ENERGY CONSUMPTION PATTERNS PROFILING AND SIMILARITY INFERENCE

KAREEM AL-SAUDI

MSc. of Computing Science
Faculty of Science and Engineering
Rijksuniversiteit Groningen

March 26, 2021 – 1.0

SUPERVISORS:
Viktoriya Degeler
Michel Medema

ACKNOWLEDGMENTS

CONTENTS

ABSTRACT

To obtain a better understanding of energy consumption patterns at an individual household level; this paper seeks to construct distinct, yet widely applicable, energy profiles that capture frequently reoccurring patterns, habits, as well as behaviors associated with the individuals that occupy said households. These energy profiles can be constructed through a variety of different techniques, that includes the likes of clustering algorithms such as K-means and Density Based Spatial Clustering of Applications with Noise (DBSCAN), that will be explored throughout the duration of this paper both in terms of their efficacy in terms of capturing the patterns present in the data as well as how the resulting energy profiles may be used within the realm of forecasting. To do this, we will be making use of publicly available historical data with regards to energy consumption, the weather, as well as public calendar holidays alongside other temporal variables in hopes of answering the following questions: are there any repeated consumption patterns that we are able to extract from historical data? If so, can we find similarities in those patterns and connect them to external factors and finally, can we find periodicity in those patterns?

Part I

INTRODUCTION

1 INTRODUCTION

Our reliance on energy is one that is ever-increasing now, more-so than ever. As our dependence on electrical appliances continues to grow over the years [1–3] so too does our need for smarter, more sophisticated and advanced power grids. Thankfully, the convergence of multiple technologies – the likes of machine learning, data mining and ubiquitous computing has led to the rise of a solution in the form of *smart (electric) grids* as well as *smart environments* that are slowly but surely taking off in terms of their popularity and availability [4]. The resulting growth in the prevalence of smart grids gives us the opportunity to both control and monitor the energy consumption of individual households on a real time usage basis [5] leading to an increase in efficiency and subsequently, an overall reduction in terms of the amount of energy we, as the human race, consume. This opens up the possibility to alleviate some of the inherent risks associated with the growth in energy consumption whether that be our overall environmental footprint on the planet or, on a much smaller scale, the financial impact on both suppliers as well as consumers due to instabilities present in current, outdated power grid systems [6].

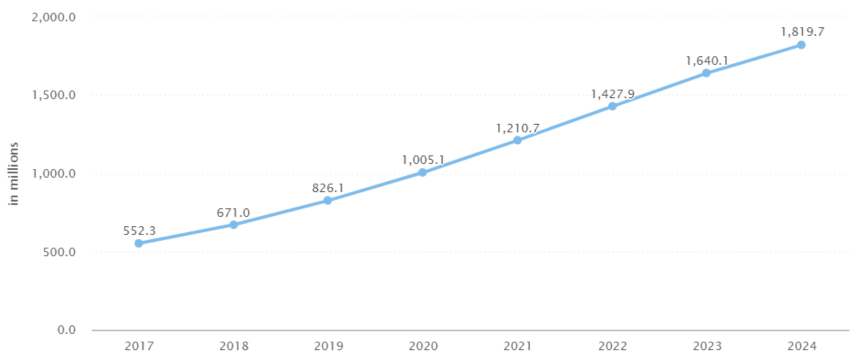


Figure 1.1: Historical/predicted growth in the number of appliances being used worldwide. Image source: [1] © 2019, Statista.

Applications developed under the increasingly popular smart grid framework provide us with the means to such an end. Existing solutions such as the Home Energy Management System (HEMS) and Battery Energy Management System (BEMS) aim to provide the end-user with the means to schedule, or otherwise manage, daily appliance operations taking into consideration external factors such as weather conditions, utility tariff rates as well as personal preference [5]. These solutions rely on the ability to capably predict or, in other words, *forecast* future trends in energy consumption [7] so as appropriately and sufficiently control and supply the correct energy load to the end-user [8]. The concept of load forecasting is far from novel having been extensively studied within the literature [9]; however, the majority of studies focus on load forecasting on the large-scale, regional level where an amalgamation of available data spanning numerous households provides more consistently obvious patterns as a result of the underlying diversity between households being lost when taking the aggregated residential level [10].

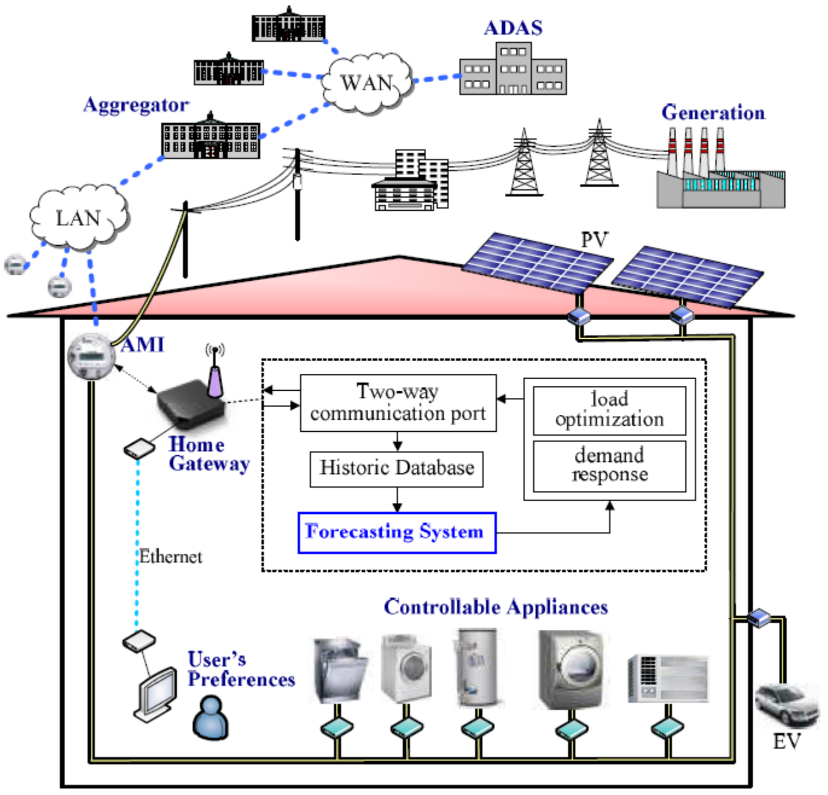


Figure 1.2: The HEMS architecture visualized. Image source: [11] © 2013, IEEE.

When exploring energy consumption at the individual household level, the diversity and complexity associated with human behaviour leads to extremely dynamic, volatile patterns that can prove to be highly dissimilar between households. In addition to this, certain households exhibit no clear pattern in energy consumption due to a high level of irregularity in the lifestyle of its occupants [10]. To account for this dissimilarity current, state-of-the-art methods generally benefit from a precursory clustering step within the forecasting pipeline [5, 6, 10] and this is the area of research that this paper seeks to tackle – how can we best construct energy profiles out of historical data and what are the effects of a clustering, or otherwise, classification step in the performance of a forecasting pipeline.

The following chapters of this paper will be organised as follows: chapter 2 will present related work within the field of clustering and classifying energy profiles so as to establish a baseline with which to compare our work to. Following that chapter 3 serves to provide a brief, intuitive explanation of the concepts related to this paper – this explanation will be fairly high-level in the name of preserving time and preventing the paper from being lengthy. Chapters 4 and 5 will both describe as well as visualize the historical data that we have on hand for the duration of this project. Ensuingly, chapter 6 will outline our methodology with regards to both our chosen clustering as well as forecasting techniques. Finally chapters 7 and 8 conclude the paper by presenting our results alongside a discussion and potential direction with regards to future work.

2 RELATED WORK

Energy management systems, such as the previously introduced HEMS and BEMS, are designed with the intent to both optimize and control the smart grid energy market. As previously stated, to be able to do this, these demand-side management systems require a priori knowledge about the load patterns and, as a result of this, the field of designing computationally intelligent load forecasting (CILF) systems has expanded quite rapidly in recent years with over 50 research papers related to the subject having been identified in existing literature [12]. In this chapter we will be exploring a compiled subset of this literature that specifically tackle the problem of energy profile construction as well as load forecasting. This is done so as to establish a baseline of understanding as to what has already been done within the field in terms of the two focal points of our forecasting pipeline: the precursory clustering step as well as the state-of-the-art forecasting models. Furthermore, by doing so we will be able to determine a benchmark with which to compare our proposed method to.

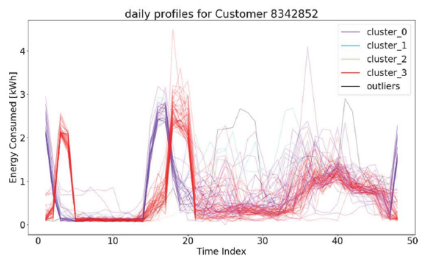
2.1 CLUSTERING AND ENERGY PROFILE CREATION

The chief issue that this paper seeks to address is that of creating interesting profiles in terms of recurrent patterns in energy consumption. To do this we will be making use of clustering algorithms that seek to partition our data into a number of clusters so that each of these clusters exhibit some metric of similarity or *goodness*. However a measure of goodness can inherently be seen as quite subjective with Backer and Jain [13] noting that, "in cluster analysis a group of objects is split up into a number of more or less homogeneous subgroups on the basis of an often subjectively chosen measure of similarity (i. e., chosen subjectively based on its ability to create "interesting" clusters) such that the similarity between objects within a subgroup is larger than the similarity between objects belonging to different subgroups.". We will be exploring papers in the existing literature that present different takes both in how they define similarity as well as their chosen clustering methodology.

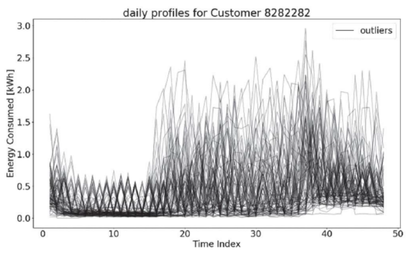
Stephen et al. [14] note that within an individual household, the daily routines and lifestyles of its occupants alongside the types of possessed major appliances may have a direct impact on the short-term load profile and cite *Practice Theory* [15] to explain the root causes of energy usage at a residential level. Practice theory states that the overall residential energy consumption is dictated by the usage of appliances through an ensemble effect of *practices*. These so-called practices can be seen as a series of doings which are governed by the diverse motives and intentions of the occupants of a household and can be classified within four major categories:

1. *Practical Understanding:* Routine activities that actors know when to do and what to do e. g., taking a shower or doing laundry.
2. *Rules:* Operations that are constrained by the technical limitations of the system e. g., a pre-programmed washing procedure of a dishwasher.
3. *Teleo-affective:* Goal-orientated behaviours e. g., making a cup of coffee or watching TV.
4. *General Understanding:* Practices with a greater degree of persistence and regular occurrence e. g., those associated with religious activities.

