8970	14340	2200	32500
2016	3	15	0	1	8,989583	93,04167	6220	9240	16430	2000	34000
2016	3	16	0	1	9,25	82,96875	4870	7530	13860	2100	28400
2016	3	17	0	1	8,604167	77,41667	4690	7760	14100	1750	28300
2016	3	18	0	1	8,479167	95,39583	3420	10120	15560	2400	31500
2016	3	19	1	1	8,757895	95,13684	2540	1400	4960	1200	10100
2016	3	20	1	1	8,923913	93,76087	2120	1610	4730	1100	9600
2016	3	21	0	2	8,252632	90,42105	7120	5680	8980	1900	23700
2016	3	22	0	2	8,578947	81,45263	5980	5100	10600	1800	23500
2016	3	23	0	2	10,38542	79,27083	5790	4430	9060	1500	20800
2016	3	24	0	2	9,020833	71,66667	4510	5240	8430	1600	19800
2016	3	25	0	2	9,46875	82,17708	1460	2050	4870	1400	9800
2016	3	26	1	2	10,96842	95,83158	2680	1730	4770	1200	10400

2016	3	27	1	2	9,434783	87,13043	1690	2410	4460	1300	9900
2016	3	28	0	2	11,10526	95,06316	5400	6580	9910	1400	23300
2016	3	29	0	2	11,79487	96,61538	5830	7300	14300	2300	29800
2016	3	30	0	2	11,79487	96,61538	5430	6710	13880	2200	28300
2016	3	31	0	2	11,79487	96,61538	7090	6890	12920	1900	28800
2016	4	1	0	2	11,79487	96,61538	6920	8440	11140	2400	28900
2016	4	2	1	2	9,760417	89,8125	1830	2660	5090	1200	10800
2016	4	3	1	2	9,760417	89,8125	1130	2650	5080	1600	10500
2016	4	4	0	2	9,760417	89,8125	7510	8470	18500	2600	37100
2016	4	5	0	2	11,14583	82,54167	6710	6420	13740	1700	28700
2016	4	6	0	2	11,14583	82,54167	3860	5780	13680	1750	25100
2016	4	7	0	2	11,14583	82,54167	5600	6640	11560	1700	25500
2016	4	8	0	2	9,428571	85,42857	6060	8440	12000	2000	28500
2016	4	9	1	2	9,428571	85,42857	2760	1760	7360	1700	13600
2016	4	10	1	2	9,428571	85,42857	1580	1830	6740	1400	11600
2016	4	11	0	2	9,428571	85,42857	5300	7230	16000	2250	30800
2016	4	12	0	2	9,760417	89,8125	6590	7050	13980	2400	30100
2016	4	13	0	2	11,14583	82,54167	7060	6990	12340	2400	28800
2016	4	14	0	2	12,5	94,65625	4930	6720	18180	2100	32000
2016	4	15	0	2	12,8125	92,13542	7330	8690	13980	2000	32000
2016	4	16	1	2	12,3125	90,15625	2140	2290	5750	1400	11600
2016	4	17	1	2	10,94792	90,40625	1890	2490	7310	1400	13400
2016	4	18	0	2	6,375	95,25	5630	6050	12700	1900	26300
2016	4	19	0	2	12,73529	86,47059	5660	6120	12830	2200	26900
2016	4	20	0	2	13,51807	84,46988	4600	5330	12260	2600	24800
2016	4	21	0	2	11,48958	95,88542	4100	5140	11730	2100	23100
2016	4	22	0	2	13,4375	90,07292	5200	6100	11000	2000	24300
2016	4	23	1	2	14,77083	80,91667	1710	2020	4250	1500	9500
2016	4	24	1	2	14,71875	75,40625	1350	1760	4750	1500	9400
2016	4	25	0	2	15,89583	71,1875	1560	1820	4800	1500	9700
2016	4	26	0	2	14,40625	76,71875	4710	6090	11720	2200	24800
2016	4	27	0	2	12,38889	78,65556	6440	4290	9850	1900	22500
2016	4	28	0	2	10,44444	92,22222	2840	3880	11750	1900	20400
2016	4	29	0	2	16,22581	56,33871	5040	4610	9750	1800	21200
2016	4	30	1	2	16,22581	56,33871	1040	1360	5780	1600	9800
2016	5	1	1	2	13,01053	61,58947	1460	970	5930	1700	10100
