Comparing the Effectiveness of LSTM, ARIMA, and GRU Algorithms for Forecasting Customer Charging Behavior in the Electric Mobility Industry in Europe

WITS
UNIVERSITY

Masters Dissertation

Robyne Chimere Pelwan
Student Number 0502168A
Supervised by Dr Clint van Alten
School of Computer Science and Applied Mathematics
University of the Witwatersrand, Johannesburg
August 20, 2023

Acknowledgement

I would like to extend my sincerest appreciation to my advisor, Dr Clint van Alten, for his advice, guidance, and remarkable patience throughout the research and writing process.

I am grateful to my brother, Chad Pelwan; my wife, Bianca van der Westhuizen; my parents, Nathalian and Bernardette Pelwan, and my friend, Hussein Jundi. Thank you for your support, encouragement and unconditional love.

To David Klinck. Thank you for moulding me into the analyst that I am. Rest in Peace.

Abstract

Forecasting, a powerful technique for unveiling potential future events, relies on historical data and methodological approaches to provide valuable insights. This dissertation delves into the domain of electric mobility, investigating the effectiveness of three distinct algorithms—Long Short-term Memory (LSTM), Autoregressive Integrated Moving Average (ARIMA), and Gated Recurrent Unit (GRU)—for predicting customer charging behavior. Specifically, it focuses on forecasting the number of charges over a 7-day period using time-series data from European electric mobility customers. In this study, we scrutinize the interplay between algorithmic performance and the intricacies of the dataset. Root mean squared error (RMSE) serves as a metric for gauging predictive accuracy. The findings highlight the supremacy of the ARIMA model in single-variable analysis, surpassing the predictive capabilities of both LSTM and GRU models. Even when additional features are introduced to enhance LSTM and GRU predictions, the superiority of ARIMA remains pronounced. Notably, this research underscores that ARIMA is particularly well-suited for time series data of this nature due to its tailored design. It contributes valuable insights for both researchers and practitioners in the electric mobility industry, aiding in the strategic selection of forecasting methodologies.

Declaration

I, Robyne Chimere Pelwan, (Student Number: 0502168A) hereby declare the contents of this dissertation to be my own work. This dissertation is submitted for a Masters of Science degree at the University of the Witwatersrand, Johannesburg. The work has not been submitted to any other university, or for any other degree.

Robyne Pelwan
Munich, Germany
August 20, 2023

Contents

Acronyms

AC Alternating Current. 43

ACF Complete Auto-Correlation Function. 17, 40, 72

ADF Augmented Dickey-Fuller. 16, 70

AIC Akaike Information Criterion. 16, 41, 73

ANN Artificial Neural Network. 29

ARIMA Autoregressive Integrated Moving Average. 2, 3, 13, 14, 70

BEV Battery Electric Vehicle. 13, 35, 43

BIC Bayesian Information Criterion. 16, 41

COVID-19 Coronavirus Disease 2019. 2, 30, 46

CSV Comma-Separated Values. 40

DC Direct Current. 43

DNN Deep Neural Network. 30

DWH Data Warehouse. 35, 40

ERD Entity Relationship Diagram. 35

EU European Union. 36

GDPR General Data Protection Regulation. 36

GRU Gated Recurrent Units. 2–4, 13, 14, 21, 63, 90, 106

KPI Key Performance Indicators. 118

LSTM Long Short-Term Memory. 2–4, 13, 14, 18, 53, 83, 100

MAE Mean Absolute Error. 23

MAPE Mean Absolute Percentage Error. 30

MSE Mean Square Error. 23

OEM Original Equipment Manufacturer. 46

PACF Partial Auto-Correlation Function. 17, 40, 72

PCA Principal Component Analysis. 26

PHEV Plug-In Hybrid Electric Vehicle. 13, 35, 43

RFE Recursive Feature Elimination. 27

RMSE Root Mean Squared Error. 23, 60

RNN Recurrent Neural Network. 18, 21, 116

List of Figures

List of Tables

Listings

List of Algorithms

1 Introduction

Forecasting stands as a pivotal instrument, unveiling the veiled trails of an uncertain future. It illuminates intricate patterns, deciphering the subtleties within data to lay bare potential trajectories. At the intersection of historical data and external influences resides the heart of precision prediction. This study embarks on an exploration into the efficacy of three distinct algorithms – Long Short-Term Memory (LSTM), Autoregressive Integrated Moving Average (ARIMA), and Gated Recurrent Units (GRU) – in forecasting customer charging behavior [Hochreiter and Schmidhuber 1997; Box and Jenkins 1976; Cho et al. 2014].

Fueled by the surging global electric vehicle market that saw a remarkable 42.9% surge in sales during 2020, the quest for accurate predictions in electric vehicle charging becomes increasingly paramount. The nucleus of this investigation lies in comprehending the unique capabilities of ARIMA, LSTM, and GRU in the context of forecasting time-series data, underpinned by their distinct algorithmic constructs. Through a comparative lens that accentuates outcome-based analysis over algorithmic intricacies, this research aims to unveil the most adept model for predicting customer charging behavior within the electric mobility sector.

Our findings spotlight the ARIMA model's supremacy in single-variable analysis, outperforming LSTM and GRU with lower root mean squared error (RMSE) and heightened predictive accuracy. Despite subjecting LSTM and GRU to augmented datasets encompassing additional features, ARIMA's favorable performance persists. This not only underscores ARIMA's resonance with this specific time-series data but also carries implications for practical decision-making within the realm of electric mobility. Beyond academic considerations, these revelations resonate in operational streamlining, resource allocation, and the intricate interplay of algorithmic evaluations, model selections, and the influence of external variables.

Embedded in this inquiry is the pivotal role of a third-party service provider, "The Company," facilitating access to charging stations for Battery Electric Vehicle (BEV) and Plug-In Hybrid Electric Vehicle (PHEV) users, championing a seamless charging experience and a maximal number of charges over time.

Structured as a multi-chapter expedition, each successive section delves deeper into the research odyssey. Chapter 2 bestows a theoretical foundation concerning the three algorithms and their architectural underpinnings. Chapter 3 meticulously navigates the dataset's genesis and transformation, while Chapter 4 meticulously dissects the influence of data. Algorithmic examinations come to the forefront in Chapters 5, 6, and 7, culminating in Chapter 8's encompassing synthesis.

2 Background and Related Work

2.1 Introduction

Stochastic models and Neural Networks have the capability to learn and generate outputs beyond the scope of the input data that they are presented with. Specifically related to this thesis proposal, these models have been imperative in forecasting probable future occurrences based on time series data. This chapter aims to introduce the three models we wish to apply in our analysis, namely Autoregressive Integrated Moving Average (ARIMA), Gated Recurrent Units (GRU), and Long Short-Term Memory (LSTM). Additionally, this chapter includes the various performance measurements that can be used to assess each of the models. Whilst these measurements are commonly used to assess regression models, it is not necessary to use all of them, but rather one or a combination of some that best fit the data at hand. In the different comparative studies that have been documented below, it can be seen that different performance measurements were used as it was dependent on the analysis at hand. Coincidentally, each of the studies produced various results, and no one model has been found to be superior to all others. The importance of these studies is to enable future analysts to have the knowledge to best choose which forecasting model to use upfront, based on the data they wish to project, without having to develop all other models.

2.2 Autoregressive Integrated Moving Average (ARIMA)

Box and Jenkins [1976] introduced the Autoregressive Integrated Moving Average (ARIMA) (also known as Box and Jenkins forecasting), which is a stochastic model for time series data that merges the Autoregressive (AR) and Moving Average (MA) processes to construct a comprehensive model. ARIMA can capture complex relationships within data by measuring the degree of dependency between one dependent variable and other changing variables [Chen 2019]. Since ARIMA does not assume any underlying models or relationships, it is widely used to forecast time series. A focal point of this methodology is that it relies on the past values of the input data and previously calculated error values to determine a forecasted value.

The acronym $ARIMA(p, d, q)$ can be explained using the three essential components of the model:

AR Autoregressive process refers to a type of stochastic process where the value of a time series at a given point in time is based on its past values. The AR component (p) is the number of past observations used to predict future values.

I Integrated process refers to the act of differencing a time series to make it stationary. Stationarity is an important property of time series data as it denotes that the statistical properties of the data (such as mean and variance) do not change over time. The I component d represents the number of times the differencing operation is performed.

MA Moving Average process refers to a type of stochastic process, where the value of a time series at a given point in time, is based on its past errors. The MA component q is the number of lagged errors used to predict future values of the time series.

Moving Average. An approach that considers the dependency between observations and the residual error terms from the moving average applied to lagged observations (q)

The AR model of order p, i.e., $AR(p)$ can be written in the form:

$$x_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + \varepsilon_t \tag{2.1}$$

where x_t is the stationary variable, c is the constant, the terms in ϕ_i are autocorrelation coefficients at lags 1, 2, ..., p. It is assumed that ε_t are the residuals of a Gaussian white noise series with mean zero and variance σ_ε^2.

The MA model of order q, $MA(q)$, is defined as :

$$x_t = \mu + \sum_{i=0}^{q} \theta_i \varepsilon_{t-i} \tag{2.2}$$

where μ is the expectation of x_t (assumed to be zero), the terms for θ_i are the weights applied to the current and prior values of the stochastic term in the time series with $\theta_0 = 1$. ε_t are the residuals of a Gaussian white noise series that have a mean zero and variance σ_ε^2.
These two models are then combined by addition forming the ARIMA model of order (p, q) as is defined as:

$$x_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + \varepsilon_t + \mu + \sum_{i=0}^{q} \theta_i \varepsilon_{t-i} \tag{2.3}$$

where $\phi_i \neq 0$, $\theta_i \neq 0$, and $\sigma_\varepsilon^2 > 0$. The parameters p and q are called the AR and MA orders, respectively.
For the ARIMA model to be successful, a few assumptions about the data and the model parameters are as follows:

The Time Series Data Does Not Contain Anomalies
Performing a descriptive analysis of the data to ensure that there are no erroneous data will alleviate this problem. Should any data fields be empty, these should be appropriately filled in, given the field requirements.

The Time Series Data is Stationary and Non-Seasonal
According to [Hyndman and Athanasopoulos 2018], a stationary time series is one that exhibits consistent statistical properties regardless of the time when the observations are taken. Seasonality, which refers to a predictable pattern of changes or trends that repeats over a set number of time periods, often causes a time series to be non-stationary due to variations in average values across certain times within the seasonal span. To transform a non-stationary time series into a stationary one, *differencing* can be applied by computing the differences between consecutive observations.

The Model Parameters and Error Term are Constant

Historic Data Dictates the Behaviour of Current and Future Time Points
By only being able to predict using the defined input data, the predicted values may not hold in stressed market data conditions.

2.2.1 Hyperparameter Optimization

In the modelling process, it is essential to determine the optimal values for the hyperparameters that govern the Autoregression (AR), Integration (I), and Moving Average (MA) components. This process, which involves finding the best values for these hyperparameters based on a given set of data, is known as hyperparameter optimization or tuning. The value for I depends on whether the data set is stationary or not. To assess stationarity, one can apply the Augmented Dickey-Fuller (ADF) test, which yields the appropriate value for I. Once I is determined, additional statistical tests, such as the Autocorrelation Function (ACF) and the Partial Autocorrelation Function (PACF), can be utilized to select candidate values for the AR and MA hyperparameters.

To determine the optimal hyperparameters for an ARIMA model, each candidate model needs to be compared using a relative measure. Two commonly used measures of goodness of fit are the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC), which help identify the best model from a pool of potential models. However, BIC is generally regarded as a more conservative measure because it penalizes models with more parameters than AIC does. Therefore, BIC is often preferred when the sample size is large, and the goal is to select a model that fits the data well while using the fewest parameters possible. In contrast, AIC is useful when the sample size is small, and the aim is to balance the model's goodness of fit with its complexity. These measures can be defined as:

$$AIC = 2k - 2\ln(L) \tag{2.4}$$

$$BIC = k\ln(n) - 2\ln(L) \tag{2.5}$$

where k is the number of parameters in the model, L is the likelihood of the data given the model, and n is the number of observations. The likelihood of the data given the model, L, refers to the probability of observing the data we have under the assumption that the model is true. More formally, if we have a sample of observations $y1, y2, ..., yn$, and the model is specified by a set of parameters $\theta1, \theta2, ..., \theta k$, then the likelihood of the data given the model is given by:

$$L = P(y1, y2, ..., yn \mid \theta1, \theta2, ..., \theta k) \tag{2.6}$$

The AIC or BIC value is calculated for each model, and the model with the lowest value is considered the best fit for the data. The lower the value, the better the model is considered to be.

Augmented Dickey-Fuller (ADF) Test

The Augmented Dickey-Fuller (ADF) test is a widely-used statistical test for examining whether a given time series data set is stationary or not. It is one of the most common statistical tests used to investigate the stationarity of a series. Being a significance test (i.e., a unit root test), it involves hypothesis testing based on null and alternative hypotheses. A test statistic is then computed and p-values are reported as a result. In probability theory and statistics, a unit root is a characteristic of some stochastic processes that can cause difficulties in statistical inference

involving time series models. Specifically, a unit root is non-stationary but not necessarily trended. The ADF test is carried out under certain assumptions, including:

> **Null Hypothesis (H_0):** Series is non-stationary or series has a unit root.
>
> **Alternate Hypothesis(H_1)**: If the null hypothesis is not rejected, then the test has not provided enough evidence to prove that the series is non-stationary.
>
> **Conditions to Reject Null Hypothesis (H_0)**: If Test statistic $<$ Critical Value and p-value < 0.05 then reject Null Hypothesis (H_0) as the time series does not have a unit root and hence, is stationary and does not have a time-dependent structure.

Should the time series data be stationary, the value of I can be set to 0. If not, the data should be differenced and the ADF test should be run on the differenced data. If the differenced data then passes the aforementioned conditions, the value of I can be set to 1. Alternatively, the processes of differencing and performing the ADF tests need to be re-initiated.

Complete Auto-Correlation Function (ACF) and Partial Auto-Correlation Function (PACF)

In order to perform time series analysis and forecasting using Autoregressive Moving Average (ARMA) models, it is necessary to select candidate models and determine the order of Autoregressive (AR) and/or Moving Average (MA) terms. This can be done by examining the Autocorrelation function (ACF) and Partial autocorrelation function (PACF) plots of the series. While these plots do not directly determine the order of the ARMA model, they can provide insight into the order and help identify models that may be a good fit for the time-series data.

The Autocorrelation Function (ACF) plot is a graphical representation of the correlation coefficients between a time series and its lagged values, expressed as a bar chart. The ACF plot provides information on how the present value of a time series is correlated with its past values at different lags (1-unit past, 2-unit past, ..., n-unit past). The y-axis of the ACF plot represents the correlation coefficient, while the x-axis indicates the number of lags. For instance, if $y(t-1), y(t), y(t-1), \ldots, y(t-n)$ represent the values of y at a given times $t, t-1, \ldots, t-n$, then the the correlation coefficient between $y(t)$ and $y(t-1)$ is the lag-1 value, lag-2 is the correlation coefficient between $y(t)$ and $y(t-2)$, and so forth.

The Partial Autocorrelation Function (PACF) is a measure of the partial correlation between a time series and its lags, which captures the direct relationship between a time series and its past values, while removing the indirect effects of other intervening variables. It can be interpreted as the correlation between a specific lag and the time series, after removing the effects of the other lags. In simple terms, PACF can be thought of as a linear regression where we predict the current value of the time series using only the values of a specific set of lags, while controlling for the effects of all other lags. By doing so, PACF helps to identify the direct causal relationship between a time series and its past values.

The order of the AR(p) model can be determined by examining significant spikes on the PACF plot. On the other hand, the order of the MA(q) model can be determined by identifying a sharp cut-off after lag q on the ACF plot.

2.3 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) that is specifically designed to overcome the vanishing gradient problem and capture long-term dependencies in time series data [Hochreiter and Schmidhuber 1997]. It achieves this by introducing a memory cell and several gates that regulate the flow of information into and out of the cell [Bengio et al. 1993]. The gates are responsible for controlling the forget, input, and output information, which allows the LSTM to selectively remember or forget information over time. This makes the LSTM well-suited for modelling sequential data that exhibits long-term dependencies, such as speech, text, and time series data. Looking into the architecture of the LTSM in Figure 2.1, it can be seen that there are four layers inside that interact together [Karkare 2018].

The LSTM model allows for the horizontal flow of information to the cell state c_t from the previous cell state c_{t-1}, without modification, thereby enabling the model to retain context from multiple time steps in the past. This makes it capable of learning long-term dependencies in the input sequence. Between cell statec_{t-1} and c_t, there is the possibility to include or remove information with the use of the sigmoid layers σ, or gates. The output of these gates ranges between 0 and 1, which is a measurement of the weight of how much information each component should be let through to control the cell state. Each of the three gates impacts the data differently, as they each have their purpose.

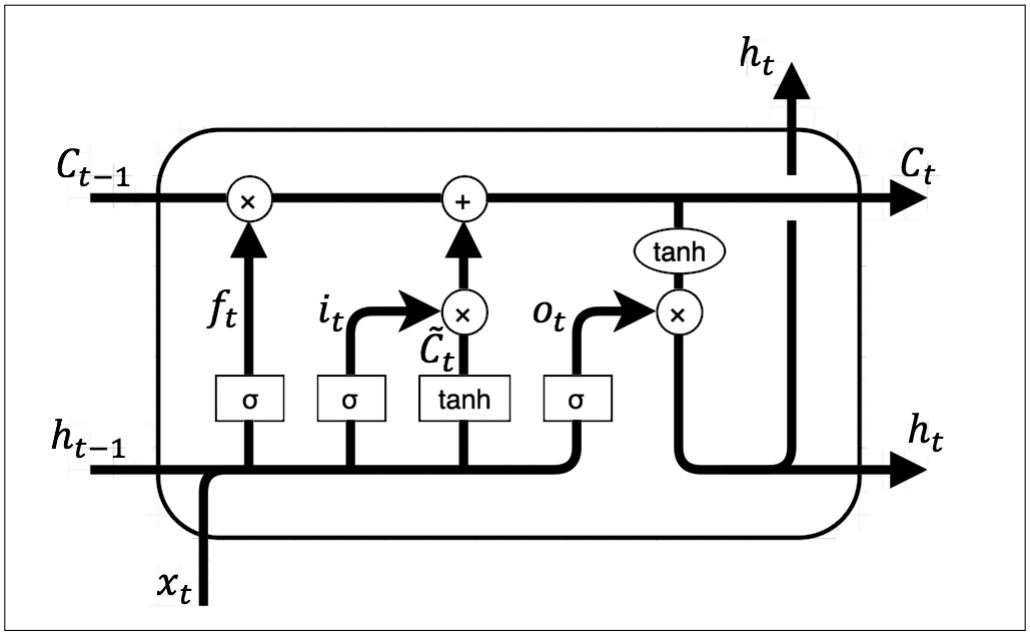


Figure 2.1: Architecture of LSTM

Let us define the LSTM equations using the following assumptions: x_t is the input at the current timestamp, h_{t-1} is the hidden state of the previous timestamp, and W is the weighted matrix associated with the input and hidden state. Then, we can express the LSTM equations as follows:

Forget Gate (f_t) determines which information to discard from the cell state. It takes in the previous hidden state h_{t-1} and the current input x_t, passes them through a sigmoid function, and outputs a number between 0 and 1 for each element in the cell state.

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{2.7}$$

Input Gate (i_t) determines which new information to be stored in the cell state. It takes in the previous hidden state h_{t-1} and the current input x_t, passes them through a sigmoid function, and outputs a number between 0 and 1 for each element in the cell state.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{2.8}$$

Candidate Cell ($\tilde{c}_t$) creates a vector of new candidate values to be added to the cell state. It takes in the previous hidden state h_{t-1} and the current input x_t, passes them through a *tanh* function, and outputs a vector of new values.

$$\tilde{c}_t = tanh(W_c * [h_{t-1}, x_t] + b_c) \tag{2.9}$$

Cell State (c_t) the long-term memory of the LSTM. It is updated by the forget gate, input gate, and candidate cell to selectively retain or discard information from previous time steps.

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \tag{2.10}$$

Hidden State (h_t) the output of the LSTM, which is based on the current cell state and the current input. It is calculated by passing the updated cell state through a *tanh* function and multiplying it by the output gate.

$$h_t = o_t * tanh(c^t) \tag{2.11}$$

Output Gate (o_t) determines which part of the hidden state to output. It takes in the previous hidden state h_{t-1} and the current input x_t, passes them through a sigmoid function, and outputs a number between 0 and 1 for each element in the hidden state. The updated hidden state is then multiplied by these values to produce the final output.

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \tag{2.12}$$

2.3.1 LSTM Model Hyperparameters

Selecting appropriate hyperparameters is crucial in ensuring the effectiveness of the learning algorithm and the resulting performance of the LSTM model. These hyperparameters are responsible for controlling the learning process, and choosing the correct values can significantly impact the quality of the model's predictions. However, hyperparameters are often volatile, and varying their values can lead to different outcomes. Hence, it is vital to compare models and adjust the hyperparameters through hyperparameter tuning to achieve optimal results. This process is called hyperparameter tuning.

The relevant hyperparameters to tune for the LSTM model are as follows:

Nodes A node is a computational unit that has one or more weighted input connections, a function that links the inputs by means of data transfer, and an output connection. Nodes are organized into layers to comprise a network, and can also be referred to as the visible layer. By convention, the number of nodes used is in exponents of 2 (2, 4, 8, 16, 32, 64, 128, 256, 512, 1024...) as this corresponds to using the efficient use of

the underlying hardware. However, there is no proof of an optimal method to select these hyperparameters

Batch Size A batch size refers to the number of samples processed before the model's parameters are updated during training. Larger batch sizes can speed up training, but smaller batch sizes can lead to more accurate models. However, it is important that the batch size contains a subset of data that is able to statistically represent all data samples to ensure that the quality of the model does not degrade.

Epochs The number of times the learning algorithm will iterate through the entire training dataset. One epoch refers to a sample of training data updating the internal model parameter once. Increasing the number of epochs can lead to more accurate models, but can also lead to overfitting.

Optimizer Responsible for controlling the performance of the algorithm by adjusting the model weights and learning rates in order to minimize the loss function.

Dropout Rate The percentage of nodes in the model randomly "drop out" during training to prevent overfitting. A higher dropout rate can increase regularization and prevent overfitting, but can also lead to underfitting.

Patience The number of epochs to wait before stopping training if the model's performance on the validation set has not improved. This is used in early stopping to prevent overfitting.

Learning Rate The magnitude of the update to the model's parameters during training. A smaller learning rate can lead to slower but more stable training, while a larger learning rate can lead to faster but less stable training.

Activation Function The activation function decides whether a neuron within an artificial network is important or not in the process of prediction using simpler mathematical operations, and hence activates or deactivates the node.

2.3.2 Hyperparameter Optimization

Hyperparameter tuning or optimization is the process of selecting the optimal hyperparameters for a model to achieve a desired metric value. Proper selection of hyperparameters can result in a successful model, while poor selection can lead to an inefficient and time-consuming training process. There are two main methods of setting hyperparameters for models, which are explained below.

Manual Hyperparameter Tuning

Manual hyperparameter tuning involves an iterative process of selecting hyperparameters manually, setting them, building machine learning models, and evaluating their performance using the underlying dataset. This method allows for the testing of different combinations of hyperparameters to determine the best combination that yields the desired performance metric.
This approach has several advantages, including (1) providing greater control over the process and (2) facilitating a deeper understanding of the tuning process and how it affects network weights. However, it also has several disadvantages, such as (1) the tedious and time-consuming nature of the process, as it often requires many trials, (2) the cost and effort involved in keeping track of the various hyperparameters and their corresponding results, and (3) the potentially large number of hyperparameters to consider, depending on the complexity of the model [ES and Bajaj 2022].

Automated Hyperparameter Tuning

Automated hyperparameter tuning refers to using algorithms to automatically search for the optimal set of hyperparameters for a given machine learning model. This approach involves specifying a range of hyperparameters and their possible values. The algorithm then runs multiple trials on the training dataset, testing different combinations of hyperparameters until it finds the best set of hyperparameters that results in optimal performance on the validation set.

The main advantage of automated hyperparameter tuning is that it saves time and resources, as the process is automated and can quickly search through a large parameter space. However, a potential disadvantage is that it may not always find the best set of hyperparameters, and the user may need to have a good understanding of the algorithm used and its limitations. We will discuss a few of these algorithms and tools that can be used to achieve automated tuning.

A selection of hyperparameter tuning methods are:

Grid Search A grid search is a commonly used method for hyperparameter optimization. It involves systematically trying out all possible combinations of a pre-defined subset of hyperparameters and applying them to a learning algorithm, then evaluating the resulting models through cross-validation (explained in Section 2.5). This can be a computationally intensive process, but it ensures that all possible hyperparameter values are considered and may lead to finding the best combination for optimal performance. The model boasting the best accuracy is naturally considered to be the best, and these hyperparameters are then selected as the optimal ones [Bergstra and Bengio 2012].

Random Search Random search is a hyperparameter tuning method that selects hyperparameters randomly and uses them to train a model. The best combination of hyperparameters is then selected. While it shares some similarities with grid search, the main difference is that it does not require setting a specific set of values for each hyperparameter. Instead, values are randomly sampled from a statistical distribution, such as a uniform distribution, for each hyperparameter. [Bergstra and Bengio 2012].

Bayesian Optimization BBayesian optimization is an iterative method that selects hyperparameters by building a probabilistic model of the objective function based on previous iterations. It then uses this model to intelligently choose the next set of hyperparameters to evaluate. This method is particularly useful when evaluating the objective function is computationally expensive or time-consuming. By using information from previous evaluations, Bayesian optimization can efficiently explore the hyperparameter space and converge on the optimal set of hyperparameters more quickly than other methods [Snoek et al. 2012].

Other tuning methods and algorithms exist; however, these have not been covered in this document as they fall outside of the scope of this analysis.

2.4 Gated Recurrent Units (GRU)

Introduced by Cho et al. [2014], the Gated Recurrent Units (GRU) is a type of neural network that is commonly used for processing sequential data. It was designed to address the vanishing gradient problem that arises in standard Recurrent Neural Network (RNN) models, where gradients can become very small, making it difficult to learn long-term dependencies. GRU accomplishes this by using gating mechanisms to selectively update and reset its hidden state,

allowing it to selectively remember or forget information as needed. The design of GRU is similar to that of LSTM, with the key difference being the number of gates and the way they are used to control the flow of information [Chung et al. 2014]. Unlike LSTM, however, the GRU only has two gates: reset gate, and update gate; as can be seen in the GRU architecture in Figure 2.2. The simplification of the cell allows it to store long-term states, but with reduced computational complexity in comparison to the LSTM [Li et al. 2021].

To understand the functioning of the GRU, refer to the GRU cell illustrated in Figure 2.2. In each time step t, the GRU cell receives an input x_t and the previous hidden state h_{t-1}. The output is a new hidden state h_t which is then forwarded to the subsequent timestamp. The initial value of h_0 is zero when $t = 0$.

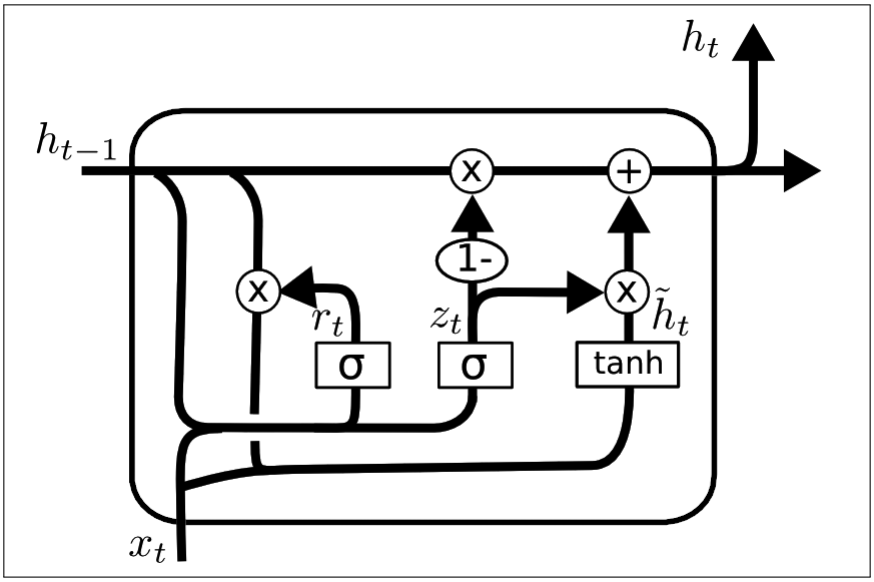


Figure 2.2: Architecture of a GRU Cell

Reset Gate (r_t) is a gating mechanism that controls how much of the previous hidden state h_{t-1} is used in computing the candidate hidden state $\tilde{h}_t$ at the current time step t.

$$r_t = \sigma(W^r * x_t + U^r * h_{t-1}) \tag{2.13}$$

where U^r and W^r are weight matrices for the reset gate. Given that the Sigmoid function is applied here, the value of r_t will range from 0 to 1.

Update Gate (z_t) is a gating mechanism that controls how much of the previous hidden state h_{t-1} is retained and how much of the candidate hidden state $\tilde{h}_t$ is incorporated into the new hidden state h_t at the current time step t.

$$z_t = \sigma(W^z * x_t + U^z * h_{t-1}) \tag{2.14}$$

where U^z and W^z are weight matrices for the reset gate. Once again, the Sigmoid function is applied here, resulting in z_t ranging from 0 to 1.

Candidate Hidden State $(\tilde{h}_t)$ is an intermediate hidden state value computed at the current time step t, based on the input x_t and the previous hidden state h_{t-1}, and potentially modified by the reset gate.

$$\tilde{h}_t = tanh(Wx_t + r_t \odot Uh_{t-1}) \tag{2.15}$$

Hidden State (h_t) is the final hidden state value output by the GRU at the current time step t, obtained by combining the previous hidden state h_{t-1} and the candidate hidden state $\tilde{h}_t$ using the update gate.

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{2.16}$$

2.4.1 GRU Model Hyperparameters and Hyperparameter Optimization

As GRU is a variation of the LSTM, it too has hyperparameters that need to be chosen and optimized. Furthermore, the hyperparameters for the GRU are the same as those described in Section 2.3.1. Similarly, the method of optimizing the hyperparameters is as per those of the LSTM described in Section 2.3.2.

2.5 Model Performance Measures

In addition to interpretability, predictive evaluation is a crucial aspect of machine learning, as it allows us to assess the performance of a model. In this analysis, we focus on regression models and, therefore, only consider regression metrics. Regression involves predicting the state of an outcome variable at a particular time point using other related independent variables, resulting in continuous values within a specific range.

Assuming that i indicates the i^{th} observational data point, n is the total number of data points, yi is the actual observations of the time series, and $\hat{y}i$ is the predicted value of the time series, the following metrics can be used to evaluate the results of the prediction:

Mean Square Error (MSE) It is calculated as the average of the squared differences between the predicted and actual values of the time series. By squaring the differences, it places more emphasis on larger errors, penalizing them more heavily than smaller errors. This property makes it a preferred metric over others as it can be easily optimized due to its differentiability.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.17}$$

Root Mean Squared Error (RMSE) is obtained by taking the square root of the mean squared error. The advantage of using RMSE is that it gives us the error value in the same unit as the target variable, which makes it easier to interpret the results. Additionally, RMSE is more sensitive to large errors than to small errors because the errors are first squared before averaging. Therefore, RMSE is useful in cases where large errors are considered to be more undesirable than small errors.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{2.18}$$

Mean Absolute Error (MAE) is the average of the absolute differences between the predicted and actual values of the time series. Compared to MSE, it is more robust to outliers and does not penalize errors as heavily. MAE is a linear score, meaning

all individual differences are equally weighted. However, it may not be suitable for applications where outliers need to be emphasized.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.19}$$

R^2 or the Coefficient of Determination enables us to assess the performance of our model relative to a constant baseline, and quantify how much our model has improved. The baseline is determined by calculating the mean of the data and plotting a line at that mean. R^2 is a scale-free metric, meaning it is not affected by the magnitude of the values, ensuring that R^2 will always be less than or equal to 1.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \tag{2.20}$$

Adjusted R^2 is an improvement over R^2 because R^2 suffers from the problem of increasing scores even when the model is not improving, which can mislead researchers. Adjusted R^2 is always lower than R^2 because it takes into account the increasing number of predictors and only shows improvement if there is a genuine improvement in the model's performance.

$$R_a^2 = 1 - \left[\left(\frac{(n-1)}{n-k-1}\right) * (1 - R^2)\right] \tag{2.21}$$

Cross Validation (CV) Cross-validation is a statistical technique used to estimate the accuracy or performance of machine learning models. It is commonly employed to guard against overfitting, especially when the available data is limited. The cross-validation process involves dividing the data into a fixed number of folds or partitions, performing the analysis on each fold, and then computing the average error estimate across all the folds. The general algorithm for this analysis is provided in Algorithm 1.

Algorithm 1 K-Fold Cross Validation Methodology

Require: Dataset, k

1: Divide the data set into k number of folds (subsets) randomly.
2: For each fold, train your model on the remaining k – 1 folds of the dataset. Finally, test the model to determine the effectiveness of the k^{th} fold
3: Repeat this for all k-folds within the data set
4: Calculate the performance metric of the model by averaging the accuracy of the $k - th$ record (called the cross validation accuracy)

This method is considered to be less biased compared to other methods as it ensures that every observation from the original dataset has the chance of appearing in the training and test set. This makes it one of the best approaches when dealing with limited input data. A drawback of this approach is that the training algorithm must be executed k times from the beginning, resulting in k times more computational resources needed for evaluation.

Algorithm 1 is not the appropriate method for evaluating time series models because shuffling the data disrupts the temporal order of events and may lead to training on

future data and testing on past data, introducing data leakages. To preserve the temporal dependency between observations during testing, cross validation on a rolling basis is a more suitable method for time series data. This involves training the model on a small subset of data, forecasting for later data points, and then assessing the accuracy of the forecasted data. The same forecasted data points are then incorporated into the next training dataset, and the process is repeated for subsequent data points. This can be seen in Figure 2.3.

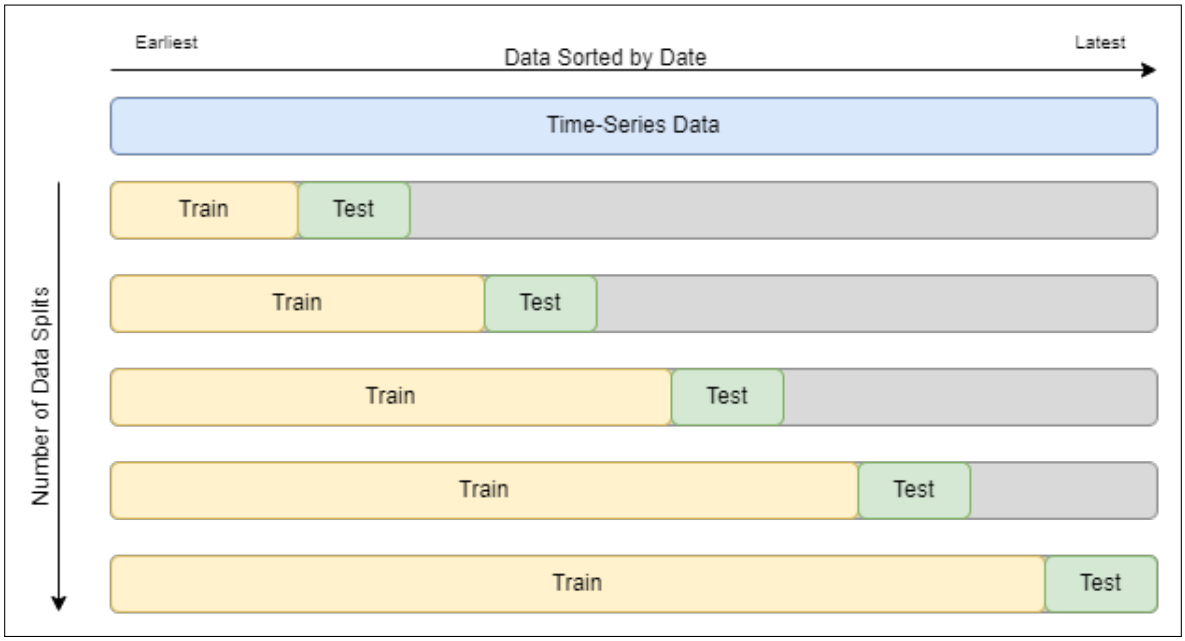


Figure 2.3: Cross Validation on Time Series

2.6 Feature Selection

Feature selection involves selecting the most consistent, relevant, and non-redundant features from a given dataset to use in building a model. As datasets continue to grow in size and complexity, it becomes increasingly important to methodically reduce their size [Guyon and Elisseeff 2003]. The primary objectives of feature selection are to enhance model performance and reduce the computational cost of modelling. Feature selection offers numerous benefits to machine learning models, including simplifying the model and improving its interpretability, decreasing training times, increasing accuracy, and reducing the risk of over-fitting.

2.6.1 Feature Selection Methods

Feature selection algorithms can be grouped into two main categories: supervised and unsupervised. Supervised methods are used for labelled data, and unsupervised methods are used for unlabeled data. Unsupervised techniques can be further classified filter methods, wrapper methods, embedded methods, and hybrid methods. We will discuss the first three below, as hybrid methods refer to a combination of these methods.

Filter-Based Methods are a type of feature selection algorithm that select features based on statistical measures, rather than relying on cross validation performance. These methods typically apply a chosen metric to assess the relevance of each feature and perform recursive feature elimination to remove irrelevant attributes. Filter-based methods can be categorized as either univariate or multivariate. Univariate methods

create an ordered ranking list of features based on their individual relevance, while multivariate methods evaluate the relevance of the features as a whole, identifying and removing redundant and irrelevant features.

Wrapper-Based Methods approach feature selection as a search problem, wherein a set of features is selected and their quality is assessed by comparing various combinations of features through preparation, evaluation, and comparison. This approach enables the identification of potential interactions among variables. These methods concentrate on identifying feature subsets that can enhance the performance of the clustering algorithm utilized for the selection process. Popular examples of such techniques are the Forward feature selection and the Boruta feature selection.

Embedded Methods involves integrating the feature selection algorithm as a part of the learning algorithm, enabling the simultaneous performance of classification and feature selection. During the training process, the classifier will extract the appropriate features that contribute the most to each iteration. Widely-used embedded methods include decision tree feature selection and random forest feature selection.

Pearson Correlation

The Pearson Correlation is a *filter-based* method. It is frequently used for numerical variables and calculates a value, denoted by r, between -1 and 1. The correlation between variables is evaluated based on this value, where 0 implies no correlation, 1 indicates a completely positive correlation, and 1 denotes a completely negative correlation [Boslaugh and Watters 2008].

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x}_i)(y_i - \bar{y}_i)}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x}_i)^2(y_i - \bar{y}_i)^2}} \tag{2.22}$$

where r is the correlation coefficient, x_i are the sample values of the x-variable, $\bar{x}_i$ is the mean of the the x-variables, y_i are the sample values of the y-variable, and $\bar{y}_i$ is the mean of the the y-variables.
In cases where the data set consists entirely of either positively or negatively correlated attributes, there is a significant likelihood of a problem called multicollinearity affecting the model's performance. This occurs when one predictor variable within a multiple regression model can be accurately predicted in a linear manner from the others, leading to results that are skewed or misleading. The simplest solution to this issue is to remove or eliminate one of the perfectly correlated features or to apply a dimension reduction algorithm such as Principal Component Analysis (PCA) [Jolliffe 2011].

Chi Squared

Chi-Squared is a *filter-based* method, that is utilized in statistics to assess the independence of two events. This method involves computing the chi-squared metric between the target and the numerical variable, and selecting only the variable with the highest chi-squared value. Given the data for two variables, we can derive the observed count O and expected count E, and then evaluate the degree of deviation (known as the degree of freedom) between the expected and observed counts. To use χ^2 for feature selection, we calculate χ^2 between each feature and the target, and choose the desired number of features with the best χ^2 scores. The underlying assumption is that if a feature is independent of the target, it does not provide relevant information for classifying observations.

$$\chi_c^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{2.23}$$

where c is the degree of freedom, O are the observed values and E are the expected values.

Recursive Feature Elimination (RFE)

Recursive Feature Elimination (RFE) is a *wrapper-based* method used in the selection of features. This method starts by fitting a model to the underlying data set, and then progressively deletes the weakest features until a predetermined number of features is achieved. Features are ranked by the model's coefficient attributes (which determine the importance of the feature); and through the removal of the number of features iteratively, the RFE is able to eliminate dependent variables and collinearity within the model. RFE requires the specification of the number of features to retain, but determining the optimal number of features beforehand is often not feasible. However, by using cross validation, the different feature subsets can be pre-evaluated and selected based on their scores.

Lasso Regression

Lasso Regression is a technique based on an *embedded* approach. This linear model utilizes a cost function to evaluate the linear relationship between two features. When two features exhibit a linear correlation, their combined presence will amplify the cost function's value. As a result, Lasso Regression endeavours to minimize the coefficient of the less significant feature to zero, in order to identify the most relevant features [Hastie et al. 2015].

$$MSE(y, y_{pred}) = \frac{1}{2N_{training}} \sum_{i=1}^{N_{training}} (y^i_{real} - y^i_{pred})^2 \tag{2.24}$$

$$l_1 = \sum_{j=1}^{n} |a_j| \tag{2.25}$$

$$LassoMSE(y, y_{pred}) = MSE(y, y_{pred}) + \alpha l_1 \tag{2.26}$$

where a_j is the coefficient of the j^{th} feature, l_1 is the penalty term, and α controls the intensity of the penalty term. As the coefficient of a feature increases, so does the corresponding value of the cost function. Therefore, a higher α coefficient leads to stronger penalization, resulting in a sparser feature set.

Tree-Based Models

Embedded methods utilize algorithms that have feature selection mechanisms built into them, such as tree-based models. Random Forest is an example of a tree-based model that employs the bagging method and decision trees for feature selection. In Random Forest, the training dataset is repeatedly resampled using the "bootstrap" procedure, with each sample containing a random subset of the original features to fit a decision tree. The number of models and features are hyperparameters that require optimization. The predictions of the trees are then combined using the mean value (for regression) or soft voting (for classification). Bagging aims to decrease the standard error and variance of the model by averaging the outputs of individual decision trees, thus balancing the bias-variance trade-off.

2.6.2 Selecting the Best Feature Selection Model

Choosing the best feature selection method depends on the input and output in consideration [Heavy.AI 2022].

Algorithm 2 Random Forest Feature Importance Methodology

Require: Numerical data matrix representing the features and predictor values

1: Implement Shadow Features by creating randomness to the features by creating duplicate features and shuffling the values in each column.
2: Train a classifier (Random Forest) on the data set and calculate the importance by assigning Z-scores (z-scores explain where the resulting score lies on a normal distribution curve)
3: Compare the importance of the original features and the shadow features. Then the algorithm checks for each of your real features if they have higher importance.
4: If the Z-score of the original feature has performed better than the shadow feature, mark the feature as important.

Numerical Input, Numerical Output To best select features from data with numerical input variables and numerical output, a widely-used method is to determine the correlation coefficient between variables by using the Pearson's correlation coefficient for linear regression or Spearman's rank coefficient for nonlinear regression.

Numerical Input, Categorical Output To best select features from data with numerical input variables and categorical output, the categorial target needs to be considered when calculating the correlation coefficient, such as ANOVA correlation coefficient for linear or Kendall's rank coefficient for nonlinear.

Categorical Input, Numerical Output In the rare case of a regression predictive modelling problem where the data used consists of categorical input variables and numerical output, a correlation coefficient such as ANOVA correlation coefficient for linear or Kendall's rank coefficient for nonlinear can once again be used, but in reverse order.

Categorical Input, Categorical Output To best select features for data with categorical input variables and categorical output, one can use a correlation coefficient such as the Chi-Squared test for known to be a successful method when dealing with agnostic data types.

2.7 Comparative Studies of the Models used to Forecast Time Series Data

Whilst the prediction of the future is impossible, we can imply a series of events given the combination of historic data and knowledge of potential future situations that may impact the prediction. However, the correctness of a prediction can only be established when compared to the actual occurrence, and it may be required to assess various prediction methodologies to determine the best algorithm given the data set at hand. By assessing previous works that compare the prediction algorithms, we can establish (1) that the aforementioned algorithms are all adequate at producing a prediction value; (2) that the algorithms can successfully analyse various kinds of time-series data, and; (3) that by using these algorithms on the time-series data presented in this thesis proposal, we can be confident that we can achieve reasonable forecasting results. Hence, the success determined in previous related works indicates the potential success of the research proposal.

2.7.1 Stock Market Price Prediction

Investors have yielded significant profits by successfully predicting the future closing stock values in the financial stock exchange. However, the ability to forecast these stock prices is challenging

to calculate due to the number of potential variables to consider as well as unpredictable noise that may contribute to the resultant prices [Shah et al. 2018]. To investigate whether ARIMA or LSTM offer a better prediction methodology to lower forecast errors and higher accuracy of forecasts; Siami-Namini et al. [2018] applied the models to historical monthly financial time series data from January 1985 to August 2018 from the Yahoo finance website. Since the data collected had non-stationary properties, the ARIMA model was selected to model and forecast these market values. LSTM, on the other hand, was chosen as a representative of deep learning-based algorithms due to its capability to train and preserve features of a given data set over a longer period of time. There is no specific empirical evidence to determine whether the more traditional economic forecasting method, such as the ARIMA, forecasts better than the deep learning algorithms, such as LSTM. However, in determining which model has a better performance in reducing error rates, Siami-Namini et al. [2018] were able to compare ARIMA and LSTM. The data preparation involved selecting stock from the market and using the "Adjusted Close" variable as the only feature of the financial time series to be fed into the ARIMA and LSTM Models. The data was split into two subsets: 70% of the dataset was used for the training set and the remaining 30% for the test set, to test the accuracy of the models. Using the RSME assessment metric, the results of the analysis showed that the LSTM-based algorithm had an 84.4% lower RMSE in comparison to the ARIMA model, stating that the deep learning-based approaches have the advantage of iterative optimization which improves the model fit of the data. It is to be noted that this analysis was only performed on single-variable data, and further investigation could be considered for other data sets with a various number of features.

Joosery and Deepa [2019] further performed a comparative analysis of the ARIMA and LSTM forecasting algorithms in the prediction of stock prices. The data used in the analysis consisted of five historical stock price data ranging from April 2009 to February 2019. There are similarities in this analysis in comparison to that of Siami-Namini et al. [2018], in that the same algorithms have been used to predict stock prices using historical time-series values. However, the difference in the analyses include (1) the efficiency of the two models was compared using the MSE instead of the RMSE; (2) The value predicted was the average of the stock value over the day; (3) the efficiency of the algorithms was tested on the data observations over different periods of time. The results of the analysis by Joosery and Deepa [2019] showed that the ARIMA model performed better on larger sets of data (over one year), whereas the LSTM model performed better on smaller data sets. The justification for this result is that the LSTM is affected by the amount of training data provided whereas ARIMA is not. When an additional mechanism was added to the LSTM to improve the memory of the LSTM, it was found that it performed similarly to that of ARIMA with large data sets and much better than that of the LSTM in smaller data sets. As LSTM can find patterns in various input data, further investigation could include external factors to improve its performance.

An earlier research article by Adebiyi et al. [2014b] examined the performance of ARIMA and an Artificial Neural Network (ANN) when predicting published stock data from the New York Stock Exchange. Whilst the ANN is not LSTM or GRU, it is still important to understand how statistical models compare to artificial neural network models. In the analysis, the closing price of the selected stock was chosen to be forecast over a 30-day period. Using the "Close Price" as a prediction variable, the selected stock chosen was the Dell Inc. stock data from August 1988 to February 2011 and consisted of a total of 5680 observations. Using the MSE as a performance indicator, Adebiyi et al. [2014b] found that there was no significant difference in the training forecasting accuracy of the ARIMA model in comparison to the ANN. However, the performance of the ANN model was shown to be better than the ARIMA model in terms

of the forecast accuracy of the test data. This is an indication that any improvements on the ANN (such as the inclusion of LSTM or GRU) could potentially improve the performance.

