L-Università ta' Malta
Faculty of Engineering

MASTER OF SCIENCE IN ENGINEERING DISSERTATION

Imputation and Anomaly Detection for Consumer Load Profiles

MICHAEL FARRUGIA

Supervised by:

DR. KENNETH SCERRI

Co-supervised by:

DR. ING. ANDREW SAMMUT

A dissertation submitted in partial fulfilment of the requirements for the degree of Master of Science in Engineering

by the

Faculty of Engineering

December 2020

Declaration

No portion of the work referred to in the dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright Notice

Abstract

The advent of smart meters has opened new possibilities for utilities. Billing is no longer the sole function of the meter. The load profile data, registered by the smart meters, can be analysed in order to obtain knowledge about various aspects. It can be used to indicate stress points in the network, calculate technical losses and explore methods for their reduction. In reality, though, it is inevitable that a certain amount of data will get lost. While doing every effort to keep this loss of data to the minimum possible, the missing portions of data must be imputed before any further analytical activity can be performed.

This thesis explores techniques for imputing such missing data in smart meter load profiles. The proposed method implements a k-nearest neighbors approach where the imputed part is calculated by searching the past consumption of the consumer for patterns that best resemble the portion around the missing part and taking an average of the parts which corresponds to the missing portion. The length of each segment and the number of segments to be considered are arbitrary and therefore suitable values had to be determined through a tuning process. Using the developed algorithm on a sample of 335 consumer load profiles, the average RMSE was 7.47% of the actual values.

An area of great concern for utilities is non-technical losses which can be made up of billing inaccuracies, faulty meters and fraudulent consumers. This thesis develops a method of anomaly detection for finding consumers with irregular behaviour which are likely to contribute to non-technical losses. The consumers are grouped into clusters having similar weekly consumption behaviour by using hierarchical clustering. For each consumer two novel coefficients are computed: the Anomaly Coefficient, which is a measure of how different the consumption behaviour of the consumer is from the other consumers in the same cluster, and the Cluster Change Coefficient, which is a measure of how irregular in consumption behaviour the consumer is, compared to all the other consumers. Consumers having high values for any, or both, of these coefficients are more likely to exhibit non-technical losses.

Acknowledgements

I would like to show my thanks and appreciation go to my supervisor Dr. Kenneth Scerri and to my co-supervisor Dr. Ing Andrew Sammut for their guidance, support and constructive feedback throughout this work. I would also thanks Enemalta Plc. for making this study possible.

I would like to dedicate this work to my late mother, who passed away during this research. For me she was always an example of how life can be enjoyed with the simplest means.

Contents

List of Figures

List of Tables

List of Abbreviations

AMM	Automated Meter Management
ANN	Artificial Neural Network
ANOVA	Analysis Of Variance
ARIMA	AutoRegressive Integrated Moving Average
BDR	Bayesian detection rate
CBMS	Cluster-based Best Match Scanning
CER	Commission for Energy Regulation (Ireland)
CHP	Combines Heat and Power
CLARA	Clustering Large Applications
CNN	Convolutional Neural Network
CoD	Coefficient of Determination
COI	Cone Of Influence
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DFT	Discrete Fourier Transforms
DGS	Distributed Generating Source
DSO	Distribution System Operator
DT	Decision Tree
DW	Durbin-Watson
ELM	Extreme Learning Machine
EM	Expectation maximization
EMSI	expectation-maximization single imputation
EneGIS	Enemalta Geographic Information System
FCM	fuzzy c-means
FFT	Fast Fourier Transform
FIS	fuzzy inference system
FSK	Frequency Shift Keying
GBC	gradient boosting classifier
GIS	Geographic information system
IOT	Internet of Things
IQR	interquartile range
IUBS	Integrated Utilities Business Systems
KNN	K-Nearest Neighbours
LD	listwise deletion
LI	Linear Interpolation
LR	Logistic Regression
LS	Least squares
LSTM	Long Short-Term Memory
LV	Low Voltage
LVC	Low Voltage Concentrator

MAPE	Mean Absolute Percentage Error
MAR	Missing At Random
MCAR	Missing Completely At Random
MI	mode imputation
MICE	imputations by chained equation
MIM	Modified Imputation Method
MNAR	Missing Not At Random
MV	Medium Voltage
NB	Naïve Bayes
NOCB	Next Observation Carried Backwards
NTL	Non-Technical Losses
OWA	Optimally Weighted Average
PAM	Partitioning around Medoid
PCA	Principal Component Analysis
PCP	Principal Components Pursuit
PLC	Power Line Communication
PSD	Power Spectral Density
RF	Random Forest
RMS	Root Mean Square
RMSE	Root Mean Squared Error
ROC	Receiver Operator Characteristics
SARIMA	Seasonal AutoRegressive Integrated Moving Average
SOM	Self Organising Maps
SVM	Support Vector Machine
TukeyHSD	Tukey Honest Significant Differences
WCSS	Within-Cluster Sum of Squares
WSC	Water Services Corporation
WT	Wavelet Transform
XGBoost	Extreme Gradient Boosted Trees

1 Introduction

Presently Malta's local fossil fuel generating capacity amounts to a total of 588.6MW [1]. All this generation capacity is located at the Delimara Power Station site, with the exception of a 37MW open-cycle gas turbine which is located in the Marsa Power Station site. Electrical energy is also imported over the Malta – Sicily interconnector which has a maximum capacity of 200MW. Additionally, a number of renewable energy sources are spread across the Maltese islands which, at the end of 2019, had a maximum generating capacity of 157.7MWp. These renewable sources are made up of 154.6MWp solar photovoltaic systems (of which 22.4MWp were installed during 2019), 3.037MWe biogas plants and 0.0698MWp micro wind generators [1]. The peak demand for the Maltese islands was 504MW and occurred in July 2019 [2].

Distribution of electricity to consumers is achieved through a four-level network, using four voltage levels, 132kV, 33kV, 11kV and 400/230V [3]. Voltages of 33kV and higher are termed High Voltage (HV), 11KV is termed Medium Voltage (MV) while 400/230V is referred to as Low Voltage (LV). Figure 1.1 shows the layout of the Enemalta plc HV distribution network. Electricity from the Delimara Power Station and the Maghtab Terminal Station, where the Malta – Sicily interconnector lands, is distributed to four 132kV distribution centres (primary substations), two at Marsa, one at Kappara and another at Mosta, through 87 kilometres of underground 132kV cables. At these centres voltage is stepped down to 33kV so that electricity can be fed to another 20 33kV distribution centres (primary substations), 17 situated in Malta, two in Gozo and one in Comino. This part of the network includes 260 kilometres of 33kV underground cables and 13 kilometres of 33kV submarine cables. As can be seen from Figure 1.1 the cables connecting the primary substations are always laid in pairs (or more) to achieve N + 1 redundancy as a minimum.

Each 33kV distribution centre feeds a number of 11kV circuits supplying over 1,400 11kV substations (secondary substations) utilising 1,134 kilometres of underground 11kV cables and a few kilometres of overhead lines, mostly in rural areas. In the 11kV secondary substations, electricity is stepped down to 400/230 V to supply the consumers' premises through a low voltage, three-phase, four-wire network comprised of underground cables and overhead lines. An electricity meter is installed at each delivery point so that consumption is measured and can be billed. At the end of 2019 Enemalta plc supplied a total of 328,308 consumers, which consisted of 173,744 residential, 100,296 domestic (second properties which may be garages) and 54,268 non-residential (commercial) properties [4].

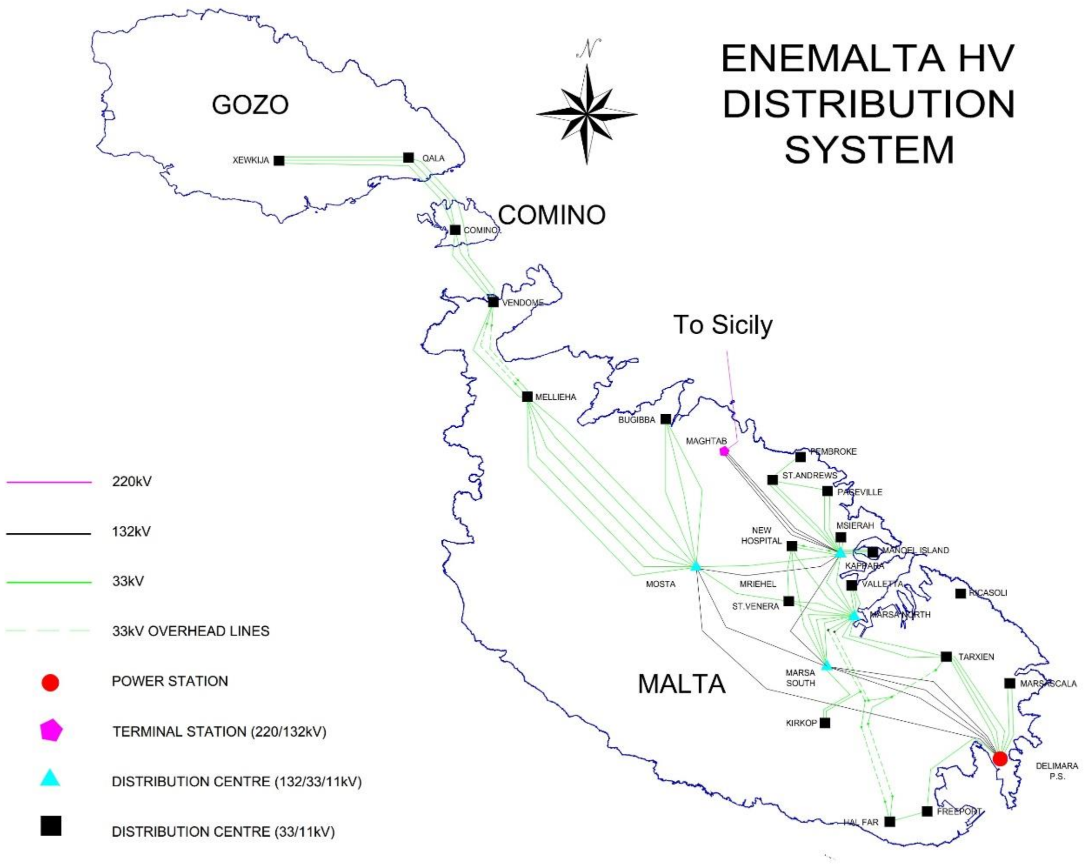

Figure 1.1. National grid diagram for the Maltese islands [3].

Having replaced almost all electricity meters with smart meters, Enemalta plc. is now capable of reading most of the meters remotely. This is of great benefit since the readings are now available without having to physically visit individual premises to read the meter in order to be in a position to bill the consumer. But apart from the billing there are other advantages which come with having a smart meter network. The smart meters register the consumption and communicate readings to the central system. This data consists of spot readings and load profiles which are read every day.

Having this data enables the utility to employ analytical methods to gain knowledge about various aspects. Used in conjunction with a Geographic Information System (GIS), the smart meter data can be utilised to quantify the consumption at various points in the network, exposing areas where the cables and aerial lines are heavily loaded and where reinforcement is needed. It can also be used to calculate the technical losses and explore methods for their reduction. This thesis develops a method for detecting anomalies in consumption behaviours which might indicate faulty meters or possibly fraud. However, due to the complex nature of the low voltage network and the vulnerability of the power line communication system adopted, it is inevitable that there will be a certain amount of data which occasionally gets lost. For statistical and analytical method to be used reliably on the data it is imperative that the data is complete without any missing data points or portions. For this reason, this thesis develops a method by which the missing data is imputed prior to identifying any anomalies in the consumer consumption.

1.1 Scope and Aims of the Study

The Enel Distribuzione Spa. smart meters installed by Enemalta plc. use Power Line Communication (PLC) [5, 6, 7] for communication with the central system. In Europe CENELEC A band, spanning from 3kHz to 95kHz, is reserved for use by utilities and smart grids. The Enel Distribuzione Spa. smart meters use Frequency Shift Keying (FSK) on a carrier of 72 kHz ± 1.2kHz (0 is 73.2kHz, 1 is 70.8kHz) at a baud rate of 2400b/s. Maximum output voltage is approximately 1.2 V_{rms} while maximum output current is about 500 mA_{rms} [8].

The advantage of using PLC is that the connecting medium is already laid and only the modems are needed for communication. However, the power distribution network was originally intended, designed and constructed, for transmission of AC power at 50Hz and so the power carrying conductors are not ideal for carrying the communication higher frequencies along considerable distances. To overcome the propagation problem, which is a limiting factor in all PLC, the system utilised by Enemalta employs a number of repeaters. Any smart meter may be automatically designated to act as a repeater at commissioning time, so that other downstream smart meters can be reached. Furthermore, the PLC signal is hampered by a number of other factors such as noise, interference caused by conducted emissions from other devices connected to the system and channel variability both in time and in frequency [8].

So, considering the complexity of the electricity distribution network on the low voltage side, and the susceptibility of the PLC method, it is understandable that there would be occasional failures in capturing the smart meter data. These failures will inevitably result in gaps in the data for a number of meters. In order to be able to perform reliable analytic activity on the collected smart meter data, the data should be complete without any interruptions and so any accumulated gaps in the data must be imputed. Having a complete load profiles allows for anomaly detection methods to be performed in order to detect irregular behaviors which may be caused by faulty or tampered metering setups. Thus, the aim of this work is to develop, first a method which imputes the missing load profile data, and then another method which detects anomalies in the consumption behaviours.

1.2 Thesis Methodology

The imputation of data is considered to be the process which prepares the data for anomaly detection. Literature reviews were carried out to identify the methodologies which have already been explored for similar work as outlined in Chapter 3.

In order to establish an effective technique for imputation of the data, an understanding of the size of the data set, the proportion of missing data, the shape of the load profiles and variability between consumers, is first obtained. With this information a thorough literature review was conducted in order to assess the various imputation techniques adopted by other researchers for similar datasets and situations. This allows the setup of the preferred imputation technique, which was then implemented and tested on the available dataset. The dataset used in this thesis is reviewed in Chapter 2.

For developing a suitable anomaly detection method, the information about the load profile dataset already obtained for the imputation part will be used. Having this knowledge, another detailed literature review was conducted to explore the anomaly detection techniques in general and especially those techniques used for smart meter datasets. Techniques which are most suitable were identified and used as basis for the development of an anomaly detection method for the Enemalta dataset.

1.3 Thesis Overview

This first Chapter has briefly outlined the scope and aims of this research. The rest of this thesis is structured as follows. Chapter 2 presents the datasets which are used in this work and also applies temporal and spectral analysis to load profiles. A literature review of the techniques used for data imputation and anomaly detection is presented in Chapter 3. This Chapter starts by exploring the relevant techniques in general and then goes on to focus on methods which are adopted for smart meter data applications. In Chapter 4 an algorithm for the imputation of missing load profiles is developed and the most appropriate parameters are determined. Chapter 5 uses an unsupervised machine learning method to compute two novel coefficients which quantify the probability of a consumer in contributing to NTL. Both Chapters 5 and 6 outline suggested enhancements for future work. Chapter 6 summarises the results achieved and draws some concluding remarks.

2 Smart Meter Data

In 2009 Enemalta Corporation (now Enemalta Plc.) had, in conjunction with Water Services Corporation (WSC), launched the Integrated Utilities Business Systems (IUBS) project. The main part of this project consisted of replacing all existing conventional meters by smart meters. Smart meters are capable of recording the consumption of electrical energy and communicating the information to a central system for monitoring and billing. Additionally, the smart meters used by Enemalta are equipped with a circuit breaker which trips if consumption exceeds a settable limit.

Smart meter data consist mainly of two types: spot readings and load profiles. The spot reading registers are counters registering the number of units of energy (KWh) consumed since the time when the meter was installed. Consumption billing is based on spot readings where the consumption between two dates is simply the difference between the readings of the counter on the two occasions. The load profile is the amount of energy in Watt Hours (Wh) consumed during periods of equal duration. The majority of meters installed by Enemata have 15-minute periods (96 readings per day), with some other meters having a 30-minute periods (48 readings per day). For this thesis data has been aggregated into 30-minute periods for all meters.

Commercial consumers have the heavier consumption and thus generate the higher revenue. Consequently, it is more effective for Enemalta to concentrate on this type of consumers rather than domestic and residential in any study that may be effective in detecting energy loss and/or fraud. This thesis has therefore considered commercial consumers for anomaly detection.

Communication to the meter is achieved through Power Line Communication (PLC). This method of communication uses the powerlines as the medium to convey the information between the devices. In the Enemalta implementation, PLC is only used on the LV part of the network. At every substation, where the voltage transformation between LV and MV takes place, a Low Voltage Concentrator (LVC) is installed. The LVC serves as an edge server (a server between two independent networks) between the central application domain, Automated Meter Management (AMM), and the distributed electricity meters. The LVC has a daily agenda to read the data (spot readings and load profiles) accumulated on all meters which are connected to the substation. The data collected from all meters is then conveyed to AMM via a GPRS/3G modem over TCP/IP using the GSM network. Figure 2.1 shows the communication setup between AMM and meters.

The TCP/IP connection is reasonably robust, and it can be assumed that all data collected by the LVC will be conveyed without error to the AMM. However, PLC has proved to be quite vulnerable, for the reasons mentioned in Section 1.1, and occasionally communication cannot be established between the LVC and the smart meter. In that eventuality all the data for that day for that particular meter is lost, causing a gap of one day in the data for that meter.

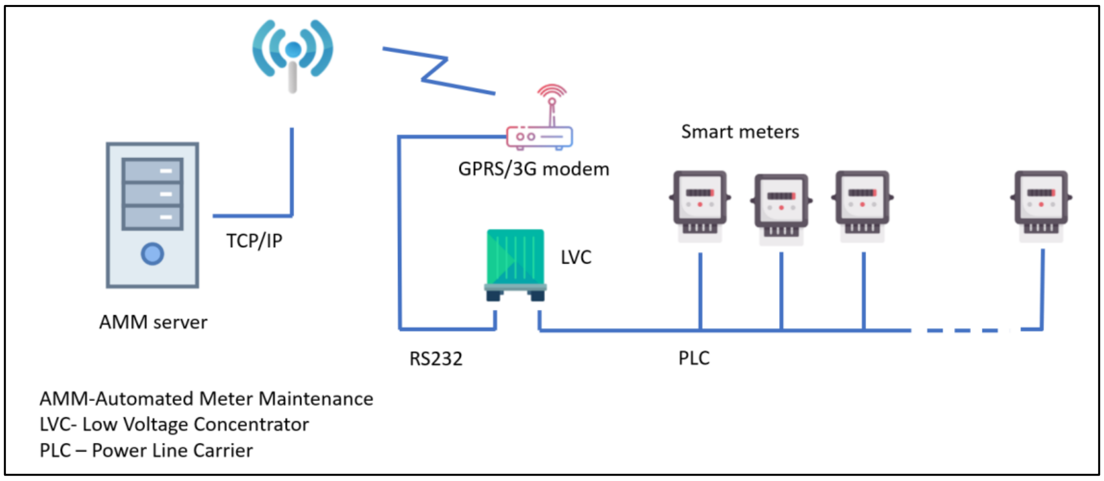

Figure 2.1. Communication between AMM server and smart meters.

Figure 2.2 shows a typical practical substation setup on the Enemalta Geographic Information System (EneGIS) and thus also on the PLC system. The red lines show underground cables which connect the aerial lines to the transformer. The consumers' meters (shown by violet and brown dots, with the red stars indicating PV meters) are connected to the aerial lines (shown as green and brown lines). Heavy consumers, with rated current exceeding 60 Amperes per phase, are supplied through dedicated underground cable passing to their premises.

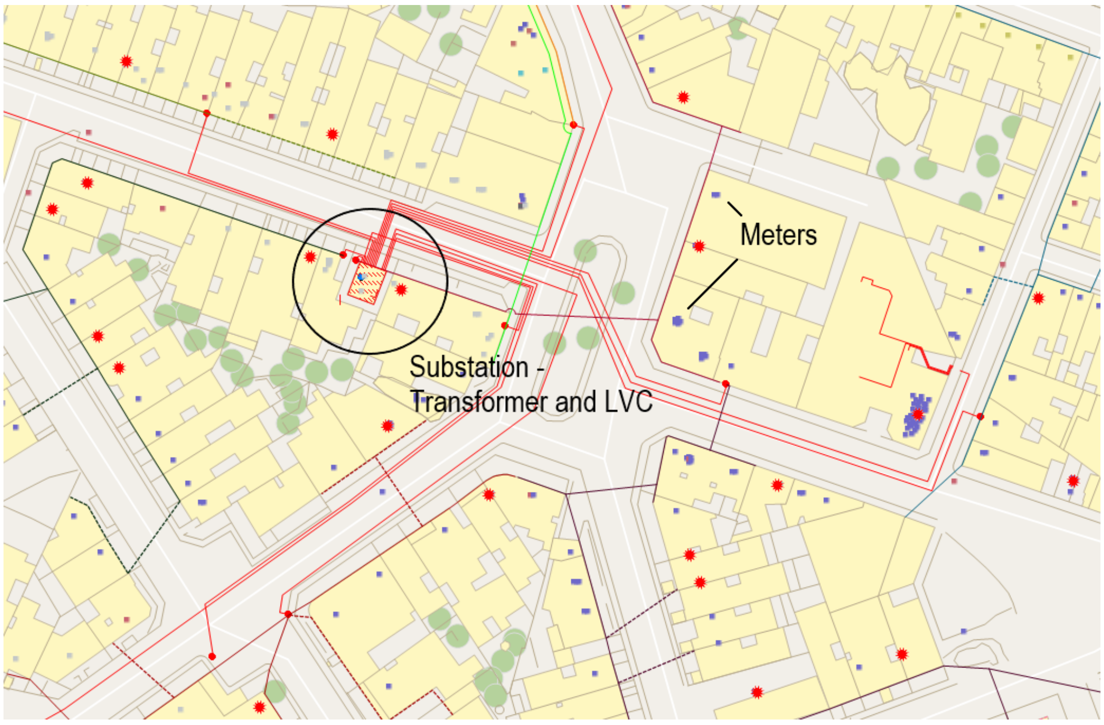

Figure 2.2. EneGIS sample layout. Red lines are underground cables, green and brown lines are overhead lines, violet and brown dots are consumer meters while the red stars are PV meters.

2.1 Introducing the Dataset

Collection of smart meter data started as early as 2009, when the first smart meters were installed. During the first months of installations the number of meters collecting data was relatively low. This number increased gradually until, at present, an average of almost 200,000 meters are being read daily.

Table 2.1 and Figure 2.3 show the number of installed meters at the end of each year since the beginning of the rollout, together with the average and maximum of daily spot readings read during each year. The spot readings here are being used as a measure of reachability since a request for spot reading is attempted every day to all installed smart meters. Average here signifies the average number of daily spot readings collected throughout the year. The maximum is the largest number of spot readings collected from the meters on a single day during that year. From these figures it is clear that a substantial amount of data is missing and thus there exists the requirement for a good imputation algorithm to be developed, so that the missing readings can be estimated and imputed.

Table 2.1. Average and maximum daily readings per year

Year	Installed	Daily spot readings	
		Average	Maximum
2010	47,587	3,904	10,507
2011	118,709	14,115	24,494
2012	189,641	38,281	55,991
2013	236,522	63,351	92,436
2014	261,937	97,665	130,521
2015	282,165	129,973	153,589
2016	293,010	149,774	166,811
2017	308,067	156,851	180,643
2018	319,364	153,528	207,411
2019	331,679	194,071	225,733

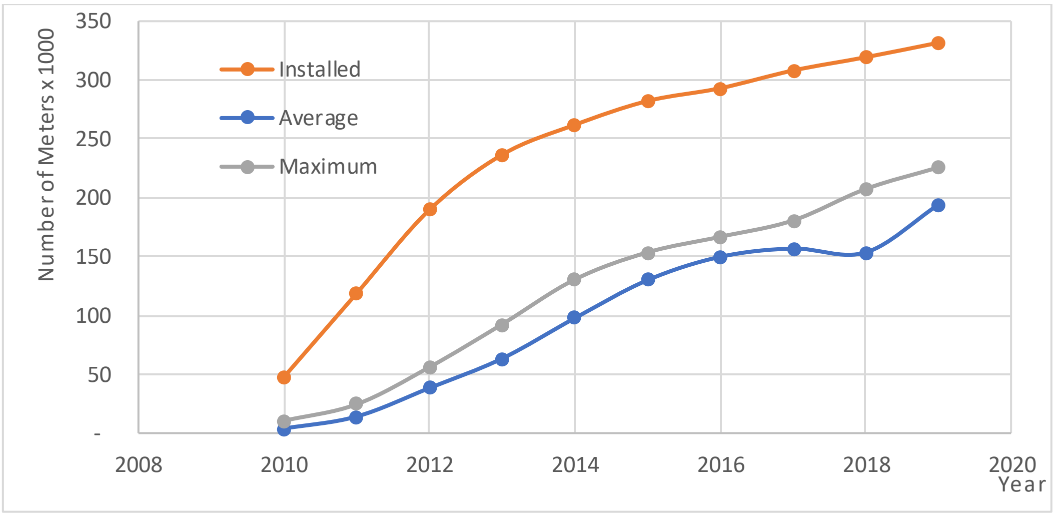

Figure 2.3. Number of smart meters installed and daily spot readings for each year.

For imputation and anomaly detection, the load profile data is considered. A preliminary study was performed on commercial consumers data from January 2013 onwards, in order to understand how many such meters have complete load profiles over how many days. This analysis showed that during the period 1st January 2016 to 31st May 2019 the average gap was of 4.51 days with a standard deviation of 21.22

days. For this period, as shown in the histogram of Figure 2.4 (Gap Max for 2016), there are 639 meters which have only 2 days of missing load profile. There are 3 meters which have only one day of missing load profile. For 335 of these meters the consumption for the missing days can be calculated from the spot readings and so these meters were selected for testing of the imputation algorithm. The importance of considering only users with a fully complete load profile is due to the fact that the missing data in this study will be simulated so that the error of the estimated values can be exactly calculated.

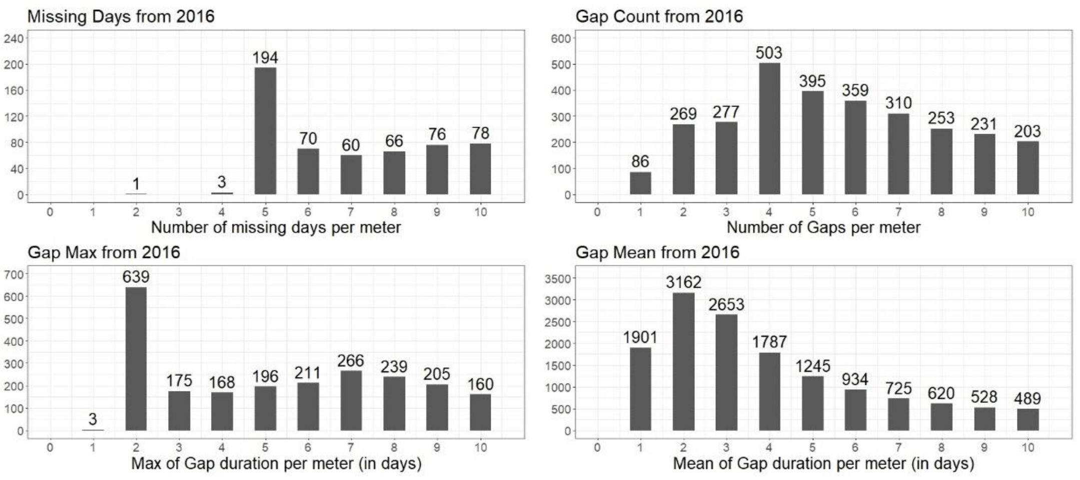

Figure 2.4. Histograms for data between 1st January 2016 to 31st May 2019.

For detecting anomalies, the approach taken in this thesis was to analyse the consumption patterns of consumers to detect those with irregular behaviour. However, consumers do normally change consumption behaviour according to the seasonal period. So, the period of observation should not span over multiple seasons so that the behaviour of most consumers will be as regular as possible. In view of this it was decided to consider 17 consecutive weeks starting 1st June 2018. For this period 3,500 commercial consumer meters were found to have a full load profile with no missing data. From the 3,500 consumers with complete load profiles, those consumers whose maximum half hourly consumption did not exceed 3KWh where excluded. For these consumers the consumption was found to be very irregular and would disrupt the analysis process. These consumers may be premises, like garages, which are still considered as commercial but are used very inconsistently and for very short times. The installations which have a renewable energy generating system have also been eliminated. Such installations have two meters each, one being the main meter, which records the imported and exported energy, and the other being the generator meter, which records the generated energy. In order to calculate the consumption for these installations the load profiles for both meters should be available for the whole 17 weeks. Then the consumption profile would be calculated as the sum of the imported and generated energy and subtracting the exported energy. However, quite a number of installations would have some days for which the data is not available for both meters. Considering the fact that in such cases tampering with any of the meters would be of financial detriment to the consumer, it is highly improbable that the consumer will tamper with the meters. Furthermore, in case of a faulty meter, the consumer will be much more reactive and would report any doubt that the meter is not recording correctly. Based on this argumentation it is considered reasonable to eliminate consumers with renewable energy sources from the dataset. Thus, the remaining 1174 meters with complete load profiles will be used for testing of the anomaly detection methods considered.

2.2 Temporal and Spectral Analysis of Load Profiles

The consumer load profiles consist of univariate observations at equally spaced time intervals and may thus be treated as a stochastic time series. Important characteristics of a time series are seasonality and trend [9]. Seasonality is a cyclic behavior that repeats over a period of time, such as weekly, monthly or yearly. Trend may be defined as the change in the mean over a substantially long period. The *substantially long* here is very subjective and depends on the time series being observed. In the case of consumer load profiles, a change in the monthly mean consumption over a year or more may be considered as exhibiting a trend. A time series which does not have a trend is also termed as stationary. The more repeatable the behavior of the consumer, the better missing observations may be estimated.

Figure 2.5 shows the load profile for a consumer, which extends over 3 years, normalised using Equation 2.1.

$$normalised\ consumption_n = \frac{consumption_n}{maximum\ consumption} \qquad \text{Equation 2.1}$$

where n corresponds to the relative period.

The yearly seasonality is very evident, with consumption increasing considerably in the summer months and again increasing slightly for the winter months. There might also be a weekly periodic behavior, which is very common with most consumers, but this is not clearly evident from this plot. This load profile can be considered to be stationary since there is no significant change in the mean over the 3-year period.

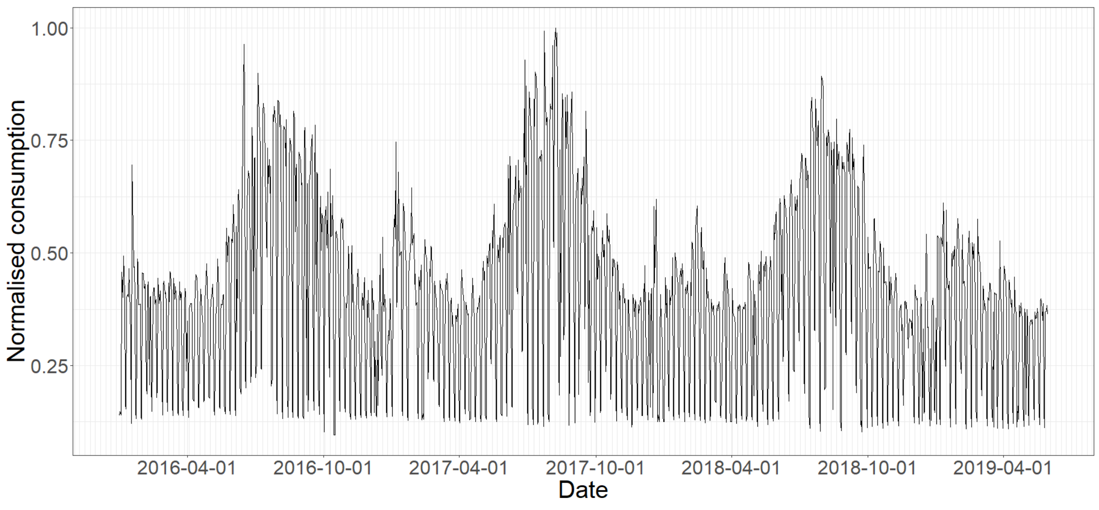

Figure 2.5. Load profile for a consumer.

Treating the load profiles as a time series allows the use of several tools which are available for time series analysis. The autocorrelation of a time series is the correlation of the time series with a lagged copy of itself and is defined as:

$$\rho_k = \frac{\sum(x_t - \mu)(x_{t-k} - \mu)}{{\sigma_x}^2} \qquad \text{Equation 2.2}$$

where ρ_k is the autocorrelation at lag k, μ is the mean and σ_x is the standard deviation of the time series.

The correlogram in Figure 2.6 shows the plot for the autocorrelation, obtained by the *acf()* function in *R*, for the load profile in Figure 2.5. The lag, in days, is iterated from 0 to the selected maximum lag, which in this case was set equal to the whole length of the series. When lag is zero the autocorrelation is actually the correlation of the signal with itself, i.e. 1. As the lag increases the autocorrelation decreases to a minimum and then starts increasing again to a localised maximum. In Figure 2.6 it can be clearly seen that the main periodicity of the oscillations is the around the 360 days lag, that is the yearly seasonality. The smaller oscillations indicate the weekly behaviour of the consumer. The two horizontal blue dotted lines indicated the 95% confidence interval and so when the correlation value lies between these two lines the two signals are not significantly correlated. When the correlation value is negative the two signals are negatively correlated i.e. an increase in one is matched with a decrease in the other, and vice versa. The correlogram gives a good visual information about the seasonality of the signal and its repeatability based on seasonal patterns.

The value of the position where the dotted blue lines are plotted is the *z*-value of the confidence interval divided by the square-root of the number of observations in the time series. In *R* the positive significant level is calculated as:

$$significanceLevel = \frac{qnorm(^{(1+ci)}/_{2})}{\sqrt{N}} \qquad \text{Equation 2.3}$$

where *ci* is the confidence interval, 0.95 in this case, and *N* is the number of observations in days. *qnorm()* calculates the quantile function for the normal distribution with mean equal to zero and standard deviation set at one.

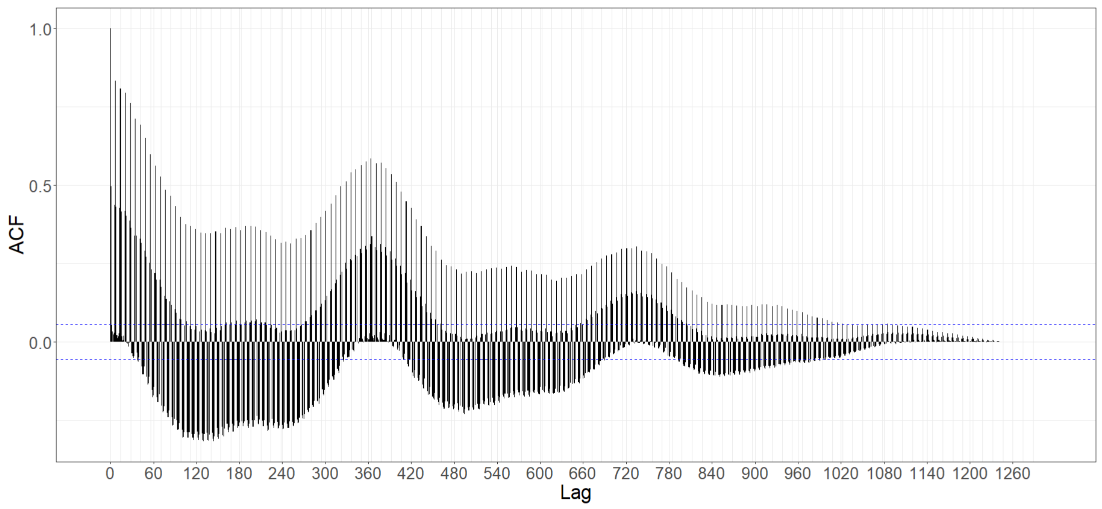

Figure 2.6. Correlogram for consumer's load profile.

The seasonality in load profile may be better analysed by transforming it into the frequency domain using the Fourier transformations [10]. The Discrete Fourier Transform (DFT) transforms a finite discrete sequence of observations in the time domain to a discrete sequence of the same size in the frequency domain. The DFT assumes that the signal is periodic with the samples in the sequence representing exactly one period.

In [10] the DFT is defined as:

$$\widetilde{H}\left(\frac{n}{NT}\right) = \sum_{k=0}^{N-1} h(kT)e^{-\frac{j2\pi nk}{N}} \qquad n = 0,1, \ldots, N\text{-}1 \qquad \text{Equation 2.4}$$

The expression in Equation 2.4 relates N samples in the time domain and N samples in the frequency domain by means of the Fourier transform. If it is assumed that the N samples of the original function $h(t)$ are exactly one period of a periodic waveform, the Fourier transform of this periodic function is given by the N samples of $\widetilde{H}\left(\frac{n}{NT}\right)$, T being the sampling interval.

If the time interval for the sequence is not exactly one period, then spectral leakage will result. Spectral leakage is the situation when a single frequency is split up into a number of neighboring frequencies. To reduce spectral leakage the sequence should be multiplied by a windowing function, such as the hamming window, so that the extremes would be smoothed. In the case of load profiles, since the load profiles are unidirectional and not zero-centered, multiplying by similar windows would interfere considerably on the shape of the signal. A windowing function will also result in reduction of frequency resolution. Instead of using a windowing function, the sequence was truncated to a size of 3 multiplied by 365, knowing, also from the autocorrelation plot, that the signal has yearly periodicity.

The Fast Fourier Transform (FFT) is a method of computing the DFT in a time-efficient manner and in *R* the function *fft()* is available. Figure 2.7 shows the magnitude (red) and phase (blue) of the of the FFT for the normalized consumer's load profile of Figure 2.5, truncated to 1095 (3 multiplied by 365, three years) observations and with the DC component (frequency zero) removed. The DC component is substantial since the waveform is unidirectional (all positive) and so removing it makes the other peaks, which correspond to frequencies of repeatability in behavior, more distinguishable. The frequency, on the horizontal axis, is in days^{-1} and represents the reciprocal of the period. The resolution in the frequency domain is $1/N$ where N is the number of observations, so here the frequency resolution is 1/1095 days^{-1}. These peaks occur at 313 (1095 divided by 313 ≈ 3.5 days), 157 (1095/157 ≈ 7 days), 3 (1095/3 ≈ 365 days) and 6 (1095/6 ≈ 182 days). Therefore, it is evident that this consumer has a behavior which shows periodicity about these number of days.

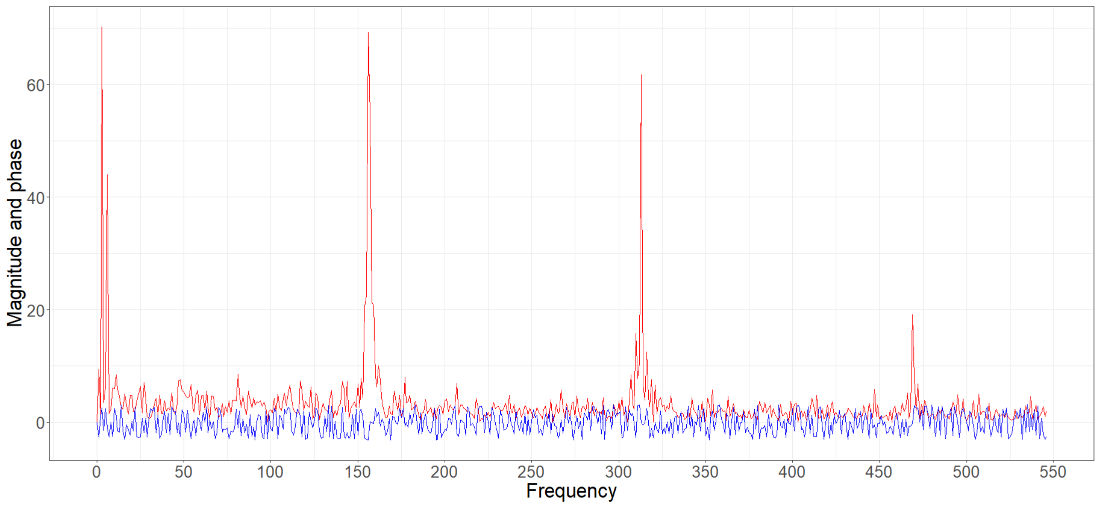

Figure 2.7. Magnitude (red) and phase (blue) of the FFT for consumer's load profile.

