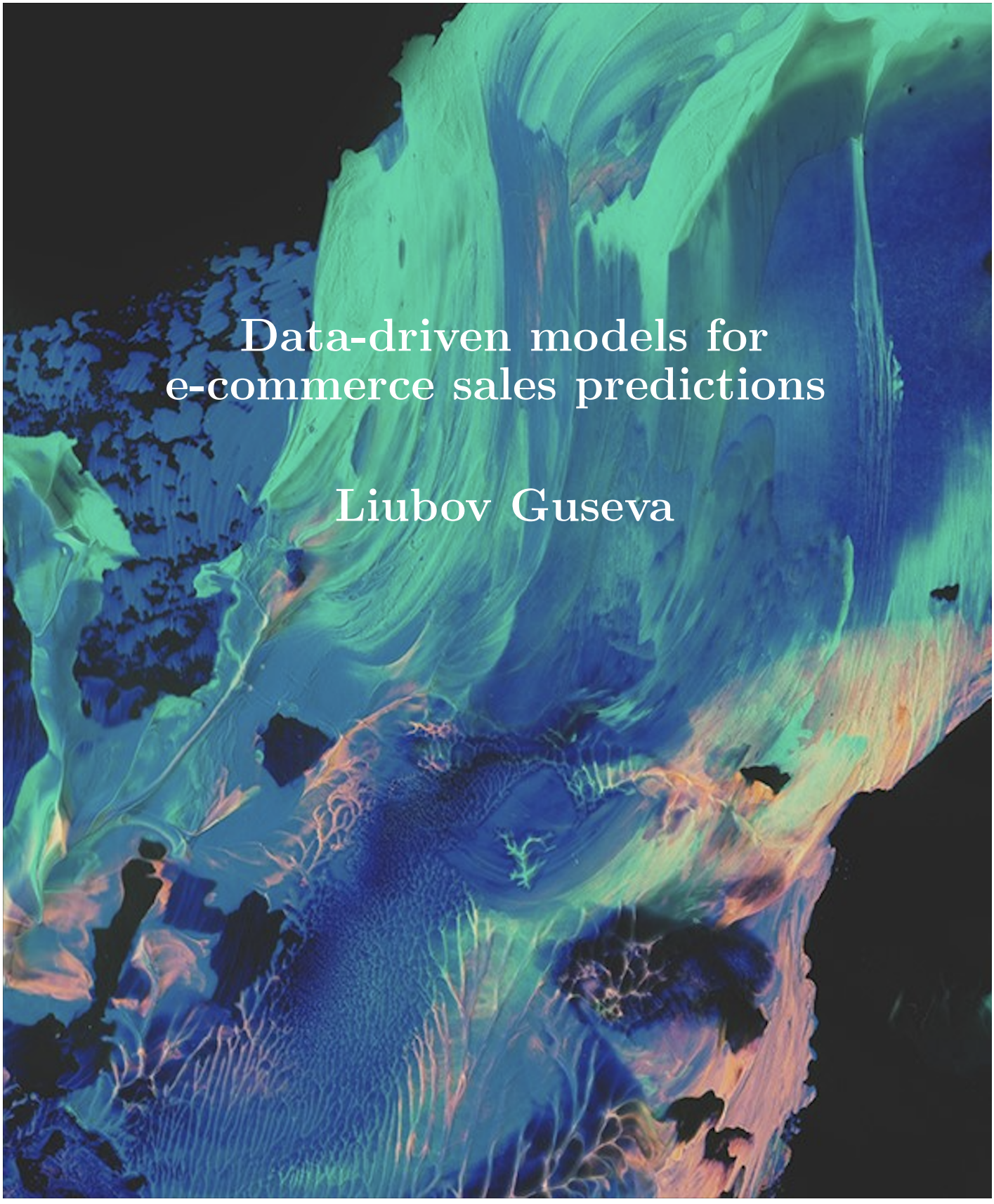
UMEÅ UNIVERSITY



Data-driven models for e-commerce sales predictions

Liubov Guseva

Master's Thesis in Engineering Physics

Spring term 2022

Data-driven models for e-commerce sales predictions

Liubov Guseva, lubagusevabg@gmail.com

Supervisors:	Martin Rosvall	Infobaleen
	Felix Djuphammar	Infobaleen
Examiner:	Ludvig Lizana	Department of Physics, Umeå University

Master of Science Thesis in Engineering Physics, 30 ECTS
Department of Physics
Umeå University
SE-901 87 Umeå, Sweden

Abstract

Future predictions have various applications, including stock prices, house market prices, and company sales. For sales, predictions can guide future expectations and suggest ways to cut costs. Due to the value of predictions, researchers have developed plenitude of prediction algorithms. Still, many companies use simplistic prediction algorithms that fail to provide accurate results. Also, the vast number of existing algorithms makes it difficult to find the best algorithm for a specific data set.

