university of groningen

ENERGY CONSUMPTION PATTERNS AND LOAD FORECASTING WITH PROFILED CNN-LSTM NETWORKS

KAREEM AL-SAUDI

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

August 28, 2021 – 1.0

Kareem Al-Saudi, MSc. of Computing Science, © August 28, 2021

SUPERVISORS:
Viktoriya Degeler
Michel Medema

ABSTRACT

By virtue of the steady societal shift to smart technologies, built on the increasingly popular smart grid framework, we have noticed an increase in the need to analyze household electricity consumption at the individual level. In order to work efficiently, these technologies rely on load forecasting to be able to optimize operations that are related to energy consumption (such as household appliance scheduling). This paper proposes a novel load forecasting method that utilizes a clustering step prior to the forecasting step to group together days that exhibit similar patterns in energy consumption. Following that, we attempt to classify new days into one of the pre-generated clusters by making use of the available context information (day of the week, month, predicted weather). Finally, using available historical data (with regards to energy consumption) alongside meteorological and temporal variables, we train a Convolutional Neural Network Long Short-Term Memory (CNN-LSTM) model on a per-cluster basis that each specializes in forecasting based on the energy profiles present within each cluster. This method leads to improvements in forecasting performance (upwards of a 10% increase in mean absolute percentage error (MAPE) scores) and provides us with the added benefit of being able to easily highlight and extract information that allows us to assess which external variables have an effect on the energy consumption of any individual household.

CONTENTS

III EMPIRICAL STUDY

IV APPENDIX

LIST OF FIGURES

LIST OF TABLES

LISTINGS

Part I

INTRODUCTION

INTRODUCTION

Over the years, our reliance on electrical appliances has been slowly increasing (as shown in Figure 1.1). As our dependence on electrical appliances increases, so too does our consumption of energy [1, 2] and, subsequently, our need for more sophisticated and advanced solutions that can accommodate this growth. Thankfully, the convergence of multiple technologies – such as 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* and *smart meters* that are slowly but surely taking off in terms of their popularity and availability [3]. 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 basis [4], and, through the utilization of applications built upon this framework, we are capable of achieving an overall reduction in terms of the amount of energy that 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 [5].

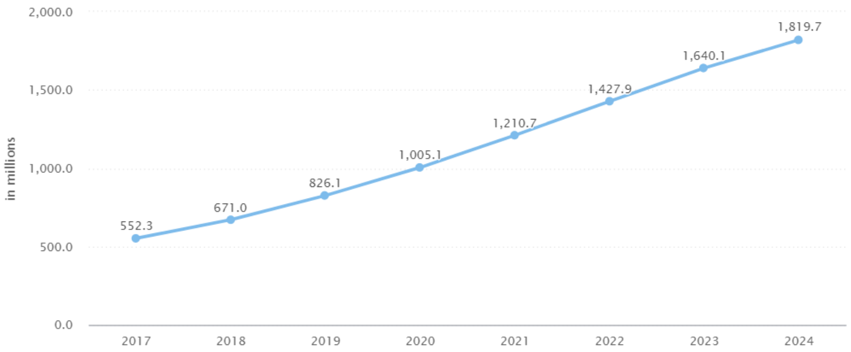


Figure 1.1: Historical/predicted growth in the number of appliances being used worldwide. Image source: [6] © 2019, Statista.

Existing solutions developed under the increasingly popular smart grid framework, 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 alongside any other personal preferences [4]. To operate efficiently, these solutions rely on our ability to capably forecast future trends in energy consumption at the individual household level. This information is required to appropriately and sufficiently control and supply the correct energy load to the end-user [7, 8]. This has lead to a shift in interest within the realm of load forecasting, in which prior research has predominantly been focused at the large-scale, regional level [9] where an amalgamation of available data spanning numerous households provides more obvious patterns as a result of the underlying diversity between households being lost [10] towards the individual household level. Furthermore, owing to the operational characteristics of both HEMS and BEMS and similar applications, load forecasting in the very short term (anywhere from a few minutes to a couple of hours), oftentimes referred to as very short-term load forecasting (VSTLF), are more relevant than the substantially studied longer term horizons that are predominantly associated with long-term network planning and operations [4].

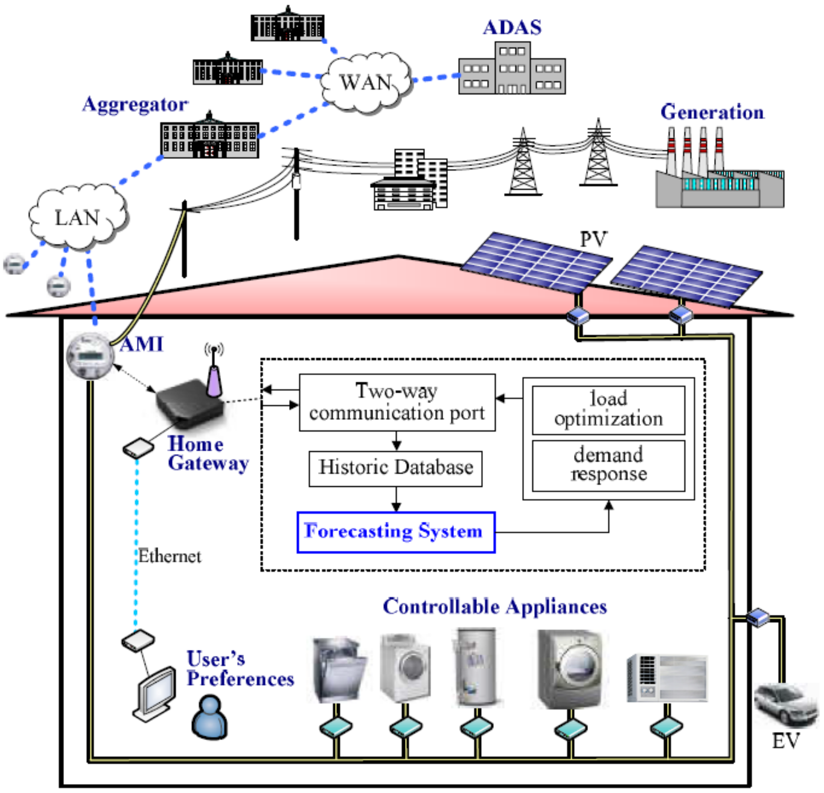


Figure 1.2: The HEMS architecture visualized. Image source: [11] © 2013, IEEE.

When exploring energy consumption at the individual household level, the diversity and complexity associated with human behavior 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 benefit from a precursory clustering step within the forecasting pipeline [4, 5, 10]. This precursory clustering step serves to amalgamate days that exhibit a measure of similarity in terms of their energy consumption patterns into the same cluster. By training individual forecasting models on a per-cluster basis we should, in theory, see an improvement in load forecasting performance as each of the respective models specializes in predicting future trends in energy consumption based on patterns present within the energy profile associated with its unique cluster. This is the area of research that this paper seeks to tackle – how can we best construct energy profiles out of historical data that truly capture repeated patterns with regards to energy consumption and what are the effects of a clustering step in the performance of a forecasting pipeline.

The following Chapters of this paper are organized as follows: Chapter 2 presents 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 important concepts that are related to this paper. Chapter 4 will both describe as well as visualize the historical data that we have on hand, outlined in Section 1.1, for the duration of this project. Ensuingly, Chapter 5 will outline our methodology with regards to both our chosen clustering as well as forecasting techniques. Finally Chapters 6 and 7 conclude the paper by presenting our results alongside a discussion and potential direction with regards to future work.

1.1 INTRODUCTION TO THE DATA

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)*" [12] which is a collaboration among the Universities of Strathclyde, Loughborough and East Anglia, as well as the *"Individual Household Electric Power Consumption"* data set [13] 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 Chapter 4 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.

1.1.1 *REFIT*

The REFIT Electrical Load Measurements data set [12] includes cleaned electrical consumption data, in watts, for a total of 20 households labeled *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 labeled 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 makes and models 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 synchronized 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 zeros.

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.

1.1.2 *UCID*

The UCID data set [13] 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.

1.1.3 *Meteorological Data*

As an addendum to both the REFIT and UCID data sets we will be incorporating meteorological data as provided by Solcast [14], a company based in Australia that aims to provide high quality and easily-accessible solar data. For the purpose of this master's thesis project we requested 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 B.3 which is located in Appendix B of this paper.

1.2 PROPOSED MODEL

To attempt to solve the previously outlined problem of VSTLF forecasting at an individual household level we propose a novel solution that utilizes a combination of statistical knowledge and machine learning techniques to both generate energy profiles that provide us with some measure of insight as to the habits of a household's occupants as well as forecast future trends in their energy consumption. A high level overview of the steps relevant to our proposed model can be seen in Figure 1.3.

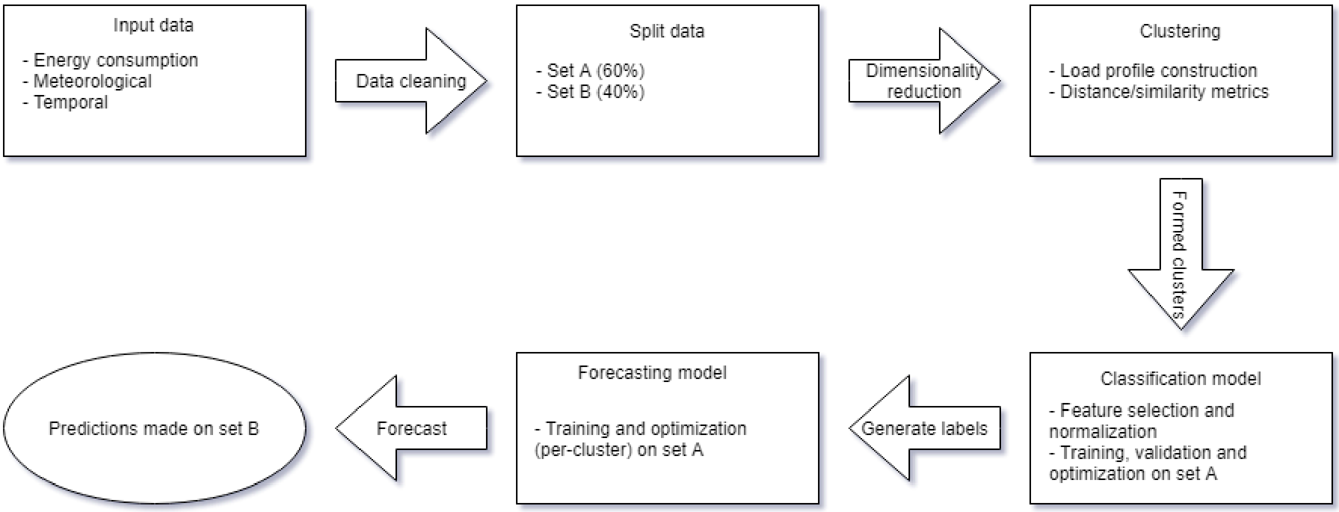

Input data
- Energy consumption
- Meteorological
- Temporal
Data cleaning
Split data
- Set A (60%)
- Set B (40%)
Dimensionality reduction
Clustering
- Load profile construction
- Distance/similarity metrics
Formed clusters
Classification model
- Feature selection and normalization
- Training, validation and optimization on set A
Generate labels
Forecasting model
- Training and optimization (per-cluster) on set A
Forecast
Predictions made on set B

Figure 1.3: High level overview of the steps pertaining to our model.

In short, we devised a method that consists of 3 steps: cluster, classify, forecast – first we cluster historical days based on similarity in terms of active power consumption, then we classify new days into one of the generated clusters and finally we generate forecasts based on models that are trained on a per-cluster basis. These steps, alongside a working example, will be discussed in-depth in Chapter 5.

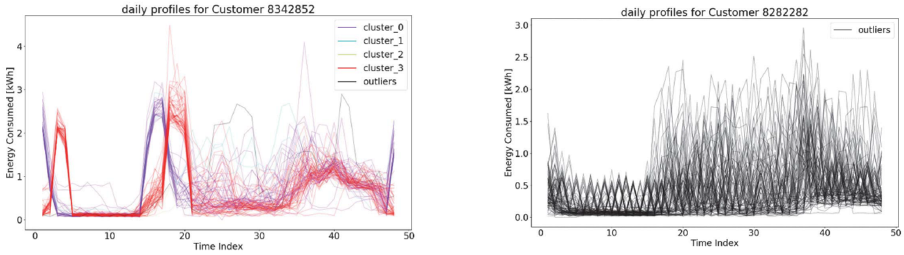
2 RELATED WORK

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 [15]. 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 position our paper with respect to the current state-of-the-art and highlight the key differences in our approach.

2.1 CLUSTERING AND ENERGY PROFILE CREATION

The main 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 [16] 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.

Kong et al. [10] attempted to justify the observations made by Stephen et al. [17] by using a density-based clustering technique known as Density Based Spatial Clustering of Applications with Noise (DBSCAN) [18] 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. [4] expand on traditional load forecasting techniques, such as the Smart Meter Based Model (SMBM) that they had previously presented [19], 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) [20] 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 [21].

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 [22] in the form of multiple linear regression to methods such as novel deep pooling Recurrent Neural Networks

(RNN) introduced by Shi, Xu, and Li [23]. 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 [15] 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. [4] 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 [21] 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 [24] 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. [25]. Furthermore, Kim and Cho [24] 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 5, will be based on these concepts.

CATEGORY	AUTHOR(S)	YEAR	METHOD(S)
Statistical based	Fumo and Rafe Biswas [22]	2015	Linear regression
	Amber, Aslam, and Hussain [26]	2015	GA & Multiple regression
Machine learning based	Lamedica et al. [27]	1996	SOM & MLP
	Yildiz et al. [4]	2018	SOM, K-means, CART & SVR
Deep learning based	Kong et al. [10]	2019	DBSCAN & LSTM network
	Kim and Cho [24]	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 serves predominantly to explain concepts and methods relevant to the research. This includes dimensionality reduction techniques, such as t-Distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP), as well as our chosen clustering and forecasting techniques (Hierarchical Density Based Spatial Clustering of Applications with Noise (HDBSCAN) and Convolutional Neural Network Long Short-Term Memory (CNN-LSTM) respectively). 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 [28], 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}:

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

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:

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

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".

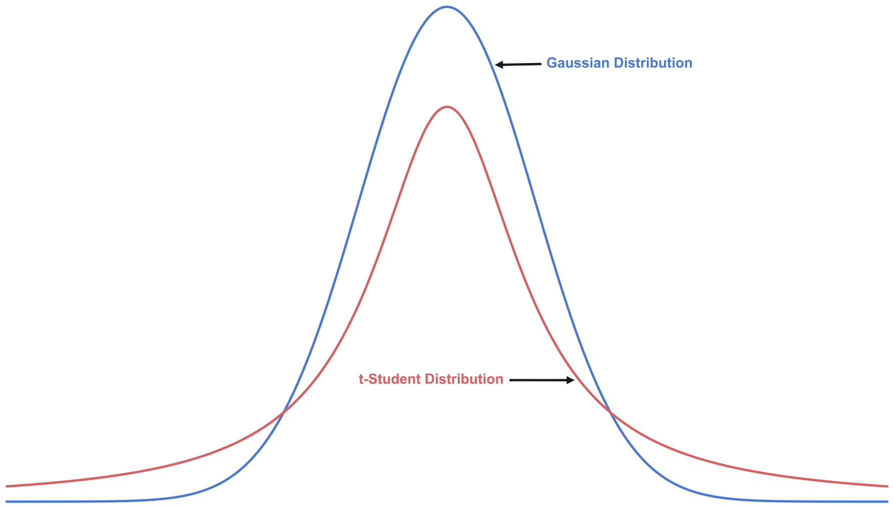


Figure 3.1: Overlapping both a Gaussian distribution and the t-Student, or Cauchy, distribution. Note the heavy tails on the t-Student distribution.

