KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Mathematical Statistics

Second Cycle 30 credits

Evaluation of Machine Learning Methods for Time Series Forecasting on E-Commerce Data

A Collaboration with QLIRO AB

PETER ABRAHAMSSON
NIKLAS AHLQVIST

Stockholm, Sweden 2022

Abstract

Within demand forecasting, and specifically within the field of e-commerce, the provided data often contains erratic behaviours which are difficult to explain. This induces contradictions to the common assumptions within classical approaches for time series analysis. Yet, classical and naive approaches are still commonly used. Machine learning could be used to alleviate such problems. This thesis evaluates four models together with Swedish fin-tech company QLIRO AB. More specifically, a MLR (Multiple Linear Regression) model, a classic Box-Jenkins model (SARIMAX), an XGBoost model, and a LSTM-network (Long Short-Term Memory). The provided data consists of aggregated total daily reservations by e-merchants within the Nordic market from 2014. Some data pre processing was required and a smoothed version of the data set was created for comparison. Each model was constructed according to their specific requirements but with similar feature engineering. Evaluation was then made on a monthly level with a forecast horizon of 30 days during 2021. The results shows that both the MLR and the XGBoost provides the most consistent results together with perks for being easy to use. After these two, the LSTM-network showed the best results for November and December on the original data set but worst overall. Yet it had good performance on the smoothed data set and was then comparable to the first two. The SARIMAX was the worst performing of all the models considered in this thesis and was not as easy to implement.

Keywords

Thesis, Time Series, Machine Learning, E-commerce, Demand Forecasting, Multiple Linear Regression, SARIMAX, XGBoost, LSTM, Model Evaluation

Sammanfattning

Inom efterfrågeprognoser, och specifikt inom området e-handel, innehåller den tillhandahållna informationen ofta oberäkneliga beteenden som är svåra att förklara. Detta motsäger vanliga antaganden inom tidsserier som används för de mer klassiska tillvägagångssätten. Ändå är klassiska och naiva metoder fortfarande vanliga. Maskininlärning skulle kunna användas för att lindra sådana problem. Detta examensarbete utvärderar fyra modeller tillsammans med det svenska fintechföretaget QLIRO AB. Mer specifikt en MLR-modell (Multiple Linear Regression), en klassisk Box-Jenkins-modell (SARIMAX), en XGBoost-modell och ett LSTM-nätverk (Long Short-Term Memory). Den tillhandahållna informationen består av aggregerade dagliga reservationer från e-handlare inom den nordiska marknaden från 2014. Viss dataförbehandling krävdes och en utjämnad version av datamängden skapades för jämförelse. Varje modell konstruerades enligt deras specifika krav men med liknande *feature engineering*. Utvärderingen gjordes sedan på månadsnivå med en prognoshorisont på 30 dagar under 2021. Resultaten visar att både MLR och XGBoost ger de mest pålitliga resultaten tillsammans med fördelar som att vara lätta att använda. Efter dessa visar LSTM-nätverket de bästa resultaten för november och december på den ursprungliga datamängden men sämst totalt sett. Ändå visar den god prestanda på den utjämnade datamängden och var sedan jämförbar med de två första modellerna. SARIMAX var den sämst presterande av alla jämförda modeller och inte lika lätt att implementera.

Nyckelord

Examensarbete, tidsserier, maskininlärning, e-handel, efterfrågeprognoser, multipel linjär regression, SARIMAX, XGBoost, LSTM, modellutvärdering

”If you torture data long enough, it will confess”
- Ronald H. Coase

Acknowledgements

The authors of this thesis woud like to raise a big thanks to our supervisor Bin Han and examiner Camilla Haldén at KTH for your help and support during this thesis. Another big thanks to Marcus Walldén at QLIRO for helping us setting up this thesis project and for your commitment during the course of its progress, it has been of tremendous help for us. The authors also want to raise a big thank you to Jiarui Xiong at QLIRO for sharing your insights within the field of machine learning. Lastly the authors also wants to raise a big thanks to Andreas Markerud and the entire QLIRO organisation for welcoming us and integrating us within the organisation in such a friendly manner. There has been a lot of pleasant talks, lunch breaks and shared insights that will be remembered.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For companies to be competitive, accurate demand forecasting of has always been an important aspect and it is especially true for demand forecasting within e-commerce. The literature presents a vast variety of different methodologies and tools for making good and accurate forecasts, and there does not seem be any consensual way of approaching time series. It is common to use naive approaches such as simply looking at sales from previous weeks, using methods such as rolling statistics or using classical methods such as linear regression or Box-Jenkins methods. Modern statistical learning methods has also seen increased interest due to their effectiveness in other areas of classifications and regression tasks. However, these modern methods shows promise, it is discussed that their performance do not make up for their complexity [2, 3, 18].

In collaboration with QLIRO AB, this thesis will investigate how different times series forecasting models perform. What will be of interest is to investigate how modern statistical methods fare on QLIRO's internal data. QLIRO has provided data containing the aggregate total reservation volumes from different e-merchants. Reservation volumes is the amount that is reserved during an online purchase that are to be handled by QLIRO. What makes this data set especially interesting is that it exhibits several seasonal components with different periodicity, a non-linear trend, as well as extreme values during sales periods, e.g. during Black Friday.

1.1 Background

QLIRO is a Swedish fin-tech company that offers a variety of different financial services to e-commerce merchants and retail customers. They are among other things active within the ”buy now, pay later” segment meaning that online customers have the option to pay in installments over time instead of the full amount directly. They have a diverse workforce representing over 30 different nationalities and serve over four million customers in the Nordics.

E-commerce has seen an upward trend in terms of total transactions. In 2020, e-commerce reached an all-time high of 18% of total global retail sales [28]. In Sweden, e-commerce saw an increase from 11% to 14% of the total retail sales between 2019 and 2021. It is also believed that this trend will continue to grow as e-commerce becomes more adopted among different age groups [29]. There has been a case study of a fast-fashion retail company that adapted towards an e-commerce oriented operations strategy during the pandemic of 2020. The results showed increased profitability and overall decrease of operational costs within the company [4]. There has also been an increasing trend and popularity in using ”buy now, pay later” services [12]. This shows that online purchasing is highly relevant and that it is on the rise to become even more prominent in the future.

As mentioned in the beginning of this chapter, QLIRO is interested in evaluating different models based on their ability to forecast future purchase volumes based on reservation volumes. More specifically, QLIRO wants to investigate how classic time series and regression models (e.g., Multiple Linear Regression and Box-Jenkins Methods), tree-based models (e.g., extreme gradient boosting trees), and Neural Network models perform and compare their results. Research in the area of forecasting within e-commerce is promising and previous research in the field is outlined in chapter 2. However, to the knowledge of the authors, there is a lack of similar research on these types of models on e-commerce within the Swedish market and the Nordics overall. E-commerce data itself is said to be highly dynamic and volatile due to temporal effects such as sales periods and holiday periods but also other aspects such as competitor behaviours and supply chain disruptions [3]. These effects can be globally and locally induced which makes it interesting to discover how such models behaves on Swedish e-commerce data.

1.2 Research Question

As the research of modern statistical methods within Swedish e-commerce is limited and with the primary objective of evaluating QLIRO AB's internal reservation volumes data, this thesis strives to answer:

- *How does modern statistical learning methods fare on QLIRO AB's reservation volume data?*

1.3 Purpose

The purpose of this thesis is to construct and evaluate different forecasting models on data provided by QLIRO. These models will then be evaluated and discussed based on their performance and their usability for QLIRO. Thus the expected outcome is an analysis that could be used internally at QLIRO and potentially provide a basis for continued work.

1.4 Relevance

As there are still discussions whether to use classical or modern statistical methods within demand forecasting, and that previous work of evaluating forecasting models on Swedish e-commerce data is limited; this thesis can be considered relevant. Also since this project aims to evaluate for an external stakeholder, this project can be considered unique. Accurate forecasting models are helpful since they simplify internal planning. Thus this thesis provides benefits for both academia and the private sector.

1.5 Stakeholders

The stakeholders of this thesis are QLIRO AB and KTH Royal Institute of Technology.

1.6 Outline

The rest of this thesis is organised as follows. In chapter 2 previous work in the field of implementing machine learning models on time series forecasting is presented.

Explanation of mathematical theory and concepts that are necessary to understand the methodologies and results of this project is carried out in chapter 3. Chapter 4 describes how the different models are implemented, what features are used, and the data processing required for each of the models. Results of the implemented models are presented in chapter 5. In chapter 6 the findings are discussed together with limitations, and future work is presented. Lastly, a conclusion of the thesis is presented in chapter 7.

Chapter 2

Previous Work

Despite the recent successes of machine learning models within classification tasks such as image recognition, the use of machine learning models within the domain of time series forecasting has not so far been considered competitive. Different types of architectures have been applied to various time series aspects and shown success, but in terms of forecasting it is debated that complex models does not provide a good enough performance to make up for their complex setup [2].

This reflects how the current setup of forecasting models are within demand forecasting, and for e-commerce in particular. Companies are still inclined to use naive approaches, classical Box-Jenkins methods or linear regression models to make forecasts due to their simplicity and ease of interpretation [3]. But as of late, the rise of machine learning models and novel application of these in other fields has led to an increased interest within forecasting as well. Since the field of demand forecasting usually contains time series which are erratic and discontinuous and therefore often violate assumptions of Gaussian errors, stationarity and homoscedasticity, classical models usually needs extensive data pre-processing and removal of important information [2, 3]. These non-linear problems could potentially be captured by machine learning methods such as deep learning to construct more accurate forecasts. This is discussed in [18] where they go through a framework of using deep learning methods within forecasting and compare them to classical models. Such deep learning models are built using Recurrent Neural Networks, or more specifically, Long Short-Term Memory Networks and they show good results on various data from the different M-competitions. Furthermore, the increased usage of data and data collection

is something that has also provided better learning foundations for deep learning methods as they are able to learn from long-term patterns within the data [18].

When it comes to tree learning methods, the Extreme Gradient Boosting tree, also known as XGBoost, has been shown to yield good results when used in various situations both within classification and regression tasks. Another popular decision tree method is the Random Forest which also shows good results. But between these aforementioned tree methods, the XGBoost is usually favoured in various competitions as it tends to yield better results at the cost of longer computational time [14, 32]. Other work has been done by utilising the combined predictability of ensembles for various machine learning methods, combining both classical- and state of the art models. These ensembles could either utilise meta-features using decision trees, e.g., a random forest, or simply taking the average of model outputs which is considered to be a very competitive option [2, 20].

Within e-commerce, Bandara et al. has constructed a deep learning model using a Long Short-Term Memory network to forecast e-commerce sales by using data from Walmart [3]. By utilising different time series based on sales for different product segments, they constructed a global model that works for all of these series and learns from similar behaviours by grouping those that shows similar patterns. Their results are very promising and could help reduce complexities since each time series would need an independent model if a classical framework were to be used whereas now only one model is needed. Inspired by this, Rémy Garnier and Arnaud Belletoile used a similar approach but instead constructed an XGBoost which showed good results using it on data provided by French e-commerce actor Cdiscount [14]. Based on the previous work, together with what was introduced in chapter 1, this thesis will further investigate a Multiple Linear Regression model, a Box-Jenkins model that utilise a Multiple linear Regression model called SARIMAX, an extreme gradient boosting tree and a Long Short-Term Memory Network.

Chapter 3

Theoretical Background

3.1 Time Series

In this section basic concepts, definitions, and components of a time series will be explained to provide a basis of theoretical knowledge to lay the other parts of this chapter upon.

3.1.1 Definition of a Time Series

A time series is a set of observations $x_t, t = 0, 1, 2 \ldots$ where x_t is a sample from a random variable and the observations are sequentially arranged in chronological order based on time. Time series can be discrete, where the observations are made at discrete time-steps, or they can be continuous where observations are recorded continuously over some time interval [5]. In this thesis a discrete univariate time series with equidistant time steps will be considered.

3.1.2 Components of a Time Series

A time series typically consists of four types of variation. These components can sometimes be separated from the time series and and analysed in isolation. These components are: (a) seasonality effect, (b) other cyclical changes, (c) trend, and (d) other irregular fluctuations. below follows a short description of the different sources of variation [8].

(a) *Seasonal effect*

Many time series exhibits variation which is periodic. For example sales of ice cream might be higher during the summer than during the winter.

(b) *Other cyclical changes*
Some time series exhibits oscillations which are non-periodic but might still be predictable. E.g. Black Week, which is a period where sales are generally much higher than during the rest of the year and does not occur with a fixed period, but is still a predictable event.

(c) *Trend*
Many time series exhibits a long-term change in the mean level. This is referred a time series trend.

(d) *Other irregular fluctuations*
These variations are often random noise and are often modelled through some white noise process.

Time series in general are built up of some or all of the components presented in (a)-(d). These may or may not be linearly related to the time series.

3.1.3 Stationarity

The models in section 3.3 requires that the time series is at least weakly stationary. Going forward weakly stationary and staionary will be used interchangeably. The formal definition of (weakly) stationarity follows below.

Let X_t be a time series with $E[X_t^2] < \infty$. The mean function of X_t is

$$\mu_X(t) = E[X_t]. \tag{3.1}$$

The covariance function of X_t is

$$\gamma_X(r,s) = \text{Cov}[X_r, X_s] = E[(X_r - \mu_X(r))(X_s - \mu_X(s))], \quad \forall\, r,s \in \mathbb{z}. \tag{3.2}$$

A time series is said to be weakly stationary if

(i) $\mu_X(t)$ is independent of t,

(ii) $\gamma_X(t+h,t)$ is independent of t for each h.