Another method to study the periodicity in the behaviour of a consumer is by observing the Power Spectral Density (PSD) through a periodogram. In *R* the *periodogram()* function, from package *TSA*, produces a periodogram as shown in Figure 2.8 for the above load profile of Figure 2.5. From [11] the periodogram of a time series $\{x_1, . . . x_n\}$ is the function:

$$I_n(\lambda) = \frac{1}{n}\left|\sum_{t=1}^{n} x_t e^{-it\lambda}\right|^2 \qquad \text{Equation 2.5}$$

where λ is one of the Fourier frequencies ω_k, e is Euler's number and $i = \sqrt{-1}$.

The horizontal axis in Figure 2.8 represents the frequency which is the reciprocal of the period in days i.e. a frequency of one represents a period of one day and a frequency of half represents a period of two days and so on.

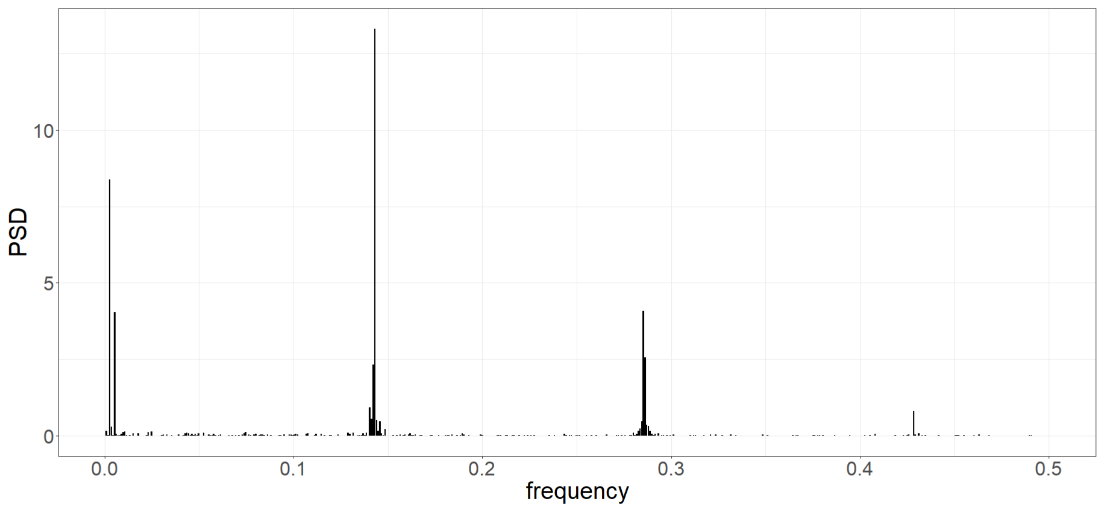

Figure 2.8. Power Spectral Density (Periodogram) for consumer's load profile.

Unlike the Fourier transform the periodogram singles out the frequency components very distinctively and thus PSD values were sorted in descending order of magnitude, as shown in Table 2.2, to highlight the most pronounced periodic behavior. As can be seen in Table 2.2, the maximum PSD is at 7 days, which is the weekly pattern. Then comes the yearly and half weekly cycles. The yearly period is given as 375 days, instead of 365, because the *periodogram()* function does not use exactly the number of observations in the sample, but a number which is convenient for computing the FFT. The effective number of observations used may be found in the *n.used* element of the returned *periodogram* object. In this case *n.used* was 1125 instead of the size of the whole sample which is 1095. So, using N=1125, the period is 1125/3 = 375.

Table 2.2. Most pronounced PSDs with the frequency and periods

PSD	Frequency	Period/days
13.315	0.143	6.988
8.376	0.003	375.000
4.084	0.285	3.505
4.040	0.005	187.500
2.569	0.286	3.494
2.333	0.142	7.031
0.925	0.140	7.120
0.804	0.428	2.334
0.553	0.141	7.075
0.525	0.144	6.944
0.481	0.146	6.860
0.468	0.284	3.516
0.349	0.287	3.483
0.312	0.288	3.472
0.307	0.004	281.250

In determining the frequency content, the DFT and PSD assume stationarity i.e. that the signal does not show a significant trend, otherwise the trend may be interpreted as an inherent frequency. The Wavelet Transform (WT) may be used to analyze time series which contain nonstationary components at different frequencies. The WT makes use of a prototype wavelet, as shown in Figure 2.9, and stretches and shifts it along the signal's waveform, using short windows at high frequencies and long windows at low frequencies. Assuming a time series, x_n, with equal time spacing δt and $n = 0 \ldots N - 1$. Also, assuming a wavelet function, $\psi_0(\eta)$, that depends on a nondimensional time parameter η. To qualify as a wavelet, a function must have zero mean and be localized in both time and frequency space. An example is the Morlet wavelet consisting of a plane wave modulated by a Gaussian [12]:

$$\psi_0(\eta) = \mu^{-1/4} e^{i\omega_0 \eta} e^{-\eta^2/2} \qquad \text{Equation 2.6}$$

where ω_0 is the nondimensional frequency.

Figure 2.9 shows a plot of the Morlet wavelet with the green line being the real part while the red line is the imaginary part.

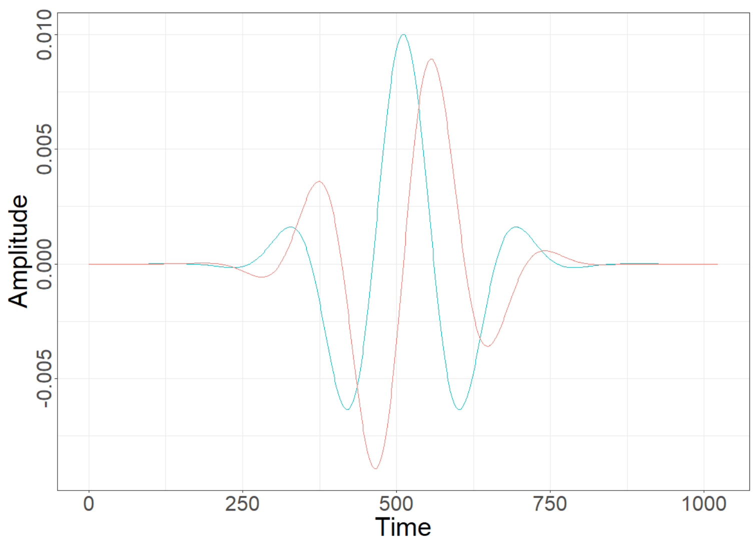

Figure 2.9. The Morlet wavelet with real in green and imaginary in red.

The continuous wavelet transform of a discrete sequence, x_n, is defined as the convolution of x_n with a scaled and translated version of $\psi_0(\eta)$ [12]:

$$W_n(s) = \sum_{n'=0}^{N-1} x_{n'} \psi^* \left[\frac{(n'-n)\delta}{s}\right] \qquad \text{Equation 2.7}$$

where the (*) indicates the complex conjugate and ψ is the normalized value of ψ_0. By varying the wavelet scale, *s*, and translating along the localized time index *n*, a scalogram is constructed showing both the amplitude of any features versus the scale together with the amplitude variations with time.

Hence, the WT represents the signal in three dimensions i.e. time, frequency and power level, in a scalogram [13]. The scalogram in Figure 2.10 shows the wavelet power spectrum for the same load profile of Figure 2.5. Time is spread along the horizontal axis, while the period is along the vertical axis. The wavelet power levels are color coded as indicated in the legend, blue representing the lower values and red representing the higher value. The scalogram was obtained by using the *R* function *analyze.wavelet()* from the package *WaveletComp* and plotted using function *wt.image()*.

The region which is lighter at the top left and right denotes the outside of the Cone Of Influence (COI). Within the COI the wavelet coefficient estimates are reliable, while outside the COI the coefficient estimates are unreliable due to edge effects. From observing the scalogram, it can be seen that there is a continuous band of high intensity at around the period of 7 days, one week, which is predominant throughout the whole span of 1246 days. Another band of high intensity is around the 365-days, 1 year, period. Some periodicity also exists at about 3.5 days, but with less consistency. There is still some periodicity at around 2.5 days, but this might be some cyclic function within the week. The thin black lines are termed wavelet ridges which are the maxima points of the normalized scalogram. The white lines contour the areas where the power is considered significant i.e. which exceeds the 95% confidence interval.

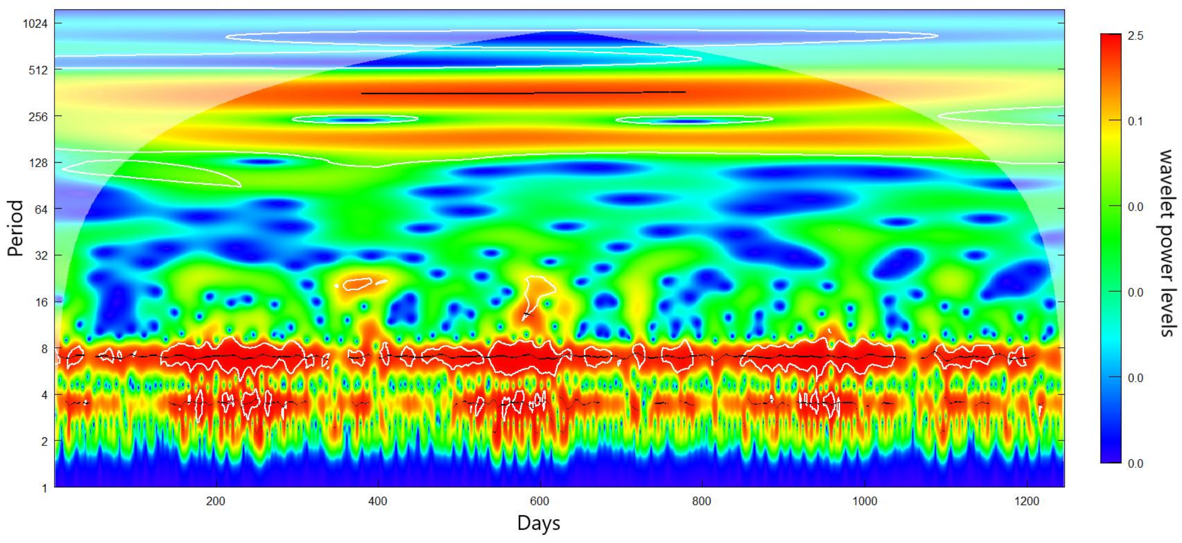

Figure 2.10. Wavelet representation of load profile.

The temporal autocorrelation, and spectral FFT, PSD and WT, analysis have shown that there is a strong periodic and cyclic element in the behavior of a typical consumer. Such cyclic and repetitive behavior indicate that both the imputation and anomaly detection tasks being considered in this thesis may be performed to a good level of accuracy. Common methods, as reported in literature to perform these tasks will be reviewed in the next Chapter.

3 Literature Review

As discussed in Chapter 2, it is inevitable for a number of consumers to have a certain amount of missing load profile data, instigating the need of an imputation method which estimates the missing values so that further analytical tools can be utilised. In Section 3.1 a thorough literature review is performed in order to determine which imputation techniques are considered for similar datasets in similar situations to be in a position to form an opinion to select an appropriate method.

In Section 3.2 literature is reviewed for anomaly detection methods as a general concept, and then delves deeper into NTL anomalies in particular, to establish an approach which would be applicable to Enemalta's consumer load profiles.

3.1 Imputation

Missing values is a common problem in real-world applications where data is measured and recorded (politics and political behaviour [14], financial stock market data [15], weather data [16]). What may determine the effectiveness of a dataset is the percentage of missing data. Any statistical or analytical process, to be performed on the recorded data, relies heavily on the completeness of the data. Hence, before any statistical or analytical tool can be used, the missing portions in the dataset have to be imputed (imputation being the process of filling of the missing values). After imputation the dataset should be considered as a complete dataset.

3.1.1 Terminology

Various works [14 , 15, 17 - 19] use the terminology Missing Completely At Random (MCAR), Missing At Random (MAR) and Missing Not At Random (MNAR) in discussion and analysis of missing data. The earliest reference found of this terminology is given by D. B. Rubin [20] who used these terms to classify missing data in his general discussion on the subject. R. J. A. Little in the book with D. B.

Rubin [21] continued using the same terminology. The three terms are also explained with examples in [17].

These explanations and examples are summarized below:

- **Missing Completely At Random (MCAR)**

Data are missing completely at random (MCAR) if all data points have the same probability of going missing. In this case, the missing data is unrelated to the model.

- **Missing At Random (MAR)**

Data are missing at random (MAR) if the missingness of a data point is random but dependent on some other observed variable in the dataset. As an example [17] gives: "a response for annual household income on a survey may be missing for several reasons. One reason is that a respondent may not know his or her household income. The missing-data mechanism may be a function of a respondent's age (e.g., very young respondents often do not know their families' incomes) but not a function of the respondent's household income."

- **Missing Not At Random (MNAR)**

Data are missing not at random (MNAR) when the missingness of a data point depends on the value of the variable which is itself being measured, i.e. values with some range, or ranges, are more prone to go missing. As an example [17] gives: "In time series, this might mean that the missing data occur in patterns or are related to the numeric values of the series (i.e., when the number of drinks consumed is very high, a subject may not record the amount)."

3.1.2 Imputation in Time Series

A time series is a sequence of signal observations typically measured at uniform time intervals. A univariate time series measures one signal while multivariate time series measures more than one signal for every time interval. Like any dataset, a time series is also prone to have missing values which need to be imputed.

S. Oh [15] used multiple imputation methods on missing values for a financial stock market dataset. The concept of multiple imputation is also used, and is explained, by Honaker and King [14] where they use multiple imputation when analysing American politics and political behaviour data. The idea is to extract relevant information from the observed portions of a data set via a statistical model, to impute multiple (around five) values for each missing cell, and to use these to construct multiple completed data sets. In each of these data sets, the observed values are the same, and the imputations vary depending on the estimated uncertainty in predicting each missing value. Honaker and King argue that the great attraction of their procedure was that after imputation, analysts can apply on each of the completed data sets any statistical method they would have used if the data contained no missing values and then use a simple procedure to combine the results.

Velicer and Colby [17] compared methods for imputation of AutoRegressive Integrated Moving Average (ARIMA) time series. The authors compared four methods of handling missing data: deletion, mean of series, mean of adjacent observations and maximum likelihood, with maximum likelihood yielding the best results. The four techniques were tested on computer simulated data. The authors argue that with simulated data the factors that may affect the outcome could be systematically manipulated. Furthermore, the use of simulated data provided population parameter values (criterion values) against which estimates could be compared. The accuracy of the estimation was also tested against the proportion of missing data. It was concluded that the higher the percentage of missing data, the poorer the model's overall fit, with poor fit beginning to occur when the percentage is more than 20%.

Caparino et al. [22] explored the effect of missing data on well-known classifiers: Naïve Bayes, One-R, K-Nearest Neighbours (KNN), C4.5 and Support Vector Machine (SVM). Results show that classification performance improves when the Modified Imputation Method (MIM) is applied to the data sets which had missing values during pre-processing. MIM is only suitable for multivariate datasets and uses Spearman correlation to find the attribute which best correlates to the missing data. The missing values are then calculated from the values of the correlated attribute.

Makaba et al. [23] also compare a number of strategies for imputing missing values and their effect on machine learning algorithms. The methods considered are Listwise Deletion (LD), mean, mode, KNN, Expectation-Maximization Single Imputation (EMSI) and Multiple Imputations by Chained Equation (MICE). These imputation methods are used on two real-life datasets, one numeric and one categorical, which are then used for six machine learning algorithms: Logistic Regression (LR), KNN, SVM, Random Forest (RF), Naïve Bayes (NB) and Artificial Neural Network (ANN). The results were evaluated based on the following performance metrics: accuracy, Root Mean Squared Error (RMSE), Receiver Operator Characteristics (ROC) and the F1-score. For the categorical dataset the classifiers showed slightly improved performance when the missing data was imputed using mode as compared to using the LD method. As for the numeric dataset there was very marginal performance difference, with EMSI and KNN performing slightly less overall.

The language *R*, being a widely accepted and versatile language for statistical analysis, is quite often used in imputation research. Different methods for univariate time series imputation were performed by Moritz et al. [18] to compare the usage of some functions available in *R* packages. Although several *R* packages (*Amelia*, *mtsdi*, *VIM*, *mice* and *imputeR*) declare that they can process univariate time series, the authors found out that practically only packages *forecast* and *zoo* do actually accept a univariate time series. The other packages only accept a multivariate time series input. The functions tested and compared were: *na.aggregate, na.locf, na.StructTS, na.interp, na.approx, ar.irmi*. The experiments were performed with four different missing data rates (10%, 30%, 50%, 70%). A self-developed algorithm was used which simulates data MCAR. The performance of each function was measured using Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) for the imputed values. The results obtained in this work show that the accuracy of the data imputed by each function depends hugely on the trend and seasonality of the dataset being imputed. Neither function performed well with all datasets which were used for testing.

Moritz & Bartz-Beielstein [24] review the *imputeTS* package in *R*. *ImputeTS* has a number of functions which perform imputation on univariate time series datasets.

However, they also concluded that the imputing function has to be chosen depending on the type of dataset.

Kim et al.[16] use and compare four methods of imputation for estimating the missing instances in meteorological data used for PV generation forecasting. The methods used are Linear Interpolation (LI), Mode Imputation (MI), KNN, and MICE. For LI a simple linear interpolation was done. This method performed well when the number of consecutive missing values was small but when the missing data interval lengths increase, the imputation accuracy degrades severely. MI is a method where missing data is replaced by the most frequently observed data. The authors state that the MI method is not suitable for the dataset being analysed, nonetheless the method was selected in order to show that when the missing data are processed in an undesired way, the prediction deviates drastically. Hence it is evident that the method of imputing missing data is very much dependent of the type of data itself.

Rahman el al. [19] develop a method for imputing missing values in multivariate time series. The method is a combination of two methods, an extension of KNN imputation with lagged correlations and the Fourier transform. The KNN approach calculates the time lagged correlations, using cross-correlation, also for the other variables in the multivariate time series, and selects the *k* nearest neighbours. These *k* values are then averaged, weighted by the strength of their correlation. The Fourier transform method computes the DFT for all the non-missing values (which should be complete, even if some of the values have been previously imputed) then the inverse DFT is performed on the frequency components to obtain the series in the time domain, comprising also the missing part. The imputed value is the average of the values obtained from the two methods.

Anava et al. [25] developed an online forecasting method, using an autoregressive model, in the presence of missing values. Autoregressive prediction is not well defined when some of the data is missing and so, to overcome this issue, the authors define a new family of autoregressive predictors. Each predictor makes use of its own past predictions to fill in the missing data, and then provides an autoregressive prediction using the completed data.

Murti et al. [26] use a *k*-nearest neighbour based method to impute data for missing values in 8 journal statistics datasets. Different quantities of missing values were simulated for each dataset and the estimated values where compared to the original ones. The KNN method used consists of calculating the Euclidian distance of the missing data instances from the complete data and then choosing the *k* instances with least distance. The imputation value is then calculated by finding the weighted average of the *k*-nearest neighbour values. The weighting used is the reciprocal of the squared Euclidian distance for each of the *k* instances.

Yodah et al. [27] examine the imputation of incomplete non-stationary seasonal time series data. The authors explore the appropriateness of Box-Jenkins approaches, Seasonal AutoRegressive Integrated Moving Average (SARIMA) and AutoRegressive Integrated Moving Average (ARIMA) models, in handling non-stationary seasonal time series with missing observations. They also discuss direct Linear Regression approaches in imputing missing observation when seasonality has been relaxed by rearranging the series in periods and then treating each period as a single series. The 3 methods are compared on 5 datasets having 5% and 7% missing data. Linear regression was found to be more efficient and effective than ARIMA and SARIMA models however it may not be applicable to all types of series.

3.1.3 Imputation in Smart Meter Data

During the last decade and a half, smart grids have been gaining in popularity and smart meters are an essential element of these systems. The data collected from smart meters can be stored and analysed. Wang et al. [28] and Liu et al. [29] provide a collection of review and benchmark analytical methods used on smart meter data. These analytical methods can be utilised for several objectives, such as for consumer characterisation [30 - 33]. These works explore methods which study consumer load profiles to establish consumer categories in order to tailor tariff packets which might be attractive to certain segments. Such methods could also be used by the utility to implement time-of-use packages which would be beneficial to both the utility as well as the consumer. However, for these analytical methods to be effective the load profile datasets should be free of any missing intervals. Hence the need for imputation.

In recent years considerable work on imputation of smart meter data has been done and published. Al-Wakeel *et al.* [34] suggest a clustering approach. In this work the available load profiles are first separated into a predefined number of clusters by using a *k*-means algorithm. Then the missing data is estimated, based on the average behaviour of the cluster to which it belongs. Yu et al. [35] use a similar approach to [33] first clustering by using *k*-means then uses KNNs to impute the missing data within groups. Here the authors term their method as Cluster-based Best Match Scanning (CBMS). These methods may not be as effective on occasions when the communication is lost to a large number of meters at the same time. In these instances, it would be difficult to estimate the missing data for a meter from adjacent meters in the same cluster because the data for these meters would most probably be missing too.

For long periods of missing data Peppanen *et al.* [36] present an Optimally Weighted Average (OWA) load power data imputation method. This method only requires the historical load power measurements from the smart meter itself. The proposed load data imputation scheme leverages two typical load data characteristics. First, the data tends to be rather continuous over a short time interval, meaning that short time intervals of missing/bad measurement samples have likely similar characteristics as the adjacent available data. Second, since the load profile data is strongly driven by human consumption patterns, the data tends to have similar characteristics over time periods with similar human activity. For example, the data characteristics of weekdays tend to be different to weekend days, mornings different to evenings, etc.

Ryu et al. [37] use a Denoising Convolutional Autoencoder (DAE). An autoencoder is a type of artificial neural network which can be summarised by two nonlinear encoding and decoding functions. For a given input vector, the network encodes the input into a latent feature, then decodes it back to the original vector. Therefore, the autoencoder learns to extract a feature vector that contains crucial information of input data. DAE is an autoencoder having denoising property, able to reconstruct the input vector from a partially corrupted version. The authors use the load profile data with missing values as the corrupted input vector and use DAE to construct the complete load profile.

Mateos and Giannakis [38, 39] use Principal Components Pursuit (PCP) to do load profile cleansing and imputation. PCP can recover the low-rank matrix (the principal components) when the data matrix is corrupted by gross sparse errors. The load profile cleansing and imputation involves identification and removal of outliers, or erroneous data, in addition to completion of the missing values from the nominal load profile matrix, and denoising of the observed ones.

Kim et al. [40] use the historical load profile data collected for the same consumer. The sequence around the missing portion, is compared to sequences of the same length in the past. KNN is employed to find the best *k* matches which are then used to estimate the values of the missing portion. A learning phase is conducted to determine the optimal values for *k*, the length of the surrounding sequence, and the historical length to search.

3.2 Anomaly detection

Smart meters generate considerable data which, apart from the conventional billing purposes, can be used much more vastly. The information that lies in the data is extensive and can be utilised in a variety of ways. Wang et al. [28] review the research being conducted on smart meter data for load forecasting, abnormal usage detection, consumer segmentation, and demand response. Liu et al. in [29] examine smart meter analytics from a software performance perspective designing a performance benchmark for typical smart meter analysis tasks which include off-line feature extraction and on-line anomaly detection. Hidayatullah et al. [41] perform an analysis of smart grids and the challenges for the future including climate change, escalating energy prices, energy security and energy efficiency. Liang et al. [42] use analytical methods for load forecasting.

In recent times, with the liberalisation of electricity markets, Distribution System Operators (DSO) are getting more interested in the individual electricity usage patterns of their consumers. Having this knowledge, they are able to classify consumers into categories based on their different behaviours. This might help the DSO to design new tariff packages, which might be beneficial to both parties, and also to channel investment to the most rewarding segments. Such consumer behaviour is studied by taking their load profiles and using clustering techniques to group them into categories [30 - 33 , 43, 44].

3.2.1 Clustering Techniques

Clustering is an unsupervised technique which groups together observations which are close to one another in some multidimensional space. The data produced by smart meters is massive, and before clustering can be performed, some kind of dimension reduction is required. Feature extraction is performed with the aim to retain only the relevant information and discard what is redundant [45].

Beckel et al. [46] establish a list of 22 features based on consumption values of individual days as well as aggregated over the entire week or over workdays and weekends separately. These features are grouped into four categories : consumption figures, ratios, temporal properties and statistical properties. Kopf el al. [47] continue to build on the features presented in [46] to establish 88 features grouped in the same categories.

Palacio-Nino and Berzal [48] discuss evaluation metrics for unsupervised learning algorithms. Kupta and Panda [49] do a validation of clustering methods Clustering Large Applications (CLARA) and *k*-Means using Silhouette and Dunn indexes. The Silhouette and Dunn indexes are metrics for evaluating the effectiveness of a clustering algorithm. Halkidi et al. [50] discuss the clustering process: feature selection, clustering algorithm, validation of results and interpretation of results. They discuss the main clustering and validation techniques.

Brock et al. [51] give the documentation for the *clValid* package in *R*. *clValid* contains functions for validating the results of a clustering analysis. The user can choose from nine clustering functions which are available in *R*. Three cluster validation measures are available and are termed: *internal, stability* and *biological*. The metrics used are dependent on the validation measure selected.

3.2.2 Anomaly detection

Anomalies are patterns in data which do not conform to a well-defined normal behaviour. Chandola et al. [52] present an extensive survey of anomaly detection techniques available in literature. The applications covered in this work are Cyber-Intrusion Detection, Fraud Detection, Medical Anomaly Detection, Industrial Damage Detection, Image Processing, Textual Anomaly Detection and Sensor Networks. For these applications the techniques used are: Classification Based, Clustering Based, Nearest-Neighbour Based, Statistical, Information Theoretic and Spectral.

Hodge and Austin [53] also conduct a survey of contemporary literature which deals with outlier detection methodologies. The methodologies mentioned are Statistical Models (Proximity-based, Parametric, Non-parametric and Semi-parametric), Neural Networks (Supervised and Unsupervised), Machine Learning and Hybrid Systems (which use at least two of the other techniques.) T. Penvy [54] uses a collection of weak classifiers which result in a strong classifier.

Aligholian et al. [55] evaluate the performance of four unsupervised machine learning methods for abnormality detection on real-world smart meter data, namely prediction-based regression, prediction-based neural network, clustered-based and projection-based methods. Different types of features, such as load-based, contextual, and environmental, are investigated to construct the data-driven models.

3.2.3 Non-Technical Losses as Anomalies and their Detection

Non-Technical Losses (NTL) have been researched quite vastly since it presents a loss of revenue for DSOs. A good part of NTL is made up of faulty meters and fraud but it also consists of errors in billing. Fraud may be accomplished by tampering with the metering setup. Consumers with faulty or a tampered metering setup would most probably have consumptions which are anomalous when compared to others which are healthy and not tampered. Hence anomaly detection methods may be used in order to single out the NTL from the benign consumption.

The main methods used for NTL detection namely Support Vector Machine (SVM), Artificial Neural Networks (ANN) and other less wide spread methods are discussed in [56 - 61] and are reviewed below.

3.2.3.1 Support Vector Machines

Support Vector Machines (SVM) are widely utilised in the literature [62 - 70] to classify the load profiles of customers for detection of energy-theft suspects. SVM is a supervised machine learning method meaning that the training has to be done on labelled data. As cited by Hearst et al. [62] and Ghori et al. [60], Vapnik has proposed the SVM classifier that creates a margin between the two classes and tries to maximise that margin. SVM is a set of machine learning methods which offers support for outlier detection, regression and classification. Toma et al. [63] use Principal Component Analysis (PCA) for dimension reduction before using the data to train the SVM.

Nagi et al. [64] extend the work done by themselves in [65] by integrating human intelligence and knowledge into the SVM-based fraud detection module with the introduction of a Fuzzy Inference System (FIS), in the form of fuzzy IF-THEN rules, as a postprocessing scheme. The FIS acts as an intelligent decision-making system together with the SVM-based detection model in [65] to shortlist customers suspected with high probabilities of fraud activities and abnormalities. The authors claim that the introduction of the FIS has increased the hit rate from 60% to 72%.

Glauner et al. [66] use monthly consumption readings for the SVM classifier. The geographical area is split into differently sized neighbourhoods. The authors claim that certain areas are more prone to cause NTLs than others, so features based on the neighbourhood area are interesting in order to improve predictions. The neighbourhood information is added as a features to the SVM classifier.

A significant concern when using supervised machine learning tools in dealing with NTL, is excessive class imbalance. Normally the percentage of consumers with NTL is very small compared to the whole population of consumers and so only a few examples of the positive class (NTL present) are seen as compared to the negative class (NTL not present). This underrepresentation of the positive class heavily affects classification.

Figueroa et al. [67] deal with the imbalance issue by over-sampling the minority class and under-sampling the majority class. Dealing with over-sampling and under-sampling involves adding and removing NTL records, so as to obtain the desired percentages in the training set. Glauner et al. [68] evaluate the performance of Boolean, fuzzy and SVM for varying NTL proportions on imbalanced real world consumption data. An optimised fuzzy model and SVM were found to significantly outperform the Boolean and unoptimized fuzzy model.

Messinis et al. [69] use meter topology data, together with SVM, to extract theoretical voltage values. As a dataset they use the Irish Commission for Energy Regulation (CER) data set [71] and modelling of NTL is simulated. Jindal et al. [70] use Decision Tree (DT) to predict the electricity consumption from the consumer features (number of heavy appliances, number of persons, season, time slot, and temperature) in real time. Then the predicted consumption, together with the consumer features are fed to the SVM classifier, which flags the consumer as good or bad. The authors claim to have reduced the occurrence of false positives and increased the detection rate.

3.2.3.2 Artificial Neural networks

Artificial Neural Networks (ANN) like SVM are supervised machine learning classifiers. Nizar et al. [72, 73] use Extreme Learning Machine (ELM). The ELM is a general learning algorithm for neural networks, which works for function approximations, classifications, and online prediction problems. The authors compare the results obtained by using the ELM to those obtained by using the more popular SVM, and claim that ELM proved to be better, both in speed and accuracy.

Yuan and Jia [74] incorporate an Internet Of Things (IOT) module so as to cater for smart meters from different manufacturers. A basic autoencoder neural network structure acts as a classifier. Hu et al. [75] use a semi-supervised deep-learning-based model with powerful feature extraction ability where labelled and unlabelled data can be utilised for training.

Huang et al. [76] use a two-layer feed-forward ANN whose inputs are the consumption of each consumer together with the entire consumption measured at the substation, and whose output is an honest coefficient for each consumer. The training of the ANN is done by simulating 2^N (N being the number of consumers on the substation) combinations of energy theft situations and randomly assigning the level of NTL to each combination.

3.2.3.3 Use of Master Meter for Anomaly Detection

A number of research studies [77 - 80] make used of a master meter installed at the low voltage side of the transformer feeding a number of smart meters to perform energy balance analysis. The addition of all the consumptions recorded by smart meters connected to the transformer, together with any technical losses, should be the same as the consumption recorded by the master meter at any time. Anything less would be NTL. Yip et al. [77, 78] devise a method to find what they term as the ‘anomaly coefficient’ for each meter.

Jokar et al. [79] compare the difference in consumption between the smart meter and all the smart metres on the transformer and when a difference is registered, all the smart meters which are fed from the transformer are selected and processed by a SVM classifier to determine which consumer is causing the NTL. Liu et al. [80] use the same principle of [79] but use a deep-learning method based on Long Short-Term Memory (LSTM) and a modified Convolutional Neural Network (CNN) to identify the malfunctioning smart meter.

3.2.3.4 Dedicated Hardware Solutions

Some researcher opted to use some hardware device connected to the smart meter in order to detect NTL. Dineshkumar et al. [81] developed an electronic module which sends an SMS on the GSM network when fraud is detected. Khoo et al. [82] proposed an RFID communication from the meter and performs a cost benefit analysis to justify the use of the module.

3.2.3.5 Additional Methods of NTL detection

Messinis and Hatziargyriou [83] use an unsupervised Bayesian detection rate (BDR) method. In this work the authors make use of the *BreakoutDetection R* package [84] for detecting sudden changes (breakouts) in the consumers load profile. For clustering this work uses PCA with four features: normalised change in mean (the difference between the mean consumption before and after the breakout, divided by the yearly mean), change in standard deviation (the difference in standard deviation before and after the break, divided by the standard deviation before the break), Im (Imaginary part of the normalised time series Fourier transformation) and Cos (Second component of the normalized yearly time series discrete cosine transformation). *k*-means, fuzzy *c*-means (FCM) and DBSCAN are used for clustering. Anomalies are simulated on two datasets: the CER dataset [71] and another dataset from Greek DSO.

Punmiya and Choe [85] use a Gradient Boosting Classifier (GBC) which is claimed to have better detection rate and improved false positive rate over other machine learning classifiers. Theft records used for testing the classifiers are simulated on the dataset. Saad and Sisworahardjo [86] use a dissimilarity matrix which is calculated for all pairs of instances which are in the same temporal instance. Then clustering is performed on the dissimilarity metric using Partitioning Around Medoid (PAM). An initial anomaly score is obtained by calculating the Euclidean distance of each instance to the medoid. Finally, the initial anomaly score is normalised to get the anomaly score in the range of 0 to 1.

Buzau et al. [87] use supervised machine learning algorithm Extreme Gradient Boosted Trees (XGBoost) for detecting NTL. Cabral et al. [88] use two methods: Self Organising Maps (SOM) and the other is a hybrid of data mining techniques. This work clusters the weekly behaviour of a single user and detects weeks with abnormal behaviour for the user.

Kee et al. [89] designed and developed a GUI-based NTL detection platform consisting of energy balance fuzzy logic and SVM. Avila et al. [90] use a maximal overlap discrete wavelet-packet transform for feature extraction and random under-sampling boosting for NTL deduction.

To overcome the imbalance problem Zhang et al. [91] propose an anomaly detection framework called semi-supervised generative Gaussian mixture model, which can be controlled using human detection indicator thresholds to adjust the intensity of detection.

Massaferro et al. [92] propose an NTL detection solution that prioritises on the potentially larger economic return. The list of potential fraudsters flagged for inspection is sorted according to the expected economic gain, factoring in also the cost of the inspection.

Being an extension of the IP network, the smart grid may be prone to cyber tampering. Consumers might be able to tamper with the data which is exchanged between their smart meter and the main server. Researchers also explored the cybersecurity of the smart grid [93 - 97].

3.3 Conclusion

This literature review was conducted to gather information on the methods which are available, and which are mostly used in the fields of data imputation and anomaly detection, to determine which are most appropriate for the Enemalta load profile dataset presented in Chapter 2.

Data imputation is used extensively in a variety of applications since missing data is a common problem in various areas. Initially the general methods of imputation were reviewed and then those specifically used for smart meter missing data. In most of the literature about smart meter data imputation the missing data was only spanning a number of hours, while in the Enemalta implementation missing data is always in multiple of days. The KNN method used by Kim et al. in [40] was chosen due to its adaptability to the problem at hand, the reported accuracy and its computational efficiency.

For anomaly detection the review was conducted to explore techniques which are adopted for detecting non-technical losses in smart meters. The most utilised methods use supervised classification, such as SVM and ANN, which have to be trained on labelled data. Due to the fact that Enemalta does not currently have such labelled information an unsupervised clustering method was preferred, literature, such as [46] and [47], establish a number of features, but these are still extensive and so a means of dimension reduction is needed. PCA, as used in [63], was chosen as a means for dimensional reduction mainly due to the fact that it maximises the variance between the selected features.

4 Data Imputation

Imputation of the missing data is essential for any analytical activity to be performed on the data efficiently. It is also beneficial to have a complete load profile so that the consumer can be presented with a complete consumption pattern, if he demands it, having the imputed part selectively marked accordingly. The imputation should be calculated based only on actual data collected from the meters, so any imputed section of the load profile is flagged so that it will not be used for imputation of future missing data.

This Chapter first presents the approach taken in developing the algorithm capable of imputing the missing load profiles. Then it covers the methods used to determine the most appropriate parameters for the algorithm. Two sample consumer load profiles are then presented to illustrate the implementation of the algorithm and the results obtained.

4.1 Approach

The method used for imputation in this thesis is based on the method used by Kim et al. in [40] where the imputed values are calculated on the historical behavior of the same consumer. The method implements a KNN approach where the imputed part is calculated by searching the past consumption of the consumer for patterns that best resemble the portion around the missing values (nearest neighbors) and taking an average of the parts which correspond to the missing portion. The method of least squares is used for selecting the patterns of closest resemblance which are then averaged.

Considering the fact that, as explained in section 1.1, the missing data, or *gaps*, in load profiles is always in multiple of days, it was decided that the imputation will be carried out in two steps. As a first step the daily load profiles are considered i.e. the imputation is calculated on the consumption of the whole day. After a suitable imputation daily pattern is determined, then the half-hourly profile is calculated by averaging the half-hourly consumption from the selected days. Consequently, the imputed values are blended with the available spot reading values.

Figure 4.1 illustrates a hypothetical load profile for a consumer having a missing portion of length *m*, in days. Here it should be noted that in Figure 4.1, and other figures that follow in the rest of this section, the horizontal axis represents the number of days in the past, so samples on the right precede those on the left. The plot shows the daily consumption which is the sum of all the 48 half-hour consumptions of each day.

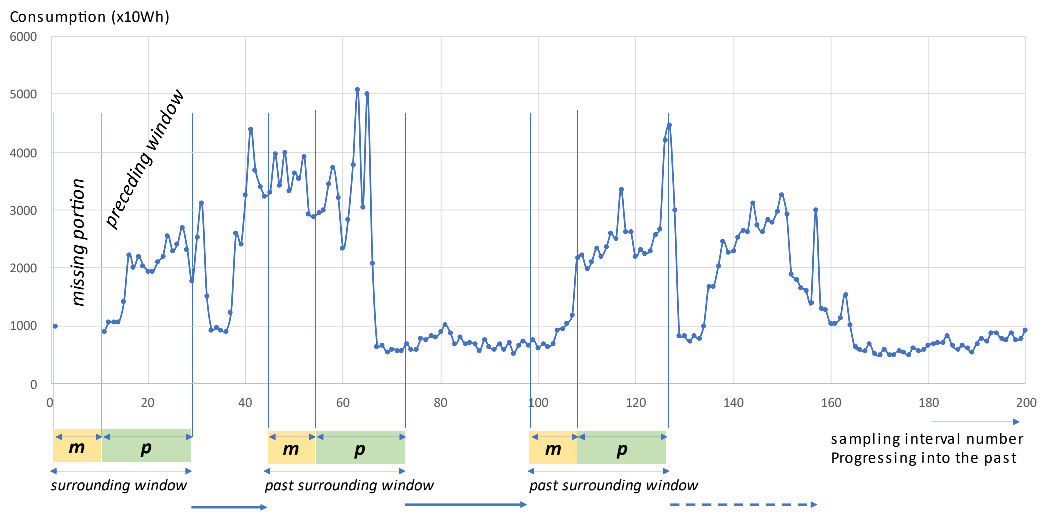

Figure 4.1. Load profile with a missing portion.

For estimating the values of the missing load profile, the *present surrounding window* needs to be considered. The *present surrounding window* is made up of one sample succeeding the *missing portion*, together with a *present preceding window* of *p* days, as shown in Figure 4.2. For every consumer, when a valid daily load profile entry is received, after a period of missing data of length *m* days, the imputation procedure is initiated. Past data is searched for patterns which resemble the *present surrounding window*. This is done by moving one sample and forming a *surrounding window* which is of the same outline as the *present surrounding window* i.e. one sample and *p* samples separated by *m* samples. This is termed as a *past surrounding window*, as in Figure 4.3.

