Scientific Research Publishing

Open Journal of Modelling and Simulation, 2026, 14(2), 63-72
https://www.scirp.org/journal/ojmsi
ISSN Online: 2327-4026
ISSN Print: 2327-4018

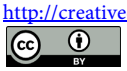
Machine Learning: An Overview (Part Two)

Mohd Izhan Mohd Yusoff

Telekom Malaysia Berhad, Kuala Lumpur, Malaysia
Email: mohdizhanmohdyusoff@gmail.com

How to cite this paper: Yusoff, M.I.M. (2026) Machine Learning: An Overview (Part Two). *Open Journal of Modelling and Simulation*, **14**, 63-72.
https://doi.org/10.4236/ojmsi.2026.142004

Received: March 26, 2026
Accepted: April 21, 2026
Published: April 24, 2026



Abstract

NILM (Nonintrusive Load Monitoring) or appliance recognition software that uses algorithms, namely Hidden Markov Model (HMM) and Factorial HMM, to detect changes in electricity values entering a building, thus inferring which appliances are used and their individual energy consumption. In this paper, a "simple" technique is introduced that uses combinatorics on a specially designed observation matrix to detect hidden appliances and their electricity usage by dissecting the building's total electricity usage.

Keywords

Nonintrusive Load Monitoring (NILM), Combinatorics, Observation Matrix, Pentagon

1. Introduction

NILMs (Nonintrusive Load Monitoring) or appliance recognition software are algorithms that detect changes in electricity values entering a building to infer which appliances are used and their individual energy consumption. NILM uses Hidden Markov Model (HMM) and Factorial Hidden Markov Model (FHMM) to determine which appliances are switched "ON" by analysing the building's electricity usage behaviour [1]-[4]. The Expectation-Maximization algorithm is used to find the final parameters of the above-mentioned techniques [5] [6]. The weaknesses of NILM are its inability to deliver cost savings and scale, as it must work with systems that collect and process data frequently (preferably every second or minute) [1].

Example of how NILM is implemented: A special sensor is placed at the electricity meter (or distribution) box of house "A" that measures the "total" electricity usage, say every minute. The system, which is installed on a server, desktop, or laptop and linked to the sensor, starts processing when the sensor has collected a block of data, say, 30 minutes. House "A" laptop (or desktop) is preferred over the

"cloud" server because of the "shared" resources, as shown by the latter, which might impact the processing speed. The sensor produces 48 blocks of 30-minute data per day. The system must process and produce (or display) results within 30 minutes before accepting the new batch of data. The word "total" presents extraordinary challenges because a single "sensor" is used instead of multiple "timer" devices (as suggested by SA^{2}VING), and without the system's aid, one must make a good "guess" about which household appliances are switched "ON".

In this paper, we would like to propose using a "simple" technique, namely combinatorics, on a specially designed observation matrix, for NILM.

2. Methods

Combinatorics, a part of mathematics, focuses on counting, arranging, and structuring finite discrete objects. Using techniques such as permutations (ordered arrangements), combinations (unordered arrangements), and the pigeonhole principle, it solves complex remuneration problems [7]. For the case of three appliances labelled, say, 1, 2, and 3, combinatorics is used to create a series of tests for a system to perform (from this point onwards, they are known as stages): (1), (2), (3), (1, 2), (1, 3), (2, 3), and (1, 2, 3), where the appliance, represented by its label, that is switched "ON", is placed in the brackets.

The following are the steps taken to create an observation matrix using appliance (electricity usage) data:

Let t(1), t(2), t(3), ..., t(m) represent the columns of an "m"-minute block of data. Let Obs(1),..., Obs(n) be the "n" observations (*i.e.*, electricity usage data when the appliance is switched "ON") collected for an appliance. For the first row, place Obs(n) in the first column, and "0.0" (meaning the appliance is switched "OFF") in the rest of the columns. For the second row, place Obs(n − 1) in the first column, Obs(n) in the second column, and "0.0" in the rest of the columns. For the third row, place Obs(n − 2) in the first column, Obs(n − 1) in the second column, Obs(n) in the third column, and "0.0" in the rest of the columns. Repeat the above process until reaching "n + (m − 1)" row where the first "m − 1" columns are filled with "0.0", and Obs(1) fills the last column. Table 1 shows an example of an observation matrix where "m" and "n" equal five.

