Robust data cleaning procedure for large scale medium voltage distribution networks feeders

Nathalie Huyghues-Beaufond

Thesis submitted for the degree of
Doctor of Philosophy

Imperial College of London
Department of Electrical and Electronic Engineering
Control and Power Research Group

March 2020

Declaration

I hereby declare that the work presented in this thesis is my own work, and, where appropriate, contributions from other people whom have been acknowledged.

Nathalie Huyghues-Beaufond
Tuesday 3rd March, 2020

Copyright

Nathalie Huyghues-Beaufond
Tuesday 3rd March, 2020

Abstract

Relatively little attention has been given to the short-term load forecasting problem of primary substations mainly because load forecasts were not essential to secure the operation of passive distribution networks. With the increasing uptake of intermittent generations, distribution networks are becoming active since power flows can change direction in a somewhat volatile fashion. The volatility of power flows introduces operational constraints on voltage control, system fault levels, thermal constraints, systems losses and high reverse power flows. Today, greater observability of the networks is essential to maintain a safe overall system and to maximise the utilisation of existing assets. Hence, to identify and anticipate for any forthcoming critical operational conditions, networks operators are compelled to broaden their visibility of the networks to time horizons that include not only real-time information but also hour-ahead and day-ahead forecasts. With this change in paradigm, progressively, large scales of short-term load forecasters is integrated as an essential component of distribution networks' control and planning tools.

The data acquisition of large scale real-world data is prone to errors; anomalies in data sets can lead to erroneous forecasting outcomes. Hence, data cleansing is an essential first step in data-driven learning techniques. Data cleansing is a labour-intensive and time-consuming task for the following reasons: 1) to select a suitable cleansing method is not trivial 2) to generalise or automate a cleansing procedure is challenging, 3) there is a risk to introduce new errors in the data. This thesis attempts to maximise the performance of large scale forecasting models by addressing the quality of the modelling data. Thus, the objectives of this research are to identify the bad data quality causes, design an automatic data cleansing procedure suitable for large scale distribution network datasets and, to propose a rigorous framework for modelling MV distribution network feeders time series with deep learning architecture. The thesis discusses in detail the challenges in handling and modelling real-world distribution feeders time series. It also discusses a robust technique to detect outliers in the presence of level-shifts, and suitable missing values imputation techniques. All the concepts have been demonstrated on large real-world distribution network data.

Acknowledgements

I would like to express my eternal gratefulness to my supervisor, Professor Goran Strbac, who provided me with continuous support and encouragement. He believed in me and trusted me by allowing me to explore a completely new area of expertise; he let me worked outside my comfort zone. He encouraged me when my confidence was at its lowest point.

I would like to thank Dr Simon Tindemans for his inspiring guidance, he is such a great individual, humble but so gifted and passionate.

I would like to thank Professor Stephen Mc Arthur and the academic team of the Centre of Doctoral Training in Future Power Networks of the University of Strathclyde and the EPSRC for funding my studies and allowing me to build a vast network in the energy industry. I met so many people who have contributed in one way or another to the success of my PhD journey. I had the great opportunity to collaborate with UKPN's innovation team. During my time at UKPN, I met the most talented and influential people in the industry, to name a few, Matthieu Michel, Alex Jakeman, Tobi Babalola, Guilia Privitera, Luca Grella, Tazi Edwards. I want to thank them all; we had such a great time collaborating on the KASM project.

I am thoroughly grateful for the support of all my colleagues in the Control and Power control group, in particular, Paola Falugi, Oyniema Nduka, Olayinka Ayo, Mingyang Sun, Dawei Qiu, Professor Tim Green, Professor Bikash Pal, Dr Adrea Junyeant-Ferre, Alexandre Moreira Da Silva, Thiago Ribeiro Furtado De Mendonca and Jenela Dragovic.

Last but not least, I owe the biggest thanks to my family members who have been a constant source of support and encouragement. Thank you, Hashem, the G. of Israel for allowing me waking up each morning, one morning after the other and fighting through the pain, the doubts, the tiredness and the isolation. You are my greatest strength and inspiration. Amen.

Dedication

To Alexandra H., Alex H.B., Sandy G., Julien P., Gregory P., Benjamin H.B., Frederic M., Emmanuel L.

Le monde ne sera sauvé, s'il peut l'être, que par des insoumis.

Andre Gide

Contents

List of Tables

List of Figures

Acronyms

ADAM	Adaptive Moment Estimation
ADMS	Advanced Distribution Management System
AIMD	Additive-Increase Multiplicative-Decrease
ANN	Artifical Neural Network
ARIMA	AutoRegressive Integrated Moving Average
ARX	AutoRegressive with Exogenous
BST	British Summer Time
CA	Contingency Analysis
CNN	Convolutional Neural Network
DER	Distributed Energy Resource
DMS	Distribution Management System
DNN	Deep Neural Network
DNO	Distribution Network Operator
DNP3	Distributed Network Protocol
DOC	Directional Overcurrent
DSL	Digital Subscriber Line

DST	Daylight Saving Time
FDNN	Feedforward Deep Neural Network
FEP	Front-End Processor
GIS	Geographic Information System
GPU	Graphics Processing Unit
GSP	Grid Supply Points
HV	High Voltage
HVDC	High-Voltage Direct Current
IED	Intelligent Electronic Device
KASM	Kent Active System Management
LAN	Local Area Network
LCNF	Low Carbon Networks Fund
LSTM	Long Short Term Memory
LV	Low Voltage
MAP	Maximum A Posteriori
MAPE	Mean Average Percentage Error
MV	Medium Voltage
NAk-foldV	Nested Adjusted k-fold validation
NAN	Not A Number
NAT	Not A Time
NROV	Nested Rolling-Origin-Validation

OLTC On Load Tap Changers

PV Solar Photovoltaic

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

RTU Remote Terminal Unit

SGD Stochastic Gradient Descent

SGT Super Grid Transformer

SI Standard Inverse

STLF Short-Term Load Forecasting

TAI International Atomic Time

TBATS Trigonometric Box-Cox ARMA Trend and Seasonal

TCP Transmission Control Protocol

TSO Transimission System Operator

UK United Kingdom

UKPN UK Power Networks

UTC Coordinated Universal Time

WAN Wide Area Network

Symbols

β ADAM smoothing parameter

$\boldsymbol{Cov}$ Covariance matrice

d Input vector dimension

$\mathcal{D}_{\mathcal{X},\mathcal{Y}}$ Input-target probability distribution

$\mathbf{E}$ Expectated value

η Learning rate

$f_{\mathcal{S}}$ Empirical target function

$\mathcal{H}$ Hypothesis space

$\mathbb{1}[.]$ indicator function

N_l Number of neurons at layer l

$\mathcal{L}$ Loss function

$l \in \{1, \ldots, L-1\}$ A hidden layer

L Number of layers

M_l Weighted edges matrix at layer l

n Number of training samples

$\mathcal{N}(0, \sqrt{\frac{2}{N_l}})$ Normal distribution of mean 0 and variance $\sqrt{\frac{2}{N_l}}$

∇ Gradient

$\mathbf{\Phi}$ Collection of adjustable parameters

$\propto$ Is proportional to

ψ Activation function

$\mathcal{S}$ Training set

$\mathcal{S}'$ Testing set

t Timestep a time t

$\mathcal{L}_{\mathcal{S}}$ Training error

$\mathcal{L}_{\mathcal{S}'}$ Testing error

$\mathcal{U}(0, \frac{1}{\sqrt{N_{l-1}}})$ Uniform distribution with parameters 0 and $\sqrt{N_{l-1}}$

Chapter 1

Introduction

'It is very difficult to predict, especially the future'. Nils Bohr

1.1 Motivation and objectives

1.1.1 The Kent Active System Management project

The motivation for this work originated from a year and a half collaboration with UKPN in the KASM project. The KASM project is a LCNF tier 2 project. The project aimed to carry out a range of technical innovation trials to demonstrate more advanced operations and planning techniques for 132 and 33 kV East Kent's networks. The East Kent area in the South East of England region is a good example of how the uptake of distributed generation is changing the way electricity networks operate. In the area, electricity networks face increasing operational and planning challenges due to large amounts of intermittent wind and solar generation being connected to the distribution network where local demand is limited. Since 2013, the region has experienced increasing volumes of renewable, with wind and PV generation connecting to transmission and distribution networks. Besides, the area is home to three HVDC interconnectors to mainland Europe, with one

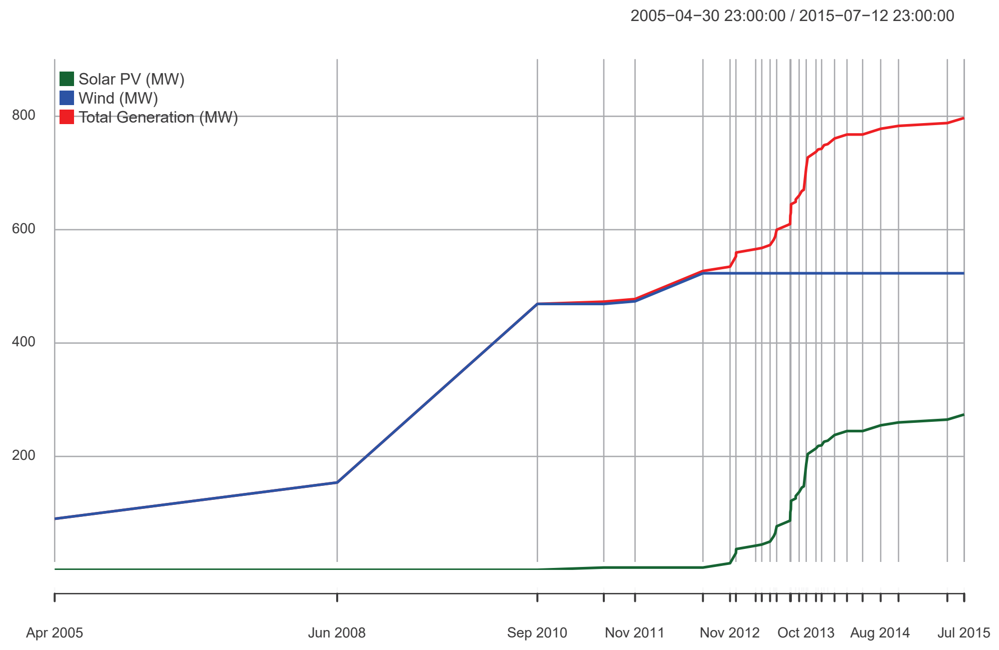

Figure 1.1: Chronology of intermittent generation connection in South East of England

more planned in 2020. Distribution and transmission networks are highly interconnected in the region, which has resulted in significant interdependence between transmission and distribution networks in the area. Since 2016, more than 800MW of wind and solar generation are connected to the distribution network, as reported in Fig. 1.1. However, since 2014, UKPN has been able to provide only a few connections, and new offers can only be scheduled beyond 2020. The main reason for new connections being difficult to accommodate in the South East of England is the system congestion that part of the network experiences under $N-1$ operating constraints. In the area, distribution and transmission networks are highly meshed and interconnected. A large portion of the 132kV distribution networks operates in parallel with the 400kV transmission network as exhibited in Fig. 1.2. As a result, power flows on either network can lead to post-fault overloads on the other. Additionally, the transmission network in the area has three HVDC interconnections with continental Europe, over which market forces almost exclusively determine the flows. One future HVDC connection is planned in the area for 2020. The variability of power flows of intermittent, and interconnector flows introduce various operational con-

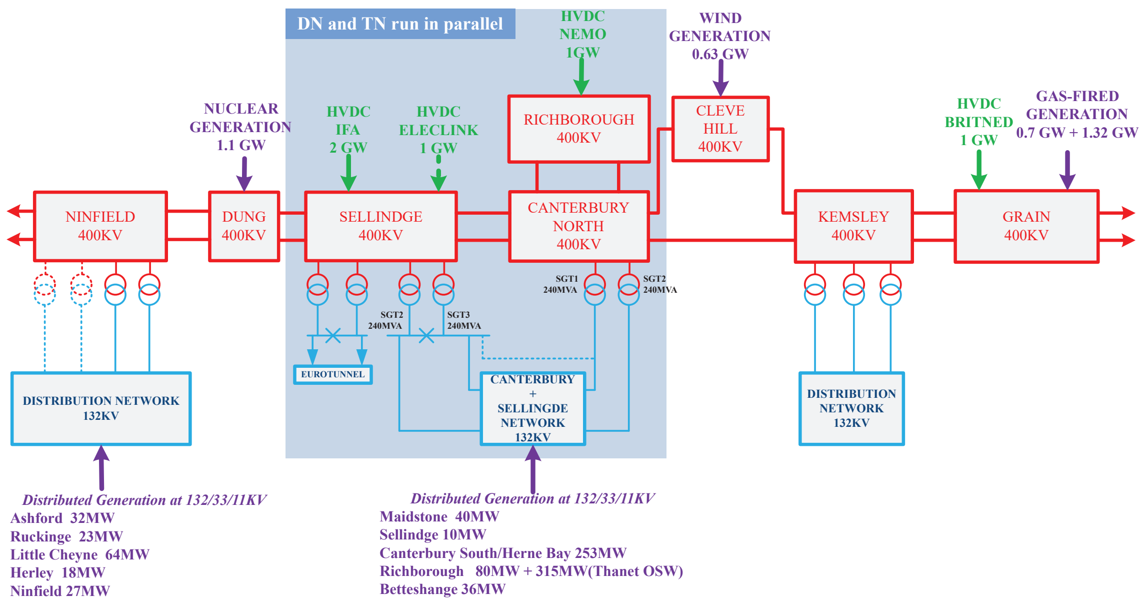

Figure 1.2: Distribution and transmission networks interdependency in East Kent operating area.

straints in the area such as voltage control, system fault levels, thermal constraints and *high reverse power* flows. The last few years have seen several GSP come under pressure due to high levels of reverse power flow. In practice, the reverse power flow capability of primary transformers is limited to 66.3% of the rated capacity of the transformer. The reverse power flow constraint is the main driver for curtailing 'green power' during the outages maintenance season. Hence, the following section investigates the causes of the reverse power limitations imposed on grid-transformers.

Reverse power flow is the active power flowing from the MV winding to the HV winding of a distribution network power transformer. Transformers have a thermal rating limit that should not be exceeded; this rating being symmetrical, it does not depend on the direction of the power flow. Therefore, in theory, the reverse power flow capability of HV and primary transformers should be limited by the nameplate rating of transformers and post-fault short-term overloading considerations. However, additional limiting factors such as OLTC mechanism and DOC protection settings prevent power transformers from carrying their full reverse power capability. The OLTC mechanism often found in primary transformers (33/11kV) can constrain the transformer's reverse power flow. The OLTC is used for voltage regulation (i.e. load drop compensation) and/or phase shifting. It varies the transformer ratio during energised condition using the *make before break contact* concept. The transformer's ratio is changed by varying the numbers of turns either on the primary or secondary winding of the transformer. A transition impedance is used as an adjacent bridging tap to transfer load from one tap to the other without disruption or noticeable change in the load current. There are two types of OLTC: the reactor type and the high-speed resistor type [42]. The high-speed resistor tap-changers are categorised as either double-resistors or single-resistor arrangements. The resistor and reactor are used as an impedance to limit the circulating current generated at bridging positions. While the reactor and double-resistors tap-changer type do not alter the inherent symmetrical attribute of transformers, the single-resistor type used in the asymmetrical pennant cycle transition method reduces the transformer reverse power flow capability [65].

In [87], non-linear optimisation models are used to compute the reverse power flow capability for one type of single-resistor tap changer which is installed on the HV side of a primary transformer. The optimised reverse power flows were computed for various bridging resistors, HV side windings configurations, transformers sizes and vector groups. The study shows that reverse power capacity can be reduced as little as 20% of the transformer nameplate rating. Also, the results demonstrate that Dy11 transformers have greater reverse power flow capability than Yy0 transformers. Depending on the resistance value and the size of the transformer, the optimised reverse power flow capability for Dy11 transformers can reach 90% of the transformer nameplate rating while the reverse power for Yy0 transformers only extends as far as 66% of the transformer nameplate rating. In mesh or ring networks with multiple infeed points, DOC relays are placed in locations where the direction of fault currents is likely to change. They also play an essential role as back-up protection, sensing high impedance fault currents (currents that are lower than the nominal current). The pick-up settings of DOC relays can impose very challenging reverse power flow constraints. These relays are designed to operate for the minimum expected fault level at their location point. Since DOC relays typically use SI IDMT characteristics, the relay current setting is selected to sense for at least half of the minimum expected fault level, which can be very low in some grid locations [121]. The reverse power restriction for accommodating distributed generation on 33kV to 132kV distribution networks was investigated in [120]. UKPN trialled a solution that combines load blinding scheme with a DOC relay. A Directional Voltage-Dependent Overcurrent (DVDO) scheme was added to the solution to prevent the maloperation of the scheme in the presence of high resistance fault.

1.1.2 Research motivation

The large amount of distributed generation connecting to the East Kent network has eroded the capacity margin that existed in the region. The congestion management issue on this complex and interconnected network has become increasingly challenging due to the high volatility of power flows on this part of the network. To overcome the congestion problem during $N-1$ operating conditions, UKPN investigated several technical solutions. The cost associated with most of the asset-based solutions were prohibitive or the solutions ineffective. For instance, the reinforcement of the Canterbury North site with the installation of a third SGT was found to exceed £20m mainly due to time constraints. The transfer of the excess power to the nearest SGT site was investigated, and the associated cost was estimated at £45m. Other technical solutions such as adding additional $N-1$ intertrip circuits were investigated and rejected due to their prohibitive costs. Active impedance devices and quad-boosters were studied and found to be an ineffective solution.

In addition to the congestion management issue, control engineers and planners have expressed concerns in managing the network in short-term and near-real-time due to lack of system visibility. The range of operating scenarios has increased as well as the uncertainties related to intermittent generation, the power injection from interconnector and the local demand. Short-term planning studies have become challenging and time-consuming when performed in the usual manual way. The impacts on system operation have affected the regional DNO, the TSO and the renewable generators of which the output is occasionally constrained as a preventive action. This has led UKPN, the regional DNO, to develop the innovative project KASM.

The project integrates a new Inter-Control Centre Communication Protocol (ICCP) link which enables real-time data exchange between National Grid Company (NGC), the TSO and UKPN. A new CA engine was integrated alongside the DMS in the UKPN control room. The CA computes online power flows on the 400 kV, 275 kV, 132 kV, 33 kV, and 11 kV networks in East Kent area and prepares contingency analysis studies within near real-time operational time frames. The KASM solution also incorporates new forecasting modules that provide STLF and short-term wind and solar generation forecasts [74]. Initially, the project was planned to span over a three years period starting February 2015 but was extended for six months. During the project development, many unexpected difficulties arose, and while the project was evolving from the conceptualisation and specification stage to the implementation and integration stage, various *data quality* issues transpired. By nature, KASM was a data-driven project, and its success lied essentially on two data properties: 1) the data consistency between the network planning tool (Power Factory) and the operational DMS (PowerOn); the data from these two systems are merged in the CA engine to create a complete dynamic model of the network (significant development time shared between UKPN and the CA's manufacturer, was spent to handle inconsistencies and discrepancies between systems) 2) The accuracy of the measurements used to estimate the state of the network in near-to-real-time and to train and produce accurate forecasts. The data quality issues which were described as in [74] have greatly motivated the present work.

The STLF modules were entirely developed by the CA's manufacturer commissioned to deliver the project. However, emerged the need for a UKPN's team member to design the factory acceptance tests (FAT) for the STLF modules. Soon, it transpired that no one within the organisation, including within the innovation team was qualified or had any prior experience (even in broad terms) with forecasting. Thereof, the situation revealed a knowledge gap. Since someone was required to initiate the investigation on the subject to plan some proven and efficient factory tests, this research project took off. While the manufacturer did not communicate to the team the technology that was used to develop the forecasters, intuitively, the ANN technology was investigated first. Later on, it appeared that the intuition was good since the manufacturer did use ANN to develop the STLF module.

KASM and the Electricity Flexibility and Forecasting System (EFFS) project conducted by Western Power Distribution (WPD) are two innovation projects that implement STLF at the distribution level. The latest is due for completion in January 2021. There is no doubt that future innovation projects will also require short-term load and generation forecasting at various timescales and in diverse applications. Thus, predictive analytics will become an integral part of distribution networks control and planning tools soon. Predictive analytics is essential to make the grid smarter, and the range of innovation that can be built on the energy industry *Big Data* is unlimited.

1.2 Literature review

The electric power industry uses the term STLF to refer to the estimation of the system demand over a time horizon ranging from less than one hour to one week. System demand concerns the electricity consumption of a large geographical area where the load is aggregated across an entire region. STLF has been used for decades by transmission system operators TSOs, and since the late sixties, significant research has been devoted to the development of methodologies for the short-term forecast of system load [23]. STLF plays an essential role in load dispatching, reserve allocation, security assessment, generation plant scheduling and unit commitment decisions. Various techniques were developed starting in the mid-sixties to predict system load power grids. STLF techniques were divided into five broad categories: multiple linear regression, spectral analysis, stochastic time series, exponential smoothing, and state-space methods (see e.g. reviews in [1, 97]). Except for spectral analysis and multiple linear regression, these techniques fell in the realm of statistical methods and rely on the assumptions of stationarity and linearity of the underlying process. Thanks to Box and Jenkins' works, a class of models capable of handling nonstationary processes were developed (see e.g. [20]). Soon, ARIMA, Kalman filter [25, 76] and general exponential smoothing [32] became the preferred modeling approaches for electricity demand. Nowadays, short-term load forecasting statistical models are built upon two main modeling frameworks, the seasonal ARIMA class models [28, 92, 98] and the exponential smoothing class models [36, 55, 75, 114]. Meanwhile, short-term load forecasting based on artificial intelligence (AI) approaches took on momentum with ANN receiving the largest share of attention due to their universal ability to learn complex nonlinear functions. Nonlinear modelling of electricity load data has long been claimed to be useful because hourly and temperature dependencies are assumed nonlinear. In [68], Hipper *et al* provide a comprehensive list of references on ANN applications with the review of 40 journal articles that reported the investigation and application of neural network to the short-term load forecasting problem. Recently, ANNs have evolved into DNN with the emergence of powerful learning architectures such as LSTM and CNN. With the advent of smart metering, large amounts of high dimensional data are available to the research community which has dedicated most recent studies in the implementation of frameworks to advance the challenging problem of short-term forecasting volatile energy demand of loads connected to low voltage distribution network.

In [108], Shi *et al.* made a first attempt in addressing household load forecasting by proposing a pooling-based deep recurrent neural network framework. The proposed framework can be assimilated to a data augmentation technique. It aims to tackle the overfitting issue associated with DNNs by utilizing correlations and interactions between neighbouring households as a mean of compensating for insufficient sample sizes. Although it involves a data cleaning process, the steps are taken and its effect on the results are not clearly described. In [81], a framework based on LSTM technology is proposed to address the short-term load forecast of individual residential households. An interesting attempt to visualize the internal states of the LSTM and track daily consumption patterns is presented in the paper. Although it was not discussed, it would have been interesting to investigate if LSTM internal states could potentially be an artificial disaggregation of the electricity demand to individual appliances consumption. Here again, there was no indication regarding the cleaning of the data.

In [24], Mengmeng *et al.* propose an interesting study that sets the problem of multistep forecasting strategies rigorously by formulating the day-ahead load forecast of commercial buildings with both, recursive and direct strategies. The study compares the results of the 24-h forecasts when forecasts are generated with SARIMAX, LSTM and CNN models. Both architectures, CNN and LSTM, were configured into recursive and direct multistep strategies. The direct multistep implementation of CNN is reported to perform best on average, but an equally important outcome is that LSTM performs better in a recursive configuration than in direct multistep implementation. A one-year period of historical data with 5% missing data was used to predict consumption on weekdays only. Missing values were handled by listwise deletion. Listwise deletion consists of removing all time-stamped rows for which one or more observations are missing. In the context of time series, listwise deletion produces an irregular spaced time series which can affect the structural dependencies of the series. The author overcame the aforementioned issues by creating multiple sections of time series bounded by the missing observations.

Demirhan *et al.* [38] were specifically concerned with the impact of missing data estimation on the accuracy of solar irradiance short-term forecast, identifying the imputation methods that generate the best estimations of solar irradiance missing values. They performed a comparison and evaluation of 36 univariate imputation strategies to real complete solar irradiance datasets. Imputation strategies include simple imputation with mean, mode and median, interpolation, Kalman filtering, Kalman smoothing, random sampling, persistence, weighted moving average and seasonal decomposition. Several rates of missing data were achieved artificially, and the localization of missing observations was spread strategically across complete data. The study has identified interpolation, weighted moving average, and Kalman filtering as the most suitable imputation strategies for solar irradiance dataset. In addition, Kalman filtering, Kalman smoothing and interpolation imputation are reported to be the methods that achieve the most accurate estimates for hourly solar irradiance. The authors did not consider structural breaks in the investigation but suggested the topic as a direction for future research.

In [104], Rahman *et al.* address the problem of training medium to long-term residential and commercial building electricity consumption forecasting models in the presence of small and large gaps (segments) of missing observations in the hourly training dataset. Small gaps are imputed using linear interpolation while segment imputation is performed using a scheme based on LSTM model. The algorithm identifies the segment of missing values then estimates the missing observations as the weighted average of predictions produced by training two LSTM models: one with the data before the segment and the other with data after the segment. The main issues with this approach are that 1) the data is assumed to experience a unique segment of missing values and 2) the location of the segment must be such that the two training datasets have a decent sample size to capture the features of the underlying process.

One of the few studies investigating the impact of outliers and level-shifts on one day ahead forecast of system load can be found in [34]. The authors identified outliers to be the aftermaths of abnormally low demand due to a slackening of industrial output while level-shifts were associated with the winter load growth. The authors proposed a robust filtering algorithm based on the Kalman filter, which allows outliers to be filtered and replaced with estimates generated by the filter. The robustification of the filter is achieved by using the one-sided Hampel function, which filters only large negative residuals that were identified as the most dangerous contamination for the predictive model.

Recently, Akouemo *et al.* proposed in [3] two data cleansing procedures tested on natural gas consumption series. Their implementations are based on ARX models and ANN models. The cleansing algorithms consist of two phases; first, an iterative process identified outliers one-by-one using a hypothesis testing procedure on residuals. At each iteration, the outlier is replaced using a 'naive interpolation'. The iterative process continues until no outlier can be found, and is followed by a second phase where all outliers are removed and imputed using either ARX or ANN.

Relatively little attention has been given to the short-term load forecasting problem of primary substations, probably due to the fact that load forecasts were not essential to secure the operation of passive distribution networks. Nowadays, with the increasing uptake of intermittent generations, distribution networks are becoming active since power flows can change direction in a rather volatile fashion. High shares of solar PV and wind generation are connected at all voltage levels in distribution networks, resulting in substantial uncertainty in their planning and operation routine [44]. The volatility of power flows introduces operational constraints on voltage control, system fault levels, thermal constraints, systems losses and high reverse power flows [44,73]. To maintain a safe overall system and maximize the utilization of existing assets, greater observability of the networks is required. DNOs are compelled to broaden their visibility of the networks to time horizons that include not only real-time information but also hour-ahead, day-ahead up to week-ahead forecasts. Also, distribution networks operators are investigating innovative ways of operating their networks such that the delivery of electricity to consumers remains secure and reliable, networks remain resilient to fault conditions while connection fees are kept as low as possible. With this change in paradigm, short-term load forecast technology is becoming an essential tool that can assist network operators and planners in identifying and anticipating any future critical operational conditions.

In [33], an earlier application of distribution feeder load forecasts is described. The article addresses the volt/var control problem and proposes a scheme that makes use of one-step-ahead forecast to optimize capacitors switching time. In the article, Civanler *et al.* adopted a time series decomposition approach for the modelling of industrial and residential feeder load. The article highlights the salient features associated with residential and industrial loads data with the main focus being on the design of a suitable model for the two types of feeders.

In [46], the treatment of bad real-time load readings is raised. Bad observations are said to be caused by thunderstorms or communication transmission outages. These outliers are detected and corrected based on specified upper/lower limits defined by offset tolerances for the normal load profiles. Chen *et al.* [31] describe an investigation of forecasts improvement of HV substation load where data quality enhancement is at the centre of the study. The article reports up to 20% of bad data and inaccurate measurements in the substation load historical data. A feeder load data correction framework is proposed; bad data are classified in two categories: outliers that are easy to detect (null data points or consecutive constant measurement) and those that are difficult to detect (subtle spikes). A detection strategy is developed for each type of outliers. The first outlier detection method uses thresholds built upon *Chebyshev's inequality* while the second compares typical daily and weekly patterns extracted by Fourier Transform from the partially cleaned data and compare the typical load curve to the raw data. Removed outliers are imputed using a linear transformation of the typical daily pattern. Forecasts are produced for raw and preprocessed testing datasets, and accuracy is reported for both data. It was applied to a scenario with a limited test set and in the absence of network configuration events.

In [41], Ding *et al.* focus on providing a steps-by-steps model design procedure to proficiently train and test ANN-based STLF for medium and low voltage distribution feeders. Its main contributions are twofold. The authors introduced an input variable selection methodology based on Gram-Schmidt orthogonalization and random probe techniques and a model selection based on virtual leave-one-out. The problem of missing values is handled by replacing missing observations with data from a similar day. In this study, 24-h ahead forecasts for two MV distribution feeders are produced using a recursive forecasting strategy. Forecasts accuracy are evaluated with the MAPE metric and reported as *15.5%* and *10.3%*. Outlier detection, structural breaks or level-shifts were not considered.

Today, short-term load forecasters are deployed at a large scale where hundreds of primary substation load time series data are to be modelled and forecast. In [73], Huyghues-Beaufond et al. provide an example of a real-world solution where a large number of MV distribution feeders forecasts are used for look-ahead contingency analysis studies. Real-world time series modelling is known to be a challenging task and MV distribution feeders time series are no exception. First, there are practical challenges associated with the manipulation of time series data (i.e. timestamps format issues, duplicate data points, timezone and daylight saving issues, diverging sampling, etc.). In addition, primary substation load profiles are mixtures of industrial, commercial and residential customers [77, 110]. Feeder data also have typical time series characteristics, such as a

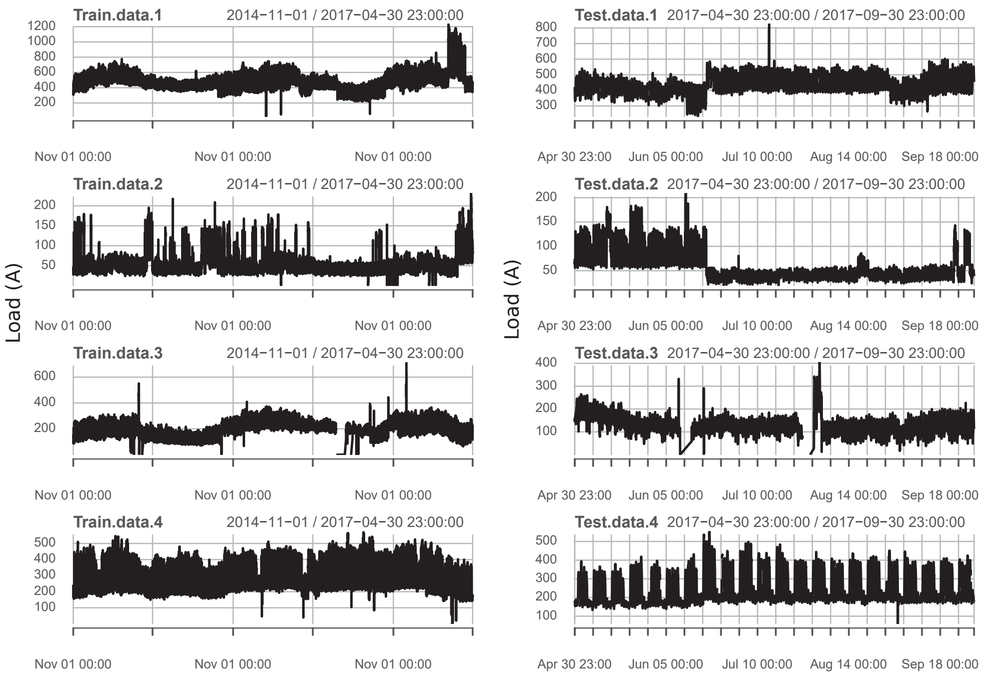

Figure 1.3: Real-world MV distribution feeders training and testing datasets

slow trend due to load growth over the years, several seasonal effects, annual cycle and pronounced dips around holidays periods [68]. Beside complex seasonal patterns, the data structure might change over time due to load transfer requirements or network reconfiguration operations [11]. Network reconfiguration is bound to happen from time to time since it is essential for 1) securing the operation of the network and 2) ensuring reliable energy delivery to the end consumer. Structural breaks in feeder time series affect the level of the data and occasion level-shifts; these features are present in historical data, and they will arise in future data. Thus, level-shifts have a double contribution to decreasing STLF accuracy. Their presence in the training and the testing datasets can affect, respectively, the estimation of the model parameters and impact on the accuracy of forecasts [34]. Another difficulty arises when one or multiple level-shifts occur in feeder series during the data preprocessing stage, particularly during the cleaning process, during which outliers are detected and removed from raw data and missing observations are estimated and imputed. Level-shifts affect outlier detection effectiveness if the detection procedure does not explicitly account for them. An example of real-world data is shown in Fig. 1.3.

1.3 Organisation of the thesis

Following this introduction and literature review, the remaining chapters of the thesis are organised as follows:

Chapter 2 provides a brief overview of statistical time series modelling techniques with a focus on modern methods. Hence, `Prophet` and TBATS models used for gaps of missing value imputation are discussed in detail.

Chapter 3 introduces the various components of deep network architectures in the context of regression problems. Also, the LSTM model used to produce point-forecasts is described.

Chapter 4 discusses the challenges that practitioners face when choosing a data cleansing strategy for real-world data. It also presents the *missingness mechanism* concept introduced by Little and Rubin. The assumption made on the missingness mechanism in this work is given, and the three main imputation techniques used to impute missing values in MV feeders' load data are described.

In **Chapter 5**, the MV feeders load datasets and weather datasets are introduced and explored. The operations used to prepare the datasets for short-term load forecasting are discussed, in particular, the challenges associated with the formatting of date-time objects (i.e. time-zone and DST) are described. Also, the root cause of the missing observations in the MV feeders load datasets is investigated in detail. GIS and domain knowledge are used to provide a possible root of the cause of the missing observations in the MV networks datasets.

Chapter 6 presents an automatic approach to detect and remove outlying observations in large scale MV feeders' time series, at an unknown location. The method combines two algorithms: a robust design of the binary segmentation algorithm which detects level-shifts in the data and, the Tukey's standard rule. The performances of the proposed method's and, an adapted version of outlier detection by hypothesis testing proposed in [4], are compared. The results have been submitted for publication. In addition, a semi-automatic outlier detection algorithm is proposed and described, it is applied to a few MV feeders data, and the algorithm's performance is demonstrated.

In **Chapter 7**, the framework used for designing and training 342 MV feeders short-term load forecasting models is described step-by-step. All the choices made during the design process are explained. The implementations of the NROV and NAk-foldV, two rigorous model evaluation procedures adapted for time series data, are described. The MV feeders data are preprocessed with eight data cleansing strategies and used to train short-term load forecasters; the performance of 24-hours-ahead forecasts are evaluated and discussed using robust statistics. The results have been submitted for publication.

In **Chapter 8**, TBATS and `Prophet` modelling techniques, and, *Imputation by windows*, a novel imputation approach, are used to fill gaps of missing values in training datasets. The imputation strategies impacts on one-step-ahead and 24-step-ahead forecasts are discussed.

Finally, **Chapter 9** summarises the findings in this work with a brief overview of the future research directions.

1.4 Contributions of the thesis

This thesis investigates the challenges associated with the STLF of large scale MV distribution feeders. It describes a detailed study of data cleansing and short-term forecasting for a large-scale MV distribution feeder dataset. To the best of author knowledge, the proposed study has not been conducted before. The paper also introduces two robust and computationally efficient data cleansing approaches for STLF of distribution feeders and performs and extensive analysis. The contributions of the thesis can be summarised as follows:

- It provides an in-depth discussion of the challenges associated with producing accurate short-term load forecasts at the medium voltage distribution level using real-world data.

- All studies are performed on real-world datasets comprising 606 (reduced to 342 MV) feeders, with measurements spanning two years and one half. The analysis simultaneously addresses outlier detection, missing values imputation, level-shift detection and short-term forecasting.

- An unsupervised automatic outlier detection for univariate time series data is proposed. It combines binary segmentation and an adapted version of the boxplot labelling rule to detect outliers in the presence of unknown numbers of level-shifts in the data. It is robust and has low computational complexity. The description of the method also includes a general formulation for the stopping criterion of the binary segmentation algorithm that is suitable for fitting L1-norm models. In addition, a semi-automatic outlier detection technique is introduced. The proposed method uses the median filter technique.

- An adaptation of Nested Cross-Validation to time series data named NROV, is proposed to tune the parameters and evaluate the models' generalisation performance. A comparison has been performed against a NAk-foldV.

- The performance of three missing data imputation techniques (Unconditional mean, Kalman smoothing and Hot Deck) is compared in combination with the outlier detection framework.

- The accuracy of short-term load forecasts is quantified and compared across the full ensemble of MV distribution feeders. Consistency of performance across different feeders and its dependence on outlier and imputation methods is analysed.

1.5 List of Publications

Journal

- N.Huyghues-Beaufond, S.Tindemans, P.Falugi, M.Sun, and G. Strbac, ‘Robust and automatic data cleansing method for short-term load forecasting of distribution feeders’, *Applied Energy journal*, Volume 261, March 2020, 114405

Conference

- N.Huyghues-Beaufond, A. Jakeman, S. Tindemans, and G. Strbac, ‘Enhancing distribution network visibility using contingency analysis tools’, *IET International Conference on Resilience of Transmission and Distribution Networks, 2018, Liverpool.*
- N.Huyghues-Beaufond, A. Jakeman, S. Tindemans, and G. Strbac, ‘Challenges in model and data merging for the implementation of a distribution network contingency analysis tool’ *CIRED, 12-15 June 2017, Glasgow.*

Chapter 2

Overview of modern statistical time series models

'The best qualification of a prophet is to have a good memory'.

Marquis of Halifax

2.1 Introduction

Single observations $\boldsymbol{y} = \{y_t\}_{t\in\mathsf{Z}}$ recorded sequentially over regular time increments are referred as *univariate time series.* Most time series are *stochastic* in that the future is only partly determined by past values. Many time series forecasting procedures are based on a statistical model that provides a stochastic data generating process which may be used to produce an entire probability distribution for a future time period $n + h$ with h referred as the *forecasting horizon.* An intrinsic feature of a time series is that adjacent observations are usually statistically correlated thereof not independent.

Univariate time series models come under different classes of stochastic frameworks. ARIMA are *autoregressive* processes that provide a wide class of generative models for describing stationary time series. Stationary stochastic processes are processes for which the statistical properties of the underlying model do not change over time. There are two forms of stationarity: strong stationarity and weak stationarity. A discrete stochastic process $\boldsymbol{y} = \{y_t\}_{t \in \mathbf{Z}}$ is *strongly stationary* if its associated probability densities are unaffected by time translation. A stochastic process is said to be *weakly stationary* if its mean $\mathrm{E}[\boldsymbol{y}]$ and autocovariance $\mathrm{Cov}(y_t, y_{t+k})$ are finite and constant through time. The variance $\mathrm{Var}[\boldsymbol{y}]$, is a special case of the latter when the lag k is zero. A process is second-order stationary if $\mathrm{E}[\boldsymbol{y}]$ and $\mathrm{Var}[\boldsymbol{y}]$ are a finite constants for all t. In real-world applications and particularly, in the electrical power industry, the stationarity assumption does not hold, which has led to the development of more flexible generative models that can account for nonstationarity in the underlying stochastic process.

Central to the statistical treatment of nonstationary time series are the *state space models* which provide a flexible framework that captures many of generative models as special cases, including autoregressive models, structural time series models and linear innovations state-space models. *Structural time series models* take full advantage of state-space frameworks [62]. In these frameworks, univariate time series models are formulated in terms of unobserved (latent) components for which a stochastic model is assumed. The level, trend and seasonal patterns are the unobserved components which represent the salient features of univariate time series. Similarly, *the innovation state space* approach of *exponential smoothing* describes a class of stochastic models that decompose univariate time series on weighted combinations of past unobserved components, with recent observations given relatively more weight than older [75].

Thus, *Bayesian modeling* plays an important role in relation to the prediction and filtering of time series since conditional distributions contain all available information about future values [43]. In Bayesian modeling, one aims to specify a posterior distribution by defining a conditional density function given in Eq. 2.1 conditioned on the data (observations) $\boldsymbol{y}$ with a set of parameters $\boldsymbol{\theta}$ explored by a Markov Chain. Hence, the Kalman filter has been widely advocated for time series filtering, prediction and smoothing. It is a powerful algorithm for the statistical treatment of structural time series models. It provides a framework for prediction error decomposition using a straight application of

Bayes rule

$$p(\boldsymbol{\alpha}_{0:n}|\boldsymbol{y}_{1:n}) = \frac{p(\boldsymbol{y}_{1:n}|\boldsymbol{\alpha}_{0:n})p(\boldsymbol{\alpha}_{0:n})}{p(\boldsymbol{y}_{1:n})} \tag{2.1}$$

$$p(\boldsymbol{\alpha}_{0:n}|\boldsymbol{y}_{1:n}) \propto p(\boldsymbol{y}_{1:n}|\boldsymbol{\alpha}_{0:n})p(\boldsymbol{\alpha}_{0:n}) \tag{2.2}$$

where $p(\boldsymbol{\alpha}_{0:n})$ is the *prior distribution* defined by the dynamical mode, $p(\boldsymbol{y}_{1:n}|\boldsymbol{\alpha}_{0:n})$ is the *likelihood* model for the measurements and $p(\boldsymbol{y}_{1:n})$, is the normalization constant. The next sections introduce *Prophet* and TBATS models and the general formulation of *state space models* for discrete signals. These models are stochastic by nature; they are used in this work either to impute gaps of missing values in MV distribution feeders data or to short-term forecast the load data. While TBATS is a univariate model, Prophet handles exogenous variables such as weather data and dummy variables. Furthermore, Prophet accommodates change-points in its formulation. Because periodic movements are an important feature of feeder time series, it was essential to consider in this experiment, models that are capable of modelling data with multiple seasonalities. It is common knowledge that electricity data exhibit repeating intraweek and intraday cycles. The methods that were considered aim to capture this feature of double seasonality. Multiple seasonal time series models divide in two main categories as displayed in Fig 2.1: *regression* and *state space*. Multiple linear regression (MLR), generalized additive model (GAM) and Prophet models belong to the regression category. Multiple seasonal ARIMA, double seasonal Holt-Winters (DSHW) and TBATS models are classified in the state space category. As shown in Fig 2.1, *Prophet* and TBATS are the most recent multiple seasonality models available in the two aforementioned model categories. Both model are from *generative* modeling approach. Nevertheless, before getting in the details of the model, it is essential to recall the state-space framework.

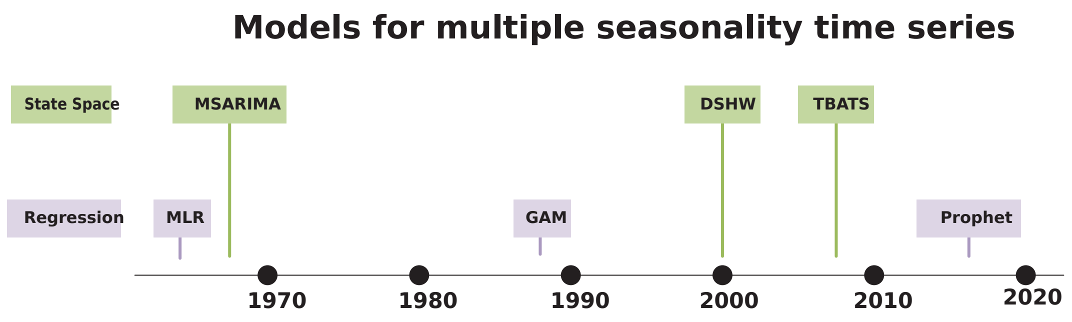

Figure 2.1: Chronology of time series models

2.2 Linear Gauss-Markov state-space model

The Bayesian inference provides the set of equations for computing the *posterior distributions* given the set of measurements once the model specifications have been identified. The state space form provides the key to the statistical treatment of dynamic structural models assumed to be *Markovian*. Let $\{\boldsymbol{y}\}_{t=1}^{n}$ be a univariate nonstationary time series generated by the state space model $\boldsymbol{\alpha_t} \in \mathbb{R}^{p\times 1}$ for which equation are provided in Eq. 2.3 and Eq. 2.4.

$$y_t = \mathbf{Z}_t\boldsymbol{\alpha}_t + \varepsilon_t, \quad \varepsilon_t \sim N(0, h_t), \text{ observation equation} \tag{2.3}$$

$$\boldsymbol{\alpha}_t = \mathbf{T}_t\boldsymbol{\alpha}_{t-1} + \boldsymbol{R}_t\boldsymbol{\eta}_t \quad \boldsymbol{\eta}_t \sim N(0, \mathbf{Q}_t), \text{ transition equation} \tag{2.4}$$

$$\boldsymbol{\alpha}_1 \sim N(\boldsymbol{a}_1, \boldsymbol{P}_1), \text{ seed vector} \tag{2.5}$$

where the matrices and vectors $\mathbf{Z}_t, \mathbf{T}_t, \mathbf{Q}_t$, $\mathbf{R}_t$ are the system matrices [43] and they are usually time invariant in filtering and smoothing applications. In many cases $\mathbf{R}_t$ is the identity matrix. The essence of state space models approach is to regard the observations as made up of some latent (hidden) components and a irregular term. The state space model associates each observation y_t with a latent *states vector* $\boldsymbol{\alpha}_t \in \mathbb{R}^{p\times 1}$, p being the number of non-observable states and $\mathbf{T}_t \in \mathbb{R}^{p\times p}$ the transition matrix. The transition equation is the mechanism for creating the inter-temporal dependencies between the values of a time series. The irregular term ε_t describes the stochastic part of y_t, it is assumed to be from a Gaussian white noise. An assumption is that the state and observation disturbance $\varepsilon_t \sim N(0, h_t)$ and $\boldsymbol{\eta} \sim N(0, \boldsymbol{Q}_t)$ are uncorrelated $\boldsymbol{Cov(\varepsilon_t, \eta_t)} = 0$ and independent, observations are conditionally independent given the latent states and y_t and $\boldsymbol{\alpha_t}$ are jointly Gaussian processes. The system variables $\mathbf{Z}_t, \mathbf{T}_t, \mathbf{Q}_t$ and h_t are chosen to represent a particular model and depend upon parameters that need to be estimated. Usually, parameters are estimated by maximizing the Gaussian Likelihood function of the chosen model. The seed states of the random vectors $\boldsymbol{\alpha}_1$ are usually treated as random vectors with a Gaussian distribution as indicated Eq 2.5. The *Kalman filter* provides a mean to compute the conditional mean value $a_{t|t-1} = \mathrm{E}[\boldsymbol{\alpha_t}|y_1, \ldots, y_{t-1}]$ and the covariance matrix $P_{t|t-1} = \mathbf{Cov}(\boldsymbol{\alpha_t}|y_1, \ldots, y_{t-1})$ of state vector $\boldsymbol{\alpha_t}$ at each time step [7].

2.3 Prophet model

The generative model `Prophet` was proposed in [115] by Facebook to address the challenges of most stochastic time series models: the parameters tuning and the lack of model flexibility. `Prophet` is a configurable model designed for non-experts forecasters that have domain knowledge about the data-generating process but little or no experience with time series models and methods. Prophet adopts a curve-fitting approach and decomposes the series in three main components: the trend function g_t, the seasonality function s_t and the effects of holidays h_t. The model accommodates piecewise trends, multiple seasonality and exogenous regressors. The general form of `Prophet` model is

$$y_t = g_t + s_t + h_t + \varepsilon_t \tag{2.6}$$

where ε_t is the error term, accounting for changes to the signal not accommodated by the model. The model combines the advantages of the generalized additive model (GAM) and exponential smoothing formulation to create a robust and flexible model, with fast fitting and easily interpretable parameters.

Trend components

There are two models of the trend function available within the `Prophet` framework's; a piecewise logistic growth model $g_t^{(1)}$ and a linear trend model $g_t^{(2)}$. Both formulations of the trend accommodate S change-points at times s_j with $j = [1, \ldots, S]$, where the growth rate is allowed to change. The models are given with the following forms

$$g_t^{(1)} = \frac{C_t}{1 + \exp(-(k + \boldsymbol{a}(t)'\boldsymbol{\delta})(t - (m + \boldsymbol{a}(t)'\boldsymbol{\gamma}))} \tag{2.7}$$

$$g_t^{(2)} = (k + \boldsymbol{a}(t)'\boldsymbol{\delta})t + (m + \boldsymbol{a}(t)'\boldsymbol{\gamma}) \tag{2.8}$$

In the logistic growth model, C_t defines the expected capacity of the system at any point in time, k is the base growth rate and m an offset parameter. Base growth rate and offset are adjusted following each change-point. The vectors containing the rate and offset adjustments are respectively $\boldsymbol{\delta} \in \mathbb{R}^S$ and $\boldsymbol{\gamma} \in \mathbb{R}^S$ impose the signal continuity between endpoints. The vector $\boldsymbol{a}(t) \in \{0, 1\}^S$ is represented as follow

$$a(t)_j = \begin{cases} 1, & \text{if } t \geq s_j \\ 0, & \text{otherwise} \end{cases} \tag{2.9}$$

The rate growth at any point t is given by $k + \sum_{j:t>s_j} \delta_j$. The offset parameter m is also adjusted for each change-points using the adjustment values in $\boldsymbol{\gamma} \in \mathbb{R}^S$ as follow

$$\gamma_j = \left(s_j - m - \sum_{i<j} \gamma_i \right) \left(1 - \frac{k + \sum_{i<j} \delta_i}{k + \sum_{i \leq j} \delta_i} \right) \tag{2.10}$$

By default, Prophet detects automatically the change-points however, forecasters may also finer control over the change-points detection process using dedicated input arguments.

