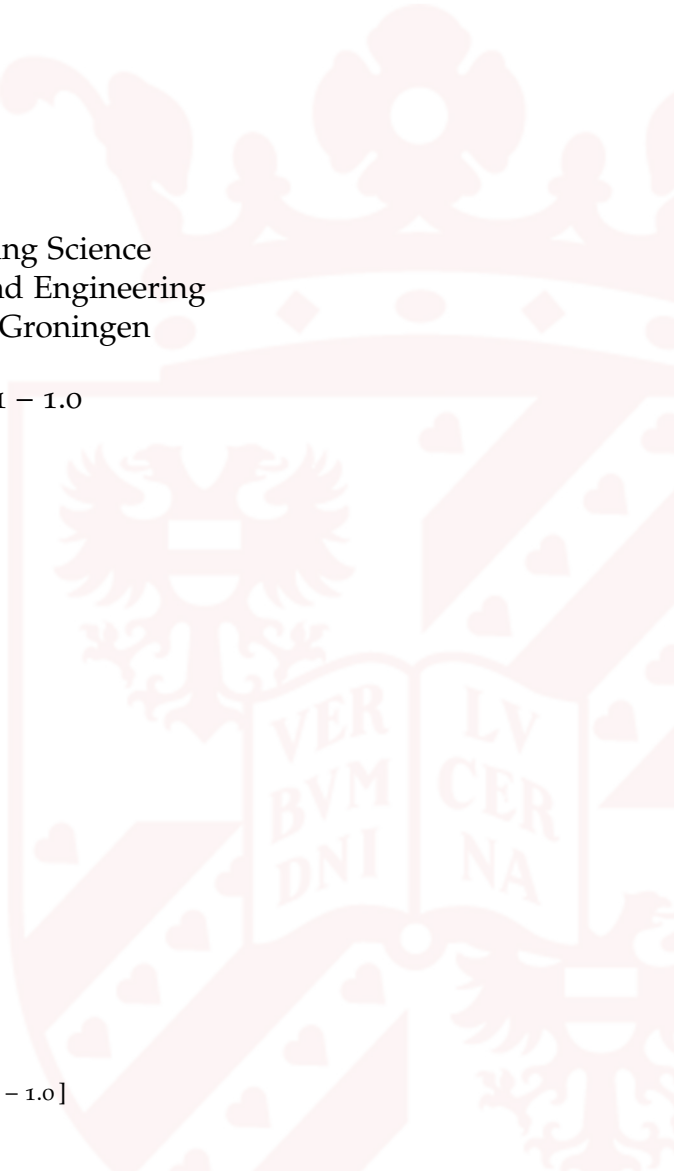
university of groningen

ENERGY CONSUMPTION PATTERNS PROFILING AND SIMILARITY INFERENCE

KAREEM AL-SAUDI



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

July 11, 2021 – 1.0

university of groningen

Kareem Al-Saudi, MSc. of Computing Science, © July 11, 2021

SUPERVISORS:
Viktoriya Degeler
Michel Medema

[July 11, 2021 at 15:15 – 1.0]

ACKNOWLEDGMENTS

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?

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

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].

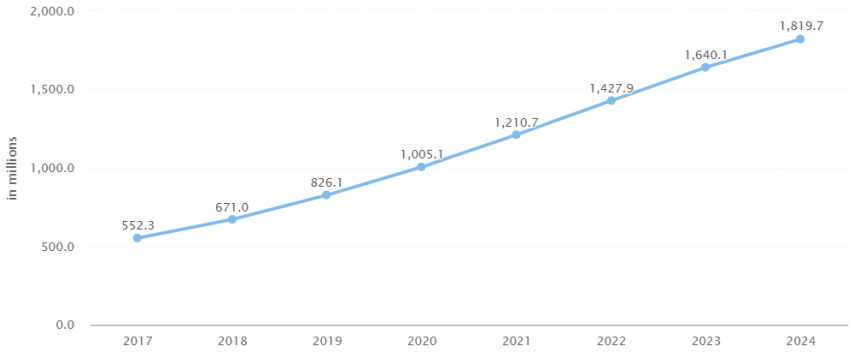


Figure 1.1: Historical/predicted growth in the number of appliances being used worldwide. Image source: [1] © 2019, Statista.

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].

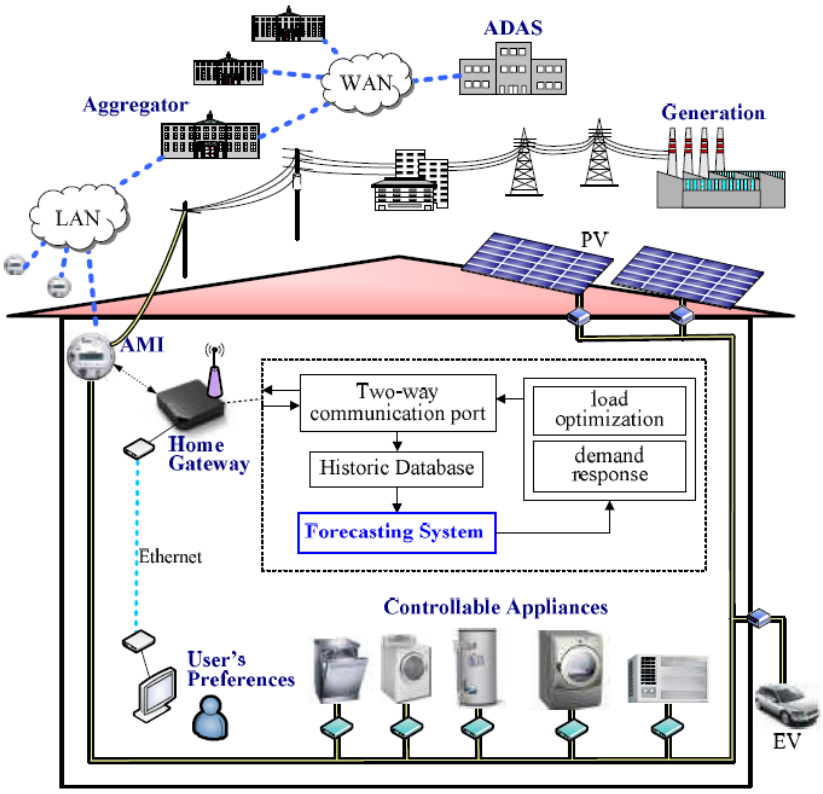


Figure 1.2: The HEMS architecture visualized. Image source: [11] © 2013, IEEE.

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

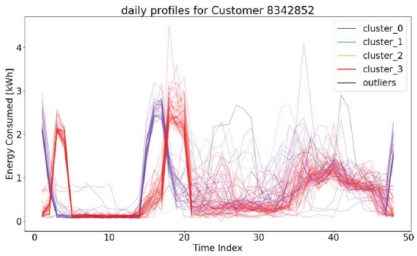
The chief issue that this paper seeks to address is that of creating interesting profiles in terms of recurrent patterns in energy consumption. To do this we will be making use of clustering algorithms that seek to partition our data into a number of clusters so that each of these clusters exhibit some 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

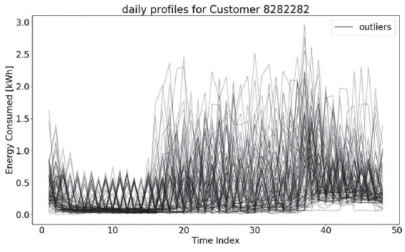
appliances through an ensemble effect of *practices*. These so-called practices can be seen as a series of doings which are governed by the diverse motives and intentions of the occupants of a household and can be classified within four major categories:

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 Density Based Spatial Clustering of Applications with Noise (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 various manifold approximation techniques (such as t-Distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP) 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 *t-Distributed Stochastic Neighbor Embedding*

t-Distributed Stochastic Neighbor Embedding (t-SNE), proposed by Laurens van der Maaten and Geoffrey Hinton [26], is a statistical, or otherwise stochastic, method, that is utilized primarily for visualizing high-dimensional data. In principle, t-SNE works by giving each high-dimensional point in the data set a location in an easy-to-digest 2 or 3-dimensional map. In contrast to Principal Component Analysis (PCA), a well-known linear dimensionality reduction technique, t-SNE is a *non-linear* technique for dimensionality reduction that allows for the separation of our data set in a manner that cannot be separated by any straight line. In order to keep things relatively simple, we can best understand how t-SNE works by breaking it down and providing an overview of the steps the algorithm undertakes.

Let us take a data point, from a subset of N total data points, x_i in the original, high-dimensional space R^D where D represents the dimensionality of said original space. A map point, y_i, that lies in the map space R^2 or, less commonly, R^3 represents one of our original points

in a lower-dimensional mapping that serves as the final representation of our data. To preserve the global structure of the data set in this lower-dimensional space we first define a conditional *similarity*, or conditional probability, that any point x_i would pick another point x_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian distribution centered around x_i with a given variance σ_i^2. This is defined by Equation 3.1.

$$p_{j|i} = \frac{\exp\left(-\left|x_i - x_j\right|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\left|x_i - x_k\right|^2 / 2\sigma_i^2\right)} \tag{3.1}$$

The variance (σ_i^2) differs between points and is chosen in such a way that points in dense areas are given a smaller variance than points in sparse areas. This is introduced as the concept of *Perplexity* within the realm of the t-SNE algorithm (determining optimal σ for each point) and is defined as:

$$\text{Perplexity} = 2^{-\sum_j p_{j|i} \log_2 p_{j|i}} \tag{3.2}$$

Similarity is the defined as a symmetrized version of the previously defined conditional probability in Equation 3.1:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N} \tag{3.3}$$

Equation 3.3 provides us with a similarity matrix for our original data set that represents the similarity between all of our data points as they lie in their original, high-dimensional space. We then define a similarity matrix for the points that lie on the mapping space as such:

$$q_{ij} = \frac{\left(1 + \left|y_i - y_j\right|^2\right)^{-1}}{\sum_{k \neq i} \left(1 + \left|y_k - y_i\right|^2\right)^{-1}} \tag{3.4}$$

The goal then is to minimize the differences between the 2 similarity matrices in order to achieve a good overall representation of our data points in the lower-dimensional, mapping space. To do this the t-SNE algorithm minimizes the Kullback-Leiber divergence between the 2 distributions p_{ij} and q_{ij}:

$$\text{KL}(P \| Q) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{3.5}$$

This *score* is minimized by performing a gradient descent that can be computed analytically and, in essence, represents the magnitude of the pull between data points in our lower-dimensional, mapping space as well as the direction of said pull:

$$\frac{\partial \mathrm{KL}}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j) \left(1 + \left|y_i - y_j\right|^2\right)^{-1} \tag{3.6}$$

The choice of using the so-called t-Student (or Cauchy) distribution, as seen in Equation 3.4, for the map points as opposed to the Gaussian distribution used for the original data points is to alleviate the *crowding problem*. This problem is defined in the original paper as follows – "the area of the two-dimensional map that is available to accommodate moderately distant data points will not be nearly large enough compared with the area available to accommodate nearby data points". In essence, using the same Gaussian distribution for the original data points and the map points would lead to an imbalance in the distribution of the distances of a point's neighbors due to the fact that the distribution of the distances varies vastly between high-dimensional spaces and low-dimensional spaces. By using the t-Student distribution, points close in the high-dimensional, original space get even closer in the lower-dimensional, mapping space while points that are further away from each other in the high-dimensional, original space get even further in the lower-dimensional, mapping space. The differences between both distributions can be seen in Figure 3.1.

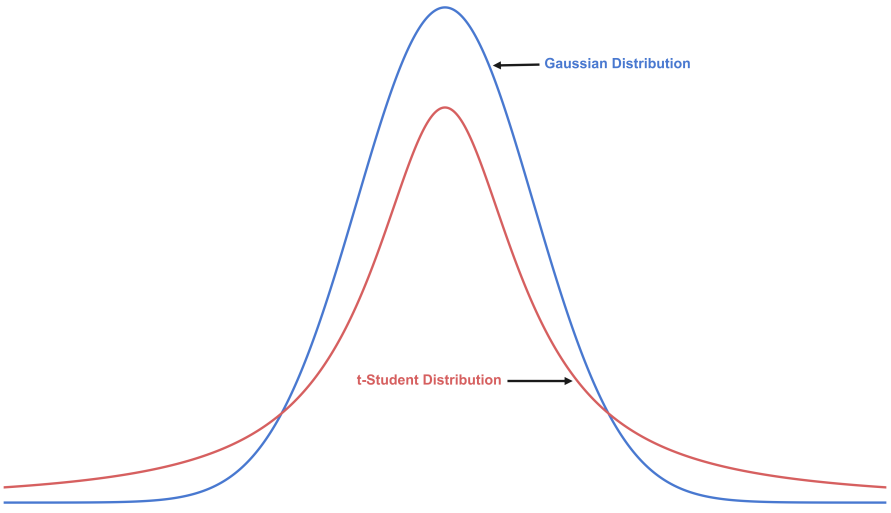


Figure 3.1: Overlapping both a Gaussian distribution and the t-Student, or Cauchy, distribution. Note the heavy tails on the t-Student distribution.

To round things off, we note that the t-SNE algorithm is heavily reliant on the hyperparameters chosen on initialization (predominantly with respect to the chosen value for *perplexity*) and, as it is a stochastic algorithm, different runs performed on the same data set will produce different, albeit similar, results. Furthermore, standard deviations between clusters (or cluster size in terms of bounding box measurements) are not representative of the relative sizes of the actual clusters and generally mean nothing. Finally, distances between clusters in the mapping space does not always give us a good sense of the global geometry – that is, in the sense that distance between clusters may (or may not) mean anything significant.

3.1.1.1 *Uniform Manifold Approximation and Projection*

Although t-SNE is fine to use as a dimensionality reduction technique it is predominantly cited as a visualization heuristic. Interpreting the resulting map obtained from performing or otherwise executing the algorithm must generally be done with some measure of caution. A novel algorithm proposed by Leland McInnes, John Healy and James Melville [27], and aptly named Uniform Manifold Approximation and Projection (UMAP) claims to be competitive with t-SNE in terms of visualization quality and argues that it preserves more of the global structure present in the data while being superior in terms of run time performance. Given the novelty of the algorithm, it lacks the rigorous testing and mathematical analysis that its counterpart, the t-SNE algorithm, is subject to. Nonetheless, we will be making use of the UMAP algorithm for the purpose of this project and refer the reader to the original paper located here [27] to better understand the core concepts of the algorithm as well as what differentiates it from the t-SNE algorithm.

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.

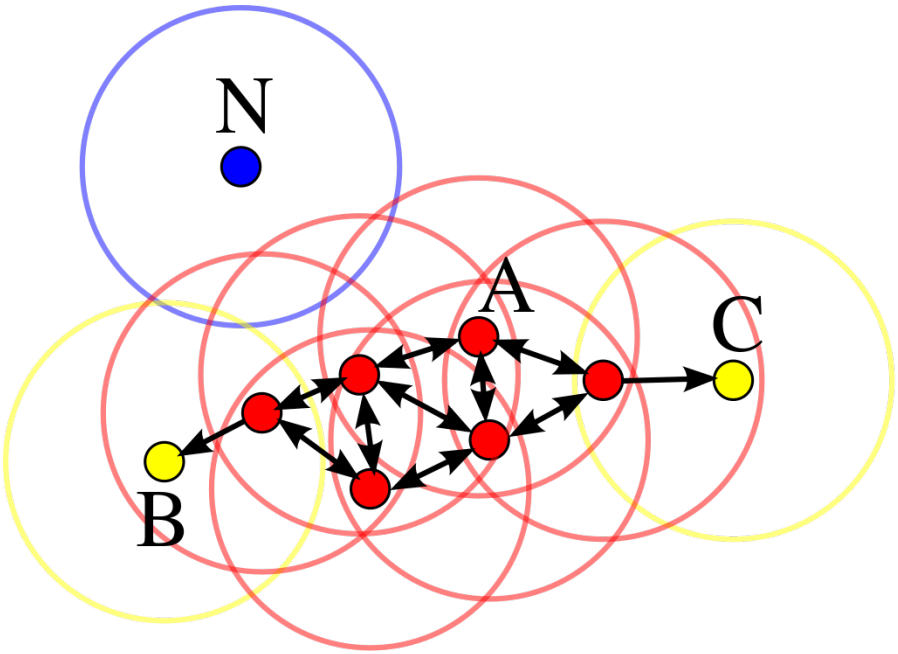
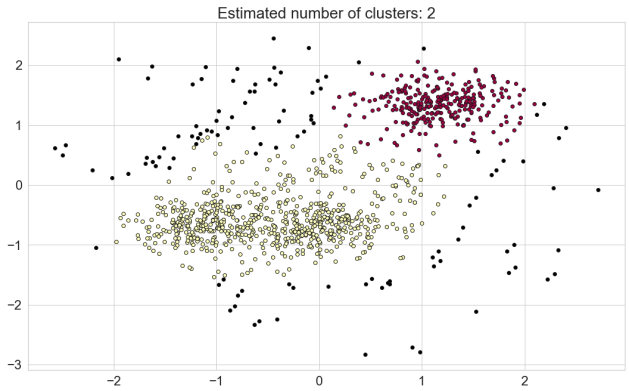


Figure 3.2: Depiction of the DBSCAN algorithm at work with minPts set to 4. Here, point A, as well as the other red points, are core points as the area surrounding these points in a radius ϵ contains at least 4 points (including the point itself). Points B and C are reachable from A through other core 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.

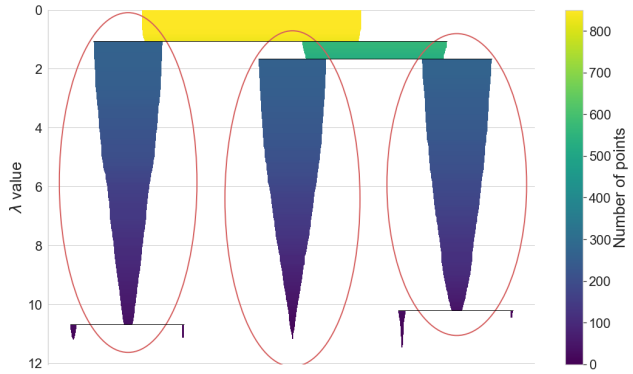
3.2.2 *HDBSCAN*

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.