Table 1. An example of the observation matrix.

t(1)	*t*(2)	*t*(3)	*t*(4)	*t*(5)
Obs(5)	0.0	0.0	0.0	0.0
Obs(4)	*Obs*(5)	0.0	0.0	0.0
Obs(3)	*Obs*(4)	*Obs*(5)	0.0	0.0
Obs(2)	*Obs*(3)	*Obs*(4)	*Obs*(5)	0.0
Obs(1)	*Obs*(2)	*Obs*(3)	*Obs*(4)	*Obs*(5)
0.0	*Obs*(1)	*Obs*(2)	*Obs*(3)	*Obs*(4)

Continued

0.0	0.0	*Obs*(1)	*Obs*(2)	*Obs*(3)
0.0	0.0	0.0	*Obs*(1)	*Obs*(2)
0.0	0.0	0.0	0.0	*Obs*(1)

Note that (electricity usage) data for each appliance is collected as follows: when the appliance is switched "ON", (electricity usage) data is collected from the same sensor as mentioned above every minute until the appliance switches "OFF" (either by hand or automatically).

As mentioned in the Introduction, the system starts to process when the sensor (located at the distribution box) has collected a block of "m"-minute data ("m" equals 30), *i.e.*, "dataTotal(1)" – "dataTotal(m)". The block of data, represents the house's total electricity usage, is subjected to the following process performed by the system: For stages 1 - 3, represented by (1) - (3), the appliance labelled, say, 1 observation matrix "om1(i, 1)" – "om1(i, m)", i = 1, 2, …, "n1 + m − 1", where each "i" is compared to the above block of data, *i.e.*, "dataTotal(1)" – "dataTotal(m)". The system displays the results if, through the comparison process, it found similarities. For stages 4 - 7, represented by (1, 2) – (1, 2, 3), the appliance labelled, say, 1 observation matrix "om1(i, 1)" – "om1(i, m)", i = 1, 2, …, "n1 + m − 1", where each "i" is paired with the appliance labelled, say, 2 observation matrix "om2(h, 1)" – "om2(h, m)", h = 1, 2, …, "n2 + m − 1". The summing of the pairs is compared to the above block of data, *i.e.*, "dataTotal(1)" – "dataTotal(m)". The system displays the results if similarities are found.

3. Results

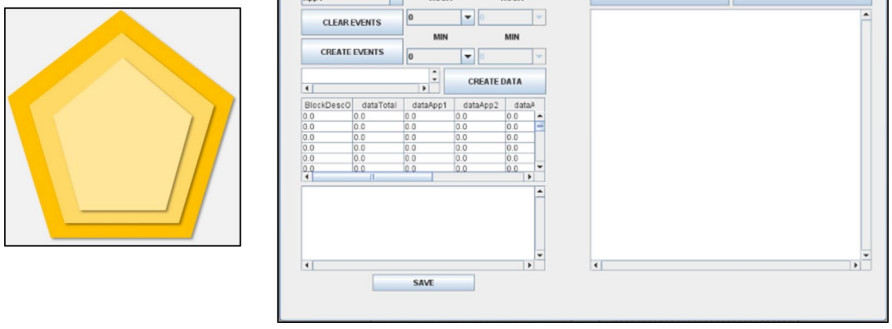
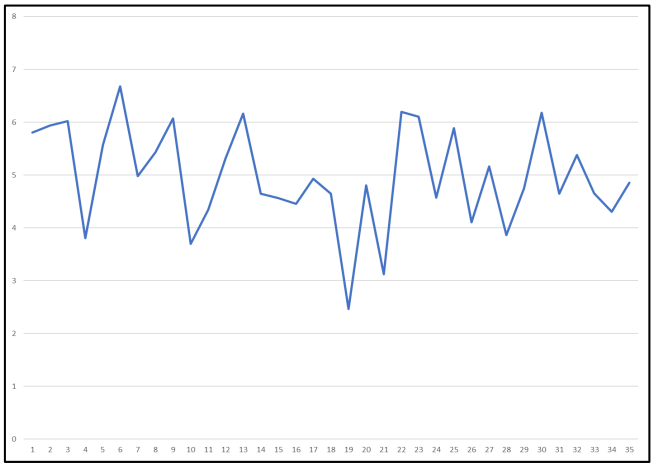


Figure 1. Pentagon.

