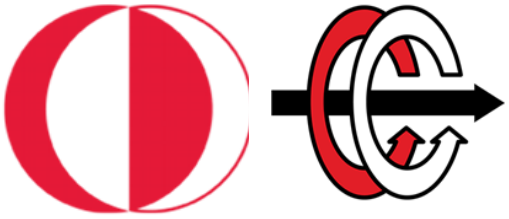
16.11.2018



MIDDLE EAST TECHNICAL UNIVERSITY
DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

EE474

Fall 2018

Term Project 1:

Forecasting the Power Demand of the Next Day with One-hour Resolution using MATLAB Environment

İVEN GÜZEL

2030831

INTRODUCTION

Prediction of electrical power required to meet the short term, medium term or long-term demand is called load forecasting. Short term load forecasting is used to make decisions ahead in order to prevent overloading. Short term load forecasting is usually done for one hour to one-week periods.

This report explains a simple, neural network based, day ahead load forecasting method using MATLAB environment. The report also explains the method I employed, why and how I used it, the methods I used to improve the accuracy of the results and the test results.

FITTING DATA USING NEURAL NETWORK

In Figure 1, the relationship between actual load demand of Ayaslı Research Center and its actual load demand from the previous day is given. It is pretty clear that the next day's load demand will be in a non-linear relationship with the load from today. If the relationship would be close to a linear one, then the points would align close to a 45^{o} line.

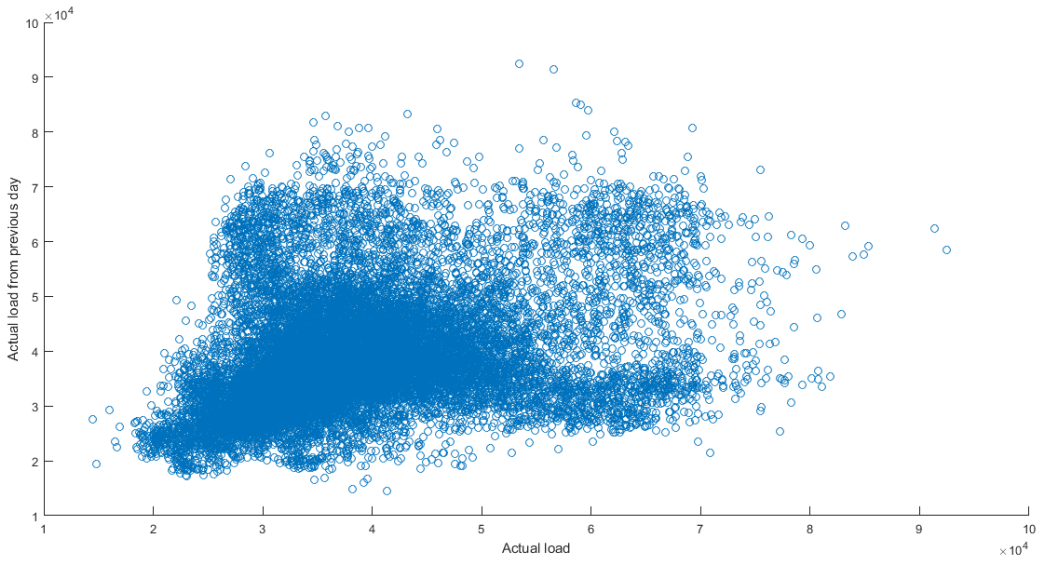


Figure 1: Actual load versus Actual load from the previous day

This complicated relationship between the input and output data led me to use neural networks. A neural network is enabled with the power of taking a decision by feeding the network with some input features and these features are processed inside the neural network's hidden layer. In the hidden layer, the network decides the relationship between the input features and the output by itself. Then the output layer compares the output of the hidden layer with the targets and adjusts the connections between input elements.

I used *nftool* in MATLAB and use Bayesian Regularization algorithm which can result in good generalization for difficult, small or noisy datasets. Training stops according to adaptive weight minimization (regularization). One advantage of using Bayesian Regularization is that,

unlike Levenberg-Marquardt algorithm, it does not stop when generalization stops improving, as indicated by an increase in the mean square error of the validation samples which is the case with our data as it can be seen in Figure 1.