Seasonal components

Prophet relies on classical Fourier series for modelling multiple seasonal patterns contained in time series. Fourier series method is particularly convenient to model seasonalities because of the simplicity and periodicity of trigonometric functions, of course, the Fourier series method is only defined for periodic functions. The Fourier transform is used to convert a series into frequency components or harmonics. Typically, only a few frequencies are needed to reconstruct the main features of a discrete signal. Seasonal effects are approximated with the following expression

$$s(t) = \sum_{n=1}^{N} (a_n \cos(\omega_n t) + b_n \sin(\omega_n t)) \tag{2.11}$$

where a_n and b_n are independent and normally distributed random variables with mean zero and standard deviations σ_n and $\omega_n = 2\pi n/P$ are harmonics over period P (number of cycles per units time i.e for yearly seasonality and hourly data $P = 8766$). A matrix of seasonality vectors $\boldsymbol{\beta} = [a_1, b_1, \ldots, a_n, b_n]^T$ is constructed with $2n$ parameters to be estimated. Each vector corresponds to the seasonal term computed at time t of the historical or future data. Even though Prophet was initially designed to fit daily data, if sub-daily data are used, daily seasonality will be fit automatically. Yearly, weekly and daily seasonalities are fit with $n = 10$, $n = 3$ and $n = 4$ Fourier terms respectively. Seasonality is selected depending on the historical data granularity and length.

Holidays and other regressors

A matrix of regressors is generated to include the effect of holidays in the models is computed in a similar way as seasonality. Days surrounding the holiday (within a defined window) are treated as holidays. A list of holidays needs to be provided to `Prophet`. An indicator function is used to indicate if time t is during holiday i so that a parameter K_i can be estimated to account for holiday effect on past and future values.

Model fit

Prophet model is set as a full probability model where the model of the observations and the model of the parameters are both probabilistic. Prophet uses the Limited-memory

Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) available in `Stan code` [26] to find the MAP to specify the optimal parameters for the historical data. Stan code is a probabilistic programming language for statistical modelling that offers users a suite of inference tools for fitting models. It provides full Bayesian inference through No-U-Turn Sampler (NUTS), an adaptive form of Hamiltonian Monte Carlo (HMC) sampler [71].

2.4 TBATS model

TBATS acronyms stand for Trigonometric seasonality, Box-Cox transformation [21], ARMA errors, Trend and Seasonal components which are the key features of the models [37]. TBATS models are rooted in the exponential smoothing innovations state-space modelling framework based on a trigonometric formulation. These models are capable of modelling time series with complex seasonalities features [36]. The Box-Cox transformation is a class of parametric **power transformation** used to normalize the data and correct skewness of the data. Box-Cox transformation helps to constrain non-negativity conditions, normalize variation in the seasonality, and linearise the trend. Let $y_t^{(\lambda)}$ be the one-parameter Box-Cox transformations on parameter $\lambda \in [-5, 5]$ of the positive observation y_t and described as follow

$$y_t^{(\lambda)} = \begin{cases} \dfrac{(y_t)^\lambda - 1}{\lambda}, \quad \lambda \neq 0 \\ \\ \log y_t, \quad \lambda = 0 \end{cases} \tag{2.12}$$

The TBATS models are described as follow

$$y_t^{(\lambda)} = l_{t-1} + \phi b_{t-1} + \sum_{i=1}^{T} s_{t-m_i}^{(i)} + d_t, \tag{2.13}$$

$$l_t = l_{t-1} + \phi b_{t-1} + \alpha d_t, \tag{2.14}$$

$$b_t = (1-\phi)b + \phi b_{t-1} + \beta d_t, \tag{2.15}$$

$$d_t = \sum_{i=1}^{p} \varphi_i d_{t-i} + \sum_{i=1}^{q} \theta_i e_{t-i} + e_t \tag{2.16}$$

where $s_{t-m_i}^{(i)}$ is the ith seasonal component, the components l_t and b_t represent the level and the trend with damping time t respectively, b represent the long-run trend d_t is the ARMA(p,q) process for residuals to model correlated error process, e_t is a Gaussian white noise. Each seasonality is modeled by a trigonometric representation based on Fourier series. One major advantage of this approach is that it requires only two seed states regardless of the length of period. Another advantage is the ability to model seasonal effects of non-integer lengths. For example, given a series of hourly observations, one can model leap years with a season of length 24×365.25. The seasonal part is given by the following trigonometric equations:

$$s_t^{(i)} = \sum_{j=1}^{k_i} s_{j,t}^{(i)}, \tag{2.17}$$

$$s_{j,t}^{(i)} = s_{j,t-1}^{(i)} \cos(w_i) + s_{j,t-1}^{*(i)} \sin(w_i) + \gamma_1^{(i)} d_t, \tag{2.18}$$

$$s_{j,t}^{*(i)} = -s_{j,t-1}^{(i)} \sin(w_i) + s_{j,t-1}^{*(i)} \cos(w_i) + \gamma_2^{(i)} d_t, \tag{2.19}$$

$$w_i = 2\pi j/m_i \tag{2.20}$$

where T is the number of seasonalities, m_i is the length of ith seasonal period, k_i is the amount of harmonics for ith seasonal period, λ is the Box-Cox transformation parameter, α, β are smoothing factors, ϕ is the trend damping, φ_i, θ_i are the $ARMA(p,q)$ coefficients and $\gamma_1^{(i)}, \gamma_2^{(i)}$ are the seasonal smoothing. Trigonometric seasonality is advantageous as it is more flexible, it reduces parametrization problem and supports non-integer seasonal periods. ARMA processes are added in state space models to correct for correlation structure remaining in the residuals. Seasonal factors with frequency s are defined by rotating states. For identification, the seasonal states set to sum to 0, reducing the required states by 1. Seasonal Fourier terms with frequency s using q harmonics where $q \leq [s/2]$. As a forecasting tool, optimizing over the full length series is unnecessary. Only the tail of the series is necessary for producing good forecasts

2.5 Summary

This section has described the formulation of the generic linear state-space framework and two generative models: TBATS and `Prophet`. Both models are fairly recent time series models, they are used in later sections because of the models' capability to fit seasonal intraday time series that exhibit repeating intraweek and intraday cycles. Here, the term 'seasonal' refers to a set of periods for which demand is assumed to be identical. In both formulations, seasonalities are modeled by a trigonometric representation based on Fourier series.

Chapter 3

Deep regression

'Artificial Intelligence is the new electricity'. Andrew NG

3.1 Problem statement

A learning problem is primarily defined upon the nature of its output. A regression problem computes a target function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ that predicts a *quantitative* output. A supervised learning algorithm receives two sets of data: a *training set* $\mathcal{S}$ of input and target pairs $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$ where $\boldsymbol{x}^{(i)}$ and $\boldsymbol{y}^{(i)}$are elements of the input space $\mathcal{X}$ and the target space $\mathcal{Y}$ respectively and a *test set* $\mathcal{S}'$. While both sets $\mathcal{S}$ and $\mathcal{S}'$ are disjoints, they are assumed to have been drawn independently from the same input-target probability distribution $\mathcal{D}_{\mathcal{X},\mathcal{Y}}$. The goal of the learning algorithm is to find a hypothesis $f_{\mathcal{S}}$:$\mathcal{X} \rightarrow \mathcal{Y}$ in the hypothesis space $\mathcal{H}$ that minimizes the training error with respect to the unknown $\mathcal{D}_{\mathcal{X},\mathcal{Y}}$ and the target function. Given a function f, a loss function $\mathcal{L}$, and a training set $\mathcal{S}$ consisting of n examples, the empirical risk minimization (ERM) problem select $f_{\mathcal{S}}$ as

$$f_{\mathcal{S}} \doteq \underset{f \in \mathcal{H}}{argmin} \ \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(\boldsymbol{x}^{(i)}), y^{(i)}) \tag{3.1}$$

The empirical error measures the average of discrepancies over all set of training pairs and the output produced by the learning algorithm. In deep-learning, the learning algorithm constructs a function $f(\boldsymbol{x}, \boldsymbol{\Phi})$ that approximates $\boldsymbol{y}$ where $\boldsymbol{\Phi}$ represents a collection of adjustable parameters. Hence, the learning problem consists of finding the values of $\boldsymbol{\Phi}$ that minimizes $\mathcal{L}_{\mathcal{S}}[f(\boldsymbol{x}, \boldsymbol{\Phi})]$.

In practice, the performance of the algorithm on the training set is of little interest since it must perform well on previously unseen inputs. Consequently, the *testing error* $\mathcal{L}_{\mathcal{S}'}[f(\boldsymbol{x'}, \boldsymbol{\Phi})]$ measures the *generalisation* performance of the learning algorithm on a sample of pairs $\{(\boldsymbol{x'}^{(i)}, y'^{(i)})\}_{i=1}^{m}$ that were collected separately. A extra *validation set* is drawn from the training set to validate the performance of the learning algorithm during training.

3.2 Artificial neuron

Deep neural networks are non-parametric estimators that have become invaluable tools for supervised learning. The building block of deep neural networks is the processing unit called *artificial neuron* represented in Fig. 3.1 . Its model was motivated by a model from neuroscience in which a cell receives multiple signals via synapses that fire (or not) with a certain intensity depending on the input signals [105]. An *artificial neuron* with *weights* $\{w_j\}_{j=1}^{n} \in \mathbb{R}$, *bias* $b \in \mathbb{R}$ and *activation function* ψ: $\mathbb{R} \to \mathbb{R}$ is defined as the function $f : \mathbb{R}^n \to \mathbb{R}$ given by

$$f(x_1, \ldots, x_n) = \psi\left(\sum_{i=1}^{n} x_i w_i - b\right) = \psi(\langle \boldsymbol{x}, \boldsymbol{w} \rangle - b) \tag{3.2}$$

where $\boldsymbol{x} = [x_i]_{i=1}^{n}$ and $\boldsymbol{w} = [w_i]_{i=1}^{n}$ are vectors. Nonlinear activation functions give neural networks their universal approximation capacity. However, the choice of activation functions affects heavily the performance of gradient descent-based optimization methods [105].

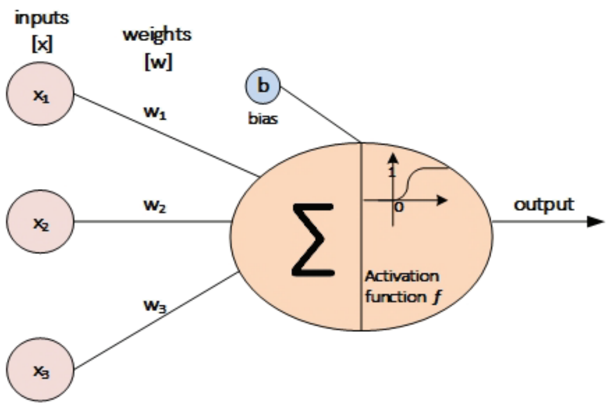

Figure 3.1: Artificial neuron

3.3 Activation functions

Activation functions with monotonic variation, zero centred nature and suitable gradient range are desirable properties of an activation function for improving training time and/or guarantee the learning algorithm convergence [96]. At present, the most popular activation function ψ is the ReLU, a non-saturating function defined as $\psi = \max(0, x)$. In past decades, ANN used smoother bounded non-linearities such as $\sigma(x) = (1 + e^{-x})^{-1}$ and $\tanh(x) = (e^{2x} - 1)(e^{2x} + 1)^{-1}$ but ReLU typically learns much faster [85].

ReLUs are one-sided piecewise linear units that enable sparse propagation of activation (by firing only a subset of neurons). This addresses the saturation problem of the hyperbolic and logistic function and the resulting vanishing gradients [99]. Their use improve gradients flow in the active part of the neural network and lead to better generalization performance and faster computation time [5, 53]. The activation function at the output layer often depends on the *loss function* $\mathcal{L}$ to be minimised. In regression tasks, there is typically one linear unit at the output layer. Linear units are more suitable for gradient-based learning algorithms because they do not saturate [54].

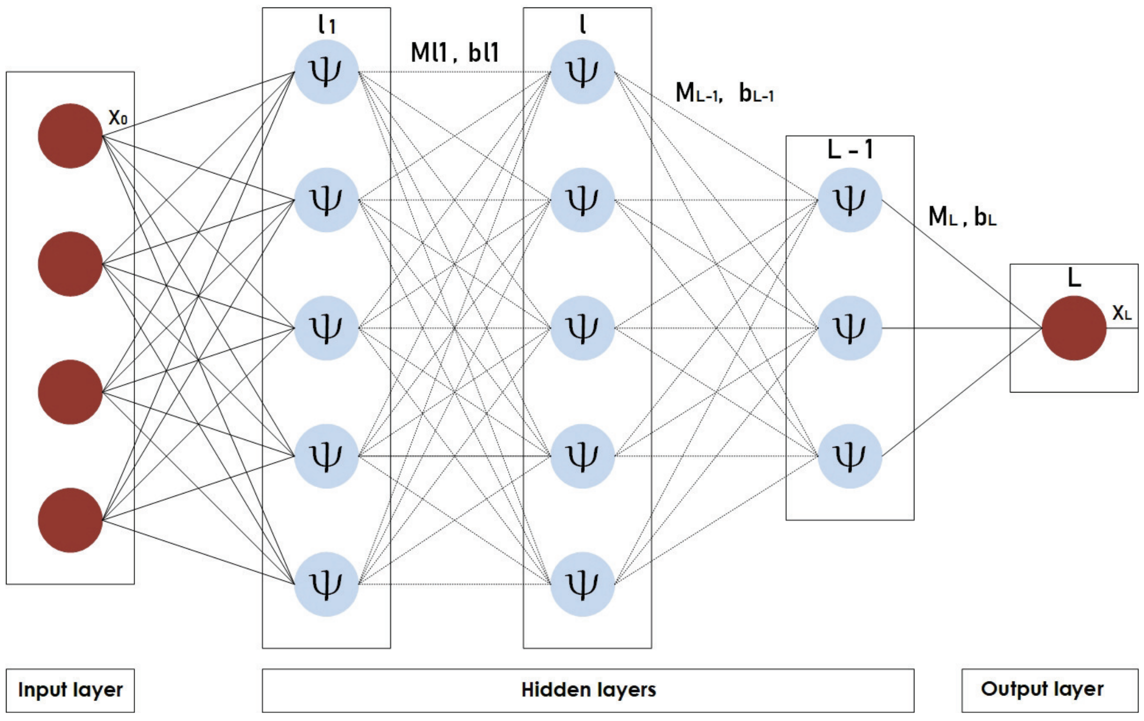

Figure 3.2: FDNN

3.4 Deep feedforward networks

Deep neural networks are used for data modelling as an alternative to standard nonlinear regression. Deep-learning methods have taken advantage of increasing amount of available computation (with the advent of faster hardware such as GPU and data (with larger training sets) and better learning regularization procedures (i.e. dropout). Deep learning methods learn very complex functions by extracting multiple levels of features from raw input data in a hierarchical fashion. Each level of features is obtained automatically by composing elementary non-linear modules that each transform input data into a higher level of features [85]. There are two mains type of deep-learning architectures: the feed forward neural network is a directed graph whose connections are acyclic and the recurrent neural network. Feed forward neural networks are hierarchical and parametric models that apply at each *layer* an affine transformation to their inputs followed by a point-wise nonlinear *activation function*. A representation of a FDNN architecture is provided Fig 3.2. The network's neurons are arranged in $L \in \mathbb{N}$ layers which are connected by weighted edges with each layer $l \in \{1, \ldots, L-1\}$ containing N_l number of neurons and the dimension of the input layer being $d \in \mathbb{N}$, . The affine transformation $\boldsymbol{W}_l : \mathbb{R}^{N_{l-1}} \to \mathbb{R}^{N_l}$ is defined by the weighted edges matrix $M_l \in \mathbb{R}^{N_l \times N_{l-1}}$ and the offsets or biases $\boldsymbol{b}_l \in \mathbb{R}^{N_l}$. Weighted edges matrix and biases form the *neural network architecture* Φ given as

$$\Phi = \{(M_l, b_l)\}_{l=1}^{L} \tag{3.3}$$

Let $L(\Phi) := L$ be the number of layers, Φ is a FDNN if $L(\Phi)$ is large. Given a nonlinear activation function $\psi : \mathbb{R} \to \mathbb{R}$, the realisation of Φ with activation function ψ maps

$$R_\psi(\Phi) : \mathbb{R}^d \to \mathbb{R}^{N_L}, \quad R_\psi(\Phi)(\boldsymbol{x}) := \boldsymbol{x_L} \tag{3.4}$$

where $\boldsymbol{x_0} := \boldsymbol{x}$ is the vector of inputs with dimension d, $\boldsymbol{x}_l = \psi(M_l \boldsymbol{x}_{l-1} - b_l)$ and $\boldsymbol{x}_L = M_L \boldsymbol{x}_{L-1} - b_L$. Given a FDNN architecture where d, L and $\{N_l\}_{l=1}^{L}$ are provided, *training the network* consists of learning the affine-linear functions $\{\boldsymbol{W}_l\}_{l=1}^{L} = \{\boldsymbol{M}_l(\cdot) - \boldsymbol{b}_l\}_{l=1}^{L}$ yielding the network $\Phi : \mathbb{R}^d \to \mathbb{R}^{N_L}$ such that

$$\Phi(\boldsymbol{x}) = \boldsymbol{W}_L \psi(\boldsymbol{W}_{L-1} \psi(\ldots \psi(\boldsymbol{W}_1(\boldsymbol{x})))), \tag{3.5}$$

3.5 Training deep neural networks

Training deep neural networks involves solving difficult optimization problems where cost surfaces are typically non-quadratic and high dimensional, with many local minima and many saddle points surrounded by flat regions [54]. The full training procedure is illustrated in Fig. 3.3. Typically, regardless the initial conditions, poor local minima are rarely a problem with large networks since the training algorithm reaches solutions of very similar quality. The learning difficulties rise because of large number of saddle points and the surface curves up in most dimensions. Gradient descent and quasi-Newton methods are almost prevalently used to train deep networks and find satisfying minimum. The most popular method for performing supervised learning tasks with gradient descent is *backpropagation* introduced by Rumelhart in [105]. The backpropagation algorithm is used to computed the gradient of the loss function with respect to all network parameters by applying the *chain rule* for gradients. The algorithm determines how to nudge the parameters of the networks to cause the most rapid decrease to the loss function. Let $\mathcal{S} = \{x_i, y_i\}_{i=1}^n \subseteq \mathbb{R}^d \times \mathbb{R}^{N_L}$ be the training set made of n samples pairs. Let $d = N_0 \in \mathbb{N}$, $N_1, \ldots, N_L, L \in \mathbb{N}$ and ψ the activation function. Considering the hypothesis space defined as

$$\mathcal{H} := \{R_\psi(\Phi) : \Phi = \{(M_l, b_l)\}_{l=1}^L, \quad M_l \in \mathbb{R}^{N_l \times N_{l-1}}, b_l \in \mathbb{R}^{N_l}\} \tag{3.6}$$

We wish to find the empirical target function $\Phi(\boldsymbol{x}) \approx f$ such that

$$F(\boldsymbol{\Phi}) := f_{\mathcal{H},\mathcal{S}} = \underset{f \in \mathcal{H}}{argmin} \sum_{i=1}^{n} \mathcal{L}(f_{\boldsymbol{\Phi}}(x^{(i)}), y^{(i)}) \tag{3.7}$$

where $\mathcal{L} : \mathcal{C}(\mathbb{R}^d, \mathbb{R}^{N_L}) \times \mathbb{R}^d \times \mathbb{R}^{N_L}$ is the *loss function.* In practice, deep neural networks are commonly trained with the *backpropagation* learning method that is based on *gradient descent* algorithm [86, 122]. Backpropagation learning methods differ with the amount of data used to compute the gradient of the objective function. *Batch learning* takes the entire training set to perform an *unique* update at each iteration (or epoch) of the gradient of the loss function w.r.t. the parameters $\boldsymbol{\Phi} = \{(M_l, b_l)\}_{l=1}^L$. The gradient is computed as $\nabla_{\boldsymbol{\Phi}} F(\boldsymbol{\Phi}) = \sum_{i=1}^n \nabla_{\boldsymbol{\Phi}} \mathcal{L}(f_{\boldsymbol{\Phi}}(x^{(i)}), y^{(i)})$, the algorithm is guaranteed to converge to the global minimum or a local minimum depending on the surface convexity. Backpropagation adjusts the parameters $\boldsymbol{\Phi}$ at each iteration t as follows

$$\boldsymbol{\Phi}_{t+1} \leftarrow \boldsymbol{\Phi}_t - \eta \nabla_{\Phi_t} F(\boldsymbol{\Phi}_t) \quad \text{for } t \in \mathbb{N} \tag{3.8}$$

where $\eta \in \mathbb{R}$ is the *learning rate*, a hyperparameter that determines the size of the steps to take to reach a minimum. Hyperparameters are parameters that appear in the training objective function, but not in the network architecture. The learning rate η can be either a constant or adaptive. Batch gradient descent can be very slow and intractable for large datasets that do not fit in memory. In contrast, the *SGD* updates the parameters for a single training example $i^* \in \{1, \ldots, n\}$ chosen uniformly at random at each iteration t. The update is given as follow

$$\mathbf{\Phi}_{t+1} \leftarrow \mathbf{\Phi}_t - \eta \nabla_{\Phi_t} \mathcal{L}\left(f_{\mathbf{\Phi_t}}(x^{(i^*)}), y^{(i^*)}\right) \tag{3.9}$$

SGD is usually much faster than batch learning and often results in better solutions because the noise in the parameters updates enables the SGD to jump to new and potentially better local minima [72]. SGD is recommended for process which the function being modeled changes over time. Unlike, batch learning which does not detected changes, SGD can track the changes and yield good approximation results. Nevertheless, the stochastic updates of the parameters causes weight fluctuations around the local minimum preventing full convergence to the exact minimum. The variance of the fluctuations is proportional to the learning rate η. To reduce the fluctuations, *mini-batches* are used is practice. Batch sizes are either fixed or gradually increased starting form a small size and an update is performed for every mini-batch of p training examples.

ADAM: a first order gradient method

ADAM is an adaptive gradient method that performs a form of learning rate annealing [79]. It has become the default algorithm used across many deep learning frameworks due to its computational efficiency. The method computes adaptive learning rates for each parameter using exponential moving averages of the gradient m_t and the squared gradient v_t where the smoothing parameters β_1 ,$\beta_2 \in [0, 1)$ control the decay rates of these moving averages. The moving averages are initialised with zero vectors causing their statistics to be biased toward zero. Hence, Adam computes bias-corrected estimates $\hat{m_t}$ and $\hat{v_t}$ to counteract the biases. *ADAM* is appropriate for non-stationary objectives and problems with very noisy and sparse gradients. Given $g_t = \nabla_{\Phi_t} F(\mathbf{\Phi}_t, \boldsymbol{x}^{(k)}, y^{(k)})$, the gradient of loss function F computed on a batch of data $\boldsymbol{x}^{(k)}, y^{(k)}$ at iteration t, Adam updates the parameters as follow

$$\mathbf{\Phi}_{t+1} \leftarrow \mathbf{\Phi}_t - \eta \cdot \frac{\hat{m_t}}{(\sqrt{\hat{v_t}} + \epsilon)} \tag{3.10}$$

where $\hat{m_t} = m_t/(1 - \beta_1^t)$ is the bias-corrected first moment estimate and $m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1)g_t$ is the updated biased first moment estimate,$\hat{v_t} = v_t/(1 - \beta_v^t)$ the bias-corrected second moment estimate and $v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ is the updated biased second moment estimate. In [79], Kingma *et al.* propose default values for $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ (β_1^t and β_2^t denote β_1 and β_2 at the power t).

3.6 Bias and variance trade-Off

Any learning algorithm must be used in conjunction with procedures that maximise its ability to *generalize*. *Generalization* is the estimator capability to predict well on test data. As previously discussed, given the sample data $\mathcal{S}$ drawn from $\mathcal{D}_{x,y}$, a natural measure of the effectiveness of the hypothesis $f(\boldsymbol{x};\mathcal{S})$ as a estimator of $\boldsymbol{y}$ can be computed as the expectation of the mean-squared error (test error) over the training set $\mathcal{S}$: $\mathrm{E}\left[(\boldsymbol{y} - f(\boldsymbol{x};\mathcal{S}))^2|\boldsymbol{x};\mathcal{S}\right]$ [50]. The sources of error of the estimation $f(\boldsymbol{x};\mathcal{S})$ can be assessed via the *bias/variance* decomposition given as

$$\mathrm{E}\left[(\boldsymbol{y} - f(\boldsymbol{x};\mathcal{S}))^2|\boldsymbol{x};\mathcal{S}\right] = \tag{3.11}$$

$$\mathrm{E}\left[(\boldsymbol{y} - \mathrm{E}[\boldsymbol{y}|\boldsymbol{x}])^2|\boldsymbol{x};\mathcal{S}\right] \quad \textit{(noise)} \tag{3.12}$$

$$(\mathrm{E}_{\mathcal{S}}[f(\boldsymbol{x};\mathcal{S})] - \mathrm{E}[\boldsymbol{y}|\boldsymbol{x}])^2 \quad \textit{(squared bias)} \tag{3.13}$$

$$+ \quad \mathrm{E}_{\mathcal{S}}\left[(f(\boldsymbol{x};\mathcal{S}) - \mathrm{E}_{\mathcal{S}}[f(\boldsymbol{x};\mathcal{S})]\right]^2 \quad \textit{(variance)} \tag{3.14}$$

where $\mathrm{E}[\boldsymbol{y}|\boldsymbol{x}]$ is the expected estimator. Detailed derivation of the *bias/variance* decomposition can be found in [50]. As can be seen in Eq. 3.13 and Eq. 3.14 respectively, the bias term measures the expected deviation of an estimator from the true value of regression function. It is a persistent error that the learning algorithm is expected to make when trained on a sample of fixed size n. The variance term provides a measure of the deviation from the expected estimator value that any particular training data is likely to produce. It is the uncertainty of the predictor. Hence, knowing that a finite sample $\mathcal{S}$ is used to compute the regression estimator, the model is intrinsically uncertain as the estimate may differ with a different training sample $\mathcal{S}^{''} \subset \mathcal{D}_{\mathcal{X},\mathcal{Y}}$. Also, as the model complexity increase, the estimator is prone to overfit the training sample which increase the model's variance. It exists a trade-off between the bias and variance contributions to the estimation error, that gives minimum expected test error [63].

As the size of training samples grows, the convergence in probability that model fit converges to the true function holds. This is called *consistency*, it is ensures that the bias error diminishes as the training sample. Consistency is formally stated by

$$\lim_{n\to\infty} f(\boldsymbol{x};\mathcal{S}_n) \xrightarrow{\text{p}} \text{E}[\boldsymbol{y}|\boldsymbol{x}] \tag{3.15}$$

If *optimal performance* and *low bias* is relevant to deep-learning methods, it comes at the price of high variance [54] as the model complexity increases with larger number of hidden units. A key technique for controlling the bias/variance trade-off for noisy problem is to select the size and architecture of the network such that the model stay stable under small variation of the training data. In general, model that presents a large variance has high sensitivity to training data noise. In the other hand, a large bias term means that the function $f(\boldsymbol{x};\mathcal{S})$ one wish to approximate modelled does not belong to the family of function that the favour of other models in F.

3.7 Regularization

Regularisation techniques are designed to reduce test error and enhance the generalisation capability of deep models. In principle, generalisation increases with the size of the training set, the larger the training set, the better the generalisation. Thus, regularisation methods are designed to improve generalisation with fixed training sets. Although in [125], explicit and implicit regularizers are found to contribute only partially to DNN's improve generalisation performance, there is a vast interest in the deep learning community in finding efficient strategies that identify suitable learning stopping criterion.

There are various methods to prevent deep models overfitting the noise of the data: regularisation techniques(i.e. weight decay, dropout), data augmentation, ensemble training and early stopping are among the most known. Regularisation techniques encompass all procedures that 1) smooth the cost function by introducing curvature in low curvature region 2) reduce variance by introducing extra bias in the cost function. A common regularization technique replaces the training error $\mathcal{L}_{train}$ with $\mathcal{L}_{train} + \lambda R(\boldsymbol{\phi})$ where $R(\cdot)$ is a function that penalizes overly complex models. A common choice for $R(\boldsymbol{\phi})$ is $\|\boldsymbol{\phi}\|^2/2$. Ultimately, regularisation aims to reduce model variance at the cost of increasing the bias error; however, increasing generalisation performance decrease the speed of learning. This is particularly the case with regularisation technique such as early stopping. *Early stopping* is widely used because it is simple to implement. The method uses the *validation data* to estimate the *generalization error* during training. The validation set is never used for weight adjustment, and a stopping criterion determines when to stop the training process. The optimisation continues until the validation error has not improved by a least a tolerance value *tol* during several iterations n_{iter}. In any case, a prior good model selection (i.e. model architecture) via cross-validation contributes equally to improve the generalisation capability of a model. In this work, we use early stopping.

3.8 Inputs representation

Neural networks are robust to the choice of inputs representation [56]. However, standardising the components of the input vector improves performance by putting the input values in a range more suitable for standard activation functions. It turns out that when the inputs are not normalised, the mean of the inputs often makes the largest eigenvalue of the Hessian even larger. Normalising the inputs reduces the largest curvature, and makes the Hessian better conditioned. Also, by restricting the input vector values to unity variance using a z-score scaling, the main assumption that is taken in the *Xavier* initialisation discussed in the following section is guaranty.

3.9 Weights initialization

The initialisation of networks' weights can have a significant effect on the training process. A good weights initialization procedure is expected to ensure that activation function outputs do not induce 'vanishing or exploding' gradients during backpropagation. Gradients that are not too small or too large compromise the convergence of the training algorithm. The multiplicative effect of weights matrices through layers causes the vanishing or exploding gradient effect that is also the main challenge encountered in training RNNs. The problem is discussed at length in [124] and will be reviewed in a later section. It was found that the 'vanishing/exploding' gradient problem can be avoided if the distribution of the outputs of each neuron has a variance close to 1. Prior to the wide adoption of ReLU, the *Xavier initialization* proposed in [52] was widely used since it favours training algorithms to reach faster convergence. In Xavier initialization, layer weights are randomly drawn from a uniform distribution $\phi_l \sim \mathcal{U}(0, \frac{1}{\sqrt{N_{l-1}}})$ where N_{l-1} is the size of the previous layer. In [64], the *He initialization* was proposed to address converging ReLU activation function nonlinearities preventing very-deep network to converge at all. He's initialization overcomes the shortcoming of Xavier initialisation by setting the variance of weights initialisation as $\phi_l \sim \mathcal{N}(0, \sqrt{\frac{2}{N_l}})$.

3.10 Long Short-Term Memory

LSTM is a family of RNNs which are powerful devices designed for tasks such as *natural language processing*. The grounding foundation behind the RNN structure was born in the '80s when machine learning researchers expressed the idea of sharing parameters across different parts of the model to extend and apply the model to examples of different length and generalize across them [54]. [45, 83]. They use iterative function loops to store infor-

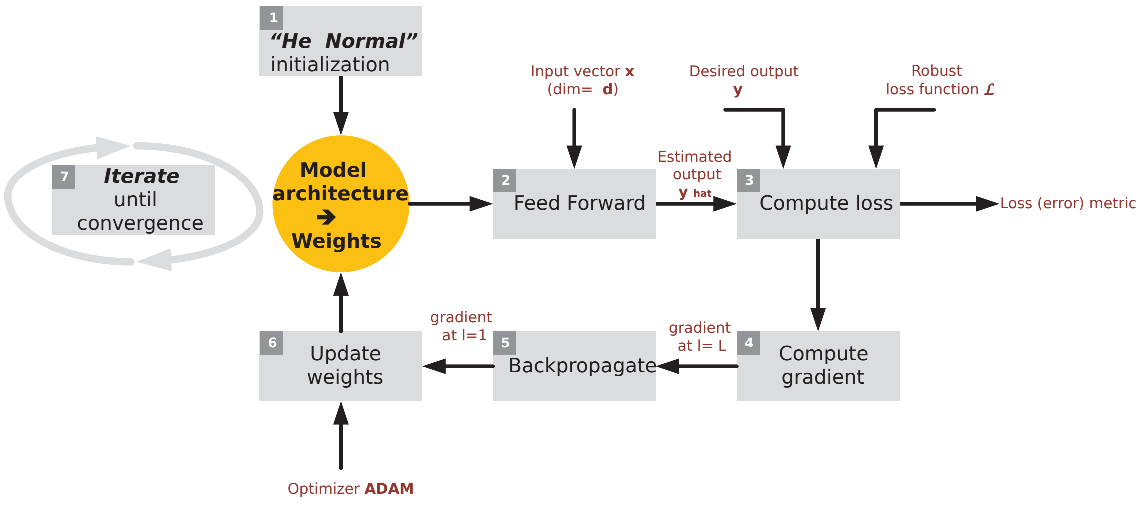

Figure 3.3: Training process block diagram

mation. They have proven to perform well at *sequences prediction* and tasks which involve sequential inputs. RNNs process an input sequence $\boldsymbol{x}_t$ one element at a time, maintaining a state vector that contains an implicit historical of all the past timesteps of the sequence in their *hidden units* $\boldsymbol{h}_t$. They differ from linear dynamical systems and Hidden Markov Models (HMMs) in that 1) they are deterministic models 2) they have **distributed** hidden states that allow them to store lots of past information efficiently 3) they are nonlinear dynamics systems. In Fig 3.4, a simple RNN is represented; on the left-hand side, it receives on its input the vector $\boldsymbol{x}_t$ at timestep t of the sequence. Its output that represents the state in which the RNN is at a timestep t causes neuron activities to reverberate as time passes, it is back-fed unto its input. On the right hide the RNN is unfolded across time, the weighs matrices for the inputs are $\boldsymbol{W}_x, \boldsymbol{W}_y$ and $\boldsymbol{W}$ the weighs matrices for the inputs, outputs and hidden states respectively. The power of RNNs architectures holds in this idea of sharing parameters across different parts of the model in order to extend and apply the model to examples of different length and generalize across them [54]. They selectively summarize an input sequence in a fixed-size state vector via a recursive update. RNNs develop a layer at each timestep where weights are shared across time. The state

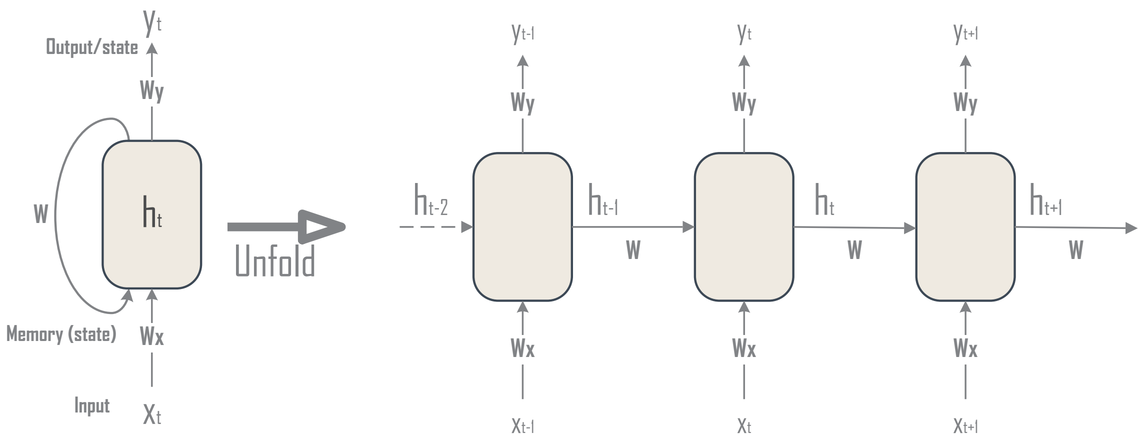

Figure 3.4: Simple Recurrent Neural Network

vector $\boldsymbol{h}_t$ and the output vector $\boldsymbol{x}_t$ equation of a RNN are given as

$$\boldsymbol{h}_t = \boldsymbol{W}\boldsymbol{\psi}(\boldsymbol{h}_{t-1}) + \boldsymbol{W}_x\boldsymbol{x_t} \tag{3.16}$$

$$\boldsymbol{y}_t = \boldsymbol{W}_y\boldsymbol{\psi}(\boldsymbol{h}_t) \tag{3.17}$$

However, RNNs have proved to be difficult to train on problems with long and complicated sequences for which they have been designed in the first place. Since RNNs potential has not been realized with their standard architecture, methods that address the difficulty of training RNNs have been proposed. The vanishing/exploding gradient problem, which causes the RNN to fail during the network training process, was investigated by Hochreiter *et. al* in [70] who proposed the vanilla LSTM block illustrated in Fig.3.5. The block features three gates (input, forget and output), a single Constant Error Carousel (CEC) cell, and peephole connections. The output of the block is recurrently connected back to the block input and all of the gates. Since LSTM has become the state-of-the-art model for a variety of machine learning problems such as handwriting recognition [56].

The output goes to every unit in the next layer. The recurrent output goes to this memory block and every other memory block in this layer. All inputs and recurrent inputs shown are the same signals (same input goes to the memory block and to the three gates).

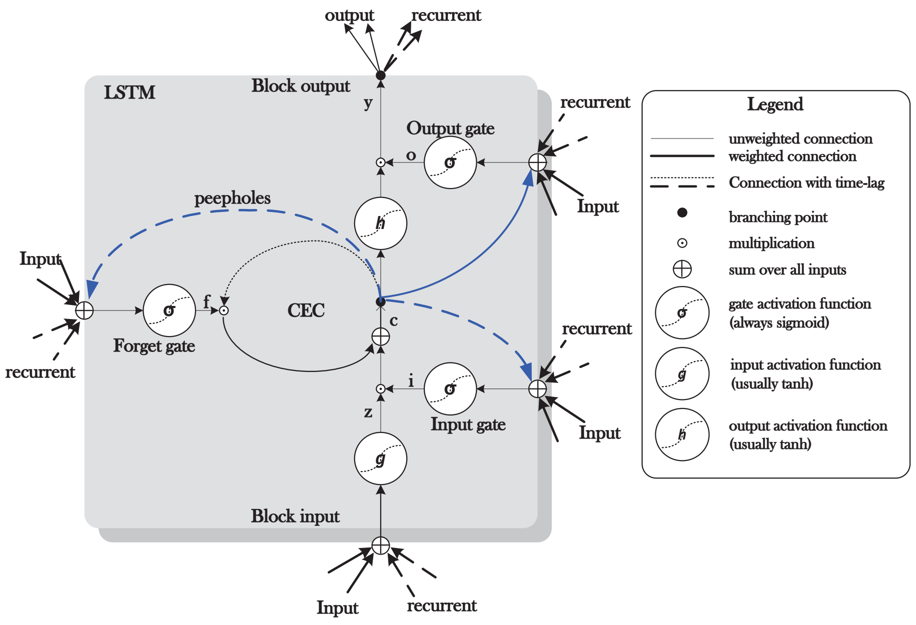

Figure 3.5: Long-Short Term Memory [57]

The state of every cell is updated in an additive way

$$\boldsymbol{z}^t = g(\mathbf{W}_z x^t + \mathbf{R}_z y^{(t-1)} + \boldsymbol{b}_z) \qquad \textit{block input} \qquad (3.18)$$

$$\boldsymbol{i}^t = \sigma(\mathbf{W}_i x^t + \mathbf{R}_i y^{(t-1)} + \boldsymbol{p}_i \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{b}_i) \qquad \textit{input gate} \qquad (3.19)$$

$$\boldsymbol{f}^t = \sigma(\mathbf{W}_f x^t + \mathbf{R}_f y^{(t-1)} + \boldsymbol{p}_f \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{b}_f) \qquad \textit{forget gate} \qquad (3.20)$$

$$\boldsymbol{c}^t = \boldsymbol{i}^t \odot \boldsymbol{z}^t + \boldsymbol{f}^t \odot \boldsymbol{c}^{(t-1)} \qquad \textit{cell state} \qquad (3.21)$$

$$\boldsymbol{o}^t = \sigma(\mathbf{W}_o x^t + \mathbf{R}_o y^{(t-1)} + \boldsymbol{p}_o \odot \boldsymbol{c}^t + \boldsymbol{b}_o) \qquad \textit{output gate} \qquad (3.22)$$

$$\boldsymbol{y}^t = \boldsymbol{o}^t \odot h(\boldsymbol{c}^t) \qquad \textit{block output} \qquad (3.23)$$

where **W** are rectangular input weight matrices, **R** are square recurrent weight matrices, ***p*** are peephole weight vectors and ***b*** are bias vectors.

3.11 Summary

Two of the most appealing deep neural network architectures used for deep regression, the FDNN and the LSTM, and their main components are described in this section.

The design of both architectures require making choices of the type and the number of neurons (activation function) and the number of layers. Although the design of deep neural networks is heuristic, it exists some level of theory which can assist the practitioner in making better choices. Deep neural networks learn by minimising a cost function for a particular set of examples, the most successful learning approaches are categorised as gradient-based learning methods. The section discusses the strategies used for improving the error minimisation process while maximising the deep neural network's ability to generalize. The effect of the bias/variance trade-off on the generalisation capability of deep learning methods are explained. Backpropagation is the learning algorithm used to train deep neural networks. The issues related to its convergence are summarised in this section; the practical tricks used to improve the performance of the backpropagation learning algorithm (i.e. regularisation, input representation, weights initialisation) are discussed.

Chapter 4

Data cleansing

'Garbage in, garbage out'

4.1 An overview

The analysis of large real-world data sets takes an ever-increasing role in the modernisation of electricity networks. Bad data quality asphyxiates innovation on some occasions, adding uncertainties at all stages of analytic procedures. Errors and anomalies in real-world data can lead to data analytics erroneous outcomes conducting users in making non-effective decisions that ultimately increase operating costs and beget unhappy customers. Data cleansing is an attempt to address data quality issues by detecting and correcting errors in data sets. Also referred to as data scrubbing or data cleaning, it is regarded as the essential first step in data analytic techniques [22].

In [103], data cleansing relates to the methods performed on data to enhance the quality and reliability of the data. Fox *et al* in [48] propose four quality dimensions for data: accuracy, completeness, consistency, and timeliness. In most applications, data cleansing procedures can only address the completeness and consistency dimensions.

Identifying the root causes of error is difficult; the cause of errors can be both simple and complex. Since there is no substitute for high-quality data, most efforts should focus on addressing data quality at the moments of data creation. Processes that fully embrace data quality enhancement must ensure the data quality persistence, which requires dedicated *data quality programs* planned at business level [93].

Instead, practitioners are often forced to spend a significant amount of time in attempting to understand and address anomalies and errors found in the data. Data cleaning is a labour-intensive and time-consuming task for the following reasons: 1) to select a suitable cleaning method is not trivial 2) to generalise or automate a cleansing procedure is challenging, 3) there is a risk to introduce new errors in the data. The definition of data cleansing is strongly dependent on the process under analysis. Since data cleansing activities require specific domain knowledge and expertise, errors detection and correction techniques are intrinsically either manual or semi-automatic.

Most data cleansing procedures incorporate domain knowledge and statistical techniques. Domain knowledge contributes to setting rules or constraints that the data must satisfy. Statistical techniques are used to screen the data, identify patterns, detect inconsistencies and outliers, and, eliminate contamination. A comprehensive data cleansing procedure defines error types, identifies and corrects the uncovered errors and, measures improvement in the data quality. Data cleaning operations encompass data exploration, data formatting, missing values imputation, eliminating duplicates, and outliers detection [93].

4.2 Missing Values Imputation

The quality of knowledge extracted from empirical data is dependent mainly on the quality of the data. Samples with missing values have a significant negative impact on the accuracy of predictive analytics. Missing values vanish due to a multitude of reasons: human errors, storing capacity restriction, malfunctioning data collection application, defective hardware, system errors, and communication interference or breakdown. Given the volume of data generated in modern electricity networks, missing data can go unnoticed for long periods; if not addressed, missing data could result in severe biases in the analyses.

Standard computational intelligence techniques such as neural networks fail to process input data with missing values. Hence, missing observations is a major obstacle to the design of predictive models. The problem of missing data imputation in time series has been vastly researched in microeconomics, and it was established that an understanding of the process of data missingness is required if reliable imputation is to occur. The treatment of missing data is independent of the learning algorithm, but it relies heavily on the mechanisms that lead the observations going missing. Missingness analysis can assist in selecting a suitable missing imputation technique.

In [89], Little and Rubin first established a classification system to differentiate between missingness mechanisms in survey data. The classification aims to describe the relationship between the observed data and the probability of missing values to determine if the missingness mechanism is Ignorable or Non-Ignorable. Three basic missingness mechanisms were identified: Missing Completely at Random (MCAR), Missing at Random (MAR) and Missing Not At Random (MNAR). Let be $\mathbf{y} = \{y_t\}_{t=1}^{n}$, a vector of random variables for the complete data which contains both the observed data $\mathbf{y_{obs}}$ and the missing values $\mathbf{y_{miss}}$ with probability density function $f_{\boldsymbol{\theta}}(\boldsymbol{\theta})$. The vector of parameters $\boldsymbol{\theta}$ is unknown and needs to be estimated. Let us define $\mathbf{M} = \{m_t\}_{t=1}^{n}$, a vector of binary variables that indicates whether the corresponding value in the complete data is missing or not. The random variable $\mathbf{M}$ can be assumed to be generated by a model described by a density function $f_{\phi}(\mathbf{M}|\boldsymbol{\phi})$ indexed by the vector of unknown parameters $\boldsymbol{\phi}$.

In practice, it is impossible to establish $f_{\phi}(M|\boldsymbol{\phi})$ with exactitude but the conditional distribution $f(M|Y, \boldsymbol{\phi})$ is used in the literature to identify the missing data mechanisms. Data are said to be missing completely at random (MCAR) if

$$p(M|\mathbf{y_{obs}}, \mathbf{y_{miss}}, \phi) = p(M|\phi) \tag{4.1}$$

Here, missingness of data does not depend on either the observed values $\mathbf{y_{obs}}$ or the unobserved values $\mathbf{y_{miss}}$. In most case, particularly in time series data logging process, it is naive to assume the data being MCAR. The second type of missing assumption is Missing at Random (MAR). Under MAR mechanism, the probability the variable **M** is missing depends only on the values of the observed data which translates as

$$p(M|\mathbf{y_{obs}}, \mathbf{y_{miss}}, \phi) = p(M|\mathbf{y_{obs}}, \phi) \tag{4.2}$$

Here, the likelihood of missing observation is entirely scrutinized by observed variables. If the probability than an observation is missing depends on observed and unobserved values, the data are said Missing Not At Random (MNAR). Under this assumption the probability that an observation is missing depends only on the values of the unobserved data which translates as

$$p(M|Y_{obs}, Y_{miss}, \phi) = p(M|Y_{miss}, \phi) \tag{4.3}$$

The MNAR mechanism is unlikely to happen in distribution feeder data collection; the contrary would indicate a malicious activity which would need to be further investigated. We understand domain knowledge is of a substantial tool in the diagnosis of the randomness mechanism of missing values. In practice, it is challenging to determine the mechanism that led to the missing data, but if missing data can be assumed to be MCAR or MAR, then the missingness mechanism is said to be ignorable. Most of data imputation techniques found in the literature lie under the assumption that the missing data are MAR, and this will be our prior assumption in this work.

In practice, it is challenging to determine the mechanism that actually led to the missing data. However, if MCAR or MAR holds, the missingness mechanism is said to be Ignorable meaning missing data and observed data come from the same distribution and a model can be derived from the observed data to impute missing data. In the context of statistical inference, imputation means *filling in the data.* Most of data imputation techniques found in the literature build their very foundations under the ignorability assumption, and it is our prior assumption for the rest of this work. Imputation methods for time series have been researched, and algorithms such as expectation maximisation, multiple imputations, cold deck and hot deck imputation techniques are presented and discussed in [39, 40, 89, 90, 94]. The following introduced the imputation techniques that we use to handle missing observations in the MV/LV feeders series.

As standing assumption for the rest of this work, the missingness mechanism in feeders load data is assumed to be Missing at Random (MAR) as per Little and Rubin [89] classification system. Under this assumption, the precise mechanism underlying the missing data can be ignored, and missing data and observed data are assumed to come from the same distribution. The imputation techniques used in this work to handle missing observations in the MV feeder time series are listed below.

4.2.1 Simple imputation: mean substitution

The simplest univariate imputation technique is mean substitution. Mean substitution is a heuristic method that substitutes missing observations by the unconditional mean of the observed data. Mean imputation is naive and should be cautiously used since it can severely distort the empirical distribution of the data and insert bias in analytic or statistical inference especially if the data is nonstationary [49, 90].