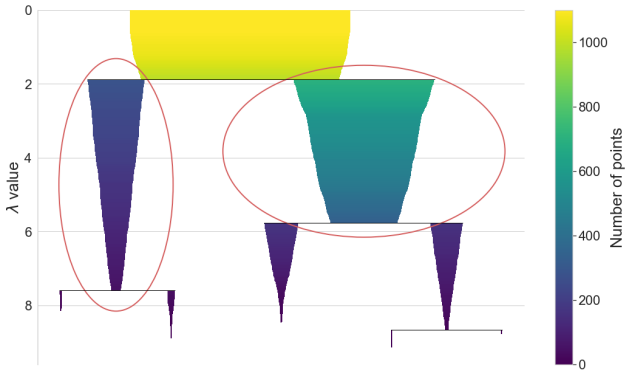
(a) HDBSCAN clustering applied in the case of 3 clusters.



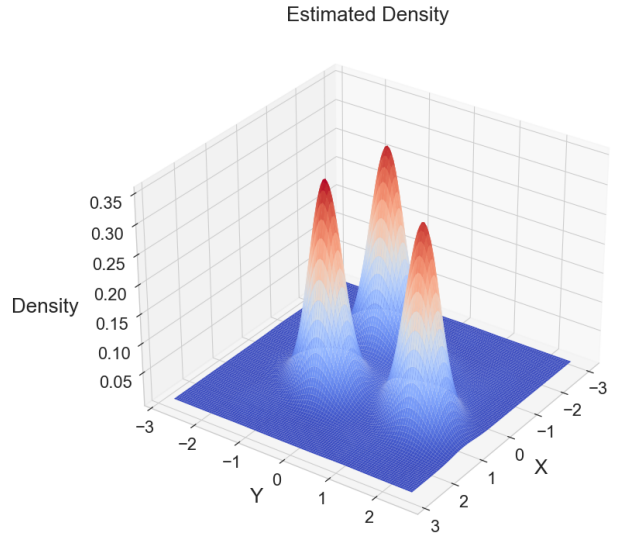
(b) HDBSCAN clustering applied in the case of 2 clusters.



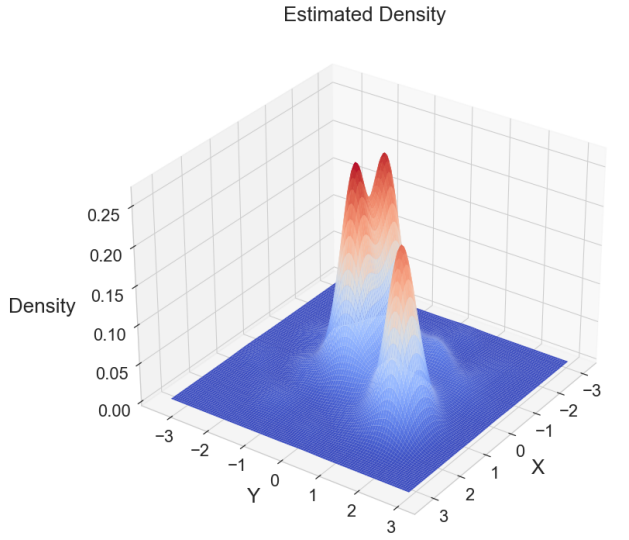
(c) Cluster hierarchy of 3.3a.



(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*

$$d_{euc}(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{3.7}$$

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:

$$D_{min}(S_k, T_k) = \underset{S_{k-1}, T_{k-1}}{\operatorname{argmin}} D_{min}(S_{k-1}, T_{k-1}) + d_{euc}(S_k, T_k | S_{k-1}, T_{k-1}) \tag{3.8}$$

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

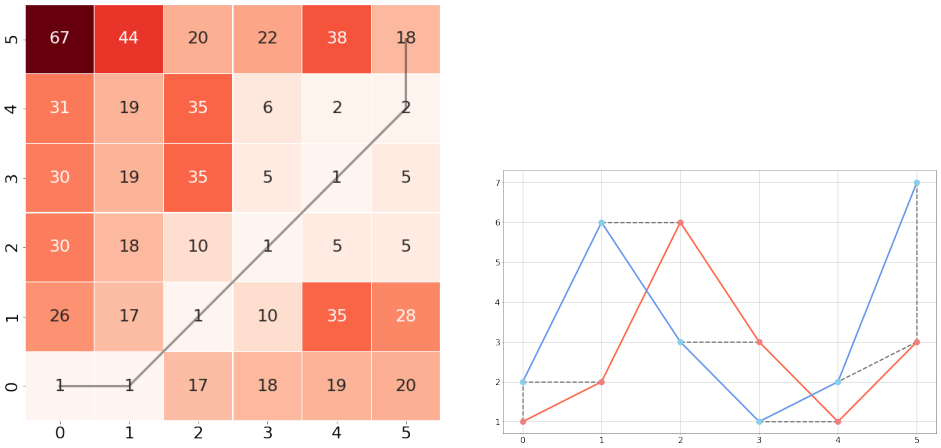
1. Horizontal moves $(x, y) \to (x, y+1)$ – insertion.
2. Vertical moves: $(x, y) \to (x+1, y)$ – deletion.
3. Diagonal moves: $(x, y) \to (x+1, y+1)$ – match.

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*.

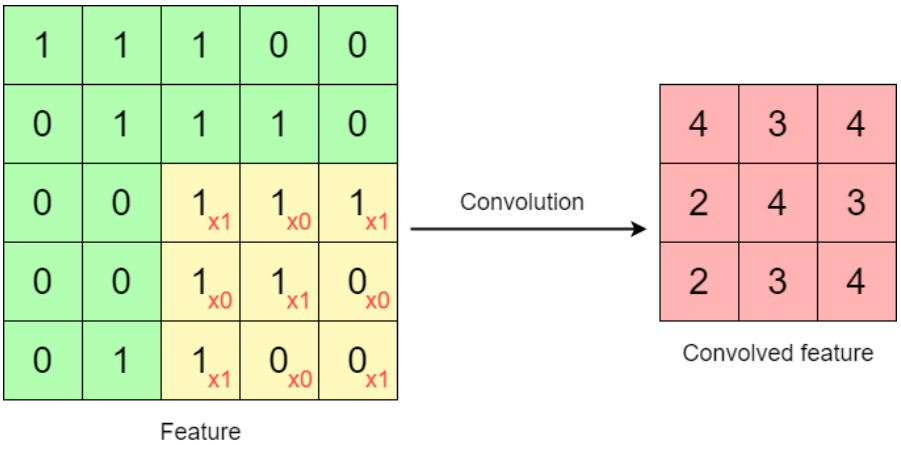


Figure 3.5: An example of a convolutional kernel at work. A 3x3 kernel traverses over a 5x5 "image" with a stride of 1 to produce the convolved feature map.

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:

$$R(z) = \max(0, z) \tag{3.9}$$

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.

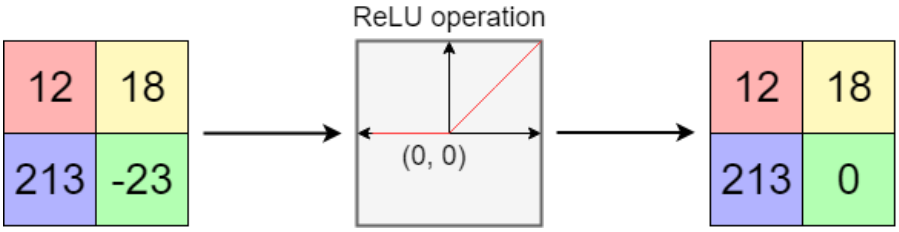


Figure 3.6: A simplified demonstration of a ReLU operation.

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.

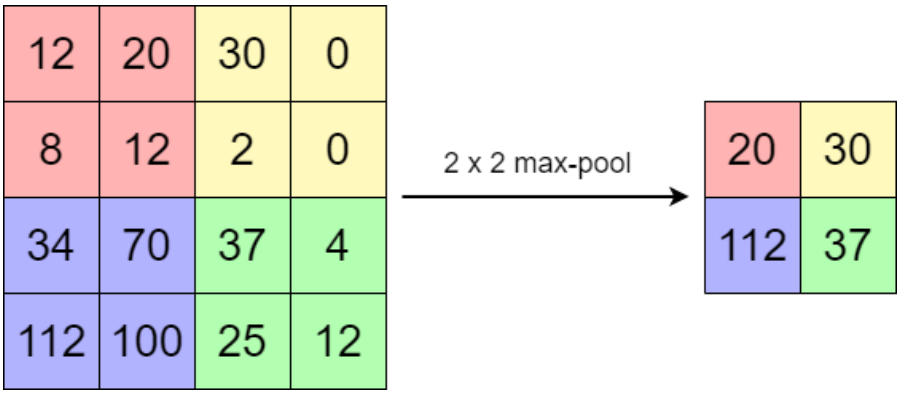


Figure 3.7: An example of max pooling using a 2x2 window.

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.

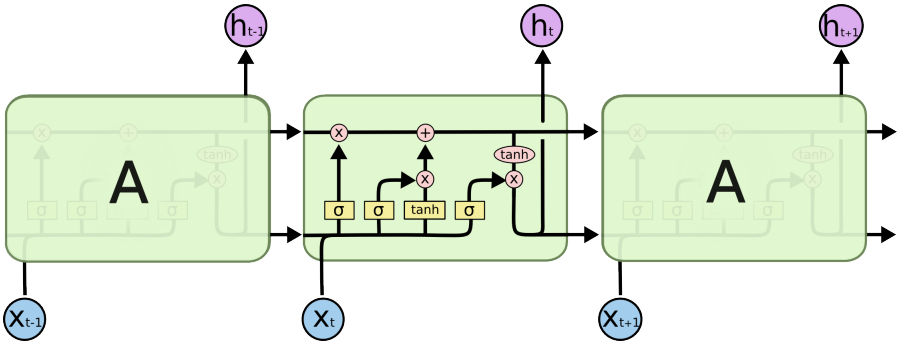


Figure 3.8: The repeating module in an LSTM that contains four interacting layers. Image source: [34] (with permission from the author).

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 3.10.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.10}$$

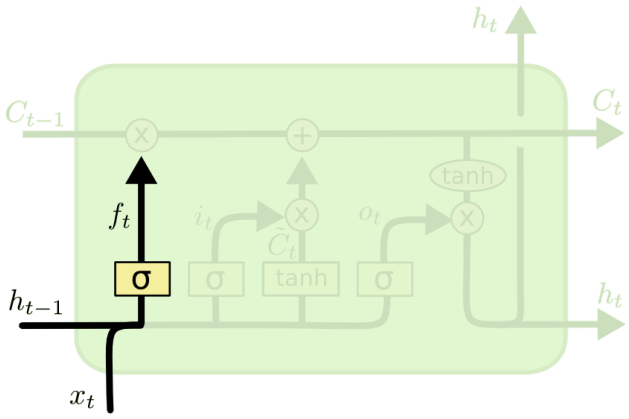


Figure 3.9: An illustration of the forget gate in an LSTM network. Image source: [34] (with permission from the author).

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}) 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 3.11.

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \end{aligned} \tag{3.11}$$

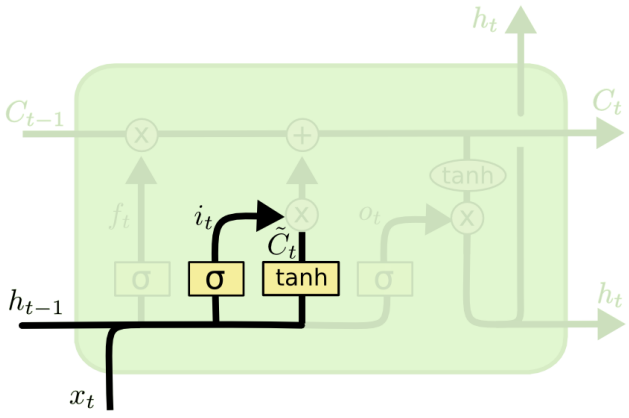


Figure 3.10: An illustration of the input gate in an LSTM network. Image source: [34] (with permission from the author).

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 3.12.

$$\begin{aligned} C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\ o_t &= \sigma(W_o \left[h_{t-1}, x_t\right] + b_o) \\ h_t &= o_t * tanh(C_t) \end{aligned} \tag{3.12}$$

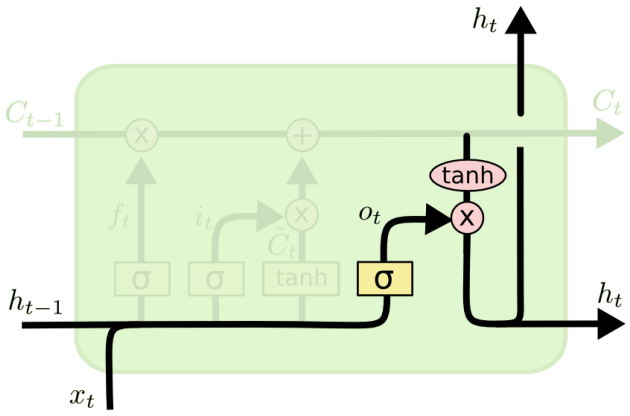


Figure 3.11: An illustration of the output gate in an LSTM network. Image source: [34] (with permission from the author).

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:

$$\text{MAE} = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{n} \tag{3.13}$$

where:

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

3.5.2 *Mean Absolute Percentage Error*

The second 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:

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{\hat{y}_i - y_i}{y_i} \right| \tag{3.14}$$

where:

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

3.5.3 *Log-Cosh Loss*

The final metric that we will be working with is the log-cosh function. Its primary purpose is to serve as a cost function that our forecasting models will seek to minimize. The primary reason behind choosing log-cosh to act as our cost function is that it is robust against the occasional wildly incorrect prediction that our networks are bound to make. Its equation is:

$$\text{Log-Cosh L} = \sum_{i=1}^{n} \log(\cosh(\hat{y}_i - y_i)) \tag{3.15}$$

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 A.1.

5 EXPLORATORY DATA ANALYSIS

Before we can get into the details of our proposed model, it behooves us to perform an initial exploratory data analysis (EDA) so that we may be able to summarize the main characteristics of the data sets that we have on hand. This will both help us understand how to perform the necessary data transformations needed to render our data serviceable as well as aid in the discovery of patterns or anomalies that might be present in the data. To this end we will be making use of a variety of visualization techniques and statistical tests.

5.1 REFIT

The first of the data sets that we will be exploring is the REFIT data set. Given that this data set consists of numerous households, each comprising its own subset of data, the first step in our EDA will be to determine which of these households contains the cleanest data to work with. Following that, the remainder of the sections, and the relevant EDA techniques associated with each section, will be centered around said single household.

5.1.1 *Issues*

As a precursory step, we will first determine to what extent each of the individual households present in the REFIT data set contain any one of a number of issues. We define issues here as any one of the following: missing periods of data, days that exhibit an incomplete number of data, or any values recorded that are labelled 'issue' by the data collection team.

5.1.1.1 *The 'Issues' Column*

The first issue that we will be taking a look at is the aptly named *Issues* column. As previously mentioned in section 4.1, the data collection team responsible for the curation of the REFIT data set appended the *Issues* column so as to indicate that the sample being recorded either contains no issues and can be treated normally, given a recorded value of 0, or that the sum of the IAMs is greater than that of the household aggregate, given a recorded value of 1. In the cases where the recorded value for the *Issues* column reads 1, the data collection team recommends either completely discarding the data or, at the very least, noting the discrepancy before moving on. Table 5.1 outlines the total number of values recorded alongside the number of values

with the *Issues* column set to 1. We note that, for the majority of the households, the number of values recorded that contain issues are rather small with only a small number of households, namely numbers 3 and 5 presenting a problematic number of values with issues and households 7, 13, 18 and 21 closely following suit. Given the overall number of households that we have to work with, and the fact that this project serves predominantly as a proof-of-concept, we can safely discard the aforementioned households that contain a substantial number of recorded values with issues.

HOUSE NO.	DATE RANGE	VALUES RECORDED	VALUES WITH ISSUES
1	2013-10-09 → 2015-07-10	6,960,008	58,183 (0.84%)
2	2013-09-17 → 2015-05-28	5,733,526	28,444 (0.5%)
3	2013-09-25 → 2015-06-02	6,994,594	408,627 (5.84%)
4	2013-10-11 → 2015-07-07	6,760,511	67,441 (1.0%)
5	2013-09-26 → 2015-07-06	7,430,755	425,766 (5.73%)
6	2013-11-28 → 2015-06-28	6,241,971	34,451 (0.55%)
7	2013-11-01 → 2015-07-08	6,756,034	161,919 (2.4%)
8	2013-11-01 → 2015-05-10	6,118,469	25,000 (0.41%)
9	2013-12-17 → 2015-07-08	6,169,525	32,226 (0.52%)
10	2013-11-20 → 2015-06-30	6,739,284	30,162 (0.45%)
11	2014-06-03 → 2015-06-30	4,431,541	40,114 (0.91%)
12	2014-03-07 → 2015-07-08	5,859,544	14,183 (0.24%)
13	2014-01-17 → 2015-05-31	4,737,371	123,796 (2.61%)
15	2013-12-17 → 2015-07-08	6,225,696	23,349 (0.38%)
16	2014-01-10 → 2015-07-08	5,722,544	14,713 (0.26%)
17	2014-03-06 → 2015-06-19	5,431,577	85,937 (1.58%)
18	2014-03-07 → 2015-05-24	5,007,721	174,490 (3.48%)
19	2014-03-06 → 2015-06-20	5,622,610	62,636 (1.11%)
20	2014-03-20 → 2015-06-23	5,168,605	19,594 (0.38%)
21	2014-03-07 → 2015-07-10	5,383,993	206,832 (3.84%)

Table 5.1: Range of dates in the REFIT data set as well as the total number of values and the total number of values that contain issues.