CHOOSING OF INPUTS

I started with the load from the previous day, the load from the previous week, day of the week, and whether that day is a holiday as my input features. X is a (m x4) matrix where m is the number of samples. Note that each day starts with the data from 00:00:00, so I added one more row to the data given with the project description since it started with 00:00:05.

I used 20 neurons. I did not predict well as it can be seen in Figure 2, which was expected because of the nonlinear relation in Figure 1.

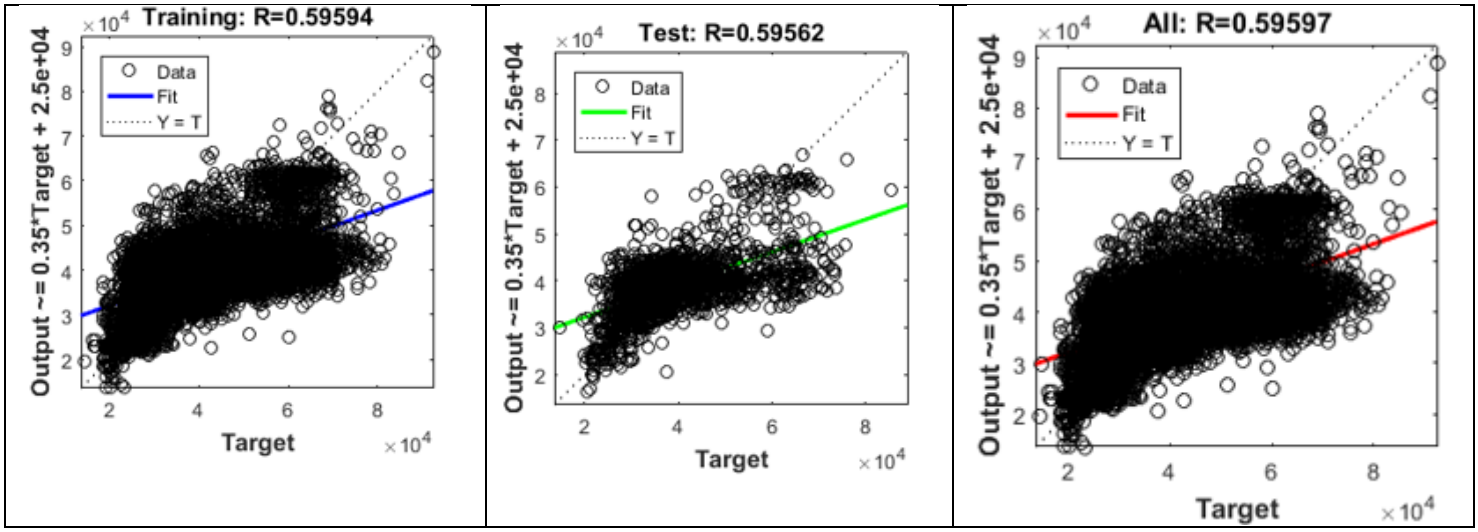


Figure 2: Regression plots of X(m,4) using 20 neurons

Next, I wanted to use column vectors as input features because from the online Deep Learning class that I am taking, I know it works better. X is a (4xm) matrix. It did not affect the results significantly but improved them as it can be seen in Figure 3. So, I kept going with this column format.