3.2 Multiple Linear Regression

Multiple Linear Regression is a model which uses k independent variables to model a dependent variable. it has the following mathematical representation [21]:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{3.3}$$

where

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\epsilon} = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}. \tag{3.4}$$

Here the vector $\mathbf{y}$ contains the dependent variable, $\mathbf{X}$ contains the independent variables, the vector $\boldsymbol{\beta}$ contains the regressor coefficients, and $\boldsymbol{\epsilon}$ is the vector of error terms.

In this model we aim to find the $\hat{\boldsymbol{\beta}}$ that minimizes the Residual Sum of Squares (RSS)

$$S(\boldsymbol{\beta}) = \sum_{i=1}^{n} \epsilon_i^2 = \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}). \tag{3.5}$$

It can be shown that the least-squares estimator can be calculated as:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X^T X})^{-1} \mathbf{X^T y}, \tag{3.6}$$

and the forecasts are calculated as:

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}. \tag{3.7}$$

3.3 Stochastic Modelling

In this section we will build up to one of the models that is used to make forecast from our time series. We will start by introducing the ARMA model, and in the following subsection introduce the ARIMA model, and after that introduce the SARIMA model. In the last subsection we will arrive at SARIMAX model which will later be used to make forecasts on our time series.

3.3.1 ARMA

An Autoregressive Moving Average (ARMA) is a linear process which consists of two parts: an Autoregressive (AR) part, and a Moving Average (MA) part [8].

In an AR(p)-model the output depends linearly on the models own previous outputs up the order p, as well as an error term. An AR(p)-process has the following mathematical representation:

$$X_t - \sum_{i=1}^{p} \phi_i X_{t-i} = \epsilon_t \quad \iff \quad \phi_p(B) X_t = \epsilon_t, \quad \epsilon_t \sim \text{WN}(0, \sigma^2), \tag{3.8}$$

where B is the backshift operator and $\phi_p(z) = 1 - \sum_{i=1}^{p} \phi_i z^i$.

In an MA(q)-process the output depends linearly on the current error term as well as past error terms up until order q. An MA(q)-process has the following mathematical representation:

$$X_t = \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} \quad \iff \quad X_t = \theta_q(B)\epsilon_t, \quad \epsilon_t \sim \text{WN}(0, \sigma^2), \tag{3.9}$$

where $\theta_q(z) = 1 + \sum_{j=1}^{q} \theta_j z^j$.

An ARMA(p, q) is a combination of both the AR(p)-model and the MA(q)- model and has the following representation:

$$X_t - \sum_{i=1}^{p} \phi_i X_{t-i} = \epsilon_t + \sum_{i=1}^{q} \theta_j \epsilon_{t-j} \quad \iff \quad \phi_p(B) X_t = \theta_q(B)\epsilon_t, \quad \epsilon_t \sim \text{WN}(0, \sigma^2). \tag{3.10}$$

3.3.2 ARIMA

One drawback of the ARMA(p, q)-model is that it can only handle stationary time series, i.e., when the conditions in 3.1.3 are satisfied. A time series can often be made stationary if it is non-stationary through an operation called differencing. In an ARIMA(p, d, q) the time series is differenced d times until stationary. The differencing operator is defined as $\nabla^d = (1 - B)^d$, where B is the backshift operator, resulting in the following representation of the ARIMA(p, d, q)[8]:

$$\phi_p(B)\nabla^d X_t = \theta_q(B)\epsilon_t. \tag{3.11}$$

3.3.3 SARIMA

The Seasonal ARIMA (SARIMA)-model allows for randomness in the seasonal pattern from one cycle to another, and does unlike the previously mentioned models not assume a deterministic seasonality. The seasonality is modelled with their own autoregressive and moving average orders with a fixed periodicity s. The SARIMA$(p, d, q)(P, D, Q)_s$ has the follwing mathematical representation[8]:

$$\phi_p(B)\Phi_P(B^s)\nabla^d\nabla_s^D X_t = \theta_q(B)\Theta_Q(B^s)\epsilon_t, \quad \epsilon_t \sim \text{WN}(0, \sigma^2), \tag{3.12}$$

where $\phi_p(z)$, $\theta_q(z)$, and ∇^d are the same as in subsections 3.3.1 - 3.3.2, and $\nabla_s^D = (1 - B^s)^D$, $\Phi_P(z^s) = (1 - \sum_{k=1}^{P} \Phi_k z^{ks})$, and $\Theta_Q(z^s) = (1 + \sum_{l=1}^{Q} \Theta_l z^{ls})$.

3.3.4 SARIMAX

The SARIMAX$(p, d, q)(P, D, Q)_s(X)$-model is a SARIMA$(p, d, q)(P, D, Q)_s$-model with the inclusion of external variables X. The external variables are modelled using MLR as

$$Y_t = \beta_0 + \sum_{j=1}^{k} \beta_j X_{j,t} + \omega_t, \tag{3.13}$$

where $X_{j,t}$, $j = 1 \dots k$ are observations of the independent variables, Y_t the dependent variable, and $\beta_0, \dots, \beta_k$ are the regressor coefficients. ω_t is the residual and can be represented using the SARIMA-equation stated above as

$$\omega_t = \frac{\theta_q(B)\Theta_Q(B^s)}{\phi_p(B)\Phi_P(B^s)\nabla^d\nabla_s^D}\epsilon_t, \tag{3.14}$$

giving the following equation for Y_t [1]

$$Y_t = \beta_0 + \sum_{j=1}^{k}\beta_j X_{j,t} + \left(\frac{\theta_q(B)\Theta_Q(B^s)}{\phi_p(B)\Phi_P(B^s)\nabla^d\nabla_s^D}\epsilon_t\right). \tag{3.15}$$

3.4 XGBoost

Within supervised learning, a popular model used in various regression and classification tasks is Extreme Gradient Boosting (XGBoost) which is a certain type of gradient boosting tree. This is an ensemble model which contains several models in a symbiotic manner to enhance performance and accuracy. The general idea is to train predictors sequentially and add new components that improves the combination of the predecessors. The core components, also known as base learners, of this ensemble are Classification And Regression Trees (known as CART) [9].

Classification And Regression Trees

Decision trees are built on regional representations that divide the predictor space into non-overlapping regions. Each region is based on recursive binary splitting, which creates the tree like structure by following a greedy approach. It is greedy because it always makes the best split based on the particular step during evaluation [13]. By considering a predictor X_j, and a split point s, we can define a pair of half-planes as:

$$R_1(j,s) = \{X|X_j < s\} \text{ and } R_2(j,s) = \{X|X_j \leq s\}. \tag{3.16}$$

Based on this position we seek to minimise the equation based on the Residual Sum of Squares (RSS):

$$\sum_{i:\ x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\ x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2. \tag{3.17}$$

Here, $\hat{y}_{R_1}$ and $\hat{y}_{R_2}$ are the mean response for the training observations in $R_1(j,s)$

and $R_2(j, s)$ respectively. This process continues to minimise the RSS by continuing down one of the identified regions and repeats for each identified region based on the predictor. This is continued until a stopping criterion is met. Now a tree is created where the data sits on top and is then defined down on each branch of the trees and then allocated to each region based on the splitting criterion. Each tree can then be written more compactly as:

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m). \tag{3.18}$$

Here the target is modeled by a constant c_m in a region R_m for a total of M regions [13].

Small and large trees can be created based on the stopping criterion. Large trees are usually prone to overfit with the training data, whereas smaller trees usually miss important patterns in the data for future predictions. To find a good balance, so called tree pruning is used. First a large tree T_0 is created (the depth is defined beforehand). From this large tree, subtrees are created $T \subset T_0$ by collapsing the original trees internal nodes, these collapsed nodes are now considered terminal nodes and are indexed by m representing region R_m [13].

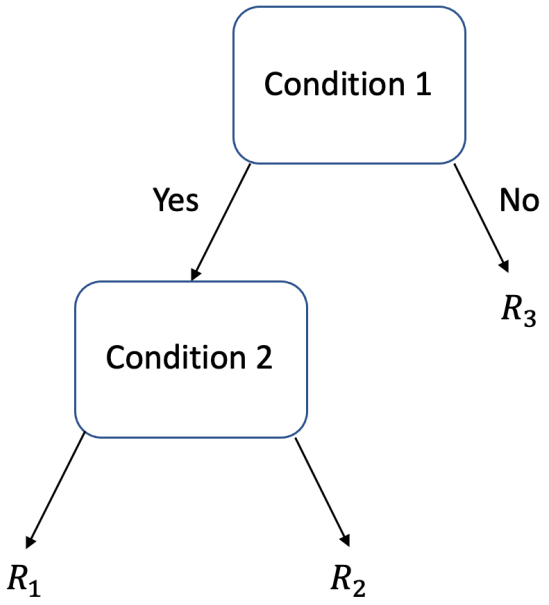

Figure 3.4.1: Representation of the tree structure

Gradient Boosting Trees

As a continuation of the ensemble algorithm described above, several trees are used together and summed up for a final score. Worth noting is that the trees complement each other to produce a final score for the ensemble [9]. This final value is written mathematically on the form:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}. \tag{3.19}$$

Here, K is the number of additive functions used in the model and $\mathcal{F}$ is the space of the regression trees. This space of regression trees is defined as $\mathcal{F} = \{f(x) = w_{q(x)}\}(q : \mathbb{R}^m \rightarrow T, w \in \mathbb{R}^T)$, where f_k corresponds to each tree, T correspond to the number of regions, w are leaf weights where w_i represents the continuous score on the i-th leaf, and lastly, q is a function assigning each data point to the corresponding leaf and is seen as an independent tree structure. To exemplify, for each observation provided the decision rules by q for each tree is used to classify into the weights w_i and summing these to a final score as the final prediction[9].

From this, we can introduce the objective function used for training the model:

$$obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i), \tag{3.20}$$

where $\sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)})$ corresponds to the training loss function and $\sum_{i=1}^{t} \Omega(f_i)$ to the regularisation parameter and is defined as:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2. \tag{3.21}$$

Since the objective function includes functions as parameters it cannot be optimised using traditional methods and is instead trained in an additive manner. That is, since the boosted trees sum the predictions of previous values it follows that the new value is added to the previous ones. For a given prediction $\hat{y}_i^{(t)}$ at the i-th instance at the t-th iteration, f_t is added to minimise the objective function:

$$obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \sum_{i=1}^{t} \Omega(f_i). \tag{3.22}$$

In this objective function, f_t is added greedily such that the model improves the most. The loss function $l(\cdot, \cdot)$ is a differentiable convex loss function, and to quickly optimise the objective function a Taylor approximation of the second order is used. Based on this second order derivative and by defining instance sets for the leafs, the loss function together with the complexity function is used to define a scoring function to appropriately assigning weights for each added tree [9]. Boosting in general can be seen as sequentially updating the weights through training according to figure 3.4.2 [15].

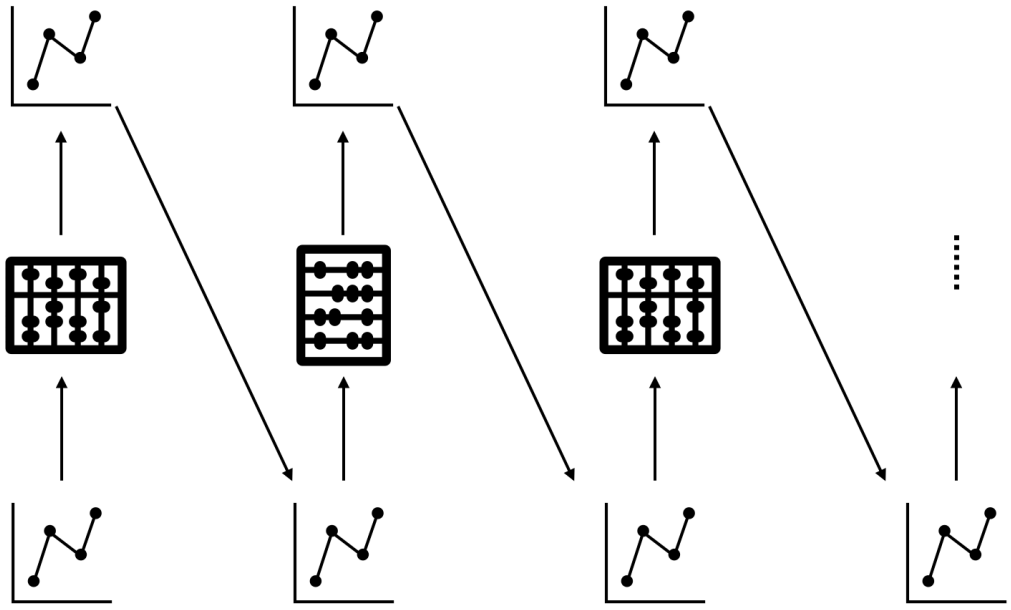

Figure 3.4.2: Concept of boosting learning algorithm

Due to its nature, Gradient Boosting can perform both regression tasks and classification tasks because of its base learners (i.e., the CARTs). What makes XGBoost stand out is that the loss function and the regularisation function given above has been tweaked to be as computationally efficient as possible, thereby pushing the computational limits of the boosting algorithm, hence its name. Also, since it builds on ensembles of CARTs that uses splitting points for the input data it is indifferent to scaling. Lastly, the representation of the trees makes it easier to interpret and the user of the model is able to deduce how the model evaluate different data points by mapping the final splitting points [15].