2016	5	2	0	2	14,22917	65,30208	3480	4430	11370	2000	21300
2016	5	3	0	2	17,29167	64	4490	3490	9890	2000	19900
2016	5	4	0	2	19,36458	58,27083	4430	3450	8920	2000	18800
2016	5	5	0	2	16,33333	85,91667	4320	3540	10640	2500	21000
2016	5	6	0	2	13,36458	98,78125	6170	4230	9400	2200	22000
2016	5	7	1	2	11,34375	95,71875	1080	2130	4270	1500	9000
2016	5	8	1	2	11,69792	93,55208	900	1720	4640	1400	8700
2016	5	9	0	2	11,09375	96,22917	3030	3640	10410	2000	19100
2016	5	10	0	2	11,25	91,38542	2590	3470	10720	1800	18600
2016	5	11	0	2	10,3125	94,66667	3960	4070	11150	1700	20900

2016	5	12	0	2	10,23958	95,29167	2550	4170	8960	1900	17600
2016	5	13	0	2	11,73958	91,375	3960	4240	9280	2000	19500
2016	5	14	1	2	12,90625	89,79167	1460	1000	5520	1500	9500
2016	5	15	1	2	15,37895	82,24211	1670	980	5100	1300	9100
2016	5	16	0	2	15,96875	77,30208	3890	4030	13860	2000	23800
2016	5	17	0	2	15,04167	82,44792	4510	4060	11640	1800	22100
2016	5	18	0	2	14,16667	76,05208	5030	4180	10490	2000	21700
2016	5	19	0	2	15	79,79167	3650	4050	10890	2300	20900
2016	5	20	0	2	18,09375	80,19792	5530	5090	7480	2900	21000
2016	5	21	1	2	15,20833	85,03125	1410	1870	5400	1600	10300
2016	5	22	1	2	14,73958	74,41667	900	1156,667	5470	1833,333	10.033
2016	5	23	0	2	15,05208	65,30208	3.378	3378,041	5470	4469,072	29.917
2016	5	24	0	2	14,40625	86,44792	900	1156,667	3060	4469,072	20.691
2016	5	25	0	2	14,41667	90,17708	3.825	4310,909	18611,13	4469,072	20.691
2016	5	26	0	2	13,64583	80,86458	3.825	4310,909	18611,13	4469,072	20.691
2016	5	27	0	2	14,30526	77,41053	3.825	4310,909	18611,13	4469,072	20.691
2016	5	28	1	2	13,07292	93,73958	3.825	4310,909	18611,13	4469,072	20.691
2016	5	29	1	2	13,47917	94,27083	3.378	3378,041	18611,13	9870,103	33.780
2016	5	30	0	2	13,72917	82,58333	3.497	3476,667	5392,783	1700	19900
2016	5	31	0	2	15,73958	74,4375	4.221	4675,357	264,9138	2089,286	18700
2016	6	1	0	2	18,3125	68,48958	3770	4530	8490	2000	18800
2016	6	2	0	2	17,91667	70,16667	3650	3940	7800	2000	17400
2016	6	3	0	2	15,88542	79,70833	4480	2290	10420	2050	19300
2016	6	4	1	2	16,23958	81,59375	920	2070	4990	1200	9200
2016	6	5	1	2	16	78,02083	900	1630	4740	1400	8700
2016	6	6	0	2	18,13542	79,82292	2560	4590	7710	2000	16900
2016	6	7	0	2	18,82292	76,1875	3470	4690	7420	2000	17600
2016	6	8	0	2	19,91304	73,36957	4170	4130	7980	1800	18100
2016	6	9	0	2	18,24468	78,05319	3730	4090	7860	1900	17600
2016	6	10	0	2	17,48958	79,41667	1380	820	5380	1600	9200
2016	6	11	1	2	16,77174	74,98913	2040	670	4270	1500	8500
2016	6	12	1	2	18,83333	79,05208	950	760	4360	1400	7500
2016	6	13	0	2	18,16667	87,42708	3540	3550	8590	1900	17600
2016	6	14	0	2	16,53684	85,57895	5370	3600	7010	1800	17800
2016	6	15	0	2	13,94681	92,17021	4155	4155	8570	2300	19200
2016	6	16	0	2	13,80208	91,05208	4690	4800	7890	2000	19400
2016	6	17	0	2	15,56989	80,29032	4710	4460	7820	2100	19100