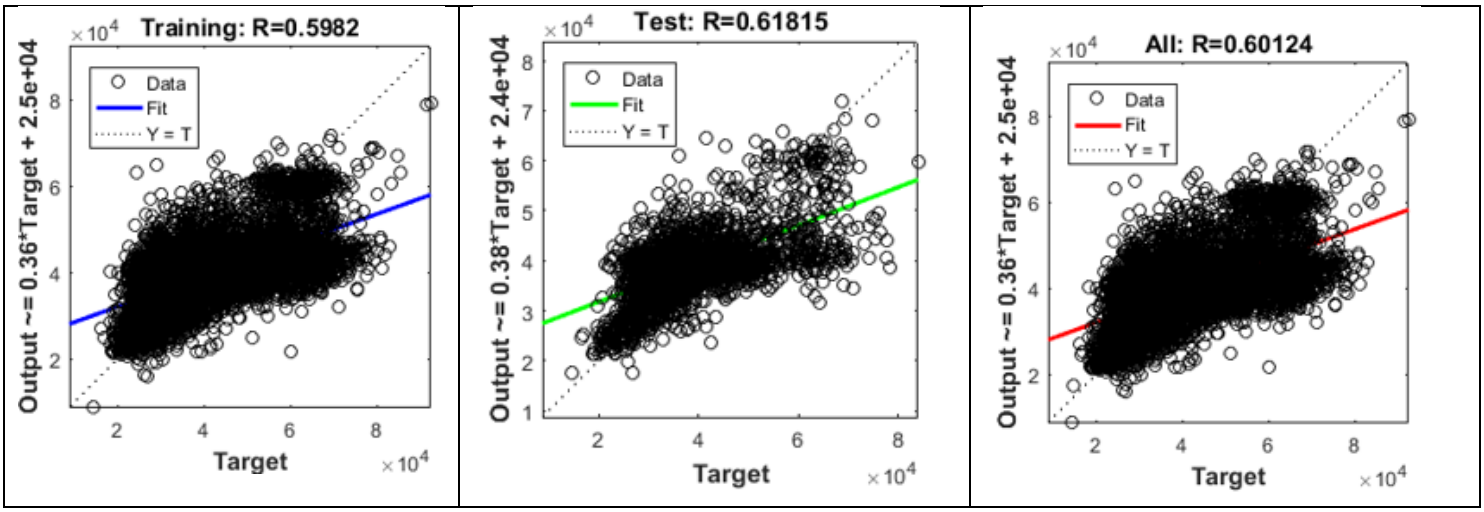


Figure 3: Regression plots of X(4,m) using 20 neurons

I added hour and minute data of the time the load is measured. X is a (6xm) matrix and I used 20 neurons. Regression plots are in Figure 4.

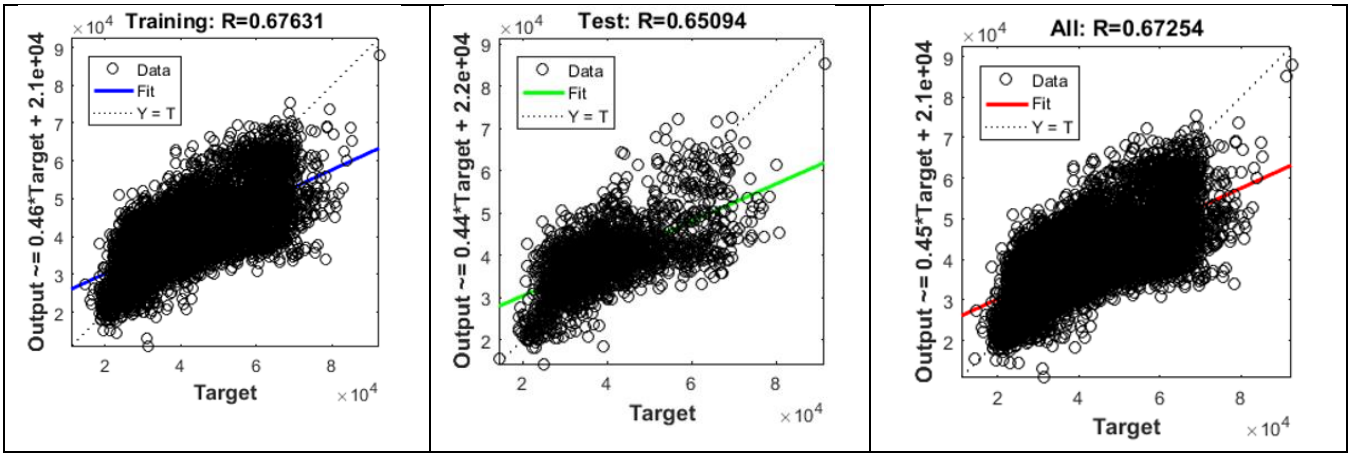


Figure 4: Regression plots of X(6,m) using 20 neurons

I added daily values of the mean, minimum and maximum values of dev point and temperature. X is a (12xm) matrix and I used 20 neurons again.