Further compounding on this Kong et al. [10] attempted to justify the observations made by Stephen et al. [14] by using a density-based clustering technique known as DBSCAN [16] to evaluate consistency in short-term load profiles. They remark on the benefits of using DBSCAN stating that as it does not require knowing the number of clusters in the data ahead of time and as it contains the notion of outliers it would be an ideal clustering technique to identify consumption patterns that repeat with a measure of noise akin to what is loosely defined by Practice Theory. Their findings are that the number of clusters as well as outliers varies greatly between households with some households exhibiting no clearly discernible patterns and some households (mostly) following fairly consistent daily profiles. This is visualized in Figures 2.1a and 2.1b.



(a) A case with only one outlier alongside two major clusters and two minor clusters.



(b) A case with no clusters at all.

Figure 2.1: Two widely different scenarios in the application of DBSCAN. Images source: [10] © 2019, IEEE.

Yildiz et al. [5] expand on traditional load forecasting techniques, such as the Smart Meter Based Model (SMBM) that they had previously presented [17], and present their own take in the form of a Cluster-Classify-Forecast (CCF) model. In traditional SMBMs, a chosen model, whether that be of a statistical variant or from the plethora of existing machine learning models, learns the relationship between the target forecasted loads when presented with some input data which, in our case, consists of some historical lags in terms of energy consumption, data with regards to the weather and temporal information with respect to the time, calendar date etc. The CCF takes this a step further by making use of both K-means and Kohonen's Self-Organizing Maps (SOM) [18] to group profiles that are most similar to each other. After obtaining and validating the output of their chosen clustering techniques they investigate the relationship between the clustering output and other temporal variables, such as the weather, by using a Classification and Regression Tree (CART) method [19].

2.2 FORECASTING MODELS

Numerous studies have been conducted with the intent to forecast energy consumption which range from methods the likes assessed by Fumo and Rafe Biswas [20] in the form of multiple linear regression to methods such as novel deep pooling Recurrent Neural Networks (RNN) introduced by Shi, Xu, and Li [21]. The majority of these forecasting models, whether they be statistical, machine learning or deep learning based, can be classified into 2 main categories: single technique models in which only a single, heuristic algorithm (e. g., a Multi-Layer Perceptron (MLP) or Support Vector Machine (SVM)) is used as the primary forecasting method and hybrid methods that encapsulate 2 or more algorithms [12] such as the Convolutional Neural Network Long Short-Term Memory (CNN-LSTM).

Kong et al. [10] employ the use of a Long Short-Term Memory (LSTM) network as it is generally the ideal candidate when attempting to learn temporal correlations within time series data sets; however, their final results are not very promising boasting a mean absolute percentage error (MAPE) of approximately 44% over variable time steps. This could be a result of poor hyperparameter tuning stating that, *"tuning 69 models for each of the candidate methods is very time-consuming for this proof-of-concept paper"* leading us to believe that there is definitely room for improvements to be made on the core concepts of their work.

Yildiz et al. [5] use the clusters they formed as described earlier alongside their assignments to build SMBMs, in this case through the use of a Support Vector Regression (SVR) model, and find that, alongside improvements to load forecasting accuracy, they are able to reveal vital information on the habitual load profiles of the households they were exploring. Unfortunately, they do not indicate any potential reasoning as to why they chose to use K-means and Kohonen's SOMs in place of potentially more effective clustering methods citing only that K-means is the most popular clustering technique [19] and that SOMs is generally used as an extension to neural networks for the purposes of clustering. Additionally, their results only include values that are indicative of their chosen technique's performance on their specific data set presenting performance metrics such as normalized root mean square error (NRMSE) and normalized mean absolute error (NMAE) rendering us unable to compare the performance of their proposed method.

Kim and Cho [22] present a more modern take on load-forecasting proposing a hybrid CNN-LSTM network that is capable of extracting both temporal and spatial features present in the data. The use of convolutional layers within the realm of load forecasting is brilliant allowing for the network to take into account the correlation between multivariate variables while minimizing noise that can eventually be fed into the LSTM section of the network that finally generates predictions. Their paper proposes such a network citing that the major difficulties with such an approach mainly boil down to hyperparameter tuning which can be remedied through a variety of means that include the likes of genetic algorithms (GA) or through the use of packages such as Keras Tuner maintained by O'Malley et al. [23]. Furthermore, Kim and Cho [22] did not explore the possibility of implementing a precursory clustering step which could have lead them to substantial improvements in their final MAPE.

2.3 SUMMARY

Table 2.1 depicts the wide variety in the different methods explored throughout the duration of chapter 2. Major takeaways here are that the use of SOMs alongside DBSCAN and a CNN-LSTM could lead to substantial improvements in load forecasting accuracy as well as provide us with energy profiles that allow us to better understand the habits present on the smaller-scale individual household level and so the proposed method, outlined in chapter 6, will be based on these concepts.

CATEGORY	AUTHOR(S)	YEAR	METHOD(S)
Statistical based	Fumo and Rafe Biswas [20]	2015	Linear regression
	Amber, Aslam, and Hussain [24]	2015	GA & Multiple regression
Machine learning based	Lamedica et al. [25]	1996	SOM & MLP
	Yildiz et al. [5]	2018	SOM, K-means, CART & SVR
Deep learning based	Kong et al. [10]	2019	DBSCAN & LSTM network
	Kim and Cho [22]	2019	CNN-LSTM network

Table 2.1: Outline of related work in the field of energy profile construction and load forecasting.

Part II

FOUNDATION

3 BACKGROUND INFORMATION

This chapter will serve predominantly to explain concepts and methods relevant to the research conducted throughout the duration of this project. Most, if not all, of the information in this chapter might be considered prior knowledge to most readers but regardless may still suffice as a brisk refresher. Feel free to click here if you would prefer to skip this chapter.

3.1 DIMENSION REDUCTION

Depending on how we choose to transform our data set(s) each individual candidate day could be represented by feature vectors of up to 96 temporal dimensions, if not more when taking into consideration supplementary external variables. As a precursory step to our clustering algorithms we can make use of the SOM to project our data onto a feature space that is much lower in terms of the overall dimensionality thus allowing us to achieve visibly clearer clusters in terms of days that express high levels of similarity in their overall energy consumption patterns.

3.1.1 *Self-Organizing Maps*

The neurobiologically inspired Self-Organizing Map (SOM), also known as the Kohonen network, is a topology preserving network proposed by Teuvo Kohonen in 1982 [18]. The SOM belongs to the class of unsupervised learning neural networks and is predominantly used within the realm of feature detection. In contrast to traditional Artificial Neural Networks (ANN), that make use of error backpropagation and gradient descent, the SOM applies a form of competitive learning in which the neurons of the network compete for the right to activate when presented with input data and as a result are forced to organize themselves. Within the realm of this paper, the principal objective of the SOM is to transform a complex, high-dimensional input into a much simpler low-dimensional discrete output space while preserving the overall relationship of the data. Generally the projected output lies in a two-dimensional space where the spatial locations (i. e., coordinates) of the neurons are indicative of inherent statistical features contained within the data [26].

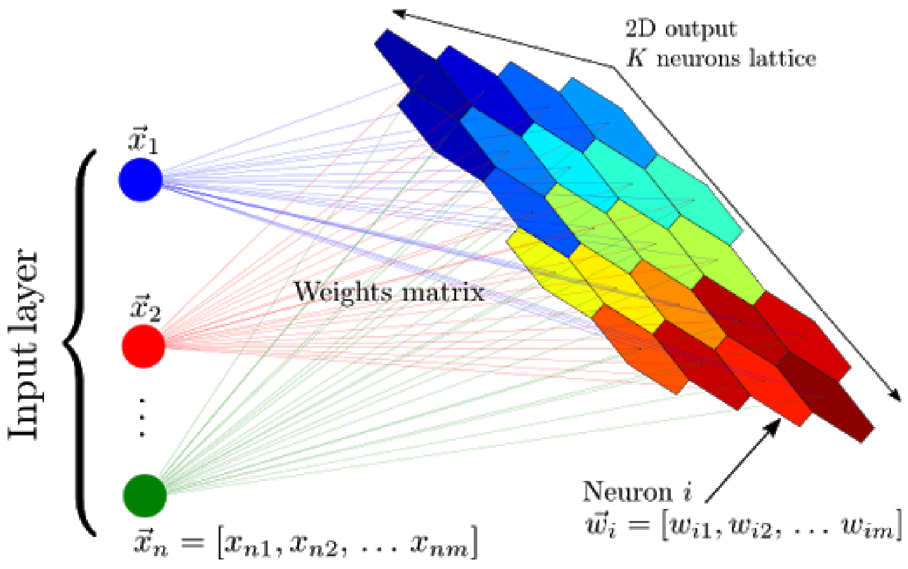

2D output
K neurons lattice
Input layer
$\vec{x}_1$
$\vec{x}_2$
Weights matrix
$\vec{x}_n = [x_{n1}, x_{n2}, \dots x_{nm}]$
Neuron i
$\vec{w}_i = [w_{i1}, w_{i2}, \dots w_{im}]$

Figure 3.1: A schematic representation of a SOM. A set of n feature vectors is mapped onto a two-dimensional lattice that consists of K neurons. The color of the map encodes the organization of groups of said feature vectors with similar properties. Image source: [27] (with permission from the author).

To best understand how the SOM works we can break it down into its core components.

1. *Initialization:* All weights are initialized to small, standardized, random values.

2. *Competition:* A random feature vector from the data set is chosen and presented to the lattice. Neurons compete within themselves by comparing their respective weights to the current input vector which is traditionally done by calculating the Euclidean distance from the weight vector to the input vector. The node with a weight vector closest to the input vector is activated in a winner-takes-all fashion and is marked as the Best Matching Unit (BMU).

3. *Co-operation:* The winning node determines a neighborhood of excited neurons that are adjusted to match the input vector more closely with neurons closer to the BMU being adjusted to a greater extent.

4. *Adaptation:* The response of a winning neuron to subsequent presentations of similar input patterns is enhanced and we can present the next feature vector to the network for evaluation.

3.2 CLUSTERING ALGORITHMS

Throughout the duration of this project we will be making use of the Hierarchical Density Based Spatial Clustering of Applications with Noise (HDBSCAN) clustering algorithm. This section serves to both introduce readers to the DBSCAN algorithm that precedes HDBSCAN as well as provide a high-level, intuitive explanation of both algorithms so that we may better understand the differences between them.