2016	6	18	1	2	16,51087	73,67391	1130	800	5250	1600	8800
2016	6	19	1	2	18,12632	71,16842	1170	780	4310	1700	8000
2016	6	20	0	2	19,71579	75,92632	3730	3510	7640	2000	16900
2016	6	21	0	2	21,07609	77,5	3830	3360	8300	2100	17600
2016	6	22	0	3	20,37634	81,13978	3950	3910	8120	2000	18000
2016	6	23	0	3	19,11702	78,29787	3490	3230	7960	1700	16400
2016	6	24	0	3	18,32967	81,84615	3760	3625	8875	2100	18400
2016	6	25	1	3	18,32967	81,84615	820	710	5060	1300	7900
2016	6	26	1	3	18,32967	81,84615	810	740	4710	1400	7700

2016	6	27	0	3	22,25	71,41667	5360	3610	8110	2200	19300
2016	6	28	0	3	20,54762	66,57143	3360	3540	9080	2100	18100
2016	6	29	0	3	20,54762	66,57143	5250	3230	9800	2500	20800
2016	6	30	0	3	20,54762	66,57143	2910	3430	9320	2500	18200
2016	7	1	0	3	18,40625	72,03125	4920	3500	8880	1900	19200
2016	7	2	1	3	18,54839	78,84946	1260	660	4760	1300	8000
2016	7	3	1	3	22,25	71,41667	1230	660	4470	1300	7700
2016	7	4	0	3	21,15625	75,23958	2660	1240	6690	1500	12100
2016	7	5	0	3	22,23958	73,04167	4580	4880	8300	2000	19800
2016	7	6	0	3	23,1978	73,62637	4225	5910	9015	2200	21400
2016	7	7	0	3	23,23958	71,85417	4240	5750	8160	1900	20100
2016	7	8	0	3	20,13542	74,90625	5130	4040	6680	2100	18000
2016	7	9	1	3	21,125	75,52083	1030	2190	5730	1400	10400
2016	7	10	1	3	19,46237	77,60215	1140	2750	4670	1400	10000
2016	7	11	0	3	18,48958	74,82292	4410	5680	7740	1800	19700
2016	7	12	0	3	17,35417	70,42708	4420	5260	8010	1900	19700
2016	7	13	0	3	18,71875	66,10417	3710	4940	7410	1700	17800
2016	7	14	0	3	20,67708	57,6875	4160	3860	7330	2200	17600
2016	7	15	0	3	23,72917	50,5	4800	3940	7520	1700	18000
2016	7	16	1	3	26	45,19792	1290	1590	5680	1400	10000
2016	7	17	1	3	25,45833	46,60417	1420	640	4900	1300	8300
2016	7	18	0	3	25,37234	51,01064	5490	4130	8930	1900	20500
2016	7	19	0	3	22,25532	59,74468	5370	2890	8000	2000	18300
2016	7	20	0	3	19,34375	64,42708	4700	4510	8040	1900	19200
2016	7	21	0	3	18,95833	76,63542	4350	3620	8680	2000	18700
2016	7	22	0	3	20,0625	76,4375	5430	3700	8320	1900	19400
2016	7	23	1	3	22,20833	63,52083	1500	1480	4870	1500	9400
2016	7	24	1	3	24,70833	53,5625	1440	770	5050	1400	8700
2016	7	25	0	3	25,5625	52,65625	4190	4120	9360	2000	19700
2016	7	26	0	3	22,64583	63,90625	4620	3070	7470	1700	16900
2016	7	27	0	3	22,41667	70,6875	4720	3770	8170	2100	18800
2016	7	28	0	3	23,73958	66,55208	4560	4150	7650	1800	18200
2016	7	29	0	3	23,66667	66,97917	5060	3960	7130	1700	17900
2016	7	30	1	3	20,38947	76,54737	1050	690	4820	1300	7900
2016	7	31	1	3	19,35417	78,21875	970	740	4150	1100	7000
2016	8	1	0	3	19,0625	79,52083	2600	2590	6370	1200	12800
2016	8	2	0	3	19,89583	78,77083	1890	3610	6850	1400	13800
2016	8	3	0	3	18,875	80,32292	1770	3600	5190	1300	11900
2016	8	4	0	3	19,97895	85,55789	2040	3980	6040	1200	13300
2016	8	5	0	3	21,12632	76,21053	2000	3050	5210	1200	11500
2016	8	6	1	3	23,16667	59,48958	1150	740	4260	1200	7400