3.5 Neural Networks

Neural Networks (NN) have had an interesting and pivotal role for machine learning's increased usage over the last century. Due to its nature, it has been applied to several different applications that utilises data to be able to produce a results. Some examples of these are image recognition, speech recognition, text generation, or simply anything that could be classified or regressed, including time series forecasting [2].

The concept of NN's derives from the depiction of neurons in the human brain and the first conceptualisation of this was the so called perceptron model that was formalised in 1958. A perceptron is fed information from inputs which are then multiplied by weights and summed together with an external bias term. This summation is then passed through an activation function and the result is passed on to its outputs. Figure 3.5.1 shows the concept of the perceptron model [2].

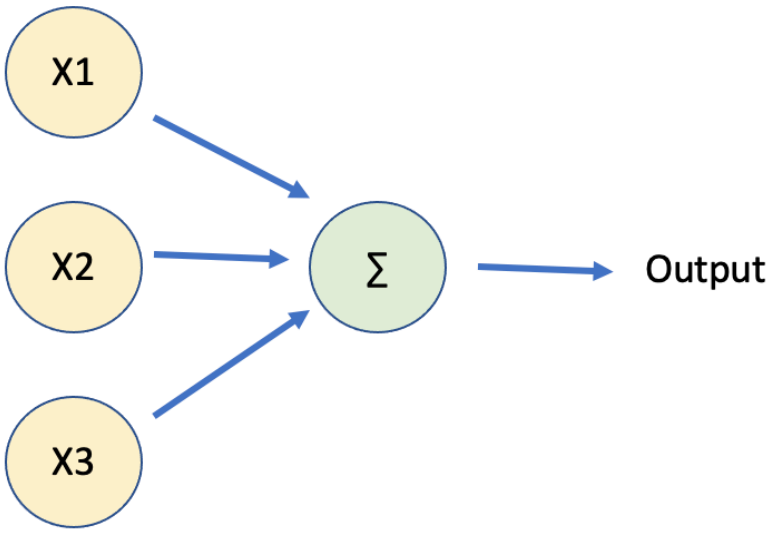

Figure 3.5.1: Concept of a perceptron

When neurons are used in combinations and stacked in several layers they are called MultiLayered Perceptron (MLP) or FeedForward Neural Networks (FFNN). The latter due to the construction of the network as it process data from its input layer through its hidden layers in a unidirectional manner until it reaches the output layer. Each node within this model functions similarly to the aforementioned perceptron. It receives inputs from the previous layer together with specific weights, but it also adds a bias term and then transforms the sum through an activation function. Thanks to this structure NN's gains advantages compared to classical models. NN's can model any

form of unknown relationship in the data and then generalise and transfer the learned relationships to unseen data [18].

In general if the intention is to predict an outcome Y_t from predictors X_t for a given t from 0 to T (i.e., $t = 0, 1, 2, \dots, T$) with a neural network with a single layer, we first ignore the aspect of time and only consider Y_t and X_t as an independent pair. By denoting each neuron in the hidden layer as $\mathbf{H}_t = \{H_{tm}\}_{m=0,\dots,M}$, the outcome of interest is then predicted as

$$Y_t = \sigma_y(\mathbf{w^T H_t} + b). \tag{3.23}$$

Each unit in the hidden layer is calculated as:

$$H_{tm} = \sigma_h(\mathbf{w}_m^T \mathbf{X_t} + b_{0m}). \tag{3.24}$$

Here, $\mathbf{w}$ is a vector of the corresponding weights, b are the biases added at each transformation and σ_h is the activation function used within the hidden layer. A more general equation for several hidden layers for calculating a neuron at layer $k+1$ is given by:

$$H_i^{k+1} = \sigma\left(\sum_{j=0}^{n} H_j^k w_{j,i}^k + b_i^k\right). \tag{3.25}$$

Again, H_i^{k+1} is the i-th neuron in the $k+1$ hidden layer. The weights between the neurons are represented as $w_{j,i}^k$ from neuron j in layer k to neuron i in layer $k+1$. Lastly, b_i^k is the bias term added from neuron i in layer k. Figure 3.5.2 shows the structure of the FFNN with one hidden layer [11].

Training of this model is done through gradient descent and weights and biases are updated according to a loss function through back-propagation. Common loss functions to use within regression tasks are mean squared error (MSE) and mean absolute error (MAE). During training of NNs, it is common to use so called epochs which are iterations during the training process. These iterations feed pre-specified sizes of batches containing training data with an updating step for each batch. When an epoch is finished, the loss function is calculated from the training data and the validation data. When all epochs are done the training of the network is complete

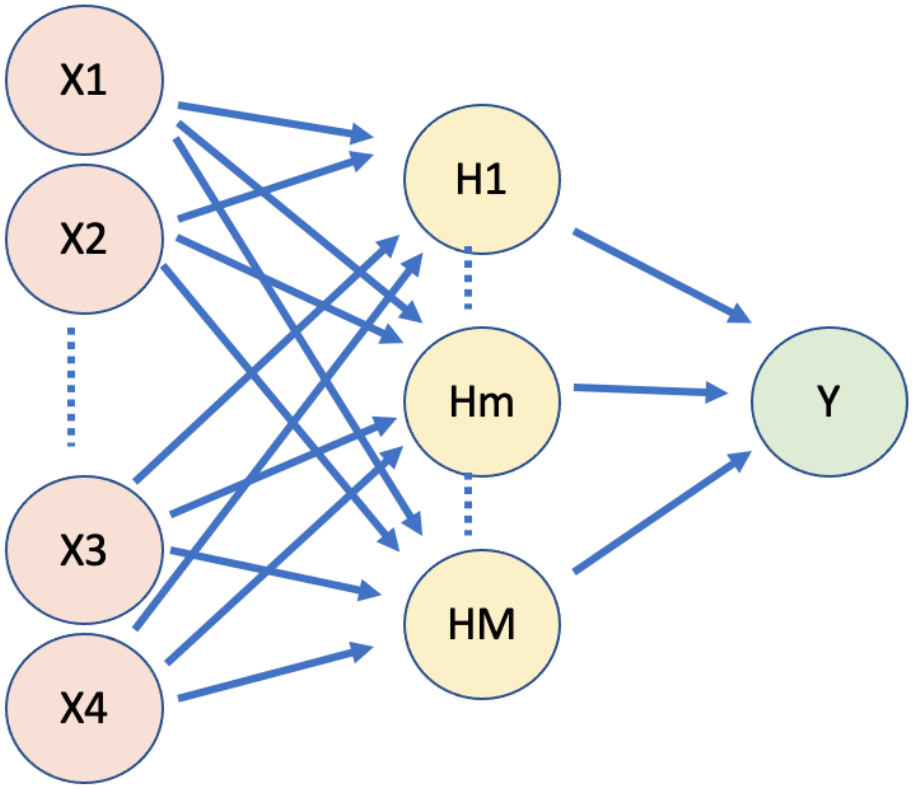

Figure 3.5.2: Feed-forward Neural Network with one hidden layer

[11].

The use of back-propagation can be computationally expensive as more layers, and neurons within each layer, are included. Another shortcoming of this is that a high amount of parameters could lead to overfitting the model. Thus, regularisation and penalisation techniques are used to prevent the model from overfitting, common methods are ridge and lasso regression. However, these types of regularisation methods adds to the computational complexity. Therefore methods such as early stopping rules or dropouts are used in practice that reduces complexity. Early stopping can be used by inspecting the plot of the loss function. When the loss function evaluated on the training data converges, or decreases slightly, and the loss function evaluated on the validation set start increasing, is an indication that the model is starting to overfit. These early stoppings can be automated by using a number called patience which is defined in the model to stop when the loss function does not decrease at a satisfying rate after a certain amount of epochs. As for the second method, dropouts, means that some neurons within the layers are randomly ”dropped” during training. Doing so reduces the dependency on certain neurons and thus reducing the risk of overfitting. It is popular to use due to its simplicity to implement, and

common dropout values are in the range of $10 - 50\%$. All nodes are later reactivated during the prediction phase. Common activation functions within a NN's are the Sigmoid Function, Hyperbolic Tangent Function (Tanh) and the Rectified Linear Unit (ReLU) function. The Sigmoid and the Tanh function are traditionally more popular but the ReLU has seen increased popularity as of late as it helps with reducing computational complexity during training. The definitions for the above mentioned activation functions can be found in table 3.5.1. When regression tasks are performed, a linear output is commonly used for the output layer in an NN [11].

Sigmoid(z)	$\frac{1}{1+e^{-z}}$
Tanh(z)	$\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$
ReLU(z)	$\max(0, z)$

Table 3.5.1: Activation functions

Reccurent Neural Networks

Feed-forward Neural Networks generally shows good results within regression tasks. A shortcoming of these type of networks is that they can be unstable when forecasting multiple steps ahead and cannot learn from long-term dependencies in the data. To overcome this issue, the Recurrent Neural Networks (RNN) was proposed to handle sequential data and addressing the problem of pattern recognition in the data. RNN's are similar to the conventional NN's. The major difference is that by using feedback loops between the layers of the network, RNN's are able to follow temporal order in the data and capture sequential dependencies. By doing so, it uses the concept of cells instead of only single neurons. Base versions of RNN's, also known as Elman recurrent unit, are constructed by having an input, output and a hidden state that continues sequentially. Figure 3.5.3 shows the relationships in a RNN. Through every iteration (i.e., time step), there is an input X_t, the hidden state that is transferred from previous iteration is h_{t-1} that together form an output $\hat{Y}_t$. During this iteration, a new hidden state h_t is calculated and then passed on to the next iteration - which resembles the memory of the RNN [18].

Even though this type of RNN is able to capture some dependence between the hidden states, it still suffers from not recognising long-term dependencies. Because of this, the Long Short-Term Memory (LSTM) was introduced. LSTM's are built as conventional NN's with input layers, hidden layers and output layers but it is within the hidden layer

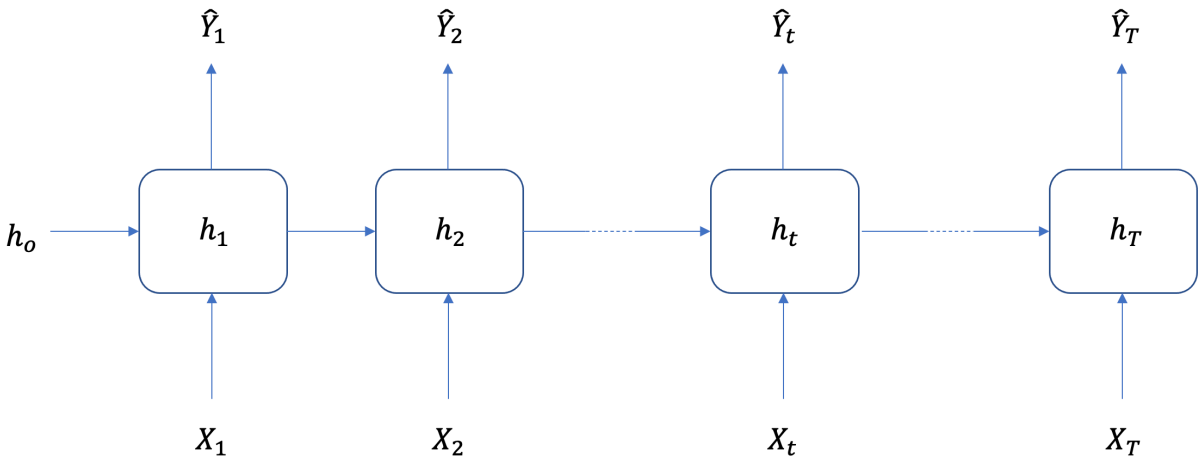

Figure 3.5.3: Recurrent Neural Network structure

the LSTM architecture takes place. It is similar to the basic RNN by utilising a hidden state h_t for short-term dependencies. But it also includes a long-term dependency between the cells called cell state c_t. Figure 3.5.4 shows the structure of the LSTM cell [18].