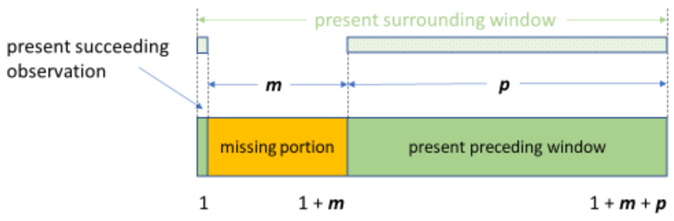

Figure 4.2. Graphical representation of the *present surrounding window*.

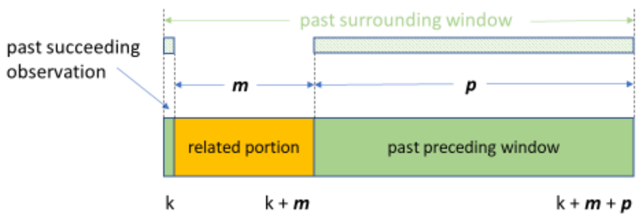

Figure 4.3. Graphical representation of a *past surrounding window*.

Figure 4.4 shows the flowchart for the imputation process which is to run every day. The process waits for one valid entry after a series of missing days of load profile and when this occurs constructs the *present surrounding window*, by adjoining the *present succeeding observation* to the *p* observations preceding the missing portion. The pointer, *n*, is then moved to just before the first value in the *present surrounding window* at $n = 1 + m + p$.

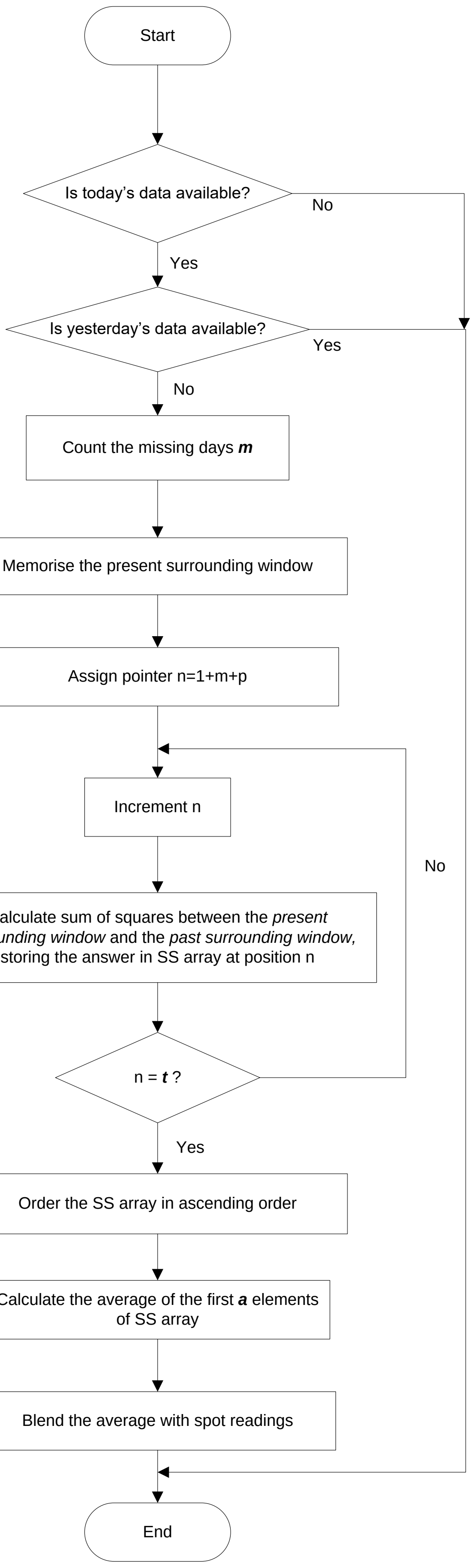
Start
Is today's data available?
No
Yes
Is yesterday's data available?
Yes
No
Count the missing days *m*
Memorise the present surrounding window
Assign pointer n=1+m+p
Increment n
No
Calculate sum of squares between the *present surrounding window* and the *past surrounding window,* storing the answer in SS array at position n
n = *t* ?
Yes
Order the SS array in ascending order
Calculate the average of the first *a* elements of SS array
Blend the average with spot readings
End

Figure 4.4. Flowchart for the imputation process.

The first *past surrounding window* starts at $n = 1 + m + p + 1$. The resemblance between the *present surrounding window* and the *past surrounding window* is computed by finding the sum squares of the difference between each element of the two arrays, which have the same length (1 + p). The smaller the sum of the squares the closer the resemblance of the two arrays i.e. profiles. The sum of squares is obtained using Equation 4.1.

$$d = \sum_{l=1}^{1+p} (present_l - past_l)^2 \qquad \text{Equation 4.1}$$

where: d is the distance, or measure of dissimilarity

$present_l$ is the value of element l in the *present surrounding window*

$past_l$ is the value of element l in the *past surrounding window*

The *past surrounding window* is moved one day at a time, by incrementing the index n, and the sum of squares, d, is calculated for each iteration. Each time the new value of d is inserted into an array SS at index n : SS[n]. After a maximum number of repetitions, t, are performed, the sum of squares array, SS, is sorted in ascending order. The indexes of the top elements in the array point to the *past surrounding windows* which best match the *present surrounding window*, these are the nearest neighbours. The first k indexes (n) are used to obtain the *related portions*, all of length m (Figure 4.3), and an average is taken to obtain the estimated daily load profile for the *missing portion*. Three types of averaging were considered: no weighting, time weighting and least-squares weighting (refer to section 4.2). To obtain the half hourly load profile for the *missing portion*, the half hourly load profiles for the same days which were used to obtain the daily load profile are averaged in the same manner.

After the averaging phase, the imputed portion is adjusted by blending in the values of any spot readings which are available. The matching process was performed on normalised values to match the behaviour, so now this blending is important to give the correct magnitude to the imputed values. Since the spot readings are taken at a time which is very close to midnight, the difference between two spot readings can be considered equal to the addition of the load profile values in between, as shown in Equation 4.2.

$$SP_i - SP_j = \sum_{n=j}^{i} LP[n] \qquad \text{Equation 4.2}$$

where SP = spot reading and LP = load profile.

Hence the values for the individual load profile values are multiplied by a factor so that their sum is equal to the difference of the spot readings. This factor is calculated by dividing the difference between two available spot readings by the sum of the imputed load profiles values between them, as shown in Equation 4.3.

$$blending\ factor = \frac{SP_i - SP_j}{\sum_{n=j}^{i} LP[n]} \qquad \text{Equation 4.3}$$

This blending process is also important so that the relationship in Equation 4.2 between load profiles and spot readings is conserved even for imputed sections.

4.2 Identifying Algorithm Parameters

The parameters p (length of *preceding window*), k (number of preceding windows considered for averaging) and t (number of samples scanned), introduced in section 4.1, are arbitrary. In order to determine appropriate values for these parameters to be used with the load profile dataset, a tuning process was conducted on a number of consumers which have full load profiles for the whole timespan under test.

As mentioned in section 2.1, the timespan of the dataset used for testing of the imputation method was selected as 1st January 2016 to 31st May 2019. During this period there were 335 consumers with very few days of missing load profiles and whose missing daily consumption could be unambiguously calculated from the spot readings. As a practical example the scenario illustrated in Table 4.1 is noted, where the load profile for 09/01/2018 was missing for a particular consumer. This was calculated from the available spot readings by using Equation 4.2. The spot reading for 06/01/2018 and 14/01/2018 are available and the consumption between these two dates is 191205417 - 190473365 = 732052. This should be equal to the sum of the load profiles between the two days. So, the consumption on 09/01/2018 is 732052 less the sum of the consumptions on all the other days from 06/01/2018 to 13/01/2018, which is 651314. So, consumption on 09/01/2016 is calculated as 732052 – 651314 =

80738Wh. Hence, the value of 80738 was inserted for date 09/01/2018 so that the load profile was completed.

Table 4.1. Calculation of missing daily consumption from spot readings.

Date	Spot Reading	Daily load profile
01/01/2018	190010311	108096
02/01/2018	190118434	95251
03/01/2018	190213546	86188
04/01/2018	190299616	92290
05/01/2018	190392056	80152
06/01/2018	190473365	87548
07/01/2018		114467
08/01/2018		83225
09/01/2018		80738
10/01/2018		92788
11/01/2018		87761
12/01/2018		90955
13/01/2018		94570
14/01/2018	191205417	101826
15/01/2018	191307188	83335
16/01/2018	191390928	88767
17/01/2018	191479465	79758
18/01/2018	191559798	81153
19/01/2018	191640454	101905

Load profiles which are sparse or do not show any sign of autocorrelation would not have their missing values reliably estimated by the algorithm and so methods were investigated to find a test which can filter out such load profiles. Ljung-Box and Box-Pierce are two such methods which are available in *R* through the function *Box.test()*.

The Ljung-Box test is defined in [98] as:

$$Q = u(u+2)\sum_{v=1}^{h}\frac{\rho_v^2}{u-v} \qquad \text{Equation 4.4}$$

where u is the sample size, ρ_v is the sample correlation at lag v, and h is the number of lags being considered. Please note that the variable names u and v have been reassigned compared to those used in the reference to avoid conflict with other variable names which are used in this dissertation.

The Box-Pierce test [99] is defined as:

$$Q_{BP} = u \sum_{v=1}^{h} \rho_v^2 \qquad \text{Equation 4.5}$$

The load profiles dataset was tested for autocorrelation by using both Ljung-Box and Box-Pierce tests. Both these tests produce a p-value (significance value) which if less than 0.05 will reject the null hypothesis which states that the series is independently distributed, without any autocorrelation. The Box-Pierce test performed slightly better than the Ljung-Box test in filtering out the sparse and uncorrelated load profiles but did not give the desired performance since there were sparse and uncorrelated load profiles which still produces p-values of less than 0.05.

The Durbin-Watson (DW) statistic is a test for autocorrelation and is defined in [100] as:

$$DW = \frac{\sum_{u=2}^{h}(e_u - e_{u-1})^2}{\sum_{u=1}^{h} e_u^2} \qquad \text{Equation 4.6}$$

where e_u is the residual at time = u, e_{u-1} is the residual at time = u-1 and h is the number of observations.

The Durbin-Watson test, like the Box-Pierce and Ljung-Box tests, are usually utilised to detect the presence of autocorrelation in the residuals from a regression process. However, in this work these functions were used to quantify the amount of autocorrelation in the load profile, since the more autocorrelation found in the load profile the better the performance of the imputation algorithm. The DW statistic lies between zero and four. A value of zero suggests the presence of perfect positive autocorrelation and a value of four suggests the presence of perfect negative autocorrelation. A value of two suggests the absence of any autocorrelation. In *R* the Durbin-Watson test is performed using the *durbinWatsonTest()* function available in the *car* package.

In order to quantify the performance of the model for different values of the parameters, the Root Mean Square Error (RMSE) between the estimated imputed values and the consumer's actual data must be measured. RMSE is defined as:

$$RMSE = \sqrt{\sum_i (y_i - f_i)^2} \qquad \text{Equation 4.7}$$

where y_i is the actual (missing) sequence and f_i is the fitted (imputed) sequence.

To calculate the average RMSE, the missing data had to be simulated by intentionally removing a portion of length *m* which would later be compared to the imputed portion. So, for each of the load profiles, the algorithm was iterated for all the 335 load profiles using various values of *m*, *p* and *k*, while *t* was kept constant at 1246 days – the whole dataset available.

Three types of averaging methods were considered, which are:

- No weighting – ordinary averaging.
- Time weighting - more recent portions are weighted more.
- Least Squares (LS) weighting - portions with lower LS values are weighted more.

The averaging was performed using:

$$[b_1 \quad b_2 \quad \cdots \quad b_m] = \frac{1}{\sum_{u=1}^{k} w_u} [w_1 \quad w_2 \quad \cdots \quad w_k] \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k1} & x_{k2} & \cdots & x_{km} \end{bmatrix} \qquad \text{Equation 4.8}$$

where:

$[b_1, b_2, \dots b_m]$ is a vector of the imputation values,

$[w_1, w_2, \dots w_k]$ is a vector of the weightings.

and x_{ij} are the observations selected using minimum least squares distance, with *i* identifying to the selected *related window*, placed in ascending order based on their least square distance value and *j* giving the relative position of the observation inside the *related window* (Figure 4.3).

The weighting vector $[w_1, w_2, \dots w_k]$ is calculated depending on the type of averaging performed. When the averaging is unweighted, then all elements of *w* are equal to 1. When the averaging is time weighted the values of *w* are calculated as $t - n$, *n* being the index of the particular *related window* selected, so that the bigger the value of *n* the smaller the weighting and vice versa. When the weighting is based on LS then the values of *w* are calculated as max(SS) – SS[*n*], *n* being the index of the particular selected *related window*, so that the bigger the value of the sum of squares the smaller the weighting and vice versa.

The algorithm was iterated using the three averaging methods for different values of *m* and *p*. Figure 4.5 compares the resulting average RMSE when using the three averaging methods with *m*=5 and *p*=10. As can be seen the *ORDINARY* averaging method does not perform much worse than the other two, actually it fared better for *k* less than nine. For *k* greater than eight the *TIME* averaging method performs better than the other two.

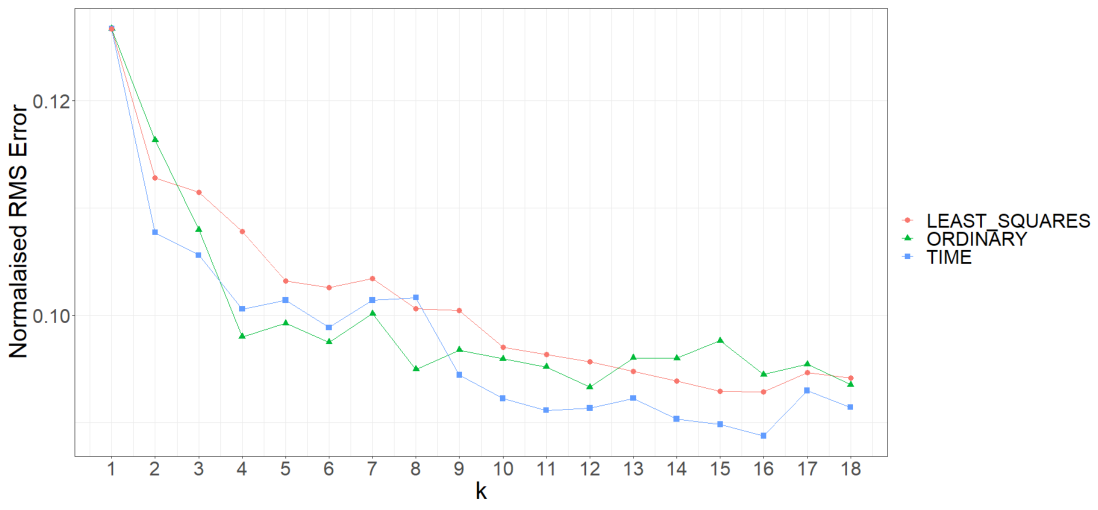

Figure 4.5. Comparison of averaging methods.

For each iteration the average error, defined as the average difference between the estimated output of the algorithm and the values of the actual load profile for all the 335 load profiles, were recorded. The normalised Root Mean Square of the Error (RMSE) was used as a metric of the goodness of fit for different values of the parameters. For normalisation the RMSE was divided by the maximum value of the missing part. To prevent division by zero conditions, when the maximum of the missing part is zero then the maximum of the imputed part is used. When the maximum of the imputed part is also zero, then the RMSE is set to zero, since it is then implied that both the missing and imputed parts are all zero and hence equal. Figure

4.6 shows how the average normalised RMSE varies with m, p and k. t was kept constant at 1246, to utilise all the available values in the dataset. Load profiles with a DW statistic less than 0.6 are included. These amount to 262 out of the total 335 consumers, or 78.2%. It can be seen how the RMSE increases as m increases and decreases as k increases.

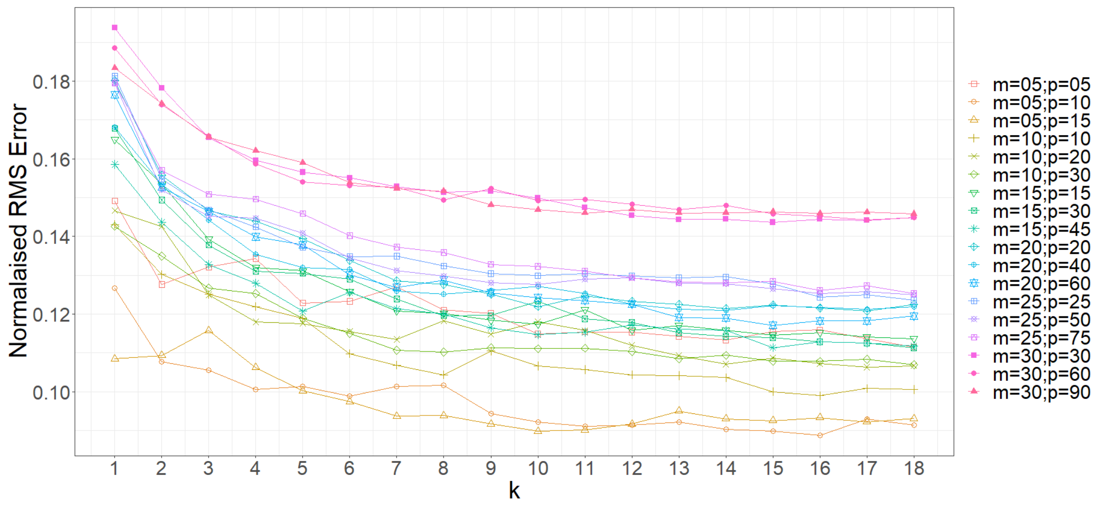

Figure 4.6. Average normalised RMS error for various m, p and k.

To distinguish better between the plots, Figure 4.7 shows only the average normalised RMSE for m = 5 and m = 30. For m = 5 it is clear that the RMSE is significantly improved between p = 5 and p = 10. Then the RMSE does not improve much for p = 15. When m = 30, the improvement on RMSE between p = 30, p = 60 and p = 90 is minimal. Same trend can be noted, from Figure 4.6, for the other values of m which were tested, i.e. 10, 15, 20 and 25. Although when m = 30 there is not much improvement in RMSE between p = 30 and p = 60, knowing from Section 2.1 that the average number of days with missing load profiles, i.e. m, is 4.51 days, and noticing the considerable improvement in RMSE when m = 5, indicates that choosing p as 2 × m is appropriate. Thus, it can be concluded that the value of p should be twice the value of m. Increasing the value of p further than that would have minimal effect on the value of the RMSE and will have a significant effect on the processing time needed for imputation.

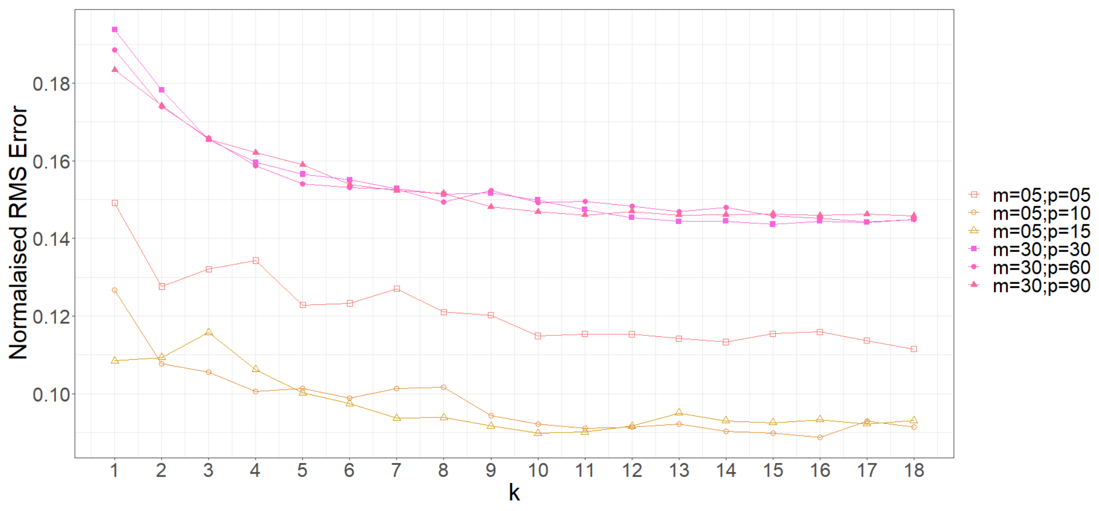

Figure 4.7. Average normalised RMSE for m = 5 and m = 30.

Another metric which was considered for quantifying the performance of the imputation model was the Coefficient of Determination (CoD) or R^2. Several definitions for the CoD are encountered in literature and not all of them are equivalent. One definition of CoD is [101]:

$$R^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y})^2} \qquad \text{Equation 4.9}$$

where y_i is the actual sequence and f_i is the fitted sequence.

In Equation 4.9 the numerator in the fractional part is the mean squared error, while the denominator is the variance multiplied by the number of samples minus one. For a perfect fit the numerator becomes zero and so R^2 is one, which is optimal. In the case of a model with a fit which is always equal to the mean of the actual value will result in an R^2 of zero. Models with worse fits will have a negative R^2. In the condition when the all the values in the sequence y are equal to a constant, the denominator of the fractional part becomes zero and so R^2 becomes undefined. This condition, of having a sequence with all values equal to a constant, is not uncommon in load profiles, especially when there is a period of zero consumption. This is one reason why RMSE was preferred as a metric for the goodness of fit over CoD.

Figure 4.8 shows the relation between the Durbin-Watson statistic and the average normalised RMSE obtained for m=20, p=40, k=10. There is a concentration of points in the bottom left area where both the RMSE and DW are close to zero. This means that the method manages to impute values with small errors when DW indicates a high level of positive autocorrelation. The RMSE starts getting larger as the value of DW starts approaching two, which indicates no autocorrelation. There are still some consumers with zero error when DW is close to two, these are mainly consumers with erratic behaviour where the missing portion was all zeroes and where the estimated values were recovered by the process of blending with the spot readings. There are also some points at the top left area where the DW shows good positive autocorrelation but the RMSE is high. These load profiles had a period of seasonal behaviour in the past but eventually became erratic.

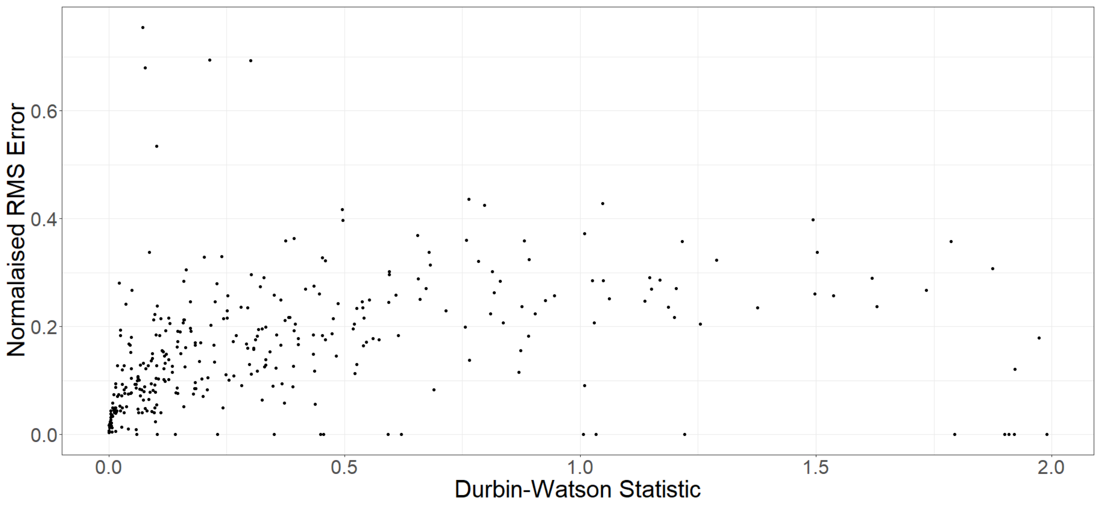

Figure 4.8. The relation between RMSE and Durbin-Watson statistic.

An analysis was conducted to establish the processing time consumed for the various values of *m*, *p*, and *k*. The value of *k* does not have significant effect on the processing time since all the samples *t* are searched, irrespective of the value of *k*, and then the nearest *k* are selected. Only the averaging process processing time is directly proportional to *k*, but that is relatively negligible when compared to the searching process. In Figure 4.9 the red horizontal line shows the average processing time aggregated for *k* and it is clear that it is constant and independent of *k*. The green line shows the average processing time aggregated for *m* and indicates that the average processing time is linearly dependent on the value of *m*. Similarly, the blue line shows the average processing time aggregated for *p*. The average processing times for both *m* and *p* have an intercept of 12.3 ms which is the overhead processing time which is independent of the values of *m* and *p*. From this analysis, and from what was already discussed before, it is clear that choosing the value of p to be twice m would give a good compromise between time performance and RMSE. The value of *k*, having no impact on the processing time, should be chosen purely on the value of RMSE. Observing Figure 4.6 and Figure 4.7 the value of the RMSE starts converging after the value of *k* = 9. So, it was decided that a value of *k* = 10 would be a reasonable value.

Note that these processing times were taken when running the imputation method on a CORE *i*7-4610M 3.00 GHz machine with 16.0 GB RAM.

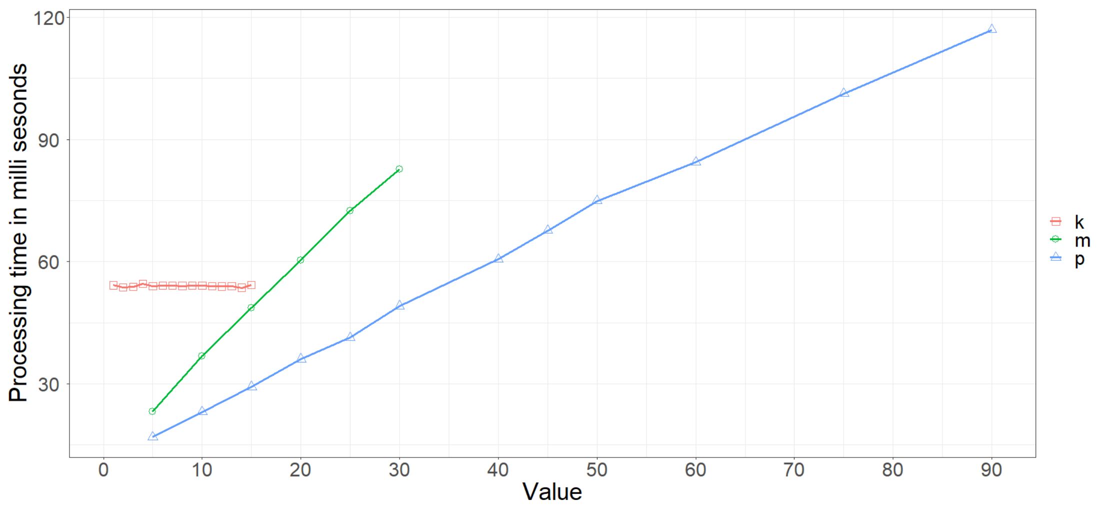

Figure 4.9. Processing time for values of *m*, *p*, and *k*.

Based on the above it was decided to set the parameters $p = 2 \times m$ *and* $k = 10$ and so a run was conducted on the daily load profiles of all the 335 consumers, varying *m* from 1 to 30, while keeping *t* constant at 1246. The obtained average normalised RMSE is shown in Figure 4.10 by the black dots. As expected, the RMSE is very small for *m* = 1, but it rises quite rapidly till *m* = 4 and then continues to rise with a gentler slope. The fitted non-linear regression line was obtained by using the *R* function *nls()*, (Nearest Least Squares) from package *stats*. The formula used as the argument for *nls()* was :

$$y \sim a * x/(b + x) - 0.8 \qquad \text{Equation 4.10}$$

Missing load profiles values can be considered to be MCAR since the occurrence of the missing data is completely unrelated to the consumption or consumption patterns but to other causes as mentioned in Chapters 1 and 2. So, the sample of 335 load profiles can be considered to be a random sample of the whole population of commercial consumers. As was mentioned in Section 2.1, the average number of days with missing load profiles is 4.51 days. The *R* function *predict()*, also from package *stats*, was used to get the average normalised RMSE value for *m* = 4.51, based on the output of *nls()*. The value for the average normalised RMSE obtained was 0.0747 or 7.47% of the actual values.

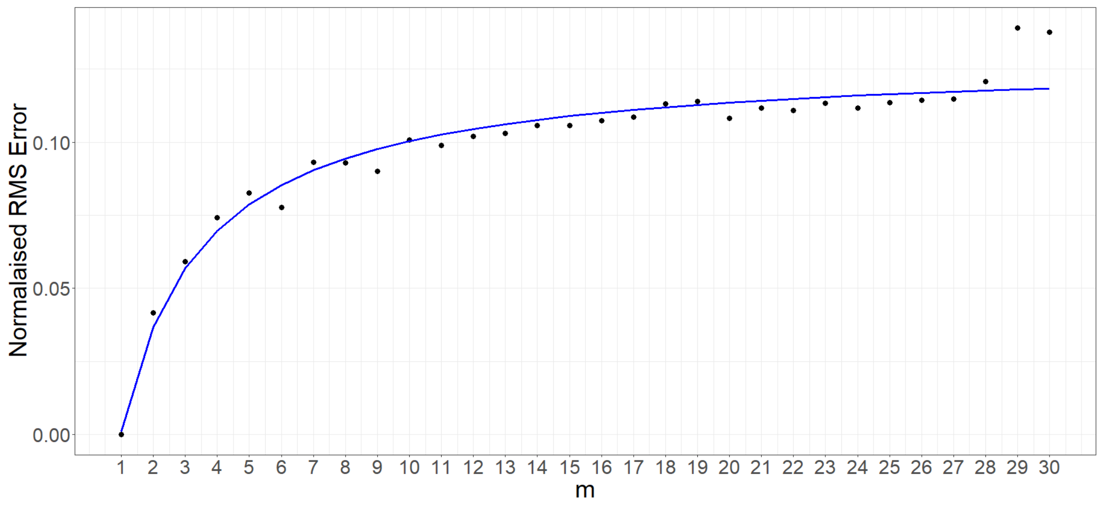

Figure 4.10. Mean of the normalised RMSE (p=2×m and k=10) for daily load profiles with non-linear regression line.

4.3 Application of the Algorithm

Two sample consumer load profiles, with a simulated missing portion of 20 days, are considered to illustrate the implementation of the algorithm and the results. The consumer load profiles in the two cases are presented mainly as a visual representation of what the imputed values look like. The load profile in case 1 has a Durbin-Watson statistic of 0.1996 and its normalised RMSE for daily load profiles was 0.0769 and that for half hourly load profiles was 0.0946. For the load profile in case 2 the Durbin-Watson statistic was 0.0810 while the normalised RMSE for the imputed daily load profile was 0.0610 and that for the half hourly load profile was 0.0644. So, for these two cases the fact that the Durbin-Watson statistic was lower for case 2, indicating a more positive autocorrelation, resulted in a better RMSE. As explained for Figure 4.8 the relationship between RMSE and autocorrelation is predominant but is not definite because there are instances of low or erratic behaviour when this relation does not hold.

4.3.1 Case 1

Figure 4.11 shows the load profile of a consumer which is used as an example for demonstrating the imputation technique adopted in this work. As already mentioned for Figure 4.1 the horizontal axis for Figure 4.11 shows the number of days in the past with day 1 being May 31st 2018. The profile shows the seasonality very clearly and does not exhibit any noticeable trend. The consumption is seen to increase in summer and decrease in winter. The value of the Durbin-Watson statistic for this load profile is 0.1996 which indicates that is has a good amount of positive autocorrelation and so the missing values are expected to be imputed with low RMSE.

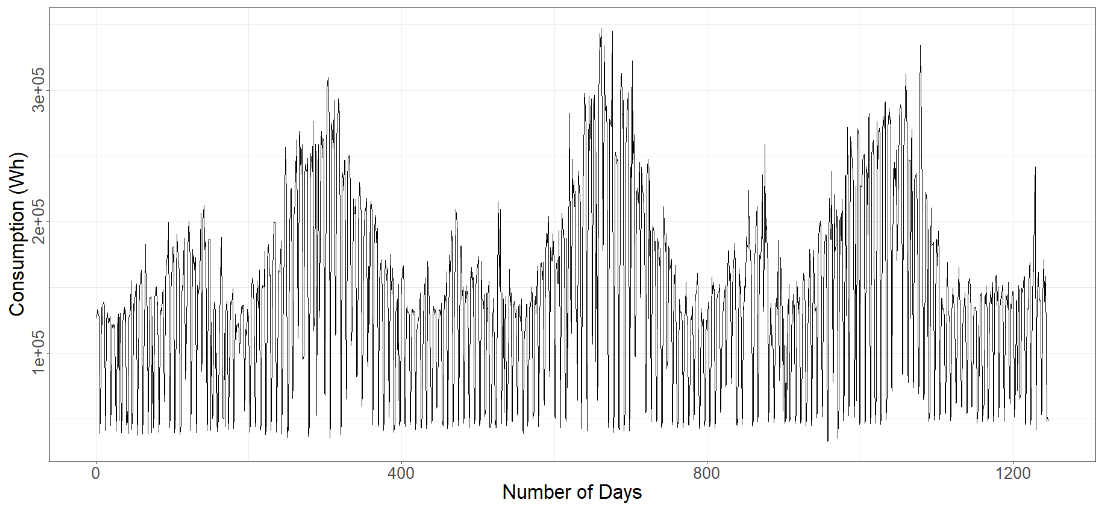

Figure 4.11. Daily load profile for consumer.

Figure 4.12 shows the same load profile of Figure 4.11 with a removed portion, shown in red with m = 20, and the corresponding best matching patterns, shown in green, with parameter p = 40. The number of green sections is not necessarily equal to k, as might be expected, because some of the *resembling windows* overlap each other. The best 10 best matching windows are found starting at n = {890, 1177, 1121, 57, 771, 533, 407, 414, 1163, 764}.

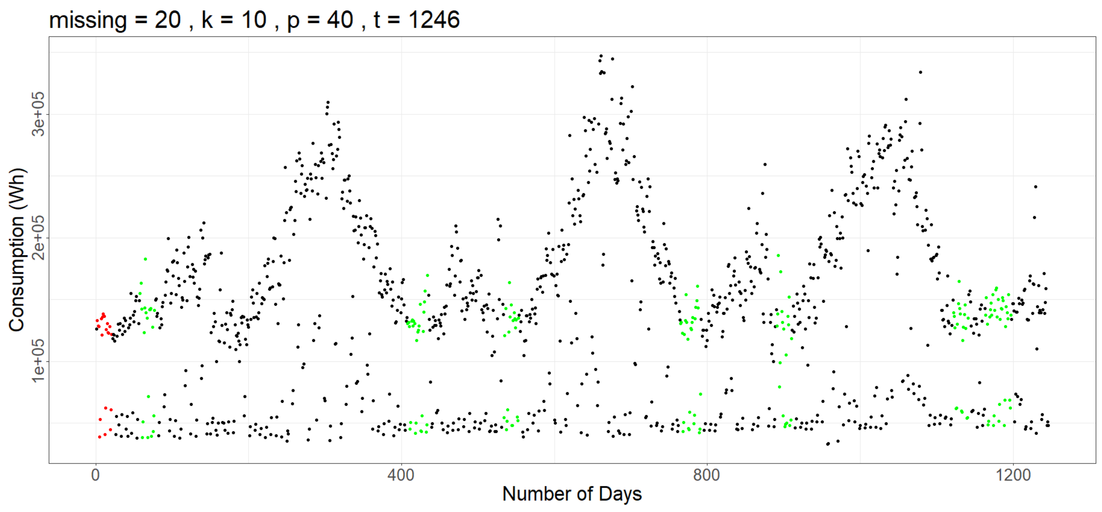

Figure 4.12. *Removed portion* in red and best match windows in green.

Figure 4.13(a) shows the *present surrounding window* with the simulated *missing portion* in red. Figure 4.13(b) through (l) shows the *past surrounding windows* which best resemble the present surrounding window, sorted ascending order by their sum of squares. The value of n and the sum squares is shown on top of each figure.

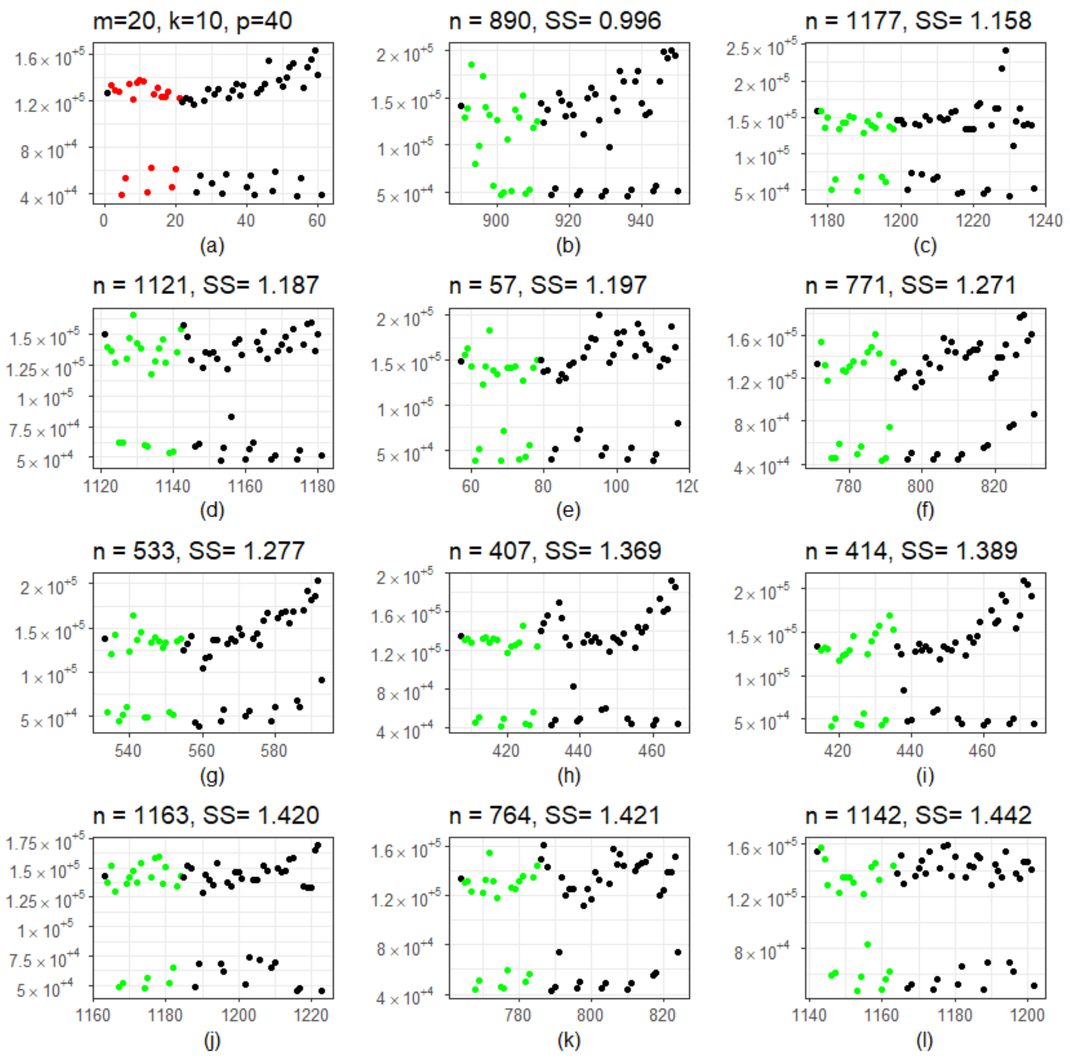

Figure 4.13. *Present surrounding window* (a) and best matched *past surrounding windows* (b – l).

Figure 4.14 shows the imputed portion for the daily load profile, in green, superimposed on the simulated missing portion. It is evident from the plot that the consumption during the weekends is reduced and the imputed values follow the pattern closely. The RMSE of the imputed values is 0.0768, or 7.68% of the actual values.