3.2.1 *DBSCAN*

Density Based Spatial Clustering of Applications with Noise (DBSCAN), proposed by Ester et al. [16], is a non-parametric data clustering algorithm that works on the principle of grouping together points that are closely packed together (i. e., located in high-density regions) while marking points that lie alone in low-density regions as outliers. This allows the DBSCAN algorithm to find arbitrarily shaped clusters while also rendering it robust to noise present in the data. Furthermore, the DBSCAN algorithm does not require us to specify, or otherwise know ahead of time, the number of clusters that our data contains and instead automatically determines this number based on the input data as well as the hyperparameters passed to the algorithm on its initialization. This leads us to the very first downside associated with the DBSCAN algorithm and that is that it is *exceptionally* sensitive to hyperparameter selection and thus it is imperative to have a solid grasp on the understanding of said hyperparameters so as to obtain ideal and meaningful results.

The DBSCAN algorithm classifies points in a feature space as either core points, density-reachable points, and outliers. To best understand how this is done we must first define the two hyperparameters that are essential to the initialization of the algorithm. The first, and arguably, most important hyperparameter is labelled ϵ and defines the maximum distance between two points or, in other words, the radius of a neighborhood with respect to a point. The second hyperparameter is aptly titled *minPts* (m_{pts}) and represents the minimum number of points that must be within distance ϵ of a point to define it as a *core* point. If a point does not contain the minimum number of points within its neighborhood to define it as a core point but *is* within distance ϵ from a core point then we consider it a *density-reachable point* and it belongs to the cluster. Any points that cannot be reached from any other point are considered outliers or noise points. In essence, any core point forms a cluster together with all points (core or not) that are within distance ϵ from said core point. Non-core points cannot be used to reach more points and belong to the *"edge"* of the cluster. The pseudocode in Listing 3.1 below depicts an explanation of how this process works.

```
DBSCAN(D, eps, minPts)
    C = 0
    foreach unvisited point P in dataset D
        P = visited
        neighborPts_P = queryNeighborhood(P, eps)
        if(neighborPts < minPts)
            P = noise
        else
            C = C + 1
            expandCluster(P, neighborPts, C, eps, minPts)

expandCluster(P, neighborPts, C, eps, minPts)
    add P to C
    foreach point Q in neighborPts_P
        if Q = unvisited
            Q = visited
            neighborPts_Q = queryNeighborhood(Q, eps)
            if(neighborPts_Q >=  minPts)
                join(neighborPts_Q, neighborPts_P)
        else
            add Q to C

queryNeighborhood(P, eps)
    return all points within distance eps to P
```

Listing 3.1: The DBSCAN algorithm.

To round things off, we present the main advantages and disadvantages of the DBSCAN algorithm:

+ No prior knowledge of the number of clusters is required.
+ Can find arbitrarily shaped clusters.
+ Contains the notion of noise and outliers.
+ Only two hyperparameters need to be set.
– Reliant on the distance metric being used.
– Choosing a meaningful distance threshold can be quite difficult if the data and scale are not well understood.

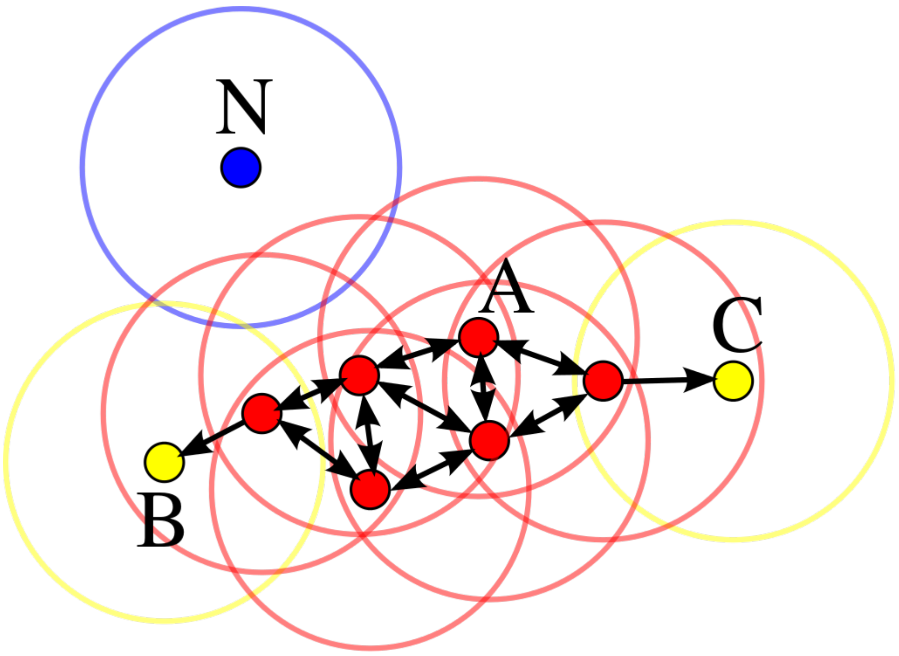
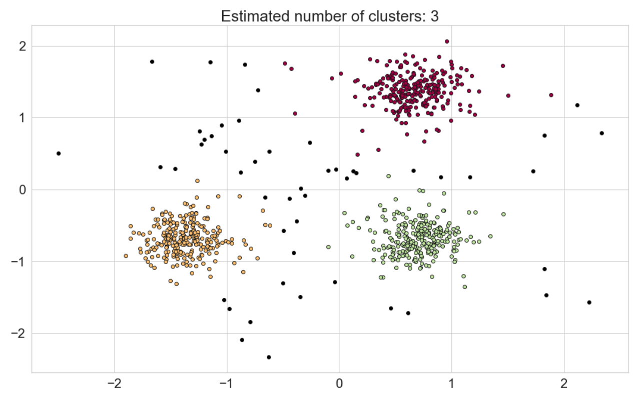


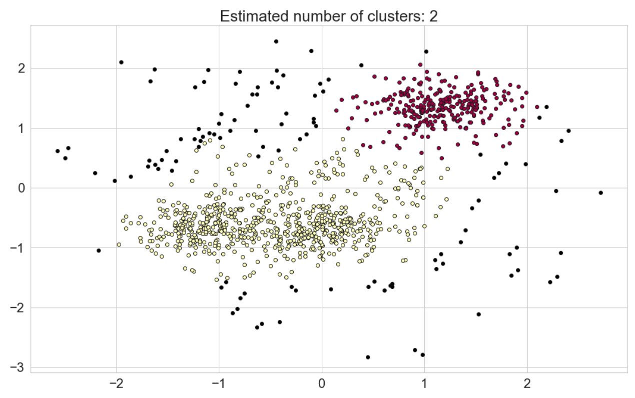
Figure 3.2: Depiction of the DBSCAN algorithm at work with minPts set to 4. Here, point A, as well as the other red points, are core points as the area surrounding these points in a radius ϵ contains at least 4 points (including the point itself). Points B and C are reachable from A through other core points and as a result can be considered density-reachable points. The cluster is made up of the core points as well as points B and C. Point N is neither a core point and cannot be reached through any of the core points and so is considered an outlier. Image source: [28], licensed under CC BY-SA 3.0.

3.2.2 *HDBSCAN*

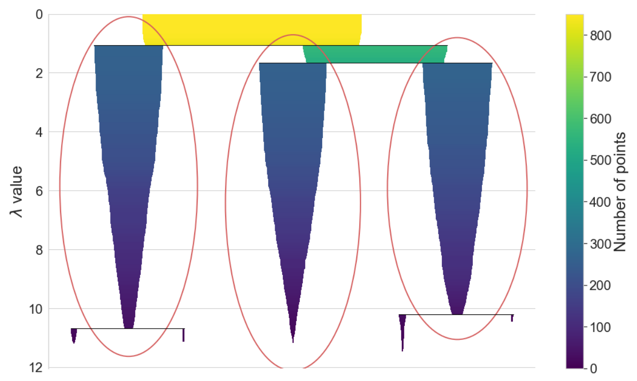
Hierarchical Density Based Spatial Clustering of Applications with Noise (HDBSCAN), proposed by Campello, Moulavi, and Sander [29], is a hierarchical non-parametric clustering algorithm that was designed to overcome the main limitations of DBSCAN. The most substantial changes come in the form of no longer explicitly needing to predefine a value for the distance threshold ϵ. Instead, HDBSCAN generates a complete density-based clustering hierarchy over variable densities from which we can extract a simplified hierarchy composed only of the most significant clusters in our data. Without delving deep into the concepts of cluster stability, minimum spanning trees and hierarchy construction we instead refer the reader to the detailed explanation in the literature [29] and leave off with Figure 3.3 that does well to illustrate how HDBSCAN works as a hierarchical clustering algorithm.



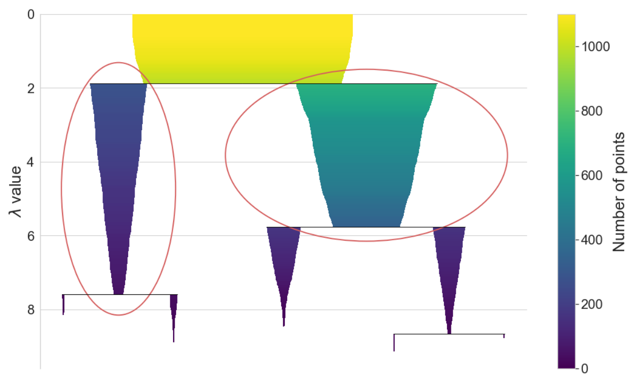
(a) HDBSCAN clustering applied in the case of 3 clusters.



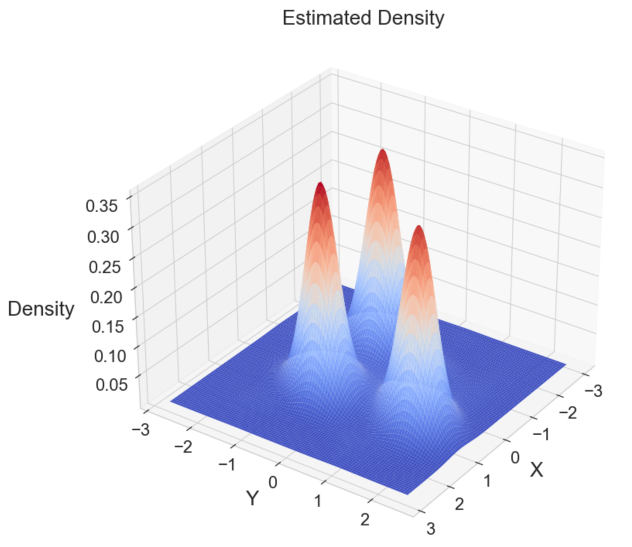
(b) HDBSCAN clustering applied in the case of 2 clusters.