A comparative study of LSTM and Deep Neural Network (DNN) for Stock Market Forecasting was also conducted by Shah et al. [2018]. This study used a web data set containing daily close prices of a particular stock from 1997 to 2017, to forecast daily and weekly movements. The split of the data was 80:10:10 (train:valid:test), which allowed the training and validation errors to coincide and stabilise quicker. In the daily predictions, it was found that the DNN produced a lower RMSE than the LSTM, whilst the LSTM resulted in a lower forecast bias. However, in the weekly predictions, the LSTM does a better job of predicting these movements, in comparison to the DNN.

An analysis of a less traditional stock price, Bitcoin, was done by Yamak et al. [2019] where the efficiency of performance of the ARIMA model, LSTM and GRU networks were compared. The Bitcoin data set, with a one-day interval and exchange rate in American Dollars, was extracted from November 2014 to June 2019. The 1639 data points were split into a training and test set at a 70:30 ratio. Once the models were trained, they were tested for efficiency using the RMSE and the Mean Absolute Percentage Error (MAPE). With both indicators, the results showed that the ARIMA model outperformed both LSTM and GRU. The result could be due to the parameters chosen per model, as well as the data size. Furthermore, the selected features may not have been sufficient to predict the Bitcoin prices accurately, as they may not have implicitly contained information about other external features

2.7.2 COVID-19 Case Prediction

Since 2019, the COVID-19 epidemic has spread from Wuhan, China to 213 countries worldwide. The spread of the virus has taken a devastating toll on the livelihoods and health of many individuals and economies. To try to mitigate its effects, detailed studies have been performed to develop short-term prediction models to predict the number of future cases. Shahid et al. [2020] proposed using LSTM and GRU to predict COVID-19 cases. For the cases in 10 different countries, a daily record of the number of confirmed cases, death cases and recovered cases were extracted from The Harvard COVID-19 database with 268 samples dated from January 2020 to June 2020. The training data set compromised of just under 60% of the sample data, and 40% was allocated to the test data. For each country, the models were expected to predict the number of given cases for the next 48 days. In the analysis, three performance measurements are used to evaluate the performance of the proposed models. These indices are the MAE, RMSE and the R^2 score. Once the models had been built and trained, it was found that ARIMA was unable to follow the trend of the features within the data, and this resulted in a higher prediction error. LSTM performed better than both GRU and ARIMA, with the lowest MAE and RMSE values for deaths as well as the highest R^2 values scored in recovered cases. These results were only provided for one country, namely China.

In a further study, ArunKumar et al. [2021] also compared the differences between deep layered neural networks, such as LSTM and GRU, in the forecasting of COVID-19 cases. This analysis was conducted using publicly available data from John Hopkins University's COVID-19 database. Unlike the previous analysis, ArunKumar et al. [2021] predict a 60-day forecast of the COVID-19 trend for the top-10 countries that are highly impacted by COVID-19. The values selected to be forecasted include cumulative confirmed cases, cumulative recovered cases and cumulative fatalities in the countries. Whilst it is not stated how many sample points were used in training and testing the models, it is known that MSE and RMSE indices were used

as performance measurements, with the lowest of the values believed to be the best model for forecasting. The results showed that in half the countries, LSTM performed better than GRU in predicting the total number of confirmed cases and recovered cases. For the forecast of the fatalities data, LSTM outperformed GRU in six out of the 10 countries. This indicates that the outputs of the models are dependent on the data given, as well as the required forecast.

2.7.3 Traffic Volume Prediction

To predict online traffic experienced on satellite networks to prevent network congestion and improve the utilisation of network resources, Li et al. [2021] proposed using a GRU neural network and comparing the results to a specialised ARIMA model can consider fractional differencing (FARIMA). This analysis had a large amount of data, namely 400,000 data points, and it was split into an 80:20 training and test ratio. Using the performance indicators, MAE, MSE and RMSE, it was found that the GRU outperformed the ARIMA model and had superior prediction accuracy.

The prediction of resource utilisation of user traffic in cellular networks was investigated by Azari et al. [2019]. This evaluation of LSTM and ARIMA was undertaken to see which model resulted in superior performance. Using a multi-variable data set consisting of cellular traffic on the 5G network to train the models, the performance was measured by RMSE under different traffic circumstances. The analysis showed that the LSTM model, in particular when augmented with additional features, demonstrated an improved prediction over the ARIMA model for future traffic predictions.

A further study, to forecast real-time network traffic in India, was done by Shelatkar, Tejas et al. [2020], where ARIMA and LSTM were used to predict the time-series data 30 days ahead of time. Using a multi-variable data structure, that included the number of user visits to a page, seasonality visit patterns, holidays, and other long-term trends, Shelatkar, Tejas et al. [2020] was able to determine that the LSTM was more effective than the ARIMA model.

Looking at a more traditional form of traffic congestion, Wang et al. [2020] investigate the prediction of truck traffic flow based on the LSTM and GRU. Sampled truck GPS data was collected between November 2018 and December 2018, which resulted in 288 data points and was used to predict data at one-hour time intervals. Each piece of GPS data consisted of the date and time that the data was recorded, as well as the geolocation coordinates, and truck identification and travel data. For the roadway network, both the LSTM and GRU were trained on optimised parameters to predict traffic flow and their performance was compared using MAPE as a performance indicator [Wang et al. 2020]. The findings show that both models have excellent performance in predicting truck traffic flows. However, for the average prediction accuracy throughout both peak and off-peak periods, LSTM had an improved accuracy of 4.1% in comparison to GRU.

2.7.4 Summary of Comparative Studies

In these studies, LSTM consistently demonstrated strong performance in predicting time-series data. ARIMA and specialized ARIMA models also showed competitive results. GRU showcased competitive performance in some scenarios. Choice of algorithm often depended on data and prediction target characteristics.
Based on the collective findings derived from the summary of the conducted comparative studies in the related work, it can be discerned that, excluding the results of our own experiments,

Study	Aim	Data	Algorithms and Findings
Stock Market Price Prediction	Predict stock market closing prices using ARIMA and LSTM.	Historical monthly financial time series data (Jan 1985 - Aug 2018).	LSTM had a lower RMSE in comparison to ARIMA, indicating better pattern capture.
Bitcoin Price Prediction	Predict Bitcoin prices using ARIMA, LSTM, and GRU.	Bitcoin exchange rate data (Nov 2014 - Jun 2019).	ARIMA outperformed LSTM and GRU in terms of RMSE, attributed to parameter tuning.
Traffic Volume Prediction	Predict traffic volume using GRU compared to specialized ARIMA models.	Large dataset with 400,000 data points.	GRU demonstrated superior prediction accuracy to ARIMA in MAE, MSE, RMSE.
Network Traffic Prediction	Forecast network traffic using LSTM and ARIMA.	Cellular network traffic data from a 5G network.	LSTM, with additional features, improved prediction accuracy in certain scenarios.
COVID-19 Case Prediction	Short-term COVID-19 case prediction using LSTM and GRU.	COVID-19 case data from 10 countries.	LSTM outperformed GRU and ARIMA in predicting case numbers.

the Long Short-Term Memory (LSTM) algorithm exhibited a consistent pattern of robust performance in predicting time-series data across various domains. This algorithm consistently demonstrated superior predictive capabilities or remained competitively comparable when juxtaposed against alternative methodologies such as Autoregressive Integrated Moving Average (ARIMA) and Gated Recurrent Unit (GRU). Nevertheless, it is imperative to underscore that the determination of the optimal algorithm is inherently contingent upon the specific attributes characterizing the dataset under consideration, as well as the particular forecasting objectives inherent to the research context.
Nonetheless, it remains important to recognize that while these conclusions draw from the synthesized insights of established studies, the definitive identification of the most suitable algorithm for the precise research inquiry at hand necessitates meticulous analysis of the distinct attributes intrinsic to the dataset under scrutiny and the particular contextual requisites of the investigation.

2.8 Research Questions

Having conducted a comprehensive literature review exploring the theoretical foundations and existing studies on electric vehicle charging behaviour prediction, this section aims to outline the specific research questions that guided our investigation. The identified research questions were designed to assess the efficacy of different forecasting algorithms and their suitability for predicting the number of electric charges performed by European customers with electric vehicles. The answers to these research questions will shed light on the most accurate and appropriate model for forecasting charging behaviour, providing valuable insights for the electric mobility industry.

- Main Research Question
  - RQ1: Which algorithm (ARIMA, LSTM, or GRU) provides the most accurate predictions for the number of electric charges on a daily basis over a 7-day period by European customers with electric vehicles?
- Subsidiary Research Questions
  - RQ2: What are the strengths and weaknesses of each algorithm in predicting customer charging behaviour based on a single variable data set (number of charges) and a multivariable dataset (Inclusion of session duration, power usage, customer type, etc.)?
  - RQ3: Did the addition of external features or data improve the prediction accuracy of LSTM and GRU models, and how did it compare to ARIMA's single variable performance?
  - RQ4: How do the algorithms compare in terms of computational efficiency and resource requirements for processing the given electric mobility data?
  - RQ5: What are the implications of the study's findings for practical applications in the electric mobility industry?

By addressing the research questions outlined above, this study contributes to the understanding of machine learning algorithms' capabilities in predicting electric vehicle charging behaviour. The insights gained from this research can be used to inform decision-making processes and optimize the charging infrastructure, fostering the sustainable growth of the electric mobility sector.

2.9 Conclusion

In this chapter, an extensive exploration of comparative studies was conducted, focusing on forecasting models and their applications. The primary aim of the chapter was to establish a comprehensive understanding of the theoretical underpinnings of the algorithms under scrutiny, providing a strong foundation for subsequent research. The investigation delved into the significance of comparative studies in evaluating the efficacy of forecasting models, serving as benchmarks for prediction accuracy and applicability. ARIMA, LSTM, and GRU emerged as prominent models due to their abilities to capture and predict complex temporal patterns. The theoretical foundations of ARIMA, LSTM, and GRU were explained. ARIMA combined autoregressive and moving average components with differencing to handle stationary and non-stationary time series data. LSTM's architecture with input, output, and forget gates addressed vanishing gradient problems in RNNs, making it adept at handling time series data. GRU's streamlined structure, integrating an update gate, enabled efficient learning of temporal dependencies. The chapter delved into feature analysis and comparative metrics. The Pearson correlation coefficient assessed interrelationships between input variables, while MAE, RMSE, and R^2 score quantified predictive accuracy. These metrics facilitated robust comparative evaluation. Methods for comparing ARIMA, LSTM, and GRU results were discussed, employing MAE, RMSE, and R^2 score along with visual representations like scatter plots and line graphs to enhance understanding. This chapter significantly advanced research by providing a sturdy framework grounded in theoretical foundations, feature analysis, and metrics comparisons. These components empowered forthcoming research to identify the optimal algorithm for electric vehicle charging behaviour prediction, bridging existing knowledge and future investigations, thus establishing a strong foundation for algorithmic forecasting in electric vehicle charging behaviour prediction.

3 Understanding the Data

3.1 Origin of the Data

For privacy reasons, the company name and exact data structure and contents will not be provided.

To understand the data, it is important to understand its origins and structure. The data set being analysed belongs to a European Company, herein known as "The Company" that provides a third-party charging service to customers that own and/or drive Battery Electric Vehicle (BEV) or Plug-In Hybrid Electric Vehicle (PHEV), by granting them access to charging stations that are owned by various charging point providers. Customers of The Company, who own BEVs or PHEVs from various vehicle brands (or suppliers) can charge their vehicle at any charging station, owned by various providers throughout the EU, at a standard agreed rate. As an underlying function, The Company has contractual agreements with different suppliers to provide a standard tariff to their customers, whilst also having other contracts with various charging point providers enabling customers the freedom to charge anywhere.
All data generated by The Company, or by customers who have some interaction with The Company and their services, are stored within the internal Data Warehouse (DWH). This thesis proposes only to analyse the charging records generated by customers who have charged their vehicles at a charging station.

3.2 Relationships within the Data

To find patterns within the data to forecast based on the requirements, it may be necessary to look at more than the data relating to the charge itself. Additional insights could be derived from extensions of the charge data, which include information about the vehicle supplier, the customer and their pertaining contract information. To better understand the data connection, a high-level Entity Relationship Diagram (ERD), using Crow's Foot notation, is shown in Figure 3.1. The ERD illustrates the type of relationship between two entities, through cardinality which are the indicators on both ends of a line that connects the two entities. When considering the relationship between tblCustomer and tblContract, for example, it can be seen that one customer can have at least 1 associated contract (with no upper limit on the number of contracts). Similarly, one contract can only be associated with one customer, and cannot be shared among customers.

3.3 Data Definition

The ERD in Figure 3.1 represents the data in a normalised fashion; however, this analysis requires the data to be denormalised. This is a technique used to combine the data tables to have one table with all the necessary information, with redundancy. The data relevant to this

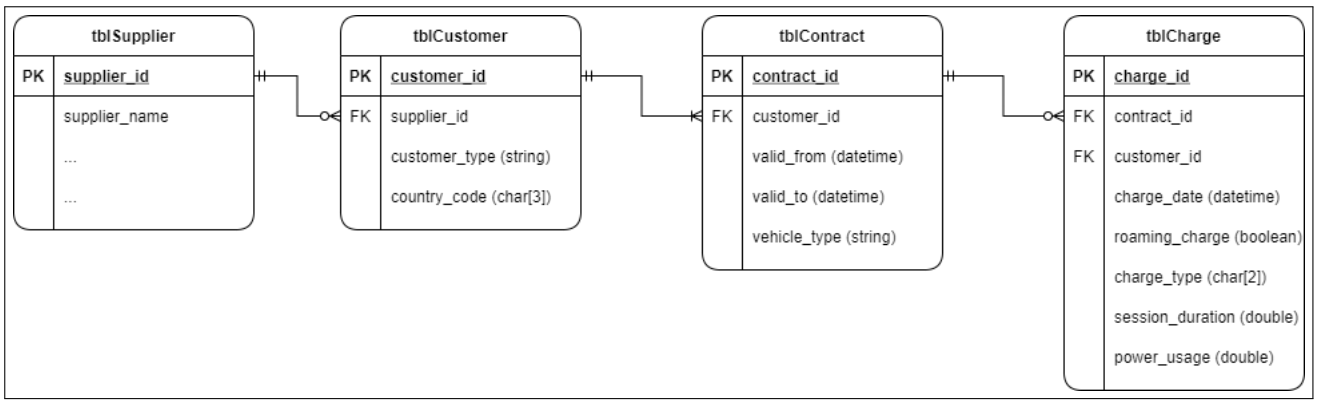


Figure 3.1: High-Level Entity Relationship Diagram

thesis has been defined in Table 3.1, with the respective data definitions.

3.4 Anonymisation of the Data

As The Company has its headquarters in Europe, it is required to comply with the law created in the European Union (EU) that protects the personal data of its citizens, known as the General Data Protection Regulation (GDPR). The GDPR stipulates what companies can and cannot do with their customers' personal data. Whilst we may understand personal data as a customer's name, surname, contact details, banking details, personal identification number, and any other methods that can directly identify a customer; data that can indirectly relate to a customer need to be anonymised too. Although the charges data does not contain any direct personal information, they do contain indirect personal information, such as the customer_id, and contract_id. Furthermore, when a customer charges, we are aware of their location, and although the customer does consent for the data to be collected and analysed anonymously, we are not able to divulge these addresses as it indicates where a customer was at a certain point in time and having the ability to track a customers' movements is against GDPR. To remain GDPR compliant, all data used within this analysis has been partially or fully anonymised, as indicated in Table 3.1.

3.5 Stationary and Non-Stationary Data Series

A time series is said to be stationary if its statistical properties, such as mean, variance, and autocorrelation, remain constant over time. In contrast, a non-stationary time series exhibits changes in these properties over time, making it difficult to model or forecast reliably. Analysis of non-stationary time series may lead to spurious results that falsely suggest a relationship between variables. Therefore, it is essential to transform non-stationary time series data into stationary data for obtaining consistent and accurate results. Differencing is a popular technique used to make non-stationary data stationary, especially for fitting an ARIMA model. In contrast, LSTM and GRU models can capture and interpret the underlying trends in non-stationary data without explicit differencing.

3.6 Data Transformation and Data Encoding

The charges data as described in Table 3.1 has only been summarised on a daily basis per individual customer. As we are interested in the *total* number of charges on a daily level, the

No	Name	Data Type	Description	Format / Content	Anonymised
01	charge_day	datetime	Date corresponding to the customer initiating the charging session.	YYYY-MM-DD	N
02	customer_id	varchar(5)	Unique value to identify the customer	Hash Function	Y
03	contract_id	varchar(15)	Unique value to identify the contract of the customer	Hash Function	Y
04	customer_type	varchar(8)	Type of customer serviced by The Company. Private customers refer to individuals whilst business refers to fleet vehicles owned by businesses	PRIVATE / BUSINESS	N
05	charging_phase	varchar(2)	Phase of the charging session	AC / DC	N
06	session_duration	double	Length of time (minutes) of the charging session	value >= 0	N
07	power_usage	double	Power usage (kWh) of the charging session	value >= 0	N
08	roaming_charge	boolean	Indicator to determine whether the charge was completed within the customer's home country or not	TRUE / FALSE	N
09	number_of_charges	double	The total number of charges completed by a customer on that day	value >= 1	N

Table 3.1: Data Descriptions of Columns within charges_data

charges data has to be transformed to have this view.

Since categorical data lacks an inherent order, it cannot be assigned an integer value with a natural ordering. Using such an encoding can lead to suboptimal performance or unpredictable outcomes such as producing predictions between categories. Instead, a one-hot encoding approach can be used where a new binary variable is introduced for each distinct category value [Brownlee 2020].

For this analysis, the categorical data will then be transformed through one-hot encoding for our machine-learning algorithms to be able to process and understand the data. Figure 3.2 shows an example of the transformed data that has one-hot encoding applied to it, using a sample data set, that will be used in the proposed analysis. The total number of charges per day has been highlighted to indicate the value to be predicted over time.

charge_day	number_of_ charges	private_charges	business_ charges	ac_charges	dc_charges	bev_charges	phev_charges	session_ duration	power_usage	valid_business_ contracts	valid_private_ contracts	new_contracts	terminated_ contracts	new_covid_ cases
2022-01-01	8,408	6,899	1,509	3,981	4,427	5,910	1,555	1,352,208	211,934	26,524	227,200	867	47	594,527
2022-01-02	11,347	9,325	2,022	4,966	6,381	8,166	1,922	1,501,973	291,908	26,543	227,790	656	14	355,898
2022-01-03	10,113	7,953	2,160	5,698	4,415	6,328	2,416	1,699,905	230,303	26,562	228,544	787	39	541,945
2022-01-04	9,284	7,283	2,001	5,465	3,819	5,742	2,282	1,570,985	203,854	26,616	229,204	753	103	932,190
2022-01-05	10,678	8,258	2,420	6,199	4,479	6,561	2,653	1,808,124	234,274	26,543	229,874	700	27	1,048,967
2022-01-06	10,763	8,618	2,145	6,101	4,662	6,928	2,492	1,783,450	243,115	26,585	230,560	754	35	904,629
2022-01-07	12,412	9,867	2,545	7,029	5,383	7,986	2,875	2,124,234	277,941	26,581	231,125	596	42	893,304
2022-01-08	12,421	9,954	2,467	6,661	5,760	8,148	2,700	1,846,296	284,464	26,564	231,585	486	49	905,804
2022-01-09	9,856	8,104	1,752	4,984	4,872	6,647	2,028	1,513,349	233,684	26,586	232,102	588	17	784,586
2022-01-10	8,724	7,205	1,519	5,463	3,261	5,205	2,433	1,775,109	181,969	26,658	232,817	804	30	715,665
2022-01-11	8,780	7,278	1,502	5,593	3,187	5,146	2,531	1,707,313	179,339	26,696	233,465	716	33	1,149,542
2022-01-12	9,396	7,800	1,596	5,935	3,461	5,711	2,556	1,775,368	190,967	26,822	234,072	766	48	1,118,501
2022-01-13	10,086	8,399	1,687	6,333	3,753	6,131	2,742	1,873,692	211,723	26,863	234,849	866	40	1,018,684
2022-01-14	11,417	9,297	2,120	6,756	4,661	7,226	2,762	1,972,479	245,529	26,856	235,421	605	70	1,041,178
2022-01-15	12,568	9,922	2,646	7,461	5,107	7,818	3,020	2,035,298	265,683	26,831	235,902	526	40	955,164

Figure 3.2: Transformed and Encoded Sample Data

3.7 Data Windowing

In order to achieve accurate time series prediction using the suggested techniques, it is necessary to reshape the data using fixed windows. This approach ensures that the model is provided with the most comprehensive information available at a particular time point from the recent past. For function X_i (where i represents the current time period), if we wish to predict X_{t+1} - next value in a time series - the data points $X_t, X_{t-1}, X_{t-2}, ...X_{t-k}$ are all fed into the model. The expectation is that the proposed algorithm will be able to calculate the auto-correlation coefficients from X_{t-k} to X_t, to predict X_{t+1}. Assuming that the data contains 30 data points, using a window size of 5 days to predict 7 days in advance, the tapered data will be structured as per Figure 3.3.

3.8 Cross Validation

Cross Validation is a technique commonly used in machine learning to evaluate the performance of a model on data that it has not yet seen. As our data set contains data over several years, we can safely assume that the number of daily charges has been impacted by both internal and external forces. Within the scope of our data set, internal forces are those that directly impact the increase or decrease of the number of charges that can be controlled by the business. Such forces can be attributed to the increase in the number of customers, price changes on the charging lower levels, and change in the sale of vehicles. These values have all been accounted for (indirectly or directly) as features in the data set.

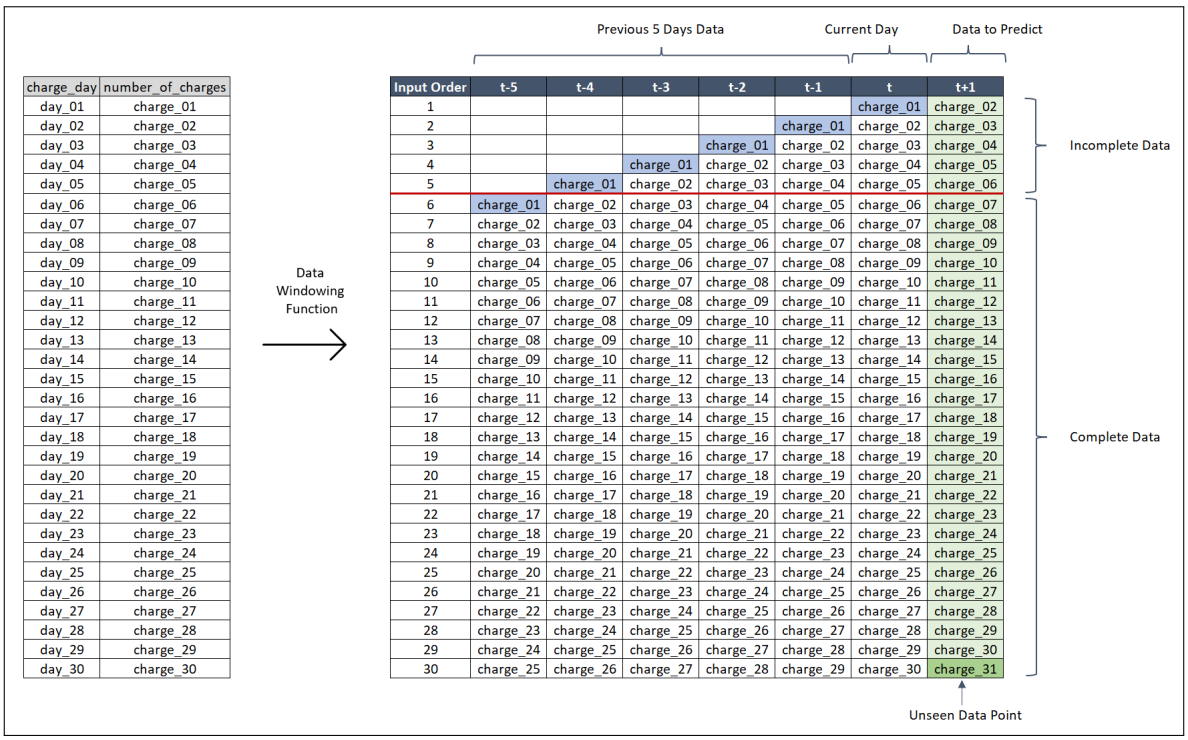


Figure 3.3: Windowing Function Example for Single Variable Data

On the other hand, the external forces are those that cannot be controlled by The Company yet affect the number of charges. A large factor that impacted the charges was COVID-19. With reference to Figure 3.4, it can be seen that the first and second lockdowns (dates as per the European Lockdown periods) had a visible impact on the number of charges as these decreased greatly. Whilst, we have considered the number of new COVID-19 cases in the data set, this cannot guarantee that we have represented its impact correctly. Further external factors include the price of energy, the price of fuel (as an alternative for PHEV), the number of vehicles manufactured, economic trends towards the use of green energy (energy that comes from natural sources) and general use of BEVs/PHEVs. The factors would all attribute to the number of charges and would have different impacts over different times.

Hence, it was decided to split the data and use the functionality of cross-validation to understand whether the models would perform in line with the results relative across all splits. Should the models produce (near) consistent results, it would mean that the selected models are able to handle different trends within the data set.

3.9 Data Assumptions

This analysis will consider the number of charges (and any related data) summarised on a daily basis from 2014-08-17 when the first charge was recorded by The Company.

Other than the anonymisation of the data, no further data exclusions have been defined; however, after the initial descriptive analysis of the data, exclusions of specific data points may be considered.

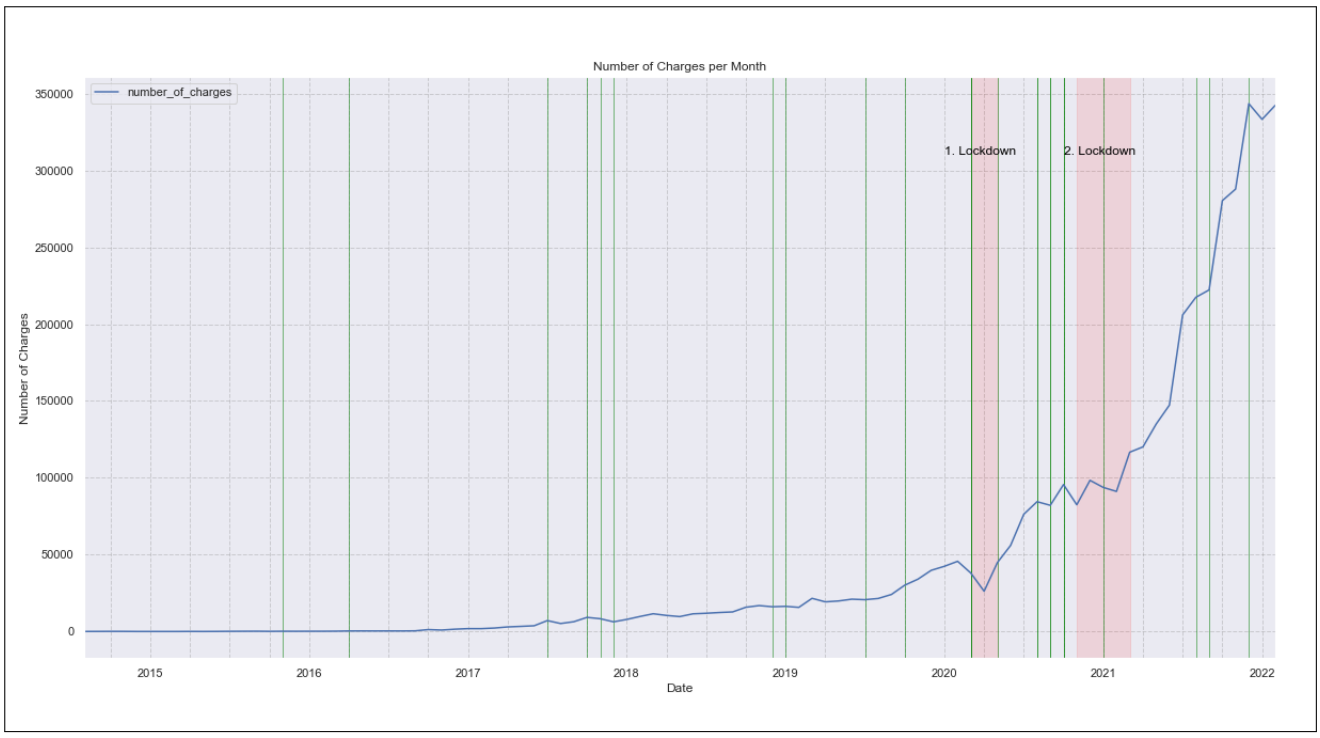


Figure 3.4: First and Second COVID-19 Lock-Downs overlaid on the Charges Data

3.10 Technical Implementation

3.10.1 Model Implementation

The raw data to be refined is housed within the main Data Warehouse (DWH) belonging to The Company. The DWH is a PostgreSQL database housed in the Microsoft Azure environment. Access to the DWH is via the database tool DBeaver version 21.0.1. Extraction of the data will be done by using SQL to create a denormalised, anonymised and one-hot encoded data table that is saved in Comma-Separated Values (CSV) format.

Using Python 3.7.4 64-bit, the Comma-Separated Values (CSV) file will then be read in and analysed accordingly. The initial descriptive analysis will require data to be plotted, and visualizations will be created using `matplotlib`; a comprehensive library for creating static, animated, and interactive visualizations in Python.

The implementation of the LSTM and GRU models will be developed using an open-source software library that provides a Python interface for artificial neural networks known as Keras. The ARIMA model will be implemented using `statsmodels` (or more specifically `statsmodels.tsa`), a Python module that contains model classes and functions useful for time series analysis.

To determine the optimised hyperparameters for the Keras models (LSTM and GRU), a tool known as Talos will be used that allows for fully automating hyperparameter tuning and model evaluation [Talos 2019]. For the ARIMA model, tuning of the parameters can be done by performing statistical correlation of the data using a Complete Auto-Correlation Function (ACF) and a Partial Auto-Correlation Function (PACF) function. The optimised hyperparameters for all models, the training data and testing data, and the predicted values generated by the models will be stored in CSV format.

The codebase for this project is available on GitHub: https://github.com/robynepelwan/masters

3.10.2 Hyperparameter Tuning Tool Selection and Implementation

Talos is a hyperparameter optimization library for deep learning models. It is an open-source tool that allows users to easily optimize the hyperparameters of their neural network models. TALOS is designed to work with the Keras deep learning library and can be used to optimize a wide range of model architectures, including feedforward neural networks, convolutional neural networks, and recurrent neural networks (such as LSTM and GRU models) [Talos 2019].

The library provides a simple and flexible interface for specifying the search space and the optimizer to be used. Once the search space and optimizer are defined, Talos will automatically perform a search over the space of hyperparameters to find the optimal set of hyperparameters for the given model. It uses Bayesian optimization to search over the space of hyperparameters, which is a method that uses a probabilistic model to guide the search process. This method has been shown to be more efficient than grid search and random search, which are other common methods for hyperparameter optimization. It also provides an option to use Hyperband which is a method that is designed to optimize the performance of a model within a fixed computation budget. This means that it will find the best models with the least number of iterations.

For the ARIMA model, we chose to use `auto_arima` to find the optimal hyperparameters. This automated algorithm used a combination of statistical tests and heuristics to find the best hyperparameters for the model. It fits a range of models with different combinations of hyperparameters, and then uses information criteria such as Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) to select the best model. It also includes a step of differencing the time series data, if necessary, to make it stationary.

4 Exploratory Analysis

4.1 Introduction

The core focus of The Company is to develop a strategy that continuously allows the number of charges to increase over time. Simply put, an increase in charges means that more and more customers are using the service of The Company which translates to increased revenue margins. Of course, understanding the development of charges over time is quite complex and needs to be investigated. When considering Figure 4.1, it can be seen that over the course of the last few years, the number of monthly charges has increased significantly - 2021 saw nearly a 300% increase from the number of charges completed in 2020. The growth of these charges are due to both internal and external factors. In this Chapter, we will look at different internal factors to determine whether these have any correlation to the factor we wish to predict. Although we have predicted the number of charges on a daily basis, the graphs and data below have been summarised on a monthly basis for ease of viewing.

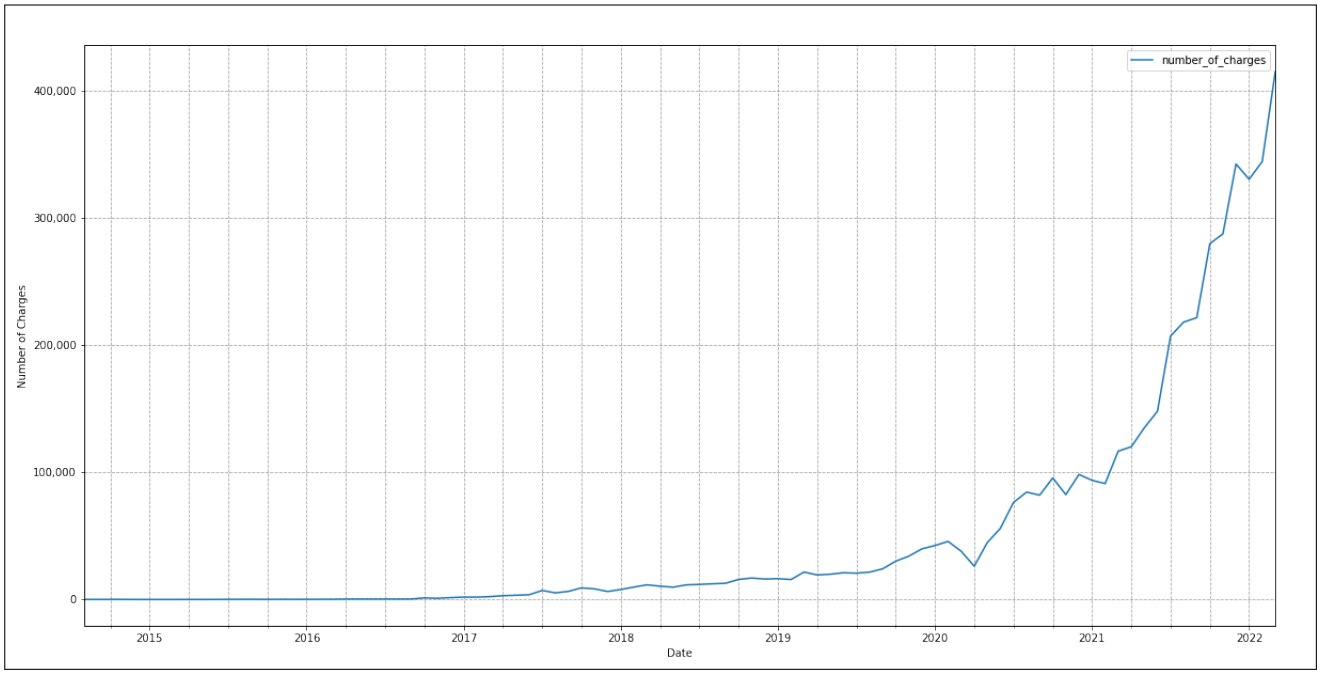


Figure 4.1: Total Number of Charges Summarised by Month

4.2 Data Deep-Dive

4.2.1 Charges by Customer Type

The Company services two kinds of customers; private customers and business customers, where private customers make up an average of 79.64% of the total customer base. The split on a monthly basis has been shown in Figure 4.2. Due to the proportion of the private customer segment that has charged their vehicles in comparison to the business customer segment, we can assume that this customer type has an influence on the prediction of the number of charges.

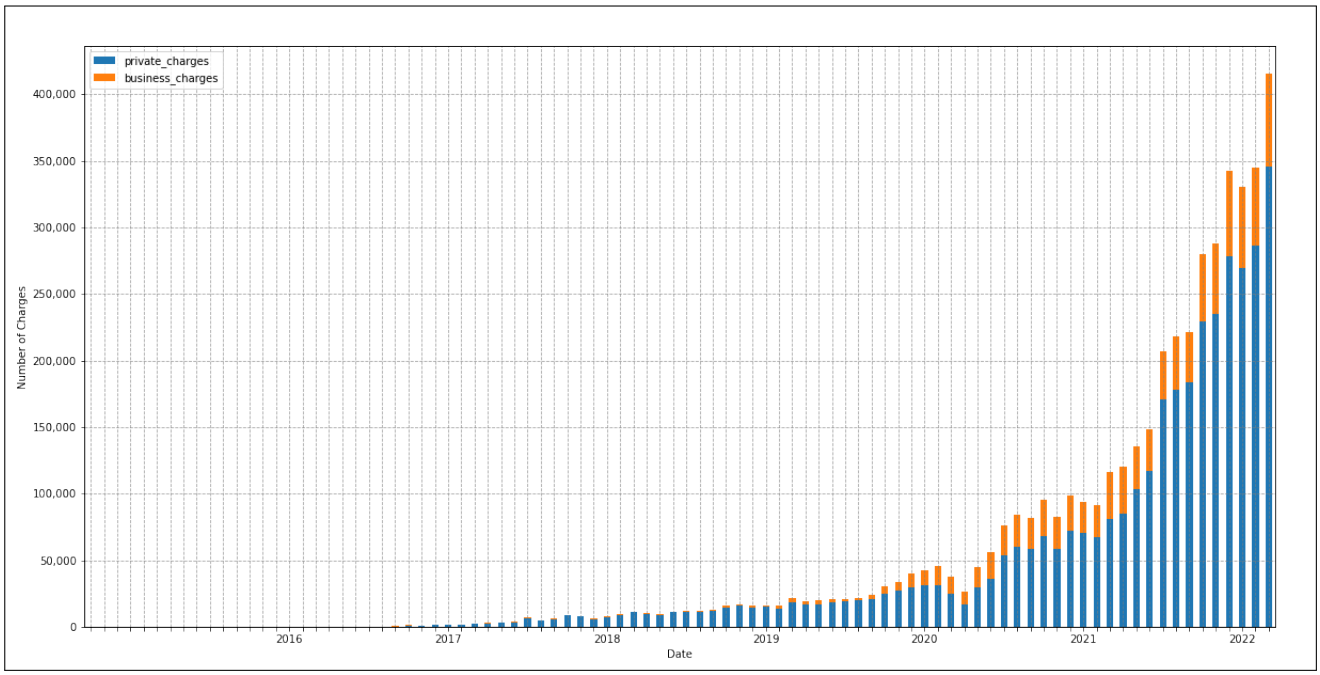


Figure 4.2: Total Number of Charges Split By Customer Type

4.2.2 Charges by Power Type

Based on the kind of vehicle that the customer has, they are allowed to charge their vehicle with either Alternating Current (AC) or Direct Current (DC). The current is important to The Company as each current type is priced differently; hence it is relevant to understand the split of the charges by current type. Prior to 2020, 84.12% of all charges were attributed to AC. This is largely due to the vehicles that were available at the time that only allowed AC charging. However, there has been an increasing demand for electric vehicles due to government backing in Europe, which would drive the demand for electric vehicles up, and hence faster charging batteries. Hence, after 2020, 62.90% of all charges were AC. Whilst the demand for DC charging stations is rising, AC still makes up for the bulk of the charges and we expect that these charges have a positive correlation to the number of charges in our thesis. The AC/DC split on a monthly basis has been shown in Figure 4.3.

4.2.3 Charges by Vehicle Type

The Company provides a service to vehicles that can be charged, namely Battery Electric Vehicle (BEV) and Plug-In Hybrid Electric Vehicle (PHEV). As BEVs rely solely on battery life,

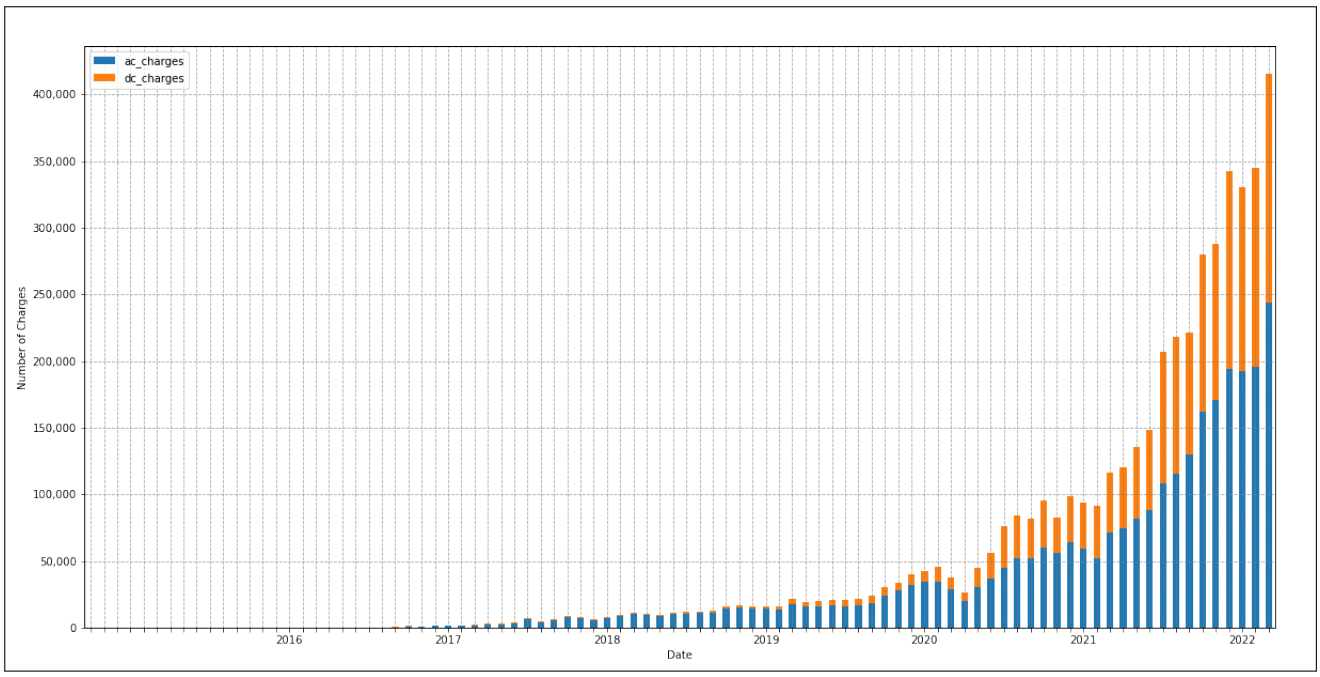


Figure 4.3: Total Number of Charges Split By Power Type

The Company wishes to gain more of these customers as the frequency of their charges could be higher than that of a PHEV. This is because PHEVs have the option to drive on a combustion engine should they have no ability to charge their vehicle. Prior to 2016, nearly all vehicles on The Company's portfolio were BEVs; between 2016 to 2019 the BEV/PHEV split was at 60/40. However, the demand for BEV is increasing and largely due to government backing (as mentioned previously). BEVs batteries are also improving over time, meaning that the vehicles can drive further / longer distances and are comparable to normal combustion engine distances now. Post 2020, we see that 75% of all vehicles that have charged are BEVs. The BEV/PHEV split on a monthly basis has been shown in Figure 4.5. It is important to note that not all vehicles have been classified as BEV/PHEV due to data errors and these vehicles have been marked as "UNKNOWN".

Research and development of the energy storage systems in BEV and PHEVs, are ongoing to reduce their relatively high cost and extend their useful life. Batteries that can last longer and be charged faster are now sought after to encourage drivers to consider vehicles using alternative fuel sources. Interestingly enough, hints at battery improvements can be seen in Figure 4.5, as post-2019, the average charging capacity of DC batteries now ranges between 40-50 kWh. AC remains between 10-20kWh. Vehicles that have batteries that last longer after a charge may not charge as often as those with lower distance batteries - however, with the lack of understanding of the customer behaviour, we cannot make such an assumption. Though, should this be considered as an important factor for predicting the number of charges, this inferred charging behaviour should be picked up by the respective model.

4.2.4 Other Features to Consider

In this analysis, we have considered several other factors within the data that could be helpful in the training of the model. These include

> **The Number of New Contracts** is the number of new contracts that are valid at The Company and able to charge on the date of inspection.

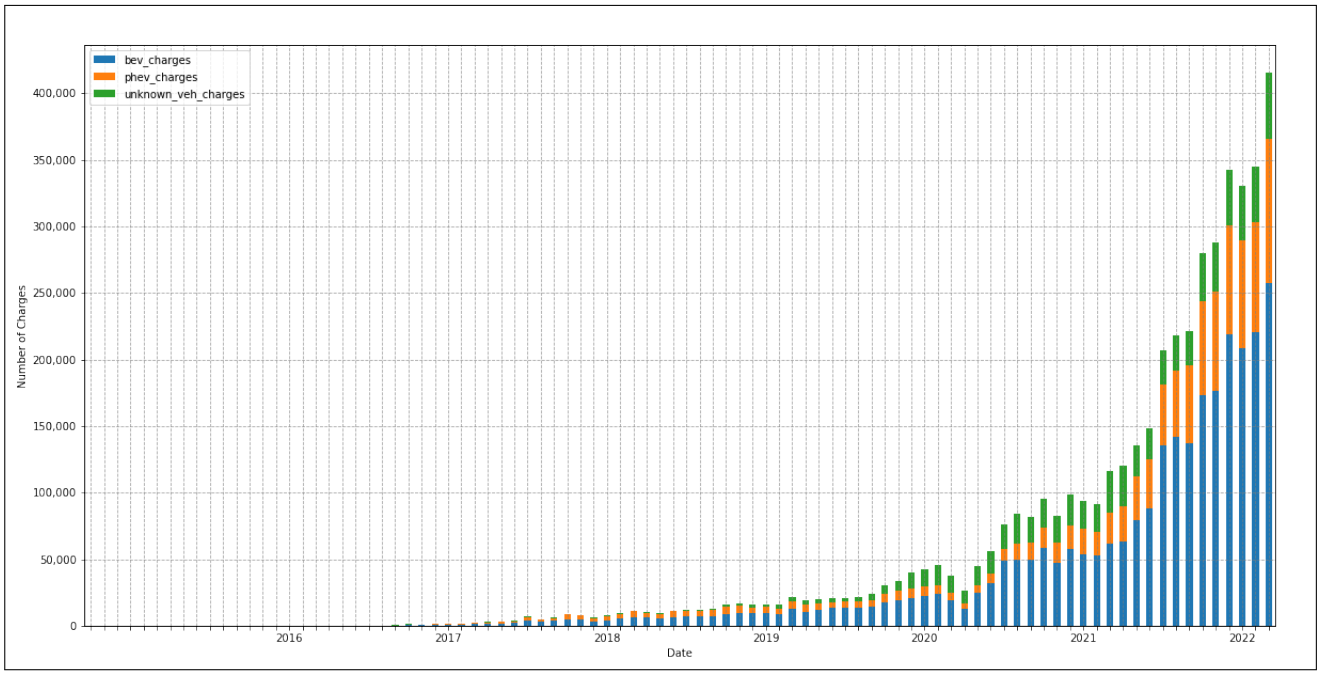


Figure 4.4: Total Number of Charges Split By Vehicle Type

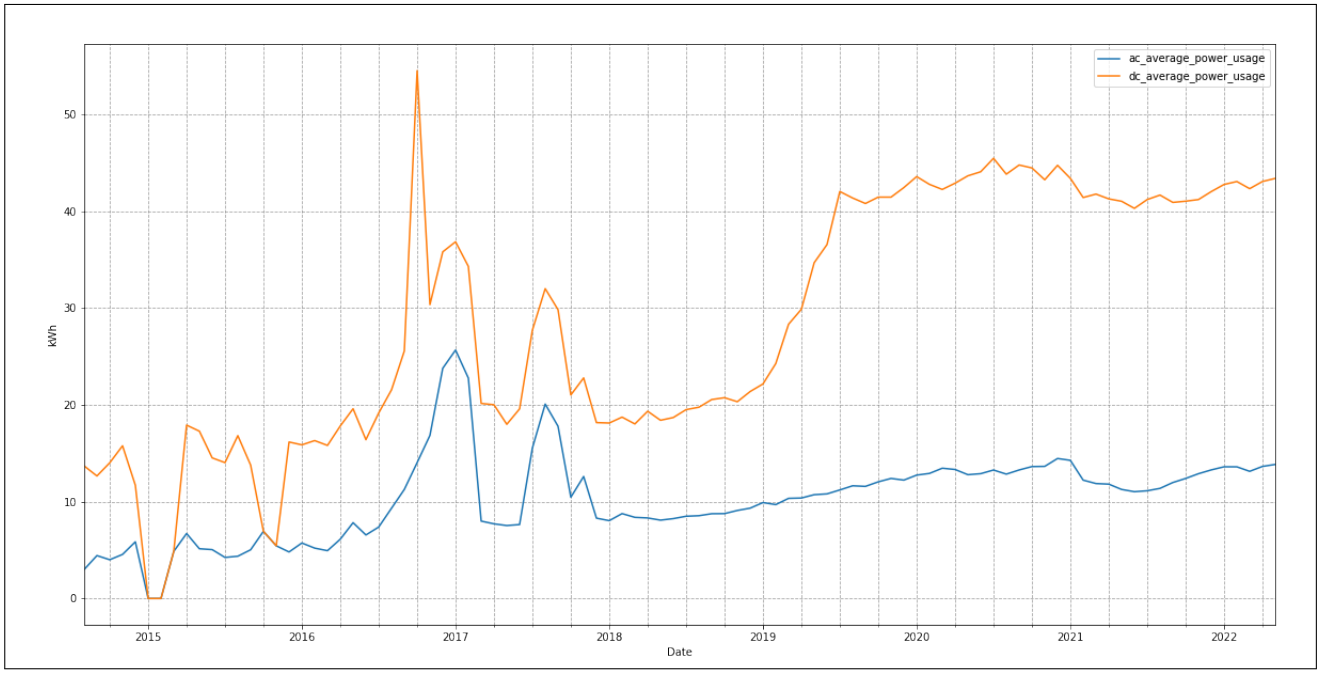


Figure 4.5: Average Power Usage (AC/DC Split)

The Number of Terminated Contracts is the number of contracts that ended their relationship with The Company and are no longer able to charge on the date of inspection.