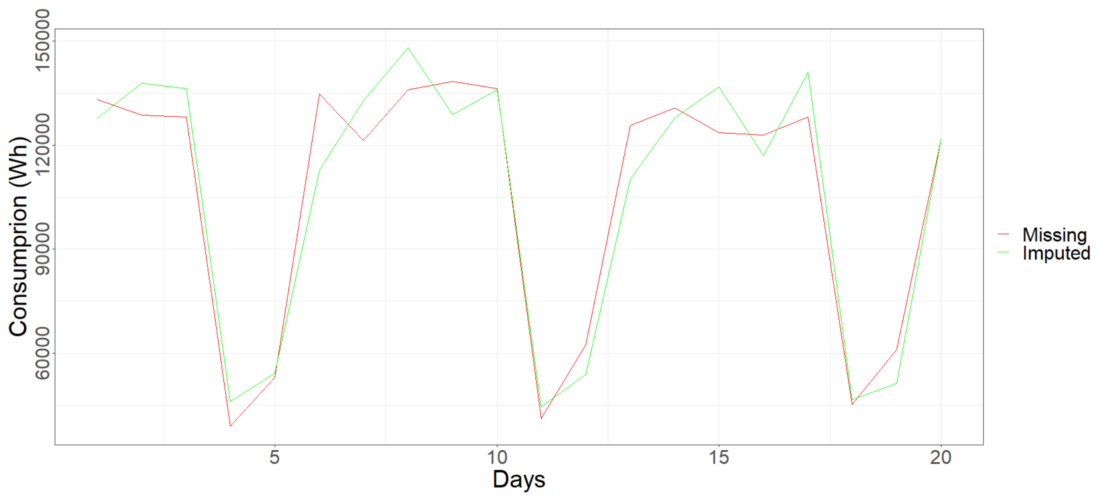

Figure 4.14. Imputed daily portion superimposed on missing portion.

Figure 4.15 shows the imputed half hourly values which were obtained for same the days of Figure 4.14, superimposed on the actual values. As was noted for Figure 4.14 this consumer has low consumption on weekends and the imputed half hour values also follow the same pattern. For most of the days the peaks of the estimated values are close to those of the actual values and the half hourly profiles are similar with some differing slightly. The higher errors are seen to occur for Saturdays. Here the RMSE for the imputed half hourly load profiles is 0.0946 or 9.46% of the actual values.

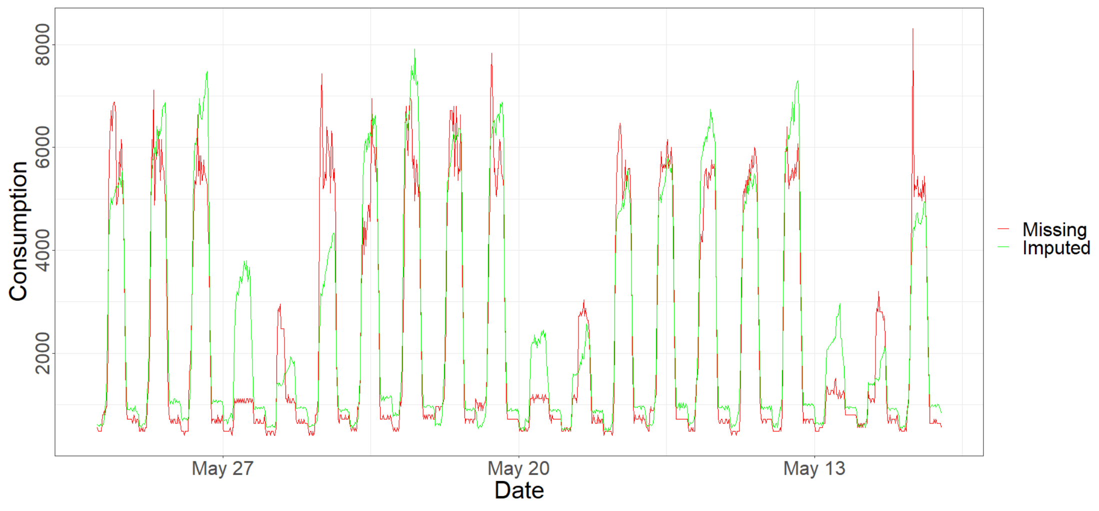

Figure 4.15. Imputed half-hourly load profile superimposed on the values.

4.3.2 Case 2

Figure 4.16 shows the load profile for another consumer. Again, the yearly seasonality is very pronounced and with a consumption pattern that is stationary over the three years. The value of the Durbin-Watson statistic for this load profile is 0.0810 which indicates more positive autocorrelation than the consumer of case 1 and so is expected to have a lower RMSE.

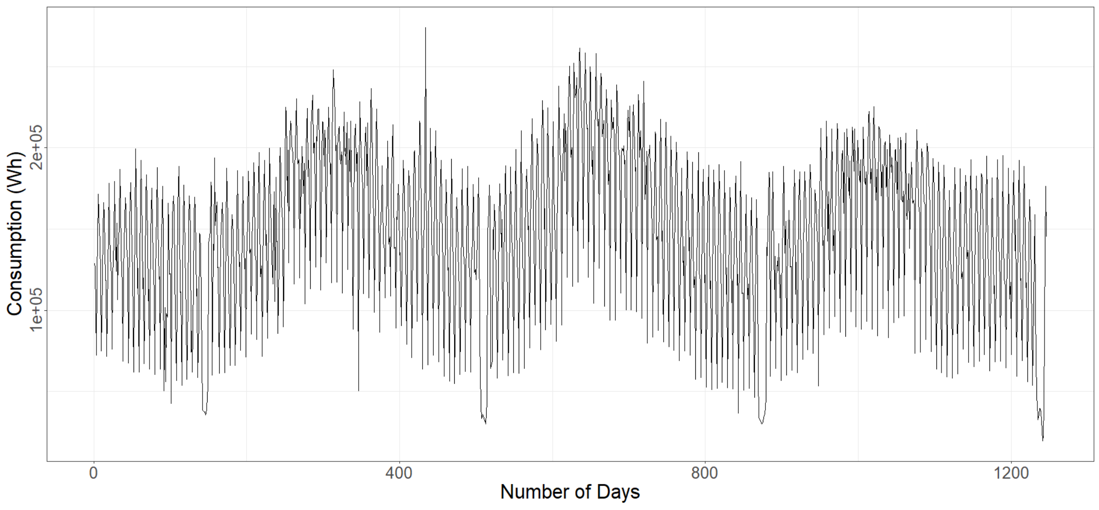

Figure 4.16. Daily load profile for a consumer over 1246 days.

Figure 4.17 shows the same load profile of Figure 4.16 with a removed portion, shown in red with *m* = 20, and the corresponding best matching patterns, shown in green, with some overlapping each other. It can be noticed, by observing the plot, that the resembling portions are found during the same time of the year. These 10 best matching *resembling windows* where found starting at n = {435, 1114, 1107, 162, 771, 1142, 1170, 743, 29, and 799}.

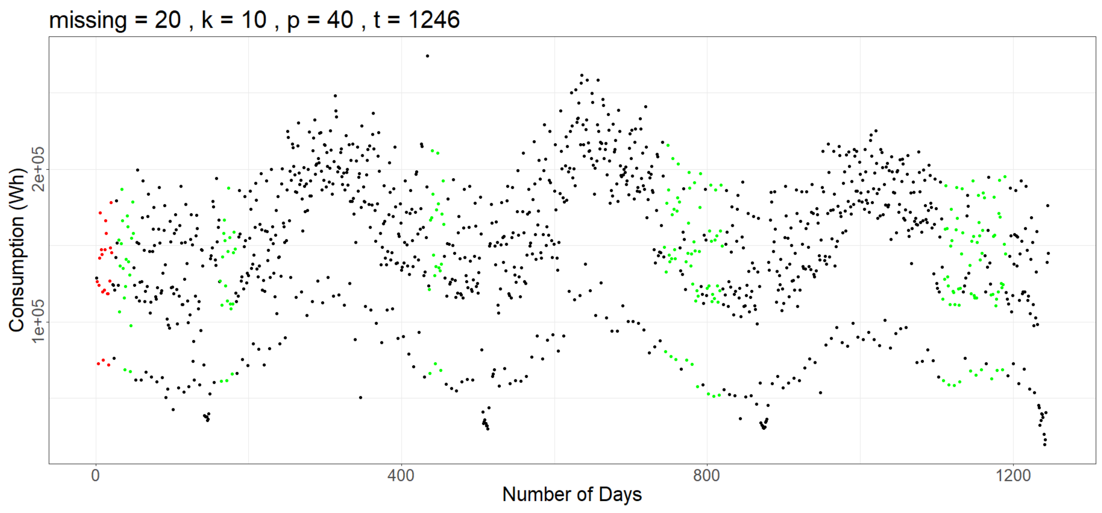

Figure 4.17. Load profile with simulated missing portion in red and best fits in green.

Figure 4.18 (a) shows the *present surrounding window* with the simulated missing portion in red. Figure 4.18 (b) through (l) show the *past surrounding windows* which best resemble the *present surrounding window*, ordered by their sum of squares in ascending order.

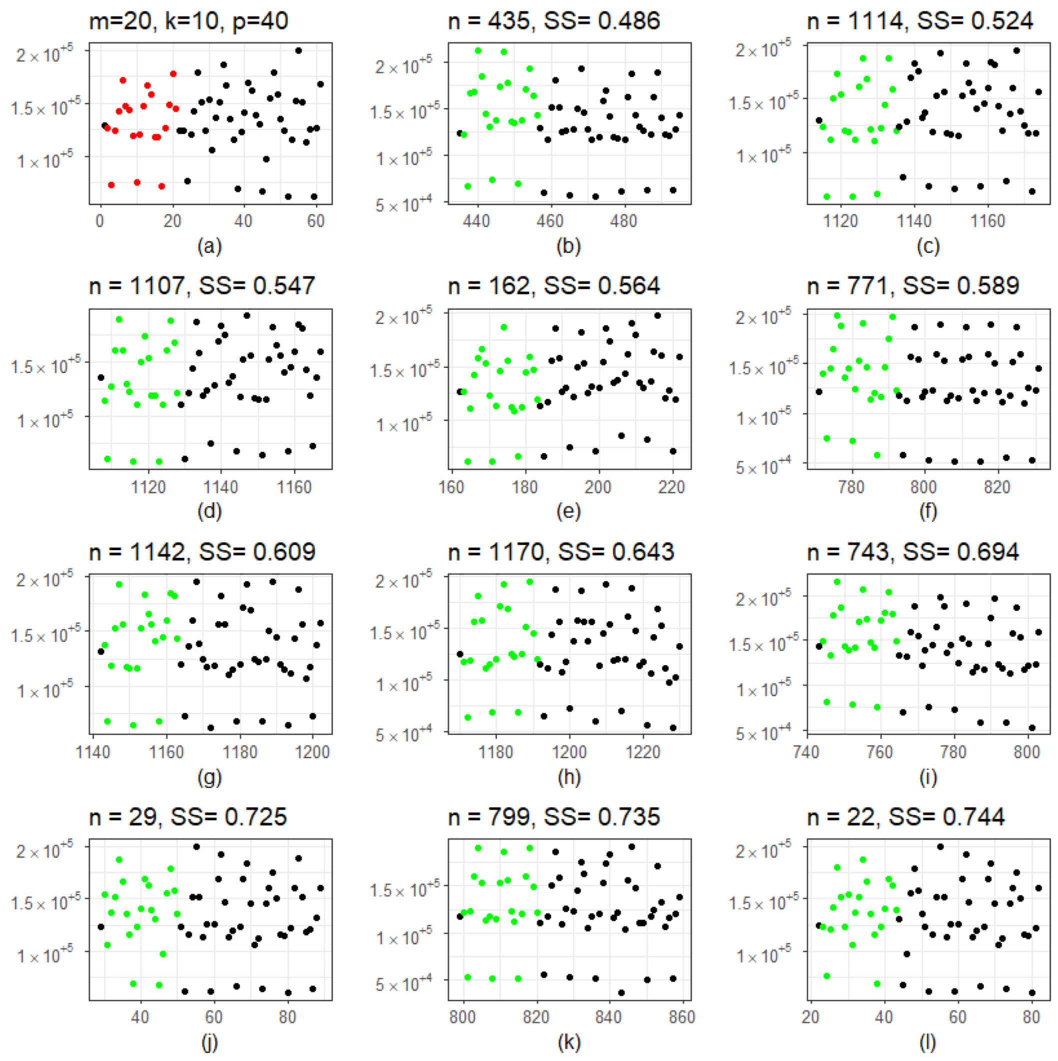

Figure 4.18. The simulated missing part in (a) and the best matches found (b – l).

Figure 4.19 shows the estimated daily load profiles for the removed portion after averaging and blending with the spot readings. As can be seen the calculated daily consumption values are close to the actual consumption for this consumer, with a normalised RMSE of 0.0610 or 6.10% of the actual values.

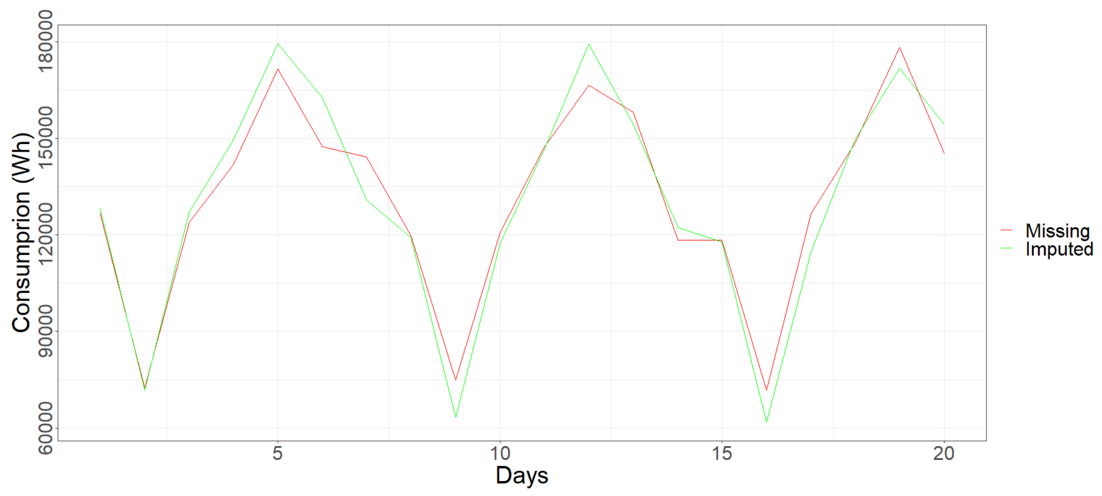

Figure 4.19. Imputed daily load profile.

Figure 4.20 shows the estimated half hour load profile obtained from the same days used for the estimated daily load profile. As can be noticed this consumer has minimal consumption on Sundays and the imputed values do follow the same pattern of behaviour. The spikes are not followed because the averaging tends to smooth them out. The imputed half hourly load profile values have an RMSE of 0.0644, or 6.44% of the actual values.

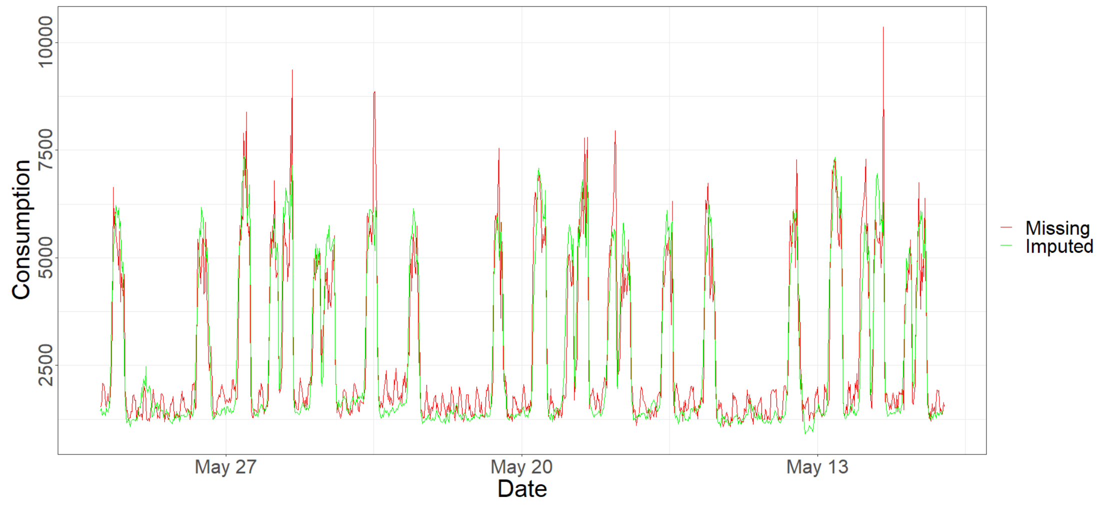

Figure 4.20. Imputed half-hourly load profile.

4.4 Minor Modifications and Future Enhancements

The imputation method developed uses the historical load profile data of the individual consumer. A suggested enhancement would be to utilise the present consumption of other consumers with similar past behaviour and give it some weighting when calculating the values to impute. This will help to better estimate the consumption when irregular events happen, like a public holiday or days with adverse weather.

In this work the amount of historical data used, *t*, was kept constant at 1246 so as to use all the available information. In future work it would be useful to experiment to find smaller values of *t* which would reduce the processing time without compromising the RMSE.

4.5 Conclusion

Having a complete dataset, without any missing observations, is essential for any statistical or analytical process. In Enemalta's smart meter set-up, occasionally, some load profiles are not successfully read for a number of days, which gives rise to missing data. Imputing this data allows the organisation to be able to offer the consumer a complete load profile, with the imputed parts indicated, instead of a partial one. A complete load profile also enables the utility to perform a number of activities like calculating the maximum current in certain nodes and hence locating points of potential failure. It can also be used for load balancing which leads to a reduction in the technical losses.

In this work an imputation method was developed so that these, and other, objectives can be achieved. The developed imputation method is based on a KNN technique which searches in the past consumption history of the consumer for behaviours similar to that during the period right before the data went missing (nearest neighbors) and taking an average of the parts which correspond to the missing portion. The method makes use also of the available spot readings so that the relationship between spot readings and load profiles is conserved. The important parameters for this method are the number of nearest neighbours to consider *k*, the size of the surrounding windows, *p*, and the amount of historical data to look into the past, *t*. Simulated runs were performed on a sample of 335 load profiles for various missing days of data, *m*, to find suitable values for these parameters. Based on the average RMSE achieved and processing time consumed, the values for the parameters were chosen to be k = 10, p = 2 × m and t = 1246. For the sample of 335 load profiles, the average RMSE of the imputed values was calculated to be 7.47% of the actual values. The completed dataset is also a building block for the achievement of the second objective in this work, anomaly detection.

5 Anomaly Detection Method

Losses are always detrimental for any utility company. Inevitably losses raise the price that consumers pay for the service. Losses are classified as Technical Losses and Non-Technical Losses (NTL). Technical losses are inherent to the delivery of electrical energy over the distribution network. There are a number of measures that electricity companies use to reduce technical losses such as, for instance, increasing the voltage to reduce power losses in transmission. Technical losses, however, will always be present in cables, overhead lines and transformers. Distributed Generating Sources (DGS), like photovoltaic generators, wind turbines and Combined Heat and Power (CHP) systems, all of which are increasingly gaining popularity, help to lower the technical losses since the lengths of the transmission lines (or cables) carrying the energy from source to sink, are reduced. Although not straightforward, technical losses can be estimated [102].

Non-technical loses (NTL) may also be considered as financial losses. They can be made up of billing inaccuracies, faulty (or stopped) metering setups, and fraudulent consumers. NTL are a concern for utilities since they directly impact the revenue. Fraudulent consumers tend to consume more than they would if they were being charged properly for their consumption. This work develops a method which identifies consumers which are most likely to have faulty or tampered metering setups.

This Chapter first outlines the approach adopted in this work for anomaly detection. Then, in the following sections, each step is explained in dept, leading to the methods used to obtain the two coefficients which quantify the probability of a consumer contributing to NTL.

5.1 Approach

Most of the approaches found in literature, which deals with NTL detection, uses supervised machine learning methods, mainly SVM and ANN. In supervised classification the machine is trained using a set of labelled values. In the case of supervised NTL detection, the training values consist of a number of inspection outcomes which have to be available before training the machine learning classifier. Normally it is expected that the number of inspections with a genuine outcome excessively outnumber those with a fraudulent one. This creates a huge imbalance in the training set which has to be dealt with [67, 68]. Moreover Enemalta plc. does not have a vast record of inspections and outcomes for genuine and fraudulent cases. It was therefore determined that an unsupervised method should be used.

The approach adopted in this work is summarized in Figure 5.1. The load profiles are collected from consumers and any missing data is imputed as already discussed in Chapter 4. The first step of dimension reduction is performed by feature extraction (Section 5.2). Feature extraction reduces the dimensionality of each data points while emphasis is given to the salient characteristics to be considered. Outliers are then detected and removed (Section 5.3) since they are prone to bias any statistical analysis to follow. Principal component analysis then follows serving as a second step of dimension reduction. The data-points are then grouped into different clusters (Section 5.5) according to their consumption behaviour which allows for the calculations of the anomaly coefficient (Section 5.9.1) and the cluster-change coefficient (Section 5.9.2). Based on these two coefficients an anomaly score is obtained to draw up a sorted list of consumers which are more likely to exhibit non-technical losses.

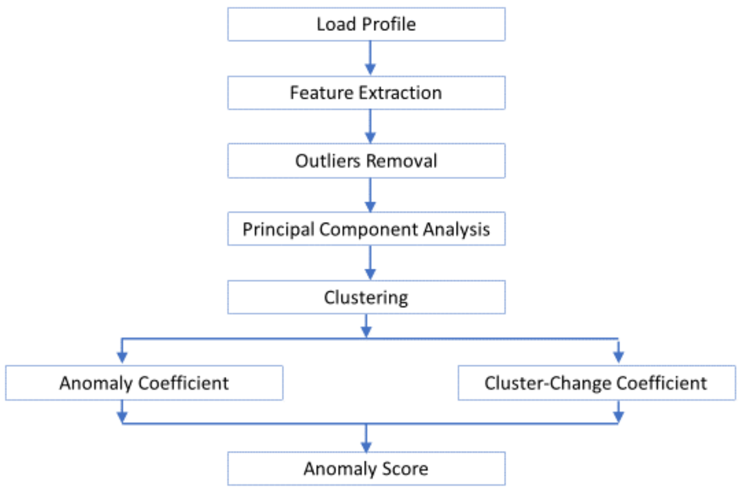

Figure 5.1. Process flow for anomaly detection.

5.2 Feature Extraction

The data used for this analysis is the consumption load profile. The load profile is the consumption by individual consumers measured at half hour intervals. Hence, there are 48 observation for every consumer every day or 336 observations for every consumer every week. Each consumer dataset for a week is therefore fairly large and thus some form of dimensional reduction has to be applied before any further processing can take place.

Feature extraction is a dimensional reduction method by which an initial set of raw variables is reduced to more manageable groups of features. The resulting features should be selected in such a way that they express the characteristics in the original data which need to be analysed and at the same time reduce redundancy. Feature extraction reduces the risk of overfitting, where the model would be too well designed for the training dataset but is not able to generalise for successive datasets.

Beckel et al. [46] established a list of 22 features for electricity consumption load profiles, which are grouped into four categories: *consumption figures*, *ratios*, *temporal properties* and *statistical properties*. Kopf el al. [47] continue to build on the features presented in [46] to establish 88 features grouped in the same categories. In this work, a set of 32 features, based on those used in [46] and [47], were derived. For some of these features, the day is divided into 5 segments as shown in Table 5.1.

Table 5.1. Time of day segments.

Time	**Segment**	**Abbreviation**
6am – 11am	Morning	M
11am- 2pm	Noon	N
2pm-7pm	Afternoon	A
7pm-11pm	Evening	E
11pm-6am	Night	G

Before starting the computation of the features, the load profile for all consumers were offset by 100Wh. This was done to avoid divisions by zero for some of the features adopted. It is not uncommon that consumers have periods of zero consumption, and in such cases, when computing ratio features, a divide by zero may occur. The 32 features established are shown in Table 5.2.

Throughout this work the analysis is done on a weekly basis, that is comparing the weekly behaviour of the consumers. So, the features were designed in a way so as to extract as much information about the weekly behaviour as possible. The days were grouped in two, weekdays and weekend days. The *Xcorrelation sum* is achieved by finding the cross-correlation for every two consecutive days and adding up each correlation value. This is a measure of the consistency of daily behaviour throughout the week. Other temporal features are used, such as *first time above mean* which is the time of day at which the consumers consumption exceeds the mean of that day for the first time. The feature *morning sum ratio* is the ratio of the consumption during the morning segment to the consumption of the whole day.

Here it is worth noting that no averages were selected as features since when, as explained later in Section 5.4, the features are normalised, the sums and averages yield the same values.

Table 5.2. List of features used.

Feature name	**Description**
WDSum	Week day sum
WDMax	Week day max
WESum	Week end sum
WEMax	Week end max
WDXcorrSum	Week day Xcorrelation sum
WEXcorrSum	Week end Xcorrelation sum
WDsDSum	Week day standard deviation sum
WEsDSum	Week end standard deviation sum
WDfTAQPSum	Week day first time above one fourth peak sum
WEfTAQPSum	Week end first time above one fourth peak sum
WDfTAHPSum	Week day first time above half peak sum
WEfTAHPSum	Week end first time above half peak sum
WDfTA3QPSum	Week day first time above three fourths peak sum
WEfTA3QPSum	Week end first time above three fourths peak sum
WDfTRMaxSum	Week day first time reach max sum
WEfTRMaxSum	Week end first time reach max sum
WDfTAMeanSum	Week day first time above mean sum
WEfTAMeanSum	Week end first time above mean sum
WDnPeaksSum	Week day number of Peaks sum
WEnPeaksSum	Week end number of Peaks sum
WDWESumRatio	Week day end sum ratio
WDWEMaxRatio	Week day end max ratio
WDMSumRatio	Week day morning sum ratio
WDNSumRatio	Week day noon sum ratio
WDASumRatio	Week day afternoon sum ratio
WDESumRatio	Week day evening sum ratio
WDGSumRatio	Week day night sum ratio
WEMSumRatio	Week end morning sum ratio
WENSumRatio	Week end noon sum ratio
WEASumRatio	Week end afternoon sum ratio
WEESumRatio	Week end evening sum ratio
WEGSumRatio	Week end night sum ratio

5.3 Removing the Outliers

An outlier is described as an observation which is dissimilar from the other observations present in the data set [103]. Outliers must be individualised and segregated before the clustering takes place because the outliers can affect the clustering process by abnormally shifting the cluster centres. [104, 105] list a number of outlier detection methods which are available and review their application.

In this work the detection of the outliers was done based on the *z*-score [104, 105]. The *z*-score is the number of standard deviations that an observation is above or below the mean of the distribution. For every consumer the *z*-score was calculated for all features. Any consumer, having a *z*-score with absolute value of more than 5, for any feature, is considered as an outlier. [104, 105] identify as outliers those observations which have a *z*-score of more than 3, however in this work it was seen as more appropriate to use a *z*-score of 5 to define a consumer as an outlier. This is because the consumers which are segregated as outliers at this stage, are themselves already anomalies because they have a behaviour which is very different from the normal, and so each one of the outliers shall already be a candidate for fraud analysis.

Another method for labelling outliers is the Interquartile Range (IQR) method [105]. The interquartile range is the range between the first and the third quartiles. The first quartile is defined as the middle number between the smallest value and the median of the data set. The third quartile is the middle value between the median and the highest value of the data set. For the IQR method, any data point that falls outside of either 1.5 times the IQR below the first quartile or 1.5 times the IQR above the third quartile is considered as outlier. For this work the IQR method was tested but it resulted in reducing the dataset quite heavily, from 1174 to 171 data consumers, while by the *z*-score method the dataset size was reduced to 1041. Note that, preference was also given to the *z*-score method on account of its faster computation.

5.4 Principal Component Analysis

Principal Component Analysis (PCA) is a technique used in modern data analysis to reduce the dimensionality of complex datasets, increasing interpretability and at the same time minimising any loss of information [106]. PCA transforms the input set of variables into a new set of uncorrelated variables that successively maximise variance.

The data processed through PCA should have a similar order of magnitude, otherwise variables of lower magnitudes lose significance in comparison to the larger values. Therefore, it is advisable to scale the variables (the features in this case) being used. Struc and Pavesic [107] list several ways in which this scaling may be achieved. One popular method is to subtract the mean and divide by the standard deviation to get the standard score:

$$x_{scaled} = \frac{x_i - \mu}{\sigma} \qquad \text{Equation 5.1}$$

where x_i is the observation that is being scaled, μ is the mean and σ is the standard deviation of the unscaled variable. The resulting dataset will have a mean of zero and a standard deviation of one.

In this work the scaling is done by the normalisation:

$$x_{normalised} = \frac{x - x_{min}}{x_{max} - x_{min}} \qquad \text{Equation 5.2}$$

thus $x_{normalised}$ obtains a value which ranges from zero to one and is unit-less.

The *prcomp()*function in *R* was used to find the Principal Components (PCs).

The [*M* x *N*] vector containing the normalised features is passed to *prcomp()* which produces a *prcomp* object consisting of :

Sdev [*N*] :	the standard deviation of the principal components.
Rotation[*N* x *N*] :	the loadings, or eigenvectors, of the variables (features).
X[*M* x *N*]:	values of the rotated data.

where *N* is the number of features and *M* is the number of consumers.

Figure 5.2 shows the biplot of the first two principal components, PC1 on the horizontal axis and PC2 on the vertical axis. The arrow lines, depicting the loadings of the features, are superimposed on the scatter plot of the consumers. A longer arrow for the feature indicates that the feature's influence on the first two principal components is stronger than other features with shorter arrows. The relative direction of the arrows relates to the correlation between the features. Features that point in the same direction are positively correlated (for PC1 and PC2), while others that point in opposite directions are negatively correlated. Features which point at right angles to each other are not correlated to each other in relation to their first two principal components. This plot therefore relates significant information about the features. From this plot it may be decided that some features can be dropped, either because their loading is very small, and so they are irrelevant, or because they point exactly in the same direction as another feature and so they are redundant. This is a method by which PCA might further aid in dimension reduction and thus computational speed-up.

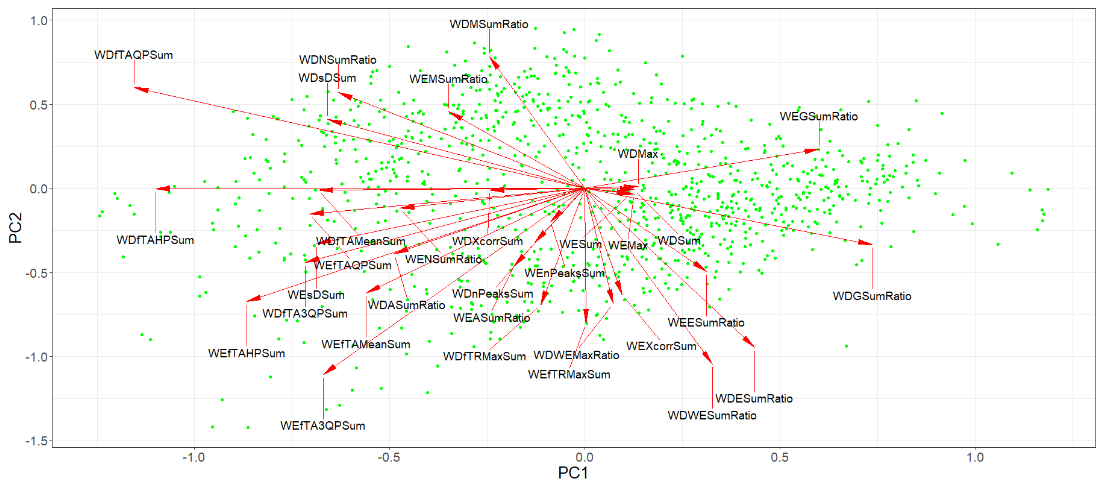

Figure 5.2. Biplot - features loadings superimposed on consumer points for PC1 and PC2.

Although PCA will return as many principal components as there are variables (32 in this case), further dimension reduction is achieved by determining how many principal components should be considered further. This decision is based on the percent variance of each principal component. Table 5.3 shows the variance, percentage variance and the cumulative percent variance for the first 14 principal components. As can be seen PC1 and PC2 contribute to 51% of the total variance, while the first 14 PCs contribute to 94.2% of the total variance

Table 5.3. Variance for the first 14 principal components.

PC	Variance	Percent Variance	Cumulative Percent Variance
1	0.214	28.5	28.5
2	0.169	22.5	51.0
3	0.084	11.2	62.2
4	0.053	7.1	69.3
5	0.039	5.2	74.5
6	0.033	4.4	78.9
7	0.024	3.2	82.1
8	0.022	2.9	84.9
9	0.018	2.5	87.4
10	0.014	1.9	89.3
11	0.010	1.4	90.7
12	0.009	1.2	91.9
13	0.009	1.2	93.1
14	0.008	1.1	94.2

The percentage variance of each PC is also plotted in Figure 5.3. Inspecting the plot of Figure 5.3, a marked drop is noted in the percent variance from PC2 to PC3. Principal components that contribute to relatively small variation may ignored. Here, principal components 1 and 2, as already seen from Table 5.3, contribute to 51% of the total variance, and subsequent principal components contribute much less. This would suggest that the focus should primarily be on principal components 1 and 2.

A second criteria that should be considered is to find which principal components contribute to more than one variable's worth of information. Had each of the 32 variables contributed equally, they would each contribute 3.125% (1/32) of the total variance, as indicated by the red line in Figure 5.3. From Table 5.3 it can be seen that the percentage variance of PC7 is 3.2% while that of PC8 is 2.9%. So, this criterion suggests that, since PC8 contributes to less than 3.2% of the total variance, it can be excluded and principal components PC3 to PC7 should be included.

So, principal components PC1 to PC7 were selected to be used for clustering since they account for most of the features in the original dataset (84.9% of the total variance). PC8 to PC32 carry much less information and were therefore removed from further analysis. Having a more concise dataset makes the clustering process simpler and less time consuming.

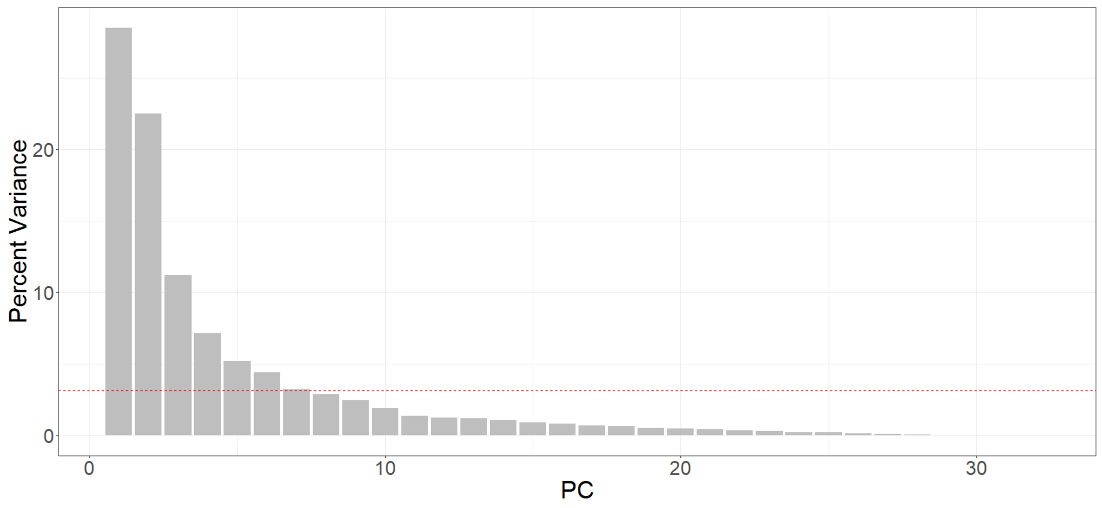

Figure 5.3. Plot for percentage variance of each principal component.

The numerical values of the PCA loadings of the first 7 components are listed in Table 5.4. The description of the feature names used is listed in Table 5.2. The loadings are the eigenvectors for each feature, which may be interpreted to express the amount of correlation between each feature and each principal component. In Figure 5.2, the loadings for PC1 and PC2 were plotted to show the correlation graphically in terms of the first two principal component directions, which account for most of the information.

Table 5.4. PCA loadings for all features.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
WDSum	0.0423	-0.0022	-0.1271	0.0556	-0.0173	0.0693	-0.3053
WDMax	0.0456	0.0034	-0.1556	0.0649	-0.0239	0.1126	-0.3705
WESum	0.0394	-0.0095	-0.1173	0.0523	-0.0195	0.0578	-0.2827
WEMax	0.0419	-0.0120	-0.1440	0.0728	-0.0262	0.0885	-0.3395
WDXcorrSum	-0.0813	-0.0033	-0.4041	-0.1679	-0.0688	0.3095	0.3578
WEXcorrSum	0.0318	-0.2113	-0.3071	0.0028	-0.0804	0.3254	0.3202
WDsDSum	-0.2197	0.1366	0.1366	-0.0111	0.2046	0.1148	0.1392
WEsDSum	-0.2291	-0.1097	0.1563	0.3376	0.2938	0.0952	0.2152
WDfTAQPSum	-0.3852	0.1997	0.0503	-0.0569	0.1887	-0.1892	0.1338
WEfTAQPSum	-0.2354	-0.0509	0.0120	0.2455	0.1357	-0.1289	0.1353
WDfTAHPSum	-0.3667	-0.0004	0.0810	-0.3010	0.1745	-0.0077	-0.1784
WEfTAHPSum	-0.2890	-0.2251	0.0675	0.1624	0.1421	0.1104	-0.0761
WDfTA3QPSum	-0.2391	-0.1473	0.1217	-0.3385	0.0234	0.0428	-0.2465
WEfTA3QPSum	-0.2237	-0.3691	0.0899	0.0652	-0.0116	0.2997	-0.1856
WDfTRMaxSum	-0.0376	-0.2316	0.0169	-0.2114	-0.0188	-0.1084	0.0579
WEfTRMaxSum	0.0009	-0.2680	-0.0482	-0.1952	-0.1246	0.0478	0.1565
WDfTAMeanSum	-0.2277	-0.0036	-0.2255	-0.2389	0.0237	-0.0300	-0.1363
WEfTAMeanSum	-0.1868	-0.2074	-0.2625	0.1587	-0.0239	0.1599	-0.1027
WDnPeaksSum	-0.0429	-0.1075	0.4512	-0.0004	-0.4018	0.1134	0.0292
WEnPeaksSum	-0.0293	-0.0665	0.4243	-0.0485	-0.4010	0.1647	0.0522
WDWESumRatio	0.1090	-0.3484	0.0103	0.1925	0.0020	-0.1157	-0.0083
WDWEMaxRatio	0.0240	-0.2287	-0.0317	0.2927	0.0166	-0.1169	0.0326
WDMSumRatio	-0.0816	0.2612	0.0264	0.0662	-0.0596	0.2991	-0.1030
WDNSumRatio	-0.2107	0.1902	-0.0988	-0.0577	-0.2542	0.0669	0.1180
WDASumRatio	-0.1631	-0.1295	-0.1365	0.0113	-0.1751	-0.5538	-0.0262
WDESumRatio	0.1449	-0.3156	0.0786	-0.1492	0.2442	-0.0404	-0.0005
WDGSumRatio	0.2458	-0.1118	0.0796	0.0741	0.1955	0.0172	0.0300
WEMSumRatio	-0.1163	0.1521	0.0233	0.2902	-0.0464	0.0930	-0.1126
WENSumRatio	-0.1583	-0.0395	-0.1112	0.1372	-0.2484	-0.1308	0.0800
WEASumRatio	-0.0608	-0.1547	-0.1211	-0.1048	-0.2365	-0.2084	0.0018
WEESumRatio	0.1039	-0.1647	0.0622	-0.2913	0.2248	0.0870	0.0412
WEGSumRatio	0.2002	0.0781	0.0879	-0.1363	0.2168	0.0660	0.0224

The principal components can be interpreted based on which features they are most correlated with, in either positive or negative direction. Table 5.5 assembles the 6 most correlated features for each of the first 7 principal components so that they can be interpreted accordingly.