5.1.1.2 *Missing & Incomplete Data*

The second issue that we will be looking at is the combination of both completely missing data as well as days that contain gaps in the recorded data. The primary difference between these 2 sub-issues is that missing data refers to dates within the range of dates, as seen in Table 5.1, that are completely missing from the data set while incomplete days refers to any days that contain less than 96 readings when resampled into a resolution of 15 minutes. Table 5.2 outlines both the number of days completely missing from our data set as well as the number of incomplete days. Furthermore, Table 5.2 contains a value indicating the longest period of consecutive days missing from our data set under the column titled *Stretch*. We note that the earlier households, in order of numbering, tend to contain a larger range of dates recorded and, subsequently, also tend to contain a larger number of completely missing days and a larger period of consecutive missing days. The largest outages seem to span the entirety of the month of February in the year 2014, which is also indicated in the documentation of the REFIT data set, and, as the earlier households tend to have been set up prior to that date it makes sense that they would also contain a larger overall number of missing days. The number of incomplete days displays no such correlation and can likely be attributed to any number of factors on a smaller-scale including household internet failure, hardware failures, network routing issues and the likes. As in section 5.1.1.1 we discard any households that contain a problematic number of missing, or otherwise incomplete, days. This is done in order to maintain a high level of integrity in the data of the households we choose to work with and so as to minimize the overall number of transformations that must be undertaken on said data so as to render it feasible to work with. Given that, we can safely discard households 1, 2, 4, 5, 6, 7, 9, 10, 13, 15, 16 and 21 given a threshold, or cutoff point, of missing or incomplete days set to 10%. When taking into consideration the previously discarded households we are left with numbers 11, 12, 17, 19 and 20 to work with. To narrow it down even further, we have chosen to select household number 12 to work with for the remainder of section 5.1 as, out of the remaining households, it contains the largest amount of days to work with alongside a relatively small amount of missing *and* incomplete number of days. This decision is somewhat arbitrary as any of the remaining households could just as likely have been chosen; however, this does not exclude them or otherwise diminish their relevance to ascertain our findings within the scope of the entire project.

HOUSE NO.	NO. OF DAYS	MISSING DAYS	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 5.2: Total number of days that are missing data in the REFIT data set as well as the number of days that contain incomplete data and the longest period of consecutive days missing data.

5.1.1.3 *Missing IAM labels*

While not necessarily as relevant, having clearly labelled IAMs may help us better understand the patterns present in our data set. Unfortunately, the households that contain cleaner, more user-friendly data to work with do not necessarily contain properly labelled IAMs. This is a trade-off that we are willing to take as the focal point of our research is centered around energy consumption patterns at the aggregate level. For the sake of consistency, Table 5.3 outlines at-a-glance information

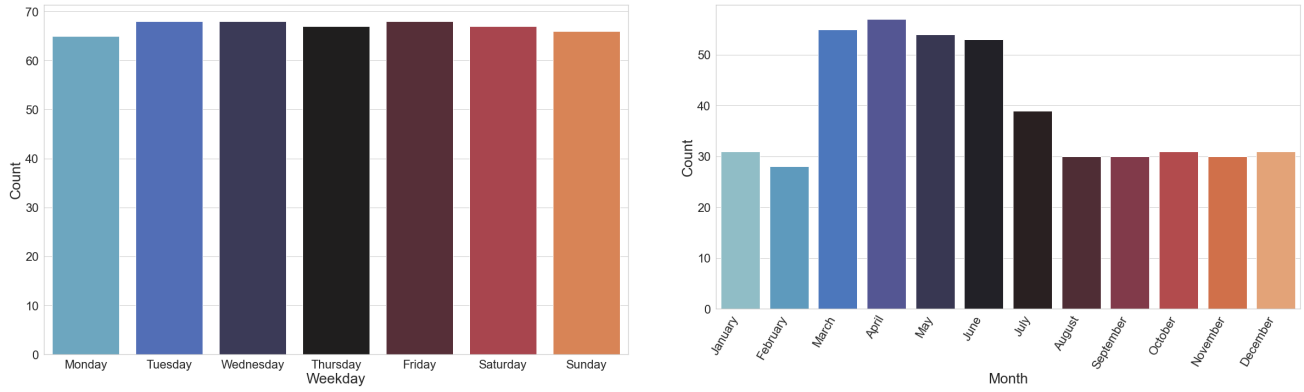
on the number of missing, or otherwise, ambiguous IAM labels per household. Here, missing IAM labels refer to IAMs that contain no recorded values throughout the duration of the data set while ambiguous IAM labels refer to IAMs that either contain numerous household appliances *or* IAMs that monitored a variety of different household appliances throughout the duration of the data collection procedure.

HOUSE NO.	NO. OF IAMS	MISSING IAMS	AMBIGUOUS IAMS
1	9	0	1
2	9	0	1
3	9	0	2
4	9	0	3
5	9	0	2
6	9	0	0
7	9	0	3
8	9	0	1
9	9	0	1
10	9	0	4
11	9	0	3
12	9	3	3
13	9	0	4
15	9	0	2
16	9	0	3
17	9	0	3
18	9	0	1
19	9	0	1
20	9	0	2
21	9	0	1

Table 5.3: Number of IAMs per household alongside IAMs that are missing/did not record any data at all and IAMs that are either ambiguously labelled or IAMs that experience a change in terms of the appliances that they are connected to.

5.1.2 *Data Visualization*

Data visualization is a rather broad term encompassing a large variety of different techniques that serve to display a variety of different aspects of our data set. Within the scope of this project we have chosen to narrow down our focus on a small subset of visualizations that display vital information relevant to the overall forecasting pipeline. The first of these visualizations include the likes of Figures 5.1a and 5.1b that serve to provide an overview of the distribution of samples over the days of the week as well as the months of the year. We note that the plots in Figures 5.1a and 5.1b represent our data set after removing days that contain an incomplete number of values. At a glance we can see that the distribution of samples over the days of the week are relatively even while the distribution of the samples over the months is heavily dominated by the months of March through June and, to a lesser extent, July. When inspecting the results of our clustering algorithm later on in this project, the impact of having nearly twice as many samples for the aforementioned months might skew the results and as such, we will have to keep that in mind when interpreting said results.



(a) 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.

(b) 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 5.1

Figure 5.2 depicts how observed electric energy consumption data can be decomposed into three main components: trend, seasonal and noise [38]. This visualization helps us better understand the problem of analyzing and forecasting patterns in energy consumption. For the purposes of this project we will be focusing predominantly on the *trend* as it captures the main essence of the energy consumption patterns present in the individual household(s) that we are exploring. With that said, we will also attempt to apply our proposed model on the observed data, as is.

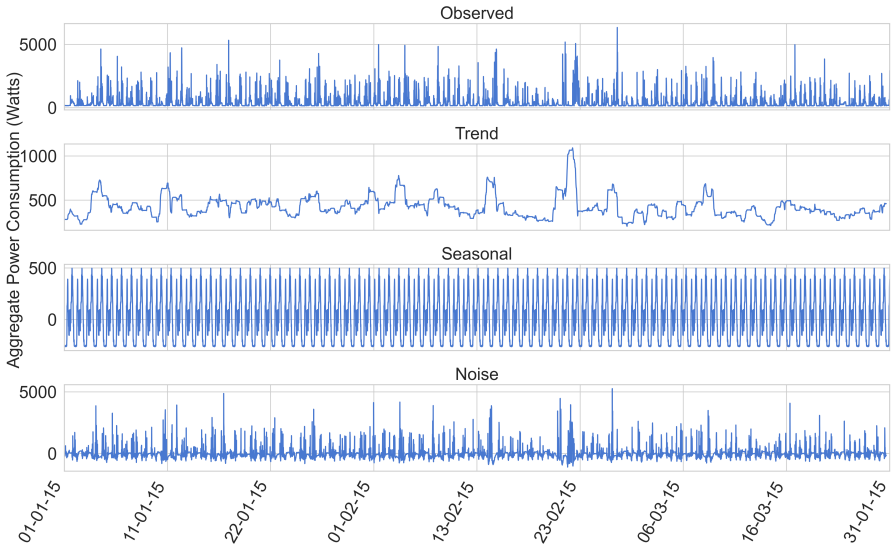
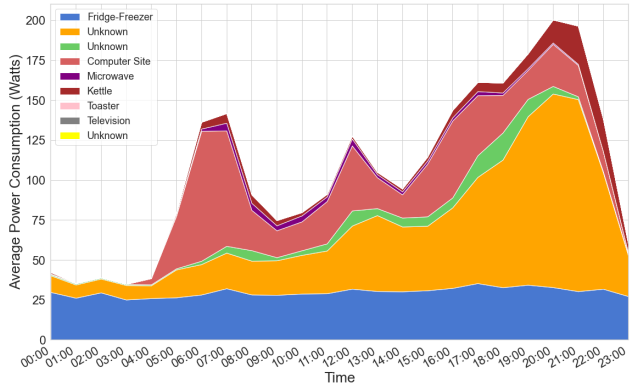
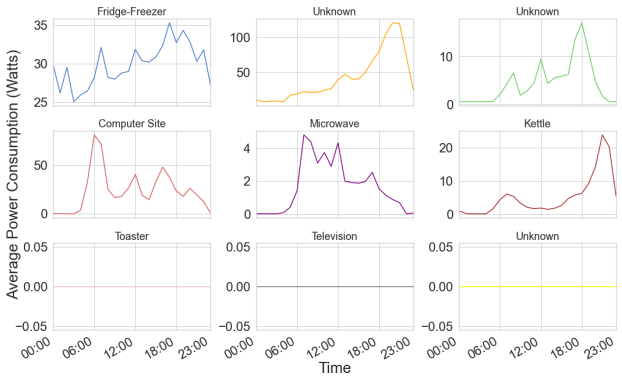


Figure 5.2: Time series decomposition. Data for these plots were pulled over a 3 month period that was resampled into a resolution of 15 minutes from CLEAN_House12.csv of the REFIT data set.

Finally, we end off section 5.1.2 by presenting a stacked area chart that provides a brief overview of the energy consumption patterns of the IAMs present in household 12 of the REFIT data set. Unfortunately, as household 12 does not contain clearly labelled IAMs we are unable to distinguish entirely which appliances make up the patterns that we see in Figures 5.3a and 5.3b. Most importantly we see that 3 IAMs in Figure 5.3b, 2 of which labelled as toaster and television and 1 of which is unlabelled, recorded absolutely no data whatsoever.



(a) 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.



(b) In-depth look at the active hours of individual appliances present in Figure 5.3a.

Figure 5.3

That said, a quick glance at Figure 5.3a shows us that the frige/freezer combination tends to run idly, consuming a consistent amount of energy throughout the day, while other appliances present noticeable spikes in the morning and much later on in the evening and, to a lesser extent, in the afternoon. The only clearly visible correlations, as a result of these missing labels, is that the computer site and microwave tend to be in use at roughly the same time, possibly around breakfast, lunch and dinner. A pair of unknown appliances spike together sometime around the evening; these could possibly be the television set alongside hi-fi or something of the sort; however, there is no clear way to completely ascertain these claims without having the IAM labels on hand.

5.1.3 *Causality & Correlation*

Given the substantial amount of features or, in other words, independent variables that we will be appending to our data set in the feature engineering step of our forecasting pipeline it is only appropriate then to perform a cursory examination as to the relative importance of each of these features with respect to their ability to aid us in forecasting our target variable which, in this case, is the aggregate power consumption of an individual household. To this end, a variety of tests, statistical or otherwise, are available that allow us to ascertain the relationship between the independent variables in our data set and our target variable.

5.1.3.1 *Granger Causality Test*

The first of these tests that we will be performing is the Granger Causality test. First proposed in 1969 by Granger [39], the Granger Causality test is a statistical hypothesis test that allows us to determine whether one time series is useful in forecasting another. In essence, one time series T_x is said to Granger-cause another time series T_y if it can be shown that, through a series of t-tests and F-tests on lagged values of both T_x and T_y, that the values present in T_x provide information that is of some statistical significance with respect to future values of T_y. The null hypothesis that we are testing here is that the past values of one time series T_x does not cause the other time series T_y. If a p-value obtained from the test is less than the significance level of 0.05 i. e., 95% confidence then we can safely reject the null hypothesis and ascertain that a relationship exists between the two time series. Figures 5.4 and A.1 depict the output of performing the Granger Causality test on the meteorological features present in our data set as well as the relevant target variable, the aggregate power consumption (*Aggregate*). We keep in mind that to perform the Granger Causality test we make the assumption that all of the variables of our data set are stationary

i. e., characteristics such as mean and variance do not change heavily over time. To confirm this we perform the Augmented Dicky-Fuller test, a unit-root test, that tests the null hypothesis that a unit root is present in our time series data set. Given a significance level of 0.05 i. e., 95% confidence then we can safely reject the null hypothesis for any p-values less than 0.05 and state that the relevant feature does not contain a unit-root and is thus stationary. Our findings can be found in Table 5.4.

FEATURE	P-VALUE	STATIONARY
AirTemp	0.0	True
AlbedoDaily	0.0	True
Azimuth	0.0	True
CloudOpacity	0.0	True
DewpointTemp	0.0	True
Dhi	0.0	True
Dni	0.0	True
Ebh	0.0	True
Ghi	0.0	True
GtiFixedTilt	0.0	True
GtiTracking	0.0	True
PrecipitableWater	0.0	True
RelativeHumidity	0.0	True
SnowDepth	0.0	True
SurfacePressure	0.0	True
WindDirection10m	0.0	True
WindSpeed10m	0.0	True
Zenith	0.0	True
Aggregate	0.0	True

Table 5.4: The results of performing the Augmented Dicky-Fuller test.

In the output of the Granger Causality test we performed, as seen in Figure 5.4, the rows represent the predictor series (T_x) while the columns represent the response series (T_y) where T_x causes T_y. The values in the matrix represent the respective p-values obtained from the test where any value that falls under the significance level of 0.05 indicates that the corresponding T_x could be considered to have an effect on, or otherwise, be causing T_y. For the purposes of our test we considered the Chi-squared test $\left(\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}\right)$ testing for causality amongst lags up to a maximum of 12. As we can see, the majority of the meteorological features seem to form a relationship with our

target variable, barre the AlbedoDaily, CloudOpacity, Direct (Beam) Horizontal Irradiance (EBH) and WindDirection which we can safely drop from our data set.

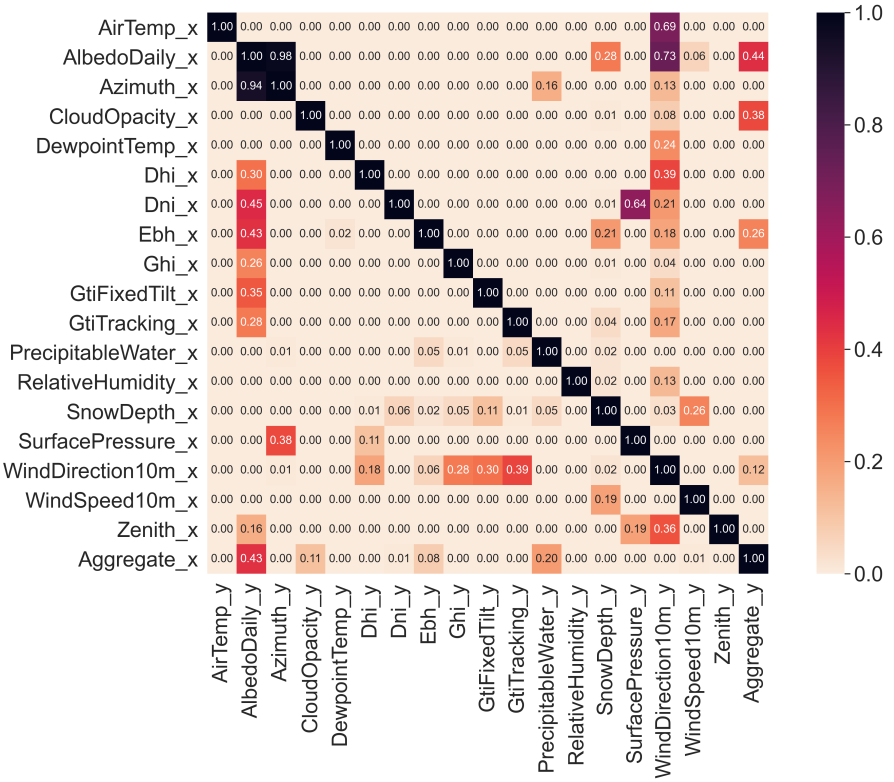


Figure 5.4: The complete Granger Causation matrix with all of the relevant features included.

A trimmed subset of the Granger Causation Matrix seen in Figure 5.4 can be found in Figure A.1 for the sake of clarity and ease of readability.

5.1.3.2 *Mutual Information Gain*

Another measure of dependence between our independent variables and our target variable would be to calculate the mutual information gain. Mutual information quantifies the "amount of information" obtained about one variable through the observation of another variable. The results of calculating mutual information of all our independent variables, including temporal variables, against our target variable can be seen in Figure 5.5. The results seen in Figure 5.5 are more or less in line with the output of the results seen in the output of the Granger Causality test further ascertaining our assumptions that certain features, such as AlbedoDaily, CloudOpacity, EBH and WindDirection, can safely be dropped from our data set and excluded from further consideration as part of this feature selection process.

