This project was supported by the Centre for Global Eco-Innovation and is partly financed by the European Regional Development Fund

Centre for Global Eco-Innovation

Master of Science (MSc) by Research Thesis on

"Predicting Future Loads of Electric Vehicles in the UK"

By

RAHUL ROY

(MSc by Research: Innovation | 35095603)

Department of Entrepreneurship, Strategy, and Innovation

(October 2019)

EUROPEAN UNION
Investing in Your Future
European Regional Development Fund 2007-13

www.cgeinnovation.org

Declaration

This thesis is submitted to the Department of Entrepreneurship, Strategy, and Innovation and the Board of Examiners of Lancaster University in partial fulfilment of the requirements for the degree of Master of Science by Research in Innovation. I hereby attest that this thesis is entirely my own work (some ideas being developed through mutual discussions with my supervisors, Dr. Trivikram Dokka and Dr. David Ellis), and has not been submitted in substantially the same form for the award of a higher degree elsewhere. All sources of information have been fully referenced.

Acknowledgement

I take this opportunity to convey my sincere gratitude to my academic supervisors, Dr. Vikram Dokka, Dr. David Ellis and Dr. Mike Hazas, and my industry supervisors, Nick Storer and Esther Dudek for their peerless guidance and constructive appraisal of my progress throughout this MSc by Research project. I also convey my sincere gratitude to the CGE team for their consistent encouragement and help throughout the project. I would also like to thank my parents for being the pillars of my strength, motivation and dedication towards my goals throughout my life. Last but not least, I am grateful to the Almighty who has been my torchbearer to be virtuous throughout my life.

Abstract

This thesis aims to propose a robust statistical model to predict the future energy demand on low voltage distribution networks based on the data obtained from the EV (electric vehicle) trials of Electric Nation project, conducted from 2017 to 2018. While the ultimate objective of Electric Nation is to assess the impact of EV charging on distribution networks and enable the distribution network operators (DNOs) to make informed decisions on demand management, this research project, as part of Electric Nation, aims to build relevant statistical models that would help the industry partner, EA Technology, to forecast the quantum of energy consumption, with high accuracy, that EV charging would lead to. In current research, we develop four statistical models based on four different algorithms: we start with time series regression as the benchmark model and iteratively improve the forecast accuracy of the benchmark model by boosting methodology. In addition, we also explore deep learning models (LSTM networks as the data is sequential) and identify that such models, with little hyperparameter tuning, deliver the best forecast accuracy among all the models.

While chapter 1 lays the foundation of the thesis, chapter 2 critically reviews relevant academic literature in the field of EV charging and impact on the electric grid. Chapter 3 gives a brief overview of Electric Nation and introduces the current research project. Chapter 4 introduces the data obtained from the Electric Nation trials and gives a comprehensive report on exploratory data analysis. Chapter 5 discusses the mathematical formulation of the project and explains the most relevant classes of algorithms applicable in this project. Furthermore, chapter 5 also discusses the various methodologies we opted in this project. Chapter 6 presents the developed models, and chapter 7 summarises the findings and discusses the way forward.

Keywords: electric vehicle (EV); EV charging; time series; forecasting; regression; ARIMA; LSTM networks.

List of Figures

LIST OF TABLES

Contents

1. Introduction

1.1 EVs, EVs and More EVs

The global EV stock crossed 2 million in 2016, which makes up nearly 0.2% of 1 billion passenger light-duty vehicles in circulation (Robinson, 2018). In 2016, China and the US had the largest shares of EVs, 32% and 28% respectively (Robinson, 2018). The UK has seen a 15-fold increase in the number of electric vehicles (EVs) between 2013 and 2015 (Godfrey, 2016). In fact, some forecasts even predict that the number of EVs would shoot to over one million by 2020 (Godfrey, 2016), although Chargemaster, UK's largest provider of EV charging infrastructure, forecasts the same figure by 2022 (Polar-Plus, n.d.). Despite the distinct forecasts on the growth of EVs as reported by different agencies, all of them indicate that the UK roads would be definitely inundated with hundreds of thousands of EVs in the next three to five years. The latest stats available as of July 4th, 2019 indicate that the sales of EVs rose by 61.7% in June 2019 compared to that in June 2018, with 2461 new battery electric vehicles (BEVs) registered in the UK in that month (Driving-Electric, 2019). The Society of Motor Manufacturers and Traders suggests that 60.3% more EVs have been sold so far in 2019 compared to the same last year (Driving-Electric, 2019). Chargemaster furthers that the current growth in EV registrations would see 60% of new cars being electric by 2030 (Polar-Plus, n.d.). Moreover, Chargemaster forecasts that one in every four vehicles would be EV by 2025 (Polar-Plus, n.d.). Keeping pace with rising trends of EV registrations, the UK automobile market would see more than 30 new EV models by 2020 giving a plethora of options to the consumers (Polar-Plus, n.d.).

1.2 Is EV Charging A Concern?

The EV charging infrastructure has been expanding to match the rise of EVs and hence, the charging requirements in the UK automobile market. POLAR, the UK's largest EV public charging network operated by Chargemaster, added more rapid chargers in 2017 than all

other public networks combined, leading to a market share of 51% (Polar-Plus, n.d.). Other major UK-wide networks include Ecotricity, Pod Point and Charge Your Car. As more EVs hit the UK roads, the charging infrastructure would be expected to expand proportionately. As more EVs and more charging stations come to reality, the demand for additional power from local distribution networks would increase.

If majority of the people in a locality start charging their EVs at the same time, then the additional demand would create stress on the distribution networks even if the duration of charging is small, the power ratings of the chargers are smaller than 7 kW and the battery ratings of the EVs are less than 75 kWh. Based on UK's National Grid's (NG) study, there would be 90% penetration of EVs by 2050 leading to an increased demand of 46 TWh ($1\ TWh = 10^9\ kWh$); this is over and above the total demand of 308 TWh in 2016 (Robinson, 2018). Robinson (Robinson, 2018) states that the impact of increased demand would depend upon the charging scenario. For instance, a sensible charging structure would encourage people to charge their EVs during off-peak periods, thereby reducing the impact on distribution networks (Robinson, 2018). My Electric Avenue project predicts that reinforcement of low-voltage (LV) distribution networks would be required when 40-70 % of the vehicles are EVs (Godfrey, 2016). Low Carbon London estimates a four times increase in the current LV network reinforcement spend during an increased EV uptake (Godfrey, 2016). Reinforcement across the whole network would also cause disruption to customers while the work is under progress (Godfrey, 2016). Besides, the rate of EV uptake may also redefine reinforcement plans (Godfrey, 2016). Furthermore, with rising uptake of EVs, there's a high probability of cluster formation within localities (Godfrey, 2016), i.e., clusters of people having charging requirements at the same time. Based on the estimates from Zap-Map as shown earlier in this section, such clusters would lead to increased load on the LV distribution networks. Hence, it is important for power distribution companies to devise plans to minimise the effect of such clustering.

In this project, we proposed deep learning models (LSTM networks) to forecast consumption of energy caused by EV charging in future scenarios. Summarised results of one such scenario in the city of Lancaster during a week in winter season in 2040 is presented below. The details can be found in section 7.1.1.

Scenario of Lancaster, UK (2040):

Start Date: 24/12/2040

End Date: 30/12/2040

Season: Winter

The mean consumption of energy for EVs based on a range of battery capacities are tabulated below (detailed description on the choice of battery capacity ranges is presented in section 4.2).

Table 1: Summary of scenario-based forecasting for the city of Lancaster, UK in 2040

Battery Capacity Range (kWh)	Estimated number of EV owners	Mean Consumption of Energy per day (kWh)
0-25	39673	106151.8
26-50	21176	99036.6
51-75	8522	42620.4
76-100	3651	18458.1

The results indicate an increased consumption of energy by 266,266.9 kWh per day in the city of Lancaster in the year 2040. Although this estimate is based on a few assumptions, it still gives a faithful indication of the quantum of stress the LV distribution networks would have in case EVs are allowed to charge without restrictions.

In further sections, we use the words ‘predictors’ and ‘features’ interchangeably. However, both the words mean the same: set of variables used to predict the target variable.

2. A Review of Academic Literature

2.1 Introduction

Estimating the energy requirements prior to EV charging is part of emergency preparedness in distribution networks as any additional bulk demand on the network may lead to frequent outages. Researchers, in the past, answered a plethora of questions relevant to EV charging, including EV uptake scenarios and the stress it causes on electricity networks, via extensive studies and analyses across different locations worldwide.

In past, researchers worked with data which included:

- Geography of consumers
- Types of vehicles (hybrid or battery operated)
- Capacity of vehicle batteries, although with small variance
- Energy consumed from chargers or batteries of vehicles
- Vehicle usage
- Household demand profile
- Demographics
- Weather conditions
- Mechanics of vehicles

Using different subsets of the aforementioned data, researchers in the past estimated consumption of energy of EVs from not only chargers but also their batteries while the vehicles were running. Although energy data were mostly sequential, modelling methods were largely restricted to rudimentary regressions and relevant time series methods were not deployed: this can be attributed to the objective that past researches primarily focussed on assessing the impact of either EV charging on distribution networks or kinematics on the consumption of energy from the batteries, with the former emphasised on understanding the stress levels EV charging would cause on the existing infrastructure of electricity networks. In the current

research project, the focus had been on developing predictive models, using relevant machine learning algorithms and time series dynamics, that would act as a tool for distribution network operators to estimate how much additional energy would be required in a local network when clusters of EVs get charged. The project leveraged machine learning and time series dynamics to develop models to forecast energy consumption during EV charging based on an initial understanding of the data that might be available to the distribution network operators.

2.2 Circa 2011 – 12

Green II et al. (Green II, et al., 2011) assessed the impact of PHEVs (plug-in hybrid EVs) on the distribution networks by analysing four important factors: driving patterns, charging characteristics, charge timing and vehicle penetration. The authors suggested that leveraging a combination of MATSim and power simulation systems to further analyse the impact of PHEVs on the distribution networks could be insightful (Green II, et al., 2011). Higgins et al. (Higgins, et al., 2012) implemented a diffusion model, linking features of multi-criteria analysis and choice modelling, and applied it to estimate the market share of different types of EVs, using the vehicle stock of 1.5 million households in the city of Victoria, Australia. The authors identified that the geographical differences in uptake of EVs were primarily attributed to driving distance, employment status and household income, with urban areas having approximately thrice the proportional uptake (Higgins, et al., 2012). Higgins et al. (Higgins, et al., 2012) tested the model for a range of incentives to demonstrate its ability to inform and evaluate policy options.

2.3 Circa 2013 – 14

Xydas et al. (Xydas, et al., 2013) leveraged data mining methods such as decision tables, decision trees, artificial neural networks and support vector machines, to forecast EV load. The model was built using data on previous day load, number of the week, day of the week, type of day, number of new plug-ins every half-hour and total charging connections every half-hour (Xydas, et al., 2013). The authors used two scenarios, next day forecast and next week

forecast, and forecasting was done for each scenario separately (Xydas, et al., 2013). The objective of the study was to validate the use of data mining methods to forecast the EV charging load. However, the authors did emphasise that more cases should be studied to have a better understanding of the key attributes that indicate the choice of a given method over another (Xydas, et al., 2013). Foley et al. (Foley, et al., 2013) studied the impacts of EV charging in an actual working electricity market in Ireland. The authors developed a model of Ireland's electricity market in 2020 using the power systems market model called PLEXOS for power systems for both peak and off-peak scenarios (Foley, et al., 2013). The authors quantified the impact of EV charging by firstly simulating a baseline scenario without any EV load (Foley, et al., 2013). The model was then run with EV load for both peak and off-peak loads (Foley, et al., 2013). The baseline scenario was then compared to both peak and off-peak scenarios to determine the effect of EV load (Foley, et al., 2013). Hoed et al. (Hoed, et al., 2013) analysed the actual usage patterns of public charging infrastructure in the city of Amsterdam, based on more than 109,000 charging events in the year 2012-13. The authors identified that as the charging infrastructure expanded in Amsterdam, the number of charging sessions also increased from 2012 to 2013 (Hoed, et al., 2013). The per month consumption of energy increased from 55 MWh (April 2012) to 109 MWh (March 2013) (Hoed, et al., 2013). The mean consumption was found to be 8.31 kWh per charging event (Hoed, et al., 2013). The authors also identified that the consumption of energy grew significantly from September 2012 onwards; this can also be verified from the plot shown below in figure 1 (Hoed, et al., 2013). The analysis was intended to deliver insights in the actual usage patterns of public charging infrastructure, eventually increasing effectiveness of the existing system and optimising the roll out of further charging stations by municipalities (Hoed, et al., 2013).

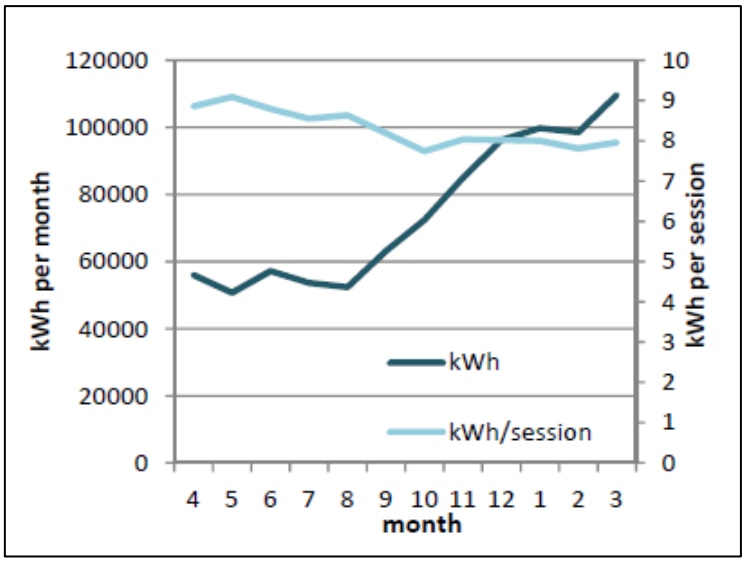

Figure 1: kWh consumption across sessions and months in Amsterdam (Hoed, et al., 2013, p. 6)

Paevere et al. (Paevere, et al., 2014) discussed a suite of models for the city of Victoria (Australia) to obtain spatial and temporal projections of charging demand and peak-shaving potentials from EVs. Paevere et al. (Paevere, et al., 2014) discussed models for future EV uptake, travel by households, household electricity demand and recharge/discharge of EVs at their home locations. An overview of the modelling strategy is shown below in figure 2 (Paevere, et al., 2014).

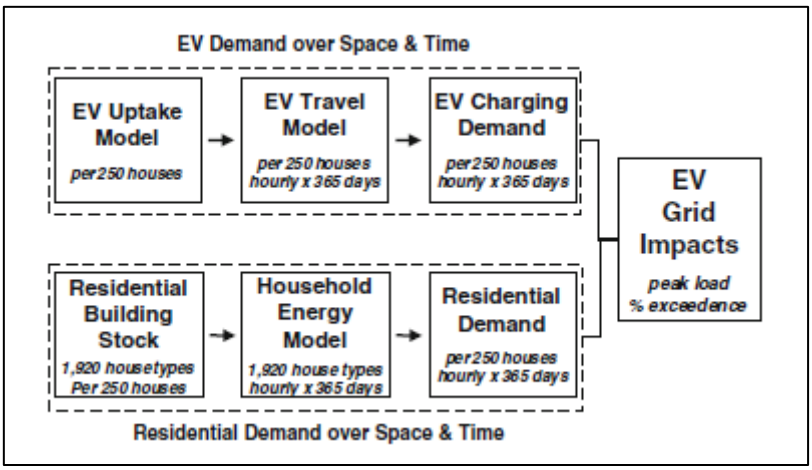

Figure 2: Modelling strategy by Paevere, et al., 2014 (Paevere, et al., 2014, p. 64)

The authors also emphasised that the shape and magnitude of EV charging demand profiles were dependent on the geography (Paevere, et al., 2014). Paevere et al. (Paevere, et al., 2014) projected that the average peak daily charging load under a demand charging scenario in Victoria was 0.66 kW per EV. The authors concluded that under the expected EV penetration in Victoria by 2033 and demand charging, the projected increase in peak electrical loads was mostly less than 10% (Paevere, et al., 2014). Khoo et al. (Khoo, et al., 2014) addressed the issue of impacts contributed by different EV user categories and models to peak loads through statistical analysis of the charge events in the Victorian EV Trial in Australia. Moreover, they also modelled (from probability distributions to regression) the relationships between EV types and attributes of the charge events such as charge duration, daily charge frequency, energy consumed, start charge hour, and time to next charge event (Khoo, et al., 2014). The Victorian EV trial saw 178 participants (70% participants being household) generating 4333 charge events with a total energy consumption of 33 MWh over a duration of 12,170 hours (Khoo, et al., 2014). Khoo et al. (Khoo, et al., 2014) projected the mean and maximum percentage increase in the energy demands between 3.27% and 5.70%, and 5.72% and 9.79% respectively in the summer of 2032/33 based on the future EV uptake scenarios.

2.3 Circa 2015 – 16

Wang et al. (Wang, et al., 2015) highlighted that energy consumption during driving of EVs is largely determined by driving behaviour, road topography information, and traffic situation. They (Wang, et al., 2015) proposed an offline algorithm that gave two energy consumption results, one for the maximum driving speed and the other for the most economical driving speed, to give a first impression to the driver on the possible energy consumption and therefore, the range which the EV can cover even before the actual trip. Furthermore, Wang et al. (Wang, et al., 2016) also proposed an online energy consumption algorithm that would help in adjusting the energy consumption prediction during driving of BEVs; this will be based on a number of factors, such as vehicle characteristics, driving behaviour, route information, traffic states and weather conditions. De Cauwer et al. (De Cauwer, et al., 2015) detected and

quantified correlations between the kinematic parameters of EVs and their energy consumption from the batteries. The authors developed three regression models for energy consumption as a function of the EV's kinematic parameters, such as travel distance, travel time, temperature, and acceleration, using three different aggregation levels for the variables (De Cauwer, et al., 2015). Neaimeh et al. (Neaimeh, et al., 2015) focussed on studying the impact of EV penetration by leveraging a probabilistic approach and network models in rural and urban households. They studied data that included 44 EVs, 85000 journeys, 19000 charging events and 125 users. Neaimeh et al. (Neaimeh, et al., 2015) analysed only domestic charging events that involved domestic load profiles of half-hourly power consumption and EV usage profile including time, battery current, voltage and state of charge. Such variables were then used to compute other variables such as duration of charging and consumption of energy (Neaimeh, et al., 2015). Monte Carlo Simulation was used to build a distribution of possible demands on the trial networks using data produced by sampling domestic load profiles and EV charging profiles (Neaimeh, et al., 2015). Neaimeh et al. (Neaimeh, et al., 2015) assessed the impact of EV charging by evaluating the effect of peak load on a given day in the month of January. Neaimeh et al. (Neaimeh, et al., 2015) suggested that it would be beneficial for the DNOs to distribute EV charging across both space and time, i.e., allowing a plethora of options, such as work, public, home and rapid, for the EV owners to charge their vehicles could be helpful in alleviating the impact of EV charging on domestic networks. Moreover, allowing the roll-out of EV charging infrastructure in association with relevant market players would be an efficient way to manage existing distribution networks (Neaimeh, et al., 2015). Xydas et al. (Xydas, et al., 2016) proposed a 'risk level' index using fuzzy logic to assess the impact of EV demand on distribution networks. Xydas et al. (Xydas, et al., 2016) developed three modules, clustering, correlation and regression, and assimilated information from all the three modules to generate a 'risk level' index. While the clustering module created daily demand profiles in a given geographical location using k-means clustering, correlations were computed between weather attributes and daily peak power of EVs' demand in a geographical location in the correlation module (Xydas, et al., 2016). The regression module involved computation of linear regression model of monthly demand as a function of time, i.e., modelling with a trend

(Xydas, et al., 2016). A high 'risk level' index indicated high risk on the distribution networks (Xydas, et al., 2016). Xydas et al. (Xydas, et al., 2016) identified Leicestershire and Nottinghamshire having high 'risk level' indices, while West Midlands with low 'risk level' index, based on the available data.

2.4 Circa 2017 – 18

Bi et al. (Bi, et al., 2018) proposed a combined model for charging time prediction as a function of the amount of state of charge based on regression and time series methods according to the data from 70 battery electric vehicles (BEVs) operating in Beijing, China. Bi et al. (Bi, et al., 2018) argued that accurate charging time prediction of BEVs could help drivers determine travel plans and driving range. Moon et al. (Moon, et al., 2018) estimated the changes in electricity charging demand based on consumer preferences for EVs, charge time of the day and types of EV supply equipment (EVSE) for the Korean market. The authors used a mixed logit model (consumers' utility function for vehicles) to estimate consumers' preferences for different types of vehicles (Moon, et al., 2018). Consumers' preferences were analysed for deriving the choice probability of EVs to assess the potential market size of EVs (Moon, et al., 2018). Moon et al. (Moon, et al., 2018) also analysed charging patterns by surveying consumers to know about their preferred EVSE (private or public) and time of day. Total electricity demand was estimated using total EV owners, average distance travelled per day, and average fuel efficiency of current EVs (Moon, et al., 2018). The authors suggested that the EV market in Korea could increase by 73,000 vehicles annually (Moon, et al., 2018). The authors also indicated that, based on their analysis and estimates, the current power grid infrastructure in Korea might not be able to meet the peak demand of energy in some areas (Moon, et al., 2018). López et al. (López, et al., 2018) proposed a demand response strategy based on machine learning to control EV charging in response to the real-time pricing, such that the overall energy cost of an EV was minimised. The authors assumed a hypothetical case when perfect knowledge of the future, such as fuel prices, vehicle data, was present and an optimal set of actions was proposed using dynamic programming, i.e., to analyse historical

data to select the best moments to charge EVs (López, et al., 2018). López et al. (López, et al., 2018) used the optimal set of actions, in combination with an information system, to learn which variables and under what conditions the charging decisions should be made, using machine learning algorithms. An overview of the proposed methodology is shown below in figure 3 (López, et al., 2018).

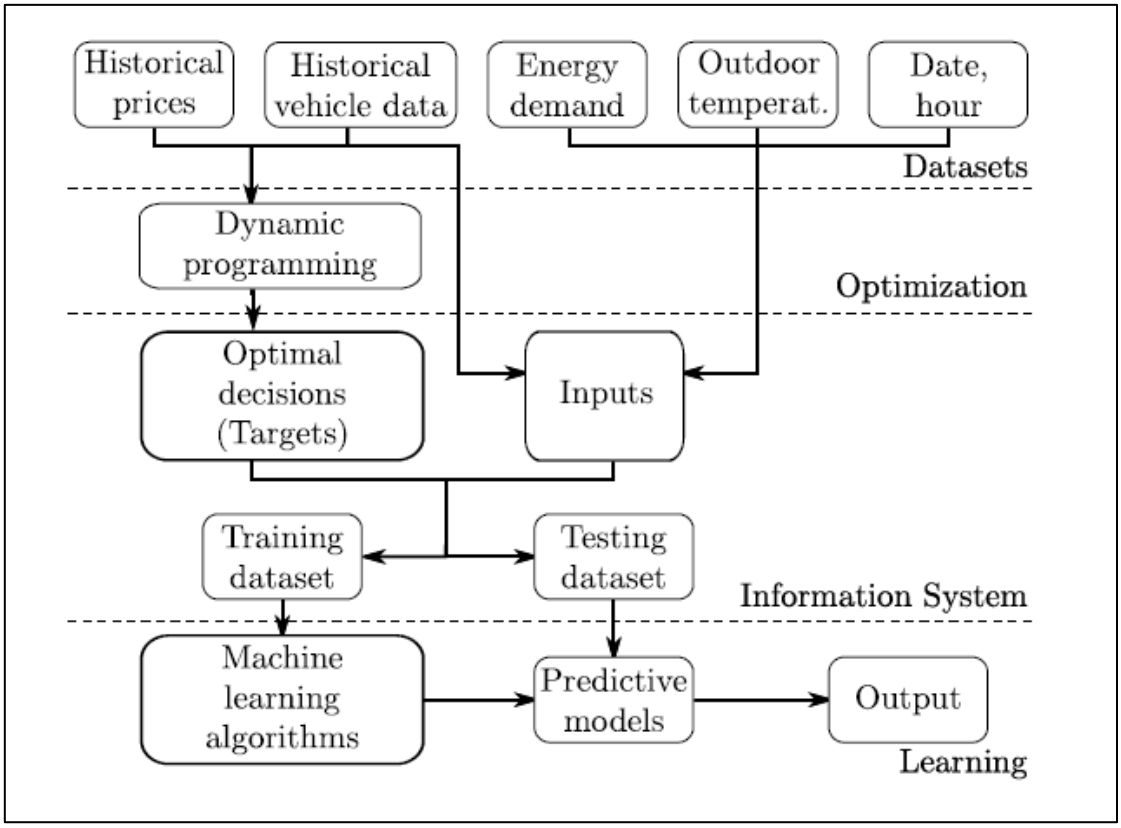

Figure 3: Methodology proposed by López, et al., 2018 (López, et al., 2018, p. 4)

3. The Project

3.1 Electric Nation

Electric Nation, the world's largest EV trial, is the customer-facing brand of CarConnect, which is a Western Power Distribution (WPD) and Network Innovation Allowance (NIA) funded project (Dudek, 2017). WPD collaborated with EA Technology, DriveElectric, Lucy Electric GridKey and TRL for the Electric Nation project (Dudek, 2017). Electric Nation attempts to answer the challenge that when 40-70 % of households within a local distribution network have EVs, at least 32% of these networks across the UK would require intervention (Dudek, 2017). The project involved developing and delivering a number of smart charging solutions to support EV uptake on local networks (Dudek, 2017). One of the major outcomes of this project would be the Network Assessment Tool (NAT) that would analyse EV related stress on networks and would identify the best economic solution (Dudek, 2017).

3.1.1 Demand Management

The essence of demand management service (DMS) is to regulate the consumption of energy to minimise the level of demand related stress on the electricity networks. The demand management system relevant to Electric Nation can be well captured in figure 4 (Dudek, 2017).

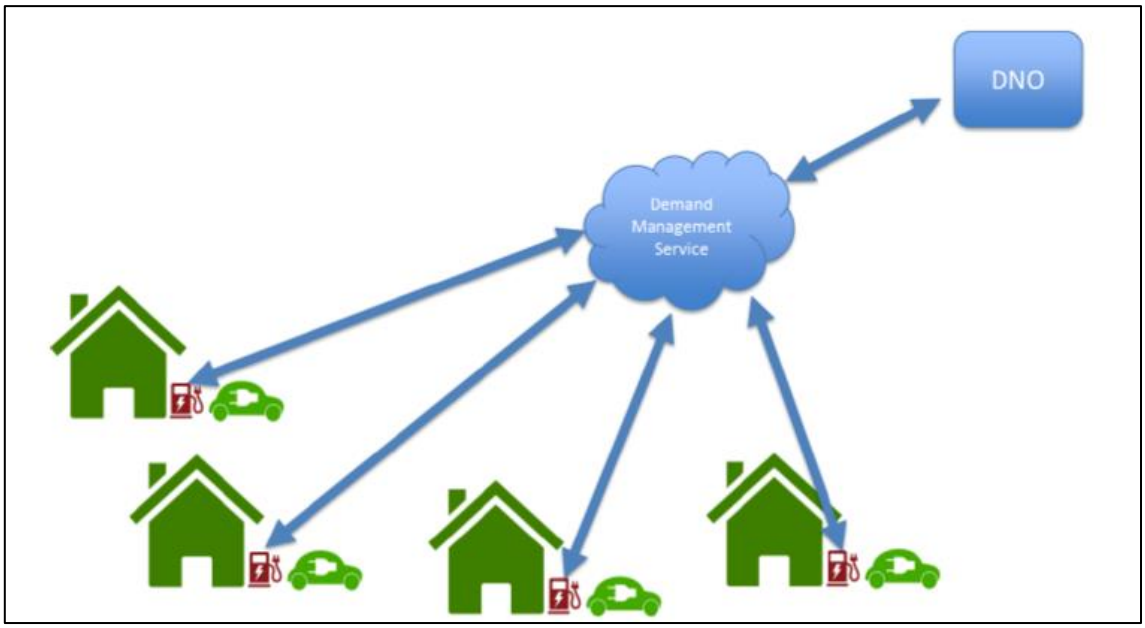

Figure 4: Demand management system (Dudek, 2017, p. 6)

The figure shows that the flow of information is bidirectional:

- Smart charger to DMS: plug-in status and status of charging, i.e., whether a car is connected and drawing current (Dudek, 2017).
- DMS to smart charger: electric current available for charging (Dudek, 2017).

This information is leveraged by the DMS to ensure that the total demand of groups of chargers under a given substation is within the limit set by the distribution network operator (DNO) (Dudek, 2017). In Electric Nation, two separate DMS providers, GreenFlux (GF) and CrowdCharge (CC), participated and a total of 673 smart chargers were installed (Dudek, 2017). Each DMS provider uses a different algorithm to allocate current to the chargers connected; moreover, the quantum of data they have and they handle, and the way the customers interact with their systems also differ (Dudek, 2017). The two DMS providers used different equipment during the trials. While CC used an APT/eVolt charger, GF worked with an ICU charger (Dudek, 2017). Both the DCS systems communicated to the customers' routers either directly or via a Wi-Fi bridge (Dudek, 2017). For the GF systems, a GSM sim was also installed inside the charger, providing a back-up form of communication (Dudek, 2017).

3.1.2 Test Rig

A test rig was designed, built and commissioned at EA Technology's premises in Capenhurst (Dudek, 2017). The test rig had twelve chargers, six APT and six ICU, with additional monitoring equipment (Dudek, 2017). The rig served two main purposes:

- To test the response of EVs to changes in the available current; this was to ascertain that the EVs would regulate their charging based on the available current (Dudek, 2017).
- To confirm the behaviour of re-configured algorithms over multiple cycles to show that current was allocated impartially among chargers without breaching the DNO limit (Dudek, 2017).

The aim of building the test rig was to test the behaviour of individual cars before flagging off the customer trials; this also involved re-configuring the algorithm (s) of demand management for both GF and CC DMS providers (Dudek, 2017). Moreover, the test rig trial also helped in confirming the behaviour of chargers if communications are lost from the DMS (Dudek, 2017). Both the DMS systems were tested and it was confirmed that their initial configurations were fit for the purpose of customer trials under varying capacity limits (Wells & Dale, 2017).

3.1.3 The 3-Pronged Approach

The aim of the Electric Nation project is to enable the DNOs to identify those parts of their networks that would be affected by EV uptake and therefore, EV charging (Wells & Dale, 2017). To achieve this, Electric Nation adopted a three-pronged approach.

1. Modelling: This phase would provide DNOs with an assessment tool to identify the areas in their networks that are most likely to be impacted by EV charging (Wells & Dale, 2017). Moreover, the tool would also enable the DNOs to have a detailed assessment of the level of risk on the networks caused by EV charging and would enable to decide on whether reinforcement would be necessary (Wells & Dale, 2017).
2. Monitoring: This phase would develop an algorithm that can be deployed on an existing substation monitoring facility, enabling the effect of charging EVs on LV networks to be retrospectively analysed (Wells & Dale, 2017).

3. Mitigation: This phase would see adapting existing smart charging technology, including V2G chargers if possible (Wells & Dale, 2017). The aim is to prove the technical and economic viability of domestic EV charging demand control and V2G services, to defer or minimise network reinforcement (Wells & Dale, 2017). This phase would involve customer trials with a wide range of EVs having a breadth of battery capacities and charging rates to reflect a diverse EV market (Wells & Dale, 2017).

3.2 Predicting Future Loads of Electric Vehicles in the UK

The MSc by Research project, 'Predicting Future Loads of Electric Vehicles in the UK' (under the aegis of Centre for Global Eco-Innovation, Lancaster University) in collaboration with EA Technology was aimed at estimating the additional load on distribution networks caused by EV (electric vehicle) charging. This project was part of the subproject 'Mitigation' under the purview of Electric Nation. Under the subproject Mitigation, customer trials were conducted between January 2017 and December 2018 (Electric-Nation, 2019). A summary of the trials can be found below.

- 673 smart chargers were installed at customers' homes throughout the licensed areas of WPD (Electric-Nation, 2019).
- The trials included 40 different types, makes or models of EVs (Electric-Nation, 2019).
- Smart charging was provided by two DMS providers, GF and CC. These providers used different algorithms to control EV charging and customer-facing systems (Electric-Nation, 2019).
- The smart chargers at customers' premises could report the plug-in time of EVs and their active charging start time (Electric-Nation, 2019). Besides, smart chargers were also capable of receiving instructions for pausing or reducing charging (Electric-Nation, 2019).
- Trial 1: Customers were not told that EV charging was regulated, and they could not interact with the smart charging system (Electric-Nation, 2019).
- Trial 2: Customers were given apps to enable them to interact with the smart charging system (Electric-Nation, 2019).

- Trial 3: Leveraged a simulated Time of Use (ToU) tariff to reward customers for changing their charging behaviour such as charging their EVs during the off-peak time (Electric-Nation, 2019).

EA Technology was interested in a robust statistical model that would give an estimate of the additional load on the LV distribution networks caused by EV charging under unsupervised conditions, i.e., if the EV owners have access to unconstrained charging, how much power would be consumed solely due to EV charging? This objective could have been addressed with different time granularities; for e.g., additional energy consumption per hour or per day, and so on. EA Technology was interested in the per day estimate of the additional load caused by EV charging. Hence, at any phase of the analysis, day-wise aggregation of data or results was indispensable as it was warranted by the project (business) objective. We did day-wise aggregation of the raw data before beginning the analyses and modelling, and converted the data into a day-wise time series (section 4.3). Another possibility (which was not analysed) could have been converting the raw data into hour-wise sequential data and then, further aggregating the results to get day-wise forecasts. However, day-wise aggregation of data prior to analyses and modelling offered an obvious advantage over other time aggregations: imputation of missing values in the day-wise sequential data was relatively easier and reliable. This is because charging events or transactions were randomly distributed in time in a given day and it was difficult to spot the exact point in time when a missing charging event could have taken place. This fact was also corroborated by EA Technology, who emphasised that based on the data available, it was challenging to identify which transactions might have been missing, both in magnitude and timestamp in a given day. Since a missing transaction with a fixed timestamp couldn't be spotted with certainty in a day but could certainly be estimated to have occurred in a particular day, working with day-wise aggregated data offered certainty relevant to aggregated charging events over other time-aggregations. Summarily, higher the frequency of time aggregations, higher was the uncertainty in imputing the missing values.

In further discussion, we will use the terms ‘forecast’ and ‘ prediction’ interchangeably. At places, we may also use the term ‘estimate’. However, all the terms hold the same meaning in the report.

4. Exploratory Data Analysis (EDA)

4.1 Description of GreenFlux (GF) and CrowdCharge (CC) Data

The trials, as part of the Electric Nation project, began in 2017. As such, the data collection, too, commenced in 2017. The GF transaction data had 80,313 observations across 12 variables (GF data collected from 01/02/2017 to 30/12/2018). The CC transaction data had 71,264 observations across 13 variables (CC data collected from 04/05/2017 to 03/11/2018). In addition, another dataset, Charger Install (CI), containing information on 673 participants involved in the trial was also present. In the CI dataset, an additional variable, *Min_Charging_Time* was created that informed the total charging time (in hours) when a battery got charged from 0% to 100% uninterruptedly. Two variables from the GF data and three variables from the CC data were dropped before preliminary analysis, as they were found to be redundant or irrelevant to the objective of the project.

The essence of delivering data-driven intelligence to any business depends on how tidy the data is (Wickham, 2014). Businesses, in general, are inundated with large volumes of data but data in its raw form can't be used to train machine learning algorithms, to generate sensible meaning and results. We need to pre-process the raw data so that the learning algorithm can be trained to yield meaningful insights and results, eventually rendering intelligence to businesses. As such, data pre-processing was imperative to our way-forward. We began with an initial inspection of the data and observed, and rectified the following issues.

- The GF data had 79,056 missing values, which were spread across the following variables: *Trial* (27,690), *GroupID* (27,690) and *ActiveChargingStart* (23,676). The CC dataset had 80,031 missing values, spread across the following variables: *ParticipantID* (3899), *CarKW* (3899), *CarKWh* (3899), *GroupID* (24,034), *Trial* (24,034), *ConsumedkWh* (1) and *ActiveChargingStart* (20,265). It's important to note that the total number of missing

values were aggregated over all the variables and hence, did not represent the actual number of rows with missing information.

- In the GF transaction data, 1 observation showed vehicle plug-in time in the year 2022 and 2 observations showed plug-out time in the year 2022. These anomalies were attributed to the communication loss between the Demand Control System (DCS) and the smart chargers. No such anomaly was observed for the CC data.
- As per the norms of the industry partner (EA Technology), observations with consumed energy less than 0.5 kWh or greater than 100 kWh were dropped from the CC data. Such observations were already dropped by EA Technology for the GF data before it was made available for analyses.
- The missing values for *Trial* and *GroupID* were actually the events when the charging was unsupervised; hence, no IDs were assigned to the two variables, thereby resulting in missing values. Such events were recoded as "Unmanaged"; this brought down the number of missing values in GF data to 23,676 and in CC data to 19,707.
- In the GF transaction data, transactions were recorded for only 301 participants, although the CI dataset showed that 345 participants had GF DCS. So, there was no information for 44 participants. In the CC data, transactions were also recorded for only 300 participants, although the CI dataset showed that 328 participants had CC DCS. So, there was no information for 28 participants. The missing information on 44 GF and 28 CC participants were attributed to the fact that the chargers for those participants never communicated with the DCS and hence, no data was collected.
- GF and CC datasets were separately combined with CI dataset to create two new datasets, GF-CI, and CC-CI respectively, having 18 variables each, including the ones created for analyses (discussed later). The missing values were present because of the unavailability of the start time of vehicle charging; this information could neither be retrieved nor be estimated and hence, were dropped from the datasets after advice from EA Technology. The final GF-CI and CC-CI datasets had 56,637 and 50,085 observations respectively.

Based on the objective, additional features, *Charging_Time_Hour*, *ActiveChargingStop,* and *capacity_kWh,* were extracted from the GF-CI and CC-CI transaction datasets (Zheng & Casari, 2018). Each observation in the dataset conveyed a unique transaction (charging event) with information captured by 18 variables. Description of the 18 variables in the combined datasets is as follows.

1. *ChargerID*: ID of charger assigned to a consumer.
2. *ParticipantID*: ID of the consumer.
3. *CarkW*: Power rating of EV battery in kW.
4. *CarkWh*: Energy rating of EV battery in kWh.
5. *Min_Charging_Time*: Time (hours) required to fully charge an EV from 0% to 100% based on CarkW and CarkWh.
6. *GroupID*: ID of the group to which a consumer was assigned during the trials. It kept changing as the trials progressed.
7. *Trial*: ID of the trials. In total, 4 trials were conducted chronologically; they were: unmanaged or unsupervised, 1, 2 and 3.
8. *AdjustedStartTime*: Time at which an EV was plugged in.
9. *ActiveChargingStart*: Time at which an EV started charging.
10. *ConsumedkWh*: Energy (kWh) consumed during a given transaction.
11. *AdjustedStopTime*: Time at which an EV was plugged out.
12. *Charging_Time_Hour*: Duration of charging (hours), assuming uninterrupted charging. It was computed by taking the ratio of *ConsumedkWh* and *CarkW*.
13. *ActiveChargingStop*: Time at which an EV stopped charging. It was estimated by adding *Charging_Time_Hour* to *ActiveChargingStart*.
14. *DCSProvider*: Type of DCS (GF or CC) connected to the charger.
15. *CarMake*: Brand of EV owned by the consumer.
16. *CarModel*: Model of EV owned by the consumer.
17. *PIVType*: Type of plug-in vehicle (PHEV, REX or BEV) owned by the consumer.
18. *capacity_kWh*: Category/bin of battery energy rating (kWh); initially, four categories/bins were created, 0-25, 26-50, 51-75 and 76-100; this was in agreement with EA Technology's

expectations (validity of the choice of bins was warranted by the conclusions drawn from clustering analysis, discussed later in section 4.2). For instance, if a consumer had owned a car with CarkWh of 33, he was assigned the bin 26-50.

Distribution of vehicles based on battery capacities can be observed from the boxplots (figure 5). We observe that the majority of the consumers in the trials preferred cars with lower battery capacities.

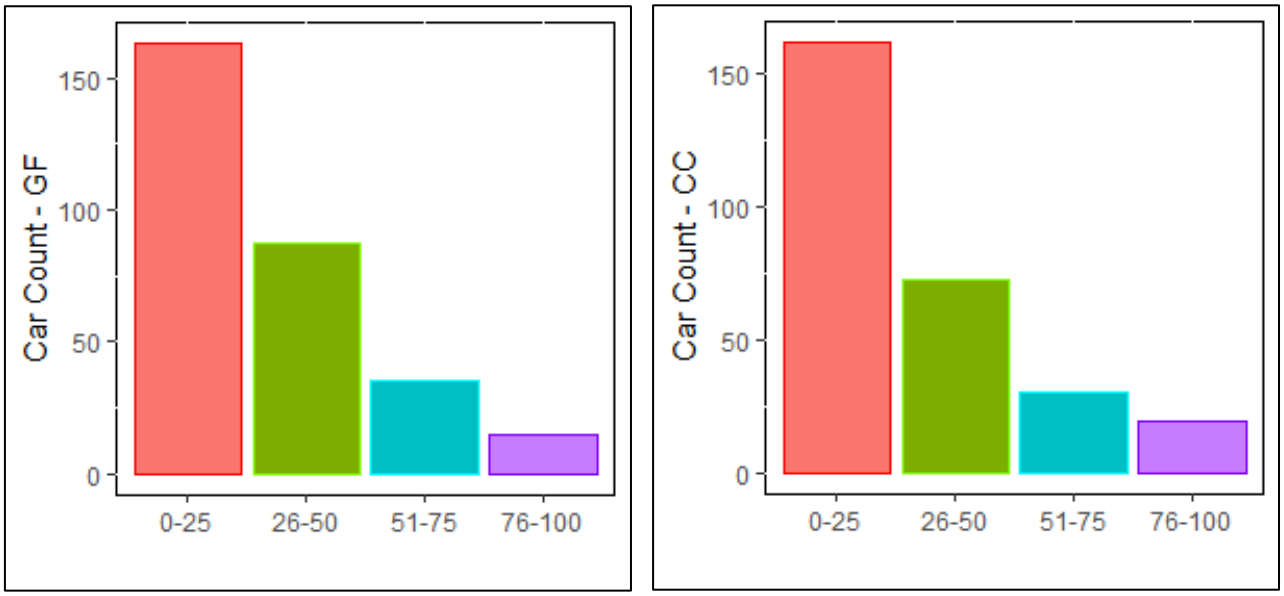

Figure 5: Car count vs bins (GF and CC)

EA Technology's objective required us to estimate the consumption of energy assuming that the charging events were unsupervised. Hence, the variable *ActiveChargingStop* was extracted from the raw data (Zheng & Casari, 2018) assuming that all the charging events were unsupervised. In principle, charging across all the trials except for trial 3 were assumed to be identical. In addition, analyses based on the assumption of unsupervised charging was to be done excluding trial 3 observations as, during trial 3, people's charging behaviour was affected because of monetary incentives given for off-peak charging. Hence, 9,650 and 4,332 observations were dropped from GF-CI and CC-CI datasets respectively. Further analyses and insights discussed in the report only involves transactions until trial 2.

4.1.1 Variability of Consumption per Transaction

From the boxplots of consumption of energy with respect to capacity range (figure 6), we observe that for both GF and CC DCS, there is significant variability of consumption per transaction (or per charging event) with capacity range. We observe that the median energy consumption is highest for consumers who own PIVs with battery capacities from 51 to 75 kWh. It is followed by consumers who own PIVs with capacities from 76 to 100 kWh. This aligns with our initial assumption that there would be the variability of frequency of charging and hence, consumption based on battery capacities.

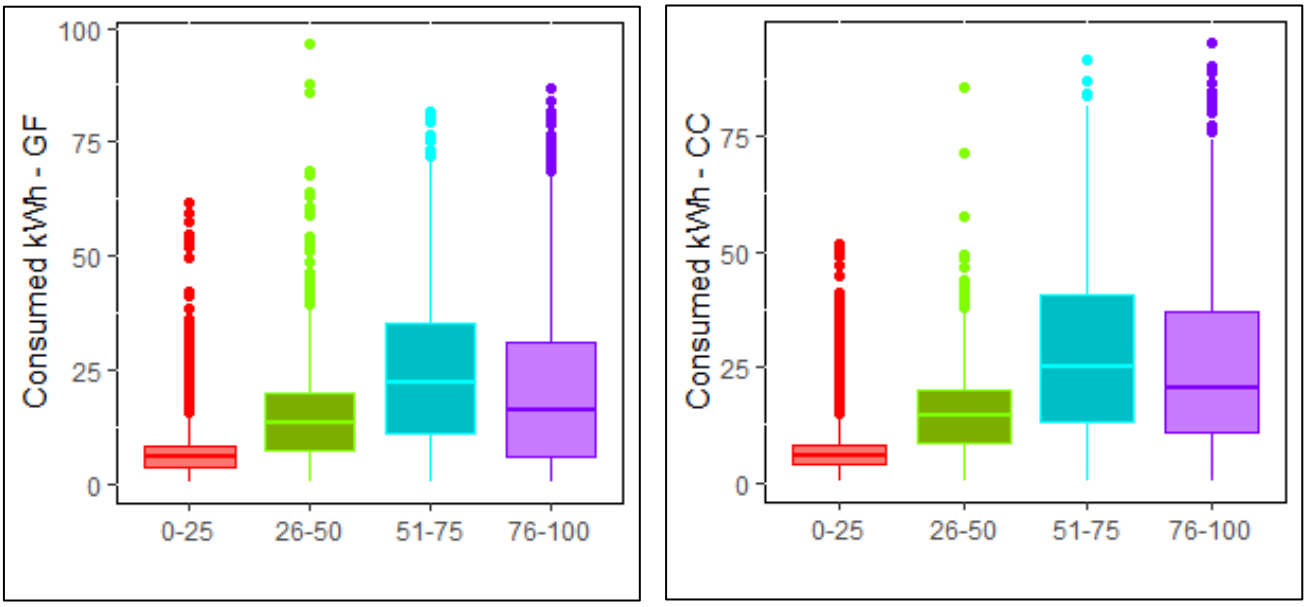

Figure 6: kWh consumption vs bins (GF and CC)

The boxplots of the count of PIVs of different car makers (figures 7 and 8) in the trials (for both GF and CC DCS) show that the top three choices for consumers are BMW, Tesla, and Nissan. These are followed by Volkswagen, Mitsubishi, and Mercedes. However, the boxplots of energy consumption with respect to car makers show that Tesla cars, in general, consume more energy than cars of other makers. Moreover, although the count of BMWs is highest among all the cars, the median energy consumption of BMWs are similar to that of many others. These observations can be attributed to the fact that while Tesla cars are found to

have higher battery capacities, for BMW cars, it is just the opposite, i.e., BMW cars are found to have lower battery capacities (figures 48 and 49, Appendix A).

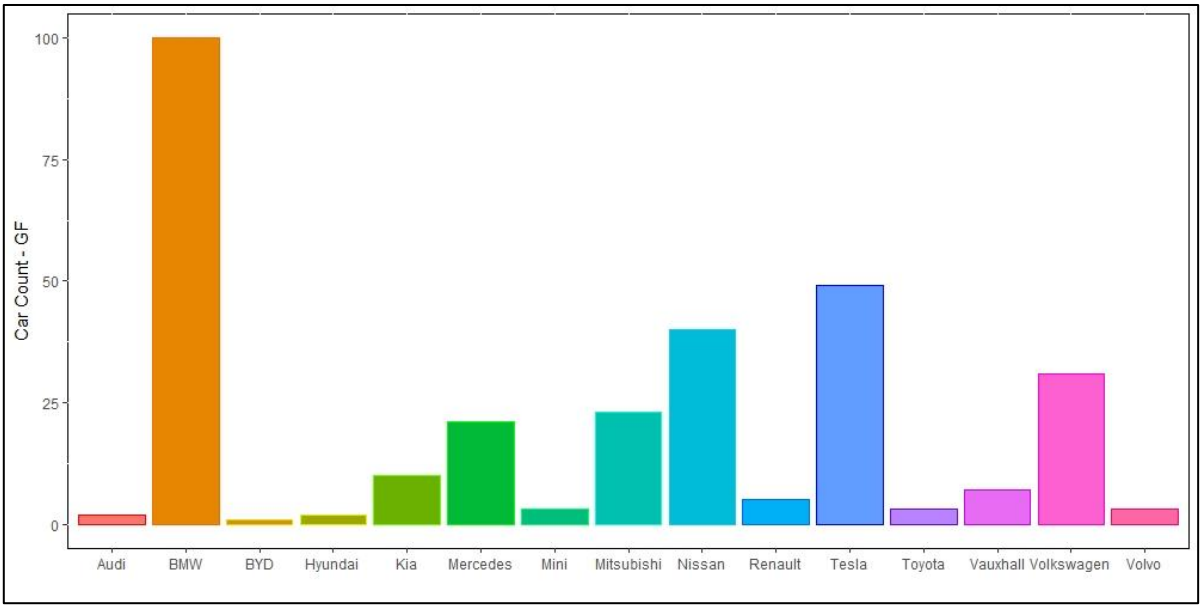

Figure 7: Car count vs car brand (GF)

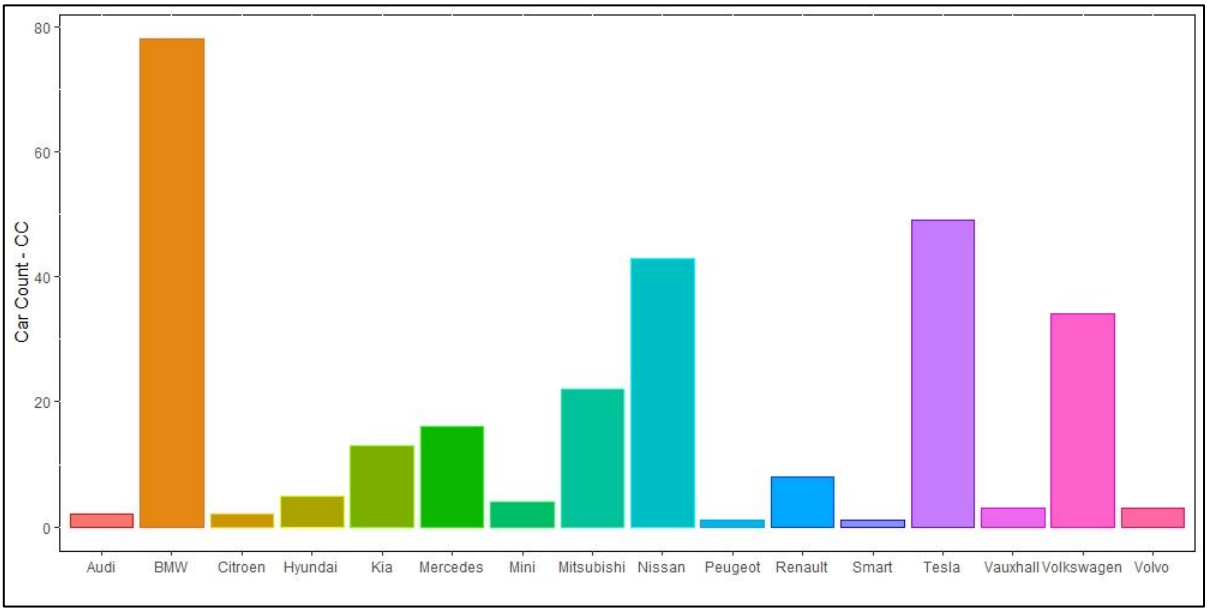

Figure 8: Car count vs car brand (CC)

Details of the median energy consumption of cars of different makers can be visualised from the boxplots (figures 9 and 10). In addition, Appendix A (figures 48 and 49) gives the boxplots for the battery capacities of different cars with respect to carmakers. Another important observation that can be drawn from the boxplots is that energy consumption gets significantly affected by the battery capacity and not by the carmaker, i.e., the effect of car make appears to be insignificant in the presence of battery capacity.