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.

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 [29], 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 (e. g., taking a mere 3 minutes to project the 784-dimensional, 70,000 point MNIST data set in contrast to the 45 minutes it would take for the t-SNE algorithm [30]). 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 [29] to better understand the core concepts of the algorithm as well as what differentiates it from the t-SNE algorithm.

At its core, UMAP works very similarly to the t-SNE algorithm. The biggest difference between the output of the UMAP algorithm in comparison to the output of the t-SNE algorithm is the balance between local as well as global structure. Given that the authors of the UMAP algorithm claim that it is often better at preserving global structure in the final projection in comparison to the t-SNE algorithm, it is safe to assume that inter-cluster relations are probably more meaningful than the projections made by the t-SNE algorithm. This preservation of global structure when running both algorithms under a similar set of hyperparameters can be seen in Figure 3.2

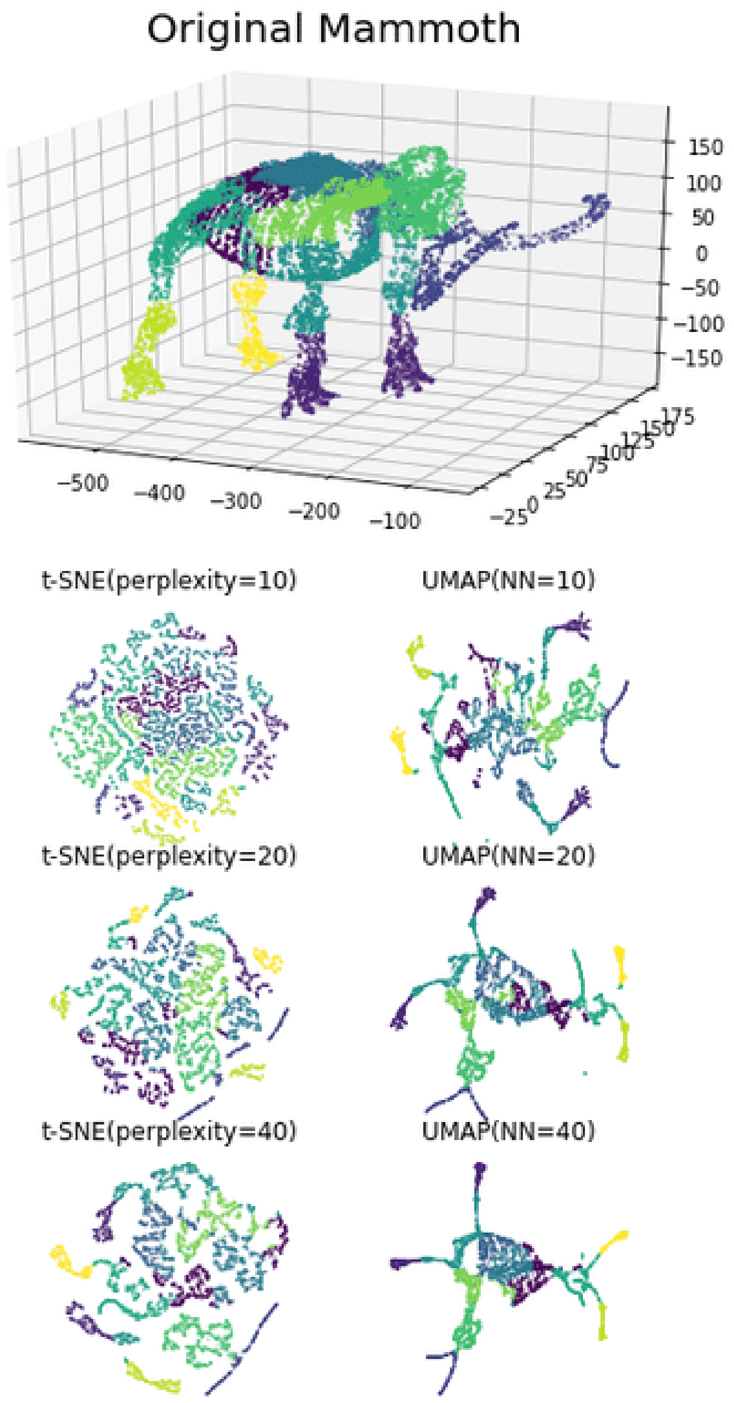


Figure 3.2: Illustrating how the UMAP algorithm captures more of the global structure in a sample data set (Mammoth data set). Image source: [31], licensed under CC BY-SA 4.0.

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. [18], 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 labeled 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.

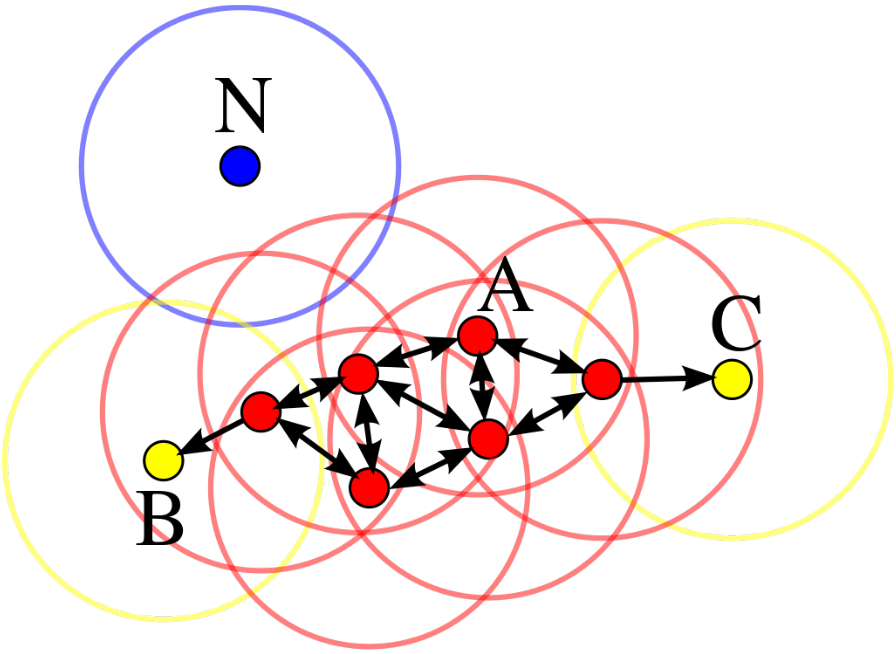
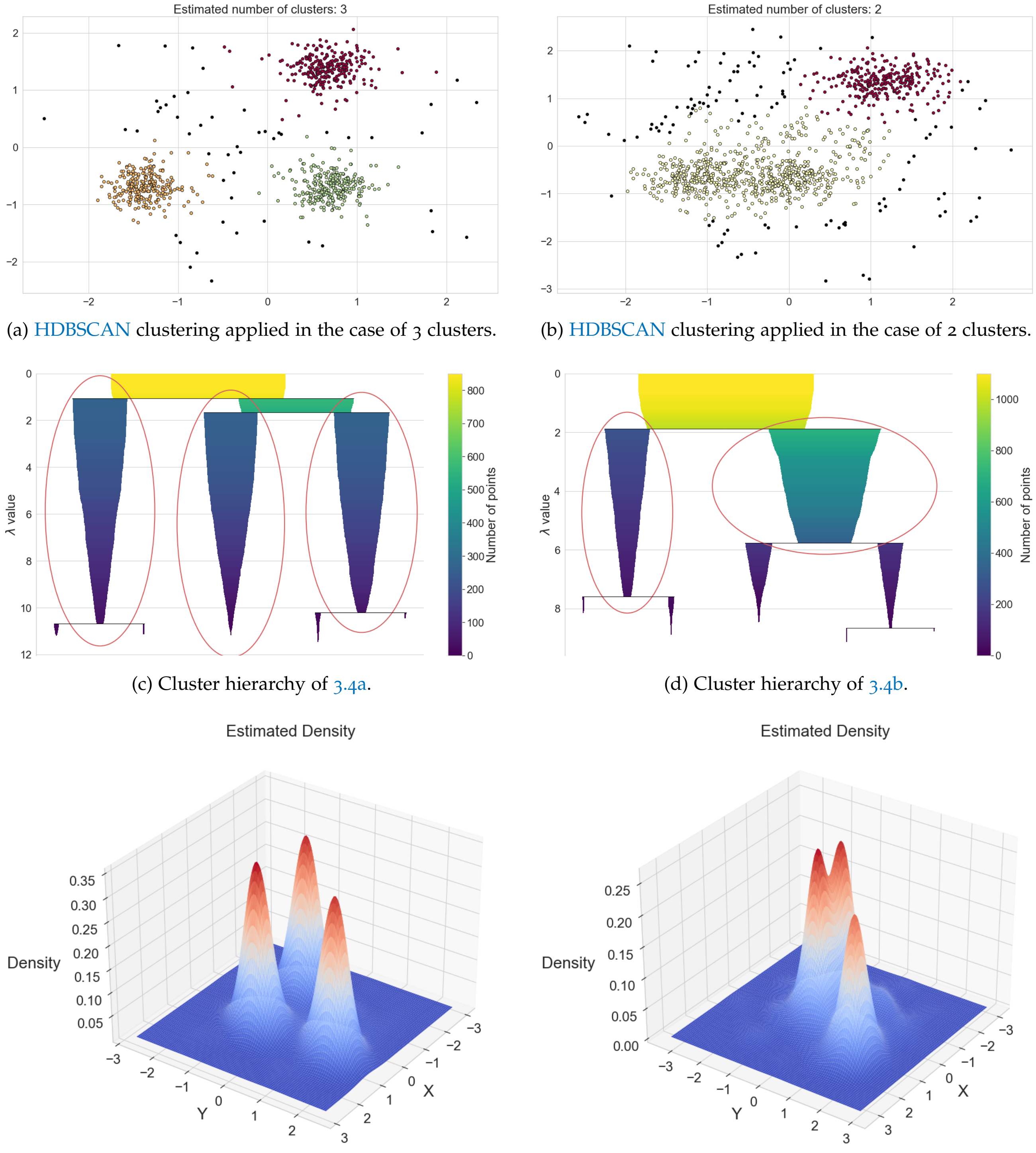


Figure 3.3: Depiction of the DBSCAN algorithm at work with minPts set to 4. Here, point A, as well as the other red points, are core points as the area surrounding these points in a radius ϵ contains at least 4 points (including the point itself). Points B and C are reachable from A through other core 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: [32], 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 [33], 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 [33] and leave off with Figure 3.4 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 3.4a.

(d) Cluster hierarchy of 3.4b.

(e) Density landscape of 3.4a.

(f) Density landscape of 3.4b.

Figure 3.4: 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 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.

3.3.1 *Convolutional Neural Networks*

Convolutional Neural Networks (CNNs) first truly started gaining traction in the 1990s when Lecun et al. [34] 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 [35], and when working with time series data sets [36]. The following sections will briefly outline key points of the inner machinations of key CNN components.

3.3.1.1 *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*.

3.3.1.2 *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:

$$R(z) = \max(0, z) \tag{3.7}$$

The purpose of the ReLU operation is to replace all negative values in the feature map by zero. This nonlinear function allows for the

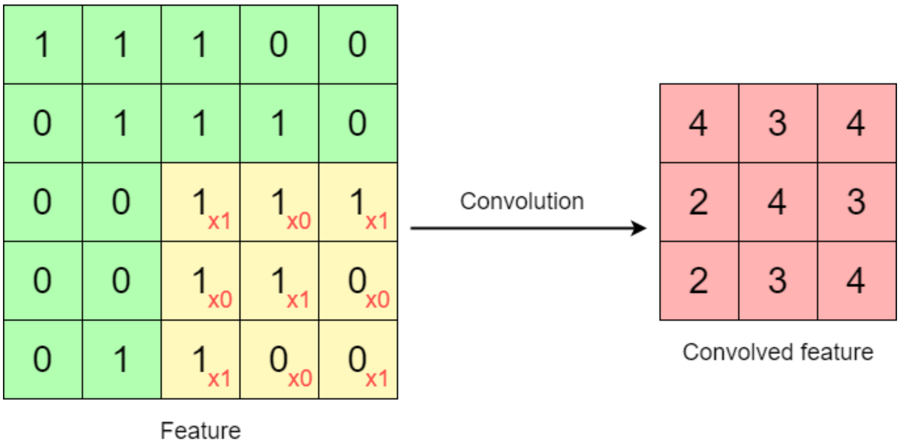


Figure 3.5: An example of a convolutional kernel at work. A 3x3 kernel traverses over a 5x5 "image" with a stride of 1 to produce the convolved feature map.

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.

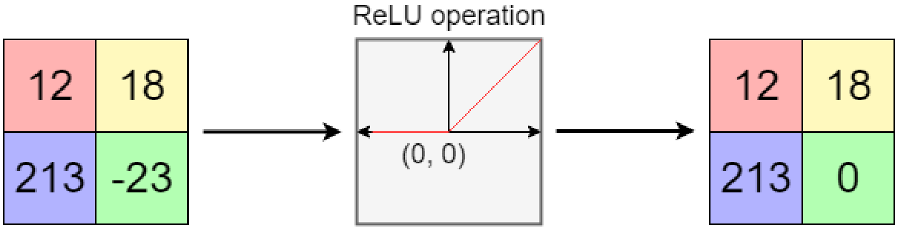


Figure 3.6: A simplified demonstration of a ReLU operation.

It is worth mentioning that in contrast to the ReLU activation function we will be utilizing the leaky ReLU activation function (illustrated in Figure 3.7) 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.

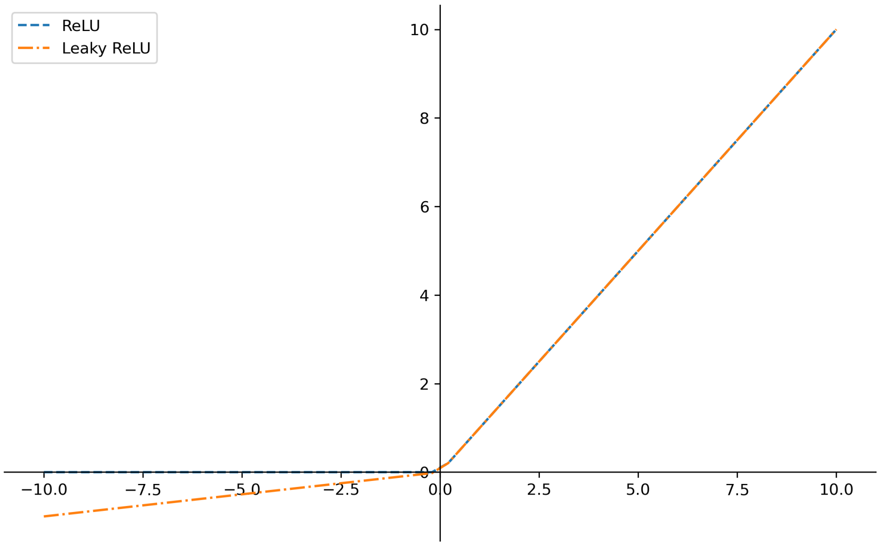


Figure 3.7: Illustration of leaky ReLU.