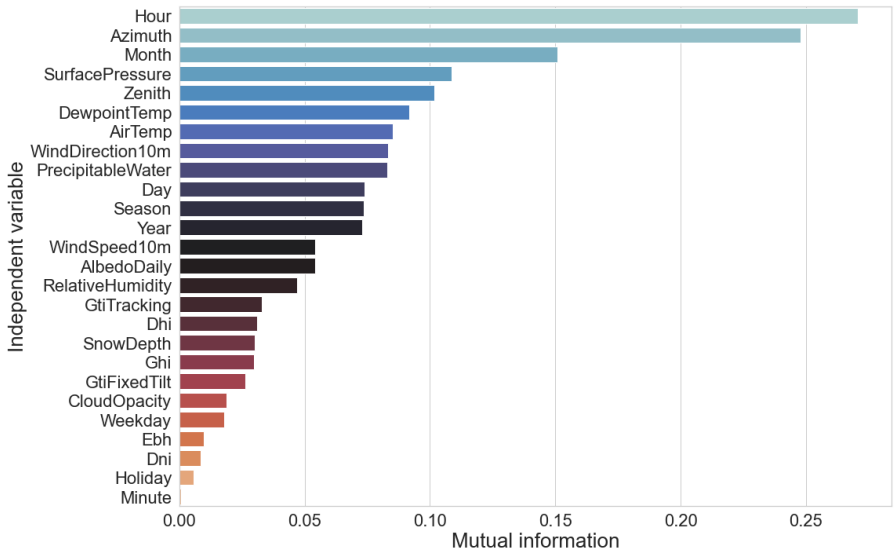


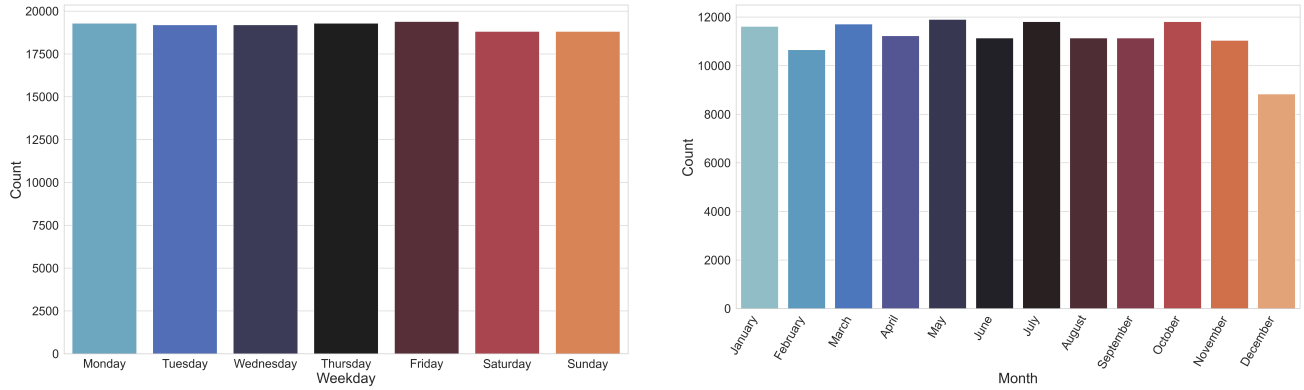
Figure 5.5: Mutual information of our independent variables against our target variable.

5.2 UCID

In contrast to the REFIT data set the UCID data set is not quite as robust in terms of the content it contains. Whereas each house in the REFIT data set contains upwards of 9 IAMs the UCID data set contains but 3 sub-meterings that give us limited insight as to where power is being drawn from within the house. Furthermore, the UCID data set represents but a single household. It does; however, contain a larger amount of data spanning a greater overall span of time. As per the REFIT data set, we will be applying many of the same EDA techniques present in Section 5.1 on the UCID data set.

5.2.1 *Data Visualization*

As per Section 5.1.2, we start off by presenting Figures 5.6a and 5.6b that serve to provide an overview of the distribution of our samples over the days of the week as well as the months of the year. We again note that these plots represent our data set *after* removing days that contain an incomplete number of values. Unlike the REFIT data set, the UCID data set has a much more even distribution over the months of the year with only the month of December standing out in terms of lack of data. Following that, Figure 5.7 depicts how the electric energy consumption data in the UCID data set can be decomposed into the three previously mentioned main time series components: trend, seasonal and noise with the trend being once again the focal point of the research done as part of this project.



(a) Number of samples per day of the week over the entirety of the UCID data set.

(b) Number of samples per month over the entirety of the UCID data set.

Figure 5.6

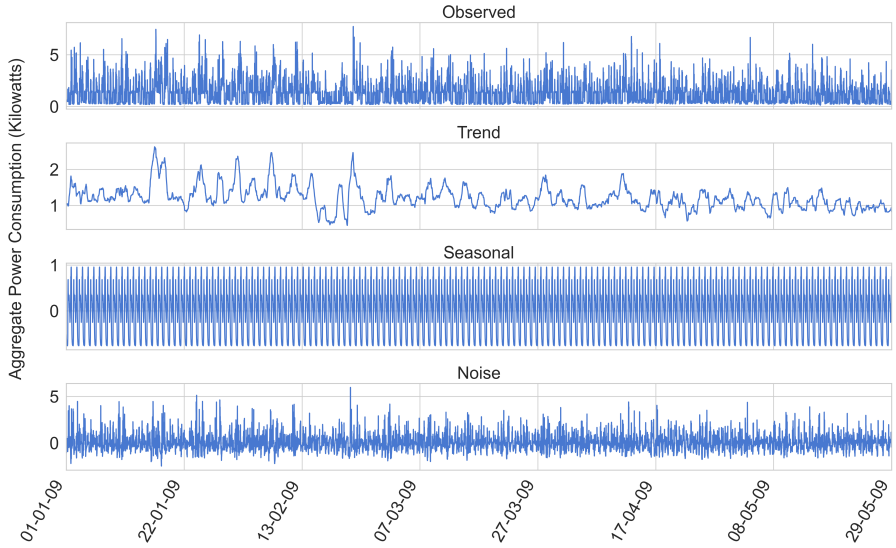


Figure 5.7: Time series decomposition. Data for these plots were pulled over a 6 month period that was resampled into a resolution of 15 minutes.

5.2.2 *Causality & Correlation*

Given that we will be appending similar features to the UCID data set as part of our feature engineering step we will be performing the same examination as to the relative importance of each of these features with respect to their ability to aid us in forecasting our target variable which, in this case, is the global active power consumption of an individual household.

5.2.2.1 *Granger Causality Test*

As per 5.1.3.1, we start off by performing the Augmented Dicky-Fuller test to determine whether our variables are stationary. The results of performing this test can be seen in Table 5.5.

FEATURE	P-VALUE	STATIONARY
AirTemp	0.0	True
AlbedoDaily	0.0	True
Azimuth	0.0	True
CloudOpacity	0.0	True
DewpointTemp	0.0	True
Dhi	0.0	True
Dni	0.0	True
Ebh	0.0	True
Ghi	0.0	True
GtiFixedTilt	0.0	True
GtiTracking	0.0	True
PrecipitableWater	0.0	True
RelativeHumidity	0.0	True
SnowDepth	0.0	True
SurfacePressure	0.0	True
WindDirection10m	0.0	True
WindSpeed10m	0.0	True
Zenith	0.0	True
Global_active_power	0.0	True

Table 5.5: The results of performing the Augmented Dicky-Fuller test.

Unlike the results we observed when performing the Granger Causality test on the REFIT data set, the results obtained when performing the test on the UCID data set, as seen in Figure 5.8, depict the entirety of the meteorological features present in the Solcast data as having some form of a relationship with our target variable. To be able to minimize the overall number of features appended to the UCID data set as part of our feature selection process and not fall victim to the curse of dimensionality it is imperative to determine which of these features presents the highest correlation with our target variable. That said, we will not be relying entirely on the Granger Causality test to determine which features to keep and which features to drop and use it more-so as a heuristic method to approximate which features could potentially prove to provide relevant information when forecasting our target variable

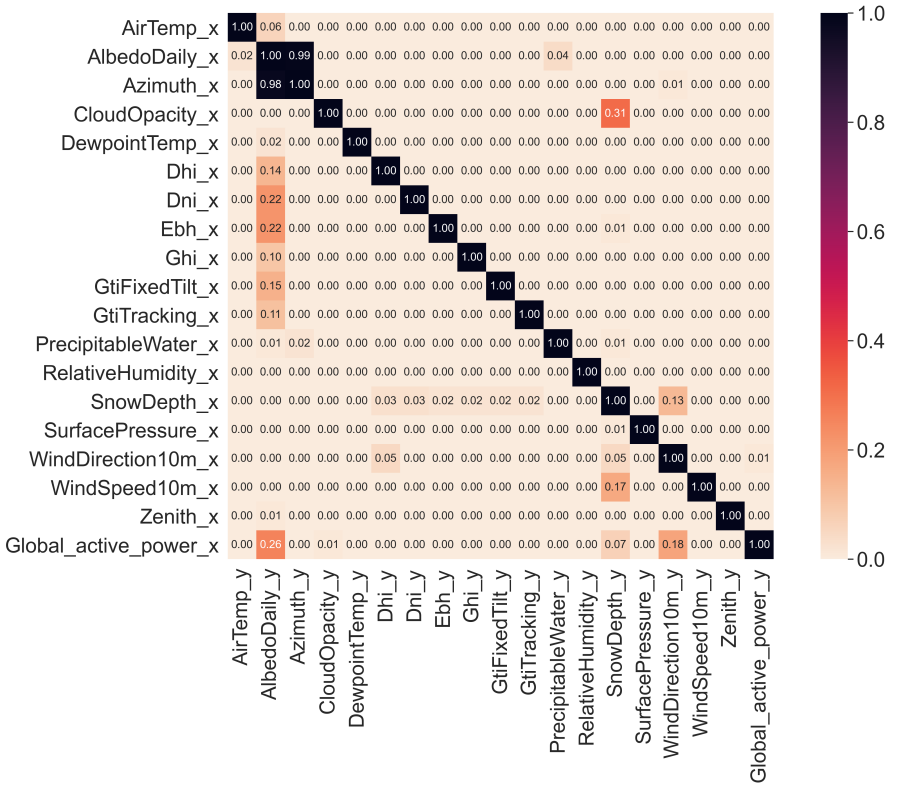


Figure 5.8: The complete Granger Causation matrix with all of the relevant features included.

A trimmed subset of the Granger Causation Matrix seen in Figure 5.8 can be found in Figure A.2 for the sake of clarity and ease of readability.

5.2.2.2 *Mutual Information Gain*

When calculating mutual information gain, our results, as seen in Figure 5.9, are more in line with what we saw when looking at the results obtained when calculating mutual information gain on the REFIT data set. This is meant in the sense that the same features, namely the AlbedoDaily, CloudOpacity, EBH and WindDirection, provide the least amount of information considering our target variable and, so as to also keep the experiments performed on both data sets more or less in line, we choose to exclude these features as part of the feature selection process.

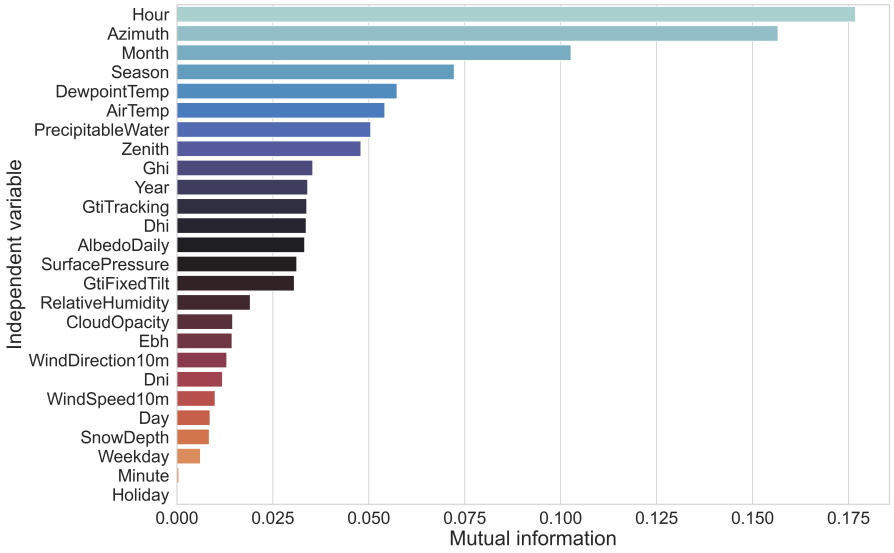


Figure 5.9: Mutual information of our independent variables against our target variable.

Part III

EMPIRICAL STUDY

6

METHODOLOGY

This paper proposes a forecasting method that utilizes dimensionality reduction and clustering techniques to group days that exhibit similarity in terms of electric consumption behaviour. Days that are grouped into the same cluster are thought to contain shared features, whether those features be temporal or meteorological or otherwise, that cause this similarity in behaviour. The formed clusters (per household) are used for 2 purposes: firstly, they will be used to train a classification model that utilizes available context information to assign a new day to the correct cluster. Secondly, and finally, a novel deep learning method will be applied on a per-cluster basis to forecast future energy consumption. An outline of the proposed method can be seen in Figure 6.1.

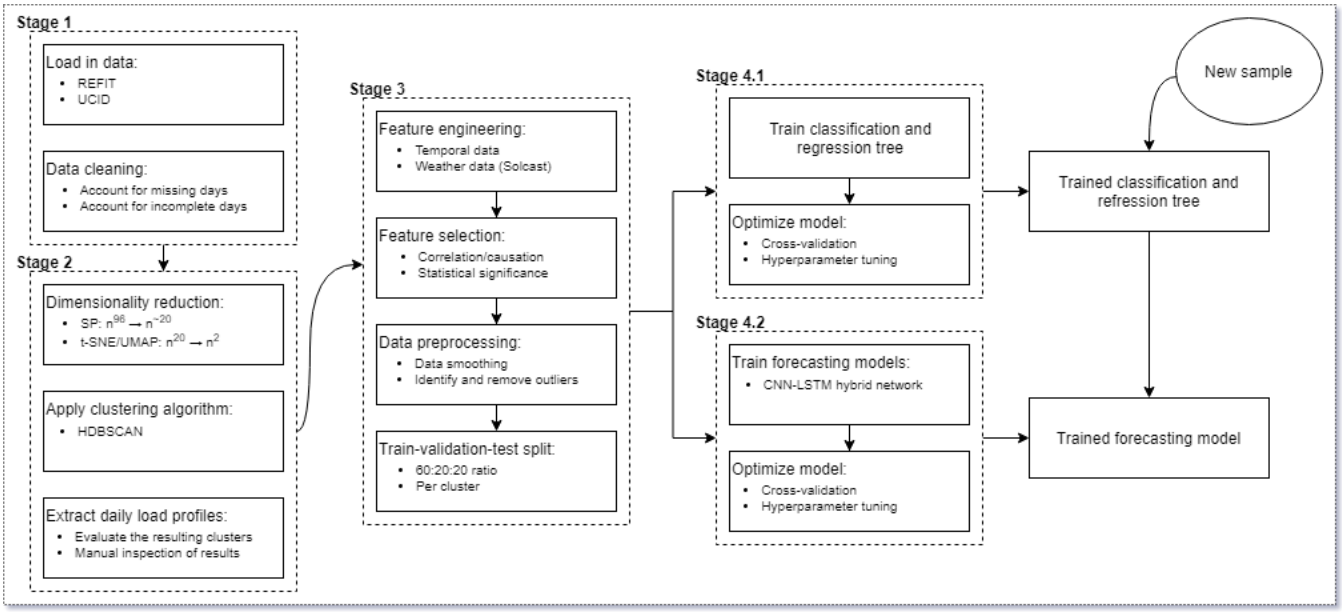


Figure 6.1: Proposed daily profile extraction and load forecasting model.

6.1 STAGE 1 - DATA COLLECTION AND CLEANING

Step 1.1: As mentioned prior, this paper utilizes available historical data with regards to energy consumption on an individual household basis. In reality, as part of stage 1 of our forecasting pipeline, time series data of daily electricity consumption would need to be collected from an individual household meter for an adequate amount of time at an ideal resolution so as to obtain acceptable results.

Step 1.2: After collecting, or in our case loading in, the data we perform common preprocessing techniques to account for noisy or otherwise missing data that occurred during the transmission of the data from the meters. In our case, the available data was resampled into a resolution of 15 minutes and any days that contained less than 96 values (given that there are 96 15 minute chunks in a day) were dropped from our data set. All other days that contained NaN values were also not considered and subsequently dropped from our data set.

6.2 STAGE 2 - DIMENSIONALITY REDUCTION AND CLUSTERING

Step 2.1: Given that each day in our data set is represented by 96 dimensions, each dimension comprising mean active power consumption over a time period of 15 minutes, the first logical step to undertake would be to transform the data in a manner that enables our clustering techniques to more efficiently determine which days exhibit similarity in terms of electric consumption behaviour. This *dimensionality reduction* step comprises 2 parts that are outlined in the substeps below.

To start things off we divide each day into 5 different periods as follows:

1: Morning: 06:00 - 11:00
2: Late morning/afternoon: 11:00 - 15:00
3: Late afternoon/early evening: 15:00 - 20:30
4: Evening: 20:30 - 23:30
5: Late evening/early morning: 23:30 - 06:00

Following that, we represent each period by its respective mean, minimum, maximum value as well as its standard deviation. The outcome of performing this is that each day is now represented by a total of 20 dimensions rather than the initial 96 which is a reduction of $\sim 80\%$.

We can reduce this even further, and even visualize our data in 2 or 3 dimensions, by making use of either of the t-SNE or UMAP algorithms outlined in Section 3.1.1. The most important hyperparameter to tune for either algorithm is the *perplexity* hyperparameter for the t-SNE algorithm and the equivalent $n_{\text{neighbors}}$ hyperparameter for the UMAP algorithm. During our research, we found that an optimal value for either of these hyperparameters is $N^{\frac{1}{2}}$ where N is the number of samples present in the data set. To better understand each of the steps of our proposed model, we will begin a series of visualizations showcasing each step as performed on the UCID data set starting off with the output of performing the t-SNE algorithm which can be seen in Figure 6.2.

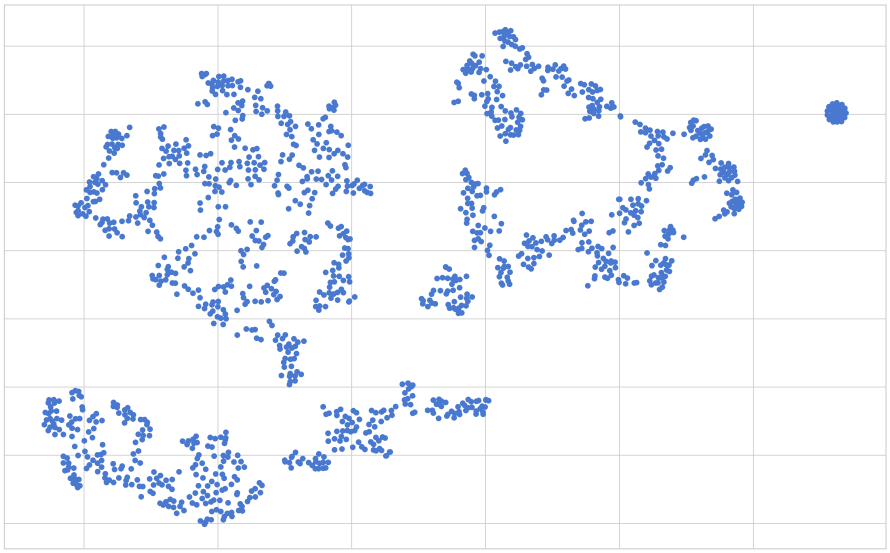


Figure 6.2: The output of performing the t-SNE algorithm on the 20-dimensional UCID data set. Each point in this figure represents a single sample (or day) within our data set mapped onto a 2-dimensional surface.