4.2.2 Hot Deck imputation: k-Nearest Neighbour

Hot Deck class of imputation techniques is widely used because it makes only minimal assumptions on the data. The procedure replaces missing values (recipient) by values extracted from responding covariates (donors) that most resembles the recipient. The algorithm widely used for matching donors to recipients is the k-Nearest Neighbour (kNN). The imputed value is either a single observation drawn from another variable (1-NN) or the weighted average of k observations drawn from k variables (k-NN) [16]. In standard kNN

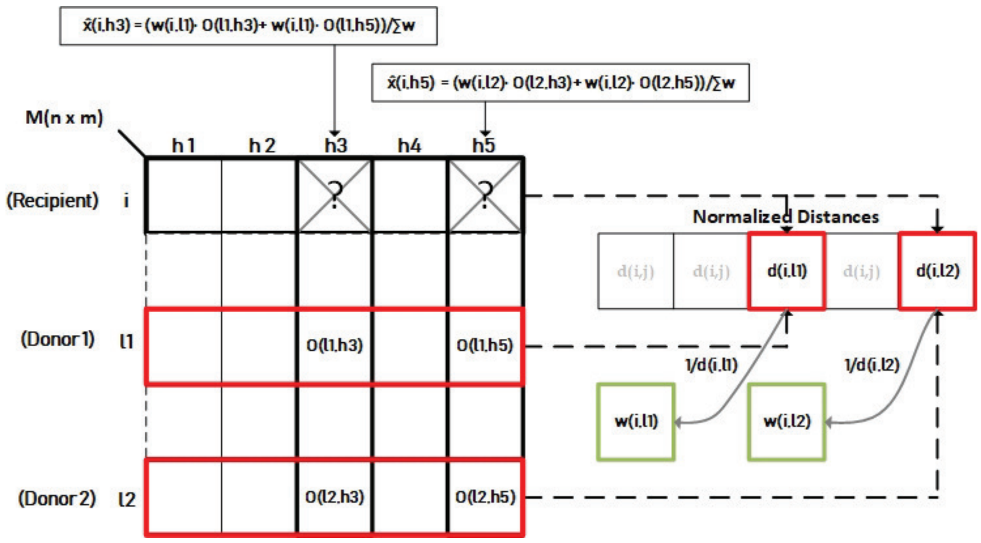

Figure 4.1: k-NN imputation technique illustration for $k = 2$. h_m = feeders, i, l_k = time

imputation, the similarity between recipient and donors is measured with the Euclidean or Manhattan distance [111]. In this work, the `optimistic` knn algorithm available in the python's `fancyimpute` library was used to impute the feeders' data. The imputation procedure is illustrated in Fig. 4.1. Feeder data sets are represented as a matrix $M \in \mathbb{R}^{n \times m}$, where each column is a time series of n regularly spaced measurement values. There are m such columns (features), one for each feeder. Missing observations are imputed on a row-by-row basis; the k-NN algorithm selects each row's k nearest neighbours (i.e. times with similar measurements) and computes their weighted average to impute the missing observations. The nearest neighbours of the ith row are identified as being the k rows with the smallest normalized Euclidean distances

$$d(i,j) = \frac{1}{n_{0(i,j)}} \sum_{h \in \Omega_{i,j}} (O_{i,h} - O_{j,h})^2 \tag{4.4}$$

where $O_{j,h}$ is the observed value for feeder h at timestamp j and the set $\Omega_{i,j}$ is defined as the set of common features between i and j (i.e. the feeders for which data is available at both timestamps) with $n_{0(i,j)} \doteq |\Omega_{i,j}| \leq m$. Note that $d(i,j) = d(j,i)$ holds for all $i, j \in \{0, \ldots, n-1\}$. Let us define the set $\mathcal{D}_i^k$ of k indices with the smallest distance from i as $\mathcal{D}_i^k \doteq \{j \neq i : d(i,j) \leq d_i^k,\ j = 0, \ldots, n-1\}$ with $d_i^k = \min_{d_x}\{d_x \ : \ \left(\sum_j \mathbb{I}_{d(i,j) \leq d_x}\right) \geq k\}$ in which $\mathbb{I}[.]$ denotes the indicator function. The imputed value $\hat{x}_{(i,h)}$ of feeder h at time stamp i is given by

$$\hat{x}_{(i,h)} = \frac{\sum_{l \in \mathcal{D}_i^k} w_{i,l} O_{l,h}}{\sum_{l \in \mathcal{D}_i^k} w_{i,l}} \tag{4.5}$$

$$w_{i,l} = \frac{1}{d(i,l)} \tag{4.6}$$

The weight $w_{i,l}$ controls the influence of the observed values $O_{l,h}$ in the computation of $\hat{x}_{(i,h)}$.

4.2.3 Kalman smoothing imputation

Kalman Filter is rooted in the field of recursive optimal linear filtering of discrete-time linear processes model in state space. It is an efficient algorithm that belongs to the Bayesian filters family; it is used to estimate the state of a system from noisy measurements. Gauss-Markov and Rao-Markov theorems are the stones of estimation theory which aims to find an estimator that minimizes the variance between prior information and posterior information. The history of optimal filtering starts from the Wiener filter which is a frequency domain solution to the problem of least squares optimal filtering of stationary Gaussian process which is also its main disadvantage as it cannot be applied to non-stationary. The Kalman filter comes ten years after the Wiener filter and extend the theory of filtering to the non-stationary process. It is a closed-form solution to the linear Gaussian filtering problem which provides the basis for prediction error decomposition. Because of the linear Gaussian model assumptions, the posterior distribution is exactly Gaussian. Let $w \doteq \{y_t\}_{t=1}^{s}$, given $\mathbf{Z}_t, \mathbf{T}_t, \mathbf{Q}_t$, h_t and w, the Kalman filter provides a means to compute the conditional mean value $a_{t|t-1} = E[\boldsymbol{\alpha_t}|w_{t-1}]$ and the covariance matrix $P_{t|t-1} = \mathrm{Cov}(\boldsymbol{\alpha_t}|w_{t-1})$ of $\boldsymbol{\alpha_t}$ at each time step [7].

$$y_t = \mathbf{Z}_t \boldsymbol{\alpha}_t + \varepsilon_t, \quad \varepsilon_t \sim N(0, h_t) \tag{4.7}$$

$$\boldsymbol{\alpha}_t = \mathbf{T}_t \boldsymbol{\alpha}_{t-1} + \boldsymbol{\eta}_t \quad \boldsymbol{\eta}_t \sim N(0, \mathbf{Q}_t) \tag{4.8}$$

In theory, any time series model that fits the data well can be used as long as the number of parameters to be estimated is kept down to a manageable size. We adopt a simple local linear trend model as suggested in [61, 100]. The structural model takes the general following form:

$$y_t = \mu_t + \varepsilon_t \tag{4.9}$$

$$\mu_t = \mu_{t-1} + \beta_{t-1} + \eta_t \tag{4.10}$$

$$\beta_t = \beta_{t-1} + \zeta_t \tag{4.11}$$

where $\varepsilon_t \sim N(0, \sigma_\varepsilon^2)$, $\eta_t \sim N(0, \sigma_\eta^2)$ and $\zeta_t \sim N(0, \sigma_\zeta^2)$ are white noise disturbances mutually uncorrelated . The local trend model can be cast in state space form as follow

$$\begin{aligned} y_t &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \mu_t \\ \beta_t \end{bmatrix} + \varepsilon_t \\ \boldsymbol{\alpha}_t = \begin{bmatrix} \mu_t \\ \beta_t \end{bmatrix} &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_{t-1} \\ \beta_{t-1} \end{bmatrix} + \begin{bmatrix} \eta_t \\ \zeta_t \end{bmatrix} \end{aligned} \tag{4.12}$$

For this work, we use the Kalman smoother imputation available in `pykalman 0.9.2` library in python 3.6. The algorithm is used in four variants. The first variant consists of smoothing the entire raw univariate time series so that the filter performs two actions: it smooths the entire signal (all observed data) and interpolates the missing observations. In the second variant, the outliers are flagged and removed using the proposed outlier detection technique, and the remaining cleaned data are smoothed while all missing observations are interpolated by the smoother. In the third and fourth variants, missing values and removed outliers are replaced by interpolated values. No smoothing is performed on healthy raw data.

Chapter 5

Data exploration

'Exploring missing data is difficult, because it is an iterative process with many dead ends and often no clear answer'. [116]

5.1 General information

UK Power Networks control and monitor all aspects of the distribution networks via a ADMS. A schematic of the communication systems between the ADMS and the network's remote equipment is provided in Fig. 5.1. The ADMS communicates with primary and secondary substations and any other automated pieces of equipment installed on the LV networks. The ADMS operates on private and dedicated WAN,LAN upon which communication transactions between the FEP and RTUs and IEDs are facilitated. LANs are typically used to connect a set of hosts within the same building. RTUs are computer-based devices installed at remote sites (i.e.primary and secondary substations) that monitor and control the equipment on the network. They are the interface between the ADMS and the network's plants. RTUs gather information about analogue values, equipment status signals and alarms collected from the field. FEP and RTUs communication are governed by a Master-slave protocol. The data gathered from remote sites are transferred to the

ADMS via the FEP which continuously polls the RTUs (typically every 10 seconds), for any changes, such as alarms, analogue values and circuit breaker and switch statuses. When it is required, the FEP instructs RTUs to control the remote actuators that command the valves, the motors, the electromagnetic switches and set analogue control values. The communication between the substations and the FEP is based on a broadcast model that dispersed messages via a shared WAN, network privately owned by the DNO. The WAN is built on fibre-optic and DSL communication mediums. A high bandwidth VSAT is used as a stand-by communication system. The interoperability between RTUs and the FEP is achieved using mainly DNP3, a standard communication protocol that operates over the TCP/IP, TCP being a packet-switching protocol. At the control room, all data are centralised and stored into the real-time data historian developed by OSIsoft. The 'PI' historian records the operational monitoring data over time in a proprietary time-series database. Each data entry is known as a PI tag. Data are passed from the PowerOn Fusion server to the PI historian server via the LAN.

This chapter describes the load and the weather datasets used to train and test the forecasting models. The load data were collected from UKPN's Data Historian. Load and weather data span two years and one half starting from January 2014 and ending in September 2017. The data sets are divided into training, and testing sets with the testing set starting in May 2017. The chapter outlines the prior processes of transforming the raw data into a usable format with the intent of making it suitable for further analytics. It discusses the need to format 'naive datetime' object to time zone-aware datetime. It details the procedure that was undertaken to allocate the weather features to the feeders' time series. Finally, the chapter discusses the identification of the missing observation patterns in the feeders' time series. It proposes a direction for further investigation with regard to the cause of bad data quality.

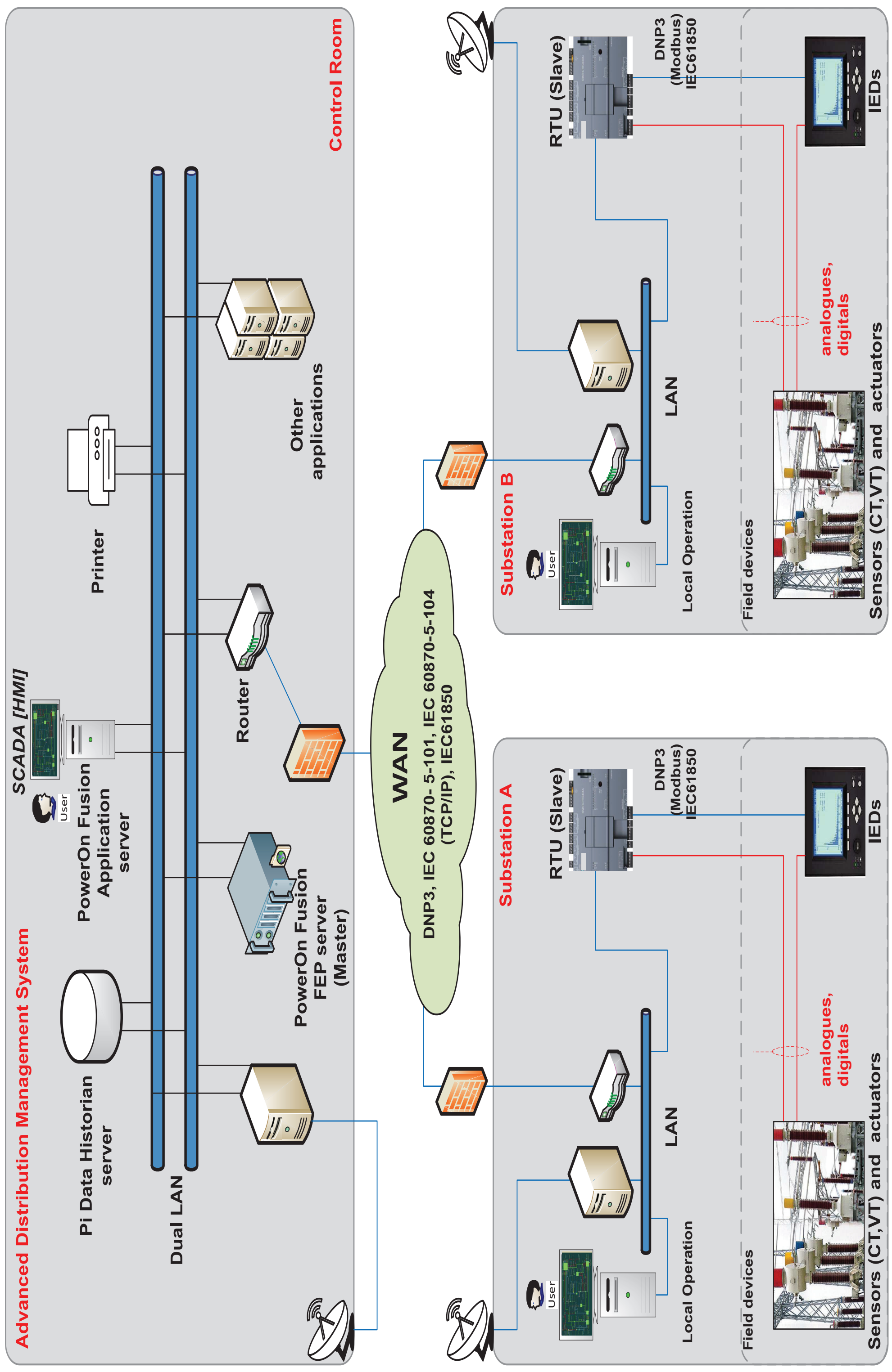

Figure 5.1: Advanced Distribution Management System Network

5.2 Load datasets

Almost three years of historical time series load data are used for this work. The data were collected from the field as *current measurements*. The feeders (from which the data have been collected) are connected to primary substations or grid substations. They feed 6.6 kV, 11 kV, 22 kV, 33 kV and 132 kV busbars. Feeders' time series were queried from UKPN's *OSIsoft* proprietary database. An initial list of 606 feeders' tags was used to query the load data. Tags are the identification numbers of feeders connected to primary and grid substations located in the East Kent area. The raw data were queried with 30 min granularity; thereafter, the half-hourly timestamps were filtered to keep only hourly observations to match the weather data granularity. Table 5.1 shows the distribution of feeders by voltage level. Some feeders are missing because their voltage level information was not available. The table indicates that the majority of the time series in the data were collected at 11 kV. This is not unexpected because it is known that the density of substations connected to the power system increases as the voltage level reaches the load centres. The training data of one feeder was randomly selected at each voltage level and plotted in Fig 5.2.

The plots are organised in voltage ascending order from top to bottom starting with 6.6 kV and ending with 132 kV. Fig 5.2 indicates that the feeders connected at 6.6 kV, 11 kV and 22 kV present similar load curve and similar features. For instance, one can notice the 'dip' in energy consumption around the winter holiday period for those feeders. The 'dips' are highlighted in pink. The 11 kV experiences a gap of missing observations; all feeders contain sporadic values that differ in amplitude to the rest of the time series. From these plots, one may anticipate that feeders connected at 6.6 kV, 11 kV and 22 kV will be relatively easy to model in comparison to the feeders connected at 33 kV and 132 kV. The 33 kV time series data is heteroscedastic and displays a major change in April 2016. The level of the data seems to have dropped as well as the 'baseload'. The plots in Fig 5.3 zoom in the 33 kV's feeder load consumption for a period of approximatively two months at a year interval. The top plot shows the load curve in July and August 2015 before

Table 5.1: Numbers of feeders by voltage level

Voltage	6 kV	11 kV	22 kV	33 kV	132 kV
No of feeders	64	433	4	41	2

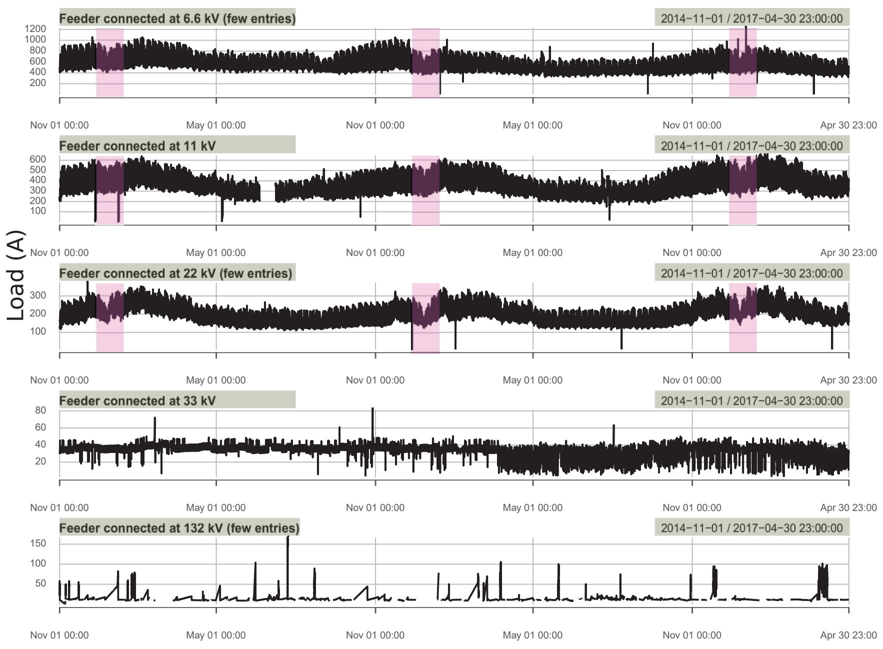

Figure 5.2: Plot of historical load data at different voltage levels

the change occurs while the bottom plot displays the load curve a year later. The shape of the load curve has significantly changed between both periods as well as the level of the time series. A general statement regarding the forecastability of each type of feeder cannot be made at this point, but the visualisation aims to highlight some of the difficulties encountered when facing large datasets with disparate or unexpected behaviours. The two 132 kV time series in the dataset was collected at the *Euro-tunnel*. The data exhibited what looks like a constant level with sporadic energy peaks, ramps, and gaps of missing values. This project will not attempt to analyse the 132 kV data further; consequently, these data have been discarded.

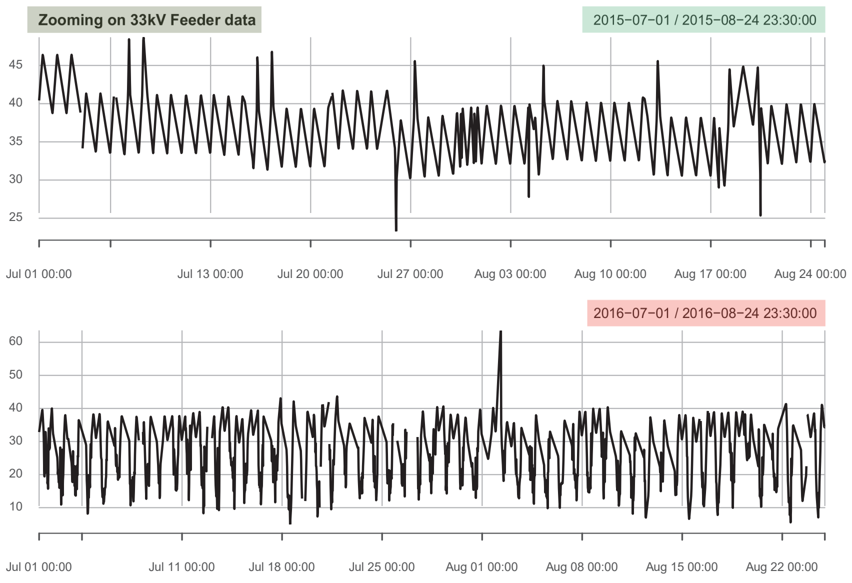

Figure 5.3: Zooming at one 33kV feeder data

5.3 Weather datasets

Observational weather data and daily weather forecast data are delivered every six hours via File Transfer Protocol (FTP) by the *Meteorological Office*. The Metropolitan office also issued one year of historical weather forecasts to train the KASM'S forecasting models. The data were collected from 11 weather stations spread across the South East of London. Weather data have hourly granularity. Forecasts for solar irradiance (W/m2), dry-bulb air temperature (C), humidity (%), wind speed (m/s) and wind direction (Deg) are pushed to UKPN's server every six hours (four times a day). Only weather *weather forecast data* were used to train the STLF models and generate 24-step ahead forecasts. Available historical data were concatenated for each substation, which gave rise to duplicated timestamps in the data. Therefore, the most up-to-date records were retained for the experiment. Each feeder was allocated a *closest weather station* using the procedure illustrated in Fig. 5.4. A list of geographical coordinates (latitude, longitude) was provided for each weather station and each of the primary and grid substations in the East Kent Area. The identification of the feeders' closest weather station required a two-stage computation procedure. During the first stage, the weather station at the vicinity of each substation was identified using

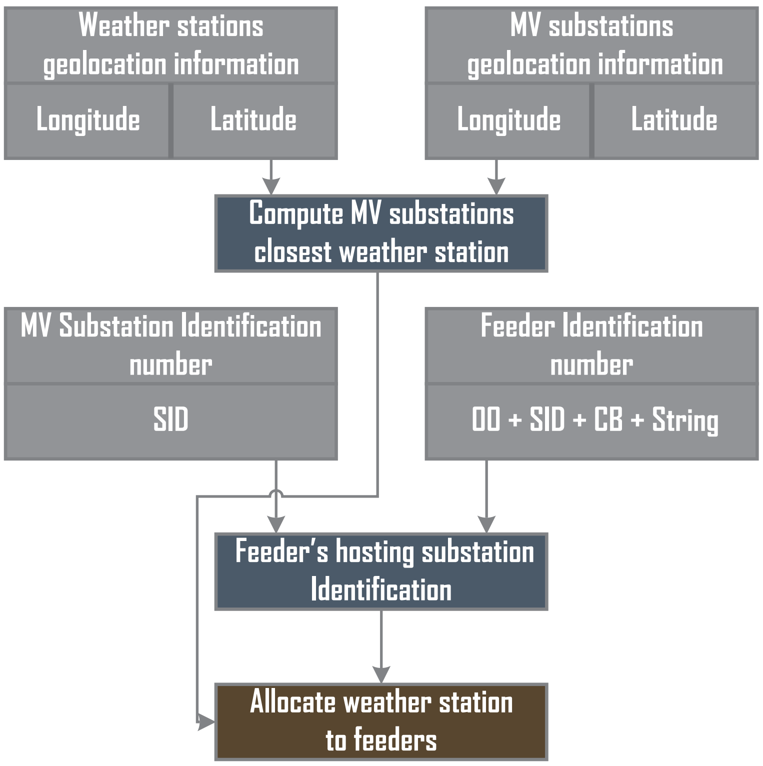
Weather stations geolocation information
Longitude
Latitude
MV substations geolocation information
Longitude
Latitude
Compute MV substations closest weather station
MV Substation Identification number
SID
Feeder Identification number
00 + SID + CB + String
Feeder's hosting substation Identification
Allocate weather station to feeders

Figure 5.4: Flow chart to assign the closest weather station to each feeder

geographical coordinate information. At the second stage, each feeder was allocated to its hosting substation by extracting the substation's identification number embedded in each feeder tag. Finally, each feeder was allocated the same closest weather station as its hosting substation.

5.4 Data manipulation

Processing data collected from the real-world can present considerable challenges if the raw data have not been pre-processed prior to the statistical inference. Processing a given data set implies one's familiarisation with the data and its entries by capturing its content and its shape through exploration. This step is time-consuming but remains essential for successful analytics.

5.4.1 Data coercion

The raw feeder data contained many unidentified strings that cast the entire data set as *object* data type, rendering the data unusable for data exploration or numerical computation. Sequence of characters such as ‘error’, ‘No data’, ‘Bad Data’, ‘No Sample’, ‘Bad Total’,‘Pt Created’, ‘Shutdown’ or ‘empty cell’ were spread across the full dataset at unknown locations. Before data wrangling, the data were *coerced* to the ‘float’ data type. From the implicit conversion of the object data type to float data type, all instances were converted to float, and the aforementioned strings were replaced by NAN.

5.4.2 Time-zone and Daylight Saving Time

Time-zone and DST are considered the most challenging problems in time series manipulation. The UKPN’s OSIsoft proprietary database returns *naive* ‘datetime’ objects. Naive ‘datetime’ objects are easy to work with but ignore some important aspects of reality. They do not contain enough information to explicitly locate themselves in relation to other ‘datetime’ objects. In addition, time-series manipulation issues arise when naive ‘datetime’ objects are subjected to DST. In practice, naive ‘datetime’ may be stored in local time or in any time-zone. With no time-zone/DST information attached to them, ‘naive’ datetime objects do not provide enough context and can be considered irrelevant, depending on the application they are used for.

UK observes DST, the mechanism by which clocks are moved one hour forward in spring and one hour backwards in autumn. The DST mechanism is a measure to reduce electricity consumption by using daylight hours more efficiently. The concept aims to better-aligning waking hours with hours of daylight. Observing DST results in having *ambiguous times* at the end of DSTs with days that are comprised of 23 or 25 hours. For example, in the UK/Eastern time-zone on the last Sunday morning on October, 01:00 BST occurs and one hour later, instead of 2:00 am the clock turned back 1 hour and 01:00 happens again, this time 01:00 UTC.

The usual strategy adopted by practitioners when they are facing time-zone and DST issues is to use non-DST-aware time-zones for both; current and historical datetime objects. Ideally, datetime information should be stored in UTC and converted back and forth for the localised user interfaces. UTC, also referred to as *standard time*, is the time standard commonly used across the world, UTC uses the measurement of a second as defined by TAI. This allows for accurate measurement of time while introducing *leap seconds*. Alternatively, timestamps should embed time-zone information by being made time-zone aware to provide users with the necessary context and useful information. An *aware* datetime object has sufficient knowledge of applicable algorithmic and political time adjustments to locate itself in relation to other time-zone aware datetime objects. Governments define the standard offset from the UTC that a geographical position follows, effectively creating a time-zone. Offsets refer to the number of hours that time-zones are from UTC. Localised times subtract the offset of its time-zone from UTC time.

For this work, the naive datetime objects of the feeders' time series were converted to UTC to avoid any DST related issues when producing forecasts. Regrettably, the conversion process did not happen smoothly since `Python` failed to process more than one DST change automatically. The presence of ambiguous multiple times in the data set was inexhaustibly failing `Python`'s *localize* method that localises naive datetime objects in a given time-zone. A series of test were implemented until a satisfying outcome was reached, following the order of this procedure:

1. **Create** time-zone object 'Europe/London'
2. **Localise** datetime objects to 'Europe/London' time-zone (disambiguate ambiguous timestamps with NAT argument)
3. **Convert** datetime objects to UTC
4. **Eliminate** duplicated timestamps and keep the first timestamp
5. **Filter** only non-NAT timestamps
6. **Resample**

To illustrate the contrast between manipulating naive datetime objects and time-zone aware ones, one propose to visualise the impact of DST change on the load curve of an 11kV feeder. In Fig 5.5, the feeder's load pattern is plotted for the week preceding and succeeding the summer' and winter' DST changes of the year 2015. There are two plots for each of the change: the top plot uses naive datetime objects while the bottom plot uses UTC. In Fig 5.5, the UTC plots to display a shift in the energy consumption for each season which does not appear in the naive plots. Thus, the UTC plots convey valuable information that is overlooked by the naive plots. Focusing on UTC plots from now on, the plot in Fig 5.5 indicate that summer and winter DST effects are consistent with daily activities in which the energy consumption shifts one hour backwards in summer and one hour forwards in winter. DST also dampens evening peak demand.

Further analysis of the plots reveals that the energy consumption falls by approximatively 10% across the entire week that follows the DST summer's change. At first, this drop in energy consumption might be associated with a decrease in the usage of electricity for lighting. This decrease in electricity consumption is prominent throughout the weekend, maybe due to business activities cease. However, the top plots (summer plots') exhibit an energy demand reduction around midday. This depletion in the energy usage could be the consequence of the PV generation that supplies the load locally (downstream the 11kV busbar) which modifies the standard load curve to what is referred as the *duck curve*. In a matter of fact, there are two impacts of DST that are indisputable: 1) the morning activities' shift one hour backwards and 2) the peak load reduction in the evening. Nevertheless, it is contentious to assert with confidence that the 10% drop in energy consumption (for the week understudy) are due to DST effects only. Because the production of PV generation is higher during summertime, both effects are coupled, so they both contribute to some proportion to potential energy savings.

One can extend the analysis to most of the feeders by computing the median of the load across all 433 feeders connected to an 11kV busbar. Using the median to represent the general tendency is a natural choice to discard outlying hourly observations. Data used for this analysis is the raw data, meaning that missing values have not been imputed at this stage. The median load is plotted for the two weeks that follow the DST changes in Fig 5.6. The analysis is done for the years 2015, 2016, and 2017. The plots show that all aforementioned remarks derived for one feeder can generalise to all feeders. These observations relate to the load curve shifting, the load curve changing to duck curve at midday and the energy consumption damping in the evenings. The plots in Fig 5.6 also show that the median load has dropped by 9.7% between 2015 and 2017. It is most likely that this load reduction is caused by the increasing utilisation of DERs downstream of the 11kV busbars. One side effect of DERs adoption is known as the *load masking* effect. It contributes to challenging some of the conventional voltage control schemes such as *load drop compensation (LDC)*. Because a share of the load is supplied locally or *behind the meter* rather than through the substation, some of the total load information is lost. By providing inaccurate load information to the automatic voltage control device, the load masking effect can cause the automatic selection of non-optimal tap positions resulting in non-compliant end-users voltage excursions.

Although consumers tend to shift backwards their energy consumption pattern during the mornings following the DST change, the plots in Fig 5.6 display a natural re-synchronisation of the morning activities happening later in the week during the first weekend. The plots also show that DST reduces not only the evening's load peak but also its duration. The median load plot for the second week that follows DST, provides further information with respect to the 11kV feeders' load behaviour. Generally, a further decrease in energy consumption occurs during the second week in 2015. That year, the median energy consumption fluctuate during the week, which represented approximatively 10% energy decrease on average. In 2016 and 2017, the second week's load curve behaviour varies across the week. The contrast between the year 2015 and the years 2016 and 2017 could be the result of different weather conditions. This hypothesis can be verified by looking at the daily load curves around midday; the presence of a *duck curve* should provide unambiguous information on weather conditions. One sees that larger feeders' load depletion manifests alongside with significant 'dips' around midday. This only confirms that the intraday curve is very much influenced by the uncertain level of PV generation production. The plots in Fig 5.6 cast certitudes that energy conservation during the periods of the day affected by the clock shifting is real. However, with the increasing level of PV generation connected to the system, the quantification of energy savings caused by DST is made even more challenging. Also, this suggests that up-to-date studies based on empirical evidence should be considered to better inform on nowadays DST's effects on energy consumption.

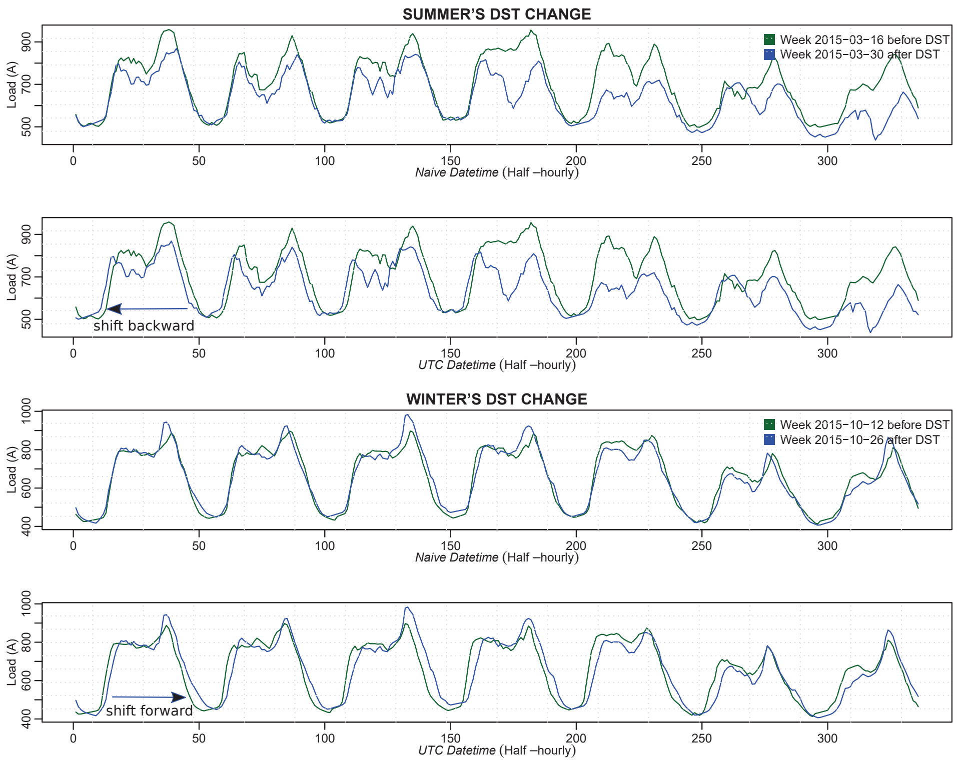

Figure 5.5: DST effect on the consumption of one 11kV feeder in 2015

In countries like Australia, India and Argentina, fairly recently studied were conducted to assess DST's effects on electricity consumption. Those studies were based on the analysis of empirical evidence. The evaluation of energy savings due to DST's effects were mostly computed using a common approach known as the difference-in-differences (DID) technique [60, 78, 82]. The method uses observational data collected from experiments or trials to partition the electricity demand into treatment group (periods during which DST is implemented) and control group (periods during which DST is not implemented). Treatment and control groups should be taken from the same country or neighbouring regions with similar features (i.e. load curve, weather conditions, holidays). The empirical evidences from the analysis published by Kellogg *et al.* in [78], Kotchen *et al.* in [82] and Hancevic *et al.* in [60] are contradictory with the expected outcomes by which DST occasioned energy savings. In [82], it was estimated that DST increases the energy consumption by 1% in India. In Argentina, the article in [60] reveals an increase in the electricity consumption in the order of 0.4% to 0.6%, the report estimated at $14.5 million USD in average the yearly cost of extra generation. In [78], Australia conducted an empirical study taking advantage of the extension of DST to facilitate the 2000's Olympic Games in Sydney. The study concluded that the potential DST's extension over an additional month would not reduce overall electricity consumption.

Similarly, the UK's parliamentary members debated on a three-year trial period of *single/double summer time (SDST)* in 2010. In fact, this is an ongoing debate since the early '90s. SDST would set clocks one hour forward from UTC in the winter and by two hours in the summer as those in the majority of Western European countries. This debate motivated the preparation of independent consultancy reports mostly based on literature reviews [15]. In the UK, most of the studies were conducted from simulations. The rare time when DST's impacts could be measured based on observational data was from the trial period in 1968 to 1971 during which British Summer Time (BST) was extended through the winter. British Summer Time (BST) can be traced back to the Summer Time Act 1916. After a few periods of deviation, DST change has been in continuous operation since 1972. At the time, *Her Majesty's Stationary office* found that under DST, mornings' energy consumption increased by 2.5% approximatively and decreased by 3.0% in the evenings', resulting in 0.5% energy savings [67]. In 2010, a study based on regression analysis was conducted by Hill *et al.* in [66] to assess potential energy savings in extending BST year-round. The analysis used National Grid data from 2001 to 2008 to train a support vector regression (SVR) model that estimates energy consumption reduction during winter. The analysis concluded that an extension of DST over the entire year would approximatively damp winter's energy consumption by 0.30% with daily peak load reduction ranging up to 4%.

Between 4 July and 16 August 2018, the European Commission conducted a public consultation on a proposal to end the seasonal clock changes. The consultation generated 4 million of replies[1] for which 84% was in favour of the revocation of Directive 2000/84/EC. The motivation behind the proposal to abolish DST changes is to ease of the European's internal market. Based on studies conducted across the Member States, the European Commission communicated that the overall energy savings caused by summer-time were marginal. An impact assessment of ceasing the DST change revealed that transitional costs would be generated as IT systems and smart technologies would have to be reprogrammed and reconfigured. The European Directive will be implemented in 2021, in which case the last transition would occur on the last Sunday of March 2021. Each Member States will decide whether they apply summer-time or winter-time year-round. The European Commission has planned to assess the impacts of the new Directive by 31 December 2024. Thereof, there will be opportunities for academics to conduct studies on the effects of DST on energy savings based on real observational data, providing the UK remains a Member State. Thus, the difference-in-difference approach could be used with the new data made available.

5.5 Missing values investigation

In this section, an attempt is made to find the cause(s) of missing observations in the load data. A full investigation would require to have: 1) a good domain knowledge in data transmission networks and data communication protocols, 2) physical access to RTUs, IEDs, communication routers and switches, 3) full access to the database, particularly to past event logs. Hence, the analysis of the missing value is carried out with objective not to assert with exactitude the sources of missing data but rather, to discard the most improbable causes and provide plausible lines of approach for further investigations.

[1]https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:52018PC0639&from=EN

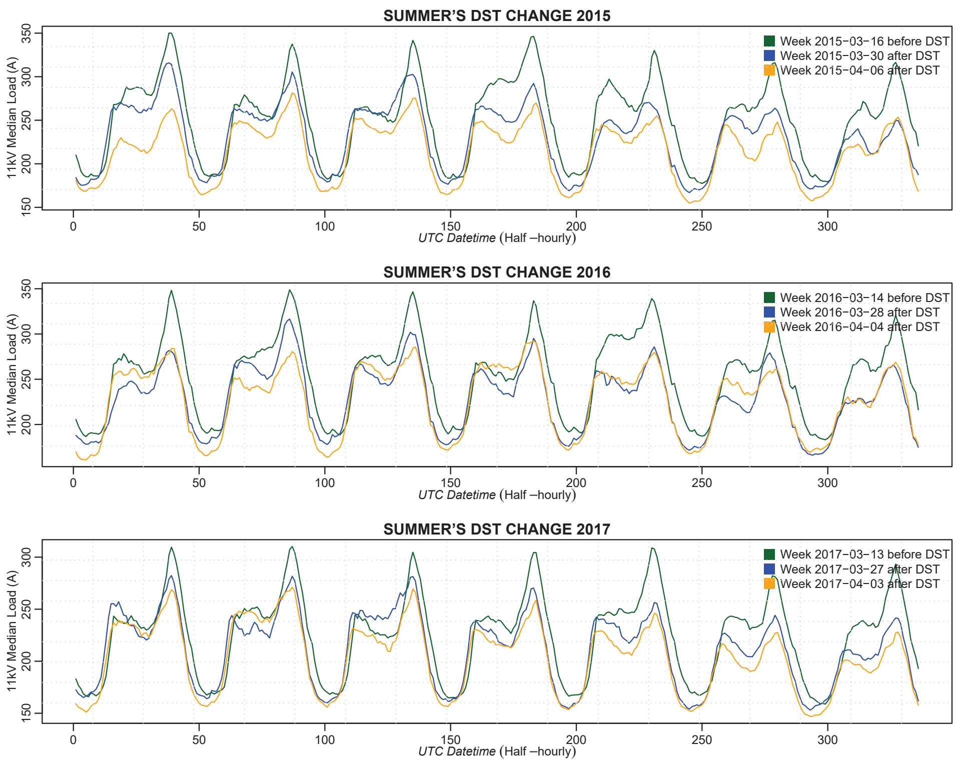

Figure 5.6: DST effect on the median consumption of 433 11kV feeders in 2015.

5.5.1 Vertical analysis

The raw data sets contained 606 variables and 25560 hourly observations per variable. The missing data are represented as a missingness matrix $M \in \mathbb{R}^{n\times m}$ displayed using a heatmap in Fig. 5.7. The matrix uses a 'shadow matrix' bounded column-wise to the raw data. The matrix is computed using R package `naniar` for which each data point d with ith row and jth column, the value m_{ij} is given by

$$m_{ij} = \begin{cases} 0, & \text{if } d_{ij} \text{ is missing} \\ 1, & \text{if } d_{ij} \text{ is observed} \end{cases} \tag{5.1}$$

The missingness matrix covers the load data set for the period starting the 01 November 2014 and ending the 30 September 2017. Each column is a time series of $n = 25560$ regularly spaced measurement values; there are $m = 606$ such columns, one for each feeder. Missing observations are displayed in black; observed data are displayed in grey.

The exploration of missing observations in the data sets can be driven vertically and horizontally.

The vertical exploration of the matrix is concerned with the analysis of missing measurements for individual feeders. In Fig. 5.7, the feeders highlighted in the red colour present a large number of missing measurements. There are five feeders' time series for which all values are missing while only one feeder (connected at 33 kV grid substation) has a complete data set. Those feeders are either not supplying to a load or might have been decommissioned. The overall distribution of feeders' percentage missing values are displayed in Fig. 5.8. The statistics of this distribution is described via a boxplot on the same figure. The median percentage of missing values in feeders' data is 0.81% and the first and third quartiles are $Q_1 = 0.54\%$ and $Q_3 = 1.44\%$ respectively. Any feeder for which the percentage of missing observations exceeds 2.75% qualifies as an outlier; there are 66 such feeders in the data set. In Fig. 5.10, the percentage of missing values for each feeder is placed on a map of South East of England using `QGIS`. The map shows that the feeders (substations) heavily affected by missing measurements are mostly located on the south-west periphery of the London area. Furthermore, the map shows that the substations located outside the areas delimited by the M25, M26 and M2 are less prone to a high percentage of missing observations.

5.5.2 Horizontal analysis - Part I

The horizontal analysis of the matrix is concerned with the investigation of simultaneous missing measurements at individual timestamps. Fig. 5.7 exhibits horizontal patterns of interrupted grey lines. The lines display feeders for which measurements are simultaneously missing. The red rectangles highlight several timestamps for which multiple feeders were affected by missing measurements. In Fig. 5.9, the distribution of missing values across timestamps is given, and its statistics are described via boxplot. The median of missing coincident observations across the period spanning from 01 November 2014 to 30 September 2017 is 13. Any timestamps that experienced more than 27 simultaneous

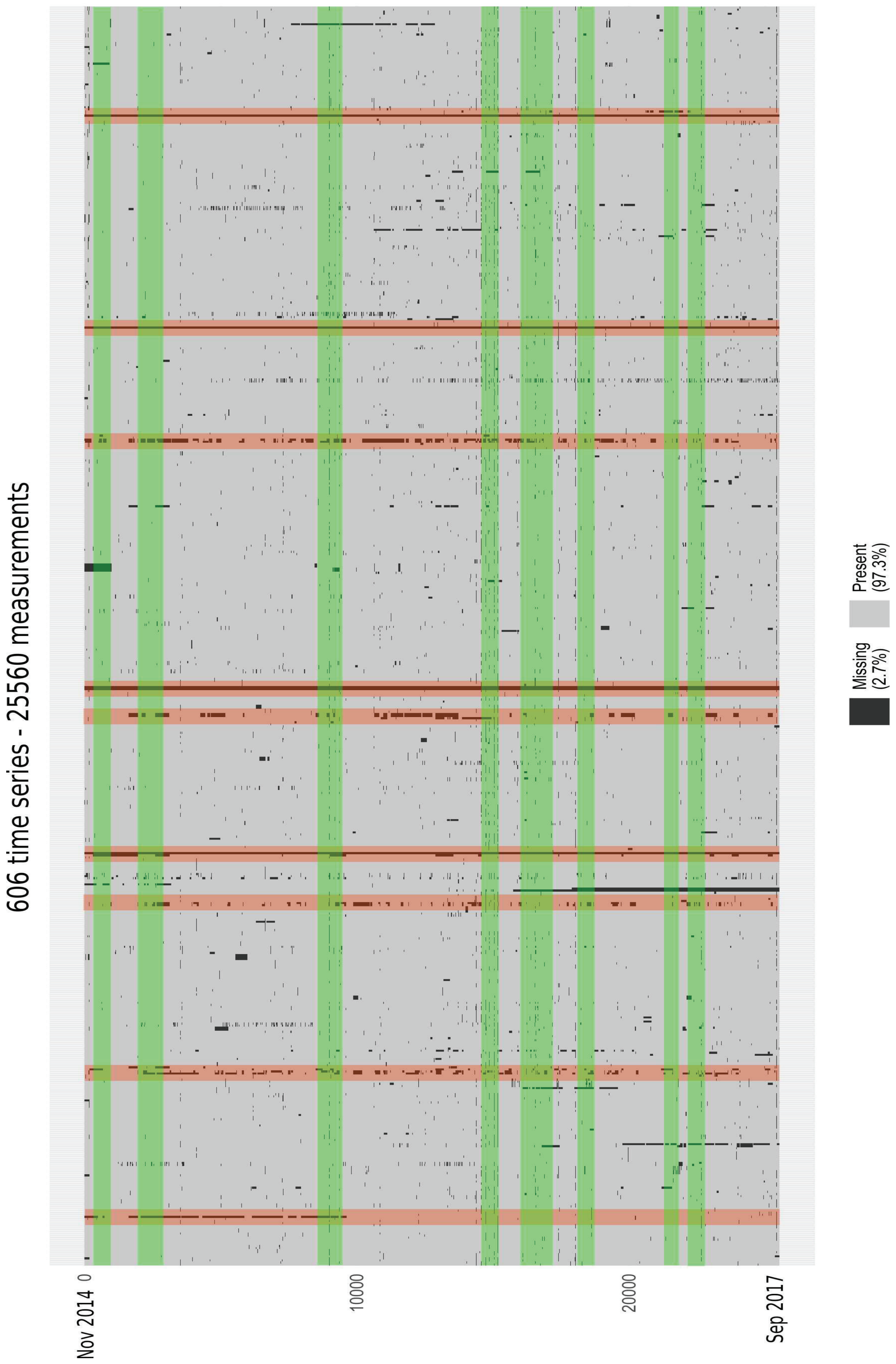

Figure 5.7: Missing values matrix 11/2014 to 09/2017

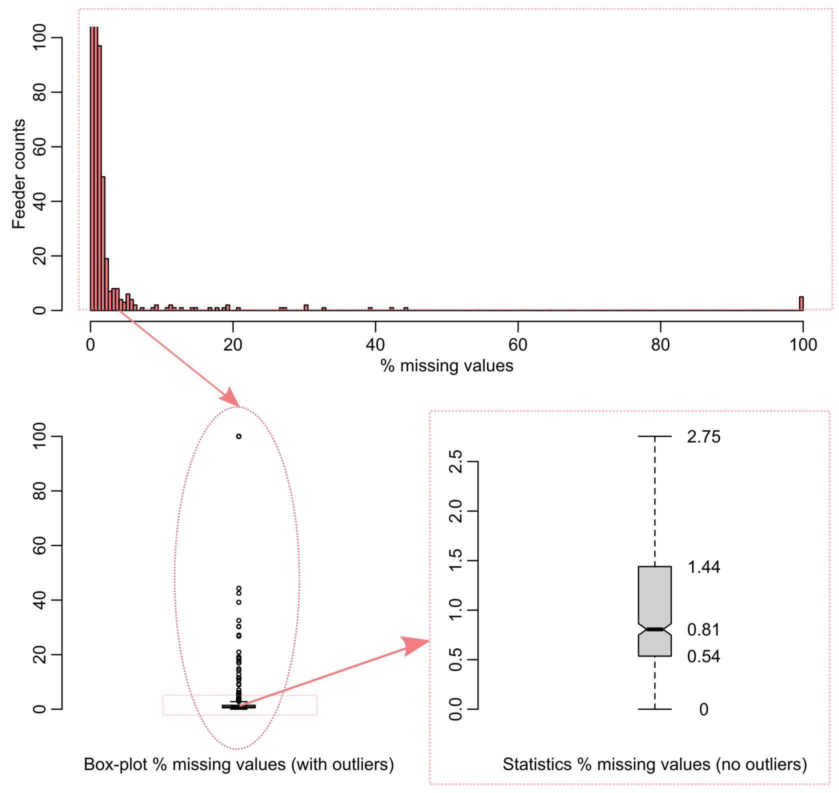

Figure 5.8: Statistics of missing values across feeders

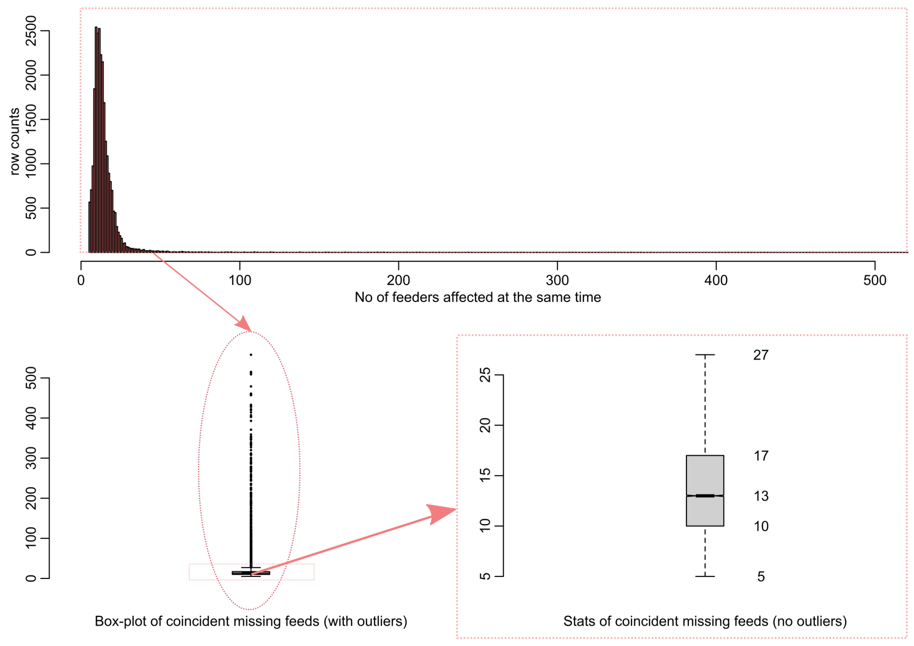

Figure 5.9: Statistics of simultaneous missing observations across timestamps

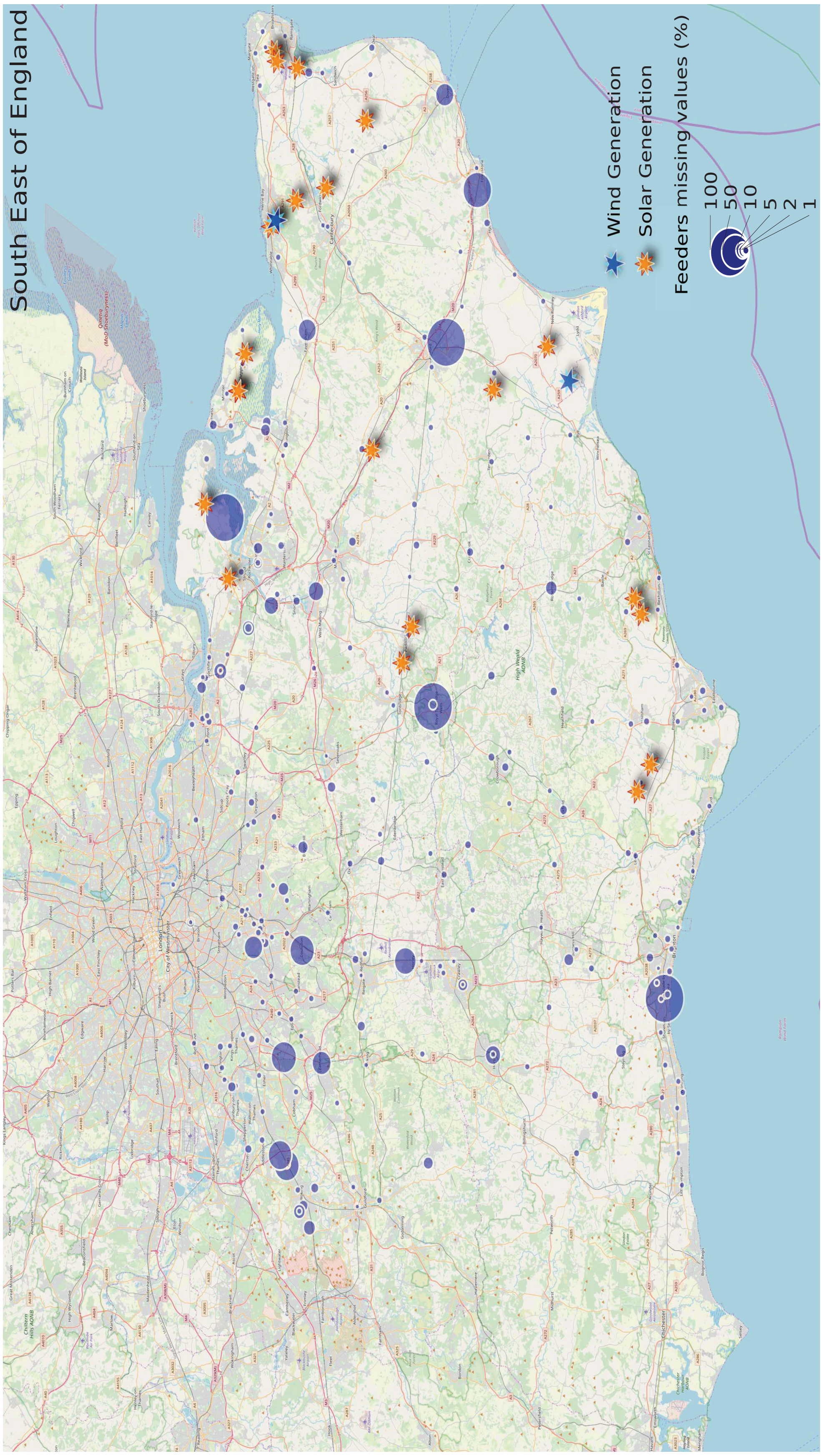

Figure 5.10: Feeders percentage missing values on map

missing measurements is an outlying event. These rare events are further analysed in the Sec.5.5.3.