Observing Table 5.5 it can be seen that PC1 is negatively correlated to *WDfTAQPSum* (week-day first time above quarter-peak), *WDfTAHPSum*, *WEfTAHPSum*, (week-day and weekend-day first time above half-peak), *WDfTA3QPSum* (week-day first time above three-fourths peak) and *WEfTAQPSum* (weekend-day first time above one-fourth peak). This correlation suggests these five features, which are all temporal variables based on the time at which the consumption reaches certain levels during the day, vary together and when one goes down, the others decrease as well. PC1 is positively correlated to *WDGSumRatio* (the ratio of night consumption to all-day consumption on week days), meaning that when the other five features decrease *WDGSumRatio* increases. This component is most predominantly correlated to *WDfTAQPSum* (week-day first time above quarter-peak) with a value of -0.3852 and then to *WDfTAHPSum* and *WEfTAHPSum* (week-day and weekend-day first time above half-peak) with values of -0.3667 and -0.2890 respectively.

PC2 is predominantly negatively correlated to *WEfTA3QPSum* (Weekend-day first time above three-fourths peak), *WDWESumRatio* (the ratio of weekend-days consumption to week-days consumption), *WDESumRatio* (the ratio of evening consumption to all-day consumption on week days), *WEfTRMaxSum*, *WDfTRMaxSum* (weekend-day and week-day first time reached max) while it is positively correlated to *WDMSumRatio* (the ratio of morning consumption to all-day consumption on week days). This correlation implies that when the former five features increase, the latter feature (*WDMSumRatio*) decreases and vice versa.

PC3 is mostly positively correlated to the features *WDnPeaksSum* and WEnPeaksSum (the week-day and weekend-day sum of number of peaks) while it is negatively correlated to *WDXcorrSum* and *WEXcorrSum* (the week-day and weekend-day cross-correlation sums). PC3 is also negatively correlated to *WEfTAMeanSum* and *WDfTAMeanSum* (the weekend-day and week-day sums of first time above mean). So PC3 is quite interesting since it groups the weekly features for number of peaks, cross-correlation and first time above mean.

Table 5.5. The 6 most correlated features for each of the first 7 PCs.

PC1	WDfTAQPSum	WDfTAHPSum	WEfTAHPSum	WDGSumRatio	WDfTA3QPSum	WEfTAQPSum
	-0.3852	-0.3667	-0.2890	0.2458	-0.2391	-0.2354
PC2	WEfTA3QPSum	WDWESumRatio	WDESumRatio	WEfTRMaxSum	WDMSumRatio	WDfTRMaxSum
	-0.3691	-0.3484	-0.3156	-0.2680	0.2612	-0.2316
PC3	WDnPeaksSum	WEnPeaksSum	WDXcorrSum	WEXcorrSum	WEfTAMeanSum	WDfTAMeanSum
	0.4512	0.4243	-0.4041	-0.3071	-0.2625	-0.2255
PC4	WDfTA3QPSum	WEsDSum	WDfTAHPSum	WDWEMaxRatio	WEESumRatio	WEMSumRatio
	-0.3385	0.3376	-0.3010	0.2927	-0.2913	0.2902
PC5	WDnPeaksSum	WEnPeaksSum	WEsDSum	WDNSumRatio	WENSumRatio	WDESumRatio
	-0.4018	-0.4010	0.2938	-0.2542	-0.2484	0.2442
PC6	WDASumRatio	WEXcorrSum	WDXcorrSum	WEfTA3QPSum	WDMSumRatio	WEASumRatio
	-0.5538	0.3254	0.3095	0.2997	0.2991	-0.2084
PC7	WDMax	WDXcorrSum	WEMax	WEXcorrSum	WDSum	WESum
	-0.3705	0.3578	-0.3395	0.3202	-0.3053	-0.2827

5.5 Hierarchical Clustering

Clustering is a technique which gathers data points into groups such that the members of one group are similar to the other members of the same group and different from members of other groups [108, 109]. The effectiveness of the clustering is superior when there is great similarity between members of the same group and great dissimilarity between members of different groups.

Clustering differs from classification in the sense that, for classification the data is labelled, and the classifier places the observations into classes based on the label of each observation – it is thus a supervised learning method. In clustering there is no foreign information about the data apart from the data itself – and is thus an unsupervised method.

There are several methods for clustering, with the most widely used being K-means and hierarchical clustering [110]. The K-means algorithm is popular because it is easy to implement, and its time complexity is linearly proportional to the number of observations in the dataset being clustered [110]. However, a major problem with K-means is that it is sensitive to the selection of the initial partitions and may converge to a local minimum of the criterion function value if the initial partition is not properly chosen [110], hence the computation has to be repeated for a number of times until the outcomes converge. Also, the K-means algorithm works well only on data sets having isotropic clusters [110]. Hierarchical clustering builds the clusters based on the data points themselves and does not depend on initial partition and works well with non-isotropic clusters. Based on the above arguments hierarchical clustering was used in this work. The time complexity for hierarchical clustering is proportional to the square of the number of observations.

The clustering pattern may be visualised on a dendrogram as shown in Figure 5.4. The horizontal axis of Figure 5.4 represents the distance or dissimilarity between clusters. The vertical axis represents the consumers and clusters. Each joining of two clusters is represented on the dendrogram by the splitting of a horizontal line into two horizontal lines. The horizontal position of the split, shown by the short vertical bar, gives the distance (dissimilarity) between the two clusters. Observing the lengths of the horizontal lines joining the clusters on the dendrogram gives an indication of where the *cut* can be made, thereby determining the number of clusters which are suitable for the dataset.

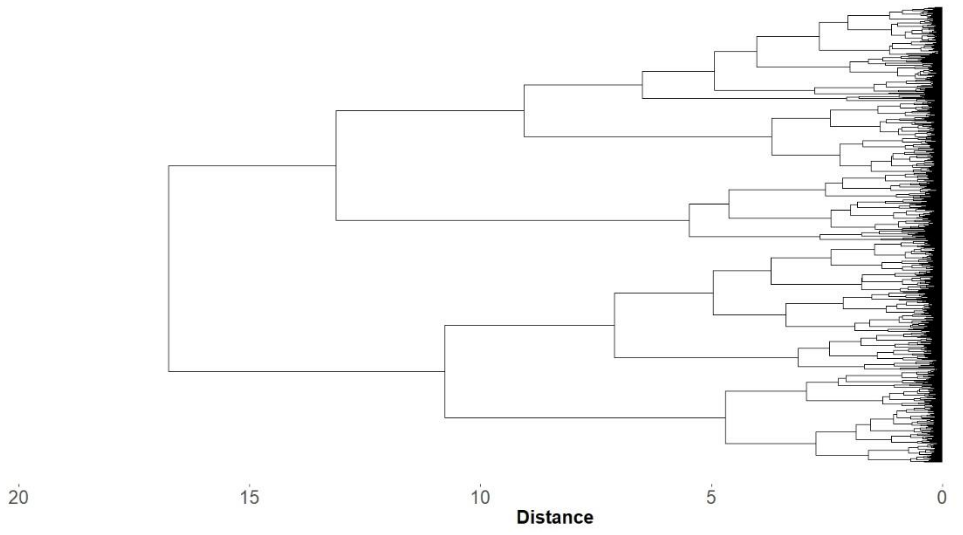

Figure 5.4. Dendrogram for the load profile dataset.

Determining the most suitable number of clusters by observing the dendrogram is a visual method and can be subjective. A classic method to determine the most appropriate number of clusters is by calculating the total Within-Cluster Sum of Squares (WCSS) [111] i.e. the total of the sums of all the squared distances between all observations within each cluster – TWCSS [111].

$$TWCSS_k = \sum_{r=1}^{k} \sum_{i=1}^{n_r-1} \sum_{j=i+1}^{n_r} \left|\left|x_i - x_j\right|\right|^2 \qquad \text{Equation 5.3}$$

where x is the data series, k is the number of clusters, n_r is the number of points in cluster r and D_r is the sum of the squares of distances between all points in cluster r.

The TWCSS gives a measure of the compactness of all the clusters since the closer the observations lie in each cluster the smaller the sum of squares. The number of clusters is iterated between 1 and a reasonable value, while each time the sum of WCSSs is calculated and plotted as in Figure 5.5. Figure 5.5 is referred to as the 'elbow plot' and the point where a noticeable kink is observed indicates an appropriate number of clusters. Package *factoextra* in *R* includes a function for generating the elbow plots, namely *fviz_nbclust()*. For hierarchical clustering *fviz_nbclust()* is called with parameters *FUN* = *hcut* and *method* = "*wss*". However, this method, being also a visual one, may sometimes be hard to interpret.

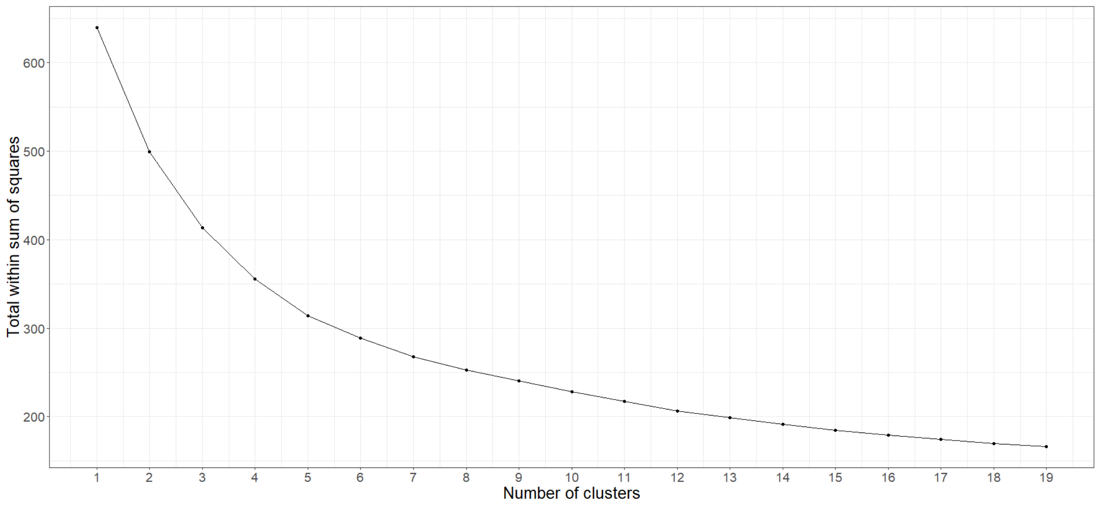

Figure 5.5. Plot of the Sum of Squares.

Another, more direct, technique used to determine the validity of a clustering scheme and the number of clusters is the method referred to as silhouette [49]. Silhouette is an algorithm which gives a value from -1 to 1 to all observations in a dataset based on their clustering. The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to the dissimilarity to other clusters (separation).

The silhouette score is calculated as follows

$$s_i = \frac{b_i - a_i}{max\{a_i, b_i\}} \text{ , } if\ n_r > 1 \qquad \text{Equation 5.4}$$

and

$$s_i = 0\ if\ n_r = 1 \qquad \text{Equation 5.5}$$

where s_i is the silhouette score for observation x_i, r is the cluster containing x_i, n_r is the number of observations in cluster *r* and

$$a_i = \frac{1}{n_r - 1} \sum_{j=1, j \neq i}^{n_r} dist(x_i, x_j) \qquad \text{Equation 5.6}$$

$$b_i = min \frac{1}{n_k} \sum_{j=1}^{n_k} dist(x_i, x_j) \text{ , } k \neq i \qquad \text{Equation 5.7}$$

where *dist*(x_i , x_j) is the distance between observations x_i and x_j.

To determine the most suitable number of clusters for this dataset the clustering was cut at different sections, obtaining different number of clusters ranging from 2 to 19. Each time the silhouette was calculated for each observation and then the mean for all observations recorded. In *R*, *fviz_nbclust()* can also be used to generated silhouette plots for an increasing number of clusters. In this case the parameters passed should be *FUNC* = *hcut* and *method* = "*silhouette*".

Figure 5.6 shows the plot of the silhouette coefficient against the number of clusters. For a small number of clusters (2 and 3) the silhouette coefficient is relatively high, then it starts falling and eventually rises until it reaches a maximum at 8 clusters. Finally, it continues to drop as the number of clusters increases. So, based on the silhouette score, it was decided to use 8 clusters.

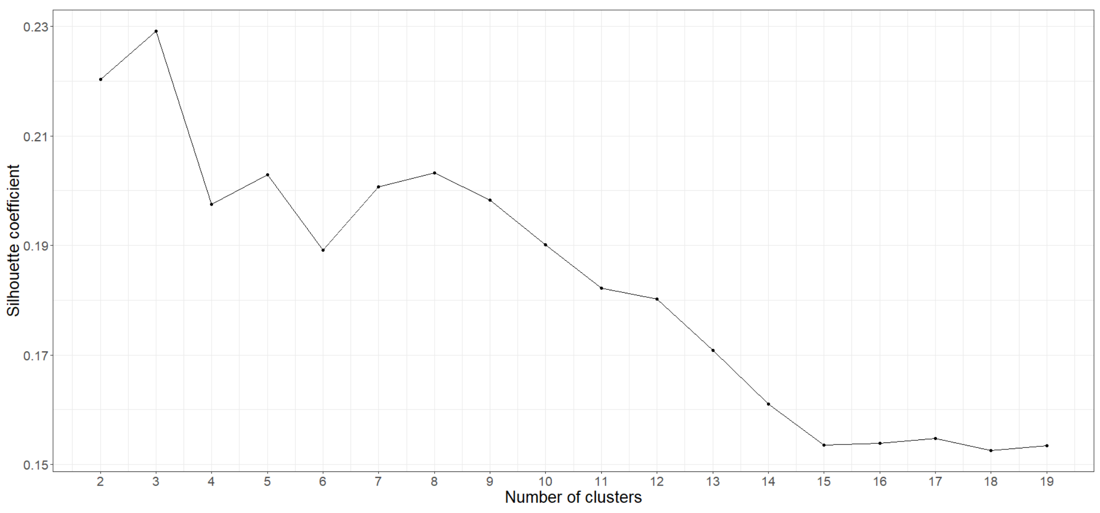

Figure 5.6. Plot of the Silhouette Coefficient.

Figure 5.7 shows the clustering scatter plot on the PC1-PC2 plane. The dots show the consumers coloured according to their cluster. Clusters 2 (green), 3 (grey), 7 (pink) and 8 (orange) are well separated from the other clusters while clusters 1 (brown) and 5 (cyan) overlap on each other quite heavily for these two components. Similarly, cluster 6 (blue) overlaps on cluster 4 (violet).

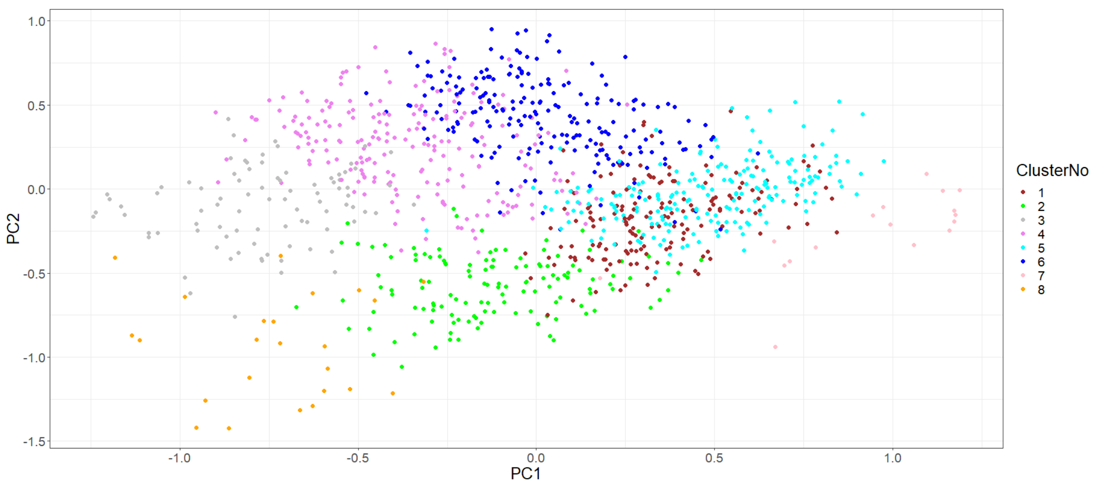

Figure 5.7. Clustering positions with respect to PC1, PC2 plane.

Figure 5.8 shows the scatter plot of the consumers' PCA scores on the PC1-PC3 plane. On this plane the points for clusters 1 and 5 (which overlap on PC1, PC2 plane) are well separated from each other. This means that the difference in clusters 1 and cluster 5 lies in features which are mostly represented by PC3. In this plane cluster 3 overlaps cluster 4 while cluster 6 overlaps clusters 2 and 4, meaning that consumers belonging to these clusters share similarities in features represented by PC3.

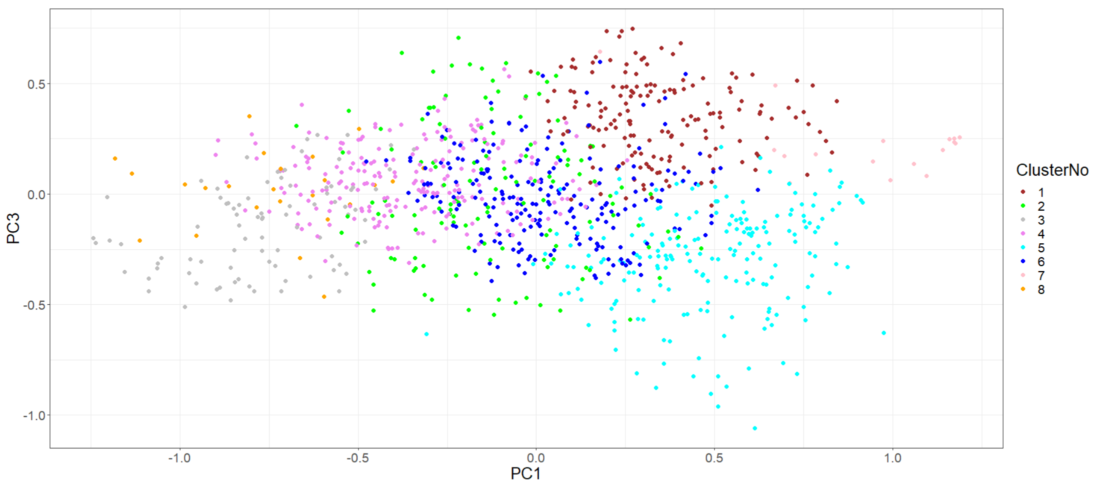

Figure 5.8. Clustering positions with respect to PC1, PC3 plane.

The scatter plot for consumers' PCA scores relative to PC2 and PC3 is shown in Figure 5.9. On this plane the points for cluster 2 (green) and cluster 3 (grey) which were fairly separated on the PC1-PC2 and PC-PC3 planes, are overlaid. Hence consumers in clusters 2 and 3 share features which are mostly represented by PC3.

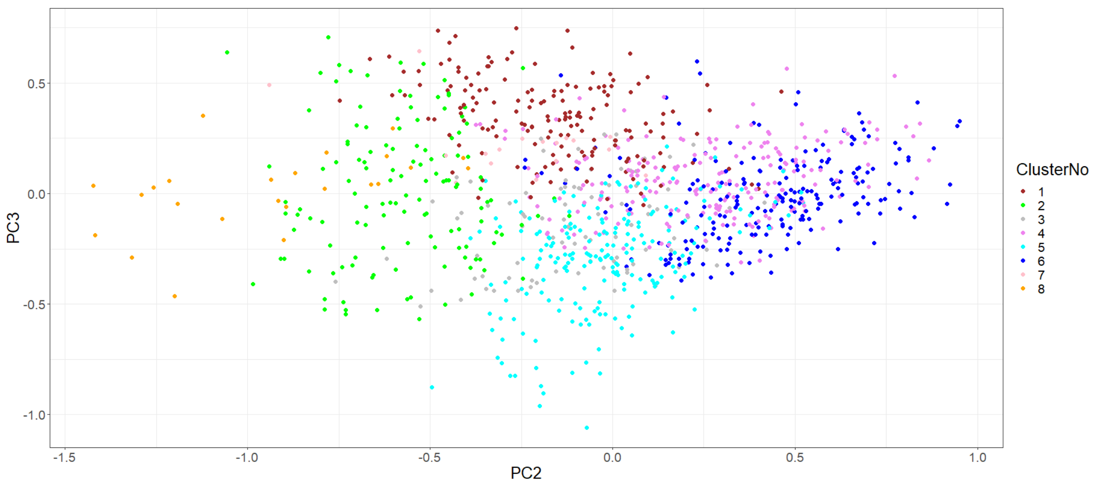

Figure 5.9. Clustering positions with respect to PC2, PC3 plane.

The scatter plot of PCA scores on the PC2-PC4 plane, Figure 5.10, shows the points for consumers in cluster 6 (blue) are well separated from the other clusters, meaning that consumers in cluster 6 differ in features which are mostly represented by PC4. Clusters 4 (pink) and 8 (orange) are also well distinguished but clusters 1 (brown), 2 (cyan) and 3 (grey) are quite superimposed, implying that the consumers in these clusters have similarities in the features which are mostly represented by PC4. In general, all other clusters are merged into each other as the variance diminishes for subsequent principal components.

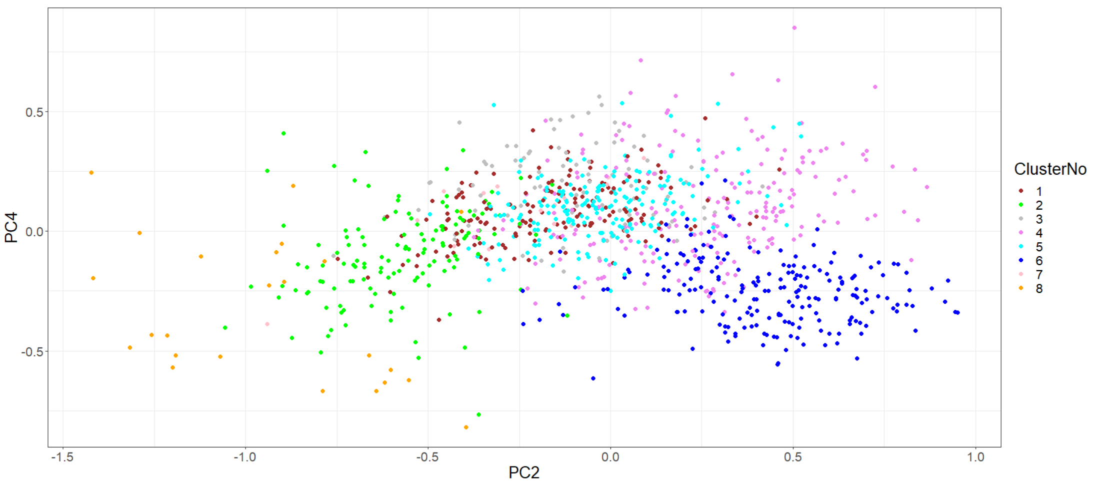

Figure 5.10. Clustering positions with respect to PC2, PC4 plane.

Figure 5.11 shows the biplot of the PC1 and PC2 loadings (eigenvectors) superimposed over the clustered PCA score positions. This figure is similar to Figure 5.2 but here the PCA scores are coloured according to the cluster to which they have been assigned. The direction and magnitude of the arrows also give an indication of the behaviour of the consumers in the clusters. The PCA loadings for *WDGSumRatio* (Week Day niGht Sum Ratio) and *WEGSumRatio* (Week End niGht Sum Ratio) point in directions which covers most of the consumers of clusters 5 and 7, suggesting that these consumers have substantial consumption during the night.

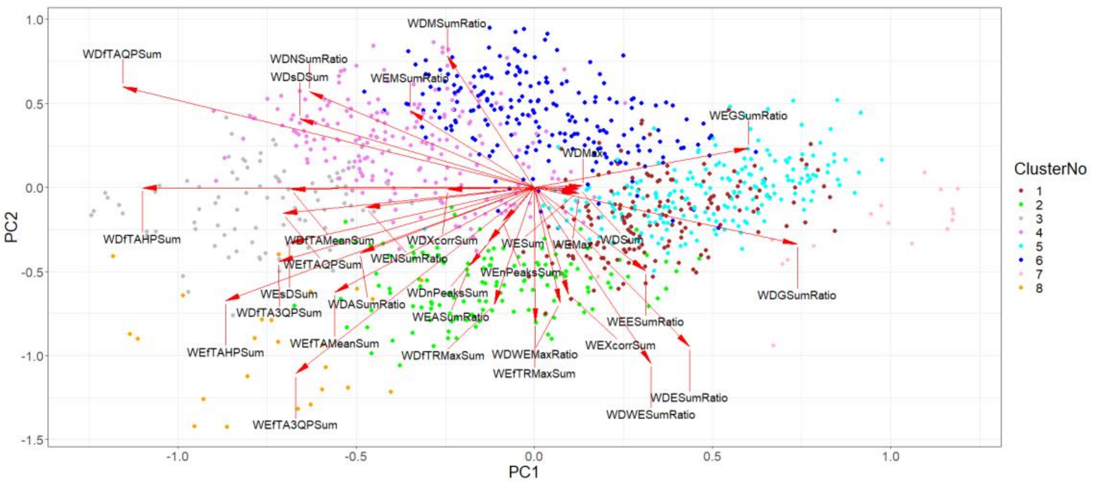

Figure 5.11. Biplot - PC1 and PC2 loadings superimposed on clustered positions.

Having been assigned on the principal components rather than directly on the consumption features, each cluster cannot be directly associated with a unique behavioural practice. An insight of the common behaviour of consumers pertaining to a cluster may be obtained by looking at the means for each consumption feature for each cluster. Table 5.6 displays such means against the feature abbreviation. The values are the means of the normalised quantities and so it is not trivial to give an interpretation. But an indication can be obtained such as for cluster 7, *WDGSumRatio* (Week Day niGht Sum Ratio) 0.7807 and *WEGSumRatio* (Week End niGht Sum Ratio) 0.7897, suggests that consumers in this clusters exhibit more night consumption than day consumption. For cluster 3 the values for WDASumRatio (Week Day Afternoon Sum Ratio), WENSumRatio (Week End Noon Sum Ratio) and WDNSumRatio (Week Day Noon Sum Ratio) all have values above 0.6 which indicates that the maximum consumption is concentrated around noon.

Table 5.6. Means for features in each cluster.

Feature	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8
WDSum	0.0066	0.0099	0.0048	0.0074	0.0932	0.0164	0.0025	0.0038
WDMax	0.0119	0.0177	0.0113	0.0185	0.1146	0.0332	0.0065	0.0110
WESum	0.0062	0.0104	0.0038	0.0041	0.0851	0.0081	0.0032	0.0042
WEMax	0.0107	0.0181	0.0104	0.0126	0.1062	0.0131	0.0074	0.0115
WDXcorrSum	0.3344	0.5659	0.6106	0.5289	0.5592	0.5623	0.5517	0.6131
WEXcorrSum	0.4538	0.6595	0.5466	0.4249	0.5985	0.4396	0.6669	0.5982
WDsDSum	0.1815	0.2221	0.4017	0.3840	0.0954	0.3190	0.3971	0.3204
WEsDSum	0.2345	0.3182	0.4923	0.3722	0.1191	0.1120	0.5711	0.4641
WDfTAQPSum	0.0420	0.1089	0.5105	0.3808	0.0115	0.2837	0.0209	0.4470
WEfTAQPSum	0.0287	0.0869	0.3693	0.1735	0.0068	0.0144	0.0207	0.3589
WDfTAHPSum	0.1731	0.4085	0.5011	0.4295	0.0784	0.4027	0.0350	0.7791
WEfTAHPSum	0.1388	0.3558	0.4155	0.2659	0.0483	0.0529	0.0410	0.6449
WDfTA3QPSum	0.3628	0.5599	0.4872	0.4406	0.2301	0.4225	0.1482	0.8026
WEfTA3QPSum	0.3379	0.5640	0.4646	0.3276	0.1927	0.1351	0.1706	0.7533
WDfTRMaxSum	0.6033	0.7371	0.6040	0.5348	0.5654	0.5169	0.5948	0.8552
WEfTRMaxSum	0.5649	0.7085	0.5605	0.4614	0.5419	0.4777	0.5405	0.7548
WDfTAMeanSum	0.2142	0.4502	0.5281	0.4500	0.3469	0.4442	0.0665	0.7255
WEfTAMeanSum	0.1926	0.4188	0.4812	0.3144	0.2998	0.1725	0.0572	0.5991
WDnPeaksSum	0.5464	0.3954	0.3102	0.3274	0.1491	0.2754	0.1766	0.3306
WEnPeaksSum	0.4173	0.3008	0.2271	0.2453	0.1074	0.2220	0.1446	0.2591
WDWESumRatio	0.5133	0.5805	0.4060	0.2844	0.4843	0.2231	0.5675	0.6111
WDWEMaxRatio	0.3551	0.4115	0.3706	0.2730	0.3657	0.1243	0.4188	0.4162
WDMSumRatio	0.3345	0.2835	0.3822	0.5137	0.3433	0.5077	0.1022	0.1876
WDNSumRatio	0.3354	0.3909	0.6324	0.5465	0.3671	0.5485	0.0822	0.2959
WDASumRatio	0.4141	0.4875	0.6672	0.4415	0.4285	0.3657	0.1408	0.5970
WDESumRatio	0.3799	0.4703	0.1575	0.1620	0.3419	0.1881	0.6894	0.6854
WDGSumRatio	0.3953	0.2957	0.1151	0.1851	0.3794	0.2187	0.7807	0.2487
WEMSumRatio	0.3334	0.2799	0.4070	0.5257	0.3413	0.3357	0.1048	0.2064
WENSumRatio	0.3413	0.4149	0.6339	0.4083	0.3571	0.3365	0.0810	0.3438
WEASumRatio	0.4398	0.5344	0.5918	0.3487	0.4437	0.4199	0.1497	0.6159
WEESumRatio	0.3865	0.4799	0.2004	0.2575	0.3610	0.3691	0.6998	0.6623
WEGSumRatio	0.4121	0.2923	0.1563	0.3087	0.4072	0.4365	0.7897	0.2496

Figure 5.12 is a plot of two consumers, m1 (red) from cluster 3 and m2 (blue) from cluster 7, for six weeks. As can be observed the consumer from cluster 3 shows most consumption around noon while the consumer from cluster 7 shows the consumption during the night. The load profiles are also normalised so that comparison is made on the pattern of the behaviour rather than on the magnitude of the consumption.

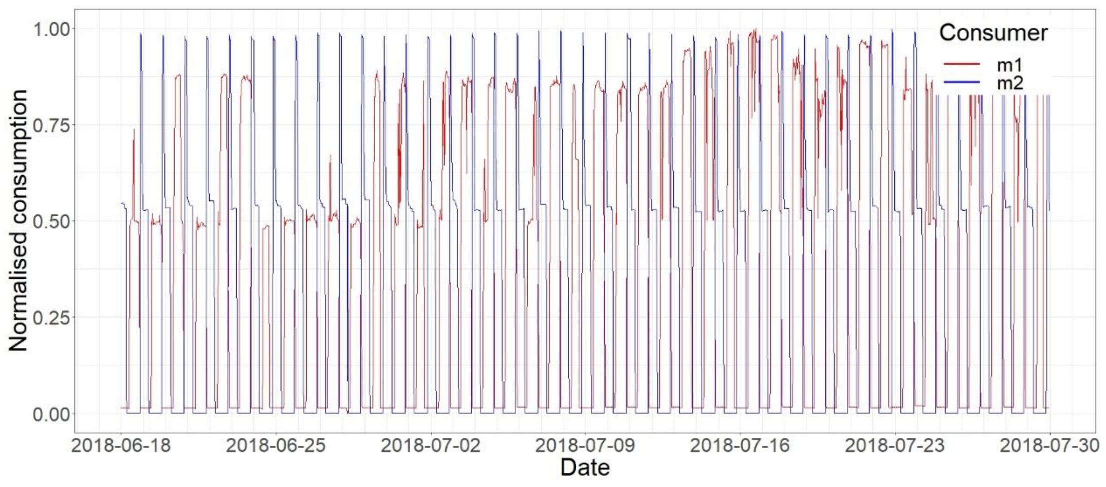

Figure 5.12. A comparison of clusters 3 and 7.

In cluster 6 the mean values for WDWESumRatio (WeekEnd / WeekDay Sum Ratio) and WDWEMaxRatio (WeekDay / WeekEnd Max Ratio) indicate that consumers in this cluster are more likely not to have consumption on weekends than consumers in cluster 3. Figure 5.13 shows the load profile of a consumer, m2 (in blue), from cluster 6 and the same consumer m1 (in red) shown in Figure 5.12. As can be seen m2 does not have consumption on weekends.

Note: In these plots the same consumer from cluster 3 (m1 in red) is shown so that comparison can be made with the load profile of the consumer from the cluster being considered.

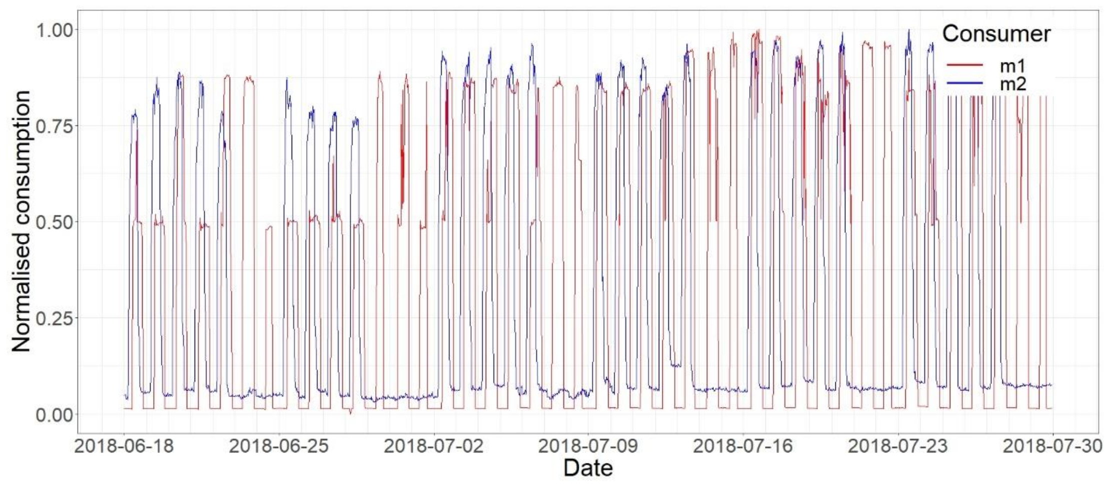

Figure 5.13. A comparison of clusters 3 and 6.

For cluster 8 the means of the temporal features all have a high value which indicate that the consumption is concentrated in the late hours of the day. Also, both *WDESumRatio* (Week Day Evening Sum Ratio) and *WEESumRatio* (WeekEnd Evening Sum Ratio) have a value above 0.6 which further confirms that the consumption for consumers in this cluster is mostly in the evenings. Figure 5.14 shows the load profile for a consumer (m2 in blue) from cluster 8 together with the same consumer m1 from cluster 3. From the plot it can be noticed that the peak of the consumption for m2 is concentrated in the evenings.

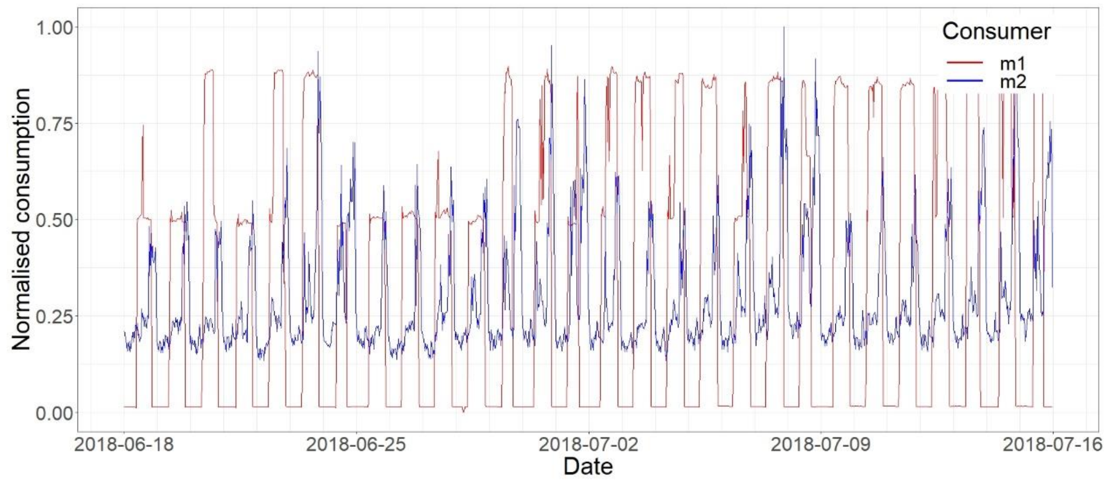

Figure 5.14. A comparison of clusters 3 and 8.

Observing the means for cluster 5 it is noted that the ratios of the sums for the morning, noon, afternoon, evening and night do not differ by much which indicated that the consumption does not differ much between different segments of the day. Moreover, the fact that the means for the cross-correlation features (*WEXcorrSum* and *WDXcorrSum*) are both above 0.55 indicates that the consumers in this cluster show quite similar behaviour on successive days. Figure 5.15 shows the load profile for a consumer m2 (blue) from cluster 5, where it can be seen that the consumption is lower during the night but not by much.

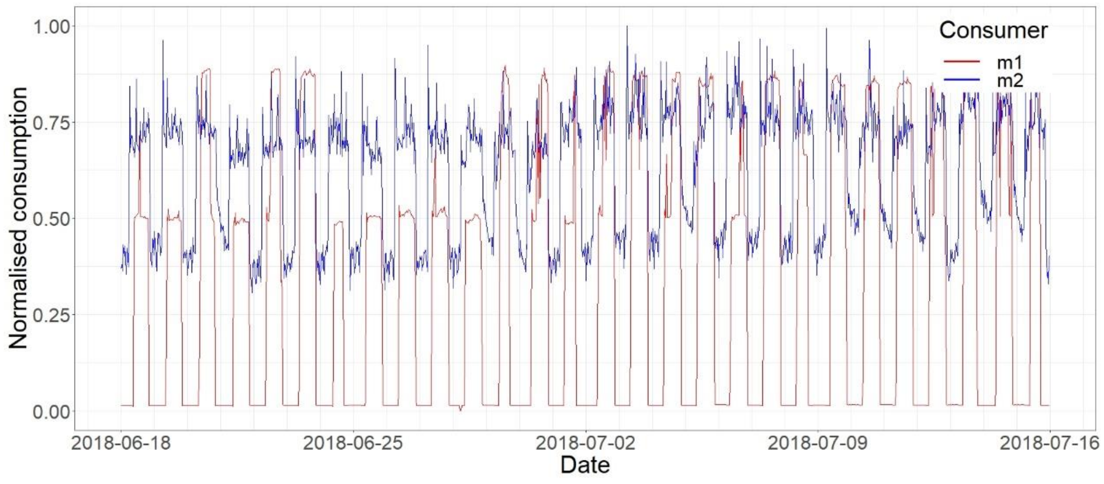

Figure 5.15. A comparison of clusters 3 and 5.

For cluster 1 the means for features WDnPeaksSum (WeekDay number of Peaks Sum) and WEnPeaksSum (WeekEnd number of Peaks Sum) are 0.5464 and 0.4173 respectively, which is quite high when compared to the means of the other features. This indicates that the consumption exhibits a number of spikes during the day. Figure 5.16 shows the load profile for a consumer m2 (in blue) from cluster 1. The consumption can be seen to have a number of peaks during each day.