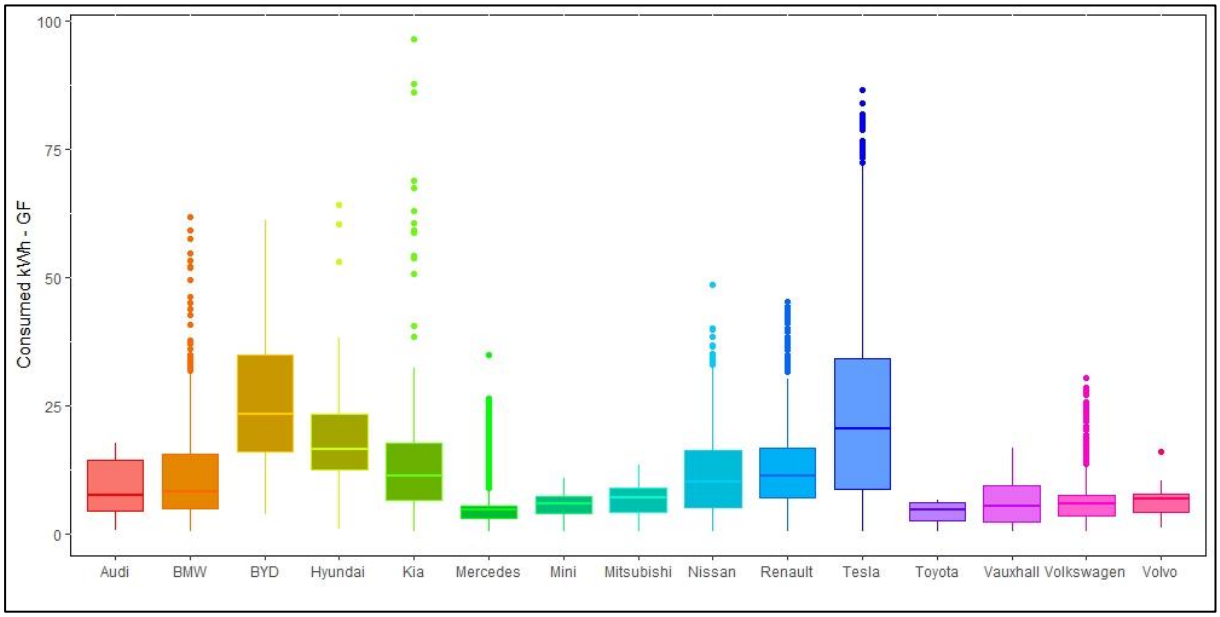

Figure 9: kWh consumption vs car brand (GF)

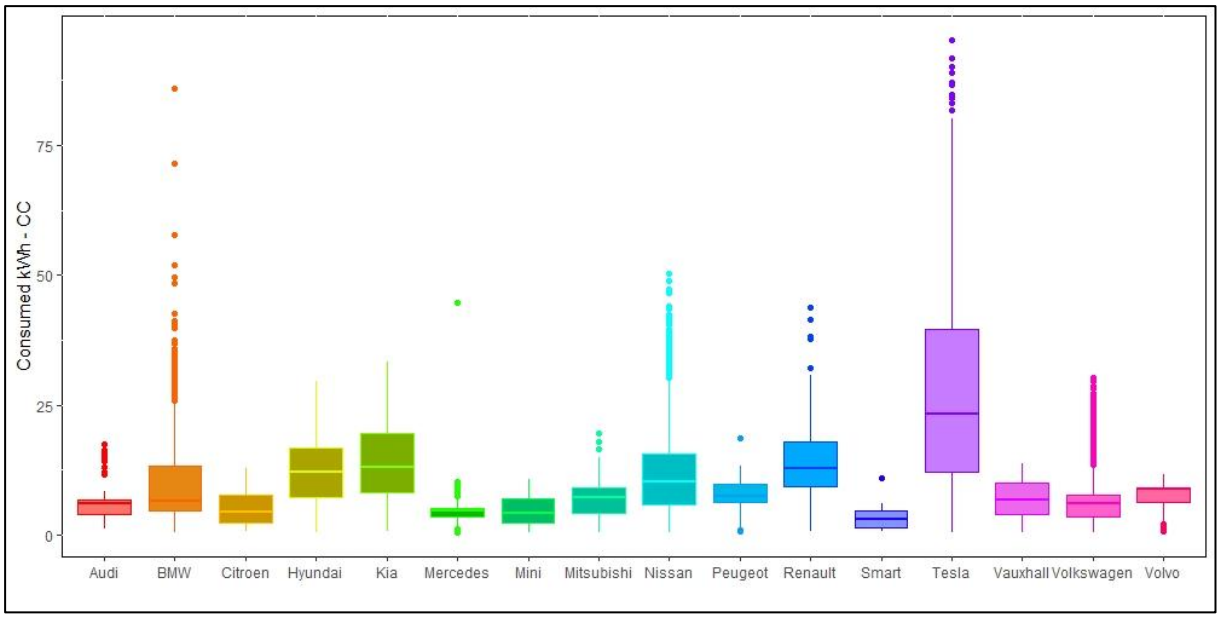

Figure 10: kWh consumption vs car brand (CC)

Boxplots of energy consumption with respect to PIV types (figure 11) show that the median energy consumption is highest for BEVs (battery electric vehicles); this is followed by REXs (range extenders) and PHEVs (plug-in hybrid electric vehicles). Although PIV type has an effect on energy consumption, the future scenario of distribution of PIV types has uncertainty. As such, EA Technology was more interested in assessing the effect of battery capacities on energy consumption. In view of the business objective, battery capacities ranged from 4.4 kWh to 100 kWh to capture a lot of variability in understanding the effect of capacities on the consumption of energy. Hence, further analyses are restricted to estimating energy consumption as a function of battery capacities and other features which would be available to DNOs (distribution network operators) in future.

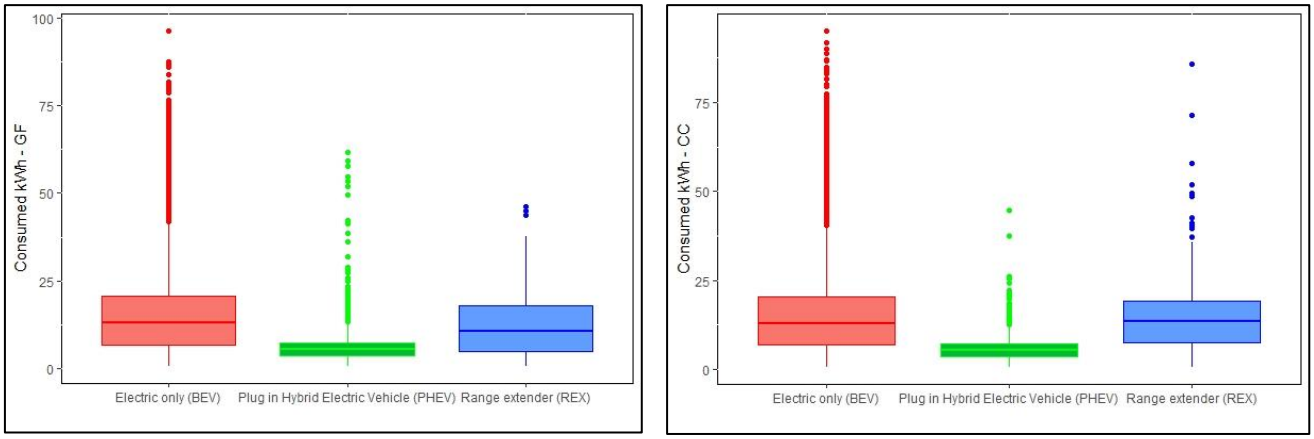

Figure 11: kWh consumption vs EV type (GF and CC)

4.2 Cluster Analysis

Although minor variations were acceptable, if trial participants within each bin showed extremely dissimilar consumption patterns, then our assumption of creating four capacity bins with equal bin width would have been seriously questioned. Cluster analysis was imperative to validate our assumption of working with four battery capacity bins to meet the business objective.

The GF transaction data was transformed to obtain consumer-wise data across all the trials except for trail 3. It is important to note that we chose only those consumers who were present in the trial for at least one day (24 hours). As such, we dropped 2 participants and summarised the data for 298 consumers. We focussed on the features that were most relevant to the future scenario: battery rating and average energy consumption per charge. It may be argued why we didn't choose the average energy consumption per day instead of the same per charge. This is because the motive behind clustering was to identify the consumption behaviour of participants every time they charge their EVs. Besides, many consumers were reported to be active for almost an entire year but charged their EVs very few times. Since we didn't know the reason for such a behaviour, we focussed on everyone's average energy

consumption per charge assuming that they used the EVs as their primary vehicles for commutation.

We started with k-means clustering (Witten, et al., 2013) for customer segmentation. A good clustering should result in small within-cluster variation (Witten, et al., 2013). The within-cluster variation for a cluster C_k is a measure that indicates how different the observations are from each other within C_k. In our analysis, we used the squared Euclidean distance to define the within-cluster variation. For cluster C_k, within-cluster variation, $W(C_k)$, is mathematically represented by (Witten, et al., 2013)

$$W(C_k) = \left(\frac{1}{|C_k|}\right) \sum_{i,i' \epsilon C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

where,

$|C_k|$ = number of observations in C_k

This entity, when summed over all the clusters, is known as the total within-cluster sum of squares. The figure shown below (figure 12) shows that as we increase the number of clusters, the total within-cluster sum of squares decreases. This result is obvious because as we increase the number of clusters (or, form smaller clusters), the within-cluster variation would decrease. However, the figure below shows an interesting observation: as we move from no cluster (k = 1) to 10 clusters, the total within-cluster sum of squares initially drops quickly from k = 1 to k = 3 but then, decreases gradually as k increases. This implies that as we increase k beyond k = 3, we don't get a significant decrease in the within-cluster variation. Hence, k = 3 is the optimum number of clusters.

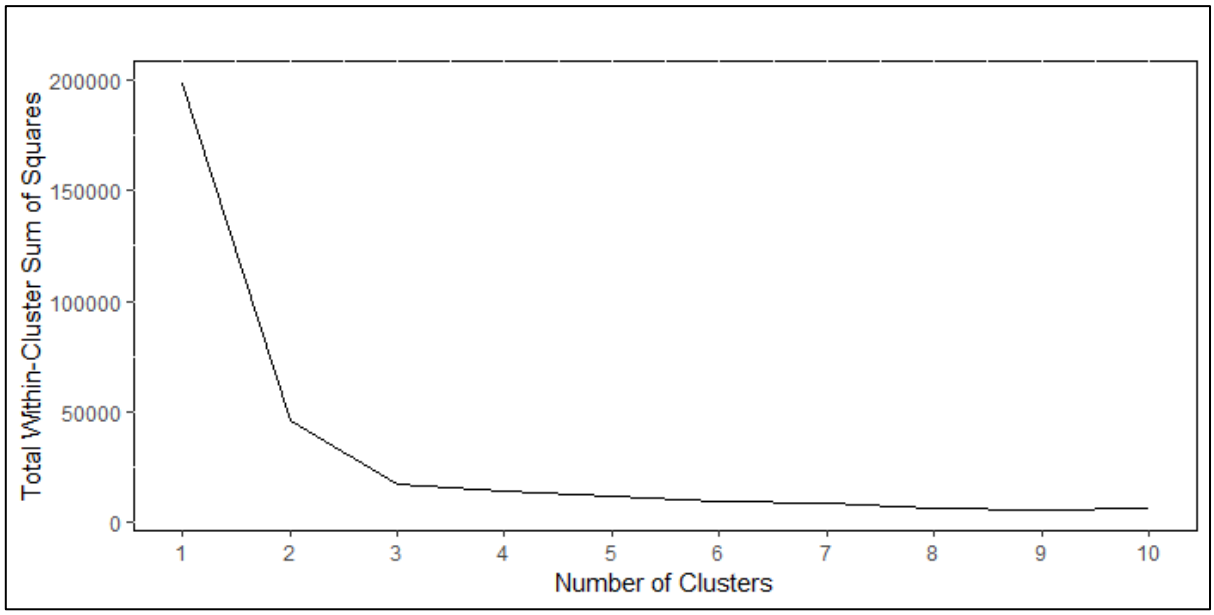

Figure 12: Total-within-cluster sum of squares vs the number of clusters

The results of k-means clustering for k = 3 are tabulated below (table 2).

Table 2: Cluster features for k = 3

Cluster	Min Rating (kWh)	Max Rating (kWh)	Mean Rating (kWh)	Mean kWh / Charge
1	4.4	18.7	9.41	5.68
2	22	41	29	14.3
3	60	100	77.8	26.8

However, we also identified the clusters for k = 4 to compare the results with those for k = 3. The results of k-means clustering for k = 4 are tabulated below (table 3).

Table 3: Cluster features for k = 4

Cluster	Min Rating (kWh)	Max Rating (kWh)	Mean Rating (kWh)	Mean kWh / Charge
1	85	100	91.3	26.4
2	4.4	18.7	9.41	5.68
3	22	41	29	14.3
4	60	75	72	27.0

We observe that, for both k = 3 and k = 4, the clusters with the two lowest mean ratings have identical mean energy consumption per charge. However, the third cluster (parent cluster) for k = 3 splits further and forms two sub-clusters when we choose k = 4. However, the mean consumption per charge differs by on 0.6 kWh for k = 4 and hence, the variability is not high in terms of mean energy consumption between the two clusters. Although the use of k = 4 doesn't help in reducing the total within-cluster sum of squares by a large margin, it is still valid and applicable, if one wishes to work with high granularity.

The suggested bins and the unique individual ratings for clustering with k = 4 are:

1. 4.4 – 18.7 kWh (4.4, 6.2, 7.1, 7.6, 8.8, 8.9, 9.2, 9.9, 11.2, 12.0, 16.0, 17.3, 18.7)
2. 22 – 41 kWh (22, 24, 27, 28, 30, 33, 40, 41)
3. 60 – 75 kWh (60, 75)
4. 85 – 100 kWh (85, 90, 100)

As mentioned earlier, EA Technology was interested in estimating the energy consumption for four ranges of battery capacities, which were equally split. The two largest battery capacity bins, 76 -100 kWh and 51 – 75 kWh, that were proposed align with two of the proposed clusters. However, the ranges of battery ratings in the clusters with two lowest mean ratings are slightly different from the proposed bins. While the proposed bins had ranges 0 – 25 kWh and 26 – 50 kWh, the clusters have ranges less than 20 kWh and greater than 20 kWh but less than or equal to 50 kWh.

The motivation behind k-means clustering was not to identify the exact capacity bins but to validate the approach. While results from k-means clustering suggested the use of three clusters, it didn't invalidate the use of four clusters but indicated that addition of a fourth cluster didn't offer any advantage relevant to within-cluster variation of consumer's energy consumption. However, the usage of four bins did allow for one advantage that couldn't be captured by the variance in the data: the addition of EVs with new battery capacities might change the range of observations in the clusters, including the number of optimal clusters based on the k-means algorithm. Hence, drawing a hard line on the range of observations within a cluster as well as the number of clusters was not recommended as this would have put constraints on the future scenarios with little room for flexibility. Hence, we adopted a more generalised binning approach to address the business objective, and approached the objective using the pre-agreed four bins of battery capacities.

Summarily, k-means clustering helped us to validate our approach of working with different battery capacity bins and develop predictive models for different bins as it helps in capturing variations at higher granularity.

4.3 Time Series Analysis

The transaction data in its original form, even after initial pre-processing, still didn't deliver the information that we needed to address the business objective. Hence, we transformed the transaction data into the day-wise dataset. The day-wise data for GF and CC were created to understand the overall maximum connected load (sum of the battery ratings in kWh for all the charging events in a given day) and consumption of energy per day for consumers across all the battery capacity bins created, i.e. we took into account variability of consumption as a function of battery capacities.

The day-wise data had nine variables which were extracted (Zheng & Casari, 2018) based on the information available in GF-CI and CC-CI datasets. The description of the variables can be found below:

1. *date*: Date of charging events.
2. *day*: Day of the week corresponding to a given date.
3. *season*: The annual season of the year based on the month (Elexon, n.d.). This agreed with the client's objective.
4. *demand*: Maximum connected load in kWh (as explained earlier) in each day; it was estimated by aggregating battery capacities for all the transactions in each day. It followed from the observation that in more than 98% of the days, number of charging events taking place per day was more than the number of consumers, thereby meaning that consumers tend to charge their EVs more than once per day. Hence, every time a consumer connected his/her EV for charging, we considered the battery capacity as his/her demand. For instance, if a consumer with a battery capacity of 4.4 kWh connected his/her EV twice in each day, we estimated the *demand* as 8.8 kWh. Such an estimation was necessary as we didn't have information on the state of the charge of an EV when it was connected. Demand puts and upper cap on the consumption of energy as the latter will always be less than or equal to the former.
5. *consumed*: Total consumed energy (kWh) in each day; it was obtained by aggregating the energy consumption data for all transactions in a given day.
6. *time*: Total duration of charging (hours) per day.
7. *owners*: Number of EV owners per day. The original data had no information on the number of EV owners per day. This information was vital as not all the people who would own EVs would charge their vehicles every day. From the original data, we could directly extract information on the number of people charging their EVs every day, but the information on EV owners per day could not be extracted without any prior assumptions. Besides, people gradually joined the trials from March 2017 onwards until the trials got over. Moreover, it would be the number of EV owners that would be known to the DNOs in future, and a sophisticated model would be required to estimate the number of users

who would actually charge their EVs. Hence, we needed to estimate EV owners so that we could estimate the number of EV owners, called users, who would actually charge their vehicles per day. To estimate the EV owners per day, we assumed that an EV owner joined the trials whenever he/she charged his/her EV for the first time; besides, we also assumed that the consumer never dropped from the trials until the trials were active. Under this assumption, the EV owners increased with time during the trials.

8. *users*: Number of EV users per day.
9. *trans*: Number of transactions per day.

4.3.1 The Influential Tail

The day-wise data obtained had missing dates between the first and last days of transactions. The missing days resulted out of lack of information in the original data. There were two possibilities: no charging took place on a given day, resulting in no data, or the charging events for the missing days were not recorded because of communications loss between chargers and DCS. The first possibility was highly unlikely and hence, interpolation of missing data was essential for further analyses. Since the data was actually a time series, to interpolate missing information, we first computed STL decomposition, which was followed by linear interpolation after seasonally adjusting the data; finally, the seasonal component was added back (Hyndman & Athanasopoulos, 2018). The methodology used for interpolation was found to render the best fit for all the univariate time series in the multivariate data and hence, helped in retaining the structure of the time series. The details of the number of observations (number of days for which charging data is available) in the datasets can be found below (table 4).

Table 4: Number of observations vs bins (GF and CC)

Battery Capacity Bin (kWh)	GF (Number of Days)	CC (Number of Days)
0-25	590	653
26-50	611	629
51-75	613	599
76-100	473	610

The difference in the number of observations arises because of the different start and end dates of the transactions for the DCS providers. From the above table, we observe that, in the majority of the cases (75% of the cases), CC datasets have a larger number of observations.

The time plots of owners vs users for both GF and CC are shown below in figure 13 (0-25 kWh data). While the time plot for GF shows sudden level shifts at certain times, the plot for CC is relatively smoother. This can be attributed to the fact that consumers were gradually added to the CC DCS provider; however, for the GF DCS, batches of consumers were added at specific time instants.

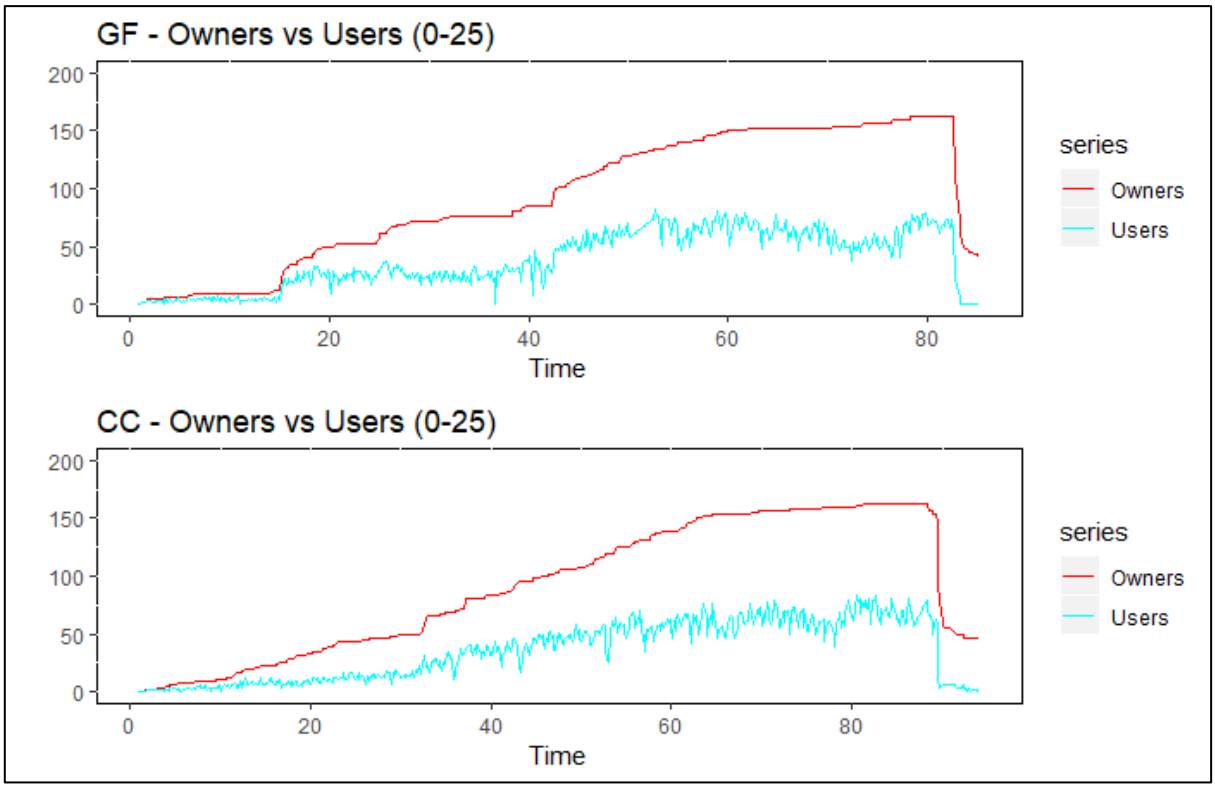

Figure 13: Owners vs users (GF and CC)

We also observe a sudden drop in owners as well as users in the time plots. Similar effects can also be observed for time plots of demand vs consumption (figure 14). The sudden drop in values can be attributed to the migration of consumers to trial 3. Although the data used for analysis didn't involve observations from trial 3, the effect of migration of consumers could be felt even before trial 3 actually began. Such observations would have adversely affected the insights and intelligence drawn from data as the drop in the number of users was not proportional to the drop in the number of owners, and so was the case with demand and consumption.

Observations in the tail of the plots, if retained, would have strongly affected the learning of forecasting algorithms as they were highly influential. Hence, for each time series based on battery capacity bins, such influential observations, which exhibited a disproportional drop in values across different variables, were dropped from further analysis. It is worth mentioning

here that most of such observations in the tail were imputed by the interpolation algorithm based on the neighbouring values and trend of the time series. As such, it was safe to drop such values as any algorithm, if used for interpolation, would have yielded similar results.

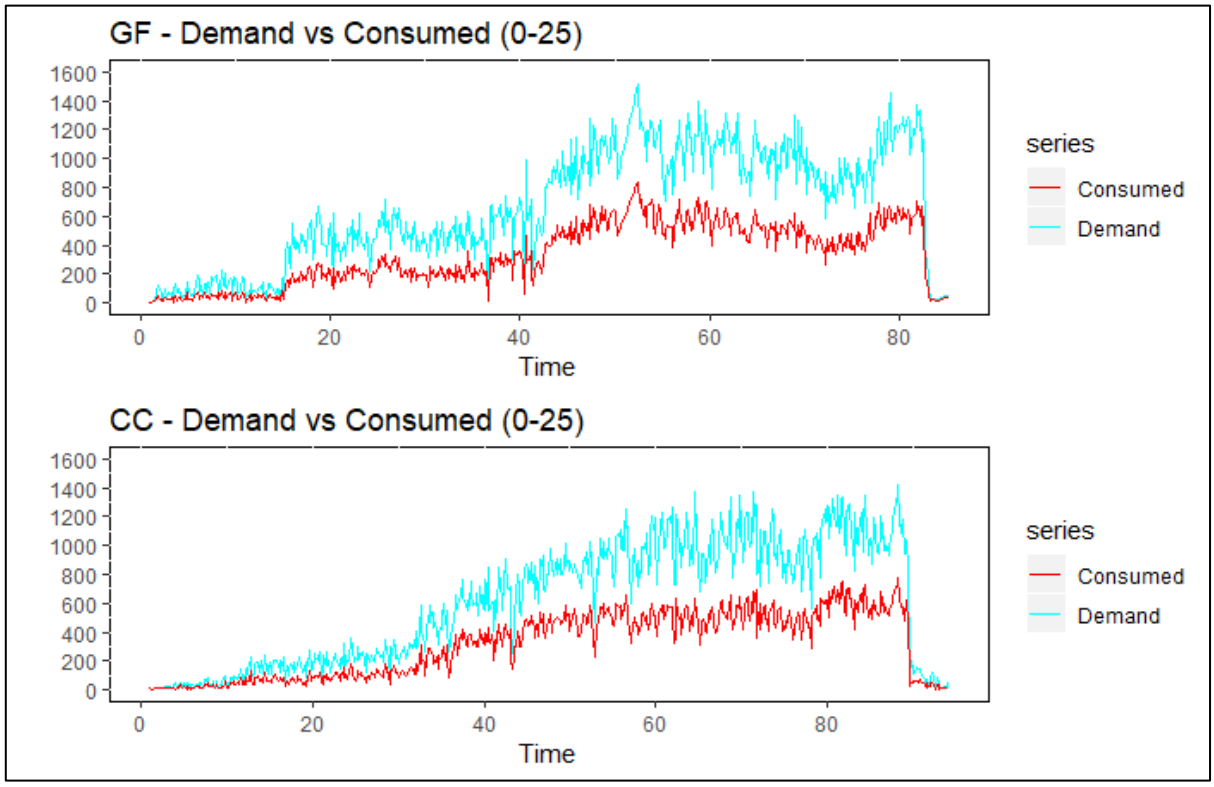

Figure 14: Demand vs consumption (GF and CC)

4.3.2 Towards Cleaner Data

The time plots for owners vs users and for demand vs consumption, after dropping the influential observations in the tail, are shown below (figure 15) for the GF dataset (plots for CC data are shown in figure 50, Appendix A).

We observe that, for both GF and CC data, a greater number of users charge every day in the 0-25 kWh bin than users in any other bin. Although this does follow from the fact that 0-25 kWh bin had the highest number of registered consumers in the trials and hence, higher

proportion of them charging, another possibility had been that consumers with smaller battery capacities needed to frequently charge their EVs than consumers with higher battery capacities, to keep their vehicles charged for usage.

However, the demand and consumption plots (figure 16) reveal that for GF data (plots for CC data are in figure 51, Appendix A), consumers seem to have similar consumption patterns for the two lowest battery capacity bins (0-25 and 26-50 kWh). Summary statistics for the GF data, as shown in tables 5 to 8 (summary statistics for the CC data can be found in tables 43 to 46, Appendix B), reveal a very interesting observation: while the mean number of users and transactions for 0-25 kWh bin is approximately double the mean values of the respective variables for 26-50 kWh bin, the mean consumption for 0-25 kWh bin is nearly the same as that of 26-50 kWh bin. However, the mean demand for 26-50 kWh bin is higher than that for the 0-25 kWh bin. We can infer that although the mean number of users and transactions for the 26-50 kWh bin is significantly smaller than those of the 0-25 kWh bin, demand and consumption per user (or per transaction) is higher for the former category than the latter. For bins with higher capacities, the number of users and transactions is extremely small compared to the values of the variables for lower capacity bins. Besides, the mean demand and consumption are also smaller than those for lower capacity bins.

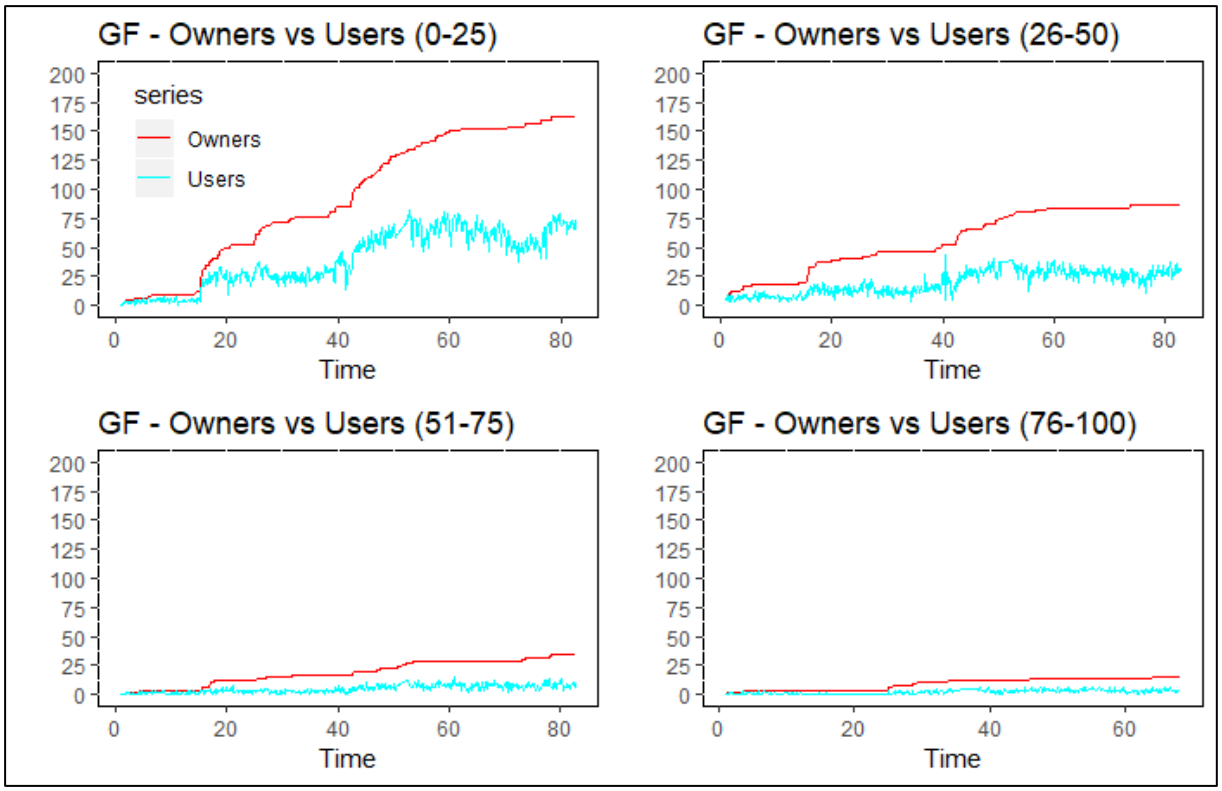

Figure 15: Owners vs users for all bins (GF)

Table 5: Summary statistics for 0-25 kWh bin (GF)

Statistics – GF – 0/25	Demand	Consumed	Time	Owners	Users	Trans
Minimum	8.8	3.8	0.54	1	1	1
1st Quartile	395.4	179.5	43.58	52	22	30
Median	692.4	341.9	82.85	85	37.5	52
Mean	685.1	344.3	82.57	93.45	39.83	53.27
3rd Quartile	1031.1	529.7	126.56	152	63	82.05
Maximum	1516.2	838.1	202.99	163	82	119

Table 6: Summary statistics for 26-50 kWh bin (GF)

Statistics – GF – 26/50	Demand	Consumed	Time	Owners	Users	Trans
Minimum	93	29.5	4.21	3	3	3
1st Quartile	447	172.6	28.43	39	11	14
Median	755	320.2	52.52	53	19	23
Mean	778.6	340.4	55.97	57.06	19.89	24.23
3rd Quartile	1081	490.4	80.05	84	29	34
Maximum	1961	945	154.75	87	40	61

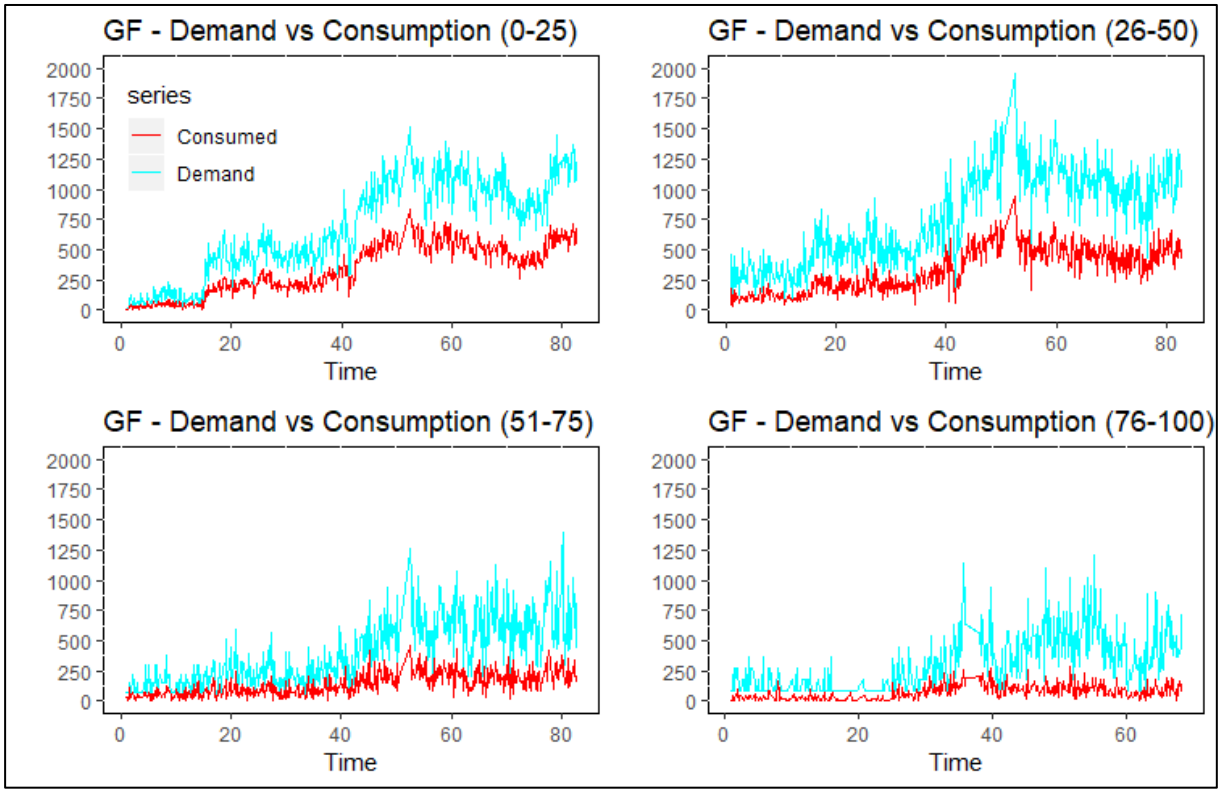

Figure 16: Demand vs consumption for all bins (GF)

Table 7: Summary statistics for 51-75 kWh bin (GF)

Statistics – GF – 51/75	Demand	Consumed	Time	Owners	Users	Trans
Minimum	60	2.4	0.34	1	1	1
1st Quartile	180	66.3	9.47	13	2	3
Median	345	119	17	17	4	5
Mean	411.2	142.1	20.31	19.16	5.02	5.83
3rd Quartile	600	204.9	29.27	29	8	8
Maximum	1395	456.8	65.26	35	15	19

Table 8: Summary statistics for 76-100 kWh bin (GF)

Statistics – GF – 76/100	Demand	Consumed	Time	Owners	Users	Trans
Minimum	85	1	0.14	1	1	1
1st Quartile	90	25.38	3.63	4	1	1
Median	270	58.9	8.41	12	2	3
Mean	318.2	75.23	10.75	9.61	2.62	3.47
3rd Quartile	460	112.45	16.06	14	4	5
Maximum	1210	285.5	40.79	15	7	13

4.3.3 The Story of Unreliable Data

Comparison of summary statistics reveals an interesting observation: the mean values of all the variables of GF data was higher than those of CC data (for 0-25 kWh datasets). A similar scenario exists for the 26-50 kWh bin too. Summary statistics of CC data can be found in Appendix B (tables 43 to 46). This shouldn't be the case as, per EA Technology, consumers were randomly assigned to both the DCS providers with no bias and preferences and hence,

the mean charging patterns across all the variables for both the DCS providers should be similar, if not identical, as long as the variability within a capacity bin is not high. Moreover, the demand and consumption plots for CC data reveal that, unlike GF data, the charging patterns for 0-25 and 26-50 kWh bins were not similar. Summary statistics for the 0-25 and 26-50 kWh bins also validate this observation. Furthermore, after initial pre-processing of data, we had data for 300 consumers for GF and 284 consumers for CC. This also validated the hypothesis that, in principle, consumers should show similar charging behaviour unless allotment of consumers between both the DCS providers involved any bias, such as consumers with lower battery capacities were allotted to CC, while those with higher battery capacities went for GF. However, such was not the case, as can be observed from the table below (table 9).

Table 9: Number of consumers vs bins (GF and CC)

Battery Capacity Bin (kWh)	GF – Consumers	CC – Consumers
0-25	163	162
26-50	87	73
51-75	35	30
76-100	15	19

Based on our observations, we suspected that either one or both the DCS providers didn't capture all the information correctly relevant to the charging of EVs. When the issue was raised with EA Technology, they informed us that unlike GF DCS, CC DCS didn't maintain a record of data if there was a communication loss between the charger and CC DCS, i.e., if a charger lost communication with CC DCS, CC would never have a record of the transaction. However, GF always recorded transactions even if there was loss of communication between charger and DCS; this obviously happened when the communication was re-established. In fact, EA Technology furthered that CC data was unreliable for the reasons specified above. To summarise, the GF dataset was completer and more reliable than the CC dataset. Having

identified the unreliability of CC data, we decided to work only with GF data in all our analyses that followed.

4.3.4 The Outlier Conundrum

The number of owners was estimated for both the datasets as explained earlier. However, for other univariate time series in the datasets, inspection for the presence of outliers was warranted to ensure reliable statistical modelling. Identification of outliers was carried out by a blend of graphical visualisation and an automated algorithm (Hyndman & Athanasopoulos, 2018) that identifies outliers and suggests reasonable replacements by identifying residuals via a periodic STL decomposition for seasonal data; residuals are labelled as outliers if they lie outside the range given by the following:

$$\pm 2(q_{0.9} - q_{0.1})$$

where, q_p= p-quantile of the residuals.

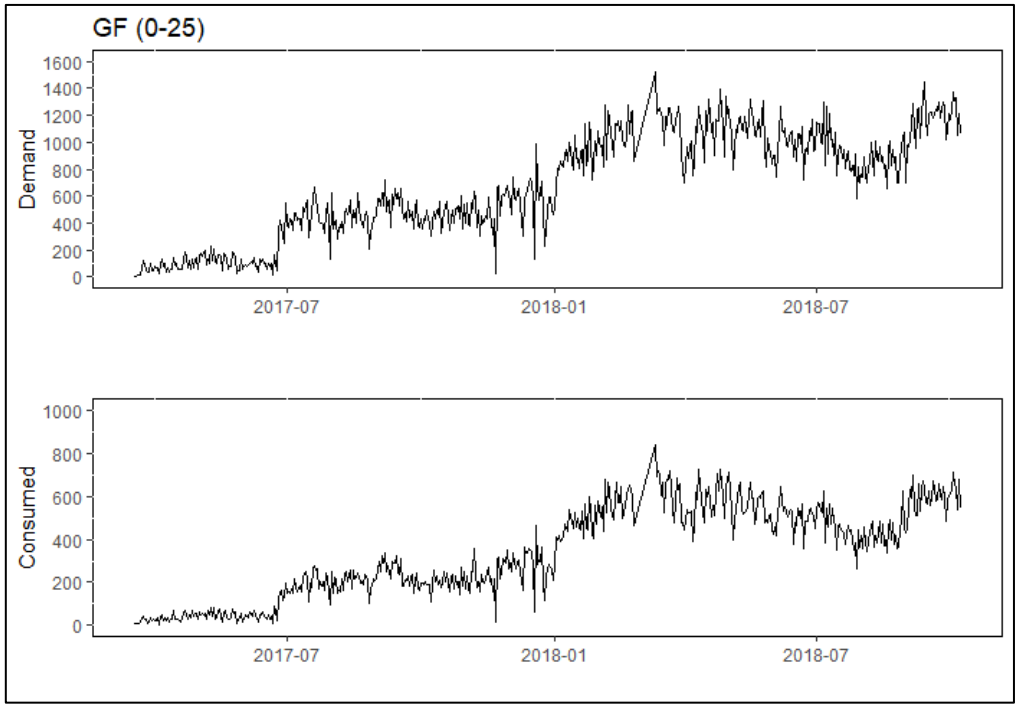

Figure 17: Demand vs consumption for 0-25 kWh (GF)

From the time plots of demand and consumption for 0-25 kWh bin, as shown in figure 17 (time plots for other bins are shown in figures 52 to 54, Appendix A), we observe a lot of variabilities as we move ahead in time. However, at certain instants we observe significant drops in the values compared to nearby values; a closer inspection would reveal that values of demand and consumption abnormally drop to extremely lower values towards the end of 2017 at three different instants. Similar results also hold for variables users, transactions and time. It is worth mentioning here that while automated algorithms usually work pretty well for lots of datasets, there may be cases that might cause problems. Hence, we compared the findings from automated algorithm with those from graphical visuals so that identification of outliers became more reliable.

However, it is worth mentioning here that not all the values were replaced with the suggestions of the algorithm by treating them as outliers; further graphical analysis was carried out to validate the findings and only those values were replaced that were identified as outliers via both algorithm and graphical analyses.

The table below summarises the number of observations in each of the GF dataset that were identified as outliers by the automated algorithm.

Table 10: Number of outliers identified by algorithm (GF)

Variable	0-25 kWh	26-50 kWh	51-75 kWh	76-100 kWh
Users	3	1	3	4
Trans	4	1	1	8
Time	3	1	0	1
Demand	3	1	1	10
Consumed	1	1	0	1

The following table shows the actual number of observations that were treated as outliers and were replaced with reasonable replacements.

Table 11: Number of observations replaced as outliers (GF)

Variable	0-25 kWh	26-50 kWh	51-75 kWh	76-100 kWh
Users	2	2	3	0
Trans	2	2	3	0
Time	2	2	3	0
demand	2	2	3	0
consumed	2	2	3	0

It might be surprising to see that for the 76-100 kWh bin, not a single observation was replaced. This was against what the algorithm suggested. We didn't replace any observation in the 76-100 kWh bin because the charging behaviour of people and hence, the estimated values of certain variables such as *trans* and *demand* didn't agree with the findings of the algorithm: we identified that transactions and demand (per day) seldom reached high values as EVs with higher battery capacities had lower charging frequencies. Hence, based on our understanding of the data, we didn't replace any observation in the 76-100 kWh bin. Similar reasons were also applicable for explaining the difference between the estimated number of outliers by the automated algorithm and the actual number of observations that were replaced.

4.3.5 Variability of Consumption (Day-Wise and Season-Wise)

We assumed that demand and consumption of energy would be the same neither across all the days of a week nor across all the seasons in a year. The boxplots shown below (figures 18 and 19) show the variability of demand and consumption across the days of a week and seasons in a year (0-25 kWh bin; plots for other bins can be found in figures 55 to 60, Appendix A).

We observe that the median demand and consumption are lowest for Sunday and respectively close to 600 kWh and 250 kWh. The median demands on all the other days are nearly the

same and approximately equal to 700 kWh. While both Thursday and Friday have their top 25 percent of the observations (demand) above 1100 kWh approximately, Monday, Tuesday and Wednesday have their top 25 percent of demand above 1000 kWh approximately.

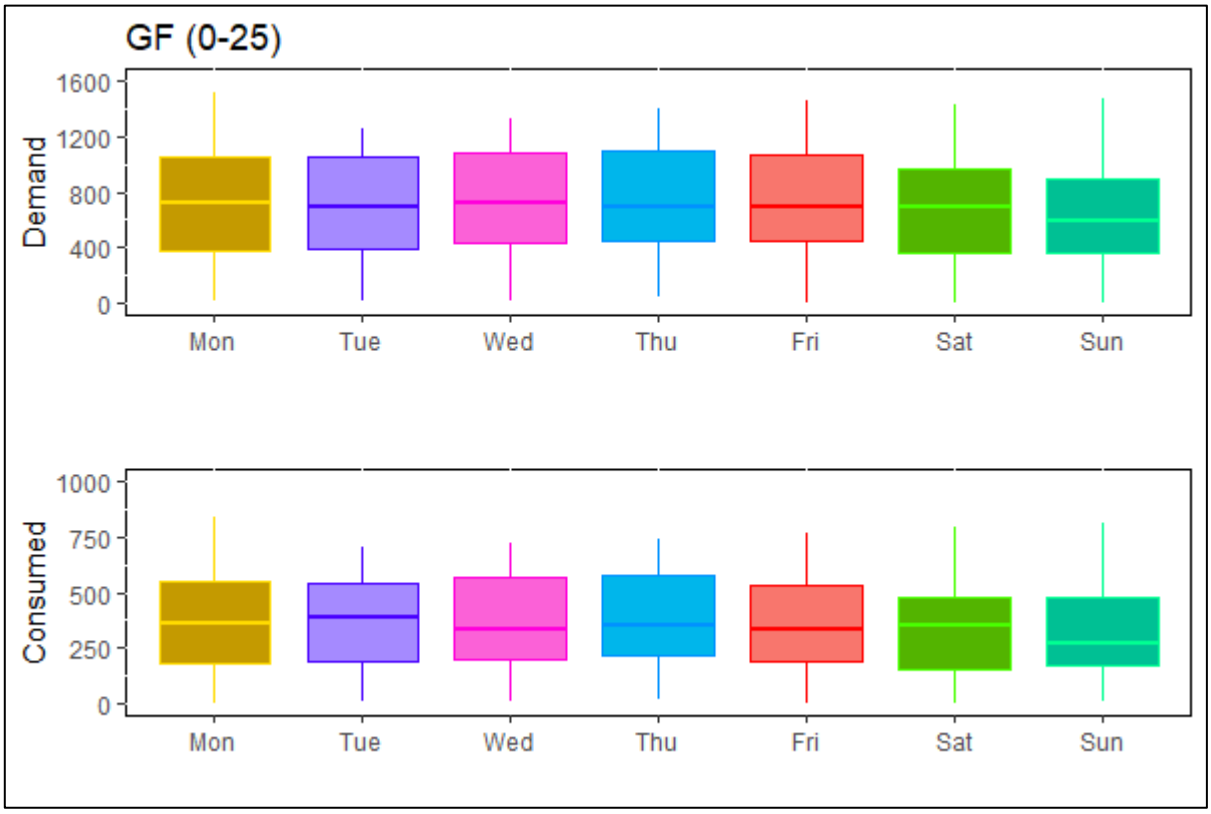

Figure 18: Demand and consumption vs day of the week for 0-25 kWh bin (GF)

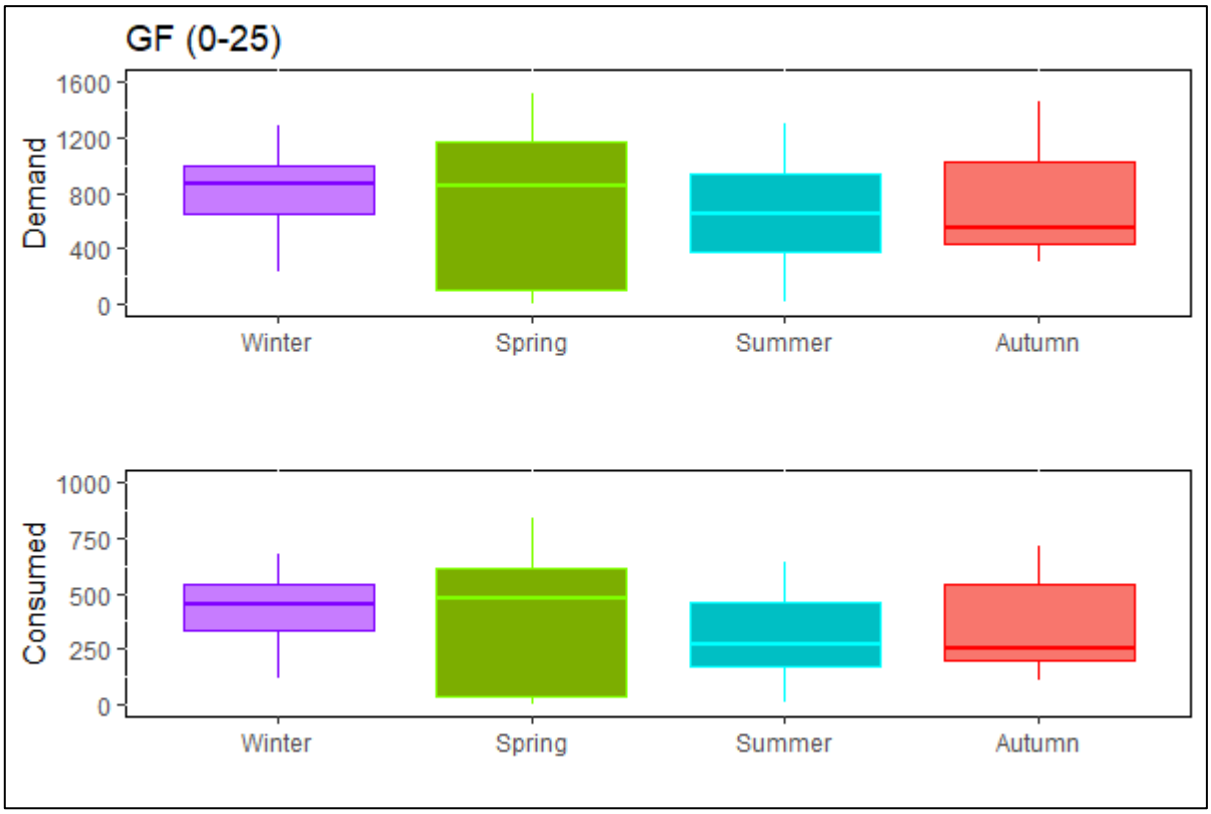

Figure 19: Demand and consumption vs season of the year for 0-25 kWh bin (GF)

The median consumption on Monday and Tuesday (approximately 375 kWh) are found to be highest among all the days. For both Saturday and Sunday, the bottom 75 percent of the observations are below 500 kWh, i.e., consumption doesn't even reach 500 kWh for the bottom 75 percent of the values. In other words, the top 25 percent of the values start below 500 kWh on weekends. However, for other days, the top 25 percent consumption is above 500 kWh. Summarily, consumption on weekends is expected to be lowest than those on other days.