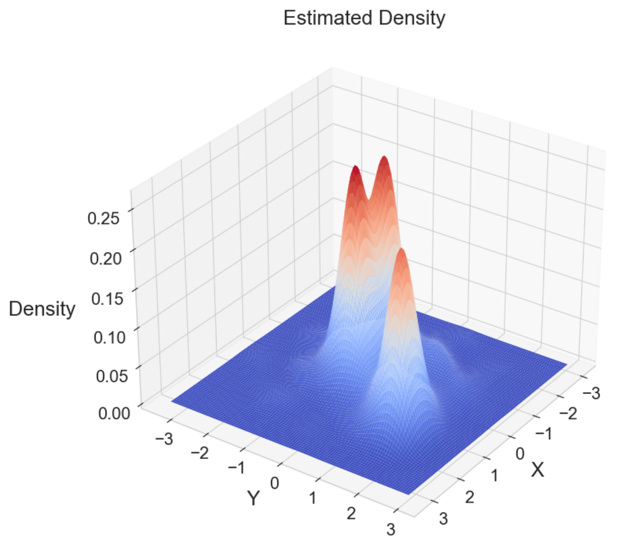
(c) Cluster hierarchy of 3.3a.



(d) Cluster hierarchy of 3.3b.



(e) Density landscape of 3.3a.



(f) Density landscape of 3.3b.

Figure 3.3: An illustration of the hierarchical aspects of the HDBSCAN algorithm. In layman's terms, when presented with the density landscape the HDBSCAN algorithm decides whether peaks of a mountain are part of the same mountain or whether they belong to different mountains where each of these mountains represent a cluster. When multiple peaks represent multiple mountains the sum of their respective volumes tends to be larger than the volume of their base. The opposite is true for when multiple peaks are just features of a singular mountain.

3.3 DISTANCE METRICS

As mentioned in section 3.2.1, the DBSCAN algorithm as well as the HDBSCAN algorithm are heavily reliant on the distance metric being used. For that reason, we explored the possibility of 2 commonly used distance metrics when working with time series data sets: Euclidean distance and Dynamic Time Warping.

3.3.1 *Euclidean Distance*

$$d_{euc}(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{3.1}$$

where:

p, q	$=$ two points in the Euclidean n-space.
p_i, q_i	$=$ Euclidean vectors starting from the origin of the space.
n	$=$ n-space.

3.3.2 *Dynamic Time Warping*

When working with time series data sets, dynamic time warping (DTW) provides us with the means of determining whether two temporal sequences exhibit any measure of similarity. To best understand how DTW works let us assume that we are working with two sequences S and T of lengths n and m respectively. We can arrange the sequences in an $n \times m$ grid where each point (x, y) is the alignment between S[x] and T[y]. Thus, we can calculate the path with minimal distance between elements of both S and T, or what is known as a warping path as follows:

$$D_{min}(S_k, T_k) = \underset{S_{k-1}, T_{k-1}}{\operatorname{argmin}} D_{min}(S_{k-1}, T_{k-1}) + d_{euc}(S_k, T_k | S_{k-1}, T_{k-1}) \tag{3.2}$$

Furthermore, the warping path can only make the following moves:

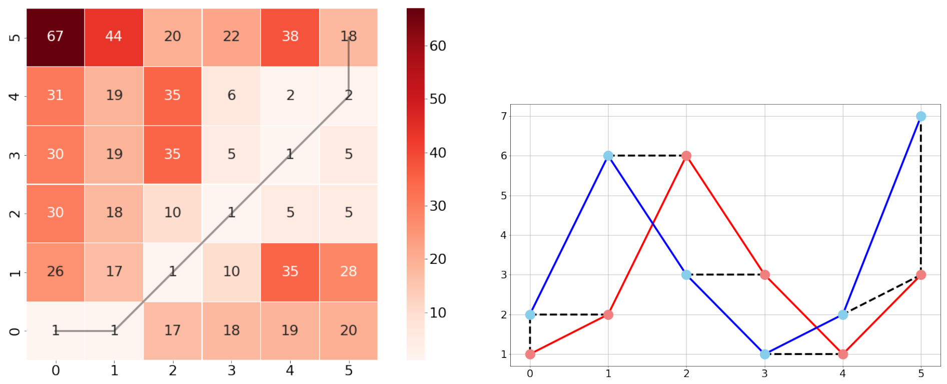
1. Horizontal moves $(x, y) \rightarrow (x, y+1)$ – insertion.
2. Vertical moves: $(x, y) \rightarrow (x+1, y)$ – deletion.
3. Diagonal moves: $(x, y) \rightarrow (x+1, y+1)$ – match.

Some final notes on the implementation of DTW:

- Each index from the first sequence must be matched with one (or more) indices from the second sequence and vice versa.

- The first and last index from the first sequence must be matched with the first and last index of the second sequence respectively.
- The mapping of indices from the first sequence to indices of the second sequence must be monotonically increasing and vice versa such that $S_{t-1} \leq S_t$ and $T_{t-1} \leq T_t$.

Figure 3.4 denotes an example of how we can calculate the DTW path of 2 sequences of length 6. The first sequence contains the values [1, 2, 6, 3, 1, 3] and the second sequence contains the values [2, 6, 3, 1, 2, 7].



(a) Accumulated cost matrix and warping path.

(b) DTW distance between our 2 sequences.

Figure 3.4: An illustration depicting how we can use DTW to calculate the distance between 2 sequences.

3.4 FORECASTING MODELS

The primary forecasting model that we will be utilizing in our forecasting pipeline is a hybrid CNN-LSTM network. This section serves to introduce readers to both the Convolutional Neural Network (CNN) component as well as the LSTM component of this network.

3.4.1 *Convolutional Neural Networks*

Convolutional Neural Networks (CNNs) first truly started gaining traction in the 1990s when Lecun et al. [30] demonstrated that a CNN which aggregates simple features into progressively more complex features can be successfully used for the task of recognizing handwritten characters. Since then, their relevance has become more and more widespread with applications in image and video classification, natural language processing [31], and when working with time series data sets [32]. The following sections will briefly outline key points of the inner machinations of key CNN components.

3.4.1.1 *Convolutional Layers*

Firstly, and most importantly, CNNs derive their name from the so-called "convolution" operator whose primary function is to extract features from the input vector while maintaining the spatial relationship between the features in said input vector. Put simply, the convolutional layer works by sliding a pre-determined number of filters, otherwise known as 'kernels', a pre-determined 'distance' or stride over an input vector and returning a feature map per filter. The value of said filters are, in practice, learned by the network during the training process while other hyperparameters, such as the number of filters as well as their respective sizes are pre-determined by the network architect. Other things to keep note of are that the resulting feature maps are reduced in dimensionality when compared to the input; however this can be offset by utilizing a variation of techniques such as the application of a form of *padding*.

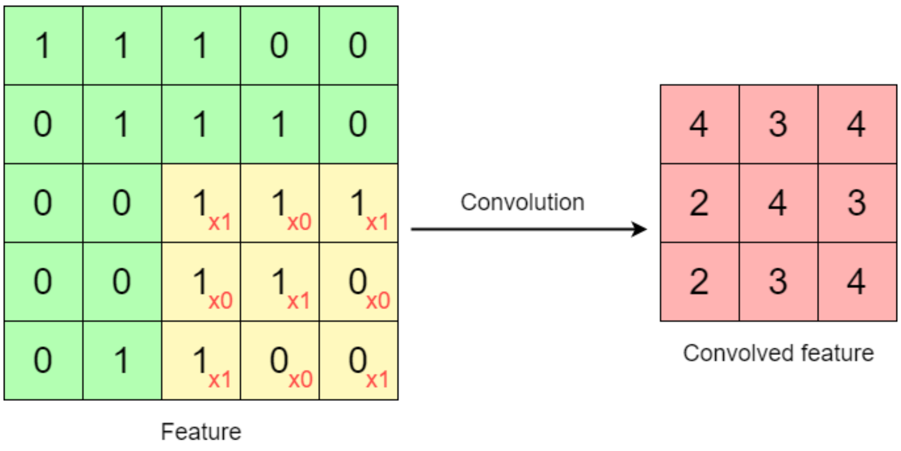


Figure 3.5: An example of a convolutional kernel at work. A 3x3 kernel traverses over a 5x5 "image" with a stride of 1 to produce the convolved feature map.

3.4.1.2 *Rectified Linear Unit Operation*

Following every convolution operation is a Rectified Linear Unit (ReLU) operation where ReLU is a non-linear operation whose output is given by:

$$R(z) = \max(0, z) \tag{3.3}$$

The purpose of the ReLU operation is to replace all negative values in the feature map by zero. This nonlinear function allows for the use of stochastic gradient descent with backpropagation of errors that enables us to learn complex relationships within the data. Other operations, or activation functions, such as *tanh* or *sigmoid* can also be used here but generally ReLU performs much better in most situations and is much quicker to perform due to its sheer simplicity.

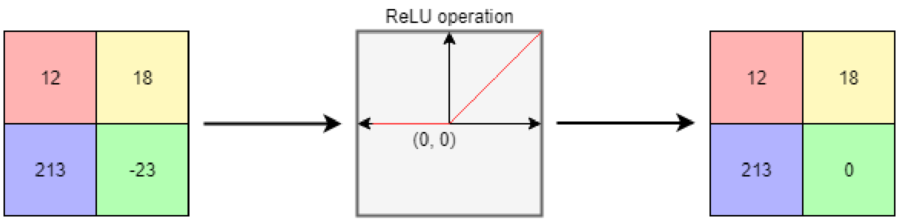


Figure 3.6: A simplified demonstration of a ReLU operation.

3.4.1.3 *Max Pooling Layers*

Inter-mingled between convolutional layers are a set of pooling layers that serve to reduce the dimensionality of each feature map while retaining the most important information. In the case of *max* pooling layers, the network defines a spatial neighborhood and takes the largest element from the rectified feature map within that window. The goal of pooling layers then is to reduce the feature dimensions of our input vectors thus making them smaller and more manageable to work with while also reducing the number of parameters and computations needed to fit our network, thus minimizing the risk of overfitting. Furthermore, this renders the network invariant to small transformations, distortions and translations in the input vector by providing us with what is essentially a scale invariant representation of our vector.