3.3.1.3 *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.

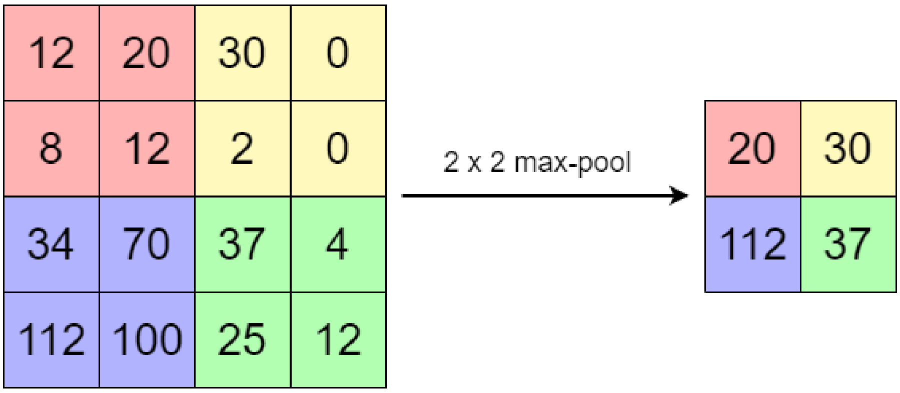


Figure 3.8: An example of max pooling using a 2x2 window.

3.3.2 *Long Short-Term Memory Networks*

Long Short-Term Memory (LSTM) networks, first proposed by Hochreiter & Schmidhuber in 1997 [37], 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.

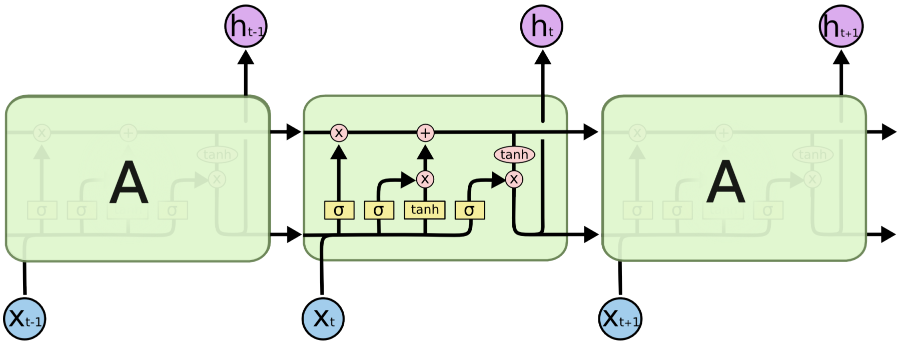


Figure 3.9: The repeating module in an LSTM that contains four interacting layers. Image source: [38] (with permission from the author).

3.3.2.1 *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 3.8.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.8}$$

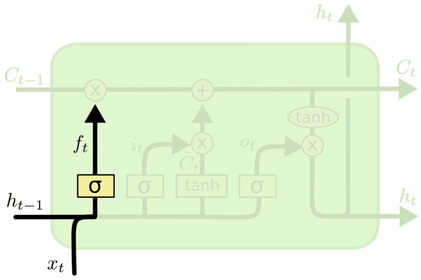


Figure 3.10: An illustration of the forget gate in an LSTM network. Image source: [38] (with permission from the author).

3.3.2.2 *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 3.9.

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \end{aligned} \tag{3.9}$$

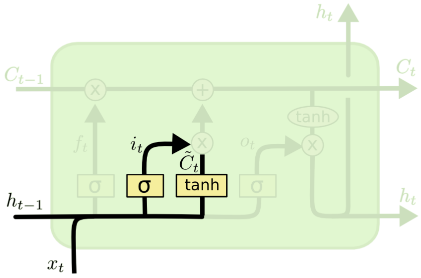


Figure 3.11: An illustration of the input gate in an LSTM network. Image source: [38] (with permission from the author).

3.3.2.3 *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 3.10.

$$\begin{aligned} C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\ o_t &= \sigma(W_o\,[h_{t-1}, x_t] + b_o) \\ h_t &= o_t * tanh(C_t) \end{aligned} \tag{3.10}$$

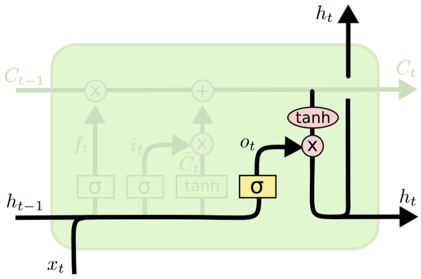


Figure 3.12: An illustration of the output gate in an LSTM network. Image source: [38] (with permission from the author).

3.4 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.

3.4.1 *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:

$$\text{MAE} = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{n} \tag{3.11}$$

where:

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

3.4.2 *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:

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{\hat{y}_i - y_i}{y_i} \right| \tag{3.12}$$

where:

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

3.4.3 *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:

$$\text{Log-Cosh L} = \sum_{i=1}^{n} \log(\cosh{(\hat{y}_i - y_i)}) \tag{3.13}$$

where:

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

y_i = actual value.

n = total number of data points.

4 EXPLORATORY DATA ANALYSIS

Before we can get into the details of our 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 applying and discussing the use of a variety of visualization techniques and statistical tests on the REFIT data set. In order to maintain the length and readability of the overall paper, the relevant Figures and Tables for the UCID data set will be shifted to the Appendix.

4.1 ISSUES

Given that the REFIT 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, or in other words, the least amount 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 labeled 'issue' by the data collection team.

4.1.1 *The 'Issues' Column*

The first issue that we will be tackling is the aptly named *Issues* column. As previously mentioned in section 1.1.1, the data collection team responsible for the curation of the REFIT data set appended an *Issues* column to the data set so as to indicate whether the sample being recorded either contains no issues and can be treated normally, given a recorded value of 0, or whether 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. Table 4.1 outlines the total number of recorded values alongside the number of values with the *Issues* column set to 1 (i. e., 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 4.1: 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.

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.

4.1.2 *Missing & Incomplete Data*

The second issue that we will be tackling is that of missing or otherwise incomplete data. Here, we refer to missing data as any dates that fall within the period of recorded data but for which we have no recorded values while incomplete data refers to any days that contain less than 96 readings when considering a resolution of 15 minutes. The results of our analysis can be seen in Table 4.2 which also includes a column that indicates the longest period of consecutive missing days.

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 4.2: 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.

We note that the earlier households, in order of numbering (houses 1 through 10), tend to contain a larger range of dates recorded and, subsequently, also tend to contain a larger number of missing days as well as 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 etc.

4.1.3 *Thresholding*

To wrap up Section 4.1, we will be selecting a single household with which to continue our EDA based on the analysis conducted in sections 4.1.1 and 4.1.2. This selection 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 to render it feasible to work with. We score each household by taking the normalized (between a range of 0 → 1) mean of the number of incomplete days, missing days and values with issues and subtracting the obtained value from 1. The results of this can be seen in Table 4.3.

HOUSE NO.	SCORE
20	0.98
12	0.92
19	0.91
11	0.89
17	0.87

Table 4.3: Weighted scores for the top 5 scoring households in the REFIT data set.

Given that, the top candidates are households 20, 12 and 19 as they make up the days that contain the least number of issues. We arbitrarily narrow our choice down to house number 12 although realistically, any of the remaining candidates would work just as fine and can be used to ascertain the findings within the scope of the entire project.

4.2 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 and these will be presented in Sections 4.2.1 and 4.2.2.

4.2.1 *Sample Distribution*

Figures 4.1 and 4.2 serve to provide an overview of the distribution of samples over the days of the week as well as the months of the year. Noting the distribution of our samples over these different criteria is essential when considering the results of our clustering algorithm as we will be attempting to classify new samples into the generated clusters based on these temporal variables later on in this project..

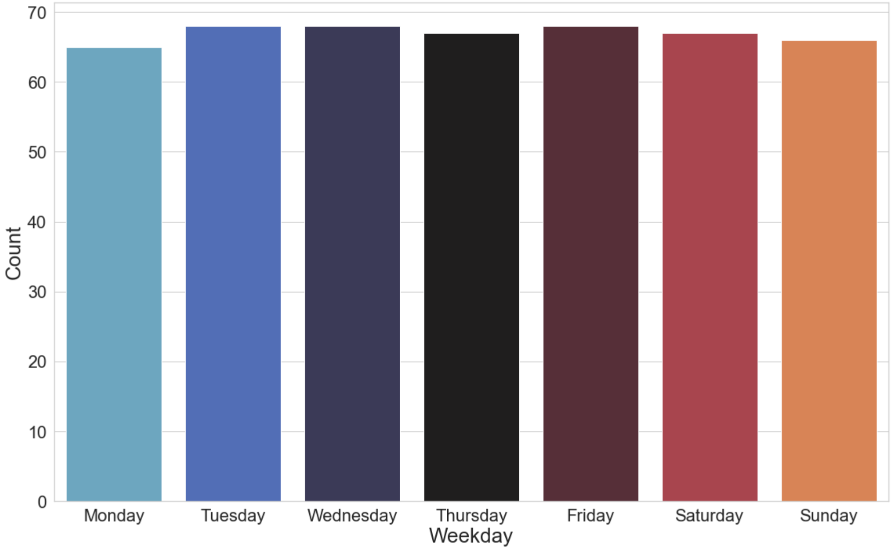


Figure 4.1: Number of samples per day of the week over the entirety of the data set. Data for this plot was pulled from CLEAN_House12.csv of the REFIT data set.

At a glance, we note that the distribution of the samples over the days of the week is relatively even with no one day containing a much larger number of samples than the other. The distribution of the samples over the months, on the other hand, is heavily dominated by the months of March through June and, to a lesser extent, July. When inspecting the results of our clustering algorithm, 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.

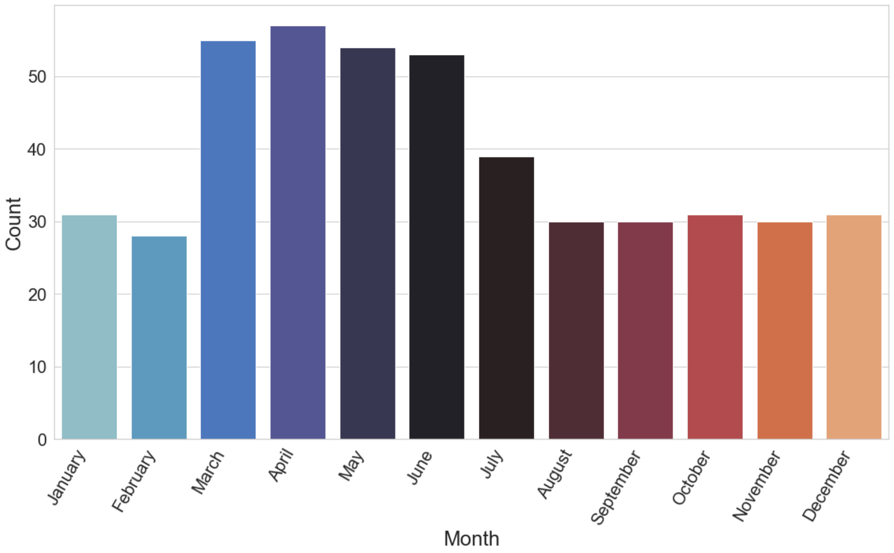


Figure 4.2: Number of samples per month over the entirety of the data set. Data for this plot was pulled from CLEAN_House12.csv of the REFIT data set.

N.B. We note that the plots in Figures 4.1 and 4.2 represent our data set after removing days that contain an incomplete number of values.

4.2.2 *Time Series Decomposition*

The decomposition of a time series is a statistical task that deconstructs it into three principle components: trend, seasonality and noise.

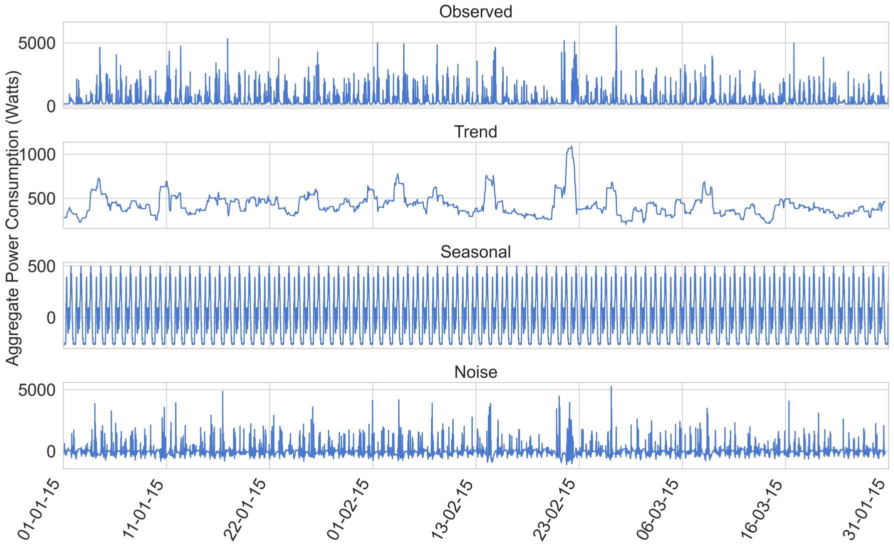


Figure 4.3: Time series decomposition. Data for these plots were pulled over a 3 month period that was resampled into a resolution of 15 minutes from CLEAN_House12.csv of the REFIT data set.

Figure 4.3 depicts the result obtained when performing additive time series decomposition on our observed electric energy consumption data [39]. Our reasoning for selecting an additive model is due to the fact that our data is stationary (i. e., no sharp increase or decrease in trend over time). Alongside the raw data, we will also be attempting to train models that learn to forecast future values for the *trend* component of this decomposition as it captures the main essence of the energy consumption patterns present in the individual household(s) that we are exploring.

4.3 causality & correlation

Given the substantial number of features or, in other words, independent variables (both temporal and meteorological) 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. In this case, our target variable is the aggregate power consumption, or global active 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.

4.3.1 *Granger Causality Test*

The first of these tests that we will be performing is the Granger Causality test. First proposed in 1969 by Granger [40], 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 A.5 and 4.4 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, as shown in Table 4.4, are that each of our independent variables are indeed stationary and so we can reaffirm the plausibility of the results obtained from performing the Granger Causality test.

FEATURE	P-VALUE	STATIONARY
AirTemp	1.68e-05	True
AlbedoDaily	3.22e-19	True
Azimuth	0.0	True
CloudOpacity	0.0	True
DewpointTemp	4.02e-12	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	1.21e-22	True
RelativeHumidity	1.85e-21	True
SnowDepth	8.96e-05	True
SurfacePressure	6.43e-14	True
WindDirection10m	6.75e-23	True
WindSpeed10m	3.02e-22	True
Zenith	0.02	True
Aggregate	0.0	True

Table 4.4: The results of performing the Augmented Dicky-Fuller test on our target variable as well as the meteorological variables introduced in Section 1.1.3 and outlined in Table B.3.