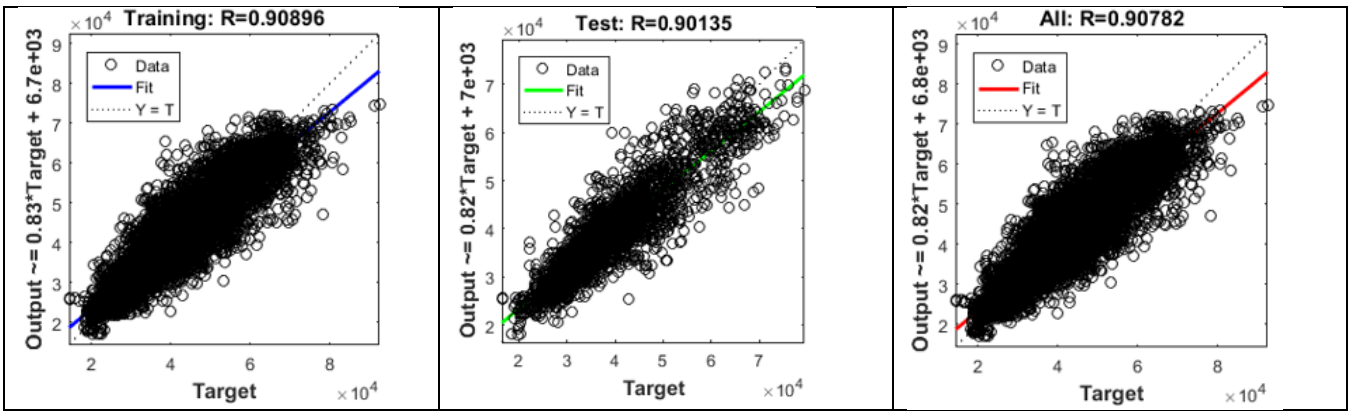


Figure 5: Regression plots of X(12,m) using 20 neurons

Since retraining changes the initial weights and biases of the network, and may produce an improved network, I decided to retrain a couple of times. Retraining the network did not improved the values so I increase neuron numbers to 50 and try again.