In this thesis, I predicted an e-commerce company's future sales based on its historical transaction data with three different models: the machine learning algorithm LSTM, a forecasting library released by Facebook called Prophet, and a model that I developed inspired by Prophet, called the Average Sales Prediction (ASP) model. I compared these models to each other and a benchmark model. The benchmark model I used is one of the simplistic algorithms that some companies currently use. It takes the mean value of the past month's sales to predict the upcoming day. Using the mean absolute percentage error (MAPE), I found that LSTM had the best overall performance on these data, with a MAPE of 18%. The second-best performing model was ASP, which resulted in a MAPE of 26%. Finally, Prophet resulted in a MAPE of 31%.

The results gauge the company's future performance and will help them improve its sales through streamlined workloads and better warehouse and transportation planning. The models can be enhanced further for sales that, for example, depend on the weather or other external factors.

Acknowledgements

I would like to express my deepest appreciation to my supervisors Martin and Felix, who always took their time and helped me. They have always been around for discussion and support during my thesis and guided me through the whole project. I am also extremely grateful to the rest of the Infobaleen team, who provided me with feedback during my work and were also always available for discussion. I also wish to thank Linn and Klas who also wrote their thesis at Infobaleen, for being a sounding board during this semester.

Finally, I would like to say a special thanks to my classmates and family for supporting me, not only through this thesis, but also through all my years at the university.

Contents

1 Introduction

Every day, companies collect a lot of data from many different sources. So how can they use these data to increase the future sales of the company?

For example, in online sales, when a customer makes a purchase, several categories of data are stored by the company, that includes time for the purchase, which items were included in the bill, from which category were those items chosen, the total amount spent on that purchase, and so on.

By analyzing and handling the data correctly, it is possible to make better sales suggestions for recurrent customers and predict how the future sales will look like. Such predictions can both help in a company's financial planning, and also have a positive environmental imprint. With accurate predictions, it is possible to reduce the transportation and storage of goods in the most efficient way.

Today the algorithms used in many companies for prediction purposes are simplistic and do not provide good results. In many cases, they are just serving as an approximate benchmark, but do not explain the results and the process of achieving them.

Due to high interest in predictions, many different algorithms have been developed over time for prediction purposes. The remaining struggle is that those algorithms are not data specific. Finding a suitable model for a specific data set is challenging.

This thesis focuses on the prediction of online sales for one particular firm. The data were provided to Infobaleen by an e-commerce company, the **client**. The predictions consist of general future predictions of sales and more detailed predictions on item category and customer level, which helps the client understand the full picture of future sales. The collected data is a **time series**, which "is a sequence of data points that occur in successive order over some period of time." [1]

The goal of this thesis is to find a suitable model for revenue predictions of the client's data. Those predictions had to cover weekly, monthly and quarterly periods as well as being applied to the total company revenue and item and customer category predictions. For item and customer predictions the time series was split into 24 smaller data sets that represented each category. Finally, they had to cover the possibility of calculating how long the current stock capacity will last. All models were also compared to a benchmark model.

To reach the goal I implemented three different algorithms and tested those on the provided data. These algorithms were chosen based on a preterm analysis of the provided time series and a pilot study, and included: LSTM, Prophet, and ASP models.

I found that LSTM has the best performance for most data sets. However, its biggest disadvantage is that it performs better with huge amounts of data. So in the cases of separate data sets, when some of them included a lot of zeros, it did not provide as good results. The other two models, Prophet and Mean model, performed similarly to each other with a slight advantage to the ASP model. Even though those models capture the mean value of the predictions quite well, daily predictions do not return as good results as the LSTM model.

From the results of this thesis, the client can better understand what to expect from future sales. It will also provide the client with a better benchmark of the total company sales, as well as sales on smaller categories level. Finally, it will help with better stock and workload planning

for the requested periods of time.

In Sec. 2 I introduce all the necessary theory for this thesis. Further on in Sec. 3, I describe my methods of analyzing the data and implementation of the algorithms. In Sec. 4 I show the prediction results, and in Sec. 5 and Sec. 6 I discuss them and draw conclusions. At the bottom, the references and appendix are presented.