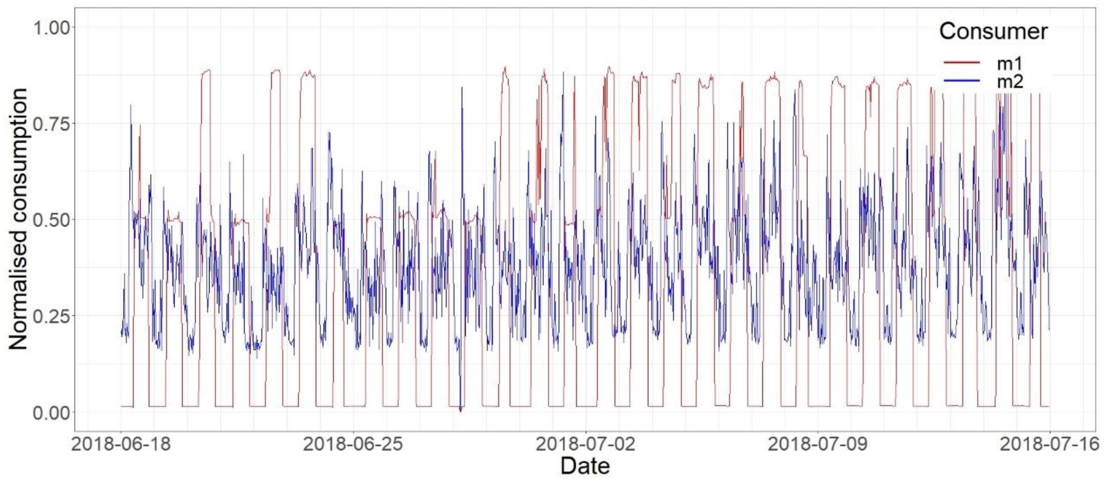

Figure 5.16. A comparison of clusters 3 and 1.

For cluster 2 the mean for WDWESumRatio (WeekEnd/WeekDay Sum Ratio) is quite high when compared with other clusters. This indicates that consumers in this cluster have more consumption on weekends than weekdays. Figure 5.17 shows the load profile for a consumer m2 (in blue) from cluster 2. As expected, the average consumption is higher on the weekends than on weekdays.

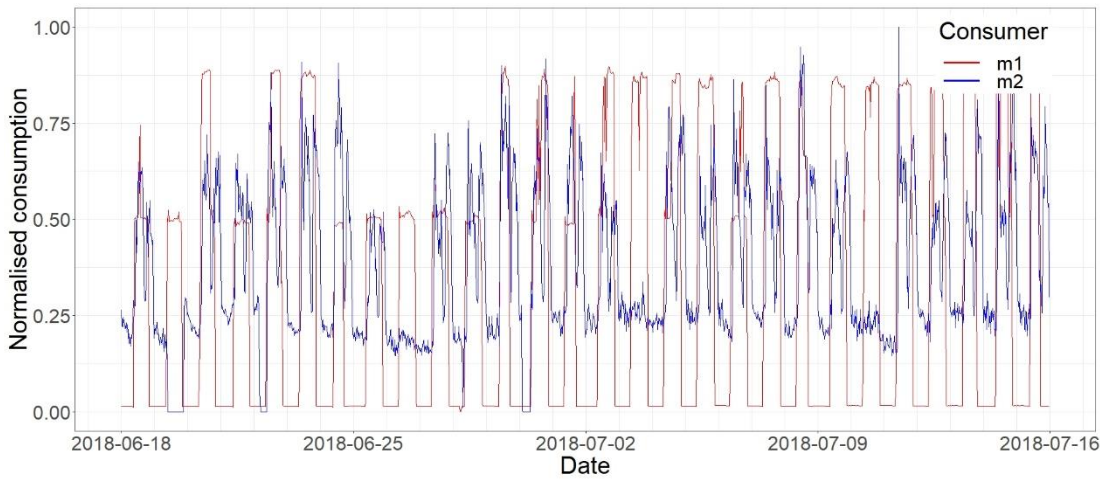

Figure 5.17. A comparison of clusters 3 and 2.

Figure 5.18 shows the load profile for a consumer (m2 in blue) from cluster 4 together with the same consumer (m1 in red) from cluster 3. It can be observed that there is no consumption on Sundays for consumer m2. All the means of the features in cluster 4 are very similar to the means of the features in cluster 6 (which has no consumption on weekends). The only means that differ are the ratios of the sums for the weekends, the means for cluster 4 are twice those for cluster 6. This indicated that the weekend consumption for consumers in cluster 4 is twice that for consumers of cluster 6. The most probable situation would be that their only consumption is on Saturdays.

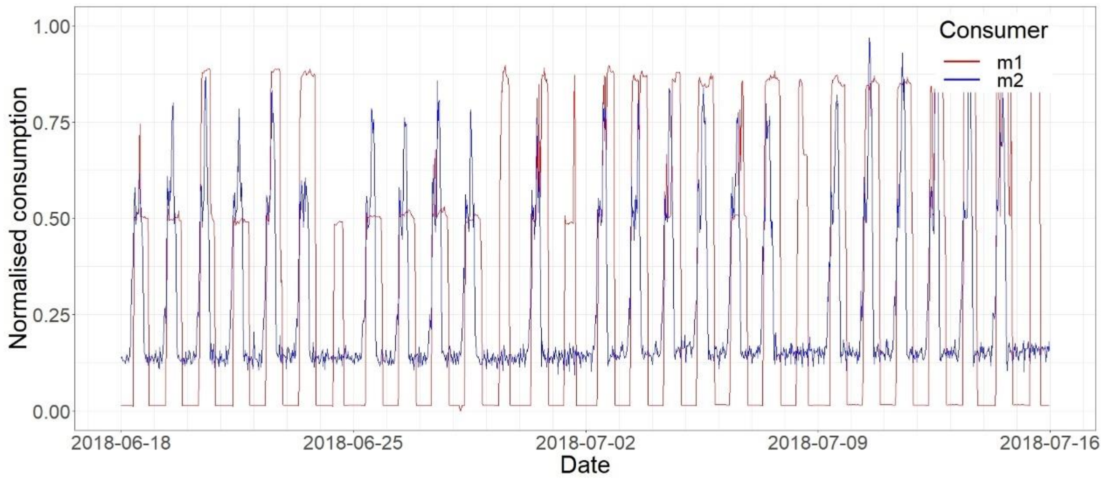

Figure 5.18. A comparison of clusters 3 and 4.

5.6 ANOVA Testing

In order to gauge the statistical quality of the clustering obtained for each principal component, an ANOVA test was performed on each of the considered principal components. Actually, for clustering the first seven principal components were considered but the ANOVA test was performed on the first eight principal components so that the quality of the clustering can be compared to that of the other seven principal components. A prerequisite for using ANOVA is that each group sample is drawn from a normally distributed population and hence a test for normality is appropriate. Here three tests for normality were used: histogram plot, Q-Q plot and Kolmogorov-Smirnov (K-S) test.

The most straight forward method for determining if a population is normally distributed is to plot the values on a histogram and judge how closely it resembles a normal distribution. The values (scores) of each PC were clustered and so the scored of each PC were considered as eight separate populations. Figure 5.19 shows the histograms for the first eight principal component scores with a horizontal bin width of 0.05. As can be noted, by observing the plots, it is not trivial to decide if the distributions are normal or how different they are from normal. PC3 and PC6 look very close to a normal distribution curve while PC4 looks a bit skewed. The other do not seem very far off from normal.

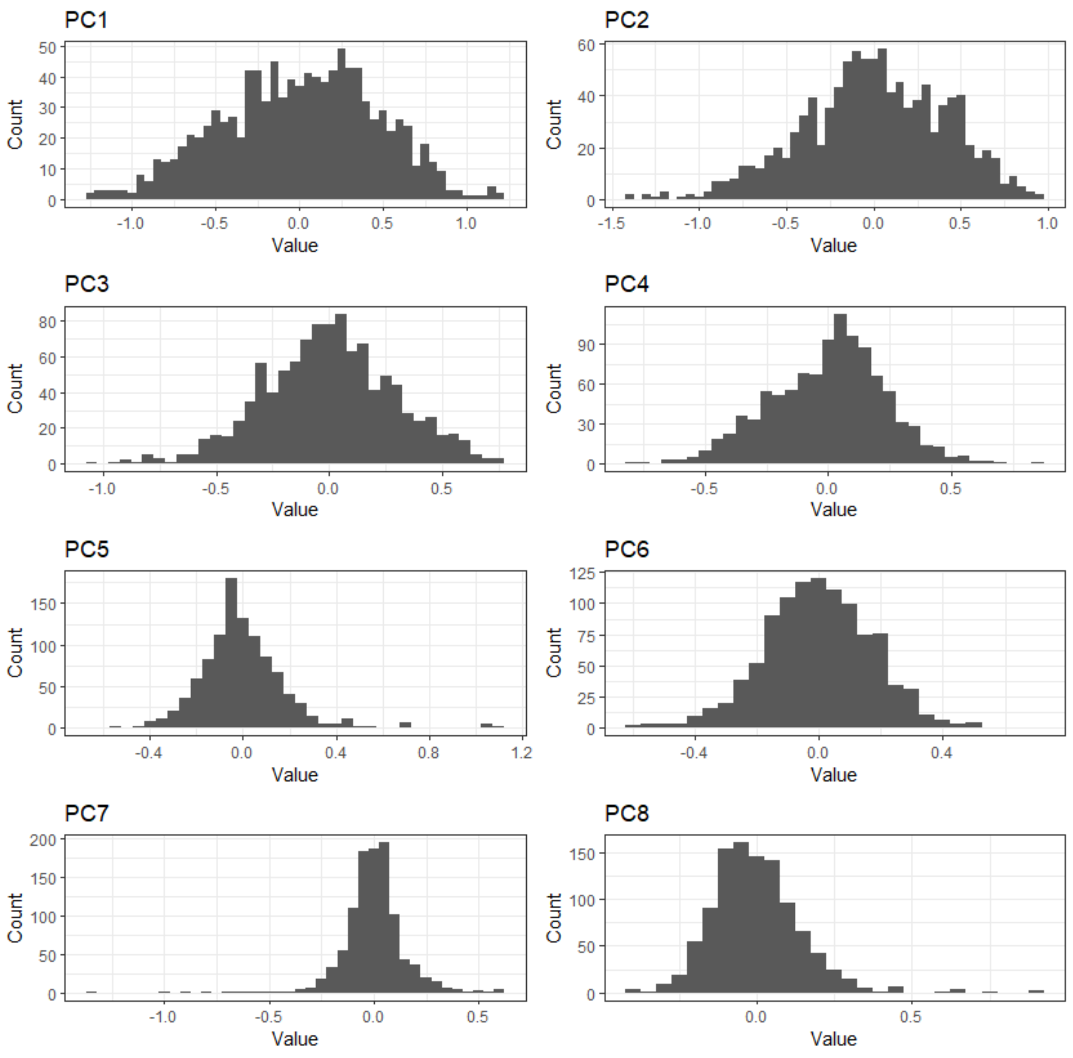

Figure 5.19. The histograms for the first eight principal components

Another method to graphically observe the normality of a distribution is the Quantile-Quantile (Q-Q) plot which is a plot of probability. A Q-Q plot is a scatterplot which plots two sets of quantiles against each other. If both sets of quantiles have the same distribution the points would lie on a straight line. Figure 5.20 shows the Q-Q plots where the horizontal axis represent the normal distribution quantiles while the vertical axis represents the quantiles for the respective PC score values. The red reference line is where the point would be if the distribution was perfectly normal and the grey areas are the confidence bands. From these Q-Q plots it can be seen that apart from PC5, PC7 and PC8, the points for the other PCs are mostly within the confidence bands.

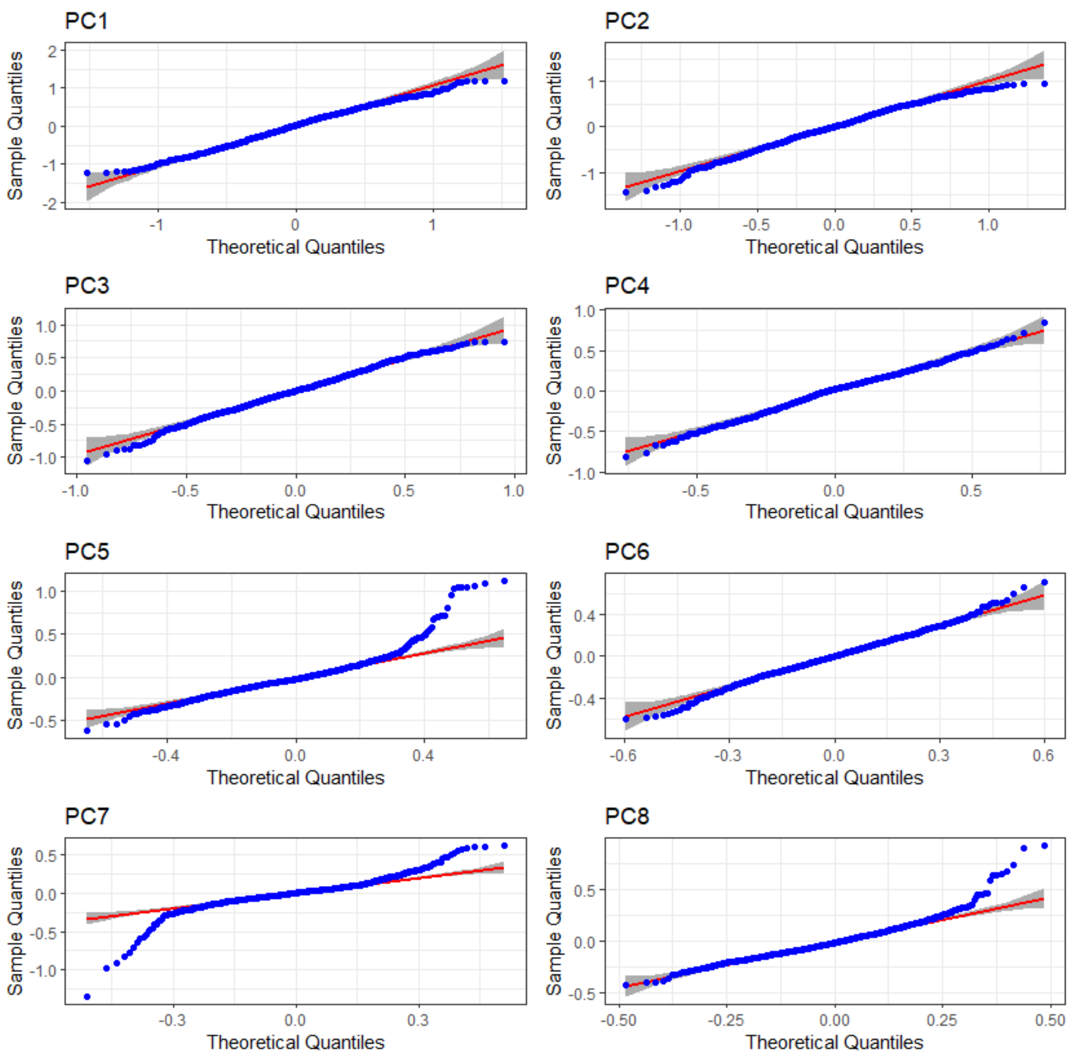

Figure 5.20. The Q-Q plots for the first eight principal components.

The Kolmogorov-Smirnov test [112] is used to check the underlying distribution of a variable by matching it to standard distributions such as the Normal, Poisson, Exponential or Uniform distribution. In *R* the Kolmogorov-Smirnov test may be performed by calling function *ks.test()* from *stats* package. The template distribution is passed as an argument to the function together with the mean and standard deviation. Table 5.7 lists the Kolmogorov-Smirnov significance values obtained for each PC. If the significance value is greater than 0.05 then the null hypothesis is accepted i.e. the distribution is normal. So PC1, PC2, PC3 and PC6 do have a normal distribution while that of PC4, PC5, PC7 and PC8 are not exactly normal.

Table 5.7. Significance value of the Kolmogorov-Smirnov statistic for each PC.

	Significance value
PC1	0.0891005
PC2	0.3721033
PC3	0.6561228
PC4	0.01312868
PC5	2.15E-07
PC6	0.5549213
PC7	5.81E-09
PC8	0.00179899

The three tests for normality that were conducted do complement each other for all PCs except for PC4 which looks close to normal on the Q-Q plot and also on the histogram.

For ANOVA, as for most statistical tests, the null hypothesis must be stated beforehand. In this case the null hypothesis is that there is no difference between the clusters and consumers could have been placed in any cluster randomly. The alternative hypothesis is that there is a difference between the clusters and consumers were placed in the relative cluster based on their behaviour.

In *R* to function *aov()* calculates the ANOVA parameters and returns an *aov* object which can be viewed by using the *summary()* function. Table 5.8 lists the summary of output from *aov()* function for each principal component from PC1 to PC8. The function *summary()* first lists the independent variables being tested in the model (in this case we have only one, *Cluster*) and then the *Residuals*. All of the variation that is not explained by the independent variables is called residual variance. The *Df* column displays the degrees of freedom for the independent variable (the number of levels in the variable minus 1), and the degrees of freedom for the residuals (the total number of observations minus one and minus the number of levels in the independent variables). The *Sum Sq* column displays the sum of squares (i.e. the total variation between the specific group mean and the overall mean). The *Mean Sq* column is the mean of the sum of squares, calculated by dividing the sum of squares by the degrees of freedom for each parameter. The *F-value* column is the test statistic from the *F* test. The *F-value* is obtained by dividing the mean square of the specific independent variable by the mean square of the residuals. The larger the *F value*, the more likely it is that the variation caused by the independent variable is real and not due to chance. The *Pr(>F)* column is the *p-value* of the *F-statistic*. The *p-value* shows how likely it is that the *F-value* calculated from the test would have occurred if the null hypothesis, of no difference among group means, were true.

As can be noted from Table 5.8, the *p-value* for all PCs are much lower than 0.05 (or even 0.01 for a confidence level of 99%) which is the level of significance above which the null hypothesis would be accepted. Hence for all PCs the null hypothesis is rejected, and the alternative hypothesis accepted. That is, there is a significant statistical difference between the consumers in each cluster. Analysing further the *F value*, it is deduced that for PC1 the clusters are more separated than for the rest of the PCs. The level of separation, indicated by the *F value*, continues to diminish for the other PCs with the lowest being PC8. This decreasing trend in *F value* further collaborated the decision to consider only the first 7 PCs.

Table 5.8. Summary of *aov()* output for each PC.

		Df	Sum Sq	Mean sq	F value	Pr(>F)
PC1	Cluster	7	172.5	24.64	511.1	<2e-16
	Residuals	1033	49.8	0.048		
PC2	Cluster	7	122.85	17.55	345.4	<2e-16
	Residuals	1033	52.49	0.051		
PC3	Cluster	7	40.58	5.797	128.3	<2e-16
	Residuals	1033	46.65	0.045		
PC4	Cluster	7	25.77	3.681	127.8	<2e-16
	Residuals	1033	29.75	0.029		
PC5	Cluster	7	17.3	2.4715	110.3	<2e-16
	Residuals	1033	23.14	0.0224		
PC6	Cluster	7	4.04	0.5772	19.81	<2e-16
	Residuals	1033	30.09	0.0291		
PC7	Cluster	7	4.051	0.5787	28.55	<2e-16
	Residuals	1033	20.937	0.0203		
PC8	Cluster	7	1.517	0.21676	10.71	4.46e-13
	Residuals	1033	20.9	0.02023		

The ANOVA test determined that not all cluster means are equal, but it does not specify that all cluster means are different. To determine which clusters are different from others a post hoc test has to be conducted such as the Tukey test. The *R* function *TukeyHSD()* (Tukey Honest Significant Differences) performs multiple pairwise-comparison between the means of groups.

The function *TukeyHD()* takes the fitted ANOVA object as an argument and produces an output such as Table 5.9 (for PC1) and Table 5.10 (for PC8).

The rows-names are the groups (clusters in this case) which are being compared. Thus row 1 shows the comparison of cluster 2 and cluster 1. Row 2 shows the comparison of cluster 3 and cluster 1, and so on.

The columns headings of Table 5.9 and Table 5.10 are :

- diff: difference between means of the two groups.
- lwr, upr: the lower and the upper end point of the confidence interval at a default of 95%.
- p adj: p-value after adjustment for the multiple comparisons.

Observing the *diff* and *p adj* values, the clusters which have significant difference in means can be determined. For PC1, Table 5.9, *p adj* is small for all cluster comparisons except for the comparison of cluster 8 to cluster 3 where it is 0.998. This means that the means of these two clusters are close. *diff* for these two clusters is small, 0.0318, further indicating the closeness of their means. For the other combinations of clusters *diff* indicates the dissimilarities between the cluster combinations.

In Table 5.10, which shows the output of *TukeyHSD()* for PC8, it can be noted that most *p adj* are greater than 0.05 and *diff* are small. This indicates that the means for most cluster combinations are very close in PC8.

Table 5.9. Tukey comparisons for PC1.

	diff	lwr	upr	p adj
2-1	-0.46439464	-0.54264083	-0.3861485	0.0000000
3-1	-1.11068338	-1.19922257	-1.0221442	0.0000000
4-1	-0.69894244	-0.76855217	-0.6293327	0.0000000
5-1	0.13703954	0.06659409	0.2074850	0.0000001
6-1	-0.32045415	-0.39006389	-0.2508444	0.0000000
7-1	0.58211491	0.41215872	0.7520711	0.0000000
8-1	-1.07880474	-1.22459898	-0.9330105	0.0000000
3-2	-0.64628874	-0.73851444	-0.5540630	0.0000000
4-2	-0.23454780	-0.30879001	-0.1603056	0.0000000
5-2	0.60143418	0.52640784	0.6764605	0.0000000
6-2	0.14394049	0.06969828	0.2181827	0.0000001
7-2	1.04650955	0.87460406	1.2184150	0.0000000
8-2	-0.61441010	-0.76247208	-0.4663481	0.0000000
4-3	0.41174094	0.32671960	0.4967623	0.0000000
5-3	1.24772292	1.16201601	1.3334298	0.0000000
6-3	0.79022923	0.70520789	0.8752506	0.0000000
7-3	1.69279829	1.51597007	1.8696265	0.0000000
8-3	0.03187864	-0.12187140	0.1856287	0.9984707
5-4	0.83598198	0.77001229	0.9019517	0.0000000
6-4	0.37848829	0.31341177	0.4435648	0.0000000
7-4	1.28105735	1.11290699	1.4492077	0.0000000
8-4	-0.37986230	-0.52354737	-0.2361772	0.0000000
6-5	-0.45749369	-0.52346339	-0.3915240	0.0000000
7-5	0.44507537	0.27657732	0.6135734	0.0000000
8-5	-1.21584428	-1.35993608	-1.0717525	0.0000000
7-6	0.90256906	0.73441870	1.0707194	0.0000000
8-6	-0.75835059	-0.90203566	-0.6146655	0.0000000
8-7	-1.66091965	-1.87230790	-1.4495314	0.0000000

Table 5.10. Tukey comparisons for PC8.

	diff	lwr	upr	p adj
2-1	-0.0344699448	-0.0851610928	0.016221203	0.4383783
3-1	0.0254469061	-0.0319124790	0.082806291	0.8803378
4-1	0.0077119089	-0.0373841800	0.052807998	0.9995651
5-1	-0.0008178061	-0.0464553072	0.044819695	1.0000000
6-1	-0.0278482695	-0.0729443583	0.017247819	0.5681551
7-1	0.2576522025	0.1475474916	0.367756913	0.0000000
8-1	-0.0190427560	-0.1134943500	0.075408838	0.9987234
3-2	0.0599168509	0.0001691932	0.119664509	0.0487623
4-2	0.0421818537	-0.0059153475	0.090279055	0.1350850
5-2	0.0336521387	-0.0149530569	0.082257334	0.4137089
6-2	0.0066216754	-0.0414755259	0.054718877	0.9998979
7-2	0.2921221473	0.1807545986	0.403489696	0.0000000
8-2	0.0154271889	-0.0804935448	0.111347923	0.9997105
4-3	-0.0177349972	-0.0728153689	0.037345375	0.9774990
5-3	-0.0262647122	-0.0817892259	0.029259801	0.8400796
6-3	-0.0532951756	-0.1083755473	0.001785196	0.0661586
7-3	0.2322052964	0.1176485965	0.346761996	0.0000000
8-3	-0.0444896621	-0.1440953533	0.055116029	0.8765123
5-4	-0.0085297150	-0.0512676348	0.034208205	0.9988044
6-4	-0.0355601784	-0.0777194658	0.006599109	0.1713315
7-4	0.2499402936	0.1410054706	0.358875117	0.0000000
8-4	-0.0267546649	-0.1198398511	0.066330521	0.9883758
6-5	-0.0270304633	-0.0697683831	0.015707456	0.5367152
7-5	0.2584700086	0.1493099430	0.367630074	0.0000000
8-5	-0.0182249498	-0.1115736304	0.075123731	0.9989635
7-6	0.2855004720	0.1765656490	0.394435295	0.0000000
8-6	0.0088055135	-0.0842796727	0.101890700	0.9999921
8-7	-0.2766949585	-0.4136410867	-0.139748830	0.0000000

The clustering for each of the first PCs is also shown in boxplot format in Figure 5.2. Here it can be further observed that PC1 exhibits the most separation between the clusters, with clusters 1 and 8 differing quite significantly and cluster 3 and cluster 8 are close as was already deduced by the *TukeyHSD()* comparison. The separation can here also be noticed to decrease until PC8 where the separation is minimal. These arguments collaborate the observations already brought forward in Section 5.5, when the separation of the clusters was examined by observing the scatter plots for the PCA scores relative to the principal components.

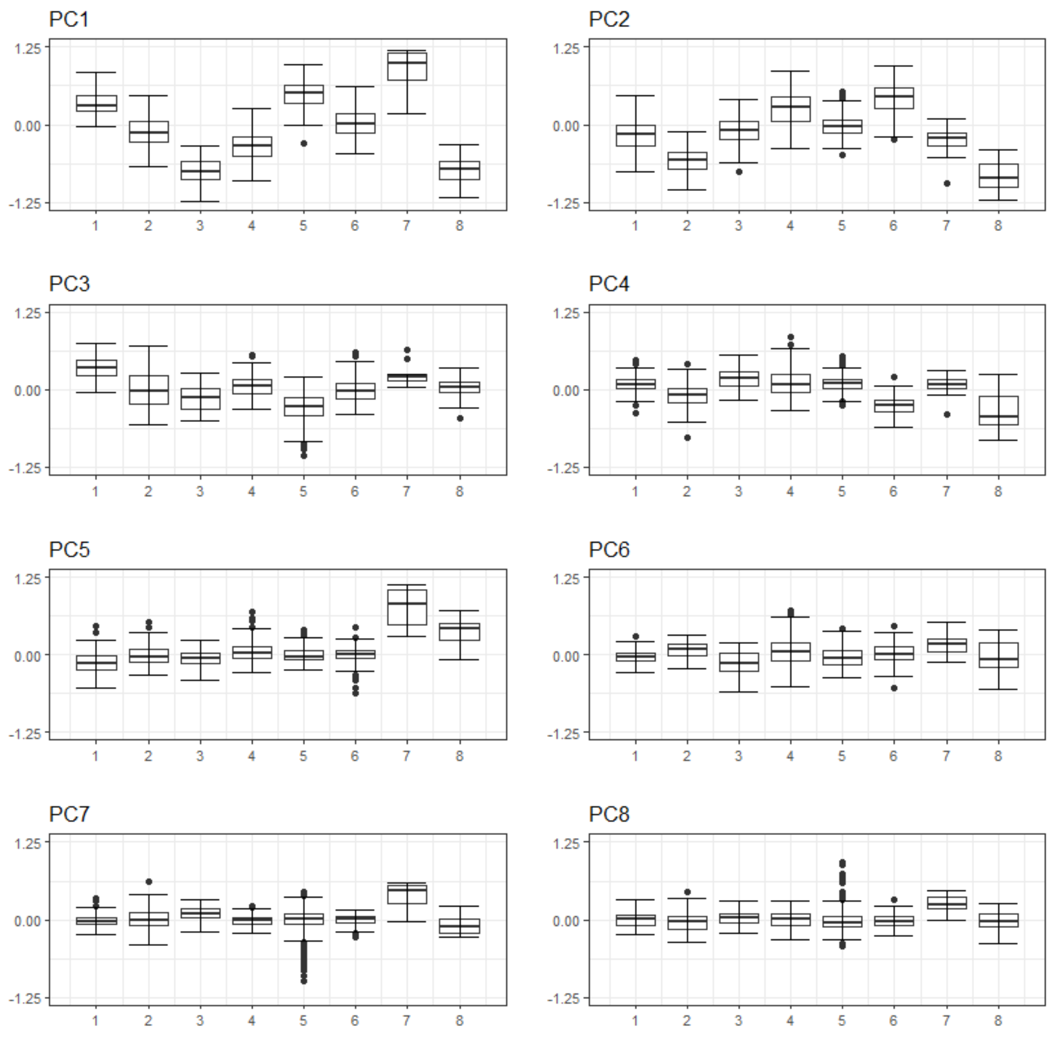

Figure 5.21. Box plots for each PC. Horizontal axis shows cluster number.

5.7 K-Nearest Neighbour Classification

K-Nearest neighbour (KNN) is a machine learning method which can be used for classification and regression. KNN is the most basic instance-based learning method [113]. Instance-based learning methods compare new problem instances with instances seen in training, which have been stored in memory, instead of performing explicit generalisation. In this work KNN is used as a classifier to complement the clustering performed by the hierarchical clustering. The clustering obtained by hierarchical clustering is passed as the training dataset to KNN, using the cluster numbers as labels and obtaining the classification for each instance in the test dataset.

This technique is used, although for a different purpose, by Mylonas et al. in [114]. Both hierarchical clustering and KNN are based on the distances between observations, not the centroids of the clusters, and thus it is reasonable to use them in conjunction with each other. In fact, the accuracy obtained for the validations performed on the customer data was good.

In *R* the function *knn3Train(),* from the package *caret*, may be used for running the KNN algorithm. The *knn3Train()* function requires the following parameters:

- *train:* *matrix* of training data – data which was processed by hierarchical clustering.
- *test:* *matrix* of test data – the test data to be classified.
- *cl:* *factor* of true classifications of training set – the cluster numbers obtained from hierarchical clustering.
- *k:* number of neighbours considered – set to 1 for this work.

knn3Train() returns an object of type *factor* with the classification of the *test* data – the predicted classes.

Note that, *k* is the number of neighbours to be considered. If k is set to one the class of the closest observation is taken. In the case where k is set to 2 the class of the closest two observation is recorded. However, if the two observations are in different clusters a tie is encountered. It is not well defined in academic literature how such ties should be resolved. In *knn3Train(),* a close inspection of the source code [115] reveals that it chooses the class randomly, via the *sample()* method. The *sample()* function returns a sample of the specified size from the elements of a list, which in this case will be made up of the tied observations. Hence, it might be a possibility to have different results in repeated tests. Another method of dealing with such ties is to continue to find the next nearest neighbour until the tie is broken and one class contains more nearest neighbours than any other class.

For this work the value of *k* was set to one since it was noted that only one nearest neighbour is needed to achieve the highest accuracy, thus also minimizing the computational cost. This was verified by iterating the value of *k* from 1 to 29, each time calculating the accuracy. The accuracy is the number of instances where the test assigned class matched the training cluster, divided by the number of observations and is thus given by:

$$accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ predictions} \qquad \text{Equation 5.8}$$

The plot of the accuracy against k is shown in Figure 5.22. The plot shows that the accuracy drops rapidly from k=1 to k=2 and then continues to diminish as k increases. Hence it is evident that k=1 gives the best value of accuracy. In the current context correct predictions are the instances when, for a given consumer, the cluster assigned (predicted) by KNN is the same as that assigned by hierarchical clustering (actual).

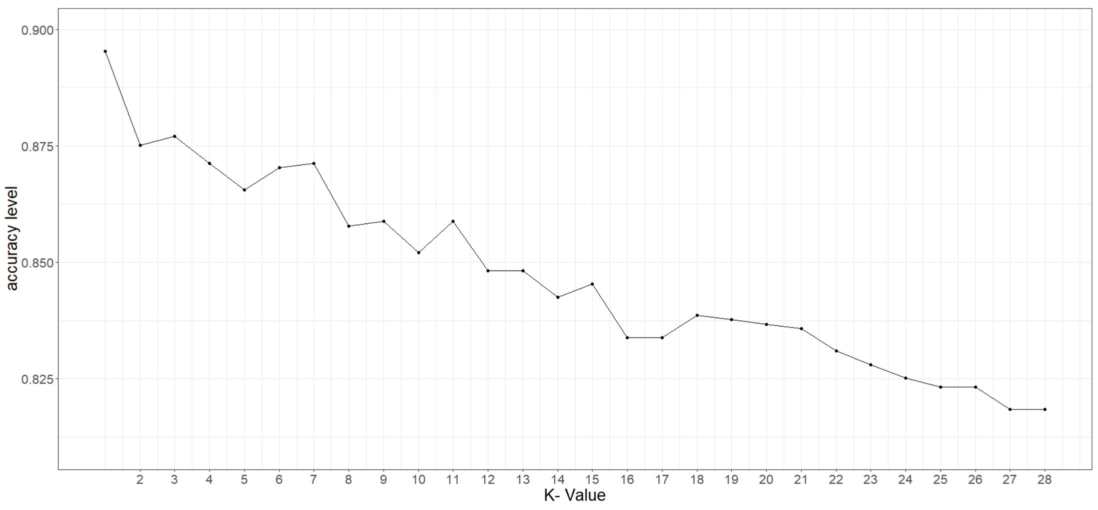

Figure 5.22. Plot of accuracy vs value of *k*.

5.8 3-Fold Cross Validation

When a machine learning mechanism is set up it is very important that the result is validated [116]. Validation is done by splitting the dataset into two parts: training and testing. The model is then trained on the training dataset and tested with the test dataset. However, if the characteristics of the data in the training portion are different from those of the data used for testing, then the model will be *over-trained* or *under-trained* and the accuracy of the model will be wrongly evaluated. K-fold cross validation tackles this issue by splitting the available dataset into *k* segments, or folds, of equal size. The training and testing are performed *k* times, each time using one fold for training and the remaining folds for testing or vice versa. In such a way it is ensured that each observation in the dataset is used for both training and testing at some point. Then an average of the accuracy is taken over the total number of folds.

In this work 3-fold cross validation was used, whereby the 15-week dataset was split into 3 folds of five weeks each fold. Three runs were carried out each time using 1 fold (5 weeks) for training and 2 folds (10 weeks) for testing. In the first run the first fold was used for training and the second and third folds were used for testing. This was repeated for the other two folds each time using one fold for training and two folds for testing. Thus the data was used in three runs as follows:

Run 1: Train with weeks 1,2,3,4,5 - test with weeks 6,7,8,9,10,11,12,13,14,15.

Run 2: Train with weeks 6,7,8,9,10 - test with weeks 11,12,13,14,15,1,2,3,4,5.

Run 3: Train with weeks 11,12,13,14,15 - test with weeks1,2,3,4,5,6,7,8,9,10.

Hence, all weeks are used once for training and twice for testing. For each run the accuracy metric was calculated by finding the sum of the occurrences where the outcome of testing was the same as that of training and dividing by the total number of samples. Then the overall accuracy was determined by taking the average for the three runs. The overall clustering accuracy tested through KNN with k=1, and three-fold validation was 79.22%.

Note that, the load profile dataset being analysed is a time series which may be subject to seasonality. Consumers tend to change behaviour depending on the time of year. The interval was intentionally chosen to be during the summer weeks such that the behaviour of consumers would be as unrelated as possible to seasonality. However, during 15 weeks the behaviour inevitably tends to change and so another method for assigning the folds was adopted. In this approach the folds are made up by grouping together alternate weeks as follows:

Run 1: Train with weeks 1,4,7,10,13- test with weeks 2,3,5,6,8,9,11,12,14,15.

Run 2: Train with weeks 2,5,8,11,14 - test with weeks 1,3,4,6,7,9,10,12,13,15.

Run 3: Train with weeks 3,6,9,12,15 - test with weeks 1,2,4,5,7,8,10,11,13,14.

By using this method, the training and the testing is spread along the whole timespan of the dataset and therefore represents a better averaged behaviour for the consumers. The weekly features were averaged over the 5 training weeks and normalised. PCA was conducted on the normalised means and the resulting, first 7, principal components were clustered using hierarchical clustering. The overall accuracy obtained by this method was 91.99%, which is a significant improvement on the 79.22% accuracy obtained in the former fold grouping.

Figure 5.23 shows the histogram for the number of consumers assigned to each cluster in run 1. Four clusters (2,4,5 and 6) share the majority of consumers quite evenly, while clusters 1, 3 and 7 have a lower but similar number of consumers. Cluster 8 has quite a relatively small number of consumers.

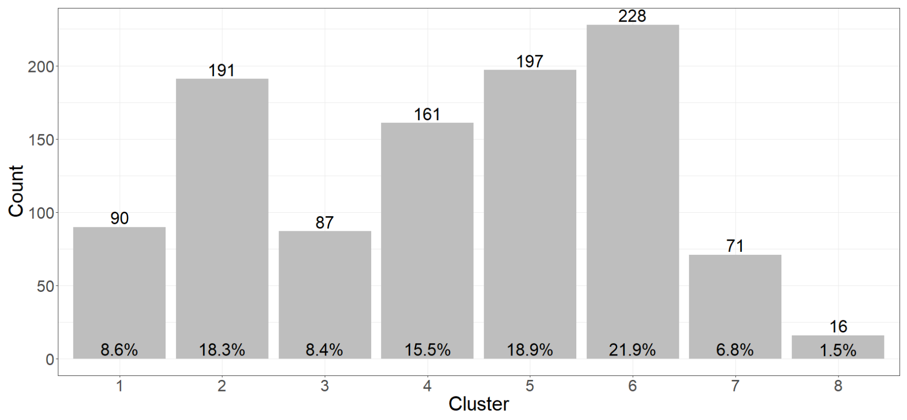

Figure 5.23. Histogram for run 1.

A confusion matrix may be used to measure the performance of a classification method and highlight possible improvements in the methodology. Table 5.11 shows the confusion table for run 1. The column names (highlighted in blue) are the clusters which were assigned to the consumers by the hierarchical clustering while the last row contains the number of consumers in that cluster. The row names (highlighted in green) are the cluster number in which KNN classified the consumers. The cluster 1 column shows the number of consumers from cluster 1 which were classified to each cluster by KNN. The values in the diagonal cells (highlighted in orange) show the numbers of consumers which were correctly classified by KNN, all others were incorrectly classified. So, adding all numbers in the diagonal cells and dividing by the total number of consumers determines the accuracy, which is 93.76%. The values in brackets are the number of consumers in that cell expressed as a percentage of the number of consumers in that cluster.

Table 5.11. Confusion matrix for run 1. Accuracy 93.76%.

	1	2	3	4	5	6	7	8
1	85 (94.4%)	6 (3.1%)	0 (0%)	0 (0%)	2 (1%)	1 (0.4%)	0 (0%)	0 (0%)
2	4 (4.4%)	171 (89.5%)	1 (1.1%)	5 (3.1%)	1 (0.5%)	1 (0.4%)	1 (1.4%)	0 (0%)
3	0 (0%)	4 (2.1%)	86 (98.9%)	1 (0.6%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
4	0 (0%)	0 (0%)	0 (0%)	153 (95%)	0 (0%)	7 (3.1%)	5 (7%)	0 (0%)
5	1 (1.1%)	7 (3.7%)	0 (0%)	0 (0%)	192 (97.5%)	3 (1.3%)	0 (0%)	0 (0%)
6	0 (0%)	3 (1.6%)	0 (0%)	1 (0.6%)	2 (1%)	211 (92.5%)	3 (4.2%)	0 (0%)
7	0 (0%)	0 (0%)	0 (0%)	1 (0.6%)	0 (0%)	5 (2.2%)	62 (87.3%)	0 (0%)
8	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	16 (100%)
Total	90	191	87	161	197	228	71	16

Figure 5.24 shows the histogram with the number of consumers assigned to each cluster in run 2. Here cluster 3, is assigned with a small number of consumers. Three clusters, 2, 5 and 7, have similar numbers while the remaining four clusters, 1, 4 and 6, have higher number of consumers.