In the output of the Granger Causality test we performed, as seen in Figure A.5, 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 among 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.

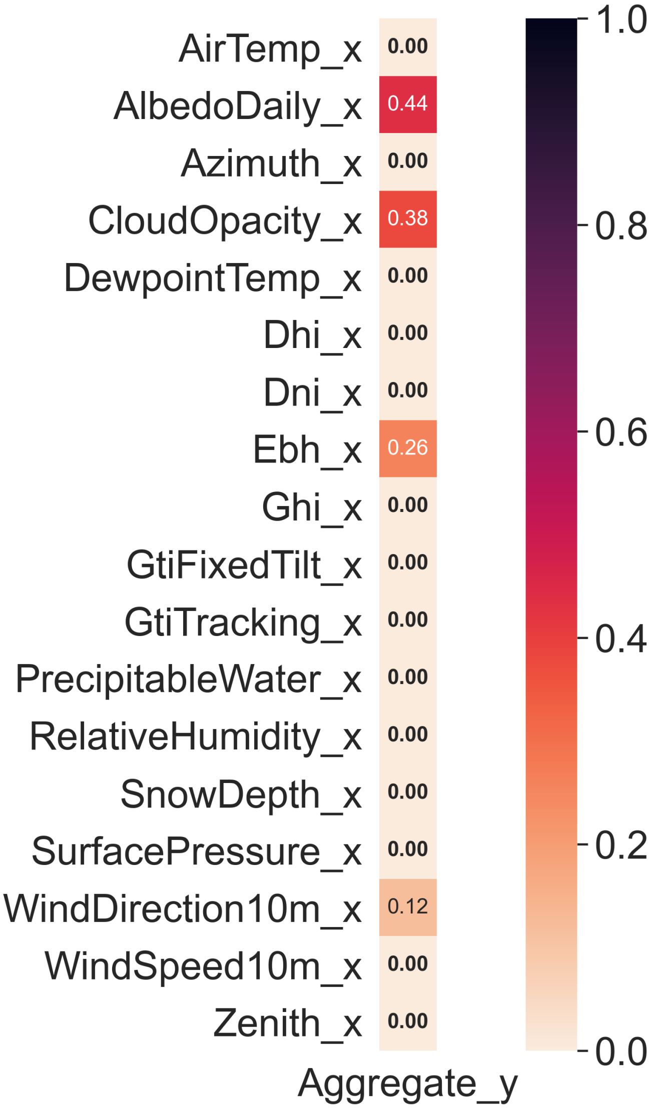


Figure 4.4: A trimmed subset of the Granger Causation matrix (Figure A.5) that displays only the relevant information with regards to our independent variables causing our target variable.

The complete Granger Causation Matrix is located in the Appendix (Figure A.5)

4.3.2 *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 4.5. The results seen in Figure 4.5 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.

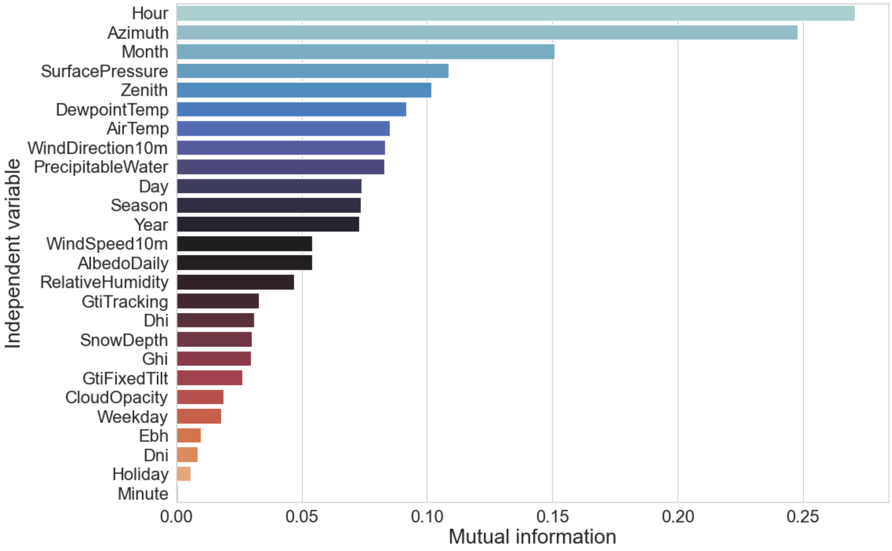


Figure 4.5: Mutual information of our independent variables against our target variable.

Part III

EMPIRICAL STUDY

5 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 behavior. 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 behavior. 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. A detailed outline of the proposed model, first introduced in Section 1.2, can be seen in Figure 5.1.

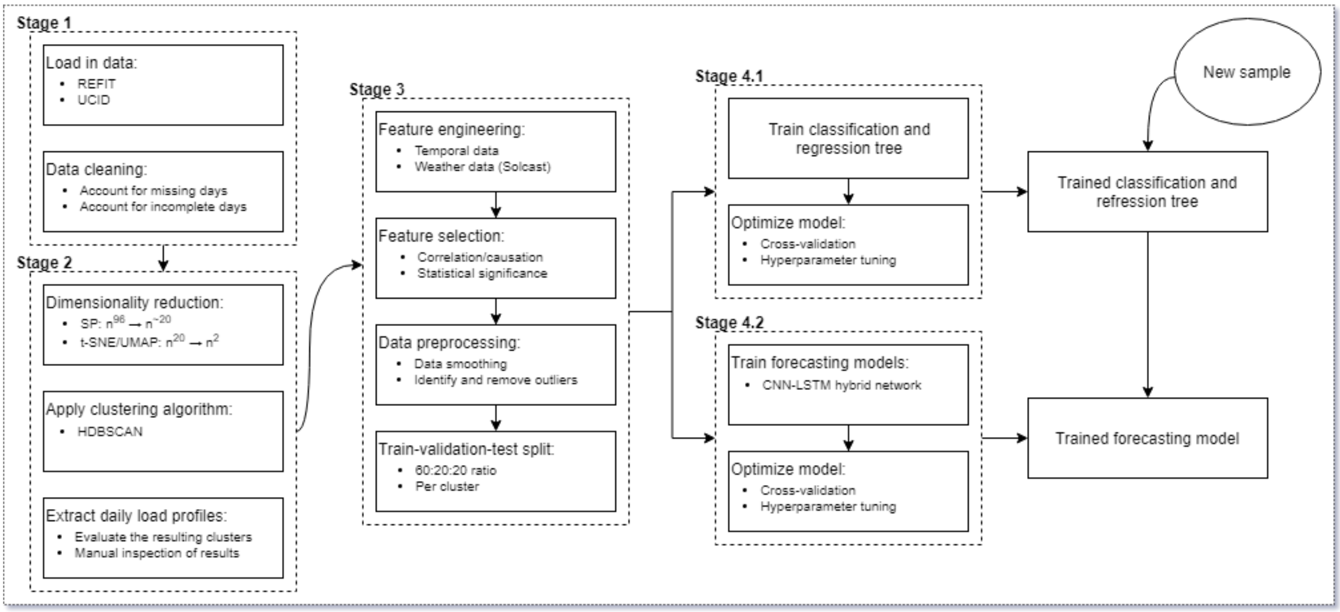

Stage 1
Load in data:
REFIT
UCID
Data cleaning:
Account for missing days
Account for incomplete days
Stage 2
Dimensionality reduction:
SP: $n^{96} \rightarrow n^{\sim 20}$
t-SNE/UMAP: $n^{20} \rightarrow n^{2}$
Apply clustering algorithm:
HDBSCAN
Extract daily load profiles:
Evaluate the resulting clusters
Manual inspection of results
Stage 3
Feature engineering:
Temporal data
Weather data (Solcast)
Feature selection:
Correlation/causation
Statistical significance
Data preprocessing:
Data smoothing
Identify and remove outliers
Train-validation-test split:
60:20:20 ratio
Per cluster
Stage 4.1
Train classification and regression tree
Optimize model:
Cross-validation
Hyperparameter tuning
Stage 4.2
Train forecasting models:
CNN-LSTM hybrid network
Optimize model:
Cross-validation
Hyperparameter tuning
New sample
Trained classification and refression tree
Trained forecasting model

Figure 5.1: Proposed daily profile extraction and load forecasting model.

In short, we start off by resampling the data present in both the REFIT data set as well as the UCID into a common resolution in order be able to directly compare results. In this case we will be resampling both data sets to a common resolution of 15 minutes per sample as this lines up well with the native resolution of the meteorological data we have on hand that was provided to us by Solcast. Following this, we clean both data sets and rid them of any days that contain an incomplete number of records (incomplete here referring to any days that contain less than 96 records given we split each day into a total of 96 chunks). After this, we take a subset of each of our data sets (60% of the total data for each of the REFIT as well as the UCID data sets henceforth referred to as *Set A*) and leave out the remaining 40% (henceforth referred to as *Set B*) in order to validate the results of our forecasting model. We then reduce the overall dimensionality of a single day, going from a total of 96 features to a much more manageable 2 through a combination of statistical and machine learning techniques and generate clusters based on the new, 2-dimensional data set by utilizing a density-based clustering algorithm. Following this, we train and optimize a classifier on Set A and use it to generate cluster labels for the previously withheld Set B. Finally, we train our forecasting models, one per cluster, on the data present in Set A and use the data present in Set B to act as both a validation set as well as a training set with which to obtain final results. In this Chapter, we will provide a working example alongside in-depth explanations of each of the steps previously discussed that make up our overall forecasting pipeline.

5.1 stage 1 - data collection and cleaning

Step 1.1: As mentioned previously, 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.

5.2 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 behavior. This *dimensionality reduction* step comprises 2 parts that are outlined in the sub-steps below.

To start things off we divide each day into 5 different periods (as per the work of Yildiz et al. [4]) 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 now begin a series of visualizations showcasing each step as performed on the UCID data set throughout the entirety of stage 2 as well as the remainder of the stages that make up our overall forecasting pipeline. We start off by presenting a scatter plot of the 2-dimensional output obtained as a result of performing the t-SNE algorithm which can be seen in Figure 5.2 that allows us to clearly visualize the 2-dimensional interpretation of the samples present in the UCID data set. Each of the points found on the 2-dimensional surface in Figure 5.2 represents a single day, and given that the UMAP algorithm claims to preserve both local as well as most of the global structure present in the data we can safely assume that distances between the samples are conducive to the similarity in terms of energy consumption as per the previously segmented (into various periods) interpretation of the data.

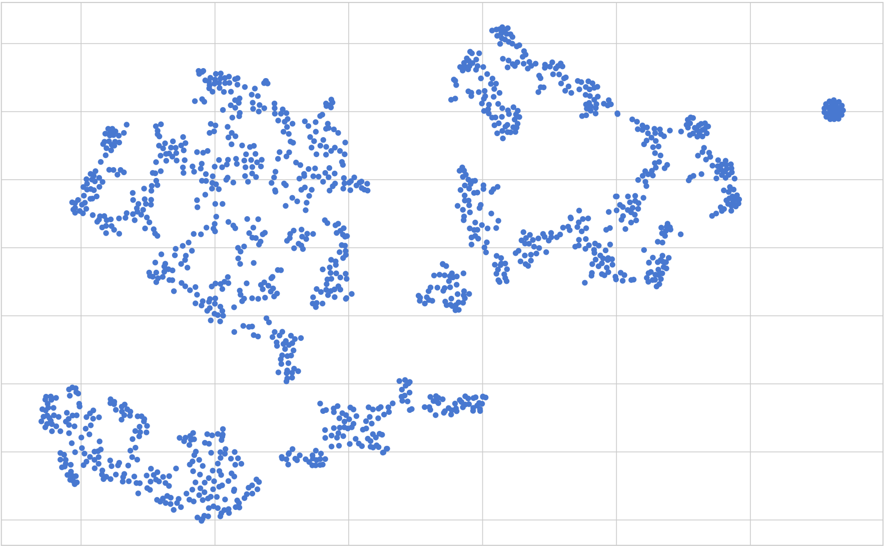


Figure 5.2: The output of performing the UMAP algorithm on the 20-dimensional UCID data set. Each point in this figure represents a single sample (or day) within our data set mapped onto a 2-dimensional surface.

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 (shown in Figure 5.2) can be seen in Figure 5.3.

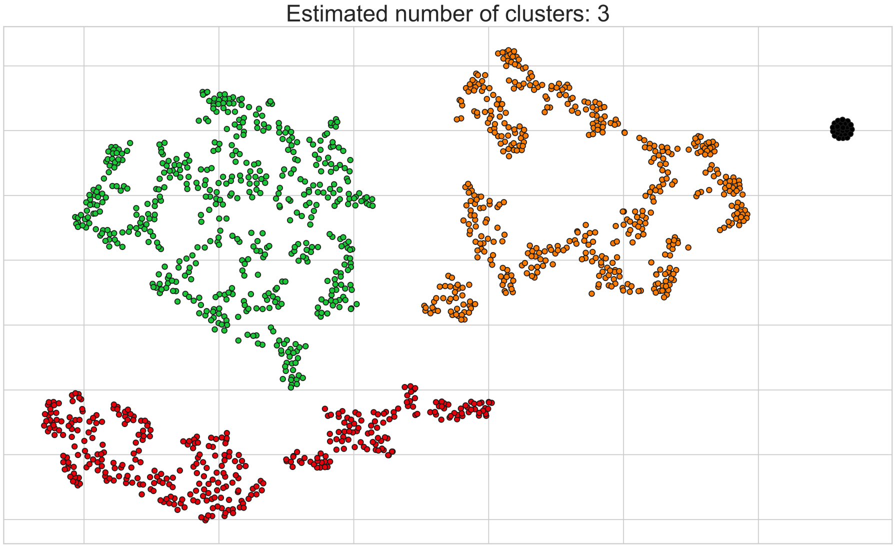


Figure 5.3: The output of performing the HDBSCAN algorithm on the 2-dimensional UCID data set previously seen in Figure 5.2.

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 5.4. 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.

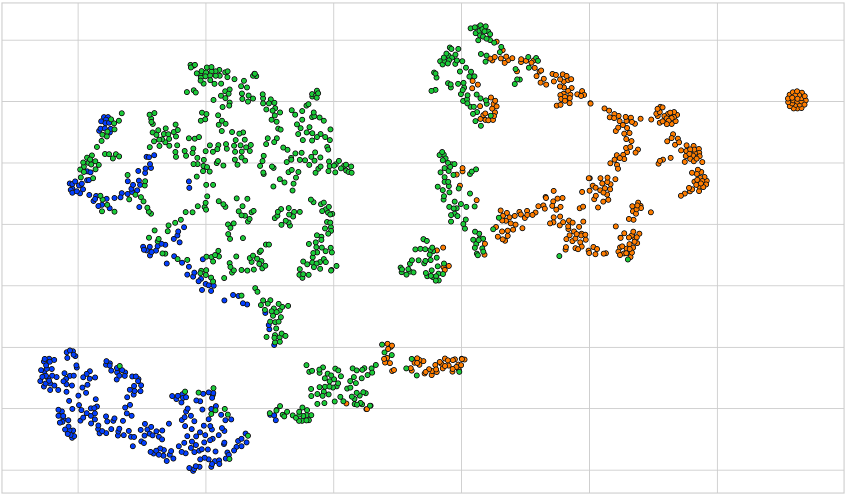


Figure 5.4: The output of performing the k-means algorithm on the 2-dimensional UCID data set previously seen in Figure 5.2.