Total Session duration is the sum of the session duration (in minutes) of all charges that were performed on the date of inspection.

Total Power Consumption is the sum of the power consumption (in kWh) of all charges that were performed on the date of inspection.

4.3 Increasing the Number of Charges

The core focus of The Company is to ensure that the number of charges increases over time, this would mean that more and more people are using the service which translates to increased revenue margins. Several aspects can be directly attributed to an increase in the number of charges, and are discussed below.

4.3.1 New Customers and Existing Customers

To gain a large chunk of customers, The Company is able to have a contractual agreement with a new original equipment manufacturer (OEM), where customers who purchase new vehicles from said OEM will automatically be signed up to the service of The Company upon delivery of their new vehicle. It is important to note, that whilst a customer with a new vehicle is signed up with The Company automatically, the customer is allowed to sign up with other service providers should they deem fit. In Figure 4.6, the number of charges has been plotted with green vertical lines where each line indicates the sign-up of a new OEM. Of course, signing up a new OEM does not translate to an immediate increase in the number of charges but there is a steady increase over time.

As stated before, it is not necessary that a customer signs up with The Company, despite their OEM contractual agreements with The Company. Hence, another manner to increase the number of customers is by appealing to these customers to join The Company. Similarly, existing customers would need to be encouraged to use the services of The Company more than their current behaviour. To push people to use the services of The Company, The Company would need to provide some incentive to encourage this behaviour.

4.3.2 COVID-19

The impact of the Coronavirus Disease 2019 (COVID-19) virus has been felt globally, affecting countries, economies and individuals. Hence, it is no surprise that the virus affected the number of charges. In Figure 4.7, the two major download periods in Europe have been highlighted against the charges data, where it can be seen that the number of charges decreased during this time. Whilst there have not been other lockdowns that have taken place, it is important to note the significance that COVID-19 has had on the data, and hence, appropriate to reflect this in the predictions that follow.

4.4 Feature Analysis

By only looking at the charges, we may not be able to fully understand why the movement of the data occurs over a period of time. In this analysis, we looked into measurable properties

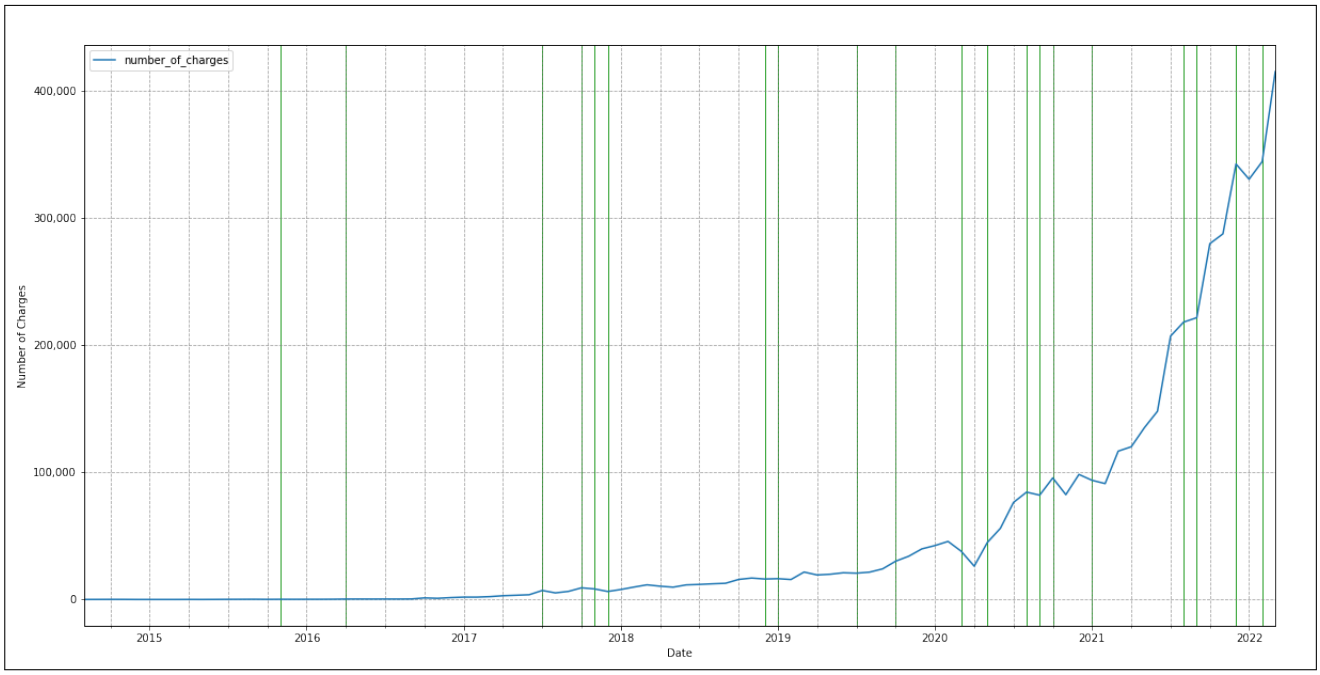


Figure 4.6: New OEM Contractual Agreement Signing with The Company

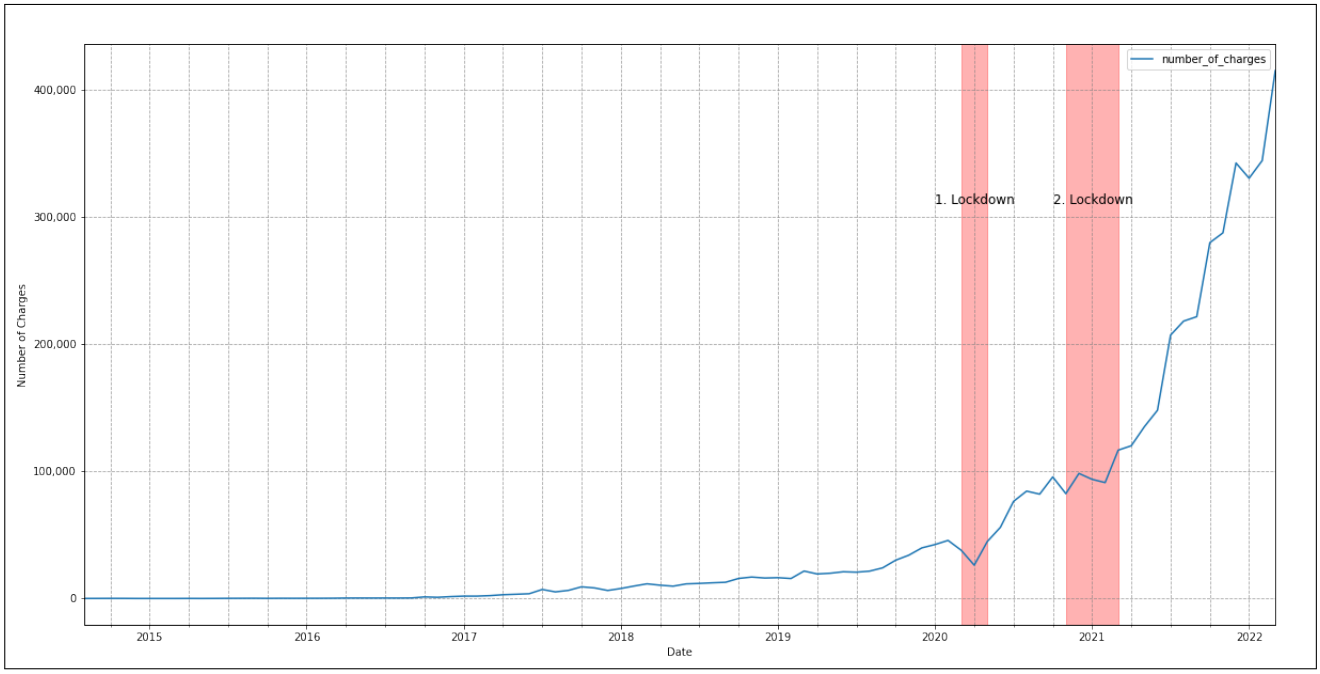


Figure 4.7: Effect on the Number of Charges during COVID-19 Lockdown Periods

within the data, known as **features**, to see if these were able to give us further insights. This exploratory analysis revealed many different features that we have considered in our analysis. As explained in Section 2.6, we aimed to select features in the data that could improve the performances of our models. As our input data was numeric, and our output data was numeric, we used a linear regression feature selection known as the Pearson correlation coefficient. This method measured the strength between different variables by comparing every variable to every other variable and returned a value between -1 and 1. A correlation coefficient of 1 indicates a strong positive relationship between the variables, whilst -1 indicates a strong negative relationship. A correlation coefficient of 0 indicates no relationship at all. Our main goal was to view the relationship of all the features compared to the number of charges, but having the comparison of all other variables would also indicate the co-dependencies in the data.

4.4.1 Feature Identification

We have identified 13 additional features that could affect the prediction of the number of charges in Table 4.1. Each of these features was calculated on a daily basis to match the format of the number of charges.

No	Feature Name	Description
01	number_of_charges	The total number of charges
02	private_charges	The number of charges performed by private customers
03	business_charges	The number of charges performed by business customers
04	ac_charges	The number of AC power charges
05	dc_charges	The number of DC power charges
06	bev_charges	The number of charges performed by customers with BEVs
07	phev_charges	The number of charges performed by customers with PHEVs
08	session_duration	The total session duration (in minutes) of the charges performed
09	power_usage	The total energy usage (in kWh) of the charges performed
10	valid_business_contracts	The number of valid business contracts that have access to charge their vehicles
11	valid_private_contracts	The number of valid private contracts that have access to charge their vehicles
12	new_contracts	The number of new contracts signed up with The Company
13	terminated_contracts	The number of contracts that have reached their end date
14	new_covid_cases	The number of new covid cases within Europe

Table 4.1: Description of the Identified features

4.4.2 Pearson's Correlation Coefficient

Pearson's Correlation Coefficient was calculated for each combination of the identified features using the formula

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{4.1}$$

where r is the Pearson Coefficient, n is number of the features, and x and y are the features compared at the time. To note, because we predicted 7 days in advance, we cannot compare the 13 features to the number of charges that happened on that day. In order to understand the relationship of the features to the number of charges, we need to compare each of the features on days D at times [D_{t-7}, D_{t-6}, ... D_{t-1}] to number_of_charges at day D_t. In doing so, we have allowed for the data to be viewed from one period of time, and be compared to another period of time without overlap, to avoid data leakage. However, since we will be comparing 7 days' worth of data per feature, we had to calculate 7 different Pearson's Correlation Coefficients per feature. In Figure 4.8, the correlation coefficients can be seen comparing each feature to every other feature, using a lag of 1 day or at time D_{t-1}. For the number of charges, we can see that many of the features have a very strong positive relationship with it - private_charges, bev_charges, valid_private_contracts scoring the highest of 0.99. Whilst this high correlation is suspicious, it is important to note that the number of charges over time was largely performed by private customers, and also largely performed by customers who drove BEVs. Furthermore, private charges can only be performed by customers that have a private contract, and hence, this feature scored highly. Further features with high scores include the ac_charges and dc_-charges (with both features ranking the same given that the number of charges roughly are split between 50% AC and 50% DC), as well as phev_charges and session_duration. On the other hand, we found that the total power usage and the number of terminated contracts have a very weak to no relationship with the number of charges as a feature. Interestingly, despite the fact that the new_covid_cases only became relevant at the end of 2019, it does have a reasonable relationship with the number of charges.

When considering the entire result from Figure 4.8, we found that the features that scored highly against the number of charges also scored highly among each other. Similarly, if a feature scored low against the number of charges, it was likely to score also low against the other features. Whilst, this is not a common result of the correlation analysis across all data sets, it is reasonable for the relationships within *this* data set given the business model and data dependencies of The Company. It was, hence, possible to further delve into the features and only compare them to the number of charges and not consider the other relationships in the data.

We compared the correlation coefficients of all variables only to the number of charges at different lag times, to see if there was a large discrepancy in the relationship over the 7 days. The relationship over the time period did not alter by much, showing a similar relationship per feature across all time lags. A visual representation of the correlation coefficients spread per feature is shown in Figure 4.9, with the median correlation coefficient per feature over the seven-day period displayed numerically.

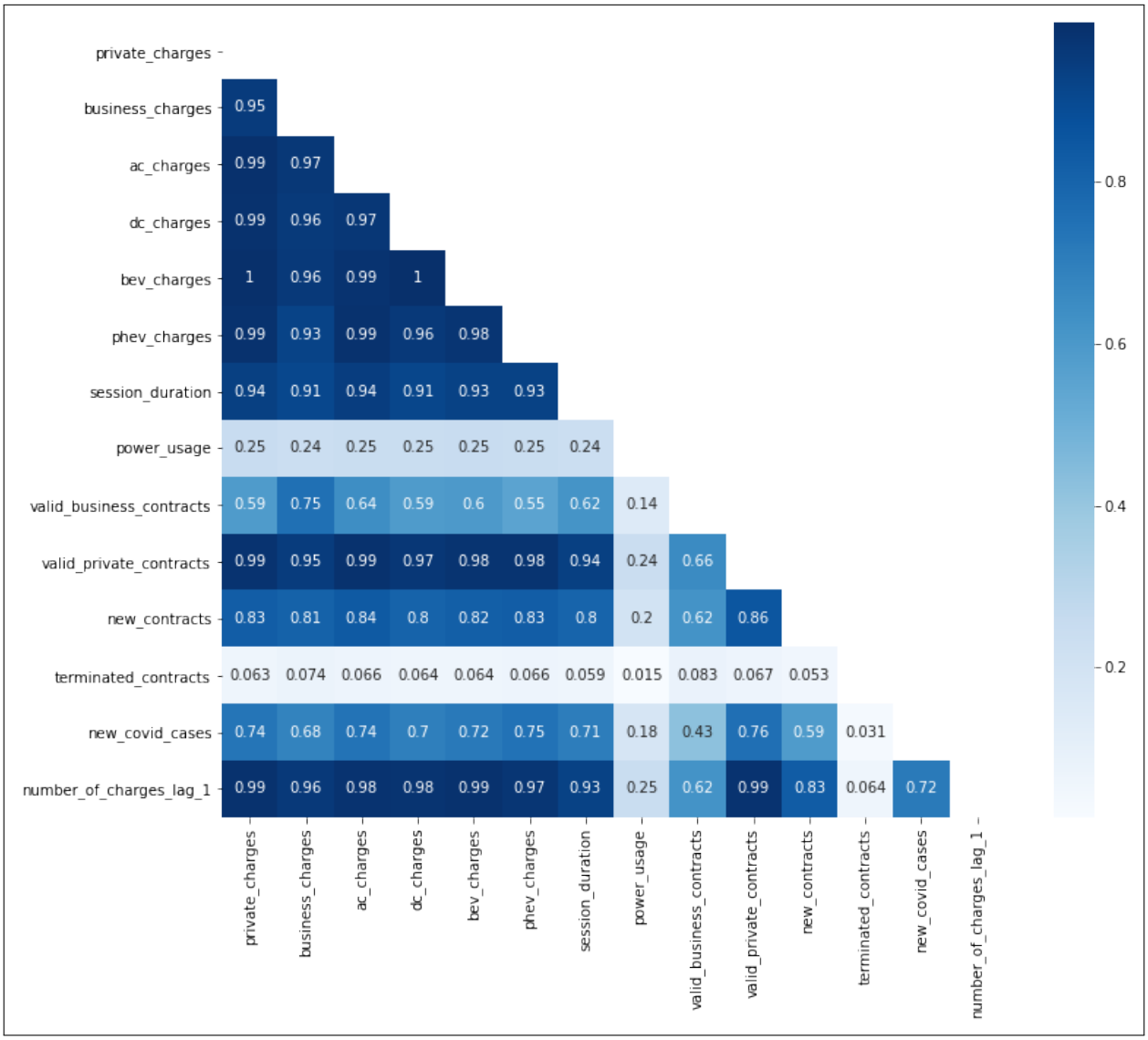


Figure 4.8: Feature Comparison using Pearson's Correlation (with 1 lag day)

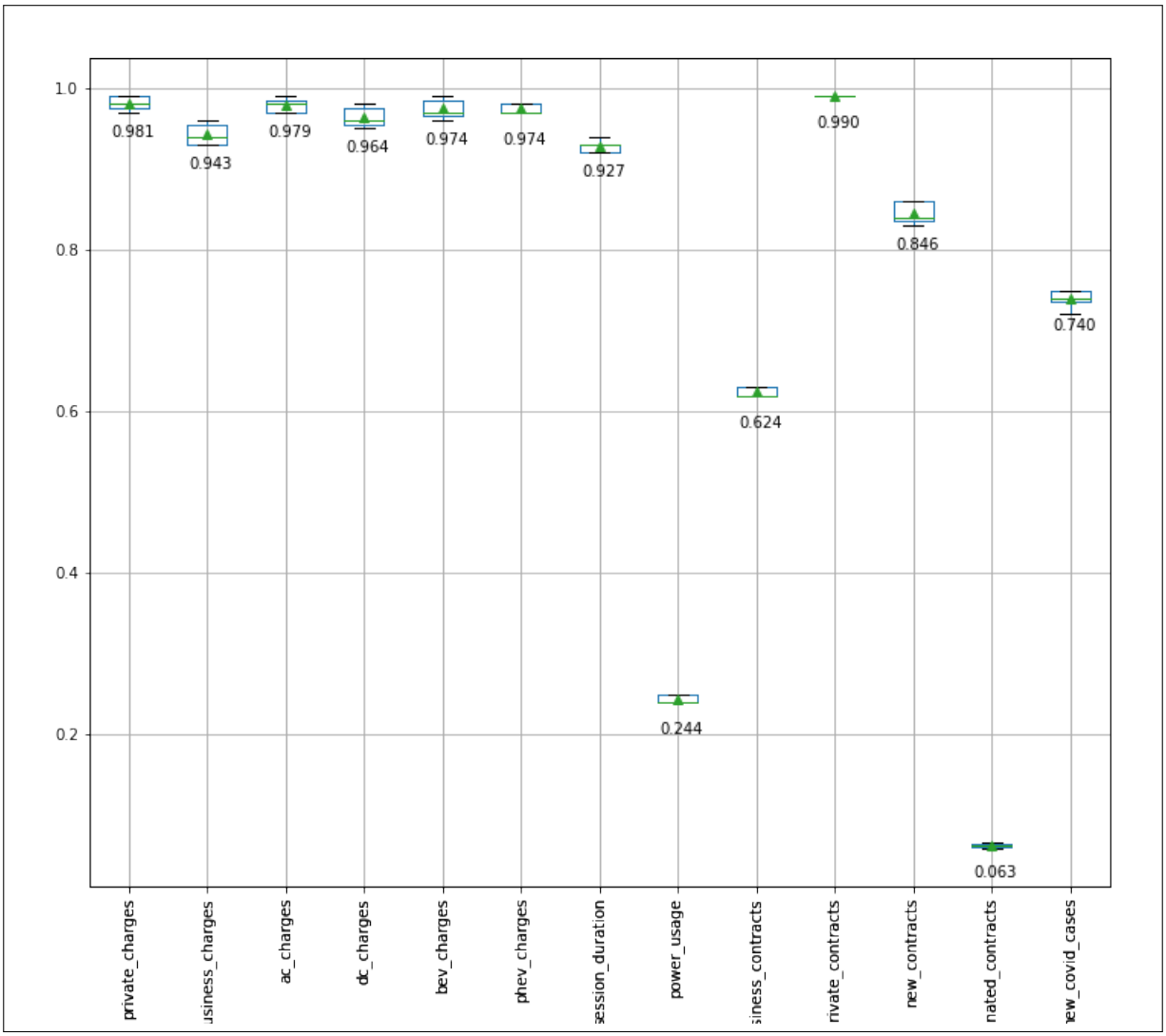


Figure 4.9: Average Correlation Coefficient per feature compared to the number of charges

4.4.3 Feature Selection Consideration

When selecting the appropriate features to model our data, we would have selected those with the highest correlation to the number of charges. However, in this case, the data is so interdependent on each other that should we select all features with the highest correlation, we may still be selecting too many features that would not deem the result we wanted - which is using the least amount of features to accurately determine the number of charges for 7 unseen days. In Chapter 6, we used all the features to determine the number of charges, and in Chapter 7, we used only three features to do the same.

5 Single Variable Analysis

5.1 Introduction

The purpose of a single variable analysis is to determine whether we can use the history of the number of charges without any other factors to forecast the future number of charges. In doing so, we assume that the past charges infer any patterns from internal and external factors. The success in predicting the number of charges for each of the selected algorithms will depend on how the algorithm can determine the patterns within the data and forecast the number of charges. Within this section, we will look into the prediction process and results for each of the selected algorithms.

5.2 Long Short-Term Memory (LSTM)

5.2.1 Model Data Preparation

Prior to any predictions, the LSTM was configured based on input-specific parameters, namely explicit parameters taken directly from the input data, and hyperparameters derived from the input data. Algorithm 3 shows the high-level process that we need to follow in order to predict and verify the number of charges.

Algorithm 3 LSTM Process to predict the number of charges

Require: `raw_data`, `window_size`, `days_to_predict`
Ensure: data contains no null values

1: Create `input_data` by transforming and tapering the `raw_data` using `window_size` and `days_to_predict`
2: Split the tapered `input_data` into three data sets: training, validation, test
3: Using the `training_data` to train the LTSM and a variety of hyperparameters, find the optimal hyperparameters using the `validation_data` to verify the result
4: Train the model on the `training_data` using the optimal hyperparameters
5: Predict the number of charges for a number of days defined by `days_to_predict`
6: Determine the rate of error between the predicted value and the `test_data` set

Once the data has been transformed and encoded, it needs to be tapered (or windowed) as per Section 3.7. Using Algorithm 3, the number of charges was predicted for the next seven days (`days_to_predict = 7`) using a windowed data period of fourteen days (`window_size = 14`). This meant that we considered two weeks prior to forecast the week that follows. Figure 5.1 shows how twenty-one days of data will be arranged by the windowing function to produce an input data set (or training data set) of fourteen days, and an expected output data set (or prediction data set) of seven days. Note that whilst the initial data set contains 2,639 data points, due to the windowing function, only 2,618 days contain complete data and can hence be used in the prediction.

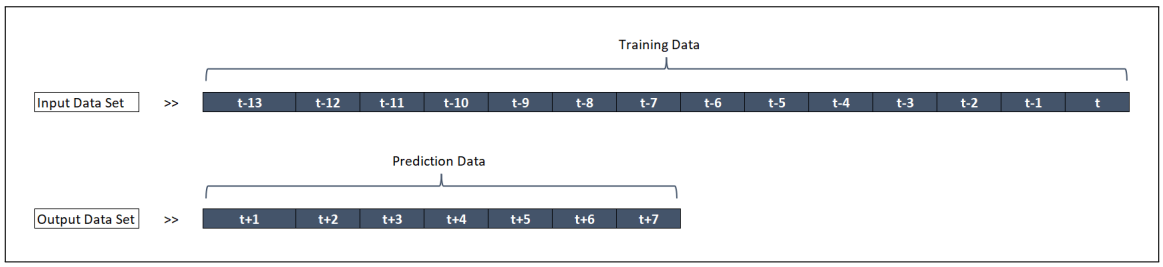


Figure 5.1: Arrangement of Training and Prediction Data created by the Windowing Function at Time t

Following the data preparation process that created the tapered data set `input_data` of size X, the data needs to be split into three data sets training, validation, and test data to ensure that there is no data leakage of the test data when training and validating the LSTM. The initial process step "I: Split Data" in Figure 5.2, reflects the splitting of the `training_data` in size k (`k = X - days_to_predict`) and the `test_data` into size `days_to_predict`. The test_data is excluded from all further steps until it is required once again in the final step to evaluate the model performance.

Process step "II: Determine Optimal Hyperparameters" defines the process of finding the optimal hyperparameters by experimenting only on the training_data taken from the previous step. To ensure that the experiments can run on one data set and then be tested on another data set, training_data is split into two data sets, namely the experiment training data - exp_training_data (of size k - `days_to_predict`) and the experiment validation data - `exp_validation_data` (size `days_to_predict`).

Thereafter, process "III: Predict the number of charges" trained the LSTM model using the training_data and the selected hyperparameters to forecast the number of charges for seven days, discussed in Section 5.2.3 and 5.2.4. Finally, to evaluate the decency of the predicted values, Process step "IV: Evaluate Model Performance" was then conducted, discussed in Section 5.2.5.

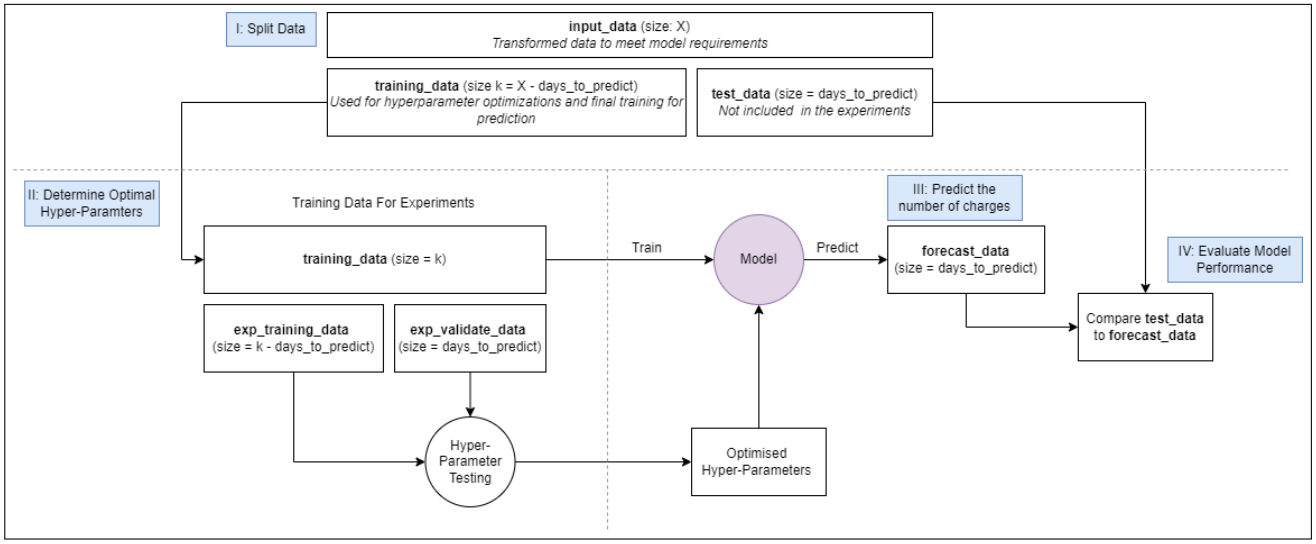


Figure 5.2: Data Split Process: Train, Validate and Test Data

5.2.2 Scaling the Data

In addition to the data tapering, we scaled the data to change the values of numeric columns in the dataset to a common scale. As the features had different ranges, it was very important to scale data before inputting it into the LSTM. The class `sklearn.preprocessing.MinMaxScaler` was used to transform all features individually such that it is in the given range on the training set, which we have selected as between zero and one. Assuming we have feature `X` defined as a numerical value per day. Using the min-max normalisation method, we had to map the numeric range of this feature to the scale range in `[0,1]` to create the scaled values defined as X' as per Equation 5.1.

$$X' = \frac{X - min(X)}{max(X) - min(X)} \tag{5.1}$$

After the scaled data set has been used to train and test the model, the model outputs predicted values (that are also scaled) that need to be unscaled using the same scaler function that was used during the scaling process.

5.2.3 Hyperparameter Tuning

To determine the best combination of hyperparameters, the LSTM model is trained using the data from `exp_training_data` and a random selection of the hyperparameters. The trained model then forecasts the number of charges for the next `days_to_predict` number of days. These forecasted values were then validated against the `exp_validation_data` using a selected model performance measure known to produce a validation loss. This process is then restarted using the same exp_training_data but with a new combination of hyperparameters, then validated against the `exp_validation_data` to produce another validation loss. Table 5.1 shows the different hyperparameters considered, as well as the range of values to be considered. Given the number of hyperparameters and the values of consideration, if we were to test every combination of the hyperparameters, we would need to run over 580 000 experiments, just for one data set.

To use Talos [2019] to identify the optimal hyperparameters, we altered the LSTM model to accept a dictionary of lists of hyperparameters (defined as `params`) that were to be tested. This implementation can be seen in Listing 5.1.

```
from keras.layers.core import Dense, Activation, Dropout
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, LSTM

def lstm_model(X_train, Y_train, X_test, Y_test, params):

    model = Sequential()
    model.add(LSTM(params['nodes'], input_shape=(X_train.shape[1],
                                X_train.shape[2])))
    model.add(Dropout(params['dropout'],seed=config.SEED))
    model.add(Dense(7))
    model.add(Activation(params['activation']))
    model.compile(loss=params['losses'], optimizer=params['optimizer'])

    history = model.fit(X_train,
```

```
                    Y_train,
                    epochs=params['epochs'],
                    batch_size=params['batch_size'],
                    validation_data=(X_test, Y_test),
                    callbacks=[EarlyStopping(monitor='val_loss',
                              patience=params['patience'])],
                    verbose=0,
                    shuffle=False)
    return history, model
```

Listing 5.1: LSTM Model Definition

We chose to run the experiment one hundred times, which meant that [Talos 2019] randomly chose one hundred combinations (to perform one hundred experiments) of the selected values from the features in Table 5.1. Per experiment, Talos trained the LSTM model defined in Listing 5.1 using a selected combination of features, and the `exp_training_data` as the input data set. The trained model then predicted the number of charges for the next seven days and these values were then compared to the actual number of charges that occurred in the `exp_validation_data`. The difference between these figures was measured by the selected loss function to produce a validation loss. The model and respective hyperparameters with the lowest validation loss were then saved to use as the final model to forecast future charges. As the input data was scaled, the validation loss values are also scaled between [0,1]. In experiment number 45, the lowest **validation loss of 0.00887** was found and corresponds to the first set of hyperparameters found in Table 5.2. The spread of the validation loss per experiment number is shown in Figure 5.3.

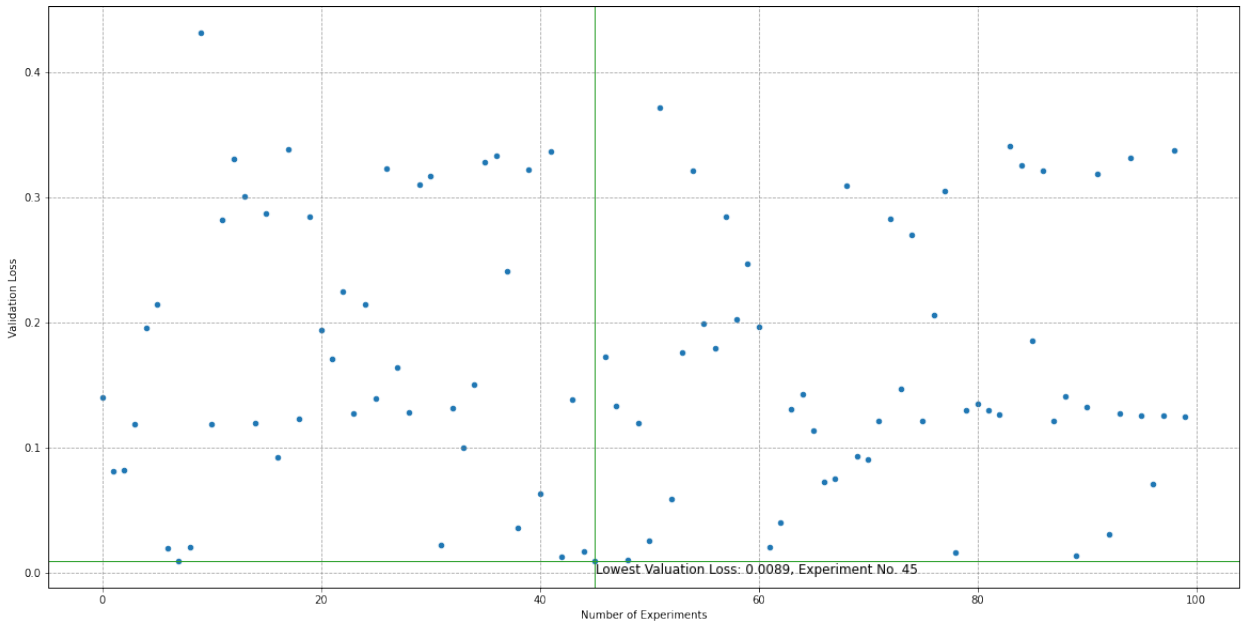


Figure 5.3: Validation Loss Result Spread for LSTM Single Variable Experiments

5.2.4 Model Training

The hyperparameters that resulted in the lowest validation loss were stored and used to train the model defined in Listing 5.1 can be seen in Table 5.2. The difference between the experimentation phase and the training phase was due to the data used when training. Instead of the `exp_training_data`, the LSTM was trained using the full `training_data` data set. Once trained, the model was then expected to predict the number of charges for the next seven days.

Parameter to Optimise	Selected Values
nodes	1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024
batch_size	1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024
epochs	20, 40, 80, 100, 200
dropout	0, 0.01, 0.05, 0.075, 0.1,0.2, 0.3, 0.5
patience	20, 30, 50, 100, 200
optimizer	"Adam", "Nadam", "RMSprop"
losses	"mean_squared_error" (MSE), "mean_absolute_error" (MAE)
activation	"relu", "sigmoid", "softmax", "tanh"

Table 5.1: Selection of Potential Values for the Hyperparameter Tuning

No	Testing Loss	Validatio Loss	Activatio	Batch Size	Drop Out	Epochs	Losses	Nodes	Optimizer	Patience
01	0.00058	**0.00887**	relu	32	0.2	200	MSE	256	RMSprop	20
02	0.00037	0.00935	tanh	2	0.075	100	MSE	1,024	RMSprop	20
03	0.00045	0.01027	tanh	8	0.1	80	MSE	128	Adam	100
04	0.00041	0.01282	relu	8	0.075	200	MSE	64	Nadam	200
05	0.00039	0.01382	tanh	8	0	80	MSE	256	Nadam	50

Table 5.2: LSTM (Single Variable): Five Lowest Validation Losses with the Associated Hyperparameters

5.2.5 Model Results

For the 7-day period ending on the 31st of March 2022, LSTM predicted a total of 94,211.69 charges. This is 542.31 charges (or 0.57%) less than the actual amount of 94,754 charges that occurred during this week. The measure of difference between the actual and predicted values, the RMSE, was 460.362; which is relatively low compared to the actual number of charges. The detailed day-by-day breakdown comparing the actual number of charges and predicted number of charges and the RMSE for the last 7 days in March 2022 were calculated and shown in Table 5.3. Figure 5.4 visually represents the actual number of charges for the last month of inspection, overlaid with the predicted number of charges for the last seven days.

Date	Actual Value	Predicted Value	Difference	% Difference
Day 01	14,878	14,827.44	-50.56	-0.34%
Day 02	15,476	15,315.16	-160.84	1.04%
Day 03	13,806	14,044.79	238.79	1.73%
Day 04	11,844	12,457.83	613.83	5.18%
Day 05	12,119	12,164.83	45.83	0.38%
Day 06	12,703	12,451.31	-251.69	-1.98%
Day 07	13,928	12,950.32	-977.68	-7.02%
Total	**94,754**	**94,211.69**	**-542.31**	**-0.57%**
RMSE: 460.362				

Table 5.3: LSTM Number of Charges: Actual vs Predicted per Day

5.2.6 Residual Analysis with Confidence Intervals

The differences between the actual and predicted values were extracted and graphed, with the resulting values fitted to a normal distribution curve. Calculating the mean and standard devia-

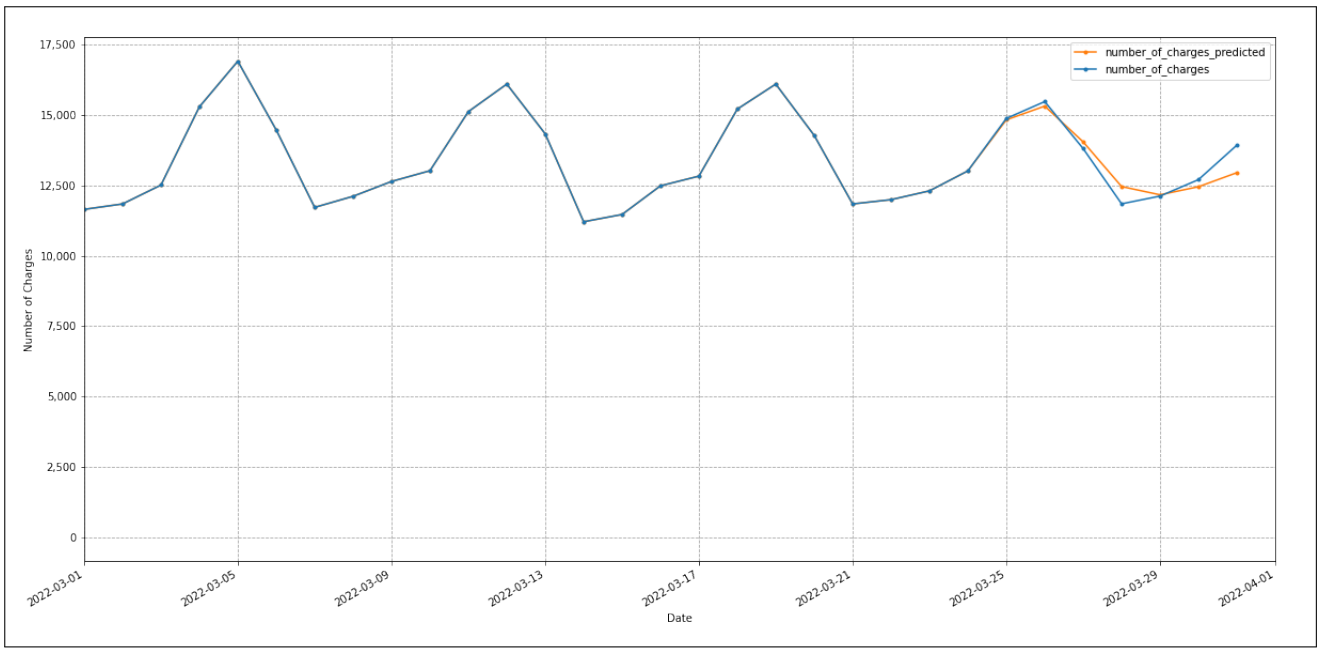


Figure 5.4: LSTM (Single Variable): Actual and Predicted Figures

tion from the data facilitated the creation of this normal curve. Opting for a normal distribution was justified by the anticipation that most residual values would center around zero, thinning out as they diverged from the mean. This corresponds to the expectation that the predicted values closely match the actual values, resulting in only a few values deviating significantly from the mean. Incorporating confidence intervals extended the graph to signify a range where, for instance, we could be 95% confident that the true residuals resided. This enhancement allowed us to visually capture the realm within which the majority of observed or simulated residuals should fall, providing a comprehensive representation of predictive uncertainty.

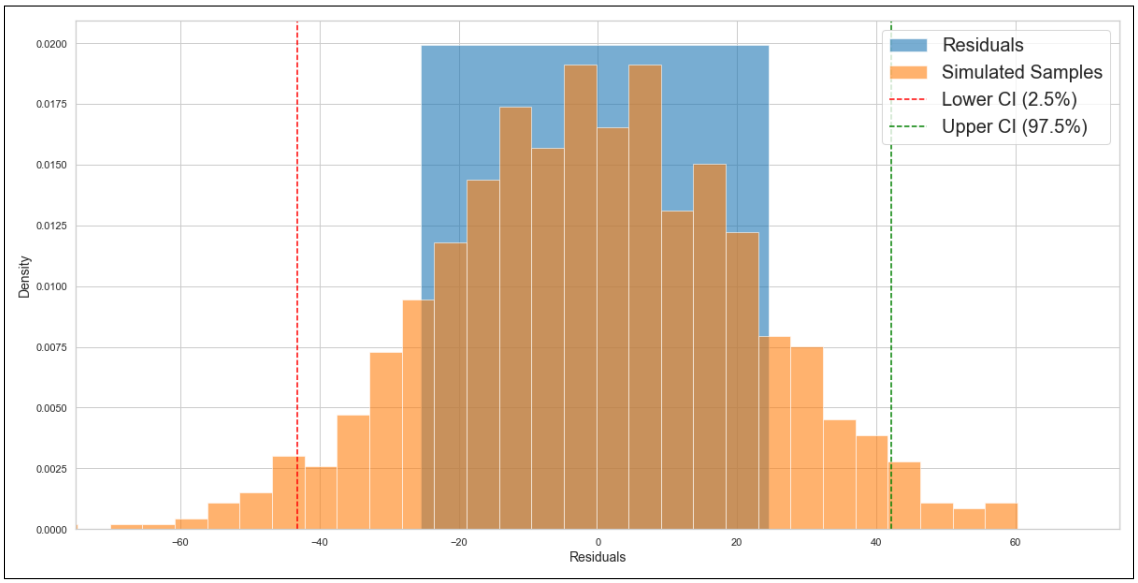


Figure 5.5: LSTM (Single Variable): Residual Analysis

Figure 5.5 presented a comprehensive analysis of the residuals, which represented the disparities between predicted and actual values for each data point. The blue histogram depicted the distribution of these residuals, indicating their frequency across various ranges. To comprehend the underlying distribution generating these residuals, simulated samples were generated from the fitted distribution and displayed as the orange histogram. These simulated samples

provided a visual representation of the potential value range that could emerge as deviations between predicted and actual outcomes.

The confidence intervals (CI), denoted by red and green dashed vertical lines, held significant importance. These intervals highlighted the range within which we could confidently assert that the true residuals lay, with a 95% confidence level. The red line marked the lower bound of the CI, signifying our reasonable confidence that the actual residuals were situated above this point. Conversely, the green line, representing the upper bound, indicated our confidence that the actual residuals were below this value.
Based on the results of the residual analysis conducted for the single variable LSTM model, pictured in Figure 5.5, several significant insights were garnered. The range of residuals was observed to span from -25 to 25, indicating that the model's predictions deviated from the actual values within this range. This deviation encompassed both negative and positive residuals, signifying instances where the model both underestimated and overestimated the actual outcomes. The confidence intervals (CIs) played a crucial role in shedding light on the uncertainty associated with the model's predictions. The lower CI of -40 indicated a high level of confidence that the true residuals fell above this threshold. This aligned with the presence of negative residuals, suggesting that the model occasionally overestimated the predicted values. Conversely, the upper CI of 40 also carried a similar degree of confidence, indicating that the actual residuals were below this value. This corresponded to the presence of positive residuals, indicating instances where the model's predictions fell short of the actual values.

5.2.7 Cross Validation Analysis

To estimate the skill of the model, we split the data set into five data sets as can be seen in Table 5.4 and further illustrated in Figure 5.6. Using these data splits, the process detailed in Algorithm 3 is then applied to each one to determine how the size and pattern of each data set affect the LSTM model and hence the predicted results.

Split No	Split Start Date	Split End Date	Number of Data Points	% of Total Data Points
01	2014-08-17	2016-06-19	528	20%
02	2014-08-17	2017-11-29	1,056	40%
03	2014-08-17	2019-05-11	1,584	60%
04	2014-08-17	2020-10-20	2,112	80%
05	2014-08-17	2022-03-31	2,639	100%

Table 5.4: Cross Validation: Split of Data Set

On each of the data splits, we optimised the hyperparameters using the same base hyperparameter selection in 5.1. The results per data split are shown in Table 5.5.

Split No	Testing Loss	Validation Loss	Activation	Batch Size	Drop Out	Epochs	Losses	Nodes	Optimizer	Patience
01	0.01645	0.12224	tanh	2	0.3	200	MSE	8	RMSprop	100
02	0.11925	0.13379	sigmoid	128	0.01	40	MSE	2	RMSprop	30
03	0.00088	0.11693	tanh	1	0.3	40	MSE	256	RMSprop	200
04	0.00096	0.03728	tanh	256	0.05	200	MSE	128	RMSprop	100
05	0.00058	0.00887	relu	32	0.2	200	MSE	256	RMSprop	20

Table 5.5: LSTM (Single Cross Validation): Lowest Validation Losses and Hyperparameters per Data Split

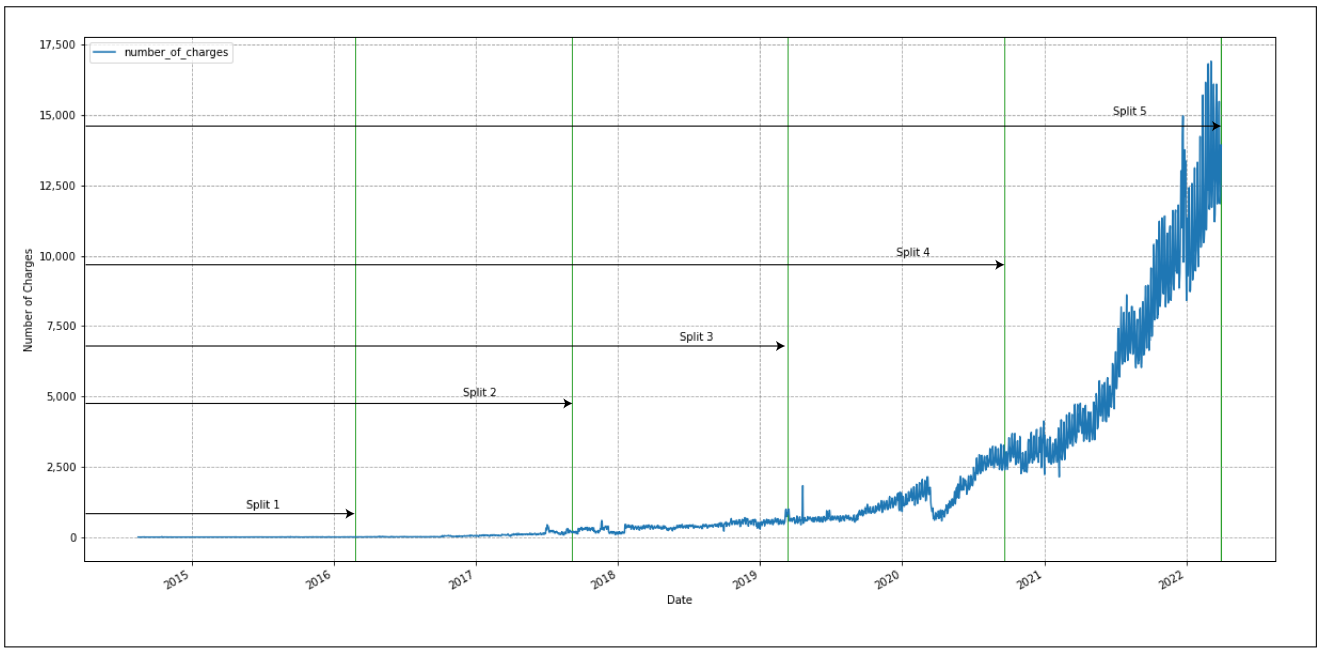


Figure 5.6: Cross Validation Data Split

Each LSTM model was trained on the data per split, with the respectively optimised hyperparameters, resulting in a seven-day prediction per split. Table 5.6 shows the actual and predicted total number of charges per prediction period of 7 days with the respective point difference comparison and the RMSE. The graphs in Figure 5.7 illustrate the daily actual and predicted values. In the figures, the actual number of charges (blue) were plotted on a daily basis over a period of 30 days, with the predicted values (orange) plotted on the last seven days of the defined period.