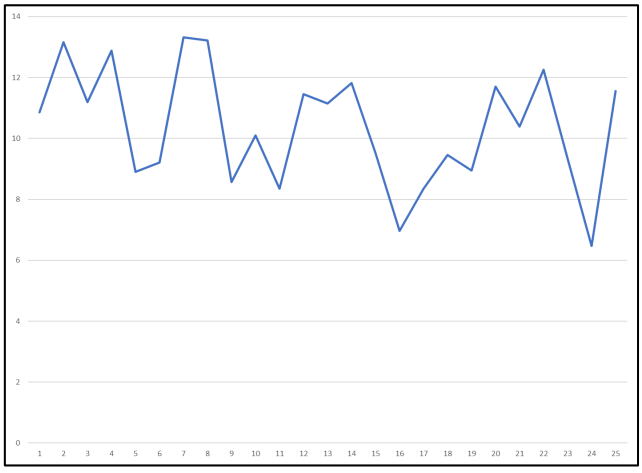
The system, called **Pentagon** (refer to Figure 1), was developed to demonstrate the effectiveness of a "combinatorics" approach (an approach "unpopular" amongst some machine learning researchers) on a "specially designed" observation matrix that can be incorporated into the "SA²VING" application [1].

The following are the assumptions used when developing the system:

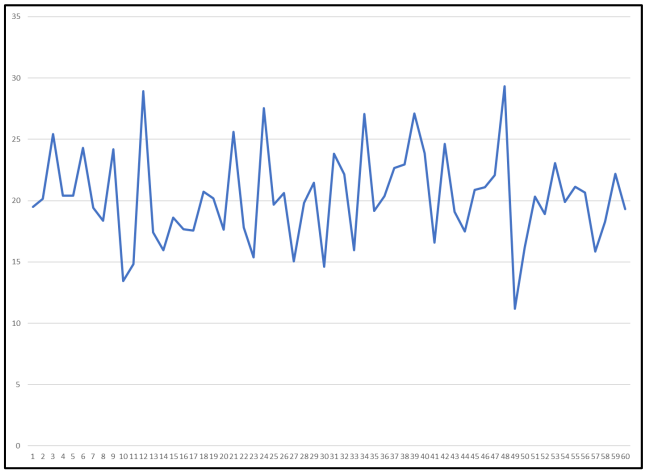
Household appliances are labelled "App1", "App2", and "App4", where "App1" takes 35 minutes to complete its tasks (or switched "OFF" automatically), whilst "App2" takes 25 minutes. A washing machine is a good example of "App1" and "App2". The duration for "App4" is not fixed; it can range from 1 minute to 60 minutes. An air-conditioner (or fan) is a good example of "App4". After an appliance has completed its tasks (or switched "OFF"), a 30-minute "cooling" period is required before switching it "ON" again. Thus, avoid any "erratic" or "random" behaviour when switching appliances "ON". "Simulated (electricity usage) data" for "App1", "App2", and "App4" are visualized graphically in **Figures 2(a)-(c)**, respectively, and are saved in the "observation matrix" database as exemplified by **Figure 3(a)** for "App1". Note that for "App4", due to none fixed duration, there are 60 observation matrices where the name of the file represents (or shows) how long "App4" is switched "ON"; for example, "App4Outcome_10.txt" means "App4" is switched "ON" for ten minutes, represented by "Obs(1)" - "Obs(10)", refer to **Figure 3(b)**.