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 can better 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 5.5. 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 us 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 (i. e., the global active power draw observed was completely stationary throughout this period); 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 5.5.

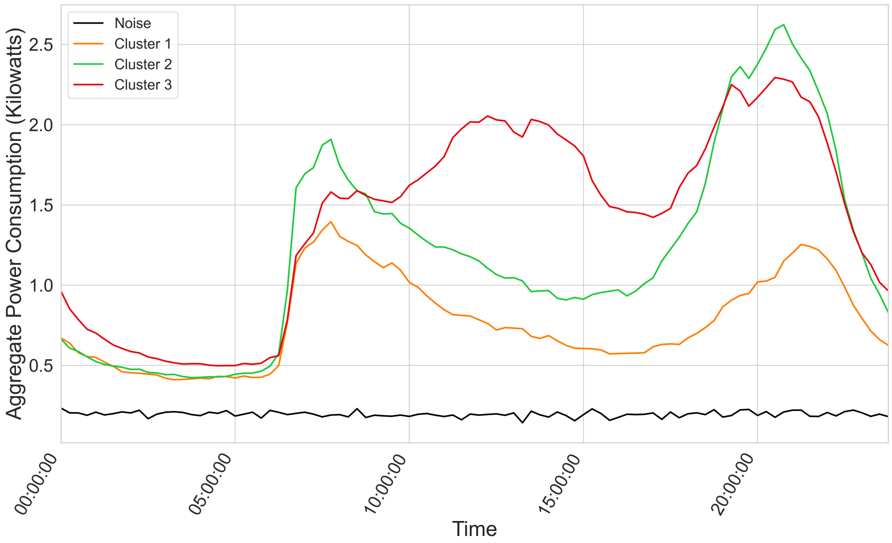


Figure 5.5: Average power consumption per hour of the day for each of the resulting clusters obtained after utilizing the HDBSCAN algorithm on our 2-dimensional representation of the UCID data set.

Following this, Figures 5.6 and 5.7 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 5.6 or Figure 5.7 respectively.

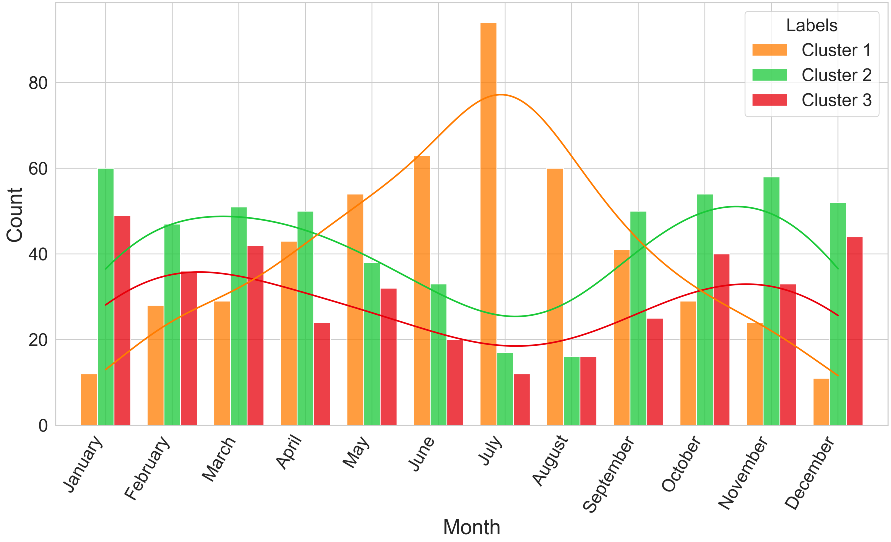


Figure 5.6: Distribution of the clusters over the different months of the year.

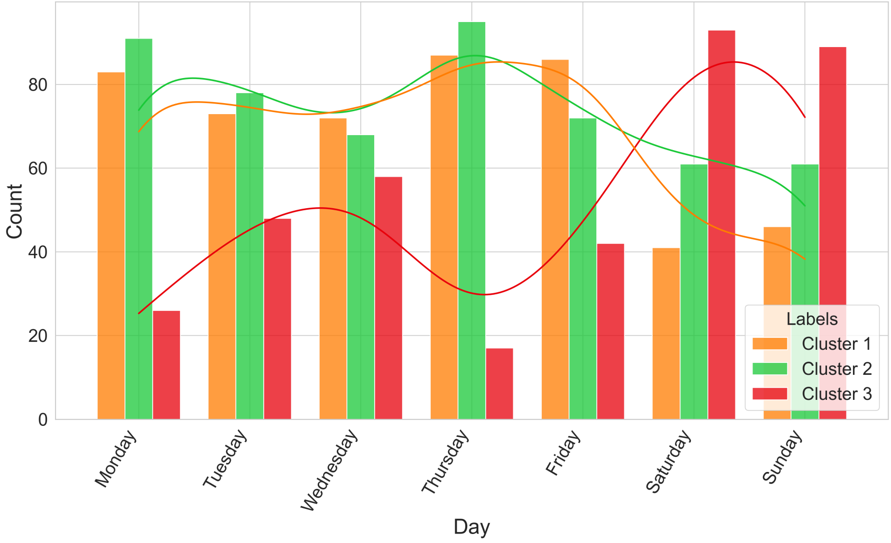


Figure 5.7: Distribution of the clusters over the different days of the week.

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, summer months peaking in terms of number of 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 ∼3 months with otherwise, generally, cooler temperatures).

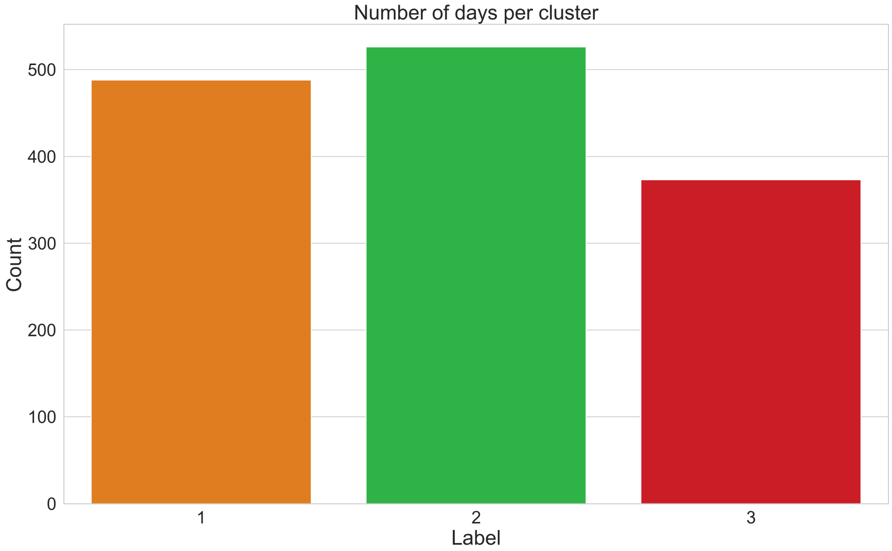
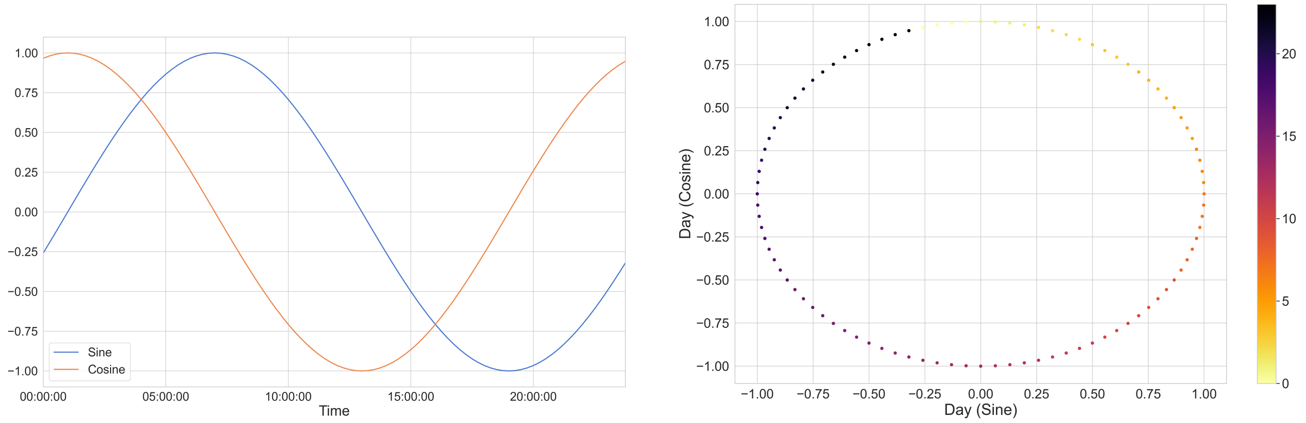


Figure 5.8: Spread in number of samples per cluster label.

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

5.3 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 Chapter 4. Nonetheless, a brief summary will be provided as part of Chapter 5. 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 B.1 pertains to the temporal variables that will be taken into consideration as part of this feature engineering step while Table B.3 pertains to the obtained, historical meteorological data that concern the regions associated with our data sets. Incidentally, as outlined in Table B.1, 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 5.9a and 5.9b.



(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.

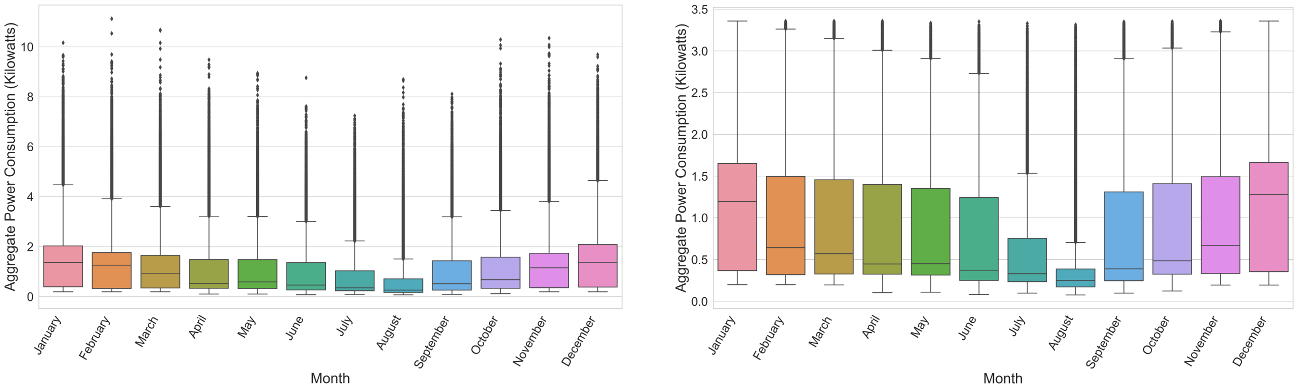
Figure 5.9: By utilizing a combination of the sine function and the cosine function, we eliminate the possibility that two different times would receive the same value had we used either function independently. The combination of both functions can be thought of as an artificial 2-axis coordinate system that represents the time of day.

Step 3.2: Following the feature engineering process, the feature selection 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 Section 4.3. 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.

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 caliber 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.9) and so this is not a feasible option. Another method, explored during prior, related research [8], 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:

$$Q1 - (1.5 * IQR) < x < Q3 + (1.5 * IQR) \tag{5.1}$$

Figures 5.10a and 5.10b represent the distribution of values for our target variable over the different months of the year both before and after removing outliers respectively.

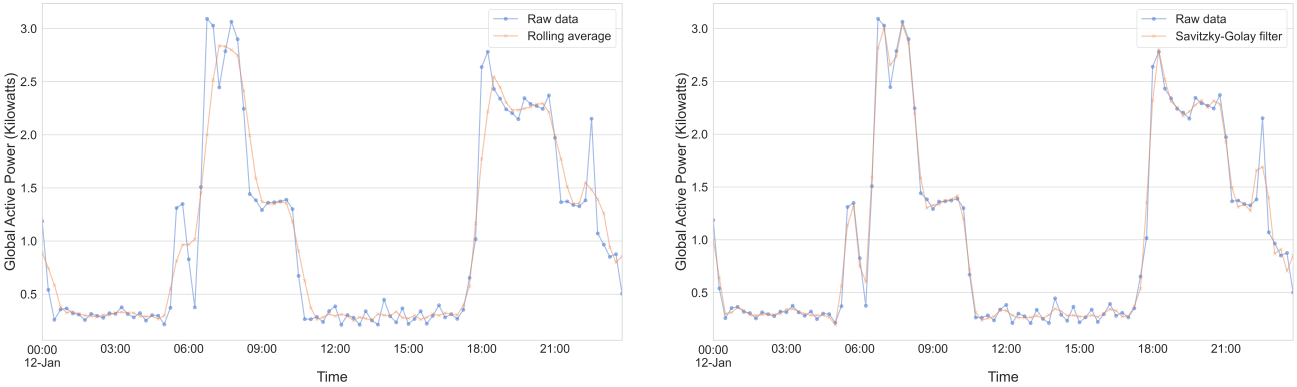


(a) Before removing outliers.

(b) After removing outliers.

Figure 5.10: 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 5.1.

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 [5] in which a moving (or rolling) average method was utilized. With regards to our proposed forecasting pipeline, we will be utilizing Savitzky-Golay filters [41] 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 5.11a and 5.11b 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 5.11: 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 (as seen in Figure 4.3); a preliminary smoothing step can be undertaken through the use of Locally Weighted Scatterplot Smoothing (LOESS) – this is illustrated in Figure 5.12.

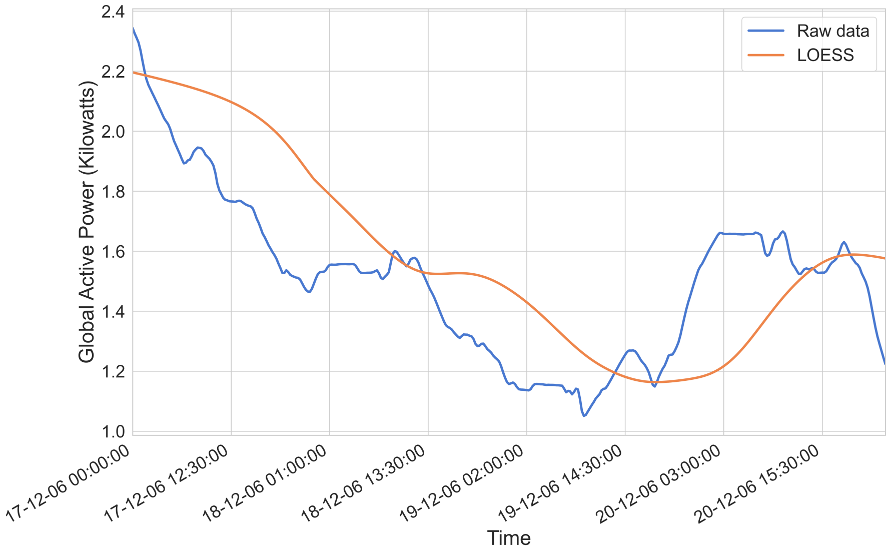


Figure 5.12: An illustration of the previously obtained trend component both with and without the application of LOESS.