The aim of this investigation is to determinate when and why many feeders' measurements are missing simultaneously. For this study, the full load datasets are plotted three different timestamps. The timestamps are selected from those that present the most significant number of simultaneous missing observations. The first event occurred on 11 November 2015 at 05.00 am, the data lost the largest count of feeders information, 558. The event is plotted in Fig 5.11, the analysis starts the 10 November 2015 at 22:00 to end on 11 November 2015 at 08:00 am. The plot displays the full dataset for eleven hours: seven hours before the most extensive loss of the feeders' measurements and five hours later. The numbers placed above the signals are the count of observations simultaneously missing at each hour (i.e. -21 indicates that 21 feeders values were lost at 22.00).

The analysis of missing hourly values in Fig 5.11 indicates that the loss of information occurred sequentially, the phenomenon could be described as a *cascade effect*; each hour, the number of vanishing records increases until its reaches a maximum of 558 missing records. Hence, most of the South East of England's electricity load information was lost at 05:00 am on 11 November 2015. The sequence of the feeders' record losses spanned nearly seven hours. The same phenomenon transpires in Fig. 5.12 which reports on a similar event that took place on 08 September 2016. On that date, a maximum number of 479 records were simultaneously lost at 20:00. The full sequence also lasted for seven hours. Fig. 5.13 illustrates two similar events on 16 August 2016, which were separated by fours hours. The first event occurred in the early morning and the second around sunset. Here again, both events spanned approximatively seven hours. One pattern that repeats across the four events is that the two last hours of the sequence experience the most drastic loss of measurements. The timestamps that display the maximum loss of information in Fig 5.11, Fig 5.12 and Fig. 5.13 were further examined in the original raw load data (the data prior to coercing them into numerical data type). It was found that most of the missing observations are shown as either 'Bad Data', 'Bad Total' or 'No Sample'.

The previous analyses suggest that the events happen sequentially and, not all feeders information are lost at the end of the sequences. Since it exists a physical communication link between the Data Historian and the Front-End Processor (FEP), the events described by the plots might not be caused by a communication failure between both systems. Thus, a physical communication link between Data Historian and the Front-End Processor (FEP) is a single point of failure and therefore, all feeders measurements should be lost at once. In addition, since most of the measurements spontaneously come back online after the event, the missing values phenomenon cannot be attributed to a hardware failure (i.e.RTUs, for instance).

To pursue the investigation on the cause(s) of coincident missing data, a time versus space analysis is proposed, supported by three geolocation maps. The 11 November 2015 event is geo-located in Fig 5.15, Fig 5.17 covers the 08 September 2016 event and Fig 5.16 displays the second event on 16 August 2016. On those maps, the feeders are identified by coloured circles; each circle is associated with a timestamp in the vicinity of the maximum loss of data points. For each timestamp, the colour and the size of the circle differ. Only the feeders for which the load information is available at the given timestamp, are identified by a sized/coloured circle. When the load value vanishes from the data set, the corresponding timestamped circle does not appear. The beginning of the sequential event in Fig 5.15, Fig 5.17 and Fig 5.16 is represented with the smallest circle in red, the end of the sequence appears in yellow with the largest circle.

The analysis approach that combines temporal and spacial features aims to determine if temporal and spatial correlations exist in the missing values phenomenon. Even though there is not an easy way to approach the analysis of 2D maps, there are few remarks that can be made from them. For instance, in Fig 5.15, the event is mapped between 23:00 and 05:00 am. At 02:00 am, only the feeders identified with the red, black, white and blue circles were still 'online'. One can see that the vanishing measurements are spread across the entire region, at various locations that can be closed to each other or diametrically opposed. This last remark applies to each timestamp; it may indicate a random component to the cause of the measurements' loss. Although there is no obvious pattern in the disappearance of the data, measurements collected from the rural areas (at the centre of the map) seem to be more affected than the ones collected from the London area or near the coast. Nevertheless, there is one pattern that arises from the three maps; it concerns the measurements that stay online during the entire sequence. It seems that the feeders for which loading information is available during the entire event are more or less the same. These feeders are identified with yellow circles on all three maps.

At this stage, it is difficult to clearly identify a cause of the cascade effect of the feeders' measurement loss. One hypothesis which can almost certainly be rejected is some potential hardware failures (isolated or synchronised) since the measurements come back after the event with no human intervention. Therefore, a communication issue is most probably causing such an event. The cascade effect needs to be further investigated for several reasons. As will be demonstrated later, these events are not sparse. In future, such data issues could affect real-time control strategies (i.e. active network management); these events also affect PV measurements as it is shown in Fig.5.14. Also, the mechanism that creates such events could be intentionally triggered; it would be safer to investigate in any case and solve the problem if it can be. An in-depth investigation requires significant domain knowledge such as expertise in network communication infrastructure and the procedure used to report communication failures.

5.5.3 Horizontal analysis - Part II

To further the study on coincident observations loss, one intends to explore the load data before November 2014. For this purpose, a new matrix with a longer span is provided in Fig. 5.19. The matrix comprises approximatively eight years of data spanning from November 2010 to May 2018. The same 606 feeders are considered, but their series consists of 60384 regularly spaced measurement instead of 25560. The black columns in the new matrix, correspond to entries for feeders that were tardily loaded. Alternatively, which have never been connected to a load. The most appealing outcome from Fig. 5.19 is that the data prior to 26 April 2014 did not experience so many missing values, not to

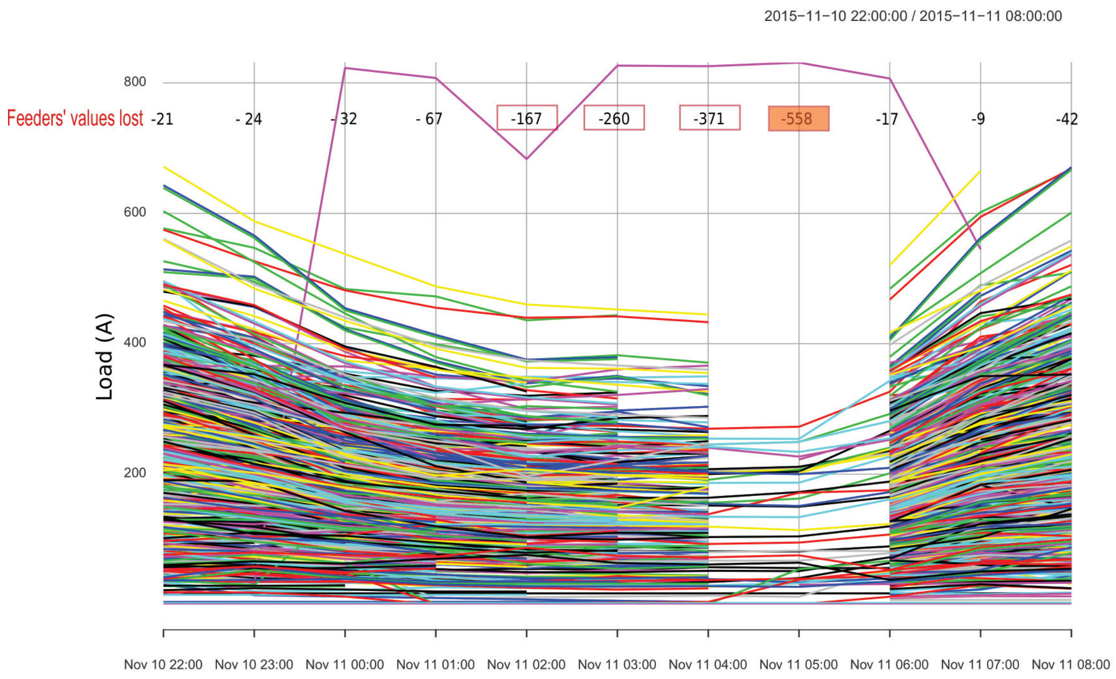

Figure 5.11: Patterns of simultaneous loss of load observations on 10-11 Nov 2015

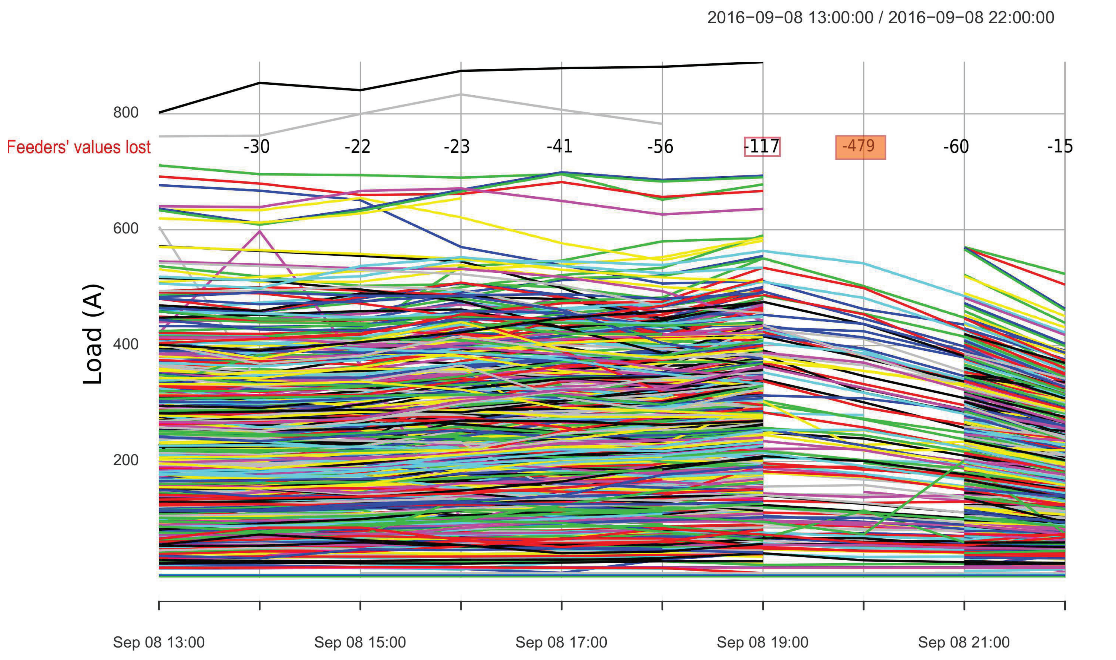

Figure 5.12: Patterns of simultaneous loss of load observations on 08 Sep 2016

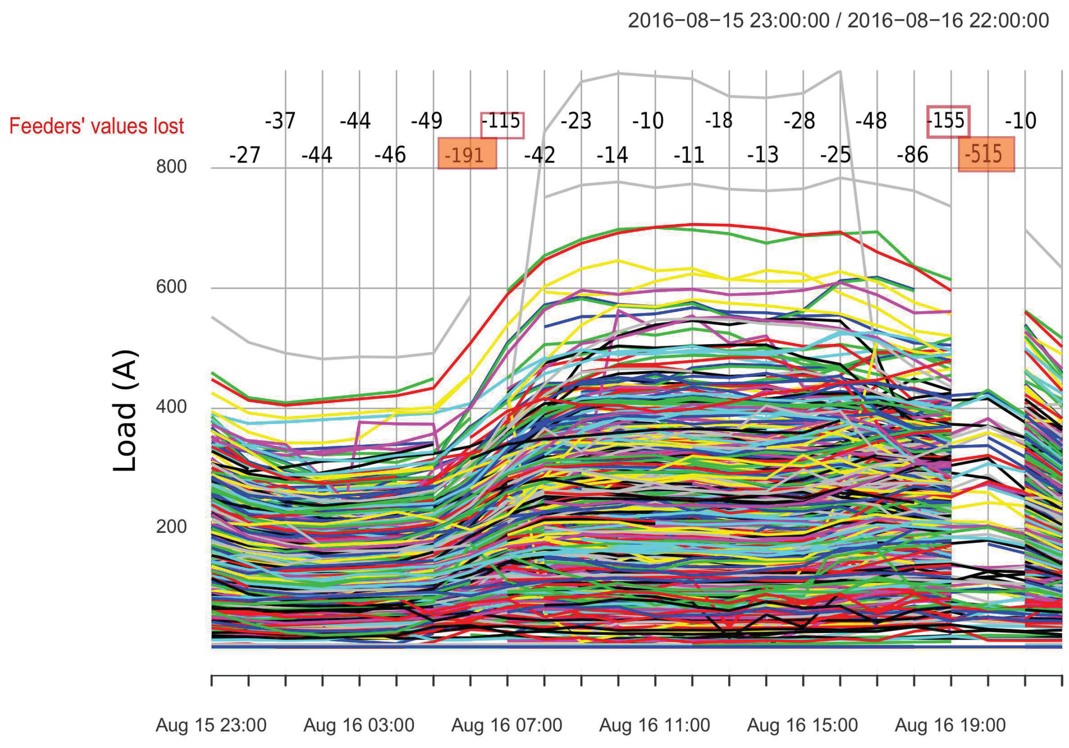

Figure 5.13: Patterns of simultaneous loss of load observations on 15-16 August 2016

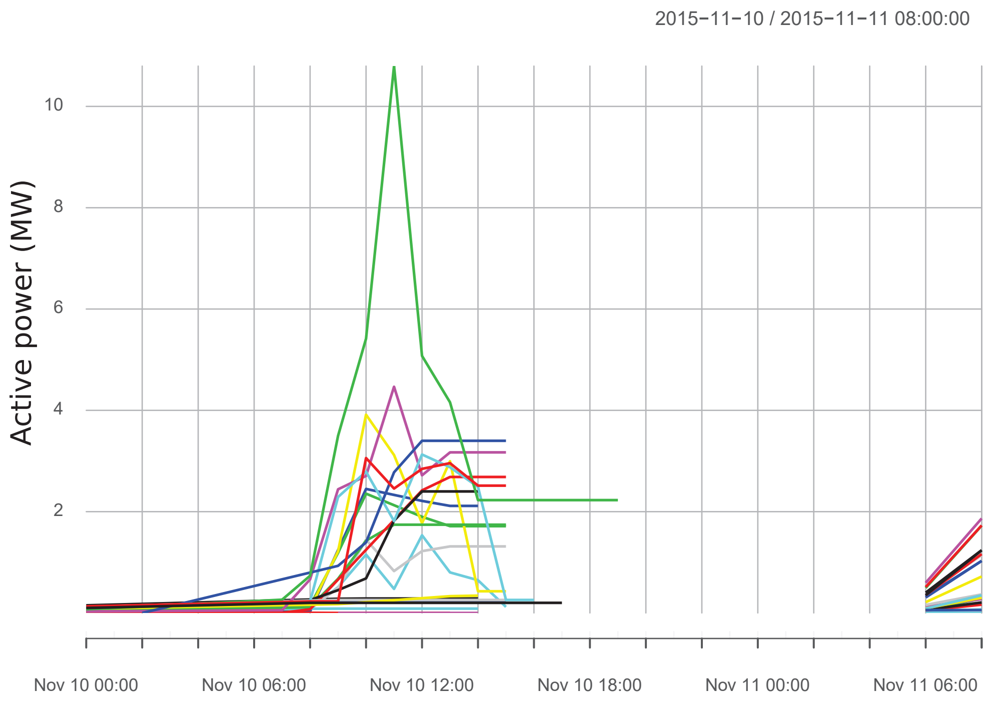

Figure 5.14: Patterns of simultaneous loss of PV observations on 10-11 November 2015

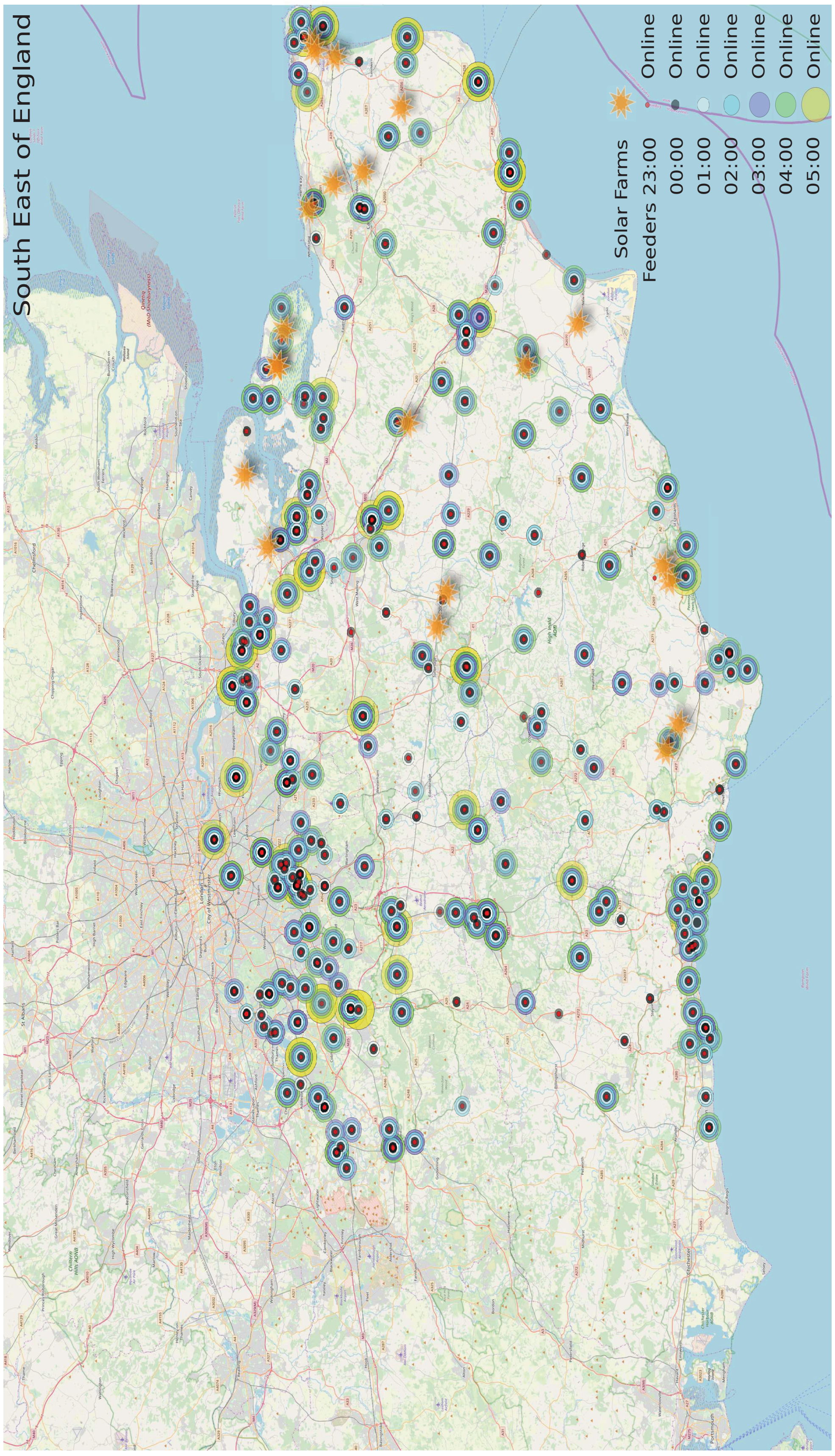

Figure 5.15: Spacio-temporal feeder information observation loss on GIS (10 to 11 Nov 2015)

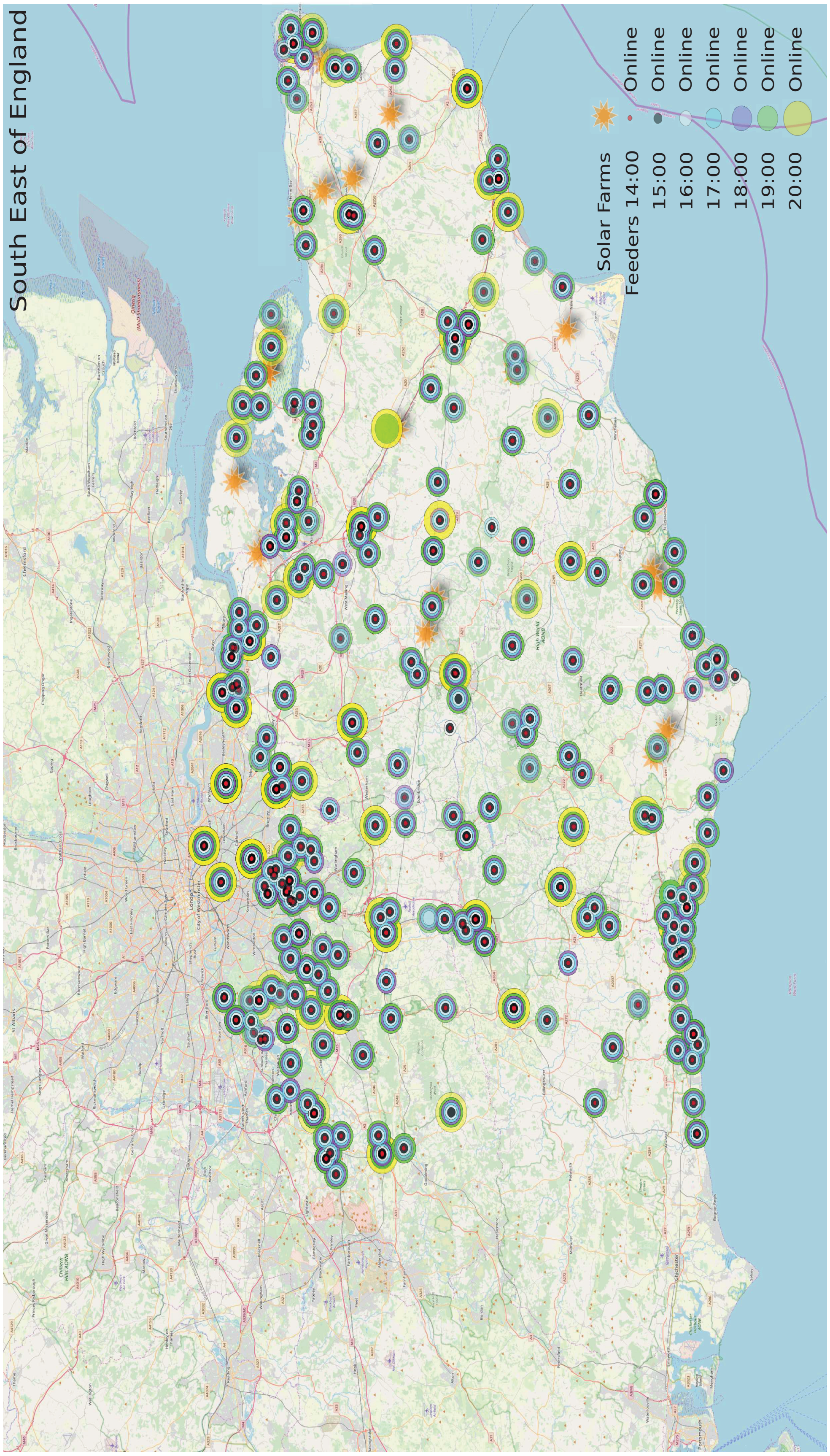

Figure 5.16: Spacio-temporal feeder information observation loss on GIS (08 Sep 2016)

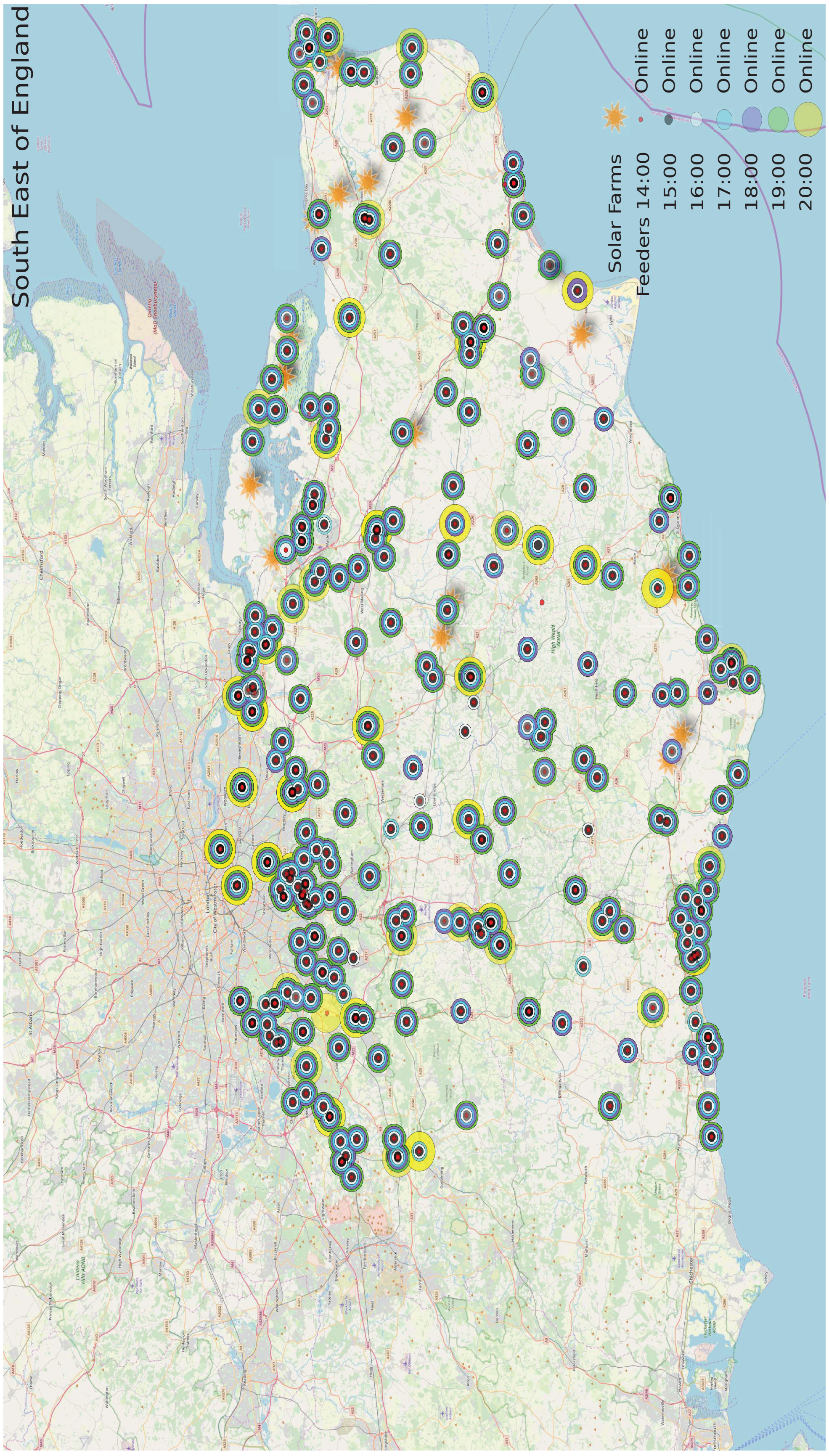

Figure 5.17: Spacio-temporal feeder information loss on GIS (16 August 2016)

mention that the cascade effect occurred only rarely as compared to the period after 26 April 2014. In a matter of fact, any count of missing coincident observations before 26 April 2014 includes all the non-loaded feeders. The plot in Fig. 5.18 displays the count of coincident observations loss thought time. It shows that posterior to 26 April 2014, the count of simultaneously observations loss becomes significant and much more frequent. The frequency and the distribution over time of these events are further analysed in Fig. 5.20. The plots report only the events for which the simultaneous loss of feeders' information exceed 100. Such event occurred only once prior to April 2014; thus, the plots concern essentially the period posterior to April 2014. In Fig. 5.20, plot (a) and plot (b) provide a 'zoom' into Fig. 5.9's outliers. In total, the events occurred 213 times between April 2014 and May 2018 with the median of coincident feeders information loss being 167. Plots (c) and (d) inform on how these phenomenons span across the years and the months. Plot (e) displays its intra-day distribution. In average, these events occurred between 40 and 50 times in 2014, 2015 and 2017 but 2016 experienced more than 100 of these events. The phenomenon can take place at any time during the year, but, July, September, and November are the most affected periods. The phenomenon happens mostly between 9.00 am and 8.00 pm, but there is also a high occurrence of these events at midnight.

5.5.4 Conclusion and discussion

The above analyses have made use of a matrix representation which has facilitated the visualisation of the full load data set. The analysis of the matrix was approached from a vertical and horizontal standpoints. The vertical analysis concerned the level of missing values in each feeder. It has shown that feeders connected at the vicinity of the urban area of the South East of England were experiencing higher percentage of missing observations. Plausible causes of data transmission error could be interferences from the environment surrounding the data transmission medium. Twisted pair copper wiring and optical fibre are the physical media used by the network; the data are transmitted as electrical pulses. There are three possible sources of interference that can attenuate, distort or corrupt the electrical signals: electromagnetic, radio frequency or crosstalk interferences [13]. Errors

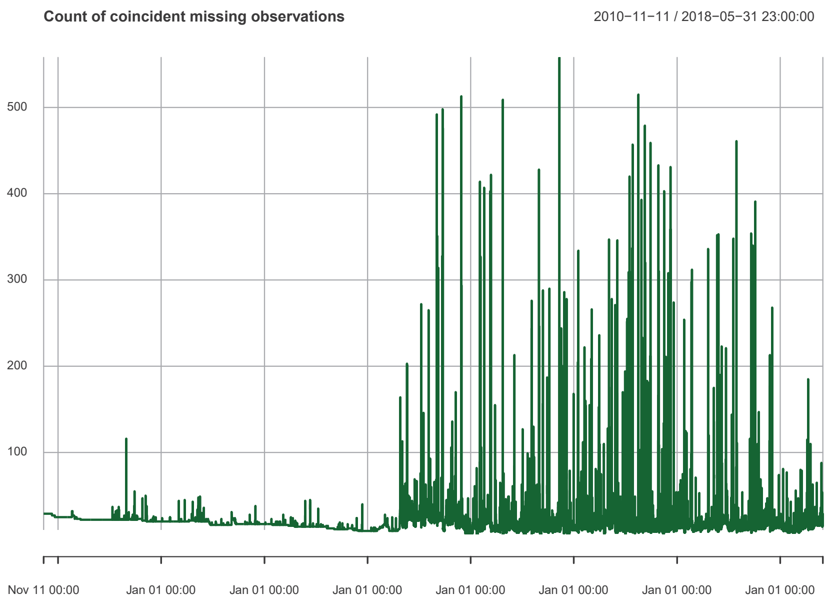

Figure 5.18: Count of coincident observations loss through time

in data transmission can also be caused by a loose cable/connector at one end of the communication link (i.e. loose connection at the RTU terminals), by a faulty cable, or by a RTU' or IED' defective communication hardware. A rigorous troubleshooting would help to investigate all potential point of failures in the data transmission path.

The horizontal analysis looked at the number of feeders that coincidently experienced a loss of information in the data set. The analysis has revealed that significant loss of feeder' values have occurred from time to time. A joint analysis of three timestamps was carried out to identify similarities between events. It was found that the loss of data points follows a sequence that spans approximately seven hours. During the sequences, feeders' information would be lost increasingly until reaching a maximum. Thereafter, the feeders' information start to be recorded again in the database. The 'events' could occur at anytime during the year but were more prone to happen in July, September, and November. The analysis has also shown that these events present diurnal pattern that is strongly correlated with human activity. Three geolocation maps that combine temporal and spacial features of feeders' information were used in an attempt of identifying if any pattern in the data vanishing process excited. No clear motif or correlated factors were

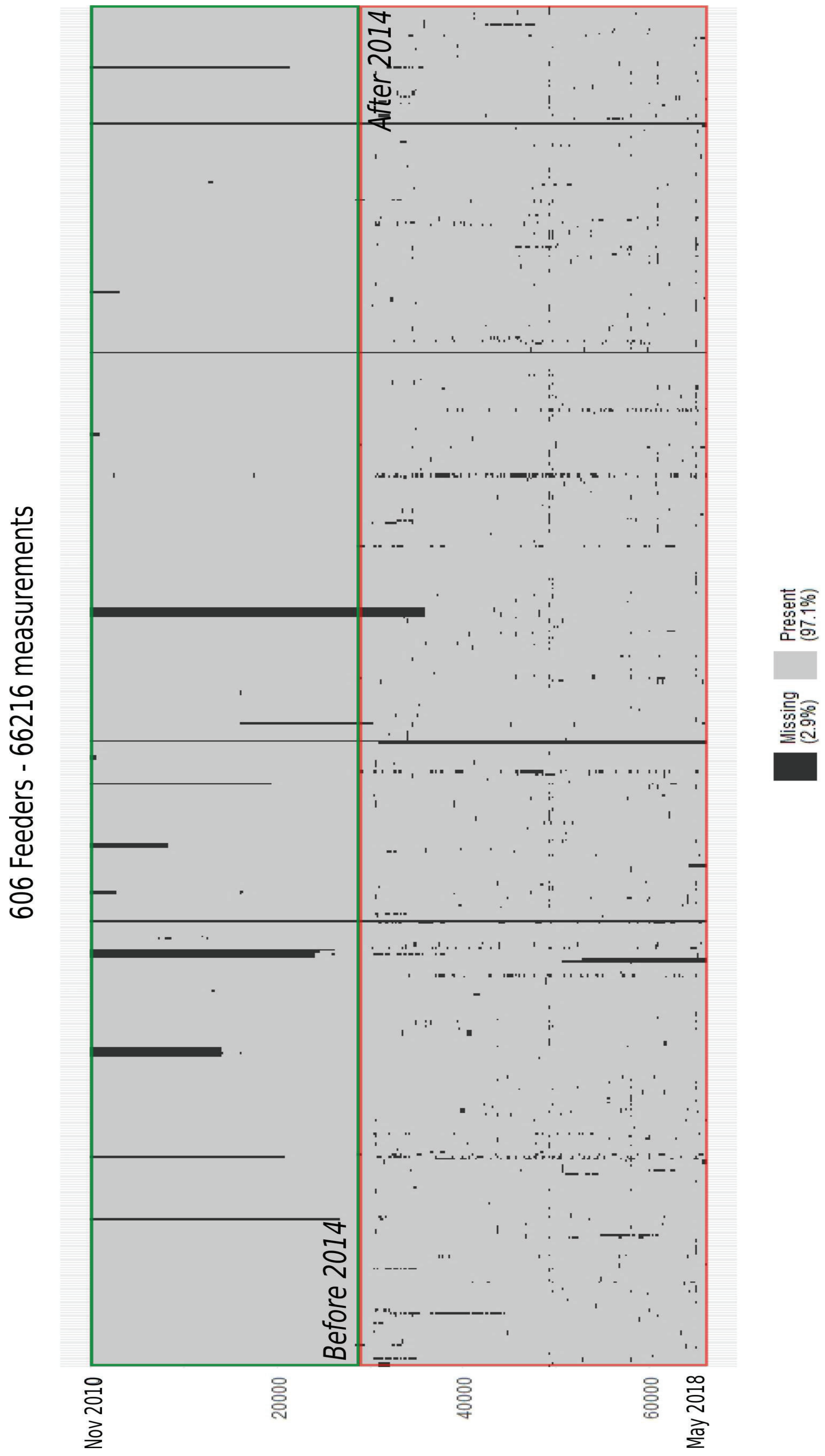

Figure 5.19: Missing values matrix from Nov 2010 to May 2018

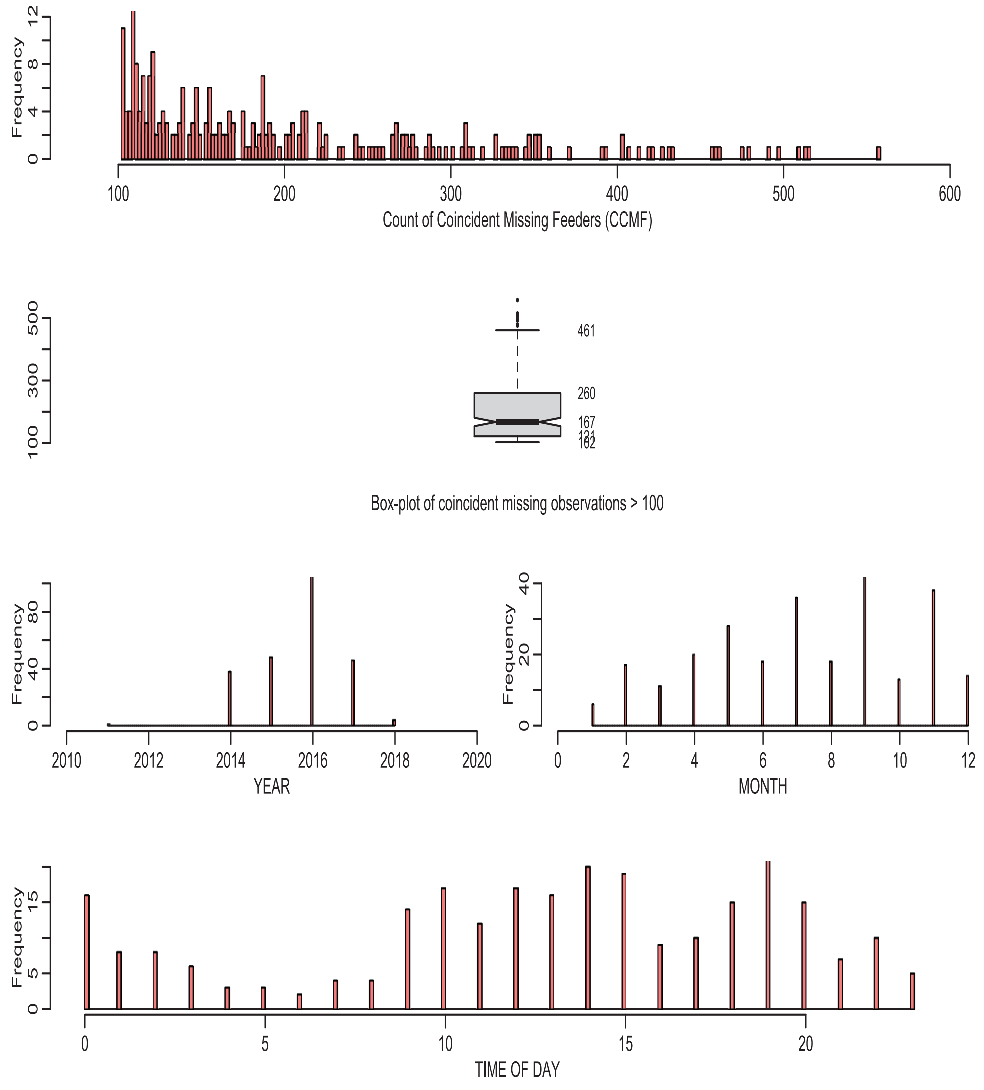

Figure 5.20: Seasonal analysis of coincident missing observations for simultaneous missing feeder > 100 (total occurrence: 213 times)

highlighted from the maps but one observed that a few feeder' values, in alike geographical locations, were less affected by the data loss sequence. It was established that the coincident data loss was unlikely to be related to some simultaneous nodes' breakdown (i.e.RTU). Ergo, it is more sensible to assume that the coincidental missing values issue results from a single cause. This assumption only narrows the sphere of possible roots of the problem (i.e. slow processor (CPU), limited flash memory (RAM)), nevertheless, it could be interesting to investigate if any connection exists between the sequence of missing values and the congestion control algorithm used by TCP protocol.

TCP is the transport layer used by most ADMS's applications. TCP has a built network congestion avoidance algorithm in which packet loss is the primary mechanism for signalling network congestion. *Congestion* arise in packet-switched networks because of insufficient buffers space, latency, timeout, packet retransmission, slow processor and, low bandwidth. In congested networks, a *bottleneck* link will space packets out in time, according to its service rate. Thereof, packets will be dropped when its buffer approaches its full capacity. Dropping of packets causes the sender to throttle back and stop flooding the bottleneck point with data [91]. TCP's congestion control is achieved with the AIMD algorithm. The algorithm adapts the size of a *congestion window* to regulate the data traffic and prevent or alleviate network congestion [95]. The algorithm combines linear growth of the congestion window when the sender receives an acknowledgement (before timeout) with an exponential reduction when congestion is detected. The approach taken is to increase the congestion/transmission window until a loss occurs. The additive increase may increase the window by a fixed amount every round-trip time (RTT). If congestion is detected, the sender decreases the window by a multiplicative factor. Let $w(t)$ be the congestion window at time t, a the additive increase parameter with $a > 0$ and b, the multiplicative decrease factor with $0 < b < 1$. The AIMD algorithm controls the congestion window as follows

$$w(t+1) = \begin{cases} w(t) + a, & \text{if congestion is not detected} \\ w(t) \times b, & \text{if congestion is detected} \end{cases} \tag{5.2}$$

with the multiplicative decrease factor b being typically 0.5. Recalling Fig. 5.20 that shows the diurnal character of the missing data events, it could reinforce the above assumption upon which the data are missing is the result of network congestion due to a full buffer somewhere on the network. However, this assumption can only hold if the duration of a TCP/IP connection between the sender and receiver exceeds the duration of the data missing sequences.

Chapter 6

Outlier detection techniques for MV feeders time series

'Naive' designs usually are more robust and better than 'optimal' designs.

Peter J. Huber

6.1 Introduction

Most of the time, real-world data are noisy and corrupted with outliers. To make a distinction between noise and anomaly, Aggarwal defines outliers as those data points that are significantly inconsistent with the remaining data [2]. Outliers relate to gross measurement errors, blunders and, measurement errors. Depending on the context, the proportion of gross errors in data is between 0.1% to 10% [59]. Most outlier detection methods are model-based, assuming the typical pattern of the data. It renders the choice of a suitable model determinant. To avoid poor fit, the model must capture the data's main properties which are closely related to the field and application domains. The criteria to choose a suitable model are the data type, data size, domain knowledge and most importantly, the interpretability of the model. Models that work directly on the attributes of the raw data with few data transformation have higher interpretability. The oldest outlier detection techniques use traditional tail inequalities such as Markov's and Chebyshev's to discriminate data points in the outer boundary of the data [6]; These techniques fall in the realm of Extreme Value Analysis.

Before the '70s, outlier detection relied on the assumption of an underlying known distribution of the data, which was assumed to be identically and independently distributed (iid). Besides, procedures for detecting outliers lied on a prior choice of the number of outliers in the data. Fox proposed the first work devoted to the detection of outliers in univariate time series in [47]. In [19, 119], Box and Tiao identified four types of outliers: additive outlier (AO), which affects only a single observation, and innovational outlier (IO) which may affect all subsequent observations, level shifts (LS) and temporary change (TC). Later, in [29, 30], Chang *et al* proposed an iterative method which applies statistical tests on the residuals to identify multiple outliers in time series. Outlier detection techniques used for time series are essentially model-specific, based on the assumption of temporal dependency, these procedures apply regression diagnostics.

The following procedure identifies outliers or inconsistent variations in a data set. The approach used avoids the need to specify the number of possible outliers in advance which is essential when the number of time series to be analysed is large.

6.2 Robust detection of outliers in univariate time series

There are two common approaches for dealing with outliers in regression problems, the *regression diagnostics* and the *robust regression*. A diagnostic approach identifies and removes the outliers from the data first and then fit the model to cleaned data, whereas a robust approach fits first a model to the entire dataset and then identifies the outliers as those data points which present large residuals. Robust regression approaches are sequential, and model parameters are re-estimated once the found outliers are removed. In a recent publication, Akouemo and Povinelli [3] adopt a robust regression approach for the treatment of outliers in daily natural gas data, based on work in the statistics community on outliers detection in time series data [19, 29, 47, 119]. Statistical techniques are parametric, relying either on prior knowledge of the data distribution or on estimating unknown parameters of an assumed family of statistical distributions. The salient downsides of parametric strategies are 1) their model dependency, 2) the estimation of the model parameters is biased by outliers, 3) they often assume stationarity of the model. Robust outlier detection procedures are classified together with non-parametric and distribution-free procedures [14]. The density-based techniques with the kNN (k nearest neighbour), the LOF (local outlier factor) and Tukey's rule are among the most popular.

Tukey's rule is a robust method that is visually related to boxplots to identify outliers. Potential outliers are flagged based on upper and lower hinges that are related to quartiles of a batch of measurements rather than distributional assumptions. To estimate the width of the central part of the data, the first quartile $q1$ (25% percentile) and third quartile $q3$ (75% percentile) are computed. The interquartile range ($iqr = q3 - q1$) has a breakdown point of 25% [58], indicative of high robustness against outliers. In Tukey's method, an observation is classed as an outlier when its value lies outside the outer fences, defined using the parameter r such that data points below $(q1 - r \times IQR)$ or above $(q3 + r \times IQR)$ are viewed as being too far from the median. The value $r = 1.5$ referred to as the main resistant rule by Tukey, and its performance is discussed in [69]. It is used in routine data analysis because it avoids the *swamping* effect in which the procedure tends to flag too many outliers. The resistant rules $r = 2$ and $r = 3$ were also proposed later for heavy tail distribution.

6.3 Robust off-line change-point detection in univariate time series

6.3.1 Binary segmentation

Change-point detection is the challenging problem of detecting the existence of abrupt changes in time series data. For an extensive overview of change-point methods, we refer to [12] and Truong *et al.* [117]. Change-points are those points in a data sequence where statistical properties such as mean, median, variance or distribution change significantly. A common strategy for change-points detection is to define a cost for a given segmentation of the data. Typically, the cost is based on defining a segment specific cost function. The sum of this specific cost function is computed over all the segmentations. To estimate the number and position of the change-points, the resulting cost is minimized. Among multiple change-points detection techniques, the Binary Segmentation (BS) is selected for its conceptual simplicity and its low computational complexity $\mathcal{O}(n \log n)$ [117]. The BS is a forward selection algorithm introduced by Scott and Knott in [107]. Let $\mathbf{y} = \{y_{1:n}\}$ denote a sample of observations from a nonstationary random process assumed to be piecewise stationary with k change-points at $\boldsymbol{\tau} = \{\tau_1, \ldots, \tau_k\} \subset \{1, \ldots, n\}$, the sequence of change-points is ordered such that $\tau_i < \tau_j$ if, and only if $i < j$. The dummy variables $\tau_0 = 0$ and $\tau_{k+1} = n$ are implicitly available. The segmentation refers to the automatic decomposition of $\mathbf{y} = \{y_{1:n}\}$ into $k + 1$ weakly stationary segments with the ith segment containing $s_i = \{y_{\tau_{i-1}+1:\tau_i}\}$.