Step 2.2: After performing the dimensionality reduction step on our data, we proceed to cluster the resulting output by applying the HDBSCAN algorithm, as outlined in Section 3.2.2. As previously mentioned, the only important parameters that need to be passed to the HDBSCAN algorithm are the minimum size we expect each cluster to be. In this case we set that value to $\frac{1}{10}(N)$ where N is the number of samples present in the data set. Our reasoning for selecting this value is predominantly based on the adequate results observed by Kong et al. [10] in their implementation of the DBSCAN algorithm in a similar setting whilst utilizing a similar selection in terms of hyperparameter settings. The other hyperparameter we choose to tune is the *min_samples* hyperparameter which, in layman's terms, denotes how conservative we would like to be with our clustering in terms of restricting clusters to progressively more dense areas and classifying samples from our data set as noise. In our case, an arbitrary value of 15 was selected, in contrast to the default value that sets $min\_samples = min\_cluster\_size$. The results of performing the HDBSCAN algorithm on our 2-dimensional representation of the UCID data set (represented by Figure 6.2) can be seen in Figure 6.3.

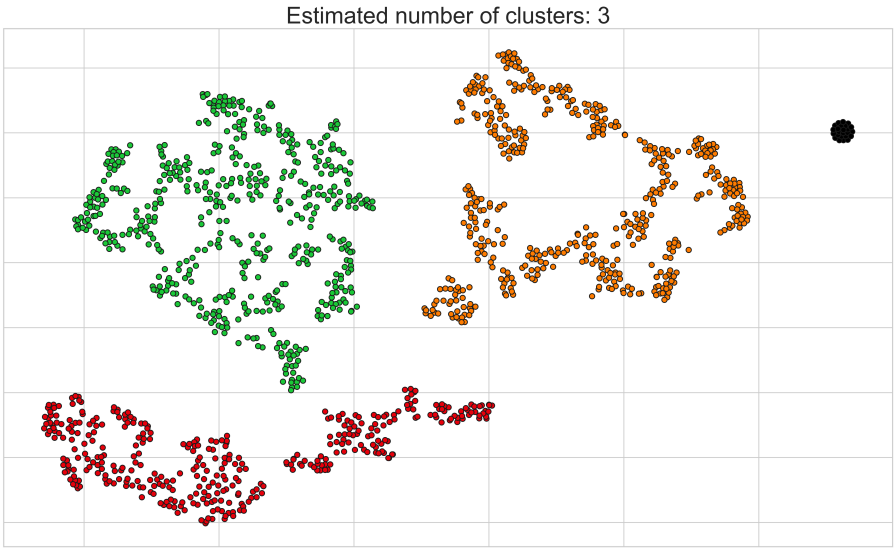


Figure 6.3: The output of performing the HDBSCAN algorithm on the 2-dimensional UCID data set previously seen in Figure 6.2.

For the sake of comparison, we present the output of applying the k-means clustering algorithm (assuming $k = 3$) on the same 2-dimensional representation of the UCID data set. This can be seen in Figure 6.4. We note immediately the capability of the HDBSCAN algorithm in capturing a better representation of the clusters present in our 2-dimensional representation of the UCID data set. The representation of outliers as noise points and not having to have a priori knowledge on the number of clusters present in the data we are working with is a definite pro as well further compounding our choice of clustering algorithm in our proposed model.

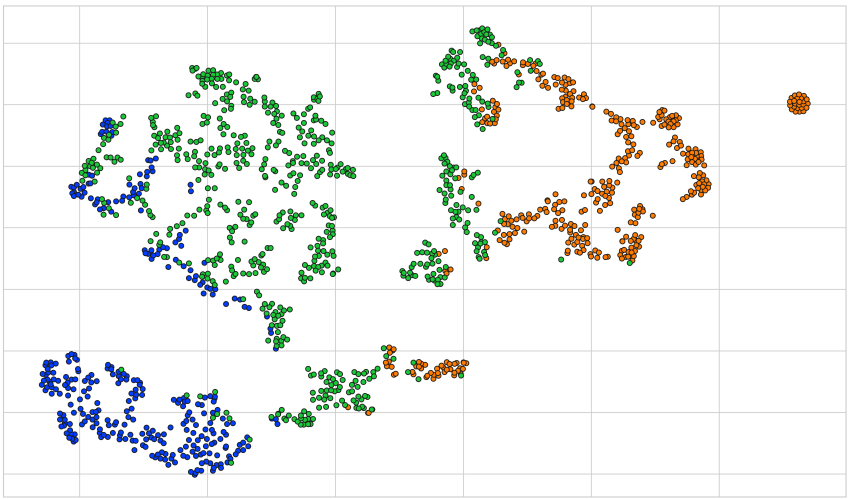


Figure 6.4: The output of performing the k-means algorithm on the 2-dimensional UCID data set previously seen in Figure 6.2.

Step 2.3: Visualizing, or otherwise manually inspecting, the clusters we obtain as a result of our application of the HDBSCAN algorithm is necessary so that we maybe be better able to understand whether our clustering algorithm truly captures the habits of the individuals residing in the households we are working with. The first step in our analysis of the resulting clusters would be to plot the averaged power consumption on a per cluster basis so that we may be able to clearly visualize the patterns in power consumption per cluster. An example of this, in line with the previous examples showcasing our proposed model on the UCID data set, can be seen in Figure 6.5. We note that, in this example, a subset of our data (24 samples in total), were recorded as noise by the HDBSCAN algorithm. Inspecting these samples manually lead to the confirmation that, of the 4 year's worth of data, these 24 days were the only days that exhibited no tangible shift in terms of power consumption throughout the entirety of the day; however, this is not explicitly outlined in the documentation of the UCID data set. This can be seen as a more or less flat line in Figure 6.5.

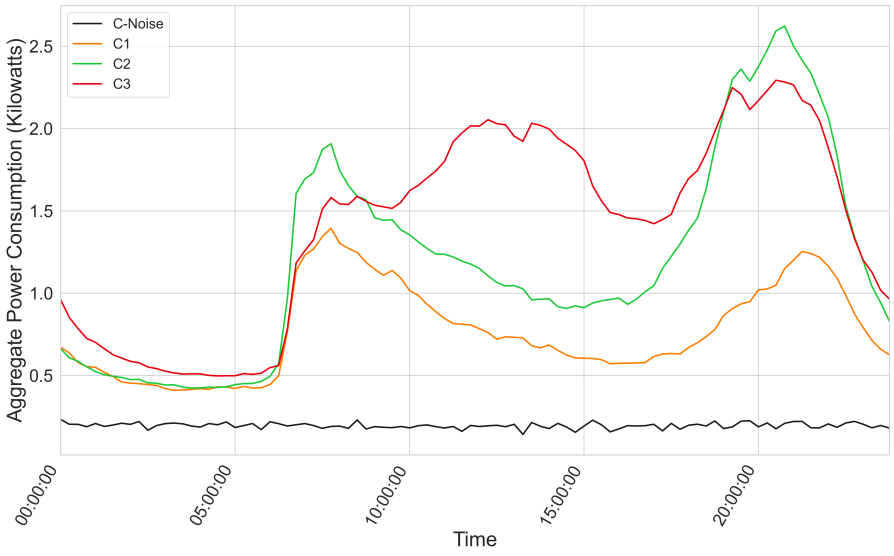


Figure 6.5: Average power consumption per hour of the day for each of the resulting clusters obtained after utilizing the HDBSCAN algorithm on our 2-dimensional representation of the UCID data set.

Following this, Figures 6.6 and 6.7 help us visualize the distribution of the clusters over the months of the year as well as the days of the week to ascertain whether any of the clusters present any correlation with these temporal variables. Given that the initial spread of the data throughout the months of the year and days of the week of the UCID were relatively uniform, we should not see any bias towards any particular month or day in either Figure 6.6 or Figure 6.7 respectively.

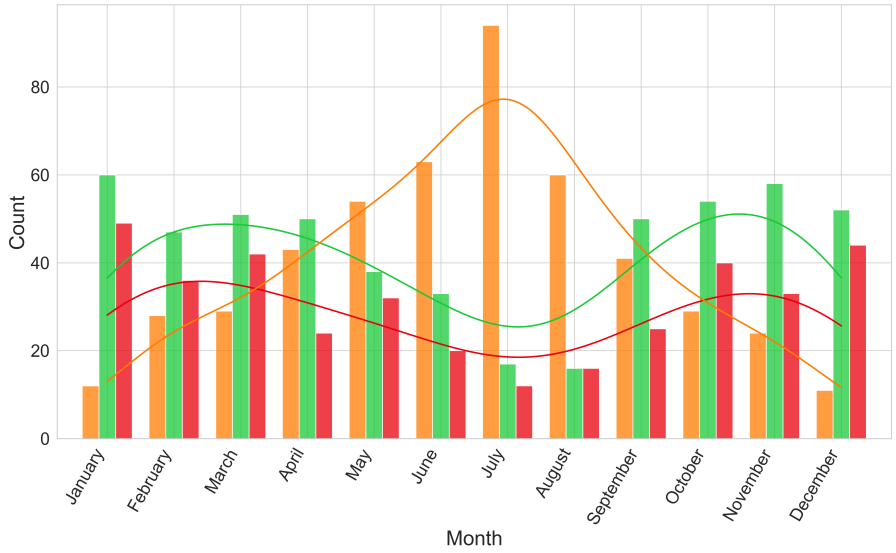


Figure 6.6: Distribution of the clusters over the different months of the year.

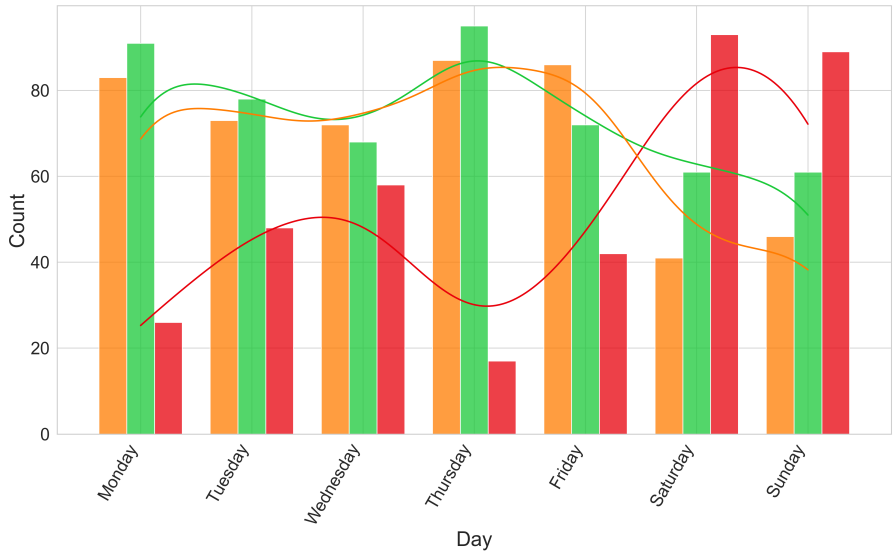


Figure 6.7: Distribution of the clusters over the different days of the week.

At a glance, we notice that clusters 1 and 2 are more likely to occur on the weekdays with cluster 3 taking over the majority share of the weekend which tends to explain the more consistent draw in power throughout the entirety of the day for samples that belong to cluster 3. Furthermore, samples in cluster 1 tend to gravitate towards the warmer, summery months peaking in terms of number occurrences in the month of July while samples in clusters 2 and 3 exhibit a more uniform spread over the remainder of the colder months which could explain the lower average draw in power present in samples that belong to cluster 1 being a result of the owners of the home not being in as often or potentially not needing to make use of appliances to

heat up their home (we note that this data was collected in Sceaux, France that experiences a warm season of $\sim$3 months with otherwise, generally, cooler temperatures).

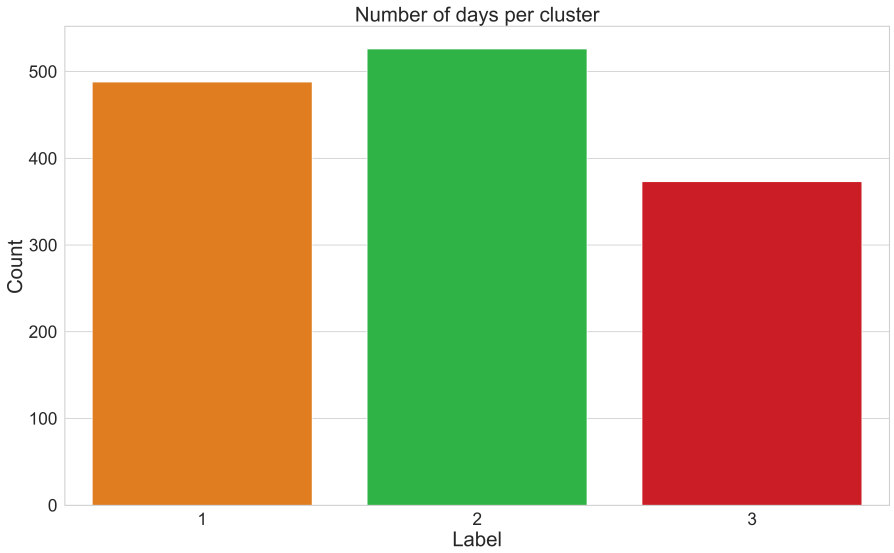
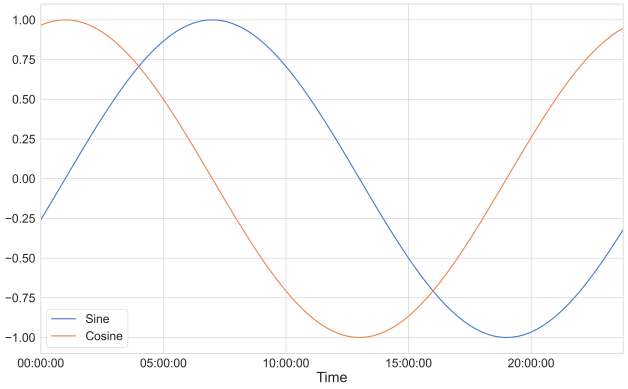


Figure 6.8: Spread in number of samples per cluster label.

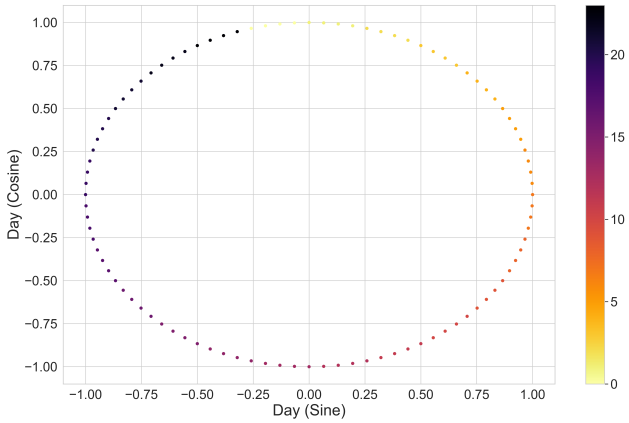
N.B.: It is worth noting that performing these same steps on households from within the REFIT data set exhibit similar results.

6.3 STAGE 3 - FURTHER DATA PREPROCESSING

Step 3.1: The details pertaining to the majority of the steps undertaken throughout the entirety of stage 3 of the proposed model have, to an extent, been explained in-depth during the EDA performed in Sections 5.1 and 5.2. Nonetheless, a brief summary will be provided as part of Chapter 6. The first step undertaken, again on a per-cluster basis, is to append both temporal data as well as meteorological data to our data sets. Table A.2 pertains to the temporal variables that we will be taken into consideration as part of this feature engineering step while Table A.1 pertains to the obtained, historical meteorological data that concern the regions associated with our data sets. Incidentally, as outlined in Table A.2, the temporal variables we have chosen to append do not hold much value given their current format. This is due mostly in part to their cyclical nature (think of how the 23rd hour of the day is rather close to hours 0 and 1). To handle this we can encode our temporal variables (for example, through the use of both the sine and cosine function) in an attempt to transpose our linear interpretation of time into a cyclical state that can be better interpreted by our deep learning model further down the line. The result of performing this so-called encoding can be seen in both Figures 6.9a and 6.9b.



(a) Representing the time of the day as a combination of both sine and cosine waves.



(b) Visualizing our cyclical encoding of the time of day.

Figure 6.9: By utilizing a combination of the sine function and the cosine function, we eliminate the possibility that two different times would receive the same value had we used either function independently. The combination of both functions can be thought of as an artifical 2-axis coordinate system that represents the time of day.

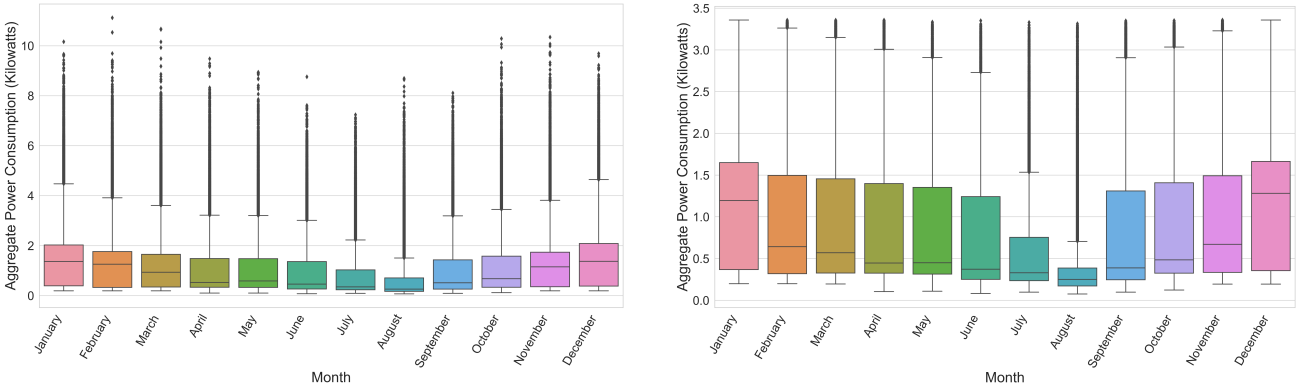
Step 3.2: Following the feature engineering process, the feature selection process (or feature minimization process), heavily revolves around minimizing the overall number of features that do not serve as good predictors of our target variable. This has been explored predominantly in Sections 5.1.3 and 5.2.2. To summarize on the steps undertaken in the sections mentioned prior – we performed a series of tests to determine whether our variables (temporal or meteorological) present a significant level of, independent (or combinatorial), correlation or causation against our target variable for each of the REFIT and UCID data sets. The primary tests conducted revolved around the concepts of Granger Causality and mutual information gain although other factors (such as a per-variable variance threshold) were also looked into. In lieu of repeating content and prolonging this section, we refer the reader to the results we procured in the previously mentioned sections.