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.

5.4 STAGE 4 - TRAINING AND TESTING

In contrast to the earlier stages, stage 4 will be subdivided into Subsections 5.4.1 and 5.4.2, where Subsection 5.4.1 serves to present an overview of our classification model while Subsection 5.4.2 serves to present an overview of our forecasting model.

5.4.1 *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.

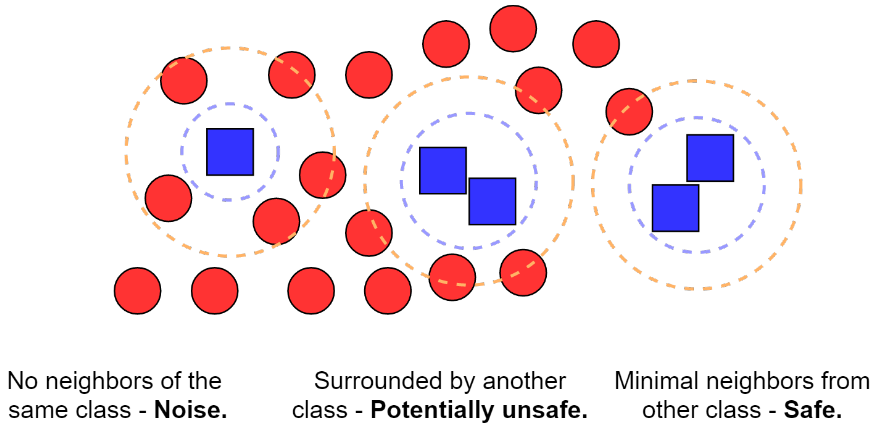

No neighbors of the same class - **Noise.**
Surrounded by another class - **Potentially unsafe.**
Minimal neighbors from other class - **Safe.**

Figure 5.13: 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 5.8. 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 to either undersample the majority class(es) or oversample 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. [42] as well as Figure 5.13 that provides a layman's explanation of the algorithm. The results of applying the SMOTE algorithm, and the overall negation of the previously mentioned class imbalance, can be seen in Figure 5.14.

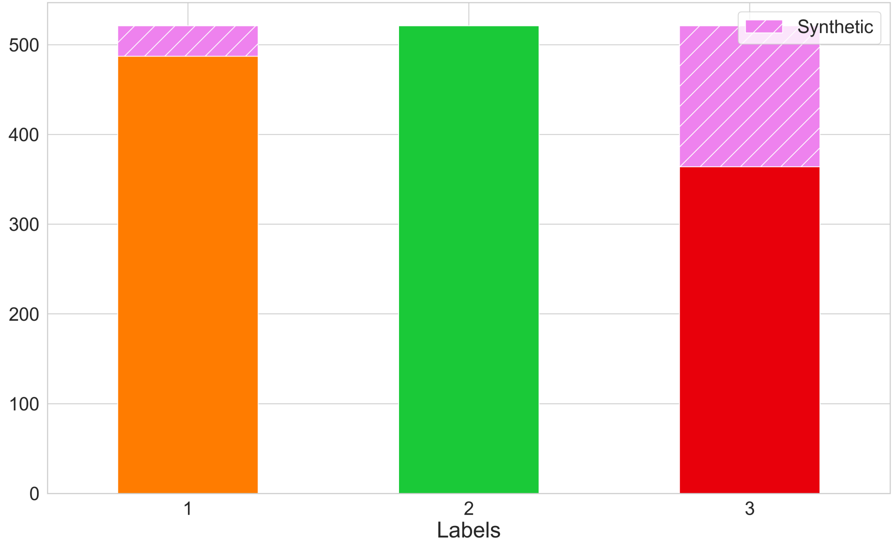


Figure 5.14: Number of samples per class label after applying the SMOTE algorithm.

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.), and historical as well as forecasted meteorological data (air temperature, humidity, cloud opacity, etc.), as provided by Solcast, and these will serve to act as the base-line number of features that our classifier will receive with which to assign a new sample into one of the previously generated clusters. Numerous methods exist to minimize the overall amount of features being passed to our classification model, some of which 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. Applying this algorithm reduces the overall number of features that our Random Forest Classifier utilizes from an initial 77 down to a mere 24 (as shown in Figure 5.15) which is an overall reduction of $\sim$ 68%.

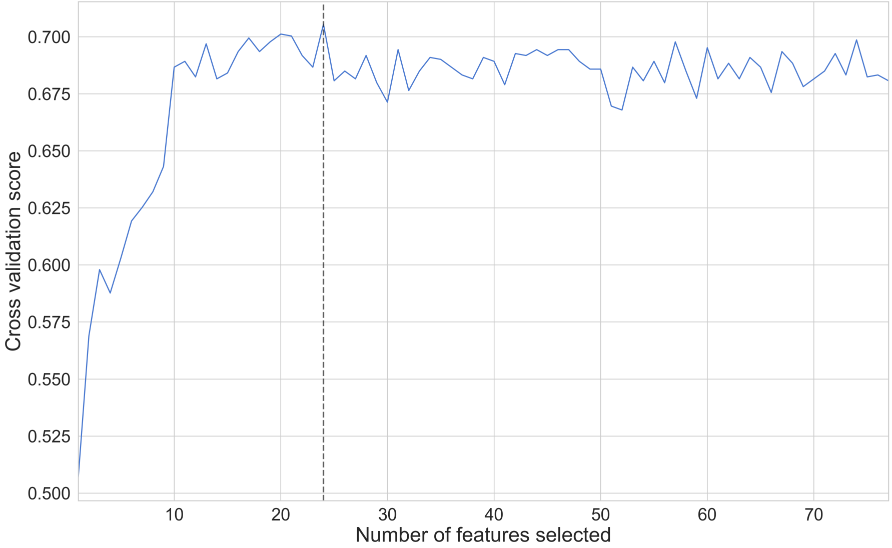


Figure 5.15: Assessing the number of important features through the use of the RFECV algorithm. In this particular scenario, the optimal number of features was pruned down from a total of 77 to a mere 24.

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 ensure an even distribution of class labels per validation set. We can conduct a quick inspection of the now fitted model by calculating 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 5.16. it is important to note that, when calculating the permutation importance 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.

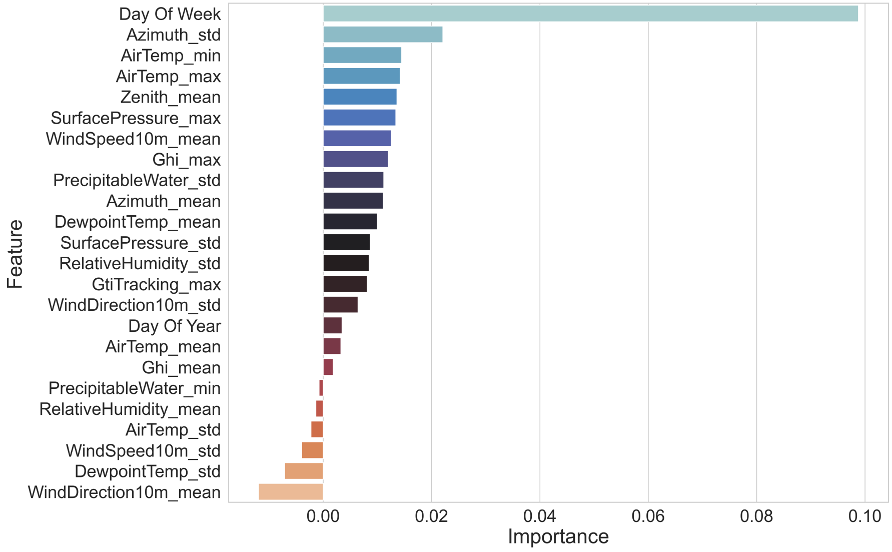


Figure 5.16: The permutation importance of each of the features chosen as part of our fitted Random Forest classifier.

The final model is then ready to accept new samples and assigns them a cluster based on the training procedure outlined through Section 5.4.1.

5.4.2 *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 (cyclical, temporal as well as meteorological) that we pass to the network as input in what we can loosely 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. 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 [24] when tackling time series problems such as that of residential energy consumption forecasting. The sample network illustrated in Figure 5.17 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.

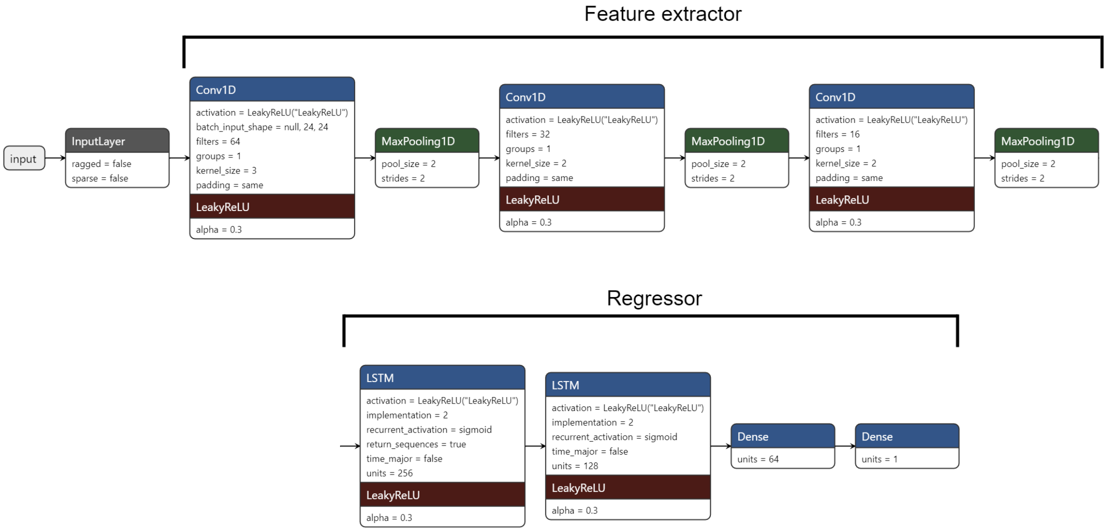


Figure 5.17: A simple, example CNN-LSTM network that makes one-step-ahead predictions.

To train our network, we will be utilizing Adam [43]: 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 [43]. Additionally, 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 5.18). 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.

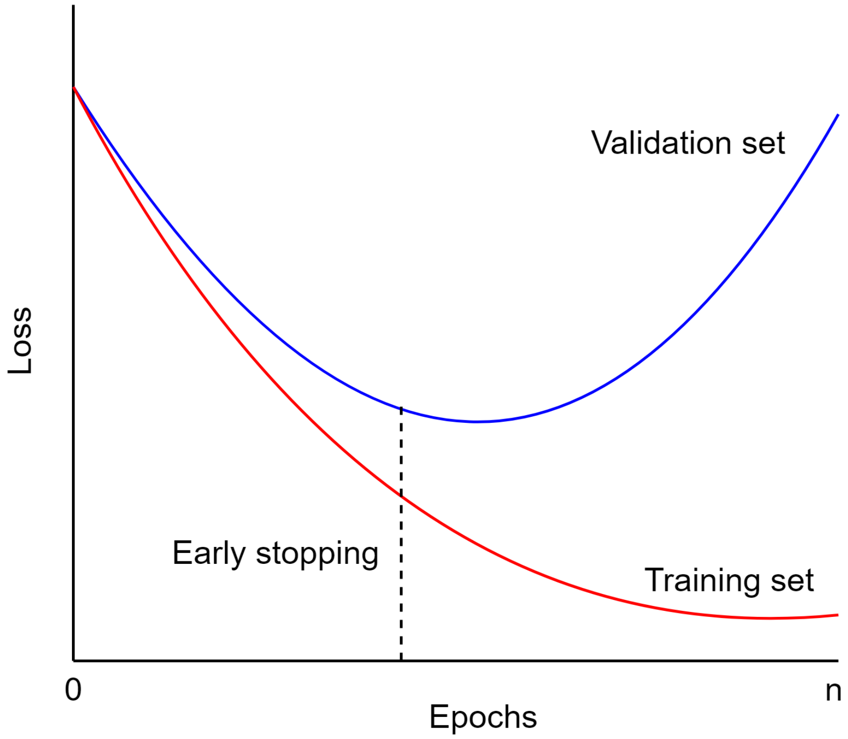


Figure 5.18: Illustration of early stopping.

For the purposes of our experiments and procuring the results showcased in Chapter 6, 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.

6 RESULTS AND DISCUSSION

Following the brief example demonstration of the method in Chapter 5, 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 method.

6.1 CLUSTER LABEL CLASSIFICATION

The first results that we will be demonstrating are that of the classification step of our method – refer to Table 6.1.

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

Table 6.1: 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 6.1 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.

Figures 6.1 and 6.2 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 number of incorrect predictions for each of the data sets and cluster 1 containing the largest amount of correct predictions.

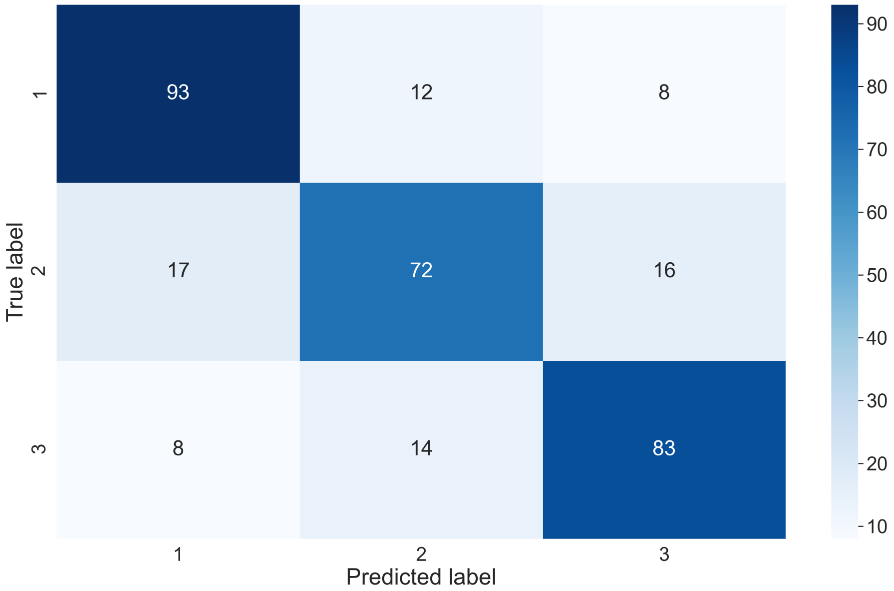


Figure 6.1: Confusion matrix - UCID.