(a)

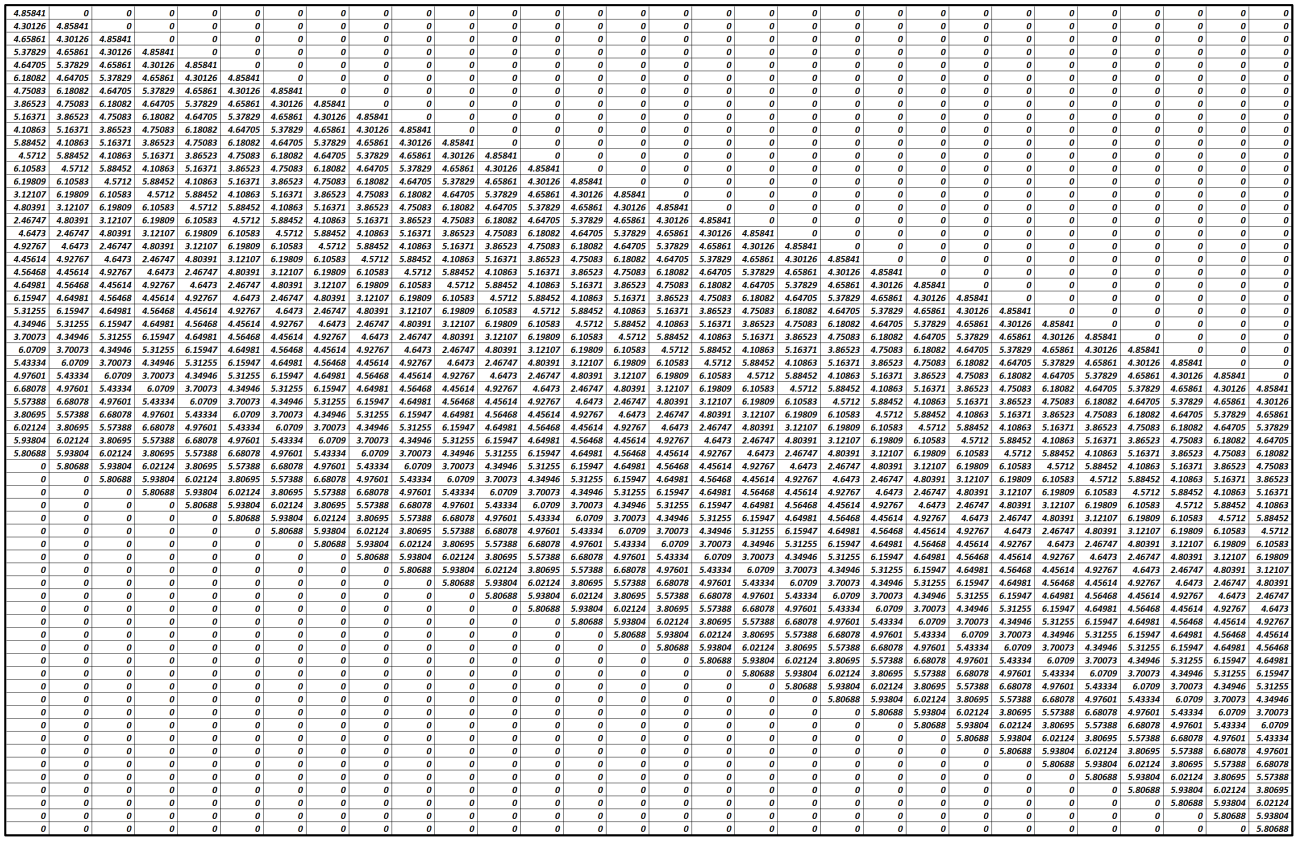


(b)