Step 3.3: When taking our target variable into consideration, the notion of outliers (and how to deal with them), is inevitable. Scaling the values in such a fashion that accounts for outliers is one possibility whilst defining a threshold and trimming outlier values is another possibility. Alternatively, leaving them in is another possibility as some level of noise is unavoidable in the data collection process and training our models on unrealistically curated data does not serve to produce an accurate representation of a real-life scenario in which a model of this calibre could be applied. Nonetheless, we explore a few possibilities with respect to dealing with outlier values. One possibility, assuming a Gaussian distribution of the values of our target variable, would be to remove all values that fall a pre-defined number of standard deviations, generally 2 or 3, away from the mean. Unfortunately, the

distribution of our target variable does not fall under this pre-condition (as seen in Figure A.3) and so this is not a feasible option. Another method, explored during prior, related research [7], worked on the basis of defining an upper and lower bound based on the interquartile range (IQR). The IQR is calculated as the difference between the 75th (Q3) and 25th (Q1) percentiles of the data and comprises the box in a traditional box and whiskers plot. Using the IQR we can define outliers as any values that are a pre-defined factor below the 25th percentile or above the 75th percentile as follows:

$$Q1 - (1.5 * IQR) < x < Q3 + (1.5 * IQR) \tag{6.1}$$

Figures 6.10a and 6.10b represent the distribution of values for our target variable over the different months of the year both before and after removing outliers respectively.

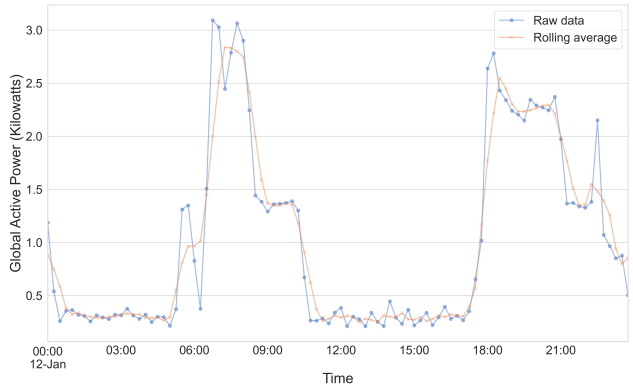


(a) Before removing outliers.

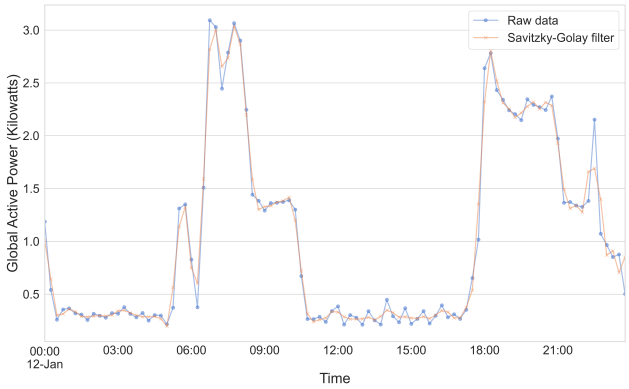
(b) After removing outliers.

Figure 6.10: Illustrating the distribution of values with respect to the global active power of the UCID data set both before and after removing outlier values as defined by Equation 6.1.

Smoothing, or otherwise filtering, the data can also be done through the use of a variety of techniques and can help alleviate some of the issues inherent to the noise present in our data as a byproduct of the data collection process. An example of performing a preliminary smoothing step on energy consumption data can be seen in the work of Hsiao [6] in which a moving (or rolling) average method was utilized. With regards to our proposed forecasting pipeline, we will be utilizing Savitzky-Golay filters to smooth our raw, electrical energy consumption data as, when compared to the moving average method, Savitzky-Golay filters tend to do a better job at preserving the integrity of the raw data. Figures 6.11a and 6.11b serve to illustrate the application of both the moving average method as well as the Savitzky-Golay filter method on a subset of our (raw) data set in order to better visualize the differences between both methods.



(a) Application of the moving average method with a window size of 3.



(b) Application of the Savitzky-Golay filter method with a polynomial order of 3 and a window size of 5.

Figure 6.11: Illustrating the application of both the moving average method as well the Savitzky-Golay filter method in smoothing on a subset of our raw data.

Similarily, when considering the trend component of our data (previously defined in Section 5.1.2 and visualized in Figure 5.7); a preliminary smoothing step can be undertaken through the use of Locally Weighted Scatterplot Smoothing (LOESS) – this is illustrated in Figure 6.12.

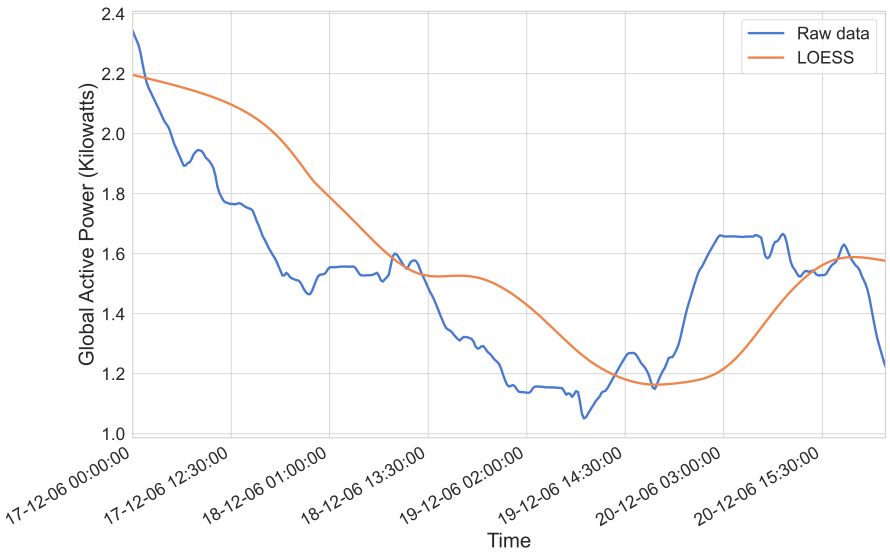


Figure 6.12: An illustration of the previously obtained trend component both with and without the application of LOESS.

Step 3.4: The final step taken as part of Stage 3 of the forecasting pipeline is to split the data into 3 subsets that serve to act as training, validation and testing sets that will be fed to both our classification tree as well as the CNN-LSTM network that we will be using for the purpose of forecasting. A split employing an arbitrarily selected ratio of 60:20:20 is

taken. Given the nature of our study, we choose not to shuffle the data in either of the generated sets as we are primarily interested in our model's capability of forecasting future trends in energy consumption given a measure of historically available data.

6.4 STAGE 4 - TRAINING AND TESTING

In contrast to the earlier stages (or sections), stage 4 will be subdivided into Subsections 6.4.1 and 6.4.2 where Subsection 6.4.1 serves to present an overview of our classification model while Subsection 6.4.2 serves to present an overview of our forecasting model.

6.4.1 *Stage 4.1 - Classification Tree*

Before we can begin attempting to forecast trends in energy consumption we will need to establish, or otherwise ascertain, our ability to correctly assign a new point (or day) to the *correct* cluster. Given that the previously discussed clustering step separated the days in our data set on the basis of similarity in terms of patterns in energy consumption; this will not be an easy feat as the remaining, available context information may not suffice in providing the relevant information to draw up a decision boundary (of sorts) that serves to differentiate individual clusters.

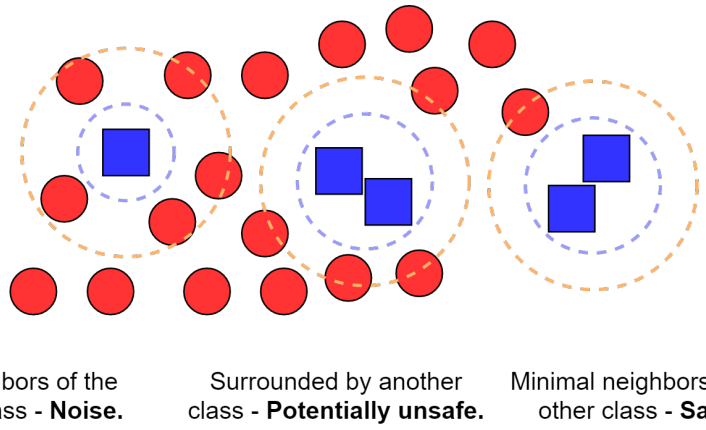


Figure 6.13: An illustration of the Synthetic Minority Oversampling Technique (SMOTE) algorithm in the case of 2 classes depicted by blue squares (minority class) and red circles (majority class). The blue square on the far left is isolated from other members of its class and is surrounded by members of the other class and is this considered to be a noise point. The cluster in the center contains several blue squares surrounded by members from the other class and thus is indicative of potentially *unsafe* points that are unlikely to be random noise. Finally, the cluster in the far right contains predominantly isolated blue squares. The algorithm would then generate new, synthetic samples prioritizing the safer regions.

The first step in insuring a decently trained classifier is to deal with the glaring problem of class imbalance that can be seen in Figure 6.8. The results of our clustering step lead us to an uneven distribution of days among the different class labels which could lead to poor predictive performance as standard classification algorithms are inherently biased to the majority class. A common means to alleviate this issue is through the of either undersampling the majority class(es) or oversampling the minority class(es). In this paper we will be implementing the SMOTE algorithm, a form of informed oversampling, that works on the basis of generalizing the decision region for minority classes and thus provides us with synthetic samples while preventing overfitting. For further explanations as to the workings, advantages as well as shortcomings of this algorithm we refer the reader to the initial paper by Chawla et al. [40] as well as Figure 6.13 that provides a layman's explanation of the algorithm. Additionally, the results of applying the SMOTE algorithm (as expected) can be seen in Figure 6.14.

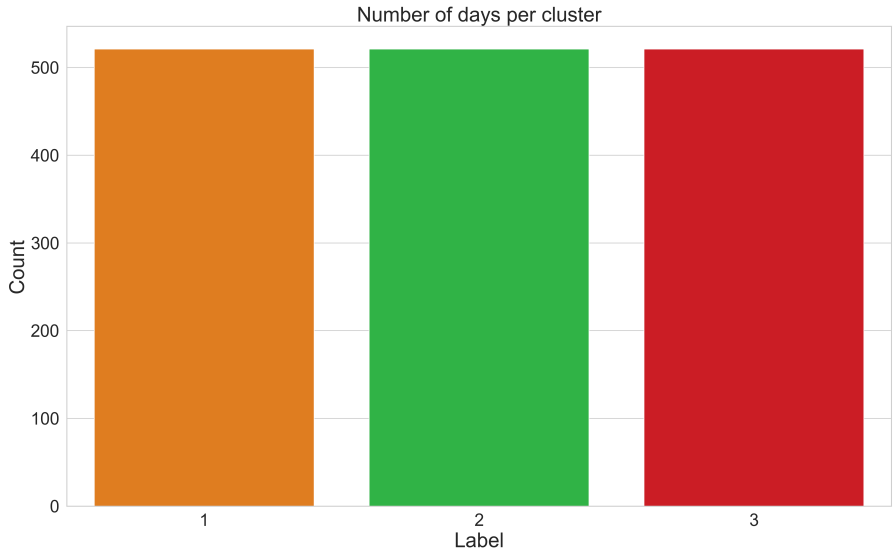


Figure 6.14: Number of samples per class label after applying the SMOTE algorithm.

After handling the class imbalance problem we can shift our attention to both the feature engineering as well as the feature selection process of this particular classification problem. In this scenario, the available context information we have is purely temporal (ordinal day of the week/year, month, season, etc.) as well as historical as well as forecasted meteorological data (air temperature, humidity, cloud opacity, etc.). Numerous methods exist to minimize the overall amount of features being passed to our classification model, some of which that we explored in Chapter 3 of this paper and can be reapplied here to similar effect. In brief, we chose to make use of a Random

Forest Classifier, the hyperparameters of which were tuned through a randomized search over a pre-determined distribution of values per hyperparameter. After assessing the optimal hyperparameters for our use-case, we passed the model as well as the complete set of features through a feature selection algorithm titled Recursive Feature Elimination and Cross-Validation (RFECV). RFECV works on the basis providing a cross-validated selection of the most important features when considering a target label and pruning the less important features. The results of applying this algorithm can be seen in Figure 6.15.

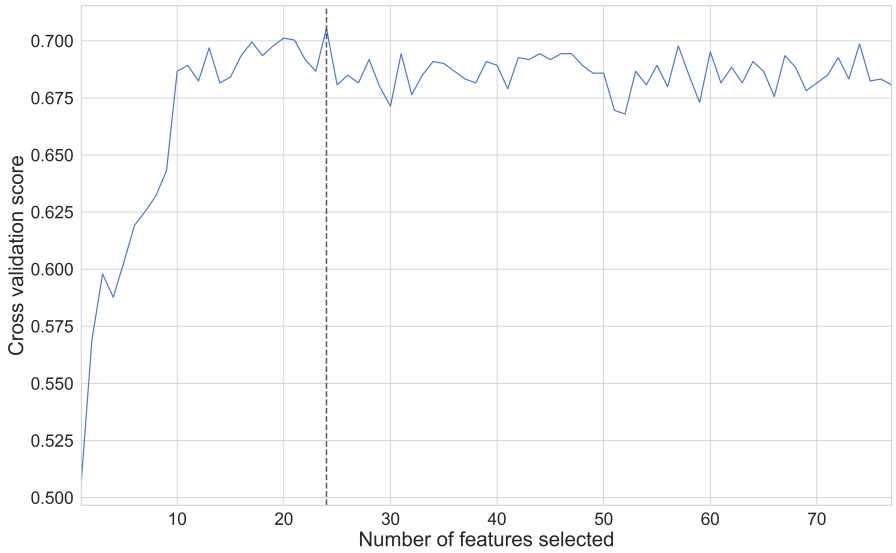


Figure 6.15: Assessing the number of important features through the use of the RFECV algorithm. In this particular scenario, the optimal number of features was pruned down from a total of 77 to a mere 24.

After transforming the data set and pruning the less important features we, once again, train the model on the new, transformed data set utilizing 5-fold stratified cross-validation to assess whether our model is overfitting at any stage and so as to insure an even distribution of class labels per validation set. We can conduct a quick inspection of the now, fitted model by calculated the permutation feature importance on a per-feature basis to validate whether the final set of features are relevant when attempting to classify a new sample into the correct cluster. By definition, the permutation importance of a feature is the overall decrease in accuracy of our model when said feature's values are randomly shuffled and, by doing so, we break the relationship between the feature and the target label. By doing this we can assess how much our model depends on said feature, the results of which can be seen in Figure 6.16. it is important to note that, when calculating the permutation important of strongly correlated features – the model will

still have access to the shuffled feature through its correlated feature which will result in a lower importance value for *both* features when they might actually be important. To address this, it is possible to further prune the data set and remove subsets of features that present strong inter-correlation.

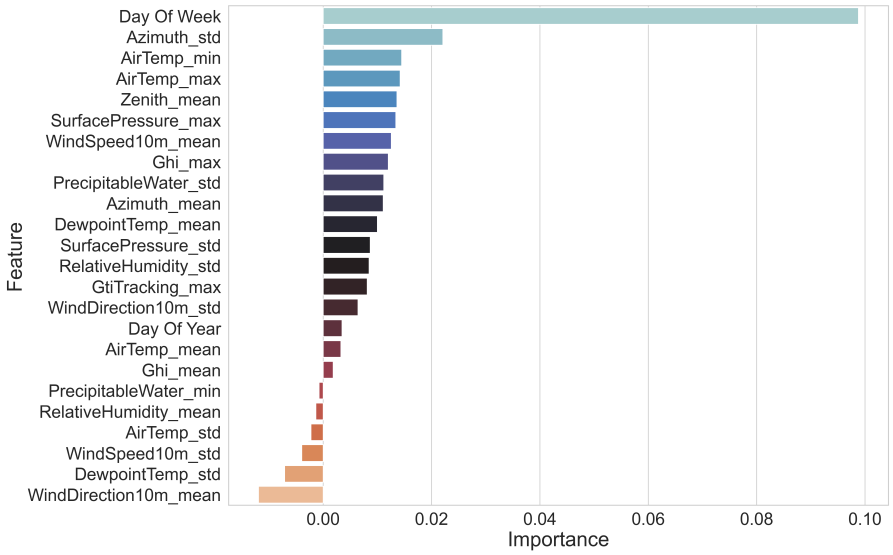


Figure 6.16: The permutation importance of each of the features chosen as part of our fitted Random Forest classifier.

The final model is then ready to accept new samples and assigns them a cluster based on the training procedure outlined through Section 6.4.1.

6.4.2 *Stage 4.2 - CNN-LSTM Network*

The focal point of our research lies in the implementation of a CNN-LSTM model in which the CNN component serves to learn the relative importance of each of the features we pass to the network as input in what we can call a *feature extraction* step. The extracted features are then passed to the LSTM portion of the network that learns the temporal relationship between past, or otherwise historical, values of said features with the present, or future, value(s) of the target variable where finally, an output prediction is made. An example network, built upon this foundational architecture, is visualized in Figure 6.17.