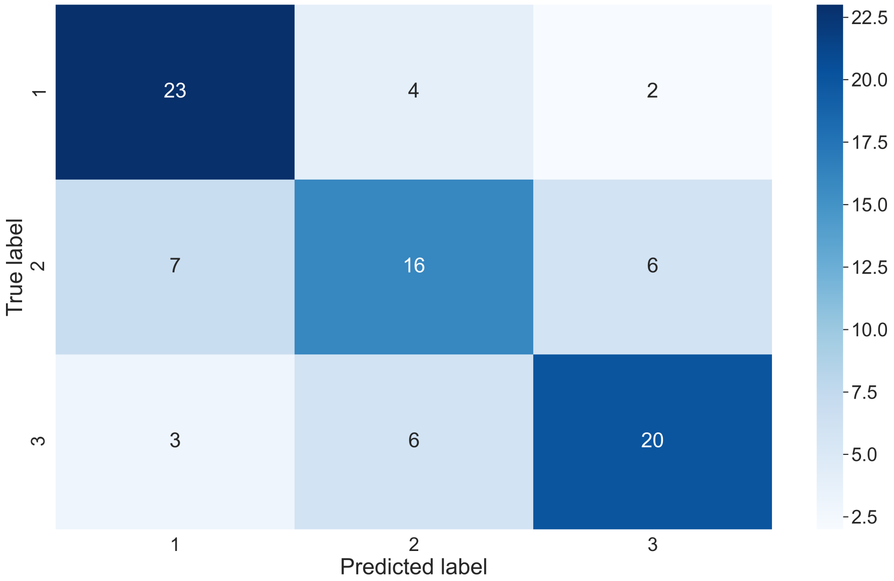


Figure 6.2: Confusion matrix - REFIT.

6.2 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 method yields superior forecasting accuracy at variable resolution. Table 6.2 presents a performance comparison of common models discussed in the literature and our 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 6.2 for the time being.

DATA SET	METHOD	MAE (KW)	RMSE (KW)	MAPE
UCID	LSTM [24]	0.62	0.86	51.45%
	CNN-LSTM [24]	0.34	0.61	34.84%
	Proposed	0.14	0.19	21.62%
REFIT	LSTM	N/A	N/A	N/A
	CNN-LSTM	N/A	N/A	N/A
	Proposed	0.11	0.17	25.77%

Table 6.2: 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 4. We attempted to tackle this problem ourselves and applied the 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 6.3. We note that the results here are considerably good, achieving a MAPE of $\sim 2\%$ for both data sets.

DATA SET	MAE (KW)	RMSE (KW)	MAPE
UCID	0.02	0.02	2.58%
REFIT	0.02	0.02	4.32%

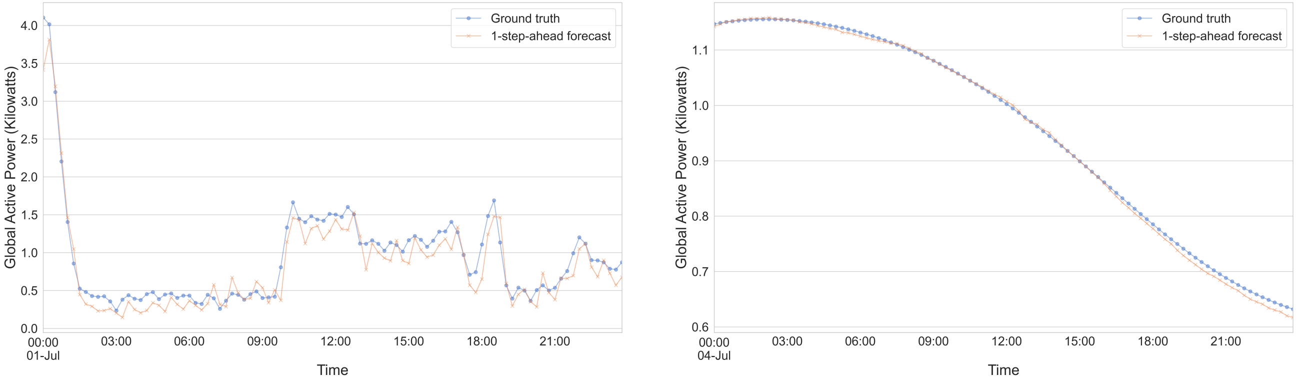
Table 6.3: Performance metrics obtained when applying our method on the trend component of each of the UCID as well as the REFIT data sets to obtain a one-step-ahead prediction.

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 6.4. 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, MAPE scores for the raw data for each of our data sets fell somewhat substantially, with an overall loss of about ∼ 10%, when moving from one-step-ahead forecasts to twelve-step-ahead forecasts which is more in line with what one could expect in this scenario.

DATA SET	METHOD	MAE (KW)	RMSE (KW)	MAPE
UCID	Raw	0.37	0.59	38.23%
	Trend	0.02	0.02	3.15%
REFIT	Raw	0.17	0.31	39.75%
	Trend	0.02	0.02	4.75%

Table 6.4: Performance metrics obtained when applying our 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.

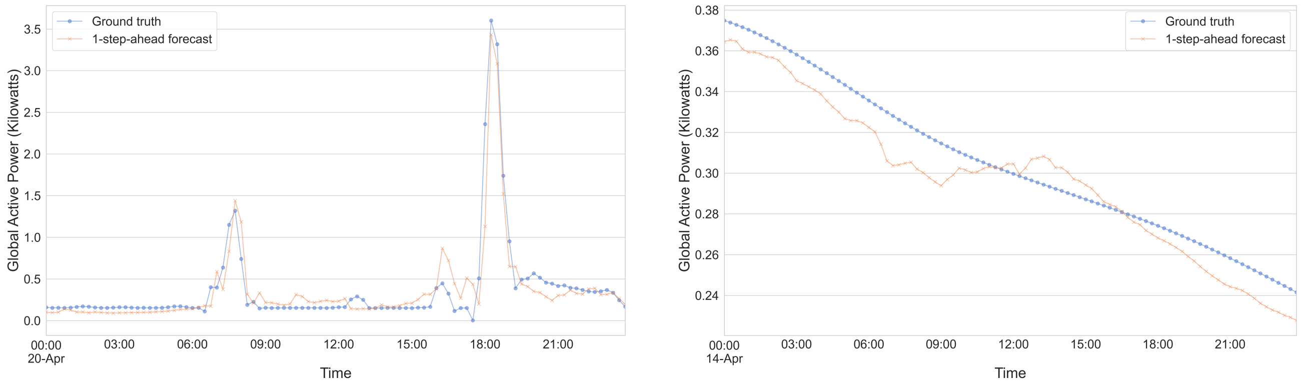
To further showcase, or otherwise visualize, the capabilities of our model we present Figures 6.3a, 6.3b, 6.4a and 6.4b that serve to illustrate one-step-ahead forecasts generated for a subset of each of the UCID data set as well as house 12 of the REFIT data set. These figures illustrate predictions made by each of our individual models on both the raw data as well as the trend component for each of the REFIT data set as well as the UCID data set over a period of 1 day out of our test set.



(a) UCID - Raw data.

(b) UCID - Trend.

Figure 6.3: Showcasing the capabilities of our method in making one-step-ahead predictions on the UCID data set.



(a) REFIT - Raw data.

(b) REFIT - Trend.

Figure 6.4: Showcasing the capabilities of our method in making one-step-ahead predictions on the REFIT data set.

A cursory glance at Figures 6.3b and 6.4b shows us that our model seems to excel at making one step predictions on the trend component of the UCID data set while the predictions being made for the trend component of the REFIT data set seem to be slightly less accurate. When considering the raw data for each of the data sets (Figures 6.3a and 6.4a) the differences are less pronounced and it seems that the model is capable of making accurate predictions one step into the future. This might vary in between days though and, given that the days chosen for these illustrations were completely random, it might be the case that the model performs better (or worse).

N.B. we note that the results obtained as part of Section 6.2 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 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.)

7 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 method.

Furthermore, one of the benefits of our 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 – for example, improvements in classifier accuracy could be seen through the utilization of a more efficient classifier. Alternatively, the current lack of contextual information that serves to explain the emergence of the clusters as part of the clustering step could likely be the reason for obtaining sub-par accuracy scores as, in its current iteration, the premise of our clustering step was to group together days that exhibited the highest similarity purely in terms of their energy consumption patterns and, given that this information is not readily available to us when considering a new day, we are left reaching for straws when 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. Additionally, 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 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.

To conclude, we note that, as a result of pre-clustering our data, and then training separate models on a per-cluster basis we achieved an improvement in overall forecasting accuracy with superior MAPE scores in contrast to the current state-of-the-art (LSTM networks, clustering based on K-means, etc.).

Part IV

APPENDIX

A

APPENDIX - FIGURES

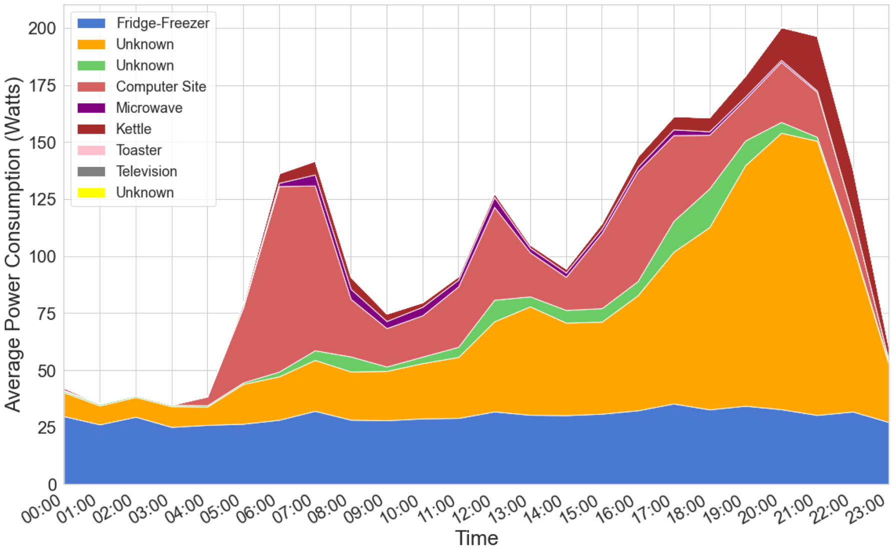


Figure A.1: A sample stacked area chart showing the readings of each appliance in each hour of a day. These readings were averaged over the entirety of the data present in the data set. Data for this plot was pulled from CLEAN_House12.csv of the REFIT data set.

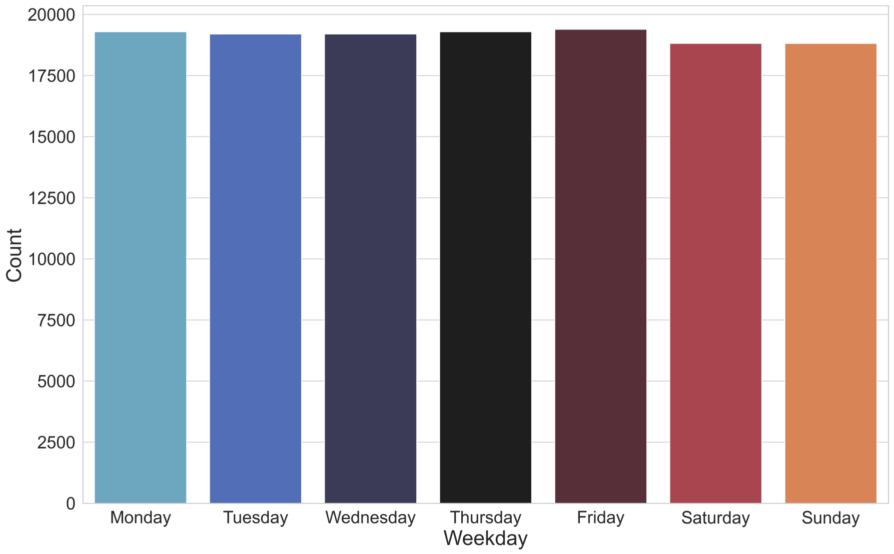


Figure A.2: Number of samples per day of the week over the entirety of the UCID data set.

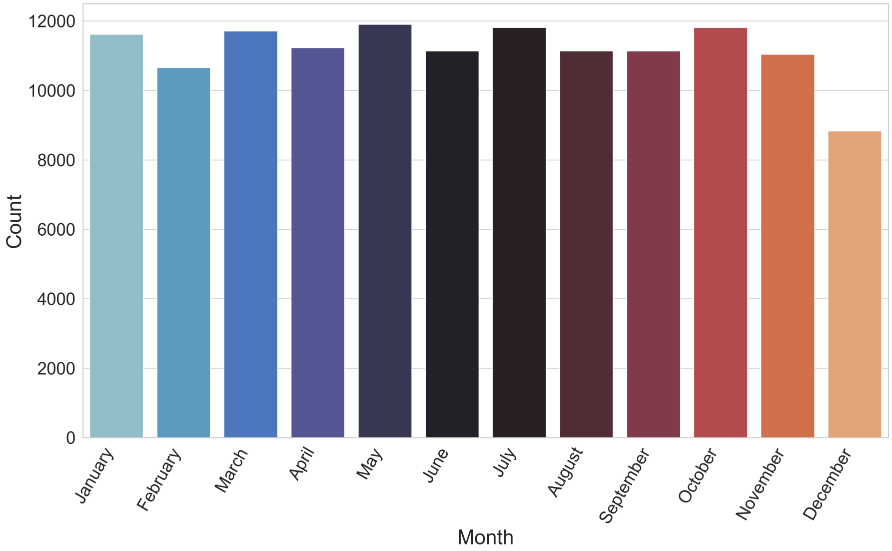


Figure A.3: Number of samples per month over the entirety of the UCID data set.

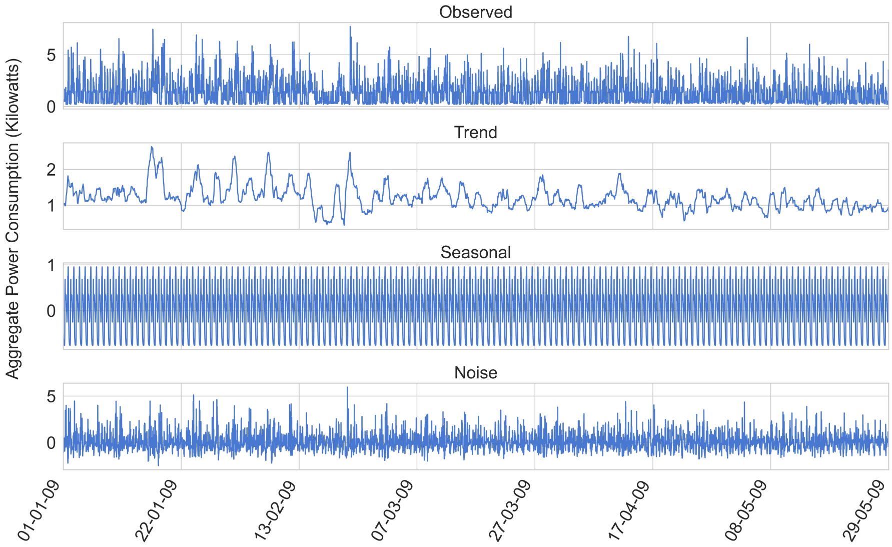


Figure A.4: Time series decomposition performed on the UCID data set. Data for these plots were pulled over a 6 month period that was resampled into a resolution of 15 minutes.