2016	8	7	1	3	25,7766	46,04255	940	680	4940	1100	7700
2016	8	8	0	3	25,6413	49,3913	3000	2580	6080	1100	12800
2016	8	9	0	3	23,02105	56,47368	3040	2280	7630	1200	14200
2016	8	10	0	3	24,24468	44,64894	1230	2590	6730	1200	11800
2016	8	11	0	3	23,76042	47,5	2760	2610	6590	1300	13300

2016	8	12	0	3	23,82292	49,1875	2680	1280	6790	1100	11900
2016	8	13	1	3	23,58696	49,52174	1150	710	4390	1100	7400
2016	8	14	1	3	21,04348	62,72826	1140	710	4210	1000	7100
2016	8	15	0	3	19,19355	73,50538	1170	680	4700	1400	8000
2016	8	16	0	3	18,66667	75,13542	1600	910	6140	1300	10000
2016	8	17	0	3	19,09677	82,56989	2230	1040	6780	1100	11200
2016	8	18	0	3	18,44211	76,54737	2520	1190	5740	1100	10600
2016	8	19	0	3	19,1828	88,45161	2400	870	5290	1100	9700
2016	8	20	1	3	18,77895	75,92632	940	800	5120	1100	8000
2016	8	21	1	3	19,59375	68,57292	950	840	4570	1100	7500
2016	8	22	0	3	21,375	66,9375	2510	2250	6290	1100	12200
2016	8	23	0	3	20,80208	74,60417	2460	1660	8240	1100	13500
2016	8	24	0	3	20,96875	77,83333	1850	1470	7630	1200	12200
2016	8	25	0	3	19,78125	83,45833	3290	2470	6290	1400	13500
2016	8	26	0	3	21,53125	79,84375	2050	1620	6590	1100	11400
2016	8	27	1	3	20,16667	82,57292	920	560	4880	1000	7400
2016	8	28	1	3	19,40625	76,54167	1120	540	4200	1100	7000
2016	8	29	0	3	17,47917	77,5625	1150	3300	7410	1100	13000
2016	8	30	0	3	18,40625	78,6875	2740	2170	7360	1200	13500
2016	8	31	0	3	18,79167	80,22917	2830	2650	5470	1400	12400
2016	9	1	0	3	21,4382	76,67416	1970	3440	7080	1700	14300
2016	9	2	0	3	20,98958	78,90625	2790	3090	6330	1500	13800
2016	9	3	1	3	21,26042	75,45833	930	1260	5350	1400	9000
2016	9	4	1	3	21,75	73,59375	940	540	4870	1400	7800
2016	9	5	0	3	22,78125	68,59375	3050	2410	7080	1500	14100
2016	9	6	0	3	25,11702	59,90426	2830	2540	8370	1500	15300
2016	9	7	0	3	19,64211	77,67368	2140	2450	6850	1700	13200
2016	9	8	0	3	18,1875	76,16667	2840	2620	5680	2000	13200
2016	9	9	0	3	17,16667	73,28125	3240	890	6320	1700	12200
2016	9	10	1	3	18,625	78,26042	2360	570	4100	1300	8400
2016	9	11	1	3	20,14583	75,97917	960	480	5300	1300	8100
2016	9	12	0	3	19,01042	77,1875	3220	3450	8030	2200	17000
2016	9	13	0	3	17,85417	81,91667	3000	3170	9040	1800	17200
2016	9	14	0	3	15,56842	79,88421	4000	3520	7600	2000	17200
2016	9	15	0	3	16,85106	86,7234	3160	3890	7320	1700	16800
2016	9	16	0	3	17,125	84,60417	4070	3020	6780	1600	15600
2016	9	17	1	3	17,21053	80,70526	1450	520	5370	1400	8800
2016	9	18	1	3	17,31579	76,34737	1520	540	4490	1500	8100
2016	9	19	0	3	17,21505	79,77419	3660	3460	7910	2100	17200
2016	9	20	0	3	18,48315	81,24719	4220	3170	9020	2700	19300
2016	9	21	0	3	16,76543	82,85185	4450	4100	9750	2300	20800
2016	9	22	0	3	15,06977	81,7093	4440	5150	8830	2000	20500
2016	9	23	0	3	16,37778	80,46667	4650	3740	7890	1700	18100
2016	9	24	1	4	17,46237	86,52688	1020	610	4710	1300	7700
2016	9	25	1	4	17,32222	84,95556	950	580	4720	1300	7600