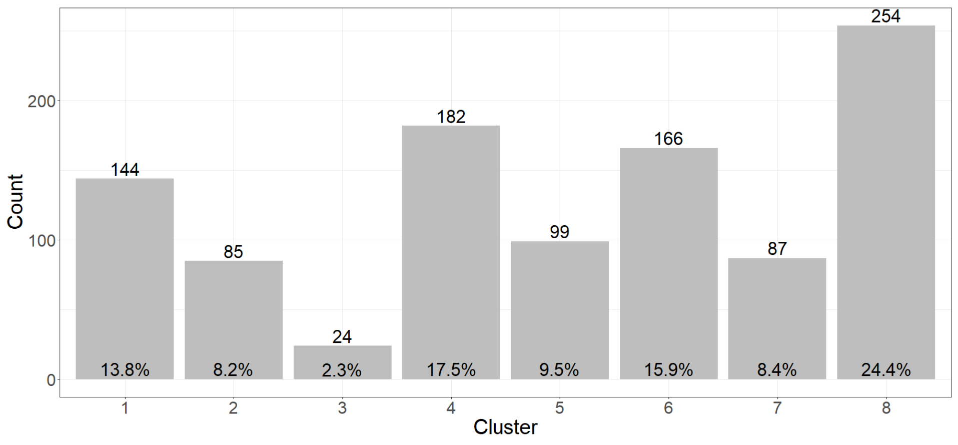

Figure 5.24. Histogram for run 2.

Table 5.12 shows the confusion matrix for run 2 where the overall accuracy was 92.7%. It can be noted that for cluster 3 the classification was 100% as all 24 consumers in this cluster were classified correctly to cluster 3. For cluster 8, from a total of 254 consumers 237 (93.3%) were classified correctly while 17 (6.7%) were classified in cluster 4. This fact may indicate that there is some resemblance between the features of consumers that form part of clusters 4 and 8. Furthermore, 7 consumers from cluster 4 (3.8%) were classified in cluster 8.

Table 5.12. Confusion matrix for run 2. Accuracy 92.70%.

	1	2	3	4	5	6	7	8
1	127 (88.2%)	10 (11.8%)	0 (0%)	0 (0%)	0 (0%)	1 (0.6%)	0 (0%)	0 (0%)
2	7 (4.9%)	72 (84.7%)	0 (0%)	1 (0.5%)	4 (4%)	0 (0%)	1 (1.1%)	0 (0%)
3	0 (0%)	0 (0%)	24 (100%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
4	1 (0.7%)	0 (0%)	0 (0%)	169 (92.9%)	3 (3%)	1 (0.6%)	0 (0%)	17 (6.7%)
5	0 (0%)	1 (1.2%)	0 (0%)	3 (1.6%)	92 (92.9%)	0 (0%)	0 (0%)	0 (0%)
6	2 (1.4%)	1 (1.2%)	0 (0%)	1 (0.5%)	0 (0%)	161 (97%)	2 (2.3%)	0 (0%)
7	5 (3.5%)	1 (1.2%)	0 (0%)	1 (0.5%)	0 (0%)	3 (1.8%)	83 (95.4%)	0 (0%)
8	2 (1.4%)	0 (0%)	0 (0%)	7 (3.8%)	0 (0%)	0 (0%)	1 (1.1%)	237 (93.3%)
Total	144	85	24	182	99	166	87	254

Figure 5.25 shows the histogram with the number of consumers assigned to each cluster in run 3. Here two clusters, 7 and 8, were assigned a small number of consumers. Three clusters, 4,5 and 6, have similarly high numbers while the remaining 3 clusters, 1,2 and 3, have a mid-range number of consumers.

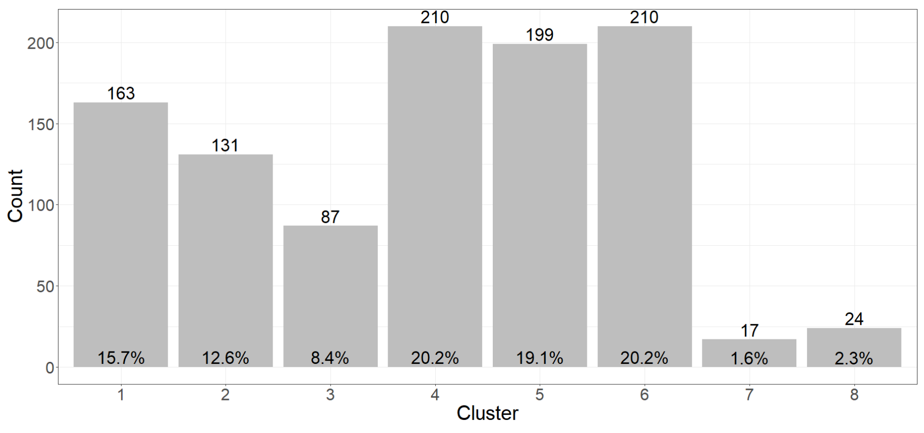

Figure 5.25. Histogram for run 3.

Table 5.13 shows the confusion matrix for run 3 where the overall accuracy was 89.53%. Here all 17 consumers in cluster 7 are correctly classified, while in cluster 3, out of 87 consumers 72 (82.8%) were correctly classified while 15 (17.2%) were classified in cluster 4. This fact might indicate some resemblance between consumers in clusters 3 and 4, but only 2 consumers (1%) from Cluster 4 were classified to cluster 3. Hence the resemblance is not so evident.

Table 5.13. Confusion matrix for run 3. Accuracy 89.53%.

	1	2	3	4	5	6	7	8
1	144 (88.3%)	5 (3.8%)	0 (0%)	3 (1.4%)	6 (3%)	1 (0.5%)	0 (0%)	0 (0%)
2	12 (7.4%)	114 (87%)	0 (0%)	1 (0.5%)	6 (3%)	0 (0%)	0 (0%)	1 (4.2%)
3	0 (0%)	4 (3.1%)	72 (82.8%)	2 (1%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
4	5 (3.1%)	3 (2.3%)	15 (17.2%)	180 (85.7%)	1 (0.5%)	6 (2.9%)	0 (0%)	0 (0%)
5	2 (1.2%)	0 (0%)	0 (0%)	0 (0%)	185 (93%)	6 (2.9%)	0 (0%)	0 (0%)
6	0 (0%)	1 (0.8%)	0 (0%)	24 (11.4%)	1 (0.5%)	197 (93.8%)	0 (0%)	0 (0%)
7	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	17 (100%)	0 (0%)
8	0 (0%)	4 (3.1%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	23 (95.8%)
Total	163	131	87	210	199	210	17	24

5.8.1 Summary of metrics for the 3 runs

Table 5.14 shows the accuracy obtained for each run which is calculated as in Equation 5.8. The overall accuracy for the method is obtained by taking the mean for the 3 runs, i.e. 0.9199 or 91.99%.

Table 5.14. Accuracy for the 3 runs.

	Accuracy
Run 1	0.9376
Run 2	0.9270
Run 3	0.8953

Table 5.15 shows other performance metrics for the classification obtained in the three runs. Sensitivity, which is also referred to as recall, may be defined as the number of correctly predicted observations divided by the actual number of observations in the cluster, that is:

$$sensitivity = \frac{number\ of\ correctly\ predicted\ observations}{number\ of\ actual\ observations\ in\ the\ cluster} \qquad \text{Equation 5.9}$$

Precision may be defined as the number of correctly predicted observations divided by the total predicted in the class, given by:

$$precision = \frac{number\ of\ correctly\ predicted\ observations}{number\ of\ predicted\ observations\ in\ the\ class} \qquad \text{Equation 5.10}$$

In the above equations of *sensitivity* and *precision*, *cluster* and *class* are used in the sense that for the classification, done by KNN, the *cluster* is the actual label for the training observation while the *class* is the prediction obtained from KNN.

F1 score is defined as twice the multiplication of sensitivity and precision divided by the sum of sensitivity and precision, that is:

$$F1_{score} = 2 * \frac{sensitivity * precision}{sensitivity + precision} \qquad \text{Equation 5.11}$$

Table 5.15. Metrics for individual clusters

		Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8
Run 1	Sensitivity	0.9444	0.8953	0.9885	0.9503	0.9746	0.9254	0.8732	1.0000
	Precision	0.9043	0.9293	0.9451	0.9273	0.9458	0.9591	0.9118	1.0000
	F1 score	0.9239	0.9120	0.9663	0.9387	0.9600	0.9420	0.8921	1.0000
Run 2	Sensitivity	0.8819	0.8471	1.0000	0.9286	0.9293	0.9699	0.9540	0.9331
	Precision	0.9203	0.8471	1.0000	0.8848	0.9583	0.9641	0.8925	0.9595
	F1 score	0.9007	0.8471	1.0000	0.9062	0.9436	0.9670	0.9222	0.9461
Run 3	Sensitivity	0.8834	0.8702	0.8276	0.8571	0.9296	0.9381	1.0000	0.9583
	Precision	0.9057	0.8507	0.9231	0.8571	0.9585	0.8834	1.0000	0.8519
	F1 score	0.8944	0.8604	0.8727	0.8571	0.9439	0.9099	1.0000	0.9020

Observing the metrics in Table 5.14 and Table 5.15 it can be noted that level of classification obtained is good, even if the dataset is rather complicated and may experience some seasonality effects. The good results obtained also verifies that KNN may be used in conjunction with hierarchical clustering since they are both based on distances between observations, rather than distances to centroids.

5.9 Anomaly Detection Methods

In this work two novel methods for anomaly detection are being proposed: the Anomaly Coefficient (AC) and the Cluster-Change Coefficient (CCC). For the anomaly coefficient method, each consumer is given a score based on its position inside its cluster. The greater the distance from all the other consumers in the cluster the more dissimilar he is from the majority of consumers in the cluster and so the higher the anomaly coefficient. For the cluster-change coefficient method, for every week each change between two clusters is given a score according to how many changes were performed between the two clusters on that particular week. Consumers who change to a cluster not common for other consumers receive a higher anomaly score. For every consumer the scores for all the weeks are then added and normalised to obtain the overall cluster-change coefficient. In Section 5.8 the clustering method was validated on 15 weeks of data, but for anomaly detection all the available 17 weeks of data were utilised.

5.9.1 Distance from Neighbours – Anomaly Coefficient

This method is based on the argument that the smaller the sum of the distances of a consumer from all the other consumers that form part of the same cluster, then the higher the similarity of his behaviour to the other consumers in the same cluster. Conversely, the greater the sum of the distances of the consumer from all other consumers in the same cluster, the more likely that his behaviour is somewhat different from that of the other consumers, although not enough to be assigned to another cluster.

The first step was to find a way to cluster the consumers into their *typical cluster*. For each consumer the *typical cluster* is the cluster where he would be classified based on the behaviour of the 17 weeks in consideration. The *typical cluster* can be determined by finding the mean for the features over the 17 weeks, perform PCA on these means and cluster on the PCs. However, it was opted that it would be better to do the clustering based on a *typical week*. The *typical week* can be thought of as the week in which the behaviour of the consumer was closest to the average behaviour that he had throughout the 17 weeks being considered. The argumentation in favour of finding the *typical week*, rather than using the average for the 17 weeks, is that the consumer might have never assumed the behaviour described by the mean. If the consumer's behaviour fluctuated between two extremes, the mean will be in the middle, but the consumer never exhibited that behaviour. The *typical week* was determined by finding the mean for all features for the full 17 weeks and then finding the week which has the smallest Euclidian distance from this mean. The features for the *typical week* are then used to cluster the consumers into 8 clusters. The actual clustering was performed by first using PCA to extract the PCs for the features of the *typical week*, then using hierarchical clustering on the first 7 PCs.

For finding the distance between the observations in a dataset the *R* function *dist()* is used. The argument passed to *dist()* is an [n x p] matrix containing n observations of p variables (PC scores in this case). The *dist()* function returns an [n x n] matrix where each row contains the distances of one observation to all other observations in the dataset, including itself. The diagonal elements of this matrix are zero since they are the distance of the observation from itself. So, the sum of each row corresponds to the sum of the distances of the observation from all other observations in the dataset.

For this application the *dist()* function was called 8 times, once for each cluster, each time passing a matrix with the considered PC scores of the consumers in the cluster as an argument. The distance measuring method can also be passed as an argument, with Euclidian being the default and the method selected in this case. The sum of the distances was then normalised for each cluster to obtain an anomaly coefficient. Consumers with an anomaly coefficient closer to one have a behaviour which is different from consumers in their respective cluster. Consumers with an anomaly coefficient close to zero have a behaviour which is similar to the other users in their cluster. Consumers with high anomaly coefficient are more likely to exhibit NTL.

This method of finding the similarity of consumers to the other consumers in the same cluster by finding the sum of their distances from each other was further verified by finding the distance of each consumer from the centroid of the cluster. The centroids of all clusters were calculated by finding the mean for all the features in each cluster. Then the Euclidian distance for each consumer from the centroid of its cluster was calculated. This distance was then normalised for every cluster and compared with the anomaly coefficient already obtained from the *dist()* method. In fact, the two methods were found to complement each other in the sense that consumers which are close to the centroid have an anomaly coefficient close to zero, while those consumers which are furthest from the centroid have an anomaly coefficient close to one.

Figure 5.26 shows the plots for the distribution of the anomaly coefficient in all 8 clusters. The horizontal axis shows the anomaly coefficient with the percentage distribution on the vertical axis. As expected, the majority of observations are concentrated at the low values of anomaly coefficient with few observations having high values. For example, in cluster 6 there are only 3 observations with anomaly coefficient exceeding 0.7 and in cluster 3 only two observations exceed 0.7.

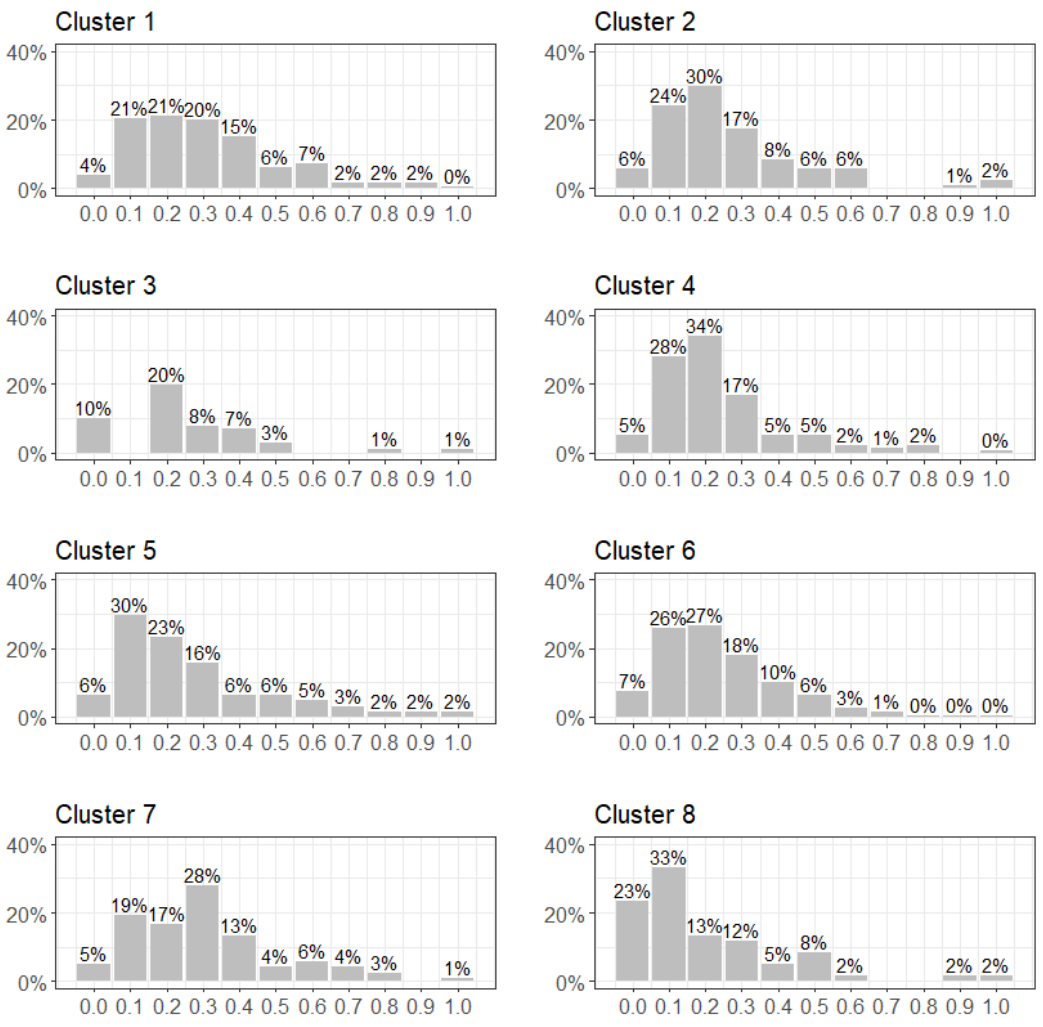

Figure 5.26. Distribution of anomaly coefficient, horizontal axis, for all 8 clusters.

The load profiles for the consumers with the lower and highest anomaly coefficients in each cluster were plotted to examine the contrast in behaviours. Figure 5.27 shows five weeks of load profiles for four consumers in cluster 1, two consumers, m1 (red) and m2 (blue), which have very low anomaly coefficients, and another two load profiles for another two consumers, m3 (green) and m4 (black), with anomaly coefficients close to 1. It should here be reminded that the clustering is based on the typical week for each consumer, which is not necessarily the same for all the consumers in the plots. The minor breaks on the horizontal axis correspond to midnight of each day. It can be seen that the behaviour of m1 and m2 are fairly similar with less consumption during weekends. m3 has almost similar behaviour as m1 and m2 but consumption at night and on weekends is more fluctuating. The night and weekend consumption for m4 is less than that for the other 3 consumers and the general behaviour is irregular. m4 might be a candidate as an NTL suspect.

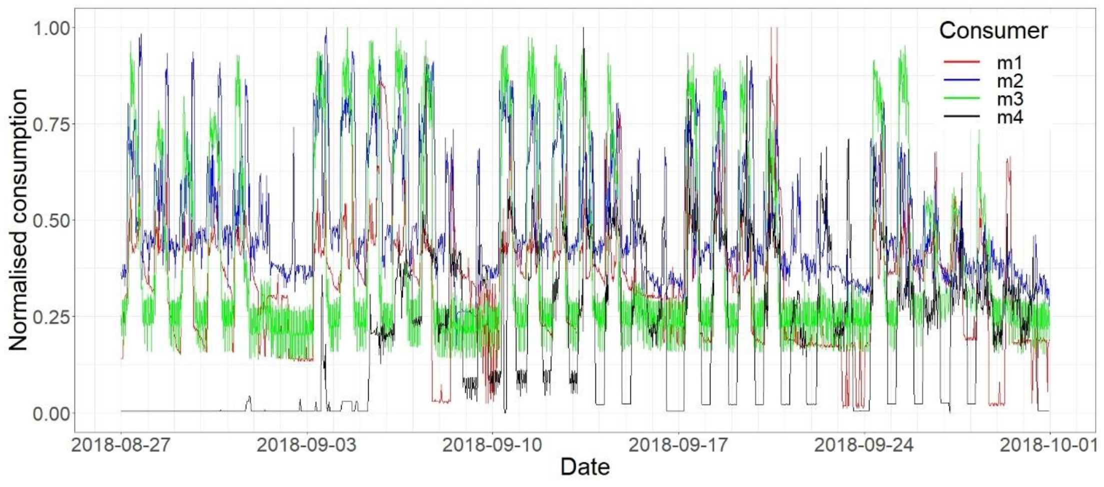

Figure 5.27. Comparing four meters from cluster 1.

Figure 5.28 shows load profiles for consumers in cluster 2, for m1 (red) and m2 (blue) the anomaly coefficient is close to zero , while for m3 (green) and m4 (black) it is close to one. In this case the behaviour of m1 is similar to m2 while m3 and m4 differ slightly. However, this difference does not amount to suspicion of NTL, since the differences are only in the time of usage and the magnitude during the nights and on weekends.

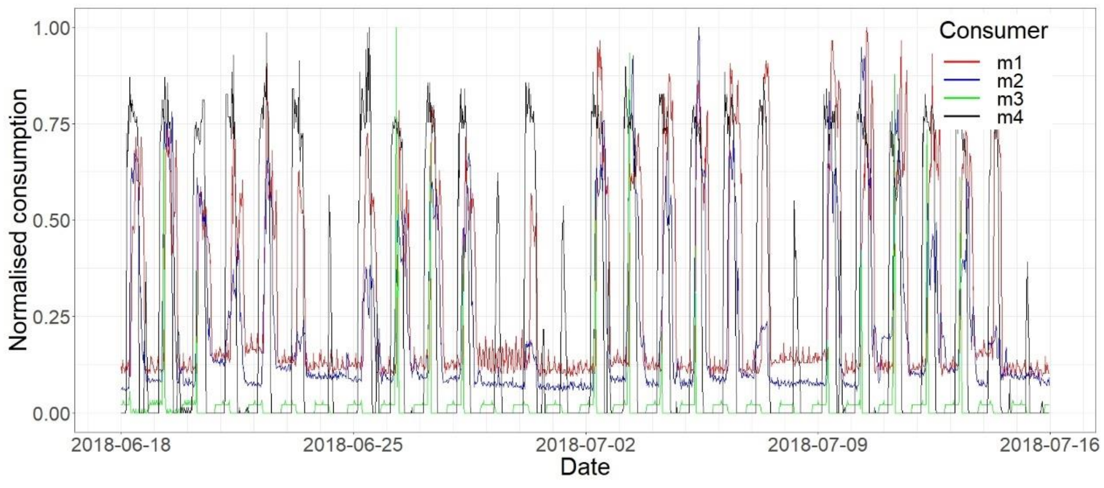

Figure 5.28. Comparing four meters from cluster 2.

Figure 5.29 shows the load profiles for four weeks of four consumers in cluster 3. Consumers m1 and m2 (red and blue) are those with a low anomaly coefficient while m3 and m4 (green and black) have a high value. Here the difference is only in the magnitude contour of the consumption, m1 and m2 exhibit a consumption with a number of peaks while for m3 and m4 the consumption is smoother and mostly at night. But m3 and m4 are not fraud suspects.

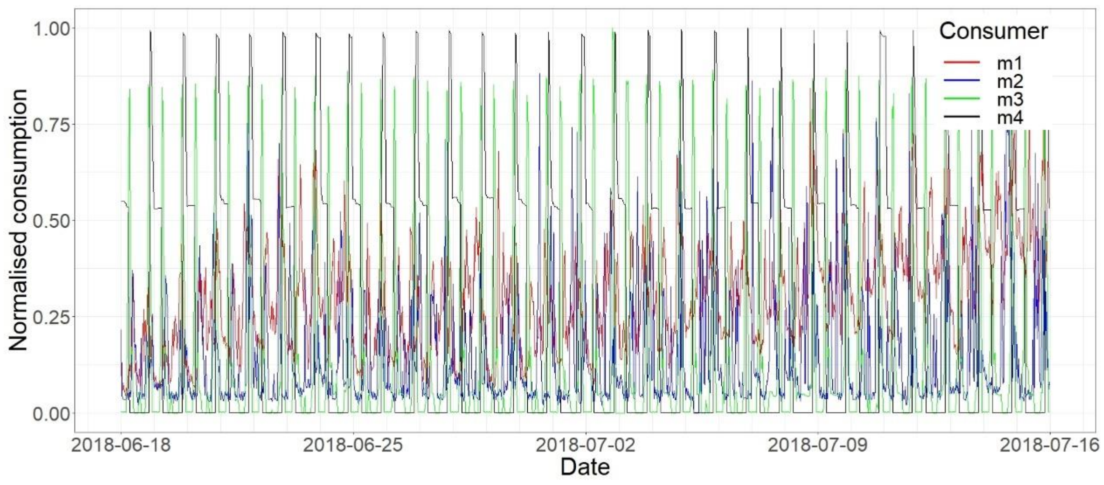

Figure 5.29. Comparing four meters from cluster 3.

The load profiles in Figure 5.30 are for four consumers from cluster 4, where m1 and m2 (red and blue) have exhibited low anomaly coefficients, while for m3 and m4 (green and black) the anomaly coefficient is high. Here four weeks are shown and there is not a considerable difference which is noted for the four consumers' behaviour.

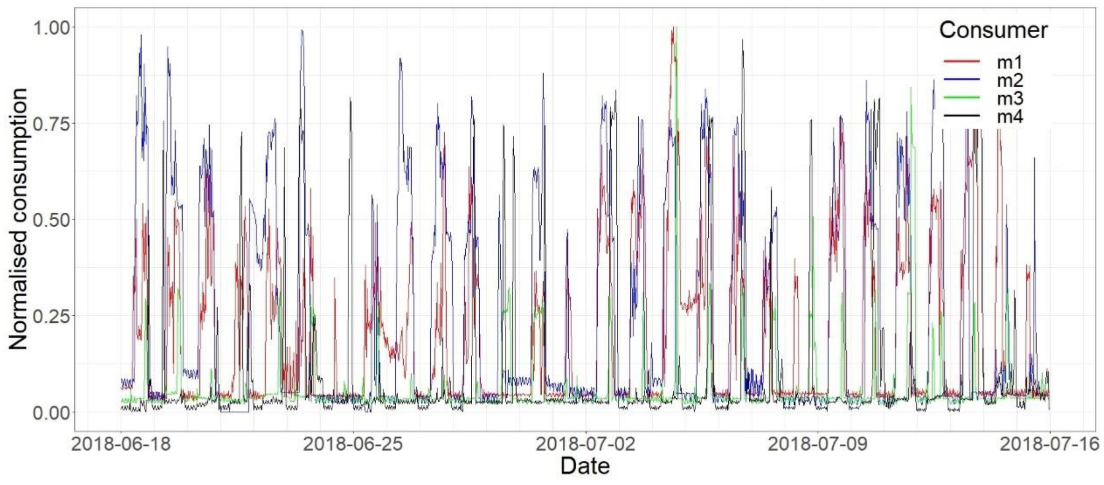

Figure 5.30. Comparing four meters from cluster 4.

Figure 5.31 shows the load profiles for four consumers form cluster 5, m1 (red) and m2 (blue) with low anomaly coefficient and m3 (green) and m4 (black) with high anomaly coefficient. From the plot it can be determined that the differences lie in magnitude of consumption during the night hours and in the fact that m3 and m4 generally do not have consumptions on Sundays, while m1 and m2 have sustained consumption all through the week. So, it is understood that the majority consumers in cluster 5 have regular consumption on all days of the week with a minority of consumers which have marginally less consumption on weekends. But neither m3 nor m4 seem to have any irregular behaviour.

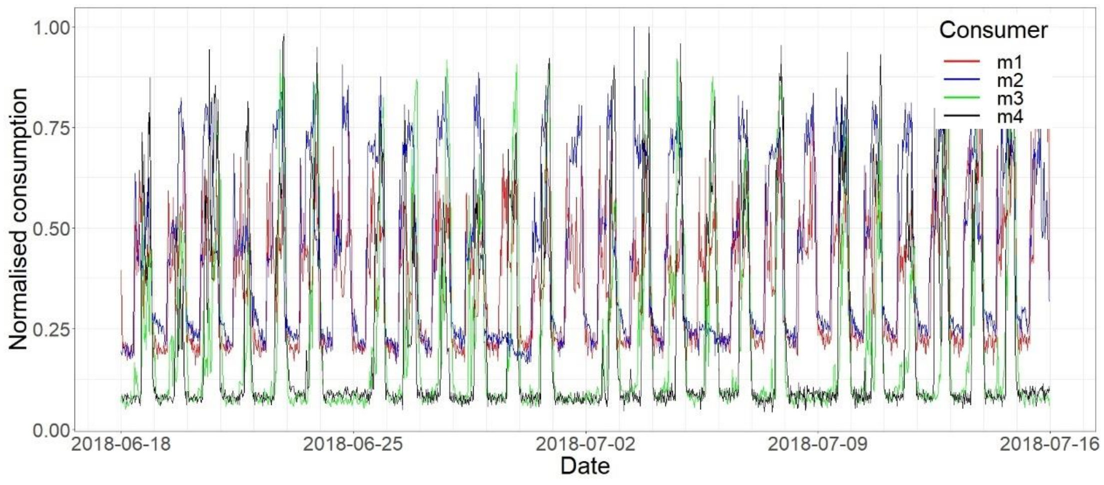

Figure 5.31. Comparing four meters from cluster 5.

Four weeks of load profile for four consumers from cluster 6 are shown in Figure 5.32. m1 (red) and m2 (blue) have low anomaly coefficient while m3 (green) and m4 (black) have high anomaly coefficient. m3 and m4 have a lower consumption during the night than m1 and m2. For two weeks, around 09/07/2018, m4 has a behaviour which might look suspicions and might be a candidate for inspection.

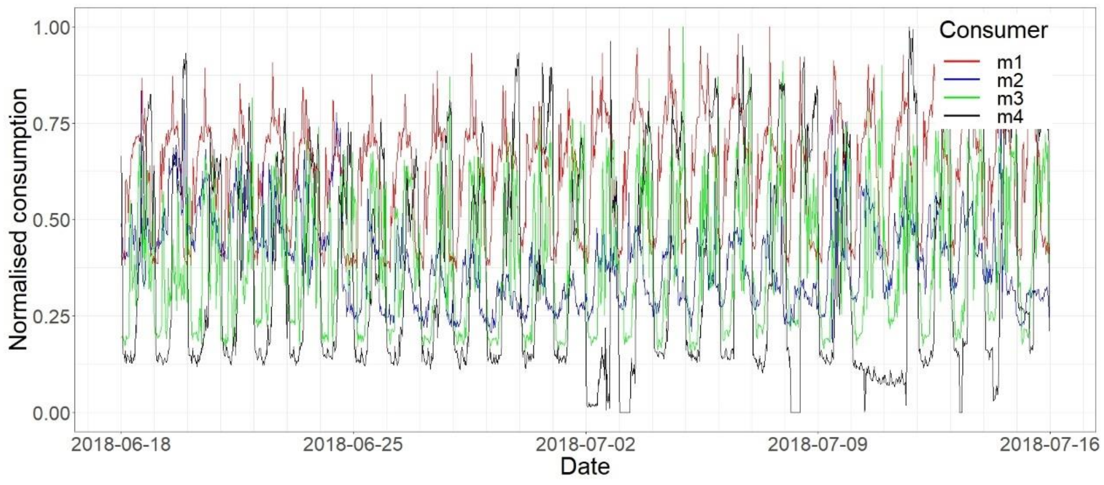

Figure 5.32. Comparing four meters from cluster 6.

Figure 5.33 shows the load profiles of four consumers during four weeks. m1 (red) and m2 (blue) have low anomaly coefficient while m3 (green) and m4 (black) have high anomaly coefficient. It can be observed that all these consumers have a low consumption on weekends. On the second week there was a public holiday, on 29/06/2018, which is distinguishable by the low consumption from all the four consumers. But m3 and m4 do not seem to have any irregular behaviour.

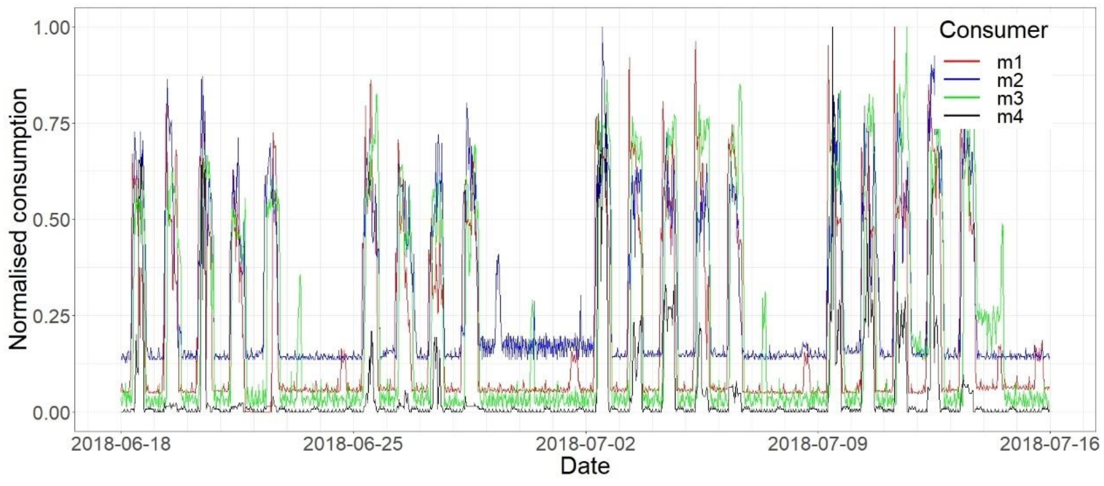

Figure 5.33. Comparing four meters from cluster 7.

The load profiles shown in Figure 5.34 are for four weeks of four consumers in cluster 8. m1 (red) and m2 (blue) have low anomaly coefficient while m3 (green) and m4 (black) have high anomaly coefficient. The profiles for m3 and m4 are very similar with the minimum consumption being very much lower than that for m1 and m2. Another dissimilarity is that m1 and m2 have their consumption concentrated during the day while for m3 and especially for m4 the consumption is almost entirely during the night. But, from looking that their load profiles, neither m3 nor m4 show any suspicious behaviour.

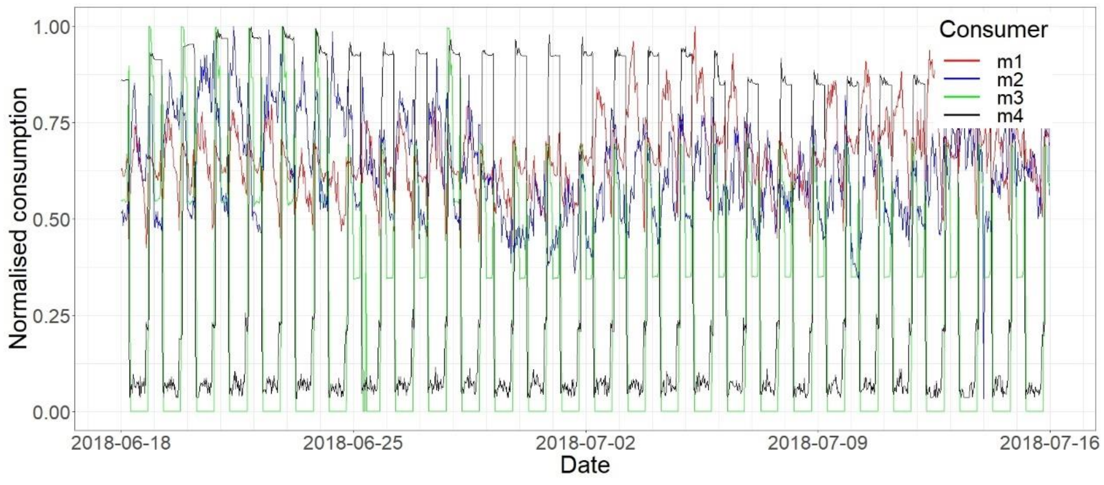

Figure 5.34. Comparing four meters from cluster 8.

By visually analysing the load profiles for consumers with high and low anomaly coefficients, consumers with suspicious irregular behaviour can be identified. However, it cannot be taken for granted that meters with high anomaly coefficients are suspects of NTL. As was observed in the above plots the behaviour of consumers with high anomaly coefficients is always different from those consumers with low anomaly coefficients, but it is not necessarily fraudulent.

5.9.2 Cluster-Change Coefficient

The objective behind this method is to use all 17 weeks of data available in the dataset and classify the consumers for each week, without doing any averaging, in order to identify which consumers change clusters most often. It is expected that consumers that stay in the same cluster for a considerable number of weeks are more regular in their consumption profile than those who change cluster often. Moreover, the cluster changes which were performed by very few consumers are more suspicious than others which are performed by a substantial number of consumers. For example, if on a particular week there was a public holiday, it is reasonable that quite a number of consumers will move to another cluster for that week, and back again the following week. This may also be the case for other external occurrences which impact on the whole, or large part, of the population like, for example, weather conditions.

The *typical cluster* for each consumer was first determined as already mentioned in section 5.9.1. Then, for each of the 17 weeks PCA was used to extract the PCs from the features and the first 7 PCs were used for classifying the consumers by using KNN – using the same procedure as in sections 5.8 and 5.9.1. The cluster numbers obtained for the typical week were used as labels for the KNN classification.

For each week, each cluster change was given a score based on the number of such cluster change that were done during that week divided by the number of consumers in the original (or source) cluster. So, if the change from cluster 1 to cluster 2 is being considered for week 3, the score will be the number of consumers who change cluster from cluster 1 to cluster 2 between week 3 and week 4, divided by the number of consumers in cluster 1 in week 3.

$$score_{12} = \frac{number\ of\ consumers\ which\ changed\ from\ 1\ to\ 2}{number\ of\ consumers\ in\ cluster\ 1}$$ Equation 5.12

Hence the score will be closer to 0 if few consumers changed clusters from cluster 1 to cluster 2, and closer to 1 if many consumers changed from cluster 1 to cluster 2.

The methodology is demonstrated for week 11 and week 12 in Table 5.16 and Table 5.17. Table 5.16 shows the number of consumers in each cluster for week 11 and week 12.

Table 5.16. Count of consumers in each cluster for weeks 11 and 12

Cluster	Week 11	Week 12
1	297	369
2	116	121
3	83	106
4	267	254
5	37	48
6	211	130
7	111	78
8	22	38

Table 5.17 shows the scores for the cluster changes for week 11 and week 12. For example between week 11 and week 12 there was only one consumer who moved from cluster 2 to cluster 3 ($2 \rightarrow 3$) and he will get a low score for that. The number of consumers who changed cluster from 2 to 3 was one. The number of consumers in cluster 2 for week 11 is 116. So, the score for a consumer changing cluster from 2 to 3 between weeks 11 and 12 is 1/116 = 0.0086.

As another example, between week 12 and week 13 there were 131 consumers which moved from cluster 1 to cluster 6 ($1 \rightarrow 6$). Calculating the score, the number of consumers who changed cluster from 1 to 6 was 131. The number of consumers in cluster 1 for week 12 was 369. So, the score for a consumer changing cluster from 1 to 6 between weeks 12 and 13 is 131/369 = 0.355.