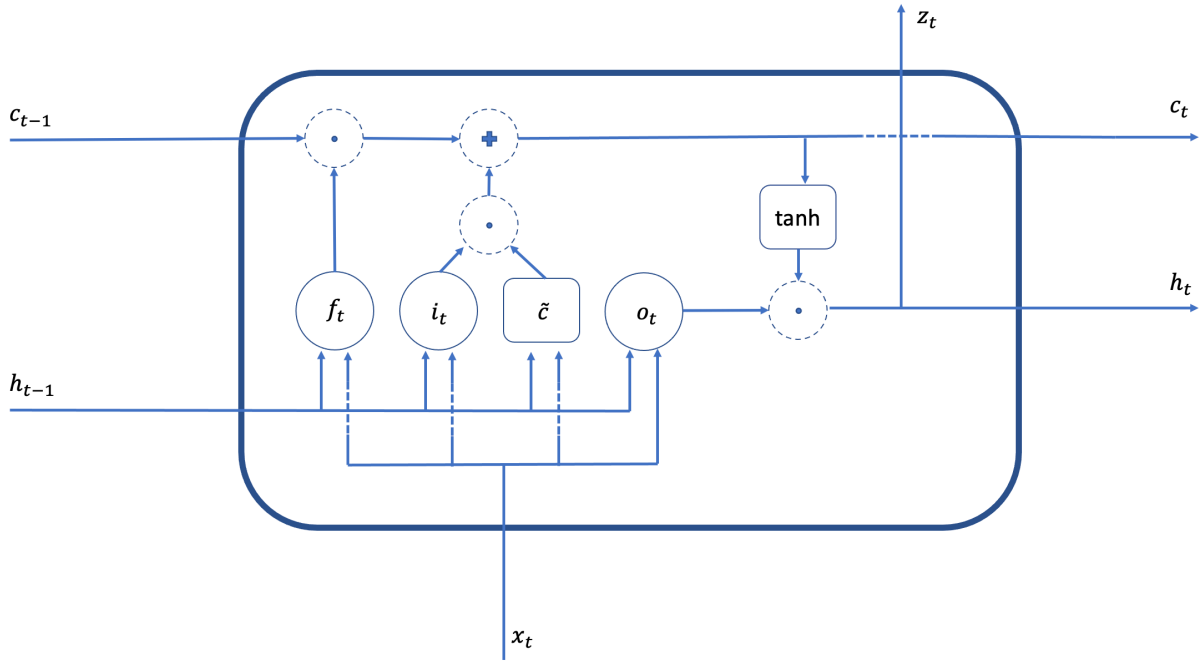

Figure 3.5.4: LSTM cell structure

Each cell then consists of a forget gate, input gate and output gate. The forget gate f_t and the input gate i_t determine how much of the past information that should be kept in the current cell and how much should be fed forward to future time steps. They utilise the Sigmoid functions taking values between 0 and 1. A value of 0 means

that nothing should be used and a value of 1 that information should be passed on. A value in between controls how much of the information is necessary for the current state from both the previous states and the current candidate cell state. The candidate cell state $\tilde{c}_t$ captures the important information from both the previous hidden state and the current input at time step t. Together by piece wise multiplication between the information from the input gate and the candidate cell, information is then added together with the previous cell state from the forget cell. Here, the cell state at time t is created and passed on to the next iteration. This information is also passed through a hyperbolic tangent function and piece wise multiplied by the information from the output gate to produce both the new hidden state and the current output z_t at time step t. This output z_t can either go directly to the output layer or to another LSTM layer if a deeper network is implemented. This concludes how the LSTM operates and each LSTM layer constitutes of several cells. Training of a LSTM is similar as to conventional NN described earlier, but by using the recurrent state, the training algorithm is now called back propagation through time (BPTT) as it uses previous information when updating the weights and not only the current information [18].

3.6 Evaluation

In the following subsections the evaluation metrics that in the end will evaluate and determine which one of the models that have the best performance will be presented. Each one of them has their respective strengths and weaknesses and they are chosen to provide a basis for an as fair evaluation as possible.

3.6.1 Mean Absolute Percentage Error

Mean Absolute Percentage Error (MAPE) is an evaluation metric that averages the absolute size of percentage error between the forecasted values F_i and the observed values A_i. It is applicable when the quantity to predict is known to be a large positive value. The MAPE has the following mathematical representation:

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{A_i - F_i}{A_i}\right|. \tag{3.26}$$

MAPE is widely used in practise because of its intuitive interpretation [22].

3.6.2 Root Mean Squared Error

Root Mean Squared Error (RMSE) is a measure that punishes large outliers in the residuals. Therefore it is an useful metrics when high accuracy in large deviating values are of great importance. It has the following mathematical representation [6]:

$$\text{RMSE} = \left(\sum_{i=1}^{n} \frac{(F_i - A_i)^2}{n} \right)^{\frac{1}{2}}. \tag{3.27}$$

3.6.3 Mean Absolute Error

Unlike RMSE, Mean Absolute Error (MAE) does not penalise large error, but gives the same weight to all errors regardless of the size. MAE has the following mathematical representation:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |F_i - A_i|. \tag{3.28}$$

Chapter 4

Methodology

4.1 Data analysis and pre-processing

The data used in this project came from QLIRO's database. It contained daily reservations volumes between the period 2014-05-22 and 2021-12-31. The corresponding reservations had been scaled by QLIRO in order not to work with the true reservation amounts. Here reservation volume refers to a certain amount of money that is reserved when customers has made an online purchase. Prior to building the models some modest data pre-processing was required and some explanatory data analysis was conducted.

4.1.1 Exploratory Data Analysis

An Exploratory Data Analysis (EDA) was conducted on the data set. The reasons for this were two-fold: it was firstly done to get to know the data, and secondly to find recurring patterns and other discrepancies in the data that later could be used to create features for our forecasting models [24]. The data set was plotted in its entirety to understand its development over time and to identify recurrent patterns. After this, the data was grouped into different temporal categories: it was first grouped at a yearly level, then grouped on a monthly level, and lastly for each weekday. By doing so, other recurrent patterns could be identified. As a follow up to this, characteristics of the empirical distributions for each member of the grouping was performed. This was done through histograms, QQ-plts, and violine plots. In total, this analysis provided a good basis for understanding how the data behaves, which features to create from the

date times, as well as what features to create for other recurring events such as Black Friday.

4.1.2 Main findings from EDA

Based on the conducted EDA, seasonality patterns could be seen. Most significant were traditional sales periods around November and December. This was expected based on intuition and assumptions about e-commerce sales patterns. Other sales periods could also be seen around spring showing there is a yearly period. Similarly, there were also weekly sales patterns. These weekly sales patterns changed between months in amplitude but also during which days the cyclical patterns moved. From this, it was clear that they were not completely deterministic and followed what could be seen as other cyclical changes described in section 3.1.2. Worth noting is that there were also comparable drops in sales that were recurring around special events such as midsummer's eve, Christmas and New Year's eve. At the same time, non-recurring increase and decrease in daily sales was observed, especially around the year 2021. Such anomalies were difficult to find an explanation as to why they appeared. Even if they were apparent, these anomalies were not very deviant compared to the recurring special day patterns. They showed some deviance compared to adjacent days and weeks but not too much. Thus, these anomalies were considered interesting for the model training and was not treated. Another aspect is that the data develops over time, meaning that new trends emerge in later years that is not necessarily showing up in earlier years. This needs to be taken into consideration during model training. Lastly, there is also an increase in mean level that can be seen from one year to the other. From these findings, special dates and periods were identified that could provide important information for the models.

4.1.3 Features

The features that were chosen for model building have their basis in the analysis conducted in subsection 4.1.2. Since significant differences between the total reservation were found in regard to what day of the week it was, which month of the year it was; these aspects was turned into categorical features. Significant outliers was also found on special holidays such as around Thanksgiving (i.e., Black Friday), Singles day, and the days prior to and after Christmas. Each event was turned into

a corresponding slack feature or categorical feature depending on context. The goal of this feature engineering step was to explain as much of the variation in the data as possible by encoding the events. In table 4.1.1 a full list of features resulting from the analysis is presented along with how they were initially coded. The different models in sections 4.2.1 - 4.2.4 requires different encoding of the features. How the features were encoded will be explained under each section respectively.

Feature	Type	Values(s)
Year	Categorical	2018 - 2021
Month	Categorical	0 - 11
Weekday	Categorical	$0-6$
Paymentweek	Boolean	0,1
Blackweek	Categorical	0 - 5
Blacktime	Boolean	0,1
Priorchristmas	Categorical	0-15
Postchristmas	Categorical	0-6
Christmas	Categorical	0 - 2
Newyearseve	Boolean	0,1
Newyear	Boolean	0,1
Greenweekend	Categorical	0 - 3
Valentine	Boolean	0,1
Mothersday	Boolean	0,1
Singlesday	Boolean	0,1
Midsummer	Categorical	0 - 2
Priormidsummer	Boolean	0,1
Postmidsummer	Boolean	0,1

Table 4.1.1: List of features used in the models to forecast

4.1.4 Data smoothing

As discussed in chapter 1, e-commerce data is usually prone to be very erratic and with varying seasonality patterns. In the article by Rémy Garnier and Arnaud Belletoile used in their talk at the Conference on Practical Applications of Artificial Intelligence (APIA) in 2019, they discuss the problems of high sales data on particular days. High sales data on certain days usually corresponds to seasonal patterns related to sales periods (e.g., Black Friday). It is argued that this data contains information regarding the effects of sales. However, these particular days could affect the forecasting ability of the next coming days due to their unique behaviour. Therefore, they constructed ”smoothed sales data” based on a certain amount of standard deviation from a moving average [14].

This type of technique could help improve the model and therefore the original data set is complemented with a smoothed alternative for comparison. Inspired by the approach described earlier, below is a description of the steps for smoothing the data. The day of interest is t and M is a period of 30 days:

- A rolling moving average of each month was calculated together with a corresponding rolling standard deviation:

$$\bar{y}_t = \frac{1}{M} \sum_{k=0}^{M} y_{t-k}, \tag{4.1}$$

$$\bar{\sigma_t} = \left(\frac{1}{M} \sum_{k=0}^{M} (y_{t-k} - \bar{y}_t)^2 \right)^{\frac{1}{2}}. \tag{4.2}$$

From these values:

- If $y_t > \bar{y}_t + \gamma \bar{\sigma}_t$, then $x_t = \bar{y}_t + \gamma \bar{\sigma}_t$,
- Otherwise $x_t = y_t$.

4.2 Modelling

In this section, each model will be presented with their respective features and implementation. Each model will make use of the features as stated in subsection 4.1.3 but also incorporate model specific implementation for these features. Each model follows the descriptions given in chapter 3 and was built using Python 3.8 [31]. Additional libraries, or packages, that is available for Python was also used which differed between each model. In total, they consisted of the following: *pandas* [30], *numpy* [16], *matplotlib* [19], *sklearn* [23], *statsmodels* [26], *pmdarima* [27] and *tensorflow keras* [10].

For each model, the entire data set was separated into three parts, i.e, into a *training*, *validation* and *test* set. The first split was done between the training and test set, based on the evaluation detailed in subsection 4.2.5. For this split, data from 2021 was used for testing of each model. As for the second split, this split was done based on a percentage split on the remaining data set to construct a training and validation set used during model training. Since the validation set contains the most up to date information, it could be necessary to include this data in the training set and then retrain the model. Such technique is discussed in [18] and showed good results.

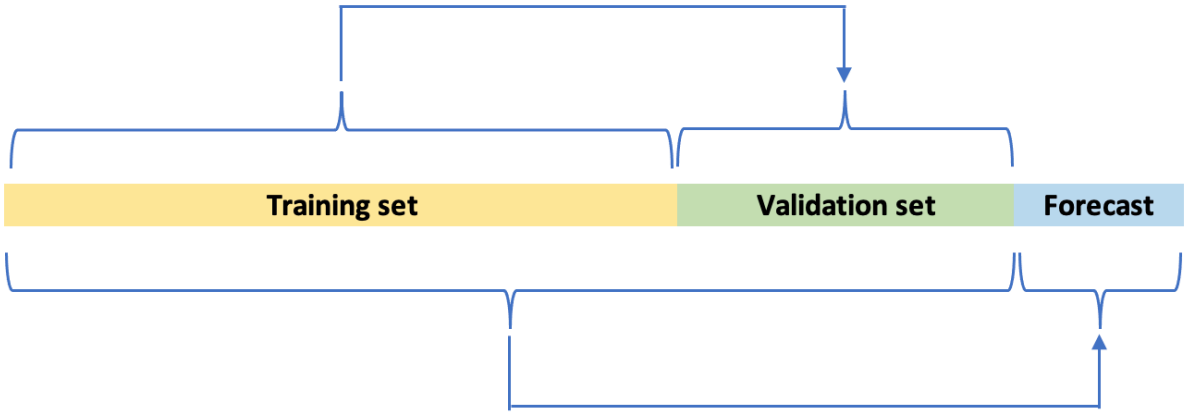

Figure 4.2.1: Training, Validation, and Test split

Before continuing onward to the model-specific information, one last note regarding the training phase is necessary. From the findings in subsection 4.1.2 it could be seen that earlier data was different from the latest data in terms of absolute values and new trends. Based on ocular inspection and initial model testing it was concluded to only use data from 2018 and onward for model training. The reason for this is that the patterns of the data has changed over time meaning that previous years lack information for later years. Such miss information could affect the models during training and result in bad accuracy by learning from the wrong behaviour.

4.2.1 MLR

For the MLR all the categorical features were one-hot encoded for the model to learn properly. See table 4.1.1 for a full list of features used to train the model. The MLR was trained using the *statsmodels* package in Python. This particular model required no hyperparameter tuning.

4.2.2 SARIMAX

Since the SARIMAX model includes a MLR-model its exogenous inputs entered the model with the same encoding as in subsection 4.2.1. The seasonal order s was chosen to be 7 days as this periodicity exhibited the most rigid seasonal pattern. for this reason the *Weekday* - feature was dropped as this pattern would be handled by the stochastic process. To find the optimal parameters p, d, q, P, D, and Q, the *pmdarima* package was used. This package searches through different combinations of hyperparameters and returns the one with the lowest AIC score. The hyperparameters that were tested

in this model are displayed in table 4.2.1.

Hyperparameter	Values(s)
s	7
d	$0, 1$
D	$0, 1$
p	$0 - 6$
q	$0 - 6$
P	$0 - 6$
Q	$0 - 6$

Table 4.2.1: SARIMAX hyperparameter space

4.2.3 XGBoost

Due to the construction of the XGBoost described in section 3.4, it can handle different types of data types following ordinal structure. Therefore, no further encoding of the features was required, and they entered the model as detailed in table 4.1.1. Another reason for this decision is that one-hot encoding categorical variables that has a high relation, such as weekdays, can cause poor performance from the model [17]. It is also similar to what was used in [14] and [32]. Other features used were rolling statistics and previous lagged targets. These did not improve the model during initial testing and it was decided to use as many similar features as possible for comparing the models. Moving over to hyperparameter tuning, the technique used was Bayesian Optimisation to find the most suitable parameters iterating between those given in table 4.2.2. Choosing Bayesian Optimisation was mostly due to its good ratio between reduced calculation time and increased performance of the model. The package used to build this model and to perform hyperparameter tuning was *sklearn*.