2016	9	26	0	4	15,21591	79,23864	4750	3640	7750	2300	18500

2016	9	27	0	4	19,78652	76,53933	4690	3840	7880	2400	19000
2016	9	28	0	4	19,60714	71,34524	5190	3630	8210	3100	20300
2016	9	29	0	4	20,25275	63,79121	4140	4380	9140	2300	20100
2016	9	30	0	4	16,86667	80,25556	4100	2970	7990	2200	17400
2016	10	1	1	4	16,92941	73,18824	1540	1780	3720	1500	8600
2016	10	2	1	4	13,71084	82,06024	2900	600	3850	1300	8700
2016	10	3	0	4	14,54118	80,27059	4310	3140	9520	2600	19700
2016	10	4	0	4	13,73333	83,3	3720	4260	8300	2200	18600
2016	10	5	0	4	16,45833	74,66667	1070	2320	4450	1300	9200
2016	10	6	0	4	14,82292	76,84375	5100	4830	8590	2400	21100
2016	10	7	0	4	13,95833	77,51042	4420	3430	8450	2100	18500
2016	10	8	1	4	14,42708	79,45833	1110	570	4370	1300	7400
2016	10	9	1	4	14,41667	80,71875	1580	560	4110	1300	7600
2016	10	10	0	4	15,26042	76,64583	5030	3180	9330	3000	20700
2016	10	11	0	4	13,79167	89,69792	4890	2780	8030	2400	18300
2016	10	12	0	4	13,5625	98,70833	5030	4410	11660	2700	23900
2016	10	13	0	4	13,23958	95,42708	4330	3220	6390	2100	16100
2016	10	14	0	4	13,15625	84,83333	4030	2920	6090	1800	14900
2016	10	15	1	4	13,47917	93,71875	1490	630	4420	1300	7900
2016	10	16	1	4	13,34375	88,42708	960	620	4560	1500	7700
2016	10	17	0	4	13,15625	97,21875	4870	3250	10440	2600	21300
2016	10	18	0	4	15,66667	94,22917	5590	3100	8700	2300	19900
2016	10	19	0	4	16,60417	91,97917	5590	4030	10170	2800	22700
2016	10	20	0	4	15,89583	86,89583	3200	4650	9310	2000	19300
2016	10	21	0	4	15,20833	89,26042	5470	3630	8100	2400	19700
2016	10	22	1	4	15,28125	97,13542	910	590	4240	1400	7200
2016	10	23	1	4	12,60417	95,55208	950	550	4440	1200	7200
2016	10	24	0	4	13,6875	89,48958	4660	3460	8810	2200	19300
2016	10	25	0	4	16,71875	82,90625	4820	3460	9110	3100	20600
2016	10	26	0	4	16,625	82,03125	5420	4960	10660	2500	23600
2016	10	27	0	4	18,54167	76,79167	5140	4100	9000	2450	21300
2016	10	28	0	4	19,37895	72,10526	4240	90	261,0457	2094,643	17.207
2016	10	29	1	4	20,30208	63,625	2220	630	3390	1300	7600
2016	10	30	1	4	16,93	72,13	930	680	4430	1400	7500
2016	10	31	0	4	15,65625	64,70833	2910	3640	10140	2600	19400
2016	11	1	0	4	18,26042	58,27083	890	1530	5420	1600	9500
2016	11	2	0	4	16,82796	72,84946	5940	4890	9700	2800	23400
2016	11	3	0	4	15,83333	75,69792	4040	4590	9520	3100	21300
2016	11	4	0	4	14,35417	95,07292	5010	4350	7930	2100	19600
2016	11	5	1	4	11,53125	94,02083	1920	660	4260	1700	8600
2016	11	6	1	4	8,1875	80,08333	990	650	4100	1400	7200
2016	11	7	0	4	7,145833	77,02083	4460	4380	10450	3200	22600
2016	11	8	0	4	7,114583	75,86458	6430	4890	11460	2400	25300
2016	11	9	0	4	12,32292	91,63542	5110	6320	11280	2200	25000
2016	11	10	0	4	9,875	92,73958	5330	6310	11430	2400	25600
2016	11	11	0	4	9,947368	89,82105	6560	4970	8610	1900	22100

2016	11	12	1	4	12,26042	95,59375	1130	850	5060	1300	8400