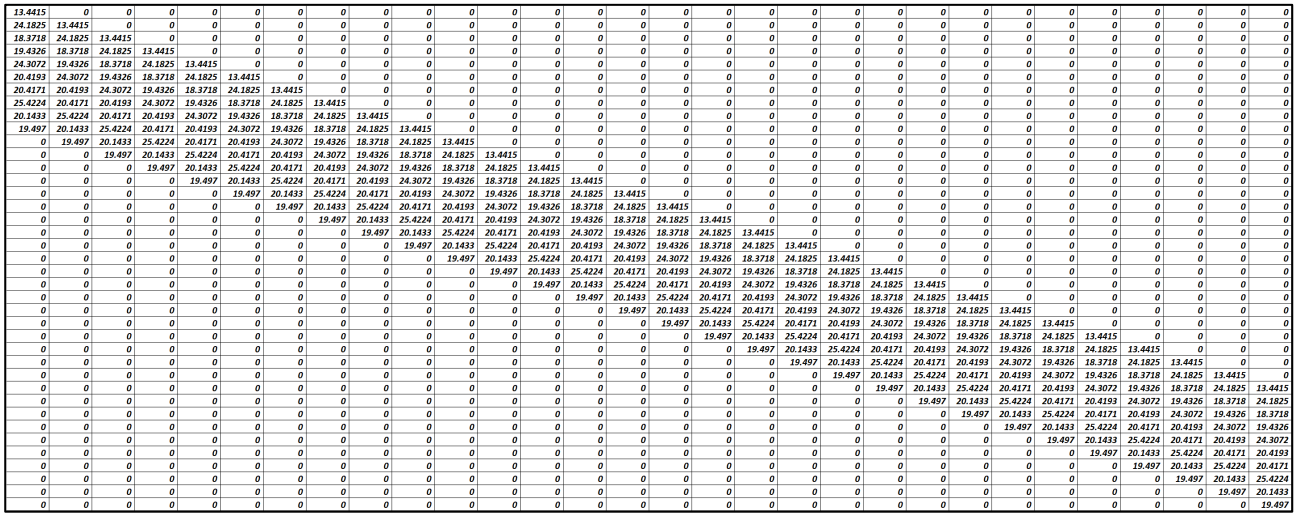


(c)

Figure 2. Simulated (electricity usage) data for (a) "App1", (b) "App2", and (c) "App4".



(a)



(b)

Figure 3. Observation matrix for (a) "App1" and (b) "App4". Note that "App4" is switched "ON" for ten minutes.

Figure 4 is a subsection of the combinatorics pane. It helps to create the appliance's events, providing users with a hands-on experience. For example, if the user selects "App1" under "Appliances" and, under "Start", "Hour", and "Min", set the hour and minute to 0 and 10, respectively, it means "App1" will start its operation at "(0:10)". The user can choose between 0 and 23 for "Hour" and 0 and 59 for

Figure 4. A subsection of the Pentagon's combinatorics pane.

"Minute". "Hour" uses the 24-hour standard (*i.e.*, "(13:00) = (1:00 pm)" and so forth). Click on the "Create Events" button. The "Clear Events" button performs the opposite. The inputs appear in the text box under the said button. Repeat the same process by selecting "App2", but this time set the hour and minute to 0 and 15 or "(0:15)", respectively. Note that, under "Finish", "Hour", and "Min" are disabled when the user selects "App1" and "App2". For "App4", set the hour and minute under "Start" to 0 and 5, respectively, and under "Finish" to 0 and 20, respectively. It means "App4" starts operation at "(0:5)" and finishes at "(0:20)". Figure 5 shows all the values entered. Note that each appliance could be switched "ON" more than once each day.

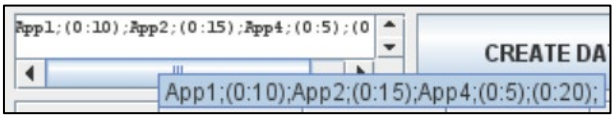


Figure 5. The event's text box displaying all the entered values.

Click "Create Data", and the table in the Combinatorics pane is updated as shown in Figure 6.