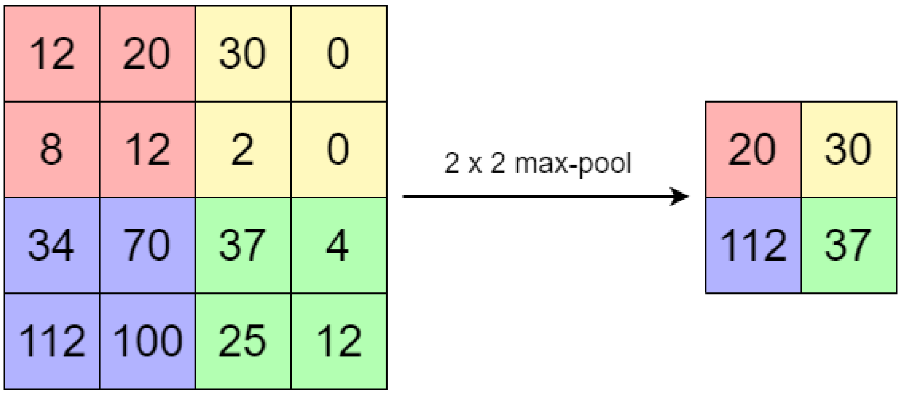


Figure 3.7: An example of max pooling using a 2x2 window.

3.4.2 *Long Short-Term Memory Networks*

Long Short-Term Memory (LSTM) networks, first proposed by Hochreiter & Schmidhuber in 1997 [33], are a special kind of RNN network that are capable of learning long-term dependencies while overcoming the main limitations that plagued traditional RNN networks (such as the exploding/vanishing gradient problem). The cell state of an LSTM can be seen as a highway that transfers relative information all the way down the sequence chain allowing information throughout the processing of the sequence to be retained giving the network a form of *"memory"*. The key to the functionality of the LSTM is through the use of a number of gates that give it the ability to remove or add information to this cell state by learning what information is relevant to keep, or otherwise forget, during training. The following sections will serve to outline the functionality of the cell state and gates so that we may gain a better understanding of them.

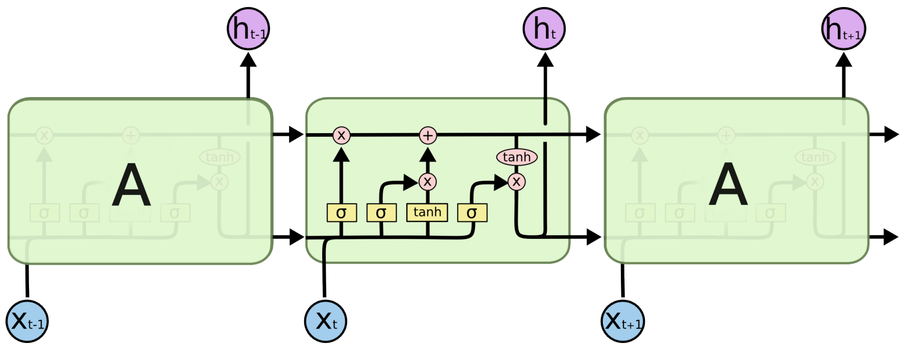


Figure 3.8: The repeating module in an LSTM that contains four interacting layers. Image source: [34].

3.4.2.1 *Forget Gate*

The first gate that we will be taking a look at is the forget gate (f_t). This gate decides what information should be thrown away and what information should be kept from prior steps. Information from the previous hidden state (h_{t-1}) and information from the current input (x_t) are passed through a *sigmoid* (σ) function where values come out between 0 and 1 for each number in the cell state (C_{t-1}). Values closer to and including 0 indicate that we should completely forget this information while values closer to and including 1 indicate that we should completely retain all of this information. The formulation of the forget gate can be seen in equation 3.4.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.4}$$

3.4.2.2 *Input Gate*

The input gate (i_t) mainly serves to decide what new information will be stored in the cell state from the current step. The input gate is a *sigmoid* (σ) function that is passed the previous hidden state (h_{t-1})

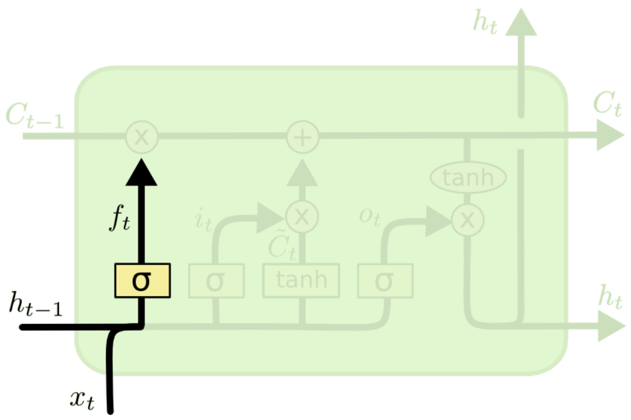


Figure 3.9: An illustration of the forget gate in an LSTM network. Image source: [34].

and the current input (x_t) and outputs values between 0 and 1 where values closer to and including 0 indicate that the information is not important while values closer to and including 1 indicate that the information is important. This value is multiplied by a *tanh* layer that serves the purpose of creating a vector of new candidate values $(\tilde{C}_t)$ that could potentially be added to the cell state. The formulation of the input gate and its respective layers can be seen in equations 3.5.

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \end{aligned} \tag{3.5}$$

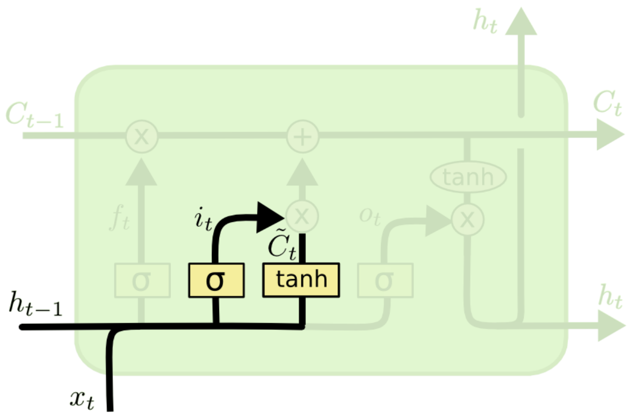


Figure 3.10: An illustration of the input gate in an LSTM network. Image source: [34].

3.4.2.3 *Output Gate*

The final gate is the output gate (o_t) which decides what the next hidden state (h_t) should be. As like in the previous gates, we pass the previous hidden state (h_{t-1}) and the current input (x_t) into a *sigmoid*

(σ) function which is multiplied by the output of the *tanh* function applied to the modified cell state (C_t) which finally gives us our new hidden state. The new hidden state as well as the new cell state are carried over to the next time step. The formulation of the output gate and its respective layers can be seen in equations 3.6.

$$\begin{aligned} C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\ o_t &= \sigma(W_o\,[h_{t-1}, x_t] + b_o) \\ h_t &= o_t * tanh(C_t) \end{aligned} \tag{3.6}$$

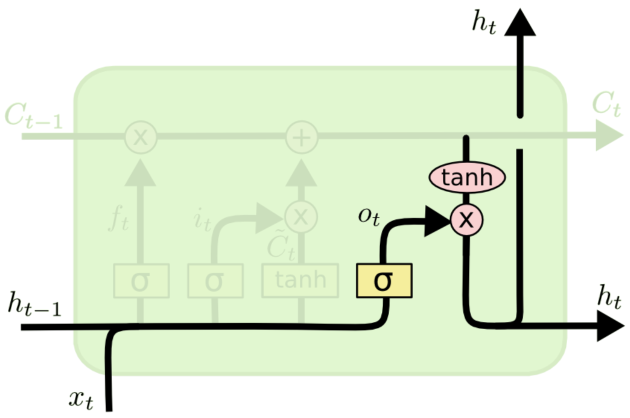


Figure 3.11: An illustration of the output gate in an LSTM network. Image source: [34].

3.5 PERFORMANCE METRICS

Throughout the duration of this project we will be making use of a variety of performance metrics. These performance metrics, as well as the reasoning behind choosing them, will be explained in the following sections.

3.5.1 *Mean Absolute Error*

The first performance metric that we will be taking a look at is the mean absolute error (MAE). It provides us with a direct interpretation of how far off the predictions made by our forecasting models were from the actual, ground truth. However, the MAE does not provide us with the capability of drawing comparisons between results obtained from disparate data sets as it is a scale-dependent metric. That said it provides a satisfactory level of insight nonetheless. Its equation is:

$$\text{MAE} = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{n} \tag{3.7}$$

where:

$\hat{y}_i$ = predicted value.

y_i = actual value.

n = total number of data points.

3.5.2 *Root Mean Squared Error*

The second performance metric that we will be working with is the root mean square error (RMSE). Its primary purpose is to serve as a cost function that our forecasting models will seek to minimize. The primary reason behind choosing RMSE as our cost function is that it penalizes larger errors. Its equation is:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}} \tag{3.8}$$

where:

$\hat{y}_i$ = predicted value.

y_i = actual value.

n = total number of data points.

3.5.3 *Mean Absolute Percentage Error*

The final performance metric we will be taking a look at is the mean absolute percentage error (MAPE). As it is a scale-invariant metric, its primary purpose is to allow us to assess the performance of our forecasting models across the multiple, disparate data sets we have on hand and draw comparisons between them. Its equation is:

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{\hat{y}_i - y_i}{y_i} \right| \tag{3.9}$$

where:

$\hat{y}_i$ = predicted value.
y_i = actual value.
n = total number of data points.

4 DATA DESCRIPTION

At our disposal are a number of publicly available data sets that contain historical data with regards to energy consumption. These include the data collected by the Engineering and Physical Sciences Research Council (EPSRC) via the project entitled "*Personalised Retrofit Decision Support Tools for UK Homes using Smart Home Technology (REFIT)*" [35] which is a collaboration among the Universities of Strathclyde, Loughborough and East Anglia and the *"Individual Household Electric Power Consumption"* data set [36] that is part of the University of California, Irvine (UCI) Machine Learning Repository and that will henceforth be acronymized as the *"UCI data set (UCID)"*. This section will serve to briefly describe the main aspects of each of these individual data sets so that we may be better able to draw comparisons between them and highlight any key differences. Further in-depth analysis of each subsequent data set can be found in section 5 of this paper. Additionally, we aim to append meteorological features (e. g., temperature, wind speed, cloud coverage, precipitation) to each of our respective data sets – an overview of this process and the data that we will be utilizing will also be presented in this section.