6.3.2 Description of the algorithm

Initially, the entire dataset is searched for one change-point, typically via a cost function to be minimized. Once a change-point is detected, the data are split into two sub-segments, defined by the detected change-point. A similar search is then performed on either sub-segment, possibly resulting in further splits. The recursion continues until a given criterion is satisfied. Here, to identify multiple unknown change-points in the data, the method adopts a general form where a contrast function $V(\boldsymbol{\tau}, \mathbf{y})$ (that penalizes a high amount of change-points in order to avoid overfitting) is minimized with respect to k and $\boldsymbol{\tau} = \{\tau_1, \ldots, \tau_k\}$. As discussed in [84,118], we assume the penalty term to be linear in $k = |\boldsymbol{\tau}|$. Under this assumption, the cardinality constrained problem to be solved can be written as

$$\min_{\boldsymbol{\tau},|\boldsymbol{\tau}|} V(\boldsymbol{\tau},\mathbf{y}) + \beta|\tau| \text{ with } \beta > 0 \tag{6.1}$$

$$V(\boldsymbol{\tau},\mathbf{y}) := \sum_{i=0}^{k} c(y_{\tau_{i-1}+1:\tau_i}) \tag{6.2}$$

$$c(y_{\tau_{i-1}+1:\tau_i}) := \sum_{t=\tau_{i-1}+1}^{\tau_i} |y_t - \overline{y}_{\tau_{i-1}+1:\tau_i}| \tag{6.3}$$

The parameter β controls the balance between model complexity and goodness of fit. Low values of β favour overfitting with too many change-points and high values of β discard most true change-points. The cost function $c(.)$ in Eq. (6.3) measures the homogeneity of the sub-signal $s_i = \{y_{\tau_{i-1}+1:\tau_i}\}$. Thus, the cost is expected to be low when the sub-signal does not contain any change-points and large when it does. Various cost functions are found in the existing literature among which piecewise linear models.

The Least Absolute Deviation (LAD) that was proposed by Bai in [10] for the estimation of level-shift points in autoregressive signals and noisy distributions is used. He considered the L1-norm because of its robustness against heavy tails distribution. [10]. In our case study, the L1-norm approach we have chosen to estimate change-point localization. The cost function is given as where $c(y_{\tau_{i-1}+1:\tau_i})$ is the sum of absolute deviations for each t from the empirical median $\overline{y}_{(\tau_{i-1}+1):\tau_i}$ of sub-signal $y_{\tau_{i-1}+1:\tau_i}$.

The Binary Segmentation approach iteratively inserts change-points in segments $s_i = \{y_{\tau_{i-1}+1:\tau}\}$ of the entire signal $\mathbf{y} = \{y_{1:n}\}$. The elementary operation is the single change-point method, it tests if a split of the segment exists such that the cost function over the two sub-segments plus the penalty term is smaller than the cost function across the entire signal $\mathbf{y} = \{y_{1:n}\}$. Under the linear assumption, the penalty term $\beta|\tau|$ is reduced to β for a single change-point search and the algorithm tests whether it exists a time index $\tau \in \{1 \dots n\}$ that satisfies

$$c(y_{1:\tau}) + c(y_{\tau+1:n}) + \beta < c(y_{1:n}) \tag{6.4}$$

If no change-point is detected, no additional change point is created and the algorithm stops. The binary segmentation is illustrated in Algorithm 1.

6.3.3 Proposed stopping penalty

In the literature, penalty terms have been proposed and justified either from theoretical assumptions or inferred from data [118,123]. The two most common penalties used in the literature were tested, namely the Akaike and Schwartz penalties, but in combination with the L1-norm cost function, the binary segmentation failed to detect any change-points. After experimentation, the value $\boldsymbol{\beta = 4 \times \log(n)}$ was selected as a suitable choice that allows the binary segmentation algorithm to approximate the number of change-points for a wide range of feeder data. A sensitivity analysis on the parameter β was carried out to assert that its value was suitable for feeder data with and without level-shifts. In addition, the proposed choice of β has been tested against scaling and shifting of the data, showing invariance properties with respect to the problem under consideration.

Algorithm 1 Binary segmentation

```
    Input: $\{y_t\}_{t=1}^{n}$ where $y_t \in \mathbb{R}$, cost function c(.), penalty constant $\beta$
    Output: $B$ set of estimated level-shifts indexes
 1: Initialize $B \leftarrow \{\}$
 2: repeat
 3:     $p \leftarrow |B|$.
 4:     $\tau_0 \leftarrow 0$ and $\tau_{p+1} \leftarrow n$
 5:     if $p > 0$ then $B = \{\tau_1, \ldots, \tau_p\}$
 6:     end if
 7:     Initialize array length $\mathbb{G}$ = p+1
 8:     for $i = 0, \ldots, p$ do
 9:         $\mathbb{G}[i] \leftarrow c(y_{\tau_i \ldots \tau_{i+1}}) - \min_{\tau_i < \tau < \tau_{i+1}} [c(y_{\tau_i \ldots \tau}) + c(y_{\tau \ldots \tau_{i+1}})]$.
10:     end for
11:     $\hat{i} \leftarrow$ argmax$_i \mathbb{G}[i]$.
12:     $\hat{\tau} \leftarrow$ argmin$_{\tau_{\hat{i}} < t < \tau_{\hat{i}+1}} [c(y_{\tau_{\hat{i}} \ldots \tau}) + c(y_{\tau \ldots \tau_{\hat{i}+1}})]$
13:     $B \leftarrow B \cup \{\hat{\tau}\}$
14: until max($\mathbb{G}[i]$) $< \beta$
```

6.4 Automatic outlier detection algorithm

In [2], Aggarwal asserts that the most effective methods for outlier detection are dataset specific and make use of contextual information to develop strategies tailored to the data in hands. The proposed strategy integrates the load seasonal features, namely, typical days of the week and the yearly cycle, in the outlier detection procedure. It is a single-step automatic procedure which identifies all outliers in a "segment" at once as opposed to the recent recursive method proposed in [4], which requires many model fits. The method proposes to adapt Tukey's univariate rule method illustrated Fig. 6.1 to detect and remove outliers from piecewise stationary segments. Segments are bounded by time indexes of detected change-points if any exists, otherwise the full dataset is processed. A segment must contain at least a complete day to be processed. The method compares observations to suitable upper bound and lower bounds at each time step. Let S be a segment of raw data to be processed. S is divided into $\mathcal{S}_p^{w,s}$, where p is the hour of day, p depends on the granularity of the data, i.e. $p = 24$ for hourly data, $p = 48$ for half-hourly data, w is the typical day, weekday (WD) or weekend (WE) and s is the season. We implement Tukey's method to construct one Upper Bound (UB) vector and one Lower Bound (LB) vector, one for each typical days at all seasons $|s|$ in S. Let $[\boldsymbol{Lb}_s^{(1)}, \boldsymbol{Ub}_s^{(1)}] \in \mathbf{R}^{p\times 2}$, the UB_s and LB_s be vectors for WD and $[\boldsymbol{Lb}_s^{(2)}, \boldsymbol{Ub}_s^{(2)}] \in \mathbf{R}^{p\times 2}$ the UB_s and LB_s vectors for WE. We compute the 5th and the 95th percentiles $q5_s^{(i)}[j]$ and $q95_s^{(i)}[j]$ respectively, and $iqr_s^{(i)}[j]$ with $i \in [1, 2]$, $j \in [1, \ldots, p]$, then we update the UB_s and LB_s vectors for both typical days as follow:

$$\boldsymbol{Lb}_s^{(i)}[j] = q5_s^{(i)}[j] - 1.5 \times iqr_s^{(i)}[j] \tag{6.5}$$

$$\boldsymbol{Ub}_s^{(i)}[j] = q95_s^{(i)}[j] + 1.5 \times iqr_s^{(i)}[j] \tag{6.6}$$

Once Tukey's hinges are computed for WD's and WE's for each season, daily observations in S are compared against the $UB_s^{(i)}$ and $LB_s^{(i)}$ vectors and outliers are flagged then removed from the data. Data are classified as outliers/non-outliers based on whether or not they fall outside the given bounds. Algorithm 2 illustrates the full outlier detection

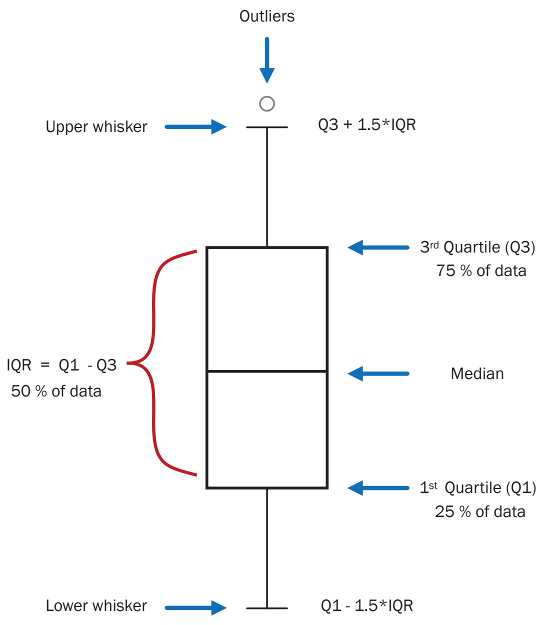

Figure 6.1: Tukey's Boxplot

procedure in the presence of level-shifts (change-points).

Algorithm 2 Outlier detection in presence of level-shifts

```
    Input: $\{y_t\}_{t=1}^n$ where $y_t \in \mathbb{R}$, cost function c(.), penalty constant $\beta$
    Output: $\{y_t^{(c)}\}_{t=1}^n$ cleaned data.
 1: $k \leftarrow 0$.                ▷ break-points to be estimated
 2: % *Binary segmentation*
 3: $B \leftarrow \{\tau_1, \ldots, \tau_k\}$.        ▷ output $k$ indexes of estimated level-shifts in $\boldsymbol{y}$
 4: % *Adapted boxplot labelling rule*
 5: $l \leftarrow \{y_{1:\tau_1}, \ldots, y_{\tau_k:n}\}$        ▷ form $k+1$ consecutive segments
 6: $y^{(c)} \leftarrow \{\}$
 7: for $s$ in $l$ do
 8:     $s^{(c)} \leftarrow$ Remove outliers from s
 9:     $y^{(c)} \leftarrow y^{(c)} \cup s^{(c)}$        ▷ Store cleansed $s^{(c)}$
10: end for
```

6.5 Case Study

6.5.1 Results and discussions

The results of the segmentation and cleaning process are illustrated in Fig.6.2. The top figure exhibits the raw data from *Feed 3* (see Table 7.8) prior to application of the outlier cleansing framework to the data. This feeder data presents multiple structural breaks and multiple outliers; piecewise stationary segments 1 to 4 are indicated with red horizontal arrows. The bottom figure illustrates the data after the outliers being removed. Table 6.1 reports the count and the percentage of removed outliers in each segment. On average less than 1% errors were detected in the training datasets with a maximum of observations removed not exceeding 2%. For distributions close to normal, the masking (false negatives) and swamping (false positive) effects on the detection error using the Tukey rule should not exceed 0.6% as per the study carried out by Hoaglin *et. al* in [69]. The forecasting performances associated with this feeder following outlier removal and missing values imputation process are presented in Table 7.8 and Table 7.9 and discussed in the case study section.

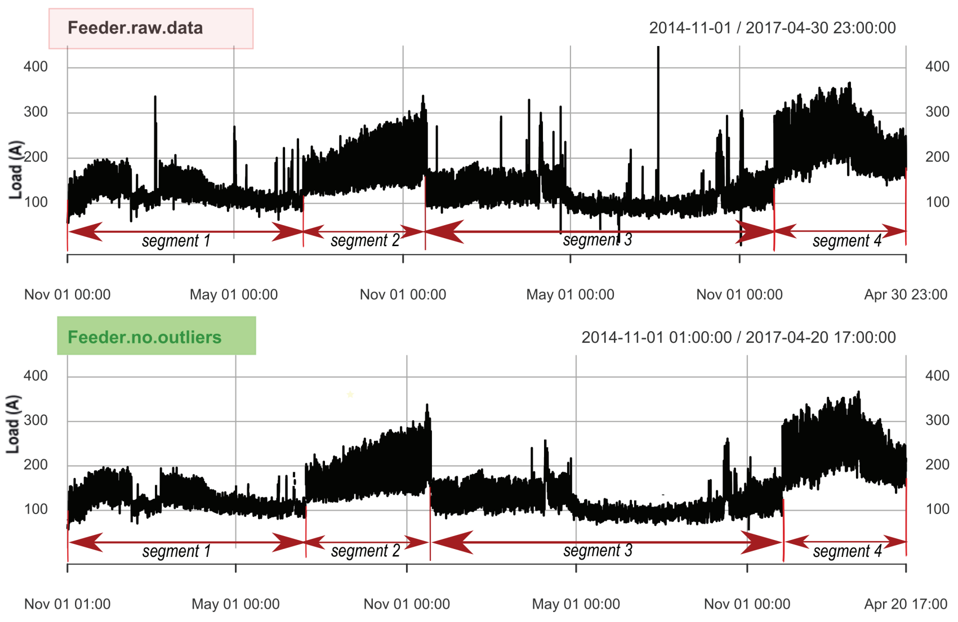

Figure 6.2: Outlier detection and removal with BS-Tukey *Feeder 3* data - the data contain multiple structural breaks - top plot (before detection), bottom plot (after detection).

Table 6.1: Count of removed outliers in each segment of *Feeder 3* training data

	segment 1	**segment 2**	**segment 3**	**segment 4**
sequence	[0, 6150]	[6151, 9380]	[9381, 18450]	[18451,21641]
No of outliers	70	18	87	23
(%) outliers	0.32	0.08	0.4	0.1

6.5.2 Computational time

The proposed cleansing procedure consists of the combination of the binary segmentation algorithm and the Tukey rule. Both procedures have a complexity of $\mathcal{O}(n \log n)$ which gives to our procedure a complexity of $\mathcal{O}(n \log n)$. Fig. 6.3 and Fig. 6.4 illustrate the performance of the outlier detection procedure for a half-hourly and hourly resolution. The boxplots show the statistics of the running time and the percentage of removed outliers relative to 342 MV feeders. In practice, however, the running time rarely exceeds one minute, which is easily doable for the size of the datasets.

Introduction to hypothesis testing

A statistical hypothesis testing is an assertion concerning one or more populations, it is formulated in terms of two statistical hypotheses: H_0, the null hypothesis and H_1, the alternative hypothesis. The null hypothesis H_0 asserts that there is no difference between one or two parameters and a specific value. The alternative hypothesis H_1 states the contrary. The aim of hypothesis testing is to determine whether the null hypothesis is likely to be true given a sample data. If there is little evidence against the null hypothesis given the data, the null hypothesis is accepted. If the null hypothesis is unlikely given the data, it might be rejected in favour of the alternative hypothesis. In hypothesis testing, the level of significance α is a probability threshold used to determine whether to reject H_0 in favour of the H_1. The level of significance $\alpha = 0.01$ relates to 99% confidence level for rejecting H_0. Incorrect conclusions made from hypothesis tests fall in one of two categories: type I error and type II error. Type I error describes a *false positive* situation where H_0 is true, but is rejected. Type II error describes a *false negative* situation where H_0 is false, but erroneously fails to be rejected.

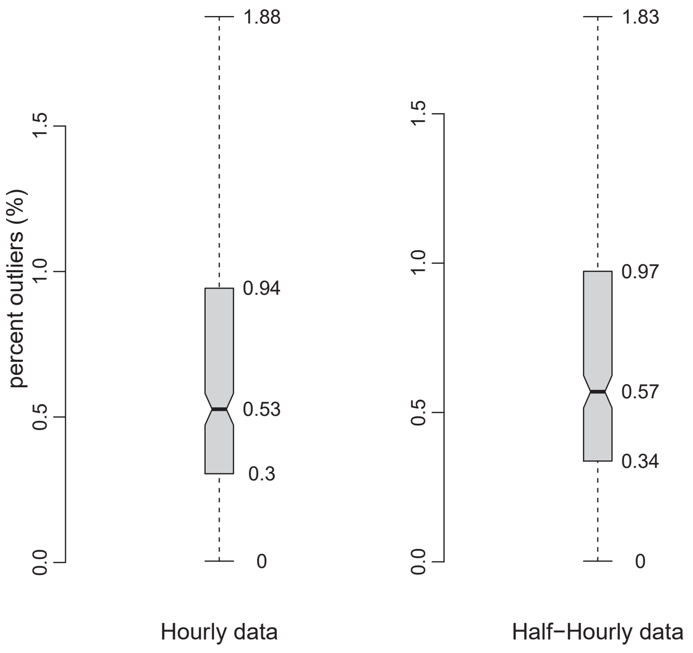

Figure 6.3: Running time statistics of the automatic outlier detection across 342 feeders for half-hourly data (43282 samples each) and hourly data (21641 samples each)

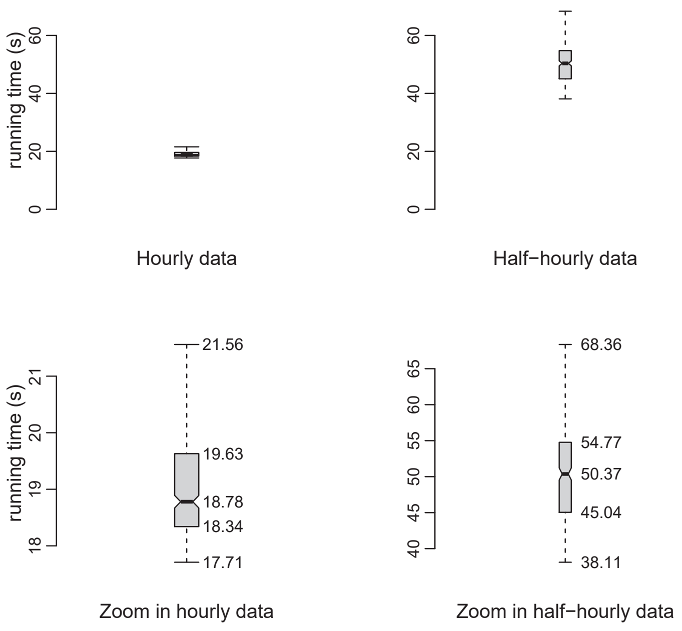

Figure 6.4: Statistics of the percentage outliers removed by the automatic outlier detection across 342 feeders for hourly data (43282 samples each) and hourly data (21641 samples each)

Outlier detection by hypothesis testing

The proposed outlier detection using Turkey's method is compared to the algorithm proposed by Akouemo and Povinelli in [3]. In the following, the hypothesis procedure is refereed to as H-test. Akouemo and Povinelli adopt a robust regression approach for the treatment of outliers in daily natural gas data. The methods identifies those observations that lie in a so-called outlier region. Hypothesis testing is used to detect outlier upon forecasting residuals results where outliers correspond to the residuals in the tail of a normal distribution. The H-test procedure is recursive; it fits unhealthy data to a given model and runs a hypothesis testing procedure on the forecast residuals to establish if an extremum is an outlier. When the probability of an extremum to be an outlier exceeds the level of significance set to $\alpha = 0.01$, the corresponding data point is removed from the dataset, imputed, and the model is retrained on healthier data. The procedure repeats until no outliers are found in the residuals.

The method proposed in [3] was adapted in a few ways to make a direct comparison possible. The authors of [3] propose to fit the data to ANN and NARX models, instead, we used the same FDNN architecture and inputs vector described in the previous section to fit the data. Moreover, we have modified the procedure in to accommodate the data in hands in three ways. First, all missing values in the raw data had to be imputed prior to train the FDNN models, they were imputed using unconditional mean. Secondly, to ensure a fair comparison, the change-points detection procedure was included in the full H-test algorithm. For each of the detected segments, we trained a model and detect outliers in the one-step-ahead forecast residuals. Lastly, we imputed the outliers using the median of segments instead of an interpolation method since the feeder data exhibit groups of consecutive outliers but not isolated outliers as it was the case with the natural gas data used in [3]. We run the H-test augmented with change-points detection as follows; outliers were searched one-by-one and temporarily imputed with the median of the segment. Outlier indices were recorded during the search. Once the algorithm had found all the outliers, we imputed all outliers with mean and we trained the FDNN forecasters so that we could compare the forecasting results with **no_m** (no outliers + mean imputation) datasets for Tukey and H-test method.

Tukey method vs Hypothesis testing

Both procedures are compared in terms of running time and detection accuracy, by applying them to load time series of two feeders: *Feed 1* and *Feed 3* (see Fig. 7.4). We first used both Tukey's procedure and the H-test to detect outliers in the training and testing data of the two aforementioned feeders. We then trained two forecasters per feeders with the cleaned data and compare the predictions from each models. In Table 6.2, the number of outliers found in the training and testing datasets by both procedures are reported. We have recorded the running time for each outlier detection procedure. Running time and 24-h ahead forecasts MAPE results are also reported in the table. Note that Tukey's method on average is 65 times faster than H-test.

Table 6.2: Outlier detection and MAPE(%) results : Tukey method vs Hypothesis testing (H-test)

	Feed 1		Feed 3	
	H-test	Tukey	H-test	Tukey
number of outliers in training data	199	191	52	198
Running time (min)	50.51	0.37	8.33	0.33
number of outliers in testing data	22	39	35	76
Running time (min)	2.23	0.055	3.60	0.061
MAPE(%)				
no_m_r	5.33	4.11	10.40	8.38
no_m_no	4.11	4.02	10.34	6.22

Looking at the running time performance of both procedures, it is obvious that Tukey's method is much faster than H-test. If we compare the MAPE results for *Feed 1* between both procedures, the performance of dataset preprocessed with Tukey and H-test are similar although H-test omitted a number of outliers in the testing data which has sightly penalized the forecast accuracy. In *Feed 3* case, H-test was significantly less successful at identifying harmful outliers. The discrepancies in the number of found outliers by the H-test algorithm can be explained; the binary segmentation only approximates the number of change-points, hence, the number of level-shifts detected in the data might not always be optimal. If the segmentation is not optimal then the residual normality assumption does not hold and the H-test cannot perform well. This demonstrates the robustness of the Tukey method. In addition, the H-test method relies on the model's parameters estimation, therefore each time the method runs, the total number of found outliers varies as opposed to Tukey method that always flagged the same observations as outliers. In definitive, both methods are relatively easy to implement, but H-test may be less suitable for large data sets, due to computational requirements. Tukey's approach is fast and robust and will be a better choice for voluminous data.

6.6 Semi-Automatic Outlier Detection

6.6.1 Description of the algorithm

In few instances, the automatic outlier detection procedure failed to cleanse the data. The failure was caused by the presence of data gaps in the historical datasets. Two examples of feeders' time series F1 and F2 for which the cleaning process has not provided satisfactory results are displayed in Fig 6.5 and Fig 6.7. The presence of these bad observations in the data occasion two issues: 1) their presence might bias the network's parameters estimation 2) some gaps are not entirely accessible for a complete missing values imputation. The presence of those strange variations at each extremity of the gaps sometimes prevent to operate an effective imputation with more accurate estimates. To overcome these issues, a semi-automatic cleaning procedure was implemented. The proposed cleansing approach applies the median filtering technique, a nonlinear method used to remove noise from image [101] and as a smoothing technique in time series analysis .

The median of m observations $\boldsymbol{y} = \{y_t\}_{t=1}^{m}$ is denoted by $med(\boldsymbol{y}_t)$ and is given by

$$med(\boldsymbol{y}_t) = \begin{cases} y_{(k+1)}, & m = 2k+1 \\ \frac{1}{2}(y_{(k)} + y_{(k+1)}) & m = 2k \end{cases} \tag{6.7}$$

where $y_{(t)}$ denotes the tth order statistic. The one-dimensional *median filter* computes the *moving median* of all windows $\boldsymbol{W}_i^\nu = \{y_t\}_{t=i}^{i+\nu}$ of length ν, sliced along the signal $\boldsymbol{y}_t$ where $i \in \mathbb{Z}$. The output of the filter m_j^ν is given by

$$m_j^\nu = med(\boldsymbol{W}_j^\nu) = \begin{cases} y_{(j+k+1)}, & \nu = 2k+1 \\ \frac{1}{2}(y_{(j+k)} + y_{(j+k+1)}) & \nu = 2k \end{cases} \tag{6.8}$$

where $j = k, k+1, \ldots, n-k$. The one tuning parameter of the filter is the window half-width k which makes its, implementation simple. The strengths of the median filter lie in, 1) its high resistance to local outliers, this allows the filter to preserve the signal edges, and 2) its low computational cost, $\mathcal{O}(n \log n)$, for sorting n observations. The proposed semi-automatic outlier detection uses the median filter technique to form a bounding envelope that encloses the time series $\boldsymbol{y}_t$ from above and below as illustrated in Fig.6.8 and Fig.6.6. The procedure takes three parameters: r_u and r_l adjust the upper and lower bounds, U_j and L_j respectively with $U_j > m_j^\nu > L_j$ and r_u and $r_l \in \mathbb{R}_{>0}$. The window size ν tunes the smoothing rate of the envelope's bounds. The upper bound U_j and the lower bound L_j are given by

$$U_j = m_j^\nu + r_u \times \bar{\boldsymbol{y}} \tag{6.9}$$

$$L_j = m_j^\nu - r_l \times \bar{\boldsymbol{y}} \tag{6.10}$$

where $\bar{\boldsymbol{y}}$ can be set as the **mean** or the **median** of the entire sequence $\boldsymbol{y}$.

6.6.2 Performance

The values of parameters adjusted to clean the two feeders are provided in Table 6.3. In Fig.6.6 and Fig.6.8, the raw data is plotted in colour blue and set behind the cleaned data plotted in green. The procedure is semi-automatic since the fine-tuning of the envelope's bounds position is achieved manually supported by visual appreciation. The main objective during the parameters adjustment is too eliminate the maximum of bad observations while avoiding the discard of too many good values.

Table 6.3: Semi-automatic outliers detection parameter settings and computational time

Feeders	ν	r_u	r_l	No of outliers	running time (s)
F2	168	0.7	0.6	64	9.0
F3	1000	0.6	0.6	503	8.89

6.6.3 Conclusion

An outlier detection technique is subjected to two effects: *swamping* effect (false positive) and *masking* effect (false negatives). Choosing an outlier detection mechanism involves a trade-off between false positives and false negatives. Two outlier detection procedures are proposed: an automatic and a semi-automatic. Both procedures are developed based on robust statistics (median and interquartile range) and have a complexity of $\mathcal{O}(n \log n)$. The fully automatic procedure computes lower bound and the upper bound using Tukey's and its main resistant rule $r = 1.5$. The rule is frequently used as a value that balances false positives and false negatives. For distributions close to normal, Tukey main resistant rule's is expected to have a detection error percentage not exceeding 0.6%. The automatic outlier detection procedure can also accommodate the presence of level-shifts in the data; it implements a robust version of the binary segmentation algorithm. The semi-automatic outlier detection uses the median filter technique to form a bounding envelope that encloses the time series from above and below. Both procedures were tested on large real-world data and have demonstrated good performances in cleansing the data from harmful observations while avoiding the swamping effect.

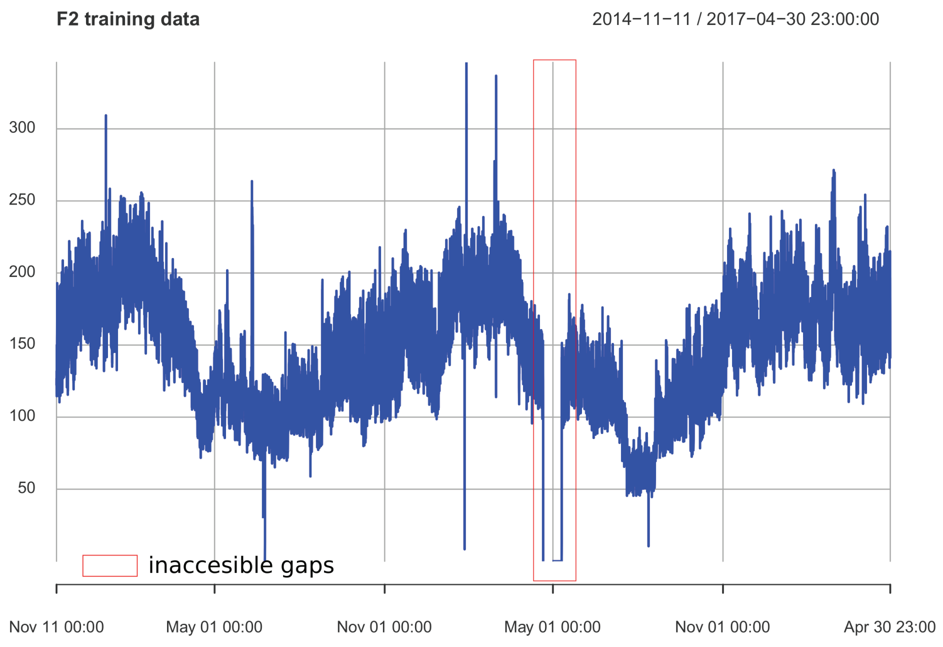

Figure 6.5: F1 feeder's data with inaccessible data gaps

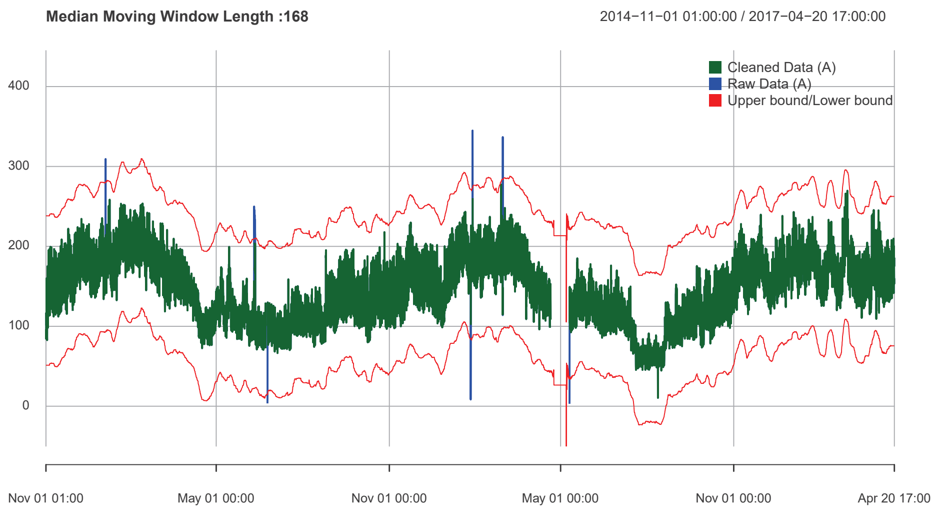

Figure 6.6: F1 feeder's data cleaned with median filter bounds

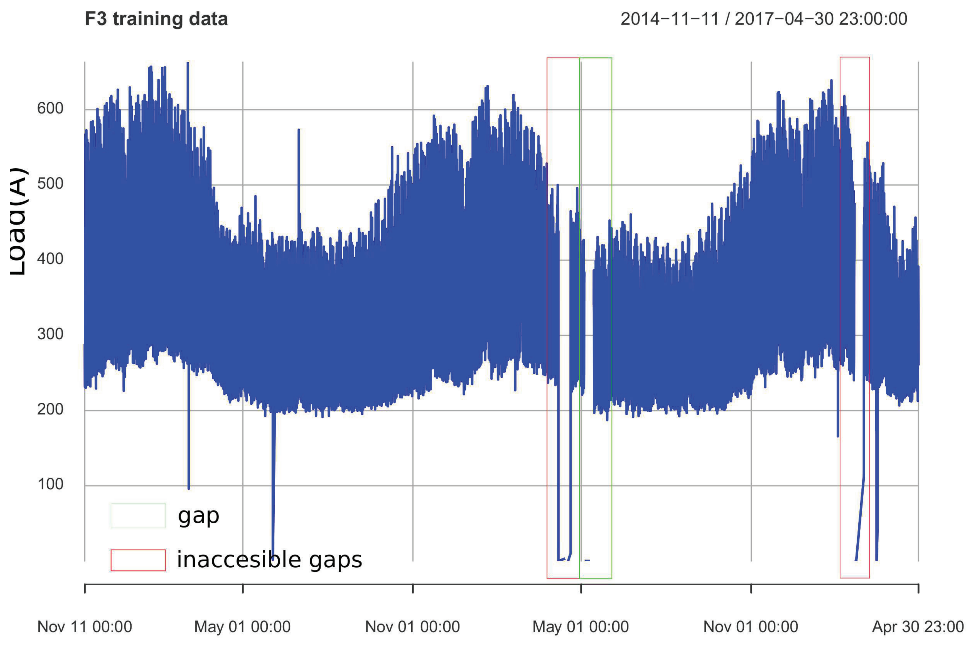

Figure 6.7: F2 feeder's data with inaccessible data gaps

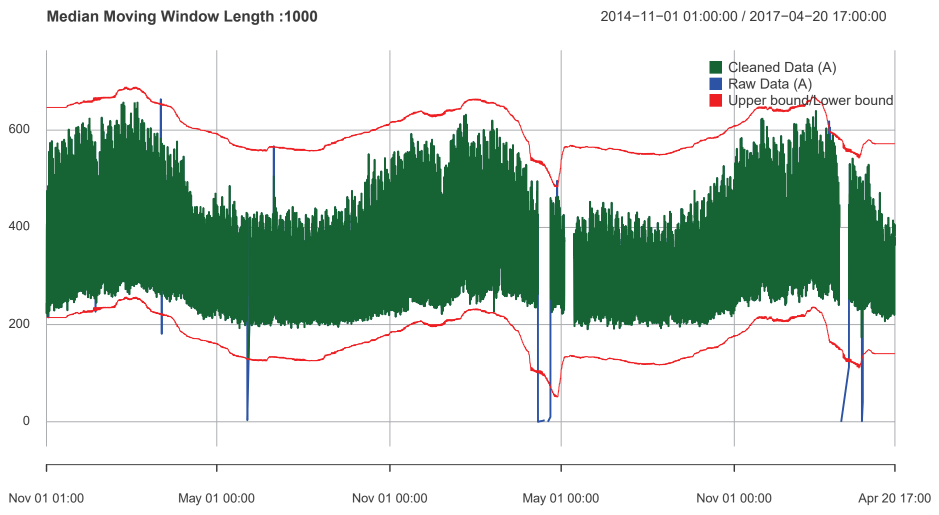

Figure 6.8: F2 feeder's data cleaned with median filter bounds

Chapter 7

Short-term load forecasting

Since the emergence of competition in the energy sector, the need for accurate and reliable energy forecasts has significantly increase across all layers of the industry: generation, transmission, distribution and energy market. STLF is essential for planning, operation and financial activities. Despite an active research, load forecasting remains a challenging task due to the multi-level time dependencies that exhibit load time series and the exogenous variables that must be taken into consideration when developing forecasting models. Forecasting model depend on geographic, climate, economic, and social characteristics. Selecting the most suitable algorithm usually is achieved by testing the algorithms on real-data.

There is not a single model or algorithm that outperforms for all utilities. Usually, companies use several load forecasting methods conjointly since there is no known mechanism that indicates prior conditions for identifying which forecasting method would be more suitable for a given load area. The accuracy evaluation of STLF models requires that the forecasting error which represents the difference between the estimated value and the target value, be computed for each timestamp of the forecasting horizon. The accuracy of STLF predictions relies on three key factors: the class of model that is being used, the quality of the historical input data set used to train the model, and the suitability and quality of the predictor data.

The methodology adopted in this work to forecast large-scale MV feeders' time-series axes on balancing the use of state-of-the-art models with computational resource and time constraints. All algorithms have been implemented in Python 3.5 and 3.6 and R. All studies were carried out on Windows 10 operating system with an Intel Xeon CPU E5-2630 2.4GHz processor.

7.1 Methodology

7.1.1 Datasets

In this study, 342, MV/LV feeders time series are forecasted. Feeders' load and weather time series data cover the period starting from January 2014 and ending in September 2017 at hourly granularity. Datasets are divided into training, and testing sets with the testing set starting in May 2017. *The testing datasets are not used during the models' selection and evaluation procedures but only to generate the final forecasts.* The number of missing observations averages 1.08% across (training and testing data combined) with a standard deviation across feeders of 1.25%. The notation 1.08±1.25% will be used to report mean and variation across feeders. After outlier removal, the fraction of missing data increases to 2.0±1.41%.

Feeders' series were preprocessed with eight preprocessing strategies producing eight training/testing datasets. The first set of four training/testing datasets **r_m**, **r_knn**, **r_kf_imp**, **r_kf_smo** was obtained following the imputation of the raw data (**r**) with no detection and removal of outliers; the missing values were imputed using either unconditional mean (**m**), 10-nearest neighbour (**knn**), Kalman imputation (**k_imp**), Kalman smoothing (**k_smo**). The other set of four training/testing datasets **no_m**, **no_knn**, **no_kf_imp**, **no_kf_smo** was processed by performing a detection/removal outliers procedure (**no**) to the raw data before imputing all missing values using the aforementioned imputation strategies. Note that training and testing data are always preprocessed with the same strategy.

7.1.2 Forecast Error Metrics

To evaluate forecasting performance, the mean absolute percentage error (MAPE) given Eq. 7.1 is computed as the forecast performance metric.

$$\text{MAPE} = \frac{1}{n}\sum_{t=1}^{n} |\frac{\hat{y}_t - y_t}{y_t}| \times 100\% \tag{7.1}$$

where $n = 24$ for 24-steps ahead forecast. The relative error percentage (RPE) given Eq. 7.2 measures one-step ahead forecast performance.

$$\text{RPE} = |\frac{\hat{y}_t - y_t}{y_t}| \times 100\% \tag{7.2}$$

MAPE and REP are computed using two ground truth datasets; the raw testing data (**r**) and the testing data cleansed from outliers (**no**). The latter allows obtaining 'clean' accuracy results since the forecast performances reduce when the ground truth data contain outliers. Note that training and testing data are always preprocessed with the same strategy.

The median and the median absolute deviation (MAD) are used as robust alternatives to the mean and the standard deviation respectively in the case study section. The MAD is defined as

$$\text{MAD} = \text{median}|X_i - \tilde{X}| \quad \text{with } \tilde{X} = \text{median } X. \tag{7.3}$$

7.1.3 Inputs Selection

Endogenous variables

The time series forecasting problem is difficult because of the autocorrelation between observations. The temporal dependency adds a complexity to the forecasting problem that requires specific handling of the data when fitting and evaluating models. Typically, the temporal dependency is unknown and must be uncovered from a detailed analysis;

these endogenous input variables refereed as *lag inputs* must be diagnosed and specified. Lag inputs selection is part of model identification in classical time series modeling. This selection is usually achieved via an analysis of the autocorrelation function which measures the extend that a time series is related with a delayed copy of itself. Fig. 7.1 and Fig. 7.2 display the lag plots of one MV feeder load time series. The horizontal axis shows lagged values of the feeder's data, each graph shows $\boldsymbol{y}_t$ plotted against $\boldsymbol{y}_{t-k}$ for 48 values of k split across the two figures. The autocorrelation between $\boldsymbol{y}_t$ and $\boldsymbol{y}_{t-k}$ is given by

$$r_k = \frac{\sum\limits_{t=k+1}^{n} (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum\limits_{t=1}^{n} (y_t - \bar{y})^2}, \tag{7.4}$$

where n is the length of the time series. In Fig. 7.1 and Fig. 7.2, the lines connect points in chronological order and the dotted blue lines represent the 95% confident threshold. The linear relationship between lagged values is strongly positive at lags 1, 24 and 48. The corresponding autocorrelation coefficients r_k calculated with Eq. 7.4 are as follow: $r_1 = 0.954$, $r_{24} = 0.919$ and $r_{48} = 0.769$. These coefficients highlight the critical importance of lagged values in this particular case of STLF. The strong linear relationship between lagged values proves that passed observations of feeder's load highly influence future demand. In particular, lag 1, for which the autocorrelation coefficient is $r_1 = 0.954$, underlines the strongest lagged value that most influences future load behaviour; this is a determinant criterion in the selection of *the recursive forecasting strategy*, which is discussed in the following section. Besides, the next section will confirm that weather variables are not prominent in the forecasting problem addressed in this thesis. The autocorrelation function analysis signifies that historical load values are the best predictors of future load values and the most important.

Exogenous and dummy variables

Besides the lagged values series, additional contextual data are fed to the network that provides exogenous information related to the environment in which the feeder's load

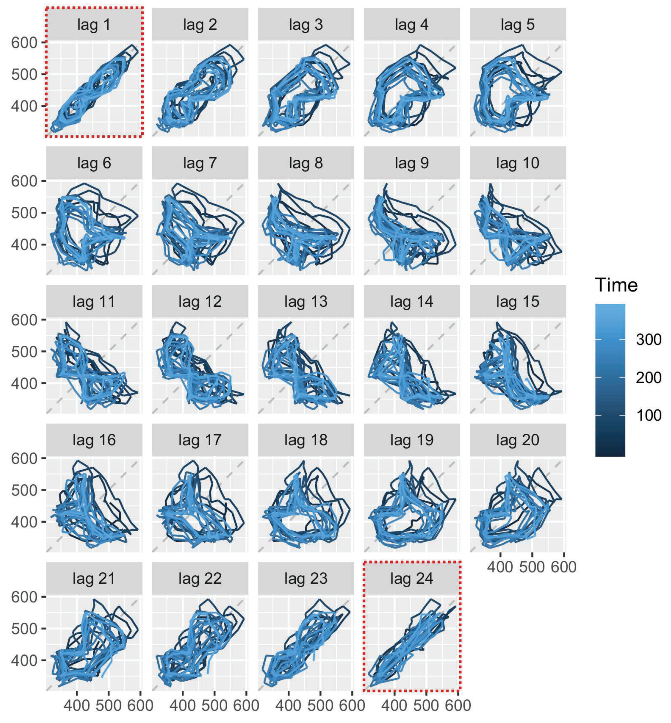

Figure 7.1: Lag plots for the first 24 lags

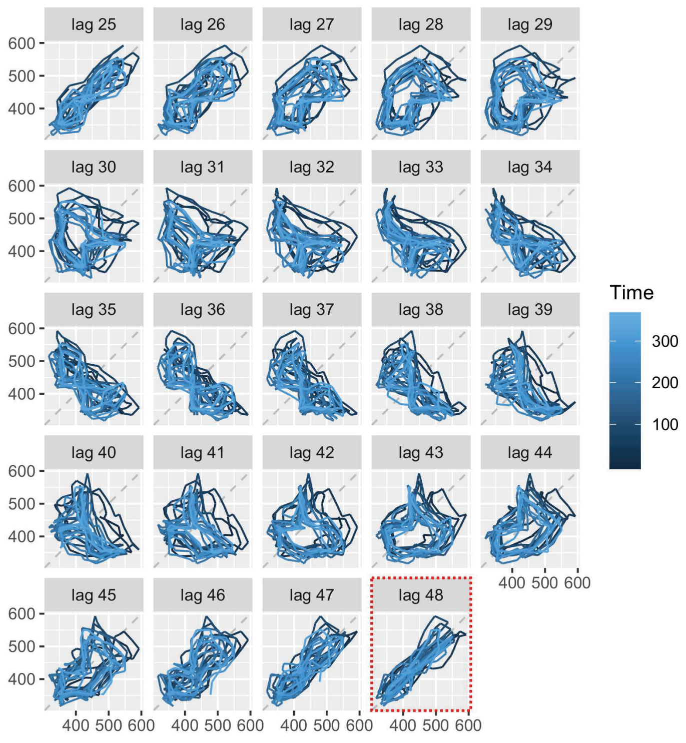

Figure 7.2: Lag plots for the 25th to 48th lags

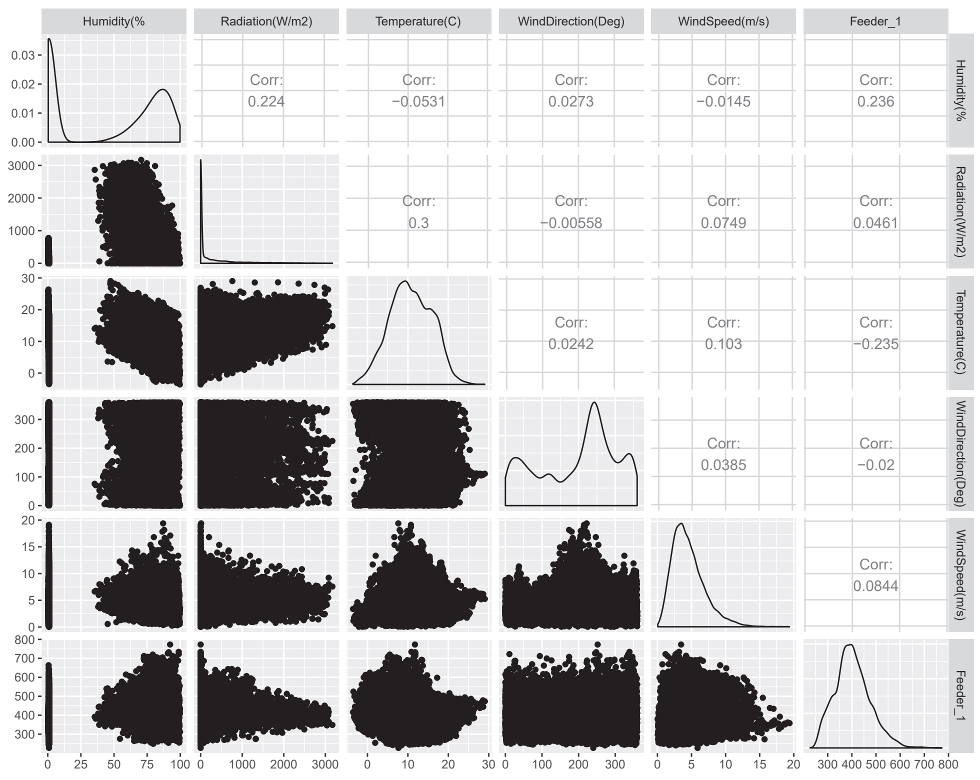

Figure 7.3: Scatterplot matrix of MV feeder load and weather predictors

evolves. Dummy variables are used to represent the calendar cycles (month and days of the week) and the day-type (weekday, weekend and holidays). High dimensional input space leads to an excessive number of input weights and poor performance [56]. To reduce the input space, dummy calendar variables are encoded with sinusoidal functions rather than using one-hot encoding as in [80, 104]. Weather inputs were selected based on correlation analysis between weather and load variables. The matrix provided in Fig 7.3 displays the scatterplots for one feeder's series against the weather data variables; this helps to visualise the relationship between the variables. In general, the matrix reveals a low correlation between the feeder's load and the weather variables. This agrees with recent studies that report 80% of heating and cooling systems in the UK are supplied with gas energy. The Pearson's correlation coefficients provided on the plot indicate that the temperature and the humidity variables as the most linearly correlated to the load with -0.235 and 0.233 correlation factors.

7.1.4 Inputs normalisation and specification

Table 7.1 provides details on the model inputs used to model the MV feeders load time series. The input vector $\boldsymbol{x}$ contains 13 inputs reported in Table 7.1. Each input is standardized with z_{score} such that each x_i was computed as $\hat{x}_i := (x_i - \mu_i)/\sigma_i$, where μ is the input mean and σ is its standard deviation.

Table 7.1: Network Inputs

Variables	Inputs (*)
Load lags (current)	y_{t-1} or $\hat{y}_{t-1}$
	y_{t-24}, y_{t-48}
Weather forecast	temp(t+h), hum(t+h)
Calendar cycles	$\cos(2\pi\mathfrak{h}/24)$, $\sin(2\pi\mathfrak{h}/24)$
	$\cos(2\pi\mathfrak{d}/7)$, $\sin(2\pi\mathfrak{d}/7)$
	$\cos(2\pi\mathfrak{m}/12)$, $\sin(2\pi\mathfrak{m}/12)$
Day-type ($\mathfrak{D}_{\mathfrak{T}}$)	Weekday/Weekend $= 0\|1$
	Holidays $= 0\|1$

(*) Notes : t is the current time, y denotes the current and ŷ denotes one-step-ahead forecast, $\mathfrak{h} \in H = \{0, \ldots, 23\}$ for time of day, $\mathfrak{d} \in D = \{0, \ldots, 6\}$ day of week with Monday $= 0$ and Sunday $= 6$, $\mathfrak{m} \in M = \{1, \ldots, 12\}$ month of year with January $= 1$ and December $= 12$.

7.1.5 Multistep-ahead forecasting strategy

The autocorrelation analysis of feeder's data has shown the significance of lag values in the short-term load forecasting problem. The highest correlation coefficient, $r_1 = 0.954$, was calculated for lag $k = 1$. Hence, this indicates that the feeder observation y_{t-1} is the most influential input in the STLF problem. The measurement at y_{t-1} will be available in the database if the prediction target is to produce one-step-ahead forecast, however, if the forecasting horizon $h > 1$, the y_{t-1} value will not be directly accessible from the historical data. It must be either estimated or not included in the network input vector. This finding has motivated the implementation of a recursive prediction strategy to produce 24-step-ahead forecasts (24 values) where predictions are generated at midnight (12 am). Let $\mathbf{y} = \{y_{1:n}\}$ be a univariate feeder time series comprising n observations, and the aim is to forecast the next 24 values of the time series. The underlying process is estimated by a model of the form m and an error term ϵ_t given by

$$y_t = m(\mathbb{y}_{t-1}, \mathbf{z}_t^{\mathcal{P}}; \boldsymbol{\theta}) + \epsilon_t \tag{7.5}$$

where $\epsilon_t \sim N(0, \sigma_t^2)$, $\boldsymbol{\theta}$ are the model parameters, $\mathbb{y}_t = [y_t, y_{t-23}, y_{t-47}]'$ and $\mathbf{z}_t^{\mathcal{P}}$ is the vector of the exogenous inputs (either known or forecast for time t) depending on the current hour, day, month and day type as reported in Table 7.1 and summarised with the parameter $\mathcal{P} := (\mathfrak{h}, \mathfrak{d}, \mathfrak{m}, \mathfrak{D}_{\mathfrak{T}})$. For simplicity, from hereon, we drop the dependence on θ in $m(\mathbb{y}_{t-1}, \mathbf{z}_t^{\mathcal{P}}; \boldsymbol{\theta})$ and use the shorthand notation $m(\mathbb{y}_{t-1}, \mathbf{z}_t^{\mathcal{P}})$. The recursive prediction consists of repeating one-step-ahead prediction several times using the previous forecast as input [24]. We compute forecasts recursively for $h = 0, \ldots, 23$ as

$$\hat{m}^{(h)}(y_{t-1:t-47}, \mathbf{z}_{t:t+h}^{\mathcal{P}}) = m([\hat{m}^{(h-1)}(y_{t-1:t-47}, \mathbf{z}_{t:t+h-1}^{\mathcal{P}}), y_{t+h-24}, y_{t+h-47}], \mathbf{z}_{t+h}^{\mathcal{P}}) \tag{7.6}$$

where the recursion is initialized by $\hat{m}^{(-1)}(y_{t-1:t-47}, \mathbf{z}_{t-1}^{\mathcal{P}}) := y_{t-1}$, and we use the conventions $y_{t-1:t-47} := [y_{t-1}, \ldots, y_{t-47}]$ and $\mathbf{z}_{t:t+h}^{\mathcal{P}} := [\mathbf{z}_t^{\mathcal{P}}, \ldots, \mathbf{z}_{t+h}^{\mathcal{P}}]$. Training and testing datasets are processed either by imputing missing values only or by detecting and removing outliers first followed by the imputation of all missing observations.