Hyperparameter	Values(s)
Max depth	$2, 3, 5, 10, 15$
Number of estimators	$250, 500, 750, 1000$
Column samples per tree	$0.5, 0.6, \dots, 1.0$
Subsample ratio	$0.5, 0.6, \dots, 1.0$
Min child weight	$1, 2, 4, 6, 8$
L2 Regularisation	$0.1, 1, 10, 100$
Learning rate	$0.05, 0.1, 0.15, 0.20, 0.3$

Table 4.2.2: XGBoost hyperparameter space

4.2.4 LSTM

When it comes to the LSTM, the use of features differ. Due to its construction, the LSTM is fed inputs from previous data points seen as lagged features. These include the target for that day but also other features that could give information to the network similar to those in subsection 4.1.3. Thus, each input node corresponds to one of the lagged data points. Also, the number of output nodes corresponds to the amount of forecasting steps that are taken. This implies that the data has to be reconstructed to follow this approach and follows that of Taken's Theorem for dynamical systems [7]. There is no consensus for what amount of lagged features that should be used for a certain amount of forecasting steps. By heuristics, Hewamalage et al. proposed that if n time steps are to be predicted, then $1.25 \cdot n$ should suffice [18]. This captures patterns of the weekly seasonality but not seasonal aspects on a yearly basis. Yearly aspects will hopefully be captured by the LSTM's long-term memory. As for the outputs, there are different strategies that could be used. Historically, the recursive strategy has been regarded as the go to method as it models single-step predictions and then uses these to predict further into the future. The shortcoming of this is that as the forecast horizon increases, the error of the forecast increases rapidly. Thus it has been suggested to use a multi-input multi-output (MIMO) strategy for LSTM's during forecasting as it is more reliable, i.e., several output nodes as described above [7, 18]. Even if the MIMO strategy is said to be more reliable, as the forecast horizon output increases, the reliability of the later steps diminish [2, 7, 18]. To increase the accuracy of the LSTM model, it should be treated as a classical time series model. That is to pre-process the data by removing trend and seasonality [18]. However, since it is also good at predicting the non-linear dependence of the data, it was chosen to feed it non-differenced data similar to the XGBoost and MLR models. This is chosen due to the simplicity of feeding the data to the model but also the problems of the other cyclical changes in described in subsection 3.1.2. In the work by Bandara et al., they also included the seasonal component [3]. Using the smoothed data set as described in subsection 4.1.4 could help relaxing the model.

To construct the aforementioned features, a so called moving window approach is used. The input window include all the lagged data points and the output window consist of the targets that are to be predicted. In addition to these lagged data points, similar features was used in the LSTM as described in subsection 4.1.3, the difference is those of ordinal structure. These features were constructed to be of cyclic nature utilising

sine and cosine functions to keep track of their periodical nature. Thus the NN better understand that Monday is closer to Sunday than Wednesday is. Using these types of features also relaxes the system of needing one-hot encoding and increasing the amount of features. A concept of the mowing window approach is given in figure 4.2.2 [3].

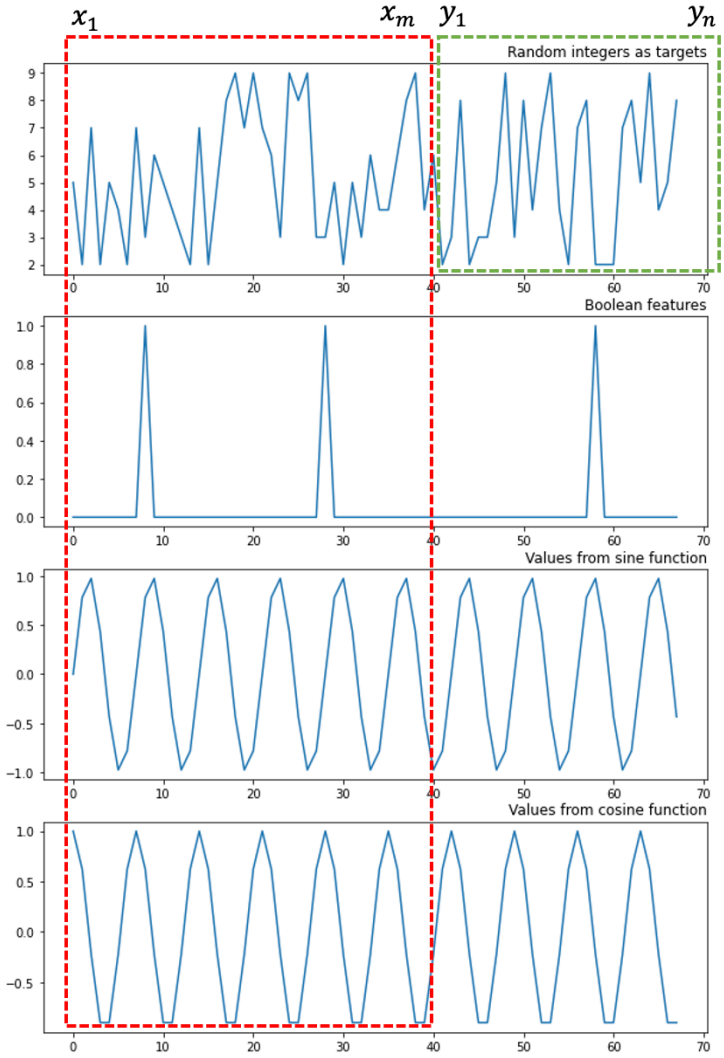

Figure 4.2.2: Concept of the moving window approach, m is the amount of lagged values and n is the forecast horizon

Furthermore, LSTM networks is said to be sensitive to the scale of the data. Therefore the data is normalised using the MinMaxScaler from the sklearn package. By using the scaler, the data is scaled between $(-1, 1)$ which is preferred for the hyperbolic tangent function. If x_s is the new scaled value, x the original value, x_{min} the minimum value and x_{max} the maximum value and lastly min and max are the end points in the feature range $(-1, 1)$. From this, the MinMaxScaler is defined as:

$$x_{std} = \frac{x - x_{min}}{x_{max} - x_{min}},$$
$$x_s = x_{std}(max - min) + min.$$

Hyperparameter tuning of network consisted of the parameters given in table 4.2.3. These parameters were tuned by using Bayesian Optimisation which is said to give good results and minimise computational complexities [18]. The LSTM network was built using KERAS and the Baysian Optimisation was carried out using KERAS tuner. The construction of LSTM in previous work shows that more than three layers does not improve the model and neurons between 32 and 128 usually give accurate results [3, 7, 11]. Thus, the proposed model is tested using one, two, and three layers; then parameters are tuned between these layers. To reduce learning complexities, early stopping is used with a patience of 50 epochs and a total length of 300 epochs.

Hyperparameter	Values(s)
Number of layers	$1, 2, 3$
Number of neurons	$32, 64, 96, 128$
Batch Sizes	$8, 16, 32$
Dropout level	$none, 0.1, 0.25, 0.5$
Learning rate	*between* $(0.0001, 0.01)$

Table 4.2.3: LSTM hyperparameter space

4.2.5 Evaluation

It was decided with the supervisor at QLIRO that the forecasting horizon to be evaluated for each model were to be 30 days. This corresponds to produce a forecast for a month during the year. Since predictions of each month during a year was of interest, the test set containing data from 2021 was divided into its respective month. Each model was then evaluated for each month following an expanding window approach. Similarly to what was described in the beginning of section 4.2, each month was appended into the total training set and used to retrain each model. It was also decided to retrain each model with the whole data set instead of only partly training it on the new data. This decision was mostly made to alleviate the problem of catastrophic inference that occurs for NN, LSTM's included [25]. For the sake of consistency, the same approach was used for the other models as well, especially since they were quicker

to train compared to the LSTM network. For each month, the metrics given in 3.6 were used to evaluate the performance of the models. When the entire year of 2021 was iterated a total average of each metric was calculated for comparison as well. It was also of interest to see the magnitude of the errors for particular days including special sales event.

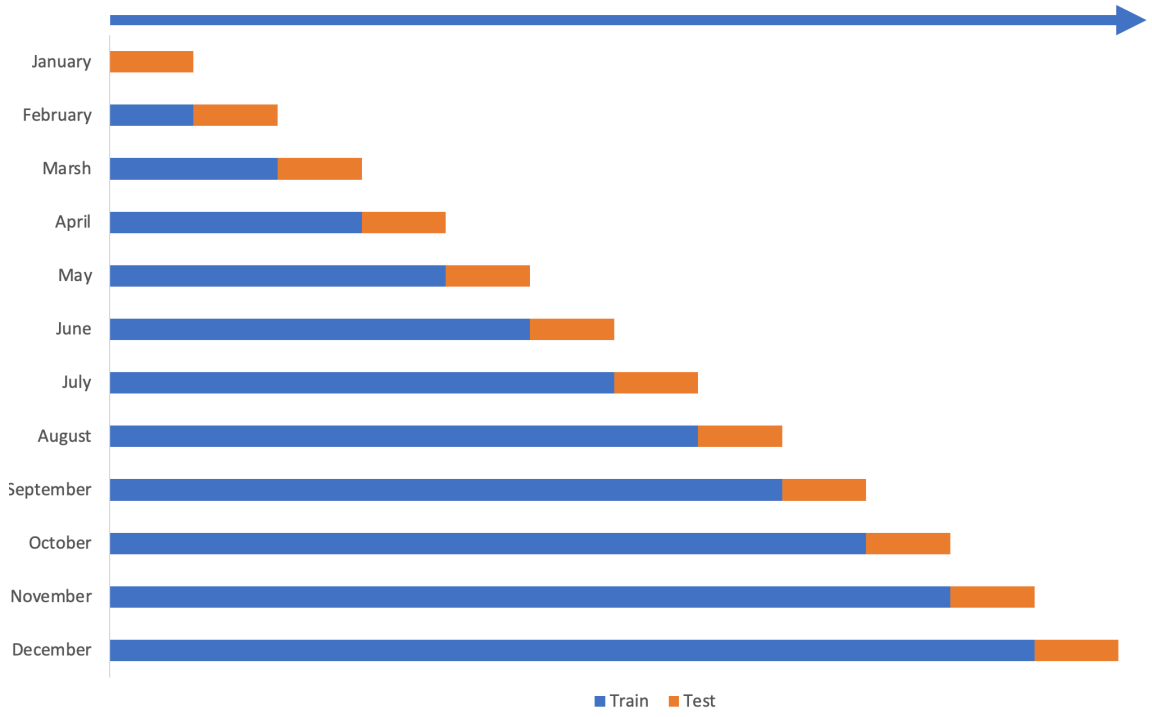

Figure 4.2.3: Concept figure of training and testing during model evaluation

Chapter 5

Results

The results chapter will have the following outline. In section 5.1 and 5.2 the results for each model using the original data set and the smoothed data set are presented respectively. Results are displayed with a table showing the models performance on a monthly basis using MAE, RMSE, and MAPE. The average of these measures over en entire year are also displayed. Following these tables, scatter plots of the standardised residuals and the absolute percentage errors are displayed.

5.1 Original data set

Within section 5.1, results from the models based on the original data set are presented. It follows the structure for how the models has been presented previously.

5.1.1 MLR

This section details the results from the MLR model using features as specified in subsection 4.2.1. From table 5.1.1 it can be seen that the performance varies in regards to both the MAE, RMSE, and the MAPE depending on which 30-day period is considered. It can further be seen from the scatter plots in figure 5.1.1. Most noticeable is the poor performance in December.

Month	MAE	RMSE	MAPE
January	549 327	745 393	0.053
February	730 274	985 228	0.070
March	853 098	1 034 778	0.081
April	1 651 434	2 336 421	0.167
May	1 110 314	1 319 630	0.112
June	1 422 729	1 693 640	0.135
July	1 478 216	1 806 983	0.171
August	883 076	1 062 348	0.085
September	1 267 346	1 860 638	0.107
October	1 037 802	1 985 579	0.083
November	3 033 190	4 541 065	0.154
December	2 598 914	3 552 113	0.253
Average	**1 384 644**	**1 910 318**	**0.123**

Table 5.1.1: Evaluation Metrics using MLR on the original data set

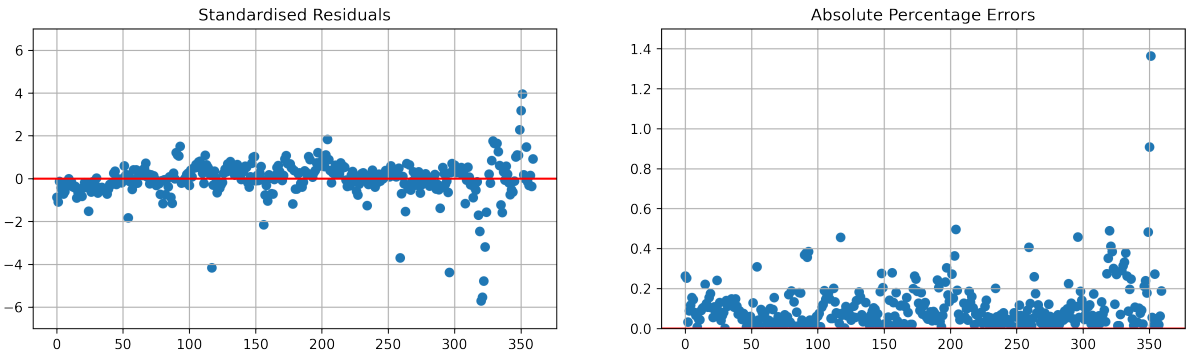

Figure 5.1.1: Standardised Residuals and Absolute Percentage Errors on the original data set using MLR

5.1.2 SARIMAX

The exhaustive search found that the optimal hyperparameters for the model resulting in the lowest AIC were

$$\text{SARIMAX}(6,1,4)(3,0,1)_7(X).$$

In comparison, this model was the most inconsistent of all the models considered in this thesis. It is clear that the features chosen could not appropriately model the variability of the data which can be clearly seen in the scatter plots in figure 5.1.2. In table 5.1.2 there can also be seen a high variability in the evaluation metrics. Interestingly though is that this model had the best performance modelling July.