4.1 REFIT

The REFIT Electrical Load Measurements data set includes cleaned electrical consumption data, in watts, for a total of 20 households labelled *House 1 - House 21* (skipping House 14) located in the Loughborough area, a town in England, over the period of 2013 through early 2015. The electrical consumption data is collected at both the aggregate level as well as the appliance level with each household containing a total of 10 power sensors that comprise of a current clamp for the household aggregate labelled as *Aggregate* in the data set as well as 9 individual appliance monitors (IAM) labelled as *Appliance 1 - Appliance 9* in the data set. The appliance list associated with each of the IAMs differs between households and comprise a measure of ambiguity as applicants may have switched appliances around during the duration of the data collection and the installation team responsible for setting up the power sensors did not always collect relevant data associated with said IAMs. The consequences of this is of course that we do not know with 100% certainty whether an appliance or set of appliances associated with an IAM is the same throughout the entirety of the data set. Additionally, some labels are inherently ambiguous taking, for example, the *television site* label which could comprise of any number of appliances including: a television, DvD player, computer, speakers etc. Finally, the models and makes of the appliances that were meant to be collected by the installation team are not always present further compounding on the previously mentioned uncertainties.

The documentation associated with the data set states that active power is collected, and subsequently recorded, at an interval of 8 seconds; however, a cursory glance at the data demonstrates that this is not always the case. A potential reason for this could be the fact that the aforementioned power sensors are not synchronised with the associated collection script which polls within a range of 6 to 8 seconds leaving us with a margin for error in the intervals between recorded data samples. Moreover, the data set is riddled with long periods of missing data making it exceptionally difficult to work with. All of that said, the data collection team made an attempt to pre-process or otherwise *clean* the data set by:

1. Correcting the time to account for the United Kingdom (UK) daylight savings.
2. Merging timestamp duplicates.
3. Moving sections of IAM columns to correctly match the appliance they were recording when said appliance was reset or otherwise moved.
4. Forward filling not a number (NaN) values or zeroing them depending on the duration of the time gap.
5. Removing spikes of greater than 4,000 watts from the IAM values and replacing them with 0s.
6. Appending an additional issues columns that is set to 1 if the sum of the sub-metering IAMs is greater than that of the household aggregate – in this case, data should either be discarded or, at the very least, the discrepancy must be noted.

4.2 UCID

The UCID data set contains a total of 2,075,259 measurements gathered in a single house located in Sceaux, a commune in the southern suburbs of Paris, France. The data within this data set was recorded throughout a duration of 47 months spanning the period between December 2006 and November 2010. Measurements were made approximately once a minute and consist of the minute-averaged active power consumption, in kilowatts, within the entire household as well as 3 energy sub-metering measurements that correspond to the kitchen, which includes a dishwasher and microwave, the laundry room that consists of a washing machine and tumble dryer, and the combination of both an electric water-heater as well as an air-conditioner respectively. The UCID data set is not without fault either containing approximately 25,979 missing measurements which make up roughly 1.25% of the entire data set; however, given the extensive range covered as well as the immense number of total measurements available on hand these missing values can easily be disregarded and subsequently discarded during the preprocessing stage of our forecasting pipeline.

4.3 METEOROLOGICAL DATA

As an addendum to both the REFIT and UCID data sets we will be incorporating meteorological data as provided by Solcast [37], a company based in Australia that aims to provide high quality and easily-accessible solar data. This service is not provided free of charge; however, public researchers and students are allotted a generous amount of credit to work with and, per request, are entitled to received additional credit as needed. For the purpose of this master's thesis project we will be requesting meteorological data in variable time resolutions (5, 10, 15 minutes) for both the Loughborough area in the UK for the REFIT data set as well as meteorological data for the Sceaux commune in the southern suburbs of Paris, France for the UCID data set. The relevant periods are the 16th of September, 2013 up to and including the 11th of July, 2015 and the 1st of December, 2006 up to and including the 30th of November, 2010 for each data set respectively. The provided data is extensive, covering a wide range of parameters that are listed, and described in detail, in Table 4.1.

PARAMETER	DESCRIPTION
GHI	The total irradiance received on a horizontal surface. It is the sum of the horizontal components of direct (beam) and diffuse irradiance. Units in W/m2.
EBH	The horizontal component of Direct Normal Irradiance (DNI). Units in W/m2.
DNI	Solar irradiance arriving in a direct line from the sun as measured on a surface held perpendicular to the sun. Units in W/m2.
Zenith	The angle between a line perpendicular to the earth's surface and the sun (90 deg = sunrise and sunset; 0 deg = sun directly overhead). Units in degrees.
Azimuth	The angle between a line pointing due north to the sun's current position in the sky. Negative to the East. Positive to the West. 0 at due North. Units in degrees.
Cloud Opacity	The measurement of how opaque the clouds are to solar radiation in the given location. Units in percentage.
Air Temperature	The air temperature (2 meters above ground level). Units in Celsius.
Dewpoint	The air dewpoint temperature (2 meters above ground level). Units in Celsius.
Relative Humidity	The air relative humidity (2 meters above ground level). Units in percentage.
SFC pressure	The air pressure at ground level. Units in hPa.
Wind Speed	The wind speed (10 meters above ground level). Units in m/s.
Wind Direction	The wind direction (10 meters above ground level). This is the meteorological convention. 0 is a northerly (from the north); 90 is an easterly (from the east); 180 is a southerly (from the south); 270 is a westerly (from the west). Units in degrees.
Preciptable Water	The total column preciptable water content. Units in kg/m2.
Snow Depth	The snow depth liquid-water-equivalent. Units in cm.
GTI Horizontal Single-Axis Tracker	The total irradiance received on a sun-tracking surface. Units in W/m2.
GTI Fixed	The total irradiance received on a surface with a fixed tilt. The tilt is set to latitude of the location. Units in W/m2.
Albedo	Average daytime surface reflectivity of visible light, expressed as a value between 0 and 1. 0 represents complete absorption. 1 represents complete reflection.

Table 4.1: List of meteorological parameters available to us as per the Solcast data sets.

5 EXPLORATORY DATA ANALYSIS

5.1 REFIT

5.1.1 *Missing & Incomplete Data*

HOUSE NO.	DAYS RECORDED	DAYS MISSING	INCOMPLETE DAYS	STRETCH
1	640	61 (9.53%)	57 (8.91%)	40 days
2	619	128 (20.68%)	58 (9.37%)	61 days
3	616	54 (8.77%)	47(7.63%)	40 days
4	635	41 (6.46%)	79 (12.44%)	13 days
5	649	21 (3.24%)	76 (11.71%)	8 days
6	578	69 (11.94%)	52 (9.0%)	32 days
7	615	61 (9.92%)	51 (8.29%)	40 days
8	556	43 (7.73%)	43 (7.73%)	38 days
9	569	74 (13.01%)	35 (6.15%)	40 days
10	588	22 (3.74%)	79 (13.44%)	8 days
11	393	31 (7.89%)	33 (8.4%)	9 days
12	489	20 (4.09%)	37 (7.57%)	8 days
13	500	89 (17.8%)	79 (15.8%)	40 days
15	569	38 (6.68%)	69 (12.13%)	8 days
16	545	52 (9.54%)	70 (12.84%)	17 days
17	471	19 (4.03%)	37 (7.86%)	8 days
18	444	15 (3.38%)	34 (7.66%)	8 days
19	472	19 (4.03%)	33 (6.99%)	8 days
20	461	19 (4.12%)	27 (5.86%)	8 days
21	491	33 (6.72%)	45 (9.16%)	14 days

Table 5.1: Number of days with missing data in the REFIT data set as well as the longest period of consecutive days with missing data.

HOUSE NO.	VALUES RECORDED	VALUES WITH ISSUES
1	6,960,008	58,183 (0.84%)
2	5,733,526	28,444 (0.5%)
3	6,994,594	408,627 (5.84%)
4	6,760,511	67,441 (1.0%)
5	7,430,755	425,766 (5.73%)
6	6,241,971	34,451 (0.55%)
7	6,756,034	161,919 (2.4%)
8	6,118,469	25,000 (0.41%)
9	6,169,525	32,226 (0.52%)
10	6,739,284	30,162 (0.45%)
11	4,431,541	40,114 (0.91%)
12	5,859,544	14,183 (0.24%)
13	4,737,371	123,796 (2.61%)
15	6,225,696	23,349 (0.38%)
16	5,722,544	14,713 (0.26%)
17	5,431,577	85,937 (1.58%)
18	5,007,721	174,490 (3.48%)
19	5,622,610	62,636 (1.11%)
20	5,168,605	19,594 (0.38%)
21	5,383,993	206,832 (3.84%)

Table 5.2: Number of values recorded that have issues in the REFIT data set.

5.1.2 *Data Visualization*

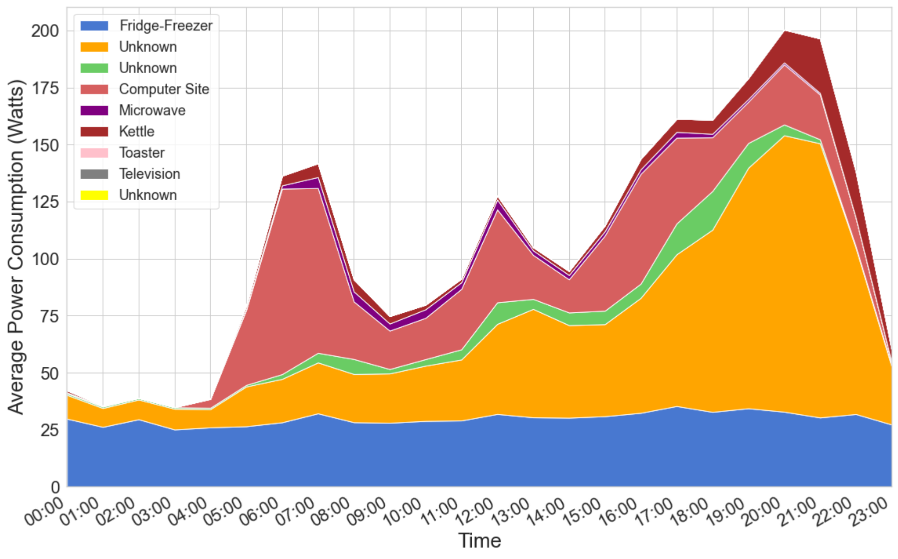


Figure 5.1: A sample stacked area chart showing the readings of each appliance in each hour of a day. These readings were averaged over the entirety of the data present in the data set. Data for this plot was pulled from CLEAN_House12.csv of the REFIT data set.