7.1.6 Learning algorithm selection

The learning algorithm selection was carried out through a preliminary investigation of three types of multivariate models: Prophet, LSTM and FDNN. The algorithm was chosen based on two criteria: the model's training computation time and forecast accuracy. All three algorithms were trained on five feeders' time series to produce a one-step-ahead prediction. The forecast accuracy results are reported in Tables 7.2, 7.3 and 7.4. The feeders' raw training and testing data are plotted in Fig. 7.4. The datasets' naming convention used in this work are reminded at the bottom of Tables 7.2, 7.3 and 7.4. For each of the modelling technique, the best point-forecast performances are highlighted in blue.

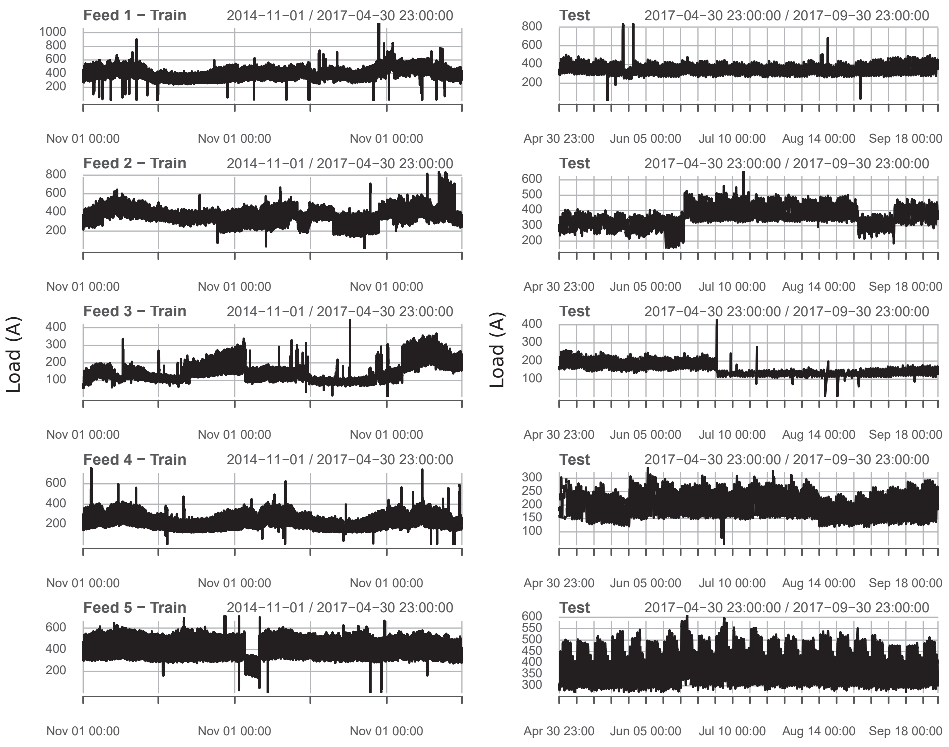

Figure 7.4: Time plots for the training (left) and testing (right) raw datasets of "Feed 1" to "Feed 5"

The forecast accuracy results described in Tables 7.2, 7.3 and 7.4 provide several key outcomes: 1) the automatic outlier detection can potentially improve forecast accuracy whichever of the three modelling algorithms is used, 2) outlier removal does not always improve one-step-ahead forecast accuracy, particularly with the FDNN, 3) the deep learning algorithms outperform Prophet on these datasets. However, based on Tables 7.3 and 7.4 results, FDNN outperforms LSTM on most of the time series (4 out of 5) which disagree with the recent claims concerning the excellent results of LSTM's on time series forecasting problems. In addition, FDNN seems to be more robust against outliers than LSTM. The results in the tables are not further discussed here as they are only shown to support the choice that was made for the algorithm selection.

LSTM has been advocated in a few recent articles as to be the state-of-the-art technology for short-term load prediction. In 2017, a comparative performance analysis of RNN's architecture applied to the short-term load forecasting problem was proposed in [18]. At the time of publication, there was no existing study reporting on the imple-

Table 7.2: **Prophet**- One-step-ahead forecast RPE(%) results for 5 feeders

Datasets	Feed 1	Feed 2	Feed 3	Feed 4	Feed 5
r_m_no	6.69	4.89	6.14	12.06	2.15
no_m_no	5.21	5.01	4.73	9.44	2.44
r_knn_no	6.36	4.51	6.47	11.60	2.08
no_knn_no	5.37	4.45	4.73	**9.41**	2.43
r_kf_imp_no	6.04	4.47	5.76	11.60	2.09
no_kf_imp_no	**4.22**	4.27	4.77	9.58	2.23
r_kf_smo_no	4.79	4.25	5.42	11.76	**1.91**
no_kf_smo_no	4.59	**4.06**	**4.64**	9.88	2.09

Table 7.3: **LSTM** - One-step-ahead forecast RPE(%) results

Datasets	Feed 1	Feed 2	Feed 3	Feed 4	Feed 5
r_m_no	2.41	4.60	5.72	3.08	2.24
no_m_no	**2.01**	4.73	6.06	3.66	2.30
r_knn_no	3.11	4.39	4.74	3.32	2.28
no_knn_no	2.15	5.02	5.31	**2.72**	**2.06**
r_kf_imp_no	2.69	4.32	4.97	4.12	2.51
no_kf_imp_no	2.03	**4.21**	6.14	3.04	2.38
r_kf_smo_no	2.24	4.41	4.80	**2.72**	2.21
no_kf_smo_no	2.61	4.36	**4.54**	3.95	2.15

Table 7.4: **FDNN** - One-step-ahead forecast RPE(%) results

Datasets	Feed 1	Feed 2	Feed 3	Feed 4	Feed 5
r_m_no	2.91	3.35	6.42	3.11	2.21
no_m_no	2.92	3.80	4.04	3.04	2.08
r_knn_no	2.68	**3.08**	3.52	2.85	2.15
no_knn_no	**2.31**	3.19	3.56	2.94	**2.05**
r_kf_imp_no	2.57	3.34	**3.32**	3.33	2.15
no_kf_imp_no	2.41	3.15	3.53	**2.57**	2.07
r_kf_smo_no	3.40	3.26	3.59	3.52	2.10
no_kf_smo_no	2.59	3.39	4.16	2.67	2.08

Note : Training and testing data preprocessing strategies are indicated in black; ground truth data are indicated in grey (r:raw data, no:no outliers, m: Unconditional mean imputation, kf_imp:Kalman filter imputation, kf_smo:Kalman smoother imputation, knn: 10-Nearest Neighbour) mentation of LSTM for real-world load time series prediction. Since, only a few articles analysed and reported on the subject [81, 108]. After careful reading of these studies, it transpired that the proposed frameworks only addressed one-step-ahead forecasts. Considering that STLF relate to the problems of up to one week ahead forecasts, a clear statement on the targeted forecasting horizon is a piece of essential information that should not be omitted. For instance, in [108] and [81], the one-step-ahead load forecast of residential house-hold is addressed with LSTM, but regrettably, the authors did not emphasise this critical point.

Multistep-ahead prediction is a more challenging problem that requires thoughtful consideration of the forecasting strategy to adopt. The only articles that address the implementation of LSTM to predict day-ahead load forecasts can be found in [35] and [24]. Both articles report on multistep-ahead forecasts of building electricity consumption using a recursive strategy and an encoder-decoder (known as sequence to sequence) architecture. Direct and recursive strategies are discussed in Sec. 7.1.5. The forecasting problem presented in these studies differ from the one discussed in this thesis from two key aspects: 1) the number of time series to be modelled and the targeted type of load. In [24], the authors show the strong correlation between the outdoor temperature and the building's electricity load demand, which emphasises the importance of having weather variables in the model's input vector. Since weather predictors are the main influences on buildings' load behaviour, the decoder necessitates weather forecasts and dummy variables essentially to generate multistep-ahead forecasts. The main issue with both studies is they only compare the forecast performances against those produced by other complex LSTM-based architectures or classical statistical methods; none of the studies included a FDNN or a standard multilayer perceptron network in their study. Yet, one of the main outcomes provided in the review proposed in [18] states the following: *'LSTM and GRU achieve outstanding results in many sequence learning problems, the additional complexity of the complicated gated mechanisms seems to be unnecessary in many time series predictions tasks.'*. In [51], Schmidhuber Jurgen (one of the developers of the LSTM architecture) and his collaborators even suggested using LSTM *'only on tasks where traditional time* *window based approaches have failed'.*

Nevertheless, the preliminary focus of this work was to efficiently train these models and assess the feasibility of using them for multistep-ahead forecasts. The implementation of LSTM models was done with the `Keras` deep learning library. Training large-scale LSTM-based models turned-out to be very much time-consuming, involving the tuning of many parameters. Furthermore, the automated architecture selection process through a rigorous model evaluation added more complexity to the problem. And besides, the implementation of the recursive approach with LSTM and the time taken to produce 24-step-ahead forecast discarded the technology conclusively. In a matter of fact, the strength of LSTM stands in its capability to remember single events for a very long time, if such extended memory is not required for the forecasting problem at hand, there is no requisite that justify the use of such complex technologies. Depending on the time series forecasting problem, complex architectures such as LSTM or CNN might not even be adequate. In definitive, the FDNN algorithm was adopted for the rest of this work as the architecture largely satisfied the two criteria mentioned above.

On a side note, the forecast accuracy results reported in Table 7.4 have raised some concerns regarding the suitability of removing outliers from the data, particularly in the training datasets. To dissipate the uncertainty raised by Table 7.4, a robust analysis of forecast accuracy is reported in Table 7.5. The robust statistics relate to one-step-ahead forecast accuracy performance across 342 feeders. The results highlighted in blue in Table 7.5 show that models that were trained on a fully cleansed dataset (i.e **no_knn_no**) perform better than those trained on imputed data only. Nevertheless, the results also reveal a slight increase in model variance; this will be discussed in later sections.

7.1.7 Robust loss functions

The choice of the loss function is a key part in deep neural network model training, $\mathcal{L}_p$-norm losses are commonly used in supervised learning [54]. During the learning process, neural networks are optimised with a stochastic backpropagation algorithm which computes the

Table 7.5: FDNN - Robust statistics for one-step-ahead forecast accuracy across 342 feeders

	MEDIAN RPE(%)	MAD RPE(%)
r_m_no	4.53	2.21
no_m_no	**4.50**	**2.46**
r_knn_no	3.98	2.14
no_knn_no	**3.86**	**2.26**
r_kf_imp_no	4.06	2.07
no_kf_imp_no	**3.85**	**2.15**
r_kf_smo_no	4.20	2.14
no_kf_smo_no	**4.06**	2.05

gradient g of a loss function $\boldsymbol{\mathcal{L}}(\boldsymbol{\Phi})$ with respect to parameters $\boldsymbol{\Phi}$, the weights of the deep neural networks. Given a training set $\mathcal{S} := \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$ of inputs and target pairs , the parameters $\boldsymbol{\Phi}$ are learned by minimising the empirical risk function given as

$$\hat{\boldsymbol{\Phi}} = \arg\min_{\boldsymbol{\Phi}} \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{\mathcal{L}}\big(f_{\boldsymbol{\Phi}}(\mathbf{x}^{(i)}, y^{(i)})\big) + \lambda \mathcal{R}(\boldsymbol{\Phi}) \tag{7.7}$$

where f is a nonlinear function such that for $i \in [1, \ldots, n]$, $f(\mathbf{x}^{(i)}, \boldsymbol{\Phi}) = \hat{y}^{(i)}$, $\boldsymbol{\mathcal{L}}$ is a particular loss function and, $\lambda\mathcal{R}(\boldsymbol{\Phi})$ a regularization term.

The gradient $g_{\boldsymbol{\Phi}} = \sum_{i=1}^{n} \nabla_{\boldsymbol{\Phi}} \mathcal{L}(f_{\boldsymbol{\Phi}}(x^{(i)}), y^{(i)})$ is used to update the parameters $\boldsymbol{\Phi}$ in that direction such that $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} - \eta \times h(g_{\boldsymbol{\Phi}})$, $\eta \in [0, \ldots, 1]$ being the learning rate and $h(.)$ varies upon the applied stochastic optimization algorithm. Hence, the parameters are updated in proportion to the product of the residual error and inputs, which causes the backpropagation algorithm to be heavily dependent upon the quality of the training data.

Robust training of ANNs in the presence of gross errors was investigated by Liano [88] and Khamis [9]. Liano studied the mechanism by which outliers affected ANNs outcome using M-estimators. Khamis conducted an analysis of variance (ANOVA) tests on training and test data, to evaluate the influence of the percentage-outliers factor and the magnitude-outliers factor on the ANNs performance. The study reported that both, the magnitude of outlier observations with respect to the unconditional mean and their percentage in the data are two contributing factors to the decrease of ANNs prediction accuracy. The square loss $\mathcal{L}(y, \hat{y}) = (y - \hat{y})^2$ is one such function that is well-suited for the purpose of regression problems, however, it suffers from one critical weakness: outliers in the data are penalized heavily by the squaring of the error. The absolute loss $\mathcal{L}(y, \hat{y}) = |y - \hat{y}|$ avoids the problem of weighting outliers too strongly by scaling the loss linearly instead of quadratically. Thus, a first design decision the analyst can make to reduce the bad effects of outliers on the model's parameter estimation is to choose a robust loss function. It exists other loss functions than the absolute loss that are less popular but robust against outliers. The *Log-Hyperbolic Cosine loss* (LOGC) in Eq. 7.8 and *Huber loss* (HL) in Eq. 7.9 are those to guarantee robustness towards outliers.

$$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \log[\cosh((y - \hat{y})] \tag{7.8}$$

$$\mathcal{L}_\delta(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{for } |y - \hat{y}| \leq \delta \\ \delta|y - \hat{y}| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \tag{7.9}$$

Table 7.6 reports on the impact of the loss function on forecasts accuracy. The table displays the statistics of one-step-ahead forecast across 342 MV feeders trained with FDNN. The data are preprocessed with mean imputation only (no outliers removal). The best forecasts are produced by the models trained on MAE loss function; these results are highlighted in colour blue. The results show that the models trained with mean square error (MSE) perform the worst; their corresponding results are highlighted in colour red. `TensorFlow` is a powerful and flexible deep learning library. It offers a wide range of

Table 7.6: Statistics of one-step ahead forecasts distribution with 4 different loss functions

Loss functions	**MSE**	**HL**	**LOGC**	**MAE**
No feeders	342	342	342	342
mean MAPE (%)	8.679	8.677	8.712	7.981
variance MAPE (%)	10.234	9.595	9.627	8.137

note: mean square error (MSE), mean absolute error(MAE), Huber loss (HL), and logistic cosh loss

preprogrammed loss functions and users also have the option to implement customised functions. However, we have found `TensorFlow` to be computationally less efficient than `Scikit-learn` on this large scale problem [109]. Therefore, the FDNN models were implemented with the `Scikit-learn` Python library because it is computationally very efficient. `Scikit-learn` offers a class of deep learning models for regression problems that are fast to train; however, the only loss function available to train the models is the mean squared error (MSE).

7.1.8 Model's architecture

FDNN are hierarchical and parametric models composed of neurons arranged in $L \in \mathbb{N}$ layers which are connected by weighted edges. There are three types of layers: the vector of inputs $\boldsymbol{x} \in \mathbb{R}^d$, multiple hidden layers $l \in [1, \ldots, L-1]$ containing each N_l number of neurons and the output layer where $N_L = 1$. Given a FDNN architecture where d, L and $\{N_l\}_{l=1}^{L}$ are provided, *training the network* consists of learning the affine-linear functions $\{\boldsymbol{W}_l\}_{l=1}^{L} = \{\boldsymbol{M}_l(\cdot) - \boldsymbol{b}_l\}_{l=1}^{L}$ yielding the network architecture $\Phi : \mathbb{R}^d \to \mathbb{R}^{N_L}$ given by Eq. 7.10 and, the model given by Eq. 7.11

$$\Phi = \{(M_l, b_l)\}_{l=1}^{L} \tag{7.10}$$

$$\Phi(\boldsymbol{x}) = \boldsymbol{W}_L \psi(\boldsymbol{W}_{L-1} \psi(\ldots \psi(\boldsymbol{W}_1(\boldsymbol{x})))), \tag{7.11}$$

In this work, FDNN models were trained to forecast load for each feeder. Network architectures were optimised by grid search via a nested-rolling-origin-validation procedure which is discussed in Sec. 7.1.9. Table 7.7 provides details on the network architecture and training algorithm specifications.

7.1.9 Model training and evaluation

Cross-validation is a statistical procedure that has two goals: 1) compare the performance of multiple models and find the best for the available data and, 2) estimate the generalisation performance of a model. Common cross-validation techniques found in the

Table 7.7: Specification of models' architecture and training algorithm

Tuning parameters method	Grid search
Inputs standardization	z-score
Hidden layers	$[1, 2, 4, 6, 8]$
Cells (per layer)	$[2, 4, 6, 8, 10]$
Batches	$[16, 32, 64]$
Activation	ReLU
Solver	ADAM
ADAM hyperparameters default settings	η= 0.001, $\beta_1 = 0.9$ $\beta_2 = 0.999$, $\epsilon = 10^{-8}$
Cost function	MSE
Early stopping	True

STLF literature are out-of-sample (OOS) evaluation and k-fold cross-validation. In general, cross-validation techniques yield to good model selection performances. However, particular care should be taken when the aim is to estimate the generalisation error of a model [8]. The out-of-sample method is a simple cross-validation technique that suffers from issues of high variance which can lead to overfitting in model selection due to *information leak* [27]. Hence, a resampling method such as k-fold cross-validation, is more suitable. However, k-fold cross-validation implementation is not straightforward when it comes to time series forecasting. Because of the serial correlation between errors in the training and testing datasets of time series data, training and testing sets are not independent, which invalidates the cross-validation [17]. In addition, the traditional setting of k-fold cross validation used future observations to predict the past.

To overcome the shortcomings of the standard k-fold cross validation, two procedures are proposed to tune the parameters and evaluate the models' generalisation performance: the NROV and the NAk-foldV illustrated in Fig. 7.6 and Fig 7.7 respectively. Each procedure implements a pair of nested loops which offers an unbiased and robust model performance evaluation technique. Model selection and model fitting procedures are integral parts of the entire model evaluation process, unlike the standard algorithm which infer hyperparameters and parameters separately. The NROV makes uses of the basic rolling-origin evaluation discussed in [113], also known as anchored walk-forward evaluation in financial optimisation. The nested validation procedure is described in the flow chart provided in Fig. 7.5. Feeders' time series are partitioned multiple times in training, validation and testing sets. Each time, the training period is moved further ahead with its origin fixed at the beginning of the series. The advantages of NROV are 1) several out-samples errors referred to as *forecast origin* in [113] are obtained which gives a better understanding of how the models perform, 2) the strategy mimics the production scenario where forecasting models are retrained on new coming historical data, and 3) the procedure output multiple optimum model architectures.

After each training period $p_i, i = [1, \ldots, m]$, an optimum architecture is selected out of k pre-selected architectures. The selection is made based on the best one-step-ahead forecast performance achieved on the validation data of each period. The optimum architecture is retrained, and forecasts are produced on the test data (out-of-sample). At the end of the NROV, there are p optimum architectures available; in Fig. 7.6, *Model selection 1* compares the relative percentage error (RPE) achieved by each optimum architecture on the out-of-sample data and picks the best model that achieved the best performance.

Arguably, *model selection 1* carries the risk of biasing the model selection because different test sets are used to quantify the performance of the various optimal architectures. As a comparison, two alternative approaches are proposed: *Model selection 2* and *Model selection 3*. In *Model selection 2*, the best architecture is selected over all possible models k as illustrated in Fig. 7.6 at the bottom of the figure. The performance of each k architecture on all m validation period is averaged, compared, and the best model is selected. *Model selection 3* applies an adapted k-fold strategy to split the training data in twelve equal splits. The procedure keeps the two last splits as validation and test data, and these remain identical for each of the 10 periods. The best architecture selection is the same as *Model selection 1*. The statistics for the performance of 24-steps ahead forecast across 342 MV feeders using *Model selection 1*, *Model selection 2* and *Model selection 3* are shown in Fig. 7.8. The preprocessed data used for this model evaluation study is the fully cleaned 10-nearest neighbours dataset (NO_KNN). The results show that NROV and NAk-foldV are both good model evaluation procedures that offer a robust measure of forecasts uncertainty through narrow confidence bands and a performance distribution

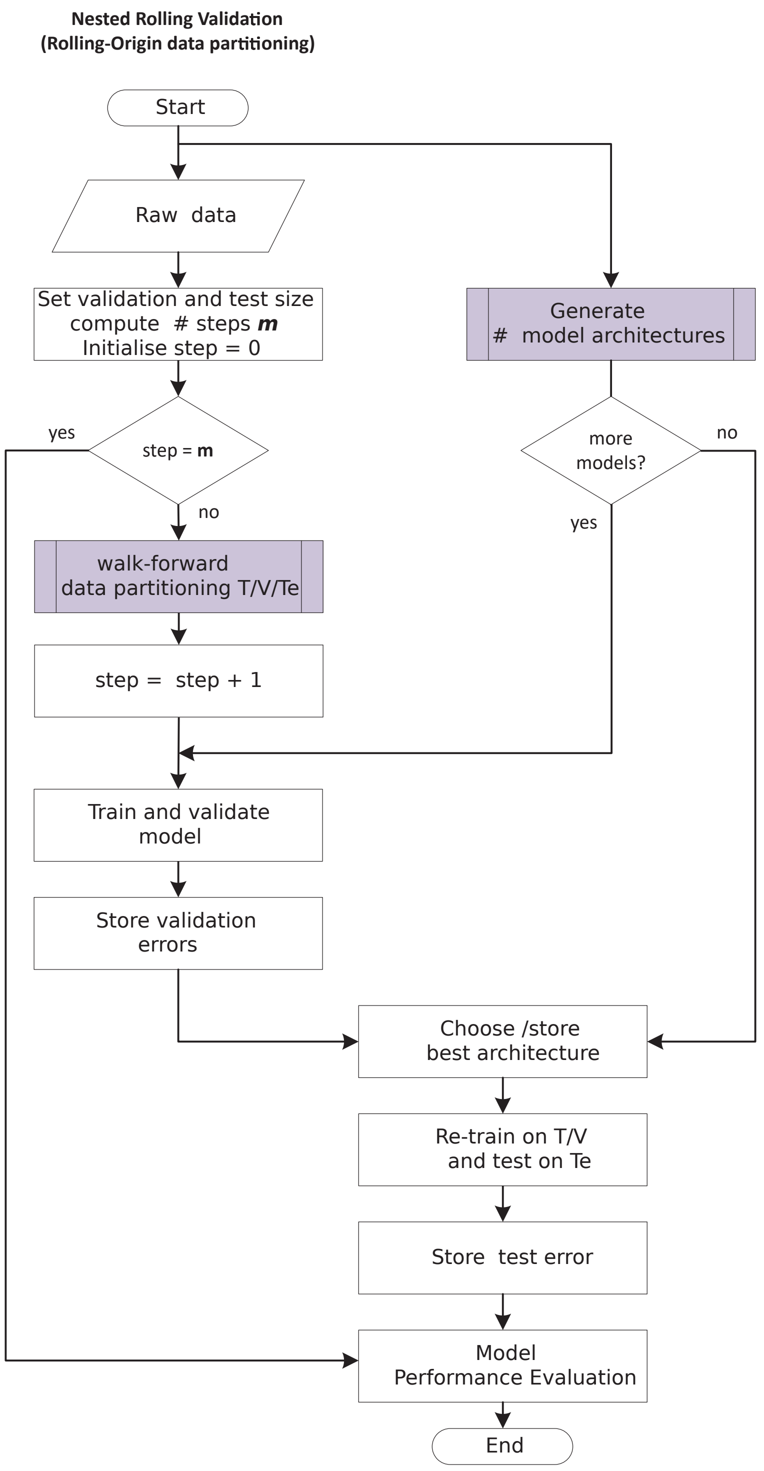

Figure 7.5: Nested Rolling-Origin Validation(NROV) flow chart - T:Training; V:Validation; Te:Test

close to normal.

In the remainder, NROV is used because the procedure does not depend on a single choice of testing data (i.e. the most recent time window). Once the best models have been identified, they are retrained on the entire training data and forecasts are produced on the new sets of test data.

7.2 Case study

7.2.1 24-step-ahead forecast analysis for individual feeder

To evaluate forecasting performance, the mean absolute percentage error metric referred to as MAPE in Eq. (7.1) is used. The MAPE was computed using two ground truth datasets; the raw testing data (**r**) and the testing data cleansed from outliers (**no**). The latter provides a 'clean' forecast accuracy because accuracy results are reduced when the ground truth data contain outliers. Note that training and testing data are always preprocessed with the same strategy.

The achieved accuracies for 24h-ahead forecasts of five feeders are reported in Table 7.8 and Table 7.9. In both tables, the left-hand-side column indicates in black the data preprocessing strategies and in gray which ground truth data were used to compute the MAPE (i.e. **r_kf_imp_no** must be understood as the model is trained and tested with raw data imputed with the Kalman filter and the ground truth has been cleaned from outliers). Table 7.8 describes the accuracy results obtained for the forecasters trained and tested on raw data with the missing values imputed. Table 7.9 outlines the performances achieved by the forecasters trained and tested on data fully processed (outliers removal + imputation). The results highlighted in red indicate the worst forecasting performances between Table 7.8 and Table 7.9.

Fig. 7.4 shows the time plots of the training and testing raw datasets for each of the five feeders under consideration. The plots help highlight the presence of gross errors

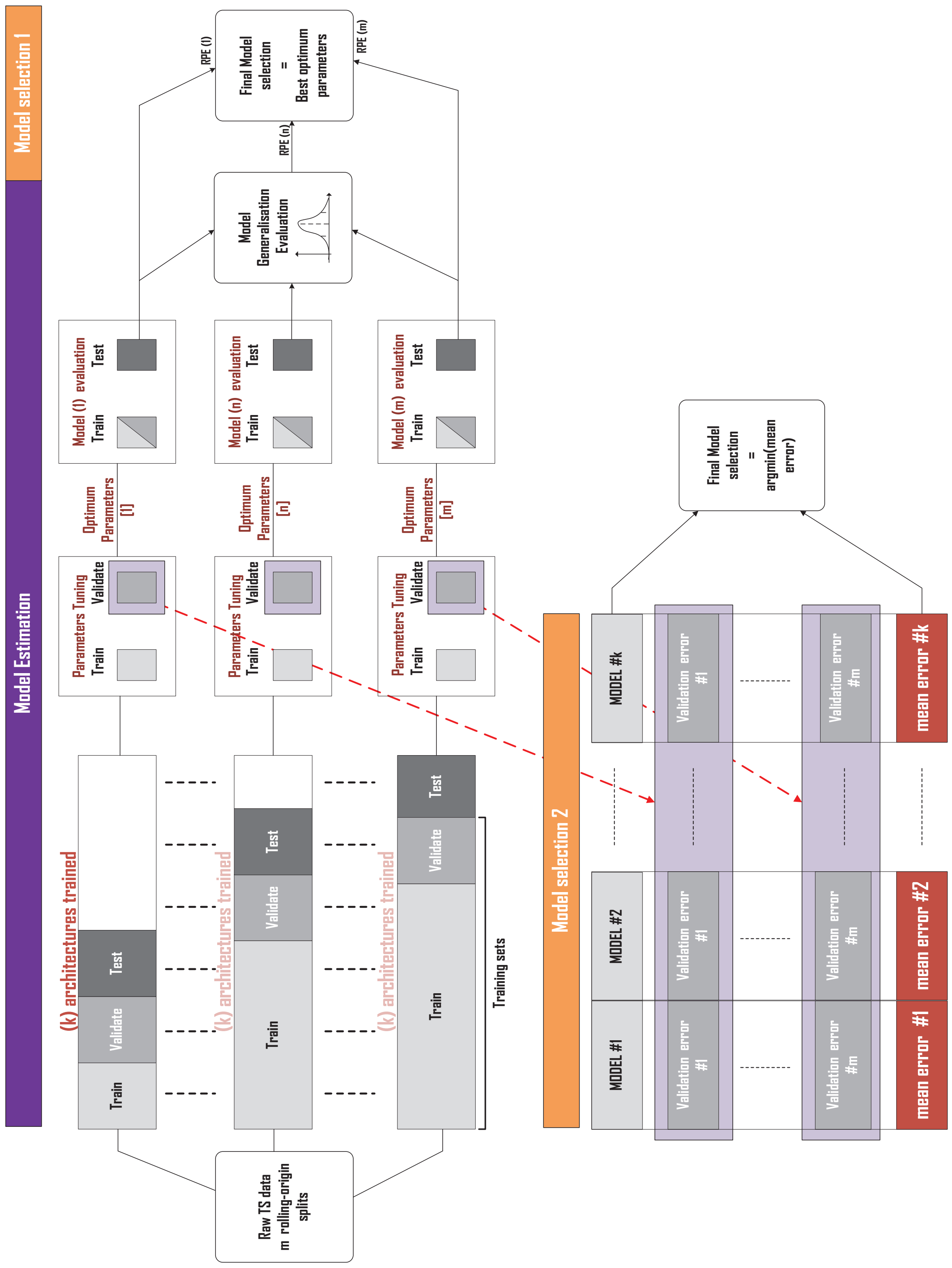

Figure 7.6: Nested rolling-origin validation (NROV) model selection 1 and model selection 3

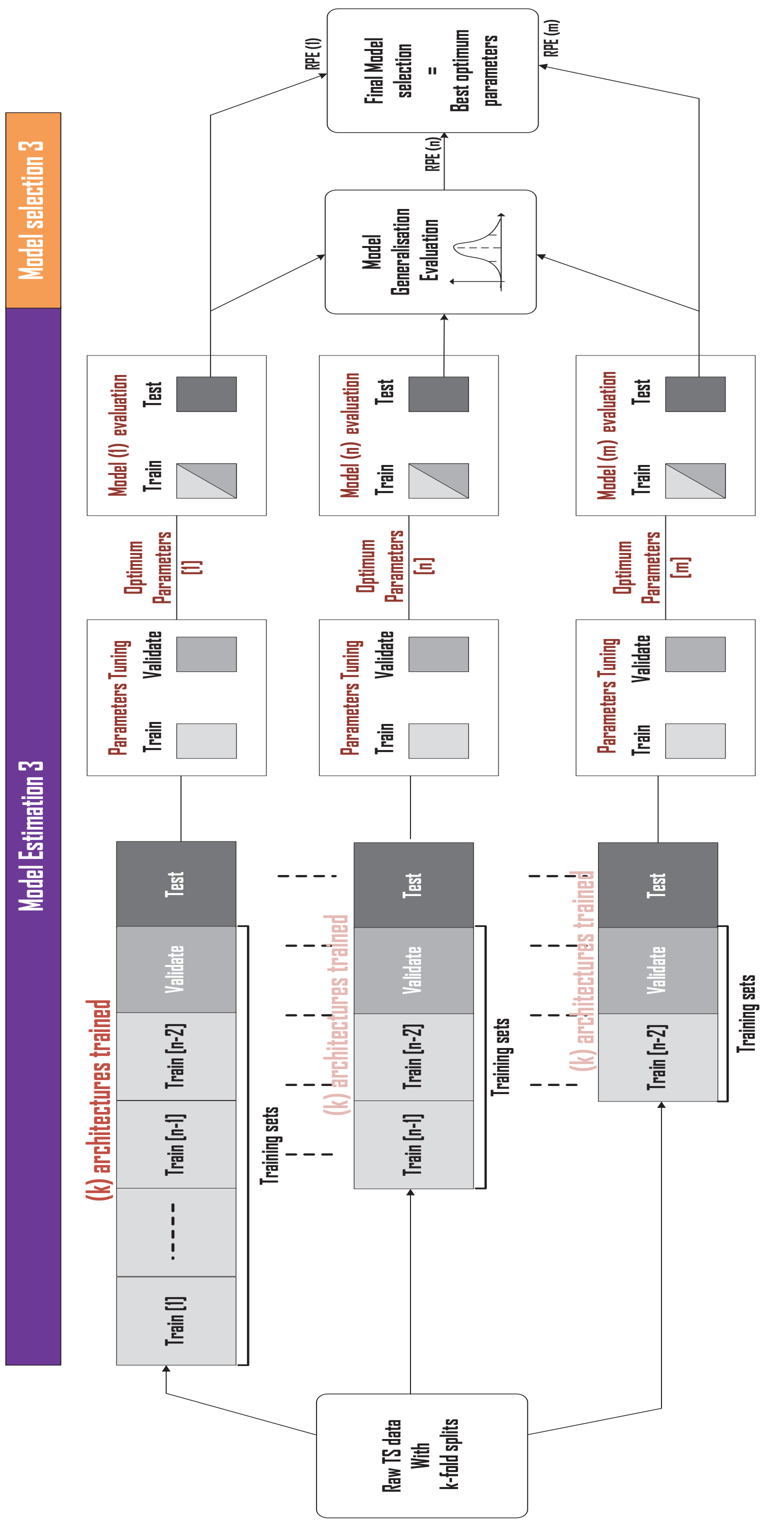

Figure 7.7: Nested Adjusted 10-fold validation (Model selection 3) - validation and test sets are the same for each data spilt

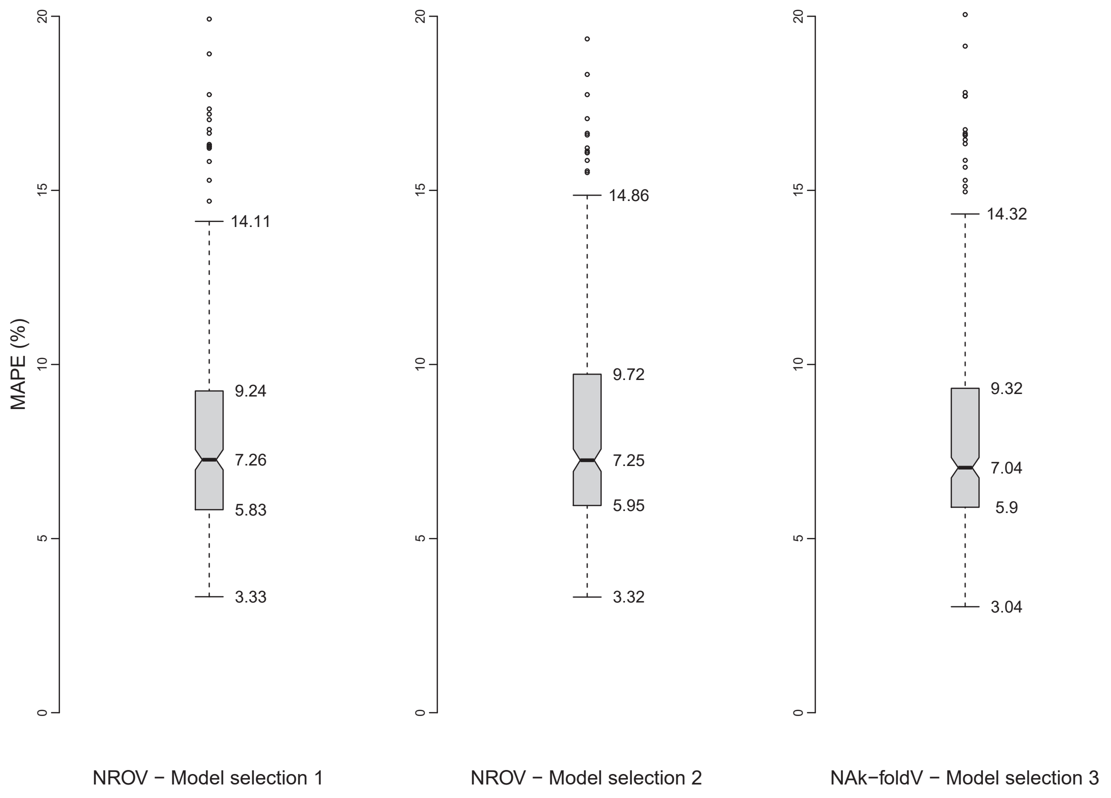

Figure 7.8: Statistics for the performance of 24-steps ahead forecast across 342 MV feeders using 1) Nested Rolling-Origin Validation (NROV) Model selection 1 (left) and 2) Nested Rolling-Origin Validation Model selection 2 (middle) and Nested Adjusted-k-fold Validation (NAk-foldV) Model selection 3 (right) (NO_KNN dataset)

Table 7.8: 24-h ahead forecast MAPE(%) results for 5 feeders - Training/testing datasets preprocessed with imputation only (no outliers cleaning)

	Feed 1	Feed 2	Feed 3	Feed 4	Feed 5
r_m_r	5.34	**6.13**	**10.37**	5.77	**6.11**
r_m_no	4.06	**5.99**	**8.39**	5.67	**6.11**
r_knn_r	5.46	6.30	9.46	**6.19**	6.27
r_knn_no	4.17	6.17	7.25	**6.07**	6.27
r_kf_imp_r	**5.22**	6.17	9.70	5.72	6.31
r_kf_imp_no	**3.94**	6.06	7.56	5.61	6.30
r_kf_smo_r	6.17	6.44	**9.01**	**5.70**	6.14
r_kf_smo_no	4.88	6.31	**6.92**	**5.59**	6.14
missing values train (%)	0.80	1.05	1.28	1.17	0.30
missing values test (%)	0.24	0.32	0.22	0.508	0.07

Table 7.9: 24-h ahead forecast MAPE(%) results for 5 feeders-Training/testing datasets preprocessed with imputation and outlier cleaning

	Feed 1	Feed 2	Feed 3	Feed 4	Feed 5
no_m_r	5.31	6.35	8.38	**4.92**	5.70
no_m_no	4.02	6.10	6.22	**4.80**	5.69
no_knn_r	5.06	**6.12**	**8.24**	4.98	**5.31**
no_knn_no	3.76	**5.98**	**6.08**	4.87	**5.30**
no_kf_imp_r	**4.89**	6.36	9.05	5.57	5.98
no_kf_imp_no	**3.59**	6.13	6.96	5.47	5.98
no_kf_smo_r	4.89	6.37	8.42	5.14	5.96
no_kf_smo_no	3.58	6.25	6.30	5.03	5.96
missing values train (%)	1.68	1.82	2.75	2.73	2.02
missing values test (%)	1.82	1.70	1.0	1.58	0.32

Note : Training and testing data preprocessing strategy is indicated in black; ground truth data are indicated in blue (r = raw data, no = no outliers, m = unconditional mean imputation, kf_imp = Kalman Filter imputation, kf_smo = Kalman smoother imputation, knn = 10-Nearest Neighbour) and level-shifts in the datasets. The plots also inform that *Feed 1* and *Feed 4* present no structural breaks in the training data and behave alike in terms of overall shape; both data contain outliers of varying amplitudes spread across the full dataset (training and testing), however, *Feed 4* has two large outliers at the end of the training data. The training and testing data of *Feed 2* and *Feed 3* contain multiple structural breaks and gross errors of various amplitude. *Feed 5* presents only one significant structural break in the middle of the training data, but there are no significant data quality issues in its testing data. At the bottom of Tables 7.8 and 7.9, the percentage of missing values (**mv**) in the feeders' training and testing data are reported. Note that missing values can only be logged before any imputation is performed.

The forecasts performances are now discussed considering the five feeders one-by-one, the feeders' MAPE reported in Table 7.8 (imputed raw datasets) and Table 7.9 (cleaned and imputed datasets) are compared. The focus in this analysis is to assess how a given data cleaning procedure improves the modeling performances, ergo, the forecasting accuracy improvement is computed as the difference between the best performances found in Tables 7.9 and 7.8.

Feed 5's raw data present a modest percentage of missing observations and consequently, most of the missing values in Table 7.9 are due to outliers being removed. The forecasts results obtained with the raw data imputed with **mean** outperformed the three other techniques. Following the outliers cleaning procedure, forecasts accuracy has improved for all imputation techniques, but the **knn** imputation did a better job on cleaner data and surpassed all the other imputation methods. Because the testing data had very few outliers, the MAPE results obtained with both type of ground truth data (**r** and **no**) are almost identical. With *Feed 5*, we achieve a 0.80 reduction of the MAPE.

Feed 4 performs better than *Feed 5* despite the level of missing values in *Feed 4* data being larger than in *Feed 5* data and despite the presence of many large gross errors in its series. This may indicate that the only presence of the level-shift in *Feed 5* training data has biased the estimation of the model parameters. *Feed 4* model's performances indicate that Kalman smoothing was the best imputation procedure used on the raw data. These good performances are a consequence of the filtering of some of the noisy data in both training and testing data. Note that imputing *Feed 4* raw training data with **knn** produced the worst forecasts possibly because contaminated observations have been used to impute missing observations. Overall, *Feed 4* forecasts accuracy has improved following outliers removal and best forecasts are achieved with **mean** imputation on cleansed data. The accuracy improvement was reported based on the MAPE obtained with the cleaned ground truth data (**no**) since *Feed 4* testing data contained several outliers. *Feed 4* *Feed 4* forecasts were improved by 0.8

Feed 3 performed the worst among all five feeders. These poor results are due to the presence of multiple level-shifts in the training data and one in the testing data. In addition, the feeder's raw data contain the highest level of missing observations exacerbated by the outlier cleansing procedure. Similar to *Feed 4*, *Feed 3*'s model performed best on smoothed raw data while **knn** imputation outperformed **mean** and **Kalman** on cleaned data. With *Feed 3*, we also achieved 0.80 accuracy improvement. Forecast results for *Feed 2* show that outlier cleansing does not always help FDNN models to perform better. The results also emphasise the analysis done for *Feed 5* and *Feed 3*; level-shifts in the training data affect negatively the model fit consequently the performance of the forecasters. *Feed 2* accuracy results indicate that the forecasts produced by the models trained on cleaned data only and imputed with **knn** or **Kalman smoothing** are approximately equal.

The forecaster trained on *Feed 1* data achieved the best forecasts among the five feeders with the lowest MAPE = 3.59% against 6.08% for *Feed 3*. As a reminder, all accuracy results reported in this section relate to 24-h ahead forecasts. *Feed 1*'s models achieved 0.33 accuracy improvement with **Kalman filter** imputation outperforming the other strategies.

Next, the carry-over effect of outliers on short-term forecasts is discussed. This effect occurs because lagged values of feeder load data are used as inputs to the forecast model. Fig. 7.9 and Fig. 7.10 illustrate the carry-over effect. The red areas in the plots highlight the contaminated observations in the testing data and their carry-over effect on predictions while the green areas outline the improved forecasts produced with the testing data cleaned from contaminated data. Level-shifts not only affect the model parameters estimation, but they also alter the forecast accuracy as illustrated in Fig. 7.9. Although in Fig. 7.9 the level does not jump significantly, the 5th day experienced a drop in level which could not be predicted by the forecaster. In closing, several factors that contribute to deteriorate forecast performance have been identified; level-shifts, outliers in historical and future data, the level of missing values and the applied imputation strategy.

7.2.2 24-step-ahead forecasts analysis at scale

In this section, the forecasting results of all 342 MV distribution feeders for which, accuracy performances are compiled in Tables 7.10, 7.11 and 7.12, are discussed. Similarly to the previous section, the forecasts relate to 24-h ahead prediction of feeders loads. In Table 7.10, the forecasting models are identified by the strategies used to preprocess the training and testing datasets. For each model (8 models per feeder), two MAPEs are computed: one uses raw ground truth data, the other uses cleaned ground truth data. The daily MAPEs were averaged across the full testing data so that a unique accuracy value per feeder is reported, giving a total of 342 values per model (each model accuracy being evaluated twice). The MAPEs of each model are reported in terms of their distribution on the histograms displayed in Fig. 7.11.

In Table 7.10, the distributions are summarized in terms of their mean and standard deviation. Because these statistics are highly sensitive to outliers, the median and the Median Absolute Deviation (MAD) as used as robust alternatives to the mean and the standard deviation, respectively.

Table 7.10 is organised as follows: the first two columns report on the distribution of the MAPEs for the models that were trained on raw datasets (missing values are imputed) while the two last columns provide MAPEs' statistics for models trained on data fully preprocessed (outliers removal + missing values imputation). In the table,

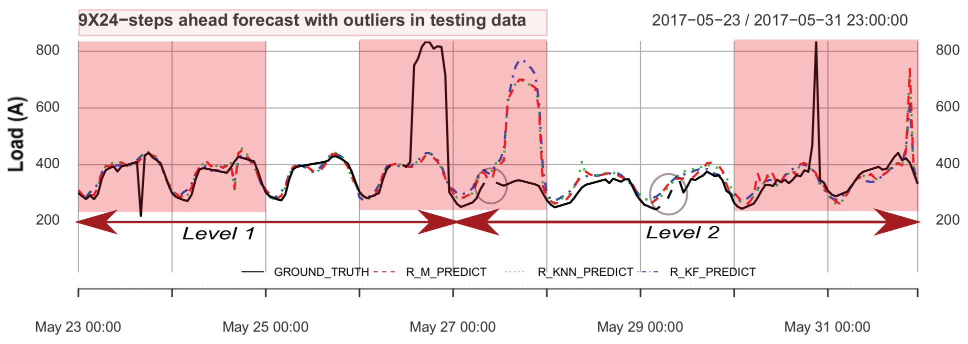
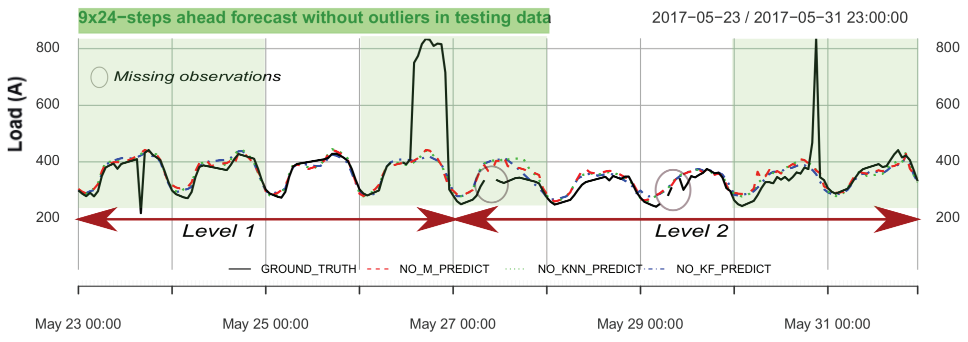

Figure 7.9: 9×24-step-ahead forecasts. Testing data contain outliers (top) while outliers are removed and missing value are imputed

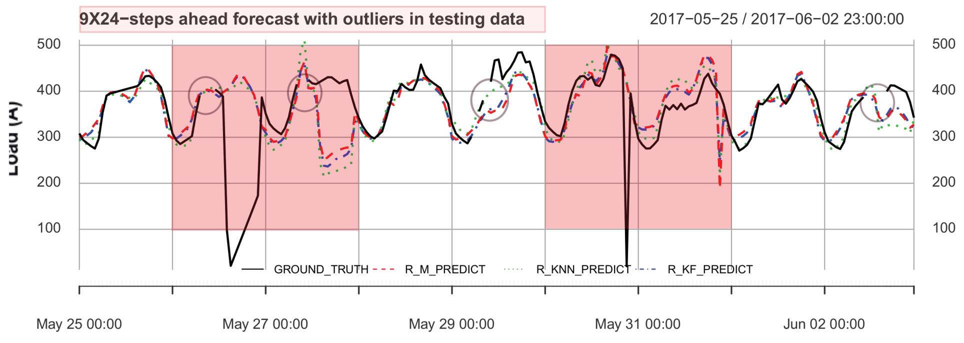
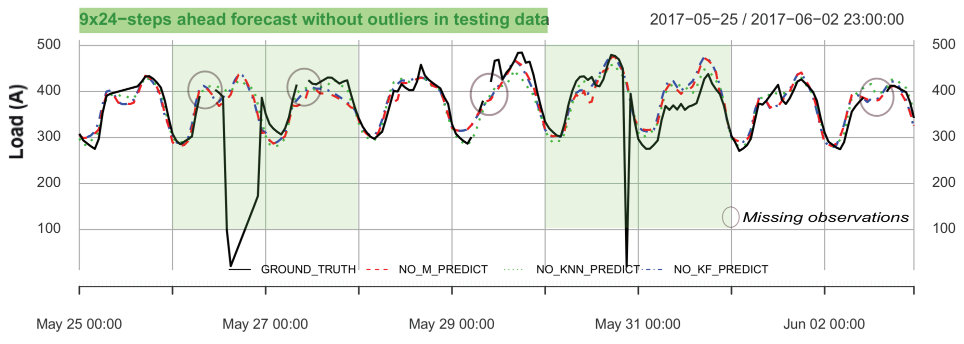

Figure 7.10: 9×24-step-ahead forecasts. Testing data contained outliers (top) while outliers were removed and missing value imputed

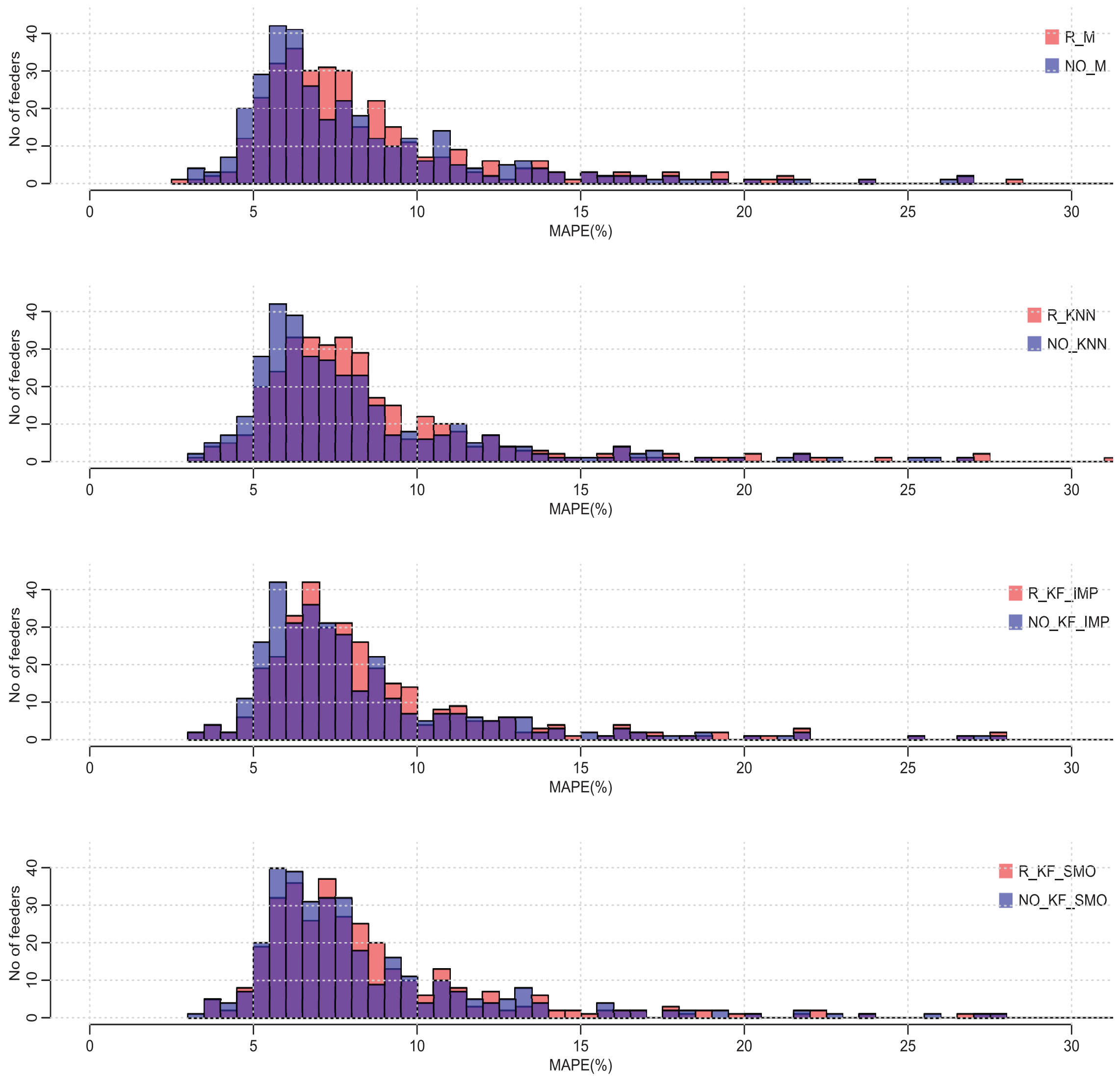

Figure 7.11: Histograms of the MAPEs distributions for raw imputed data and fully cleansed data

each block of four rows refers to a different imputation strategy. Comparing the mean and median of each model, they are quite dissimilar, which indicates significant skewness in the results (see Fig. 7.11). Consequently, the rest of the analysis is based on the median and the MAD. The robust statistics of the MAPEs are highlighted in colour blue in the 2nd and the 4th columns of Table 7.10. These statistics correspond to the accuracy performance computed with cleaned ground truth data; the following discussions always refer to them.