BlockDescO	dataTotal	dataApp1	dataApp2	dataA
Block1;(0:5...	19.497049...	0.0	0.0	19.497(
Block1;(0:6...	20.143263...	0.0	0.0	20.143
Block1;(0:7...	25.422393...	0.0	0.0	25.422
Block1;(0:8...	20.417080...	0.0	0.0	20.417(
Block1;(0:9...	20.4192684	0.0	0.0	20.419
Block1;(0:1...	30.114093...	5.8068810...	0.0	24.307

Figure 6. The table showing all the data generated from the entered events.

Click "Nilm(Real)", and the system uses Combinatorics, observation matrix, and all the information stored in the database to split the "dataTotal" into "dataApp1", "dataApp2", and "dataApp4" (which is the system's aim or goal). To understand how "Nilm(Real)" works, click "Nilm(Sim)" first, which uses "predefined" data input. The results appear in the table and in the text box under the said button, as shown in Figure 7. Note that, for each block, the results are divided into "(Input):", "(Start:###)", "(Finish:###)", and "(Results):". Refer to the follow-

ing figures. The "(Start:###)" and "(Finish:###)" show how long the system takes to process the block involving at most seven stages. The seven stages are in the file "ThreeApps.txt" located in the folder "Appliances Events" and are mentioned in "Methods".

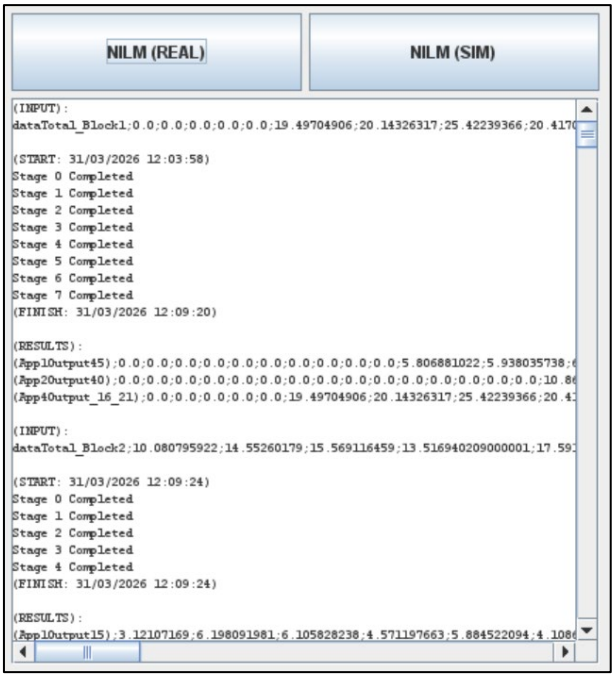


Figure 7. The text box displays the results when the user clicks the "Nilm(Real)" button.

Click "Save" (located under the table) to save all of them (*i.e.*, the results) in two files, namely "data.txt" and "results.txt", in the folder "combinatorics".

The system provides (apart from the table) a visual presentation of the results in the "Graphs" pane. After selecting "data" and "block" from the drop-down list located near the top of the pane, click "Plot", and graphs appear as shown by the following figures. The block descriptions are in the drop-down list located near the top of the pane. For example, the graphs show that "NILMdataApp1" fits "dataApp1" perfectly; refer to **Figure 8(a)** and **Figure 8(b)**. "dataApp2" and "dataApp4" show similar results; for the former, refer to **Figure 8(c)** and **Figure 8(d)**, and the latter, **Figure 8(e)**. The results are further verified and shown in **Table 2**. To facilitate comparison, the word "NILM" is added to the title (or name) of the data generated when the "NILM (Real)" button is clicked.

4. Discussion

Referring to the previous section, the system, using a "simple" technique, namely combinatorics, on a specially designed observation matrix, manages to reveal hidden appliances and electricity usage for each detected appliance by dissecting the total electricity usage. Due to the combinatorial technique's limitations, the