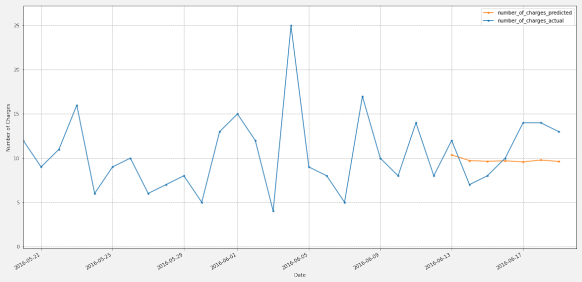
Split No	Date From	Date To	Actual Value	Predicted Value	Difference	% Difference	RMSE
01	2016-06-12	2016-06-19	78	68.41	-9.59	-12.29%	2.9677
02	2017-11-22	2017-11-29	2,234	2,557.97	323.97	14.50%	90.9312
03	2019-05-04	2019-05-11	4,644	4,637.06	-6.94	-0.15%	31.1418
04	2020-10-13	2020-10-20	21,479	21,982.67	503.67	2.34%	206.1429
05	2022-03-24	2022-03-31	94,754	94,211.69	-542.31	-0.57%	460.362

Table 5.6: LSTM (Single Variable Cross Validation): Actual vs Predicted Charges Results per Split

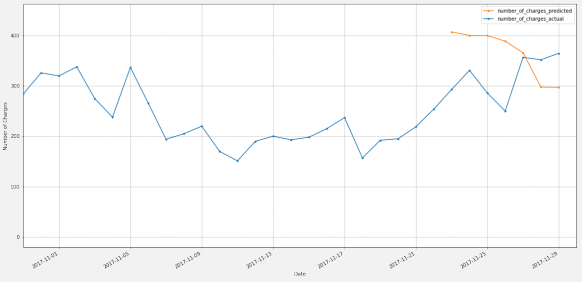
The RMSE is an indicator of the goodness of fit, and the lower the RMSE, the better the fit. However, this only applies to a comparison of several fits on the **same** dataset. For every split, we have a different set of data, and hence, the assumption that the results from Split 1 are better than those from any other Split as it has the lowest RMSE, is incorrect. To compare different RMSE values across different data splits, we need to normalise all RMSE values so that they are comparable as per Equation 5.2, where $\bar{y}$ is the average of the observed values over the time of investigation (seven days in our case). The normalised RMSE uses the average value of the *actual* number of charges and compares this to the RMSE to produce a value between 0 and 1. The closer the normalised RMSE is to 0, the better the result.

$$RMSE\_normalised = \frac{RMSE}{\bar{y}} \tag{5.2}$$

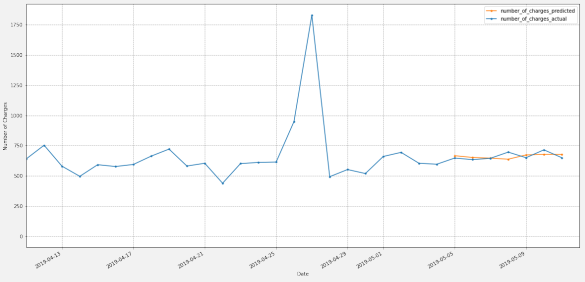
In Figure 5.8, we displayed the RMSE with the Normalised RMSE per data split. As explained



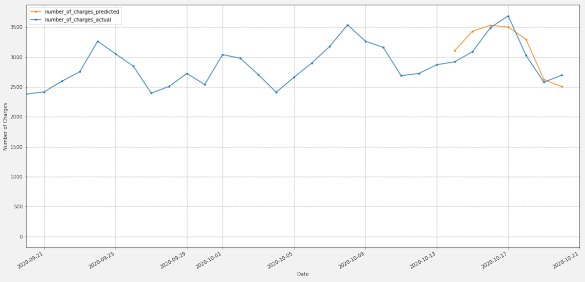
(a) Split 1 Comparison (RMSE: 2.9677)



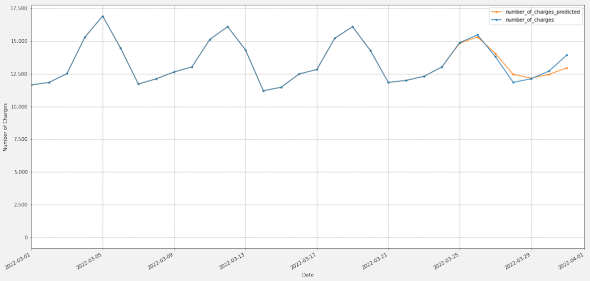
(b) Split 2 Comparison (RMSE: 90.9312)



(c) Split 3 Comparison (RMSE:31.1418)



(d) Split 4 Comparison (RMSE: 206.1429)



(e) Split 5 Comparison (RMSE: 460.362)

Figure 5.7: LSTM (Single Variable) Cross Validation: Actual vs Predicted Charges Results per Split

Split No	Minimum Charges	Maximum Charges	RMSE	Normalised RMSE
01	7	14	2.968	0.424
02	250	365	90.931	0.791
03	635	716	31.142	0.384
04	2,579	3,686	206.143	0.186
05	11,844	15,476	460.362	0.127

Table 5.7: LSTM: RMSE and Normalised RMSE Comparison

above, considering only the RMSE values would mean that the results in Split 1 gave the best results, whilst Split 5 gave the worst (RMSE 2.968 vs 460.362). However, the direct comparison of the RMSE values per split is not valid, as each RMSE is calculated on a different data set. Whilst each split contains seven data points, Split 1 has a daily average of 11 charges whilst Split 5 has a daily average of over thirteen thousand charges. Hence, a difference of 1 charge per day (when comparing the actual values to the predicted values) will affect Split 1 far more than it will affect Split 5. Hence, the RMSE values need to be normalised using Equation 5.2 to facilitate a comparison between the different data splits. When comparing the Normalised RMSE, we can now see that Split 5 produced the best-predicted charges when compared to the other splits. These results showed that despite the fact that the original data set contained different trends over different time periods (which includes the impact of COVID-19), the model performed better having more data points. It also indicated that the windowing period of fourteen days contained enough information to predict the number of charges for seven days within Split 5 as indicated by a low rate of error.

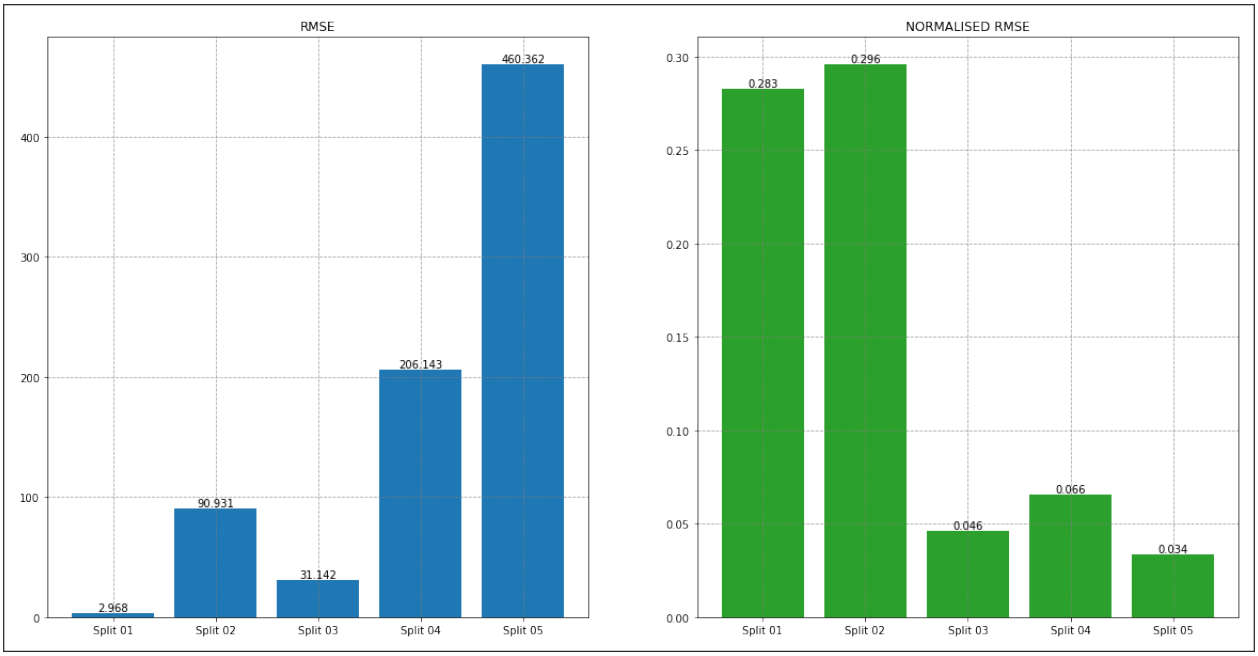


Figure 5.8: LSTM: RMSE and Normalised RMSE Visual Comparison

5.3 Gated Recurrent Units (GRU)

5.3.1 Model Data Preparation

Due to the similarity between the LSTM and the GRU, the configuration of the GRU took on the same modelling preparation method. Using the univariate input data, we derived the optimal hyperparameters to have a paradigmatic forecasting model. Algorithm 4 shows the high-level approach taken to predict and verify the number of charges.

Algorithm 4 GRU Process to predict the number of charges

Require: `raw_data`, `window_size`, `days_to_predict`
Ensure: data contains no null values

1: Create `input_data` by transforming and tapering the `raw_data` using `window_size` and `days_to_predict`
2: Split the tapered `input_data` into three data sets: training, validation, test
3: Using the `training_data` to train the GRU and a variety of hyperparameters, find the optimal hyper-parameters using the `validation_data` to verify the result
4: Train the model on the `training_data` using the optimal hyper-parameters
5: Predict the number of charges for the number of days defined by `days_to_predict`
6: Determine the rate of error between the predicted value and the `test_data` set

The data was also prepared as described in Section 3.7, where we transformed, encoded and tapered (or windowed) the univariate time series. Maintaining the variable input from the LSTM, we predicted the same number of charges (`days_to_predict = 7`) using a windowed data period of fourteen days (`window_size = 14`). The twenty-one days' worth of data was arranged again as in Figure 5.1 that produced an input data set (or training data set) of fourteen days, and an expected output data set (or prediction data set) of seven days. By creating these training and testing sets, the full data set containing 2,639 data points was decreased to only 2,625 data points that could be used in the prediction. This was due to the windowing function. The data was then split as described in Section 5.2.1 and graphically depicted in Figure 5.2

5.3.2 Hyperparameter Tuning

To determine the best combination of hyperparameters, the GRU model was trained using the data from `exp_training_data` and a random selection of the hyperparameters. The trained model then forecasts the number of charges for the next `days_to_predict` number of days. These forecasted values were then validated against the `exp_validation_data` using a selected model performance measure known to produce a validation loss. This process was then reiterated using the same exp_training_data but with a different combination of hyperparameters. The range of the hyperparameters is the same as those used in the LSTM model as shown in Table 5.1.

```
from keras.layers.core import Dense, Activation, Dropout
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, GRU

def gru_model(X_train, Y_train, X_test, Y_test, params):
    model = Sequential()
    model.add(GRU(params['nodes'], input_shape=(X_train.shape[1],
                                X_train.shape[2])))
    model.add(Dropout(params['dropout'],seed=config.SEED))
```

```
model.add(Dense(7))
model.add(Activation(params['activation']))
model.compile(loss=params['losses'], optimizer=params['optimizer'])

history = model.fit(X_train,
                    Y_train,
                    epochs=params['epochs'],
                    batch_size=params['batch_size'],
                    validation_data=(X_test, Y_test),
                    callbacks=[EarlyStopping(monitor='val_loss',
                               patience=params['patience'])],
                    verbose=0,
                    shuffle=False)
return history, model
```

Listing 5.2: GRU Model Definition

We chose to run the experiment one hundred times, which meant that [Talos 2019] randomly chose one hundred combinations (to perform one hundred experiments) of the selected values from the features in Table 5.1. Per experiment, Talos trained the GRU model defined in Listing 5.2 using a selected combination of features, and the `exp_training_data` as the input data set. The trained model then predicted the number of charges for the next seven days and these values were then compared to the actual number of charges that occurred in the `exp_validation_data`. The difference between these figures was measured by the selected loss function to produce a validation loss. The model and respective hyperparameters with the lowest validation loss were then saved to use as the final model to forecast future charges. As the input data was scaled, the validation loss values are also scaled between [0,1]. In experiment number 54, the lowest **validation loss of 0.01351** was found and corresponds to the first set of hyperparameters found in Table 5.8. The spread of the validation loss per experiment number is shown in Figure 5.9.

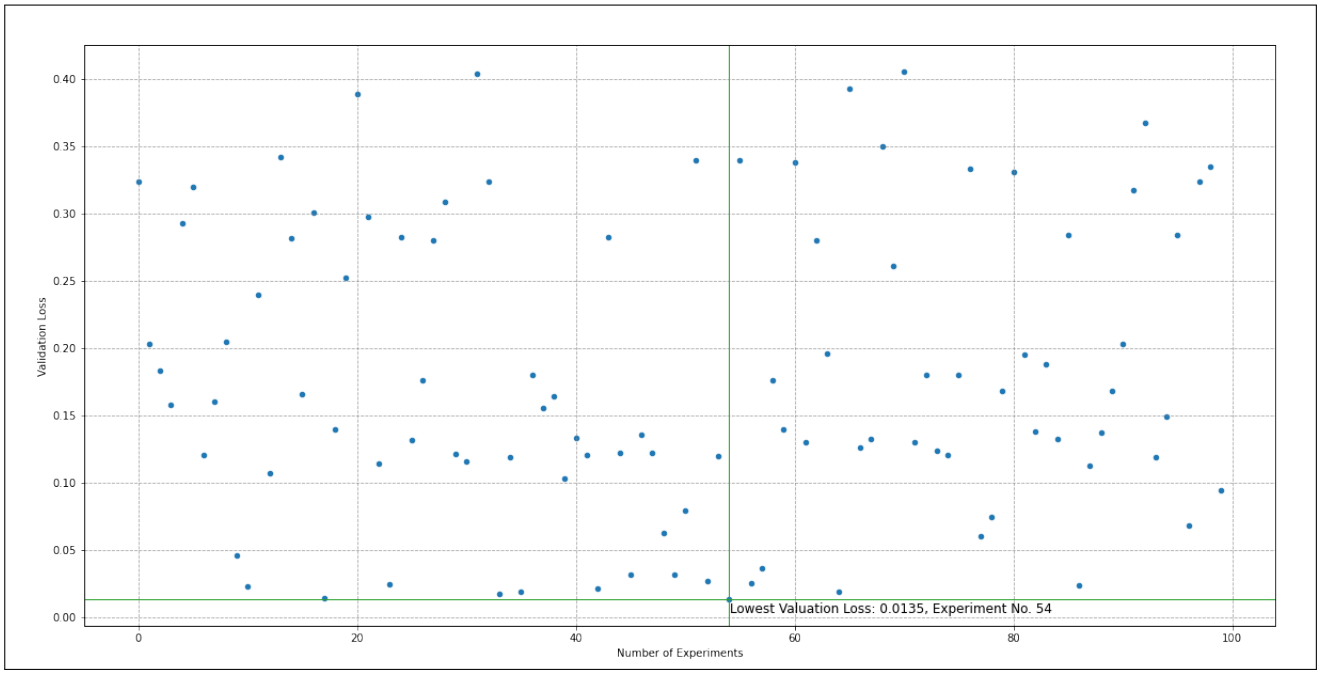


Figure 5.9: Validation Loss Result Spread for GRU Single Variable Experiments

No	Testing Loss	Validatio Loss	Activatio	Batch Size	Drop Out	Epochs	Losses	Nodes	Optimizer	Patience
01	0.00043	**0.01351**	tanh	64	0.01	80	MSE	1024	Nadam	20
02	0.00044	0.01436	sigmoid	4	0.075	100	MSE	256	Adam	50
03	0.00043	0.01803	tanh	4	0	100	MSE	8	Adam	50
04	0.00051	0.01891	sigmoid	16	0.5	200	MSE	256	Adam	200
05	0.00046	0.01909	tanh	2	0.01	100	MSE	16	Nadam	30

Table 5.8: GRU (Single Variable): Five Lowest Validation Losses and their Associated Hyperparameters

5.3.3 Model Training

The hyperparameters that resulted in the lowest validation loss were stored and used to train the model defined in Listing 5.2 can be seen in Table 5.8. The difference between the experimentation phase and the training phase was due to the data used when training. Instead of the `exp_training_data`, the GRU was trained using the full `training_data` data set. Once trained, the model was then expected to predict the number of charges for the next seven days.

5.3.4 Model Results

For the 7-day period ending on the 31st of March 2022, GRU predicted a total of 93,019.86 charges. This is 1,734.14 charges (or 1.83%) less than the actual amount of 94,754 charges that occurred during this week. The measure of difference between the actual and predicted values, the RMSE, was 531.225; which is low compared to the actual number of charges. The detailed day-by-day breakdown comparing the actual number of charges and predicted number of charges and the RMSE for the last 7 days in March 2022 were calculated and shown in Table 5.9. Figure 5.10 visually represents the actual number of charges for the last month of inspection, overlaid with the predicted number of charges for the last seven days.

Date	Actual Value	Predicted Value	Difference	% Difference
Day 01	14,878	14,890.03	12.03	0.08%
Day 02	15,476	14,783.15	-692.85	-4.48%
Day 03	13,806	14,076.08	270.08	1.96%
Day 04	11,844	12,367.57	523.57	4.42%
Day 05	12,119	11,588.94	-530.06	-4.37%
Day 06	12,703	12,049.54	-653.46	-5.14%
Day 07	13,928	13,264.54	-663.46	-4.76%
Total	**94,754**	**93,019.86**	**-1,734.14**	**-1.83%**
RMSE: 531.225				

Table 5.9: Number of Charges: Actual vs Predicted per Day

5.3.5 Residual Analysis with Confidence Intervals

After conducting the residual analysis for the single variable GRU model, as seen in Figure 5.11, several notable findings emerged. The analysis revealed that the residuals spanned from -5 to 40, illustrating the range within which the GRU model's predictions deviated from the actual values. This range encompassed both negative and positive residuals, indicating instances where the model underestimated and overestimated the actual outcomes, respectively. The confidence intervals (CIs) added an important layer of insight into the model's prediction uncertainties.

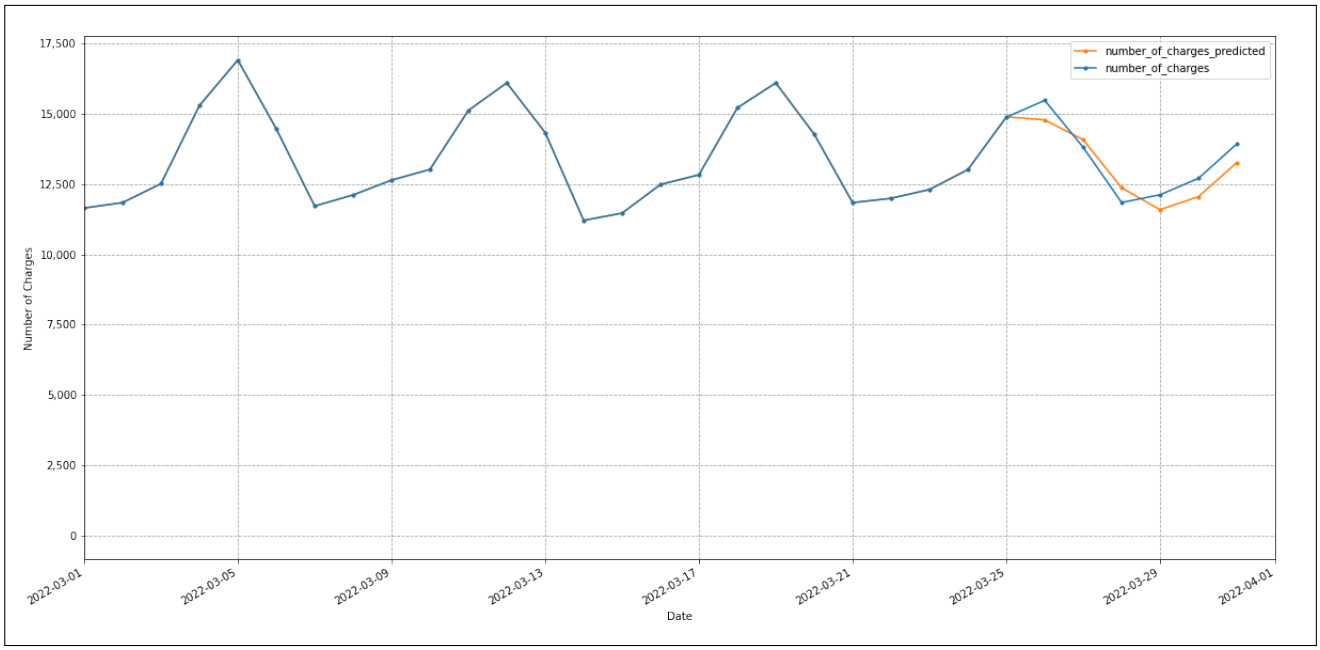


Figure 5.10: GRU (Single Variable): Actual and Predicted Figures

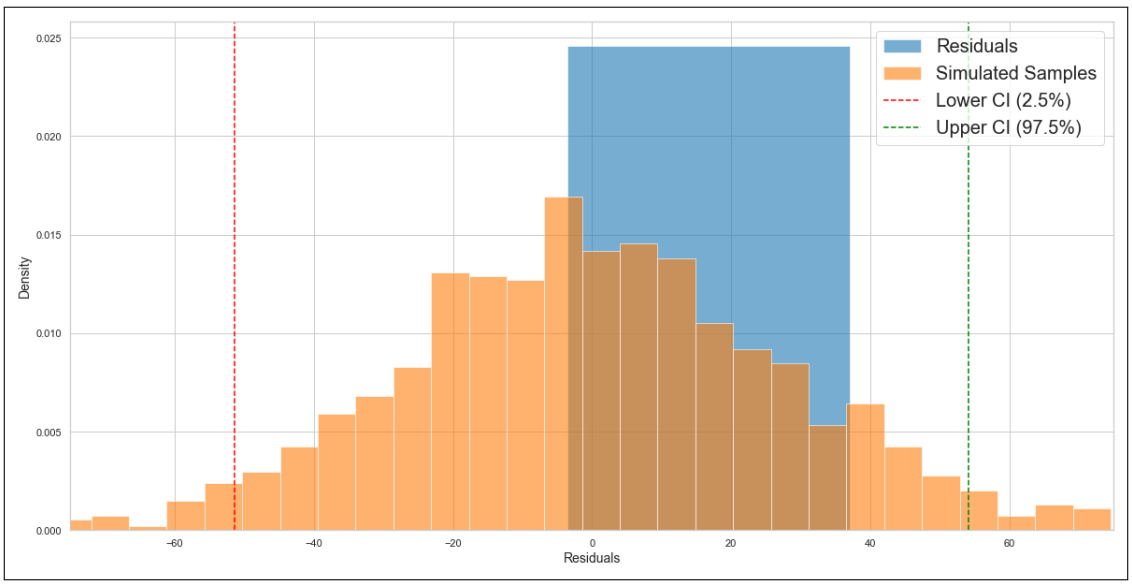


Figure 5.11: GRU (Single Variable): Residual Analysis

Split No	Testing Loss	Validation Loss	Activation	Batch Size	Drop Out	Epochs	Losses	Nodes	Optimizer	Patience
01	0.01514	0.11969	sigmoid	2	0.05	200	MSE	1024	RMSprop	20
02	0.16631	0.1293	sigmoid	512	0	20	MSE	8	Nadam	20
03	0.00099	0.11429	tanh	2	0.075	40	MSE	512	RMSprop	100
04	0.00096	0.04679	sigmoid	16	0.05	100	MSE	1024	Nadam	200
05	0.00043	0.01351	tanh	64	0.01	80	MSE	1024	Nadam	20

Table 5.10: GRU (Single Variable Cross Validation): Lowest Validation Losses and Hyperparameters per Data Split

The lower CI of -50 indicated a strong level of confidence that the actual residuals exceeded this value. This observation aligned with the presence of negative residuals, indicating instances where the model's predictions were higher than the actual values. Conversely, the upper CI of 30 also carried a significant degree of confidence, suggesting that the actual residuals were lower than this value. This correlated with the occurrence of positive residuals, showcasing instances where the model's predictions fell below the actual values.

5.3.6 Cross Validation Analysis

We split the data set into five data sets as can be seen in Table 5.4 and further illustrated in Figure 5.6. Using these data splits, the process detailed in Algorithm 4 was then applied to each one to determine how the size and pattern of each data set affect the GRU model and the predicted results.

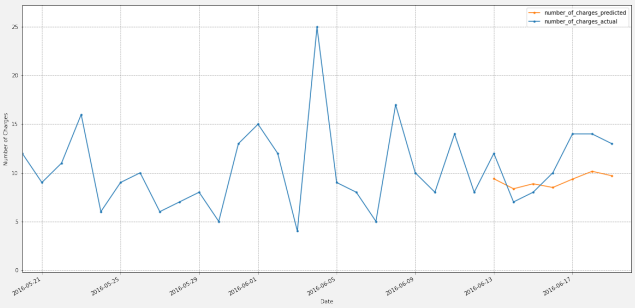
On each of the data splits, we optimised the hyperparameters using the same base hyperparameter selection in 5.1. The results per data split shown in Table 5.11

Each GRU model was trained on the data per split, with the respectively optimised hyperparameters, resulting in a seven-day prediction per split. Table 5.11 shows the actual and predicted total number of charges per prediction period of 7 days with the respective point difference comparison and the RMSE. The graphs in Figure 5.12 illustrate the daily actual and predicted values. In the figures, the actual number of charges (blue) were plotted on a daily basis over a period of 30 days, with the predicted values (orange) plotted on the last seven days of the defined period.

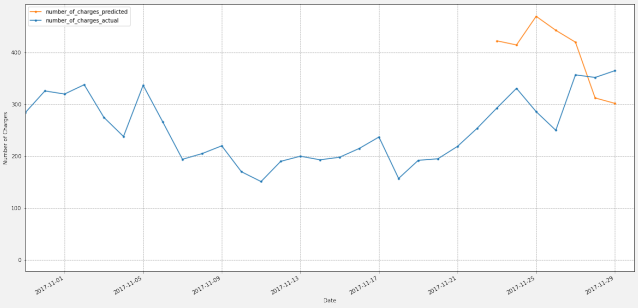
Split No	Date From	Date To	Actual Value	Predicted Value	Difference	% Difference	RMSE
01	2016-06-12	2016-06-19	78	64.31	-13.69	-17.55%	2.905
02	2017-11-22	2017-11-29	2,234	2,785.23	551.23	24.67%	122.184
03	2019-05-04	2019-05-11	4,644	4,608.74	-35.26	-0.76%	28.677
04	2020-10-13	2020-10-20	21,479	22,604.39	1,125.39	5.24%	219.744
05	2022-03-24	2022-03-31	94,754	93,019.86	-1,734.14	-1.83%	531.225

Table 5.11: GRU (Single Variable) Cross Validation: Actual vs Predicted Charges Results per Split

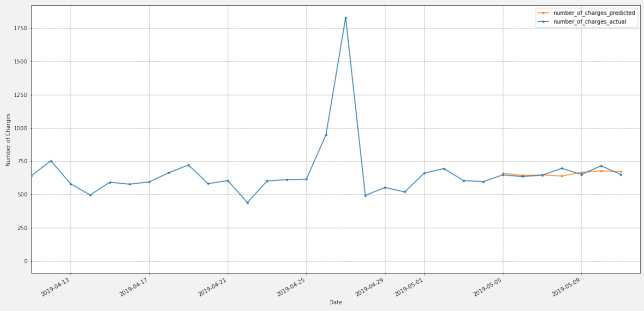
Using the RMSE as an indicator of the goodness of fit, we can compare the different RMSE values to determine which split has the lowest score. However, as the RMSE values were not all calculated on the **same** datasets, we need to calculate the normalised RMSE as per Equation 5.2.



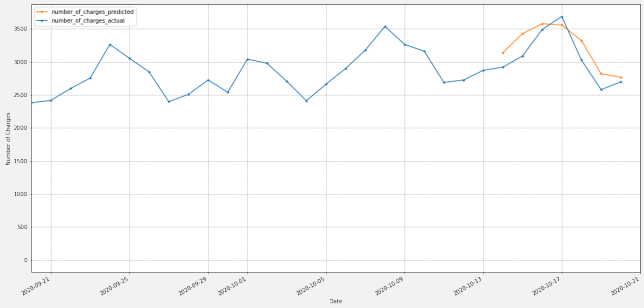
(a) Split 1 Comparison (RMSE: 2.905)



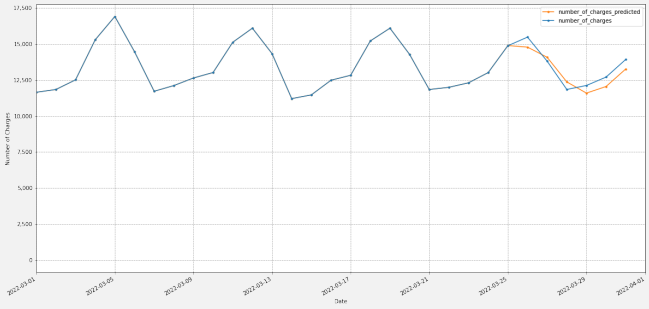
(b) Split 2 Comparison (RMSE: 122.184)



(c) Split 3 Comparison (RMSE:28.677)



(d) Split 4 Comparison (RMSE: 219.744)



(e) Split 5 Comparison (RMSE: 531.225)

Figure 5.12: GRU (Single Variable) Cross Validation: Actual vs Predicted Charges Results per Split

Split No	Minimum Charges	Maximum Charges	RMSE	Normalised RMSE
01	7	14	2.905	0.277
02	250	365	122.184	0.397
03	635	716	28.677	0.042
04	2,579	3,686	219.744	0.070
05	11,844	15,476	531.225	0.039

Table 5.12: GRU (Single Variable Cross Validation): RMSE and Normalised RMSE Comparison

To better understand why we cannot use the RMSE, consider the results of Split 1 vs Split 5. Here we see that Split 1 has a lower RMSE than Split 5 (RMSE 2.905 vs 531.225 respectively), and hence, we could use this to determine that Split 1's predicted values were better than those from Split 5. When comparing the Normalised RMSE, we can now see that Split 5 produced the best-predicted charges when compared to the other splits. These results showed that despite the fact that the original data set contained different trends over different time periods (which includes the impact of COVID-19), the model performed better having more data points. It also indicated that the windowing period of fourteen days contained enough information to predict the number of charges for seven days within Split 5 as indicated by a low rate of error.

Figure 5.13 visually displays the RMSE with the Normalised RMSE per data split for the GRU. Similar to the LSTM results, when only comparing the normalised RMSE, it can be seen that Split 5 produced the best-predicted charges when compared to the other splits.

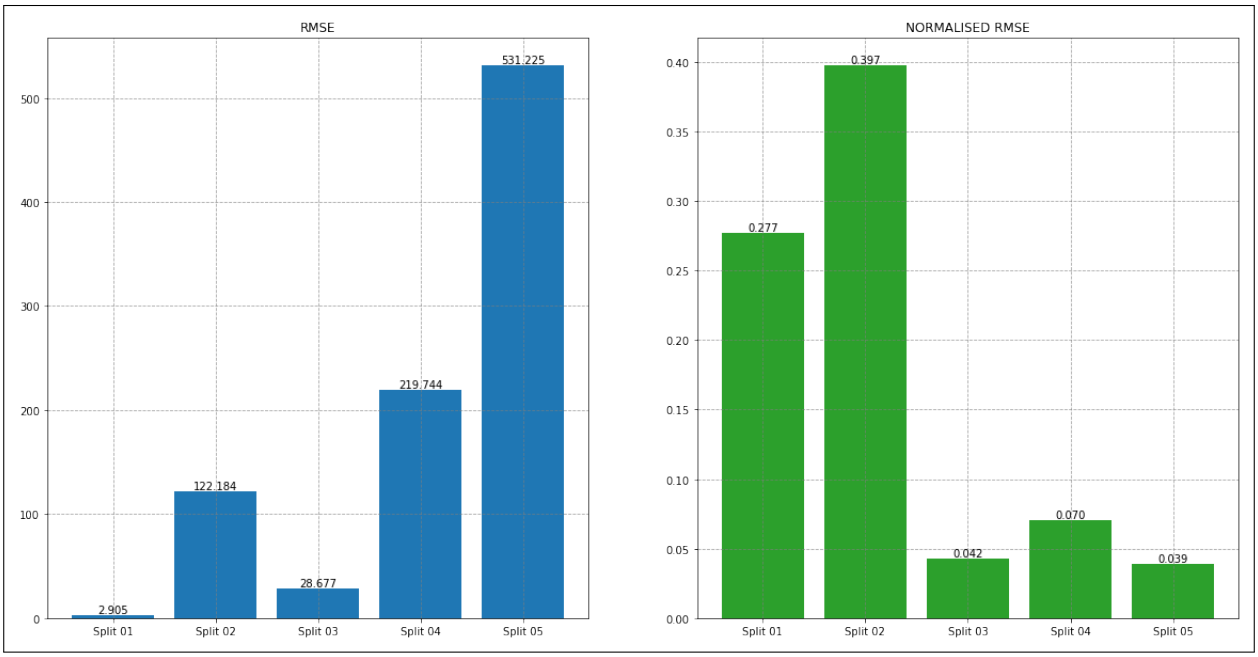


Figure 5.13: GRU (Single Variable Cross Validation): RMSE and Normalised RMSE Visual Comparison

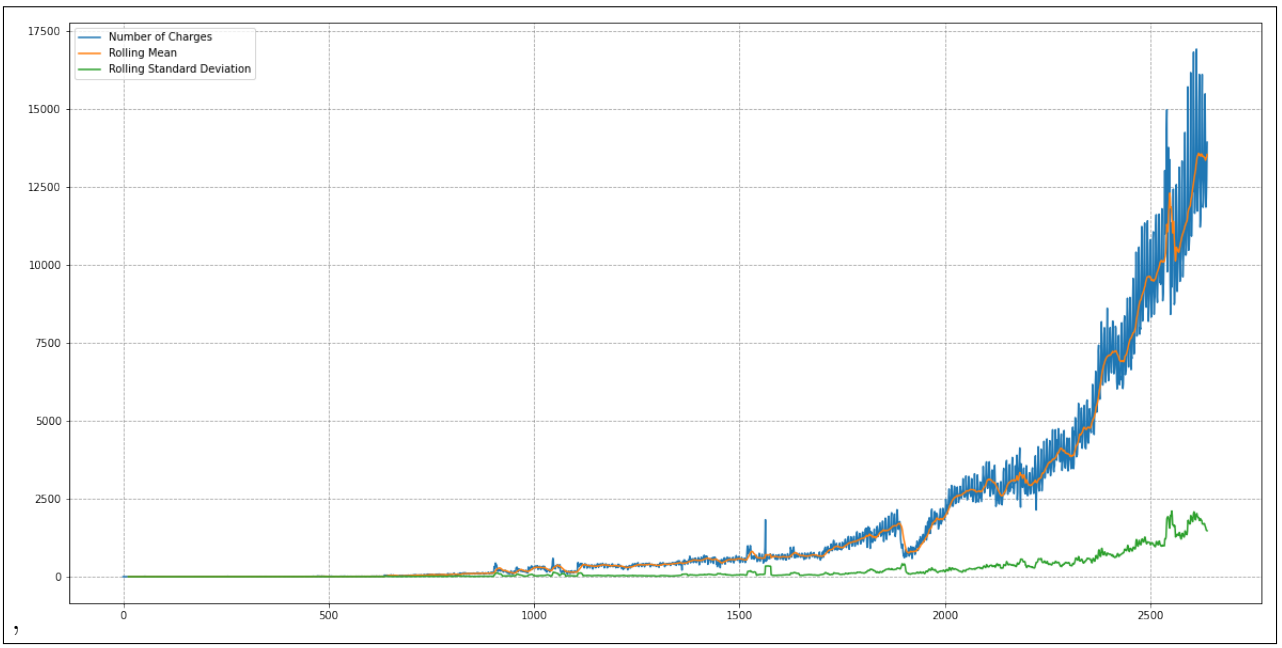


Figure 5.14: Number of Charges plotted against the Rolling Mean and Rolling Standard Deviation to understand Stationarity in the Original Time Series Data

5.4 Autoregressive Integrated Moving Average (ARIMA)

5.4.1 Stationarity Analysis

A stationary time series is one whose properties do not depend on the time at which the series is observed [Hyndman and Athanasopoulos 2018]. A stationary process has the property that the mean, and variance (or standard deviation) do not change over time. Considering the visual representation of the number of charges and its respectively plotted mean and standard deviation in Figure 5.14, the data is not stationary.
To further test for stationarity known as the Augmented Dickey-Fuller (ADF) test, which determines whether the change in the number of charges at a point can be explained by a lagged value (such as time t-1) and by a linear trend. As ADF is a statistical significance test, it was used to inform the degree to which a null hypothesis H_0 can be rejected or fail to be rejected. Here we defined the following hypothesis.

Null Hypothesis (H_0): The series of data (number of charges over time) is non-stationary
Alternate Hypothesis(H_1): If the null hypothesis is rejected, then the series of data (number of charges over time) is stationary.

Based on the ADF, we found that the test statistic was 6.079685, the p-value was 1.000000 and the critical value at a significance of 5% was -2.862647.

Since the test statistic is greater than the critical value, and the p-value is greater than 0.05, given a significance level of 5%, we cannot reject the null hypothesis (H_0). Hence, the series of data is non-stationary - confirming the visual analysis.

For us to transform the non-stationary charges data to stationary, we used a technique known as *differencing*, which computes the differences between consecutive observations. The ADF test calculated on the difference of the number of charges data showed a p-value = 1.792536E-19 which is less than 0.05 and hence, our original null hypothesis, which stated that the time series

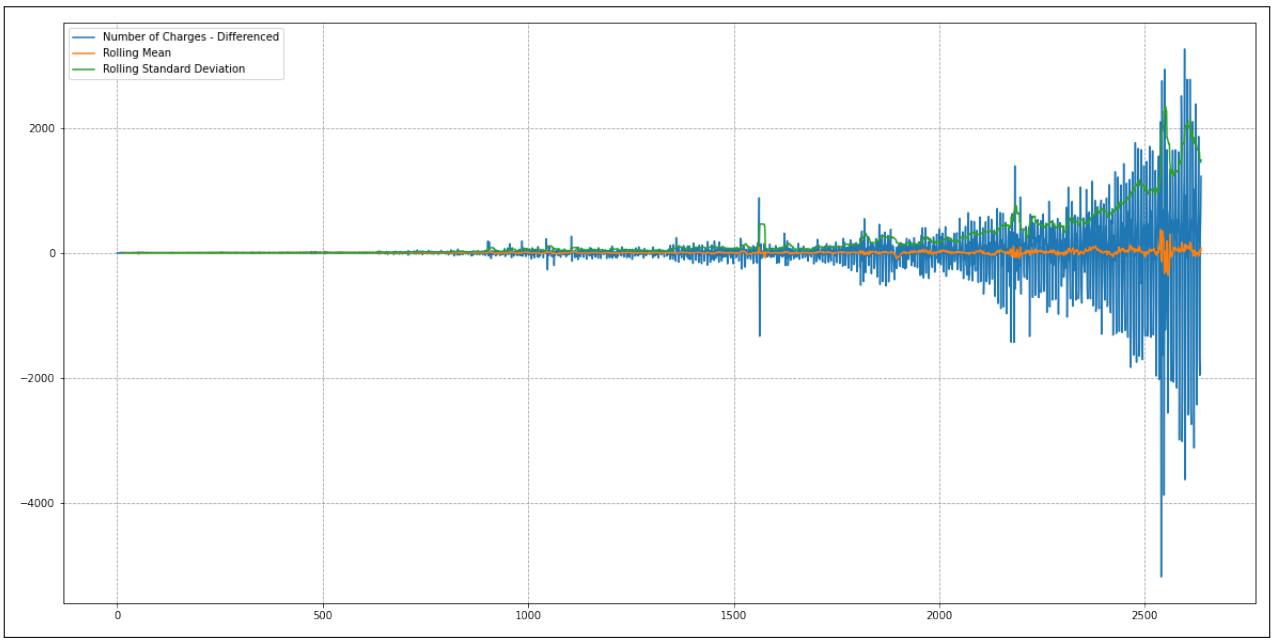


Figure 5.15: Number of Charges plotted against the Rolling Mean and Rolling Standard Deviation to understand Stationarity in the Differenced ($d = 1$) Time Series Data

is non-stationary, is false. Hence, differencing the number of charges transformed the series into a stationary data set to be used in the ARIMA model. As the ADF results concluded that the null hypothesis was proven false after the data set was differenced after one iteration, we can set the value of d to 1.

5.4.2 Model Data Preparation

The only transformation to the data for the ARIMA model occurred when the data needed to be made stationary and non-seasonal. Unlike the previous models, ARIMA does not require that the data be tapered. Hence, the unicolumnar stationary data can be split into the relevant training, validation, and test data sets, and used in the training and testing of the ARIMA Model. Algorithm 5 shows the high-level process to predict the number of charges and verify whether the prediction is valid.

Algorithm 5 ARIMA Process to predict the number of charges

Require: `raw_data`, `days_to_predict`
Ensure: *ABC*

1: Using statistical methods, determine if `raw_data` is stationary and non-seasonal
2: **if** `raw_data` is not stationary **then**
3: transform the data to a stationary data set to create `input_data`
4: **else**
5: copy `raw_data` as is to create `input_data`
6: **end if**
7: Split the `input_data` into three data sets: training, validation, test
8: Using the `training_data` to train the ARIMA model and a variety of hyperparameters, find the optimal hyperparameters using the `validation_data` to verify the result
9: Train the model on the `training_data` using the optimal hyperparameters
10: Predict the number of charges for a number of days defined by `days_to_predict`
11: Determine the rate of error between the predicted value and the `test_data` set

5.4.3 Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF)

Plotting the ACF and PACF of the data helped to understand the order of AR and MA for use in the model as an estimate for a good fit for the time-series data. We know from the analysis in Section 5.4.1, that the order of differencing is 1, and hence, we will focus our analyses on the first-order differencing graphs shown in Figure 5.16. The ACF and the PACF are known as lollipop plots. The original and second-order differencing graphs have been provided for reference.

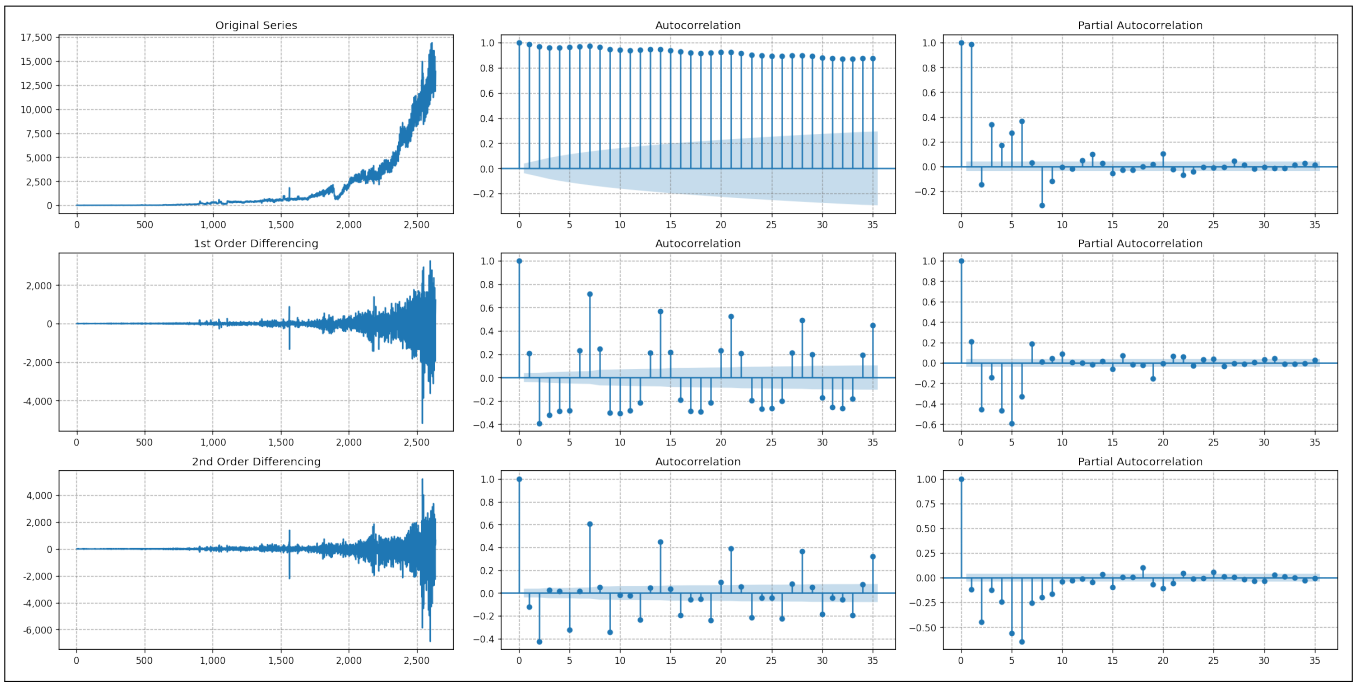


Figure 5.16: ACF and PACF Plots on the Number of Charges

The lag 0 of both the ACF and PACF represents the correlation of the time series with itself, resulting in a correlation of 1. The blue region in the ACF and PACF plots represents the 95% confidence interval, which is a measure of statistical significance. Thus, any value within the blue region indicates that it is statistically close to zero, while any value outside the blue region is statistically significant.

Simply stated, to determine the order of the model, we count the number of lollipops that are above or below the confidence interval before the next lollipop enters the blue area.

In the partial autocorrelation for the first order differencing, we observed (1) there were several non-zero significant autocorrelations prior to lag=7, (2) there was a high degree of autocorrelation between adjacent (lag = 7) in the PACF plot, and (3) after lag=7, there was a geometric decay in ACF plot. This led us to conclude that we have a significant lag at lag = 7 and hence, the autoregressive lag value of p is optimal at 7.