While the median demand and consumption are highest during winter and spring and are approximately equal to 800 kWh and 500 kWh respectively, they are nearly the same during summer and autumn (median demand and consumption are nearly 600 kWh and 250 kWh respectively). Summer and winter are two extreme weather conditions and spring and autumn lie at their interface, i.e., while the transition from summer to winter experiences autumn, the

transition from winter to summer experiences spring. Hence, we observe a lot of variability in demand and consumption during autumn and spring. Summarily, demand and consumption are expected to be higher in spring and winter, with a lot of variabilities expected during spring.

4.3.6 Correlations

The multiple scatterplots shown below (figure 20) highlight the correlation among all the numeric variables in the GF dataset for the 0-25 kWh bin (scatterplots for other bins can be found in figures 61 to 63, Appendix A). All the variables show strong correlations (correlation coefficient > 0.9) among each other.

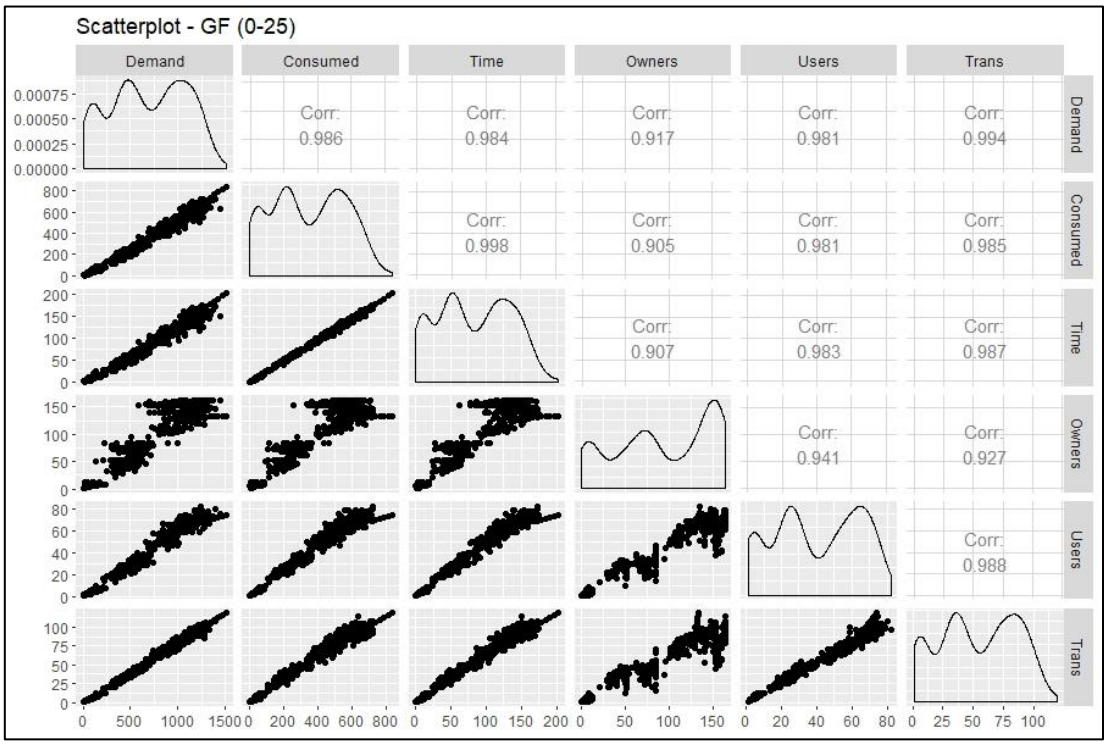

Figure 20: Correlations among numeric variables for 0-25 kWh bin (GF)

Variables (predictors) which have high correlations among each other would have relatively large standard errors in partial regression coefficients. This makes the partial regression coefficients unstable, thereby leading to varying coefficients across different samples of data

and eventually unreliable forecasts. Since all the variables were strongly correlated among each other, careful consideration was given during variable selection during the development phase of forecasting models as too many variables would have posed problems of multicollinearity and overfitting.

4.3.7 Time Series Decomposition

Time series decomposition deconstructs a time series into its various components, each representing one of the underlying categories of patterns. We split a time series into three components: trend-cycle, seasonal and a remainder component. The trend-cycle component indicates how the time series progresses (sloping upward, sloping downward or no slope) with time, including any cyclic patterns for smaller time periods. The seasonal component indicates the periodic variations in the time series, which occurs after a given fixed interval of time. For instance, a daily sales time series may show peak sales on every Sunday for all the weeks, while relatively lower sales on other days; the pattern of sales for all weeks would be similar, if not same. We call this type of seasonality as weekly as it repeats every week. After filtering out the trend-cycle and seasonal components, we get the remainder component. Time series decomposition not only helps to understand the time series better but also can be used to improve the forecast accuracy.

The plots shown (figure 21) below give the details of the components of the time series for GF data (0-25 kWh); plots of other bins are shown in figures 64 to 66, Appendix A. Since we converted the raw transaction data into day-wise time series after data cleaning (Wickham, 2014) and we had data for less than two years, we only opted for weekly seasonality and excluded annual seasonality as we didn't have two full periods of annual data. The plots are shown for demand and consumption of energy; each plot shows four subplots: original time series, trend, seasonal and remainder components.

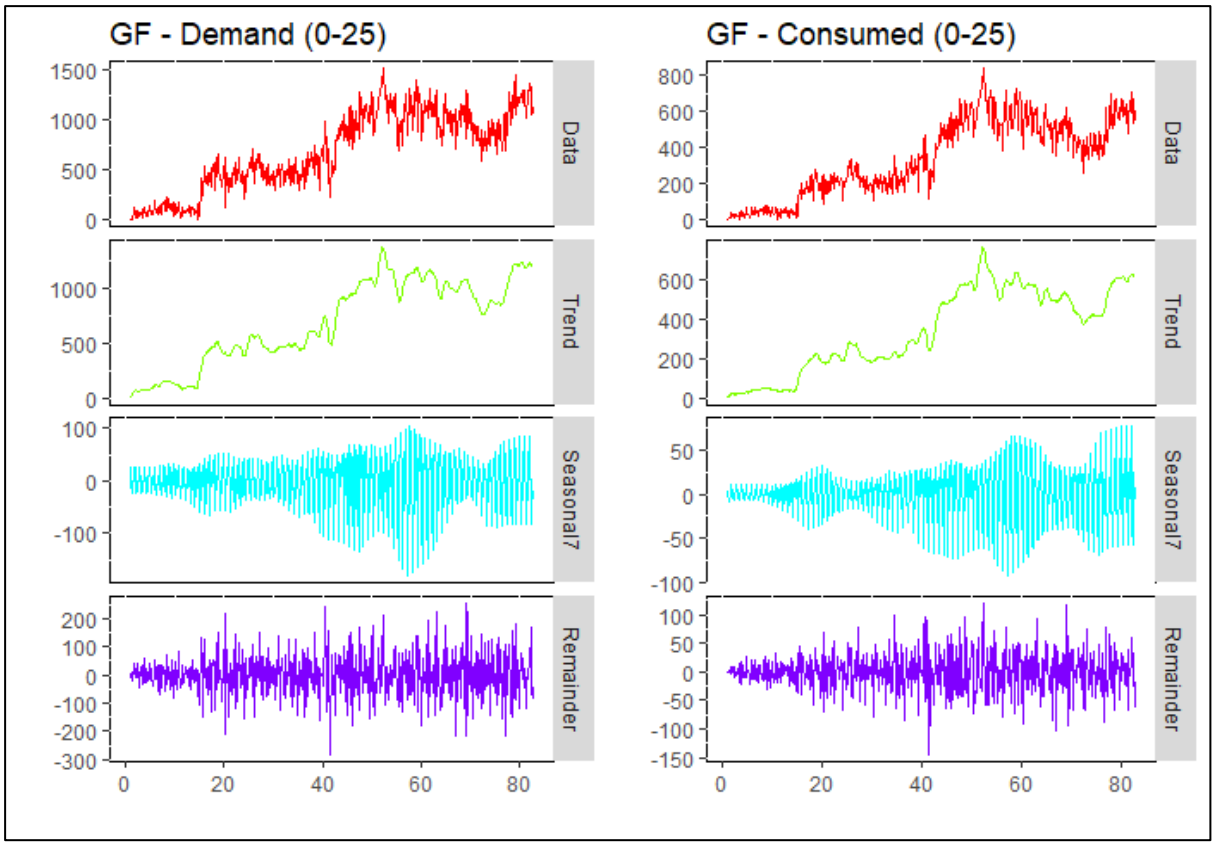

Figure 21: Time series decomposition for 0-25 kWh bin (GF)

The decomposition was carried out using STL (Cleveland, et al., 1990). STL is an acronym for "Seasonal and Trend decomposition using Loess". STL offers several advantages over other decomposition methods (Hyndman & Athanasopoulos, 2018). Some of them are discussed below.

- STL handles any type of seasonality.
- The seasonal component is allowed to change over time, with the provision of the rate of change being allowed to be controlled by the user.
- The smoothness of trend-cycle is also user-controllable.
- It can be robust to outliers, provide the user specifies for a robust STL decomposition.

We observe a clear upward trend for both demand and consumption; the upward trend is caused by an increasing number of users charging their EVs as time progresses. Both time series also show strong weekly seasonal components having both troughs and crests, with the

troughs corresponding to weekends when both demand and consumption are on the lower side. The variation in seasonal components is in agreement with the inferences drawn from the boxplots of demand and consumption with respect to days of the week. The seasonal components also experience variation in magnitude with time; this can be attributed to the fact that variability of demand and consumption increase with an increasing number of EV owners and therefore, users. When the number of EV owners and therefore, the users are at their peak, demand and consumption are at their maximum too, but with higher variations.

4.3.8 Autocorrelations and Partial Autocorrelations

In this section, we discuss the time series dynamics of correlations of the target variable that also play an important role in determing the value of the target variable; this is because such dynamics contain information that need to be extracted to enhance the predictive accuracy of the models. Since predictive accuracy of the model is vital to the objective, discussion on autocorrelations are essential and necessary.

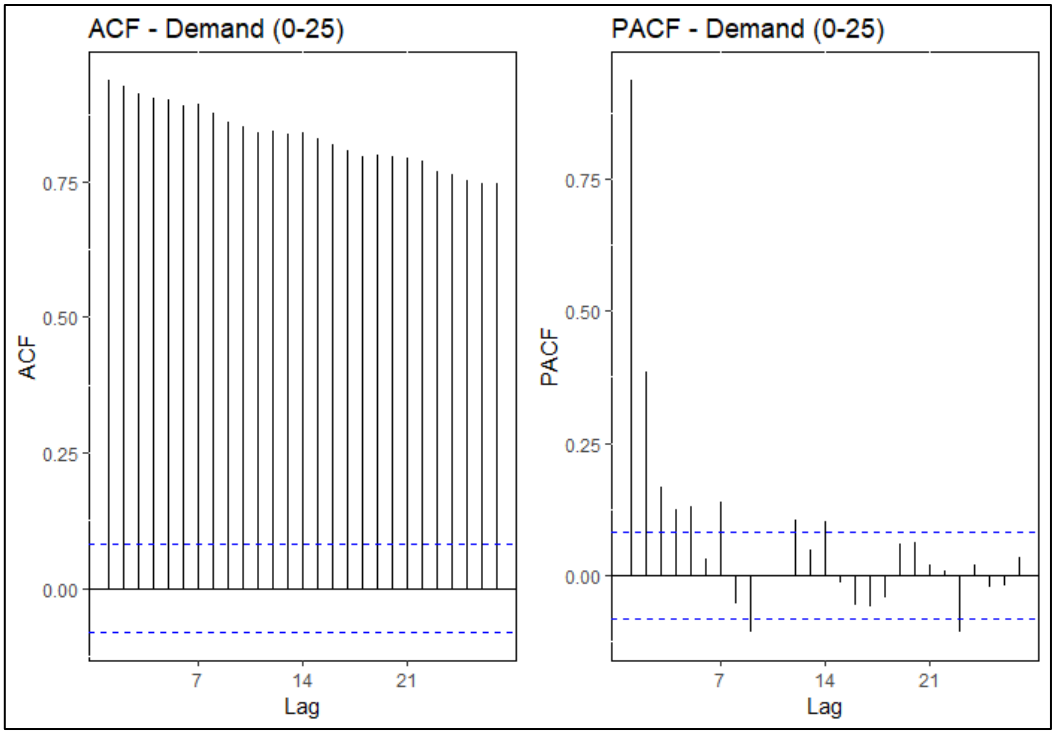

Figure 22: ACF and PACF of demand for 0-25 kWh bin (GF)

The autocorrelations and partial autocorrelations for the demand and consumption of 0-25 kWh bin are shown above via the ACF and PACF plots in figures 22 and 23 (ACF and PACF for other bins are shown in figures 67 to 72, Appendix A).

ACF (autocorrelation function) shows the correlations among lagged variables of a time series. We observe that for both demand and consumption, there is a slow decrease in the values of ACF. This means that the time series has a trend. However, a closer inspection would also show that at lags 7, 14, 21 and so on, there are spikes, i.e., ACF values at these lags are higher than the neighbouring values. This happens because of the presence of weekly seasonality in time series. The presence of both trend and seasonality gives a "scalloped" shape to the ACF plots. The dashed blue lines are an indication of whether the autocorrelations are significantly different from zero. If the ACF lies within the blue lines, then it means that the time series is white noise as the time series shows no autocorrelation.

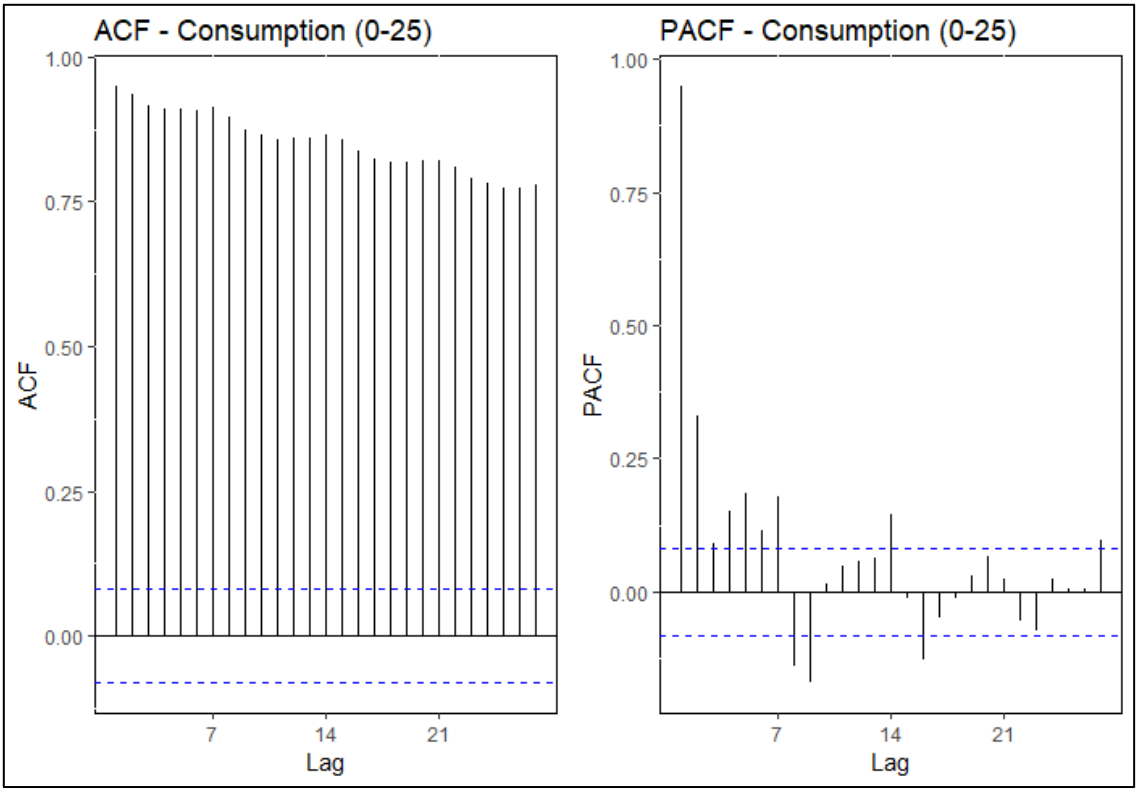

Figure 23: ACF and PACF of consumption for 0-25 kWh bin (GF)

ACF plot measures the correlation between y_t and y_{t-l} for different values of l, where l is the lag. However, if y_t and y_{t-l} are correlated, and y_{t-l} and y_{t-m} are correlated, then y_t and y_{t-m} will also be correlated as both of them have a correlation with y_{t-l}. This doesn't mean that y_{t-m}has any new information that may help in forecasting. To overcome this problem, we use PACF, partial autocorrelation function (Hyndman & Athanasopoulos, 2018).

PACF measures the correlation between y_t and y_{t-l} after removing the effects of intermediate lags. Hence, PACF is a better measure to understand if there's an actual relationship between y_t and y_{t-l}. PACF plot of demand shows that only lags 1 to 5, 7, 12 and 14 have significant correlations with y_t. Similarly, the PACF plot of consumption tells that lags 1, 2, 4 to 7 and 14 have significant correlations with y_t. Although partial autocorrelation between y_t and y_{t-3} is just significant (as the spike is slightly above the dashed blue line), we can neglect the partial correlation at lag 3.

However, the standard ACF and PACF plots become unreliable for large lags and hence, we also present improved versions of ACF and PACF plots (figures 24 to 26), known as tapered ACF and PACF plots (Hyndman, 2014).

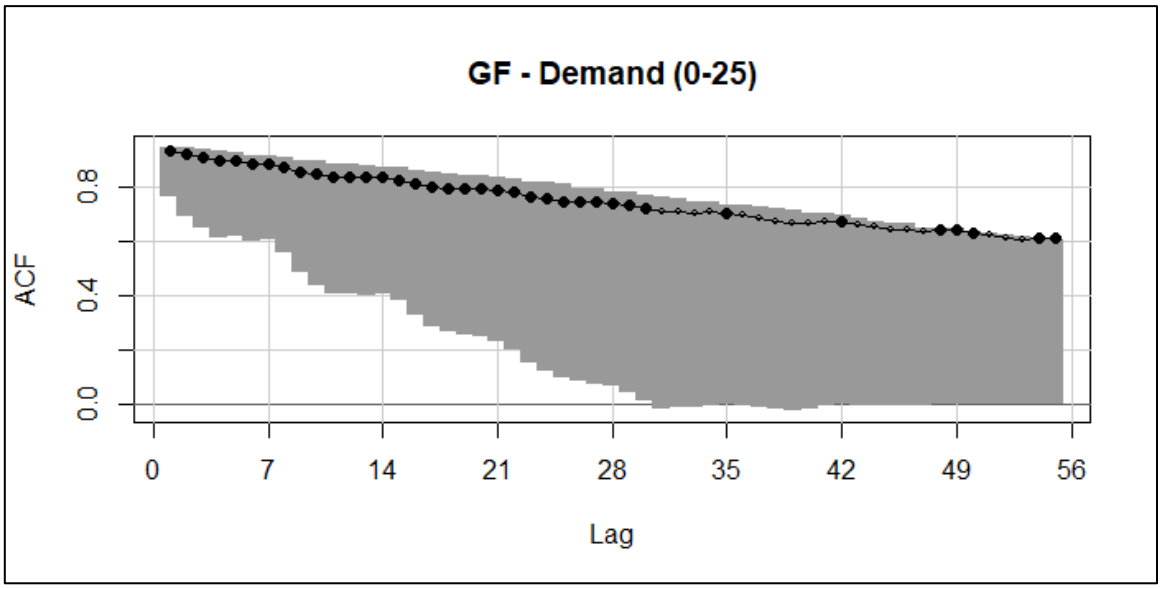

Figure 24: Tapered ACF of demand for 0-25 kWh bin (GF)

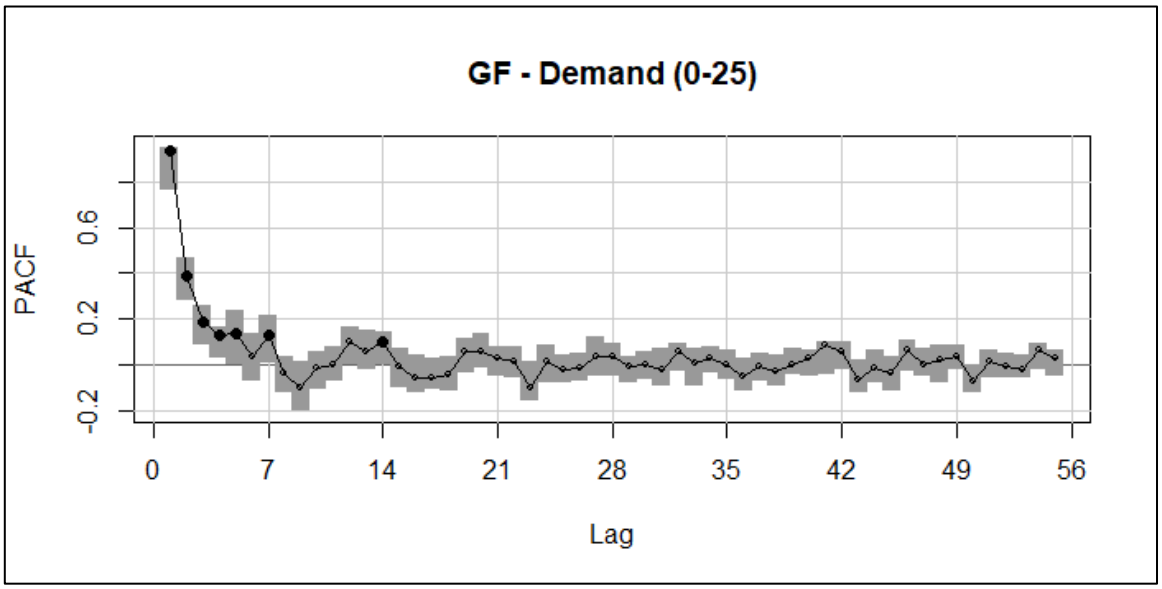

Figure 25: Tapered PACF of demand for 0-25 kWh bin (GF)

Autocorrelations that are significantly different from zero are shown using the dark solid circles, while insignificant circles are shown using bubbles (small open circles). The shaded region represents the 95% bootstrapped confidence intervals (Hyndman, 2014). The ACF plot (figure 24) shows that for demand, the first 29 autocorrelations are significant; after this, lags

at multiples of 7 remain statistically significant, with autocorrelations at neighbouring lags seldom significant. The tapered PACF plot for demand (figure 25) shows that partial autocorrelations from lags 1 to 5, and at lags 7 and 14 are statistically significant. Similar insights can also be drawn from tapered ACF and PACF plots for consumption (figures 26 and 27) of 0-25 kWh GF time series.

A better understanding of ACF and PACF help us in making a better decision on the order p of a $AR(p)$ or order q of a $MA(q)$ process. Hence, we shared some lights on tapered ACF and PACF plots as they can be more useful and easier to interpret than the conventional ACF and PACF plots (Hyndman, 2014). However, we would restrict ourselves to the conventional ACF and PACF plots as tapered plots would be of little use to our objective.

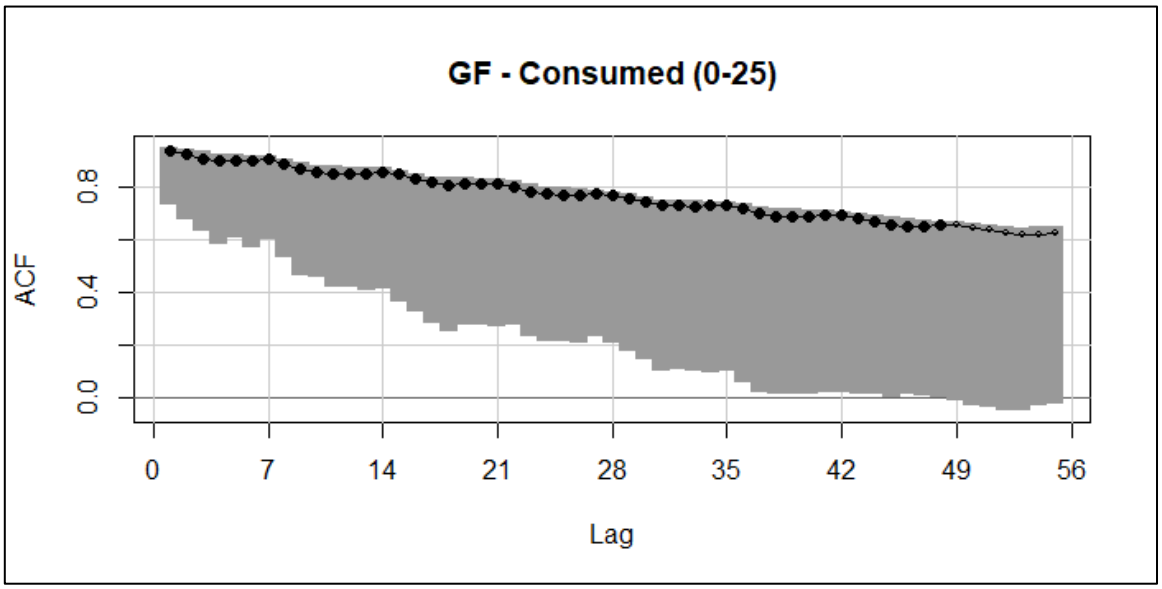

Figure 26: Tapered ACF of consumption for 0-25 kWh bin (GF)

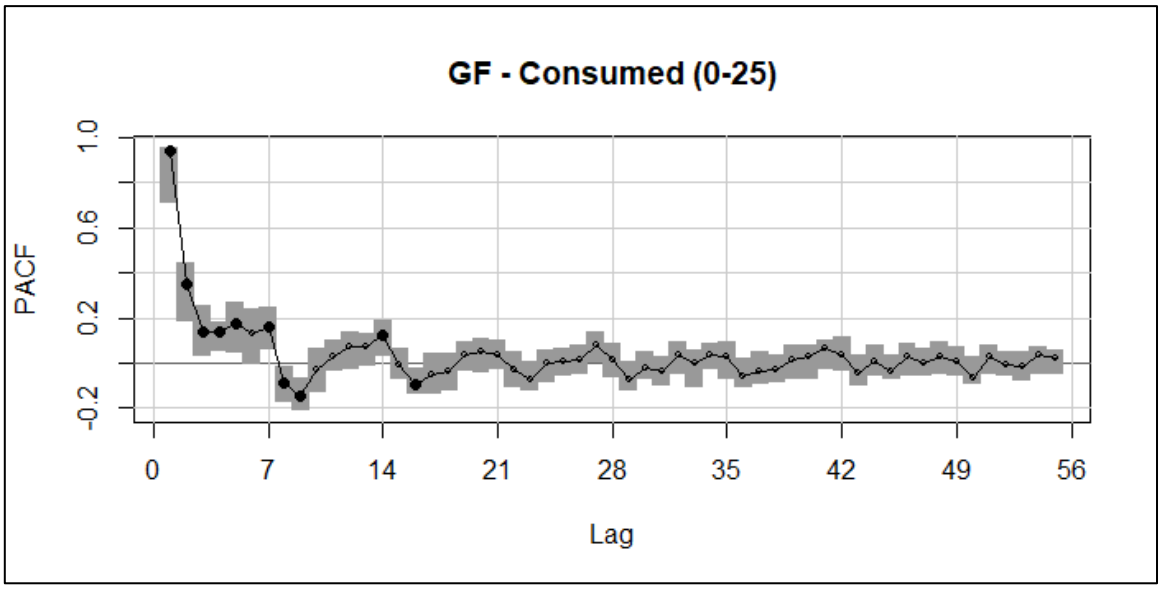

Figure 27: Tapered PACF of consumption for 0-25 kWh bin (GF)

5. Forecasting

5.1 A Retrospect of Project Objective

Based on the discussion with EA Technology, the primary objective of the project was to estimate the EV load on distribution networks in the UK when the roads would be inundated with a higher number of EVs (section 3.2). Furthermore, the estimate should be based on the information available to the DNOs (distribution network operators) in the future. While the available data had many features (predictors), not all of them would be available to the DNOs in the future. The constraints on the breadth of information available to the DNOs seriously limit the modelling approach we could have taken to forecast the energy demand. EA Technology asked us to come with a statistical model that would give an estimate of energy consumption due to EV charging; the energy consumption should be based on number of EV owners in the market.

The mathematical representation of the statistical model based on the aforementioned objective would be:

$$c = f(o) + \epsilon$$

where,

c = energy consumption

o = number of EV owners

f = statistical function that maps o and c

ϵ = irreducible random error which has zero mean

The objective, in its original form, missed on a lot of variabilities that EV charging might have. For instance, the energy consumption, in a given charging event (or transaction), of an EV with smaller battery could be less than that of an EV with larger battery; besides, the frequency of charging for an EV with smaller battery could be higher than that of an EV with larger battery.

Moreover, the consumption may vary from one day of the week to another and from one season of the year to another. Many such possibilities were discussed with EA Technology, and after a comprehensive discussion on what information would be available to the DNOs in future, we reformulated the objective with additional variables as shown below. It's worth mentioning that only such variables were considered that were available or accessible to us as well as of concern to EA Technology.

The reformulated objective can be mathematically represented as:

$$c = f(o, d, s, b) + \epsilon$$

where,

c = energy consumption

o = number of EV owners

d = day of the week

s = annual season of the year (spring, summer, autumn or winter)

b = battery capacity of EV

f = statistical function of that maps o and c

ϵ = irreducible random error which has zero mean

It is worth mentioning that the data available to the DNOs would only have information on the number of EV owners in their distribution network, day of the week, battery capacity of EVs and annual season of the week; hence, forecasting models were developed around the information that would be available to the DNOs in future.

We know from earlier discussions in section 4.3 that the number of users who charge their EVs was smaller than the number of owners; there were, however, exceptions when the users equalled the owners; however, it happened for very small number of owners and hence, can be neglected as it would highly unlikely in future scenarios. In general, we would observe that the number of users who are charging their EVs is always less than the number of actual owners in the market.

The time plots shown below (figure 28) depict how the proportion of EV owners who are active per day changes over time.

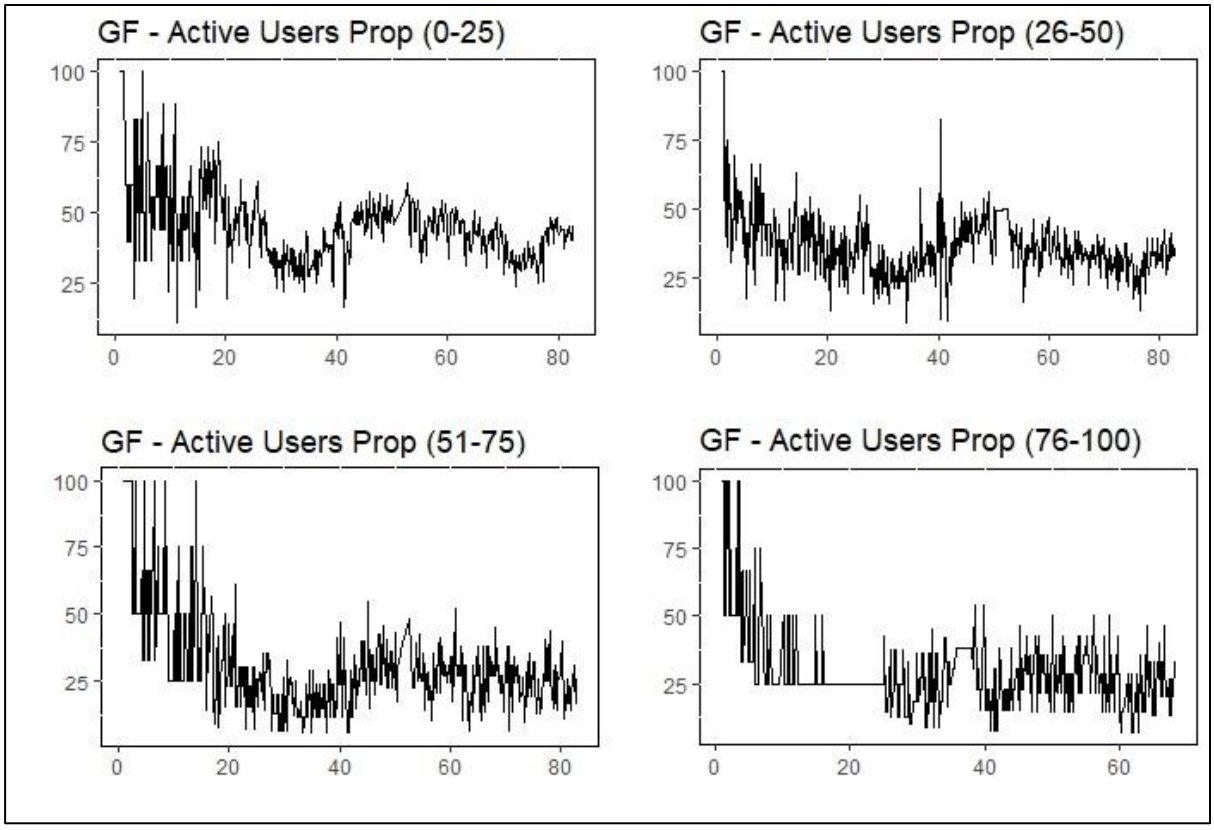

Figure 28: Active users proportion for all bins (GF)

Initially, the proportion is at 100% as the number of EV owners is very small and hence, the active users are equal to the number of EV owners. However, as the number of EV owners begins to rise, the proportion of active users per day declines and never even crosses the 60% mark. Based on our observations, we can ascertain that the number of users is a random variable that needs to be predicted before we can forecast energy consumption.

The summary statistics shown below corroborate the findings from the time plots shown above.

Table 12: Summary statistics of active users proportion for all bins (GF)

Statistics	0-25 kWh	26-50 kWh	51-75 kWh	76-100 kWh
Minimum	11.11	8.51	5.88	7.14
1st Quartile	36.84	29.27	20.00	22.40
Median	44.44	34.88	25.81	25.00
Mean	45.29	36.14	30.61	29.16
3rd Quartile	50.98	41.46	35.29	35.71
Maximum	100.00	100.00	100.00	100.00

We observe that for users who fall in the smallest battery capacity bin, the mean percentage of active users is even less than 50. For other users, the mean is always less than 40%. The mean would go down even further if we drop the few observations at the beginning of the trials as the number of EV owners and hence, users are very small, resulting in extremely higher percentage values of active users; this, however, would not be the case, in general, and would never occur in real-life scenario. While building predictive models, we would drop such observations that would never appear in a real-life set-up. However, we would maintain consistency across all the models that we would build and hence, would discuss later on the number of observations to be dropped.

The boxplots shown below (figure 29) exhibit variability of users as a function of the season of the year and day of the week. Hence, the mathematical formulation for forecasting the number of users per day would be identical to the one we devised to forecast energy consumption, i.e., we would fit a statistical model that would explain the variability in the number of users as a function of EV owners, day of the week, season of the year and battery capacity.

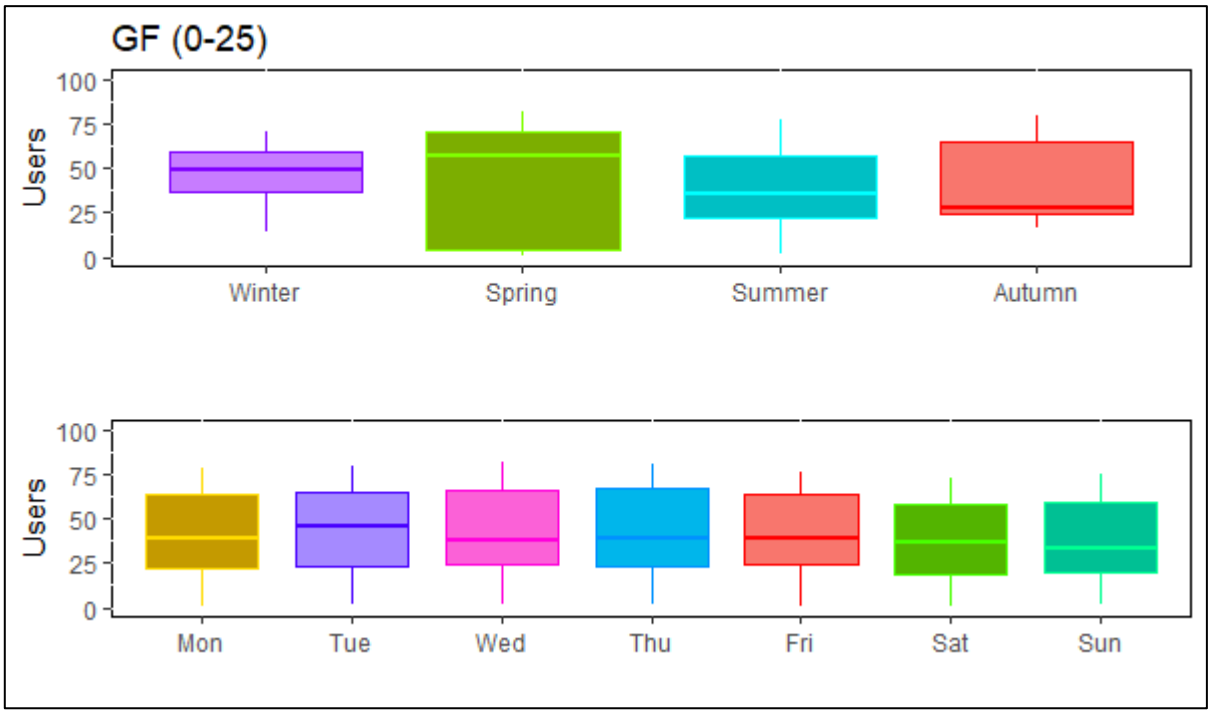

Figure 29: Variability of users with season and day for 0-25 kWh bin (GF)

Furthermore, from the scatterplots discussed in section 4.3.6 we observed that consumption had a stronger correlation with users (0.981) than with owners (0.905), i.e., users may be a better explanatory variable (predictor) than owners to forecast demand. The choice of explanatory variables for forecasting a given response variable (target) would be discussed later in this chapter. Since the maximum number of variables (predictors) available to the DNOs would be only four (as advised by EA Technology), selection of predictors to find the model with best forecast performance doesn't require sophisticated feature selection algorithms (Hmamouche, et al., 2017; Yoon & Shahabi, 2006; Yang, et al., 2005; Tyralis & Papacharalampous, 2017; Rahajoe, et al., 2017). In case the formulation of the objective was in high-dimensional space, i.e., if we had a very high number of explanatory variables (), we could have chosen one or more of the approaches based on our objective to select the optimal combination of variables (predictors) in our forecast model.

5.2 Ex-ante vs Ex-post vs Scenario-based Forecasting

Depending on what is assumed to be known when forecasting, we can classify the different types of forecasts into three broader categories (Hyndman & Athanasopoulos, 2018).

- Ex-ante forecasts are the ones which are computed using the information that is available in advance. In this case, to forecast the response variable, we need to forecast the predictors (explanatory variables) too as the forecaster has no information of the predictors as well. These are genuine forecasts as the forecaster has no knowledge of the future whatsoever.
- Ex-post forecasts are those which are computed using the actual information available on predictors, i.e., information on predictors is available prior to forecasting. It's worth mentioning that in ex-post forecasting, only the information on predictors is known and no knowledge of the response variable exists.
- Scenario-based forecasts are the ones which are computed assuming possible scenarios for the predictors that are of interest to the objective. In scenario-based forecasting, the uncertainty associated with the confidence intervals of the forecasts does not assume any uncertainty in the values of the predictors as it is believed that the predictors are known with certainty.

Based on the project requirements, we mathematically formulated the objective to incorporate the future scenario, i.e., the future information available to the DNOs. Using the available information, the DNOs (or authorities concerned) should be able to estimate the energy consumption caused by EV charging using a robust statistical model. While additional information relevant to EV charging and EVs might have enhanced the predictive performance of the model, EA Technology wanted the model to be robust so that it could forecast the energy consumption with minimal errors under increased uncertainty; this was definitely caused by the constraints on the quantum of information we could work with.

The mathematical formulation of the objective manifested itself in the form of a multiple regression problem: *predict the response based on the available predictors*. In our case, the

client would like to work with different plausible scenarios, with each scenario having its own set of values of the predictors, and we would be required to forecast the response variable based on the prior information on predictors for different scenarios; this was a case of scenario-based forecasting. For instance, the client might be interested in estimating the energy demand on a Sunday in the winter season for an EV ownership of 10,000 people having battery capacity within a given range, such as EVs with battery capacity less than 25 kWh.

5.3 The Curious Case of Univariate Forecasting

The day-wise time series we had was multivariate; hence, forecasters might argue that the simplest approach to start with could be univariate forecasting for the variables of interest (or individual time series as each time series in a multivariate set-up is also a variable). To address this concern, we would discuss our observations from the results obtained from one such univariate forecasting approach. Although EA Technology referred to the consumption of energy as the demand for energy, we made a clear distinction between the two terms. While the total consumption per day was referred to as *consumed*, the total connected load caused by all the transactions was referred to as *demand*. Hence, we focussed on two variables, *demand* and *consumed*, separately. While *consumed* would give an estimate of the mean consumption of energy per day, *demand* would indicate the mean connected load caused by all the transactions. Besides, since *demand* is the aggregate of the total connected load caused by all the charging events or transactions, it gives an estimate of the energy consumption under worst-case scenario when every charging event leads to maximum energy consumption governed by the capacity of the battery in kWh. In other words, a *demand* model gives an upper boundary prediction of consumption of energy and helps in estimating the total consumption of energy under worst-case scenario, although this is a hypothetical estimation and the actual consumption of energy given by the *consumption* model will always be less than that given by the *demand* model. We built seasonal ARIMA (Hyndman & Athanasopoulos, 2018) models for *owners*, *demand* and *consumed* (Although the primary objective of the project was to estimate consumption of energy, we included estimate of the total connected

load too to indicate that not every charging event would lead to full battery charge and hence, consumption would be usually less than the total connected load). We set the forecasting horizon to four years from the last day of observation, which was 9th October 2018. The time plots shown below highlight the estimated values of the three univariate time series (for the 0-25 kWh bin) with respective confidence intervals, both 80%, and 95%. The x-axis shows time in weeks; so, four years is a little over 200 weeks. It is not important how long the forecasting horizon is in the given instance as the entire idea to discuss univariate approaches is to show why such standard forecasting algorithms are not applicable in our case irrespective of any forecasting horizon. Moreover, it is also not important how the ARIMA parameters are chosen as we once again reiterate that the idea is to show why such approaches are not applicable in our case. Even the best ARIMA or exponential smoothing models do not solve the purpose. Hence, discussion on how the parameters are chosen is irrelevant.

We observe in figure 30 that as time progresses all the time series exhibit a strong linear upward trend. In principle, this does reflect the future scenario as the number of EV owners and hence, the demand and consumption of energy caused by EV charging would rise. Besides, the table (table 13) following the time plots (figure 30) show the last 6 estimates of all the three univariate time series towards the end of the forecasting horizon of 4 years (365.25 * 4 = 1461 days).

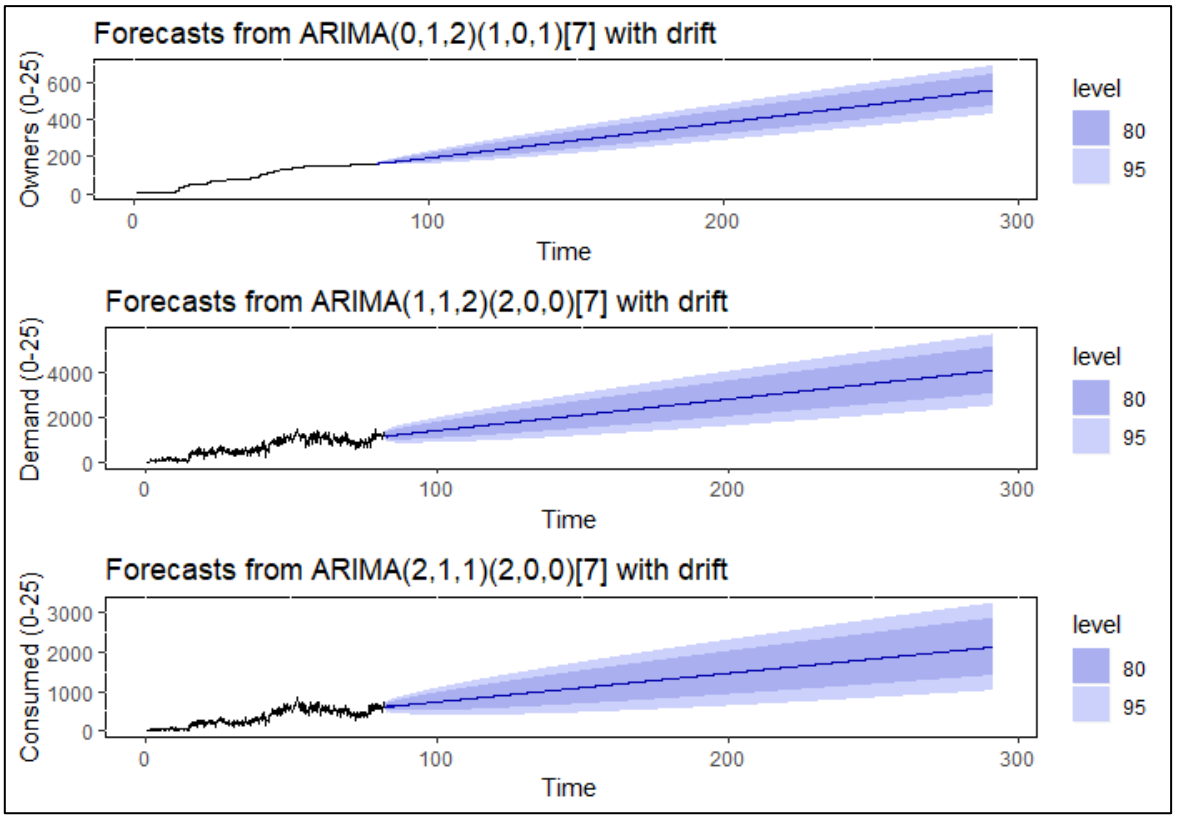

Figure 30: Forecasts using seasonal ARIMA for 0-25 kWh bin (GF)

Table 13: Last six forecasts of the forecast horizon for 0-25 kWh bin (GF)

Observation Count	Owners (0-25)	Demand (0-25)	Consumed (0-25)
1456	560.20	4086.22	2120.79
1457	560.48	4088.03	2121.83
1458	560.75	4090.03	2122.87
1459	561.03	4092.04	2123.92
1460	561.30	4094.04	2124.96
1461	561.58	4096.04	2125.99

Although the estimates of owners are non-integral, we can realise the nearest lowest or greatest possible integral values for the estimates of owners. If we assume the nearest greatest possible integral values, the last 6 estimates of owners are 560, 560, 561, 561, 561, 562.

In the trials, the upward trend of EV owners was controlled by the authorities concerned. Since the estimates of owners depend on the learning of forecasting algorithm, the estimates are biased because of the presence of this element of control: how the participants (EV owners) had gradually risen in the trials. The rise of EV owners in the trials doesn't represent the actual trend in the real-world scenario. While the estimates of demand and consumption might have been captured well by the model as they depended upon how many people had EVs and hence, how many were actually charging per day, the estimates for all the time series suffer from the following issues and hence, do not completely reflect the future scenario.

- The estimates suggest that towards the end of four years from 9th October 2018, the number of EV owners with battery capacities less than 25 kWh would be 562, and the demand and consumption would vary as observed from the table above. Although the demand and consumption of energy are a function of the EV owners, their estimates towards the end of the forecasting horizon (four years) are biased as the estimates of EV owners are biased. It is highly likely that the UK roads would have more than 562 EVs with battery capacities less than 25 kWh around October 2022 and hence, the estimates of demand and consumption do not reflect the true future scenario of additional load on distribution networks. Moreover, the estimates only indicate that when the EV ownership reaches 562 for battery capacity less than 25 kWh, mean demand and consumption would be approximately 4096 and 2126 kWh respectively.
- Although the univariate models might give a good estimate of demand and consumption as a function of EV owners, we lose the variability of demand and consumption across different seasons of the year and days of the week. This can be attributed to the fact that estimates of demand and consumption for a given number of EV owners would only be valid for a given day of the week and season of the year if we opt for univariate forecasting. For instance, the 1461st estimate from the table above would fall somewhere in Autumn and on a particular day of the week. As such, the univariate model would not be able to tell the effect of any other season or day on the demand and consumption for an EV ownership of 562. This seriously undermines the project objective.

From the previous discussions on the constraints offered by univariate forecasting models, we can conclude that univariate forecasting doesn't align with the projective objective, although it might still generate good estimates of demand and consumption. Hence, based on our understanding of the project objective and analyses, we decided to focus on multiple regression algorithms applicable to time series data as they aligned with what the client sought.

5.4 The Nested Approach

The flow diagram shown below (figure 31) is a pictorial representation of the mathematical formulation of the objective. As we mentioned in the previous section that we would not only be forecasting the consumption of energy per day but also the total connected load caused by all the transactions per day. While consumption tells us the actual energy consumed, demand tells us the maximum connected load for all the charging events in a given day. The flow diagram, in its original form, miss out on a lot of important information that was present in the data or was extracted from the data. For instance, the scatterplots shown in the section 4.3.6 reveal that variables such as transactions and users have a stronger correlation with the two response variables of interest than owners have with them. So, the two predictors may turn out to be more useful than owners in forecasting the response variables. Similarly, duration of charging (time) and demand also show strong correlations with consumption. Hence, a lot of plausible combinations existed in forecasting demand and consumption of energy. It was a matter of developing models with different combinations of predictors to eventually identify which model gave the best forecasting performance.

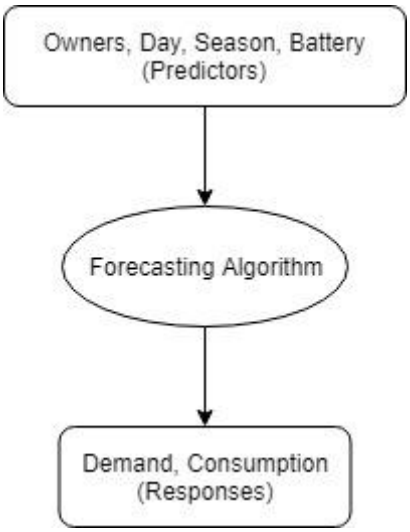

Figure 31: Flow diagram to show the project objective

An important consideration we had to keep in mind while developing models was that when the model would be used for forecasting, the available information would be restricted to only four variables, which had already been shown in the mathematical formulation as well as the flow diagram above. However, we needed to incorporate the variables which showed stronger correlations with the response variables. Hence, we decided to adopt the following strategy and called it the 'Nested Approach'.

- We used information from the day-wise time series (GF data) to train the forecasting algorithm, i.e., training was done using any subset of the features in the day-wise time series. For instance, we might have used the information on demand for each battery capacity bin to train an algorithm to forecast consumption; the forecasts obtained on the training data were then fitted values (fitted values are the in-sample predictions or the predictions made on the training data). However, when forecasting consumption on test data (scenario-based data), we couldn't use demand as this feature wasn't present in the scenario-based data (section 5.1 explains the structure of test data). Hence, we needed to devise an approach that would help us in determining demand (and all other variables of interest) and then, could use the estimates of demand (and other variables) as a feature to forecast consumption (or any other variable of interest).