2016	11	13	1	4	11,36458	89,125	1160	720	4570	1400	7900
2016	11	14	0	4	9,09375	74,77083	6510	5640	10540	2400	25200
2016	11	15	0	4	7,458333	70,66667	6190	5070	9110	2800	23300
2016	11	16	0	4	6,238095	78,28571	5890	6670	9740	2600	25000
2016	11	17	0	4	8,15625	80,05208	5000	7290	10220	3000	25600
2016	11	18	0	4	9,53125	93,77083	6310	6930	9610	2500	25400
2016	11	19	1	4	8,75	84,3125	1010	1530	4900	1400	8900
2016	11	20	1	4	13,36458	97,30208	1030	1720	5300	1300	9400
2016	11	21	0	4	9	98,64516	5990	5190	12310	3100	26700
2016	11	22	0	4	7,75	91,05208	7600	5870	11040	3200	27900
2016	11	23	0	4	4,65625	83,08333	6740	5700	12560	2000	27100
2016	11	24	0	4	5,546512	91,01163	8130	8100	12970	2700	32000
2016	11	25	0	4	9,09375	95,4375	10590	7500	13150	2000	33300
2016	11	26	1	4	9,302083	89,01042	1460	1640	6140	1200	10500
2016	11	27	1	4	5,895833	91,19792	1960	2240	5740	1200	11200
2016	11	28	0	4	6,21875	89	10460	8110	12800	2300	33800
2016	11	29	0	4	7,364583	82,97917	6940	8390	13260	2500	31300
2016	11	30	0	4	13,05208	67,48958	9120	7200	15790	2600	34800
2016	12	1	0	4	12,89583	81,03125	1140	3890	5700	1600	12400
2016	12	2	0	4	11,3125	86,16667	8420	7660	11630	2200	30000
2016	12	3	1	4	12,73958	89,90625	1630	3060	4640	1300	10700
2016	12	4	1	4	13,48958	92,5	1090	2390	4870	1200	9600
2016	12	5	0	4	12,67708	87,17708	7810	8370	11250	2300	29900
2016	12	6	0	4	10,85417	81,29167	8520	7540	9800	2400	28400
2016	12	7	0	4	10,52632	80,14737	8060	7220	12090	2900	28950
2016	12	8	0	4	10,9375	68,94792	3550	3430	4470	1500	13000
2016	12	9	0	4	13,65625	65,30208	9000	7360	9370	2200	28000
2016	12	10	1	4	13,44792	75,29167	1800	2490	4160	1300	9800
2016	12	11	1	4	9,84375	82,16667	1340	2820	4600	1500	10300
2016	12	12	0	4	8	85,11458	7060	7730	12650	2500	30100
2016	12	13	0	4	8,5	89	5070	9250	12780	2350	30100
2016	12	14	0	4	9,84375	92,30208	7510	7230	11670	2000	28500
2016	12	15	0	4	6,6875	87,85417	6950	10540	16450	3000	37100
2016	12	16	0	4	8,4375	85,125	6050	8620	15300	1800	31800
2016	12	17	1	4	7,90625	83,05208	2810	1330	6720	1200	12100
2016	12	18	1	4	5,614583	80,59375	1790	1680	5490	1300	10300
2016	12	19	0	4	4,977778	74,35556	6940	7980	13330	2100	30700
2016	12	20	0	4	5,555556	77,59259	6940	7890	11260	1800	28000
2016	12	21	0	4	5,253521	80,5493	5190	7830	13230	2000	28400
2016	12	22	0	1	5,78125	82,28125	6200	7950	9540	1800	25600
2016	12	23	0	1	5,375	86,77083	4070	1910	7640	1400	14000
2016	12	24	1	1	4,25	86,04167	4070	1910	7640	1400	14000
2016	12	25	1	1	4,833333	81,05556	4070	1910	7640	1400	14000
2016	12	26	0	1	3,866667	87,66667	4070	1910	7640	1400	14000
2016	12	27	0	1	5,666667	70,47619	4070	1910	7640	1400	14000

2016	12	28	0	1	7,522727	72,65909	4070	1910	7640	1400	14000
2016	12	29	0	1	5,171429	73,6	4070	1910	7640	1400	14000
2016	12	30	0	1	4,838235	73,57353	4070	1910	7640	1400	14000
2016	12	31	1	1	5,412698	76,19048	4070	1910	7640	1400	14000