In the autocorrelation for the first order differencing, we observed (1) there were several significantly non-zero autocorrelations, (2) there was a high degree of autocorrelation between adjacent (lag = 1) in ACF plot, and (3) after lag = 1, there was a geometric decay in PACF plot. This led us to conclude that we have a significant lag at lag = 1 and hence, the moving average lag value of q is optimal at 1.

5.4.4 Hyperparameter Tuning and Model Training

The hyperparameters required for tuning in forecasting model ARIMA are the autoregressive parameter p, the differencing parameter d and the moving average q. In our statistical analyses in Sections 5.4.1 and 5.4.3, we calculated that the optimal values can be set at $d = 1$, $p = 7$ and $q = 1$.

In the implementation of the ARIMA model, we chose to use the *pmdarima.arima.auto_arima* function which conducted a search over the provided model constraints and returned the best model according to the AIC value. Unlike the models shown for LSTM and GRU, where the parameters needed to be individually selected, *auto_arima* works by conducting differencing tests to determine the order of differencing (which we have set to 1 given the analyses produced in Section 5.4.1), d, and then fitting models to the numeric values defined within ranges between $start\_p$ and max_p for parameter p, and between $start\_q$ and max_q for parameter q. Given that the data were daily, $m = 7$ was set as this indicates daily data. As the original data was not stationary, this variable was set to false, and $d = 1$ to indicate that the data needed to be differenced once as per our analysis. Similarly, the data was found to be non-seasonal, and hence the seasonal parameter was set to false. To find the best model, auto_arima optimized on $information\_criterion = 'aic'$. This implementation can be seen in Listing 5.3.

```
from pmdarima.arima import auto_arima

def arima_model(Y_train):
    model = auto_arima(y = Y_train,
                       start_p = 1,
                       start_q = 1,
                       start_P = 1,
                       max_p = 14,
                       max_q = 14,
                       m = 7,
                       seasonal = False,
                       stationary = False,
                       d = 1,
                       information_criterion = 'aic')
    return model
```

Listing 5.3: ARIMA Model Definition

The *auto_arima* function performed a stepwise search to minimise the AIC based on a defined range of values with results shown in Table 5.13. It can be seen that the lowest AIC value resulted from the model ARIMA(7,1,1) (or p=7, d=1 and q=1) which matches our statistical analyses, and they were considered as the optimal hyperparameters for the ARIMA model.

5.4.5 Model Results

For the 7-day period ending on the 31st of March 2022, ARIMA predicted a total of 94,731.24 charges. This is 2.76 charges (or 0.02%) less than the actual amount of 94,754 charges that occurred during this week. The measure of difference between the actual and predicted values, the RMSE, was 365.070; which is relatively low compared to the actual number of charges. The detailed day-by-day breakdown comparing the actual number of charges and predicted number of charges and the RMSE for the last 7 days in March 2022 were calculated and shown

No	Model Parameters (p,d,q)	AIC	Time (sec)
01	ARIMA(1,1,1)	38,878.48	1.26
02	ARIMA(0,1,0)	39,286.62	0.11
03	ARIMA(1,1,0)	39,176.96	0.21
04	ARIMA(0,1,1)	38,948.63	0.86
05	ARIMA(0,1,0)	39,284.99	0.05
06	ARIMA(2,1,1)	37,776.65	2.09
07	ARIMA(2,1,0)	38,572.36	0.22
08	ARIMA(3,1,1)	37,756.94	2.98
09	ARIMA(3,1,0)	38,522.14	0.45
10	ARIMA(4,1,1)	37,102.70	3.45
11	ARIMA(4,1,0)	37,895.74	0.43
12	ARIMA(5,1,1)	36,588.16	2.70
13	ARIMA(5,1,0)	36,759.20	0.61
14	ARIMA(6,1,1)	36,371.72	5.46
15	ARIMA(6,1,0)	36,444.88	1.10
16	ARIMA(7,1,1)	36,337.32	5.90
17	ARIMA(7,1,0)	36,343.30	1.27
18	ARIMA(8,1,1)	36,339.02	7.59
19	ARIMA(7,1,2)	36,341.00	7.18
20	ARIMA(6,1,2)	36,354.07	6.26
21	ARIMA(8,1,0)	36,342.81	1.69
22	ARIMA(8,1,2)	36,338.54	8.04
23	ARIMA(7,1,1)	36,344.96	2.06

Table 5.13: Hyperparameter results

in Table 5.14. Figure 5.17 visually represents the actual number of charges for the last month of inspection, overlaid with the predicted number of charges for the last seven days.

Date	Actual Value	Predicted Value	Difference	% Difference
Day 01	14,878	14,867.85	-10.15	-0.07%
Day 02	15,476	15,718.4	242.4	1.57%
Day 03	13,806	14,195.23	389.23	2.82%
Day 04	11,844	12,249.75	405.75	3.43%
Day 05	12,119	11,971.2	-147.8	-1.22%
Day 06	12,703	12,456.45	-246.55	-1.94%
Day 07	13,928	13,272.35	-655.65	-4.71%
Total	**94,754**	**94,731.24**	**-22.76**	**-0.02%**
RMSE: 365.070				

Table 5.14: Number of Charges: Actual vs Predicted per Day

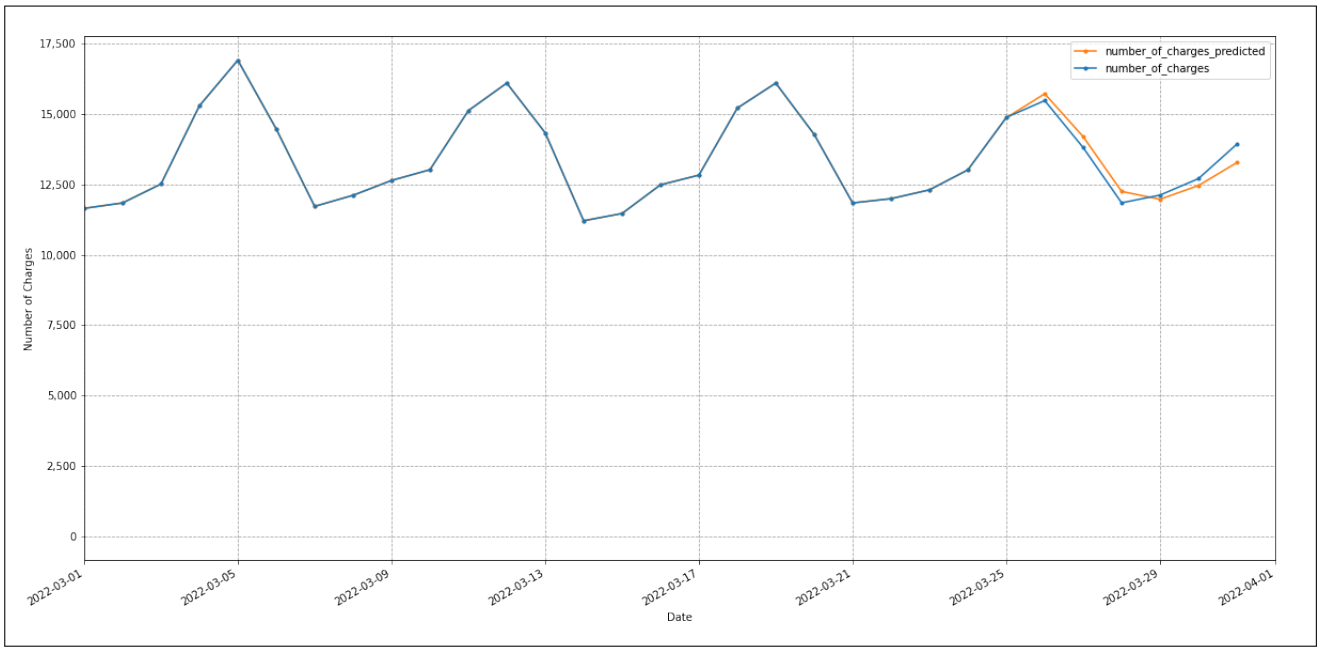


Figure 5.17: ARIMA: Single Variable Actual and Predicted Figures

5.4.6 Residual Analysis with Confidence Intervals

In the evaluation of the ARIMA forecasting model, an essential aspect of assessing the model's predictive accuracy involved the consideration of confidence intervals. A confidence interval (CI) is a statistical range within which the true value of the variable was expected to lie with a certain level of confidence. For instance, in our analysis, we adopted a 95% confidence interval, which indicated that we were 95% confident that the actual value would fall within the specified range of the predicted value.

Upon performing the ARIMA model evaluation, a notable observation emerged. All of the predictions generated by the ARIMA model fell comfortably within the boundaries of the 95% confidence interval. This result signified the model's consistency in capturing the inherent variability and uncertainty present in the dataset. The alignment of predictions within the specified confidence interval reinforced the model's capability to provide reliable forecasts while

acknowledging the potential variability in the true values. It further highlighted the robustness and precision of the ARIMA model in offering predictions that were both accurate and appropriately bounded by the level of confidence established. The results can be seen in Figure 5.18.

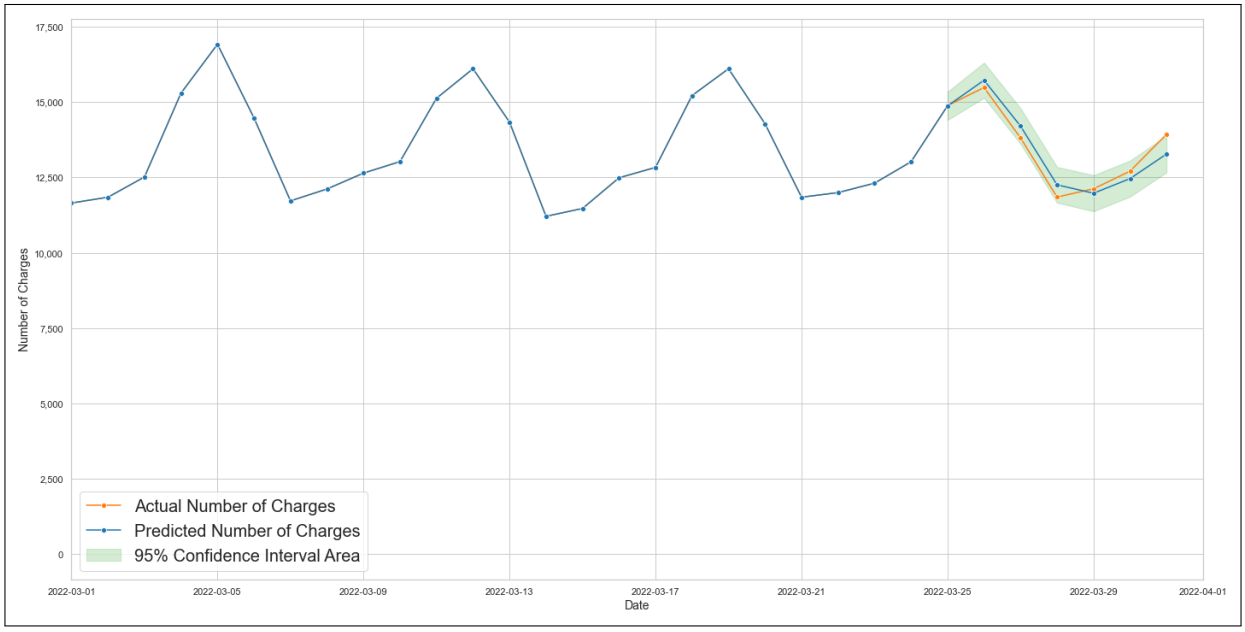


Figure 5.18: ARIMA: Single Variable Actual and Predicted Figures - 95% CI

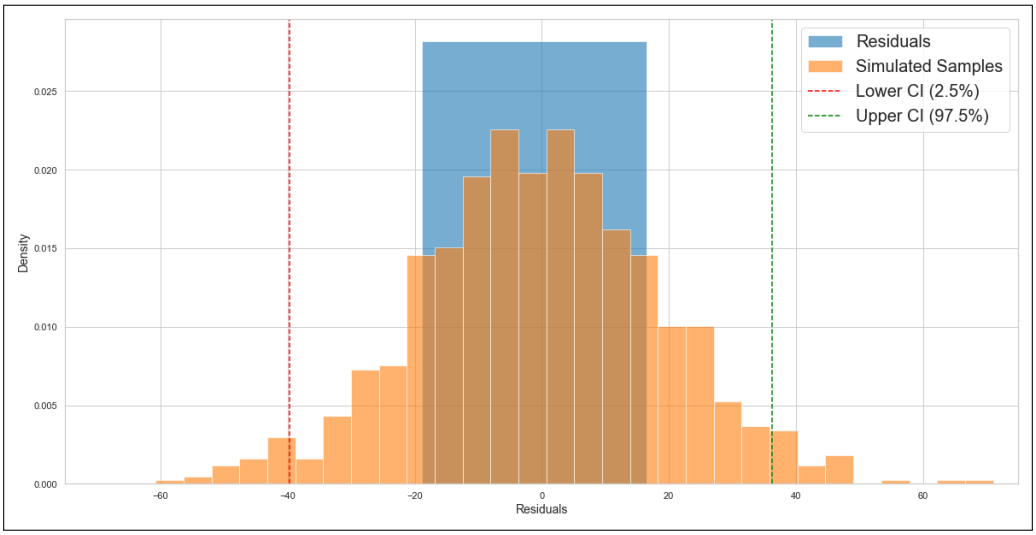


Figure 5.19: ARIMA: Residual Analysis

After conducting the residual analysis for the single-variable ARIMA model, as seen in Figure 5.19, several key insights were obtained. The analysis revealed that the residuals ranged from -20 to 20, representing the deviations between the ARIMA model's predicted values and the actual data points. The inclusion of confidence intervals (CIs) added an important dimension to the analysis. The lower CI of -40 indicated a high level of confidence that the actual residuals were greater than this value. This observation was in line with the presence of negative residuals, where the model's predictions exceeded the actual values. Conversely, the upper CI of 20 carried a significant degree of confidence, suggesting that the actual residuals were lower than this value. This correlation aligned with the occurrence of positive residuals, indicating instances where the model's predictions fell short of the actual values.

5.4.7 Cross Validation Analysis

Each ARIMA model was trained on the data per split, with the respectively optimised hyperparameters, resulting in a seven-day prediction per split. Table 5.15 shows the actual and predicted total number of charges per prediction period of 7 days with the respective point difference comparison and the RMSE. The graphs in Figure 5.20 illustrate the daily actual and predicted values. In the figures, the actual number of charges (blue) were plotted on a daily basis over a period of 30 days, with the predicted values (orange) plotted on the last seven days of the defined period.

Split No	Date From	Date To	Actual Value	Predicted Value	Difference	% Difference	RMSE
01	2016-06-12	2016-06-19	78	74.69	-3.31	4.42%	2.684
02	2017-11-22	2017-11-29	2,234	2,023.74	-210.26	-9.41%	49.154
03	2019-05-04	2019-05-11	4,644	4,600.44	-43.56	-0.94%	28.407
04	2020-10-13	2020-10-20	21,479	22,298.25	819.25	3.81%	181.567
05	2022-03-24	2022-03-31	94,754	94,731.24	-22.76	-0.02%	356.070

Table 5.15: Cross Validation: Actual vs Predicted Charges Results per Split

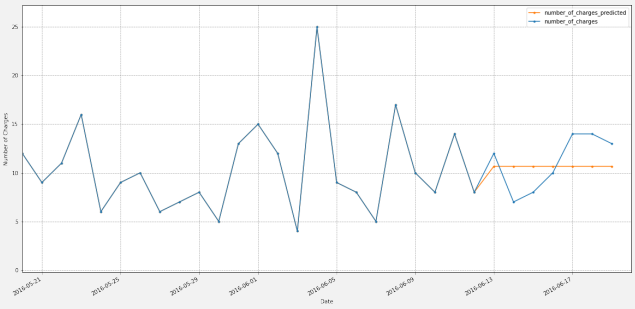
Using the RMSE an indicator of the goodness of fit, we can compare the different RMSE values to determine which split has the lowest score. However, as the RMSE values were not all calculated on the **same** datasets, we need to calculate the normalised RMSE as per Equation 5.2.

Split No	Minimum Charges	Maximum Charges	RMSE	Normalised RMSE
01	7	14	2.684	0.259
02	250	365	49.154	0.160
03	635	716	28.407	0.042
04	2,579	3,686	181.567	0.058
05	11,844	15,476	356.070	0.026

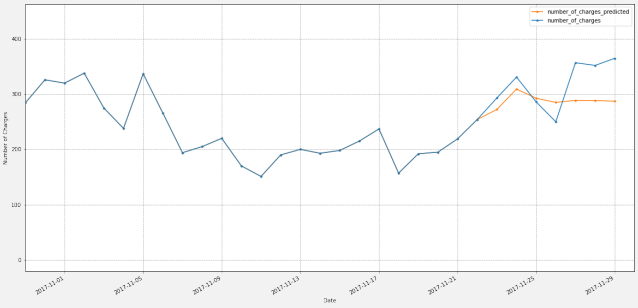
Table 5.16: ARIMA: RMSE and Normalised RMSE Comparison

To better understand why we cannot use the RMSE, consider the results of Split 1 vs Split 5. Here we see that Split 1 has a lower RMSE than Split 5 (RMSE 2.684 vs 356.070 respectively), and hence, we could use this to determine that Split 1's predicted values were better than those from Split 5. When comparing the Normalised RMSE, we can now see that Split 5 produced the best-predicted charges when compared to the other splits. These results showed that despite the fact that the original data set contained different trends over different time periods (which includes the impact of COVID-19), the model performed better having more data points. It also indicated that the windowing period of fourteen days contained enough information to predict the number of charges for seven days within Split 5 as indicated by a low rate of error.

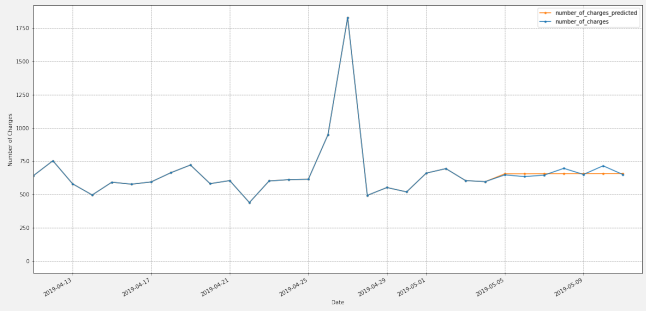
Figure 5.21 visually displays the RMSE with the Normalised RMSE per data split for the ARIMA. Only when the normalised RMSE values were compared, could it be seen that Split 5 produced the best-predicted charges when compared to the other splits.



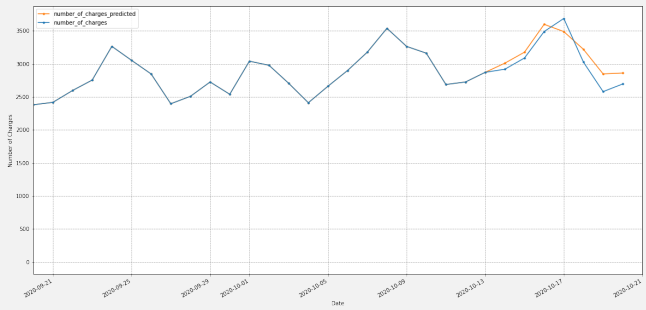
(a) Split 1 Comparison (RMSE: 2.684)



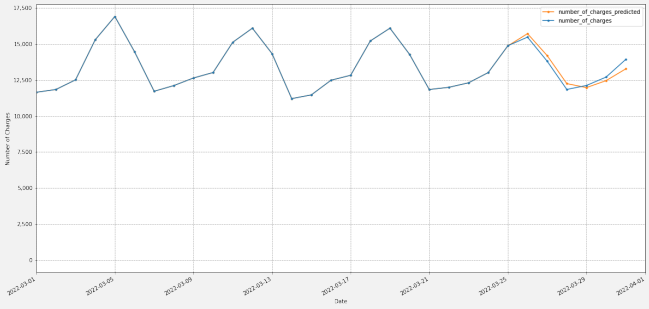
(b) Split 2 Comparison (RMSE: 49.154)



(c) Split 3 Comparison (RMSE:28.407)



(d) Split 4 Comparison (RMSE: 181.567)



(e) Split 5 Comparison (RMSE: 356.070)

Figure 5.20: ARIMA (Single Variable) Cross Validation: Actual vs Predicted Charges Results per Split

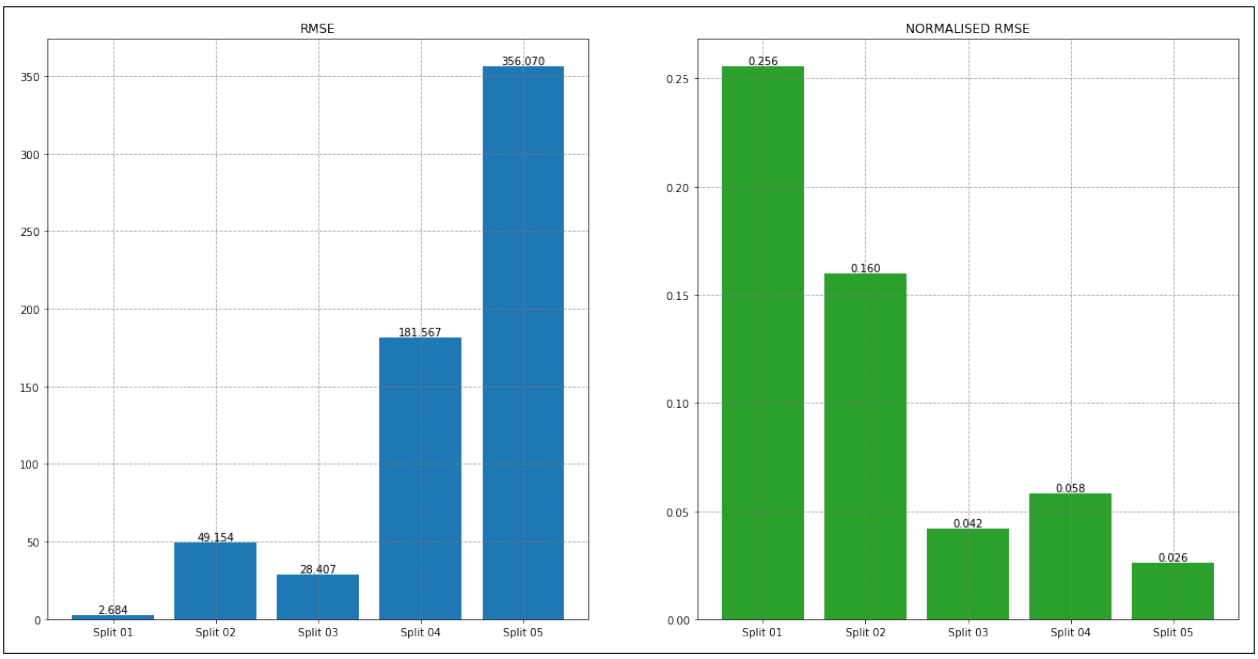


Figure 5.21: ARIMA: RMSE and Normalised RMSE Visual Comparison

5.5 Results Comparison for All Models

Once the experiments were completed and each of the models predicted the number of charges for the last 7 days in March 2022, we were then able to compare the results per model. To recapitulate, we considered only the univariate data for the number of charges completed per day from August 2014. The total data set consisted of 2,639 data points, where the last 7 data points were considered as the test data. Each model (LSTM, GRU, and ARIMA) used the training data to determine the best hyperparameters required to produce the best test result. Once these hyperparameters were defined, each model then predicted the number of charges for the next seven days which were then compared to the test data set.

5.5.1 Hyperparameter Optimization

Both LSTM and GRU models each had eight hyperparameters to optimise, whilst ARIMA had three. The time per model to find the optimised hyperparameters can be seen in Table 5.17. As expected, finding the optimal hyperparameters for the ARIMA model took less time than the other two models - and the LSTM and GRU times are similar given the similarity of the models and the required hyperparameters.

No	Model	Hyperparameters	Time (hh:mm:ss)
01	ARIMA	3	00:01:02
02	LSTM	8	02:03:51
03	GRU	8	02:34:54

Table 5.17: Time taken to find the optimal hyperparameters

5.5.2 Model Results

Once the hyperparameters were found, each model predicted the number of charges per day for a period of seven days. These were then plotted against the actual values of the charges over

those seven days, where the RMSE of each model result was calculated based on the actual charges, as shown in Table 5.18. On a total level, ARIMA **underpredicted** the actual values by **0.02%**, LSTM **underpredicted** by **0.57%**, GRU **underpredicted** by **1.83%**. When comparing the RMSE values, again, the ARIMA model had the least amount of errors in its predictions. Figure 5.22 visually represented the results per model against the actual number of charges.

Date	Actual Value	ARIMA Predicted Value	LSTM Predicted Value	GRU Predicted Value
Day 01	14,878	14,867.85	14,827.44	14,890.03
Day 02	15,476	15,718.4	15,315.16	14,783.15
Day 03	13,806	14,195.23	14,044.79	14,076.08
Day 04	11,844	12,249.75	12,457.83	12,367.57
Day 05	12,119	11,971.2	12,164.83	11,588.94
Day 06	12,703	12,456.45	12,451.31	12,049.54
Day 07	13,928	13,272.35	12,950.32	13,264.54
Total	**94,754**	**94,731.24**	**94,211.69**	**93,019.86**
% Diff		**-0.02%**	**-0.57%**	**-1.83%**
Difference		**-22.76**	**-542.31**	**-1,734.14**
RMSE		**356.070**	**460.362**	**531.225**

Table 5.18: The Actual Number of Charges vs The Predicted Number of Charges per Model

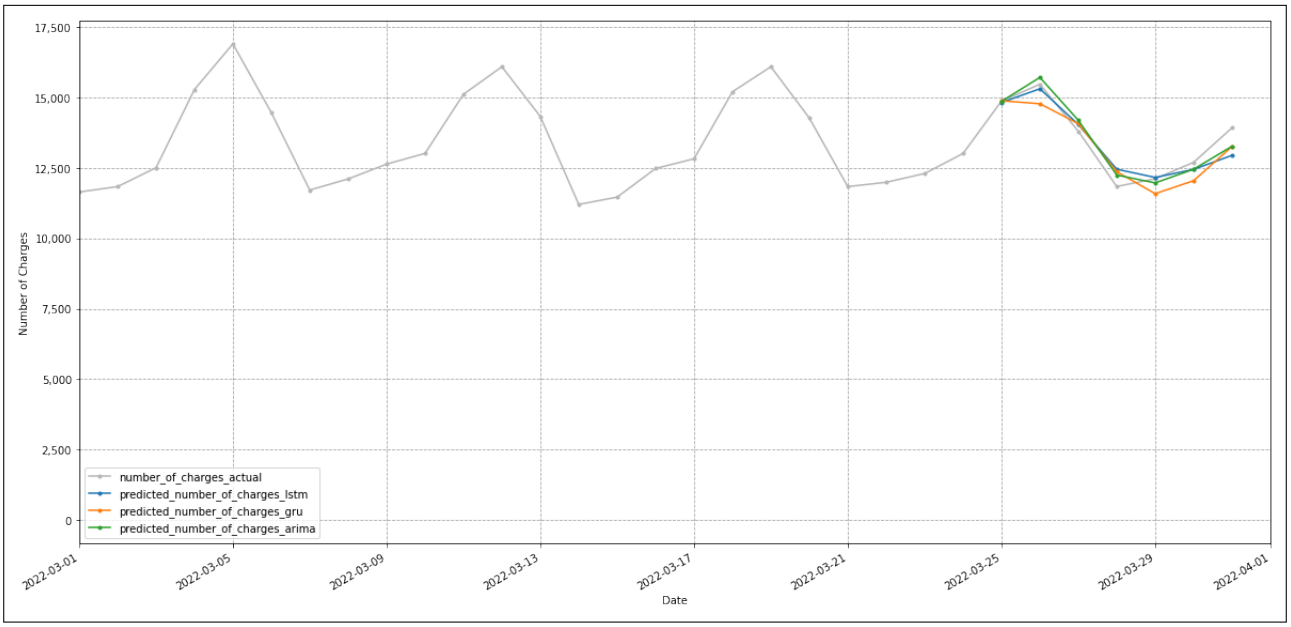


Figure 5.22: Day predictions for all models

5.5.3 Residual Analysis Comparison

The investigation into residuals and their associated confidence intervals for the ARIMA, LSTM, and GRU models provided distinct insights into their predictive performance. The ARIMA model showcased a narrow range of residuals and tight confidence intervals, signifying accurate and consistent predictions closely aligned with actual values. In contrast, the LSTM model's

residuals spanned from under -25 to just over 25, with confidence intervals around -40 and 40, indicating moderate alignment with occasional deviations. The GRU model, however, displayed a wider range of residuals, extending from -1 to around 60, with confidence intervals at -50 and 50-60. These findings collectively revealed the ARIMA model's superior precision, the LSTM model's moderate alignment, and the GRU model's higher variability in forecasting accuracy. Such analyses provided nuanced insights that enabled informed model selection for specific forecasting requirements.

5.5.4 Cross Validation Results

The RMSE results per data split can be seen in Table 5.19 and seen graphically in Figure 5.22. The RMSE allowed us to compare the difference between actual vs. expected per individual split, and it can be seen that for every split, the ARIMA model has the lowest RMSE in comparison to the LSTM and GRU models. To be able to compare the RMSE per model across all splits, the RMSE values were normalised and in every iteration, on average, the ARIMA model outperformed the other models. Furthermore, the normalised RMSE decreased as the data increased in size for all models. This can be seen graphically in Figure 5.24.

Splits	**ARIMA**	**LSTM**	**GRU**
RMSE			
Split 01	2.684	2.968	2.905
Split 02	49.154	90.931	122.184
Split 03	28.407	31.142	28.677
Split 04	181.567	206.143	219.744
Split 05	356.070	460.362	531.225
RMSE Normalised			
Split 01	0.259	0.283	0.277
Split 02	0.160	0.296	0.397
Split 03	0.042	0.046	0.042
Split 04	0.058	0.066	0.070
Split 05	0.026	0.034	0.039

Table 5.19: Comparison of the RMSE and Normalised RMSE per Model

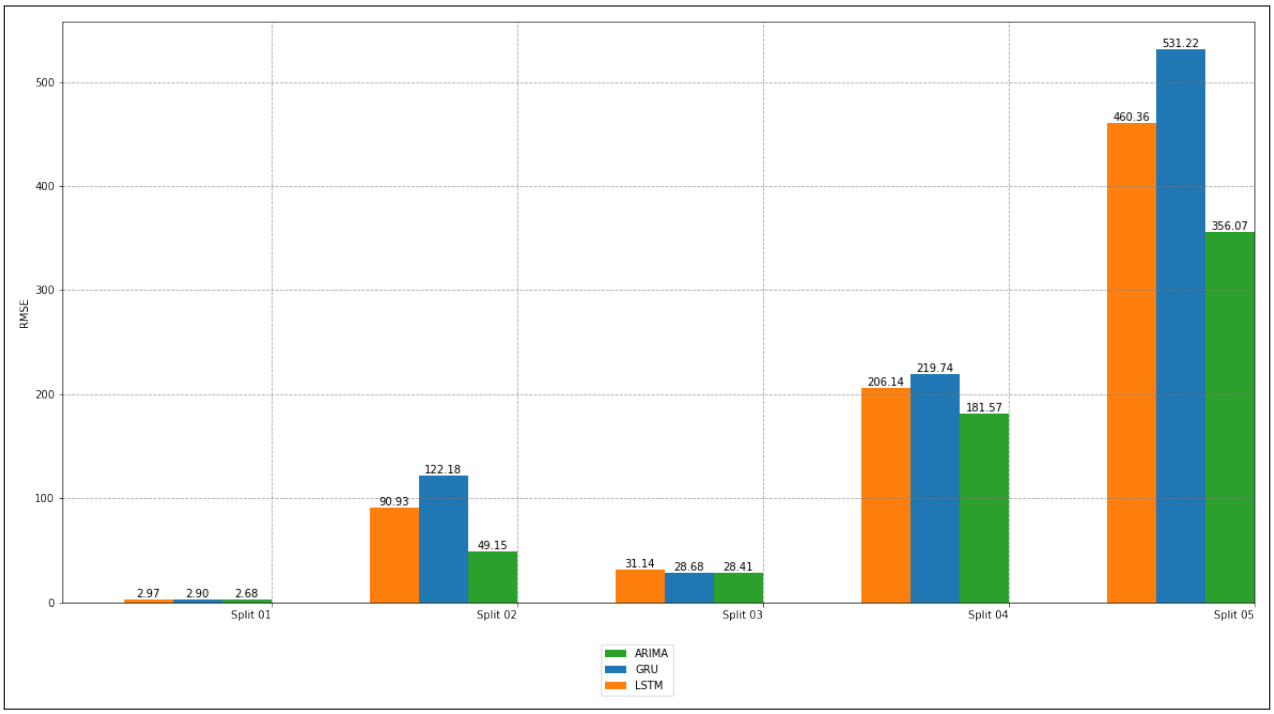


Figure 5.23: Normalised RMSE Comparison for All Models

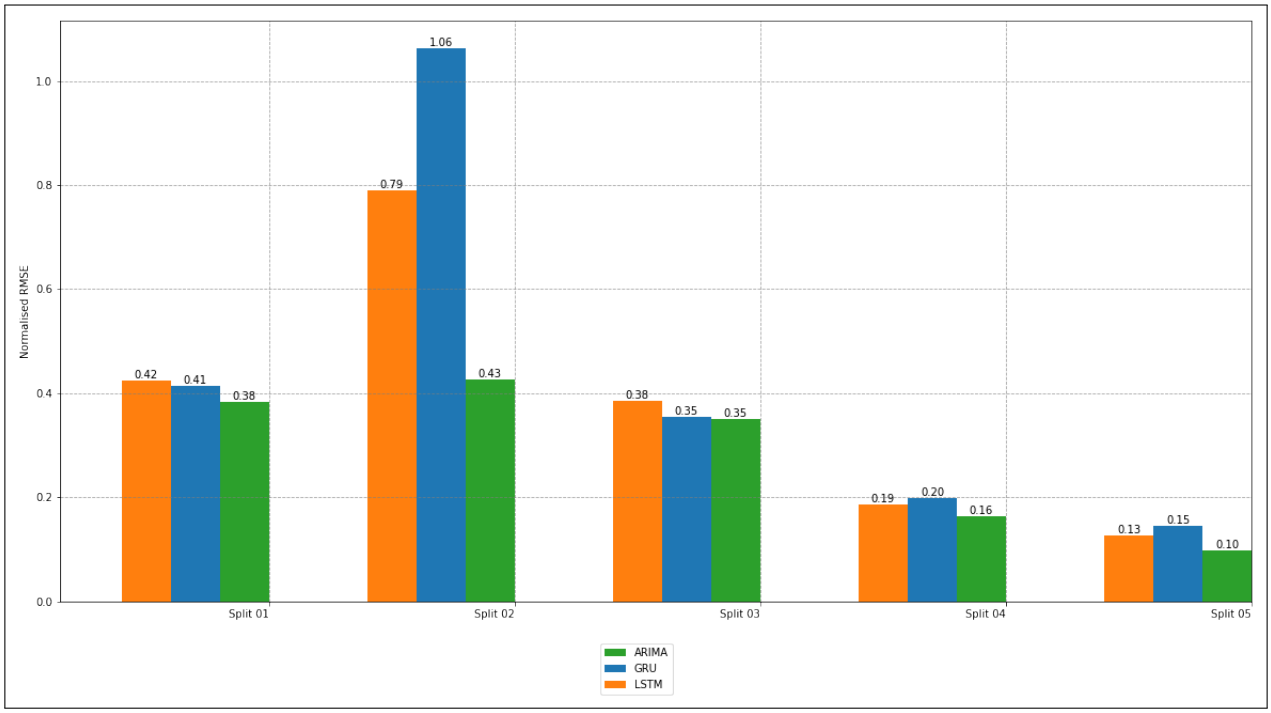


Figure 5.24: LSTM: RMSE and Normalised RMSE Visual Comparison

6 Multivariable Analysis (All Features)

6.1 Introduction

The multivariable analysis was based on observations of more than one feature at a time. The purpose of this multivariable analysis was to determine whether we could use the history of the number of charges, as well as additional influential features, to forecast the future number of charges. As with the single variable analysis, the success of the selected algorithms depended on how well they could determine patterns within the underlying data to forecast the number of charges. In Section 4.4, we discussed the potential features within the underlying data and described them in Table 4.1. To understand the relationship between the features, a Pearson coefficient correlation analysis was performed and this revealed that the number of charges had several highly correlated features (that, in themselves, are also highly correlated), and few features that had very little to no correlation. In the analyses presented, we considered two approaches to identify the features' impact on the prediction of the number of charges by (1) using all of the identified features, and (2) using only a selection of the features.

Furthermore, as ARIMA is a methodology best used for single variable analyses, we only implemented the multivariable analyses on LSTM and GRU.

6.2 Long Short-Term Memory (LSTM)

6.2.1 Model Data Preparation

The preparation of the multivariable LSTM model, in comparison to the single variable LSTM model, only differs in how the data was prepared. In this analysis, we kept the window size and number of days to predict the same as the single variable model so that we could compare the outputs. Hence, Algorithm 3 was also used as the high-level process that we followed to predict the number of charges.

After the data had been transformed and encoded, it needed to be tapered by using the variables `days_to_predict = 7` (that defined seven as the number of days forward we wished to predict) and `window_size = 14` (that defined fourteen as the window period). Assuming that $F1$ represented the number of charges, figure 6.1 visually represents how 21 days of transformed data $F1_{t-13}, F1_{t-12}, ..., F1_t$ was arranged to produce the input data (or training data) of fourteen days $F1_{t+1}, F1_{t+2}, ..., F1_{t+7}$, and the expected output (test data) of seven days. In addition to the training data, we also included other factors represented by $F2, F3, ...Fn$ where we tapered these as per the factor $F1$. This tapered data now represented the `input_data`.
To create the train, validation, and test data, the process defined in Figure 5.2 was used. As the multivariable data set had only increased in the number of columns due to the extra added features, we could compare the rows to those in the single dataset as those remained the same, i.e. the daily number of data points had remained the same.

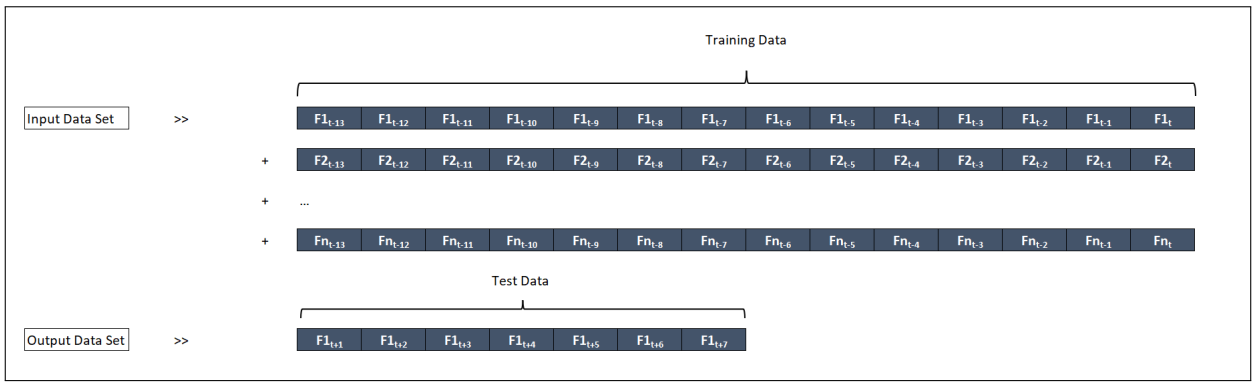


Figure 6.1: Actual Windowing Function for Multivariable Data

6.2.2 Scaling the Data

As per the single variable model, we scaled the different numeric variables using Equation 5.1 which ensured that all values were mapped to a value between `[0,1]`. The scaled data set was then used to train, validate, and test the model, producing predicted values that were scaled. To better understand the results in the context of the original data set, these predicted values were unscaled using the same initial scaler that was applied to the input.

6.2.3 Hyperparameter Tuning

With the data sets created, we needed to determine the best combination of hyperparameters that can be used to train the LSTM. With reference to Figure 5.2, Process Step "II: Determine Optimal Hyper-Parameter", the data from `exp_training_data` was trained in combination with various hyperparameters and then required to forecast the number of charges for the next seven days. These forecasted values were then validated against `exp_validation_data` using the selected model performance measurement to produce a validation loss. This process was repeated fifty more times using the same dataset, but using a different combination of hyperparameters each time to produce fifty different validation loss values. At the end of this process (also known as the experimentation phase), the validation loss values were compared and the hyperparameters of the lowest validation loss were stored as the optimal hyperparameter set that we will use in the final forecasting of the number of charges. The selection of hyperparameters is as per Table 5.1, and once again, [Talos 2019] was used to conduct the experimentation.

It is important to note that due to the increased data size, the experiments would have a longer run-time. Hence, to complete the experimentation in a reasonable time and also ensure that we get to the best hyperparameters possible, we ran fifty experiments. As the input data was scaled, the validation loss values were also scaled between `[0,1]`.

In experiment number 36, the lowest validation loss of 0.03950 was found and corresponds to the first set of hyperparameters found in Table 6.1. A visual representation of each scaled validation loss per experiment performed can be seen in Figure 6.2

6.2.4 Model Training

As shown in Table 5.1, the hyperparameters that resulted in the lowest validation loss were stored and used to train the model defined in Listing 5.1. The difference between the experimentation phase and the training phase was due to the data used when training. Instead of the `exp_training_data`, the LSTM was trained using the full `training_data` data set. Once

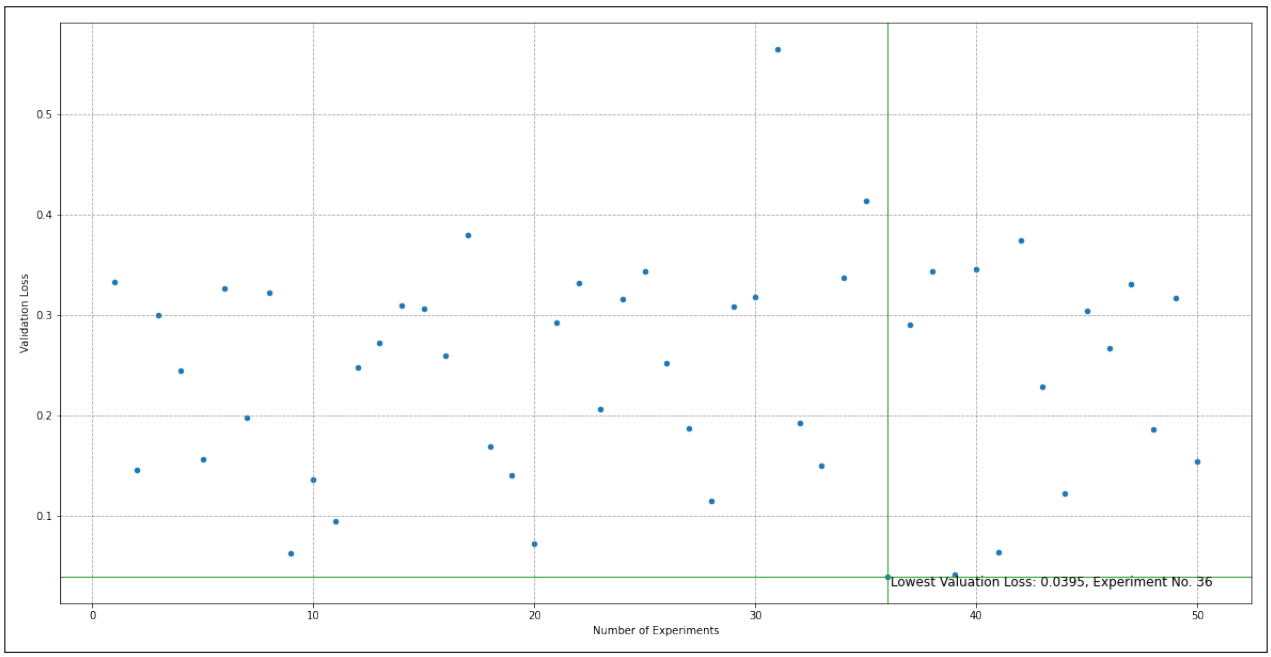


Figure 6.2: Validation Loss Result Spread for LSTM Multivariable (All Features) Experiments

trained, the model was then expected to predict the number of charges for the next seven days.

The hyperparameters, that resulted in the lowest validation loss, were stored and used to train the model defined in Listing 5.1; they can be seen in Table 6.1. The difference between the experimentation phase and the training phase was due to the data used when training. Instead of the `exp_training_data`, the LSTM was trained using the full `training_data` data set. Once trained, the model was then expected to predict the number of charges for the next seven days.

No	Testing Loss	Validatio Loss	Activatio	Batch Size	Drop Out	Epochs	Losses	Nodes	Optimizer	Pat-ience
01	0.00108	**0.03950**	tanh	512	0.05	20	MSE	256	Adam	20
02	0.00044	0.04177	relu	16	0.01	40	MSE	128	Nadam	100
03	0.00079	0.06304	relu	512	0.05	40	MSE	64	Adam	50
04	0.00150	0.06388	relu	1024	0.5	40	MSE	512	Nadam	100
05	0.00790	0.07221	relu	64	0.2	40	MSE	512	Nadam	50

Table 6.1: LSTM (Multivariable - All Features): Five Lowest Validation Losses and their Associated Hyperparameters

6.2.5 Model Results

For the 7-day period ending on the 31st of March 2022, LSTM predicted a total of 96,400.90 charges. This is 1,646.90 charges (or 1.71%) more than the actual amount of 94,754 charges that occurred during this week. The measure of difference between the actual and predicted values, the RMSE, was 627.454; which is low compared to the actual number of charges. The detailed day-by-day breakdown comparing the actual number of charges and predicted number of charges and the RMSE for the last 7 days in March 2022 were calculated and shown in Table 6.2. Figure 6.3 visually represents the actual number of charges for the last month of inspection, overlaid with the predicted number of charges for the last seven days.