- Variables such as users, transactions, time, demand and consumption had stronger correlations with each other than they had with owners. Hence, there was a possibility that variables other than owners could have given a better estimate of other variables. However, this was to be done keeping in mind the causal relationship between any two variables, i.e., if X caused Y, then it didn't mean that Y also caused X. For instance, users caused transactions but not vice-versa, and hence, transactions could be modelled as a function of the user, but not the other way around. So, we could train algorithms to forecast transactions as a function of owners, or as a function of users (users being a function of owners) but not both as this led to overfitting (section 5.6). Similarly, time could be modelled as a function of owners, users or transactions but not as a function of all at the same time. Similar analysis held true for demand and consumption. The choice of the final model to forecast consumption was based on forecasting performance.

The flow diagram shown (figure 32) below depicts one such nested forecasting approach to eventually forecast our response variables. In the approach shown below, the four predictors (scenario-based test data) are used to forecast users. Once we get estimates of users, we use them to forecast transactions. We, then, use estimates of transactions to forecast both time and demand and eventually use estimates of demand to forecast consumption. This approach helps to extract more information from the available test data that wouldn't be apparently present at first place, and help develop better forecasting models than the ones that could be developed using only the information directly available in test data.

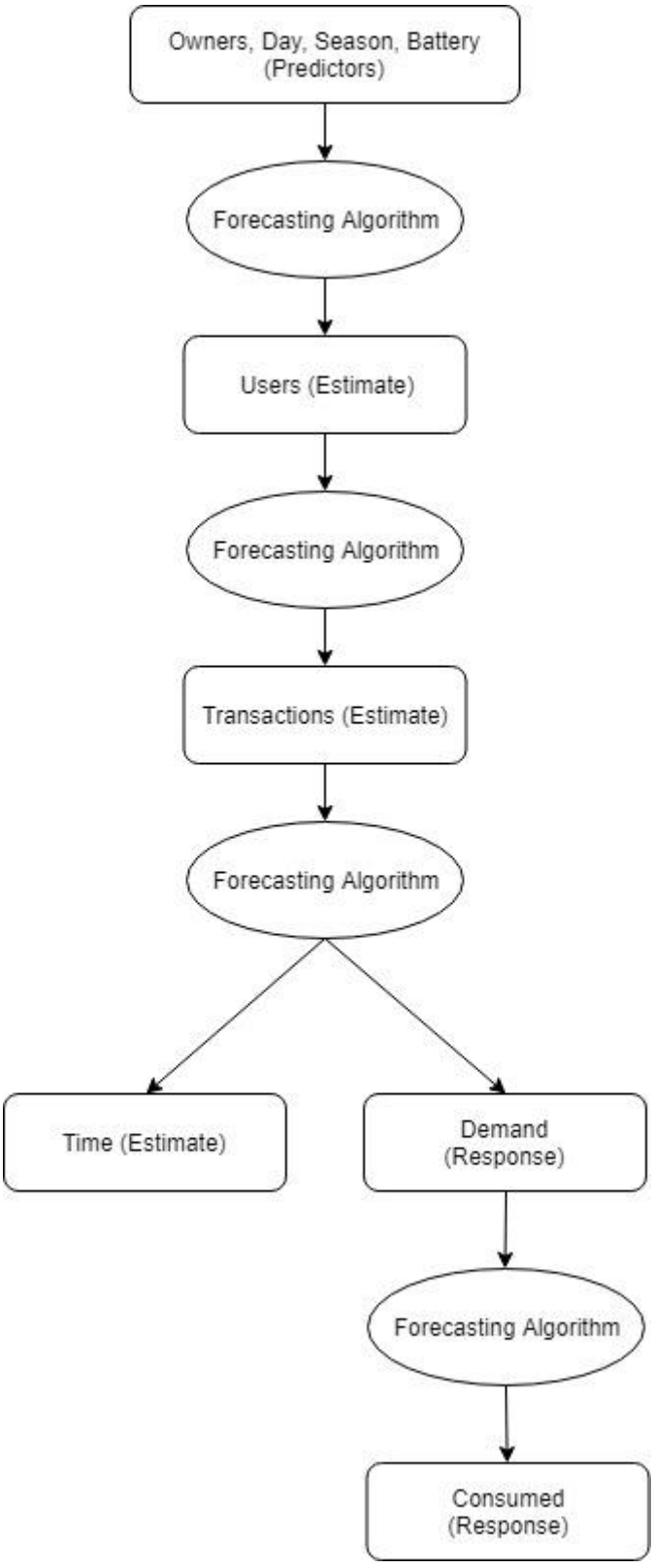

Figure 32: The nested modelling approach

5.5 Algorithms

In this section, we discuss different algorithms that aligned with the business objective to forecast consumption of energy.

5.5.1 Time Series Regression

The fundamental concept of time series regression is that we forecast the time series y_t as a linear function of some other time series x_t. We observed in section 4.3.6 that all the time series showed strong correlations with each other and their relationships were significantly linear. Hence, the scatterplots validated the use of time series regression to forecast our response variables.

The mathematical formulation of a simple time series linear regression model is shown below.

$$y_t = \beta_0 + \beta_1 x_t + \varepsilon_t$$

where,

y_t = response

x_t = predictor

ε_t = random error

β_0 = predicted value of y_t when x_t is zero

β_1 = average predicted change in the value of y_t when the value of x_t changes by one unit

5.5.2 Regression with ARIMA Errors

An improved version of regression is to fit an ARIMA (Hyndman & Athanasopoulos, 2018) model on the residuals, to boost forecasting. This would be possible as the overall forecast would be a combination of forecasts from the regression part and the ARIMA part (Hyndman & Athanasopoulos, 2018). Regression with ARIMA errors is useful when the residuals show high autocorrelations, thereby meaning that the model didn't capture all the information from the data. The regression equation described in the previous section can be reformulated to incorporate ARIMA errors as described below.

$$y_t = \beta_0 + \beta_1 x_t + \eta_t$$
$$\eta_t = \phi \eta_{t-1} + \varepsilon_t + \theta \varepsilon_{t-1}$$

where,

y_t = response

x_t = predictor

η_t = regression error

β_0 = predicted value of y_t when x_t is zero

β_1 = average predicted change in the value of y_t when the value of x_t changes by one unit

ε_t = ARIMA error

ϕ = parameter of the AR part of the ARIMA model

θ = parameter of the MA part of the ARIMA model

The ARIMA models fitted on the residuals use the Hyndman-Khandakar algorithm for automatic ARIMA modelling (Hyndman & Khandakar, 2008; Hyndman & Athanasopoulos, 2018).

5.5.3 Distributed Lag Models

Assume an instance when a user with a low battery EV charges his/her vehicle on a Monday. It is highly likely that the user would charge the EV again on Tuesday if the EV is the primary vehicle for commutation. If the user doesn't charge the EV on Tuesday, then the probability of charging the EV on Wednesday increases. On the contrary, the probability of a user charging his/her EV with a higher battery on consecutive days is comparatively less. The time plots on the active proportion of EV owners per day also corroborate these assumptions. We understand that previous predictor values (such as the number of users in the previous day (s)) can impact the consumption of energy at present. We can summarise that it is possible that the impact of a predictor which is included in a forecasting model may not be simple and immediate (Hyndman & Athanasopoulos, 2018). As such we need to allow for lagged values of predictors too as they may help improve the forecasting accuracy. Such an approach is

called 'modelling with distributed lags'. The mathematical representation of such an approach is shown below.

$$y_t = f(x_t, x_{t-1}, x_{t-2}, \dots)$$

where,

y_t = response

x_t = predictor

x_{t-l} = lagged value of a predictor at a lag l

f = function that maps the relationship between y_t and x_t

We incorporate distributed lags in regression with ARIMA errors. The mathematical formulation of regression with ARIMA errors including distributed lags is as follows.

$$y_t = \beta_0 + \gamma_0 x_t + \gamma_1 x_{t-1} + \gamma_2 x_2 + \cdots + \gamma_l x_{t-l} + \eta_t$$
$$\eta_t = \phi \eta_{t-1} + \varepsilon_t + \theta \varepsilon_{t-1}$$

where,

y_t = response

x_t = predictor

x_{t-l} = lagged value of a predictor for a lag l

η_t = regression error

β_0 = predicted value of y_t when all predictors are zero

γ_l = average predicted change in the value of y_t when value of x_{t-l} changes by one unit

ε_t = ARIMA error

ϕ = parameter of the AR part of the ARIMA model

θ = parameter of the MA part of the ARIMA model

5.5.4 LSTM

In conventional feed-forward neural networks (NNs), all cases are considered to be independent, i.e., to estimate the future value of a sequential data (for instance, a time series),

no consideration is given to the values in the past. We can say that such neural networks have no memory (Chollet & Allaire, 2018). Recurrent neural networks (RNNs), which have loops in them allowing information to persist, address this issue (Olah, 2015). RNN processes sequential data by iterating through the sequence elements and maintaining a state (s_t) containing information based on what the network has seen until then (Chollet & Allaire, 2018). The state of the RNN is reset before processing the next independent sequence (Chollet & Allaire, 2018); this makes one sequence a single data point, i.e., a single input to the network (Chollet & Allaire, 2018). The pseudocode (Chollet & Allaire, 2018) to explain RNN is depicted in the figure shown below (figure 33).

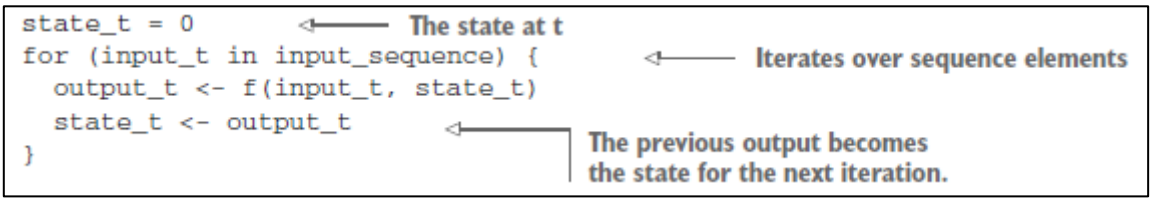

Figure 33: Pseudocode to explain RNN (Chollet & Allaire, 2018, p. 181)

The RNN loops over timesteps (a chunk of sequential data treated as one sample or data point), and at each timestep, it produces an output based on the current input and the current state. For instance, at time $t = k$, output (h_k) is a function of the input (x_k) and state (s_k). This current output, h_k, then becomes the state for the next timestep. The previous output for the first timestep is not defined; hence, the initial state (s_o) is set to zero.

However, simple RNNs have a major issue: although they should theoretically be able to retain at present time the information about inputs seen many timesteps before, practically, such long-term dependencies are impossible to learn (Chollet & Allaire, 2018) because of vanishing gradient problem (Hochreiter, 1998).

LSTM (Long Short-Term Memory) networks are an advanced version of RNNs, capable of learning long-term dependencies (Olah, 2015; Chollet & Allaire, 2018). LSTM networks save

information that can be used later, thereby preventing past information to gradually fade with time (Chollet & Allaire, 2018).

5.6 Performance Assessment

We mentioned in the previous section that in our project, a model is chosen based on its forecasting performance as achieving a higher predictive accuracy is intrinsic to our objective. The reasons for choosing forecasting performance over model fit as the criterion to select the best model can be summarised below (Hyndman & Athanasopoulos, 2018).

- A model that fits the training sample well doesn't necessarily forecast well as a better model fit can result from capturing patterns in that data that occur by chance (randomly). Such a case is called overfitting.
- A better fit can be obtained by using enough parameters while training the algorithm.

In the following sections, we discuss the approach we chose to evaluate the forecast models and our choice of error metric.

5.6.1 Model Evaluation and Variable Origin

A forecast model can be evaluated on unseen or new data. A model X that gives higher forecast accuracy than another model Y on new data is a better model for forecasting. It is worth mentioning that no part of the new data should be used to fit the forecast model.

In practice, it's not always possible to find new data to evaluate our forecast models. Under such circumstances, it is a common practice to split the data into two components: training and test data. We fit the model on the training data and evaluate the model performance on the test data. The decision on the split is user-specific; however, practitioners choose 20% of the data as the test data. However, the choice of the split also depends on the length of the data. If the data is small, then creating a training-test split may render us with an even smaller training sample to fit the model. A popular approach to evaluate model performance is time

series cross-validation (Hyndman & Athanasopoulos, 2018). In this case, we do not create training and test samples but create successive test sets with a single observation. The corresponding training sets consist of all the observations that occur before the test sample in the time series data. This approach is also called 'evaluation of forecast performance on rolling origin' as the size of training sample gradually increase as we move the test sample observation by one time-unit forward.

However, our objective is scenario-based forecasting and hence, we do not have a fixed forecasting horizon. We, in fact, need to forecast for years ahead in time and the length of the period is also user-specific. Hence, working with a small number of origins seems unreasonable to our case as we need to test our model on larger samples of data. Hence, we propose a modified version of cross-validation, 'evaluation of forecast performance on variable origin, which is similar to the rolling origin methodology. However, in our proposed methodology, we work on larger samples of test data by changing the position of the origin (hence the name variable origin) in such a way that the time series dynamics are retained. Hence, random sampling is strictly prohibited.

The diagram shown below (figure 34) depicts our proposed methodology. In this project, we decided to include only three origins so that we obtain three possible train-test splits as shown below. We start with 70% of the data as the test sample and keep decreasing that proportion in steps of 10% until it reaches 10%. We fit three different models on different training samples and evaluate their performance on respective test samples. This eliminates any bias caused by choosing a fixed length for the training sample to fit the model. We compute the mean performance of the models by using suitable measure of accuracy. The mean performance measure eventually helps in comparing models based on different algorithms. The model with the best accuracy is the one chosen for forecasting.

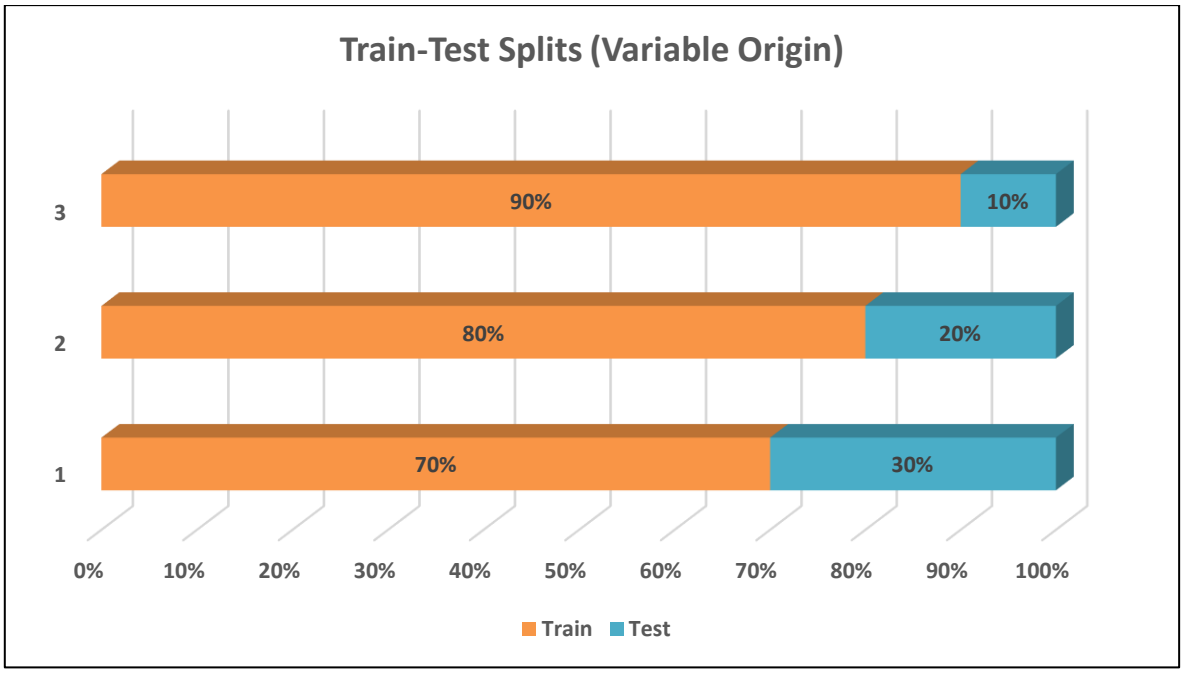

Figure 34: Variable origin

5.6.2 Error Metrics

When we forecast a variable, we are always left with an unpredictable part of an observation of that variable; this part is called forecast error. It is the difference between the observed value of a variable and its forecast. Obviously, better forecasting performance warrants lower forecast errors. For an h-step forecast, the forecast error can be mathematically represented as follows (Hyndman & Athanasopoulos, 2018).

$$e_{T+h} = y_{T+h} - \hat{y}_{T+h|T}$$

where,

y_{T+h} = observed value at time $T + h$

$\hat{y}_{T+h|T}$ = forecast at time $T + h$ using all the observations till time T

e_{T+h} = forecast error

Forecasts are calculated on test data, and hence are different from residuals which are calculated on training data (Hyndman & Athanasopoulos, 2018). A residual is the difference between an observed value and the estimated (fitted) value of that observation in the training

sample. If y_t is the actual value of an observation, and $\hat{y_t}$ is the estimated value of that observation, then residual is defined as

$$e_t = y_t - \hat{y_t}$$

Scale-dependent Errors

Forecast errors are on the same scale as the data (Hyndman & Athanasopoulos, 2018). Such error metrics cannot be used to compare performance across series having different units. If e_t is the forecast error, the two most commonly used scale-dependent error metrics are MAE and RMSE, which are defined below.

Mean Absolute Error: $MAE = mean(|e_t|)$

Root Mean Square Error: $RMSE = \sqrt{mean(e_t^2)}$

Percentage Errors

Percentage error, p_t, is given by $p_t = 100\ e_t / y_t$. Percentage errors are unit free and hence, can be used to compare performance across series with different units (Hyndman & Athanasopoulos, 2018). A popular percentage error metric is MAPE, which is defined below.

Mean Absolute Percentage Error: $MAPE = mean(|p_t|)$.

However, MAPE suffers from the disadvantage that for zero or very low values of y_t, it becomes undefined. Besides, MAPE also has another disadvantage of putting a heavier penalty on negative errors than on positive errors.

Since we will be comparing models to forecast consumption, the corresponding series across all the models will have the same unit and hence, scale-dependent error metrics can be used. For the sake of interpretability, we choose MAE as the error metric to compare the performance of different models. Besides, we would also use MAPE to compare models as it

can be used to compare models to forecast consumption with the models to forecast users; this would help in assessing how the model performance degrades from as we move from forecasting users to consumption of energy. Moreover, just like MAE, MAPE also offers the advantage of easy interpretability. Summarily, we would use MAE and MAPE to compare performance of models.

6. The Forecasters

6.1 Introduction

In this section, we discuss different predictive models developed to address the ultimate aim of identifying the best model that would forecast the consumption of energy with minimum errors. We discussed our modelling strategy in section 5.4, where we explained 'The Nested Approach' to bridge the gap between the data we had and the data the DNOs would have in the future. All the models were built using the same approach: we first leveraged the concept of variable origin, discussed in section 5.6.1, to fit three models on different train-test splits. However, before fitting the models, the data was normalized to ensure convergence of the learning algorithms as we were dealing with multivariate data with different scales. The performance of a given algorithm for forecasting was computed using the mean of MAE and MAPE across all the test samples for all the models. We, then, used the entire data to obtain the final model. Models across different family of algorithms were compared using the mean MAE and MAPE. In all the following sections, we would share the details of three models with *users*, *demand*, and *consumption* as target variable. It is worth mentioning that based on the current objective, time series regression is the benchmark algorithm, and in the following sections we also focus on how other algorithms perform with respect to the benchmark.

In the subsequent sections, we indicate a predictor with a prime (') if the estimate of the predictor is used for scenario-based forecasting. For instance, if the predictor, *users*, is used for forecasting *consumption*, we indicate the predictor as *users'* instead of *users*, to differentiate between the situations when actual values of predictors are used and when estimates of predictors are used. Since only *owners*, *day* and *season* are the predictors that will be available to the DNOs, any other predictor, if used for forecasting, will be indicated with a prime (') as they will not be available to DNOs and only their estimates can be leveraged for forecasting.

6.2 Time Series Regression

We share the details of the time series regression models for the 0-25 kWh bin. The results for other bins are present in Appendix C.

1. $\boldsymbol{users = f(owners, day, season) + error}$

We identified in section 5.1 that the number of consumers who charge their EVs every day (active users or simply, users) is a random variable that also needs to be predicted as this variable varies with every season and day of the week and is always less than the number of people who own EVs. Since we have two categorical variables, season and day, we use 3 dummies for season (3 dummies for autumn, spring, and summer) and 6 dummies for day (6 dummies for all the days except for Sunday) as the number of dummies should always be less than the number of factors in a categorical variable to avoid the dummy variable trap (Hyndman & Athanasopoulos, 2018). We observe that factors 'spring' and 'sat' are statistically insignificant at 5% significance level. For all the days whose coefficients are statistically significant, we observe that when the day changed from Sunday to any other day, there is always an increase in the number of users. Similarly, for all the seasons with statistically significant coefficients, we observe that when the season changed from winter to any other season, there is a decrease in the number of users. The estimates of the coefficients of the predictors and the mean error metrics across all the test samples are tabulated below (tables 14 and 15). The chosen model explains 90.94% ($adjusted\ R^2 = 0.9094$) of the variability in users as a function of the predictors.

Table 14: Coefficients of the regression model to predict users for 0-25 kWh bin (GF)

Coefficients				
Predictors	**Estimate**	**Std. Error**	**t-value**	**p-value**
intercept	0.023379	0.014803	1.579	> 0.05
owners	0.816852	0.011017	74.143	< 0.05
autumn	-0.082634	0.012224	-6.76	< 0.05
spring	0.010211	0.011801	0.865	> 0.05
summer	-0.07167	0.01148	-6.243	< 0.05
mon	0.046637	0.014004	3.33	< 0.05
tue	0.061698	0.014004	4.406	< 0.05
wed	0.064494	0.014047	4.591	< 0.05
thu	0.067558	0.014048	4.809	< 0.05
fri	0.043191	0.014004	3.084	< 0.05
sat	-0.007239	0.014004	-0.517	> 0.05

Table 15: MAE and MAPE of the regression model to predict users for 0-25 kWh bin (GF)

Predictors	**MAE**	**MAPE**
owners, season, day	10.69	19.45

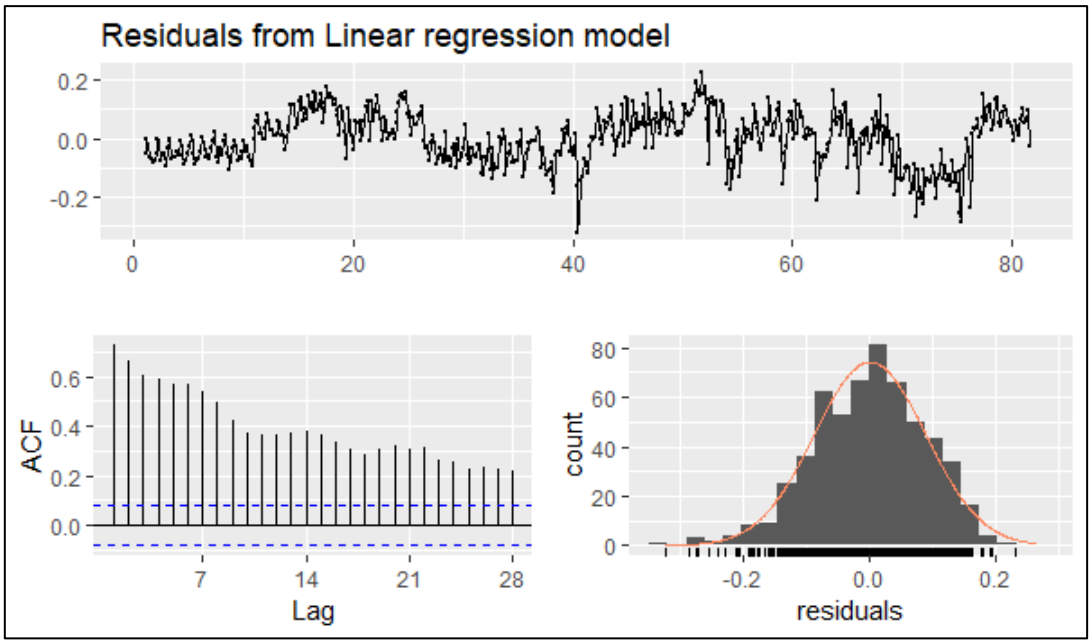

Figure 35: Residuals of the regression model to predict users for 0-25 kWh bin (GF)

From the plots shown above (figure 35), we observe that the residuals show a strong autocorrelation among themselves, thereby implying that there is still significant information left in the data that the model failed to capture (Hyndman & Athanasopoulos, 2018).

2. $\boldsymbol{demand = f(time') + error}$

The total connected load, called demand, could be modelled as a function of four different sets of predictors. The error metrics for forecasting demand as a function of different sets of predictors are shown below (table 16). We observe that the error metrics for time' as the predictor are nearly the same as those for owners, season, day as predictors. However, we chose the model with fewer predictors to minimise model complexity.

Table 16: MAE and MAPE of the regression model to predict demand for 0-25 kWh bin (GF)

Predictors	MAE	MAPE
owners, season, day	195.14	21.47
users'	213.83	23.42
trans'	206.06	22.7
time'	195.92	21.19

The estimates of the coefficients of the predictors are tabulated below (table 17). The positive coefficient in the model suggests that as the duration of charging (time') increases, the total connected load (demand) also increases. It is worth mentioning that time is modelled as a function of owners, season, day as the model gives the lowest error metrics (MAE = 26.87, MAPE = 25.26) than models with other predictors.

Table 17: Coefficients of the regression model to predict demand for 0-25 kWh bin (GF)

Coefficients				
Predictors	**Estimate**	**Std. Error**	**t-value**	**p-value**
intercept	0.03289	0.003748	8.774	< 0.05
time'	1.021479	0.007794	131.057	< 0.05

The model explains 96.82% of the variability in demand as a function of time ($adjusted\ R^2 = 0.9682$). Moreover, this model validates 'The Nested Approach' that we proposed in section 5.4. Although the model captures the variability of response variable pretty well (96.82%), the residuals have significant autocorrelations among each other as shown below (figure 36).

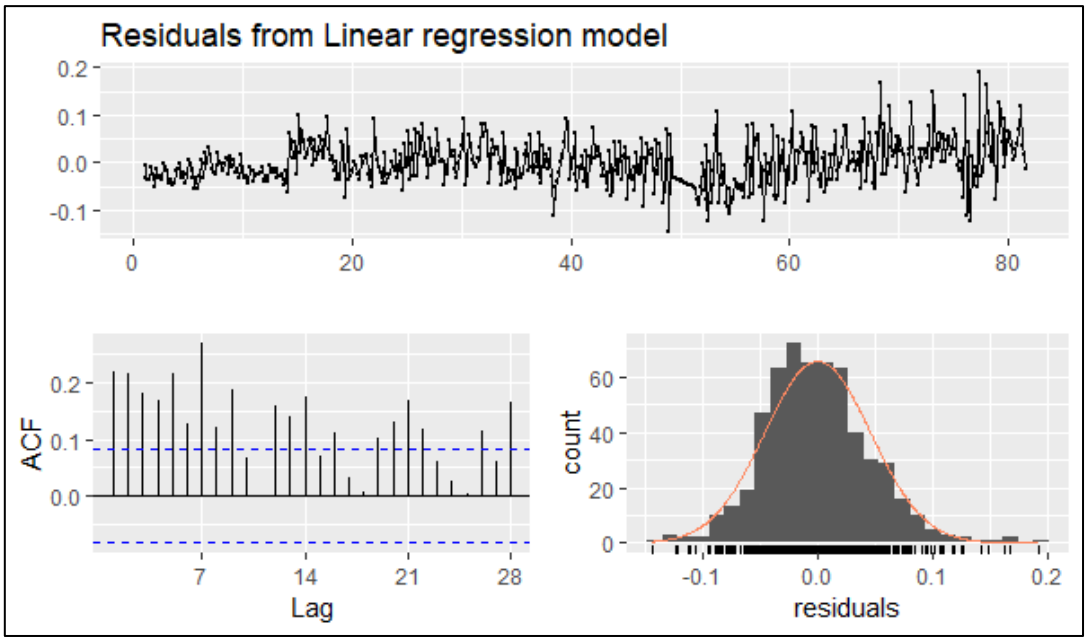

Figure 36: Residuals of the regression model to predict demand for 0-25 kWh bin (GF)

3. $\boldsymbol{consumed = f(demand') + error}$

Modelling consumption was possible with five different sets of predictors. The performance of different models to forecast consumption as a function of different sets of predictors can be observed from the table shown below (table 18). We identify that the model with demand' as the predictor gives the best mean performance over a variable origin (minimum error metric).

Table 18: MAE and MAPE of the regression model to predict consumption for 0-25 kWh bin (GF)

Predictors	MAE	MAPE
owners, season, day	112.69	25.05
users'	124.8	27.95
trans'	119.6	26.88
time'	114.07	25.38
demand'	110.93	24.6

The model with demand' as the predictor explains 97.18% of the variability in consumption ($adjusted\ R^2 = 0.9718$). In addition, the model also suggests that as the connected load (demand) increases, the consumption also increases (indicated by the positive sign of the coefficient of demand' in the model, table 19).

Table 19: Coefficients of the regression model to predict consumption for 0-25 kWh bin (GF)

Coefficients				
Predictors	**Estimate**	**Std. Error**	**t-value**	**p-value**
intercept	-0.018975	0.003582	-5.297	< 0.05
demand'	0.956531	0.006866	139.31	< 0.05

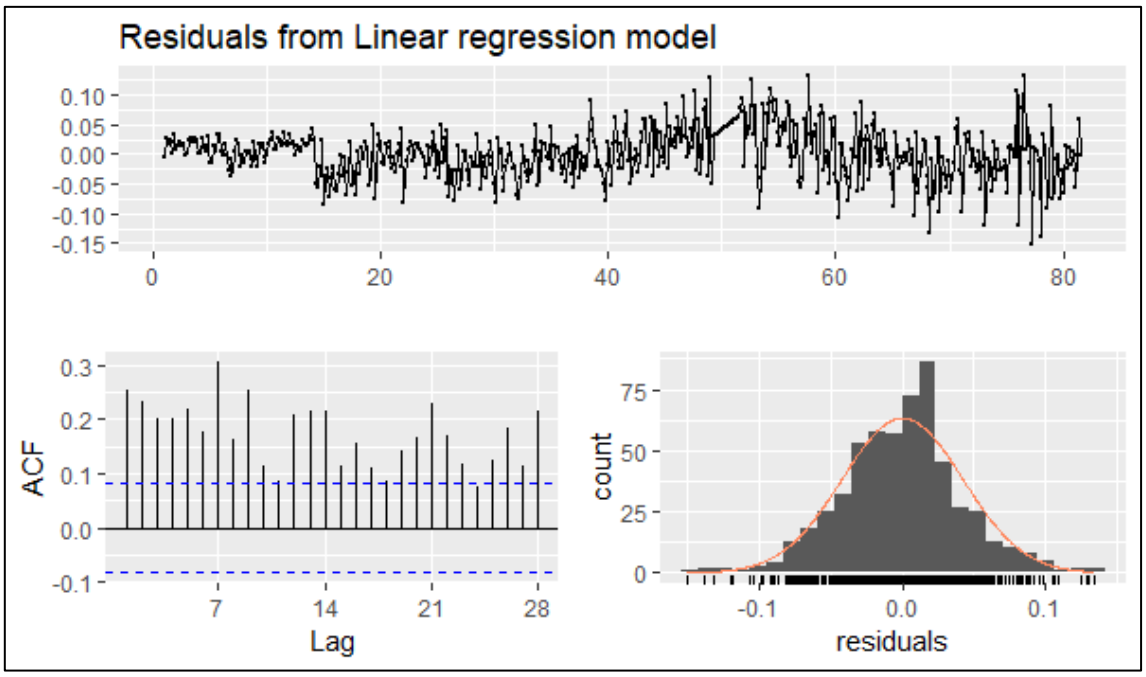

Figure 37: Residuals of the regression model to predict consumption for 0-25 kWh bin (GF)

The above figure shows (figure 37) that the residuals analysis have significant correlation among themselves, thereby indicating that the model fails to capture a lot of information from the data and hence, can be further improved.

6.3 Regression with ARIMA Errors

We observed in time series regression that the residuals were never even close to being white noise, i.e., the residuals exhibited strong autocorrelations among each other, leading to huge scope for extracting more information from the data to improve forecasts. Since the residuals obtained from time series regression didn't resemble white noise, we leveraged a boosting approach to fit an ARIMA model on the regression errors. The errors obtained after fitting ARIMA model, ARIMA errors (residuals), should represent white noise.

We share the details of the regression models with ARIMA errors for the 0-25 kWh bin. The results for other bins are present in Appendix C.

1. $\boldsymbol{users = f(owners, day, season) + ARIMA\ error}$

The estimates of the coefficients of the model are shown below (table 20). We see four additional predictors in the table below: *ar1*, *ma1*, *sar1,* and *sma1*. These four predictors are introduced into the model as we fitted an ARIMA model on the regression errors (section 5.5.2). A seasonal $(1, 0, 1)(1, 0, 1)_7$ ARIMA model is fitted on the regression errors. We also observe that no differencing is done before fitting the ARIMA model as the regression errors are found to be stationary.

Table 20: Coefficients of regression with ARIMA model to predict users for 0-25 kWh bin (GF)

Coefficients		
Predictors	**Estimate**	**Std. Error**
ar1	0.9417	0.0203
ma1	-0.5612	0.047
sar1	0.9091	0.0509
sma1	-0.8266	0.0684
owners	0.8187	0.0539
autumn	-0.0068	0.0342
spring	0.0155	0.0305
summer	-0.0105	0.0337
mon	0.0438	0.013
tue	0.0576	0.0136
wed	0.0607	0.0139
thu	0.0637	0.0139
fri	0.0403	0.0135
sat	-0.0095	0.013

The mean error metrics on a variable origin are shown below (table 21). We observe a marginal improvement in the performance of the regression model when an ARIMA model is fitted on the regression errors.

Table 21: MAE and MAPE of regression with ARIMA model to predict users for 0-25 kWh bin (GF)

Predictors	**MAE**	**MAPE**
owners, season, day	10.46	17.97

Although the model performance using regression with ARIMA errors marginally improves than that of the earlier model using time series regression, we observe a significant

reduction in the autocorrelation among the residuals, as shown in figure 38 (ARIMA errors); this makes the prediction intervals more reliable.

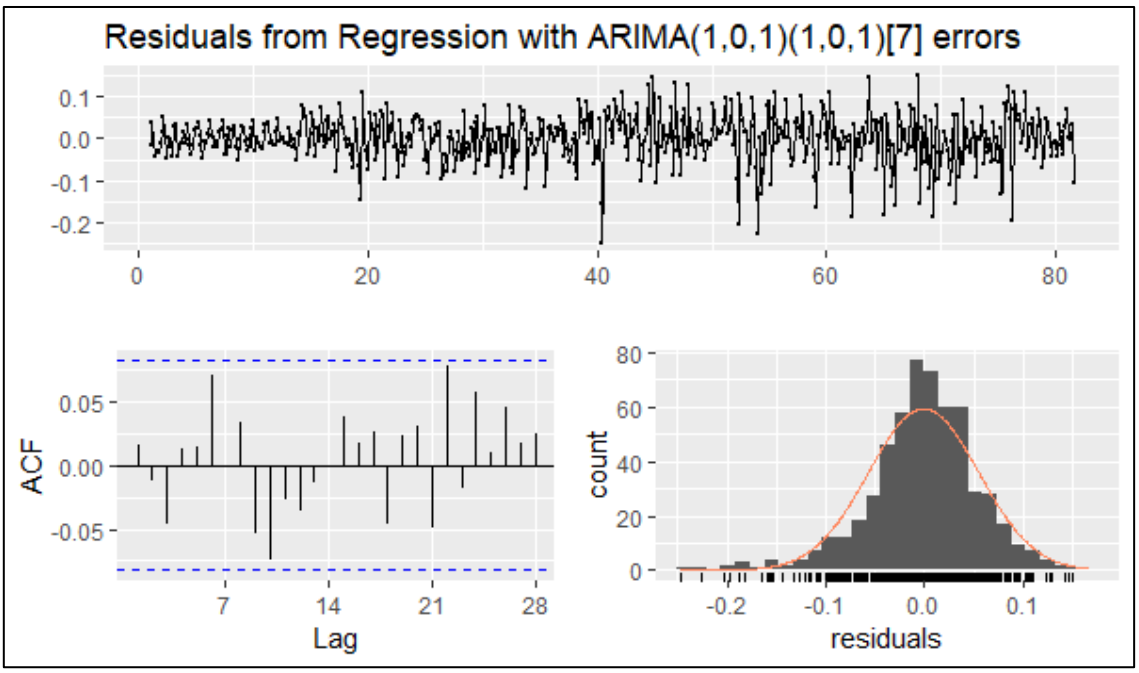

Figure 38: Residuals of regression with ARIMA model to predict users for 0-25 kWh bin (GF)

2. $demand = f(time') + ARIMA\ error$

The total connected load (demand) can be modelled using four possible sets of predictors. The mean performance of four such models on variable origin can be found below (table 22).

Table 22: MAE and MAPE of regression with ARIMA model to predict demand for 0-25 kWh bin (GF)

Predictors	MAE	MAPE
owners, season, day	161.12	18.02
users'	197.05	20.19
trans'	154.29	17.14
time'	144.53	15.43

We see that the model with time' as the predictor gives the best performance among all the models. Besides, the error metrics see a significant reduction in their values using regression with ARIMA errors compared to when only regression is used. Moreover, we can observe from the figure below (figure 39) that the autocorrelations among residuals get significantly reduced, thereby making the prediction intervals more reliable.

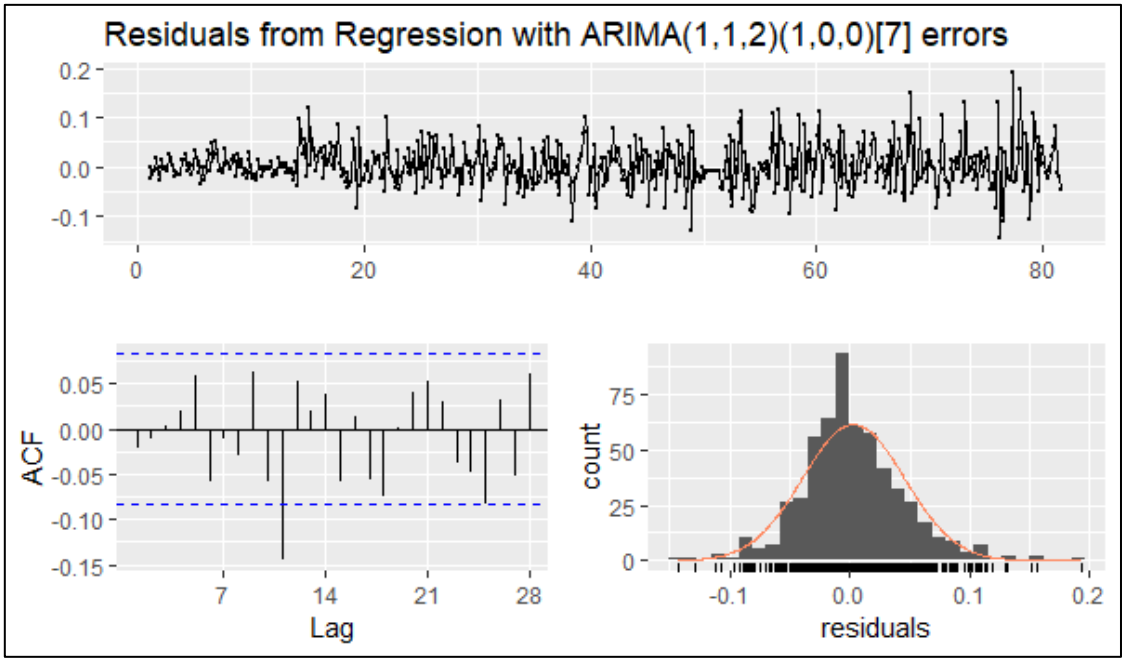

Figure 39: Residuals of regression with ARIMA model to predict demand for 0-25 kWh bin (GF)

The estimates of the coefficients with time' as predictor have been tabulated below (table 23). A seasonal $(1,1,2)(1,0,0)_7$ ARIMA model is fitted on the regression errors. We also observe that non-seasonal differencing is done before fitting the non-seasonal ARIMA part; this emphasises that the regression errors are not stationary and hence, differencing is warranted.

Table 23: Coefficients of regression with ARIMA model to predict demand for 0-25 kWh bin (GF)

Coefficients		
Predictors	**Estimate**	**Std. Error**
ar1	0.7911	0.1013
ma1	-1.6594	0.1134
ma2	0.6683	0.1092
sar1	0.1343	0.0447
time'	0.9697	0.0237

3. $\boldsymbol{consumed = f(demand') + ARIMA\ error}$

Consumption can be modelled using five distinct sets of predictors; the mean performance metrics for each model have been tabulated below (table 24). We observe that with demand' as the predictor, the model to forecast consumption gives the lowest error metrics. Hence, consumption is modelled as a function of demand'.

Table 24: MAE and MAPE of regression with ARIMA model to predict consumption for 0-25 kWh bin (GF)

Predictors	MAE	MAPE
owners, season, day	113.99	23.52
users'	112.63	23.18
trans'	84.29	18.55
time'	85.45	18.66
demand'	82.91	17.37

The estimates of the coefficients are shown below (table 25). The positive coefficient of demand' indicates that as demand increases, consumption also increases. We also observe that a seasonal $(1,1,1)(2,0,0)_7$ ARIMA model is fitted on the non-stationary regression errors after carrying out non-seasonal differencing.

Table 25: Coefficients of regression with ARIMA model to predict consumption for 0-25 kWh bin (GF)

Coefficients		
Predictors	**Estimate**	**Std. Error**
ar1	0.1243	0.0488
ma1	-0.9034	0.0222
sa1	0.2316	0.0441
sar2	0.0897	0.0423
demand'	0.7498	0.0209

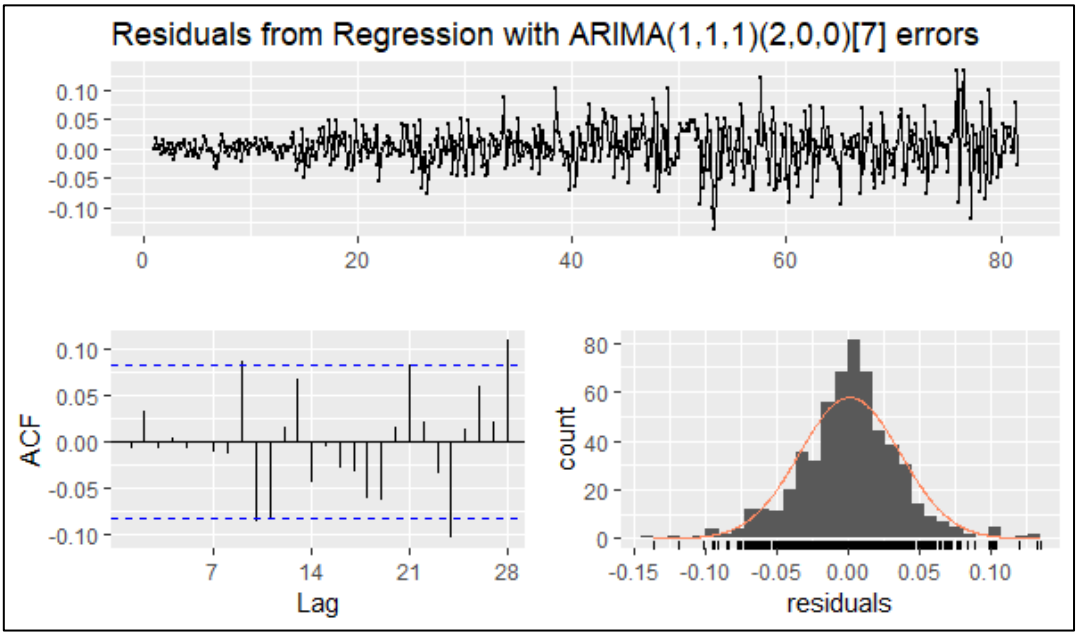

Figure 40: Residuals of regression with ARIMA model to predict consumption for 0-25 kWh bin (GF)

The ARIMA errors too have reduced autocorrelations than the earlier model built to forecast consumption using regression (figure 40). The reduced correlations, obviously, make the prediction intervals more reliable than earlier.

Comparing the two modelling approaches discussed so far, we observe that regression with ARIMA errors not only improves forecasting performance but also makes the prediction

intervals more reliable. However, we still believe that forecasting can be improved as we can modify the modelling using regression with ARIMA errors by incorporating distributed lags (Hyndman & Athanasopoulos, 2018). The modelling approach to be followed incorporates distributed lags (DL) as predictors in the regression with ARIMA errors approach.

6.4 Distributed Lag Models (with ARIMA errors)

We share the details of the distributed lag models (with ARIMA errors) for the 0-25 kWh bin. The results for other bins are present in Appendix C.

1. $\boldsymbol{users = f(owners, day, season, lagged\ values\ of\ owners) + ARIMA\ error}$

As we added lagged values of owners to the model, the error metrics over a variable origin gradually decreased for the model. With one lagged value of owners, the mean MAE and MAPE were 10.36 and 17.79 respectively. However, the minimum error metrics were observed for six lagged values of owners. The error metrics with six lagged values are shown below (table 26).

Table 26: MAE and MAPE of regression with distributed lag model to predict users for 0-25 kWh bin (GF)

Predictors	MAE	MAPE
owners, season, day, 6 lagged values of owners	8.03	14.53

A non-seasonal $(2, 0, 1)$ ARIMA model is fitted on the regression errors without any differencing prior to model fitting. Although the ARIMA errors (residuals) exhibit some degree of autocorrelation (figure 41), it is significantly less than the residuals obtained in time series regression and hence, the prediction intervals are more reliable than earlier.

The estimates of the coefficients have been tabulated below (table 27). We can observe that not all the lagged values of owners have a positive effect on the number of users. If the values

of owners at lags 1, 3 and 4 increase, the value of users also increases. However, for owners at lags 2, 5 and 6, an increase in their values would see a decrease in the value of users.

Table 27: Coefficients of regression with distributed lag model to predict users for 0-25 kWh bin (GF)

Coefficients		
Predictors	**Estimate**	**Std. Error**
ar1	1.1265	0.0756
ar2	-0.1555	0.0682
ma1	-0.7048	0.0596
owners	1.421	0.4695
autumn	-0.0058	0.0334
spring	0.0209	0.0291
summer	-0.0124	0.0317
mon	0.0446	0.0073
tue	0.0585	0.008
wed	0.0641	0.0082
thu	0.066	0.0081
fri	0.0403	0.008
sat	-0.0088	0.0073
owners-lag1	0.4508	0.6129
owners-lag2	-0.8389	0.6197
owners-lag3	0.1315	0.6267
owners-lag4	0.5477	0.6258
owners-lag5	-0.3627	0.6268
owners-lag6	-0.5506	0.48

Figure 41: Residuals of regression with distributed lag model to predict users for 0-25 kWh bin (GF)

2. $demand = f(time') + ARIMA\ error$

Modelling demand could be done using four different sets of predictors with or without their lagged values. When demand is modelled as a function of owners and their lagged values as well as season and day, the best performance is obtained with one lagged value of owners (MAE = 160.64, MAPE = 17.96). If users', trans' or time' is used as a predictor, the best performance is obtained when no lagged values are used. Adding lagged values only deteriorates the model performance for any of the subject predictors. The best performance of the four distinct models can be found below (table 28).

Table 28: MAE and MAPE of regression with distributed lag model to predict demand for 0-25 kWh bin (GF)

Predictors	MAE	MAPE
owners, season, day, 1 lagged value of owners	160.64	17.96
users'	159.8	17.07
trans'	153.29	17.02
time'	143.71	15.33

We observe that the lowest error metrics are obtained when demand is modelled as a function of time'. The estimates of the coefficients are tabulated below (table 29). A non-seasonal $(1, 1, 2)$ ARIMA model is fitted on the regression errors after non-seasonal differencing.

Table 29: Coefficients of regression with distributed lag model to predict demand for 0-25 kWh bin (GF)

Coefficients		
Predictors	**Estimate**	**Std. Error**
ar1	0.8535	0.0713
ma1	-1.7224	0.0824
ma2	0.7299	0.0789
time'	0.9538	0.0230

The residuals (ARIMA errors, as shown in figure 42) also exhibit a significant reduction in autocorrelations, leading to more reliable prediction intervals than the benchmark model.

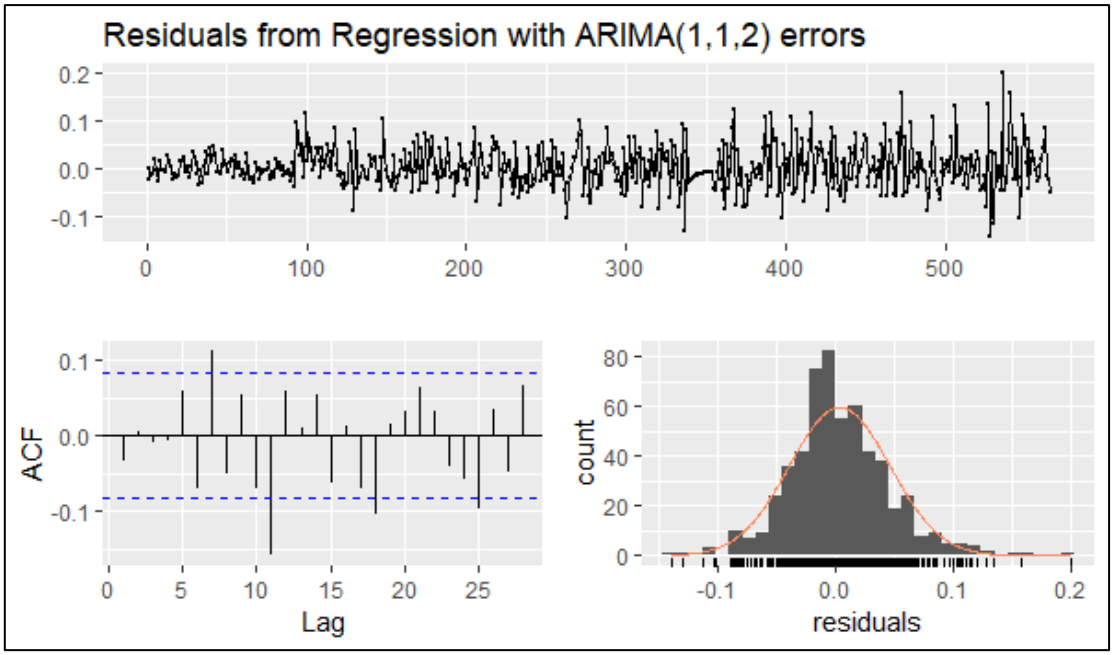

Figure 42: Residuals of regression with distributed lag model to predict demand for 0-25 kWh bin (GF)

3. $consumed = f(demand') + ARIMA\ error$

Consumption can be modelled using five different sets of predictors, which may include lagged values of predictors too. When owners, season, day are used as predictors, the best performance is obtained with six lagged values of owners (MAE = 86.92, MAPE = 19.17). With one lagged value of owners, the mean MAE and MAPE are respectively 113.41 and 23.40. As we add more lagged values of owners, the performance improves and reaches its best at six lagged values of owners as mentioned above. If users', trans', time' or demand' is used are predictor, the best performance is obtained when no lagged values are used. The mean error metrics for the best model with every possible predictor (s) are tabulated below (table 30).