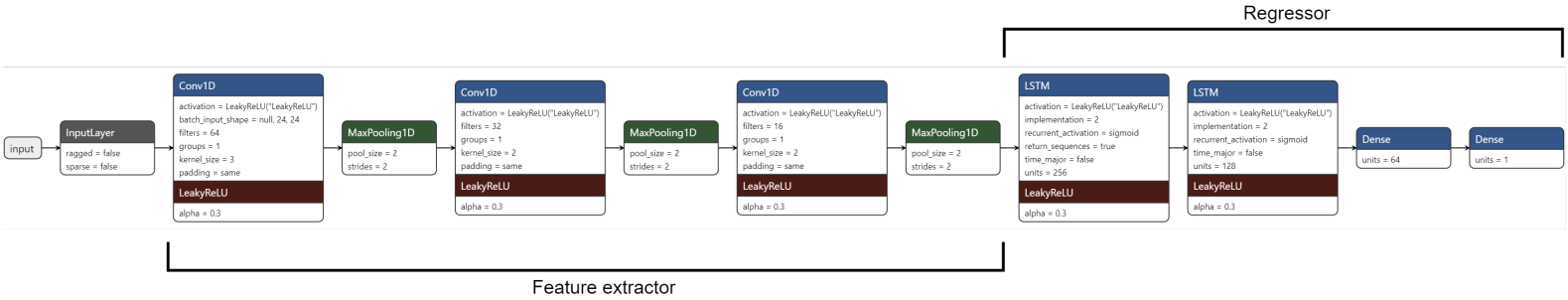


Figure 6.17: A simple, example CNN-LSTM network that makes one-step-ahead predictions.

The combination of both CNN and LSTM components allows the network to learn spatio-temporal relationships between the features being passed as input and the target variable that we are attempting to forecast. In contrast to other architectures and forecasting models, this architecture is demonstrably more efficient [22] when tackling time series problems such as that of residential energy consumption forecasting. The sample network illustrated in Figure 6.17 can be expanded to forecast multiple time steps ahead with minor adjustments and is capable of understanding patterns at variable time resolutions. For the purposes of this example, we will be moving forward with the previously defined resolution of 15 minutes using a window of 24 historical values $(t-24, t-23, ..., t)$ to make a prediction one step into the future $(t+1)$ for both the previously established trend component as well as the raw, unadulterated data.

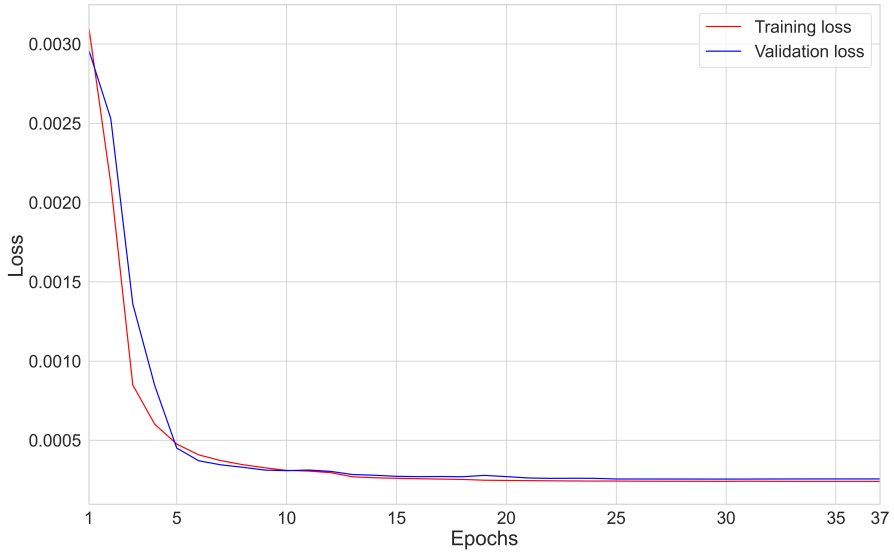
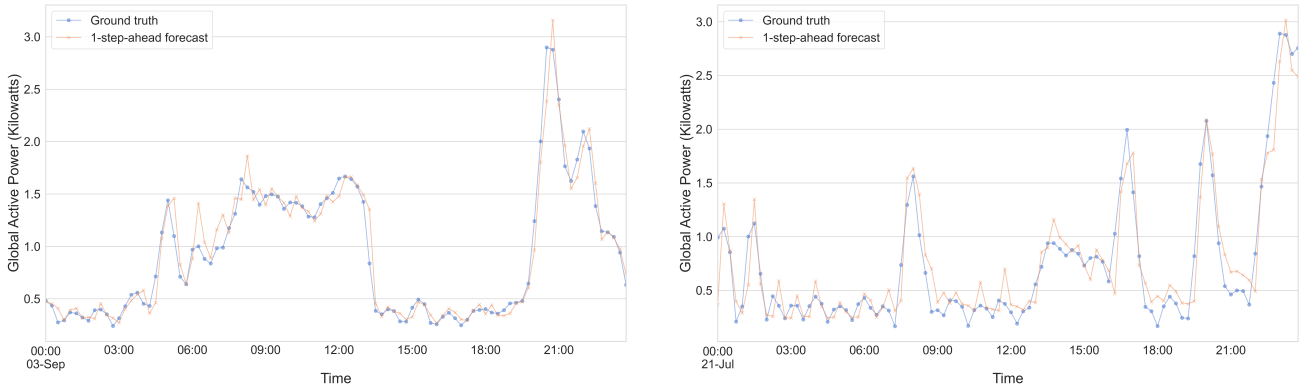


Figure 6.18: Training and validation loss when applying the previously defined network (illustrated in Figure 6.17) on the raw UCID data set in an attempt to make predictions one time step into the future.

To train our network, we will be utilizing Adam [41]: an adaptive learning rate optimization algorithm that was designed specifically for training deep neural networks. In contrast to the ever-familiar Stochastic Gradient Descent (SGD), Adam leverages the power of adaptive learning rate methods and momentum to allocate individual learning rates for each parameter of the network being trained. For further explanations as to the workings of this algorithm we refer the reader to the initial paper by Kingma and Ba [41].



(a) An example of where our network performs optimally, achieving a MAPE of $\sim$ 11%.

(b) An example of where our network performs sub-optimally, achieving a MAPE of $\sim$ 26%.

Figure 6.19: An illustration of one-step-ahead forecasts on 2 separate days in an attempt to showcase cases in which our network performs both optimally and sub-optimally.

When training our network(s), we will be making use of a variety of techniques to improve generalization and prevent overfitting to the training data set(s). The first of these techniques is the notion of *early stopping* (illustrated in Figure 6.20). Early stopping is a form of regularization that monitors the validation loss (or generalization error) and aborts the training when the monitored values either begin to degrade or do not shift for an arbitrarily set number of epochs. The second technique we will be using works on the notion of employing a variable learning rate that, in theory, facilitates convergence of our weight update rule and prevents learning from stagnating thus allowing us to break through plateaus and avoid settling at local minima.

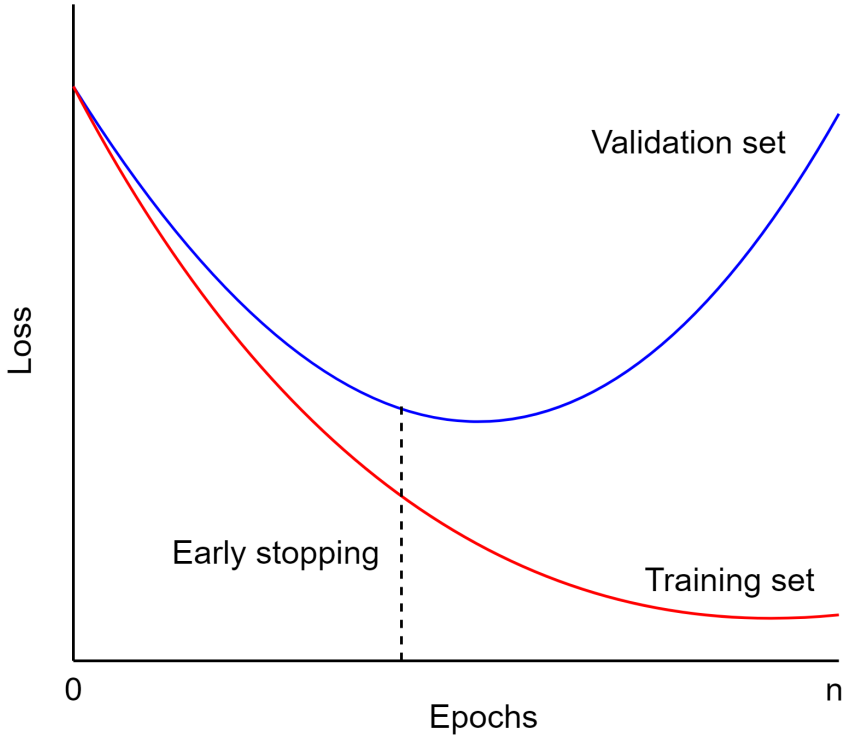


Figure 6.20: Illustration of early stopping.

For the purposes of our experiments and procuring the results showcased in Chapter 7, we will be implementing a network on a per-cluster basis for both the raw data as well as the trend component of each of our data sets. The networks implemented will serve to provide one-step-ahead forecasts as well as one-shot 12-step-ahead (3 hour) forecasts as proof of concept.

N.B.: It is worth mentioning that in contrast to the ReLU activation function defined in Section 3.4.1.2 we will be utilizing the leaky ReLU activation function (illustrated in Figure 6.21) so as to avoid the "dying" ReLU problem

in which the ReLU neurons of a network always output a value of 0 thus effectively not contributing anything to the learning of the network.

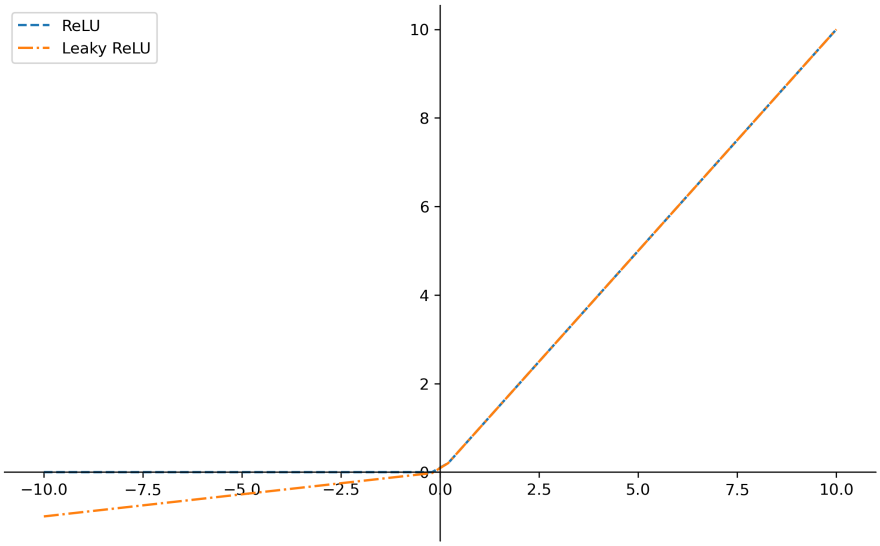


Figure 6.21: Illustration of leaky ReLU.

7

RESULTS AND DISCUSSION

Following the brief example demonstration of the proposed method in Chapter 6, we extended the implementation to house 12 of the REFIT data set. The subsequent sections serve to demonstrate the efficacy of both the classification step as well as the forecasting step of our proposed method.

7.1 CLUSTER LABEL CLASSIFICATION

The first results that we will be demonstrating are that of the classification step of our proposed method – refer to Table 7.1.

DATA SET	NO. OF CLUSTERS	ACCURACY
UCID	3	76%
REFIT - House 12	3	66%

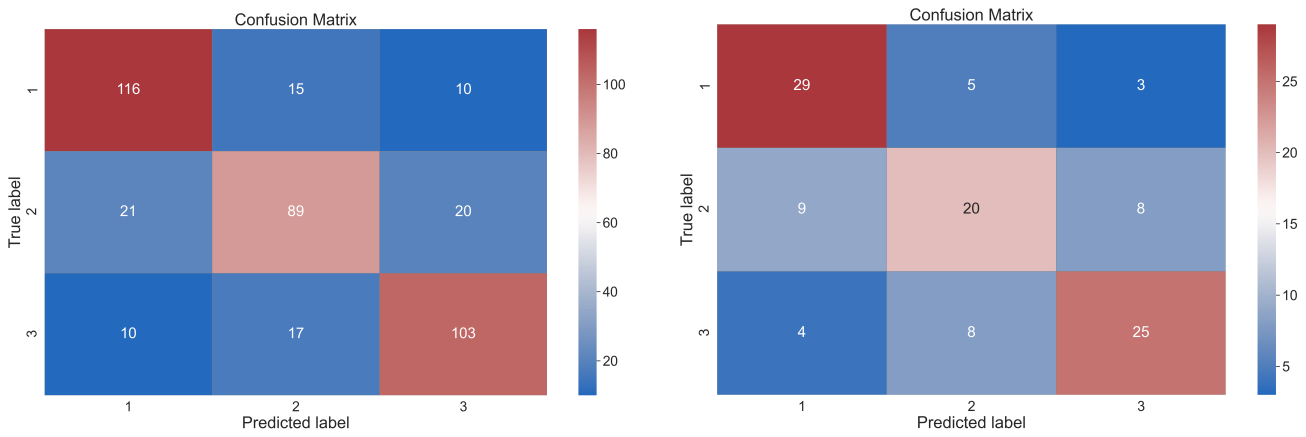
Table 7.1: Result of training, optimizing and evaluating a random forest classifier on the cluster labels obtained for each of the UCID as well as the REFIT data sets.

Being able to correctly assign new samples into the correct cluster is imperative so as to insure the highest likelihood of achieving consistently reliable forecasting accuracy. Given that we had an equal number of 3 clusters per data set and that we were working with a (synthetic) uniform distribution of samples over the different clusters; the scores outlined in Table 7.1 are fairly good (a random predictor would achieve an accuracy of 33.3%). The disparity in the results between the 2 data sets could predominantly be linked to the following 2 reasons:

1. The UCID data set contained a much larger number of samples (days).
2. The distribution of the samples over the different days of the week as well as the months is much more uniform in the UCID data set (refer to Figures 5.1 and 5.6 of Chapter 5).

Figures 7.1a and 7.1b allow us to clearly visualize both the correct as well as the incorrect predictions made by our model. Interestingly, given that both the clusters formed for each of the UCID as well as

the REFIT data set were quite similar in terms of the overall patterns that were captured, the fitted model per data set seems to be making mistakes, or otherwise incorrect predictions, of a similar magnitude with cluster 2 containing the largest amount of incorrect predictions for each of the data sets and cluster 1 containing the largest amount of correct predictions.



(a) Confusion matrix - UCID.

(b) Confusion matrix - REFIT.

Figure 7.1: Confusion matrices for each of the UCID as well as the REFIT data sets.

One of the benefits of our proposed method that we previously discussed is that no prior knowledge of the number of clusters is required; as there is no guarantee that any 2 individual households contain a similar number of *repeating* patterns we avoid running into the problem of overly generalizing a single working solution that may or may not work given said change in energy consumption patterns and instead present a solution that could potentially extend to a much larger scale. A potential issue with this implementation however, is that an individual household *may* contain a large number of repeating consumption patterns which could possibly lead to an overall decline in what can already be considered sub-par performance from our classifier. That said, there is definitely room for improvement that could accommodate these potential risks, specifically with regards to the feature engineering step and this will be discussed in Chapter 8 of this paper. Finally, regardless of the fact that the performance of our forecasting model is the highlight of this paper, it is interesting to note that a byproduct of our proposed method is the potential to extract insights into variables that have an effect on the daily energy consumption patterns of unique households. A cursory glance at applying our method to a portion of the data at hand, as an example of the insights that we can obtain, shows us that some households have frequently occurring patterns that tend to deviate among the different days of the week while other households have an even bigger

separation across months of the year or even among meteorological factors such as the temperature or chance of rain.

7.2 FORECASTING ACCURACY

When compared to the current state-of-the-art in modern literature, particularly with regards to data available on hand pertaining to the UCID data set, our proposed method yields superior forecasting accuracy at variable resolution. Table 7.2 presents a performance comparison of common models discussed in the literature and our proposed method. We note that, at the time of writing, no published results attempting to forecast energy consumption on the REFIT data set could be found and thus, barre attempting to recreate the results ourselves, we have had to omit them from Table 7.2 for the time being.

DATA SET	METHOD	MAE (KW)	RMSE (KW)	MAPE
UCID	LSTM	0.62	0.86	51.45%
	CNN-LSTM	0.34	0.61	34.84%
	Proposed	0.11	0.16	18.13%
REFIT	LSTM	N/A	N/A	N/A
	CNN-LSTM	N/A	N/A	N/A
	Proposed	0.07	0.11	19.27%

Table 7.2: Performance comparison of different methods on each of UCID as well as the REFIT data set. Note that these results were obtained for one-step-ahead prediction at a resolution of 15 minutes over the raw data sets.

Another component that is frequently (attempted to be) forecasted in the literature is the trend component obtained as part of a time-series decomposition step that was previously discussed in Chapter 5. We attempted to tackle this problem ourselves and applied the proposed method to both the smoothed, trend component of the UCID data set as well as house 12 of the REFIT data set, the results of which can be seen in Table 7.3. We note that the results here are considerably good, achieving a MAPE of $\sim 2\%$ for both data sets.

N.B. we note that the results obtained as part of Section 7.2 are the averaged results obtained from training, optimizing and assessing multiple models, one for each of the respective clusters obtained as part of stage 2 of our proposed method. Furthermore, all results were obtained at a resampled resolution of 15 minutes per time-step; however, similar results have been observed for variable time resolutions (1 minute, 1 hour etc.)