Date	Actual Value	Predicted Value	Difference	% Difference
Day 01	14,878	15,135.87	257.87	1.73%
Day 02	15,476	14,652.85	-823.15	-5.32%
Day 03	13,806	14,223.95	417.95	3.03%
Day 04	11,844	13,087.43	1,243.43	10.50%
Day 05	12,119	12,173.08	54.08	0.44%
Day 06	12,703	13,238.34	535.34	4.21%
Day 07	13,928	13,889.37	-38.63	-0.28%
Total	**94,754**	**96,400.90**	**1,646.90**	**1.71%**
RMSE: 627.454				

Table 6.2: Multivariable LSTM Number of Charges: Actual vs Predicted per Day

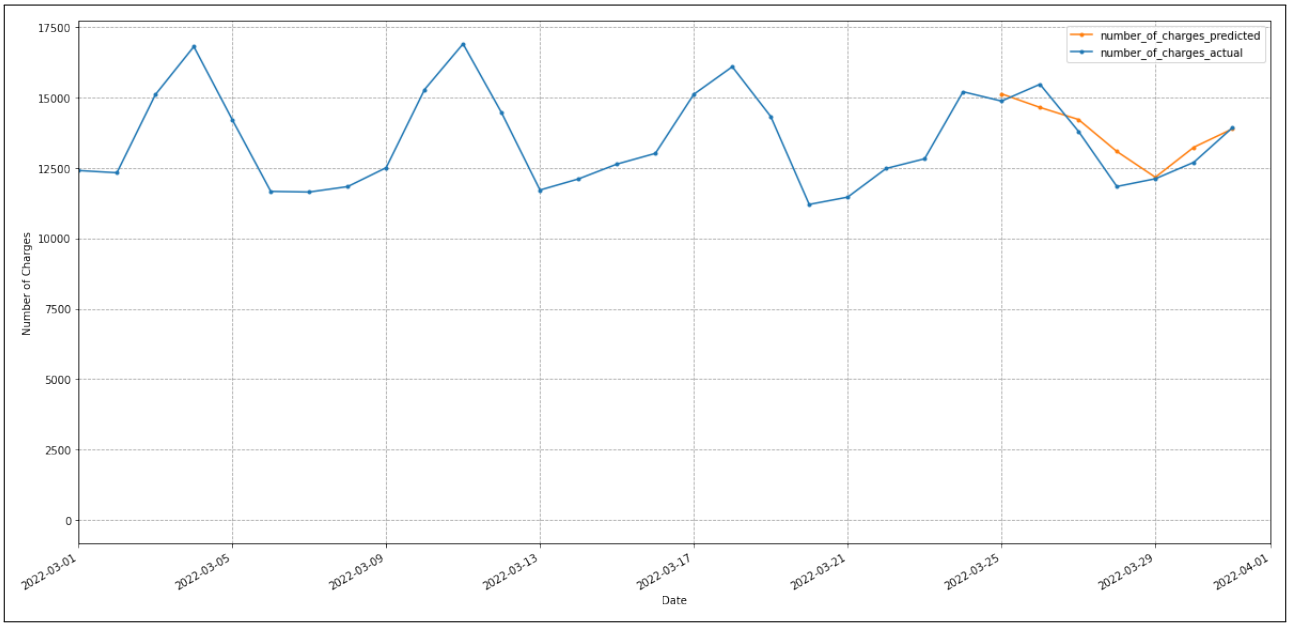


Figure 6.3: LSTM (Multivariable - All Features): Actual and Predicted Figures

Split No	Date From	Date To	Actual Value	Predicted Value	Difference	% Difference	RMSE
01	2016-06-12	2016-06-19	78	67.26	-10.74	-13.77%	3.400
02	2017-11-22	2017-11-29	2,234	2,440.83	206.83	9.26%	84.237
03	2019-05-04	2019-05-11	4,644	4,481.73	-162.27	-3.49%	39.317
04	2020-10-13	2020-10-20	21,479	22,699.49	1,220.49	5.68%	243.153
05	2022-03-24	2022-03-31	94,754	96,400.9	1,646.9	1.74%	627.454

Table 6.3: LSTM Cross Validation: Actual vs Predicted Charges Results per Split

6.2.6 Residual Analysis with Confidence Intervals

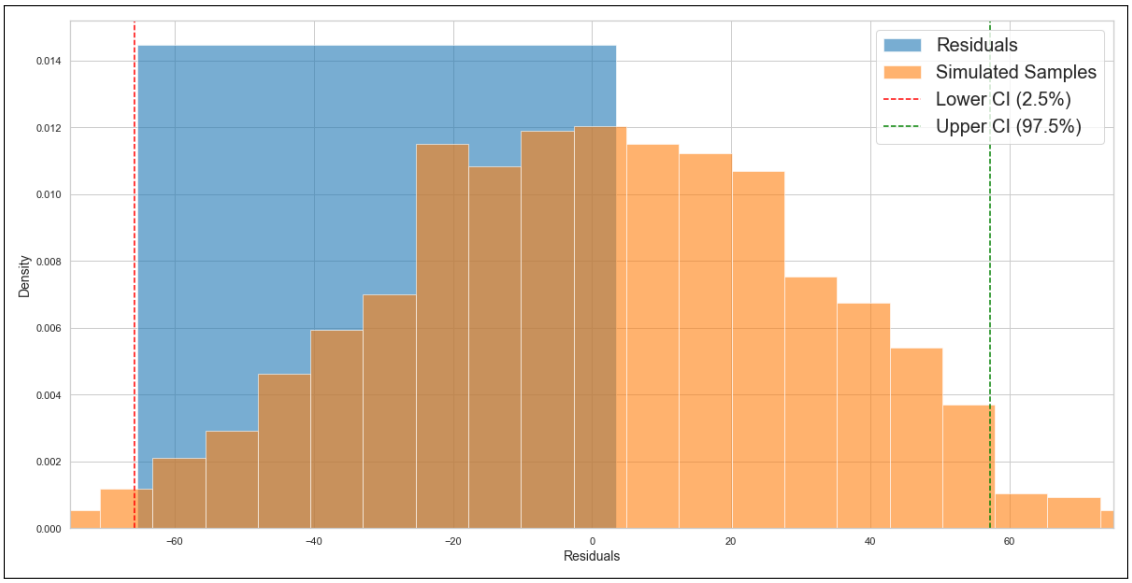


Figure 6.4: LSTM (Multivariable): Residual Analysis

Based on the results of the residual analysis conducted for the multivariable LSTM model, pictured in Figure 6.4, several key insights were drawn. The range of residuals was observed to span from -60 to 5, indicating that the model's predictions exhibited variations from the actual values within this range. The presence of mainly negative positive residuals signified instances where the model tended to underestimate the actual outcomes. Furthermore, the confidence intervals (CIs) provided valuable information about the uncertainty associated with the model's predictions. The lower CI of -60 indicated a high level of confidence that the true residuals fell above this threshold. This corresponded to the model's tendency to occasionally overestimate the predicted values, as evidenced by the presence of negative residuals. Conversely, the upper CI of 60 indicated a similar degree of confidence that the actual residuals were below this value.

6.2.7 Cross Validation Analysis

With the split data set seen in Table 5.4 (and further illustrated in Figure 5.6), we applied Algorithm 3 to each data split to determine how the size and pattern of each data set affect the LSTM model and hence the predicted results.

On each of the data splits, we optimised the hyperparameters using the same base hyperparameter selection in 5.1. The results per data split are shown in Table 6.3.

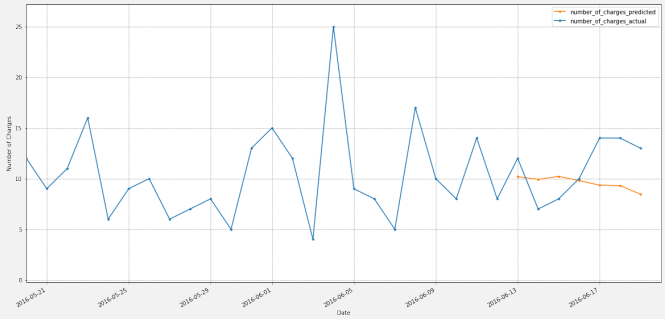
Each LSTM model per data split was trained using the respectively optimised hyperparameters, which resulted in a seven-day prediction per split. Table 6.3 shows the actual and predicted

total number of charges per prediction period of 7 days with the respective point difference comparison and the RMSE. The graphs in Figure 6.10 illustrate the daily actual and predicted values. In the figures, the actual number of charges (blue) were plotted on a daily basis over a period of 30 days, with the predicted values (orange) plotted on the last seven days of the defined period.

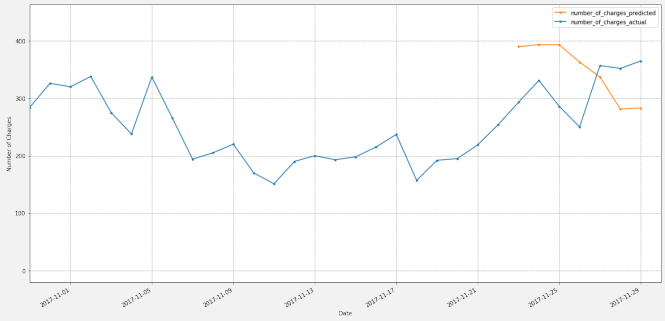
The RMSE allowed us to compare the results per model, but only to a respective split. Hence, we calculated the normalised RMSE, as shown in Table 6.4 in order to compare the RMSE across splits. Split 05 produced the lowest normalised RMSE (0.046), indicating that this Split produced the best prediction values vs actual values in comparison to all splits. Graphically, the RMSE and the normalised RMSE per split for the multivariable LSTM (with all features) can be seen in Table 6.6.

Split No	**Minimum Charges**	**Maximum Charges**	**RMSE**	**Normalised RMSE**
01	7	14	3.400	0.324
02	250	365	84.237	0.274
03	635	716	39.317	0.058
04	2,579	3,686	243.153	0.078
05	11,844	15,476	627.454	0.046

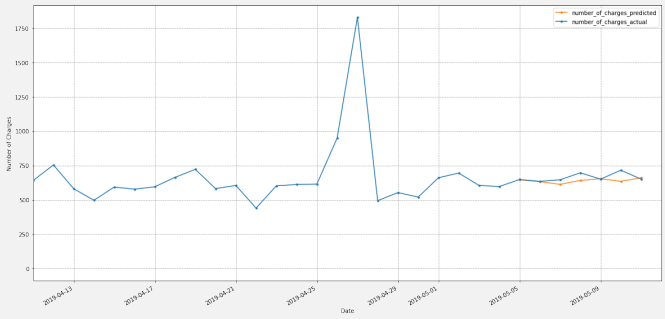
Table 6.4: Multivariable LSTM: RMSE and Normalised RMSE Comparison



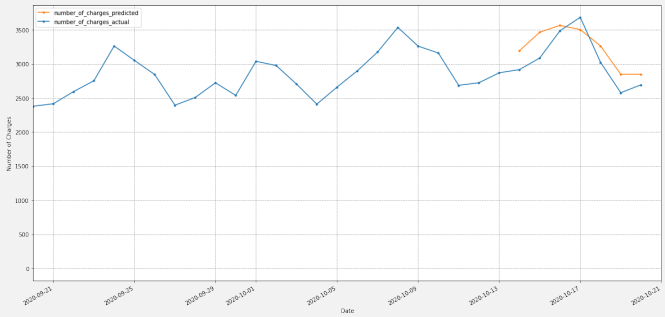
(a) Split 1 Comparison (RMSE: 3.400)



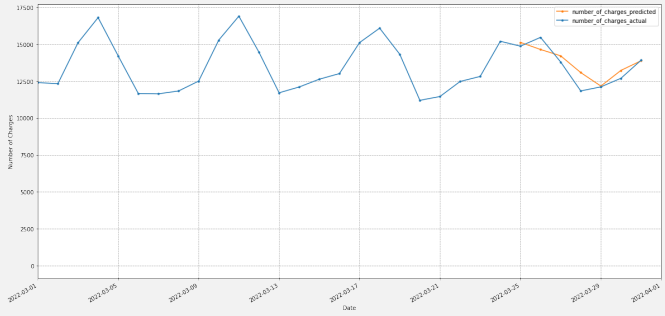
(b) Split 2 Comparison (RMSE: 84.237)



(c) Split 3 Comparison (RMSE:39.317)



(d) Split 4 Comparison (RMSE: 243.153)



(e) Split 5 Comparison (RMSE: 627.454)

Figure 6.5: LSTM (Multivariable - Selected Features) Cross Validation: Actual vs Predicted Charges Results per Split

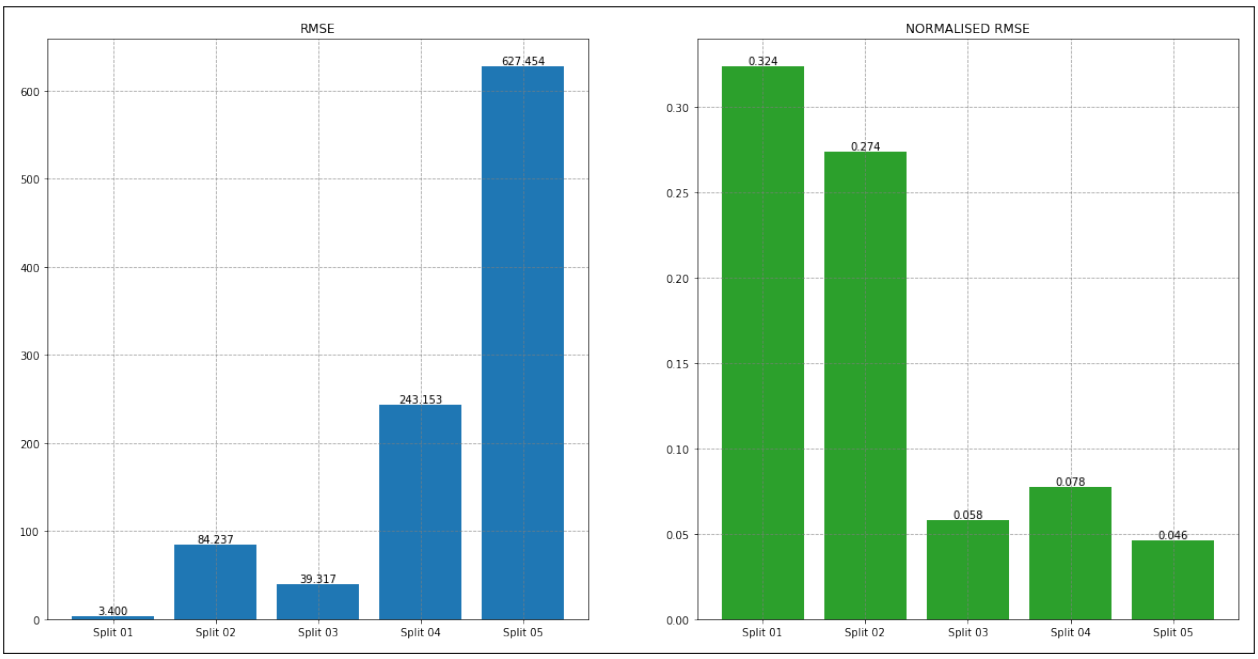


Figure 6.6: LSTM: RMSE and Normalised RMSE Visual Comparison

6.3 Gated Recurrent Units (GRU)

6.3.1 Model Data Preparation

The preparation of the multivariable GRU model, in comparison to the single variable GRU model, only differed in how the data was prepared. As with the prior analyses, the window size and number of days to predict remained the same as the single variable model so that we could compare the outputs. Hence, Algorithm 4 was also used as the high-level process to predict the number of charges.

After the data had been transformed and encoded, it was then tapered by using the variables `days_to_predict = 7` and `window_size = 14`. As explained in Section 6.2.1 the `input_data` was visually represented in Figure 6.1 showing how 21 days of transformed data were arranged to produce the 14-day input data (or training data), and the expected seven-day output (test data).

6.3.2 Scaling the Data

As per the single variable model, the `input_data` was scaled using Equation 5.1 mapping the data to a value between `[0,1]`. The scaled data set was then used to train, validate, and test the model, producing predicted values that were scaled. To better understand the results in the context of the original data set, these predicted values were unscaled using the same initial scaler that was applied to the input.

6.3.3 Hyperparameter Tuning

The process of determining the best combination of hyperparameters to train the GRU was similar to that described in Section 6.2.3 where we detailed the hyperparameter tuning process for LSTM, using [Talos 2019] was used to conduct the experimentation with the same selection of hyperparameters are as per Table 5.1. The only difference here is that the model used was the GRU. Due to the increased size of data, the experiments have a longer run-time when

compared to that of the single variable model. Hence, to complete the experimentation in a reasonable time, and also ensure that we get to the best hyperparameters possible, we ran fifty experiments. As the input data was scaled, the validation loss values were also scaled between `[0,1]`.

In experiment number 10, the lowest validation loss of 0.0236 was found and corresponds to the first set of hyperparameters found in Table 6.5. A visual representation of each scaled validation loss per experiment performed can be seen in Figure 6.7.

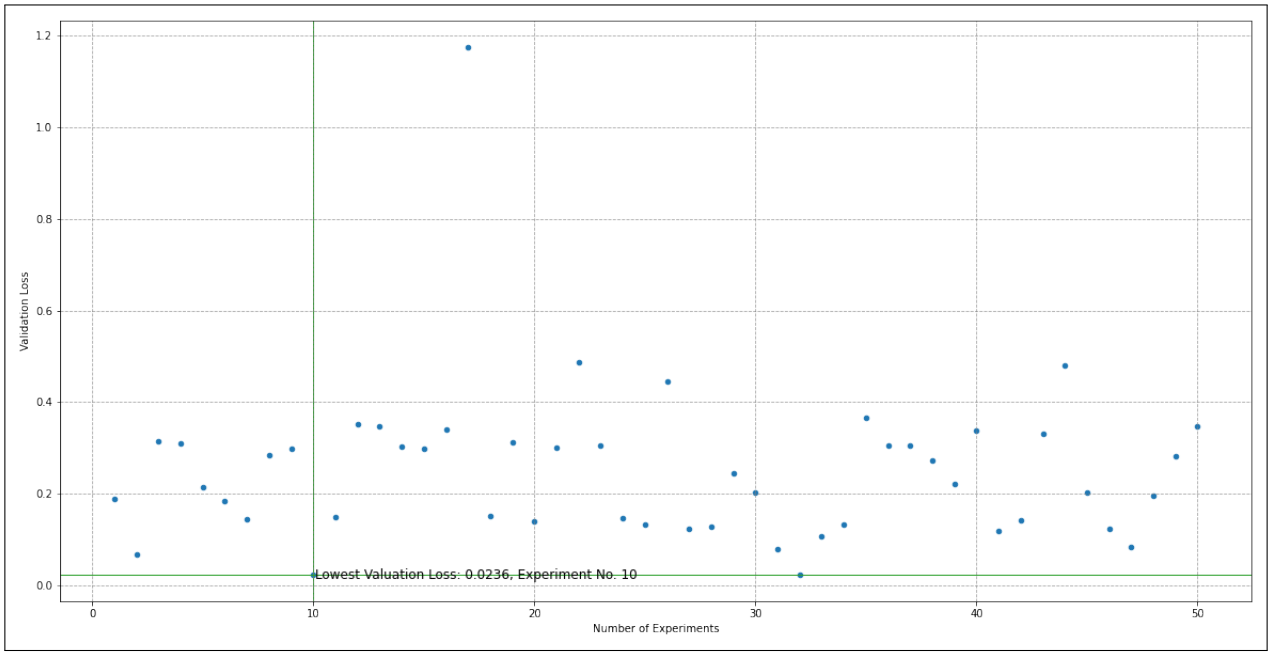


Figure 6.7: Validation Loss Result Spread for GRU Multivariable (All Features) Experiments

6.3.4 Model Training

As shown in Table 5.1, the hyperparameters that resulted in the lowest validation loss were stored and used to train the model defined in Listing 5.2. The difference between the experimentation phase and the training phase was due to the data used when training. Instead of the `exp_training_data`, the GRU was trained using the full `training_data` data set. Once trained, the model was then expected to predict the number of charges for the next seven days.

The hyperparameters that resulted in the lowest validation loss were stored and used to train the model defined in Listing 5.2 can be seen in Table 6.5. The difference between the experimentation phase and the training phase was due to the data used when training. Instead of the `exp_training_data`, the GRU was trained using the full `training_data` data set. Once trained, the model was then expected to predict the number of charges for the next seven days.

6.3.5 Model Results

For the 7-day period ending on the 31st of March 2022, GRU predicted a total of 94,425.24 charges. This is 328.76 charges (or 0.35%) less than the actual amount of 94,754 charges that occurred during this week. The measure of the difference between the actual and predicted values, the RMSE, was 514.998; which is low compared to the actual number of charges. The detailed day-by-day breakdown comparing the actual number of charges and predicted number of charges and the RMSE for the last 7 days in March 2022 were calculated and shown in

No	Testing Loss	Validatio Loss	Activatio	Batch Size	Drop Out	Epochs	Losses	Nodes	Optimizer	Pat-ience
01	0.00064	**0.02359**	tanh	128	0.3	200	MSE	256	Nadam	50
02	0.00050	0.02387	relu	32	0.01	80	MSE	32	Nadam	200
03	0.00079	0.06712	sigmoid	256	0.5	200	MSE	128	RMSprop	50
04	0.00055	0.07971	sigmoid	2	0.05	100	MSE	128	Adam	30
05	0.00453	0.08342	relu	512	0.2	100	MSE	16	RMSprop	100

Table 6.5: GRU (Multivariable - All Features): Five Lowest Validation Losses and their Associated Hyperparameters

Table 6.6. Figure 6.8 visually represents the actual number of charges for the last month of inspection, overlaid with the predicted number of charges for the last seven days.

Date	Actual Value	Predicted Value	Difference	% Difference
Day 01	14,878	14,738.52	-139.48	-0.94%
Day 02	15,476	14,824.00	-652	-4.21%
Day 03	13,806	14,239.74	433.74	3.14%
Day 04	11,844	12,768.42	924.42	7.80%
Day 05	12,119	12,079.70	-39.3	-0.32%
Day 06	12,703	12,300.30	-402.7	-3.17%
Day 07	13,928	13,474.57	-453.43	-3.26%
Total	**94,754**	**94,425.24**	**-328.76**	**-0.35%**
RMSE: 514.998				

Table 6.6: Multivariable GRU Number of Charges: Actual vs Predicted per Day

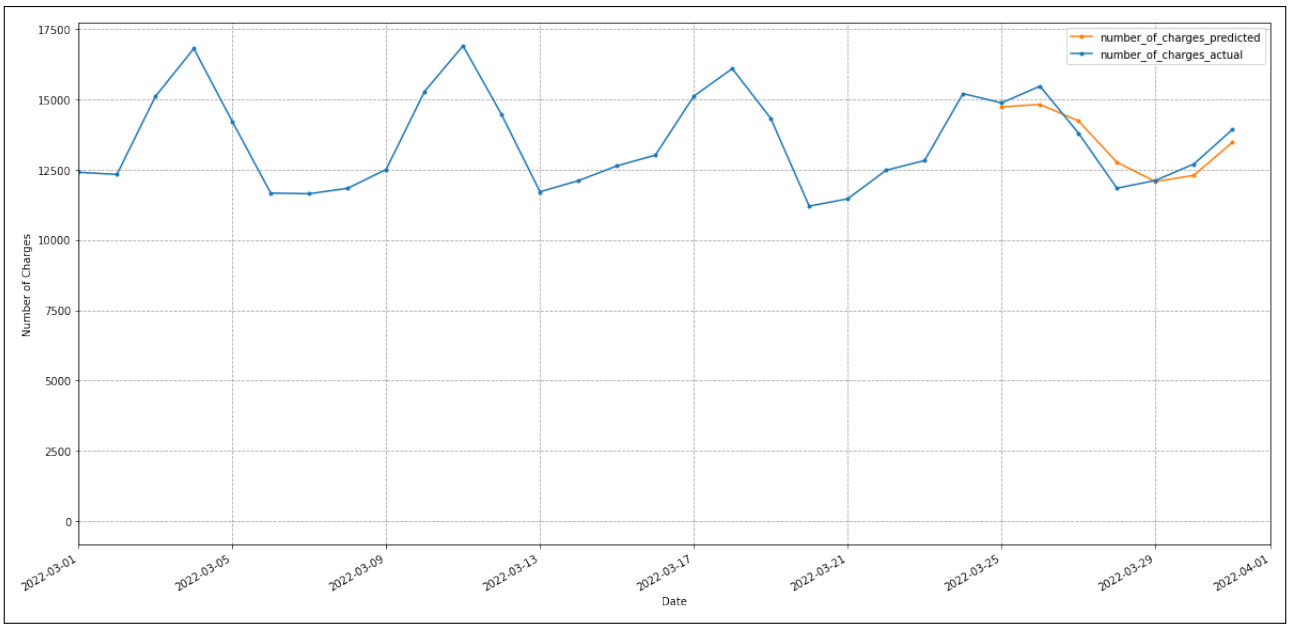


Figure 6.8: GRU: Multivariable Actual and Predicted Figures

6.3.6 Residual Analysis with Confidence Intervals

Based on the results of the residual analysis for the multivariable GRU model pictured in Figure 6.9, several insights can be gleaned. The range of residuals, spanning from -20 to

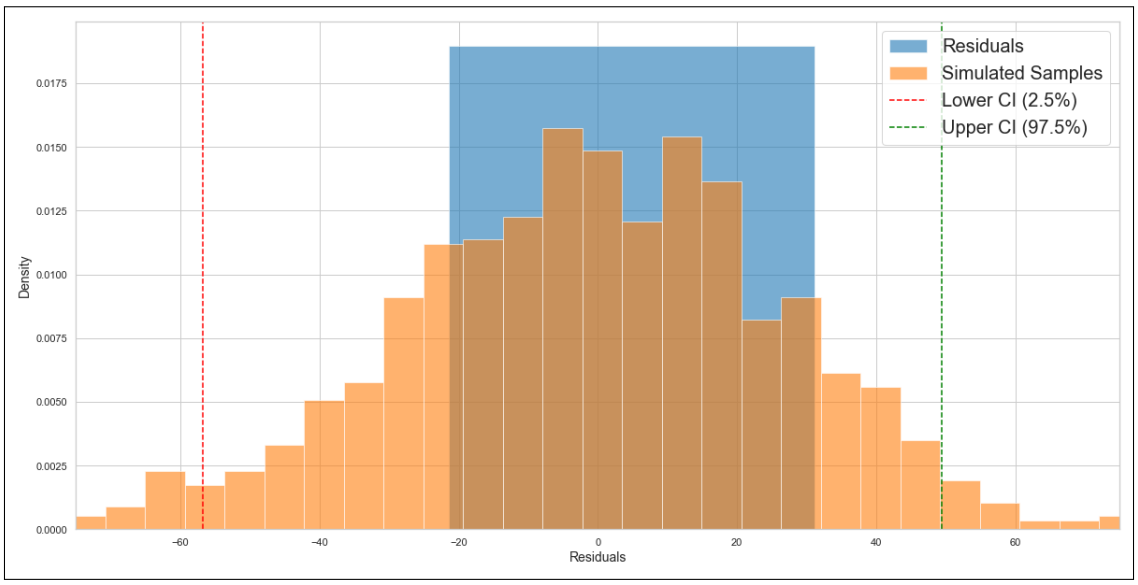


Figure 6.9: GRU (Multivariable): Residual Analysis

30, suggested that the model's predictions exhibit variations from the actual values within this range. The presence of negative residuals indicates that, on occasion, the model tended to overestimate the predicted values. Conversely, the presence of positive residuals signifies instances where the model underestimated the actual outcomes. Furthermore, the confidence intervals (CIs) provided valuable information regarding the uncertainty associated with the model's predictions. The lower CI of -60 suggested that there is a high degree of confidence that the true residuals fall above this threshold. This could be due to the model's tendency to occasionally overestimated the predicted values, as indicated by the presence of negative residuals. On the other hand, the upper CI of 50 implied a similar level of confidence that the actual residuals are below this value. This aligned with the presence of positive residuals, indicating instances where the model's predictions fall short of the actual values.

6.3.7 Cross Validation Analysis

With the split data set seen in Table 5.4 (and further illustrated in Figure 5.6), we applied Algorithm 4 to each data split to determine how the size and pattern of each data set affect the GRU model and hence the predicted results.

On each of the data splits, we optimised the hyperparameters using the same base hyperparameter selection in 5.1. The results per data split shown in Table 5.11.

Split No	Date From	Date To	Actual Value	Predicted Value	Difference	% Difference	RMSE
01	2016-06-12	2016-06-19	78	74.77	-3.23	-4.14%	2.883
02	2017-11-22	2017-11-29	2,234	2,567.44	333.44	14.93%	99.125
03	2019-05-04	2019-05-11	4,644	4,487.33	-156.67	-3.37%	40.069
04	2020-10-13	2020-10-20	21,479	22,256.38	777.38	3.62%	210.957
05	2022-03-24	2022-03-31	94,754	94,425.24	-328.76	-0.35%	514.998

Table 6.7: GRU (Multivariable - All Features) Cross Validation: Actual vs Predicted Charges Results per Split

Each GRU model was trained on the data per split, with the respectively optimised hyperparameters, resulting in a seven-day prediction per split. Table 6.7 shows the actual and predicted total number of charges per prediction period of 7 days with the respective point difference comparison and the RMSE. The graphs in Figure 6.10 illustrate the daily actual and predicted values. In the figures, the actual number of charges (blue) were plotted on a daily basis over a period of 30 days, with the predicted values (orange) plotted on the last seven days of the defined period.

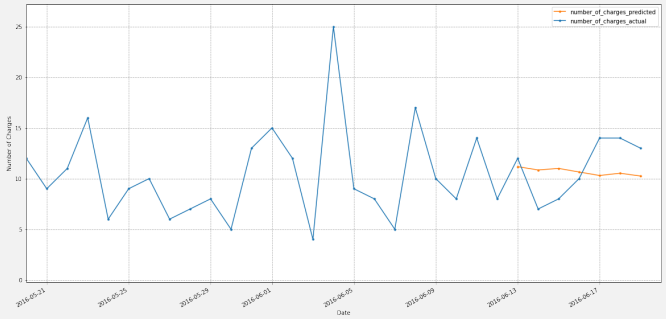
Using the RMSE as an indicator of the goodness of fit, we can compare the different RMSE values to determine which split has the lowest score. However, as the RMSE values were not all calculated on the **same** datasets, we need to calculate the normalised RMSE as per Equation 5.2.

Split No	Minimum Charges	Maximum Charges	RMSE	Normalised RMSE
01	7	14	2.883	0.275
02	250	365	99.125	0.322
03	635	716	40.069	0.059
04	2,579	3,686	210.957	0.067
05	11,844	15,476	514.998	0.038

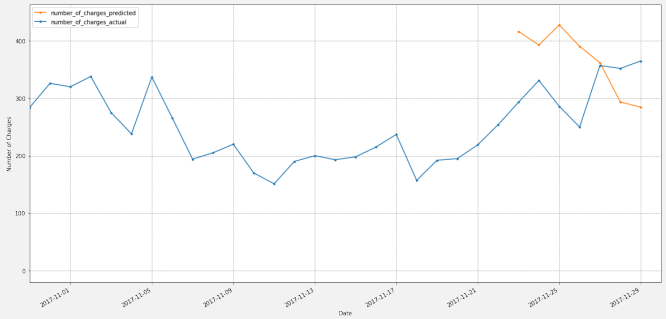
Table 6.8: Multivariable GRU: RMSE and Normalised RMSE Comparison

To better understand why we cannot use the RMSE, consider the results of Split 1 vs Split 5. Here we see that Split 1 has a lower RMSE than Split 5 (RMSE 2.883 vs 514.998 respectively), and hence, we could use this to determine that Split 1's predicted values were better than those from Split 5. When comparing the Normalised RMSE, we can now see that Split 5 produced the best-predicted charges when compared to the other splits. These results showed that despite the fact that the original data set contained different trends over different time periods (which includes the impact of COVID-19), the model performed better having more data points. It also indicated that the windowing period of fourteen days contained enough information to predict the number of charges for seven days within Split 5 as indicated by a low rate of error.

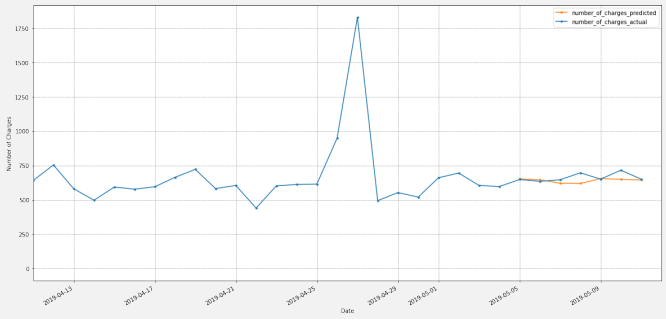
Figure 6.11 visually displays the RMSE with the Normalised RMSE per data split for the GRU. When only comparing the normalised RMSE, it can be seen that Split 5 produced the best-predicted charges when compared to the other splits.



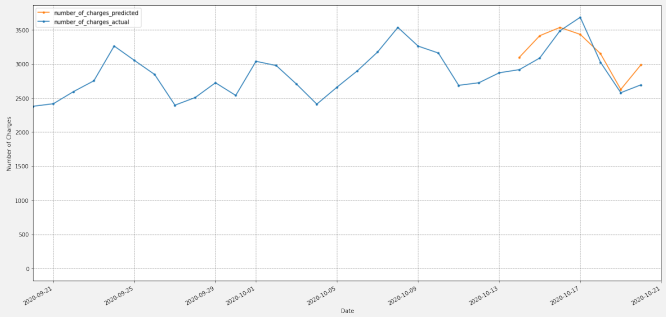
(a) Split 1 Comparison (RMSE: 2.883)



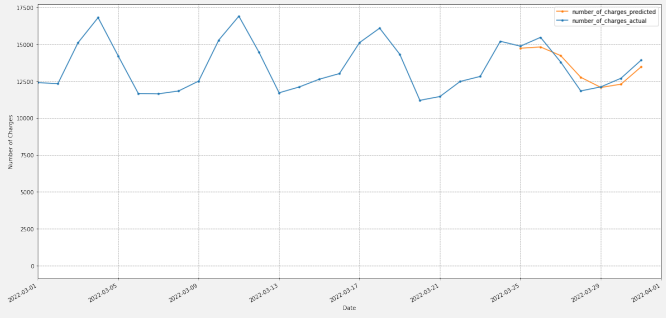
(b) Split 2 Comparison (RMSE: 99.125)



(c) Split 3 Comparison (RMSE:40.069)



(d) Split 4 Comparison (RMSE: 210.957)



(e) Split 5 Comparison (RMSE: 514.998)

Figure 6.10: GRU (Multivariable - All Features) Cross Validation: Actual vs Predicted Charges Results per Split

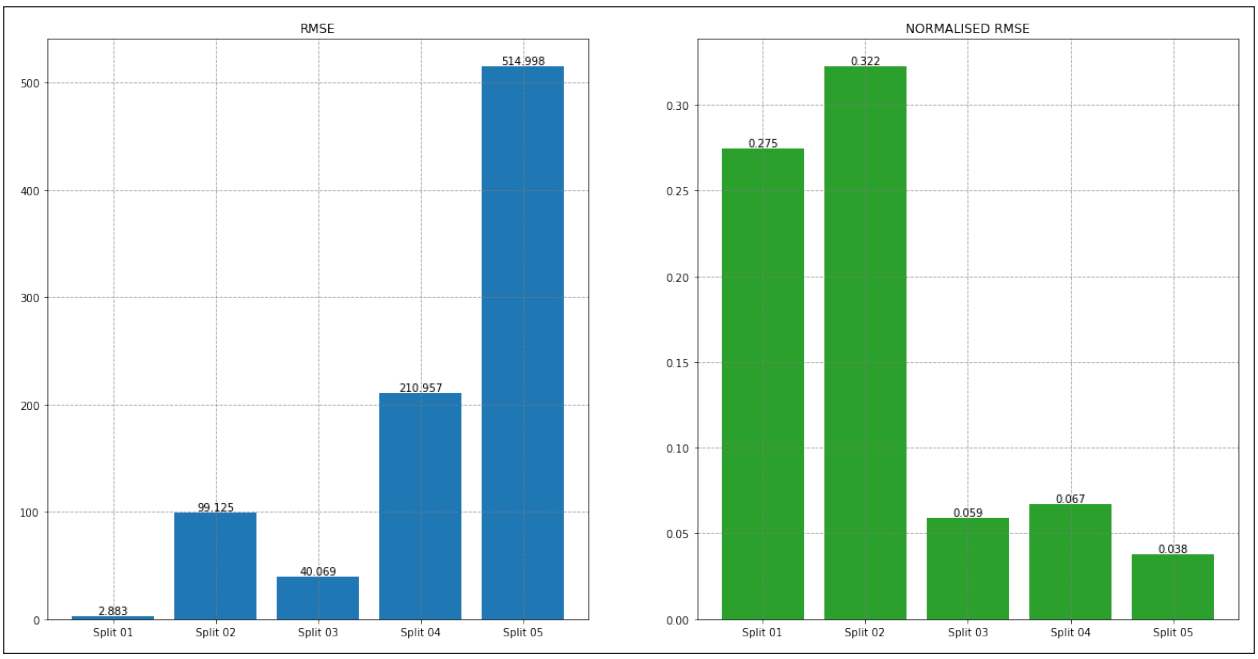


Figure 6.11: GRU (Multivariable Cross Validation): RMSE and Normalised RMSE Visual Comparison

No	Model	Hyperparameters	Time (hh:mm:ss)
01	LSTM	8	02:22:01
02	GRU	8	04:24:54

Table 6.9: Time taken to find the optimal hyperparameters for the Multivariable Analysis

6.4 Results Comparison

Once the experiments were completed and each of the models predicted the number of charges for the last 7 days in March 2022, we were then able to compare the results per model. In these analyses, we considered all data points including fourteen data features and the number of charges that were completed per day from August 2014. The total data set consisted of 2,639 data points per feature, where the last 7 data points were considered as the test data. Both models (LSTM, and GRU) used the training data to determine the best hyperparameters required to produce the best test result. Once these hyperparameters were determined, each model then predicted the number of charges for the next seven days which were then compared to the test data set.

6.4.1 Hyperparameter Optimization

Both LSTM and GRU models each had eight hyperparameters to optimise, and the respective time to find the optimised hyperparameters per model was shown in Table 7.9. Finding the optimal hyperparameters for LSTM took 2 hours less time than the GRU. We would expect that the GRU and LSTM have similar times given the similarity of the models and the required hyperparameters. Hence it is assumed that because the finding of the optimal hyperparameters occurred on a local machine, any event on the machine (automatic software updates, for example) could have caused this delay.

Date	Actual Value	LSTM Predicted Value	GRU Predicted Value
Day 01	14,878	15,135.87	14,738.52
Day 02	15,476	14,652.85	14,824.00
Day 03	13,806	14,223.95	14,239.74
Day 04	11,844	13,087.43	12,768.42
Day 05	12,119	12,173.08	12,079.70
Day 06	12,703	13,238.34	12,300.30
Day 07	13,928	13,889.37	13,474.57
Total	**94,754**	**96,400.90**	**94,425.24**
Difference		**1,646.90**	**-328.76**
% Diff		**1.71%**	**-0.35%**
RMSE		**627.454**	**514.998**

Table 6.10: The Actual Number of Charges vs The Predicted Number of Charges per Model

6.4.2 Model Results

Once the hyperparameters were found, each model predicted the number of charges per day for a period of seven days. These were then plotted against the actual values of the charges over those seven days, where the RMSE of each model result was calculated based on the actual charges, as shown in Table 6.10. On a total level, LSTM **overpredicted** the actual values by **1.71%**, and GRU **underpredicted** by **0.35%**. When comparing the RMSE values, the GRU had the least amount of errors in its predictions in comparison to the LSTM (514.998 vs 627.454). Figure 6.12 visually represented the results per model against the actual number of charges.

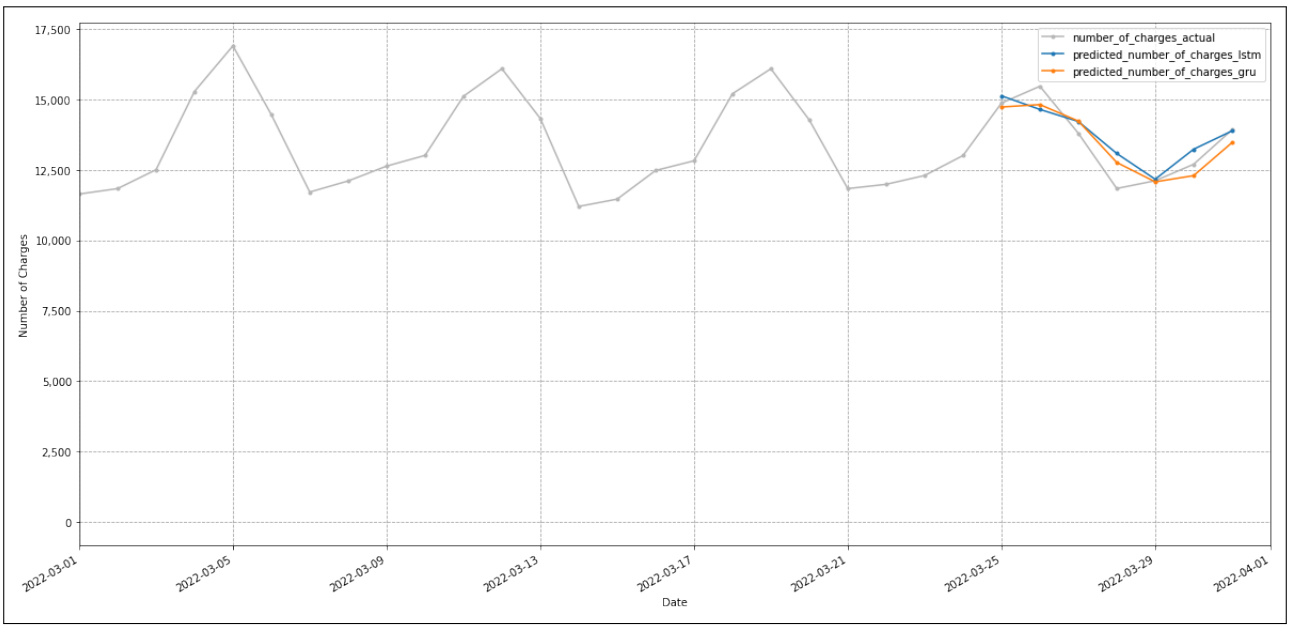


Figure 6.12: Prediction Results for LSTM and GRU

6.4.3 Residual Analysis Comparison

The investigation into residuals and their associated confidence intervals for LSTM and GRU models provided distinct insights into their predictive performance. The GRU model showcased a narrow range of residuals and tight confidence intervals, signifying accurate and consistent

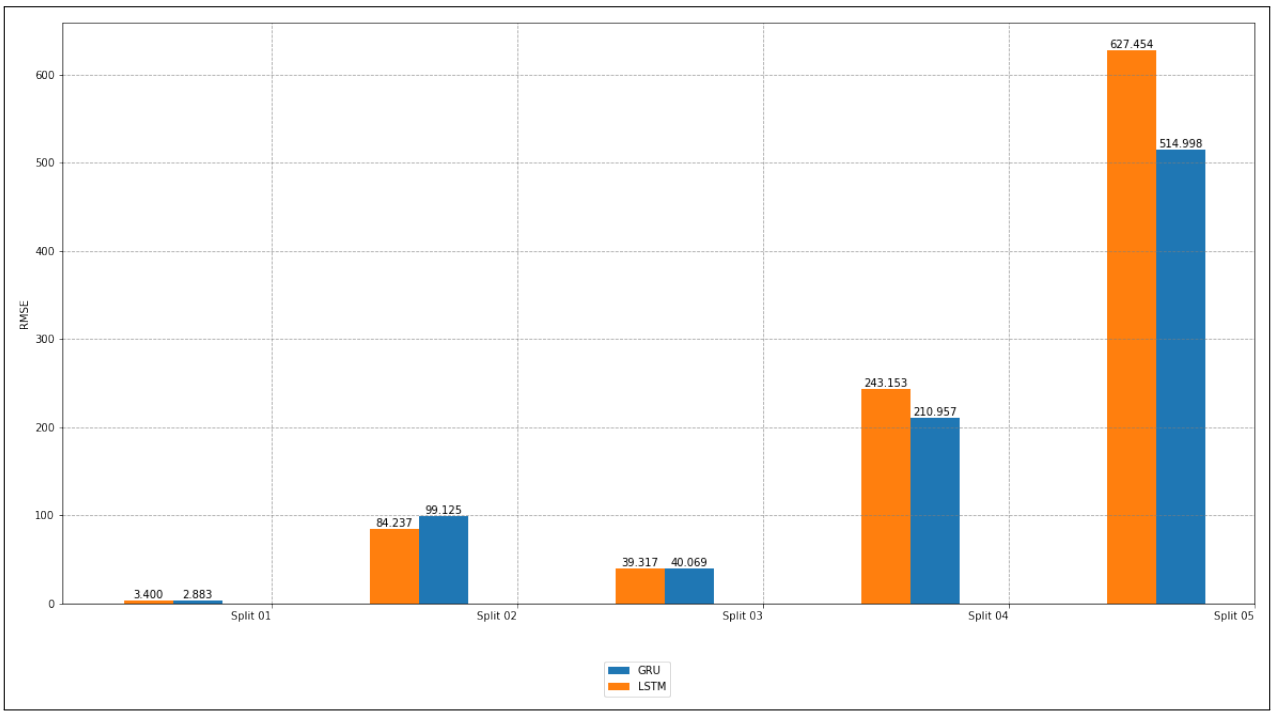


Figure 6.13: RMSE Comparison for LSTM and GRU

predictions closely aligned with actual values. In contrast, the LSTM model's residuals were mostly negative, with confidence intervals around -60 and 60. These findings revealed the GRU model's superior precision and the LSTM model's higher variability in forecasting accuracy. These results are in alignment with the conclusions found in Table 6.10 with regard to the RMSE results.

6.4.4 Cross Validation Results

The RMSE and normalised RMSE results per data split can be seen graphically in Figures 6.13 and 6.14. Whilst the LSTM performs better in Split 02 and marginally better in Split 03, the RMSE results from the GRU are far better in the final two splits. To be able to compare the RMSE per model across every split, the RMSE values were normalised and on average, the GRU model outperformed the LSTM models. Furthermore, the normalised RMSE decreased as the data increased in size for all models. This can be seen graphically in Figure 6.14.

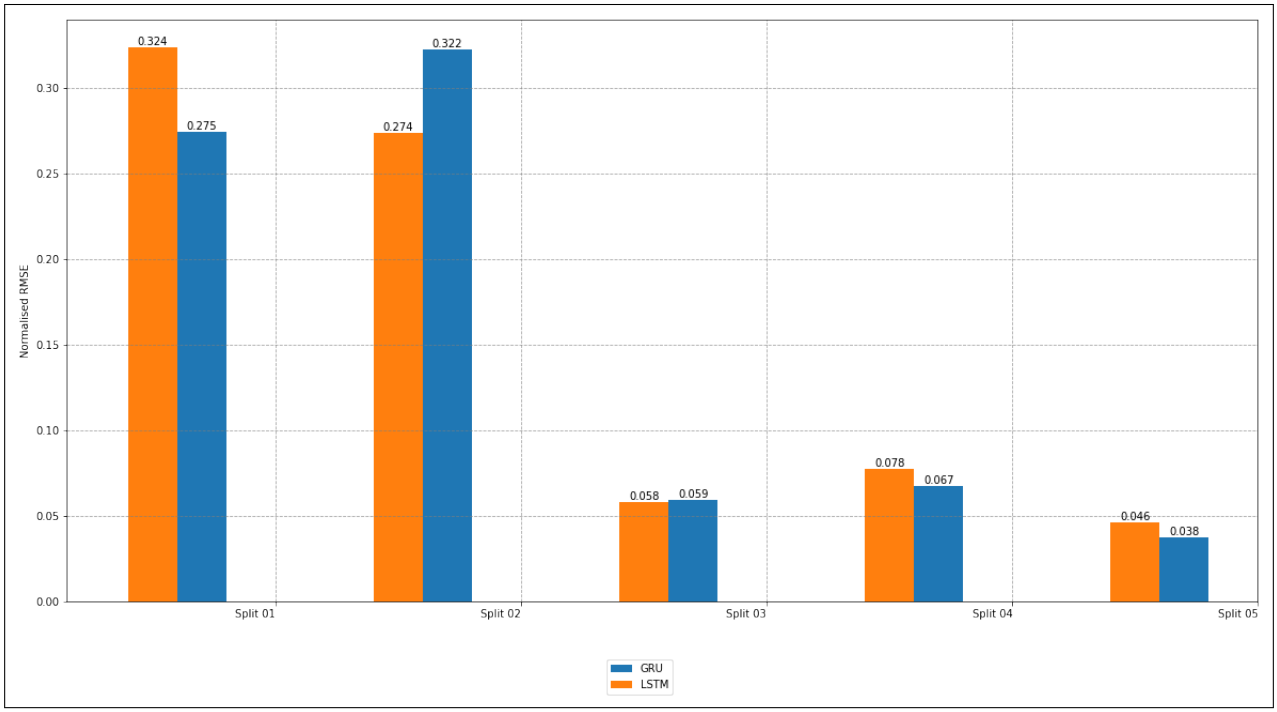


Figure 6.14: Normalised RMSE Comparison for LSTM and GRU

7 Multivariable Analysis (Selected Features)

7.1 Introduction

As shown in Section 6, the multivariable analysis required features to the analysis additional than the number of charges. As with the previous section, the purpose of this multivariable analysis was to determine whether there were features that could improve the difference between the actual and forecasted figure, but by only using a selected number of features. Section 4.4 detailed the feature analysis and feature correlation, and using these results, we determined it best to use **three features** to predict the number of charges.