Table 30: MAE and MAPE of regression with distributed lag model to predict consumption for 0-25 kWh bin (GF)

Predictors	MAE	MAPE
owners, season, day, 6 lagged values of owners	86.92	17.17
users'	91.47	19.62
trans'	83.79	18.43
time'	84.96	18.53
demand'	82.5	17.27

We observe that the best performance is obtained when consumption is modelled as a function of demand'. Estimates of the model coefficients are shown below (table 31).

Table 31: Coefficients of regression with distributed lag model to predict consumption for 0-25 kWh bin (GF)

Coefficients		
Predictors	**Estimate**	**Std. Error**
ar1	0.0190	0.0598
ar2	-0.0247	0.0516
ar3	-0.1230	0.0499
ar4	-0.1073	0.0506
ma1	-0.8120	0.0482
demand'	0.7680	0.0212

A non-seasonal $(4, 1, 1)$ ARIMA model is fitted on the regression errors after carrying out non-seasonal differing of the errors. While we do see some degree of autocorrelation in the residuals (figure 43), it is significantly less than what we observed in the case of the benchmark model.

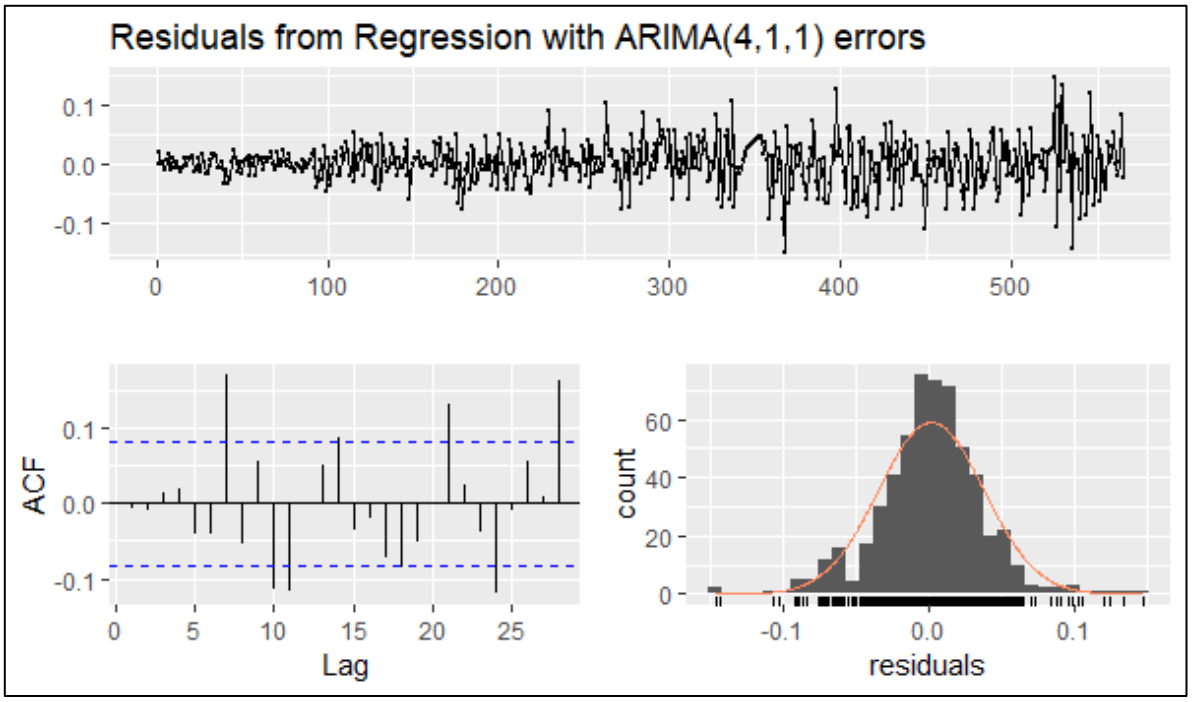

Figure 43: Residuals of regression with distributed lag model to predict consumption for 0-25 kWh bin (GF)

It might be argued that models to estimate consumption of energy for both regression (with ARIMA errors) with distributed lags and regression with ARIMA errors use demand' as the predictor, with the former model using no distributed lags. In that case, how did the performance improve? The answer lies in the way we built up to the final model. In case of regression with ARIMA errors, consumption was modelled as a function of demand', demand was modelled as a function of time', and time was modelled as a function of owners, day and season. The difference lies in the way time was modelled in case of regression (with ARIMA errors) with distributed lags. It was modelled as a function of transactions' with no distributed lags, but in turn, was modelled as a function of owners with lags, day and season, which delivered much better performance. Hence, the subsequent estimates also improved, leading to the final model being better than the one built using ARIMA errors only.

6.5 LSTM Networks

We set up a basic architecture and then gradually tuned a few hyperparameters to improve the performance of LSTM networks. The basic architecture is described below.

- *Number of hidden layers (LSTM layers)* = 2
- *Activation function* = Rectified Linear Unit
- *Loss function* = Mean Squared Error
- *Optimiser* = RMSprop
- *Metrics* = MAE and MAPE

We kept the aforementioned hyperparameters fixed while tuning the others. We tuned the following hyperparameters to improve the performance of the LSTM networks.

- *Number of neurons in each LSTM layer*
- *Epochs*
- *Batch size*
- *Dropout and recurrent dropout rates*
- *Learning rate*

We initially tested eleven different combinations of neurons for the two LSTM networks to forecast the number of active users as a function of EV owners, day of the week and season of the year. The details of the eleven configurations are shown below (table 32).

Table 32: Initial eleven configurations of hidden layers (GF)

Configuration	Neurons (Layer 1)	Neurons (Layer 2)
1	20	10
2	20	20
3	30	20
4	30	30
5	40	30
6	40	40
7	100	100
8	50	40
9	30	10
10	40	10
11	40	20

We tuned the epochs for three different values: 50, 100 and 150. We observe from the plots shown below (figures 44 and 45) that except for networks with configurations 1 and 2, all other networks exhibited lowest error metrics (MAE and MAPE) at epochs = 50. Besides, the error metrics for configurations 1 and 2 were higher than any other configurations. Hence, in further analyses, we dropped the configurations 1 and 2. We also tested the networks at epochs = 25, but the error metrics increased, suggesting that epochs should be set to 50.

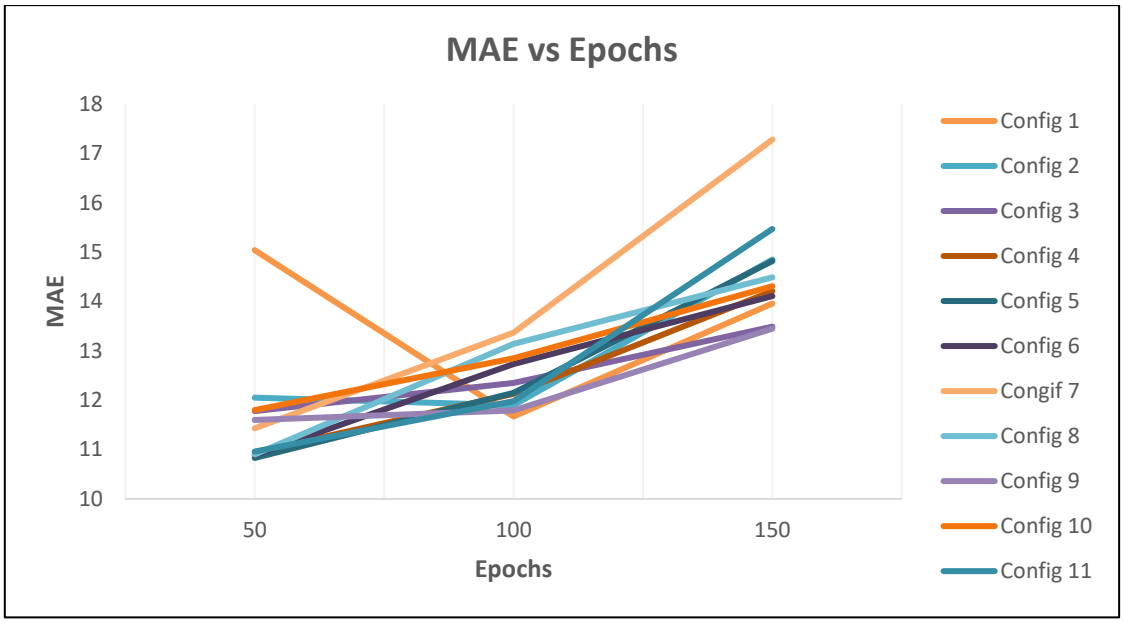

Figure 44: MAE vs epochs

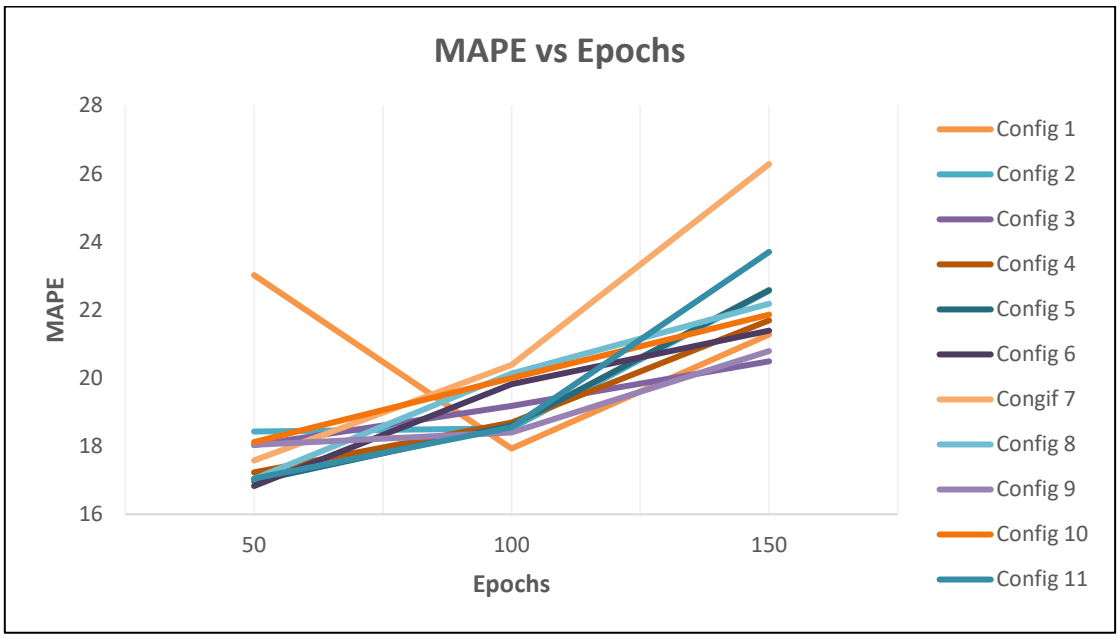

Figure 45: MAPE vs epochs

We also tested the configurations for three different batch sizes (1, 7, and the default value of 32) during model training. We found that while the default number of batch sizes (= 32) gave decent performance, the error metrics increased when the batch size was set to 1.

However, for a batch size of 7, the error metrics were found to be lowest. As such the batch size was set to 7. We also observed that the networks gave a decent performance without any dropout of neurons in both the LSTM layers. When both dropout and recurrent dropout were set to 0.2, the performance deteriorated. However, when the dropouts were set to 0.1 each, the performance was found to be the best than the other two combinations of dropouts. Hence, both dropout and recurrent dropout were set to 0.1. Moreover, the learning rate was also adjusted to 0.05 and 0.005 (default value = 0.001); however, in both the cases, the performance was found to deteriorate compared to the case when the learning rate was left to its default value. In further analyses, learning rate, epochs, dropout, and recurrent dropout were set to 0.001, 50, 0.1 and 0.1 respectively.

With fixed values of the learning rate, epochs, dropout and recurrent dropout (as mentioned above), we again tested twelve configurations and found that the lowest error metrics were obtained for the configuration 9 (table 33, figures 46 and 47). We observed that as we increased the number of neurons in each layer, the error metrics decreased initially. However, after configuration 9, where each layer had 50 neurons, the error metrics increased on increasing the number of neurons. Hence, we fixed the number of neurons in each LSTM layer to 50 for all the models we built henceforth.

Table 33: Final twelve configurations of hidden layers (GF)

Configuration	Neurons (Layer 1)	Neurons (Layer 2)	MAE	MAPE
1	30	10	11.88	18.37
2	30	20	11.85	18.67
3	30	30	11.09	17.52
4	40	10	11.96	18.85
5	40	20	11.11	17.86
6	40	30	10.53	16.64
7	40	40	9.73	15.45
8	50	40	9.49	15.2
9	50	50	8.37	13.64
10	60	50	9.28	14.93
11	60	60	8.58	14.04
12	100	100	9.24	15.43

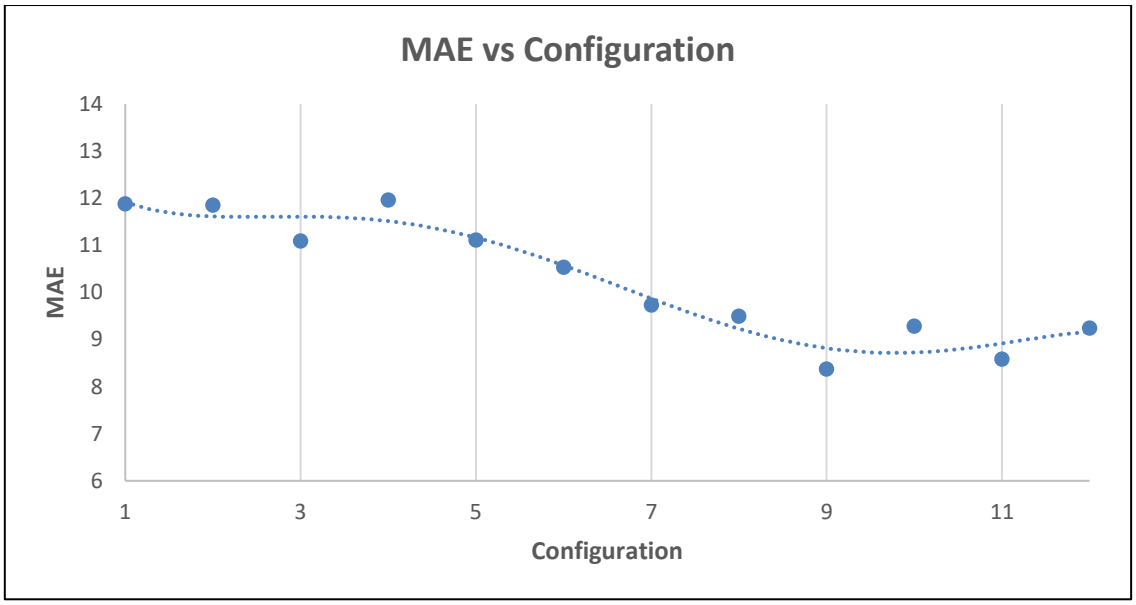

Figure 46: MAE vs configurations

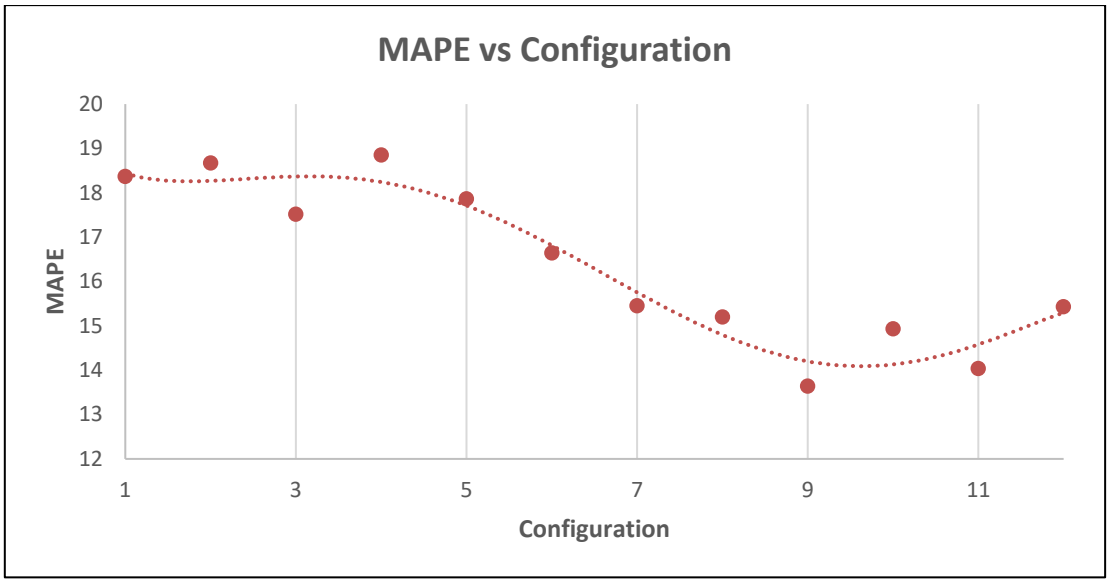

Figure 47: MAPE vs configurations

The final architecture of the LSTM network we used to build all the models can be found below.

- *Number of hidden layers (LSTM layers)* = 2
- *Activation function* = Rectified Linear Unit
- *Loss function* = Mean Squared Error
- *Optimiser* = RMSprop
- *Metrics* = MAE and MAPE
- *Number of neurons in each LSTM layer* = 50
- *Epochs* = 50
- *Batch size* = 7
- *Dropout and recurrent dropout rates* = 0.1
- *Learning rate* = 0.001

As in the case of previously discussed algorithms, we also tested all combinations of features for each response variable. We present the details of error metrics for all the models for *users*, *demand* and *consumed* for the 0-25 kWh and present the details of error metrics of only the best models for other bins in Appendix C.

1. $\boldsymbol{users = f(owners, day, season)}$

The error metrics are shown below (table 34).

Table 34: MAE and MAPE of LSTM network to predict users for 0-25 kWh bin (GF)

Predictors	MAE	MAPE
owners, season, day	8.37	13.64

2. $\boldsymbol{demand = f(owners, day, season)}$

The error metrics for predicting the total connected load (demand) are shown below (table 35). We observe that the best model is obtained when owners, season and day are used to predict the total connected load.

Table 35: MAE and MAPE of LSTM network to predict demand for 0-25 kWh bin (GF)

Predictors	MAE	MAPE
owners, season, day	128.11	12.36
users'	186.32	17.13
trans'	187.42	17.17
time'	171.21	15.75

3. $\boldsymbol{consumed = f(owners, day, season)}$

The error metrics for predicting the consumption of energy (consumed) are shown below (table 36). We observe that the best model is obtained when owners, season and day are used to predict the consumption.

Table 36: MAE and MAPE of LSTM network to predict consumption for 0-25 kWh bin (GF)

Predictors	MAE	MAPE
owners, season, day	59.18	11.94
users'	101.51	18.63
trans'	104.71	18.74
time'	91.68	16.66
demand'	83.19	15.37

7. Conclusions

7.1 Summary of Results

We observed that deep learning networks (LSTMs) delivered the best performance over a variable origin among all algorithms. While the objective of the project was to predict future energy load caused by EV charging on local distribution networks, we also developed models to estimate the number of active users who would charge their EVs every day and to estimate the total connected load (demand) on the distribution networks caused by all the EV transactions. While the models on predicting active users were meant to give an estimate of the mean number of users out of the total population of EV owners who would charge their EVs per day, the models on predicting demand was meant to indicate an upper cap on the energy requirements caused by multiple transactions of EV charging. However, in subsequent sections we will focus our discussion only on the actual business objective of EA Technology: predicting energy consumption due to EV charging. A summary of the performance of the relevant models (for all the bins) is presented below (tables 37 to 40). It is important to note that the best performing model across different algorithms might have used different set of predictors.

Table 37: Performance comparison of all models to predict consumption for 0-25 kWh bin (GF)

Models (0-25 kWh)	MAE	MAPE
Time Series Regression	110.93	24.6
Regression with ARIMA errors	82.91	17.37
Regression (with ARIMA errors) with Distributed Lags	82.5	17.27
LSTM networks	59.18	11.94

For the model in 0-25 kWh bin (table 37), we see an improvement of 12.66 percentage points in MAPE as we move from the benchmark model (Time Series Regression) to LSTM networks.

Besides, the MAE improves by a value of 51.75 (equivalent to 46.65%) over the benchmark. A similar analysis is also true for other bins. However, at this stage it is important to understand that in future, DNOs will be dealing with thousands of EV owners and hence, more users and transactions every day, thereby leading to high consumption of energy due to EV charging. Hence, keeping in mind the future scenario, relative error metrics would be a more suitable performance evaluation measure than absolute metrics. So, we would be focussed on MAPE in the discussions to follow. Comparing the performance of all algorithms for the bins, we observed that deep learning method (LSTM networks), with little hyperparameter tuning, outperformed all the other algorithms.

Table 38: Performance comparison of all models to predict consumption for 26-50 kWh bin (GF)

Models (26-50 kWh)	MAE	MAPE
Time Series Regression	91.8	24.26
Regression with ARIMA errors	87.1	22.89
Regression (with ARIMA errors) with Distributed Lags	86.88	22.82
LSTM networks	75.79	18.13

Table 39: Performance comparison of all models to predict consumption for 51-75 kWh bin (GF)

Models (51-75 kWh)	MAE	MAPE
Time Series Regression	64.77	40.49
Regression with ARIMA errors	65.04	43.67
Regression (with ARIMA errors) with Distributed Lags	66.82	42.72
LSTM networks	64.34	36.9

Table 40: Performance comparison of all models to predict consumption for 76-100 kWh bin (GF)

Models (76-100 kWh)	MAE	MAPE
Time Series Regression	49.23	150.01
Regression with ARIMA errors	40.17	110.77
Regression (with ARIMA errors) with Distributed Lags	40.07	109.06
LSTM networks	42.65	93.68

7.1.1 Scenario-based Forecasting (Case of Lancaster, UK in 2040)

We consider a future scenario in the city of Lancaster (UK) in 2040. Detailed results of one such scenario-based forecasting is presented below (summarised results outlined in section 1.2). By 2040, the projected number of EVs in the UK is 36 million (Evans, 2018). Besides, by 2040, the UK and Lancaster population are projected to rise to 72 million and 0.146043 million respectively (Nash, 2017; Lancashire-County-Council, n.d.). Based on the aforementioned statistics, we estimate the number of EV owners in Lancaster under the following assumption.

1. Distribution of EVs in the UK will be uniform
2. Distribution of EVs across battery capacity bins will be same as observed in EV trials

Estimated number of EVs in Lancaster in 2040 = $\left(\frac{36}{72}\right) \times 146043$ = 73022

The distribution of EVs (73022) across various battery capacity bins is tabulated below.

Table 41: Distribution of EVs across battery capacity bins in Lancaster in 2040

Battery Capacity Bins (kWh)	Count of EVs in trials	% of EVs (p) among total EVs (= 300) in trials	Estimate of EVs in 2040 (= 73022 * p/100)
0-25	163	54.33	39673
26-50	87	29.00	21176
51-75	35	11.67	8522
76-100	15	5.00	3651

Based on the estimates of EVs in Lancaster in 2040, the LSTM networks forecast the following kWh consumption (tabulated below) during a given week (from 24/12/2040 to 30/12/2040) in winter season in 2040.

Table 42: Scenario-based forecasting of kWh consumption per day in winter season in Lancaster in 2040

Day	kWh Consumed (0-25 bin)	kWh Consumed (26-50 bin)	kWh Consumed (51-75 bin)	kWh Consumed (76-100 bin)
Monday	106156.1	99035.5	42616.0	18455.0
Tuesday	106145.2	99036.5	42622.3	18464.7
Wednesday	106134.8	99018.9	42614.5	18452.2
Thursday	106157.5	99038.8	42619.7	18456.5
Friday	106163.0	99048.1	42622.0	18461.7
Saturday	106163.5	99044.1	42628.1	18457.3
Sunday	106142.5	99034.1	42620.1	18459.3

We observe that for extremely higher number of EVs, the effect of day in determining the energy consumption is apparently insignificant as the observed kWh consumption is very high and vary by marginal values across days. However, the LSTM networks developed are expected to be robust and work well for local distribution networks with relatively smaller number of EV owners (a few hundreds). Hence, the models should be leveraged to forecast energy consumption for smaller clusters of EV owners as is the case of actual local distribution networks, to ensure more reliable forecasts. However, such forecasts (tabulated above) do indicate the quantum of stress that distribution networks would have in case EVs are allowed to charger without any constraints.

7.2 Discussion

While we found that LSTM networks are the way forward to estimate the consumption of energy based on the current objective, we also realised that the deep learning models can be improved to incorporate a more realistic scenario with better predictive accuracy. The following comments have summarised the constraints in the project with plausible suggestions on how to improve predictive performance in the future.

- We observed in section 4.3.8 that the response variable (consumption of energy) had significant autocorrelations with its lagged values, thereby meaning that lagged values of the response variable also carried information that could have been used to forecast the response variable. Since the objective was not a conventional forecasting problem (we didn't have to forecast ahead of time for a given forecasting horizon based on trailing data), we couldn't leverage the autocorrelations among lagged values of the response variable, leading to loss of relevant information in forecasting the response variable. This adversely affected the predictive accuracy of the models. A plausible solution to take on this issue would be either real-time or near time forecasting of energy consumption. In both cases, autoregressive models can be leveraged to incorporate the strong correlations among the lagged values of our target variable. In real-time forecasting, we forecast the future value of energy consumption based on a set of predictors as well as the past values of energy consumption (lagged values) as EV charging takes place in real-time. In near time forecasting, we would also forecast energy consumption but not in real-time, i.e., we would collect data on EV charging in a local distribution network and predict the future consumption of energy for a given forecasting horizon; this approach would help in regulating demand management system in future. In addition, we can also leverage the temporal aggregation of forecasts to get long-term forecasts (Athanasopoulos, et al., 2017).
- We discussed in section 5.1 that the scenario-based forecast of energy consumption was seriously constrained by the future information available to DNOs. This compelled us to propose a nested modelling approach so that the feature space of scenario-based data

could be extended to match the training data (actual data available). However, in nested modelling approach, errors get multiplied if we use forecasts of one variable to forecast another variable, leading to poor predictive accuracy. Hence, if the breadth of information available to DNOs could be increased, it would likely improve the predictive accuracy.

- In the project, we transformed the raw transaction data to day-wise time series data for analyses and modelling. However, it could be also possible to work with high-frequency time series data (hour-wise time series) and aggregate results to get day-wise predictions. Working with high-frequency data may provide higher variability in the available information, leading to better algorithm training. However, it is a mere possibility and needs to carefully analysed.
- It can be argued that analyses and modelling can also be done on the raw transaction data (after data cleaning) and the results can then be suitably aggregated to get day-wise predictions. We observed in section 4.3.1 that the data had missing values which were imputed to obtain the regular time series. However, the problem of working with transition data is that imputing the missing values in the data is a challenging task. This is because the charging events are randomly scattered within a day and imputing a missing charging event with a reasonably accurate timestamp would be really challenging.
- We observed in section 4.2 that k-means clustering suggested the use of a minimum of three bins and adding any fourth bin didn't lead to any additional variability across the clusters. However, we chose to work with four bins (as agreed with EA Technology) to develop predictive models. However, it would also be interesting to identify what hierarchical clustering has to suggest. After comparing the results obtained from both clustering approaches, different combinations of clusters can be tried and tested to see if the results improved or deteriorated.
- We identified in section 5.1 that the number of users charging their EVs was very small in the initial phase of the trials, leading to a very small consumption of energy. High absolute errors (poor predictions) on such smaller values amplify the percentage errors (as the denominator is a small number) more than they do on larger values (as the denominator

is a large number). A high value of percentage error can be misleading as it fails to capture the real picture and, in turn, gives a false alarm caused by the poor forecasts of smaller values. Hence, to have a more realistic understanding of the predictive accuracy of the models, we need to work with data which is more realistic. Dropping observations that do not reflect the real-world scenario might help in getting rid of the false alarms.

- Hyperparameter tuning of deep learning networks is time-consuming and varies from one case to another, i.e., a good architecture to predict a given response variable might not be good enough to predict another response variable. In our case, we had tuned a few hyperparameters to get good predictive accuracy for the model on users. The architecture also delivered the best results among all the different family of algorithms for the model on the consumption of energy. However, it might be possible that the performance of the LSTM networks to forecast energy consumption could be improved by using different architectures and more hyperparameter tuning. Besides, the hyperparameters were tuned for the 0-25 kWh bin and the same architecture was used for all the other bins. It could be possible that different architectures work better for other bins.

7.3 Concluding Remarks

In the project, we developed predictive models to forecast the daily energy consumption caused by EV charging. The models were developed keeping in view of the information that would likely be available to the DNOs in the future. We observed that the energy consumption varied with day of the week and season of the year, with highest median consumption during winter and mid-week. However, it is likely that there are more contributors to the variation in energy consumption, and gathering more information and data would help in better understanding of the variability of energy consumption, leading to the identification of more important predictors for forecasting consumption of energy. We proposed deep learning models as the way forward and even suggested plausible solutions to counter the constraints we faced during the project for enhancing the predictive accuracy. It would be interesting to

see in future work how deep learning can be leveraged to build forecast models with minimal errors (or high predictive accuracy) that serve the business objective and enable the DNOs to make better, informed decisions on demand management.

References

Athanasopoulos, G., Hyndman, R. J., Kourentzes, N. & Petropoulos, F., 2017. Forecasting with Temporal Hierarchies. *European Journal of Operational Research,* Volume 262, pp. 60-74.

Bi, J., Wang, Y., Sun, S. & Guan, W., 2018. Predicting Charging Time of Battery Electric Vehicles Based on Regression and Time-Series Methods: A Case Study of Beijing. *Energies,* 11(5).

Chollet, F. & Allaire, J. J., 2018. *Deep Learning with R.* s.l.:Manning.

Cleveland, R. B., Cleveland, W. S., McRae, J. E. & Terpenning, I., 1990. STL: A Seasonal-Trend Decomposition Procedure Based on Loess. *Journal of Official Statistics,* 6(1), pp. 3-33.

De Cauwer, C., Van Mierlo, J. & Coosemans, T., 2015. Energy Consumption Prediction for Electric Vehicles Based on Real-World Data. *Energies,* Volume 8, pp. 8573-8593.

Driving-Electric, 2019. *Driving Electric.* [Online]
Available at: https://www.drivingelectric.com/news/678/electric-car-sales-uk-2019-stars-110-jump
[Accessed 30 06 2019].

Dudek, E., 2017. *Algorithm Development and Testing: Electric Nation,* s.l.: Western Power Distribution: Innovation.

Electric-Nation, 2019. *Summary of the Findings of the Electric Nation Smart Charging Trial,* s.l.: s.n.

Elexon, n.d. *Elexon - Glossary Term: BSC Season.* [Online]
Available at: https://www.elexon.co.uk/glossary/bsc-season/
[Accessed 15 January 2019].

Evans, S., 2018. *CarbonBrief.* [Online]
Available at: https://www.carbonbrief.org/rise-uk-electric-vehicles-national-grid-doubles-2040-forecast
[Accessed 2019].

Foley, A., Tyther, B., Calnan, P. & Ó Gallachóir, B., 2013. Impacts of Electric Vehicle charging under electricity market operations. *Applied Energy,* Volume 101, pp. 93-102.

Godfrey, B., 2016. *Car Connect: Balancing Act Conference,* s.l.: Western Power Distribution (Innovation).

Green II, R. C., Wang, L. & Alam, M., 2011. The impact of plug-in hybrid electric vehicles on distribution networks: A review and outlook. *Renewable and Sustainable Energy Reviews,* Volume 15, pp. 544-553.

Higgins, A., Paevere, P., Gardner, J. & Quezada, G., 2012. Combining choice modelling and multi-criteria analysis for technology diffusion: An application to the uptake of electric vehicles. *Technological Forecasting & Social Change,* Volume 79, pp. 1399-1412.

Hmamouche, Y., Casali, A. & Lakhal, L., 2017. *A Causality Based Feature Selection Approach for Multivariate Time Series Forecasting.* Barcelona, Spain, DBKDA 2017, The Ninth International Conference on Advances in Databases, Knowledge, and Data Applications.

Hochreiter, S., 1998. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* 06(02), pp. 107-116.

Hoed, v. d. R., Helmus, J., Vries, R. d. & Bardok, D., 2013. *Data analysis on the public charge infrastructure in the city of Amsterdam.* Barcelona, Spain, World Electric Vehicle Symposium and Exhibition (EVS27) 2013 - IEEE.

Hyndman, R., 2014. Better ACF and PACF plots, but no optimal linear prediction. *Electronic Journal of Statistics.*

Hyndman, R. J. & Athanasopoulos, G., 2018. *Forecasting: Principles and Practice.* 2nd ed. s.l.:OTexts.

Hyndman, R. J. & Khandakar, Y., 2008. Automatic Time Series Forecasting: The forecast Package for R. *Journal of Statistical Software,* 27(3).

Jiming, H. et al., 2017. *A Review of Demand Forecast for Charging Facilities of Electric Vehicles.* Shanghai, China, IOP Conference Series: Materials Science and Engineering 199 012040.

Khoo, Y. B., Wang, C.-H., Paevere, P. & Higgins, A., 2014. Statistical modeling of Electric Vehicle electricity consumption in the Victorian EV Trial, Australia. *Transportation Research Part D,* Volume 32, pp. 263-277.

Lancashire-County-Council, n.d. *Population Projections (Lancashire County Council).* [Online] Available at: https://www.lancashire.gov.uk/lancashire-insight/population-and-households/population/population-projections/
[Accessed 2019].

López, K. L., Gagné, C. & Gardner, M.-A., 2018. Demand-Side Management using Deep Learning for Smart Charging of Electric Vehicles. *IEEE Transactions on Smart Grid.*

Moon, H., Park, S. Y., Jeong, C. & Lee, J., 2018. Forecasting electricity demand of electric vehicles by analyzing consumers' charging patterns. *Transportation Research Part D,* Volume 62, pp. 64-79.

Nash, A., 2017. *National Population Projections: 2016-based statistical bulletin.* [Online] Available at: https://www.ons.gov.uk/peoplepopulationandcommunity/populationandmigration/populationproje

ctions/bulletins/nationalpopulationprojections/2016basedstatisticalbulletin
[Accessed 2019].

Neaimeh, M. et al., 2015. A probabilistic approach to combining smart meter and electric vehicle charging data to investigate distribution network impacts. *Applied Energy,* Volume 157, pp. 688-698.

Neaimeh, M. et al., 2015. A probabilistic approach to combining smart meter and electric vehicle charging data to investigate distribution network impacts. *Applied Energy,* Volume 157, pp. 688-698.

OGL, 2019. *2018 UK GREENHOUSE GAS EMISSIONS, PROVISIONAL FIGURES,* s.l.: OGL (Department for Business, Energy & Industrial Strategy).

Olah, C., 2015. *Understanding LSTM Networks.* [Online]
Available at: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
[Accessed 25 06 2019].

Paevere, P. et al., 2014. Spatio-temporal modelling of electric vehicle charging demand and impacts on peak household electrical load. *Sustainability Science,* 9(1), pp. 61-76.

Polar-Plus, n.d. *Polar Plus.* [Online]
Available at: https://polar-network.com/uk-course-60-new-cars-electric-2030/
[Accessed 30 06 2019].

Rahajoe, D. A., Winarko, E. & Guritno, S., 2017. A Hybrid Method for Multivariate Time Series Feature Selection. *IJCSNS International Journal of Computer Science and Network Security,* Volume 17.

Robinson, D., 2018. *Electric vehicles and electricity,* s.l.: The Oxford Institute of Energy Studies.

Tyralis, H. & Papacharalampous, G., 2017. Variable Selection in Time Series Forecasting Using Random Forests. *Algorithms,* Volume 10.

Wang, J., Besselink, I. & Nijmeijer, H., 2015. Electric vehicle energy consumption modelling and prediction based on road information. *World Electric Vehicle Journal,* Volume 7, pp. 447-458.

Wang, J., Besselink, I. & Nijmeijer, H., 2016. Online prediction of battery electric vehicle consumption. *World Electric Vehicle Journal,* Volume 8, pp. 213-224.

Wells, A. & Dale, M., 2017. *NIA Major Project Progress Report: Electric Nation (CarConnect) ,* s.l.: Wester Power Distribution: Innovation.

Wickham, H., 2014. Tidy Data. *Journal of Statistical Software,* 59(10).

Witten, D., James, G., Tibshirani, R. & Hastie, T., 2013. *An Introduction to Statistical Learning with Applications in R.* s.l.:Springer.

Xydas, E. et al., 2016. A data-driven approach for characterising the charging demand of electric vehicles: A UK case study. *Applied Energy,* Volume 162, pp. 763-771.

Xydas, E. S. et al., 2013. *Electric Vehicle Load Forecasting using Data Mining Methods.* London, UK, IET Hybrid and Electric Vehicles Conference 2013 (HEVC 2013).

Yang, K., Yoon, H. & Shahabi, C., 2005. *A Supervised Feature Subset Selection Technique for Multivariate Time Series.* Hanoi, Vietnam, PAKDD'05 Proceedings of the 9th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining.

Yoon, H. & Shahabi, C., 2006. *Feature Subset Selection on Multivariate Time Series with Extremely Large Spatial Features.* Hong Kong, China, Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06).

Zap-Map, 2019. *Zap-Map.* [Online]
Available at: https://www.zap-map.com/tools/home-charging-calculator/
[Accessed 30 06 2019].

Zheng, A. & Casari, A., 2018. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists.* 1st ed. s.l.:O'Reilly.

Appendices

Appendix A

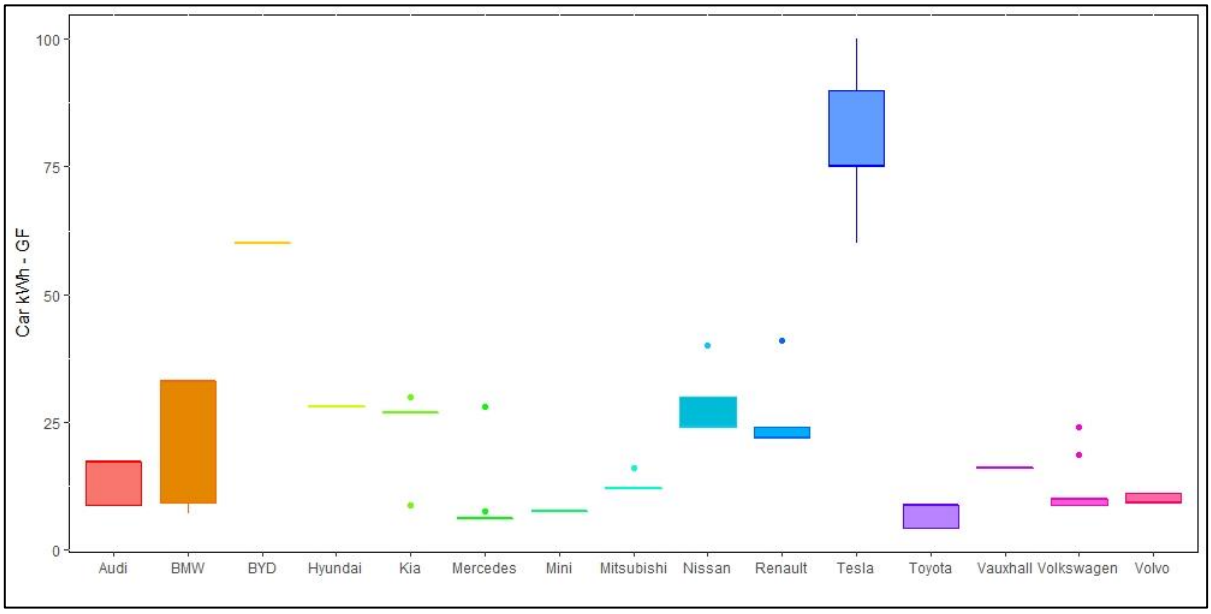

Figure 48: Battery rating vs car brand (GF)

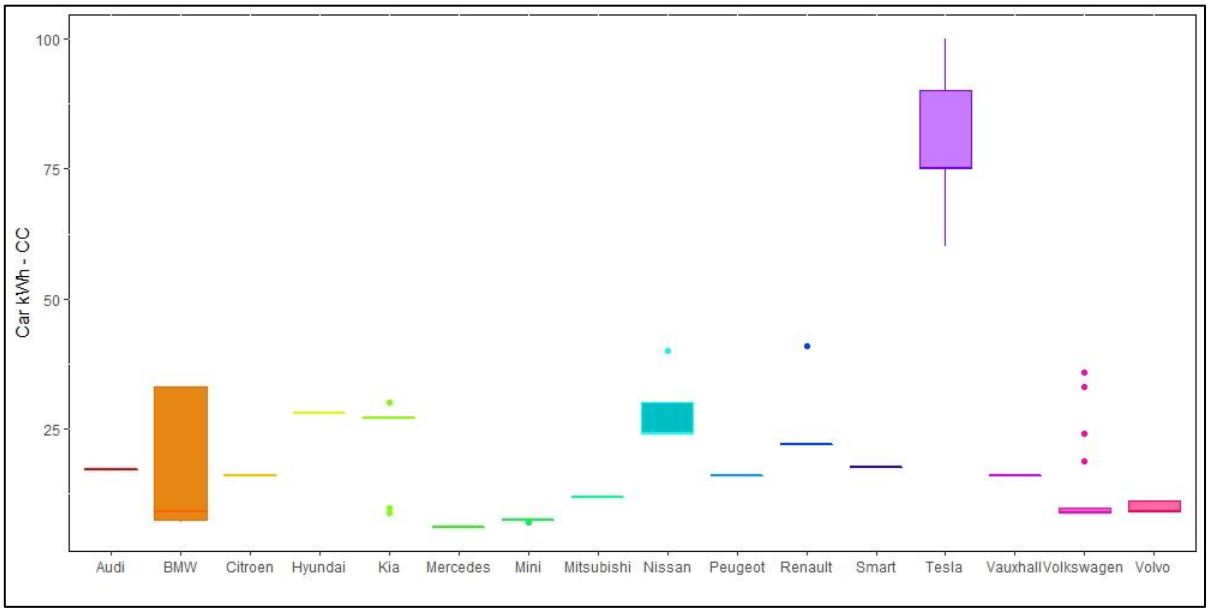

Figure 49: Battery rating vs car brand (CC)

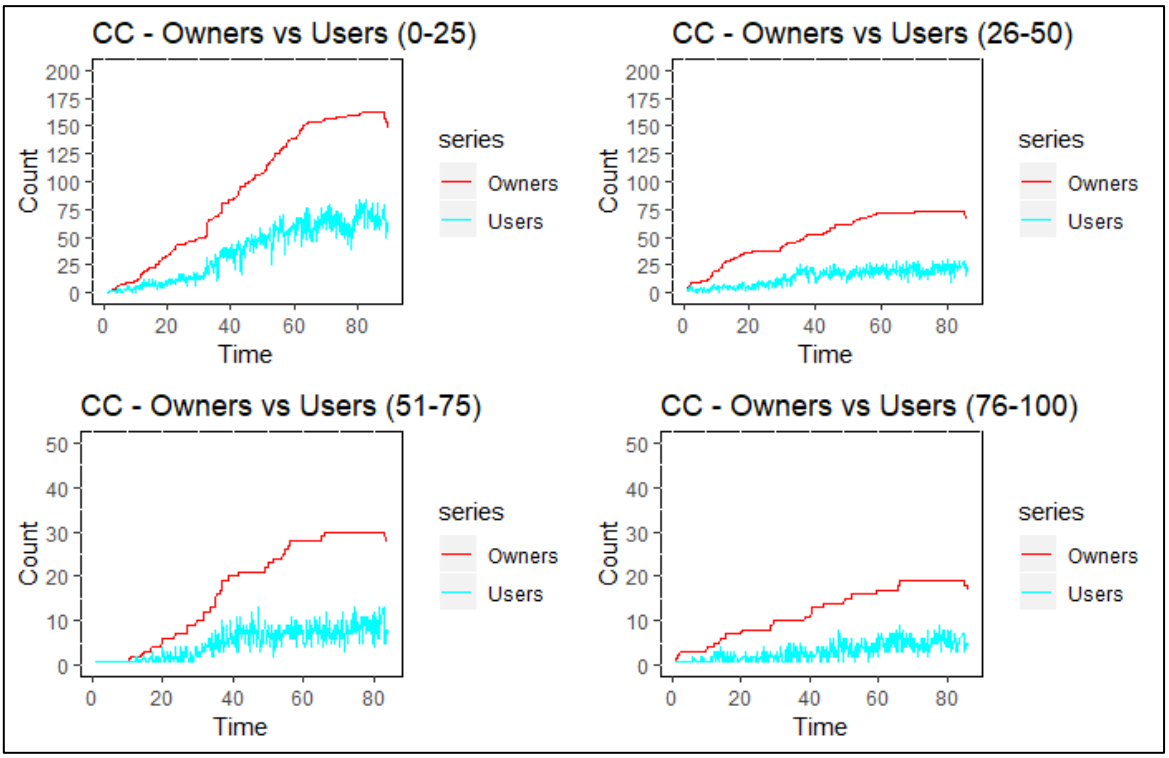

Figure 50: Owners vs users for all bins (CC)

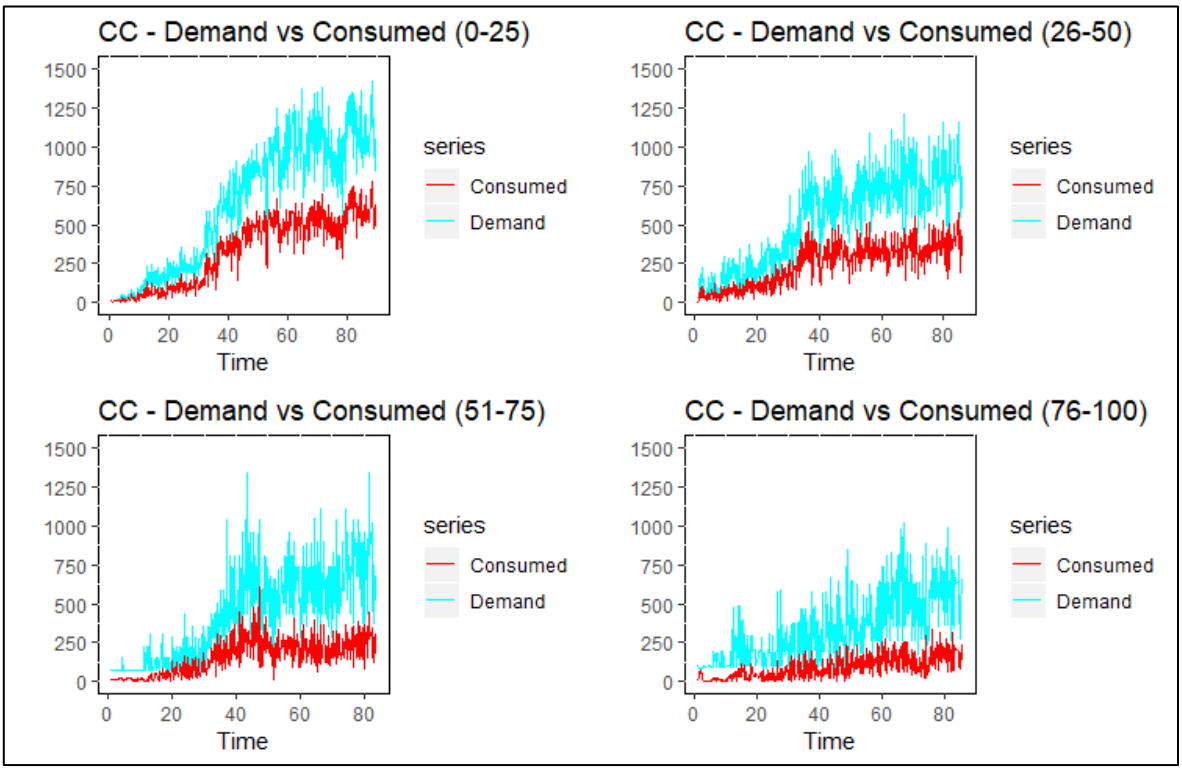

Figure 51: Demand vs consumption for all bins (CC)

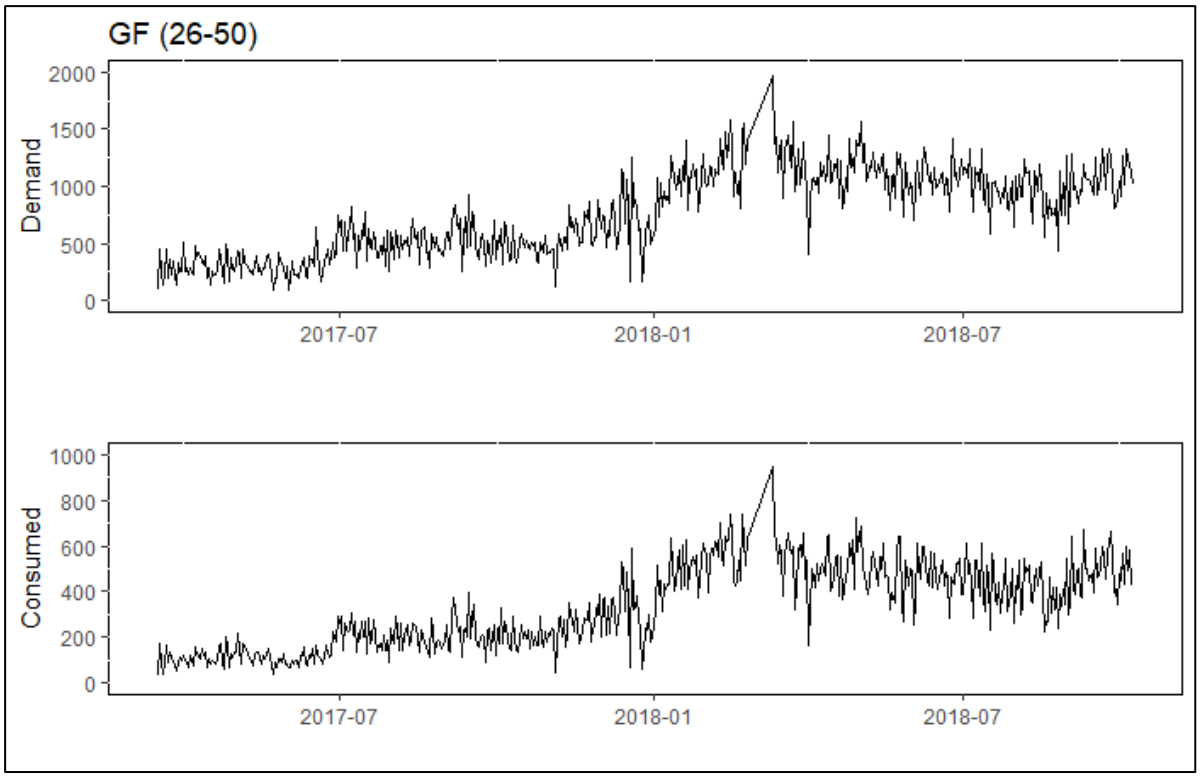

Figure 52: Demand vs consumption for 26-50 kWh (GF)