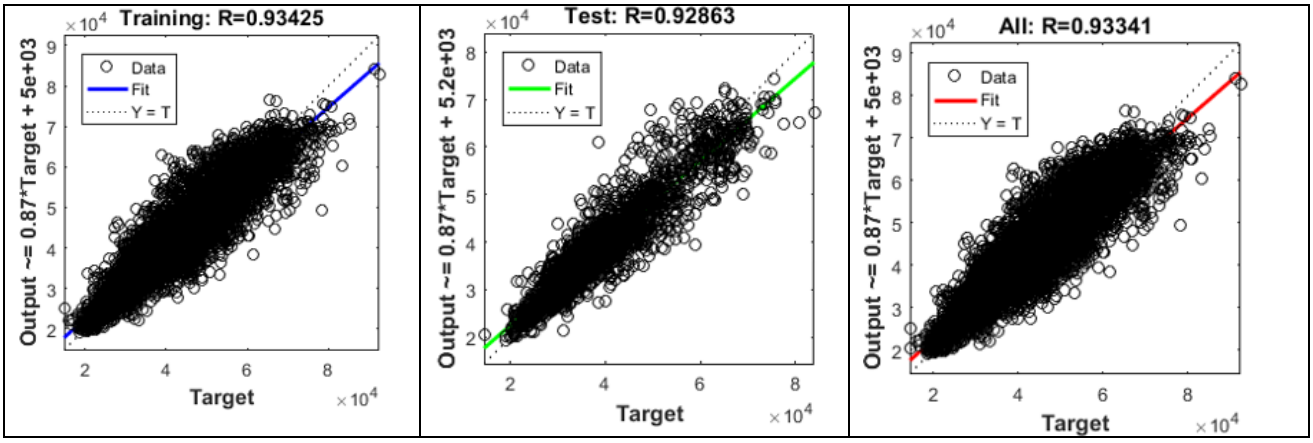


Figure 6: Regression plots of X(12,m) using 50 neurons

Finally, I added daily and hourly mean and standard deviation of the load from the previous day using 20 neurons. Now, X is a (16,m) matrix. It gave better results (Figure 7) comparing to Figure 5. Therefore, I increase the neuron number and retrain a couple of times but as it can be seen from Figure 8, it gave similar results with Figure 6. I conclude that, since a larger network provides better fitting by acquiring more complicated relations between input features, with increased number of neurons daily and hourly mean and standard deviation of the load information may be redundant. However, I decided to keep them as input features anyway, since they do not hurt the results. Error histogram of the training can be observed in Figure 9.