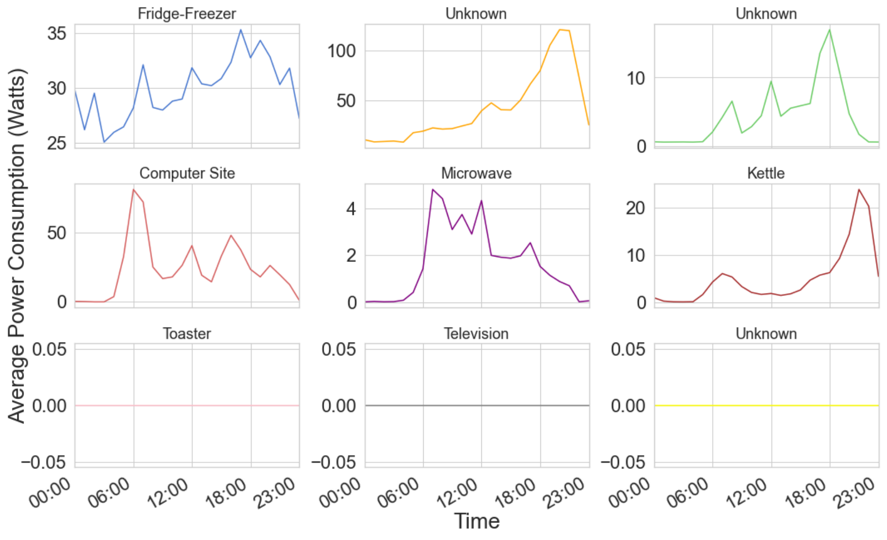


Figure 5.2: In-depth look at the active hours of individual appliances present in Figure 5.1.

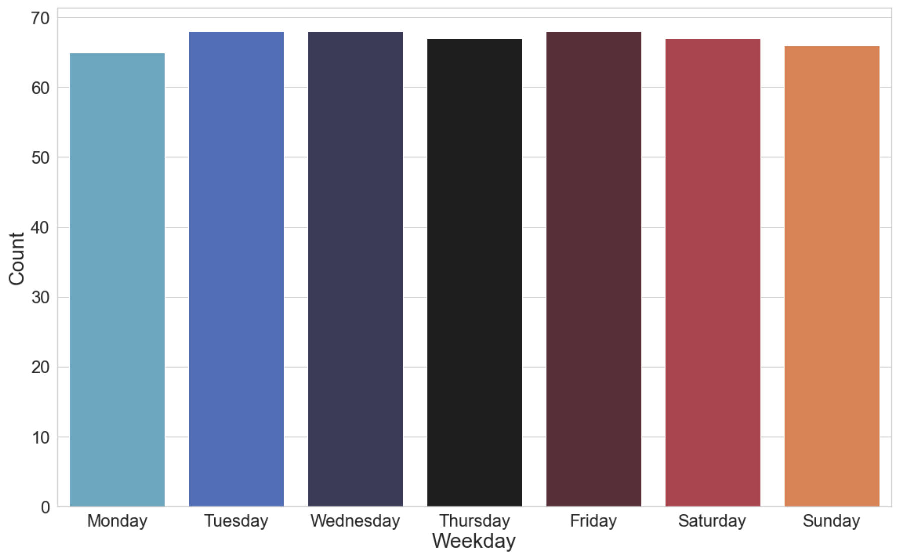


Figure 5.3: Number of samples per day of the week over the entirety of the data set. Data for this plot was pulled from CLEAN_House12.csv of the REFIT data set.

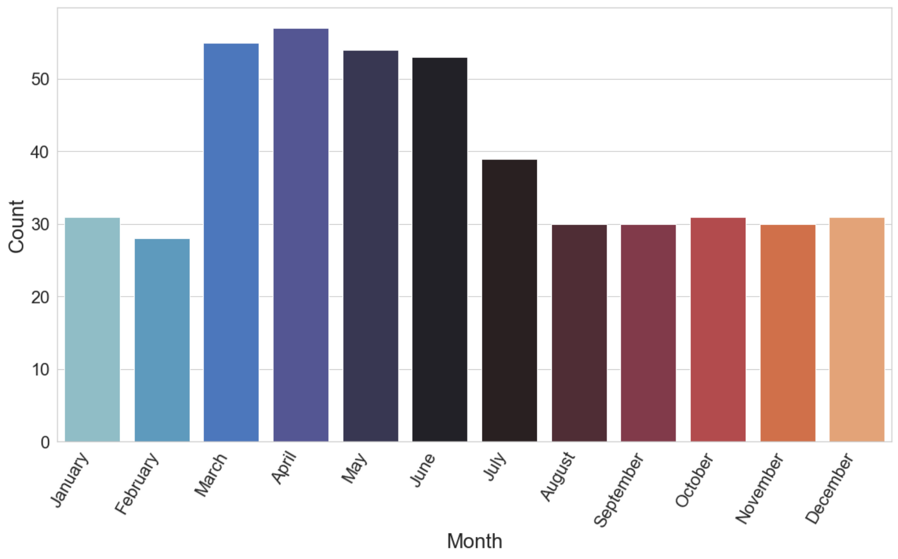


Figure 5.4: Number of samples per month over the entirety of the data set. Data for this plot was pulled from CLEAN_House12.csv of the REFIT data set.

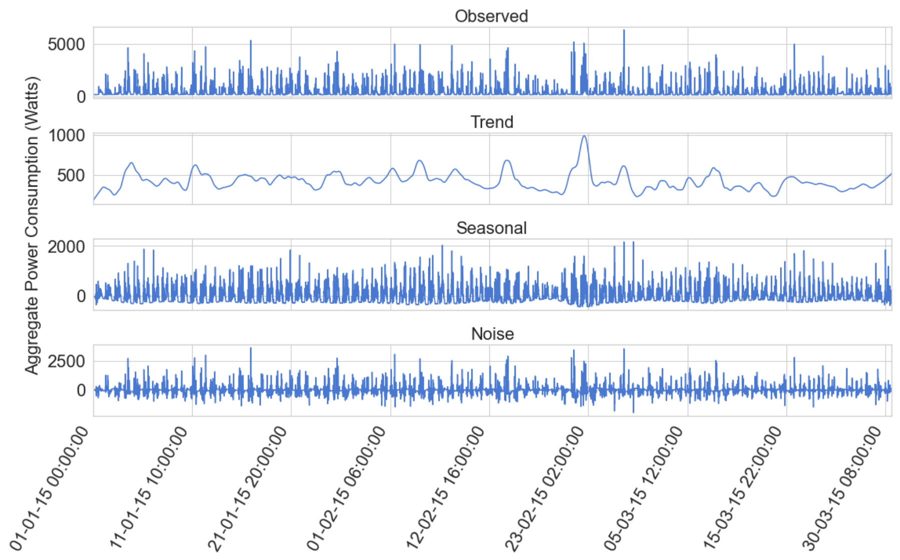


Figure 5.5: Time series decomposition. Data for these plots were pulled over a 3 month period that was resampled into a resolution of 15 minutes from CLEAN_House12.csv of the REFIT data set.

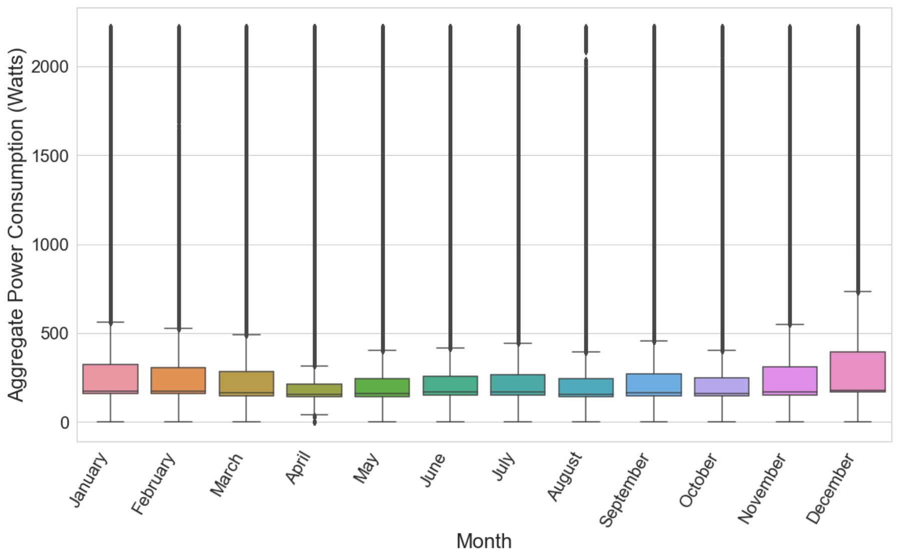


Figure 5.6: Box and whiskers plot over the aggregate power consumption (in watts) on a per-month basis *after* removing outliers that were 3 standard deviations away from the mean. Data for this plot was pulled from CLEAN_House12.csv of the REFIT data set.

5.1.3 *Causality & Correlation*

5.2 UCID

Part III

EMPIRICAL STUDY

6 METHODOLOGY

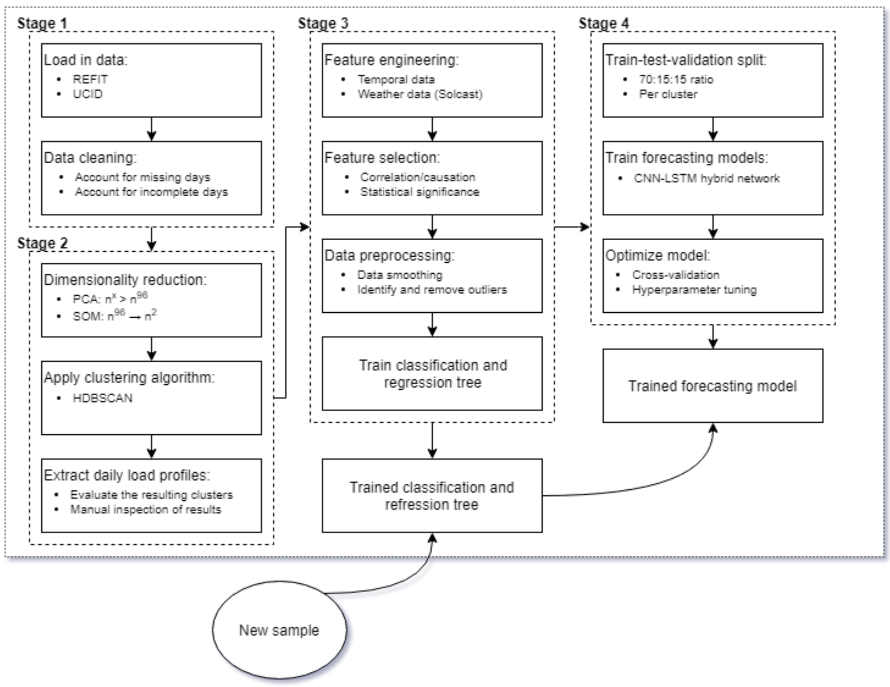


Figure 6.1: Proposed daily profile extraction and load forecasting model.