Month	MAE	RMSE	MAPE
January	1 206 176	1 481 349	0.118
February	3 237 740	3 496 870	0.323
March	1 238 915	1 663 061	0.108
April	1 629 971	2 686 851	0.138
May	2 775 906	2 976 010	0.280
June	1 995 523	2 520 239	0.164
July	801 277	1 007 909	0.090
August	2 224 830	2 496 668	0.200
September	1 453 269	1 961 508	0.126
October	1 560 417	2 454 122	0.120
November	3 364 202	5 468 381	0.143
December	3 377 557	4 666 323	0.342
Average	**2 072 149**	**2 739 941**	**0.179**

Table 5.1.2: Evaluation Metrics using SARIMAX on the original data set

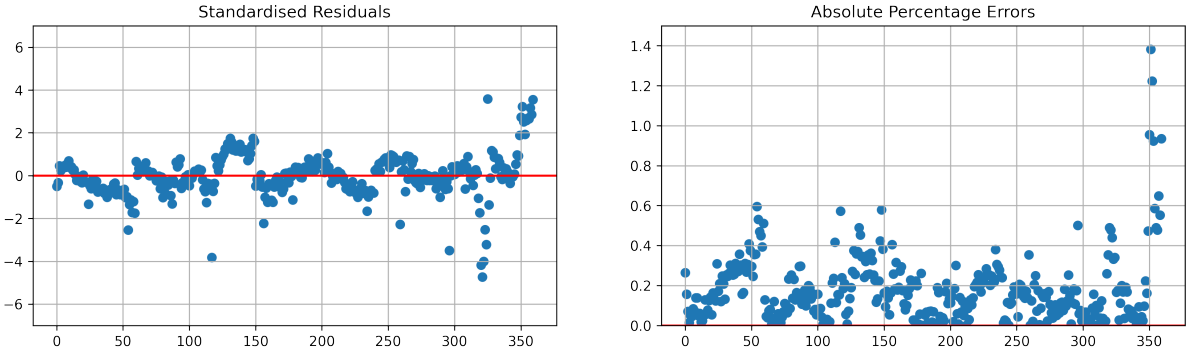

Figure 5.1.2: Standardised Residuals and Absolute Percentage Errors on the original data set using SARIMAX

5.1.3 XGBoost

The hyperparameter tuning resulted in the following hyperparameters for the XGBoost model:

Hyperparameter	Values(s)
Max depth	3
Number of estimators	750
Column samples per tree	0.9
Subsample ratio	0.9
Min child weight	1
L2 Regularisation	10
Learning rate	0.3

Table 5.1.3: Values of hyperparameters for XGBoost on the original data set

The XGBoost model gave the most consistent results as can be seen in table 5.1.4 and figure 5.1.3. In disregard of the evaluation metrics for January, this model gave the best performance. It follows weekly patterns and special events very well. However, it was more difficult to follow unexplained days (either an increase or a decrease). If looking at the period around black week it seems to overestimate quite a bit but also underestimate the days prior to Black Friday specifically.

Month	MAE	RMSE	MAPE
January	2 680 976	2 793 501	0.262
February	729 226	1 085 408	0.069
March	796 274	1 083 890	0.070
April	1 205 089	2 110 617	0.112
May	1 101 240	1 312 329	0.105
June	1 098 120	1 508 774	0.092
July	1 428 438	1 766 201	0.158
August	986 723	1 153 545	0.088
September	1 149 753	1 814 161	0.094
October	1 125 106	2 043 051	0.089
November	2 941 303	4 81 9919	0.160
December	1 778 975	2 434 240	0.146
Average	**1 418 435**	**2 230 142**	**0.121**

Table 5.1.4: Evaluation Metrics using XGBoost on the original data set

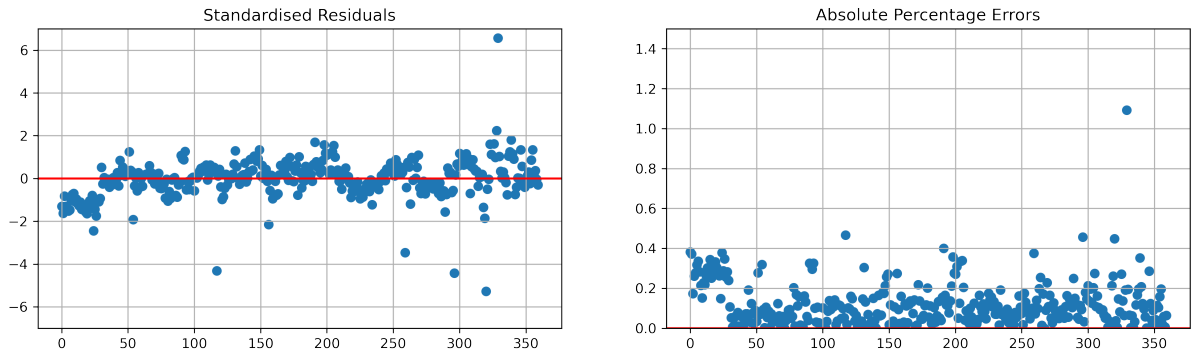

Figure 5.1.3: Standardised Residuals and Absolute Percentage Errors on the original data set using XGBoost

5.1.4 LSTM

The hyperparameter tuning resulted in the following hyperparameters for the LSTM model:

Hyperparameter	Values(s)
Number of layers	1
Number of neurons	128
Batch Size	16
Dropout level	0.25
Learning rate	0.0009476

Table 5.1.5: Values of hyperparameters for LSTM on the original data set

As can be seen in table 5.1.6 this model gave the largest forecasting errors with consistently high evaluation metrics on almost every month. Also from figure 5.1.4 one can see that month long chunks of the forecast deviate more than 20%. What is peculiar is that the worst month is October, usually one of the better months for the other models, whereas it shows very strong results during January, November and December for which are usually the more difficult months overall for the other models. If it was not for an overestimation prior to singles day and an underestimation exactly at singles day, November would probably have been below 10%.

Month	MAE	RMSE	MAPE
January	782 675	984 683	0.076
February	2 148 600	2 289 569	0.213
Marsh	2 283 909	2 406 000	0.217
April	2 050 804	3 046 684	0.178
May	2 828 903	3 070 363	0.266
June	2 321 754	2 787 518	0.201
July	2 718 367	2 820 854	0.292
August	3 469 706	3 656 920	0.319
September	2 218 545	2 799 463	0.183
October	3 922 367	4 407 323	0.334
November	2 089 362	2 943 963	0.118
December	1 303 459	1 665 444	0.127
Average	**2 344 871**	**2 739 899**	**0.210**

Table 5.1.6: Evaluation Metrics using LSTM on the original data set

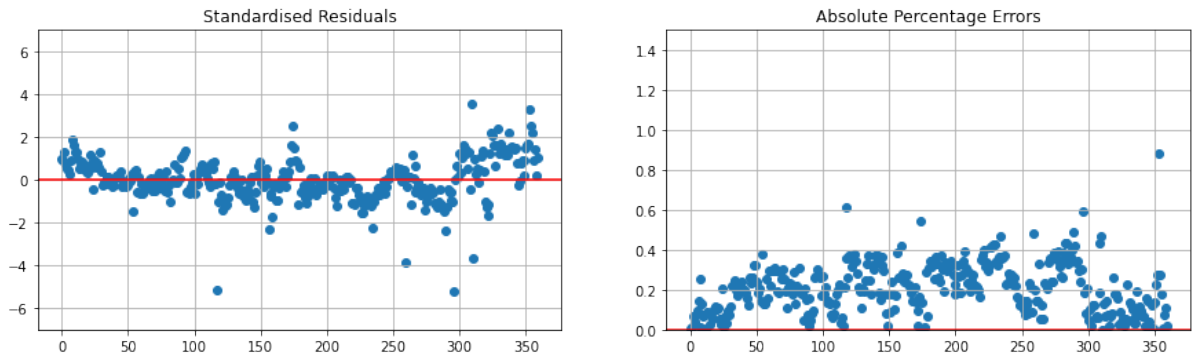

Figure 5.1.4: Standardised Residuals and Absolute Percentage Errors on the original data set using LSTM

5.2 Smoothed data set

Section 5.2 follows the same structure as 5.1. But here results from the smoothed data set are presented.

5.2.1 MLR

On the smoothed data set the MLR was the best performing model in regards to both the MAE and the MAPE. In table 5.2.1 it can be seen that the performance is consistent throughout the year, with December being the only exception. This can also be seen in the far right of the scatter plots in figure 5.2.1.

Month	MAE	RMSE	MAPE
January	1 243 277	1 402 496	0.121
February	908 845	1 221 388	0.087
March	743 570	1 016 357	0.067
April	1 117 233	1 415 064	0.122
May	890 083	1 038 534	0.087
June	991 872	1 240 330	0.091
July	1 115 970	1 380 814	0.128
August	761 930	989 698	0.071
September	790 481	1 119 181	0.067
October	889 269	1 221 562	0.076
November	1 869 294	2 469 021	0.115
December	1 865 608	2 693 244	0.189
total	**1 098 953**	**1 530 024**	**0.102**

Table 5.2.1: Evaluation Metrics using MLR on the smoothed data set

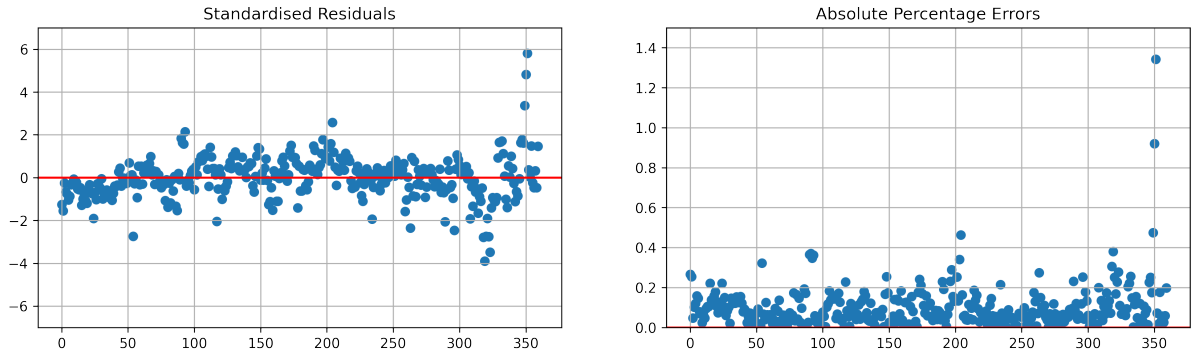

Figure 5.2.1: Standardised Residuals and Absolute Percentage Errors on the smoothed data set using MLR

5.2.2 SARIMAX

The exhaustive search found that the optimal hyperparameters for the model resulting in the lowest AIC were

$$\text{SARIMAX}(6,1,3)(3,0,1)_7(X). \tag{5.1}$$

The model has bad performance in comparison to the other models once again. It can be seen in table 5.2.2 that this model has the highest average of the evaluation metrics. There is also a high variability in performance for the different months. The scatter plots in figure 5.2.2 shows that there are clear patterns in the residuals that the model was not able to account for.

Month	MAE	RMSE	MAPE
January	1 980 360	2 418 351	0.191
February	3 316 277	3 591 960	0.329
March	997 833	1 276 625	0.092
April	1 883 898	2 289 319	0.178
May	2 210 781	2 421 593	0.226
June	1 783 351	2 145 650	0.151
July	745 663	943 648	0.081
August	1 695 317	1 977 018	0.149
September	1 213 890	1 414 733	0.113
October	1 758 637	2 024 249	0.150
November	2 135 574	3 021 256	0.123
December	3 651 662	4 883 746	0.369
Average	**1 947 770**	**2 367 346**	**0.179**

Table 5.2.2: Evaluation Metrics using SARIMAX on the smoothed data set

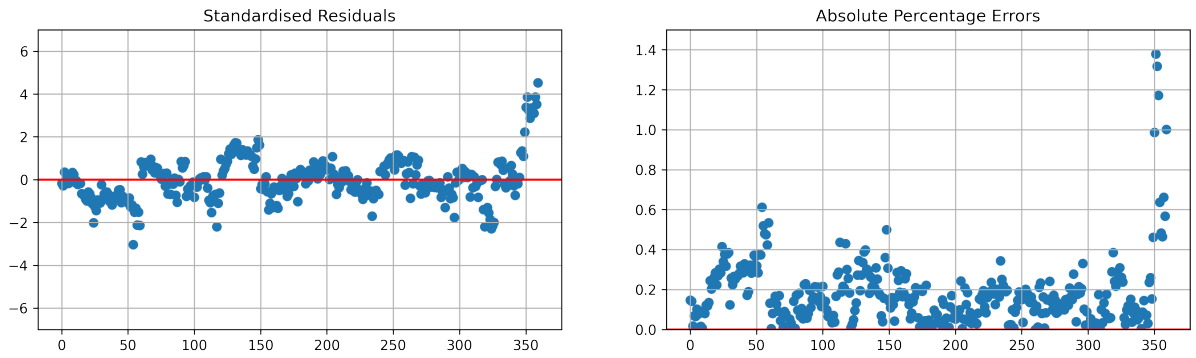

Figure 5.2.2: Standardised Residuals and Absolute Percentage Errors on the smoothed data set using SARIMAX

5.2.3 XGBoost

The hyperparameter tuning resulted in the following hyperparameters for the XGBoost model on the smoothed data set:

Hyperparameter	Values(s)
Max depth	2
Number of estimators	500
Column samples per tree	0.6
Subsample ratio	0.5
Min child weight	2
L2 Regularisation	10
Learning rate	0.3

Table 5.2.3: Values of hyperparameters for XGBoost on the smoothed data set

The XGBoost model had the best performance in regards to RMSE. From table 5.2.4 it is evident that the model would be the best performing model overall if January was not to be considered. With the exception of January the XGBoost models performance is highly consistent throughout the year which can be seen in figure 5.2.3. Compared to the original data set it is mostly November and December that yields better results which is expected.