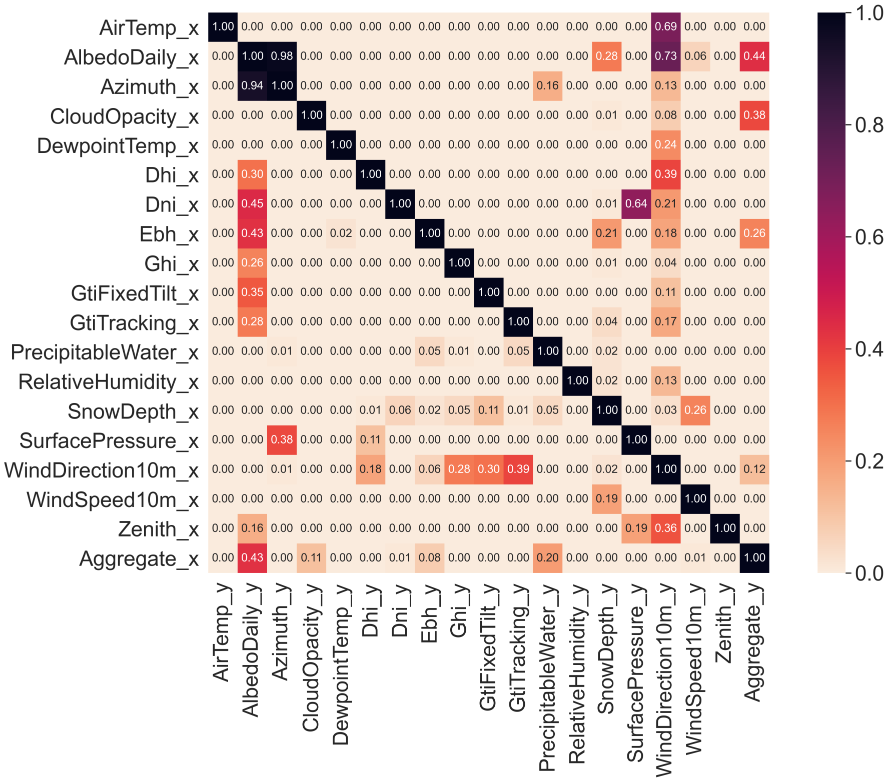


Figure A.5: The complete Granger Causation matrix for the REFIT data saet with all of the relevant features included.

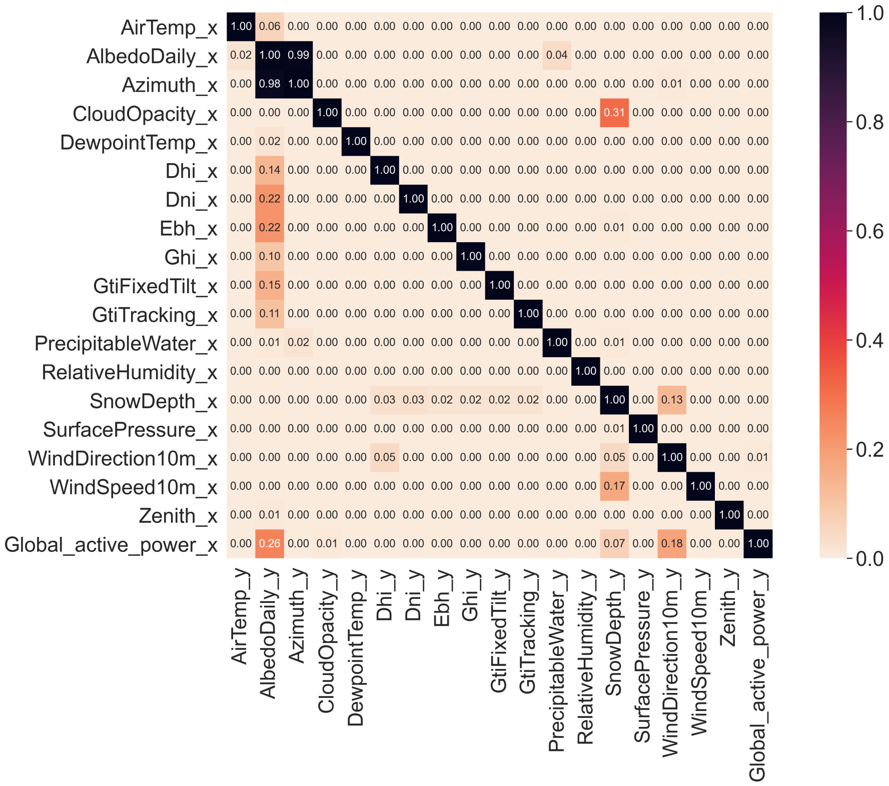


Figure A.6: The complete Granger Causation matrix for the UCID data set with all of the relevant features included.

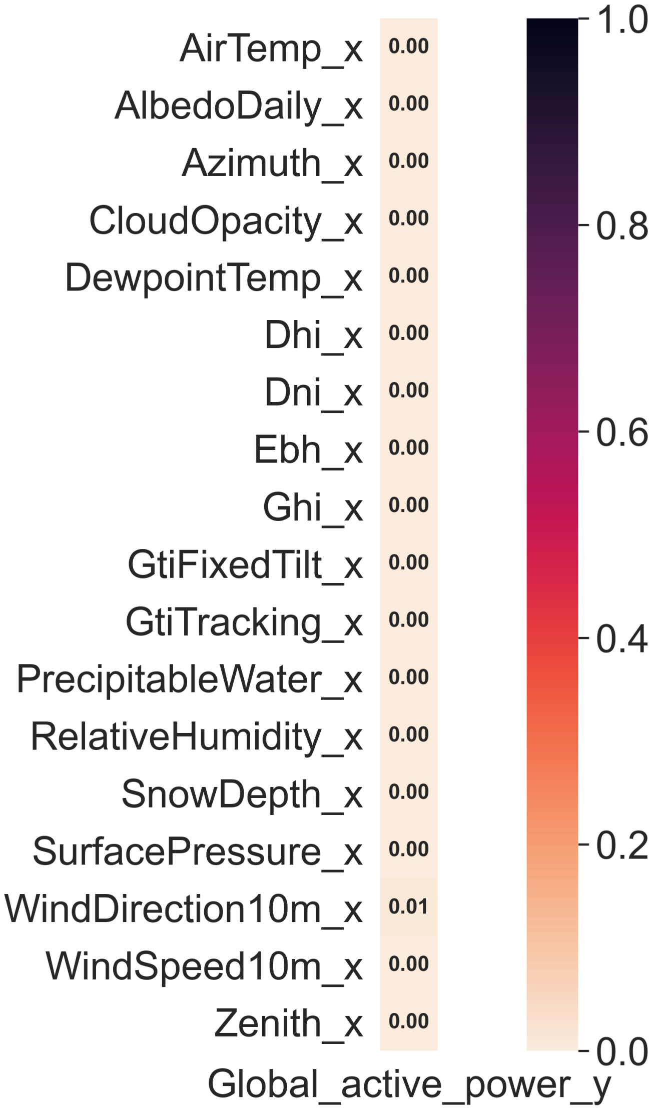


Figure A.7: A trimmed subset of the Granger Causation matrix (Figure A.6) that displays only the relevant information with regards to our independent variables causing our target variable.

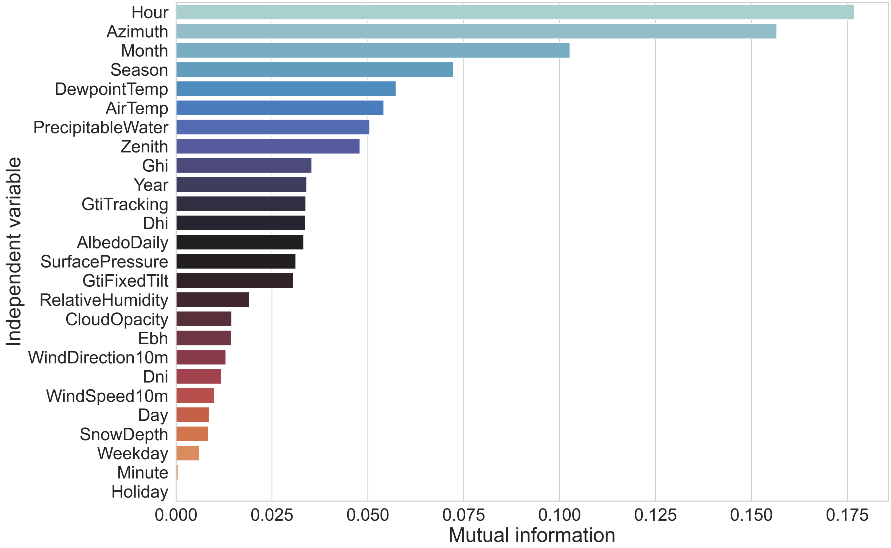


Figure A.8: Mutual information of our independent variables against our target variable.

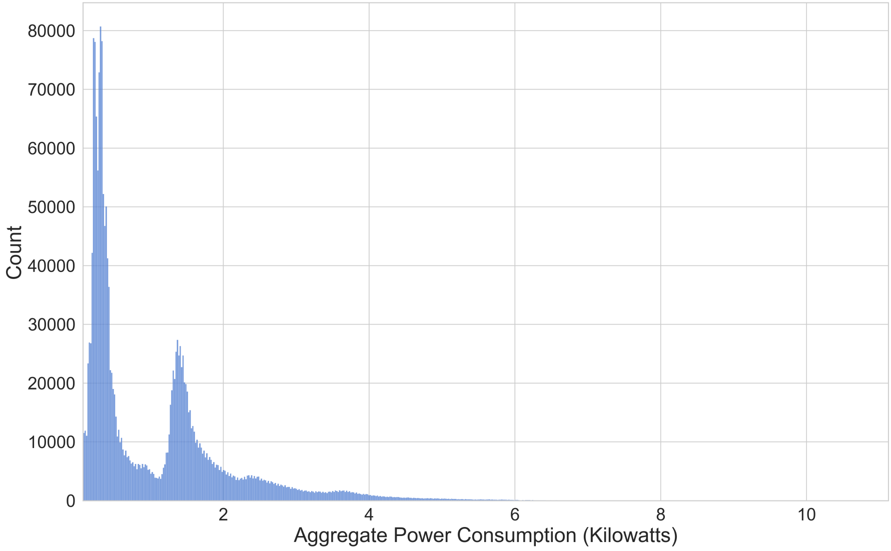


Figure A.9: Distribution of values with regards to our target variable.

B APPENDIX - TABLES

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 B.1: List of temporal variables that are taken into consideration during the feature engineering process as outlined in Section 5.3.

FEATURE	P-VALUE	STATIONARY
AirTemp	6.37e-04	True
AlbedoDaily	4.07e-27	True
Azimuth	0.0	True
CloudOpacity	0.0	True
DewpointTemp	5.06e-15	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	3.26e-26	True
RelativeHumidity	1.29e-23	True
SnowDepth	1.98e-26	True
SurfacePressure	1.29e-22	True
WindDirection10m	0.0	True
WindSpeed10m	0.0	True
Zenith	0.04	True
Global_active_power	0.0	True

Table B.2: The results of performing the Augmented Dicky-Fuller test on our target variable as well as the meteorological variables introduced in Section 1.1.3 and outlined in Table B.3 for the UCID data set.

PARAMETER	DESCRIPTION
Air Temperature	The air temperature (2 meters above ground level). Units in Celsius.
Albedo	Average daytime surface reflectivity of visible light, expressed as a value between 0 and 1. 0 represents complete absorption. 1 represents complete reflection.
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.
Dewpoint	The air dewpoint temperature (2 meters above ground level). Units in Celsius.
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.
EBH	The horizontal component of Direct Normal Irradiance (DNI). Units in W/m2.
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.
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.
GTI Horizontal Single-Axis Tracker	The total irradiance received on a sun-tracking surface. Units in W/m2.
Preciptable Water	The total column preciptable water content. Units in kg/m2.
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.
Snow Depth	The snow depth liquid-water-equivalent. Units in cm.
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.
Wind Speed	The wind speed (10 meters above ground level). Units in m/s.
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.

Table B.3: List of meteorological parameters available to us as per the Solcast data sets.

GLOSSARY

BEMS	Battery Energy Management System. 2, 3, 8
CART	Classification and Regression Tree. 9, 11
CCF	Cluster-Classify-Forecast. 9
CILF	Computationally intelligent load forecasting. 8
CNN	Convolutional Neural Network. 22, 55
CNN-LSTM	Convolutional Neural Network Long Short-Term Memory. iii, ix, 10, 11, 22, 55, 56
DBSCAN	Density Based Spatial Clustering of Applications with Noise. vi, xi, 8, 9, 11, 18–20, 44
DNI	Direct Normal Irradiance. 74
EBH	Direct (Beam) Horizontal Irradiance. 39, 74
EDA	Exploratory data analysis. 30, 33, 48
EPSRC	Engineering and Physical Sciences Research Council. 4
GA	Genetic algorithm. 11
GHI	Global Horizontal Irradiance. 74
GTI	Global Tilted Irradiance. 74
HDBSCAN	Hierarchical Density Based Spatial Clustering of Applications with Noise. vii, viii, 18, 20, 21, 44–46, 63
HEMS	Home Energy Management System. vi, 2, 3, 8
IAM	Individual appliance monitor. 5, 6, 30
IQR	Interquartile range. 50
LOESS	Locally Weighted Scatterplot Smoothing. viii, 51
LSTM	Long Short-Term Memory. vii, 10, 11, 22, 25–27, 55, 64
MAE	Mean absolute error. 28

MAPE	Mean absolute percentage error. iii, 10, 11, 60, 61, 64
MLP	Multi-Layer Perceptron. 10, 11
NAN	Not a number. 6, 42
NMAE	Normalized mean absolute error. 10
NRMSE	Normalized root mean square error. 10
PCA	Principal Component Analysis. 13
REFIT	Personalised Retrofit Decision Support Tools for UK Homes using Smart Home Technology. vii, ix–xi, 4–6, 30–35, 41, 42, 48, 49, 58–62, 66, 68
RELU	Rectified Linear Unit. vii, 22–24
RFECV	Recursive Feature Elimination and Cross-Validation. ix, 54
RNN	Recurrent Neural Network. 9, 25
SGD	Stochastic Gradient Descent. 56
SMBM	Smart Meter Based Model. 9, 10
SMOTE	Synthetic Minority Oversampling Technique. ix, 52, 53
SOM	Self-Organizing Map. 9–11
SVM	Support Vector Machine. 10
SVR	Support Vector Regression. 10, 11
T-SNE	T-Distributed Stochastic Neighbor Embedding. 13, 14, 16, 17, 43, 63
UCI	University of California, Irvine. 4
UCID	UCI data set. viii–xi, 4, 6, 30, 41–46, 49, 50, 58–62, 66, 67, 69, 73
UK	United Kingdom. 5, 6
UMAP	Uniform Manifold Approximation and Projection. vi, viii, 13, 16, 17, 43, 44
VSTLF	Very short-term load forecasting. 3, 7

BIBLIOGRAPHY

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

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

[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] 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.

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

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

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

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

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

[30] *Understanding UMAP*. URL: https://pair-code.github.io/understanding-umap/.

[31] Yingfan Wang et al. "Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMAP, and PaCMAP for Data Visualization." In: (Dec. 2020).

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

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

[39] 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.

[40] 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.

[41] Abraham. Savitzky and M. J. E. Golay. "Smoothing and Differentiation of Data by Simplified Least Squares Procedures." In: *Analytical Chemistry* 36.8 (1964), pp. 1627–1639. DOI: 10.1021/ac60214a047. URL: https://doi.org/10.1021/ac60214a047.

[42] 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.

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