Table 7.10: MAPE(%) statistics of 24-h ahead forecasts across 342 feeders

	r_m_r	**r_m_no**	**no_m_r**	**no_m_no**
mean	13.08	9.35	12.98	9.14
std	12.66	6.93	13.53	7.56
median	8.48	**7.50**	8.06	**6.99**
mad	7.89	**3.72**	8.21	**3.98**
	r_knn_r	**r_knn_no**	**no_knn_r**	**no_knn_no**
mean	13.15	9.46	12.85	9.04
std	12.57	6.84	12.99	6.85
median	8.42	**7.69**	8.04	**7.26**
mad	7.89	**3.69**	7.97	**3.73**
	r_kf_imp_r	**r_kf_imp_no**	**no_kf_imp_r**	**no_kf_imp_no**
mean	13.15	9.46	12.9	9.15
std	12.42	6.54	12.77	6.70
median	8.48	**7.63**	8.02	**7.33**
mad	7.83	**3.64**	7.84	**3.68**
	r_kf_smo_r	**r_kf_smo_no**	**no_kf_smo_r**	**no_kf_smo_no**
mean	13.13	9.44	13	9.24
std	12.49	6.67	12.71	6.61
median	8.35	**7.59**	8.09	**7.28**
mad	7.87	**3.70**	7.92	**3.71**

Note : Training/testing data preprocess strategy is in black - ground truth data are in blue with r = raw data, no = no outliers, m = unconditional mean imputation, kf_imp = Kalman Filter imputation, kf_smo = Kalman smoother imputation, knn = 10-Nearest Neighbour

Several outcomes can be drawn from the statistics provided in Table 7.10: the outlier cleaning procedure did improve the median performance of the forecasters regardless of which imputation strategy was adopted. Nonetheless, the removal of outliers tends to slightly increase the dispersion of the MAPEs (as evidenced by the MAD), which indicates an increase in the models' performance uncertainty. Overall, the imputation techniques used for the study perform alike on the MV feeders datasets. Surprisingly, the simple unconditional mean imputation technique achieved a good score with the lowest median of the MAPEs.

To contrast between the performance of each cleansing strategy, the counts of feeders for which the accuracy does not exceed the upper bounds of 5%, 7.5%, 10% and 15% is provided in Table 7.11. Feeders for which the MAPE exceed 15% are considered to be outliers; these represent approximately 8.80% of all feeders. The experimental results indicate that most of the feeders for which forecasts accuracy were notably improved had a *raw* performance not exceeding 7.5%. The imputation strategy for which outliers removal has been the most beneficial are **knn** and **mean** with 32 feeders and 18 feeders respectively that were led under the 7.5% upper bound performance. Table 7.11 shows that the **mean** strategy presents more models that perform below 5.0 % MAPE accuracy.

Table 7.11: Models performances - count of feeders per data preprocessing

	MAPE (%) upper bounds				
Data preprocessing	≤ 5	≤ 7.5	≤ 10.0	≤ 15.0	> 15
r_m_no	**19**	**171**	264	311	31
no_m_no	**34**	**189**	263	312	29
r_knn_no	**17**	**158**	258	312	30
no_knn_no	**26**	**190**	266	313	29
r_kf_imp_no	14	160	265	312	30
no_kf_imp_no	19	185	266	313	28
r_kf_smo_no	15	165	260	312	30
no_kf_smo_no	17	179	265	312	30

Table 7.12: MAPE's adjustment after full data cleansing

	Δmedian	ΔMAD
r_m −> no_m	0.49	0.26
r_knn −> no_knn	**0.43**	**0.04**
r_kf_imp −> no_kf_imp	0.30	0.04
r_kf_smo −> no_kf_smo	0.31	0.01

Note : In blue is the best trade-off between improving the accuracy across all feeders while maintaining the variability of MAPE to a low level

The results in Table 7.11 also show that automatic outliers detection has succeeded in improving the forecasts whenever it was possible without deteriorating the overall accuracy. The main drawback with removing outliers in the training set is the increased level of missing observations to be estimated. Thus, it exists a bias-variance trade-off associated with the choice between cleaning contaminated observations and creating additional missing values to be estimated. The outcomes of the study are summarized in Table 7.12 where the variations in MAPE distributions between models trained and tested on raw data and models trained and tested on cleaned data are showed. Although **mean** imputation exhibits the best performance improvement with 0.49 MAPE improvement, the statistics show that it is at the cost of an increase in the variability of the models' performance. Hence, **mean** displays the highest MAD increase even if the increased model variability remains low. In conclusion, the **knn** algorithm enhanced with the proposed cleaning strategy achieves the best compromise between improving the forecasters' performance and keeping the uncertainty of the model as low as possible.

7.2.3 Further performance investigation

The 30 feeders for which the MAPE exceeded 15% were further investigated. By inspecting the feeders' data and the produced forecasts, three main reasons for obtaining lousy prediction performances were identified. 1) The quality of the testing data was poor; hence, it may be difficult to assess the out-of-sample performance of the forecaster from test datasets that are unreliable. 2) The model was fitted on poor quality training data; therefore, the model was misspecified. In that case, the model must be retrained on better quality data. 3) The features selection was inappropriate. Generally, any issues related to the model specification should be identified and solved during the model evaluation.

To investigate the outlying forecast performances, a comparison analysis between forecasts collected during the nested-rolling-origin-validation and those obtained during the final testing phase is conducted. As it is reported in Table 7.13, the MAPE performance of the 30 outlying feeders differ significantly from training to testing phases. The analysis from the table shows that only nine feeders performed with a MAPE exceeding 15% during the training; the remaining feeders (21) were tested on bad quality data. Table 7.13 also shows that two feeders performed badly during both the training and testing phase; the quality of the historical data cause these performances. In definitive, only nine feeders require more in-depth investigation.

Table 7.13: Distribution of training and testing MAPE (binned) for 30 feeders with testing MAPE> 30%

	Upper bounds							
MAPE(%)	≤ 5	≤ 7.5	≤ 10	≤ 15	≤ 30	≤ 60	≤ 200	≤ 1000
Training	3	9	4	5	6	1	1	1
Testing	0	0	0	0	23	5	2	0

7.2.4 Conclusion

The key outcomes drawn from these experiments and results are 1) there is no perfect missing values imputation technique that works well for any datasets. It was found that mean imputation outperforms the Kalman smoothing method despite the latter being highly recommended in the literature given its optimality properties. 2) The **knn** is an effective imputation technique that generally performed well on the studied dataset but, to take full advantage of the method, the data must be cleaned from contaminated observations. 3) Structural changes lead to biased parameter estimates and forecasts. However, the forecasters handle level-shifts well by adapting quickly to changes. 4) The past 24-h lagged values are significantly weighted in the short-term forecasts problem, which creates the need for an online data cleaning procedure.

The study has identified several factors that contribute to deteriorating forecast performance: level-shifts, outliers in historical and future data and the imputation of the missing values strategy. The results reported in Table 7.11 show that automatic outlier detection has succeeded in improving the forecasts whenever it was possible without deteriorating the overall accuracy. The main drawback with removing outliers is the increased level of missing observations to be estimated. Thus, there exists a bias-variance trade-off associated with the choice between cleaning contaminated observations and creating additional missing values to be estimated. The outcomes of the analysis are summarised in Table 7.12 where the variations in MAPE distributions between models trained and tested on raw data and, models trained and tested on cleaned data are displayed. Although **mean** imputation exhibits the best performance improvement with 0.49 MAPE improvement, the statistics show that it is at the cost of an increase in the variability of the models' performance. Hence, **mean** displays the highest MAD increase even if the increased model variability remains low. A comparison of MAPE distributions is provided in Figure 7.11. The results show that the **knn** algorithm enhanced with the proposed cleaning strategy achieves the best compromise between improving the forecasters' performance and keeping the uncertainty of the model as low as possible.

Residuals analysis

A careful analysis of residuals helps to assess adequately the risks associated with the use of forecasts on decision making or control strategy. Fig. 7.12 is used to investigate the average performances of the 24-steps-ahead forecasts across the full forecasting horizons. The hourly forecast residuals and hourly percentage error have been clustered using all the forecasts. The boxplots at the top of Fig. 7.12 display the distribution of the hourly MAPE the bottom plot exhibits the median of hourly residuals of the forecasts.

Because feeder loads vary in magnitude, the residuals were normalised using the *feeders' load median* as base-load. The fully cleansed **knn** datasets were chosen for the analysis. The forecast horizons for which forecasts are the most uncertain are highlighted in yellow (top panel); these hours correspond to the morning peak-hours of the day. The corresponding forecast residuals shown on the bottom panel indicates that most of the time, the models underestimate the load. These results corroborate the analysis found in [34] where the authors stipulate that robust models penalise the prediction of peak demand by down-weighting their estimation.

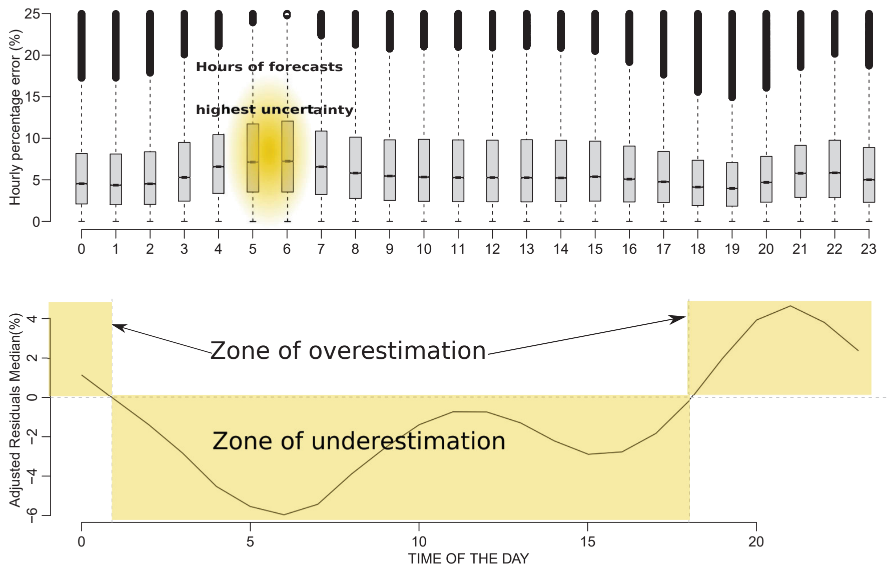

Figure 7.12: Hourly analysis of 24-steps-ahead forecasts across 342 MV/LV feeders for the NO_KNN training dataset. Hourly boxplots of the MAPE (top) and median normalised hourly forecast residuals (bottom)

Chapter 8

Forecasting with gaps imputation

8.1 Description of the imputation strategies

The prospect of producing accurate forecasts by filling gaps of missing values with advanced imputation techniques is investigated. This experiment assesses, in particular, the impact of imputation on one-step-ahead and 24-step-ahead forecasts when different imputation strategies are used to fill gaps in the training data. The study is conducted on four training datasets, namely F1 to F4 are used; each data experience multiple gaps for which the time intervals are reported in Table 8.1; the table indicates the number of missing values in each gap and the corresponding count of consecutive missing days. Note that the gaps arise after a yearly cycle in each feeders' data except for F4 (see bottom plots in Fig 8.1 to Fig 8.4). F3 experience the largest gap with 24.5 consecutive missing days. F2 experience the largest number of gaps; most of the gaps spread along the second half of the data. F4 is the single feeder for which the biggest gap resides in the first half of the training set. The concept of a 'gap' as it is understood in this experiment is described below. Prior to imputation, F1 to F4 training data were cleaned from outliers using the semi-automatic procedure discussed in Sec. 6.6. The missing values are imputed with three main strategies as it is reported in Table 8.2 and Tables 8.3 : *Standard Imputation*, *Imputation by windows* and *Model-based Imputation*.

Table 8.1: F1 to F4 sequences of missing values in historical training data ($l = 12$)

	Start gap	End gap	missing values	count in days
F1	2015-06-22 13:00:00	2015-06-24 00:00:00	36	1.5
	2016-04-20 13:00:00	2016-05-10 00:00:00	468	19.5
F2	2015-06-02 13:00:00	2015-06-03 00:00:00	12	0.5
	2016-04-06 13:00:00	2016-04-19 00:00:00	300	12.5
	2016-05-04 13:00:00	2016-05-14 00:00:00	228	9.5
	2017-02-21 01:00:00	2017-03-02 00:00:00	216	9
	2017-03-17 01:00:00	2017-03-17 12:00:00	12	0.5
F3	2016-01-26 13:00:00	2016-02-20 00:00:00	588	24.5
	2016-08-18 01:00:00	2016-08-18 12:00:00	12	0.5
F4	2015-05-20 01:00:00	2015-06-08 12:00:00	468	19.5
	2016-09-26 13:00:00	2016-09-28 00:00:00	36	1.5
	2017-02-21 13:00:00	2017-02-22 00:00:00	12	0.5

The *Standard Imputation* implements two methods: **full_knn** and **full_mean** where all missing observations (gaps and non-gaps) are imputed with **knn** and **unconditional mean** respectively. For these methods, the testing datasets are imputed alike the training data. The *Imputation by windows* is a procedure proposed for this study; the procedure is only used to impute gaps; the remaining of the missing observations are filled using either **knn**, **mean** or **kf_imp** imputation strategies. Similarly, F1 to F4 testing datasets that did not contain gaps were imputed as the remaining data of the training data. Hence, the *Imputation by windows (iw)* consists in three hybrid strategies: **iw_mean** and **iw_knn** and **iw_kf_imp**. The *Model-based Imputation* holds two hybrid strategies: **tbats_knn** and **prophet_knn**. The **tbats_knn** strategy uses TBATS model to impute the gaps; the remaining of missing values are imputed using **knn**. TBATS is a univariate model that filters through the training data sequentially and produces h forecasts at the end of the training set. A computationally efficient implementation of TBATS model is available in the R's `forecast` library. TBATS was fit to the multi-seasonal time series object `msts` for all feeders' time series. Because the gaps occur after a full yearly cycle, the `msts`'s seasonal periods were set to daily, weekly and yearly seasonality for feeders F1 to F3. For feeder F4, the seasonal periods were set two daily and weekly patterns only because of the position of the gap; however, the TBATS model failed to produce reasonable estimates. Likewise, the **prophet_knn** procedure imputes gaps using `Prophet`, the remaining of missing values are estimated with **knn**. As a multivariate time series forecasting procedure, `Prophet` was configured to use the input variables described in Table 7.1. The **knn** imputation is carried out as described in Sec. 4.2.2. When gaps have been filled, the feeders' data are concatenated alongside the other 342 cleansed data, and the **knn** imputation is computed. Once the F1 to F4's training sets were fully imputed, one-step-ahead forecasts and 24-step-ahead forecasts were generated using FDNN. The models' selection and evaluation were performed as rigorously as described before using the NROV procedure.

Description of *Imputation by windows*

A gap consists of a minimum of $l > 1$ consecutive missing observations. To determine the position of each gap in the data, a moving average procedure is applied to a *shadow vector*. Let $\{y_t\}_{t=1}^{n}$ be the feeder' time series, where $y_t \in \mathbb{R}$, the *shadow vector* is defined as $\{\bar{y}_t\}_{t=1}^{n}$ where, $\bar{y}_t \in \{0, 1\}$; $\bar{y}_t = 1$ indicates a missing observation a time t and $\bar{y}_t = 0$, indicates the presence of a measurement. The moving average uses a sliding windows procedure with a windows of size l and step-size=1 across $\boldsymbol{y}_t$ to identify gaps. The top plots in Fig. 8.1 to Fig. 8.4 display the moving average for $\{y_t^{(j)}\}_{t=1}^{n}$ standardized with z_{score} and $\{\bar{y}_t^{(j)}\}_{t=1}^{n}$ where $j \in \{1, \ldots, 4\}$ for F1 to F4 and $l = 12$. The windows size was chosen based on the assumption that for gaps of size $l < 12$, any of the imputation methods used previously would be sufficient. Imputation by windows fills feeder's data gaps with a neighbouring data sample borrowed from the feeder's own series. Let $G^{(j)}$ be the set of gaps in feeder j and $g_i^{(j)} = \{y_t^{(j)}\}_{t=s_i}^{s_i+p} = \emptyset$ the ith gap of length $p \geq l$, $p \in \mathbb{Z}$ and $i \in [1, \ldots, |G^{(j)}|]$. The left and the right windows used for imputation are defined respectively as $lw_i^{(j)} = \{y_t^{(j)}\}_{t=s_i-p}^{s_i}$ and $rw_i^{(j)} = \{y_t^{(j)}\}_{t=s_i+p}^{s_i+2p}$ where $s_i - p \geq 0$ and $s_i + 2p \leq n$, $|lw_i^{(j)}| > \frac{3p}{4}$ and $|rw_i^{(j)}| > \frac{3p}{4}$. Each gap is filled as follow

$$g_i^{(j)} = \underset{x}{argmin}\ \mathrm{std}(x) \tag{8.1}$$

$$x \in \{lw_i^{(j)}, rw_i^{(j)}\} \tag{8.2}$$

where std(x) denote the standard deviation of x.

8.2 Case study

All the results reported and discussed in this section concern F1, F2, F3 and F4 feeders. In this study, imputation methods are used to training a model. Hence, seven models are compared to one another. On the plots which are described in more details below, the yellow colour highlights unpredictable variations of the load, places in the ground truth data where missing values were imputed are highlighted in green and, blue rectangles that surround the name of a dataset indicate the model performs the best for a give imputation strategy. These exact performance values are reported in Tables 8.2 and 8.3.

8.2.1 Results description

Gaps' estimate visualisation

The bottom plots in Fig 8.1, 8.2, 8.3 and 8.4 show the *Imputation by windows* of data gaps. `Prophet` and TBATS gaps' imputation are shown in Fig 8.5, 8.6, 8.7 and 8.8. Each of the figure shows the TBATS imputation at the top plot and `Prophet` imputation at the bottom. In every figures, non-imputed and imputed data are superimposed with the non-imputed data appearing at first ; this allows gap's estimates to be shown in dark-green.

Summary of forecasting performance

Tables 8.2 and 8.3 report the forecast accuracies for all feeders and for each one of the imputation strategies introduced in this study. Table 8.2 and 8.3 show the one-step-ahead and 24-step-ahead forecast results for forecasts generated with the FDNN models; Best forecasting performances are highlighted in blue. The results highlighted in red are further discussed in Sec. 8.2.2.

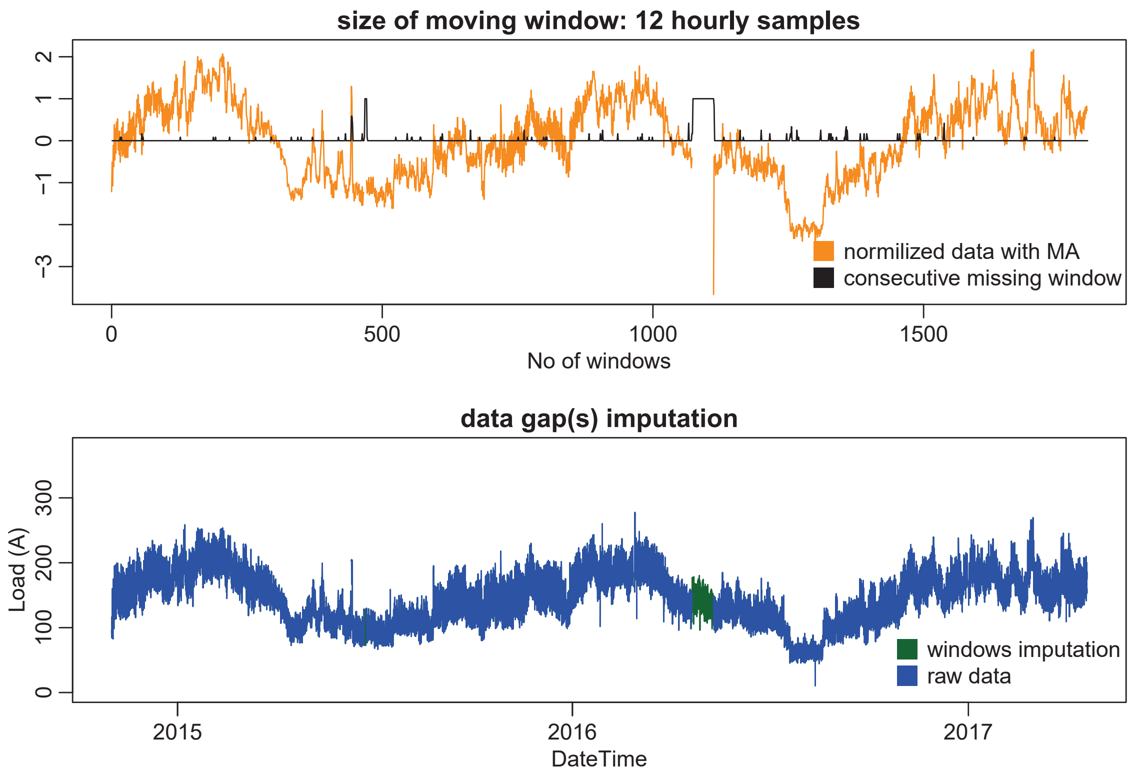

Figure 8.1: Gaps imputation with 'Imputation by windows' strategy - F1

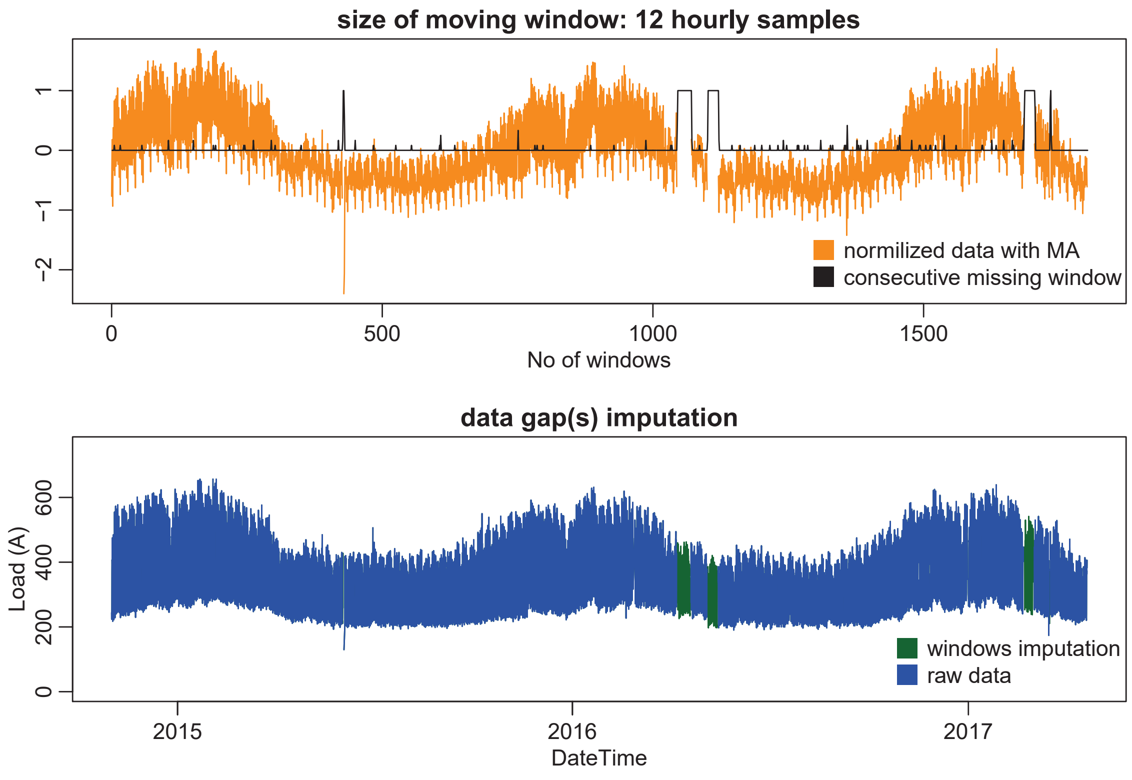

Figure 8.2: Gaps imputation with 'Imputation by windows' strategy - F2

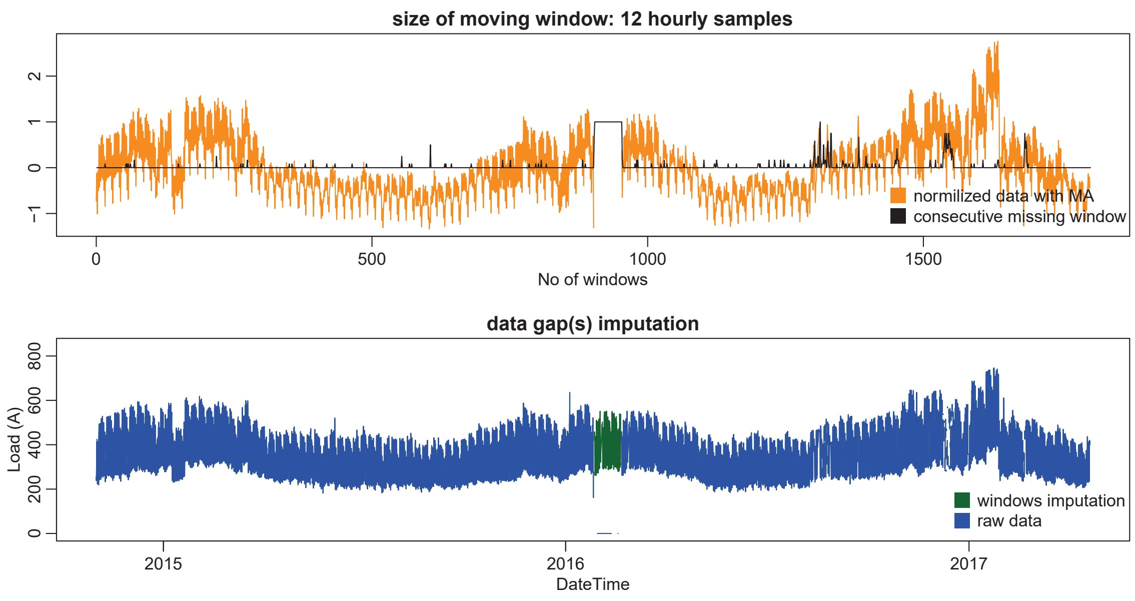

Figure 8.3: Gaps imputation with ‘Imputation by windows’ strategy - F3

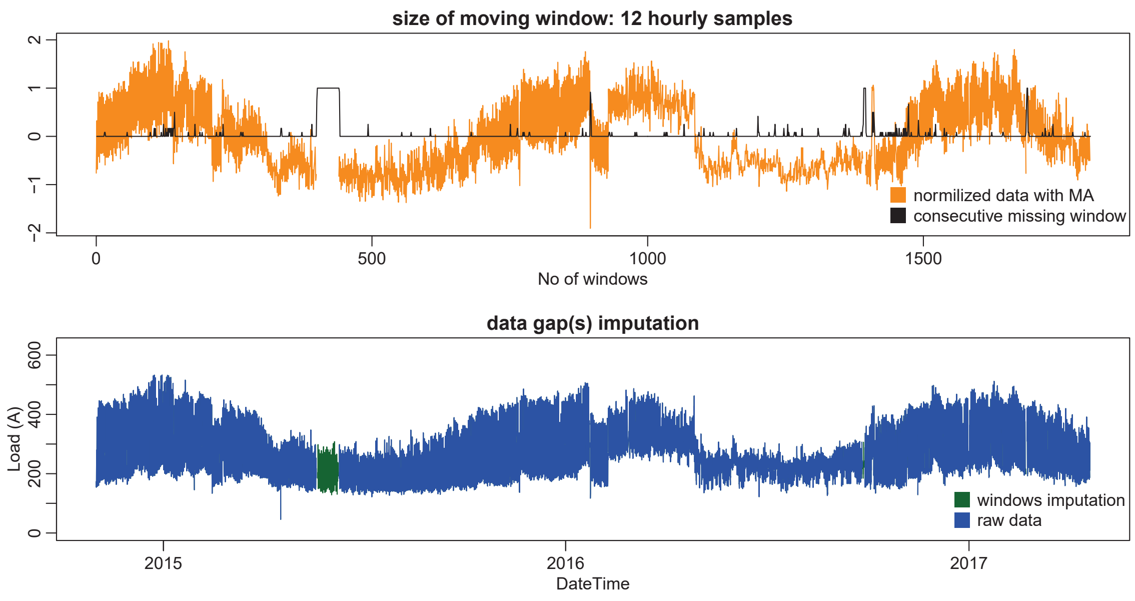

Figure 8.4: Gaps imputation with ‘Imputation by windows’ strategy - F4

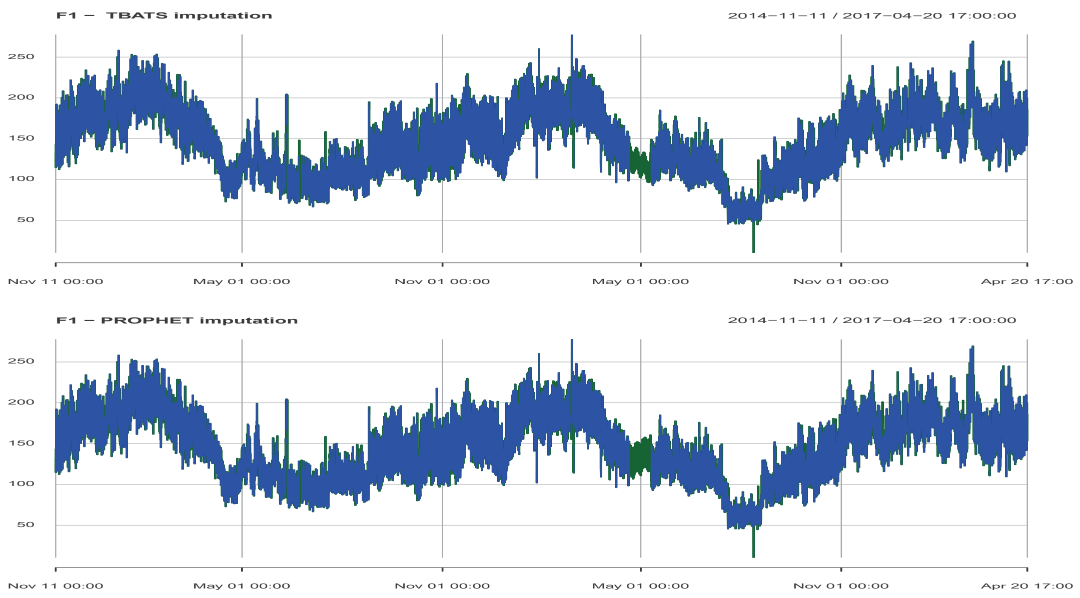

Figure 8.5: Gaps imputation with TBATS (top plot) and `Prophet`(bottom plot) - F1 - (the imputed gaps are shown in dark-green)

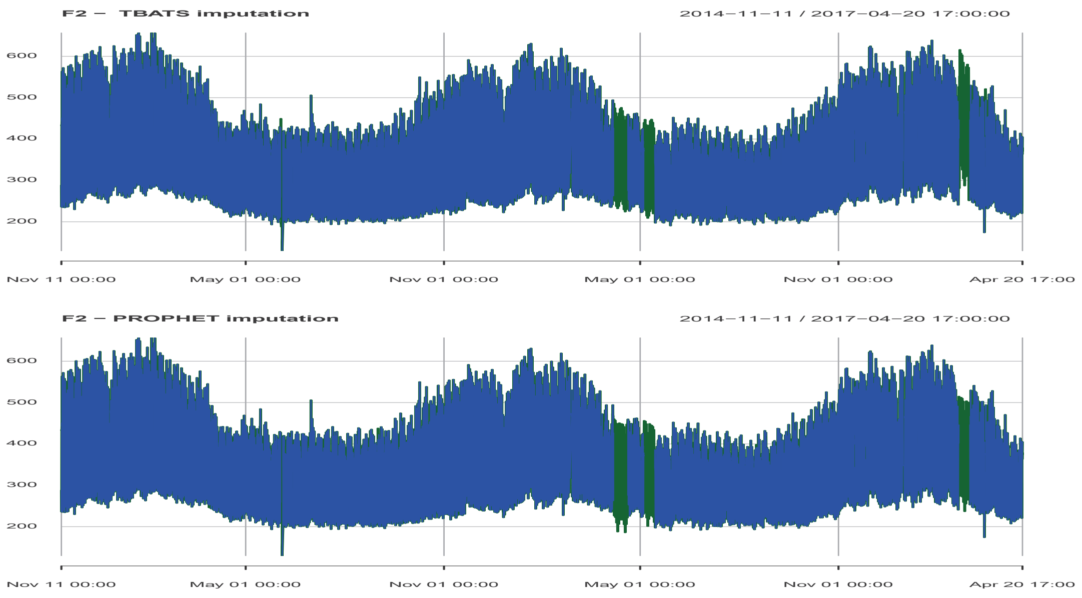

Figure 8.6: Gaps imputation with TBATS (top plot) and `Prophet`(bottom plot) - F2

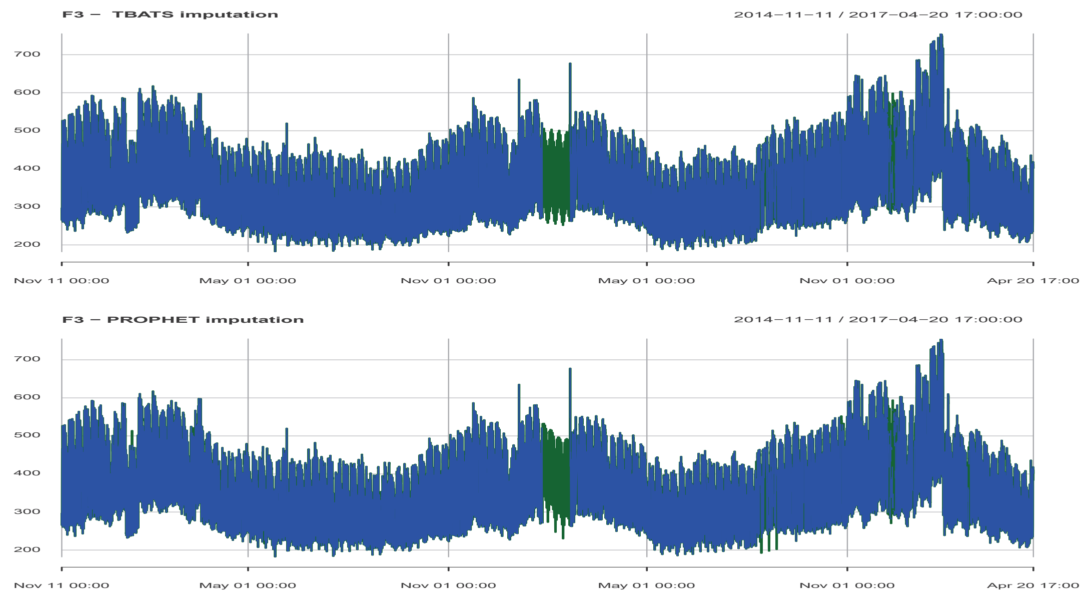

Figure 8.7: Gaps imputation with TBATS (top plot) and `Prophet`(bottom plot) - F3

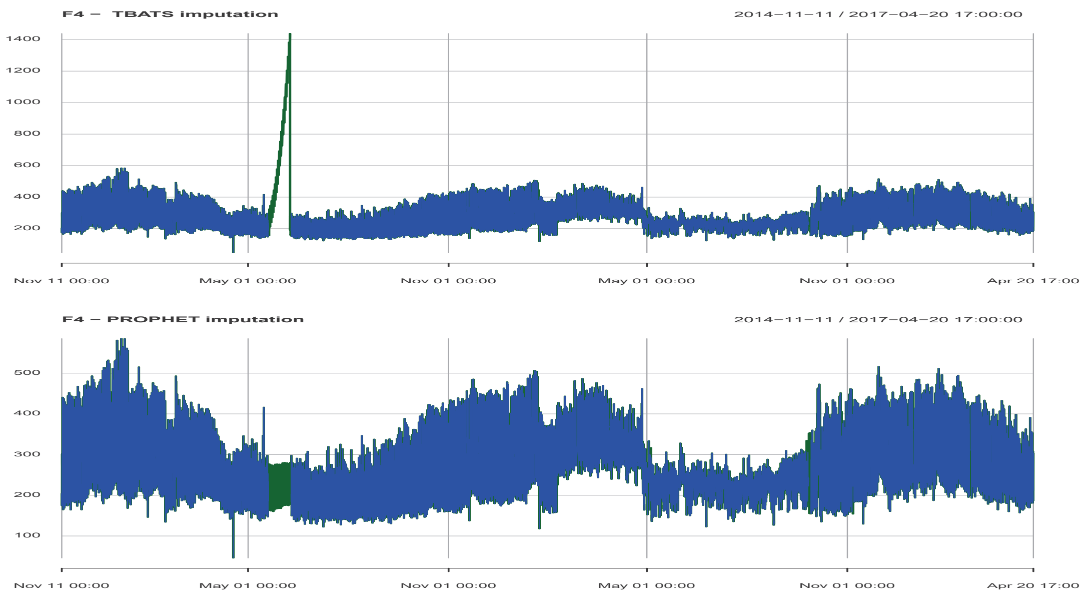

Figure 8.8: Gaps imputation with TBATS (top plot) and `Prophet`(bottom plot) - F4

Forecasts analysis by visualisation

For each one of the feeders, one-step-ahead and 24-step-ahead forecasts models are plotted against the ground truth; only a small part of the testing data are reported (testing data are described in Sec. 7.1.1). Fig 8.9 and Fig 8.10 display F1 forecasts, Fig 8.11 and Fig 8.12 report F2 forecasts, Fig 8.13 and Fig 8.14 exhibit F3 forecasts and Fig 8.15 and Fig 8.16 display F4 forecasts. In each figure, the plots are organised by imputation strategies. The two methods used in *Standard imputation* are plotted at the top of the figures, the middle plots report the three methods used in the *Imputation by windows*, and the bottom plots relate the two imputation methods found in the *Model-based Imputation* strategy.

8.2.2 Results and discussion

One-step-ahead forecasts

The forecasting performances are disparate from one feeder to another, therefore, the results reported in Table 8.2 must be juxtaposed to the forecasting plots in Figs 8.9, 8.11, 8.13 and 8.15 to perform an effective analysis. The plots provide information on the quality of the testing data. These data reflect on the lagged values that were used by the models to produce the forecasts. On the plots, ground truth data represent the raw version of the testing data; there are drawn in colour black consistently.

The table displays consistent best one-step-ahead forecasts performances for models trained on *Imputation by windows* imputed data. Hence, this suggests that the *Imputation by windows* strategy is a good option for repairing training datasets. Besides, looking at the one-step-ahead forecast plots, the models trained on these data demonstrate good abilities to forecast base-load with precision. This heuristic strategy outperforms the more sophisticated strategies, TBATS and `Prophet` on these datasets. Also, it seems that *Imputation by windows* provides some 'damping' capability to the models. This can be seen on F1 and F3 plots. By taking into account that the models are uncertainty, a fair evaluation of results in Table 8.2 establishes that all the imputation strategies have provided good forecasts for F2 feeder. However, F3 accuracy results show without a doubt that *Imputation by windows* outperformed all other imputation strategies with `Prophet` achieving a similar level of performance.

In general, the forecasts performances (tables and plots) show that all the models perform well on raw and on persistent testing data. When an 'unpredictable' variation of the load shape occurs, or, if the raw data have missing observations, the forecast accuracy decreases. Any deviation from the expected load profile leads the models to perform poorly. Unexpected variations in the testing data affect the 24-step-ahead forecast accuracy more than the point forecasts. The results also show that, whenever the load lagged values are generated from imputation, it affects the forecasts shape noticeably; especially when estimate are generated form **mean**imputation. Hence, not surprisingly, feeder F2 reaches the best forecast performances, its data are pure through the entire testing sample. There is only one place where the ground truth required to be imputed. The feeders that perform the worst one-step-ahead forecasts are F3 and F4 where feeder F3 performing really badly in some occasion. For instance, it is not clear why the F3 performance is 10.7% for the *Imputation by windows* with **knn** while the two *Imputation by windows* conjoint methods perform relative well. Most likely, these bad results can be granted to the **knn** imputation. Among all the feeders, the plots show that F3 is the feeder that contains the most missing values in its ground-truth data. This emphasizes the critical importance of choosing suitable *online* imputation methods.

24-step-ahead forecasts

The day-ahead forecasts general performances follow those of point forecasts with the best accuracies achieved with the *Imputation by windows* imputed models. The accuracy decreases because of the greater time horizons to be forecasted and the lack of hourly update on the system latest state. The experimental results in the table indicate that a model which produced the best one-step-ahead forecasts might not necessarily perform the best in producing multistep-ahead forecasts. However, here again, this comment has to be put into perspective with the uncertain nature of any models. Therefore, generally, the models' performances are consistent whenever the models are used to produce point forecasts or 24-step-ahead forecasts.