Month	MAE	RMSE	MAPE
January	2 675 304	2 744 741	0.265
February	687 734	975 408	0.066
March	670 705	909 193	0.060
April	975 854	1 228 994	0.106
May	938 988	1 093 212	0.090
June	847 957	1 061 242	0.074
July	1 130 944	1 397 085	0.124
August	874 194	1 043 961	0.078
September	880 055	1 084 703	0.080
October	797 211	1 099 387	0.069
November	1 305 440	1 880 535	0.096
December	1 556 894	1 918 640	0.134
Average	**1 111 773**	**1 465 945**	**0.104**

Table 5.2.4: Evaluation Metrics using XGBoost on the smoothed data set

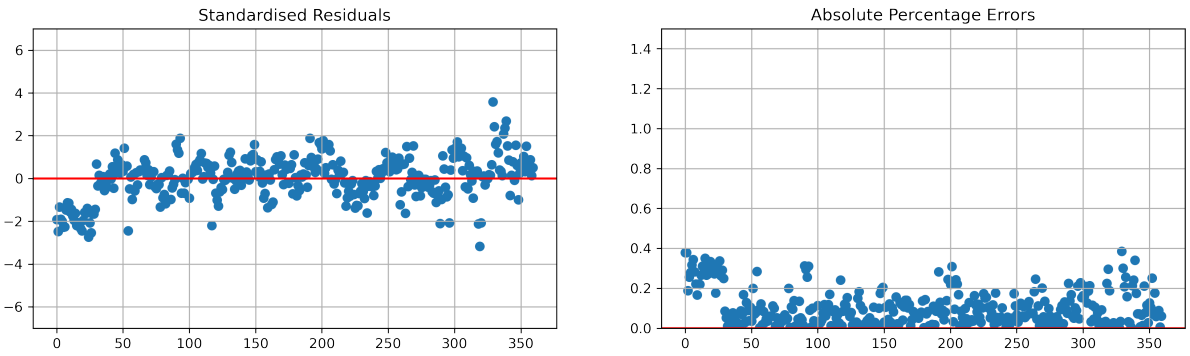

Figure 5.2.3: Standardised Residuals and Absolute Percentage Errors on the smoothed data set using XGBoost

5.2.4 LSTM

The hyperparameter tuning resulted in the following hyperparameters for the LSTM model on the smoothed data set:

Hyperparameter	Values(s)
Number of layers	1
Number of neurons	64
Batch Size	16
Dropout level	*none*
Learning rate	0.0011256

Table 5.2.5: Values of hyperparameters for LSTM on the smoothed data set

The LSTM saw a great improvement when applied to the smoothed data set. From

table 5.2.6 it is evident that all the evaluation metrics have improved significantly and have higher consistency throughout the year. From figure 5.2.4 its possible to observe that there are no clear patterns in the standardised residuals and that a majority of the deviations are below 20%. Most are within 10% except for December which saw an increase compared to the original data set. One of the reasons for this is that the model seems to suggest an increase of sales prior to December 25th and thus showing high sales during December 24th which is always to lowest during the month.

Month	MAE	RMSE	MAPE
January	1 058 505	1 264 000	0.106
February	821 539	1 098 582	0.085
Marsh	1 360 792	1 644 637	0.124
April	1 037 711	1 336 003	0.106
May	995 143	1 254 519	0.098
June	1 798 706	2 010 939	0.166
July	961 912	1 134 073	0.105
August	1 245 885	1 360 209	0.114
September	1 276 876	1 451 910	0.112
October	1 051 314	1 261 979	0.092
November	1 511 455	1 991 108	0.095
December	1 579 230	2 153 000	0.162
Average	**1 224 922**	**1 496 747**	**0.114**

Table 5.2.6: Evaluation Metrics using LSTM on the smoothed data set

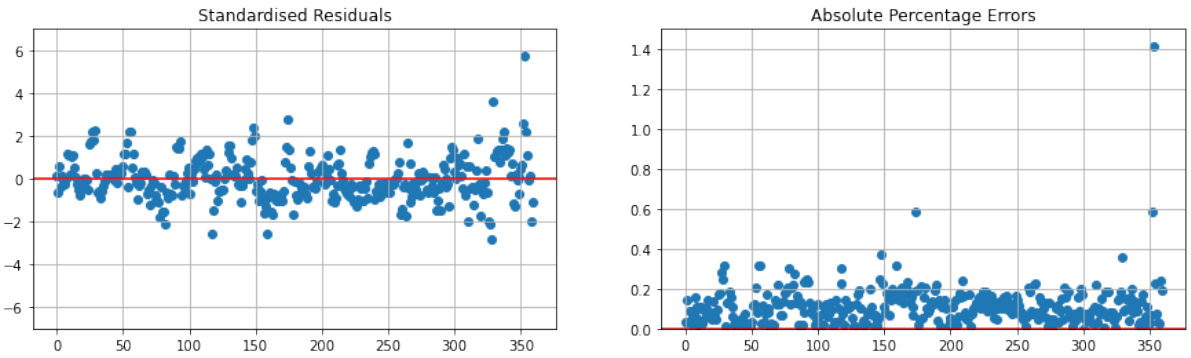

Figure 5.2.4: Standardised Residuals and Absolute Percentage Errors on the smoothed data set using LSTM

Chapter 6

Discussion

In this chapter analysis of results in chapter 5 and a discussion of the different models will be conducted. The models will be discussed based on the results together with additional information that was noted during the model exploration phase. Worth noting is that the additional information provided are based on the experience from the exploration phase and not necessarily provided in the aforementioned chapter.

6.1 Analysis of results

On both the original data set and the smoothed data set the XGBoost model gave the best overall performance with low scores on all of the evaluation metrics, and good looking standardised residuals and absolute percentage errors. The only downside of the model was that it struggled to forecast the month of January on both of the data sets. Both the satisfactory performance on the months February - December, and the bad performance on January can probably be attributed to the models structure, and the choice of features. As discussed in section 3.4 the XGBoost model can, unlike the other models, handle ordinal data. What this means is that fewer features were used compared to the other models to describe the same relationships. This also means that it will perform badly when met with a new ordinal value that it has not yet seen before, e.g. when forecasting January while only having data up until December the previous year.

The second best overall performance on both of the data sets came from the MLR. On the original data set it consistently resulted in low scores on the evaluation metrics

in comparison to the other models, with the only exception being December. On the other hand it performed flawlessly on the smoothed data set. The reason for the bad performance in December using the original data set can probably be attributed to the fact that the MLR can not handle ordinal features, unlike the XGBoost. This resulted in there being more than triple the amount of features in this model compared to the XGBoost, many of which were in December. it can be argued that there simply was not enough data for the MLR to learn the regressor coefficients for the features in December properly.

The LSTM model performed very differently depending on which data set was used. On the original data set it was the worst performing model, all metrics considered, and the residuals showed some patterns that the model was not able to account for. However, on the smoothed data set the model's performance increased profoundly and there were no clear patterns in the residuals. As the LSTM is a sequential model it makes sense that its performance should increase when trained on less erratic data which is exactly what is being done in the case of using the smoothed data set. This relaxation could also be attributed to what is briefly described in subsection 4.2.4. Since NN's usually performs better when the seasonality component is removed, conversely, removing erratic behaviour and non-linear patterns should help it learn the seasonal component better. This could also explain as to why the model performed worse on the original data set yet still being able to learn non-linear patterns in the data such as peaks on specific days. Another reason that could contribute to the difficulties for the LSTM network to predict on the original data set is the lack of total data points. Of course, most of the models in this thesis would benefit from more data, but it is particularly true for the LSTM as it need large amounts of data to learn long-term patterns as discussed in chapter 2. With that said, the feasibility of this model might increase as time passes and more data is available for training, given that the data is more consistent over time. An additional insight is its possibility to learn longer forecasting time steps. Usually it is said that a forecasting horizon of 10 steps yields good results where the next coming steps usually shows diminishing results. In this case, there were little difference between the 30 steps and proves that this is an area that is worth investigating.

The worst overall model, on both data sets considered, was the SARIMAX. It resulted in the second worst and worst evaluation metrics on the original and smoothed data set respectively. In both cases clear patterns emerge in both the standardised residuals

and the absolute percentage errors meaning that the model was not able to capture all variability. It is clear that there are difficulties using the symbiotic nature of the SARIMAX model compared to simply using an MLR alone.

Another aspect worth noting for each model is the problem of domain knowledge during feature engineering. For this thesis, the decision was made to use similar features for the models. This could induce limitations as deep diving into respective model with proper knowledge within e-commerce could provide better results. This is for example evident when comparing the original data set and the smoothed data set as the LSTM error dropped significantly Smoothing the data shows its ability to learn certain patterns in the data. However, what usefulness does it bring to remove information from the data if it does not provide more accurate forecasts? This type of application can be of use in combination with other models that are able to forecast the difference between the smoothed and the original data set. That is, one model that predicts during the erratic and specific cyclical events such as Black Friday and one for longer dependencies. However, if only used as an instrument to alleviate complexities for the model, it should not be considered sufficient during forecasting.

6.2 Model evaluation

In this section follows a short evaluation of each model with regard to the following criteria:

- Implementation
- Feature Creation
- Interpretability of Code
- Interpretability of Method
- Computational Time
- Performance

MLR

The MLR was easy to build, train, and forecast with using only a few lines of code. However, since this type of regression is unable to handle ordinal data some work had

to be done with the features in order for the model to work properly. All categorical variables had to be one-hot encoded which resulted in a much messier data set. Since this is such a standard way of making predictions the code required is easy to follow and adopt to the problem at hand, and no hyperparameter tuning was required. The mathematical methods used are well known and is taught to almost all university students within a technical field, and it is easy to understand how the different regressor coefficients directly affect the predictions. It took only a few seconds to train and it produced consistently good forecast regardless of the type of data it was fed.

SARIMAX

The SARIMAX was not as straight forward to implement as the MLR. This model requires a more thorough initial analysis of the time series before any modelling can start. Generally, one simply looks at the Partial AutoCorrelation Function (PACF) and the AutoCorrelation Function (ACF) to decide the hyperparameters based on inspection. Instead of that approach, it was opted to perform an exhaustive search and pick the model that yielded the lowest AIC score, i.e., the lowest Akaikes Information Criterion. Doing so helps automate the procedure of picking the best parameters. This exhaustive search resulted in a code that had a low level of interpretability and was quite difficult to understand. Furthermore, all the categorical features had to be one-hot encoded for the model to work properly. The mathematical knowledge require to understand how the model works is quite high and it is not as intuitive as the MLR. It takes up to several hours to find the right hyperparameters and it takes quite long time to make forecasts, and despite this it had the worst performance overall of all the models considered in this thesis.

XGBoost

It is easy to quickly crown the XGBoost as the most impressive of the model in terms of overall performance. This model yields very good overall results even though it is not necessarily strong around Black Friday as compared to the MLR model. Why this is the case could be the fact that, when observed, the overall trend of Black Friday being the primary sales day is changing and being more distributed over the entire week. Thus, the previous data does not reflect the increased sales earlier in the week which could explain the slight decrease in sales of Black Friday in 2021.

Even if it is possible to understand how the XGBoost make certain decision based on the feature importance, it is still difficult to understand as to why certain values are being generated. One explanation in the result section could be the fact that it only learns from the temporal features and makes splits in relation to those. There are no previous values in consideration in an autoregressive manner. It should be stated that it was tried to use lagged values from the previous month, rolling statistics and similar measures. However, these did not improve the model, rather the exact opposite. From this, it is of course reasonable that the model generates values based on previous data and not learning particular non-linear movements or changing patterns over time. As described in the previous section, certain feature engineering could of course improve this flaw and shows the importance of good domain knowledge within the field of e-commerce. Despite this, the model is still very consistent and shows good results. In addition to that it is very forgiving to those who do not have previous knowledge in operating such model and how to incorporate features. It is computationally quick both during training and prediction making it easy to tweak certain factors. Since the data also does not need prior data pre-processing as it is indifferent to scaling (see section 3.4), together with the fact that it is able to handle ordinal features, it is also very intuitive during data pre-processing. To summarise, the XGBoost provides a good combination of performance, computational efficiency and also being intuitive to use.