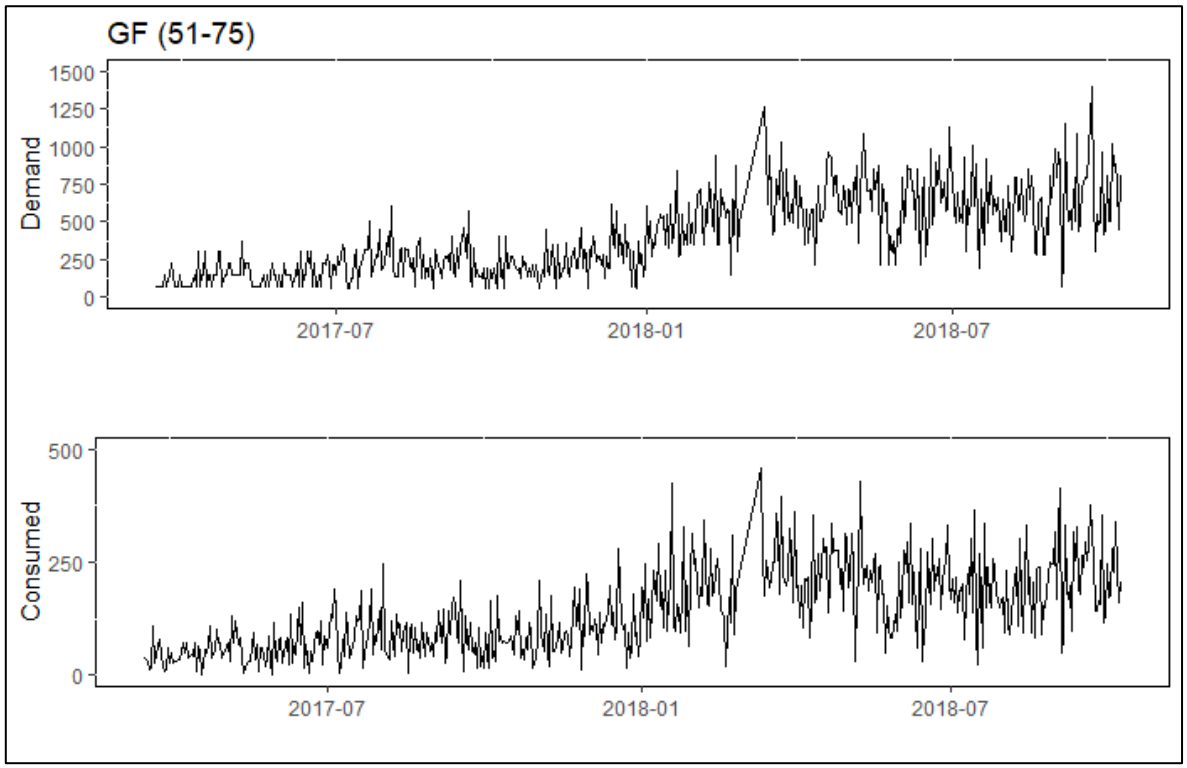

Figure 53: Demand vs consumption for 51-75 kWh (GF)

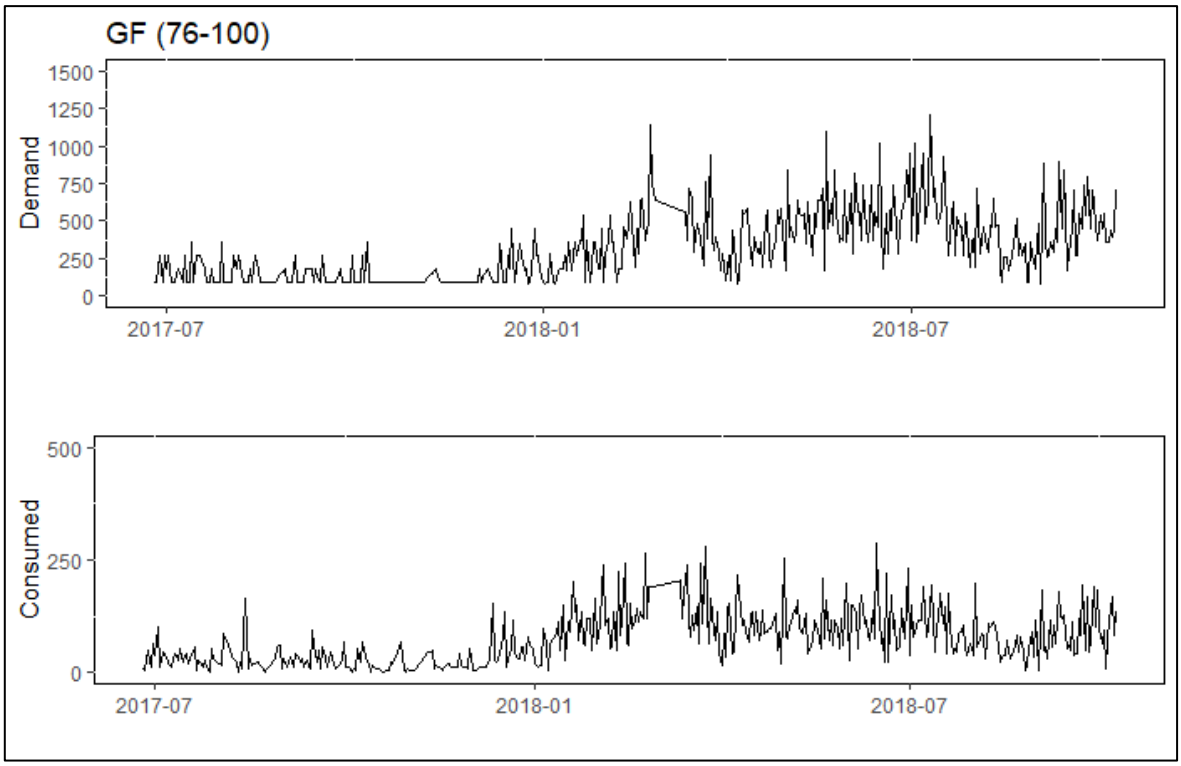

Figure 54: Demand vs consumption for 76-100 kWh (GF)

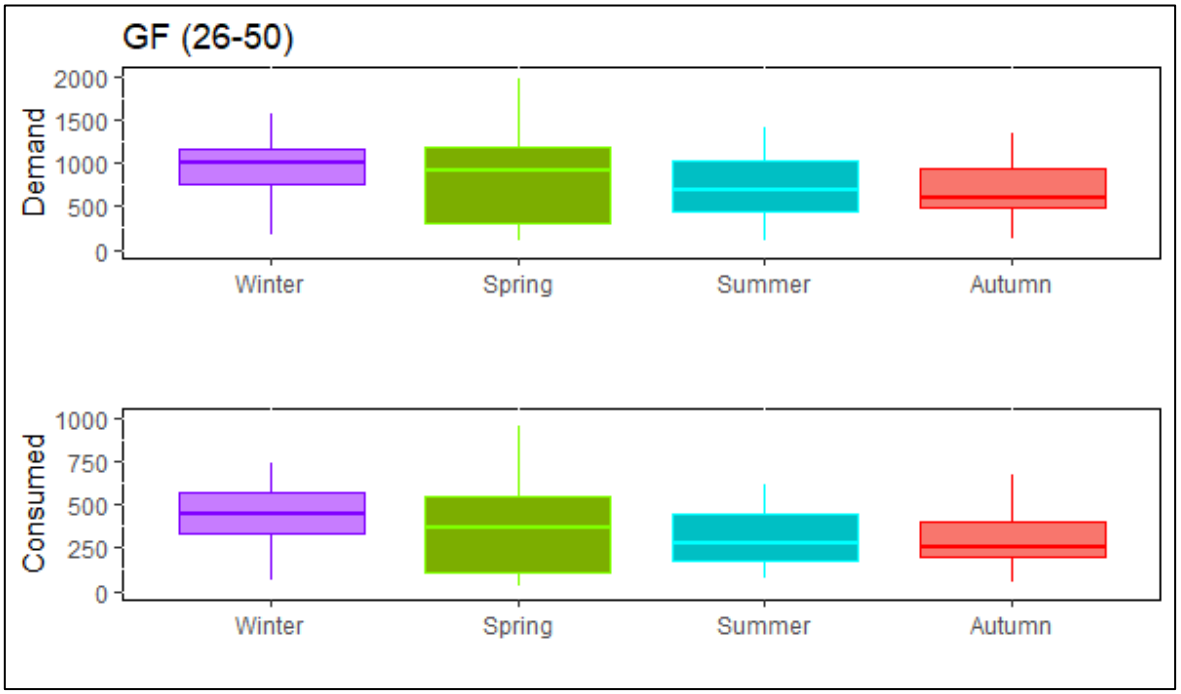

Figure 55: Demand and consumption vs season of the year for 26-50 kWh bin (GF)

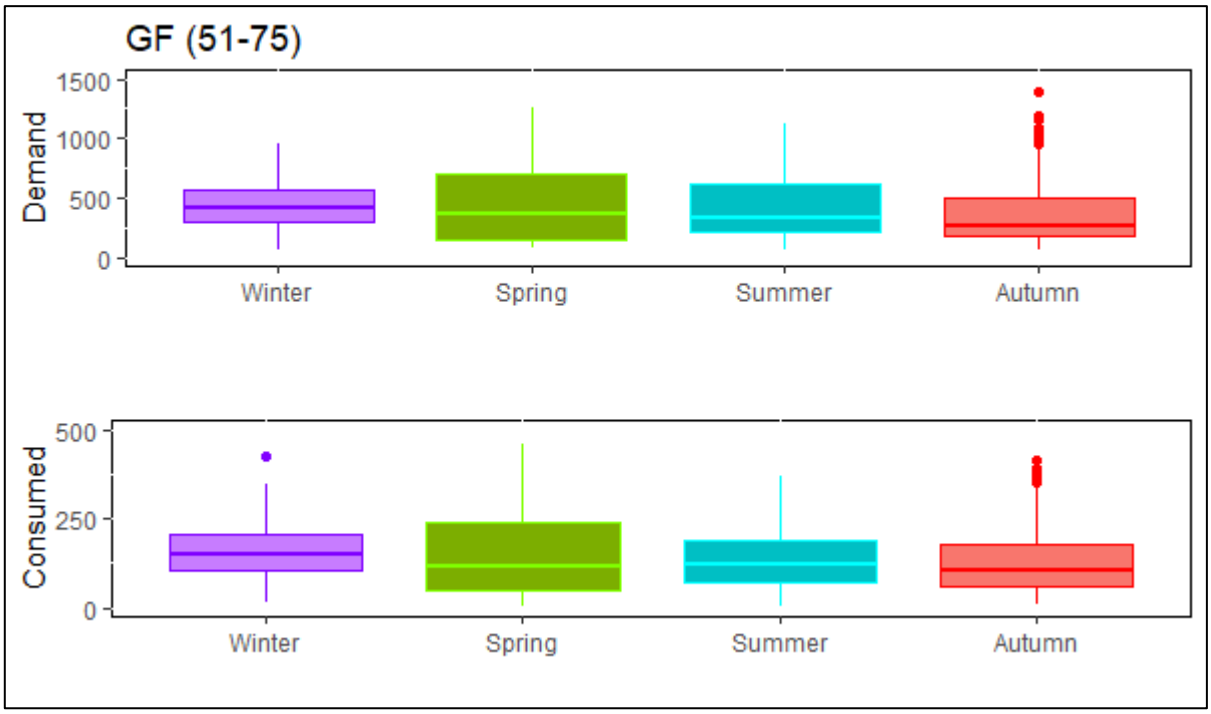

Figure 56: Demand and consumption vs season of the year for 51-75 kWh bin (GF)

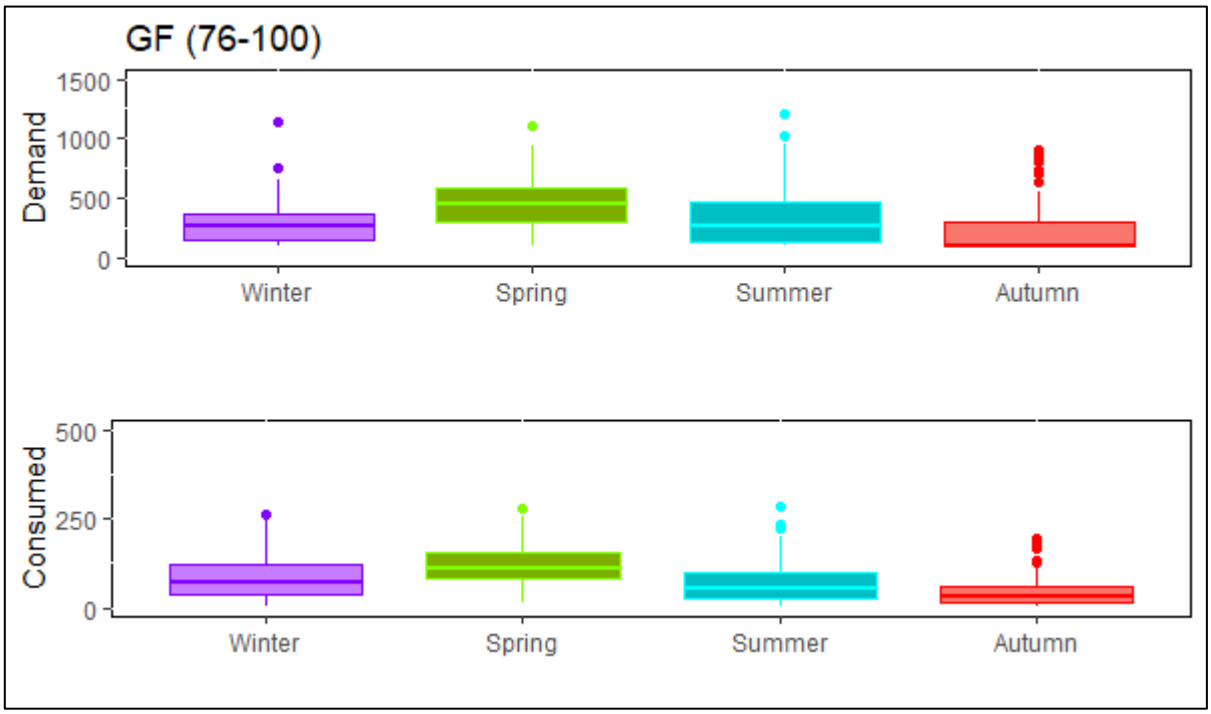

Figure 57: Demand and consumption vs season of the year for 76-100 kWh bin (GF)

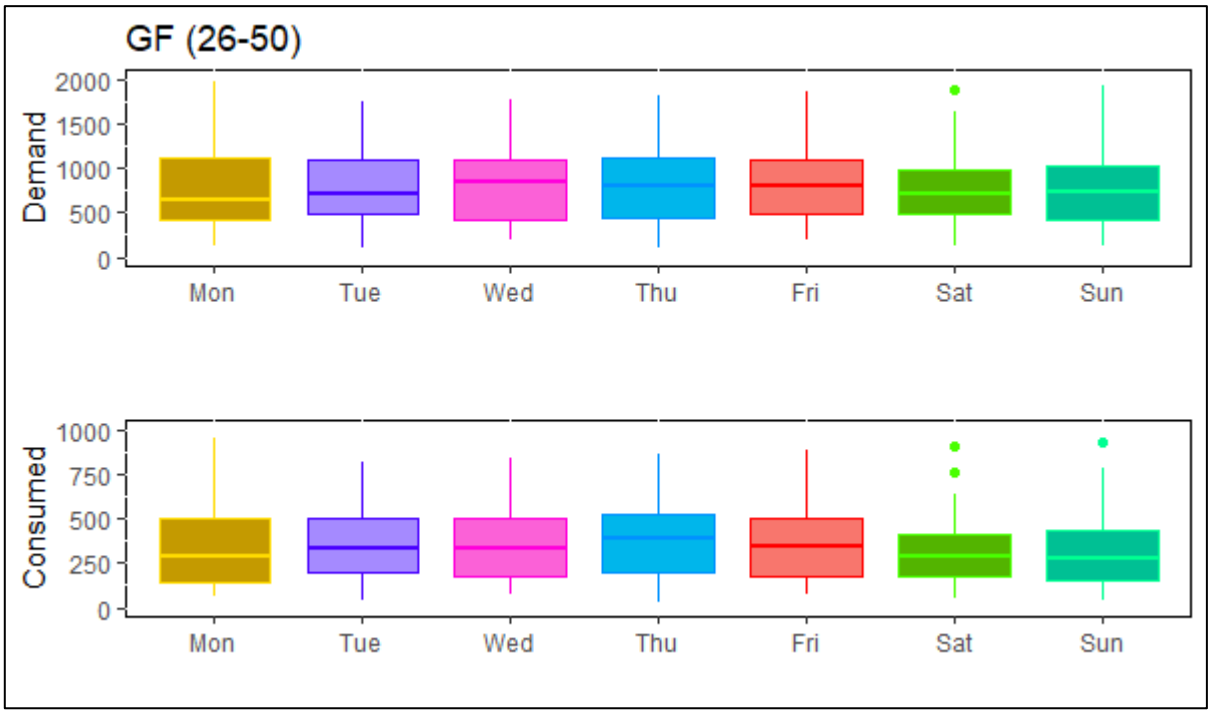

Figure 58: Demand and consumption vs day of the week for 26-50 kWh bin (GF)

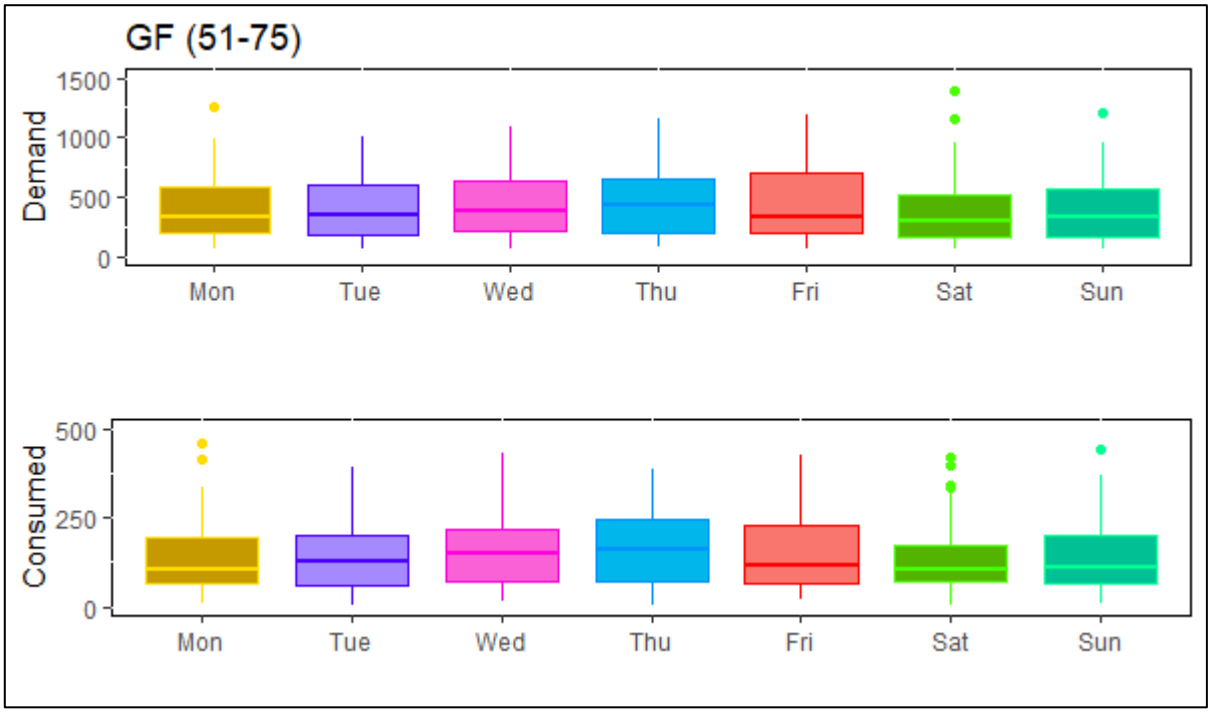

Figure 59: Demand and consumption vs day of the week for 51-75 kWh bin (GF)

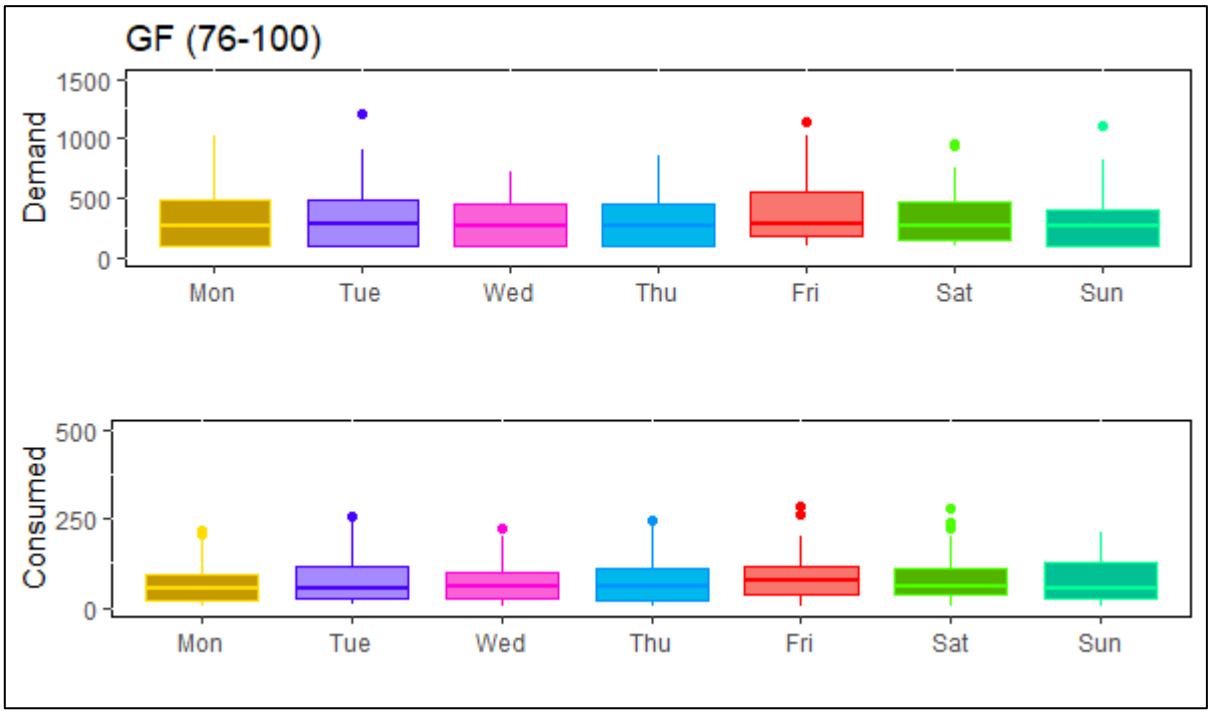

Figure 60: Demand and consumption vs day of the week for 76-100 kWh bin (GF)

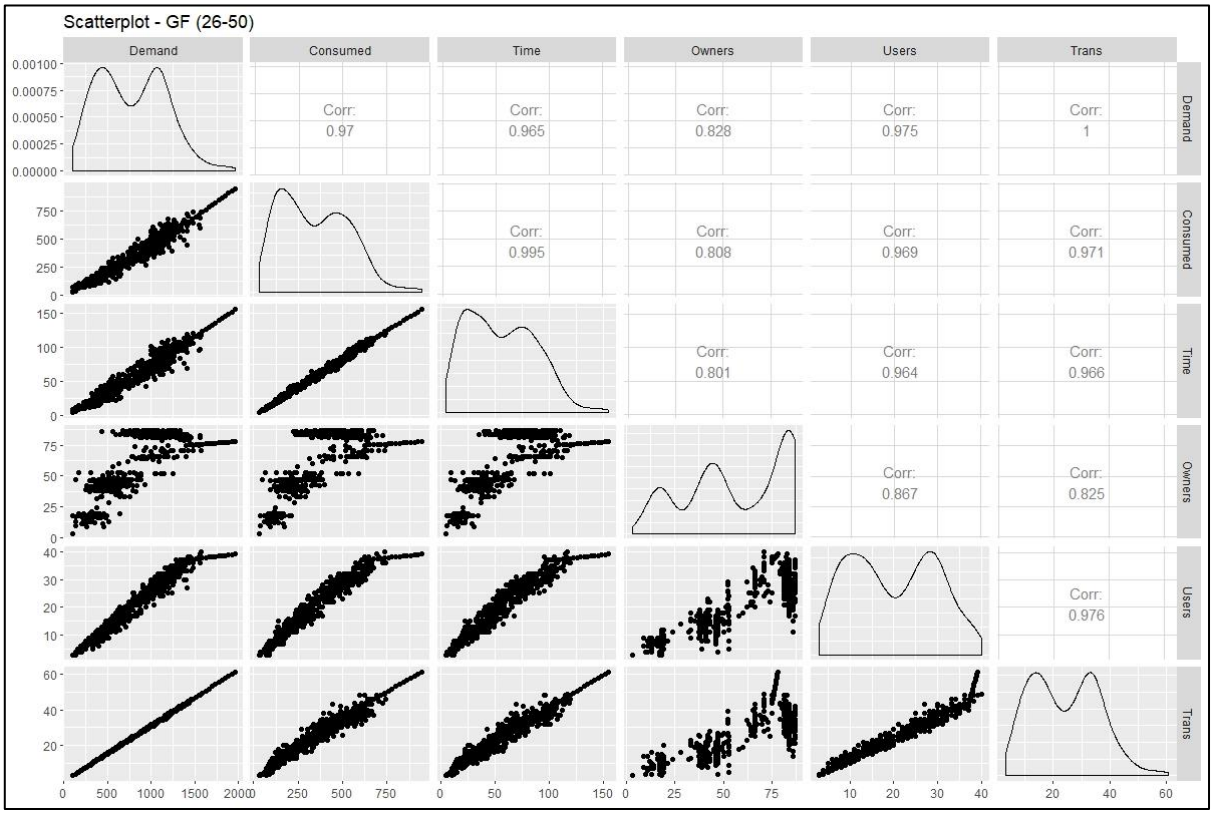

Figure 61: Correlations among numeric variables for 26-50 kWh bin (GF)

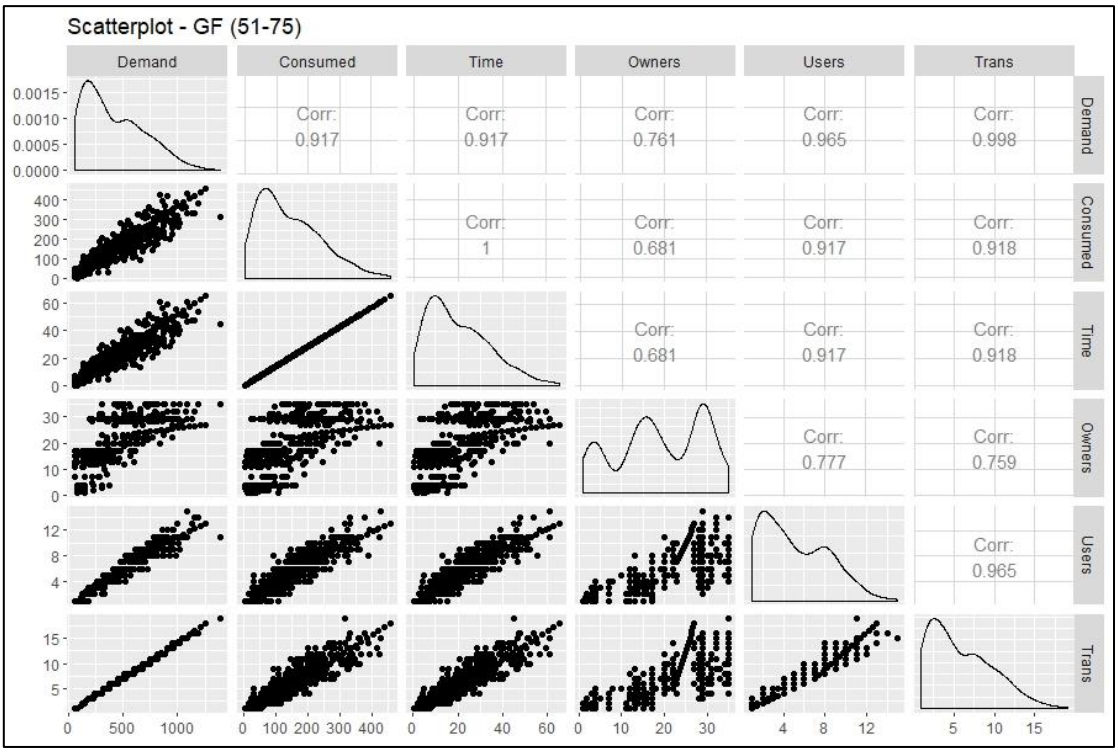

Figure 62: Correlations among numeric variables for 51-75 kWh bin (GF)

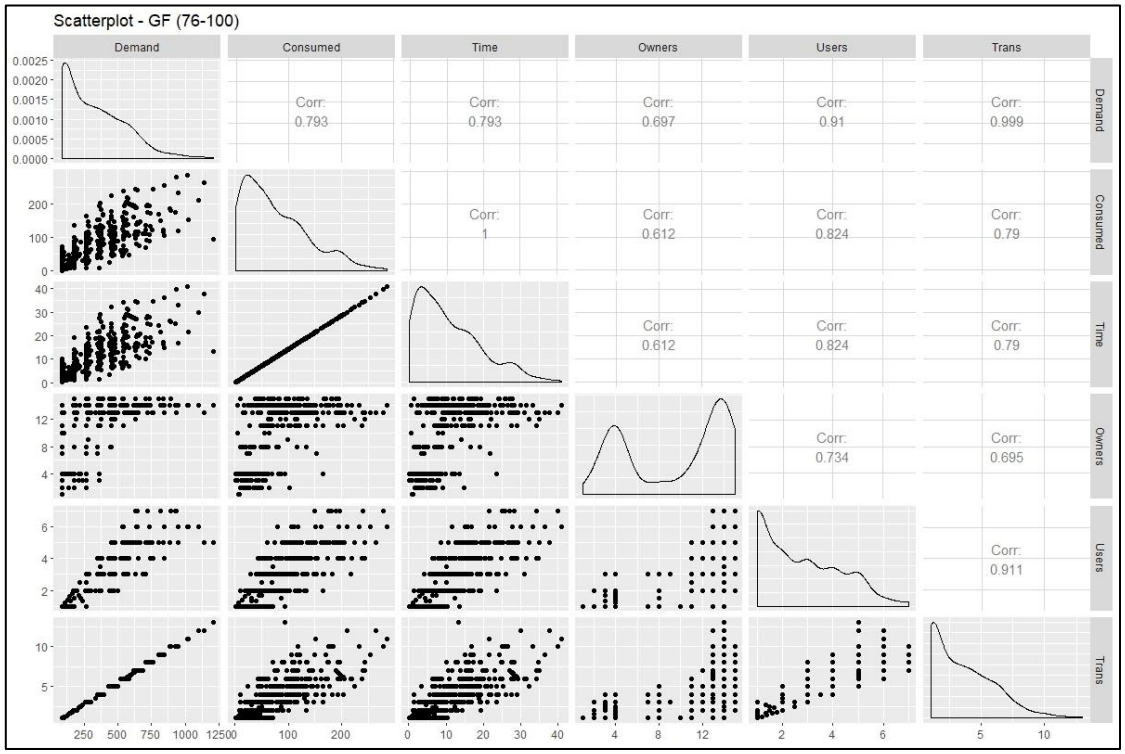

Figure 63: Correlations among numeric variables for 76-100 kWh bin (GF)

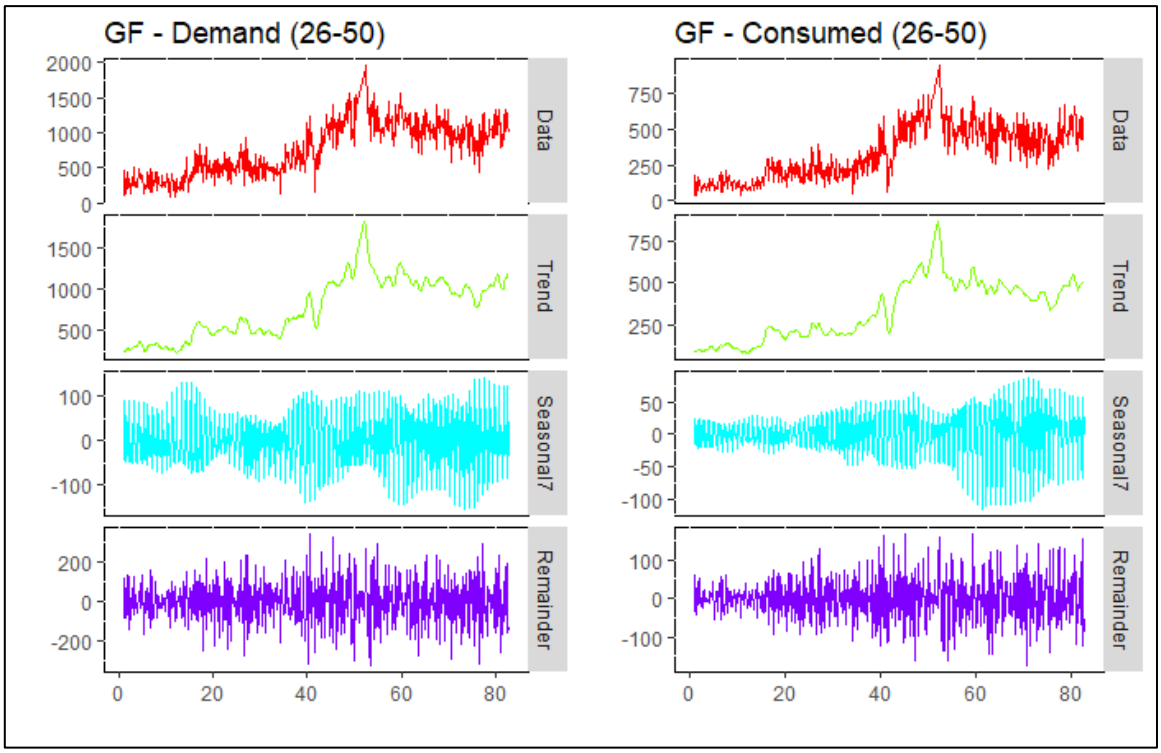

Figure 64: Time series decomposition for 26-50 kWh bin (GF)

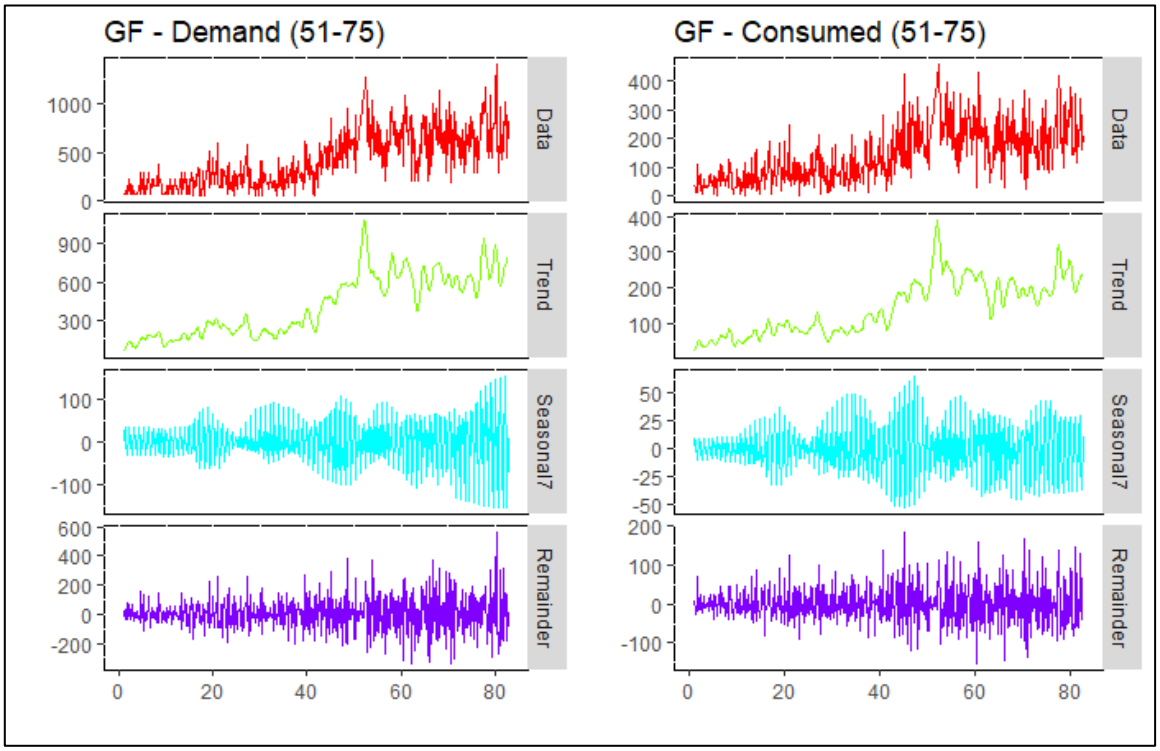

Figure 65: Time series decomposition for 51-75 kWh bin (GF)

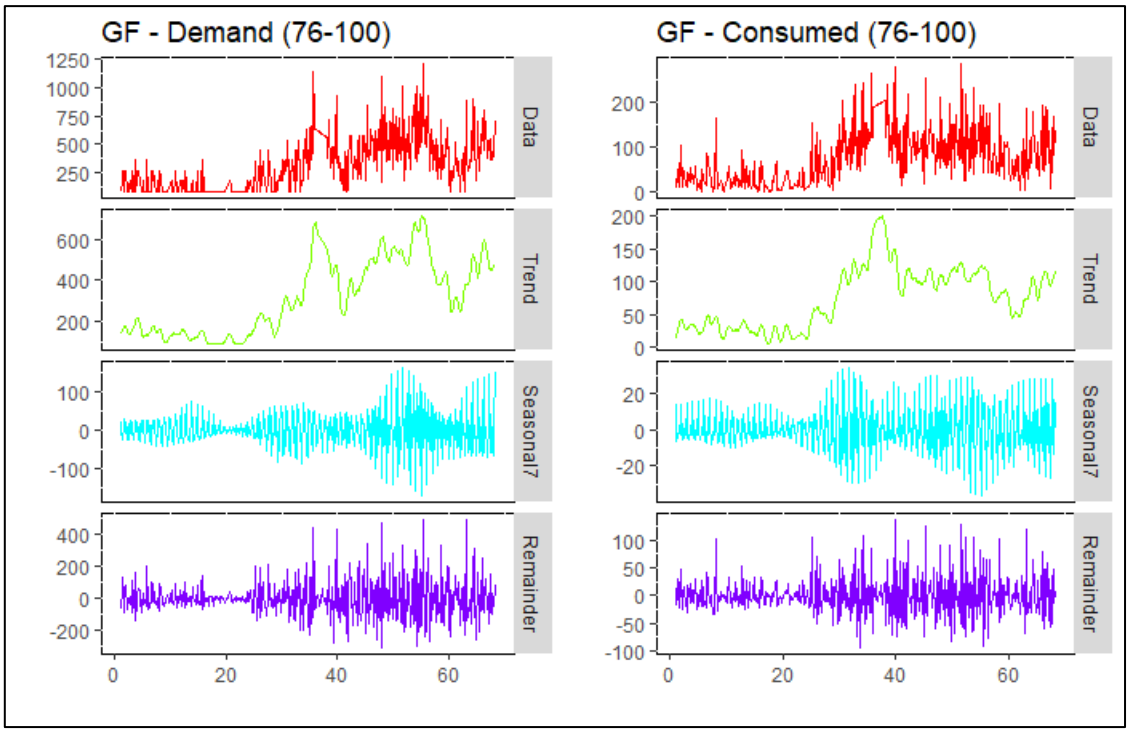

Figure 66: Time series decomposition for 76-100 kWh bin (GF)

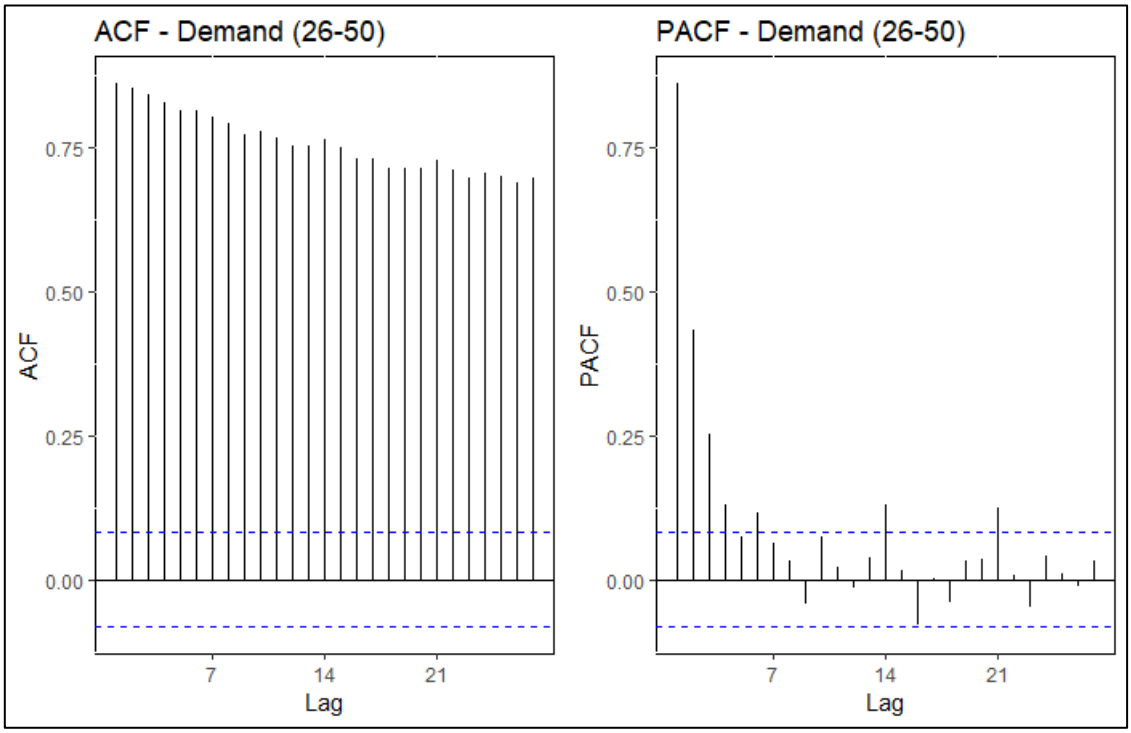

Figure 67: ACF and PACF of demand for 26-50 kWh bin (GF)

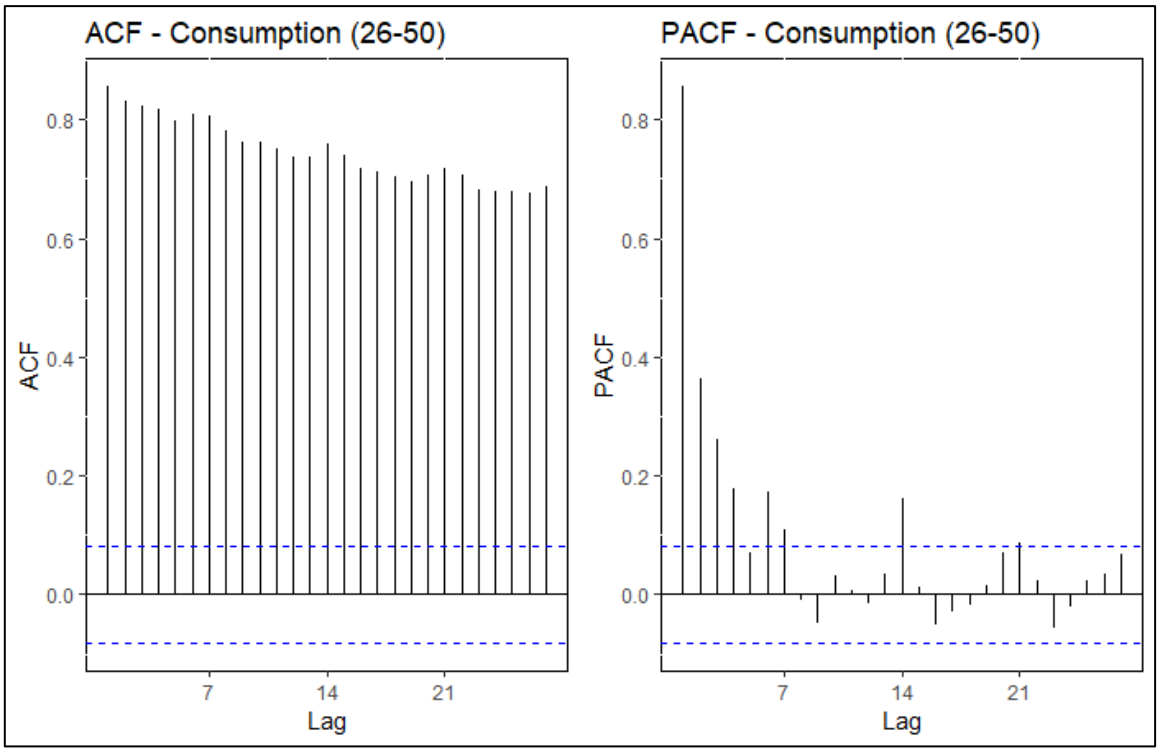

Figure 68: ACF and PACF of consumption for 26-50 kWh bin (GF)

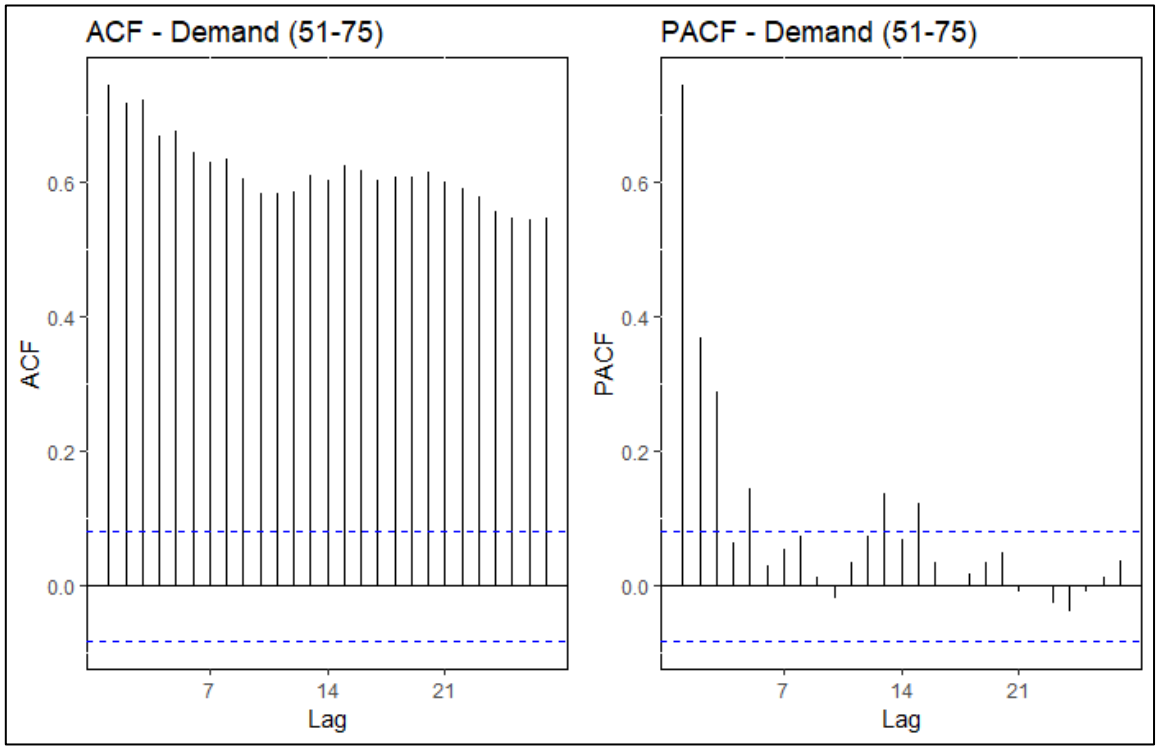

Figure 69: ACF and PACF of demand for 51-75 kWh bin (GF)

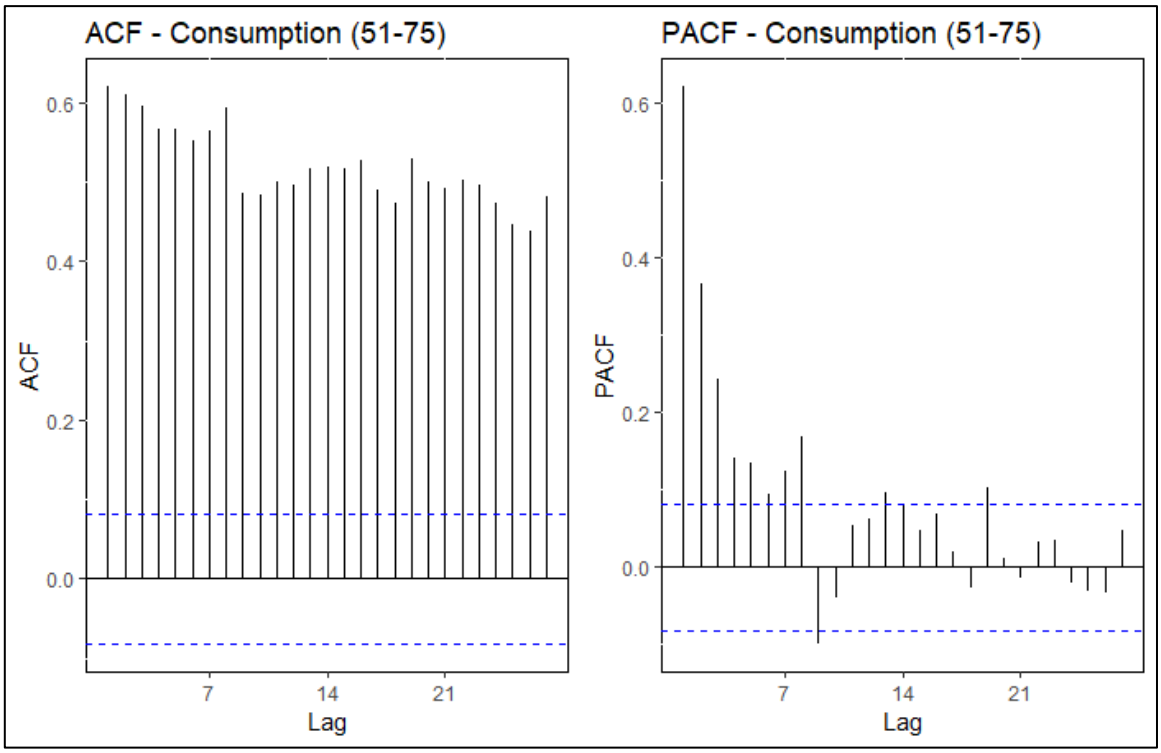

Figure 70: ACF and PACF of consumption for 51-75 kWh bin (GF)