Table 5.17. Scores for cluster changes weeks 11 and 12

Week 11						Week 12					
Change	Count	Score	Change	Count	Score	Change	Count	Score	Change	Count	Score
1 -> 1	150	0.5051	5 -> 1	11	0.2973	1 -> 1	125	0.3388	5 -> 1	2	0.0417
1 -> 2	20	0.0673	5 -> 2	0	0.0000	1 -> 2	7	0.0190	5 -> 2	0	0.0000
1 -> 3	19	0.0640	5 -> 3	12	0.3243	1 -> 3	37	0.1003	5 -> 3	5	0.1042
1 -> 4	21	0.0707	5 -> 4	7	0.1892	1 -> 4	24	0.0650	5 -> 4	15	0.3125
1 -> 5	6	0.0202	5 -> 5	7	0.1892	1 -> 5	6	0.0163	5 -> 5	21	0.4375
1 -> 6	46	0.1549	5 -> 6	0	0.0000	1 -> 6	131	0.3550	5 -> 6	2	0.0417
1 -> 7	23	0.0774	5 -> 7	0	0.0000	1 -> 7	30	0.0813	5 -> 7	1	0.0208
1 -> 8	12	0.0404	5 -> 8	0	0.0000	1 -> 8	9	0.0244	5 -> 8	2	0.0417
2 -> 1	14	0.1207	6 -> 1	118	0.5592	2 -> 1	4	0.0331	6 -> 1	38	0.2923
2 -> 2	38	0.3276	6 -> 2	9	0.0427	2 -> 2	33	0.2727	6 -> 2	3	0.0231
2 -> 3	1	0.0086	6 -> 3	3	0.0142	2 -> 3	0	0.0000	6 -> 3	6	0.0462
2 -> 4	45	0.3879	6 -> 4	8	0.0379	2 -> 4	34	0.2810	6 -> 4	13	0.1000
2 -> 5	1	0.0086	6 -> 5	2	0.0095	2 -> 5	0	0.0000	6 -> 5	0	0.0000
2 -> 6	7	0.0603	6 -> 6	46	0.2180	2 -> 6	2	0.0165	6 -> 6	44	0.3385
2 -> 7	10	0.0862	6 -> 7	3	0.0142	2 -> 7	45	0.3719	6 -> 7	5	0.0385
2 -> 8	0	0.0000	6 -> 8	22	0.1043	2 -> 8	3	0.0248	6 -> 8	21	0.1615
3 -> 1	22	0.2651	7 -> 1	22	0.1982	3 -> 1	24	0.2264	7 -> 1	14	0.1795
3 -> 2	0	0.0000	7 -> 2	35	0.3153	3 -> 2	0	0.0000	7 -> 2	12	0.1538
3 -> 3	46	0.5542	7 -> 3	1	0.0090	3 -> 3	57	0.5377	7 -> 3	2	0.0256
3 -> 4	7	0.0843	7 -> 4	24	0.2162	3 -> 4	5	0.0472	7 -> 4	11	0.1410
3 -> 5	4	0.0482	7 -> 5	0	0.0000	3 -> 5	11	0.1038	7 -> 5	0	0.0000
3 -> 6	2	0.0241	7 -> 6	2	0.0180	3 -> 6	9	0.0849	7 -> 6	0	0.0000
3 -> 7	1	0.0120	7 -> 7	27	0.2432	3 -> 7	0	0.0000	7 -> 7	38	0.4872
3 -> 8	1	0.0120	7 -> 8	0	0.0000	3 -> 8	0	0.0000	7 -> 8	1	0.0128
4 -> 1	31	0.1161	8 -> 1	1	0.0455	4 -> 1	8	0.0315	8 -> 1	10	0.2632
4 -> 2	18	0.0674	8 -> 2	1	0.0455	4 -> 2	37	0.1457	8 -> 2	0	0.0000
4 -> 3	24	0.0899	8 -> 3	0	0.0000	4 -> 3	6	0.0236	8 -> 3	0	0.0000
4 -> 4	139	0.5206	8 -> 4	3	0.1364	4 -> 4	169	0.6654	8 -> 4	0	0.0000
4 -> 5	28	0.1049	8 -> 5	0	0.0000	4 -> 5	14	0.0551	8 -> 5	0	0.0000
4 -> 6	18	0.0674	8 -> 6	9	0.4091	4 -> 6	7	0.0276	8 -> 6	20	0.5263
4 -> 7	9	0.0337	8 -> 7	5	0.2273	4 -> 7	12	0.0472	8 -> 7	0	0.0000
4 -> 8	0	0.0000	8 -> 8	3	0.1364	4 -> 8	1	0.0039	8 -> 8	8	0.2105

A high score indicates normal behaviour while a low score indicates anomalous behaviour. Every consumer for every week, except the last week, was given this score based on the cluster changes they performed. So, each consumer ended up with 16 scores. To combine the 16 scores into one for each consumer the scores were added.

Adding was preferred over multiplication where if there is even one score close to zero, the overall score will also be close to zero, even if all the other 15 scores are close to 1. Then the overall score for all users was normalised so that the users can be compared in a range from zero to one. In order to allow its use in conjunction with the anomaly coefficient derived in section 5.9.1, the overall score was subtracted from one to obtain the cluster-change coefficient:

$$cluster_{change}coefficient = 1 - overall\ score \qquad \text{Equation 5.13}$$

So, consumers with cluster-change coefficient which is close to one are more likely to exhibit NTL.

Figure 5.35 shows eight weeks of load profiles of four consumers which have different cluster change coefficients for comparison. m1 (red) and m2 (blue) have very low cluster change coefficients which means that they are very consistent throughout the 17 weeks. On the contrary, m3 (green) and m4 (black) have cluster change coefficients which are close to one, hence showing erratic, or at least inconsistent, behaviour. m3 has a very unsteady consumption with spikes once or twice a week having very different peaks. The consumption behaviour of m4 is also very irregular, there are two weeks in June and one week in July with very low consumption relative to rest. While it is not a certainty that these two consumers have suspicious behaviours, it is surely advisable that their meters are checked.

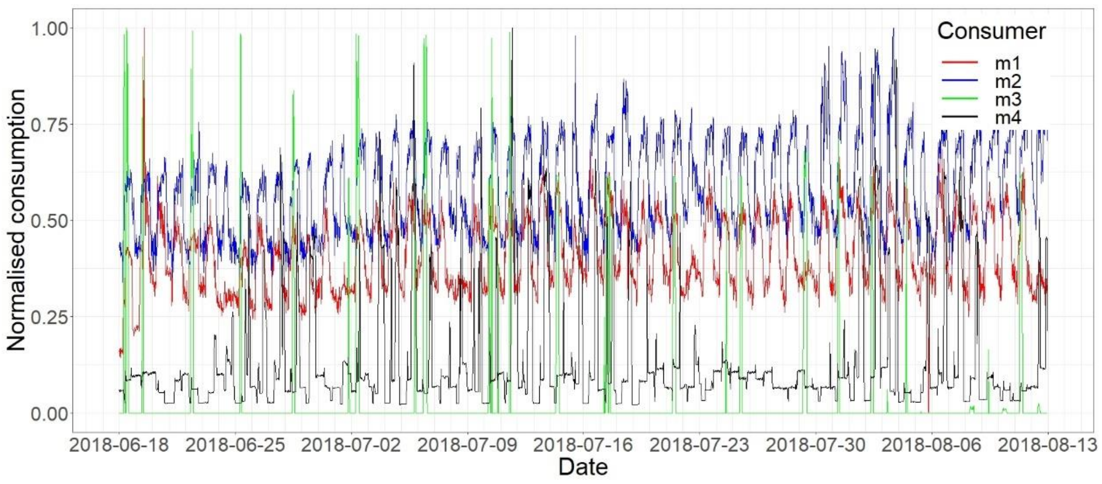

Figure 5.35. Load profiles for consumers with different cluster-change coefficients.

5.10 Minor Modifications and Future Enhancements

An enhancement to this work would be the selection of the features. The selected features can make considerable difference in the results and so have to be chosen very carefully. So, it would be an improvement if the feature selection can be fine-tuned depending on the results.

In future work the use of sequential hierarchical clustering [117] can be explored for the determination of the cluster change coefficient, whereby the clustering for each week will continue on the clusters obtained for the previous week. The intention is that conventional hierarchical clustering will be done for the first week and then sequential hierarchical clustering will be performed for the second and successive weeks. Consumers can then be allocated in the same, other or even new clusters. New consumers should also be processed and placed in their appropriate cluster. Using this approach there will not be the need to find the typical week for each consumer since the grouping will be done entirely by hierarchical clustering without any need for the KNN classification.

The results that will be obtained from inspections suggested by the coefficients determined in this work should be recorded in detail so that in future a supervised approach can be explored. The two methods, supervised and unsupervised, can be combined together to make the NTL detection more efficient.

5.11 Conclusion

Anomaly detection is nontrivial, and detection of non-technical losses is quite a challenge, mainly because of the unbalance in the numbers between genuine and fraudulent consumers and, in the case of local data, the lack of labelled information.

In this work a set of features were derived to describe the weekly behaviour of the consumers while serving as a means of dimension reduction. PCA was used as a method for further dimension reduction. Both unsupervised and supervised learning machines were utilised to classify behaviours by using hierarchical clustering for training and KNN for testing. The model was validated by using 3-fold cross validation which gave an overall accuracy of 91.99%. Finally, two coefficients were derived which indicate the possibility that a consumer is liable to contribute to non-technical losses. The anomaly coefficient gives a measure of how much the behaviour of the consumer is similar to others which generally have similar features. The cluster-change coefficient gives a measure of how well behaved the consumer is over a number of weeks by keeping track of the cluster changes done with respect to the cluster changes of the majority of consumers. The anomaly coefficient is more conservative than the cluster change coefficient since it is based on the behaviour of a typical week, while the cluster-change coefficient is based on the behaviour of each week. It is recommended to combine these two coefficients to give an Anomaly Score which can be used to list the consumers which are more likely to exhibit NTL.

6 Conclusion

Due to the fact that not all the data generated by smart meters is read successfully every day and thus a portion of the data gets lost, it is important for Enemalta to have a data imputation strategy in place. In the presence of missing data, statistical and analytical processes cannot be conducted effectively and reliably. Hence the first objective of this work was to develop a method which fills in the missing load profiles for commercial consumers. First a study was conducted to determine the extent of missing data and from this study, it was decided to use data from 2016 to 2019 as a test dataset for imputation. The developed imputation method uses a KNN based algorithm which searches in the past daily consumption of the consumer for behaviour similar to that during the period right before the data went missing. The *k* best matching profiles are averaged to get the estimated daily consumption for the missing portion. Then the half hourly load profiles for the same days which were in the best matching profiles are averaged to get the values for the imputed half hourly load profiles. Finally, the imputed load profiles are scaled for their sum to equal the difference between any available spot readings. This factoring restores the correct magnitude to the imputed load profiles since the matching is done on normalised values to match the behaviour. The factoring is also important so that the relationship between spot readings and load profiles is conserved. The average RMSE obtained for a sample of 335 load profiles was 7.47%.

Having a complete load profile, where any missing data has been imputed, paves the way for the second objective of this thesis, that is developing a method for detecting anomalous consumption behaviour. Anomalies might be caused by faulty or tampered metering setups which both contribute to non-technical, i.e. financial losses for Enemalta. Commercial consumers have much higher consumption than residential and domestic consumers and so it makes more economical sense to concentrate on them. A number of consumers with full load profiles for a period between 1st June 2018 and 30th September 2018 (17 weeks) were selected. The period was selected to span over the summer season so that the behaviour of the consumers is as regular as possible. These consumers were split into 8 clusters, by using hierarchical clustering, based on their behaviour. A two-step dimension reduction process was performed before clustering to reduce the amount of data processed by the clustering algorithm. Feature extraction was the first step in dimension reduction where 32 features were selected which well explain the weekly consumption behaviour of clients while reducing the redundancy in the data. The second step of dimension reduction was performed through Principal Component Analysis. The clustering process was performed on the first 7 principal components which were shown to be the major contributors for the total variance.

The clustering model was tested by using 3-fold validation. The load profile data for 15 weeks was split into 3 5-week folds which were then used for training and testing. The data in one fold was used for training while the data in the other two folds was used for testing. This was repeated 3 times so that each week of data was utilised both for training and for testing. In order to reduce the effect of seasonality between training and testing, the weeks making up each fold were selected to be alternate rather than adjacent so that the training data was spread over the whole 15 weeks, as was the testing data. For testing, classification was performed using KNN based on the clusters obtain through hierarchical clustering. The overall accuracy based the 3-fold verification process was 91.99%.

For each consumer an anomaly coefficient was then derived by calculating the sum of the distances from all the other consumers in the same cluster. The larger the sum of the distances, the more dissimilarity a particular consumer is from the general behaviour of that cluster. These distances where then normalised to obtain the anomaly coefficient by which all users in all clusters can be compared. Consumers with high anomaly coefficient are more likely to exhibit NTL.

Another method to assess the regularity of a consumer is to monitor the cluster changes during the observation period of 17 weeks. The number of times a consumer changes from one cluster to the other is directly related to the regularity of his behaviour. A consumer who changes clusters every week is more likely to exhibit NTL than a consumer who stayed in the same cluster for a considerable number of weeks. Moreover, the cluster changes which were performed by very few consumers are more suspicious than others which were performed by a substantial number of consumers. Based on these changes an algorithm was developed which calculates the cluster-change coefficient for each consumer. Consumers with cluster-change coefficient closer to 1 have unstable consumption which might indicate NTL.

The two coefficients can be combined by addition to give an Anomaly Score which can be used to list the consumers which are most likely to have their metering setup faulty or tampered.

In conclusion this research presented an algorithm capable of imputing missing values with an average RMSE of 7.47% of the actual values. It also presented two novel coefficients which can be used on their own or combined to indicated consumers who are most likely to contribute to non-technical losses.

7 References

1. “Malta's Annual Report 2020” available at https://www.rews.org.mt/#/en/rewsfa/27, accessed 02/04/2021.
2. “NR161/2020”, available at https://nso.gov.mt/en/News_Releases/View_by_Unit/Unit_02/Regional_and_Geospatial_Statistics/Pages/Electricity-Supply.aspx, accessed 02/04/2021.
3. https://www.enemalta.com.mt/about-us/our-network, accessed 02/04/2021.
4. “Annual Report 2019 & Financial Statements” available at https://www.rews.org.mt/#/en/rewsfa/26, accessed 02/04/2021.
5. G. Bucci, F. D’Innocenzo, S. Dolce, E. Fiorucci and F. Ciancetta, “Power line communication, overview of standards and applications,” XXI IMEKO World Congress “Measurement in Research and Industry”, August 30 - September 4 2015, Prague, Czech Republic.
6. N. Pavlidou, A. J. Han Vinck, J. Yazdani and B. Honary, “Power line communications: state of the art and future trends,” IEEE Communications Magazine, vol. 41, no. 4, pp. 34-40, April 2003, DOI: 10.1109/MCOM.2003.1193972.
7. H. C. Ferreira, H. M. Grove, O. Hooijen and A. J. Han Vinck, “Power line communications: an overview,” Proceedings of IEEE. AFRICON '96, Stellenbosch, South Africa, 1996, pp. 558-563 vol.2, DOI: 10.1109/AFRCON.1996.562949.
8. CESI SpA, “Electronic meters theory and practice,” Course material 18th - 22nd October 2010.
9. C. Chatfield, “The analysis of time series,” sixth edition 2003, Chapman & Wall/CRC, ISBN: 1-58488-317-0.
10. I. E. Oran Brigham, “The fast Fourier transform and its applications,” 1988, Prentice-Hall, ISBN: 0-13-307505-2.
11. P. J. Brockwell and R. A. Davis, “Introduction to time series and forecasting,” Second Edition 2002, Springer text in statistics, ISBN: 978-0387-95351-9.
12. C. Torrence and G. P. Compo, “A practical guide to wavelet analysis,” 1998, Bulletin of the American Meteorological Society, 79, pp. 605-618.
13. O. Rioul and M. Vetterli, “Wavelets and signal processing,” IEEE Signal Processing Mag., vol. 8, pp. 14-38, 1991.

14. J. Honaker and G. King (2010), “What to do about missing values in time-series cross-section data,” American Journal of Political Science, vol. 54, No. 2, April 2010, pp. 561-581, ISSN 0092-5853.
15. S. Oh, “Multiple imputation on missing values in time series data,” Masters dissertation, Duke University, 2015.
16. T. Kim, W. Ko and J. Kim, “Analysis and impact evaluation of missing data imputation in day-ahead PV generation forecasting,” Applied Sciences. 2019, 9, 204, DOI:10.3390/app9010204.
17. W. F. Velicer and S. M. Colby, “A comparison of missing-data procedures for ARIMA time-series analysis,” Educational and Psychological Measurement 2005 vol. 65, No. 4, pp. 596-615.
18. S. Moritz, A. Sardá, T. Bartz-Beielstein, M. Zaefferer and J. Stork, “Comparison of different methods for univariate time series imputation in R,” 2015, 1510.03924.
19. S. A. Rahman, Y. Huang, J. Claassen and S. Kleinberg, “Imputation of missing values in time series with lagged correlations,” IEEE International Conference on Data Mining Workshop, Shenzhen, 2014, pp. 753-762, DOI: 10.1109/ICDMW.2014.110.
20. D. B. Rubin, “Inference and missing data,” Biometrika 1976, vol. 63, No. 3, pp. 581-592.
21. R. J. A. Little and D. B. Rubin, “Statistical analysis with missing data,” New York: John Wiley, 1987, ISBN:9780471183860.
22. E.T. Caparino, A.M. Sison and R.P. Medina, “Application of the modified imputation method to missing data to increase classification performance,” IEEE 4th International Conference on Computer and Communication Systems, Singapore 23-25 February 2019, pp. 134-139, DOI: 10.1109/CCOMS.2019.8821632.
23. T. Makaba and E. Dogo, (2019), “A comparison of strategies for missing values in data on machine learning classification algorithms,” International Multidisciplinary Information Technology and Engineering Conference (IMITEC), 21-22 November 2019, South Africa, pp. 1-7, DOI: 10.1109/IMITEC45504.2019.9015889.
24. S. Moritz and T. Bartz-Beielstein, “imputeTS: time series missing value imputation in R,” 2017, The R Journal, DOI: 10.32614/RJ-2017-009

25. O. Anava, E. Hazan and A. Zeevi, “Online time series prediction with missing data,” Proceedings of the 32nd International Conference on Machine Learning, Lille, France 2015. JLMR: W&CP volume 37.

26. D. M. P. Murti, U. Pujianto, A. P. Wibawa and M. I. Akbar, “K-Nearest Neighbor (K-NN) based missing data imputation,” 5th International Conference on Science in Information Technology (ICSITech), Yogyakarta, Indonesia, 23-24 October 2019, pp. 83-88, DOI: 10.1109/ICSITech46713.2019.8987530.

27. W. O. Yodah, J. M. Kihoro, K.H.O. Athiany and H. W. Kibunja, “Imputation of incomplete non-stationary seasonal time series data,” Mathematical Theory and Modeling, vol. 3, No.12, pp. 142–154, 2013, ISSN 2224-5804.

28. Y. Wang, Q. Chen, T. Hong and C. Kang, “Review of smart meter data analytics: applications, methodologies, and challenges,” IEEE Transactions on Smart Grid, vol. 10, no. 3, pp. 3125-3148, May 2019, DOI: 10.1109/TSG.2018.2818167.

29. X. Liu, L. Golab, W. Golab, I. F. Ilyas, and S. Jin, “Smart meter data analytics: systems, algorithms and benchmarking,” A C M Transactions on Database Systems, vol. 42, No. 1, 2016.

30. F. McLoughlin, “Characterising domestic electricity demand for customer load profile segmentation,” PhD Dissertation, School of Civil and Building Services Engineering Dublin Institute of Technology Bolton Street, Dublin 1, Ireland, February 2013.

31. 41G. Chicco, R. Napoli, P. Postolache, M. Scutariu and C. Toader, “Customer characterization options for improving the tariff offer,” IEEE Transactions on Power Systems, vol. 18, no. 1, pp. 381-387, February. 2003, DOI: 10.1109/TPWRS.2002.807085.

32. V. Figueiredo, F. Rodrigues, Z. Vale and J. B. Gouveia, “An electric energy consumer characterization framework based on data mining techniques,” in IEEE Transactions on Power Systems, vol. 20, no. 2, pp. 596-602, May 2005, DOI: 10.1109/TPWRS.2005.846234.

33. I. B. Sanchez, I. D. Espinos, L. M. Sarrion, A. Q. Lopez and I. N. Burgos, “Clients segmentation according to their domestic energy consumption by the use of self-organizing maps,” 6th International Conference on the European Energy Market, Leuven, 2009, pp. 1-6, DOI: 10.1109/EEM.2009.5207172.

34. A. Al-Wakeel, J. Wu and N. Jenkins, “k-means based load estimation of domestic smart meter measurements,” Applied Energy Volume 194, 15 May 2017, pp. 333-342.

35. W. Yu, W. Zhu, G. Liu, B. Kan, T. Zhao and H. Liu, “Cluster-based best match scanning for large-scale missing data imputation,” 2017 3rd International Conference on Big Data Computing and Communications (BIGCOM), Chengdu, 2017, pp. 232-238.

36. J. Peppanen, X. Zhang and S.Grijalva, “Handling bad or missing smart meter data through advanced data imputation,” 2016 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT).

37. S. Ryu, M. Kim and H. Kim, “Denoising autoencoder-based missing value imputation for smart meters,” in IEEE Access, vol. 8, pp. 40656-40666, 2020, DOI: 10.1109/ACCESS.2020.2976500.

38. G. Mateos and G. B. Giannakis, “Spatiotemporal load curve data cleansing and imputation via sparsity and low rank,” 2012 IEEE Third International Conference on Smart Grid Communications (SmartGridComm), Tainan, 2012, pp. 653-658, DOI: 10.1109/SmartGridComm.2012.6486060.

39. G. Mateos and G. B. Giannakis, “Load curve data cleansing and imputation via sparsity and low rank," IEEE Transactions on Smart Grid, vol. 4, no. 4, pp. 2347-2355, December. 2013, DOI: 10.1109/TSG.2013.2259853.

40. M. Kim, S. Park, J. Lee, Y. Joo and J. K. Choi, “Learning-based adaptive imputation method with kNN algorithm for missing power data,” Energies, MDPI, Open Access Journal, vol. 10(10), pp. 1-20, October 2017.

41. N. A. Hidayatullah, B. Stojcevski and A. Kalam, “Analysis of distributed generation systems smart grid technologies and future motivators influencing change in the electricity sector,” Smart Grid and Renewable Energy, 2011, vol. 2, pp. 216-229. DOI:10.4236/sgre.2011.23025.

42. Y. Liang, D. Niu and W.Hong, “Short term load forecasting based on feature extraction and improved general regression neural network model”, Energy (2018), DOI: 10.1016/j.energy.2018.10.119.

43. C. Beckel, L. Sadamoriy and S. Santini, “Automatic socio-economic classification of households using electricity consumption data,” e-Energy’13, May 21–24, 2013, Berkeley, California, USA, ACM 978-1-4503-2052-8/13/05.

44. K. Gajowniczek, T. Zabkowski and M. Sodenkamp, “Revealing household characteristics from electricity meter data with grade analysis and machine learning algorithms,” Appl. Sci. 2018, vol. 8, pp. 1654; DOI:10.3390/app8091654
45. J.G. Dy and C. E. Brodley, “Feature selection for unsupervised learning,” Journal of Machine Learning Research 5, 2004, pp. 845-889.
46. C. Beckel, L. Sadamori and S. Santini, “Towards automatic classification of private households using electricity consumption data,” Buildsys’12, November 6, 2012, Toronto, ON, Canada. ACM 978-1-4503-1170-0.
47. K. Hopf, M. Sodenkamp, I. Kozlovkiy and T. Staake, “Feature extraction and filtering for household classification based on smart electricity meter data,” Comput Sci Res Dev 2016, vol. 31, pp. 141-148, DOI 10.1007/s00450-014-0294-4.
48. J. Palacio-Nino and F. Berzal, “Evaluation metrics for unsupervised learning algorithms,” arXiv:1905.05667v2 [cs.LG], 23 May 2019.
49. T. Gupta and S. P. Panda, “Clustering validation of CLARA and K-means using silhouette & DUNN measures on iris dataset,” 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (Com-IT-Con), India, 14-16 February, 2019.
50. M. Halkidi, Y. Batistakis and M. Vazirgiannis, “On clustering validation techniques," Journal of Intelligent Information Systems, 17:2/3, pp. 107-145, 2001.
51. G. Brock, V. Pihur, S. Datta, and S. Datta, “clValid , an R package for cluster validation,” Department of Bioinformatics and Biostatistics, University of Louisville. Journal of Statistical Software, March 2008
52. V. Chandola, A. Banerjee and V. Kumar, “Anomaly detection: a survey,” Comput. Surv. 41, 3, Article 15, July 2009, 58 pages. DOI: 10.1145/1541880.1541882.
53. V. J. Hodge and J. Austin, “A survey of outlier detection methodologies,” Artificial Intelligence Review 22: 85–126, 2004.
54. T. Pevny, "LODA: Lightweight on-line detector of anomalies,” Mach Learn. DOI 10.1007/s10994-015-5521-0.
55. A. Aligholian, M. Farajollahi and H. Mohsenian-Rad, “Unsupervised learning for online abnormality detection in smart meter data,” 2019 IEEE Power & Energy Society General Meeting (PESGM), Atlanta, GA, USA, 2019, pp. 1-5, DOI: 10.1109/PESGM40551.2019.8973564.

56. R. Jiang, R. Lu, Y. Wang, J. Luo, C. Shen and X. Shen, “Energy-theft detection issues for advanced metering infrastructure in smart grid,” Tsinghua Science and Technology, vol. 19, no. 2, pp. 105-120, April 2014, DOI: 10.1109/TST.2014.6787363.
57. J. L. Viegas, P. R. Esteves, R. Melicioc, V. M. F. Mendes and S. M. Vieira, “Solutions for detection of non-technical losses in the electricity grid: a review,” Article in Renewable and Sustainable Energy Reviews, December 2017, DOI: 10.1016/j.rser.2017.05.193.
58. G. M. Messinis and N. D. Hatziargyriou, “Review of non-technical loss detection methods,” Electric Power Systems Research 158, 2018, pp. 250-266.
59. P. Glauner, J. A. Meira, P. Valtchev, R. State and F. Bettinger, “The challenge of non-technical loss detection using artificial intelligence: a survey,” arXiv:1606.00626v3 [cs.AI], 25 July 2017.
60. K. M. Ghori, R. A. Abbasi, M. Awais, M. Imran, A. Ullah and L. Szathmary, “Performance analysis of different types of machine learning classifiers for non-technical loss detection,” IEEE Access, vol. 8, pp. 16033-16048, 2020, DOI: 10.1109/ACCESS.2019.2962510.
61. C. Papadimitriou, G. Messinis, D. Vranis, S. Politopoulou and N. Hatziargyriou, “Non-technical losses: detection methods and regulatory aspects overview,” CIRED - Open Access Proceedings Journal, vol. 2017, no. 1, pp. 2830-2832, 10 2017, DOI: 10.1049/oap-cired.2017.0825.
62. M. A. Hearst, S. T. Dumais, E. Osman, J. Platt and B. Scholkopf, “Support vector machines,” IEEE Intell. Syst. Appl., vol. 13, no. 4, pp. 1828, Jul./Aug. 2008.
63. R. N. Toma, M. N. Hasan, A. Nahid and B. Li, “Electricity theft detection to reduce non-technical loss using support vector machine in smart grid,” 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), Dhaka, Bangladesh, 2019, pp. 1-6, DOI: 10.1109/ICASERT.2019.8934601.
64. J. Nagi, K. S. Yap, S. K. Tiong, S. K. Ahmed and F. Nagi, “Improving SVM-based nontechnical loss detection in power utility using the fuzzy inference system,” IEEE Transactions on Power Delivery, vol. 26, no. 2, pp. 1284-1285, April 2011, DOI: 10.1109/TPWRD.2010.2055670.

65. J. Nagi, K. S. Yap, S. K. Tiong, S. K. Ahmed and M. Mohamad, "Nontechnical loss detection for metered customers in power utility using support vector machines," IEEE Transactions on Power Delivery, vol. 25, no. 2, pp. 1162-1171, April 2010, DOI: 10.1109/TPWRD.2009.2030890.
66. P. Glauner, J. A. Meira, L. Dolberg, R. State, F. Bettinger and Y. Rangoni, "Neighborhood features help detecting non-technical losses in big data sets," IEEE/ACM 3rd International Conference on Big Data Computing Applications and Technologies (BDCAT), Shanghai, 2016, pp. 253-261.
67. G. Figueroa, Y. Chen, N. Avila and C. Chu, "Improved practices in machine learning algorithms for NTL detection with imbalanced data," 2017 IEEE Power & Energy Society General Meeting, Chicago, IL, 2017, pp. 1-5, DOI: 10.1109/PESGM.2017.8273852.
68. P. Glauner, A. Boechat, L. Dolberg, R. State, F. Bettingert, Y. Rangonit and D. Duartet, "Large-scale detection of non-technical losses in imbalanced data sets," 2016 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), Minneapolis, MN, 2016, pp. 1-5, DOI: 10.1109/ISGT.2016.7781159.
69. G. M. Messinis, A. E. Rigas and N. D. Hatziargyriou, "A hybrid method for non-technical loss detection in smart distribution grids," IEEE Transactions on Smart Grid, vol. 10, no. 6, pp. 6080-6091, November. 2019, DOI: 10.1109/TSG.2019.2896381.
70. A. Jindal, A. Dua, K. Kaur, M. Singh, N. Kumar and S. Mishra, "Decision tree and SVM-based data analytics for theft detection in smart grid," IEEE Transactions on Industrial Informatics, vol. 12, no. 3, pp. 1005-1016, June 2016, DOI: 10.1109/TII.2016.2543145.
71. Irish Social Science Data Archive. [Online]. Available at: http://www.ucd.ie/issda/data/commissionforenergyregulationcer/ accessed on 10/08/2020.
72. A. H. Nizar, Z. Y. Dong and Y. Wang, "Power utility nontechnical loss analysis with extreme learning machine method," IEEE Transactions on Power Systems, vol. 23, no. 3, pp. 946-955, August. 2008, DOI: 10.1109/TPWRS.2008.926431.

73. A. H. Nizar and Z. Y. Dong, “Identification and detection of electricity customer behaviour irregularities,” IEEE/PES Power Systems Conference and Exposition, Seattle, WA, 2009, pp. 1-10, DOI: 10.1109/PSCE.2009.4840253.

74. Y. Yuan and K. Jia, “A distributed anomaly detection method of operation energy consumption using smart meter data,” International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), Adelaide, SA, 2015, pp. 310-313, DOI: 10.1109/IIH-MSP.2015.38.

75. T. Hu, Q. Guo, X. Shen, H. Sun, R. Wu and H. Xi, “Utilizing unlabelled data to detect electricity fraud in AMI: a semisupervised deep learning approach,” in IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 11, pp. 3287-3299, November. 2019, DOI: 10.1109/TNNLS.2018.2890663.

76. H. Huang, S. Liu and K. Davis, “Energy theft detection via artificial neural networks,” 2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), Sarajevo, 2018, pp. 1-6, DOI: 10.1109/ISGTEurope.2018.8571877.

77. S. Yip, W. Tan, C. Tan, M. Gan and K. Wong (2018), “An anomaly detection framework for identifying energy theft and defective meters in smart grids,” International Journal of Electrical Power & Energy Systems, vol. 101, pp. 189-203. 10.1016/j.ijepes.2018.03.025.

78. S. Yip, C. Tan, W. Tan, M. Gan, K. Wong and R. C. Phan, “Detection of energy theft and metering defects in advanced metering infrastructure using analytics,” International Conference on Smart Grid and Clean Energy Technologies (ICSGCE), Kajang, 2018, pp. 15-22, DOI: 10.1109/ICSGCE.2018.8556710.

79. P. Jokar, N. Arianpoo and V. C. M. Leung, “Electricity theft detection in AMI using customers’ consumption patterns,” IEEE Transactions on Smart Grid, vol. 7, no. 1, pp. 216-226, January. 2016, DOI: 10.1109/TSG.2015.2425222.

80. M. Liu, D. Liu, G. Sun, Y. Zhao, D. Wang, F. Liu, X. Fang, Q. He and D. Xu, “Detection of malfunctional smart electricity meters based on deep learning of electricity usage data”, pp. 1-24, 2019.

81. K. Dineshkumar, P. Ramanathan and S. Ramasamy, “Development of ARM processor based electricity theft control system using GSM network,” International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015], Nagercoil, 2015, pp. 1-6, DOI: 10.1109/ICCPCT.2015.7159401.

82. B. Khoo and Y. Cheng, “Using RFID for anti-theft in a Chinese electrical supply company: A cost-benefit analysis,” Wireless Telecommunications Symposium (WTS), New York City, NY, 2011, pp. 1-6, DOI: 10.1109/WTS.2011.5960892.
83. G. M. Messinis and N. D. Hatziargyriou, “Unsupervised classification for non-technical loss detection,” Power Systems Computation Conference (PSCC), Dublin, 2018, pp. 1-7, DOI: 10.23919/PSCC.2018.8442797.
84. BreakoutDetection R package, [Online] Available at: https://github.com/twitter/BreakoutDetection, accessed 11/06/2020
85. R. Punmiya and S. Choe, “Energy theft detection using gradient boosting theft detector with feature engineering-based preprocessing,” IEEE Transactions on Smart Grid, vol. 10, no. 2, pp. 2326-2329, March 2019, DOI: 10.1109/TSG.2019.2892595.
86. A. Saad and N. Sisworahardjo, “Data analytics-based anomaly detection in smart distribution network,” International Conference on High Voltage Engineering and Power Systems (ICHVEPS), Sanur, 2017, pp. 1-5, DOI: 10.1109/ICHVEPS.2017.8225855.
87. M. M. Buzau, J. Tejedor-Aguilera, P. Cruz-Romero and A. Gómez-Expósito, “Detection of non-technical losses using smart meter data and supervised learning," IEEE Transactions on Smart Grid, vol. 10, no. 3, pp. 2661-2670, May 2019, DOI: 10.1109/TSG.2018.2807925.
88. J. E. Cabral, J. O. P. Pinto and A. M. A. C. Pinto, “Fraud detection system for high and low voltage electricity consumers based on data mining,” IEEE Power & Energy Society General Meeting, Calgary, AB, 2009, pp. 1-5, DOI: 10.1109/PES.2009.5275809.
89. K. K. Kee, S. M. F. Shahab and C. J. Loh, “Design and development of an innovative smart metering system with GUI-based NTL detection platform,” 4th IET Clean Energy and Technology Conference (CEAT 2016), Kuala Lumpur, 2016, pp. 1-8, DOI: 10.1049/cp.2016.1293.
90. N. F. Avila, G. Figueroa and C. Chu, “NTL detection in electric distribution systems using the maximal overlap discrete wavelet-packet transform and random undersampling boosting,” IEEE Power & Energy Society General Meeting (PESGM), Atlanta, GA, USA, 2019, pp. 1-1, DOI: 10.1109/PESGM40551.2019.8973473.

91. Q. Zhang, M. Zhang, T. Chen, J. Fan, Z. Yang and G. Li, "Electricity theft detection using generative models," IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI), Volos, 2018, pp. 270-274, DOI: 10.1109/ICTAI.2018.00050.
92. P. Massaferro, J. M. D. Martino and A. Fernández, "Fraud detection in electric power distribution: an approach that maximizes the economic return," IEEE Transactions on Power Systems, vol. 35, no. 1, pp. 703-710, Jan. 2020, DOI: 10.1109/TPWRS.2019.2928276.
93. Y. Tang, C. Ten and K. P. Schneider, "Inference of tampered smart meters with validations from feeder-level power injections," IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia), Chengdu, China, 2019, pp. 2783-2788, DOI: 10.1109/ISGT-Asia.2019.8881675.
94. V. B. Krishna, G. A. Weaver and W. H. Sanders, "PCA-based method for detecting integrity attacks on advanced metering infrastructure," 12th International Conference on Quantitative Evaluation of Systems (QEST 2015), September 2015 DOI: 10.1007/978-3-319-22264-6_5.
95. M. Raciti and S. Nadjm-Tehrani, "Embedded cyber-physical anomaly detection in smart meters," Lecture Notes in Computer Science, vol 7722. Springer, Berlin, Heidelberg, DOI: 10.1007/978-3-642-41485-5_4.
96. S. McLaughlin, B. Holbert, A. Fawaz, R. Berthier and S. Zonouz, "A multi-sensor energy theft detection framework for advanced metering infrastructures," IEEE Journal on Selected Areas in Communications, vol. 31, no. 7, pp. 1319-1330, July 2013, DOI: 10.1109/JSAC.2013.130714.
97. M. Nabil, M. Ismail, M. Mahmoud, M. Shahin, K. Qaraqe and E. Serpedin, "Deep recurrent electricity theft detection in AMI networks with random tuning of hyper-parameters," 24th International Conference on Pattern Recognition (ICPR), Beijing, 2018, pp. 740-745, DOI: 10.1109/ICPR.2018.8545748.
98. R. Serra and A. C. Rodríguez, "The Ljung-Box test as a performance indicator for VIRCs," International Symposium on Electromagnetic Compatibility - EMC EUROPE, Rome, 2012, pp. 1-6, DOI: 10.1109/EMCEurope.2012.6396834.
99. G. E. P. Box and D. A. Pierce, "Distribution of residual autocorrelations in autoregressive integrated moving average time series models," Journal of the American Statistical Association, vol. 65, No. 332, December 1970, pp. 1509-1526.

100. M. Abdollahian and R. Foroughi, “Regression analysis of ozone data,” International Conference on Information Technology: Coding and Computing (ITCC'05), vol. 2, Las Vegas, NV, 2005, pp. 174-178 Vol. 1, DOI: 10.1109/ITCC.2005.242.

101. J. P. Barrett, “The coefficient of determination - some limitations,” The American Statistician, 28:1, pp. 19-20, 1974, DOI: 10.1080/00031305.1974.10479056.

102. C. C. B. Oliveira, N. Kagan, A. Meffe, S. Jonathan, S. Caparroz and J. L. Cavaretti, “A new method for the computation of technical losses in electrical power distribution systems,” 16th International Conference and Exhibition on Electricity Distribution, Part 1: Contributions. CIRED. (IEE Conf. Publ No. 482), Amsterdam, Netherlands, 2001, pp. 5 pp. vol.5-, DOI: 10.1049/cp:20010889.

103. H. C. Mandhare and S. R. Idate, “A comparative study of cluster based outlier detection, distance based outlier detection and density based outlier detection techniques,” International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, 2017, pp. 931-935, DOI: 10.1109/ICCONS.2017.8250601.

104. H. Aguinis, R. K. Gottfredson and H. Joo, “Best-practice recommendations for defining, identifying, and handling outliers,” Organizational Research Methods 16(2), 2013: pp. 270-301, DOI: 10.1177/1094428112470848.

105. S. Seo, “A review and comparison of methods for detecting outliers in univariate data sets,” Masters dissertation, 2006, University of Pittsburgh.

106. J. Shlens, “A tutorial on principal component analysis,” Educational. 51. arXiv:1404.1100v1 [cs.LG], 2014.

107. V. Struc and N. Pavesic, “A comparison of feature normalization techniques for PCA-based palm-print recognition,” Proceedings of the Conference on Mathematical Modelling, MATHMOD’09, 2009, pp. 2450–2453.

108. A. Ben Ayed, M. Ben Halima and A. M. Alimi, “Survey on clustering methods: towards fuzzy clustering for big data,” 2014 6th International Conference of Soft Computing and Pattern Recognition (SoCPaR), Tunis, 2014, pp. 331-336, DOI: 10.1109/SOCPAR.2014.7008028.

109. Nisha and P. Jai Kaur, “A survey of clustering techniques and algorithms,” 2nd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, 2015, pp. 304-307.

110. A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: a review,” ACM computing surveys (CSUR), vol. 31, no. 3, pp. 264–323, 1999.

111. L. Yu and C. Zhou, “Determining the best clustering number of K-means based on bootstrap sampling,” 2nd International Conference on Data Science and Business Analytics (ICDSBA), Changsha, 2018, pp. 78-83, DOI: 10.1109/ICDSBA.2018.00022.
112. T. J. Bihl, K. W. Bauer, M. A. Temple and B. Ramsey, “Dimensional reduction analysis for physical layer device fingerprints with application to ZigBee and Z-Wave devices,” MILCOM 2015, IEEE Military Communications Conference, Tampa, FL, 2015, pp. 360-365, DOI: 10.1109/MILCOM.2015.7357469.
113. T. M. Mitchell, “Machine learning,” McGraw-Hill Companies, Inc., 1997, ISBN: 0070428077.
114. P. Mylonas, M. Wallace and S. Kollias, “Using k-nearest neighbor and feature selection as an improvement to hierarchical clustering,” Lecture Notes in Computer Science, May 2004, pp. 191-200, Springer-Verlag Berlin Heidelberg, DOI:10.1007/978-3-540-24674-9_21.
115. “caret-package.R”, available at https://github.com/topepo/caret/blob/master/pkg/caret/R/knn3.R#L265, accessed 11/09/2020.
116. S. Yadav and S. Shukla, “Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification,” IEEE 6th International Conference on Advanced Computing (IACC), Bhimavaram, 2016, pp. 78-83, DOI: 10.1109/IACC.2016.25.
117. A. Demir and H. E. Çetingül, “Sequential hierarchical agglomerative clustering of white matter fiber pathways,” IEEE Transactions on Biomedical Engineering, vol. 62, no. 6, pp. 1478-1489, June 2015, DOI: 10.1109/TBME.2015.2391913.