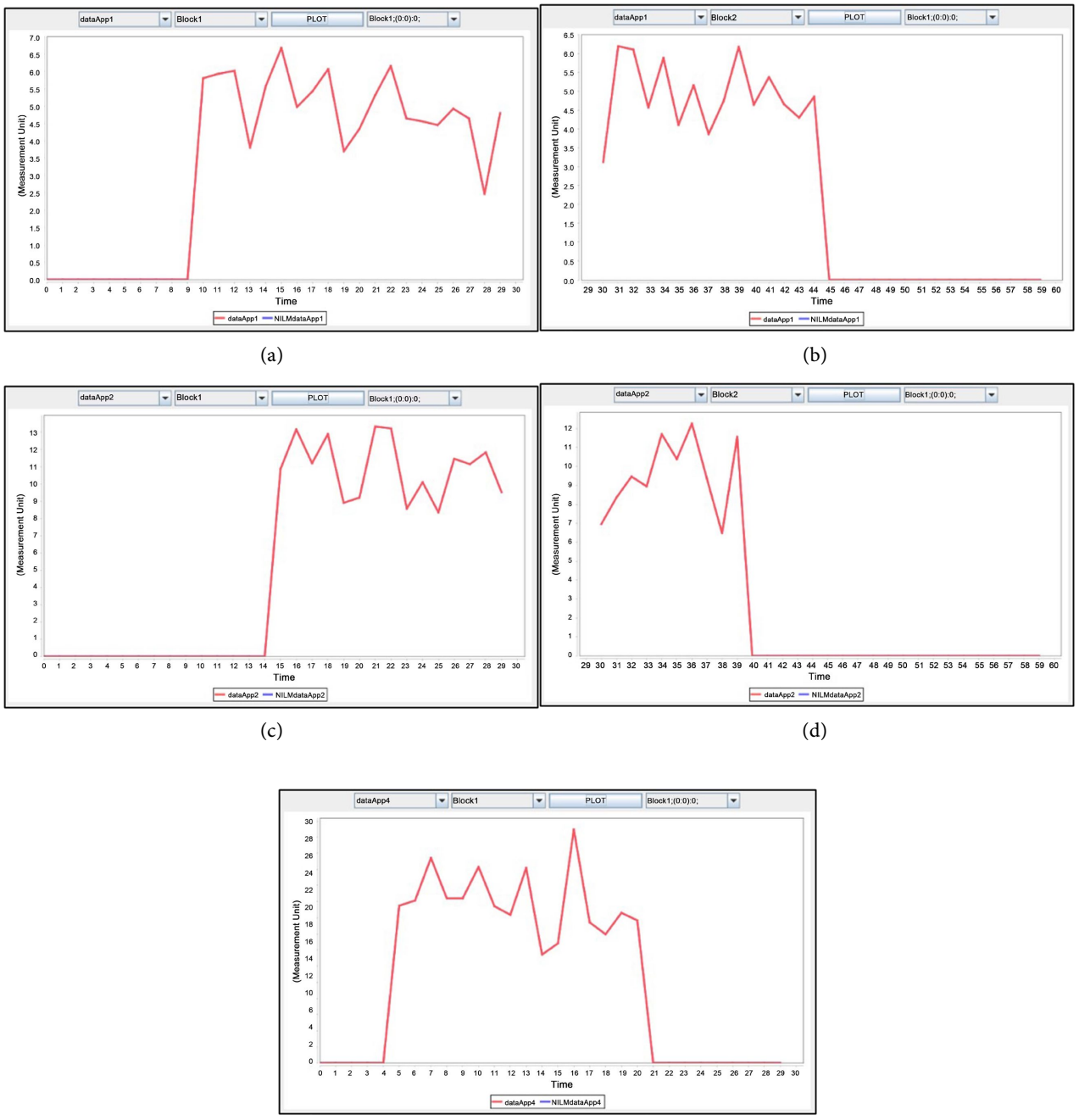


Figure 8. Pentagon's Graphs pane displays the results graphically when the user clicks the "Nilm(Real)" button. Note that, for each "Block" and for comparison purposes, two graphs are plotted for "dataApp1", "dataApp2", and "dataApp4".

Table 2. The mean and standard deviation of errors are calculated for "dataApp1", "dataApp2", and "dataApp4".