Private Charges The `private_charges` feature had an average correlation coefficient of 0.981 with the number of charges. Due to this strong positive relationship between the two features, it was decided that the other features with such a strong correlation coefficient (>0.94) should not be used as these may not add any additional information to the model.

Session Duration The `session_duration` feature also has a relatively strong relationship with the `number_of_charges` and hence was also selected.

New Contracts To add a bit of variety to the data, and try to include information in the models that may have been missed in the single variable models, the `new_contracts` feature was selected to be included in the analysis.

The features with a correlation coefficient below 0.65 were excluded as their relationships with the `number_of_charges` may not have been beneficial to the model. The feature `new_covid_cases` had a relatively high correlation score of 0.740 but was ultimately not selected as the cases may have only been relevant to a selected period of the data set.

7.2 Long Short-Term Memory (LSTM)

7.2.1 Model Data Preparation and Data Scaling

The data preparation explained in Section 6.2.1 along with Algorithm 3 was also used as the high-level process that we followed to predict the number of charges. The reduction of the features in this data set only impacts the number of columns in the data, with no other impacts on the model.

As per the multivariable model, the data was scaled using Equation 5.1 that mapped all values between `[0,1]`. This data set was then used to create the data sets to train, validate, and test the model, producing predicted values that were scaled. These predicted values were then unscaled using the same initial scaler as used in the scaling process.

7.2.2 Hyperparameter Tuning and Model Training

To remain in line with the previous experiments from the Multivariable Analysis (All Features), we ran fifty experiments. As the input data was scaled, the validation loss values were also scaled between `[0,1]`. The selection of hyperparameters for this data set can be seen as per Table 5.1, and once again, [Talos 2019] was used to conduct the experimentation.

In experiment number 20, the lowest validation loss of 0.0215 was found and corresponds to the first set of hyperparameters found in Table 7.1. A visual representation of each scaled validation loss per experiment performed can be seen in Figure 7.1

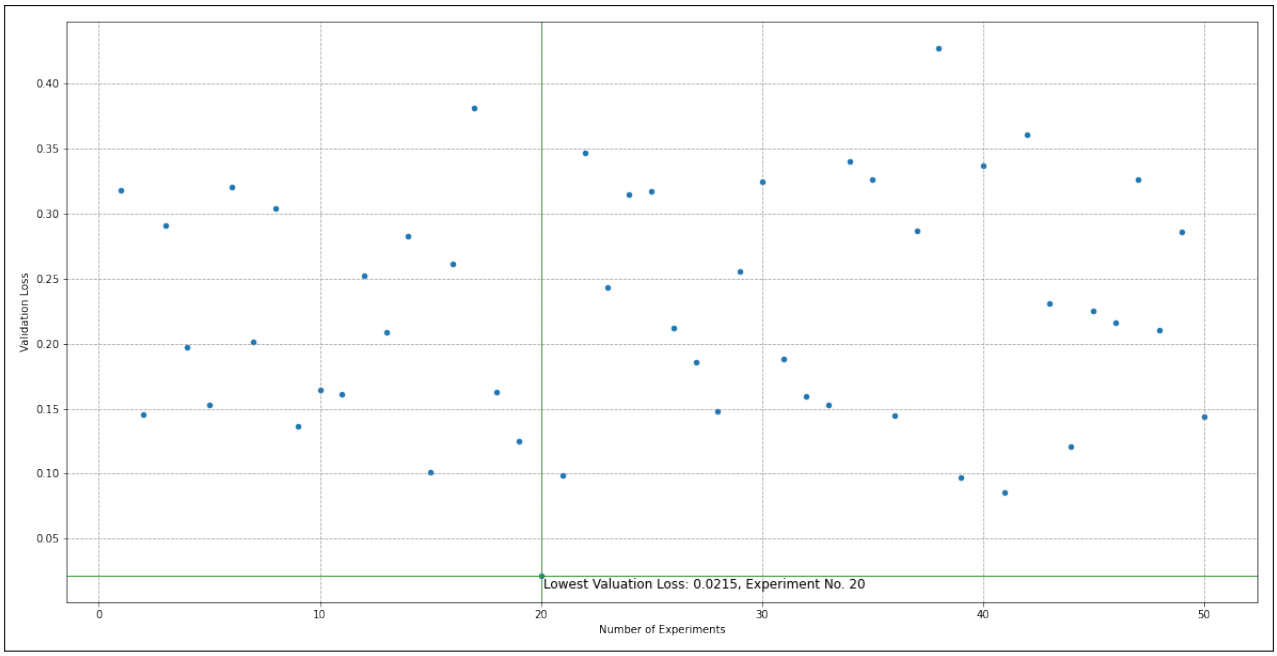


Figure 7.1: Validation Loss Result Spread for LSTM Multivariable (Selected Features) Experiments

No	Testing Loss	Validatio Loss	Activatio	Batch Size	Drop Out	Epochs	Losses	Nodes	Optimizer	Patience
01	0.00070	**0.02152**	relu	64	0.2	40	MSE	512	Nadam	50
02	0.00094	0.08571	relu	1024	0.5	40	MSE	512	Nadam	100
03	0.00042	0.09751	relu	16	0.01	40	MSE	128	Nadam	100
04	0.00088	0.09880	sigmoid	4	0.05	200	MSE	8	Adam	200
05	0.00052	0.10105	sigmoid	2	0.075	100	MSE	64	Nadam	20

Table 7.1: LSTM (Multivariable - Selected Features): Five Lowest Validation Losses and their Associated Hyperparameters

7.2.3 Model Results

For the 7-day period ending on the 31st of March 2022, LSTM predicted a total of 97,265 charges. This is 2,511 charges (or 2.65%) more than the actual amount of 94,754 charges that occurred during this week. The measure of difference between the actual and predicted values, the RMSE, was 488.768; which is low compared to the actual number of charges. The detailed day-by-day breakdown comparing the actual number of charges and predicted number of charges and the RMSE for the last 7 days in March 2022 were calculated and shown in

Date	Actual Value	Predicted Value	Difference	% Difference
Day 01	14,878	15,474.57	596.57	4.01%
Day 02	15,476	15,563.38	87.38	0.56%
Day 03	13,806	14,339.91	533.91	3.87%
Day 04	11,844	12,759.47	915.47	7.73%
Day 05	12,119	12,464.25	345.25	2.85%
Day 06	12,703	12,900.81	197.81	1.56%
Day 07	13,928	13,762.86	-165.14	-1.19%
Total	**94,754**	**97,265.23**	**2,511.23**	**2.65%**
RMSE: 488.768				

Table 7.2: LSTM (Multivariable - Selected Features) Number of Charges: Actual vs Predicted per Day

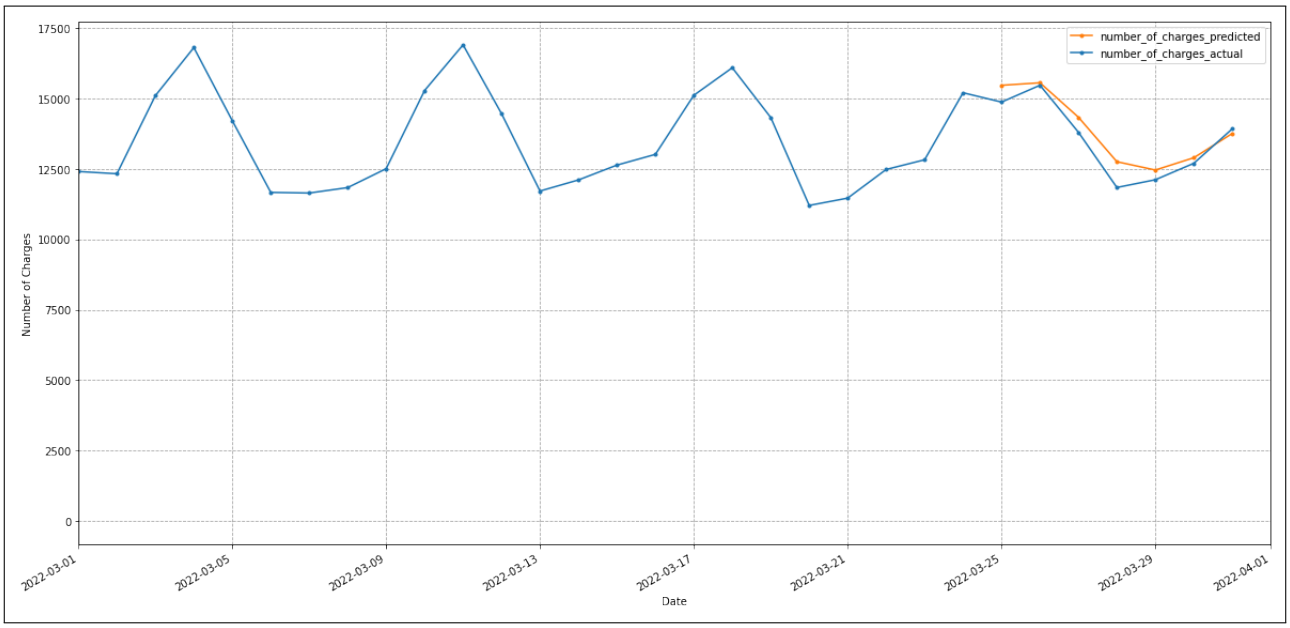


Figure 7.2: LSTM: Multivariable Actual and Predicted Figures

Table 7.2. Figure 7.2 visually represents the actual number of charges for the last month of inspection, overlaid with the predicted number of charges for the last seven days.

7.2.4 Residual Analysis with Confidence Intervals

Based on the results of the residual analysis conducted for the multivariable (selected features) LSTM model, pictured in Figure 7.3, several key insights were drawn. The range of residuals was observed to span from -20 to 20, indicating that the model's predictions exhibited variations from the actual values within this range. This range encompassed both negative and positive residuals, indicating instances where the model underestimated and overestimated the actual outcomes, respectively. Furthermore, the confidence intervals (CIs) provided valuable information about the uncertainty associated with the model's predictions. The lower CI of -50 indicated a high level of confidence that the true residuals fell above this threshold. This corresponded to the model's tendency to occasionally overestimate the predicted values, as evidenced by the presence of negative residuals. Conversely, the upper CI of 50 indicated a similar degree of confidence that the actual residuals were below this value.

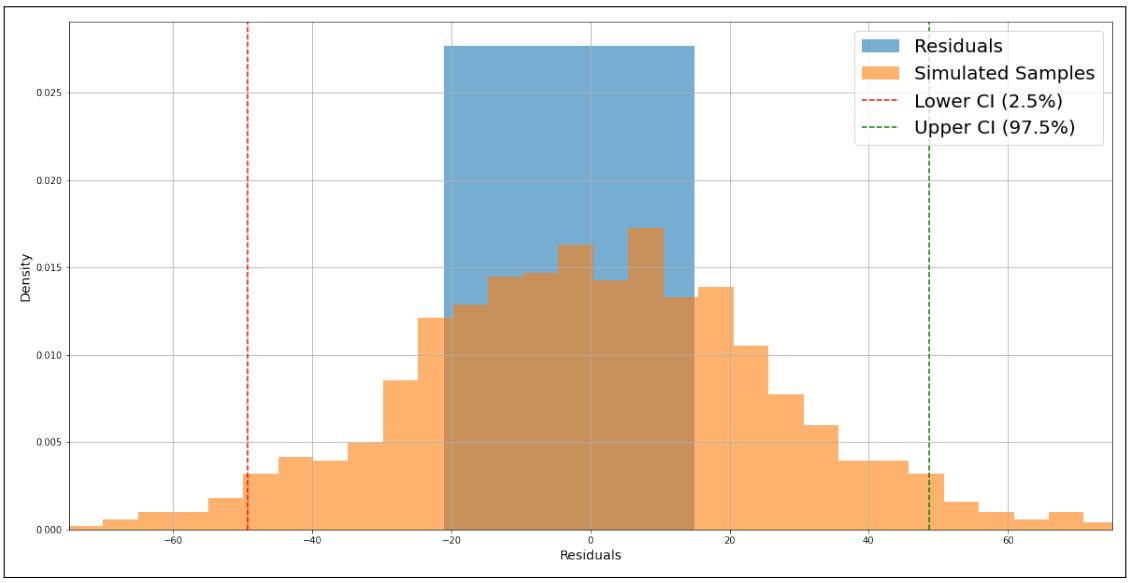


Figure 7.3: LSTM (Multivariable - Selected Features): Residual Analysis

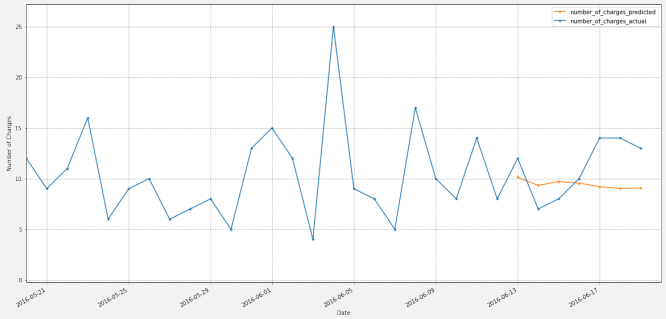
Split No	Date From	Date To	Actual Value	Predicted Value	Difference	% Difference	RMSE
01	2016-06-12	2016-06-19	78	66.03	-11.97	-15.37%	3.277
02	2017-11-22	2017-11-29	2,234	2,660.13	426.13	19.07%	110.780
03	2019-05-04	2019-05-11	4,644	4,533.39	-110.61	-2.38%	33.004
04	2020-10-13	2020-10-20	21,479	22,330.90	851.9	3.97%	209.778
05	2022-03-24	2022-03-31	94,754	97,265.23	2,511.23	2.65%	488.768

Table 7.3: LSTM (Multivariable - Selected Features) Cross Validation: Actual vs Predicted Charges Results per Split

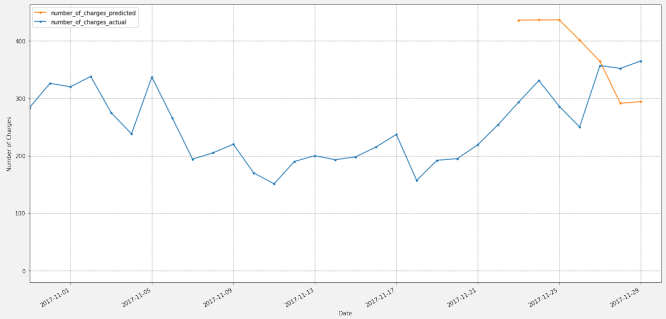
7.2.5 Cross Validation Analysis

We applied Algorithm 3 to each data split (shown in Table 5.4) to determine how the size and pattern of each data set affect the LSTM model and hence the predicted results. Each LSTM model per data split was trained, using the respectively optimised hyperparameters, which resulted in a seven-day prediction per split. Table 7.3 shows the actual and predicted total number of charges per prediction period of 7 days with the respective point difference comparison and the RMSE. A visual representation of the actual and predicted charges can be seen in Figure 7.4. In the figures, the actual number of charges (blue) were plotted on a daily basis over a period of 30 days, with the predicted values (orange) plotted on the last seven days of the defined period.

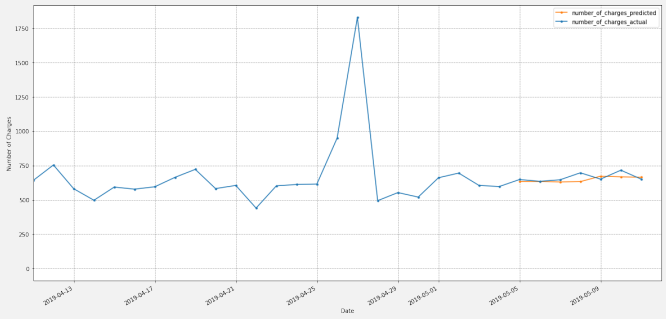
To compare the RMSE across splits, we calculated the normalised RMSE, as shown in Table 7.4. Split 05 produced the lowest normalised RMSE (0.036), indicating that this split produced the best prediction values vs actual values in comparison to all splits. Graphically, the RMSE and the normalised RMSE per split for the multivariable LSTM (with selected features) can be seen in Table 7.5.



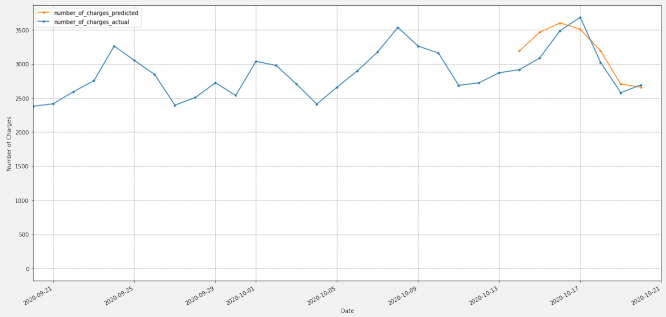
(a) Split 1 Comparison (RMSE: 3.277)



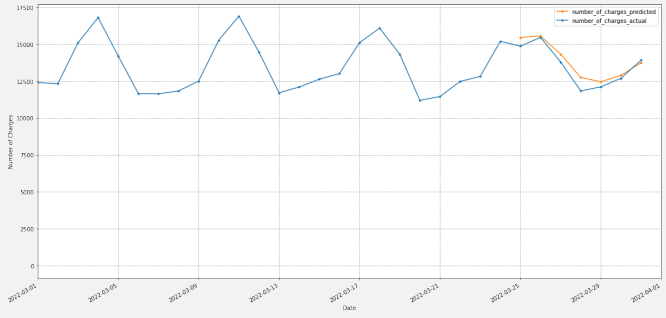
(b) Split 2 Comparison (RMSE: 110.780)



(c) Split 3 Comparison (RMSE: 33.004)



(d) Split 4 Comparison (RMSE: 209.778)



(e) Split 5 Comparison (RMSE: 488.768)

Figure 7.4: LSTM (Multivariable - Selected Features) Cross Validation: Actual vs Predicted Charges Results per Split

Split No	Minimum Charges	Maximum Charges	RMSE	Normalised RMSE
01	7	14	3.277	0.312
02	250	365	110.78	0.36
03	635	716	33.004	0.049
04	2,579	3,686	209.778	0.067
05	11,844	15,476	488.768	0.036

Table 7.4: LSTM (Multivariable - Selected Features): RMSE and Normalised RMSE Comparison

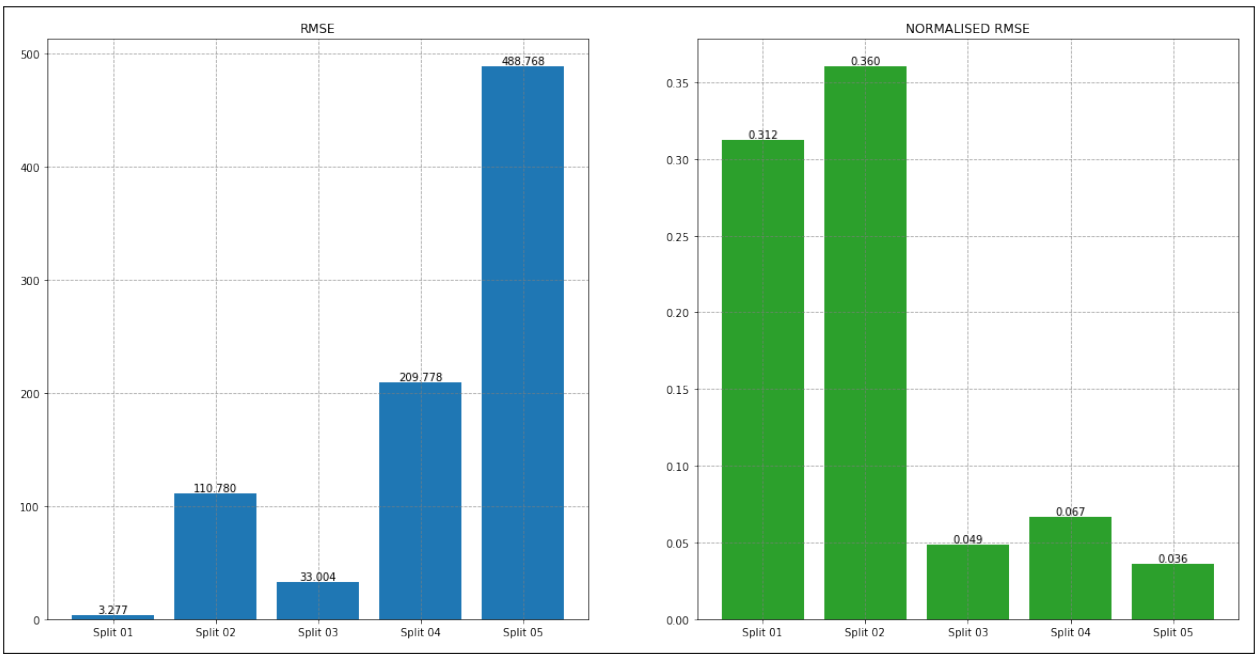


Figure 7.5: LSTM (Multivariable - Selected Features): RMSE and Normalised RMSE Visual Comparison

7.3 Gated Recurrent Units (GRU)

7.3.1 Model Data Preparation and Data Scaling

The data preparation explained in Section 6.3.1 along with Algorithm 4 was also used as the high-level process that we followed to predict the number of charges. The reduction of the features in this data set only impacts the number of columns in the data, with no other impacts on the model.

As per the multivariable model, the data was scaled using Equation 5.1 that mapped all values between `[0,1]`. This data set was then used to create the data sets to train, validate, and test the model, producing predicted values that were scaled. These predicted values were then unscaled using the same initial scaler as used in the scaling process.

7.3.2 Hyperparameter Tuning and Model Training

With the data sets created, we determined the best combination of hyperparameters that can be used to train the GRU as described in 6.3.3. The hyperparameters corresponding to the lowest validation loss values found in the experimentation phase were stored as the optimal hyperparameter set used in the final forecasting of the number of charges. The selection of hyperparameters for this data set can be seen as per Table 5.1, and once again, Talos [2019] was used to conduct the experiment.

To remain in line with the previous experiments from the Multivariable Analysis using all features, once again, we ran fifty experiments. As the input data was scaled, the validation loss values were also scaled between `[0,1]`.

In experiment number 37, the lowest validation loss of 0.0284 was found and corresponds to the first set of hyperparameters found in Table 7.5. A visual representation of each scaled validation loss per experiment performed can be seen in Figure 7.6

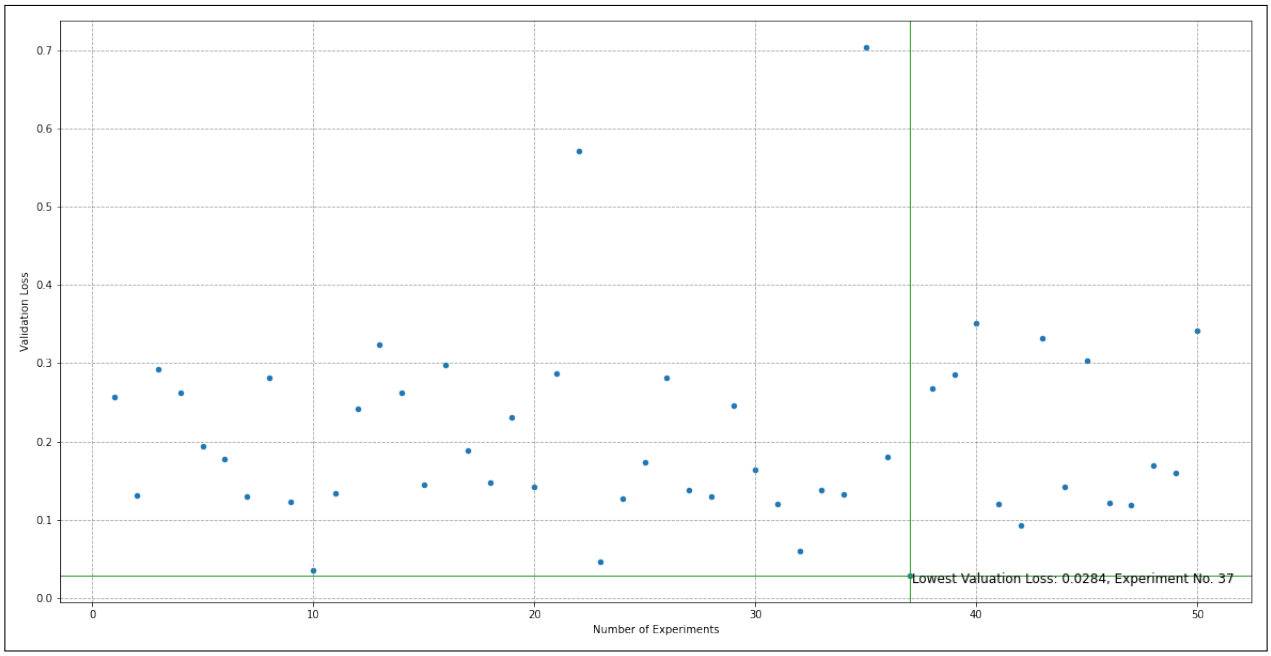


Figure 7.6: Validation Loss Result Spread for GRU Multivariable (Selected Features) Experiments

No	Testing Loss	Validatio Loss	Activatio	Batch Size	Drop Out	Epochs	Losses	Nodes	Optimizer	Patience
01	0.00070	**0.02152**	relu	64	0.2	40	MSE	512	Nadam	50
02	0.00094	0.08571	relu	1024	0.5	40	MSE	512	Nadam	100
03	0.00042	0.09751	relu	16	0.01	40	MSE	128	Nadam	100
04	0.00088	0.09880	sigmoid	4	0.05	200	MSE	8	Adam	200
05	0.00052	0.10105	sigmoid	2	0.075	100	MSE	64	Nadam	20

Table 7.5: GRU (Multivariable - Selected Features): Five Lowest Validation Losses and their Associated Hyperparameters

Date	Actual Value	Predicted Value	Difference	% Difference
Day 01	14,878	15,532.75	654.75	4.40%
Day 02	15,476	15,981.8	505.8	3.27%
Day 03	13,806	14,591.17	785.17	5.69%
Day 04	11,844	13,080.78	1,236.78	10.44%
Day 05	12,119	12,971.86	852.86	7.04%
Day 06	12,703	13,007.11	304.11	2.39%
Day 07	13,928	13,627.8	-300.2	2.16%
Total	**94,754**	**98,793.27**	**4,039.27**	**4.26%**
RMSE: 731.009				

Table 7.6: GRU (Multivariable - Selected Features) Number of Charges: Actual vs Predicted per Day

The hyperparameters that resulted in the lowest validation loss were stored and used to train the model defined in Listing 5.2. Once trained, the model was then expected to predict the number of charges for the next seven days.

7.3.3 Model Results

For the 7-day period ending on the 31st of March 2022, GRU predicted a total of 98,793 charges. This is 4,039 charges (or 4.26%) less than the actual amount of 94,754 charges that occurred during this week. The measure of the difference between the actual and predicted values, the RMSE, was 731.009; which is low compared to the actual number of charges. The detailed day-by-day breakdown comparing the actual number of charges and predicted number of charges and the RMSE for the last 7 days in March 2022 were calculated and shown in Table 7.6. Figure 7.7 visually represents the actual number of charges for the last month of inspection, overlaid with the predicted number of charges for the last seven days.

7.3.4 Residual Analysis with Confidence Intervals

Based on the results of the residual analysis conducted for the multivariable (selected features) GRU model, pictured in Figure 7.8, several key insights were drawn. The range of residuals was observed to span from -45 to 5, indicating that the model's predictions exhibited variations from the actual values within this range. The presence of mainly negative positive residuals signified instances where the model tended to underestimate the actual outcomes. Furthermore, the confidence intervals (CIs) provided valuable information about the uncertainty associated with the model's predictions. The lower CI of -75 indicated a high level of confidence that the true residuals fell above this threshold. This corresponded to the model's tendency to occasionally overestimate the predicted values, as evidenced by the presence of negative residuals.

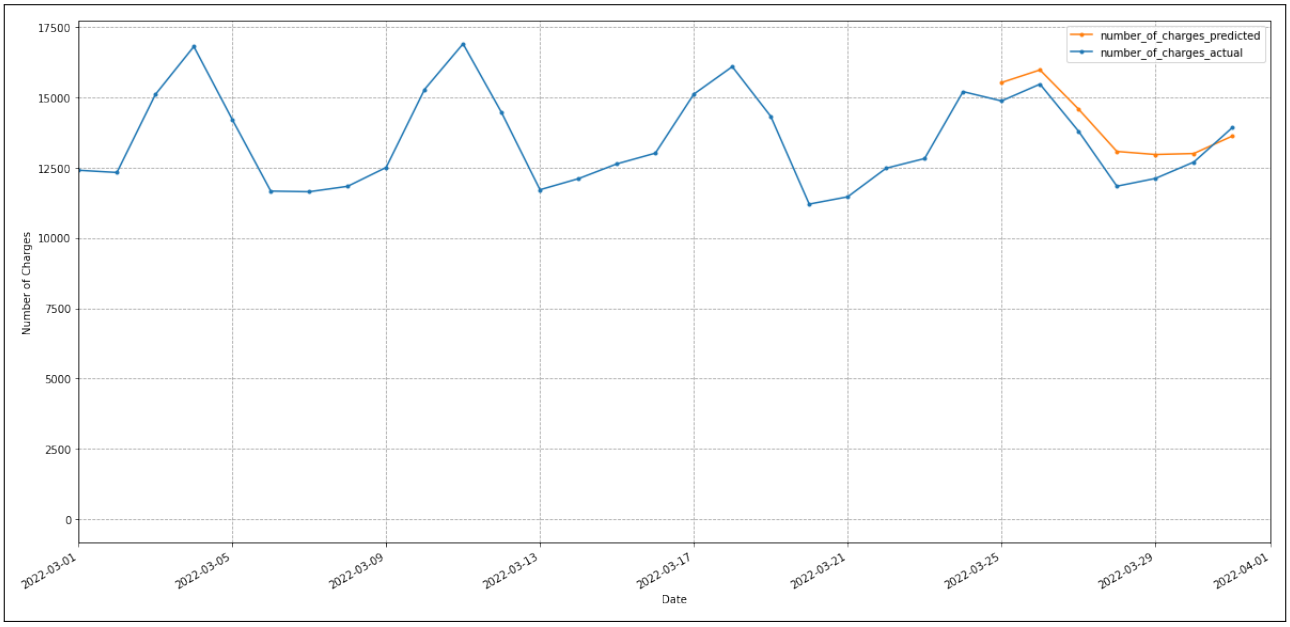


Figure 7.7: GRU (Multivariable - Selected Features): Actual and Predicted Figures

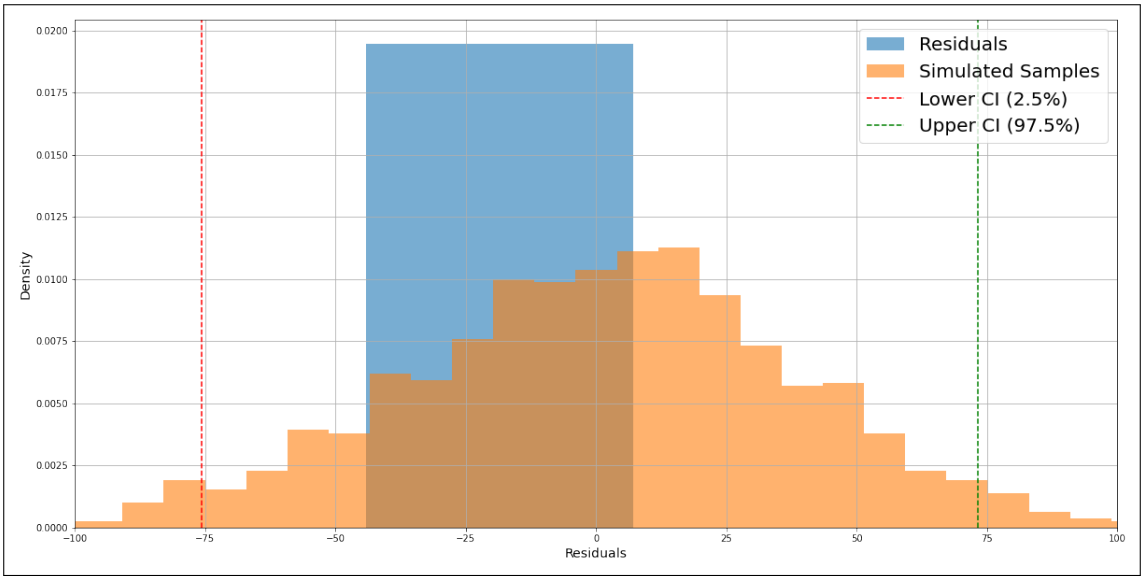


Figure 7.8: GRU (Multivariable - Selected Features): Residual Analysis

Split No	Date From	Date To	Actual Value	Predicted Value	Difference	% Difference	RMSE
01	2016-06-12	2016-06-19	78	67.1	10.9	-13.97%	3.217
02	2017-11-22	2017-11-29	2,234	2,834.51	600.51	26.88%	134.514
03	2019-05-04	2019-05-11	4,644	4,499.81	-144.19	-3.10%	47.488
04	2020-10-13	2020-10-20	21,479	23,402.38	1,923.38	8.95%	323.600
05	2022-03-24	2022-03-31	94,754	98,793.27	4,039.27	4.26%	731.009

Table 7.7: GRU (Multivariable - Selected Features) Cross Validation: Actual vs Predicted Charges Results per Split

Split No	Minimum Charges	Maximum Charges	RMSE	Normalised RMSE
01	7	14	3.217	0.306
02	250	365	134.514	0.437
03	635	716	47.488	0.070
04	2,579	3,686	323.600	0.103
05	11,844	15,476	731.009	0.054

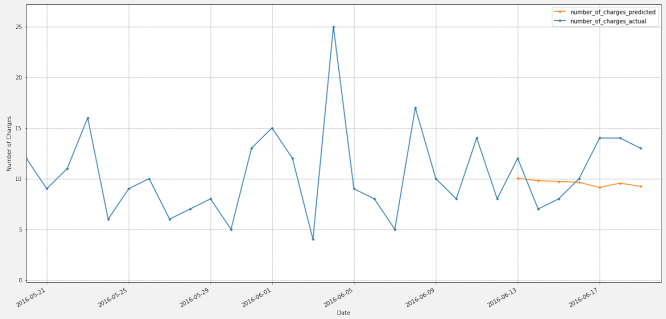
Table 7.8: GRU (Multivariable - Selected Features): RMSE and Normalised RMSE Comparison

Conversely, the upper CI of 75 indicated a similar degree of confidence that the actual residuals were below this value.

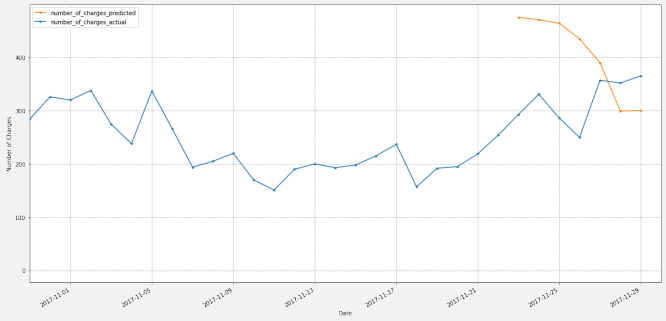
7.3.5 Cross Validation Analysis

We applied Algorithm 4 to each data split (shown in Table 5.4) to determine how the size and pattern of each data set affect the GRU model and hence the predicted results. Each GRU model per data split was trained, using the respectively optimised hyperparameters, which resulted in a seven-day prediction per split. Table 7.7 showed the actual and predicted total number of charges per prediction period of 7 days with the respective point difference comparison and the RMSE. A visual representation of the actual and predicted charges can be seen in Figure 7.9. In the figures, the actual number of charges (blue) were plotted on a daily basis over a period of 30 days, with the predicted values (orange) plotted on the last seven days of the defined period.

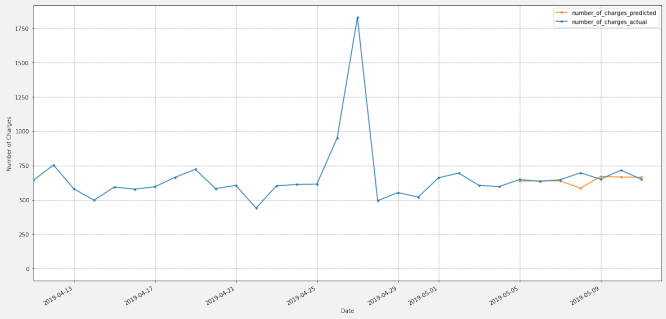
To compare the RMSE across splits, we calculated the normalised RMSE, as shown in Table 7.8. Split 05 produced the lowest normalised RMSE (0.054), indicating that this split produced the best prediction values vs actual values in comparison to all splits. Graphically, the RMSE and the normalised RMSE per split for the multivariable GRU (with selected features) can be seen in Table 7.10.



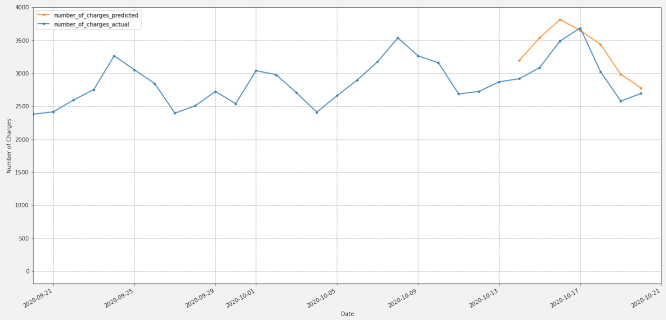
(a) Split 1 Comparison (RMSE: 3.217)



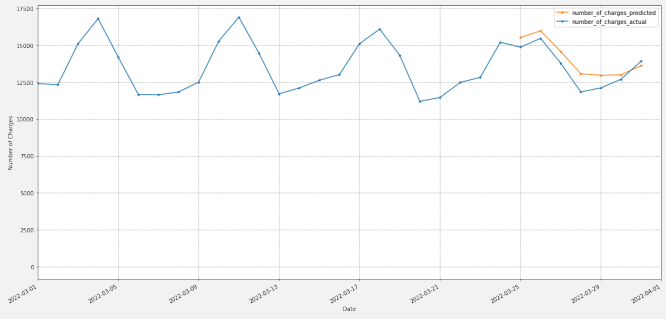
(b) Split 2 Comparison (RMSE: 134.514)



(c) Split 3 Comparison (RMSE: 47.488)



(d) Split 4 Comparison (RMSE: 323.600)



(e) Split 5 Comparison (RMSE: 731.009)

Figure 7.9: GRU (Multivariable - Selected Features) Cross Validation: Actual vs Predicted Charges Results per Split

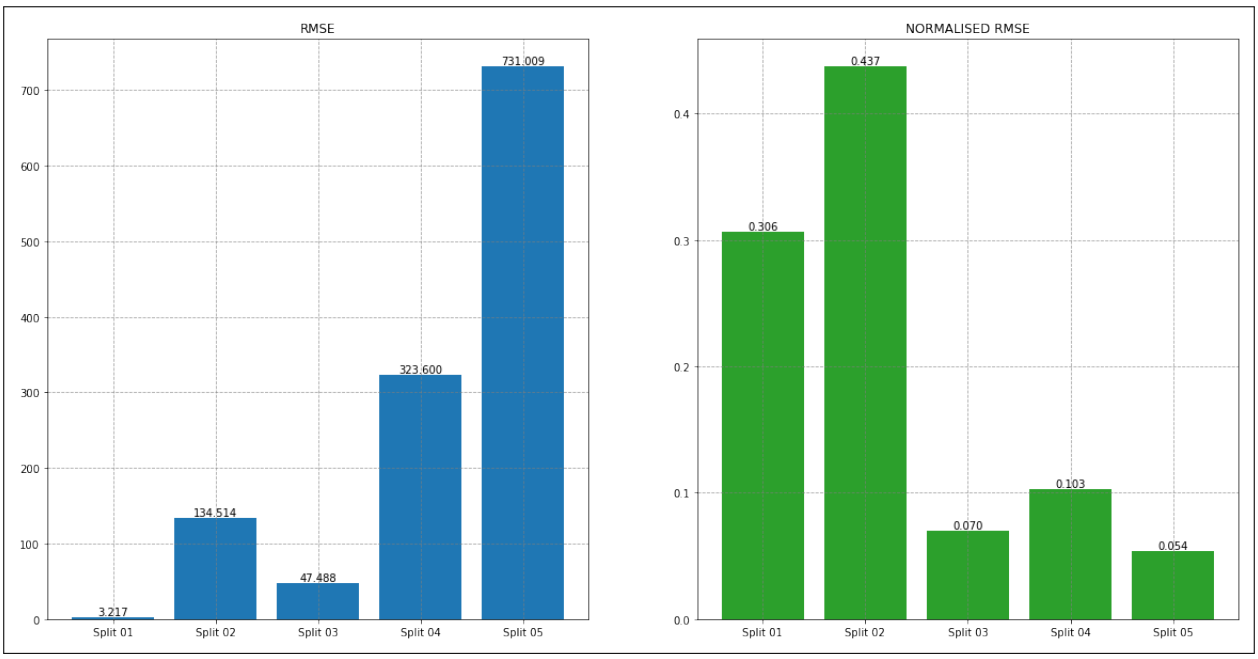


Figure 7.10: GRU (Multivariable - Selected Features): RMSE and Normalised RMSE Visual Comparison

No	Model	Hyperparameters	Time (hh:mm:ss)
01	LSTM	8	02:22:01
02	GRU	8	04:24:54

Table 7.9: Time taken to find the optimal hyperparameters for the Multivariable Analysis

7.4 Results Comparison

Once the experiments were completed and each of the models predicted the number of charges for the last 7 days in March 2022, we were then able to compare the results per model. In these analyses, we considered only the number of charges as well as the data points from three additional features (private charges, session duration and new contracts) that were registered per day from August 2014. The total data set consisted of 2,639 data points per feature, where the last 7 data points were considered as the test data. Both models (LSTM, and GRU) used the training data to determine the best hyperparameters required to produce the best test result. Once these hyperparameters were defined, each model then predicted the number of charges over the next seven days which were then compared to the test data set.

7.4.1 Hyperparameter Optimization

Both LSTM and GRU models each had eight hyperparameters to optimise, and the respective time to find the optimised hyperparameters per model was shown in Table 7.9. As expected, finding the optimal hyperparameters for the ARIMA model took less time than the other two models - and the LSTM and GRU times are similar given the similarity of the models and the required hyperparameters.

7.4.2 Model Results

Once the hyperparameters were found, each model predicted the number of charges per day for a period of seven days. These were then plotted against the actual values of the charges over

Date	Actual Value	LSTM Predicted Value	GRU Predicted Value
Day 01	14,878	15,474.57	15,532.75
Day 02	15,476	15,563.38	15,981.80
Day 03	13,806	14,339.91	14,591.17
Day 04	11,844	12,759.47	13,080.78
Day 05	12,119	12,464.25	12,971.86
Day 06	12,703	12,900.81	13,007.11
Day 07	13,928	13,762.86	13,627.80
Total	**94,754**	**97,265.23**	**98,793.27**
Difference		**2,511.23**	**4,039.27**
% Diff		**2.65%**	**4.26%**
RMSE		**488.77**	**731.01**

Table 7.10: The Actual Number of Charges vs The Predicted Number of Charges per Model

those seven days, where the RMSE of each model result was calculated based on the actual charges, as shown in Table 6.10. On a total level, both LSTM and GRU **overpredicted** the actual values by **2.65%** and **4.26%**, respectively. When comparing the RMSE values, LSTM had the least amount of errors in its predictions in comparison to GRU. Figure 7.11 visually represented the results per model against the actual number of charges.

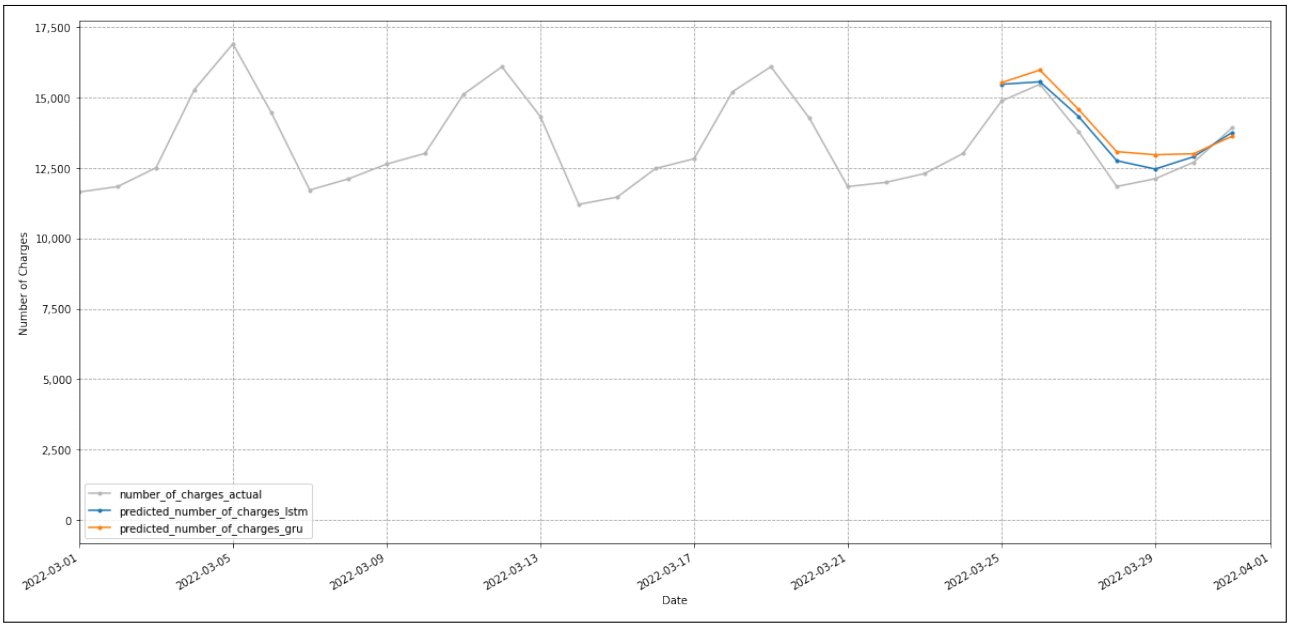


Figure 7.11: Prediction Results for LSTM and GRU

7.4.3 Residual Analysis Comparison

The investigation into residuals and their associated confidence intervals for multivariable (selected features) LSTM and GRU models provided distinct insights into their predictive performance. The LSTM model showcased a narrow range of residuals, with slightly looser confidence intervals, signifying accurate and consistent predictions closely aligned with actual values. In contrast, the GRU model's residuals were mostly negative, with confidence intervals around -75 and 75 (a larger range in comparison to the LSTM). These findings revealed the LSTM model's superior precision and the GRU model's higher variability in forecasting accuracy. These results are in alignment with the conclusions found in Table 7.10 with regard to the RMSE results.