DATA SET	MAE (KW)	RMSE (KW)	MAPE
UCID	0.01	0.01	1.43%
REFIT	0.01	0.01	2.3%

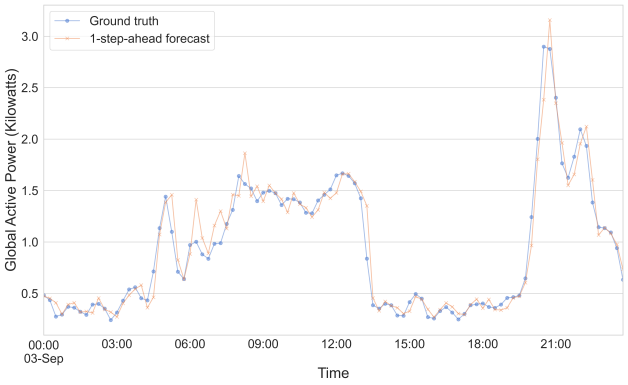
Table 7.3: Performance metrics obtained when applying the proposed method on the trend component of each of the UCID as well as the REFIT data sets to obtain a one-step-ahead prediction.

Finally, we attempted to extend our model by scaling up the number of predictions from a singular step (15 minutes into the future in this scenario) to a total of 12 sequential steps (leading to a grand total of 3 hours being forecasted given the previously mentioned step size of 15 minutes) the results of which can be seen in Table 7.4. Oddly enough, for both the UCID data set as well as house 12 of the REFIT data set, we achieved marginal improvements with regards to MAPE scores when attempting to build twelve-step-ahead forecasts on their respective trend components. On the other hand though, MAPE scores for the raw data for each of our data sets fell somewhat substantially, with an overall loss of about $\sim$ 10% when moving from one-step-ahead forecasts to twelve-step-ahead forecasts which is more in line with one could expect in this scenario.

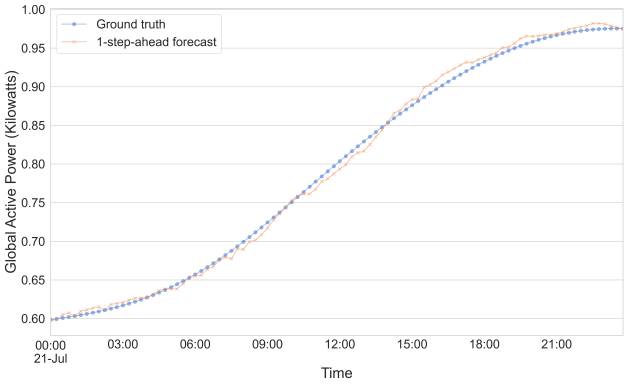
DATA SET	METHOD	MAE (KW)	RMSE (KW)	MAPE
UCID	Raw	0.31	0.45	29.84%
	Trend	0.01	0.01	1.2%
REFIT	Raw	0.12	0.23	36.55%
	Trend	0.01	0.01	2.2%

Table 7.4: Performance metrics obtained when applying the proposed method on both the raw data as well as trend component of each of the UCID as well as the REFIT data sets to obtain twelve-step-ahead predictions.

To further showcase, or otherwise visualize, the capabilities of our model we present Figures 7.2a, 7.2b, 7.3a and 7.3b that serve to illustrate one-step-ahead forecasts generated for a subset of each of the UCID data set as well as house 12 of the REFIT data set.

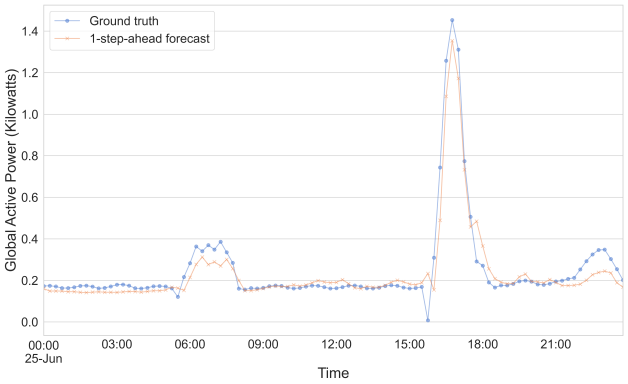


(a) UCID - Raw data.

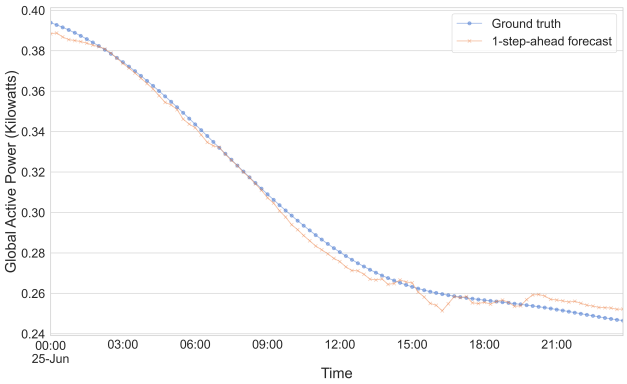


(b) UCID - Trend.

Figure 7.2: Showcasing the capabilities of the proposed method in making one-step-ahead predictions on the UCID data set.



(a) REFIT - Raw data.



(b) REFIT - Trend.

Figure 7.3: Showcasing the capabilities of the proposed method in making one-step-ahead predictions on the REFIT data set.

8

CONCLUSION AND FUTURE WORK

In this study, we have shown that the application of a clustering step that utilizes dimensionality reduction techniques such as t-SNE and hierarchical, density-based clustering in the form of HDBSCAN leads to significant improvements in forecasting accuracy when taking individual households into consideration. While this technique is certainly more complex, in particular with regards to the number of steps and moving parts associated with the entire pipeline, we maintain that the benefits in terms of improved forecasting accuracy outweigh the overall increase with regards to the time and effort it would take to train and set up such a model. The practicality of the model lies in the availability of the data that it requires to function – primarily with respect to historical energy consumption data for the individual households in question (which is becoming easier and easier to obtain thanks to the prevalence of smart meters) and meteorological data that can easily be obtained from numerous sources. Furthermore, it is highly likely that, given enough historical data, the need to further train the model(s) after the initial setup is rather low further compounding the efficacy of our proposed method.

With that said, at the time of writing and testing, the classification step of our pipeline is definitely lacking – an averaged accuracy of ~ 71, while not necessarily bad, is not anything to write home about and could cause issues down the line. Room for improvement lies both within the classifier used and the optimization process; however, we note that a lack of contextual information that serves to explain the emergence of the clusters as part of the clustering step could very well likely be the reason for obtaining sub-par accuracy scores. As it currently stands, the clustering step was built upon grouping together days that exhibited the highest similarity in terms of their energy consumption patterns. Given that this information is not available to us when considering a new day, we are left reaching for straws attempting to explain when any individual household is likely to observe energy consumption patterns that fall within any of the obtained clusters. Evidently, temporal and meteorological information is not enough to explain the emergence of said clusters and other information (perhaps patterns in terms of cluster labels leading up to the new sample) could serve to improve classifier accuracy. This is definitely an area of this study that could be looked into as part of future research.

On the other hand, were we to disregard the shortcomings of the clas-

sification step of the pipeline, the overall improvements in forecasting accuracy that our CNN-LSTM network managed to achieve over other models and setups available in the current literature. Given enough time, further improvements can probably be made given changes to the overall network architecture and hyperparameter optimization. Overall, initial results seem quite promising and pave the way for further improvements to be made down the line, both in terms of forecasting accuracy as well as the overall structure of the entire pipeline and, in general, with regards to elaboration and increased clarity over each of the steps undertaken to achieve said results.

Part IV

APPENDIX

A

APPENDIX

A.1 FIGURES

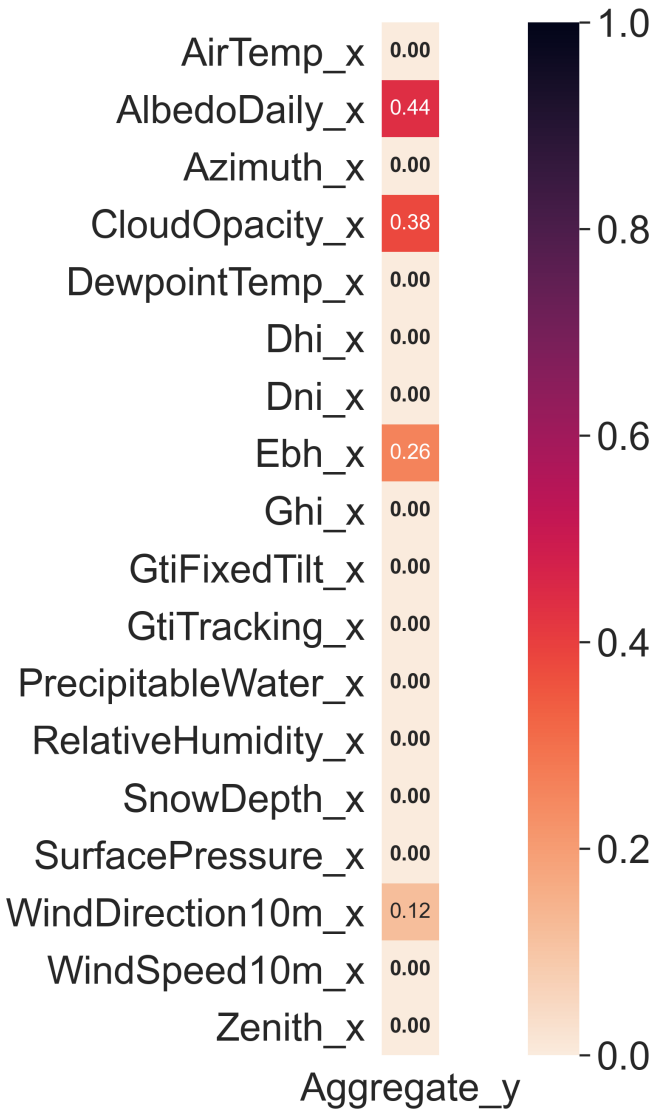


Figure A.1: A trimmed subset of the Granger Causation matrix present in Figure 5.4 that displays only the relevant information with regards to our independent variables causing our target variable.

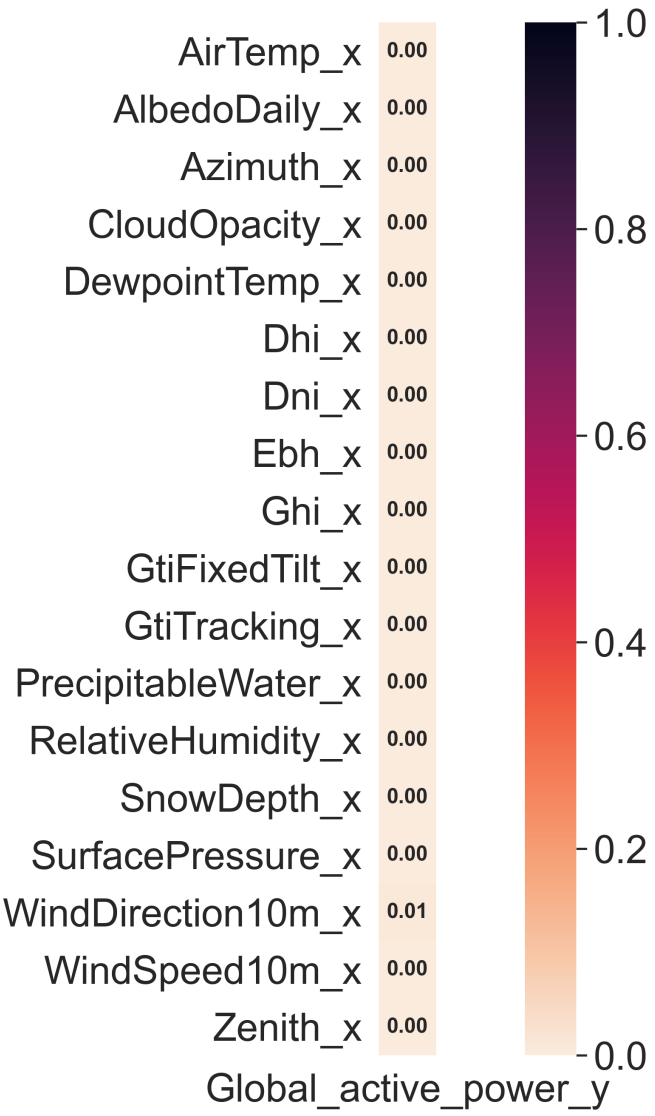


Figure A.2: A trimmed subset of the Granger Causation matrix present in Figure 5.8 that displays only the relevant information with regards to our independent variables causing our target variable.

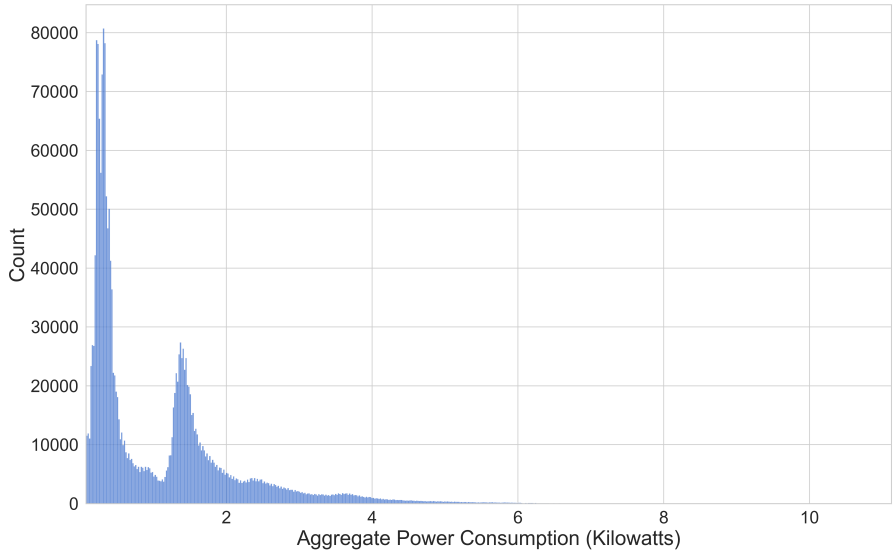


Figure A.3: Distribution of values with regards to our target variable.

A.2 TABLES

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 A.1: List of meteorological parameters available to us as per the Solcast data sets as outlined in Section 4.3.

VARIABLE	DESCRIPTION
Day	An integer value between 1 and 31.
Weekday	An integer value between 0 and 6 denoting the different days of the week.
Month	An integer value between 1 and 12.
Year	An integer value between 2007 and 2010.
Hour	An integer value between 0 and 23.
Minute	An integer value between 0 and 45 in increments of 15.
Season	An integer value between 0 and 3 where 0 denotes Spring, 1 denotes Summer, 2 denotes Fall and 3 denotes Winter.
Holiday	A categorical variable that takes on an integer value of 1 when the day concerned is a public holiday and 0 otherwise.

Table A.2: List of temporal variables that are taken into consideration during the feature engineering process as outlined in Section 6.3.

GLOSSARY

BEMS	Battery Energy Management System. 2, 5
CART	Classification and Regression Tree. 7, 9
CCF	Cluster-Classify-Forecast. 7
CILF	Computationally intelligent load forecasting. 5
CNN	Convolutional Neural Network. 20, 21, 61
CNN-LSTM	Convolutional Neural Network Long Short-Term Memory. x, 7–9, 20, 61, 71
DBSCAN	Density Based Spatial Clustering of Applications with Noise. iv, vii, xii, 6, 8, 9, 15–17, 19, 49
DNI	Direct Normal Irradiance. 76
DTW	Dynamic time warping. vii, 19, 20
EBH	Direct (Beam) Horizontal Irradiance. 40, 44, 76
EDA	Exploratory data analysis. 31, 41, 53
EPSRC	Engineering and Physical Sciences Research Council. 28
GA	Genetic algorithm. 8, 9
GHI	Global Horizontal Irradiance. 76
GTI	Global Torizontal Irradiance. 76
HDBSCAN	Hierarchical Density Based Spatial Clustering of Applications with Noise. vii–ix, 15, 17–19, 49–51, 70
HEMS	Home Energy Management System. vii, 2, 3, 5
IAM	Individual appliance monitor. xi, 28, 29, 31, 34, 35, 37, 38, 41
IQR	Interquartile range. 55
LOESS	Locally Weighted Scatterplot Smoothing. ix, 56
LSTM	Long Short-Term Memory. viii, 7–9, 20, 23–25, 61

MAE Mean absolute error. 26

MAPE Mean absolute percentage error. 7, 8, 62, 67, 68

MLP Multi-Layer Perceptron. 7, 9

NAN Not a number. 29, 48

NMAE Normalized mean absolute error. 8

NRMSE Normalized root mean square error. 8

PCA Principal Component Analysis. 11

REFIT Personalised Retrofit Decision Support Tools for UK Homes using Smart Home Technology. viii, x, xi, 28, 30–34, 36, 37, 41, 43, 44, 53, 54, 65–69

RELU Rectified Linear Unit. viii, x, 21, 22, 63, 64

RFECV Recursive Feature Elimination and Cross-Validation. x, 59

RNN Recurrent Neural Network. 7, 23

SGD Stochastic Gradient Descent. 62

SMBM Smart Meter Based Model. 7

SMOTE Synthetic Minority Oversampling Technique. ix, x, 57, 58

SOM Self-Organizing Map. 7–9

SVM Support Vector Machine. 7

SVR Support Vector Regression. 7, 9

T-SNE T-Distributed Stochastic Neighbor Embedding. viii, 11, 12, 14, 48, 49, 70

UCI University of California, Irvine. 28

UCID UCI data set. viii–xi, 28–30, 41–43, 48–51, 54, 55, 62, 65–69

UK United Kingdom. 29, 30

UMAP Uniform Manifold Approximation and Projection. 11, 14, 48

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] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE." In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[27] Leland McInnes, John Healy and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2020. arXiv: 1802.03426 [stat.ML].

[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/.

[38] Pasapitch Chujai, Nittaya Kerdprasop, and Kittisak Kerdprasop. "Time Series Analysis of Household Electric Consumption with ARIMA and ARMA Models." In: *Lecture Notes in Engineering and Computer Science* 2202 (Mar. 2013), pp. 295–300.

[39] C. W. J. Granger. "Investigating Causal Relations by Econometric Models and Cross-spectral Methods." In: *Econometrica* 37.3 (1969), pp. 424–438. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/1912791.

[40] N. V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique." In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357. DOI: 10.1613/jair.953.

[41] D.P. Kingma and J. Ba. "Adam: A Method For Stochastic Optimization." In: *ICLR* (2015).

DECLARATION

-

Groningen, The Netherlands, July 11, 2021

Kareem Al-Saudi