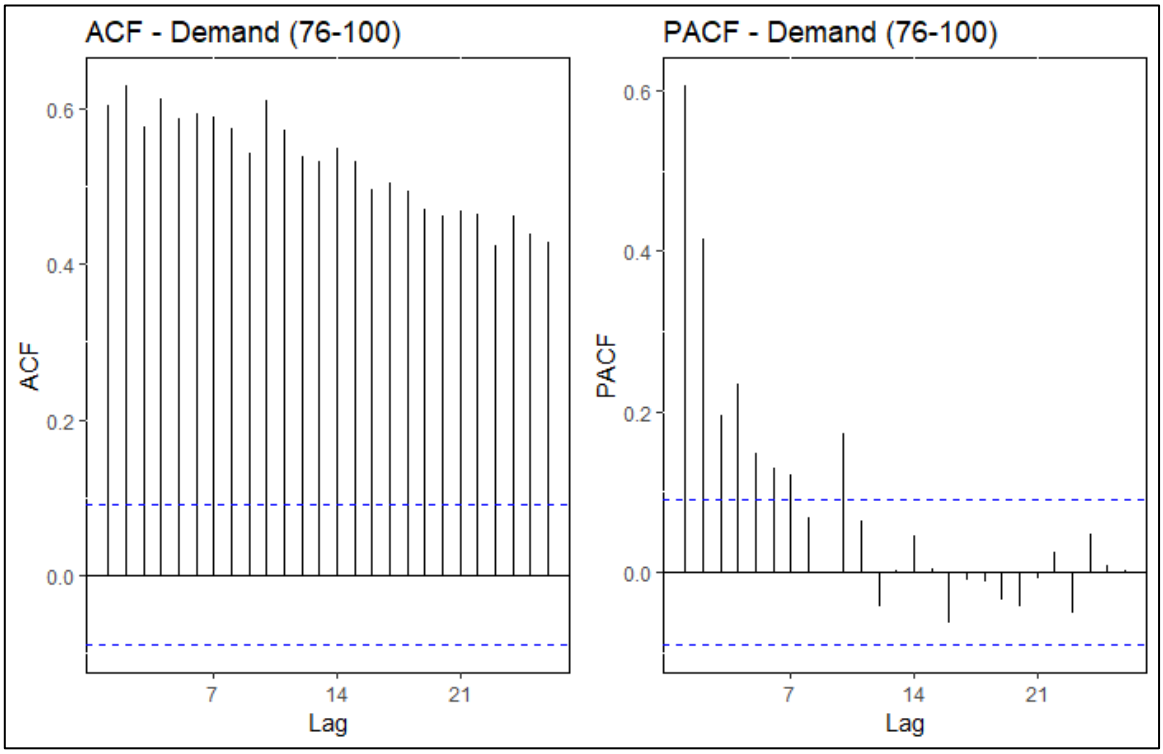

Figure 71: ACF and PACF of demand for 76-100 kWh bin (GF)

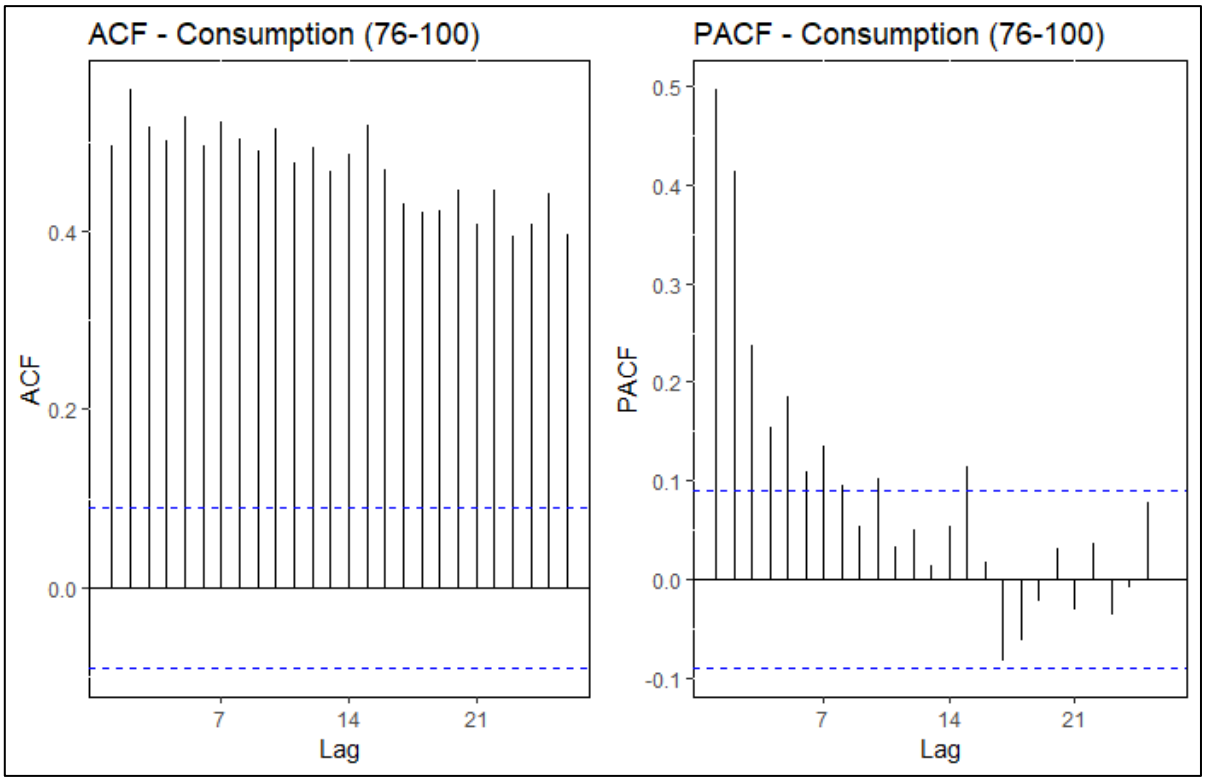

Figure 72: ACF and PACF of consumption for 76-100 kWh bin (GF)

Appendix B

Table 43: Summary statistics for 0-25 kWh bin (CC)

Statistics – CC – 0/25	Demand	Consumed	Time	Owners	Users	Trans
Minimum	6.2	3.86	1.07	1	1	1
1st Quartile	208.6	91.52	21.89	43	12	15.75
Median	720	389.42	91.8	98.5	41.5	57
Mean	630.4	330.88	79.12	92.68	37.66	50.46
3rd Quartile	982.1	523.37	124.49	153	60	81
Maximum	1414.7	772.75	185.36	162	84	114

Table 44: Summary statistics for 26-50 kWh bin (CC)

Statistics – CC – 26/50	Demand	Consumed	Time	Owners	Users	Trans
Minimum	28	0.92	0.13	1	1	1
1st Quartile	261.2	115.41	17.77	37	7	8
Median	572.4	270.2	42.68	55	16	18
Mean	530.3	247.27	39.54	51.59	14.52	16.85
3rd Quartile	755	349.94	56.62	72	21	24
Maximum	1201.8	572.48	99.51	73	30	38

Table 45: Summary statistics for 51-75 kWh bin (CC)

Statistics – CC – 51/75	Demand	Consumed	Time	Owners	Users	Trans
Minimum	60	1.06	0.15	1	1	1
1st Quartile	150	53.43	7.63	6	2	2
Median	435	157.07	22.44	21	5	6
Mean	425.9	158.84	22.69	17.25	5.12	5.79
3rd Quartile	660	240.56	34.37	28	8	9
Maximum	1335	604.86	86.41	30	13	18

Table 46: Summary statistics for 76-100 kWh bin (CC)

Statistics – CC – 76/100	Demand	Consumed	Time	Owners	Users	Trans
Minimum	85	0.51	0.07	1	1	1
1st Quartile	174.2	33.66	4.81	8	1.29	1.94
Median	280	75.33	10.76	13	3	3
Mean	331.4	89.88	12.84	12.24	3.15	3.6
3rd Quartile	465	135.41	19.35	17	4	5
Maximum	1020	337.17	48.17	19	9	11

Appendix C

Time Series Regression

1. $\boldsymbol{users = f(owners, day, season) + error}$

Table 47: Coefficients of the regression model to predict users for 26-50 kWh bin (GF)

Coefficients (26 – 50 kWh)				
Predictors	**Estimate**	**Std. Error**	**t-value**	**p-value**
intercept	0.11129	0.01926	5.778	< 0.05
owners	0.62417	0.01431	43.606	< 0.05
autumn	-0.1413	0.01559	-9.064	< 0.05
spring	-0.04265	0.01502	-2.84	< 0.05
summer	-0.12526	0.01462	-8.567	< 0.05
mon	0.01899	0.01784	1.065	> 0.05
tue	0.03714	0.01784	2.082	< 0.05
wed	0.03758	0.0179	2.1	< 0.05
thu	0.04115	0.01784	2.306	< 0.05
Fri	0.04499	0.01784	2.522	< 0.05
sat	0.02463	0.01784	-1.381	> 0.05

Table 48: Coefficients of the regression model to predict users for 51-75 kWh bin (GF)

Coefficients (51 – 75 kWh)				
Predictors	**Estimate**	**Std. Error**	**t-value**	**p-value**
intercept	0.0105705	0.0106166	0.996	> 0.05
owners	0.1969915	0.00916	21.506	< 0.05
autumn	-0.0389401	0.0087681	-4.441	< 0.05
spring	0.0035499	0.0084149	0.422	> 0.05
summer	-0.0147009	0.0082128	-1.79	> 0.05
mon	-0.0098429	0.0100318	-0.981	> 0.05
tue	-0.0076959	0.0100318	-0.767	> 0.05
wed	-0.0006814	0.0100633	-0.068	> 0.05
thu	0.0096027	0.0100328	0.957	> 0.05
Fri	0.0031601	0.0100322	0.315	> 0.05
sat	-0.0199669	0.0100318	-1.990	< 0.05

Table 49: Coefficients of the regression model to predict users for 76-100 kWh bin (GF)

Coefficients (76 – 100 kWh)				
Predictors	**Estimate**	**Std. Error**	**t-value**	**p-value**
intercept	-0.05496	0.03173	-1.732	> 0.05
owners	0.52164	0.02619	19.917	< 0.05
autumn	-0.01224	0.02514	-0.487	> 0.05
spring	0.05219	0.02769	1.885	> 0.05
summer	0.04648	0.02391	1.944	> 0.05
mon	-0.02945	0.03109	-0.947	> 0.05
tue	0.00105	0.03109	0.034	> 0.05
wed	-0.04424	0.03121	-1.417	> 0.05
thu	-0.01146	0.03121	-0.367	> 0.05
Fri	0.04357	0.03121	1.396	> 0.05
sat	0.01092	0.03121	0.350	> 0.05

Table 50: Performance comparison of models to predict users (GF)

Models (Bin-wise)	MAE	MAPE
26-50 kWh	4.34	19.13
51-75 kWh	2.04	38.8
76-100 kWh	1.32	54.09

2. $demand = f(time') + error$

Table 51: Coefficients of the regression model to predict demand for 26-50 kWh bin (GF)

Coefficients (26 – 50 kWh)				
Predictors	**Estimate**	**Std. Error**	**t-value**	**p-value**
intercept	0.048763	0.004482	10.88	< 0.05
time'	0.942945	0.011182	84.33	< 0.05

Table 52: Coefficients of the regression model to predict demand for 51-75 kWh bin (GF)

Coefficients (51 – 75 kWh)				
Predictors	**Estimate**	**Std. Error**	**t-value**	**p-value**
intercept	-0.001931	0.006735	-0.287	> 0.05
time'	1.343359	0.027679	48.533	< 0.05

Table 53: Coefficients of the regression model to predict demand for 76-100 kWh bin (GF)

Coefficients (76 – 100 kWh)				
Predictors	**Estimate**	**Std. Error**	**t-value**	**p-value**
intercept	0.011506	0.009122	1.261	< 0.05
time'	0.751222	0.026951	27.873	< 0.05

Table 54: Performance comparison of models to predict demand (GF)

Models (Bin-wise)	MAE	MAPE
26-50 kWh	171.65	19.8
51-75 kWh	210.56	37.64
76-100 kWh	161.28	53.69

3. $\boldsymbol{consumed = f(demand') + error}$

Table 55: Coefficients of the regression model to predict consumption for 26-50 kWh bin (GF)

Coefficients (26 – 50 kWh)				
Predictors	**Estimate**	**Std. Error**	**t-value**	**p-value**
intercept	-0.027745	0.004395	-6.313	< 0.05
demand'	0.980022	0.010369	94.516	< 0.05

Table 56: Coefficients of the regression model to predict consumption for 51-75 kWh bin (GF)

Coefficients (51 – 75 kWh)				
Predictors	**Estimate**	**Std. Error**	**t-value**	**p-value**
intercept	0.063421	0.005887	10.77	< 0.05
demand'	0.929057	0.017432	53.30	< 0.05

Table 57: Coefficients of the regression model to predict consumption for 76-100 kWh bin (GF)

Coefficients (76 – 100 kWh)				
Predictors	**Estimate**	**Std. Error**	**t-value**	**p-value**
intercept	0.088747	0.008701	10.20	< 0.05
demand'	0.834093	0.029924	27.87	< 0.05

Table 58: Performance comparison of models to predict consumption (GF)

Models (Bin-wise)	MAE	MAPE
26-50 kWh	91.8	24.26
51-75 kWh	64.77	40.49
76-100 kWh	49.23	150.01

Regression with ARIMA Errors

1. $\boldsymbol{users = f(owners, day, season) + error}$

Table 59: Coefficients of regression with ARIMA model to predict users for 26-50 kWh bin (GF)

Coefficients (26 – 50 kWh)		
Predictors	**Estimate**	**Std. Error**
ar1	0.9423	0.0241
ma1	-0.7205	0.0442
sar1	0.8127	0.0939
sar2	0.0842	0.0472
sma1	-0.8133	0.0894
owners	0.6738	0.0584
autumn	-0.0064	0.0465
spring	0.0096	0.039
summer	-0.0391	0.0427
mon	0.0231	0.0228
tue	0.0383	0.023
wed	0.0422	0.0233
thu	0.0411	0.0232
fri	0.0475	0.023
sat	-0.021	0.0227

Table 60: Coefficients of regression with ARIMA model to predict users for 51-75 kWh bin (GF)

Coefficients (51 – 75)		
Predictors	**Estimate**	**Std. Error**
ma1	0.2363	0.0415
ma2	0.1826	0.0431
ma3	0.1919	0.0421
ma4	0.1338	0.0383
ma5	0.1987	0.0422
owners	0.5857	0.0273
autumn	-0.1083	0.0257
spring	0.0164	0.0227
summer	-0.0306	0.0229
mon	-0.0008	0.0163
tue	-0.0032	0.0152
wed	0.0348	0.0157
thu	0.0582	0.0157
fri	0.0388	0.0152
sat	-0.0317	0.0163

Table 61: Coefficients of regression with ARIMA model to predict users for 76-100 kWh bin (GF)

Coefficients (76 – 100 kWh)		
Predictors	**Estimate**	**Std. Error**
ar1	0.9426	0.0257
ma1	-0.8116	0.0403
owners	0.4988	0.0533
autumn	-0.0267	0.0452
spring	0.0257	0.057
summer	-0.0048	0.0474
mon	-0.0349	0.0265
tue	-0.0043	0.0267
wed	-0.0497	0.0268
thu	-0.0163	0.0269
fri	0.0368	0.0268
sat	0.0057	0.0266

Table 62: Performance comparison of models to predict users (GF)

Models (Bin-wise)	MAE	MAPE
26-50 kWh	4.47	19.74
51-75 kWh	1.99	31.13
76-100 kWh	1.2	47.19

2. $\boldsymbol{demand\ (26-50\ kWh) = f(time') + error}$
$\boldsymbol{demand\ (51-75\ kWh) = f(trans') + error}$
$\boldsymbol{demand\ (76-100\ kWh) = f(trans') + error}$

Table 63: Coefficients of regression with ARIMA model to predict demand for 26-50 kWh bin (GF)

Coefficients (26 – 50 kWh)		
Predictors	**Estimate**	**Std. Error**
ma1	-0.924	0.0223
sar1	0.1222	0.044
sar2	0.096	0.0423
time'	0.7987	0.0224

Table 64: Coefficients of regression with ARIMA model to predict demand for 51-75 kWh bin (GF)

Coefficients (51 – 75 kWh)		
Predictors	**Estimate**	**Std. Error**
ar1	-0.874	0.2011
ma1	-0.0854	0.2375
ma2	-0.7169	0.2318
sar1	0.0966	0.0539
trans'	0.9546	0.0034

Table 65: Coefficients of regression with ARIMA model to predict demand for 76-100 kWh bin (GF)

Coefficients (76 - 100 kWh)		
Predictors	**Estimate**	**Std. Error**
ar1	-0.3912	0.0252
ar2	-0.9552	0.0312
ma1	-0.3669	0.0359
ma2	0.5692	0.0395
ma3	-0.7956	0.0355
trans'	0.9749	0.0028

Table 66: Performance comparison of models to predict demand (GF)

Models (Bin-wise)	MAE	MAPE
26-50 kWh	167.73	19.52
51-75 kWh	192.39	34.00
76-100 kWh	164.44	57.94

3. $\boldsymbol{consumed\ (26-50\ kWh) = f(demand') + error}$
$\boldsymbol{consumed\ (51-75\ kWh) = f(trans') + error}$
$\boldsymbol{consumed\ (76-100\ kWh) = f(trans') + error}$

Table 67: Coefficients of regression with ARIMA model to predict consumption for 26-50 kWh bin (GF)

Coefficients (26 – 50 kWh)		
Predictors	**Estimate**	**Std. Error**
ar1	-0.41	0.2543
ma1	0.5223	0.2369
sar1	0.9307	0.0311
sma1	-0.8154	0.0537
sma2	0.0123	0.0465
intercept	-0.0152	0.0082
demand'	0.9454	0.0169

Table 68: Coefficients of regression with ARIMA model to predict consumption for 51-75 kWh bin (GF)

Coefficients (51 – 75 kWh)		
Predictors	**Estimate**	**Std. Error**
ma1	-0.9566	0.9096
trans'	0.0124	0.0259

Table 69: Coefficients of regression with ARIMA model to predict consumption for 76-100 kWh bin (GF)

Coefficients (76 – 100 kWh)		
Predictors	**Estimate**	**Std. Error**
ma1	-0.9233	0.0185
sar1	-0.6682	0.0947
sar2	0.2338	0.0484
sma1	0.856	0.0872
trans'	0.7943	0.0368

Table 70: Performance comparison of models to predict consumption (GF)

Models (Bin-wise)	MAE	MAPE
26-50 kWh	87.1	22.89
51-75 kWh	66.06	43.01
76-100 kWh	40.17	110.77

Regression (with ARIMA Errors) with Distributed Lags

1. $\boldsymbol{users = f(owners, day, season) + error}$

Table 71: Coefficients of regression with distributed lag model to predict users for 26-50 kWh bin (GF)

Coefficients (26 – 50 kWh)		
Predictors	**Estimate**	**Std. Error**
ar1	0.9546	0.0205
ma1	-0.7417	0.0436
autumn	-0.0121	0.0446
spring	0.02	0.0366
summer	-0.0419	0.0395
mon	0.0185	0.0137
tue	0.0375	0.0139
wed	0.0366	0.014
thu	0.042	0.014
fri	0.0452	0.0139
sat	-0.0266	0.0137
owners	0.4591	0.7047
owners-lag1	0.4699	1.0977
owners-lag2	0.5404	1.1085
owners-lag3	-2.0542	1.1038
owners-lag4	2.4823	1.1201
owners-lag5	-1.515	1.1198
owners-lag6	0.2896	0.7309

Table 72: Coefficients of regression with distributed lag model to predict users for 51-75 kWh bin (GF)

Coefficients (51 – 75 kWh)		
Predictors	**Estimate**	**Std. Error**
ma1	0.2312	0.0417
ma2	0.1861	0.0433
ma3	0.1965	0.0424
ma4	0.1339	0.0384
ma5	0.196	0.0433
autumn	-0.1024	0.0259
spring	0.0232	0.0232
summer	-0.0248	0.0233
mon	-0.0008	0.0162
tue	-0.0041	0.015
wed	0.0343	0.0155
thu	0.0585	0.0155
fri	0.0386	0.015
sat	-0.03	0.0162
owners	1.3671	0.7141
owners-lag1	-0.2650	1.0204
owners-lag2	-0.8563	1.0437
owners-lag3	-1.2728	1.0355
owners-lag4	2.9522	1.0429
owners-lag5	-0.5884	1.0479
owners-lag6	-0.7574	0.741

Table 73: Coefficients of regression with distributed lag model to predict users for 76-100 kWh bin (GF)

Coefficients (76 – 100 kWh)		
Predictors	**Estimate**	**Std. Error**
ar1	0.9413	0.0262
ma1	-0.8047	0.042
autumn	-0.0214	0.0459
spring	0.0329	0.0575
summer	0.0015	0.0475
mon	-0.0407	0.0264
tue	-0.0042	0.0264
wed	-0.0481	0.0266
thu	-0.0164	0.0267
fri	0.0357	0.0266
sat	0.0043	0.0264
owners	1.7716	0.4966
owners-lag1	-1.2836	0.4979

Table 74: Performance comparison of models to predict users (GF)

Models (Bin-wise)	MAE	MAPE
26-50 kWh	4.44	19.63
51-75 kWh	1.96	30.15
76-100 kWh	1.14	42.72

2. $\boldsymbol{demand\ (26-50\ kWh) = f(time') + error}$
$\boldsymbol{demand\ (51-75\ kWh) = f(owners, day, season) + error}$
$\boldsymbol{demand\ (76-100\ kWh) = f(owners, day, season) + error}$

Table 75: Coefficients of regression with distributed lag model to predict demand for 26-50 kWh bin (GF)

Coefficients (26 – 50 kWh)		
Predictors	**Estimate**	**Std. Error**
ar1	-0.0666	0.0608
ar2	-0.1175	0.0537
ar3	-0.1357	0.0507
ar4	-0.1666	0.0504
ma1	-0.8032	0.0506
time'	0.794	0.0217

Table 76: Coefficients of regression with distributed lag model to predict demand for 51-75 kWh bin (GF)

Coefficients (51 – 75 kWh)		
Predictors	**Estimate**	**Std. Error**
ar1	0.2334	0.0413
ar2	0.1395	0.0422
ar3	0.1969	0.0415
autumn	-0.0753	0.0291
spring	0.0206	0.0255
summer	-0.0232	0.026
mon	-0.0061	0.0158
tue	0.0016	0.0162
wed	0.0209	0.0156
thu	0.0445	0.0156
fri	0.031	0.0162
sat	-0.0067	0.0158
owners	0.5235	0.0313

Table 77: Coefficients of regression with distributed lag model to predict demand for 76-100 kWh bin (GF)

Coefficients (76 - 100 kWh)		
Predictors	**Estimate**	**Std. Error**
ar1	0.1231	0.1491
ar2	0.7985	0.1444
ma1	-0.0277	0.1685
ma2	-0.6224	0.147
autumn	-0.006	0.0409
spring	-0.0351	0.0526
summer	-0.0282	0.0454
mon	0.0064	0.0207
tue	0.0207	0.0202
wed	-0.0146	0.0206
thu	0.0203	0.0207
fri	0.0426	0.0203
sat	0.0057	0.0207
owners	1.0474	0.3872
owners-lag1	-0.6693	0.3883

Table 78: Performance comparison of models to predict demand (GF)

Models (Bin-wise)	MAE	MAPE
26-50 kWh	167.0	19.44
51-75 kWh	193.44	33.5
76-100 kWh	163.51	56.91

3. $consumed = f(demand') + error$

Table 79: Coefficients of regression with distributed lag model to predict consumption for 26-50 kWh bin (GF)

Coefficients (26 – 50 kWh)		
Predictors	**Estimate**	**Std. Error**
ar1	1.0674	0.0508
ar2	-0.1755	0.0619
ar3	0.001	0.0618
ar4	0.0895	0.0449
ma1	-0.9102	0.0308
demand'	0.9146	0.0162

Table 80: Coefficients of regression with distributed lag model to predict consumption for 51-75 kWh bin (GF)

Coefficients (51 – 75 kWh)		
Predictors	**Estimate**	**Std. Error**
ma1	-0.9579	0.0122
demand'	0.9517	0.0271

Table 81: Coefficients of regression with distributed lag model to predict consumption for 76-100 kWh bin (GF)

Coefficients (76 – 100 kWh)		
Predictors	**Estimate**	**Std. Error**
ma1	-0.9075	0.0187
demand'	0.7801	0.0385

Table 82: Performance comparison of models to predict consumption (GF)

Models (Bin-wise)	MAE	MAPE
26-50 kWh	86.88	22.82
51-75 kWh	66.82	42.72
76-100 kWh	40.07	109.06

LSTM Networks

Table 83: Performance comparison of LSTM networks (GF)

Bin	Target	Predictors	MAE	MAPE
26-50 kWh	*users*	*owners, season, day*	3.74	15.21
26-50 kWh	*demand*	*users'*	147.86	15.75
26-50 kWh	*consumed*	*owners, season, day*	75.79	18.13
51-75 kWh	*users*	*owners, season, day*	2.09	28.59
51-75 kWh	*demand*	*owners, season, day*	65.56	36.26
51-75 kWh	*consumed*	*owners, season, day*	64.34	36.9
76-100 kWh	*users*	*owners, season, day*	1.36	40.43
76-100 kWh	*demand*	*time'*	178.26	48.49
76-100 kWh	*consumed*	*owners, season, day*	42.65	93.68