	Mean	*Standard deviation*
"dataApp1"	0.0	0.0
"dataApp2"	0.0	0.0
"dataApp4"	0.0	0.0

processing time will increase (exponentially) if more than three appliances are added to its database. Grouping appliances and labeling them as "Others" to reduce the number of appliances involved will increase the complexity of creating the observation matrix. The collection of each appliance's (electricity usage) data to create an observation matrix is done in the building itself. The process could also be done in the laboratory, where the appliances are identified or selected through marketing studies of famous brands. If an appliance, say, a fan, has three speeds and each has its own electricity usage signature, then the observation matrix should be created for each speed. The labeling should include, in addition to the appliance's name, the speed as well. Some Pentagon features would be deleted if implemented in the field, for example, "Create Events", "Clear Events" and "Create Data". In its place, new features such as "Create Observation Matrix" and "Create (Experimental) Stages" are introduced.

If the "cooling" period of 30 minutes, as mentioned in the previous section, is not included and to "capture" the erratic or random behaviour of a user when switching "ON" an appliance, then the observation matrix is modified in the following manner, where Table 1 is used as an example:

Table 1's first row is "expanded", as shown in Table 3. It involves creating four rows where, in the first row, "Obs(5)" fills the first column, "0.0" fills the second to fourth columns, and "Obs(1)" fills the fifth column. In the second row, "Obs(5)" fills the first column, "0.0" fills the second to third columns, "Obs(1)" and "Obs(2)" fill the fourth and fifth columns, respectively. Repeat the above process for the third to fourth column, until finally "Obs(5)" fills the first column, "Obs(1)" - "Obs(4)" fill the second to fifth column.

Table 3. The expansion of Table 1's first row.

$t(1)$	$t(2)$	$t(3)$	$t(4)$	$t(5)$
$Obs(5)$	0.0	0.0	0.0	$Obs(1)$
$Obs(5)$	0.0	0.0	$Obs(1)$	$Obs(2)$
$Obs(5)$	0.0	$Obs(1)$	$Obs(2)$	$Obs(3)$
$Obs(5)$	$Obs(1)$	$Obs(2)$	$Obs(3)$	$Obs(4)$

Table 1's second through fourth rows are "expanded" using the same approach. All of them, including Table 1, are saved separately in the database.

Note that the above modification is workable if the appliance's duration, represented by the number of observations, equals or exceeds the number of time stamps. Further research is required for cases where the appliance's duration, represented by the number of observations, is not fixed or less than the number of time stamps.

Further research is also needed, especially if more than one appliance has the same electricity usage "signature"; for example, the building has two air conditioners with the same features, such as horsepower and brand. For this case, they

should be given different "labels". Although this action would make it easier to create the observation matrix, the results produced by both are the same. The system will not be able to give a definitive answer to which of the two is switched "ON".

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Yusoff, M.I.M. (2024) Machine Learning: An Overview. *Open Journal of Modelling and Simulation*, **12**, 89-99. https://doi.org/10.4236/ojmsi.2024.123006

[2] Rabiner, L.R. (1989) A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, **77**, 257-286. https://doi.org/10.1109/5.18626

[3] Ghahramani, Z. and Jordan, M.I. (1997) Factorial Hidden Markov Models. *Machine Learning*, **29**, 245-273. https://doi.org/10.1023/a:1007425814087

[4] Kolter, J.Z. and Jaakkola, T. (2012) Approximate Inference in Additive Factorial HMMs with Application to Energy Disaggregation. *International Conference on Artificial Intelligence and Statistics*, La Palma, 21-23 April 2012, 1472-1482.

[5] Dempster, A.P., Laird, N.M. and Rubin, D.B. (1977) Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **39**, 1-22. https://doi.org/10.1111/j.2517-6161.1977.tb01600.x

[6] Bilmes, J.A. (1998) A Gentle Tutorial of the EM Algorithm and Its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. Technical Report, University of Berkeley, ICSI-TR-97-021. https://share.google/zmNEPysuvqbfdMZSf

[7] Navadi, W. (2011) Statistics for Engineers and Scientists. 3rd Edition, The McGraw-Hill Companies.