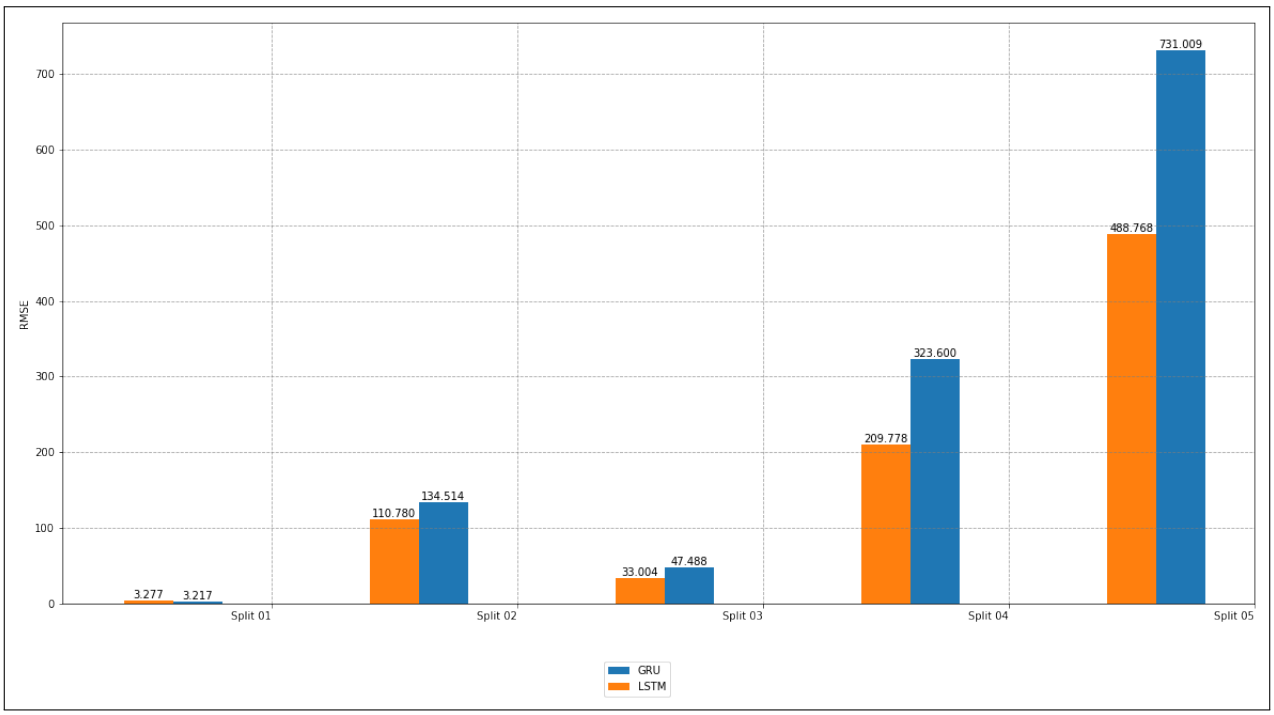


Figure 7.12: RMSE Comparison for LSTM and GRU

7.4.4 Cross Validation Results

The RMSE and normalised RMSE results per data split can be seen graphically in Figures 7.12 and 7.13. Whilst the GRU performed marginally better in Split 01, LSTM outperformed the GRU in every split that followed. To be able to compare the RMSE per model across every split, the RMSE values were normalised and on average, the LSTM models outperformed the GRU model. Furthermore, the normalised RMSE decreased as the data increased in size for all models and can be seen graphically in Figure 7.13.

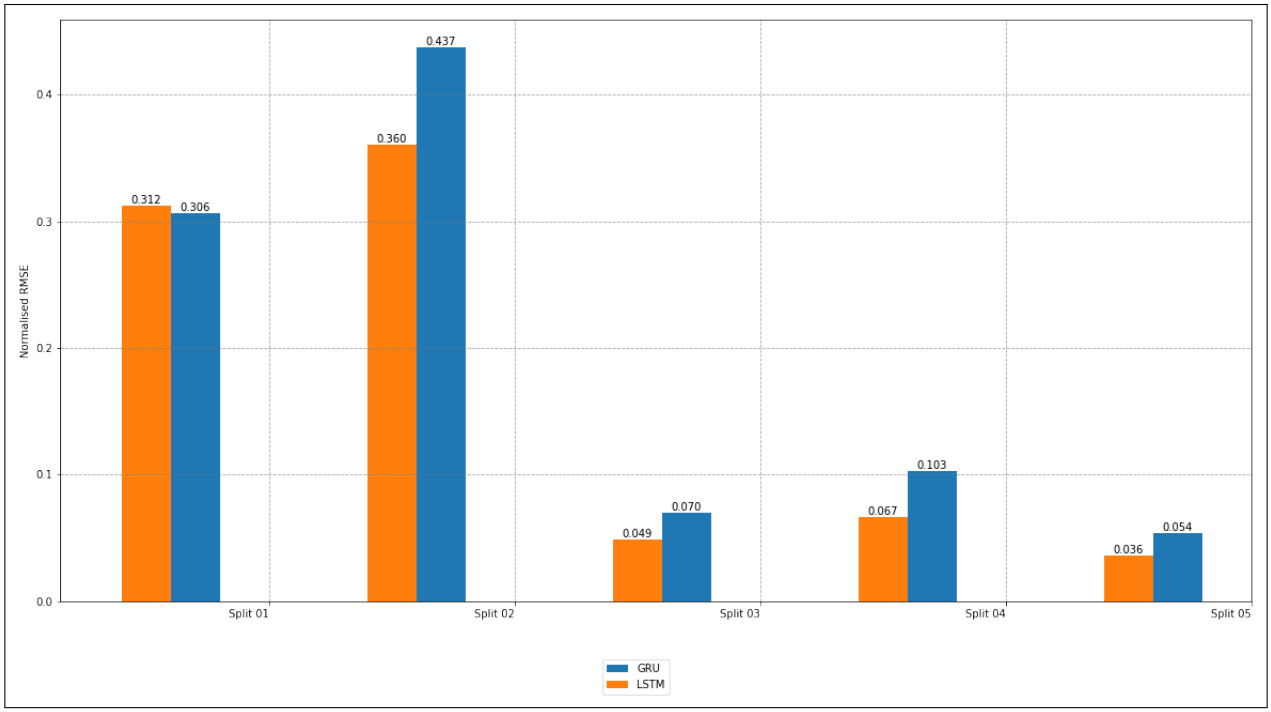


Figure 7.13: Normalised RMSE Comparison for LSTM and GRU

8 Comparative Analysis

8.1 Introduction

With the completion of all experiments, we were then able to compare the ARIMA, LSTM, and GRU models for univariable data, and the LSTM, and GRU models for multivariable data (all features and selected features).

To forecast the number of charges for seven days at the end of March 2022, the best-fitting models of ARIMA, LSTM, and GRU were utilized. Univariable models were trained using the number of charges, while multivariable models were trained using additional features, both dating from 2014, with a rolling window period of 14 days before the prediction period. The predicted values were then compared with the actual values to assess the models' predictive capability. To evaluate the prediction performance, RMSE was used, with lower values indicating better performance.

8.2 Results Analysis and Discussion

The results in Table 8.1 summarise important factors considered to compare the models. These factors include the number of features used, the number of hyperparameters that needed to be optimized, the number and duration of the experiments, and the values of actual vs expected values and RMSE. For the single variable analysis, the ARIMA model showed lower values than both the LSTM and GRU models, indicating the accuracy of the statistical model was better than the neural networks. ARIMA model produced a lower error value (356.070) than the LSTM and GRU models (460.362 and 531.225, respectively)in the seven-day prediction values which indicated that ARIMA was more successful than LSTM and GRU models forecasting.

In trying to determine whether the error values for the LSTM and GRU models could be improved, we included additional features to the data set. Initially, fourteen additional data features were added to find the best-fitting models and compare the error rates calculated from the forecasted values when compared to the actual values. The LSTM model produced error values higher than their single variable counterparts. For LSTM, the multivariable model produced an error rate of 627.454 in comparison to the LSTM single variable model's error rate of 460.362. The increase was likely caused by the over-fitting of the features. As seen in Chapter 4.4, the correlation analysis showed that the features were all very similar and overtrained the model on the training data deeming it unable to reasonably predict future daily charges. On the other hand, the multivariable GRU model produced a lower error rate of 514.998 in comparison to the GRU single variable model's error rate of 531.225. However, this difference is not significant and hence shows that the GRU can handle all the features, but could also do without them.

The fourteen features were then reduced to three and added to then find the best-fitting models for both LSTM and GRU. Here we saw that the LSTM with three features performed signif-

icantly better than the LSTM with all features (488.768 vs. 627.454 respectively). However, the GRU with three features performed worse than the GRU with all features (731.009 vs. 514.998). The reason for both results is that the selected features were a better input for the LSTM but not for the GRU.

When comparing all error rates, the ARIMA model had the lowest and hence, predicted better than all other models. Whilst we can state that the ARIMA model is the better predictor of the number of charges, it should be noted that the underlying data dictates the success or failure of a model. This was seen in the comparison of the GRU multivariable models (all features vs selected features). We could have achieved a better result with the GRU had we selected a different subset of features.

	Single Variable			Multivariable (All Features)		Multivariable (3 Features)	
	ARIMA	**LSTM**	**GRU**	**LSTM**	**GRU**	**LSTM**	**GRU**
Model Type	Statistical Model	RNN	RNN	RNN	RNN	RNN	RNN
Number of Features	0	0	0	14	14	3	3
Number of Hyperparameters	3	8	8	8	8	8	8
Number of Experiment Iterations	196	100	100	50	50	50	50
Hyperparameter Optimisation (hh:mm:ss)	00:01:02	02:03:51	02:34:54	02:22:01	04:24:54	01:06:35	01:43:41
Total % Difference	-0.02%	-0.57%	-1.83%	1.74%	-0.35%	2.65%	4.26%
RMSE	356.070	460.362	531.225	627.454	514.998	488.768	731.009
Normalised RMSE	0.026	0.034	0.039	0.046	0.038	0.036	0.054

Table 8.1: Model Summary: Comparison of Models

We conducted a comprehensive analysis of the performance of different algorithms for forecasting the number of charges. We not only compare the performance of the ARIMA, LSTM, and GRU models on the entire dataset, but we also applied cross-validation techniques to split the data into five parts. This allowed us to evaluate the performance of the models on different subsets of the data, providing a more robust assessment of their forecasting capabilities.

Cross-validation is a technique used in machine learning to assess the performance of a model by dividing the data into several subsets, training the model on one subset and evaluating its performance on the remaining subsets. In the case of time series data, it is important to use techniques such as time-series cross-validation or rolling window cross-validation to ensure that the temporal dependencies in the data are preserved during the splitting process.

The results of our analysis showed that the ARIMA model outperformed the LSTM and GRU models based on the RMSE values. The results can be seen in Table 8.2. The ARIMA model had the lowest RMSE values across all data splits, indicating that it was able to make more accurate predictions than the other models. Additionally, we also included the percentage difference between actual vs predicted values, and we found that the LSTM model did have lower percentage difference values for some splits.

		Single Variable			Multivariable (All Features)		Multivariable (3 Features)	
Split No	**Actual Charges**	**ARIMA**	**LSTM**	**GRU**	**LSTM**	**GRU**	**LSTM**	**GRU**
Number of Charges								
01	78	74.69	68.41	64.31	67.26	74.77	66.03	67.10
02	2,234	2,023.74	2,557.97	2,785.23	2,440.83	2,567.44	2,660.13	2,834.51
03	4,644	4,600.44	4,637.06	4,608.74	4,481.73	4,487.33	4,533.39	4,499.81
04	21,479	22,298.25	21,982.67	22,604.39	22,699.49	22,256.38	22,330.90	23,402.38
05	94,754	94,731.24	94,211.69	93,019.86	96,400.9,	94,425.24	97,265.23	98,793.27
% Difference (Actual and Predicted)								
01		-0.04%	-0.12%	-17.55%	-13.77%	-4.14%	-15.35%	-13.97%
02		-9.41%	14.50%	24.67%	9.26%	14.93%	19.07%	26.88%
03		-0.94%	-0.15%	-0.76%	-3.49%	-3.37%	-2.38%	-3.10%
04		3.81%	2.34%	5.24%	5.68%	3.62%	3.97%	8.95%
05		-0.02%	-0.57%	-1.83%	1.74%	-0.35%	2.65%	4.26%
RMSE								
01		2.684	2.968	2.905	3.400	2.883	3.277	3.217
02		49.154	90.931	122.184	84.237	99.125	110.78	134.514
03		28.407	31.142	28.677	39.317	40.069	33.004	47.488
04		181.567	206.143	219.744	243.153	210.957	209.778	323.6
05		356.07	460.362	531.225	627.454	514.998	488.768	731.009
Normalised RMSE								
01		0.256	0.283	0.277	0.324	0.275	0.312	0.306
02		0.160	0.296	0.397	0.274	0.322	0.360	0.437
03		0.042	0.046	0.042	0.058	0.059	0.049	0.070
04		0.058	0.066	0.070	0.078	0.067	0.067	0.103
05		0.026	0.034	0.039	0.046	0.038	0.036	0.054

Table 8.2: Model Summaries: Result Comparison per Data Split

Given our results, we saw that ARIMA was a better predictor for stationary time series data. One reason for this was that ARIMA models were specifically designed for time series data and were able to take into account the temporal dependencies in the data. These models were able to identify patterns in the data and make predictions based on those patterns. In contrast, LSTM and GRU models were generally used for non-time series data and may not have been as well-suited for this type of data.

Another reason why ARIMA may have been a better predictor was that it was a simpler model than LSTM and GRU. This made it easier to understand and interpret the results of the model. Additionally, simpler models can be less prone to overfitting, which can lead to more accurate predictions.

It was important to note that the choice of model would depend on the specific data and forecasting problem at hand. For example, if the time series data is non-stationary, other methods such as exponential smoothing may have been more appropriate. Additionally, if the data has high levels of noise or irregular patterns, more complex models such as LSTM or GRU may have been necessary. However, for stationary time series data - as we have used in this dissertation, ARIMA was often a good choice as it was simple, well-understood, and able to capture temporal dependencies in the data.

8.3 Impact of Results from a Business Perspective

Forecasting important Key Performance Indicators (KPI) for a shorter period, such as within the next few weeks, has several advantages compared to long-term planning or forecasting for the next few years. Specifically, in this dissertation, we looked at the short-term forecasting of the number of charges - and we forecasted 7 days into the future.

One advantage of short-term forecasting is that it allows for more accurate predictions. Since we considered a short time horizon, fewer unknown factors could affect the outcome and make it easier to identify patterns and trends in the data. This we saw with the cross validation results that included the COVID-19 data set. This affected the data set a lot less than expected given that we only forecasted one week. Additionally, short-term forecasts can be updated more frequently, and given the short training and prediction times of the models, the results could be produced when necessary and hence can prove to be beneficial when making quick decisions.

Another advantage of short-term forecasting is that it can help identify and respond to potential issues more quickly. For example, if The Company does see a decline in the number of charges (which has a direct impact on revenue) in the short-term forecast, it can take immediate action to address the issue before it becomes a larger problem. Given that The Company is still fairly new to the market, it still can make short-term adjustments.

Whilst this short-term forecasting is unable to assist with making strategic decisions based on long-term trends, having a week-by-week estimation of the number of charges will help in the day-to-day operations to enable quick decision-making that could in turn allow The Company to meet their long-term goals.

In this dissertation, we aimed to analyze the forecasted values of seven days, specifically the last week of March 2022. We found that the ARIMA model was the most effective in predicting

these values, and we, therefore, decided to see what this same model would predict up until December 31, 2022. A summary of the actual and predicted charges for each month (including the originally predicted seven days) has been summarised in Table 8.3.

Date From	Date To	Actual Charges	Predicted Charges	RMSE	Difference	Perc Diff
2022-03-25	2022-03-31	94,754	94,731.24	356.070	-22.76	-0.024%
2022-04-01	2022-04-30	465,891	409,835.75	2,199.035	-56,055.25	-12.032%
2022-05-01	2022-05-31	471,451	424,692.71	2,120.783	-46,758.29	-9.918%
2022-06-01	2022-06-30	495,768	416,351.7	2,907.180	-79,416.30	-16.019%
2022-07-01	2022-07-31	616,332	435,641.31	6,060.390	-180,690.69	-29.317%
2022-08-01	2022-08-31	602,584	439,570.97	5,514.946	-163,013.03	-27.052%
2022-09-01	2022-09-30	589,215	430,421.88	5,606.242	-158,793.12	-26.95%
2022-10-01	2022-10-31	643,093	449,400.9	6,560.519	-193,692.10	-30.119%
2022-11-01	2022-11-30	663,804	439,423.33	7,718.856	-224,380.67	-33.802%
2022-12-01	2022-12-31	787,850	458,886.51	10,996.032	-328,963.49	-41.755%

Table 8.3: Monthly Charging Predictions for 2022 (Original Model)

Our analysis revealed that while the ARIMA model was capable of predicting the number of charges that far in advance, it was only able to do so with reasonable accuracy for the immediate three months. For April, May and June, the RMSE remained below 3,000 with a percentage difference of under roughly 16%. Given that these results are forecasted, a general acceptance criteria of 15% difference has been used at The Company. The number of charges until June has been shown in Figure 8.1. After July, however, the RMSE increased quite substantially - as the predicted values were not in line with the actual values. This was because the trained did not represent the future data set as well as it represented the validation data - and the decay in the pattern of the predicted values was evident hereof. This does not mean that the model we used was insufficient, but rather that the future data had different intrinsic patterns that were not substantially represented in the historic data.

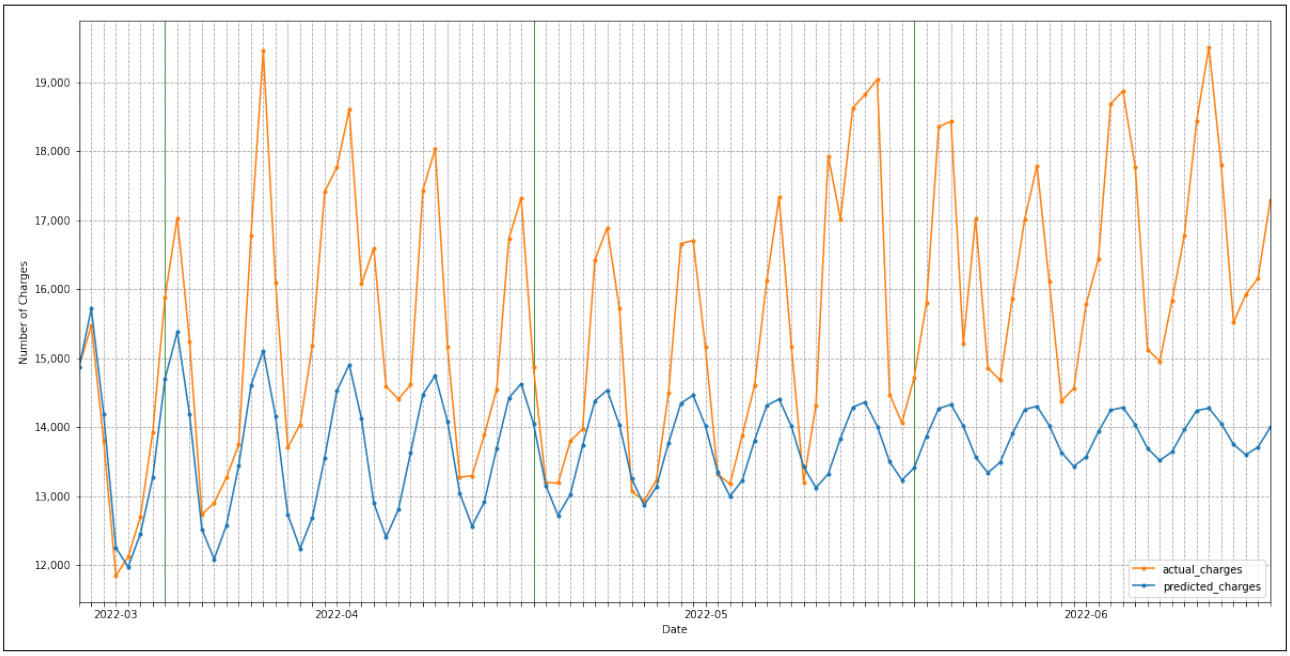


Figure 8.1: Visual Representation of Monthly Charging Predictions for 2022 (Original Model)

Our analysis also showed that the actual data revealed a significant increase in the number of charges compared to the predicted model. While this may have been attributed to external factors such as market growth or new goals set internally by The Company, it highlighted the

importance of updating the model to incorporate any changes that may impact the number of charges.

Therefore, we concluded that the ARIMA model could be used to predict the number of charges up to a year in advance, with the understanding that the results were dependent on the current situation. After three months, it would be necessary to update the model with the latest information if we wished to stay on track with our predictions.

To test this hypothesis, we decided to update the data and run the ARIMA model again, this time incorporating data up until June 30, 2022. A summary of the actual and predicted charges for each month (where the seven-day prediction had been moved to the last week in June) has been summarised in Table 8.4.

Date From	Date To	Actual Charges	Predicted Charges	RMSE	Difference	Perc Diff
2022-06-24	2022-06-30	120,658	117,578.21	536.356	-3,079.79	-2.55%
2022-07-01	2022-07-31	616,332	526,157.55	3,070.839	-90,174.45	-14.63%
2022-08-01	2022-08-31	602,584	523,565.40	2,681.542	-79,018.61	-13.11%
2022-09-01	2022-09-30	589,215	515,138.74	2,719.132	-74,076.26	-12.57%
2022-10-01	2022-10-31	643,093	535,742.96	3,725.959	-107,350.04	-16.69%
2022-11-01	2022-11-30	663,804	519,909.42	4,980.472	-143,894.58	-21.68%
2022-12-01	2022-12-31	787,850	544,816.35	8,332.265	-243,033.65	-30.85%

Table 8.4: Monthly Charging Predictions for 2022 (Updated Model)

Our results showed that updating the model with three months of data did improve the prediction of the number of charges until the end of 2022. This analysis revealed that July, August and September had lowered their RMSE values in comparison to the first run - and also showed that the percentage difference between actual and expected had fallen below the recommended 15%. The number of charges until September has been shown in Tables 8.2. This showed us that the model was still most effective in predicting the number of charges for the immediate three months, and also that the results for October, November, and December were better in the second run compared to the first. This could be an indication that customer charging behaviour had stabilized toward the end of the year, which allowed the model to predict the values more accurately. Overall, this analysis showed that while the ARIMA model was effective in predicting the number of charges, it needed to be updated regularly to ensure that it continued to accurately reflect the current situation.

8.4 Research Findings and Contributions

The present study investigated the efficacy of machine learning algorithms in predicting customer charging behaviour in the electric mobility industry. By focusing on European customers with electric vehicles, the research aimed to enhance the accuracy of charging behaviour forecasts and provide valuable insights for the sustainable growth of the electric vehicle ecosystem.

In Subsection 2.8, we formulated a primary research question and subsidiary research questions to investigate the efficacy of machine learning algorithms for predicting customer charging behaviour in the electric mobility industry. In light of the comprehensive results and analyses presented in the previous sections, we will now proceed to provide a concise summary, aligning our findings with the respective research questions.

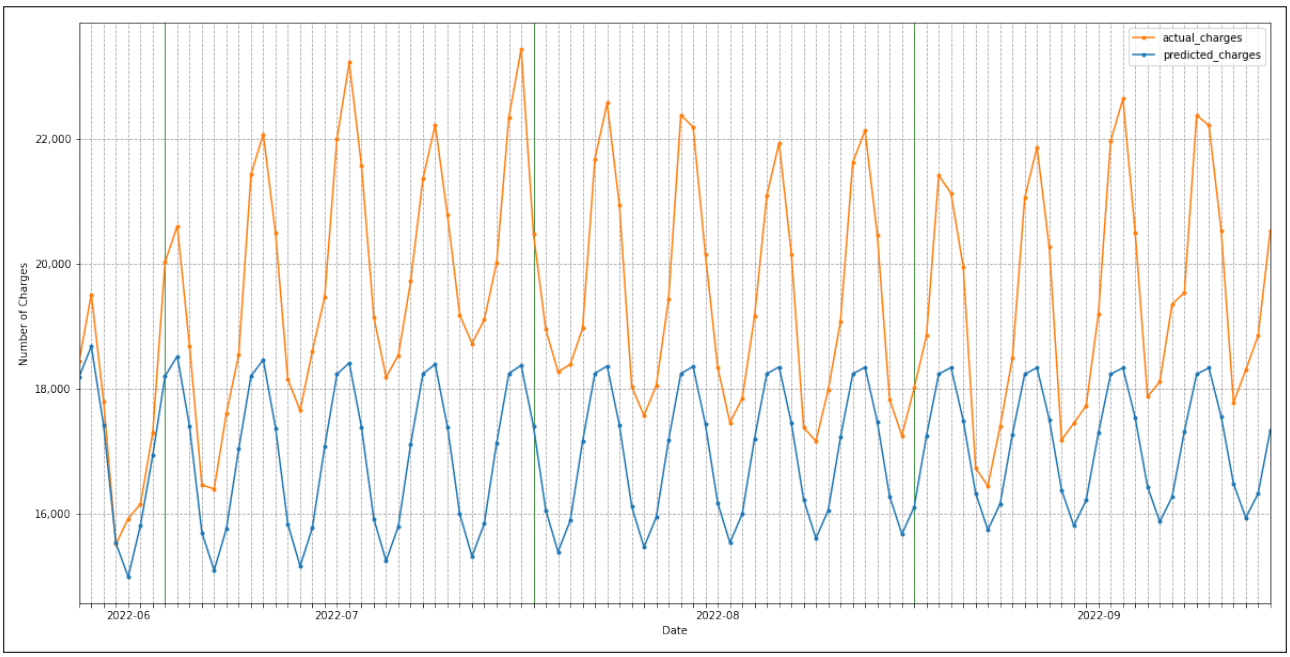


Figure 8.2: Visual Representation of Monthly Charging Predictions for 2022 (New Model)

RQ1: Which algorithm (ARIMA, LSTM, or GRU) provides the most accurate predictions for the number of electric charges daily over 7 days by European customers with electric vehicles?

We conducted a comprehensive analysis of the performance of the ARIMA, LSTM, and GRU models on the time series data of electric charges by European customers with electric vehicles. The evaluation metric used was the root mean squared error (RMSE), which measures the accuracy of predictions by calculating the difference between actual and predicted values. The results were as follows:

- ARIMA Model: RMSE = 356.070
- LSTM Model: RMSE = 460.362
- GRU Model: RMSE = 531.225

Based on the results, the ARIMA model outperformed both LSTM and GRU models, exhibiting the lowest RMSE value. This indicates that ARIMA provided the most accurate predictions for the number of electric charges in the given dataset.

RQ2: What are the strengths and weaknesses of each algorithm in predicting customer charging behaviour based on a single variable data set (number of charges) and a multivariable dataset (Inclusion of session duration, power usage, customer type, etc.)?

Single Variable Analysis

- ARIMA Model achieved the lowest RMSE value, indicating its strength in handling the time series data of electric charges by European customers based on the number of charges.
- LSTM Model produced a higher RMSE value compared to ARIMA, suggesting its weakness in handling the single-variable data.
- GRU Model performed the worst in comparison to LSTM and ARIMA.

Multivariable Analysis

- LSTM Model showed increased error values (higher RMSE) when the data included additional features (e.g., session duration, power usage, customer type). Overfitting of features may have contributed to the decrease in performance.
- GRU Model demonstrated a slight performance improvement when applied to the multivariable dataset, as indicated by a lower RMSE compared to its single-variable analysis.

RQ3: Did the addition of external features or data improve the prediction accuracy of LSTM and GRU models, and how did it compare to ARIMA's single variable performance?

LSTM Model with Additional Features showed a decrease in performance (higher RMSE) compared to its single variable counterpart, likely due to overfitting of the added features. GRU Model with Additional Features demonstrated a slight performance improvement (lower RMSE) when applied to the multivariable dataset, indicating better handling of the additional features compared to LSTM. The results from both LSTM and GRU models with additional features did not outperform ARIMA's single variable performance, which had the lowest RMSE. Thus, ARIMA remained the superior choice for forecasting customer charging behaviour based on this dataset.

RQ4: How do the algorithms compare in terms of computational efficiency and resource requirements for processing the given electric mobility data?

ARIMA Model required the fewest hyperparameters (3) and had the shortest hyperparameter optimization time (00:01:02). For the LSTM and GRU Models, both required more hyperparameters (8) and longer optimization times (LSTM: 02:03:51, GRU: 02:34:54). Hence, ARIMA was found to be computationally efficient, as it was a simpler model compared to LSTM and GRU, making it easier to interpret and implement.

RQ5: What are the implications of the study's findings for practical applications in the electric mobility industry?

The use of the ARIMA model is recommended for forecasting the number of electric charges by European customers with electric vehicles, as it demonstrated superior accuracy in the provided dataset. The results highlight the importance of considering the characteristics of the data and the specific forecasting task when selecting an appropriate algorithm. ARIMA's suitability for stationary time series data made it the preferred choice for this application. Practical decision-making processes, resource planning, and infrastructure optimization in the electric mobility sector can benefit from the accurate predictions provided by the ARIMA model. It should be acknowledged that the choice of the model ultimately depends on the nature of the data and the specific requirements of the forecasting task. Non-stationary time series data or data with high levels of noise might necessitate the use of alternative methods.

8.4.1 Reconsidering Benchmarking: Implications for Algorithm Selection

It is worth noting that this research did not incorporate a benchmark model for comparison alongside the evaluated algorithms. While benchmarks can provide a point of reference, their inclusion can sometimes introduce bias or oversimplification. In our study, the algorithms

were compared directly with each other, taking into account the specific characteristics of the dataset at hand. The focus remained on assessing the relative strengths and weaknesses of the algorithms in the context of electric vehicle charging behaviour prediction. The absence of a benchmark allows for a more nuanced understanding of algorithm performance and its applicability to real-world scenarios.

8.4.2 Initial Expectation and Unforeseen Results

Initially, there was a prevailing expectation that the LSTM model, with its capability to capture intricate temporal patterns, would outperform other algorithms. However, the research findings revealed a counterintuitive outcome—the ARIMA model surpassed both the LSTM and GRU models in predictive accuracy for the provided dataset. This unexpected outcome emphasizes the significance of empirical evaluation in algorithm selection, as theoretical assumptions may not always align with practical outcomes. The study's findings contribute to the nuanced understanding of algorithm performance and the importance of data-specific analyses.
By addressing these research questions, the study has contributed valuable insights into the effectiveness of machine learning algorithms in forecasting customer charging behaviour within the electric mobility industry. The research aids in resource optimization and practical applications within the field, advancing knowledge in electric mobility forecasting. The methodological approach and data transformations employed during the research process also offer guidance for future investigations in this domain.

8.5 Methodology Limitations and Considerations

While the selected algorithms (ARIMA, LSTM, GRU) were carefully chosen for their prominence in time series forecasting, it was important to recognize that the field encompasses a broader range of algorithms that were not explored in this research. These algorithms might have offered distinct advantages or faced specific challenges that were not accounted for in this study.

Furthermore, the feature selection process, including the incorporation of external factors such as COVID-19 data, introduced a layer of complexity. The chosen features represented a subset of potential variables influencing charging behaviour, potentially overlooking other crucial contributors. The single dataset used for training and evaluation also demanded attention, as dataset-specific noise and trends could have limited the models' ability to capture broader patterns accurately.

Generalizability concerns stemmed from the fact that the study's findings primarily relied on a specific dataset. Variations in geographical, temporal, or cultural contexts could have resulted in varying charging behaviours, thereby warranting caution when applying the findings to diverse scenarios. Assumptions inherent in predictive modelling, such as stationarity and linearity, introduced an additional layer of complexity, as these assumptions might not have held for all charging behaviour patterns.

The residual analysis provided valuable insights into model performance, but the interpretation of confidence intervals required careful consideration. Misinterpretation of confidence intervals could have led to inaccurate conclusions about the models' accuracy. Moreover, the methodology's focus on machine learning algorithms and historical data might have inadvertently disregarded influential external factors, such as evolving policies or technological advancements.

Finally, the choice of evaluation metrics, while appropriate for the study's objectives, invited reflection. Employing metrics beyond RMSE, such as MAE or MAPE, could have offered additional perspectives on model performance. Ensuring data quality was crucial, as the accuracy of model predictions hinged on the reliability of input data.

By acknowledging these limitations and considerations, this study aimed to foster transparency, enhance the interpretation of results, and provide a balanced assessment of the research methodology's strengths and weaknesses. These insights prompted critical evaluation and paved the way for future research to address these challenges and contribute to the evolving field of electric mobility forecasting.

8.6 Future Work

This study has provided valuable insights into the prediction of electric vehicle charging behaviour using various machine learning models. Beyond the current scope, several promising areas for future research could advance the field and contribute to the sustainable growth of the electric mobility industry.

1. **Hyperparameter Optimization:** Further investigation into fine-tuning hyperparameters for each model could yield improved forecasting accuracy. This might involve exploring more advanced optimization techniques or developing automated methods for parameter selection.

2. **Ensemble Methods:** An interesting avenue is to explore ensemble methods that combine the strengths of multiple models. Ensembling techniques like stacking or blending could potentially enhance predictive performance by leveraging the complementary aspects of individual models.

3. **Time Series Decomposition:** Applying time series decomposition techniques, such as Seasonal and Trend decomposition using LOESS (STL), could help in capturing underlying patterns and seasonality more effectively, potentially leading to improved forecasting results.

4. **External Factors:** Inclusion of additional external factors beyond the COVID-19 dataset could be explored. Factors like weather conditions, social events, or public holidays might contribute to a more comprehensive model that better captures real-world complexities in the electric mobility sector.

5. **Long-Term Forecasting:** Extending the forecasting horizon beyond the 7 days could provide insights into longer-term trends and help in planning for more extended timeframes in the context of electric vehicle charging behaviour.

6. **Transfer Learning:** Investigating the potential of transfer learning from related domains could be beneficial. Transferring knowledge from similar time series forecasting tasks might help in enhancing the predictive capabilities of the models, specifically tailored for the electric mobility industry.

7. **Model Interpretability:** Enhancing the interpretability of machine learning models could be crucial for practical deployment in the electric mobility sector. Exploring techniques for explaining model predictions, especially in complex models like LSTM and GRU, could facilitate better decision-making and resource allocation.

8. **Dynamic Feature Engineering:** Implementing dynamic feature engineering techniques that adapt to changing patterns and trends in the data could lead to more robust and adaptive forecasting models for electric vehicle charging behaviour.

9. **Deployment in Real-World Scenarios:** Testing the models in real-world scenarios within the electric vehicle charging infrastructure could provide valuable insights into their practical utility and uncover potential challenges that might not arise in a controlled experimental setup.

10. **Hybrid Models:** The combination of traditional time series models (e.g., ARIMA) with deep learning models (e.g., LSTM) could harness the strengths of both approaches, potentially resulting in more accurate and stable predictions, tailored to the dynamic demands of the electric mobility industry.

Exploring these areas of future work could contribute to the continuous advancement of forecasting methodologies for electric vehicle charging behaviour. As the electric mobility industry strives for sustainable and efficient charging infrastructure, these research directions could provide actionable insights for optimizing resource allocation, enhancing operational efficiency, and supporting decision-making processes.

9 Conclusion

Based on historical data, the prediction of the likelihood of a future outcome is imperative in business as it highlights possible future risky scenarios that can be mitigated. Of course, merely predicting the future is not good enough, as the expected outcome should match the actual occurrence as time progresses. For a business like The Company, that means being able to assess whether there will be growth in the business over time. The simple measurement and trend analysis on the **number of charging sessions** dictates whether The Company is growing in line with the global markets.

In this dissertation, we aimed to predict the number of charges for the next seven days. To do this, we compared the performance of the ARIMA model, LSTM model, and GRU model. We first used a single-variable model, which uses only one variable to make predictions.

To begin the process, we first had to prepare the data. We collected the number of charges that were summarized on a daily basis from 17.08.2014 to 31.03.2022. We then preprocessed the data by using a window period of 14 days, to predict the charges for the next seven days. This data was then referred to as the input dataset. Data is typically split into a training and test set to evaluate the performance of a model. The training data is used to train the model and the test data is used to evaluate the performance of the model on unseen data. This helps in assessing the model's generalization capability and avoiding overfitting.

After preparing the data, we then used the training data in the experimentation. For each algorithm, we took the training data and split it once again into the experiment training data and the experiment validation set. This validation set was used to evaluate the performance of the model during training, and to avoid overfitting.

With the prepared data, we then had to optimize the hyperparameters for each algorithm. Hyperparameter optimization is the process of finding the best values for the parameters of a model. It is done to fine-tune the parameters of the model to improve its accuracy. The ARIMA model had 3 hyperparameters to optimize, while the LSTM and GRU models had 8 each.

After the experimentation was complete, we had the training data and the optimized hyperparameters per model. We then predicted the results for the next seven days, per algorithm and compared these figures to those of the test set, using the Root Mean Squared Error (RMSE) as a metric. The ARIMA model outperformed both LSTM and GRU models by having the lowest RMSE.

We then decided to include additional features to the data set to determine whether additional information could improve the prediction results. We only did this for LSTM and GRU to determine whether we could improve their performance. The process was the same as the above, with the only change being that the extra features were added to the data.

In our second analysis, we looked at how all 13 identified features predicted the number of charges for the next seven days. For this case, the multivariable GRU performed better than the multivariable all-features LSTM model. However, the RMSE for the multivariable all-feature GRU was still not better than the initial ARIMA RMSE.

In our third analysis, we selected 3 features as part of our experiment (private charges, session duration, and new contracts) to predict the number of charges. These features were found using feature analysis or Pearson's correlation. Here we saw that the RMSE for the LSTM with selected features performed much better than the GRU for selected features. LSTM selected features also did better than LSTM multivariable all features - but was still not better than the LSTM single variable model.

In all cases, the ARIMA model outperformed all the other models, despite the addition of features. This meant that the number of charges contained enough information to predict the next seven days of charges. ARIMA model is specifically designed for time-series data, it is a statistical model that captures the temporal dependencies in data, such as trends and seasonality. The model uses past observations to forecast future values, which makes it well-suited.

For each of the models above, we also considered the impact of cross validation of the different data sets and how these would impact the results. By using this technique, we could mitigate the issue of overfitting the model to the training data, given that the time series data has a temporal dependence. In time series cross validation, we split the data into five different subsets using k-fold cross validation and trained and evaluated each model on different subsets of the data. using the RMSE as a metric, we found that ARIMA also outperformed all other models when predicting the seven days of charges per data split.

One limitation of this dissertation that we initially identified was that we only predicted seven days. While a week is long enough to determine which algorithm is better for prediction, in terms of usefulness for actual problems, seven days of prediction may be considered short. It can aid in short-term planning but is less useful for long-term planning. Given that ARIMA had the lowest RMSE score when predicting the number of charges, the prediction analysis was extended to predict the number of charges until the end of December 2022. Our analysis revealed that the model was capable of predicting that far in advance; however, it was only able to do so with reasonable accuracy for the immediate three months that followed the training period. For April, May and June 2022, the RMSE remained below 3,000. The model was updated to include training data up to June 2022, and once again called to predict the number of charges until December 2022. This analysis revealed that July, August and September 2022 had lowered their RMSE values in comparison to the first run. This decrease highlighted that the updated data represented the "future data" better than it did in our first run. As a potential direction for future work, we could explore predicting more values with updated data sets as these contained intrinsic patterns more aligned with the predicted values. In general, more data is generally better for machine learning models, as it allows the model to better capture the underlying patterns in the data.

Another limitation of this thesis was that the data used was highly correlated and internal to the company. One option for expanding the data would be to bring in external data that can influence the number of charges. While we considered the effect of COVID-19, there may be other factors that can also affect these charges. If the included factors are more complex than the presented time-series data, then perhaps the LSTM and GRU models may perform better.

It is important to note that the best algorithm for a problem depends on the underlying data. ARIMA models are best suited for time-series data with a clear trend and seasonality. LSTM and GRU models are more suited for time-series data with more complex patterns, such as long-term dependencies or multiple seasonalities. Given the fact that we were able to simplify the number of charges into a data set with clear trend and seasonality, ARIMA best predicted the number of charges, returning the lowest RMSE value when compared to the LSTM and GRU models for all data splits.

Bibliography

[Adebiyi et al., 2014a] Adebiyi, A., Adewumi, A., and Ayo, C. (2014a). Stock Price Prediction using the ARIMA Model. *International Conference on Computer Modelling and Simulation*. https://doi.org/10.1109/UKSim.2014.67.

[Adebiyi et al., 2014b] Adebiyi, A. A., Adewumi, A. O., and Ayo, C. K. (2014b). Comparison of ARIMA and Artificial Neural Networks Models for Stock Price Prediction. *Journal of Applied Mathematics*, 2014. https://doi.org/10.1155/2014/614342.

[ArunKumar et al., 2021] ArunKumar, K. E., Kalaga, D. V., Kumar, C. M. S., Kawaji, M., and Brenza, T. M. (2021). Forecasting of COVID-19 using deep layer Recurrent Neural Networks (RNNs) with Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM) cells. *Chaos, Solitons & Fractals*, 146:110861. https://doi.org/10.1016/j.chaos.2021.110861.

[Azari et al., 2019] Azari, A., Papapetrou, P., Denic, S., and Peters, G. (2019). Cellular Traffic Prediction and Classification: A Comparative Evaluation of LSTM and ARIMA. pages 129–144.

[Bengio et al., 1993] Bengio, Y., Frasconi, P., and Simard, P. (1993). Problem of learning long-term dependencies in recurrent networks. *1993 IEEE International Conference on Neural Networks*, pages 1183 – 1188 vol.3. https://doi.org/10.1109/ICNN.1993.298725.

[Bergstra and Bengio, 2012] Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

[Boslaugh and Watters, 2008] Boslaugh, S. and Watters, P. (2008). *Statistics in a Nutshell: A Desktop Quick Reference*. In a Nutshell (O'Reilly). O'Reilly Media.

[Box and Jenkins, 1976] Box, G. E. P. and Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control*. San Francisco: Holden-Day.

[Brownlee, 2020] Brownlee, J. (2020). Ordinal and One-Hot Encodings for Categorical Data. https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/. Accessed: 2021-02-06.

[Budiharto, 2021] Budiharto, W. (2021). Data science approach to stock prices forecasting in Indonesia during Covid-19 using Long Short-Term Memory (LSTM). *Journal of Big Data*, 2021. https://doi.org/10.1515/10.1186/s40537-021-00430-0.

[Chen, 2019] Chen, J. (2019). Autoregressive Integrated Moving Average (ARIMA). https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp. Accessed: 2021-01-28.

[Cho et al., 2014] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. pages 1724–1734. https://doi.org/0.3115/v1/D14-1179.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *NIPS 2014 Workshop on Deep Learning, December 2014*.

[Ding and Qin, 2019] Ding, G. and Qin, L. (2019). Study on the prediction of stock price based on the associated network model of LSTM. *International Journal of Machine Learning and Cybernetics*, page 1307–1317. https://doi.org/10.1007/s13042-019-01041-1.

[ES and Bajaj, 2022] ES, S. and Bajaj, A. (2022). Hyperparameter Tuning in Python: a Complete Guide. https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide. Accessed: 2023-01-04.

[Guo et al., 2019] Guo, A., Jiang, A., Lin, J., and Li, X. (2019). Data mining algorithms for bridge health monitoring: Kohonen clustering and LSTM prediction approaches. *The Journal of Supercomputing*, page 932–947. https://doi.org/10.1007/s11227-019-03045-8.

[Guyon and Elisseeff, 2003] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3(null):1157–1182.

[Hastie et al., 2015] Hastie, T., Tibshirani, R., and Wainwright, M. (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. CRC Press.

[Heavy.AI, 2022] Heavy.AI (2022). Feature Selection Definition. https://www.heavy.ai/technical-glossary/feature-selection. Accessed: 2022-06-17.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-term Memory. *Neural computation*, 9:1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.

[Hyndman and Athanasopoulos, 2018] Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: Principles and Practice*. OTexts: Melbourne, Australia. OTexts.com/fpp2.

[IEA, 2021] IEA (2021). How global electric car sales defied Covid-19 in 2020. https://www.iea.org/commentaries/how-global-electric-car-sales-defied-covid-19-in-2020. Accessed: 2021-04-17.

[Jin et al., 2019] Jin, Z., Yang, Y., and Liu, Y. (2019). Stock closing price prediction based on sentiment analysis and LSTM. *Neural Computing and Applications*, 2020:9713–9729. https://doi.org/10.1007/s00521-019-04504-2.

[Jolliffe, 2011] Jolliffe, I. (2011). *Principal Component Analysis*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Joosery and Deepa, 2019] Joosery, B. and Deepa, G. (2019). Comparative Analysis of Time-Series Forecasting Algorithms for Stock Price Prediction.

[Karkare, 2018] Karkare, P. (2018). A 7 Minute Introduction to LSTM: Powerful deep learning algorithm widely used in sequence modelling. https://medium.com/x8-the-ai-community/a-7-minute-introduction-to-lstm-5e1480e6f52a. Accessed: 2021-01-31.

[Ke et al., 2019] Ke, K., Hongbin, S., Chengkang, Z., and Brown, C. (2019). Short-term electrical load forecasting method based on stacked auto-encoding and GRU neural network. *Evolutionary Intelligence*. https://doi.org/10.1007/s12065-018-00196-0.

[Kuhn and Johnson, 2018] Kuhn, M. and Johnson, K. (2018). *Applied Predictive Modeling, 1st ed. 2013, Corr. 2nd printing 2018 Edition*. Springer.

[Li et al., 2021] Li, N., Hu, L., Deng, Z.-L., Su, T., and Liu, J.-W. (2021). Research on GRU Neural Network Satellite Traffic Prediction Based on Transfer Learning. *Wireless Personal Communications*. https://doi.org/10.1007/s11277-020-08045-z.

[Malik, 2018] Malik, F. (2018). Understanding Auto Regressive Moving Average Model - ARIMA. https://medium.com/fintechexplained/understanding-auto-regressive-model-arima-4bd463b7a1b. Accessed: 2021-01-28.

[Murat et al., 2018] Murat, M., Malinowska, I., Gos, M., and Krzyszczak, J. (2018). Forecasting daily meteorological time series using ARIMA and regression models. *Int. Agrophys*, 2014:253–264. https://doi.org/10.1515/intag-2017-0007.

[ReportLinker, 2021] ReportLinker (2021). Electric Vehicle Charging Station Market by Level of Charging, DC fast Charging - Global Forecast to 2027. https://www.reportlinker.com/p05404284. Accessed: 2021-04-17.

[Shah et al., 2018] Shah, D., Campbell, W., and Zulkernine, F. (2018). A Comparative Study of LSTM and DNN for Stock Market Forecasting. *In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data)*. https://doi.org/10.1109/BigData.2018.8622462.

[Shahid et al., 2020] Shahid, F., Zameer, A., and Muneeb, M. (2020). Predictions for COVID-19 with deep learning models of LSTM, GRU and Bi-LSTM. *Chaos, Solitons Fractals*, 140:110212. https://doi.org/10.1016/j.chaos.2020.110212.

[Shelatkar, Tejas et al., 2020] Shelatkar, Tejas, Tondale, Stephen, Yadav, Swaraj, and Ahir, Sheetal (2020). Web Traffic Time Series Forecasting using ARIMA and LSTM RNN. *ITM Web Conf.*, 32:03017. https://doi.org/10.1051/itmconf/20203203017.

[Siami-Namini et al., 2018] Siami-Namini, S., Tavakoli, N., and Siami Namin, A. (2018). A Comparison of ARIMA and LSTM in Forecasting Time Series. pages 1394–1401. https://doi.org/10.1109/ICMLA.2018.00227.

[Snoek et al., 2012] Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, page 2951–2959, Red Hook, NY, USA. Curran Associates Inc.

[Talos, 2019] Talos, A. T. C. S. (2019). Talos. http://github.com/autonomio/talos.

[Torres et al., 2021] Torres, J. F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F., and Troncoso, A. (2021). Deep learning for time series forecasting: A survey. *Big Data*, 9(1):3–21. http://doi.org/10.1089/big.2020.0159.

[Wang et al., 2020] Wang, S., Zhao, J., Shao, C., Dong, C., and Yin, C. (2020). Truck Traffic Flow Prediction Based on LSTM and GRU Methods With Sampled GPS Data. *IEEE Access*, 8:208158–208169. https://doi.org/10.1109/ACCESS.2020.3038788.

[Yamak et al., 2019] Yamak, P., Yujian, L., and Gadosey, P. (2019). A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting. pages 49–55. https://doi.org/10.1145/3377713.3377722.