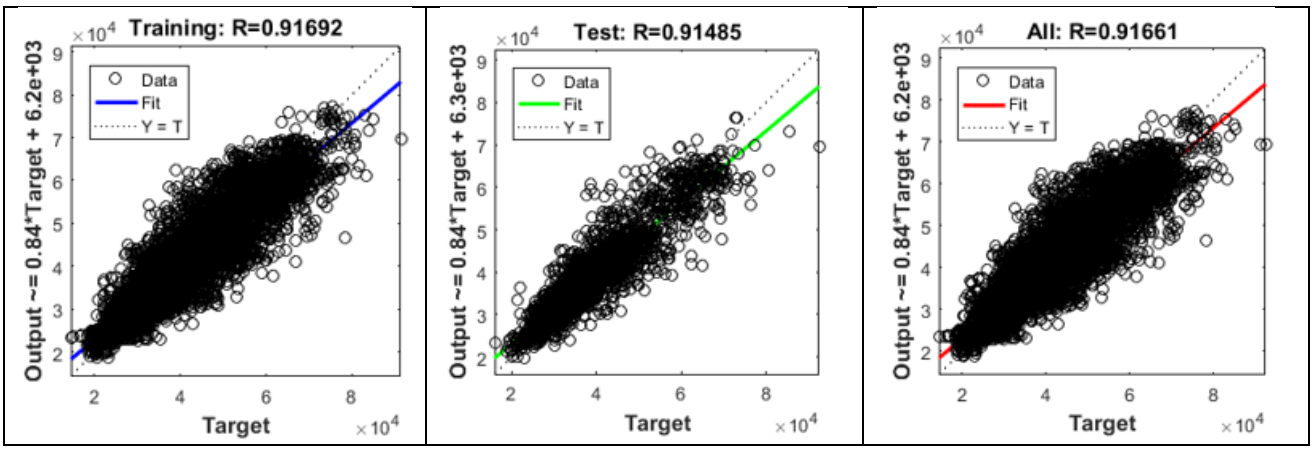


Figure 7: Regression plots of X(16,m) using 20 neurons

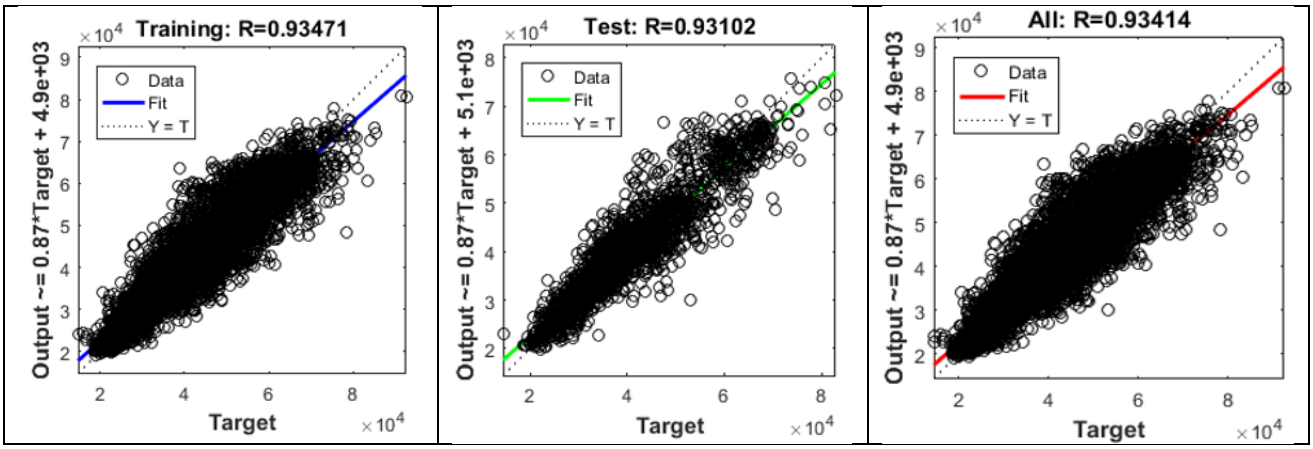


Figure 8: Regression plots of X(16,m) using 50 neurons

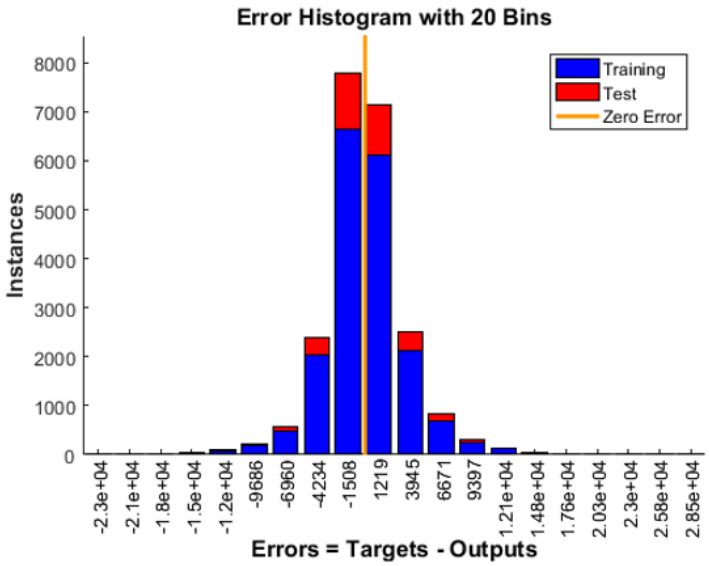


Figure 9: Error Histogram when X(16,m) using 50 neurons

As it can be seen in Figure 9, error of the forecasting is not satisfactory. I also plotted the actual load versus forecasted load and error which can be observed in Figure 10.

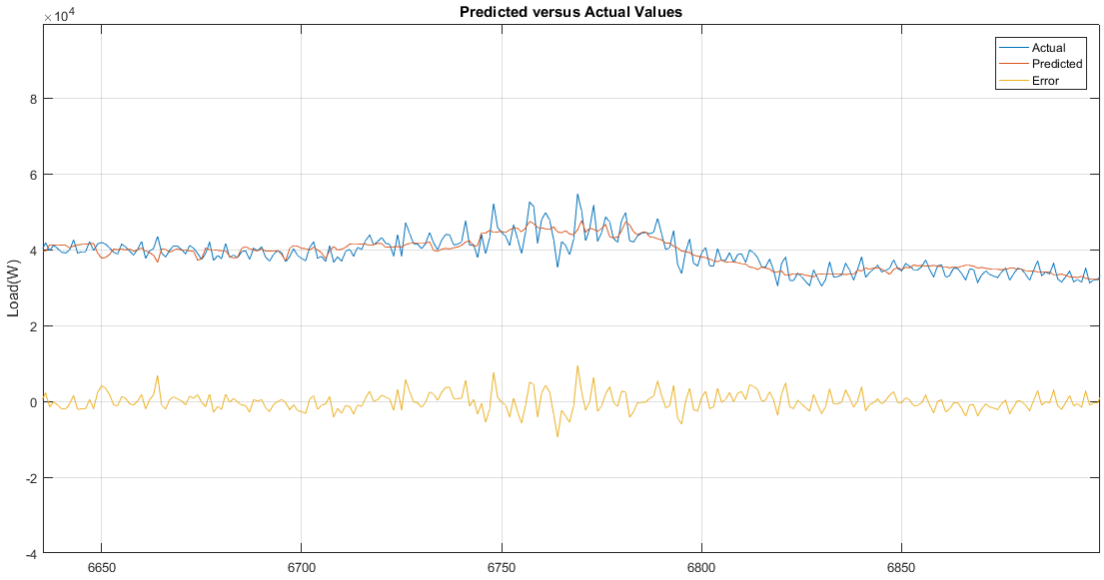


Figure 10: Predicted versus Actual Load Demand

A Guide to Use the forecasting function:

e203083_guzel_EE474_P1.m takes xls file name as input and gives means of the predicted load demands in every hour. Your xlsx file should look similar to Figure 11. Furthermore, since the function generates means of every hour the number of rows should be 12 or its integer multipliers, i.e., 12,24,36,..., 288. There should be 288 rows for prediction of one day.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	21918	17406	0	0	3	0	5	1	-4	6	3	0	26008,11	5552,119	20459	1375,31
2	18708	17406	0	5	3	0	5	1	-4	6	3	0	26008,11	5552,119	20459	1375,31
3	20316	18426	0	10	3	0	5	1	-4	6	3	0	26008,11	5552,119	20459	1375,31
4	22794	21996	0	15	3	0	5	1	-4	6	3	0	26008,11	5552,119	20459	1375,31
5	19908	18786	0	20	3	0	5	1	-4	6	3	0	26008,11	5552,119	20459	1375,31
6	19602	18864	0	25	3	0	5	1	-4	6	3	0	26008,11	5552,119	20459	1375,31
7	20148	18144	0	30	3	0	5	1	-4	6	3	0	26008,11	5552,119	20459	1375,31
8	21660	21024	0	35	3	0	5	1	-4	6	3	0	26008,11	5552,119	20459	1375,31
9	20496	19548	0	40	3	0	5	1	-4	6	3	0	26008,11	5552,119	20459	1375,31
10	19572	17592	0	45	3	0	5	1	-4	6	3	0	26008,11	5552,119	20459	1375,31
11	18372	18798	0	50	3	0	5	1	-4	6	3	0	26008,11	5552,119	20459	1375,31
12	22014	19974	0	55	3	0	5	1	-4	6	3	0	26008,11	5552,119	20459	1375,31
13	20310	19542	1	0	3	0	5	1	-4	6	3	0	26008,11	5552,119	19783	1580,044

Figure 11: xlsx file example

A: Load from the previous day

B: Load from the previous week

C: Hour

D: Minute

E: The day of the week (1 is Sunday)

F: Holiday or weekday (1 is holiday)

G: Maximum dev point of the day

H: Average dev point

I: Minimum dev point

J: Maximum temperature of the day

K: Average temperature

L: Minimum temperature

M: Daily load mean of the previous day

N: Daily standard deviation

O: Hourly mean of the data from the previous day

P: Hourly standard deviation

Table 1 provides some weather information:

Table1: Daily weather values of February 2018

Time	Temperature (° C)			Dew Point (° C)		
Feb	Max	Avg	Min	Max	Avg	Min
1	12	2	-7	-1	-4	-8
2	9	2	-6	0	-2	-6
3	11	7	2	2	1	-1
4	11	6	1	2	1	-2
5	9	4	1	4	2	-1
6	10	5	0	4	2	-1
7	13	7	0	4	2	-1
8	11	6	2	6	4	2
9	11	6	1	5	1	-3
10	10	4	-1	5	1	-2
11	16	8	0	6	2	-1
12	10	6	0	3	1	-2
13	8	3	-2	1	-2	-5
14	11	3	-4	2	-2	-5
15	7	2	-3	3	1	-2
16	12	7	2	4	1	-1
17	13	5	-3	2	-1	-4
18	10	3	-4	-2	-3	-5
19	12	3	-5	0	-3	-6
20	12	4	-4	1	-3	-6
21	9	7	4	5	3	1

Time	Temperature (° C)			Dew Point (° C)		
22	13	6	-2	5	2	-2
23	14	8	2	5	3	2
24	13	7	-1	4	2	-1
25	7	3	0	3	0	-4
26	7	3	0	2	-1	-3
27	10	6	2	3	1	-2
28	12	6	1	4	2	-2