Table 8.2: FDNN - One-step-ahead forecast accuracy with gaps imputation RPE (%)

	F1	F2	F3	F4
	Standard Imputation			
full_knn	6.59	2.71	11.09	6.39
full_mean	4.13	2.87	10.43	6.34
	Imputation by window (iw)			
iw_mean	4.26	2.81	**4.61**	6.41
iw_knn	3.89	2.83	**10.70**	5.81
iw_kf_imp	**3.70**	**2.35**	6.18	**5.55**
	Model-based Imputation			
tbats_knn	6.87	2.63	8.36	5.92
prophet_knn	6.60	2.55	**4.74**	6.09

Table 8.3: FDNN - 24-step-ahead forecast accuracy with gaps imputation RPE (%)

	F1	F2	F3	F4
	Standard Imputation			
ful_knn	9.54	4.99	11.93	9.18
full_mean	**7.91**	4.77	11.96	9.01
	Imputation by window (iw)			
iw_mean	8.66	4.82	**7.15**	8.96
iw_knn	8.05	**4.64**	11.44	8.89
iw_kf_imp	**7.97**	5.31	9.08	**8.6**
	Model-based Imputation			
tbats_knn	9.75	4.82	10.74	9.25
prophet_knn	9.54	**4.64**	**7.35**	9.06

Best performances are highlighted in colour blue for each feeder

8.3 Conclusion

In this experiment, gaps of missing observations in training datasets are imputed with one of the following strategies: **knn**, **mean**, `Prophet`, TBATS and, *Imputation by windows* which was designed for this study. The proposed *Imputation by windows* strategy has demonstrated consistently better performances than those achieved by standard and

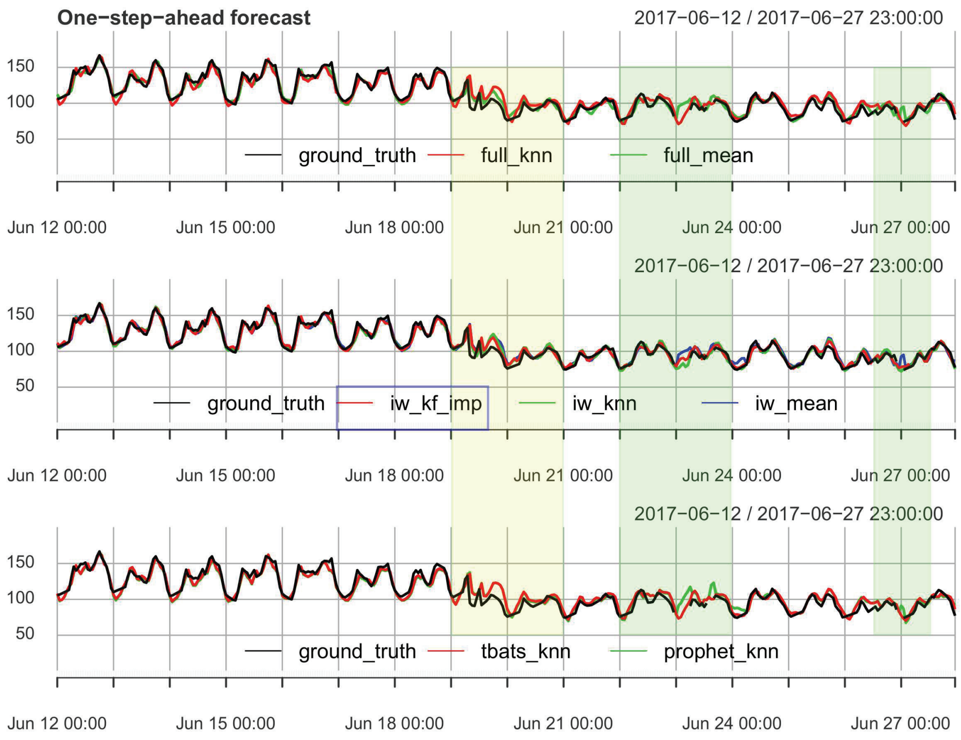

Figure 8.9: F1's OSA forecasts with FDNN

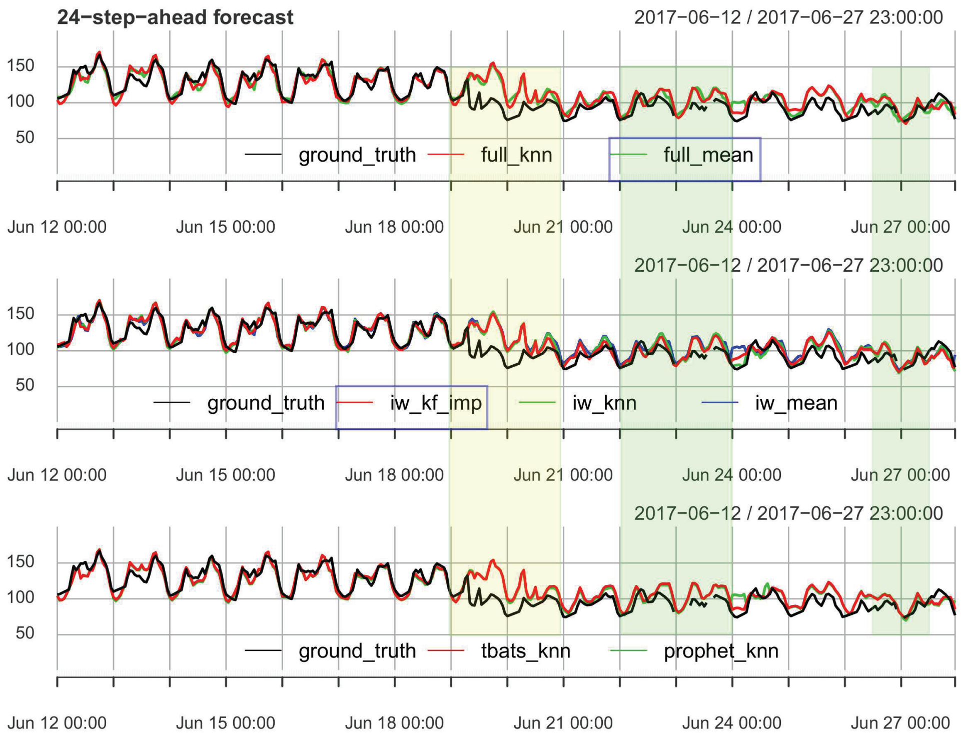

Figure 8.10: F1's MSA forecasts with FDNN

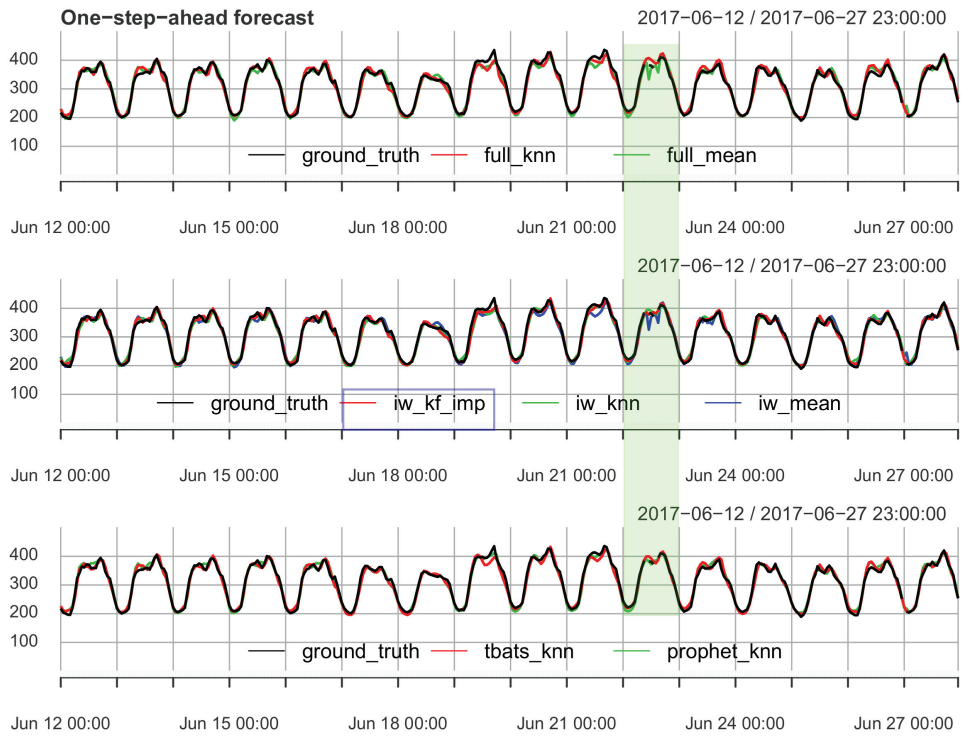

Figure 8.11: F2's OSA forecasts with FDNN

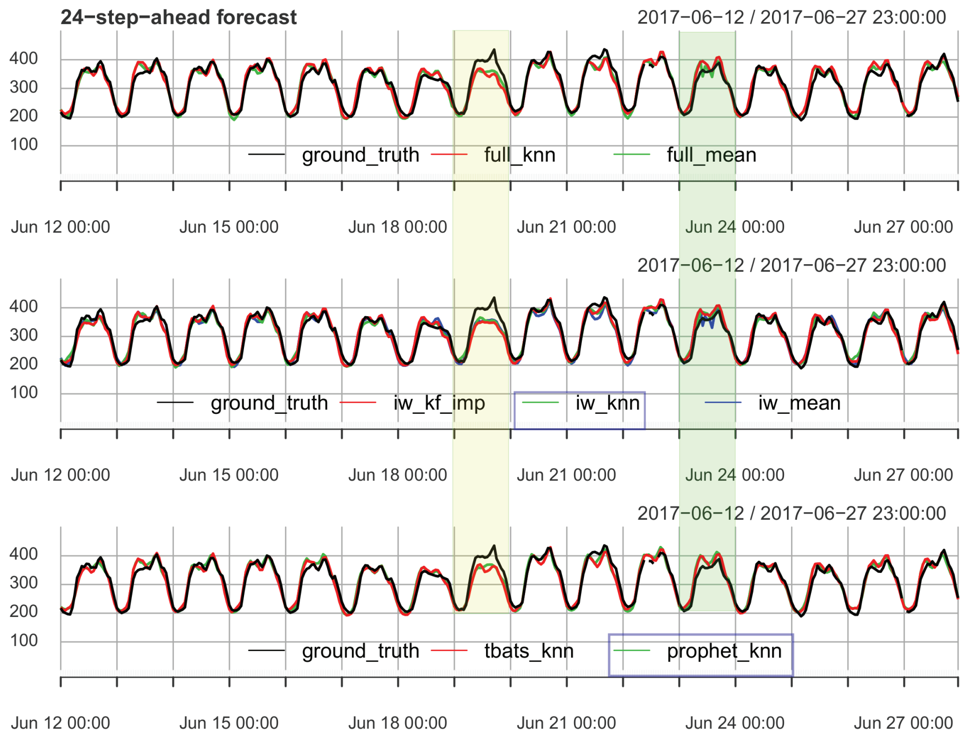

Figure 8.12: F2's MSA forecasts with FDNN

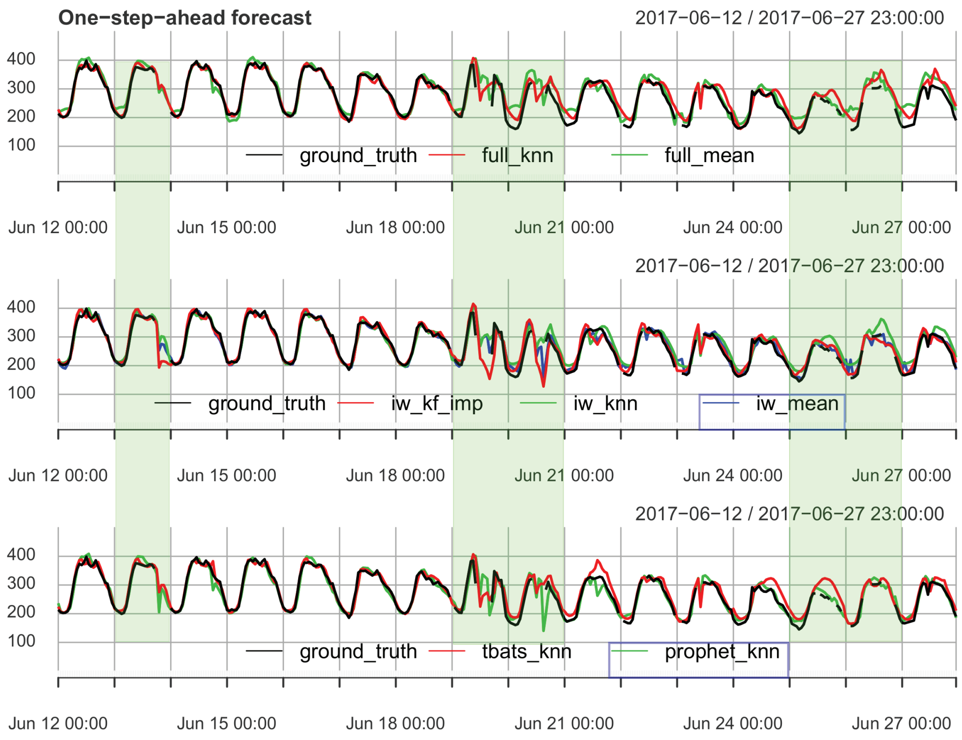

Figure 8.13: F3's OSA forecasts with FDNN

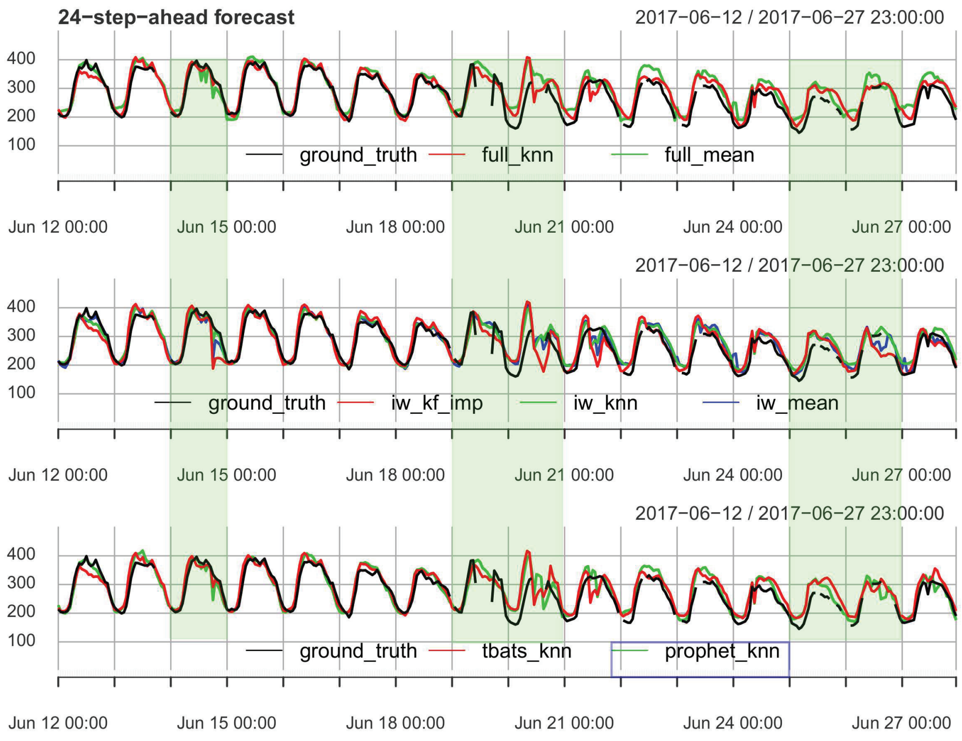

Figure 8.14: F3's MSA forecasts with FDNN

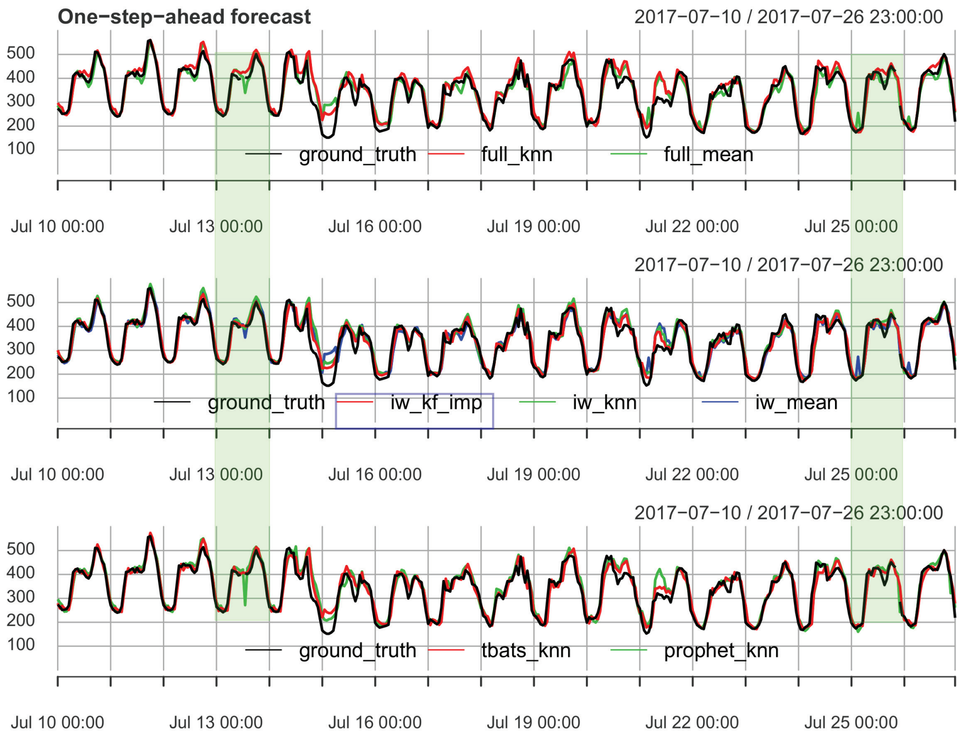

Figure 8.15: F4's OSA forecasts with FDNN

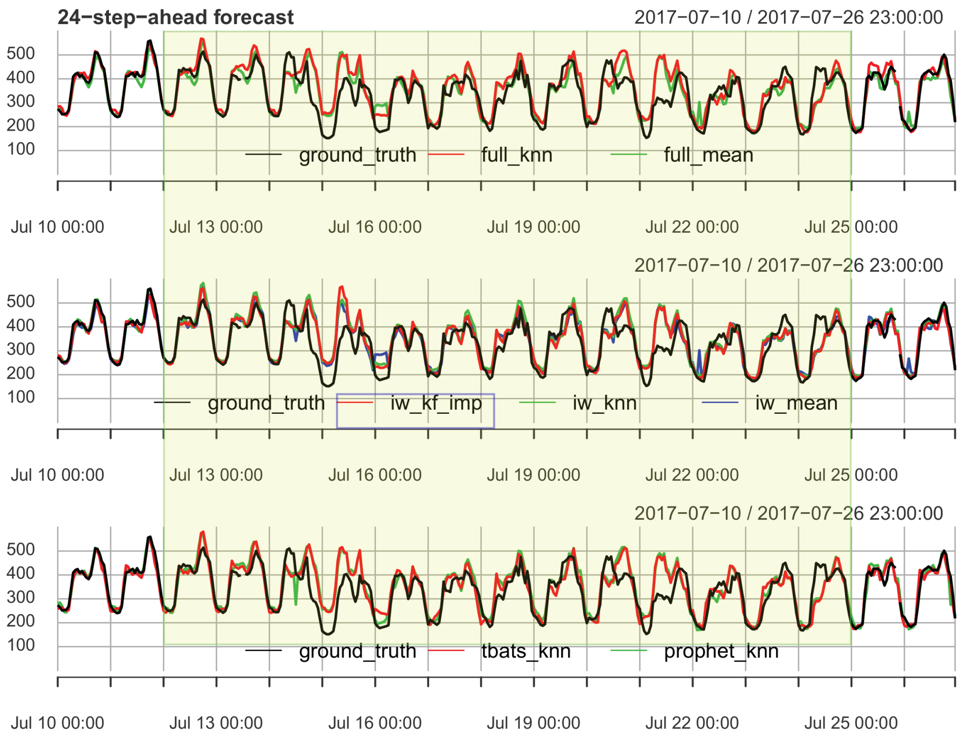

Figure 8.16: F4's MSA forecasts with FDNN

sophisticated imputation techniques. The method fills feeder's data gaps with a neighbouring data sample borrowed from the feeder's series. The promising performances of this technique might suggest that the algorithm should be extended to the entire dataset. However, the study also demonstrates the negative impact that imputation techniques can have on short-term load forecasts. It highlights the complexity in finding a method which can provide the best results for any data if it is even possible. This emphasizes the critical importance of choosing suitable *online* imputation methods. The best strategy is to avoid missing values imputation by ensuring quality data at their creation whenever possible.

Chapter 9

Conclusion

9.1 Summary of thesis achievements

Today, data quality issues draw much more attention than they were in the past because data have properties, unlike other assets. Data quality issues of electricity networks datasets can be real bottlenecks to the smooth implementation of innovation projects. Moving forward in smart-grid paradigm, data quality is key to delivering benefits from a smart grid. The development of applications for advanced analytical studies relies on the integrity of the data input; therefore, it becomes increasingly important to ensure that data-quality issues are monitored and mitigated. Identifying the root causes of error is difficult; the cause of errors can be both simple and complex.

This thesis attempted identifying the roots causes of missing observations in large scale MV feeder load datasets. The analysis made use of a shadow matrix of the raw data to facilitate the visualisation of the full load dataset. The analysis of the matrix was approached from a vertical and horizontal standpoint. The vertical analysis has shown that feeders connected at the vicinity of the urban area of the South East of England experience a higher percentage of missing observations. The horizontal analysis revealed that simultaneous loss of network measurements could occur from time to time. This thesis stated the hypothesis that some congestions in communication networks could be the source of coincidental loss of observations.

Also, this thesis identified potential future opportunities to conduct studies on the effects of DST on energy savings based on real observational data. The European Commission has planned to implement a new directive in 2021 that abolishes DST changes; it has also planned to assess the impacts of the new Directive by 31 December 2024. This thesis found that the difference-in-difference approach could be used with if new observational data are made available.

Data quality and good practices in model design and evaluation are key features to achieve good forecast performances. A comprehensive investigation into the problem of short-term load forecasting for large numbers of MV distribution network feeders was proposed in this thesis. Multiple data cleansing strategies suitable for the MV feeders time series were investigated. The thesis implemented an automatic approach to detect and remove outlying observations placed at an unknown location in the modelling datasets. The hybrid method combines two algorithms: a robust design of the binary segmentation algorithm which detects level-shifts in the data and, the Tukey's standard rule. In addition, a robust semi-automatic outlier detection algorithm is also proposed and described. Both procedures were implemented on real-world datasets.

The thesis considered three major imputation techniques for the estimation of missing observations : Unconditional Mean, Hot Deck (k-NN) and Kalman smoothing. Outlier detection and missing values imputation techniques were combined to preprocess 342 MV feeders. MV feeders data were modelled with feed-forward deep neural networks. Two rigorous model evaluation techniques tailored for time series data, were proposed to evaluate the performance of the models.

A recursive forecasting strategy was preferred to a direct forecasting strategy for to forecast day-ahead. It has the advantage of reducing the problem of 24-step-ahead forecasting to the training of a single model. The thesis found that a bias and variance trade-off existed in the data quality enhancement problem. Ideally, the problem should address low bias and variance simultaneously, but in practice, a decrease of the bias generates an increase of the variance and vice versa.

Finally, this thesis proposed a robust framework for developing and reporting large scale short-term forecasting projects at MV distribution network level. It provided guidelines on how to address the challenges related to producing multistep-ahead forecasts with real-world datasets.

9.2 Future work

- In this thesis, several unsuccessful attempts were made to automate the selection of the preprocessed dataset. The impact of various data cleansing procedures on time-series forecastability was investigated but not addressed. Ideally, an automated data cleansing procedure would select the most suitable data imputation strategy for a given time series prior to the modelling stage. With this objective in mind, a fair amount of time was spent in identifying a suitable index to quantify the complexity of time series and therefore its forecastability. Among the time series irregularity evaluation indexes, approximate entropy ($ApEn$), sample entropy and permutation entropy were investigated but did not provided conclusive results [102, 106]. Further research studies should be carried in that direction.

- This work has demonstrated that the production of good forecasts highly relies on healthy load measurements. Therefore, suitable online cleansing strategies should be developed.

- In this thesis, forecasts are computed recursively. The recursive formulation is computationally more efficient than the direct method, which requires to train one model per forecasting horizon h. The advantage of using a recursive strategy, therefore, grows when a large number of time series and multiple forecast horizons are involved. If the model's parameters $\boldsymbol{\theta}$ are chosen adequately during the model evaluation, the one-step-ahead prediction is unbiased. However, in [112], Taieb *et al.* show that the same unbiasedness property does not hold for forecast horizons $h \geq 2$, mainly when the model is nonlinear. The authors show that bias increases with the curvature of the nonlinear function. In future works, we should consider the impact on the bias of this recursive strategy. Additionally, if computational resources are available, one should consider using a direct strategy or a hybrid recursive-direct strategy to improve forecast accuracy possibly.

- The Nested Rolling-Origin-Validation (NROV) returns multiple optimum models. An improved model selection procedure would involve ensemble learning as a model averaging technique. Of course, it would require substantial computational resources.

- A full structural model consisting of a trend, two seasonal components, cycle component and white noise should be implemented to model the MV feeders' time series for Kalman filter imputation. `Stan code` should be used to estimate $\mathbf{Z}_t, \mathbf{T}_t, \mathbf{Q}_t$, h_t, the state-space parameters as given as bellow

$$y_t = \mathbf{Z}_t\boldsymbol{\alpha}_t + \varepsilon_t, \quad \varepsilon_t \sim N(0, h_t) \tag{9.1}$$

$$\boldsymbol{\alpha}_t = \mathbf{T}_t\boldsymbol{\alpha}_{t-1} + \boldsymbol{\eta}_t \quad \boldsymbol{\eta}_t \sim N(0, \mathbf{Q}_t) \tag{9.2}$$

The observation equation is given as

$$y_t = \mu_t + \gamma_t^{(1)} + \gamma_t^{(2)} + \psi_t + \varepsilon_t \tag{9.3}$$

where y_t is to the observation at time t, μ_t is to the trend component, $\gamma_t^{(1)}$ and $\gamma_t^{(2)}$ are to the daily and weekly seasonal components respectively, ψ_t is to the yearly seasonal component, and ε_t is the observation irregular term. Each component state-space expressed are discussed below

The local linear trend model

The trend component represents the long-term evolution in the generative process modeled by a slope. It is a dynamic extension of a regression model with a a generalization of the time-trend that can dynamically vary across time, It can be written:

$$y_t = \mu_t + \varepsilon_t \tag{9.4}$$

$$\mu_t = \mu_{t-1} + \beta_{t-1} + \eta_{t-1} \tag{9.5}$$

$$\beta_t = \beta_{t-1} + \zeta_{t-1} \tag{9.6}$$

where $\eta_t \sim N(0,\, \sigma_\eta^2)$ and $\zeta_t \sim N(0, \sigma_\zeta^2)$ are uncorrelated. The trend model can be written in the state space form

$$\begin{aligned} y_t &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \mu_t \\ \beta_t \end{bmatrix} + \varepsilon_t \\ \begin{bmatrix} \mu_t \\ \beta_t \end{bmatrix} &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_{t-1} \\ \beta_{t-1} \end{bmatrix} + \begin{bmatrix} \eta_{t-1} \\ \zeta_{t-1} \end{bmatrix} \end{aligned} \tag{9.7}$$

The seasonal components

It is common practice to model seasonal patterns by a set of trigonometric cycles. To fully model all seasonal variations of periodicity, the seasonal pattern is formed of a sum of harmonics λ_j, multiple of seasonal fundamental frequency s. The seasonal components can also be made stochastic by including an error terms allowing for the seasonal effects to vary over time. The parameters of γ_t is given as

$$\begin{aligned} \gamma_t &= \sum_{j=1}^{[s/2]} \gamma_{j,t} \\ \gamma_{j,t} &= \gamma_{j,t-1} \cos \lambda_j + \gamma_{j,t-1}^* \sin \lambda_j + \omega_{j,t-1} \\ \gamma_{j,t}^* &= -\gamma_{j,t-1} \sin \lambda_j + \gamma_{j,t-1}^* \cos \lambda_j + \omega_{j,t-1}^*, \\ \omega_{j,t-1} &\sim N(0, \sigma_{\omega^2}) \\ \omega_{j,t-1}^* &\sim N(0, \sigma_{\omega^2}) \end{aligned} \tag{9.8}$$

with $\lambda_j = 2\pi j/s$ for j = 1, . . . , [s/2] and t = 1, . . . , n. From a practical point of view, it is preferable to let the parameters variances be the same although different variances would allow each harmonic to evolve at difference pace. The seasonal model can be written in the state space form as follow

$$\begin{bmatrix} \gamma_{j,t} \\ \gamma^*_{j,t} \end{bmatrix} = \begin{bmatrix} \cos\lambda_j & \sin\lambda_j \\ -\sin\lambda_j & \cos\lambda_j \end{bmatrix} \begin{bmatrix} \gamma_{j,t-1} \\ \gamma^*_{j,t-1} \end{bmatrix} + \begin{bmatrix} \omega_{j,t-1} \\ \omega^*_{j,t-1} \end{bmatrix} \tag{9.9}$$

The number of harmonics for daily and weekly pattern should be investigated. Because seasonal patterns change relatively smoothly over the year, it might be possible to use only the fundamental or drop few higher-order harmonics.

The cycle component

The cyclical component is intended to capture cyclical effects at time frames much longer than captured by the seasonal components. It frequency is substantially greater than the daily and weekly periods s_1 and s_2. In its simplest form, the cycle component is a pure sine wave of frequency λ_c and period $2\pi/\lambda_c$. Likewise to the other components, the cycle can be made stochastic and is described as follow:

$$\psi_t = \psi_{t-1}\cos\lambda_c + \psi^*_{t-1}\sin\lambda_c + \tilde{\omega}_{t-1} \tag{9.10}$$

$$\psi^*_t = -\psi_{t-1}\sin\lambda_c + \psi^*_{t-1}\cos\lambda_c + \tilde{\omega}^*_{t-1} \tag{9.11}$$

where $t = [1, \dots, n]$ ω_t, ω^*_t are assumed to be uncorrelated white noises with same variance. This yields to the state-space form

$$\begin{bmatrix} \psi_t \\ \psi^*_t \end{bmatrix} = \begin{bmatrix} \cos\lambda_c & \sin\lambda_c \\ -\sin\lambda_c & \cos\lambda_c \end{bmatrix} \begin{bmatrix} \psi_{t-1} \\ \psi^*_{t-1} \end{bmatrix} + \begin{bmatrix} \tilde{\omega}_{t-1} \\ \tilde{\omega}^*_{t-1} \end{bmatrix} \tag{9.12}$$

Finally, the hidden state of the feeder's load process at time step t is given by

$$\boldsymbol{\alpha}_t = \begin{bmatrix} \mu_t & \beta_t & \psi_t & \psi^*_t & \gamma^{(1)}_{1,t} & \gamma^{*(1)}_{1,t} & \gamma^{(1)}_{2,t} & \dots & \gamma^{(2)}_{1,t} & \dots \end{bmatrix}' \tag{9.13}$$

and the system matrices

$$\mathbf{Z}_t = \begin{bmatrix} Z_\mu, & Z_\psi, & Z_{\gamma^{(1)}}, & Z_{\gamma^{(2)}} \end{bmatrix}, \quad \mathbf{T}_t = \text{diag}\left(T_\mu, \quad T_\psi, \quad T_{\gamma^{(1)}}, \quad T_{\gamma^{(2)}} \right)$$
$$\mathbf{R}_t = \text{diag}\left(R_\mu, \quad R_\psi, \quad R_{\gamma^{(1)}}, \quad R_{\gamma^{(2)}} \right), \quad \mathbf{Q}_t = \text{diag}\left(Q_\mu, \quad Q_\psi, \quad Q_{\gamma^{(1)}}, \quad Q_{\gamma^{(2)}} \right) \tag{9.14}$$

where

$$\begin{aligned} Z_\mu &= \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad Z_\psi = \begin{bmatrix} 1 & 0 \end{bmatrix} \\ Z_{\gamma^{(1)}} &= \begin{bmatrix} 1 & 0 & 1 & 0 & \dots & 0 \end{bmatrix}, \quad Z_{\gamma^{(2)}} = \begin{bmatrix} 1 & 0 & 1 & 0 & \dots & 0 \end{bmatrix} \end{aligned} \tag{9.15}$$

$$\begin{aligned} T_\mu &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad T_\psi = \begin{bmatrix} \cos\lambda_c & \sin\lambda_c \\ -\sin\lambda_c & \cos\lambda_c \end{bmatrix} \\ T_{\gamma^{(1)}} &= \text{diag}\begin{pmatrix} \cos\frac{2\pi j}{s} & \sin\frac{2\pi j}{s} \\ -\sin\frac{2\pi j}{s} & \cos\frac{2\pi j}{s} \end{pmatrix} \text{ for j} = 1, \ . \ . \ . \ , s/2 \text{ and } s = s_1 \\ T_{\gamma^{(2)}} &= \text{diag}\begin{pmatrix} \cos\frac{2\pi j}{s} & \sin\frac{2\pi j}{s} \\ -\sin\frac{2\pi j}{s} & \cos\frac{2\pi j}{s} \end{pmatrix} \text{ for j} = 1, \ . \ . \ . \ , s/2 \text{ and } s = s_2 \end{aligned} \tag{9.16}$$

$$\begin{aligned} R_\mu &= \mathbf{I}_2, \quad R_\psi = \mathbf{I}_2 \\ R_{\gamma^{(1)}} &= \mathbf{I}_{s_1}, \quad R_{\gamma^{(2)}} = \mathbf{I}_{s_2}, \end{aligned} \tag{9.17}$$

$$\begin{aligned} Q_\mu &= \begin{bmatrix} \sigma_\eta^2 & 0 \\ 0 & \sigma_\zeta^2 \end{bmatrix}, \quad Q_\psi = \begin{bmatrix} \sigma_{\tilde{\omega}^2} & 0 \\ 0 & \sigma_{\tilde{\omega}^2} \end{bmatrix} \\ Q_{\gamma^{(1)}} &= \sigma^2_{\omega^{(1)}}\mathbf{I}_{s_1-1}, \quad Q_{\gamma^{(2)}} = \sigma^2_{\omega^{(2)}}\mathbf{I}_{s_2-1} \end{aligned} \tag{9.18}$$

The state prior distribution should be set as $\boldsymbol{\alpha}_o \sim N(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$.

Bibliography

[1] Mohamed A. Abu-El-Magd and Naresh K. Sinha. Short-Term Load Demand Modeling and Forecasting: A Review. *IEEE Transactions on Systems, Man and Cybernetics*, 12(3):370–382, 1982.

[2] Charu C. Aggarwal. *Outlier Analysis.* Springer Science, springer edition, 2013.

[3] Hermine N. Akouemo and Richard J. Povinelli. Data Improving in Time Series Using ARX and ANN Models. *IEEE Transactions on Power Systems*, 32(5):3352–3359, 2017.

[4] Hermine N. Akouemo and Richard J. Povinelli. Data Improving in Time Series Using ARX and ANN Models. *IEEE Transactions on Power Systems*, 32(5):3352–3359, 2017.

[5] Krizhevsky Alex, Sutskever Ilya, and Geoffrey.E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 8, pages 713–772, 2012.

[6] Brett G. Amidan, Thomas A. Ferryman, and Scott K. Cooley. Data outlier detection using the chebyshev theorem. *IEEE Aerospace Conference Proceedings*, 2005:3814–3819, 2005.

[7] Brian D O Anderson, John B Moore, Lindsey Simon Di Franco Rmm, and Melsa Sage. Optimal Filtering. Information and system science series. 1979.

[8] Sylvain Arlot and Alain Celisse. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4:40–79, 2009.

[9] Azme Khamis. The Effects of Outliers Data on Neural Network Performance. *Journal of Applied Sciences*, pages 1394–1398, 2005.

[10] Jushan Bai. Least absolute deviation regression. *Econometric theory*, pages 403–436, 1995.

[11] Mesut E. Baran and Felix F. Wu. Network reconfiguration in distribution systems for loss reduction and load balancing. *IEEE Transaction on Power Delivery*, Vol. 34 No:1401, 1989.

[12] Michele Basseville and Igor V. Nikiforov. Detection of abrupt changes. Theory and application. *Control Engineering Practice*, 2(4):729–730, 1994.

[13] Behrouz A. Forouzan. Data Communication and Networking, 2007.

[14] Irad Ben-gal. Outlier Detection. *Data Mining and Knowledge Discovery Handbook*, pages 131–146, 2005.

[15] Olivier Bennet and Hannah Cromarty. British Summer Time. *The British Journal of Psychiatry*, 111(479):1009–1010, 2016.

[16] Lorenzo Beretta and Alessandro Santaniello. Nearest neighbor imputation algorithms: a critical evaluation. *BMC Medical Informatics and Decision Making*, 16(S3):74, 2016.

[17] Christoph Bergmeir and José M. Benítez. On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191:192–213, 2012.

[18] Filippo Maria Bianchi, Enrico Maiorino, Michael C. Kampffmeyer, Antonello Rizzi, and Robert Jenssen. An overview and comparative analysis of Recurrent Neural Networks for Short Term Load Forecasting. pages 1–41, 2017.

[19] George Box and George Tiao. Intervention Analysis with Applications to Economic and Environmental Problems. *Journal of the American Statistical Association*, 70(349):70–79, 1975.

[20] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. Time Series Analysis. *Chapman & Hall/CRC texts in statistical science series*, page 380, 2008.

[21] GEP Box and DR Cox. An analysis of transformations: Applying the Box-Cox transformation. *Journal of the Royal Satistical Society*, Series B(26):211–252, 1964.

[22] Ronald J Brachman and Tej Anand. The Process of Knowledge Discovery in databases. *Advanced in knowledge discovery and data mining*, pages 37–57, 1996.

[23] Derek W. Bunn and E D Farmer. *Comparative models for electrical load forecasting.* 1985.

[24] Mengmeng Cai, Manisa Pipattanasomporn, and Saifur Rahman. Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques. *Applied Energy*, 236(October 2018):1078–1088, 2019.

[25] R Campo and P Ruiz. Adaptative weather sensitive short-term load forecast. *IEEE Transactions on Power Systems,*, PWRS-2, No(August 1987):224–225, 1987.

[26] Bob Carpenter, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A Probabilistic Programming Language. *Journal of Statistical Software*, 76(1), 2017.

[27] Gavin C. Cawley and Nicola L. Talbot. On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *Journal of Machine Learning Research*, 11:2079–2107, 2010.

[28] Yacine Chakhchoukh, Patrick Panciatici, and Lamine Mili. Electric load forecasting based on statistical robust methods. *IEEE Transactions on Power Systems*, 26(3):982–991, 2011.

[29] Ih Chang, George C Tiao, Chung Chen, and Ih Chang. Estimation of Time Series Parameters in the Presence of Outliers. 30(2):193–204, 1988.

[30] Chung Chen and Lon-Mu Liu. Joint Estimation of Model Parameters and Outlier Effects in Time Series. *Journal of the American Statistical Association*, 88(421):284–297, 1993.

[31] Xinyu Chen, Chongqing Kang, Xing Tong, Qing Xia, and Junfeng Yang. Improving the accuracy of bus load forecasting by a two-stage bad data identification method. *IEEE Transactions on Power Systems*, 29(4):1634–1641, 2014.

[32] W.R Christiaanse. Short-term load forecasting using general exponential smoothing. *IEEE Transactions on Power Apparatus and Systems*, (2):900–911, 1971.

[33] S. Civanlar and J. J. Grainger. Forecasting distribution feeder loads: Modeling and application to volt/var control. *IEEE Transactions on Power Delivery*, 3(1):255–264, 1988.

[34] J. T. Connor. Robust neural network filter for electricity demand prediction. *Journal of Forecasting*, Vol 15:6(May):p 437–458, 1996.

[35] Marino Daniel L, Stefan Hosein, and Patrick Hosein. Building Energy Load Forecasting using Deep Neural Networks. *7th IEEE International Conference on Smart Grid Communications (SmartGridComm 2016)*, 2016.

[36] Alysha M. De Livera, Rob J. Hyndman, and Ralph D. Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, 106(496):1513–1527, 2011.

[37] Alysha M. de Livera, Rob J. Hyndman, and Ralph D. Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, 106(496):1513–1527, 2011.

[38] Haydar Demirhan and Zoe Renwick. Missing value imputation for short to mid-term horizontal solar irradiance data. *Applied Energy*, 225(March):998–1012, 2018.

[39] A.P. Dempster, N.M. Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B Methodological*, 39(1):1–38, 1977.

[40] Joenssen Dieter William and Bankhofer Udo. *Hot Deck Methods for Imputing Missing Data*. Springer edition, 2012.

[41] Ni Ding, Clementine Benoit, Guillaume Foggia, Yvon Besanger, and Frederic Wurtz. Neural network-based model design for short-term load forecast in distribution systems. *IEEE Transactions on Power Systems*, 31(1):72–81, 2016.

[42] Dieter Dohnal. On-load tap changers for power transformers. *MR Knowledge Base*, page 24, 2013.

[43] Koopman S. J. Durbin J. *Time Series Analysis by State Space Methods*. Oxford University Press, 2012.

[44] Ali Ehsan and Qiang Yang. State-of-the-art techniques for modelling of uncertainties in active distribution network planning: A review. *Applied Energy*, 239(January):1509–1523, 2019.

[45] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[46] J Y Fan and J D Mcdonald. A real-time implementation of short-term load forecasting for distribution power systems. *IEEE Transactions on Power Systems*, 9(2):988–994, 1994.

[47] A. J. Fox. Outliers in Time Series. *Journal of the Royal Statistical Society*, 34(3):350–363, 1972.

[48] Christopher Fox, Anany Levitin, and Thomas Redman. The notion of data and its quality dimensions. *Information Processing and Management*, 30(1):9–19, 1994.

[49] Andrew Gelman and Jennifer Hill. Missing-data imputation. *Data Analysis Using Regression and Multilevel/Hierarchical Models*, pages 529–543, 2007.

[50] Stuart Geman and Eli Bienenstock. Neural networks and the bias/variance dilemma. *Neural computation*, 4:1–58, 1992.

[51] Felix A. Gers, Douglas Eck, and Jürgen Schmidhuber. Applying LSTM to time series predictable through time-window approaches. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 2130, pages 669–676, 2001.

[52] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *International Conference on Artificial Intelligence and Statistics*, 9:249–256, 2010.

[53] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 15:315–323, 2011.

[54] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. *Nature Methods*, 13(1):35–35, 2015.

[55] Phillip G. Gould, Anne B. Koehler, J. Keith Ord, Ralph D. Snyder, Rob J. Hyndman, and Farshid Vahid-Araghi. Forecasting time series with multiple seasonal patterns. *European Journal of Operational Research*, 191(1):205–220, 2008.

[56] Alex Graves. *Supervised Sequence Labeling with Recurrent Neural Networks*, volume 12. 2013.

[57] Klaus Greff, Rupesh K. Srivastava, Jan Koutnik, Bas R. Steunebrink, and Jurgen Schmidhuber. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 2016.

[58] Frank R Hampel. A general qualitative definition of robustness. *The Annals of Mathematical Statistics*, 42(6):1887–1896, 1971.

[59] Frank R Hampel. The Influence Curve and Its Role in Robust Estimation. *Journal of the American Statistical Association*, 69(346):383–393, 1974.

[60] Pedro Hancevic and Diego Margulis. Daylight saving time and energy consumption: The case of Argentina. *Economic Policy*, (2116):0–33, 2013.

[61] A Harvey and S Peters. Estimation procedures for structural time series models. *Journal of Forecasting*, 9(2):89–108, 1990.

[62] Andrew c Harvey. *Forecasting, structural time series models and the Kalman filter.* 1989.

[63] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The Elements of Statistical Learning. *Elements*, 1:337–387, 2009.

[64] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter:1026–1034, 2015.

[65] Martin J. Heathcote and David Peter Franklin. The J & P Transformer Book: A Practical Technology of the Power Transformer. *The J & P Transformer Book*, page 945, 1998.

[66] S. I. Hill, F. Desobry, E. W. Garnsey, and Y. F. Chong. The impact on energy consumption of daylight saving clock changes. *Energy Policy*, 38(9):4955–4965, 2010.

[67] Mayer Hillman and Jon Parker. More daylight, less electricity. *Energy Policy*, 16(5):514–515, 1988.

[68] Henrique Steinherz Hippert, Carlos Eduardo Pedreira, and Reinaldo Castro Souza. Neural networks for short-term load forecasting : A review and evaluation. *IEEE Transactions on Power Systems*, 16(1):4333, 2001.

[69] David C. Hoaglin, Boris Iglewicz, and John W. Tukey. Performance of some resistant rules for outlier labeling. *Journal of the American Statistical Association*, 81(396):991–999, 1986.

[70] Sepp Hochreiter and J Urgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[71] Matthew D. Hoffman and Andrew Gelman. The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15:1593–1623, 2014.

[72] David Hutchison and John C Mitchell. *Neural Networks: Tricks of the Trade*, volume 7700. 2012.

[73] Nathalie Huyghues-Beaufond, Alex Jakeman, S.H. Tindemans, and Goran Strbac. Enhancing distribution network visibility using contingency analysis tools. In *IET International Conference on Resilience of Transmission and Distribution Networks*, pages 4 (6 .)–4 (6 .), 2018.

[74] Nathalie Huyghues-Beaufond, Alex Jakeman, Simon Tindemans, and Goran Strbac. Challenges in model and data merging for the implementation of a distribution network contingency analysis tool. *CIRED - Open Access Proceedings Journal*, 2017(1):1621–1624, 2017.

[75] Rob Hyndman, Anne Koehler, Keith Ord, and Ralph Snyder. *Forecasting with Exponential Smoothing The state Space Approach*. 2008.

[76] D. G. Infield and D. C. Hill. Optimal smoothing for trend removal in short term electricity demand forecasting. *IEEE Transactions on Power Systems*, 13(3):1115–1120, 1998.

[77] M.A. Kashem, V. Ganapathy, and G.B. Jasmon. Network reconfiguration for load balancing in distribution networks. *IEE Proceedings - Generation, Transmission and Distribution*, 146(6):563, 2002.

[78] Ryan Kellogg and Hendrik Wolff. Daylight time and energy: Evidence from an Australian experiment. *Journal of Environmental Economics and Management*, 56(3):207–220, 2008.

[79] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. pages 1–15, 2014.

[80] Weicong Kong, Zhao Yang Dong, Youwei Jia, David J. Hill, Yan Xu, and Yuan Zhang. Short-Term Residential Load Forecasting based on LSTM Recurrent Neural Network. *IEEE Transactions on Smart Grid*, 3053(c):1–11, 2017.

[81] Weicong Kong, Zhao Yang Dong, Youwei Jia, David J. Hill, Yan Xu, and Yuan Zhang. Short-Term Residential Load Forecasting based on LSTM Recurrent Neural Network. *IEEE Transactions on Smart Grid*, 10(1):841–851, 2019.

[82] Matthew J. Kotchen and Laura E. Grant. Does Daylight Saving Time Save Energy? Evidence From A Natural Experiment In Indiana. *Review of Economics and Statistics*, 93(4):1172–1185, 2011.

[83] Kevin J. Lang, Alex H. Waibel, and Geoffrey E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3(1):23–43, 1990.

[84] Marc Lavielle. Using penalized contrasts for the change-point problem. *Signal Processing*, 85(8):1501–1510, 2005.

[85] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[86] Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus Robert Müller. Efficient backprop. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7700 LECTU:9–48, 2012.

[87] Victor Levi, Mike Kay, and Ian Povey. Reverse power flow capability of tap-changers. *Electricity Distribution, 2005. CIRED 2005. 18th International Conference and Exhibition on*, 2, 2005.

[88] Kadir Liano. Robust error measure for supervised neural network learning with outliers. *IEEE Transactions on Neural Networks*, 7(1):246–250, 1996.

[89] R. J. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, 1987.

[90] Roderick J A Little. Missing-Data Adjustments in Large Surveys. *Journal of Business & Economic Statistics*, 6(No.3), 1988.

[91] Pete Loshin. *TCP/IP Clearly Explained, Fourth Edition (The Morgan Kaufmann Series in Networking)*. 2003.

[92] Gordon Lowry, Felix U. Bianeyin, and Nirav Shah. Seasonal autoregressive modelling of water and fuel consumptions in buildings. *Applied Energy*, 84(5):542–552, 2007.

[93] Oded Maimon and Lior Rokach. *The Data Mining and Knowlegde Discovery Handbook*. Tel-Aviv, tel-aviv u edition, 2005.

[94] Ms R Malarvizhi and Antony Selvadoss Thanamani. K-Nearest Neighbor in Missing Data Imputation. *International Journal of Engineering Research*, 5(1):5–07, 2012.

[95] Deep Medhi and Karthik Ramasamy. Packet Queueing and Scheduling Deep. *Network Routing*, 2018.

[96] Hrushikesh Narhar Mhaskar and Charles a Micchelli. How to Choose an Activation Function. *Advances in Neural Information Processing Systems 6*, pages 319–326, 1994.

[97] I. Moghram and S. Rahman. Analysis and evaluation of five short-term load forecasting techniques. *IEEE Transactions on Power Systems*, 4(4):1484–1491, 1989.

[98] Norizan Mohamed, Maizah Hura Ahmad, and Zuhaimy Ismail. Improving Short Term Load Forecasting Using Double Seasonal Arima Model. *World Applied Sciences Journal 15*, 15(2):223–231, 2011.

[99] Vinod Nair and Geoffrey.E Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th International Conference on International Conference on Machine Learning*, (3):807–814, 2010.

[100] F Palmer. Multiple imputation of time series: an application to the construction of historical price indexes. *Biltoki*, pages 1–10, 2005.

[101] Ronald K. Pearson, Yrjö Neuvo, Jaakko Astola, and Moncef Gabbouj. Generalized Hampel Filters. *Eurasip Journal on Advances in Signal Processing*, 2016(1), 2016.

[102] M Pincus, R Cummins, G Haddad, New Haven, and M Steven. Heart rate control infants. 1993.

[103] Erhard Rahm and Hong Hai Do. Data Cleaning: Problems and Current Approaches Erhard. *DOLAP: Proceedings of the ACM International Workshop on Data Warehousing and OLAP*, pages 49–56, 2007.

[104] Aowabin Rahman, Vivek Srikumar, and Amanda D. Smith. Predicting electricity consumption for commercial and residential buildings using deep recurrent neural networks. *Applied Energy*, 212(December 2017):372–385, 2018.

[105] David E Rumelhart, Geoffrey E Hinton, and R J Williams. Learning Internal Representations by Error Propagation, 1986.

[106] P. J. Santos, A. G. Martins, and A. J. Pires. Designing the input vector to ANN-based models for short-term load forecast in electricity distribution systems. *International Journal of Electrical Power and Energy Systems*, 29(4):338–347, 2007.

[107] A. J. Scott and M. Knott. A Cluster Analysis Method for Grouping Means in the Analysis of Variance. *Biometrics*, 30(3):507, 1974.

[108] Heng Shi, Minghao Xu, and Ran Li. Deep Learning for Household Load Forecasting A Novel Pooling Deep RNN. *IEEE Transactions on Smart Grid*, 3053(c):1–1, 2017.

[109] Shaohuai Shi, Qiang Wang, Pengfei Xu, and Xiaowen Chu. Benchmarking state-of-the-art deep learning software tools. *Proceedings - 2016 7th International Conference on Cloud Computing and Big Data, CCBD 2016*, pages 99–104, 2017.

[110] Mukwanga Willy Siti, Dan Valentin Nicolae, Adisa A. Jimoh, and Abhisek Ukil. Reconfiguration and load balancing in the LV and MV distribution networks for optimal performance. *IEEE Transactions on Power Delivery*, 22(4):2534–2540, 2007.

[111] Yenduri Sumath and Iyengar S. Performance Evaluation of Imputation Methods for Incomplete Datasets. *International Journal of Software Engineering and Knowledge Engineering*, 17(01):127–152, 2007.

[112] Souhaib Ben Taieb and Amir F. Atiya. A Bias and Variance Analysis for Multistep-Ahead Time Series Forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, 27(1):62–76, 2016.

[113] Leonard J. Tashman. Out-of-sample tests of forecasting accuracy: An analysis and review. *International Journal of Forecasting*, 16(4):437–450, 2000.

[114] J. W. Taylor. Short-term electricity demand forecasting using double seasonal exponential smoothing. *Journal of the Operational Research Society*, 54(8):799–805, 2003.

[115] Sean J. Taylor and Benjamin Letham. Forecasting at Scale. *American Statistician*, 72(1):37–45, 2018.

[116] Nicholas J Tierney and Dianne H Cook. Expanding tidy data principles to facilitate missing data exploration, visualization and assessment of imputations. pages 1–41, 2018.

[117] Charles Truong, Laurent Oudre, and Nicolas Vayatis. A review of change point detection methods. 2018.

[118] Charles Truong, Laurent Oudrez, and Nicolas Vayatis. Penalty learning for change-point detection. *25th European Signal Processing Conference, EUSIPCO 2017*, 2017-Janua:1569–1573, 2017.

[119] Ruey S Tsay. Outliers, Level Shifts, and Variance Changes in Time Series. *Journal of forecasting*, 7(May 1987), 1988.

[120] UK Power Networks. Flexible Plug and Play Low Carbon Networks: Novel Protection Relay Trial Report. (December):48, 2012.

[121] UK Power Networks. An Investigation into Alternatives to Directional Overcurrent Protection on Grid Transformers to Improve the Network Capacity to Accommodate Reverse Power Flow. 2015.

[122] Rene Vidal, Joan Bruna, Raja Giryes, and Stefano Soatto. Mathematics of Deep Learning. 2017.

[123] Yi Ching Yao. Estimating the number of change-points via Schwarz' criterion. *Statistics and Probability Letters*, 6(3):181–189, 1988.

[124] Paolo Frasconi Yoshua Gengio, Patrice Simard. Learning Long-Term Dependencies with Gradient Descent is difficult. *IEEE transaction on Neural Networks*, VOL.5, 1994.

[125] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. 2016.