7 RESULTS AND DISCUSSION

8 CONCLUSION AND FUTURE WORK

Part IV

APPENDIX

LIST OF FIGURES

LIST OF TABLES

LISTINGS

GLOSSARY

ANN	Artificial Neural Network. 11
BEMS	Battery Energy Management System. 2, 5
BMU	Best Matching Unit. 12
CART	Classification and Regression Tree. 7, 9
CCF	Cluster-Classify-Forecast. 7
CILF	Computationally intelligent load forecasting. 5
CNN	Convolutional Neural Network. 18, 19
CNN-LSTM	Convolutional Neural Network Long Short-Term Memory. 7–9, 18
DBSCAN	Density Based Spatial Clustering of Applications with Noise. vi, 6, 8, 9, 13–15, 17, 40, 42
DNI	Direct Normal Irradiance. 29
DTW	Dynamic time warping. 17, 18, 41
EBH	Direct (Beam) Horizontal Irradiance. 29
EPSRC	Engineering and Physical Sciences Research Council. 26
GA	Genetic algorithm. 8, 9
GHI	Global Horizontal Irradiance. 29
GTI	Global Torizontal Irradiance. 29
HDBSCAN	Hierarchical Density Based Spatial Clustering of Applications with Noise. 13, 15–17, 40
HEMS	Home Energy Management System. 2, 3, 5, 40
IAM	Individual appliance monitor. 26, 27
LSTM	Long Short-Term Memory. 7–9, 18, 21–23, 41
MAE	Mean absolute error. 24

MAPE	Mean absolute percentage error. 7, 8
MLP	Multi-Layer Perceptron. 7, 9
NAN	Not a number. 27
NMAE	Normalized mean absolute error. 8
NRMSE	Normalized root mean square error. 8
REFIT	Personalised Retrofit Decision Support Tools for UK Homes using Smart Home Technology. 26, 28, 30–34, 41, 42
RELU	Rectified Linear Unit. 19, 20, 41
RMSE	Root mean square error. 24
RNN	Recurrent Neural Network. 7, 20, 21
SMBM	Smart Meter Based Model. 7
SOM	Self-Organizing Map. 7–9, 11, 12, 40
SVM	Support Vector Machine. 7
SVR	Support Vector Regression. 7, 9
UCI	University of California, Irvine. 26
UCID	UCI data set. 26–28
UK	United Kingdom. 27, 28

BIBLIOGRAPHY

[1] *Household Appliances - Worldwide: Statista Market Forecast*. URL: https://www.statista.com/outlook/256/100/household-appliances/worldwide.

[2] *Energy Efficiency in Buildings*. URL: https://www.wbcsd.org/programs/cities-and-mobility/energy-efficiency-in-buildings.

[3] Yixuan Wei et al. "A Review of Data-Driven Approaches for Prediction and Classification of Building Energy Consumption." In: *Renewable and Sustainable Energy Reviews* 82 (2018), pp. 1027–1047. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2017.09.108. URL: https://www.sciencedirect.com/science/article/pii/s136403211731362x.

[4] Chao Chen, Barnan Das, and Diane Cook. "Energy Prediction in Smart Environments." In: Jan. 2010, pp. 148–157. DOI: 10.3233/978-1-60750-638-6-148.

[5] Baran Yildiz et al. "Household Electricity Load Forecasting Using Historical Smart Meter Data with Clustering and Classification Techniques." In: *2018 IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia)* (2018). DOI: 10.1109/ISGT-ASIA.2018.8467837.

[6] Yu-Hsiang Hsiao. "Household Electricity Demand Forecast Based on Context Information and User Daily Schedule Analysis From Meter Data." In: *IEEE Transactions on Industrial Informatics* 11.1 (2015), pp. 33–43. DOI: 10.1109/TII.2014.2363584.

[7] Kareem Al-Saudi. *The Effectiveness of Different Forecasting Models on Multiple Disparate Datasets*.

[8] Muhammad Qamar Raza and Abbas Khosravi. "A Review on Artificial Intelligence Based Load Demand Forecasting Techniques for Smart Grid and Buildings." In: *Renewable and Sustainable Energy Reviews* 50 (2015), pp. 1352–1372. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2015.04.065. URL: https://www.sciencedirect.com/science/article/pii/s1364032115003354.

[9] Aurélie Foucquier et al. "State of the Art in Building Modelling and Energy Performances Prediction: A Review." In: *Renewable and Sustainable Energy Reviews* 23 (2013), pp. 272–288. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2013.03.004. URL: https://www.sciencedirect.com/science/article/pii/s1364032113001536.

[10] W. Kong et al. “Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network.” In: *IEEE Transactions on Smart Grid* 10.1 (2019), pp. 841–851. DOI: 10.1109/TSG.2017.2753802.

[11] Hong-Tzer Yang, Jian-Tang Liao, and Che-I Lin. “A Load Forecasting Method for HEMS Applications.” In: *2013 IEEE Grenoble Conference* (2013). DOI: 10.1109/PTC.2013.6652195.

[12] Seyedeh Narjes Fallah et al. “Computational Intelligence Approaches for Energy Load Forecasting in Smart Energy Management Grids: State of the Art, Future Challenges, and Research Directions.” In: *Energies* 11.3 (2018). ISSN: 1996-1073. URL: https://www.mdpi.com/1996-1073/11/3/596.

[13] Eric Backer and Anil K. Jain. “A Clustering Performance Measure Based on Fuzzy Set Decomposition.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-3.1 (1981), pp. 66–75. DOI: 10.1109/TPAMI.1981.4767051.

[14] B. Stephen et al. “Incorporating Practice Theory in Sub-Profile Models for Short Term Aggregated Residential Load Forecasting.” In: *IEEE Transactions on Smart Grid* 8.4 (2017), pp. 1591–1598. DOI: 10.1109/TSG.2015.2493205.

[15] Elizabeth Shove, Mika Pantzar, and Matt Watson. *The Dynamics of Social Practice: Everyday Life and How It Changes*. Sage, 2012.

[16] Martin Ester et al. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.” In: KDD’96. Portland, Oregon: AAAI Press, 1996.

[17] B. Yildiz et al. “Recent Advances in the Analysis of Residential Electricity Consumption and Applications of Smart Meter Data.” In: *Applied Energy* 208 (2017), pp. 402–427. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2017.10.014.

[18] Teuvo Kohonen. “The Self-Organizing Map.” In: *Proceedings of the IEEE* 78.9 (1990), pp. 1464–1480.

[19] Gareth James et al. *An Introduction to Statistical Learning with Applications in R*. Springer, 2017.

[20] Nelson Fumo and M.A. Rafe Biswas. “Regression Analysis for Prediction of Residential Energy Consumption.” In: *Renewable and Sustainable Energy Reviews* 47 (2015), pp. 332–343. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2015.03.035. URL: https://www.sciencedirect.com/science/article/pii/s1364032115001884.

[21] Heng Shi, Minghao Xu, and Ran Li. “Deep Learning for Household Load Forecasting – A Novel Pooling Deep RNN.” English. In: *IEEE Transactions on Smart Grids* 9.5 (Sept. 2018), pp. 5271–5280. ISSN: 1949-3053. DOI: 10.1109/TSG.2017.2686012.

[22] Tae-Young Kim and Sung-Bae Cho. "Predicting Residential Energy Consumption Using CNN-LSTM Neural Networks." In: *Energy* 182 (2019), pp. 72–81. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2019.05.230.

[23] Tom O'Malley et al. *Keras Tuner*. https://github.com/keras-team/keras-tuner. 2019.

[24] K.P. Amber, M.W. Aslam, and S.K. Hussain. "Electricity Consumption Forecasting Models for Administration Buildings of the UK Higher Education Sector." In: *Energy and Buildings* 90 (2015), pp. 127–136. ISSN: 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2015.01.008.

[25] R. Lamedica et al. "A Neural Network Based Technique for Short-Term Forecasting of Anomalous Load Periods." In: *IEEE Transactions on Power Systems* 11.4 (1996), pp. 1749–1756. DOI: 10.1109/59.544638.

[26] Umut Asan and Secil Ercan. "An Introduction to Self-Organizing Maps." In: Jan. 2012, pp. 299–319. DOI: 10.2991/978-94-91216-77-0_14.

[27] Matias Carrasco Kind and Robert J. Brunner. "SOMz: photometric redshift PDFs with self-organizing maps and random atlas." In: *Monthly Notices of the Royal Astronomical Society* 438.4 (2014), pp. 3409–3421. DOI: 10.1093/MNRAS/STT2456.

[28] *DBSCAN*. Feb. 2021. URL: https://en.wikipedia.org/wiki/DBSCAN.

[29] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. "Density-Based Clustering Based on Hierarchical Density Estimates." In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Jian Pei et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 160–172. ISBN: 978-3-642-37456-2.

[30] Yann Lecun et al. "Gradient-Based Learning Applied to Document Recognition." In: *Proceedings of the IEEE* 86 (Dec. 1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[31] Ronan Collobert and Jason Weston. "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning." In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: Association for Computing Machinery, 2008, pp. 160–167. ISBN: 9781605582054. DOI: 10.1145/1390156.1390177. URL: https://doi.org/10.1145/1390156.1390177.

[32] A. Tsantekidis et al. "Forecasting Stock Prices from the Limit Order Book Using Convolutional Neural Networks." In: *2017 IEEE 19th Conference on Business Informatics (CBI)*. Vol. 01. 2017, pp. 7–12. DOI: 10.1109/CBI.2017.23.

[33] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory.” In: *Neural Comput.* 9.8 (1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/NECO.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[34] *Understanding LSTM Networks*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[35] David Murray et al. “A Data Management Platform for Personalised Real-Time Energy Feedback.” In: *Proceedings of the 8th International Conference on Energy Efficiency in Domestic Appliances and Lighting*. Aug. 2015.

[36] *UCI Machine Learning Repository: Individual Household Electric Power Consumption Data Set*. URL: https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption.

[37] *Global Solar Irradiance Data and PV System Power Output Data*. 2019. URL: https://solcast.com/.

DECLARATION

-

Groningen, The Netherlands, March 26, 2021

Kareem Al-Saudi