Appendix D

R-Codes

```
## User-defined functions ------------------------------------------------------------------------

# Load packages
library(tidyverse)
library(lubridate)
library(fpp2)
library(dummies)
library(caret)
library(tictoc)
options(max.print = 999999)
options(scipen = 999)

# Function to choose plot (z = factor variable with n factors in aesthetics)

# For plots with facets, add the following after calling "myplot":
# facet_grid(~as.factor(z))
# scale_color_manual(values=rainbow(n))
myplot <- function(data, x, y, plot_fun, z, x_label, y_label, plot_title){
  attach(data)
  if(missing(y)){
    if(missing(z)){
      ggplot(data, aes(x = x)) +
        plot_fun +
        theme(legend.title = element_blank(), panel.background = element_rect(fill = "white",
                                                                               colour = "black")) +
        labs(title = plot_title, x = x_label, y = y_label)
    }else{
      ggplot(data, aes(x = x, col = as.factor(z), fill = as.factor(z))) +
        plot_fun +
        theme(legend.title = element_blank(), panel.background = element_rect(fill = "white",
                                                                               colour = "black")) +
        labs(title = plot_title, x = x_label, y = y_label)
    }
  }
  else{
    if(missing(z)){
      ggplot(data, aes(x = x, y = y)) +
        plot_fun +
        theme(legend.title = element_blank(), panel.background = element_rect(fill = "white",
                                                                              colour = "black")) +
        labs(title = plot_title, x = x_label, y = y_label)
    }else{
      ggplot(data, aes(x = x, y = y, col=as.factor(z), fill=as.factor(z))) +
        plot_fun +
        theme(legend.title = element_blank(), panel.background = element_rect(fill = "white",
                                                                              colour = "black")) +
        labs(title = plot_title, x = x_label, y = y_label)
    }
  }
}

# Function to plot multiple scatterplots
scatterplot <- function(data, columns, column_names, ...){
  GGally::ggpairs(data, columns = columns, columnLabels = column_names, ...)
}
```

```r
# Function to calculate NA values column-wise
na_values <- function(data){
  na_vals <- matrix(0, nrow =1, ncol = ncol(data))

  for(i in 1:ncol(data)){
    na_vals[, i] = sum(is.na(data[, i]))
  }

  na_vals <- as.data.frame(na_vals)
  colnames(na_vals) <- colnames(data)

  na_vals <<- data.frame(variable = colnames(na_vals), na_vals = t(na_vals),
                         prop_na = t(100*na_vals/nrow(data)), row.names = NULL)

}

# Function to normalize dataframe (min-max normalization)
standardize <- function(data, new_data_name){
  attach(as.data.frame(data))
  for(i in 1:ncol(data)){
    if(class(data[[i]]) == "numeric" || class(data[[i]]) == "integer" ||
       class(data[[i]]) == "ts" || class(data[[i]]) == "msts"){
      numerator <- data[, i] - min(data[, i])
      denominator <- max(data[, i]) - min(data[, i])
      norm_var <- as.data.frame(numerator/denominator)
      names(norm_var) <- paste0(names(data[i]), "_", "norm")
      data <- cbind(as.data.frame(data), norm_var)
    }
    else {
      next
    }
  }
  assign(paste0(new_data_name), data, inherits = T)
}

# Function for linear interpolation to fill missing data
linear_interpolation <- function(data, new_data_name){
  attach(data)
  for(i in 1:ncol(data)){
    if(class(data[[i]]) == "numeric" || class(data[[i]]) == "integer" ||
       class(data[[i]]) == "ts" || class(data[[i]]) == "msts"){
      interp <- as.numeric(na.interp(data[, i]))
      names(interp) <- paste0(names(data[i]), "_", "interp")
      data <- cbind(as.data.frame(data), interp)
    }
    else {
      next
    }
  }
  assign(paste0(new_data_name), data, inherits = T)
}


## EDA ---------------------------------------------------------------------------------------------------

# Function to set working directory
wd <- function(path){
  setwd(path)
}

# Set working directory
path <- "F:/EA Tech MSc Research/Codes/Data/Data 3.0/R"
wd(path)

# Source user-defined functions
source("R_Functions.R")
```

```
# ------------------------------------------------------------------------------------------------------------------------

# Function to calculate EV owners per day based on kWh, piv and kW

EV_owners <- function(df, df_name){
  # calculate EV owners based on kWh (battery capacity bin)
  owners_kwh <- df%>%group_by(ParticipantID, capacity_kWh)%>%
    summarise(date = parse_date_time(min(date(ActiveChargingStart)), orders = "ymd"))%>%
    group_by(date, capacity_kWh)%>%
    summarise(owners = length(unique(ParticipantID)))%>%
    mutate(owners_0_25 = if_else(capacity_kWh == "0-25", owners, as.integer(0)),
           owners_26_50 = if_else(capacity_kWh == "26-50", owners, as.integer(0)),
           owners_51_75 = if_else(capacity_kWh == "51-75", owners, as.integer(0)),
           owners_76_100 = if_else(capacity_kWh == "76-100", owners, as.integer(0)))%>%
    group_by(date)%>%
    summarise(owners_0_25 = sum(owners_0_25), owners_26_50 = sum(owners_26_50),
              owners_51_75 = sum(owners_51_75), owners_76_100 = sum(owners_76_100))

  # calculate EV owners based on type of EV (PHEV, BEV or REX)
  owners_piv <- df%>%group_by(ParticipantID, PIVType)%>%
    summarise(date = parse_date_time(min(date(ActiveChargingStart)), orders = "ymd"))%>%
    group_by(date, PIVType)%>%
    summarise(owners = length(unique(ParticipantID)))%>%
    mutate(owners_PHEV = if_else(PIVType == "Plug in Hybrid Electric Vehicle (PHEV)", owners, as.integer(0)),
           owners_REX = if_else(PIVType == "Range extender (REX)", owners, as.integer(0)),
           owners_BEV = if_else(PIVType == "Electric only (BEV)", owners, as.integer(0)))%>%
    group_by(date)%>%
    summarise(owners_PHEV = sum(owners_PHEV), owners_REX = sum(owners_REX), owners_BEV = sum(owners_BEV))

   # calculate EV owners based on battery kW
   owners_kw <- df%>%group_by(ParticipantID, CarkW)%>%
     summarise(date = parse_date_time(min(date(ActiveChargingStart)), orders = "ymd"))%>%
     group_by(date, CarkW)%>%
     summarise(owners = length(unique(ParticipantID)))%>%
     mutate(owners_3.6 = if_else(CarkW == 3.6, owners, as.integer(0)),
            owners_7.0 = if_else(CarkW == 7.0, owners, as.integer(0)))%>%
     group_by(date)%>%
     summarise(owners_3.6 = sum(owners_3.6), owners_7.0 = sum(owners_7.0))

   df2 <- full_join(owners_kwh, owners_piv, by = "date")%>%full_join(owners_kw, by = "date")
   assign(paste0(df_name), df2, inherits = T)

}
```

```
# ------------------------------------------------------------------------------------------------------------------------

# Function to calculate active users per day based on kWh, piv and kW (trial 3 excluded)

active_user <- function(df, df_name){
  # calculate active users based on kwh (battery capacity bin)
  active_users_kwh <- df%>%filter(Trial != 3)%>%
    group_by(date = parse_date_time(date(ActiveChargingStart), orders = "ymd"), capacity_kWh)%>%
    summarise(transactions = length(ParticipantID), active_users = length(unique(ParticipantID)))%>%
    mutate(Car_kWh_0_25 = if_else(capacity_kWh == "0-25", active_users, as.integer(0)),
           Car_kWh_26_50 = if_else(capacity_kWh == "26-50", active_users, as.integer(0)),
           Car_kWh_51_75 = if_else(capacity_kWh == "51-75", active_users, as.integer(0)),
           Car_kWh_76_100 = if_else(capacity_kWh == "76-100", active_users, as.integer(0)),
           trans_0_25 = if_else(capacity_kWh == "0-25", transactions, as.integer(0)),
           trans_26_50 = if_else(capacity_kWh == "26-50", transactions, as.integer(0)),
           trans_51_75 = if_else(capacity_kWh == "51-75", transactions, as.integer(0)),
           trans_76_100 = if_else(capacity_kWh == "76-100", transactions, as.integer(0)))%>%
    group_by(date)%>%
    summarise(users_0_25 = sum(Car_kWh_0_25), users_26_50 = sum(Car_kWh_26_50),
              users_51_75 = sum(Car_kWh_51_75), users_76_100 = sum(Car_kWh_76_100),
              trans_0_25 = sum(trans_0_25), trans_26_50 = sum(trans_26_50), trans_51_75 = sum(trans_51_75),
              trans_76_100 = sum(trans_76_100))
```

```
  # calculate EV owners based on type of EV (PHEV, BEV or REX)
  active_users_piv <- df%>%filter(Trial != 3)%>%
    group_by(date = parse_date_time(date(ActiveChargingStart), orders = "ymd"), PIVType)%>%
    summarise(transactions = length(ParticipantID), active_users = length(unique(ParticipantID)))%>%
    mutate(PHEV = if_else(PIVType == "Plug in Hybrid Electric Vehicle (PHEV)", active_users, as.integer(0)),
           REX = if_else(PIVType == "Range extender (REX)", active_users, as.integer(0)),
           BEV = if_else(PIVType == "Electric only (BEV)", active_users, as.integer(0)),
           trans_PHEV = if_else(PIVType == "Plug in Hybrid Electric Vehicle (PHEV)", transactions, as.integer(0)),
           trans_REX = if_else(PIVType == "Range extender (REX)", transactions, as.integer(0)),
           trans_BEV = if_else(PIVType == "Electric only (BEV)", transactions, as.integer(0)))%>%
    group_by(date)%>%
    summarise(users_PHEV = sum(PHEV), users_REX = sum(REX), users_BEV = sum(BEV),
              trans_phev = sum(trans_PHEV), trans_rex = sum(trans_REX), trans_bev = sum(trans_BEV))

  # calculate EV owners based on battery kW
  active_users_kW <- df%>%filter(Trial != 3)%>%
    group_by(date = parse_date_time(date(ActiveChargingStart), orders = "ymd"), CarkW)%>%
    summarise(transactions = length(ParticipantID), active_users = length(unique(ParticipantID)))%>%
    mutate(Car_kW_3.6 = if_else(CarkW == 3.6, active_users, as.integer(0)),
           Car_kW_7.0 = if_else(CarkW == 7.0, active_users, as.integer(0)),
           trans_3.6 = if_else(CarkW == 3.6, transactions, as.integer(0)),
           trans_7.0 = if_else(CarkW == 7.0, transactions, as.integer(0)))%>%
    group_by(date)%>%
    summarise(users_3.6 = sum(Car_kW_3.6), users_7.0 = sum(Car_kW_7.0),
              trans_3.6 = sum(trans_3.6), trans_7.0 = sum(trans_7.0))

  df2 <- full_join(active_users_kwh, active_users_piv, by = "date")%>%full_join(active_users_kW, by = "date")
  assign(paste0(df_name), df2, inherits = T )
}
# ------------------------------------------------------------------------------------------------------------------------

# Function to give day-wise summarised data (trial 3 excluded)

data_day_summary <- function(df1, df_ev_owners_all, df_active_users, df4_ev_owners_t3, df_name){
  df <- df1%>%filter(Trial != 3)%>%
    mutate(date = parse_date_time(date(ActiveChargingStart), orders = 'ymd'))%>%
    group_by(date, day = as.character(wday(ActiveChargingStart, label = T)))%>%
    summarise(transactions = length(ParticipantID), active_users = length(unique(ParticipantID)),
              max_demand = sum(Carkwh), total_kwh = sum(Consumedkwh), duration_hour = sum(Charging_Time_hour),
              Consumption_peruser = total_kwh/active_users,
              Consumption_pertrans = total_kwh/transactions)%>%
    mutate(annual_season = case_when(month(date) >= 3 & month(date) <= 5 ~ "Spring",
                                     month(date) >= 6 & month(date) <= 8 ~ "Summer",
                                     month(date) >= 9 & month(date) <= 11 ~ "Autumn",
                                     month(date) >= 12 ~ "Winter",
                                     month(date) >= 1 & month(date) < 3 ~ "Winter"))%>%
    full_join(df_ev_owners_all, by = "date")%>%
    full_join(df_active_users, by = "date")%>%
    full_join(df4_ev_owners_t3, by = "date")

  for(i in 1:nrow(df)){
    for(j in 1:ncol(df)){
      if(is.na(df[i, j])){
        df[i, j] = 0
      } else {
        next
      }
    }
  }

  assign(paste0(df_name), df, inherits = T)

}
```

```
# ------------------------------------------------------------------------------------------------------------------------

# Function to calculate cumulative ev owners based on kwh, piv and kw (trial 3 excluded)

cum_owners <- function(df, df_name, ev_name){
  df2 <- sapply(df[, c(11:19, 38:46)], cumsum)
  owners <- data.frame(integer(nrow(df)), integer(nrow(df)), integer(nrow(df)), integer(nrow(df)),
                       integer(nrow(df)), integer(nrow(df)), integer(nrow(df)), integer(nrow(df)),
                       integer(nrow(df)))

  colnames(owners) <- c("owners_0_25", "owners_26_50", "owners_51_75", "owners_76_100", "owners_PHEV",
                        "owners_REX", "owners_BEV", "owners_3.6", "owners_7.0")

  for(i in 1:ncol(owners)){
    for(j in 1:nrow(owners)){
      owners[j, i] =  df2[j, i] - df2[j, i + 9]
    }
  }

  df2 <- cbind(as.data.frame(df[, 1]), owners)
  df3 <- full_join(df[, c(-11:-19, -38:-46)], df2, by = "date")
  df3 <- df3%>%mutate(total_owners = owners_3.6 + owners_7.0)
  assign(paste0(ev_name), df2, inherits = T)
  assign(paste0(df_name), df3, inherits = T)
}
# ------------------------------------------------------------------------------------------------------------------------

# Function to create dataset for modelling (DCS based)

model_data <- function(df, feature_name, factor_name, ev_df, active_df, df_name){
  attach(df)
  df1 <- df%>%filter(Trial != 3 & feature_name == factor_name)%>%
    mutate(date = parse_date_time(date(ActiveChargingStart), orders = 'ymd'))%>%
    group_by(date, day = as.character(wday(ActiveChargingStart, label = T)))%>%
    summarise(max_demand = sum(CarkWh), total_kWh = sum(ConsumedkWh), duration_hour = sum(Charging_Time_hour))%>%
    mutate(annual_season = case_when(month(date) >= 3 & month(date) <= 5 ~ "Spring",
                                     month(date) >= 6 & month(date) <= 8 ~ "Summer",
                                     month(date) >= 9 & month(date) <= 11 ~ "Autumn",
                                     month(date) >= 12 ~ "Winter",
                                     month(date) >= 1 & month(date) < 3 ~ "Winter"))

  df2 <- left_join(df1, ev_df, by = "date")%>%
    left_join(active_df, by = "date")%>%
    arrange(date)

  assign(paste0(df_name), df2, inherits = T)
}
# ------------------------------------------------------------------------------------------------------------------------

# Function to interpolate values for missing dates and create new dataframe

regular_data <- function(data, start_date, end_date, new_data_name){

  # Sequence of dates to impute missing values
  seq_date <- data.frame(date = parse_date_time(seq(from = ymd(start_date), to = ymd(end_date), by = "days"),
                                                orders = "ymd"))%>%
    mutate(day = as.character(wday(date, label = T)),
           annual_season = case_when(month(date) >= 3 & month(date) <= 5 ~ "Spring",
                                     month(date) >= 6 & month(date) <= 8 ~ "Summer",
                                     month(date) >= 9 & month(date) <= 11 ~ "Autumn",
                                     month(date) >= 12 ~ "Winter",
                                     month(date) >= 1 & month(date) < 3 ~ "Winter"))

  # Create a new dataframe to include missing dates
  data2 <- left_join(seq_date, data, by = c("date", "day", "annual_season"))

  # Impute missing values by linear interpolation and seasonal adjustment
  for(i in 1:ncol(data2)){
    if(class(data2[[i]]) == "numeric" || class(data2[[i]]) == "integer" ||
       class(data2[[i]]) == "ts" || class(data2[[i]]) == "msts"){
      interp <- as.numeric(na.interp(data2[, i]))
      names(interp) <- paste0(names(data2[i]), "_", "interp")
      data2 <- cbind(data.frame(data2), interp)
    }
    else {
      next
    }
  }
```

```
  # Remove the redundant columns
  data2 <- data2[, -4:-9]

  # Rename columns of new dataframe
  names(data2) <- c("date", "day", "season", "demand", "consumed", "time", "owners", "users", "trans")

  # Save the new dataframe
  assign(paste0(new_data_name), data2, inherits = T)
}

# ------------------------------------------------------------------------------------------------------------------------

# Function to replace outliers in the dataset

ts_clean_data <- function(data, index_list, replacement_list, clean_data_name){

  # data = data with outliers
  # index_list = list of indices where outliers are present
  # replacement_list = list of reasonable replacements
  # clean_data_name = user-specified name for cleaned data

  # Replace outliers with reasonable replacements

  if(0 %in% index_list){
    data2 <- data
  }
  else{
    data2 <- data
    data2[index_list, c("demand", "consumed", "time", "users", "trans")] <- replacement_list

    # Impute missing values by linear interpolation and seasonal adjustment
    for(i in 1:ncol(data2)){
      if(class(data2[[i]]) == "numeric" || class(data2[[i]]) == "integer" ||
         class(data2[[i]]) == "ts" || class(data2[[i]]) == "msts"){
        interp <- as.numeric(na.interp(data2[, i]))
        names(interp) <- paste0(names(data2[i]), "_", "interp")
        data2 <- cbind(data.frame(data2), interp)
      }
      else {
        next
      }
    }

    # Drop redundant columns from cleaned data
    data2 <- data2[, -4:-9]
    names(data2) <- c("date", "day", "season", "demand", "consumed", "time", "owners", "users", "trans")
  }

  # Save cleaned data to workspace
  assign(paste0(clean_data_name), data2, inherits = T)
}

# ------------------------------------------------------------------------------------------------------------------------

# Function to create regular time series (factors are converted into dummies)

regular_ts <- function(data, ts_name){

  # Create dummy variables for annual season and day of the week
  season_dummy <- dummy(data[, "season"])
  day_dummy <- dummy(data[, "day"])

  # Add the dummies to the new dataframe and rename columns
  data2 <- cbind(data.frame(data), season_dummy, day_dummy)[, c(-1:-3)]
  names(data2) <- c("demand", "consumed", "time", "owners", "users", "trans", "autumn", "spring", "summer", "winter",
                    "fri", "mon", "sat", "sun", "thu", "tue", "wed")

  # Rearrange the columns to sttructure the dataframe
  data2 <- data2[, c(1:10, 14, 12, 16:17, 15, 11, 13)]

  # Convert data into time series
  ts <- msts(data2, seasonal.periods = 7)

  # Save the new dataframe
  assign(paste0(ts_name), ts, inherits = T)
}
```

```
# ------------------------------------------------------------------------------------------------------------------------

# Function to create training and test data with distributed lags
lag_data<- function(data, d_lag, p){

  # Create a copy of data
  ts_data <- data

  # Combine main data with DL and drop first 7 obervations
  ts_data2 <- msts(cbind(ts_data, d_lag), seasonal.periods = 7)[-1:-7, ]

  # Rename columns of data with DL
  colnames(ts_data2)[c(1:ncol(ts_data), (ncol(ts_data) + 1):ncol(ts_data2))] <-
    c(colnames(ts_data), c("lag0", "lag1", "lag2", "lag3", "lag4", "lag5", "lag6", "lag7"))

  if(missing(p)){
    # Complete data with lags
    ts_data <<- msts(ts_data2, seasonal.periods = 7)

  }else{
    # Create partition for train-test split
    subset <- floor(p * nrow(ts_data2))

    # Complete data with lags
    ts_data <<- msts(ts_data2, seasonal.periods = 7)

    # Training sample
    ts_train <<- msts(ts_data2[1:subset, ], seasonal.periods = 7)

    # Test sample
    ts_test <<- msts(ts_data2[-1:-subset, ], seasonal.periods = 7)
  }

}

# ------------------------------------------------------------------------------------------------------------------------

# Function to create dataset for a fixed number of EV owners and a chosen duration
scenario_data <- function(start_date, end_date, owners, scenario_data_name){

  # start_date = start date of choice to calculate demand and consumption of energy
  # end_date = end date of choice to calculate demand and consumption of energy
  # owners = fixed number of EV owners to compute demand and consumption of energy between start and end dates
  # scenario_data_name = user defined name for the scenario data

  # Creates sequence of dates between user specified start and end dates with labelled weekdays and annual seasons
  seq_date <- data.frame(date = parse_date_time(seq(from = ymd(start_date), to = ymd(end_date), by = "days"),
                                                orders = "ymd"))%>%
    mutate(day = as.character(wday(date, label = T)),
           season = case_when(month(date) >= 3 & month(date) <= 5 ~ "Spring",
                              month(date) >= 6 & month(date) <= 8 ~ "Summer",
                              month(date) >= 9 & month(date) <= 11 ~ "Autumn",
                              month(date) >= 12 ~ "Winter",
                              month(date) >= 1 & month(date) < 3 ~ "Winter"))

  # Create dataframe of dates, days, annual seasons and EV owners
  data <- cbind(seq_date, owners)

  # Save the data to workspace
  assign(paste0(scenario_data_name), data, inherits = T)
}

# Function to convert scenario-based data to all numeric for predictions
season_data <- function(data, data_name){

  if(data[["season"]][[1]] %in% "Winter"){
    data_season <- data.frame(spring = rep(0, nrow(data)), summer = rep(0, nrow(data)), autumn = rep(0, nrow(data)))
  } else if(data[["season"]][[1]] %in% "Summer"){
    data_season <- data.frame(autumn = rep(0, nrow(data)), winter = rep(0, nrow(data)), spring = rep(0, nrow(data)))
  } else if(data[["season"]][[1]] %in% "Spring"){
    data_season <- data.frame(summer = rep(0, nrow(data)), autumn = rep(0, nrow(data)), winter = rep(0, nrow(data)))
  } else if(data[["season"]][[1]] %in% "Autumn"){
    data_season <- data.frame(winter = rep(0, nrow(data)), spring = rep(0, nrow(data)), summer = rep(0, nrow(data)))
  }

  # Create dummy variables for annual season and day of the week
  season_dummy <- dummy(data[, "season"])
  day_dummy <- dummy(data[, "day"])

  # Add the dummies to the new dataframe and rename columns
  data2 <- cbind(data.frame(data), day_dummy, season_dummy, data.frame(data_season))[, c(-1:-3)]

  # Rearrange columns
  data2 <- data2[, c(1, 3, 7:8, 6, 2, 4:5, 9, 10, 11, 12)]
```

```
  # Rearrange columns
  data2 <- data2[, c(1, 3, 7:8, 6, 2, 4:5, 9, 10, 11, 12)]

  # Change column names
  names(data2)[1:8] <- c("owners", "mon", "tue", "wed", "thu", "fri", "sat", "sun")

  if(data[["season"]][[1]] %in% "Winter"){
    names(data2)[9] <- "winter"
  } else if(data[["season"]][[1]] %in% "Summer"){
    names(data2)[9] <- "summer"
  } else if(data[["season"]][[1]] %in% "Spring"){
    names(data2)[9] <- "spring"
  } else if(data[["season"]][[1]] %in% "Autumn"){
    names(data2)[9] <- "autumn"
  }

  # Change class to numeric
  data_class <- apply(data2, 2, as.numeric)

  # Convert data into time series
  ts <- msts(data2, seasonal.periods = 7)

  # Save the new dataframe
  assign(paste0(data_name), ts, inherits = T)
}

# ------------------------------------------------------------------------------------------------------------------------

# Load package
library(esquisse)

# Transaction data (inspection and wrangling) ----------------------------------------------------------------
gf_final <- read.csv("GF.csv", stringsAsFactors = F, header = T, na.strings = c("", "NA", "NULL"))
cc_final <- read.csv("CC.csv", stringsAsFactors = F, header = T, na.strings = c("", "NA", "NULL"))

# Structure of transaction data
str(gf_final)
str(cc_final)
na_values(gf_final)
na_values(cc_final)

# Modification in data structure
gf_final2 <- gf_final[,-5:-6]
names(gf_final2)[c(3:4,10)] <- c("CarkW", "CarkWh", "ActiveChargingStart")

cc_final2 <- cc_final[, c(-3:-4,-6)]
cc_final2 <- cc_final2[,c(1,4,5:6,2,7:9,3,10)]
names(cc_final2)[c(3:4,9:10)] <- c("CarkW", "CarkWh", "ConsumedkWh", "ActiveChargingStart")

# Structure of modified datasets
str(gf_final2)
str(cc_final2)

# Missing values
na_values(cc_final2)
na_values(gf_final2)

# Date-time parsing
gf_final2$AdjustedStartTime <- parse_date_time(gf_final2$AdjustedStartTime, orders = 'dmy_HM')
gf_final2$AdjustedStopTime <- parse_date_time(gf_final2$AdjustedStopTime, orders = 'dmy_HM')
gf_final2$ActiveChargingStart <- parse_date_time(gf_final2$ActiveChargingStart, orders = 'dmy_HM')

cc_final2$AdjustedStartTime <- parse_date_time(cc_final2$AdjustedStartTime, orders = 'dmy_HM')
cc_final2$AdjustedStopTime <- parse_date_time(cc_final2$AdjustedStopTime, orders = 'dmy_HM')
cc_final2$ActiveChargingStart <- parse_date_time(cc_final2$ActiveChargingStart, orders = 'dmy_HM')

# Invalid date-time entries (if any)
unique(year(gf_final2$AdjustedStartTime))
unique(year(gf_final2$AdjustedStopTime))
unique(year(gf_final2$ActiveChargingStart))
sum(year(gf_final2$AdjustedStartTime) == 2022)
sum(year(gf_final2$AdjustedStopTime) == 2022)

unique(year(cc_final2$AdjustedStartTime))
unique(year(cc_final2$AdjustedStopTime))
unique(year(cc_final2$ActiveChargingStart))
```

```
# Charger install data (inspection and wrangling) ------------------------------------------------------------
ci <- read.csv("ChargerInstall.csv", stringsAsFactors = F, header = T)
str(ci)
ci <- ci[,c(-4,-6,-13)]
colnames(ci)[2] <- "ChargerID"
na_values(ci)

# Date-time parsing
ci$ChargerInstallDate <- parse_date_time(ci$ChargerInstallDate, orders ='dmy')
ci$CarInstallDate <- parse_date_time(ci$CarInstallDate, orders ='dmy')
ci <- ci%>%arrange(ChargerInstallDate)

# Minimum time for full charging
ci <- ci%>%mutate(Min_Charging_Time = CarkWh/CarkW)

# Charger install data for GF
ci_gf <- ci%>%filter(DCSProvider == "Greenflux")
ci_gf_cid <- as.data.frame(unique(ci_gf$ParticipantID))
names(ci_gf_cid) <- "ParticipantID"
ci_gf_cid <- ci_gf_cid%>%arrange(ParticipantID)

# Missing participant IDs from GF transaction data
gf_cid <- as.data.frame(unique(gf_final2$ParticipantID))
names(gf_cid) <- "ParticipantID"
gf_cid <- gf_cid%>%arrange(ParticipantID)
missingid_gf <- anti_join(ci_gf_cid, gf_cid)
missingid_gf <- missingid_gf%>%arrange(ParticipantID)

# GF transaction and charger install data combined
ci_gf2 <- ci_gf[,c(-3,-5)]
gf_final3 <- left_join(gf_final2, ci_gf2, by = "ParticipantID")
gf_final4 <- gf_final3[,c(-11,-16:-18)]
names(gf_final4)[c(1, 3:4)] <- c("ChargerID", "CarkW", "CarkWh")
na_values(gf_final4)

# Charger install data for CC
ci_cc <- ci%>%filter(DCSProvider == "Crowd Charge")
ci_cc_cid <- as.data.frame(unique(ci_cc$ParticipantID))
names(ci_cc_cid) <- "ParticipantID"
ci_cc_cid <- ci_cc_cid%>%arrange(ParticipantID)

# Missing participant IDs from CC transaction data
cc_cid <- as.data.frame(unique(cc_final2$ParticipantID))
names(cc_cid) <- "ParticipantID"
cc_cid <- cc_cid%>%arrange(ParticipantID)
str(cc_cid)
str(ci_cc_cid)
cc_cid$ParticipantID <- as.character(cc_cid$ParticipantID)
ci_cc_cid$ParticipantID <- as.character(ci_cc_cid$ParticipantID)
missingid_cc <- anti_join(ci_cc_cid, cc_cid)
missingid_cc <- missingid_cc%>%arrange(ParticipantID)

# Missing participant IDs for GF and CC
missing_id <- rbind(missingid_gf, missingid_cc)

# CC transaction and charger install data combined
ci_cc2 <- ci_cc[,c(-3,-5)]
cc_final2$ParticipantID <- as.character(cc_final2$ParticipantID)
ci_cc2$ParticipantID <- as.character(ci_cc2$ParticipantID)
cc_final3 <- left_join(cc_final2, ci_cc2, by = "ParticipantID")
cc_final4 <- cc_final3[,c(-11,-16:-18)]
names(cc_final4)[c(1, 3:4)] <- c("ChargerID", "CarkW", "CarkWh")
na_values(cc_final4)

# Filtering CC data for consumed kWh <0.5 or >100
cc_final5 <- cc_final4%>%filter(ConsumedkWh>=0.5)%>%
  filter(ConsumedkWh<=100)

na_values(cc_final5)

# CC data processing for modelling/testing ------------------------------------------------------------------------
cc_final6 <- cc_final5
cc_final6$GroupID[is.na(cc_final6$GroupID)] <- "Unmanaged"
cc_final6$Trial[is.na(cc_final6$Trial)] <- "Unmanaged"
na_values(cc_final6)

cc_final7 <- cc_final6%>%filter(!is.na(ActiveChargingStart))
na_values(cc_final7)

unique(year(cc_final7$AdjustedStartTime))
unique(year(cc_final7$AdjustedStopTime))
unique(year(cc_final7$ActiveChargingStart))
```

```
cc_final8 <- cc_final7%>%mutate(Charging_Time_hour = ConsumedkWh/CarkW,
                                ActiveChargingStop = ActiveChargingStart + 3600 * Charging_Time_hour)

cc_final8 <- cc_final8[, c(1:4, 15, 5:6, 7, 10, 9, 8, 16:17, 11:14)]

cc_final9 <- cc_final8%>%mutate(capacity_kWh = case_when(CarkWh <= 25 ~ "0-25",
                                                         CarkWh >= 26 & CarkWh <= 50 ~ "26-50",
                                                         CarkWh >= 51 & CarkWh <= 75 ~ "51-75",
                                                         CarkWh >= 76 & CarkWh <= 100 ~ "76-100"))

# Summary by date - CC (trial 3 excluded)
EV_owners(cc_final9, "ev_owners_cc")
active_user(cc_final9, "active_users_cc")
EV_owners(cc_final9%>%filter(Trial == 3), "ev_cc_t3")
data_day_summary(cc_final9, ev_owners_cc, active_users_cc, ev_cc_t3, "data_day_cc_summary")
cum_owners(data_day_cc_summary, "data_day_cc", "ev_cc")
data_day_cc <- data_day_cc%>%mutate(prop_active = 100 * active_users/total_owners)

# GF data processing for modelling/testing ------------------------------------------------------------------------
gf_final5 <- gf_final4
gf_final5$GroupID[is.na(gf_final5$GroupID)] <- "Unmanaged"
gf_final5$Trial[is.na(gf_final5$Trial)] <- "Unmanaged"
na_values(gf_final5)

gf_final6 <- gf_final5%>%filter(!is.na(ActiveChargingStart))
unique(year(gf_final6$AdjustedStartTime))
unique(year(gf_final6$AdjustedStopTime))
unique(year(gf_final6$ActiveChargingStart))

gf_final7 <- gf_final6%>%mutate(Charging_Time_hour = ConsumedkWh/CarkW,
                                ActiveChargingStop = ActiveChargingStart + 3600 * Charging_Time_hour)

gf_final7 <- gf_final7[, c(1:4, 15, 5:6, 7, 10, 9, 8, 16:17, 11:14)]

gf_final8 <- gf_final7%>%mutate(capacity_kWh = case_when(CarkWh <= 25 ~ "0-25",
                                                         CarkWh >= 26 & CarkWh <= 50 ~ "26-50",
                                                         CarkWh >= 51 & CarkWh <= 75 ~ "51-75",
                                                         CarkWh >= 76 & CarkWh <= 100 ~ "76-100"))

# Summary by date - GF (trial 3 excluded)
EV_owners(gf_final8, "ev_owners_gf")
active_user(gf_final8, "active_users_gf")
EV_owners(gf_final8%>%filter(Trial == 3), "ev_gf_t3")
data_day_summary(gf_final8, ev_owners_gf, active_users_gf, ev_gf_t3, "data_day_gf_summary")
cum_owners(data_day_gf_summary, "data_day_gf", "ev_gf")
data_day_gf <- data_day_gf%>%mutate(prop_active = 100 * active_users/total_owners)

# Count of consumers based on car specifications (car maker and battery capacity) - GF
# User CarMake or capacity_kWh as a grouping variable along with ParticipantID
car_count_gf <- gf_final8%>%filter(Trial != 3)%>%
  group_by(capacity_kWh, ParticipantID)%>%
  summarise(count = length(unique(ParticipantID)))%>%
  group_by(capacity_kWh)%>%
  summarise(count = sum(count))

# Count of consumers based on car specifications (car maker and battery capacity) - CC
# User CarMake or capacity_kWh as a grouping variable along with ParticipantID
car_count_cc <- cc_final9%>%filter(Trial != 3)%>%
  group_by(capacity_kWh, ParticipantID)%>%
  summarise(count = length(unique(ParticipantID)))%>%
  group_by(capacity_kWh)%>%
  summarise(count = sum(count))

# Call function to sample GF/CC data based on battery capacity bin
model_data(df = gf_final8, feature_name = capacity_kWh, factor_name = "51-75",
           ev_df = ev_gf[, c(1, 4)], df_name = "gf_data_51_75", active_df = active_users_gf[, c(1, 4, 8)])

# Call function to interpolate values for missing dates and create new dataframe
regular_data(data = gf_data_0_25, start_date = "2017-03-17", end_date = "2018-10-27",
             new_data_name = "gf_0_25_interp")

# Summary statistics for GF/CC data (interpolated)
summary(cc_51_75_interp[, 4:9])

# Time Series (all data)
# GF data
gf_0_25_ts <- msts(gf_0_25_interp[, 4:9], seasonal.periods = 7)
gf_26_50_ts <- msts(gf_26_50_interp[, 4:9], seasonal.periods = 7)
gf_51_75_ts <- msts(gf_51_75_interp[, 4:9], seasonal.periods = 7)
gf_76_100_ts <- msts(gf_76_100_interp[, 4:9], seasonal.periods = 7)
```

```
# CC data
cc_0_25_ts <- msts(cc_0_25_interp[, 4:9], seasonal.periods = 7)
cc_26_50_ts <- msts(cc_26_50_interp[, 4:9], seasonal.periods = 7)
cc_51_75_ts <- msts(cc_51_75_interp[, 4:9], seasonal.periods = 7)
cc_76_100_ts <- msts(cc_76_100_interp[, 4:9], seasonal.periods = 7)

# Time series plots (customize y-axis scale based on requirements)
# Plots
p1 <- autoplot(gf_0_25_ts[, 1], col = T, series = "Demand") +
  labs(title ="GF - Demand vs Consumed (0-25)", y ="") +
  theme(panel.background = element_rect(fill = "white", colour = "black")) +
  autolayer(gf_0_25_ts[, 2], series = "Consumed") +
  scale_color_manual(values = rainbow(2)) +
  scale_y_continuous(limits = c(0, 1600), breaks = c(0, 200, 400, 600, 800, 1000, 1200, 1400, 1600))

p2 <- autoplot(cc_0_25_ts[, 1], col = T, series = "Demand") +
  labs(title ="CC - Demand vs Consumed (0-25)", y ="") +
  theme(panel.background = element_rect(fill = "white", colour = "black")) +
  autolayer(cc_0_25_ts[, 2], series = "Consumed") +
  scale_color_manual(values = rainbow(2)) +
  scale_y_continuous(limits = c(0, 1600), breaks = c(0, 200, 400, 600, 800, 1000, 1200, 1400, 1600))

# Combine plots into a grid
gridExtra::grid.arrange(p1, p2, ncol = 1)

# GF/CC data after removing observations in the tail (outliers not processed)
# GF dataset
gf_0_25_int_clean <- gf_0_25_interp[1:572, ]
gf_26_50_int_clean <- gf_26_50_interp[1:573, ]
gf_51_75_int_clean <- gf_51_75_interp[1:573, ]
gf_76_100_int_clean <- gf_76_100_interp[1:472, ]

# CC dataset
cc_0_25_int_clean <- cc_0_25_interp[1:620, ]
cc_26_50_int_clean <- cc_26_50_interp[1:596, ]
cc_51_75_int_clean <- cc_51_75_interp[1:580, ]
cc_76_100_int_clean <- cc_76_100_interp[1:595, ]

# Outliers (GF/CC data)
# Replace outliers with suitable replacements
# Replacements should be chosen after careful inspection of time series patterns
# Suggested replacements shouldn't be used directly without careful inspection of data
# If not satisfied with suggested replacements, replace with NA and then call function for replacements
# List of outliers (indices and suggested replacements)
tsoutliers(msts(gf_51_75_int_clean[, "time"], seasonal.periods = 7))

# Indices of outliers
# If not observation is to be replaced, set index_list = 0
index <- c(342, 538, 539)

# Suggested replacements
# Order of replacements in list: demand > consumed > time > users > trans
# For users and trans, use lowest possible integer greater than or equal to the suggested replacements
# For demand, consumed and time, use numeric values upto one decimal place
replacement <- list(c(NA, NA, NA), c(NA, NA, NA), c(NA, NA, NA),
                    c(7, 8, NA), c(NA, NA, NA))
```

```
# Call function to replace outliers
ts_clean_data(data = gf_51_75_int_clean, index_list = index, replacement_list = replacement, "gf_51_75_clean")

# Plots
# Choose type of plot for GF/CC DCS
p3 <- myplot(data = gf_0_25_clean, x = season, y = users, plot_fun = geom_boxplot(),
             x_label = "", y_label = "Users", plot_title = "GF (0-25)", z = season) +
  scale_color_manual(values = rainbow(4)) +
  theme(legend.position = "none") +
  scale_y_continuous(limits = c(0, 100), breaks = c(0, 25, 50, 75, 100)) +
  scale_x_discrete(limits = c("Winter", "Spring", "Summer", "Autumn"))

# Choose type of plot for GF/CC DCS
p4 <- myplot(data = gf_0_25_clean, x = day, y = users, plot_fun = geom_boxplot(),
             x_label = "", y_label = "Users", plot_title = "", z = day) +
  scale_color_manual(values = rainbow(7)) +
  theme(legend.position = "none") +
  scale_y_continuous(limits = c(0, 100), breaks = c(0, 25, 50, 75, 100)) +
  scale_x_discrete(limits = c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"))

# Combine plots into a grid
gridExtra::grid.arrange(p3, p4, nrow = 2)

# Multiple scatterplots
scatterplot(data = gf_0_25_clean, columns = 4:9, title = "Scatterplot - GF (0-25)",
         column_names = c("Demand", "Consumed", "Time", "Owners", "Users", "Trans"))

# Active proportion of EV owners
active_gf_0_25 <- msts(100 * gf_0_25_clean[, "users"]/gf_0_25_clean[, "owners"], seasonal.periods = 7)
active_gf_26_50 <- msts(100 * gf_26_50_clean[, "users"]/gf_26_50_clean[, "owners"], seasonal.periods = 7)
active_gf_51_75 <- msts(100 * gf_51_75_clean[, "users"]/gf_51_75_clean[, "owners"], seasonal.periods = 7)
active_gf_76_100 <- msts(100 * gf_76_100_clean[, "users"]/gf_76_100_clean[, "owners"], seasonal.periods = 7)

# Summary statistics for active proportion of EV owners
summary(active_gf_51_75)
# Variability of active proportion of EV owners
# Plots
p5 <- autoplot(active_gf_0_25, col = T) +
  labs(x = "", title = "GF - Active Users Prop (0-25)", y = "") +
  theme(panel.background = element_rect(fill = "white", colour = "black"), legend.position = "none")

p6 <- autoplot(active_gf_26_50, col = T) +
  labs(x = "", title = "GF - Active Users Prop (26-50)", y = "") +
  theme(panel.background = element_rect(fill = "white", colour = "black"), legend.position = "none")

p7 <- autoplot(active_gf_51_75, col = T) +
  labs(x = "", title = "GF - Active Users Prop (51-75)", y = "") +
  theme(panel.background = element_rect(fill = "white", colour = "black"), legend.position = "none")

p8 <- autoplot(active_gf_76_100, col = T) +
  labs(x = "", title = "GF - Active Users Prop (76-100)", y = "") +
  theme(panel.background = element_rect(fill = "white", colour = "black"), legend.position = "none")

# Combine plots into a grid
gridExtra::grid.arrange(p5, p6, p7, p8, nrow = 2, ncol = 2)
```

```
## Time Series Analysis ----------------------------------------------------------------------------------------

# Call function to create regular time series with dummies
regular_ts(data = gf_51_75_clean, ts_name = "ts_gf_51_75")

# Summary statistics for GF time series
summary(ts_gf_0_25[, 1:6])

# Time series plots (customize y-axis scale based on requirements)
# Plots
p1 <- autoplot(ts_gf_0_25[, 1], col = T, series = "Demand") +
  labs(title ="GF - Demand vs Consumption (0-25)", y ="") +
  theme(panel.background = element_rect(fill = "white", colour = "black"), legend.position = c(0.2, 0.7)) +
  autolayer(ts_gf_0_25[, 2], series = "Consumed") +
  scale_color_manual(values = rainbow(2)) +
  scale_y_continuous(limits = c(0, 2000), breaks = c(0, 250, 500, 750, 1000, 1250, 1500, 1750, 2000))

p2 <- autoplot(ts_gf_26_50[, 1], col = T, series = "Demand") +
  labs(title ="GF - Demand vs Consumption (26-50)", y ="") +
  theme(panel.background = element_rect(fill = "white", colour = "black"), legend.position = "none") +
  autolayer(ts_gf_26_50[, 2], series = "Consumed") +
  scale_color_manual(values = rainbow(2)) +
  scale_y_continuous(limits = c(0, 2000), breaks = c(0, 250, 500, 750, 1000, 1250, 1500, 1750, 2000))

p3 <- autoplot(ts_gf_51_75[, 1], col = T, series = "Demand") +
  labs(title ="GF - Demand vs Consumption (51-75)", y ="") +
  theme(panel.background = element_rect(fill = "white", colour = "black"), legend.position = "none") +
  autolayer(ts_gf_51_75[, 2], series = "Consumed") +
  scale_color_manual(values = rainbow(2)) +
  scale_y_continuous(limits = c(0, 2000), breaks = c(0, 250, 500, 750, 1000, 1250, 1500, 1750, 2000))

p4 <- autoplot(ts_gf_76_100[, 1], col = T, series = "Demand") +
  labs(title ="GF - Demand vs Consumption (76-100)", y ="") +
  theme(panel.background = element_rect(fill = "white", colour = "black"), legend.position = "none") +
  autolayer(ts_gf_76_100[, 2], series = "Consumed") +
  scale_color_manual(values = rainbow(2)) +
  scale_y_continuous(limits = c(0, 2000), breaks = c(0, 250, 500, 750, 1000, 1250, 1500, 1750, 2000))

# Combine plots into a grid
gridExtra::grid.arrange(p1, p2, p3, p4, nrow = 2, ncol = 2)

# Decomposition of time series data
# Plots
p5 <- mstl(ts_gf_76_100[, 5])%>%autoplot(facet = T, col = T) +
  labs(x = "", title = "GF - Users (76-100)") +
  theme(panel.background = element_rect(fill = "white", colour = "black"), legend.position = "none") +
  scale_color_manual(values = rainbow(4))

p6 <- mstl(ts_gf_76_100[, 6])%>%autoplot(facet = T, col = T) +
  labs(x = "", title = "GF - Transactions (76-100)") +
  theme(panel.background = element_rect(fill = "white", colour = "black"), legend.position = "none") +
  scale_color_manual(values = rainbow(4))

# Combine plots into a grid
gridExtra::grid.arrange(p5, p6, nrow = 1)

# PACF plots
p7 <- ggAcf(ts_gf_0_25[, "demand"]) +
  theme(panel.background = element_rect(fill = "white", colour = "black")) +
  labs(title = "ACF - Demand (0-25)")

p8 <- ggPacf(ts_gf_0_25[, "demand"]) +
  theme(panel.background = element_rect(fill = "white", colour = "black")) +
  labs(title = "PACF - Demand (0-25)")

# Combine plots into a grid
gridExtra::grid.arrange(p7, p8, nrow = 1)

# Tapered ACF
taperedacf(ts_gf_0_25[, "consumed"], main = "GF - Consumed (0-25)", nsim = 50, level = 95)

# Tapered PACF
taperedpacf(ts_gf_0_25[, "consumed"], main = "GF - Consumed (0-25)", nsim = 50, level = 95)
```

```
## k-means clustering ------------------------------------------------------------------------------------------------

# User-wise data
user_data <- gf_final8%>%filter(Trial != 3)%>%
  group_by(ParticipantID)%>%
  summarise(battery = unique(CarkW), rating = unique(CarkWh), total_kWh = sum(ConsumedkWh),
            first_charge = min(date(ActiveChargingStart)), last_charge = max(date(ActiveChargingStop)),
            days_elapsed = as.numeric(difftime(last_charge,first_charge,units="days")),
            charging_count = n(), charge_perday = 100 * charging_count/days_elapsed,
            Avg_kWh_percharge = total_kWh/charging_count,
            Avg_kWh_perday = total_kWh/days_elapsed,
            car_make = unique(CarMake), car_model = unique(CarModel),
            piv = unique(PIVType))%>%
  filter(days_elapsed != 0)%>%
  arrange(first_charge)

# Summary statistics
summary(user_data)

# Numerical user-wise data
user_num <- user_data[, c(3, 10)]

# Use map_dbl to run many models with varying value of k(centers)
set.seed(2019)
tot_withinss <- map_dbl(1:10,  function(k){
  model <- kmeans(x = user_num, centers = k, nstart = 10)
  model$tot.withinss
})

# Generate a data frame containing both k and tot_withinss
elbow_df <- data.frame(k = 1:10, tot_withinss = tot_withinss)

# Elbow plot
myplot(data = elbow_df, x = k, y = tot_withinss, plot_fun = geom_line(), z = , x_label = "Number of Clusters",
       y_label = "Total Within-Cluster Sum of Squares", plot_title = "") +
  scale_x_continuous(breaks = 1:10)

# k-means clustering for k = 4
set.seed(2019)
cluster_kmeans <- kmeans(user_num, 4, nstart = 20)

# Create dataframe with cluster assignments
cluster_users_kmeans <- user_data%>%mutate(cluster = cluster_kmeans$cluster)

# Summarise by clusters
cluster_kmeans_summary <- cluster_users_kmeans%>%group_by(cluster)%>%
  summarise(min_rating = min(rating), max_rating = max(rating), avg_rating = mean(rating),
            avg_kWh_percharge = mean(Avg_kWh_percharge))

# Cluster summary
print(cluster_kmeans_summary)


# Function to create lagged variables for data (maximum number of lags = 7)
lag_variables <- function(data, variable_name, matrix_name){
  # Create a matrix of 7 lagged variables
  lag_var <- cbind(lag0 = stats::lag(data[, variable_name], 0),
                   lag1 = stats::lag(data[, variable_name], -1),
                   lag2 = stats::lag(data[, variable_name], -2),
                   lag3 = stats::lag(data[, variable_name], -3),
                   lag4 = stats::lag(data[, variable_name], -4),
                   lag5 = stats::lag(data[, variable_name], -5),
                   lag6 = stats::lag(data[, variable_name], -6),
                   lag7 = stats::lag(data[, variable_name], -7))

  # Remove NA values
  lag_var2 <- lag_var[-(nrow(lag_var) - 6):-nrow(lag_var), ]

  # Save the matrix of lagged variable
  assign(paste0(matrix_name), lag_var2, inherits = T)
}
```

```
# ------------------------------------------------------------------------------------------------------------------------

# Time series regression with variable origin
ts_reg <-function(data, formula, x_test, y, bin, ...){

  # data = time series data
  # formula = formula for regression
  # x_test = matrix of predictors if mean of forecast objects are used
  # y = response variable for evaluating forecast performance
  # bin = battery capacity bin of time series data
  # ... = optional arguments

  # Matrix to save model AICCc and error metrics
  model <- data.frame(Model = 1:3, AICc = rep(NA, 3), MAE = rep(NA, 3), MAPE = rep(NA, 3))

  # Vector to save forecast objects
  fcast_object_list <- vector(mode = "list", length = 3)

  # Vector to save number of rows for ith test sample
  test_iter <- vector("numeric", 4)
  test_iter[1] <- 0

  # Drop the first 7 observations from data (this eliminates high proportion of active users)
  data2 <- msts(data[-1:-7, ], seasonal.periods = 7)

 # Model fitting with variable origin
 for(i in 1:3){
   # p-value decides train and test split
   # 0.7 <= p <= 0.9
   p = 0.7 + (i - 1)/10
   subset <- floor(p * nrow(data2))

   # Training sample
   train <- msts(data2[1:subset, ], seasonal.periods = 7)

   # Test sample
   test <- msts(data2[-1:-subset, ], seasonal.periods = 7)

   # Test sample row count
   test_iter[i + 1] <- test_iter[i] + nrow(test)

   # Normalization of training and test data
   train_norm <- scale(train, center = apply(train, 2, min),
                       scale = apply(train, 2, max) - apply(train, 2, min))

   test_norm <- scale(test, center = apply(train, 2, min),
                      scale = apply(train, 2, max) - apply(train, 2, min))

   # Model for ith training sample
   fit <- tslm(formula = formula, data = train_norm, ...)

   # AICc for ith model
   model[i, 2] <- CV(fit)[["AICc"]]
```

```
  # Forecast on test data (scenario-based data)
  if(missing(x_test)){
    fcast <- forecast(fit, newdata = data.frame(test_norm))
  }
  # Forecast on forecast object as predictor
  else{
    fcast <- forecast(fit, newdata = x_test[(test_iter[i] + 1):test_iter[i + 1], 1])
  }
  # Assign forecast for ith model to the list
  fcast_object_list[[i]] <- fcast

  # Minimum value of target variable in train data
  minimum <- min(train[, y])

  # Maximum value of target variable in train data
  maximum <- max(train[, y])

  # Difference between max and min of target variable
  multiplier <- maximum - minimum

  # Normalized forecasts as dataframe
  fcast_norm <- as.data.frame(fcast$mean)

  # Empty dataframe to save denormalized forecasts
  fcast_denorm <- as.data.frame(fcast$mean)
  fcast_denorm[, 1] <- NA

  # Denormalization of forecasts
  for(j in 1:nrow(fcast_norm)){
      fcast_denorm[j, 1] <- multiplier*fcast_norm[j, 1] + minimum
    }

   # MAE and MAPE for ith model on test sample
   model[i, 3] <- accuracy(msts(fcast_denorm, seasonal.periods = 7), as.vector(test[, y]))["Test set", "MAE"]
   model[i, 4] <- accuracy(msts(fcast_denorm, seasonal.periods = 7), as.vector(test[, y]))["Test set", "MAPE"]

}
# Save the forecast object list to workspace
assign(paste0("fcast_obj_reg_", y, "_", bin), fcast_object_list, inherits = T)

# AICc and error metrics of all models
print("Model Details")
print(model)

# Mean MAE and MAPE of all models
print("Mean Performance")
print(model%>%summarise(MAE = mean(MAE), MAPE = mean(MAPE)))

# Model on complete data
data3 <- scale(data2, center = apply(data2, 2, min),
               scale = apply(data2, 2, max) - apply(data2, 2, min))

fit_final <- tslm(formula = formula, data = data3, ...)

# Summary details of model on full data
print("Summary of Model")
print(summary(fit_final))

# CV metrics of model
print("CV Metrics of Model")
print(CV(fit_final))

 # Residuals of model
 print("Residual Analysis")
 checkresiduals(fit_final)

 # Mean of residuals
 print("Mean of Residuals")
 print(mean(fit_final$residuals))

 # Save the final model to workspace
 assign(paste0("fit_reg_", y, "_", bin), fit_final, inherits = T)

 # Combine the forecasts from all the lists into a single dataframe
 fcasts_list <- vector("list")
 for(i in 1:3){
   fcasts_list[[i]] <- data.frame(as.numeric(fcast_object_list[[i]][["mean"]]))
 }
 fcast_list <- rbind(fcasts_list[[1]], fcasts_list[[2]], fcasts_list[[3]])
 names(fcast_list) <- paste0(y)

 # Save the forecast list to workspace
 assign(paste0("fcast_reg_", y, "_", bin), fcast_list, inherits = T)
}
```

```
# ------------------------------------------------------------------------------------------------------------------------

# Function to build regression model with arima errors
ts_reg_arima <- function(data, y, x, xreg_test, bin, ...){

  # data = time series data
  # y = response variable
  # x = vector of predictors
  # xreg_test = matrix of predictors if mean of forecast objects are used
  # bin = battery capacity bin of time series data
  # ... = optional arguments depending on model requirements

  # Matrix to save model AICCc and error metrics
  model <- data.frame(Model = 1:3, AICc = rep(NA, 3), MAE = rep(NA, 3), MAPE = rep(NA, 3))

  # Vector to save forecast objects
  fcast_object_list <- vector(mode = "list", length = 3)

  # Vector to save number of rows for ith test sample
  test_iter <- vector("numeric", 4)
  test_iter[1] <- 0

  # Drop the first 7 observations from data (this eliminates high proportion of active users)
  data2 <- msts(data[-1:-7, ], seasonal.periods = 7)

  # Model fitting with variable origin
  for(i in 1:3){
    # p-value decides train and test split
    # 0.7 <= p <= 0.9
    p = 0.7 + (i - 1)/10
    subset <- floor(p * nrow(data2))

   # Training sample
   train <- msts(data2[1:subset, ], seasonal.periods = 7)

   # Test sample
   test <- msts(data2[-1:-subset, ], seasonal.periods = 7)

   # Test sample row count
   test_iter[i + 1] <- test_iter[i] + nrow(test)

   # Normalization of training and test data
   train_norm <- scale(train, center = apply(train, 2, min),
                      scale = apply(train, 2, max) - apply(train, 2, min))

   test_norm <- scale(test, center = apply(train, 2, min),
                     scale = apply(train, 2, max) - apply(train, 2, min))

   # Forecast model
   fit <- auto.arima(train_norm[, y], approximation = F, stepwise = F, xreg  = train_norm[, x], ...)

   # AICc for ith model
   model[i, 2] <- fit[["aicc"]]

   # Forecast on test data (scenario-based data)
   if(missing(xreg_test)){
     fcast <- forecast(fit, xreg = test_norm[, x])
   }
   # Forecast on forecast object as predictor
   else{
     fcast <- forecast(fit, xreg = xreg_test[(test_iter[i] + 1):test_iter[i + 1], 1])
   }
   # Assign forecast for ith model to the list
   fcast_object_list[[i]] <- fcast
```

```
  # Minimum value of target variable in train data
  minimum <- min(train[, y])

  # Maximum value of target variable in train data
  maximum <- max(train[, y])

  # Difference between max and min of target variable
  multiplier <- maximum - minimum

  # Normalized forecasts as dataframe
  fcast_norm <- as.data.frame(fcast$mean)

  # Empty dataframe to save denormalized forecasts
  fcast_denorm <- as.data.frame(fcast$mean)
  fcast_denorm[, 1] <- NA

  # Denormalization of forecasts
  for(j in 1:nrow(fcast_norm)){
    fcast_denorm[j, 1] <- multiplier*fcast_norm[j, 1] + minimum
  }

  # MAE and MAPE for ith model on test sample
  model[i, 3] <- accuracy(msts(fcast_denorm, seasonal.periods = 7), as.vector(test[, y]))["Test set", "MAE"]
  model[i, 4] <- accuracy(msts(fcast_denorm, seasonal.periods = 7), as.vector(test[, y]))["Test set", "MAPE"]

}
# Save the forecast object list to workspace
assign(paste0("fcast_obj_ra_", y, "_", bin), fcast_object_list, inherits = T)

# AICc and error metrics of all models
print("Model Details")
print(model)

# Mean MAE and MAPE of all models
print("Mean Performance")
print(model%>%summarise(MAE = mean(MAE), MAPE = mean(MAPE)))

# Model on complete data
data3 <- scale(data2, center = apply(data2, 2, min),
               scale = apply(data2, 2, max) - apply(data2, 2, min))

fit_final <- auto.arima(data3[, y], approximation = F, stepwise = F, xreg  = data3[, x], ...)

# Summary details of model on full data
print("Summary of Model")
summary(fit_final)

# Residuals of model
print("Residual Analysis")
checkresiduals(fit_final)

# Mean of residuals
print("Mean of Residuals")
print(mean(fit_final$residuals))

# Save the final model to workspace
assign(paste0("fit_ra_", y, "_", bin), fit_final, inherits = T)

# Combine the forecasts from all the lists into a single dataframe
fcasts_list <- vector("list")
for(i in 1:3){
  fcasts_list[[i]] <- data.frame(as.numeric(fcast_object_list[[i]][["mean"]]))
}
fcast_list <- rbind(fcasts_list[[1]], fcasts_list[[2]], fcasts_list[[3]])
names(fcast_list) <- paste0(y)
```

```
  # Save the forecast list to workspace
  assign(paste0("fcast_ra_", y, "_", bin), fcast_list, inherits = T)
}

# ------------------------------------------------------------------------------------------------------------------------

# Function to build regression model with arima errors
ts_dl <- function(data, d_lag, y, x, xreg_test, bin, ...){

  # data = time series data
  # d_lag = distributed lags pf predictors
  # y = response variable
  # x = vector of predictors
  # xreg_test = matrix of predictors if mean of forecast objects are used
  # bin = battery capacity bin of time series data
  # ... = optional arguments depending on model requirements

  # Matrix to save model AICCc and error metrics
  model <- data.frame(Model = 1:3, AICc = rep(NA, 3), MAE = rep(NA, 3), MAPE = rep(NA, 3))

  # Vector to save forecast objects
  fcast_object_list <- vector(mode = "list", length = 3)

  # Vector to save number of rows for ith test sample
  test_iter <- vector("numeric", 4)
  test_iter[1] <- 0

  # Combine main data with DL and drop first 7 obervations
  data2 <- msts(cbind(data, d_lag), seasonal.periods = 7)[-1:-7, ]

 # Rename columns of data with DL
 colnames(data2)[c(1:ncol(data), (ncol(data) + 1):ncol(data2))] <-
   c(colnames(data), c("lag0", "lag1", "lag2", "lag3", "lag4", "lag5", "lag6", "lag7"))

 # Model fitting with variable origin
 for(i in 1:3){
   # p-value decides train and test split
   # 0.7 <= p <= 0.9
   p = 0.7 + (i - 1)/10
   subset <- floor(p * nrow(data2))

   # Training sample
   train <- msts(data2[1:subset, ], seasonal.periods = 7)

   # Test sample
   test <- msts(data2[-1:-subset, ], seasonal.periods = 7)

   # Normalization of training and test data
   train_norm <- scale(train, center = apply(train, 2, min),
                       scale = apply(train, 2, max) - apply(train, 2, min))

   test_norm <- scale(test, center = apply(train, 2, min),
                      scale = apply(train, 2, max) - apply(train, 2, min))

   # Forecast model
   fit <- auto.arima(train_norm[, y], approximation = F, stepwise = F, xreg  = train_norm[, x], ...)

   # AICc for ith model
   model[i, 2] <- fit[["aicc"]]
```

```
# Forecast
if(missing(xreg_test)){
  fcast <- forecast(fit, xreg = test_norm[, x])

  # Minimum value of target variable in train data
  minimum <- min(train[, y])

  # Maximum value of target variable in train data
  maximum <- max(train[, y])

  # Difference between max and min of target variable
  multiplier <- maximum - minimum

  # Normalized forecasts as dataframe
  fcast_norm <- as.data.frame(fcast$mean)

  # Empty dataframe to save denormalized forecasts
  fcast_denorm <- as.data.frame(fcast$mean)
  fcast_denorm[, 1] <- NA

  # Denormalization of forecasts
  for(j in 1:nrow(fcast_norm)){
    fcast_denorm[j, 1] <- multiplier*fcast_norm[j, 1] + minimum
  }

  # MAE and MAPE for ith model on test sample
  model[i, 3] <- accuracy(msts(fcast_denorm, seasonal.periods = 7), as.vector(test[, y]))["Test set", "MAE"]
  model[i, 4] <- accuracy(msts(fcast_denorm, seasonal.periods = 7), as.vector(test[, y]))["Test set", "MAPE"]


}
else{
  xreg_test_df <- data.frame(xreg_test)
  xreg_col <- ncol(xreg_test_df)
  na_drop <- sum(is.na(xreg_test_df[, xreg_col]))

  if(na_drop != 0){
    xreg_test2 <- xreg_test[-1:-(na_drop),]
    test2 <- test[-1:-(na_drop), ]

    # Test sample row count
    test_iter[i + 1] <- test_iter[i] + nrow(test2)
    fcast <- forecast(fit, xreg = xreg_test2[(test_iter[i] + 1):test_iter[i + 1],])

    # Minimum value of target variable in train data
    minimum <- min(train[, y])

    # Maximum value of target variable in train data
    maximum <- max(train[, y])

    # Difference between max and min of target variable
    multiplier <- maximum - minimum

    # Normalized forecasts as dataframe
    fcast_norm <- as.data.frame(fcast$mean)

    # Empty dataframe to save denormalized forecasts
    fcast_denorm <- as.data.frame(fcast$mean)
    fcast_denorm[, 1] <- NA
```

```
      # Denormalization of forecasts
      for(j in 1:nrow(fcast_norm)){
        fcast_denorm[j, 1] <- multiplier*fcast_norm[j, 1] + minimum
      }

      # MAE and MAPE for ith model on test sample
      model[i, 3] <- accuracy(msts(fcast_denorm, seasonal.periods = 7), as.vector(test2[, y]))["Test set", "MAE"]
      model[i, 4] <- accuracy(msts(fcast_denorm, seasonal.periods = 7), as.vector(test2[, y]))["Test set", "MAPE"]

    }
    else{
      # Test sample row count
      test_iter[i + 1] <- test_iter[i] + nrow(test)
      fcast <- forecast(fit, xreg = xreg_test_df[(test_iter[i] + 1):test_iter[i + 1],])

      # Minimum value of target variable in train data
      minimum <- min(train[, y])

      # Maximum value of target variable in train data
      maximum <- max(train[, y])

      # Difference between max and min of target variable
      multiplier <- maximum - minimum

      # Normalized forecasts as dataframe
      fcast_norm <- as.data.frame(fcast$mean)

      # Empty dataframe to save denormalized forecasts
      fcast_denorm <- as.data.frame(fcast$mean)
      fcast_denorm[, 1] <- NA

      # Denormalization of forecasts
      for(j in 1:nrow(fcast_norm)){
        fcast_denorm[j, 1] <- multiplier*fcast_norm[j, 1] + minimum
      }

       # MAE and MAPE for ith model on test sample
       model[i, 3] <- accuracy(msts(fcast_denorm, seasonal.periods = 7), as.vector(test[, y]))["Test set", "MAE"]
       model[i, 4] <- accuracy(msts(fcast_denorm, seasonal.periods = 7), as.vector(test[, y]))["Test set", "MAPE"]

    }
  }
  # Assign forecast for ith model to the list
  fcast_object_list[[i]] <- fcast
}
# Save the forecast object list to workspace
assign(paste0("fcast_obj_dl_", y, "_", bin), fcast_object_list, inherits = T)

# AICc and error metrics of all models
print("Model Details")
print(model)

# Mean MAE and MAPE of all models
print("Mean Performance")
print(model%>%summarise(MAE = mean(MAE), MAPE = mean(MAPE)))

# Model on complete data
data3 <- scale(data2, center = apply(data2, 2, min),
               scale = apply(data2, 2, max) - apply(data2, 2, min))

fit_final <- auto.arima(data3[, y], approximation = F, stepwise = F, xreg  = data3[, x], ...)

# Summary details of model on full data
print("Summary of Model")
summary(fit_final)

# Residuals of model
print("Residual Analysis")
checkresiduals(fit_final)
```

```
  # Mean of residuals
  print("Mean of Residuals")
  print(mean(fit_final$residuals))

  # Save the final model to workspace
  assign(paste0("fit_dl_", y, "_", bin), fit_final, inherits = T)

  # Combine the forecasts from all the lists into a single dataframe
  fcasts_list <- vector("list")
  for(i in 1:3){
    fcasts_list[[i]] <- data.frame(as.numeric(fcast_object_list[[i]][["mean"]]))
  }
  fcast_list <- rbind(fcasts_list[[1]], fcasts_list[[2]], fcasts_list[[3]])
  names(fcast_list) <- paste0(y)

  # Save the forecast list to workspace
  assign(paste0("fcast_dl_", y, "_", bin), fcast_list, inherits = T)
}

# ------------------------------------------------------------------------------------------------------------------------

# Function to build LSTM models
lstm <- function(data, x, y, x_fcast, n, y_name, bin){

  # data = data for modelling
  # x = predictors
  # y = target variable
  # x_fcast = forecast as predictor
  # n = number of epochs
  # y_name = name of target variable
  # bin = battery capacity bin

  # Matrix to save model error metrics
  model <- data.frame(Model = 1:3, MAE = rep(NA, 3), MAPE = rep(NA, 3))

  # Vector to save forecasts
  fcast_list <- vector(mode = "list", length = 3)

  # Vector to save number of rows for ith test sample
  test_iter <- vector("numeric", 4)
  test_iter[1] <- 0

  # Convert the time series into matrix
  data2 <- as.matrix(data)

  # Remove the column names to prepare the data for NNs
  dimnames(data2) <- NULL

  for(i in 1:3){

    # p-value decides train and test split
    # 0.7 <= p <= 0.9
    p = 0.7 + (i - 1)/10
    subset <- floor(p * nrow(data2))

    # Training sample
    train <- as.matrix(data2[1:subset, ])

    # Test sample
    test <- as.matrix(data2[-1:-subset, ])

    # Test sample row count
    test_iter[i + 1] <- test_iter[i] + nrow(test)

    # Normalization of training and test data
    train_norm <- scale(train, center = apply(train, 2, min),
                        scale = apply(train, 2, max) - apply(train, 2, min))

    test_norm <- scale(test, center = apply(train, 2, min),
                       scale = apply(train, 2, max) - apply(train, 2, min))

    # Create targets and features for training and test samples
    # Test sample target is not normalized for performance evaluation
    x_train <- as.matrix(train_norm[, x])
    y_train <- as.matrix(train_norm[, y])
    x_test <- as.matrix(test_norm[, x])
    y_test <- as.matrix(test[, y])
```

```
  # Convert matrices to 3D arrays to input to NNs
  dim(x_train) <- c(nrow(x_train), 1, ncol(x_train))
  dim(x_test) <- c(nrow(x_test), 1, ncol(x_test))

  # Define the model for ith training sample
  model_keras <- keras_model_sequential()%>%
    layer_lstm(units = 50, activation = "relu", dropout = 0.1, recurrent_dropout = 0.1, return_sequences = T,
               input_shape = c(dim(x_train)[[2]], dim(x_train)[[3]]))%>%
    layer_lstm(units = 50, activation = "relu", dropout = 0.1, recurrent_dropout = 0.1)%>%
    layer_dense(units = 1)%>%
    compile(loss = "mse", optimizer = optimizer_rmsprop(), metrics = c("mae", "mape"))

  # Train the model
  fit_keras <- model_keras%>%
    fit(x = x_train, y = y_train, epochs = n, validation_split = 0, verbose = 0, shuffle = F, batch_size = 7)

  # Forecast on test data (scenario-based data)
  if(missing(x_fcast)){
    fcast_list[[i]] <- as.matrix(predict(model_keras, x_test))
  }
  # Forecast on forecast as predictor
  else{
    x_fcast2 <- x_fcast[(test_iter[i] + 1):test_iter[i + 1]]
    dim(x_fcast2) <- c(length(x_fcast2), 1, 1)
    fcast_list[[i]] <- as.matrix(predict(model_keras, x_fcast2))
  }

  # Minimum value of target variable in train data
  minimum <- min(as.matrix(train[, y]))

  # Maximum value of target variable in train data
  maximum <- max(as.matrix(train[, y]))

 # Difference between max and min of target variable
 multiplier <- maximum - minimum

 # Normalized forecasts as dataframe
 fcast_norm <- as.data.frame(fcast_list[[i]])

 # Empty dataframe to save denormalized forecasts
 fcast_denorm <- as.data.frame(fcast_list[[i]])
 fcast_denorm[, 1] <- NA

 # Denormalization of forecasts
 for(j in 1:nrow(fcast_norm)){
   fcast_denorm[j, 1] <- multiplier*fcast_norm[j, 1] + minimum
 }

 # Evaluate model accuracy
 eval <- accuracy(msts(fcast_denorm, seasonal.periods = 7), y_test)["Test set", c("MAE", "MAPE")]

 # MAE and MAPE for ith test sample
 model[i, 2] <- eval[[1]]
 model[i, 3] <- eval[[2]]
}

# Error metrics of all models
print("Model Details")
print(model)

# Mean MAE and MAPE of all models
print("Mean Performance")
print(model%>%summarise(MAE = mean(MAE), MAPE = mean(MAPE)))
```

```
  # Combine all the forecasts into a 1D array
  fcast <- rbind(fcast_list[[1]], fcast_list[[2]], fcast_list[[3]])

  # Save the forecast list to workspace
  assign(paste0("fcast_lstm_", y_name, "_", bin), fcast, inherits = T)

  # Model on complete data
  data3 <- scale(data2, center = apply(data2, 2, min),
                 scale = apply(data2, 2, max) - apply(data2, 2, min))

  # Create targets and features for training sample
  x_train_final <- as.matrix(data3[, x])
  y_train_final <- as.matrix(data3[, y])

  # Convert matrices to 3D arrays to input to NNs
  dim(x_train_final) <- c(nrow(x_train_final), 1, ncol(x_train_final))

  # Model on complete data
  model_keras <- keras_model_sequential()%>%
    layer_lstm(units = 50, activation = "relu", dropout = 0.1, recurrent_dropout = 0.1, return_sequences = T,
               input_shape = c(dim(x_train_final)[[2]], dim(x_train_final)[[3]]))%>%
    layer_lstm(units = 50, activation = "relu", dropout = 0.1, recurrent_dropout = 0.1)%>%
    layer_dense(units = 1)%>%
    compile(loss = "mse", optimizer = optimizer_rmsprop(), metrics = c("mae", "mape"))

  # Train the model
  fit_keras_final <- model_keras%>%
    fit(x = x_train_final, y = y_train_final, epochs = n, validation_split = 0, verbose = 0, shuffle = F,
        batch_size = 7)

  # Save the model to workspace
  assign(paste0("fit_lstm_", y_name, "_", bin), fit_keras_final, inherits = T)
}


# set up parallel processing

install.packages("doParallel")
library(doParallel)

# set parallel processing
registerDoParallel(cl = makePSOCKcluster(2))

# Call function for time series regression with variable origin
# Leave x-test blank in case forecasts are not used as predictors
# Set x_test = data.frame(predictor = fcast_reg_variable_bin) in case forecasts are used as predictors
# Specify bin using "_" between the bin limits
ts_reg(data = ts_gf_51_75,
       formula = consumed ~ demand,
       x_test = data.frame(demand = fcast_reg_demand_51_75), y = "consumed", bin = "51_75")

# ------------------------------------------------------------------------------------------------------------------------

# Run all the functions together to compute the run-time
# Start timer
tic()
# Regression with arima errors
# Call function to forecast using regression with arima errors
# Set xreg_test = fcast_ra_predictor_bin if and only if mean forecasts are used as predictors
# For transformation of response variable, specify value of lambda based on requirements and set biasadj = T
ts_reg_arima(data = ts_gf_51_75,
             y = "consumed",
             x = "trans",
             xreg_test = fcast_ra_trans_51_75,
             bin = "51_75",
             lambda = , biasadj = )
# Stop timer
toc()
```

```
# ------------------------------------------------------------------------------------------------------------------------

## Distributed lag models (DL)
# Create lag variables for train data
lag_variables(data = ts_gf_26_50, variable_name = "owners", matrix_name = "owners_lag")

lag_variables(data = ts_gf_26_50, variable_name = "users", matrix_name = "users_lag")

lag_variables(data = ts_gf_26_50, variable_name = "trans", matrix_name = "trans_lag")

lag_variables(data = ts_gf_26_50, variable_name = "time", matrix_name = "time_lag")

lag_variables(data = ts_gf_26_50, variable_name = "demand", matrix_name = "demand_lag")

# Create lag variables for forecasts (create only after the forecasts are generated)
# Set data= msts(forecast_name, seasonal.periods = 7)
lag_variables(data = msts(fcast_dl_users_26_50, seasonal.periods = 7), variable_name = "users",
              matrix_name = "fcast_users_lag")

lag_variables(data = msts(fcast_dl_trans_26_50, seasonal.periods = 7), variable_name = "trans",
              matrix_name = "fcast_trans_lag")

lag_variables(data = msts(fcast_dl_time_26_50, seasonal.periods = 7), variable_name = "time",
              matrix_name = "fcast_time_lag")

lag_variables(data = msts(fcast_dl_demand_26_50, seasonal.periods = 7), variable_name = "demand",
              matrix_name = "fcast_demand_lag")

# Run all the functions together to compute the run-time
# Start timer
tic()
# Call function to forecast using DL
# Set xreg_test = fcast_predictor_lag[, vector of columns] if and only if forecasts are used as predictors
ts_dl(data = ts_gf_26_50,
      d_lag = demand_lag,
      y = "consumed",
      x = c(18),
      xreg_test = fcast_demand_lag[, 1],
      bin = "26_50",
      lambda = , biasadj = )
# Stop timer
toc()

# ------------------------------------------------------------------------------------------------------------------------

## LSTM models
library(keras)
use_session_with_seed(2019)
install_keras()

# Create lag variables for train data
lag_variables(data = ts_gf_26_50, variable_name = "users", matrix_name = "users_lag")

# Call function to create time series data with lags
# Output = ts_data
lag_data(ts_gf_26_50, users_lag)

# data = time series data (ts_data)
# x = vector of columns to subset predictors from feature space
# y = column to subset target

data <- ts_data
x <- c(5)
y <- c(1)

# Run all the functions together to compute the run-time
# Start timer
tic()
# Call lstm function to build model
# Set x_fcast = fcast_lstm_predictor_bin if and only if mean forecasts are used as predictors
# Leave x_fcast blank if mean forecasts are not used as predictors
set.seed(2019)
lstm(data = data,
     x = x,
     y = y,
     x_fcast = fcast_lstm_users_26_50,
     n = 50,
     y_name = "demand",
     bin = "26_50")
# Stop timer
toc()

# ------------------------------------------------------------------------------------------------------------------------
```

```
## Predictions on user-defined scenario-based data
# Function to build LSTM models
lstm_pred <- function(train_data, x, y, n){

  # train_data = data for modelling
  # x = predictors
  # y = target variable
  # n = number of epochs

  # Convert the time series into matrix
  data2 <- as.matrix(train_data)

  # Remove the column names to prepare the data for NNs
  dimnames(data2) <- NULL

  # Create targets and features for training and test samples
  # Test sample target is not normalized for performance evaluation
  x_train <- as.matrix(data2[, x])
  y_train <- as.matrix(data2[, y])

  # Convert matrices to 3D arrays to input to NNs
  dim(x_train) <- c(nrow(x_train), 1, ncol(x_train))

  # Define the model for ith training sample
  model_keras_lstm <<- keras_model_sequential()%>%
    layer_lstm(units = 50, activation = "relu", dropout = 0.1, recurrent_dropout = 0.1, return_sequences = T,
               input_shape = c(dim(x_train)[[2]], dim(x_train)[[3]]))%>%
    layer_lstm(units = 50, activation = "relu", dropout = 0.1, recurrent_dropout = 0.1)%>%
    layer_dense(units = 1)%>%
    compile(loss = "mse", optimizer = optimizer_rmsprop(), metrics = c("mae", "mape"))

  # Train the model
  fit_keras_lstm <<- model_keras_lstm%>%
    fit(x = x_train, y = y_train, epochs = n, validation_split = 0, verbose = 0, shuffle = F, batch_size = 7)

}

# prepare train data
train_data <- ts_data
x <- c(4, 7:9, 12:17)
y <- c(2)
n <- 50

# call function to build LSTM model
lstm_pred(train_data, x, y, n)

# scenario of Lancaster (2022)
scenario_data(start_date = "2022/12/19", end_date = "2022/12/25", owners = 1145,
              scenario_data_name = "data_0_25")

# scenario-based data converted to all numeric
season_data(data_0_25, "winter_0_25")

# prepare test data
test_data <- as.matrix(winter_0_25[, c(1:7, 10:12)])
dimnames(test_data) <- NULL
dim(test_data) <- c(nrow(test_data), 1, ncol(test_data))

# Forecast on test data (scenario-based data)
fcast <<- as.matrix(predict(model_keras_lstm, test_data))
```