LSTM

As for last, the LSTM could be considered a wildcard compared to the other models. Results were varying depending on different aspects such as what type of features it was fed, tweaking of hyperparameters and usage of different amount of previous data. Also, retraining of the model without properely saving it could yield different results. This shows the stochastic nature caused by the gradient descent during model training. Even if overall metrics did not change much, the forecasting results differed. When the data set was smoothed the model showed very good results over the entire year. Similar results were obtained when only using temporal features as cyclic functions such as weekday and months. The trade-off being subpar performance during important periods, such as November, completely missing the increased sales happening those particular days. On the contrary, when continuously being re-trained on the original data set together with a higher feature dimension including one-hot encoded features

for the black week period and Christmas, it instead showed worse results overall, but best during November and December. This shows the complexities of implementing a good performing LSTM, especially when there is not large amount of data available. The LSTM also has disadvantages when it comes to computational complexities as it was by far one of the more difficult models to train. Data pre-processing also needed several steps remapping the data into a specific input shape depending on how the sequence input should look like for the model. Further, it is widely considered a black box and this is truly the case as it is very difficult to understand how it makes certain decision based on the calculated wights, biases and the internal cells memory. Thus, juxtaposed to the XGBoost, this model does not provide a good trade-off between model performance and complexity. However, it is one of the models that shows good results and makes predictions based on the memory of previous trends in the data. This would mean that it could find patterns that are yet to be seen which could explain the good results during November and December on the original data set. Also, the possibility to use the LSTM as a global model for several time series is a huge upside as described in [3]. This will be further discussed in the next section.

6.3 Application at QLIRO

The above discussion shows the difficulties of only finding one refined model that can be universally used. As for QLIRO, or any actor practicing demand forecasting, each model needs to fulfill certain criteria. For this thesis, three such criteria will be in consideration. The first two are performance and usability which importance has already been discussed in this chapter. Model life cycle is the third criteria which indicates how long the model could be considered useful within their system before being replaced. Since each model has its drawbacks, it is difficult to only recommend one such model. However, based on the results and the experience of implementing such model, the XGBoost definitely proves to be a worthy candidate. It fulfills all of the aforementioned criteria. And by applying proper domain knowledge during feature engineering the XGBoost could probably provide even better results. Yet, the same could be said of the other models as well. In this current setup the XGBoost does not learn from previous values in sequences like the LSTM and could therefore miss certain trends or non-linear patterns in the data. Thus, a combination could be preferred at the cost of added complexity by having to maintain and understand

different applications simultaneously. As for the usefulness of the model, it can help automate internal planning especially during a long period such as one month ahead of time. However, it depends on the accepted level of forecasting error. Even if the models shows good results, they still produce high error margin in terms of absolute percentage. If comparing this to using a naive approach of looking at similar days for the previous year and adding an increasing trend then the naive approach could suffice. However, such planning is time consuming as it needs more manual work, especially since the data contains a lot of other cyclical patterns that is not repeated on the same date each year. This is were forecasting models are useful as they could give quick suggestions.

Another aspect worth discussing that was mentioned in the previous section is the use of global models. Both the XGBoost and the LSTM have shown the possibility to be used as a single model over several similar time series. This could provide a useful application if looking at specific market segments or industries and grouping them accordingly. By doing so, more data becomes available and the models can learn from the certain groups if they show similar patterns, e.g., high sales during weekends and low sales during beginning of weeks. This could help alleviate the problems of limited data that was discussed previously. It could also provide other applications as an option to judgmental forecasting. Judgmental forecasting is a forecast made based on intuition and research without prior data. Usually it is done when implementing a new product line or in QLIRO's case it could be introducing a new merchant within their system. If such a model provide sufficient performance, it could provide a very quick way to perform judgmental forecasting.

6.4 Limitations

There are three main limitations with the work done in this thesis: Data scarcity, changes in consumer patterns, and the inclusion or exclusion of merchants. Data scarcity in this case refers to the fact that the data set is rather short. Short in terms of measuring yearly cyclical periods. In total there were only seven observations for special events such as Black Friday. On top of that, the time interval was shortened due to the development of the data which led to only including three such periods for model training. Data scarcity also relates to that the data is on a daily frequency together with limited information regarding the merchants. More granular data could improve the

performance of the models if it was possible to include product categories, regions and hourly reservations. It could also be whether a large merchant, or several of them, are about to launch a campaign that could influence the sales patterns. As for changes in consumer patterns, the first aspect of consumer being recurrent costumers or new entrants are of course influencing the data. In relation to this, e-commerce has seen an increased inclusion within retail, especially so during the pandemic of COVID-19 which has led some customers to use internet as their primary purchase infrastructure. This means that the patterns in the data used for training have seen changes since the pandemic began and including certain periods were low and high sales periods that was not observed before. This is something only time will solve, and the more the situation stabilises the market can saturate and converge to more repeating patterns. Lastly inclusion or exclusion of large merchants might change the patterns in the time series. Some might also rework their sales strategies meaning previous patterns are obsolete. Such changes can be difficult to explain at this point and thus changing how the data set behaves over time.

6.5 Future work

First off, each model could be improved using proper domain knowledge within e-commerce during feature engineering and data pre-processing. It could therefore be useful with additional investigation as to what extent certain features influence each model. This could for example be shown with the smoothed data set which relaxed the complexity and thus improved model performance. Additional information could also be included using exogenous features such as temperature, humidity and social media trends, all of which might influence sales patterns. Evaluation of training techniques and forecasting horizons is another area to be investigated that has the potential to improve the performance of the models. Secondly, how to build the models could be further investigated. Using ensemble techniques such as meta features and decision trees to decide the predictions of several models could improve the overall performance. By doing so, if possible, the model's strengths combined would reduce their individual weaknesses over different periods whether it be a regular period or a more erratic one. Another area of investigation in relation to this is the approach to build one global model over several time series. In QLIRO's case, additional information from different market segments and industries could be used to produce

several time series which gives more training data for a global model as discussed previously in this chapter. Such model has the potential to learn more patterns within the data which could improve forecasting performance. Lastly, since data scarcity was one limitation, further investigation on how to produce artificial data for the models could help improve their performance within the field of e-commerce forecasting.

Chapter 7

Conclusion

In this thesis, four models has been constructed and evaluated based on their forecasting accuracy and the complexity to implement respective model. Out of the four different models, it was found that the forecasting accuracy varied significantly. According to almost all evaluation measures and plot inspections two models stood out from the rest, those being the MLR and the XGBoost. These had good accuracy and were computational efficient to use. The other two models, SARIMAX and LSTM, had inconsistent forecasting accuracy and was either hard to work with or to optimise. Yet the LSTM showed substantial improvements when smoothing of the erratic behaviour in the data was done. It also showed good results during certain periods compared to the other models. This concludes the answer to the research question as to how modern statistical methods fares on QLIRO's sales reservation volume data. Since the MLR and the XGBoost yielded similar results it was also interesting to compare their usability. That is, which model that should be preferred for implementation at an organisations such as QLIRO. Between the two, the deciding factor came down to which one is easier to use. According to this perspective the XGBoost has a clear advantage since it handles features in a more intuitive way compared to the MLR. It is thus easier to maintain the model by trying out new ideas and further tuning it, but also when passing it on to new colleagues with less experience. Suggestion for further research has also been provided that could improve the accuracy of the different models but also increase the usability of certain models. Such improvements could be the use of ensemble techniques, use of a global model, and applying more domain knowledge from the field of e-commerce during feature engineering.

Bibliography

[1] Arunraj, N, Ahrens, D, and Fernandes, M. “Application of SARIMAX Model to Forecast Daily Sales in Food Retail Industry”. eng. In: *International journal of operations research and information systems* 7.2 (2016), pp. 1–21. ISSN: 1947-9328.

[2] Auffarth, B. *Machine Learning for Time-Series with Python: Forecast, Predict, and Detect Anomalies with State-Of-the-art Machine Learning Methods*. eng. Birmingham: Packt Publishing, Limited, 2021. ISBN: 1801819629.

[3] Bandara, K, Shi, P, Bergmeir, C, Hewamalage, H, Tran, Q, and Seaman, B. “Sales Demand Forecast in E-commerce Using a Long Short-Term Memory Neural Network Methodology”. eng. In: *Neural Information Processing*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 462–474. ISBN: 3030367177.

[4] Bilińska Reformat, K and Dewalska Opitek, A. “E-commerce as the predominant business model of fast fashion retailers in the era of global COVID 19 pandemics”. eng. In: *Procedia computer science* 192 (2021), pp. 2479–2490. ISSN: 1877-0509.

[5] Brockwell, P. *Introduction to Time Series and Forecasting*. eng. 3rd ed. 2016.. Springer Texts in Statistics. 2016. ISBN: 3-319-29854-2.

[6] Chai, T and Draxler, R.R. “Root mean square error (RMSE) or mean absolute error (MAE)? -Arguments against avoiding RMSE in the literature”. eng. In: *Geoscientific model development* 7.3 (2014), pp. 1247–1250. ISSN: 1991-959X.

[7] Chandra, R, Goyal, S, and Gupta, R. “Evaluation of Deep Learning Models for Multi-Step Ahead Time Series Prediction”. eng. In: *IEEE access* 9 (2021), pp. 83105–83123. ISSN: 2169-3536.

[8] Chatfield, C. *The analysis of time series : an introduction*. eng. 4. ed.. London: Chapman and Hall, 1989. ISBN: 0-412-31810-5.

[9] Chen, T and Guestrin, C. "XGBoost: A Scalable Tree Boosting System". eng. In: *Proceedings of the 22nd ACM SIGKDD International Conference on knowledge discovery and data mining*. Vol. 13-17-. KDD '16. ACM, 2016, pp. 785–794. ISBN: 1450342329.

[10] Chollet, F et al. *Keras*. `https://keras.io`. 2015.

[11] Clark, S, Hyndman, R, Pagendam, D, and Ryan, L. "Modern Strategies for Time Series Regression". eng. In: *International statistical review* 88.1 (2020), S179–S204. ISSN: 0306-7734.

[12] Dickler, J. *Buy now, pay later plans are booming in the Covid economy*. URL: `https://www.cnbc.com/2020/12/14/buy-now-pay-later-plans-are-booming-in-the-covid-economy.html`. (accessed: 18.01.2022).

[13] Gareth, J, Witten, D, Hastie, T, and Tibshirani, R. *An introduction to statistical learning: With applications in R*. eng. Vol. 103. Springer texts in statistics. New York: Springer, 2013. ISBN: 9781461471370.

[14] Garnier, R and Belletoile, A. "A multi-series framework for demand forecasts in E-commerce". eng. In: (2019).

[15] Géron, A. *Hands-on machine learning with Scikit-learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems*. eng. Second edition.. 2019. ISBN: 1-4920-3259-X.

[16] Harris, C et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: `10.1038/s41586-020-2649-2`. URL: `https://doi.org/10.1038/s41586-020-2649-2`.

[17] Hastie, T, Tibshirani, R, and Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. eng. Springer Series in Statistics. New York, NY: Springer, 2013. ISBN: 9780387952840.

[18] Hewamalage, H, Bergmeir, C, and K, Bandara. "Recurrent Neural Networks for Time Series Forecasting: Current status and future directions". eng. In: *International journal of forecasting* 37.1 (2021), pp. 388–427. ISSN: 0169-2070.

[19] Hunter, J. “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[20] Hyndman, R and Athanasopoulos, G. *Forecasting: Principles and Practice*. English. 3rd. Australia: OTexts, 2021.

[21] Montgomery, D. *Introduction to linear regression analysis*. eng. Fifth edition.. Wiley series in probability and statistics. 2012. ISBN: 1-119-18017-1.

[22] Myttenaere, A, Golden, B, Grand, B, and Rossi, F. “Mean Absolute Percentage Error for regression models”. eng. In: *Neurocomputing (Amsterdam)*. Advances in artificial neural networks, machine learning and computational intelligence — Selected papers from the 23rd European Symposium on Artificial Neural Networks (ESANN 2015) 192 (2016), pp. 38–48. ISSN: 0925-2312.

[23] Pedregosa, F, Varoquaux, G, Gramfort, A, Michel, V, Thirion, B, Grisel, O, Blondel, M, Prettenhofer, P, Weiss, R, Dubourg, V, Vanderplas, J, Passos, A, Cournapeau, D, Brucher, M, Perrot, M, and Duchesnay, E. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[24] Saunders, M, Lewis, P, and Thornhill, A. *Research methods for business students: 7. ed.* eng. Harlow u.a: Pearson, 2016. ISBN: 1292016620.

[25] Schak, M and Gepperth, A. “A Study on Catastrophic Forgetting in Deep LSTM Networks”. eng. In: *Artificial Neural Networks and Machine Learning – ICANN 2019: Deep Learning*. Vol. 11728. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 714–728. ISBN: 3030304833.

[26] Seabold, S and Perktold, J. “statsmodels: Econometric and statistical modeling with python”. In: *9th Python in Science Conference*. 2010.

[27] Smith, T et al. *pmdarima: ARIMA estimators for Python*. [Online; accessed 2021-3-1]. 2017–. URL: http://www.alkaline-ml.com/pmdarima.

[28] Statista.com. *E-commerce share of total global retail sales from 2015 to 2024*. URL: https://www.statista.com/statistics/534123/e-commerce-share-of-retail-sales-worldwide/. (accessed: 18.01.2022).

[29] SvenskHandel. *Läget i handeln, 2021 års rapport om branschens ekonomiska utveckling*. URL: https://www.svenskhandel.se/rapporter/laget-i-handeln/. (accessed: 18.01.2022).

[30] team, The pandas development. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[31] Van Rossum, G and Drake, F. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[32] Zhang, L, Bian, W, Qu, W, Tuo, L, and Wang, Y. "Time series forecast of sales volume based on XGBoost". eng. In: *Journal of Physics: Conference Series*. Vol. 1873. 1. Bristol: IOP Publishing, 2021, p. 12067.