2 Theory

In this section, the necessary theory is presented. It is covering some theory on the Time series and pre- and post-processing of data. Further, it covers three different models that were used for prediction purposes.

2.1 Time series

A time series is a sequence of data points that are measured over some period of time. The most common approach is working with discrete time series, where the observations are not continuous, but fixed in time. That can be yearly, monthly, daily, hourly, etc. time steps. In this thesis, the time step that was used is daily. In general, a time series in sales are built up from three different components: trend (long term change), seasonal (calendar-related changes), and irregular (short term changes). Based on those components a time series can be decomposed as:

$$y(t) = g(t) + s(t) + \epsilon_t, \tag{1}$$

here $g(t)$ corresponds to the trend component, $s(t)$ is the seasonal component and ϵ_t is the irregular component, also usually called noise.

The possibility for such decomposition results in a lot of different prediction models that can be used for time series. Some examples of such algorithms are linear regressions, machine learning algorithms, SARIMA models, etc. However, only three different models will be discussed in this work. Two of those models were chosen based on a previous study on a similar type of data [2].

2.2 Pre- and post-processing of data

Before starting to work with a time series it is quite important to visualize the data and pre-process it. From the visualization, one can receive a lot of information on the time series that can be useful for further processing and predictions. That information contains the absence of data for any timestamps, presence of any trends and seasonality in the series, and presence of outliers. To be able to make good predictions for an average day it is important to remove the missing data and the outliers from the series. The method used for outlier detection and removal will be shown further on in Sec. 2.2.1.

As for the post-processing, all data shown in this thesis was normalized due to secrecy. The method that was used for normalization will be explained in Sec. 2.2.3.

2.2.1 Outliers

Outliers are extremes in a data set, they can be either very small or very large values and in that way affect the overall observations from the data. That brings the need to discard them from the data set. The question arises, how can those values be detected and removed from the data? There are many different methods for that goal, but for the purpose of this thesis Box plot, also called Whisker's plot, was chosen. That is a graphical method that is based on finding quartiles and inter quartiles of the data, which define the upper and lower limits beyond which any value can be considered as an outlier. The upper and lower quartiles 25% and 75% of the data need to be identified, by letting **N** be the number of observations they can be found as:

$$\begin{aligned} Q_l &= 0.25(1+N) \\ Q_u &= 0.75(1+N). \end{aligned} \tag{2}$$

Further on an inter quirtile range (IRQ) is defined as:

$$IQR = Q_u - Q_l \tag{3}$$

To define upper and lower limit for the outlier detection following range is used:

$$\begin{aligned} \text{Lower limit} &= Q_l - 1.5IQR \\ \text{Upper limit} &= Q_u + 1.5IQR \end{aligned} \tag{4}$$

2.2.2 Error estimation

An important part while doing prediction is to know how well the model performs. That can be achieved by dividing the data set into two parts: training and testing sets. The training set is used to train the model and the test set is further on used to evaluate the error of the results. There are several different types of errors that can be used for such a purpose. In this thesis mean square error (MSE), mean average error (MAE) and mean average percentage error (MAPE) were used.

MSE is one of the most widely used types of error, it represents the square difference between actual and estimated values. The final error is calculated by:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \tilde{y}_i\right)^2, \tag{5}$$

here N is the total number of observations, y_i is the actual value and $\tilde{y}_i$ is the predicted value.

MAE is the average magnitude of errors in the forecasted data, it measures the accuracy for continuous variables. This error is calculated as following:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \tilde{y}_i\right|, \tag{6}$$

where once again N represents the total number of observations, y_i is the actual value and $\tilde{y}_i$ is the predicted value.

The final error that was used was MAPE which represents the average of the absolute percentage error between the predicted and the actual value. The error is calculated in the following way:

$$\text{MAPE} = \frac{1}{N}\sum_{t=1}^{N}\left|\frac{y_i - \tilde{y}_i}{y_i}\right| \tag{7}$$

As a result we have three error estimators that are good indicators of how well a model performs. The lower the error indicators are, the better the performance is [3].

2.2.3 Normalization

All data that is to be presented in this thesis has been normalized due to secrecy. Normalization is a statistical term that implies scaling of a data set in such way that the normalized data falls in the range between 0 and 1. In this thesis so called max-min normalisation was used. That is calculated by using:

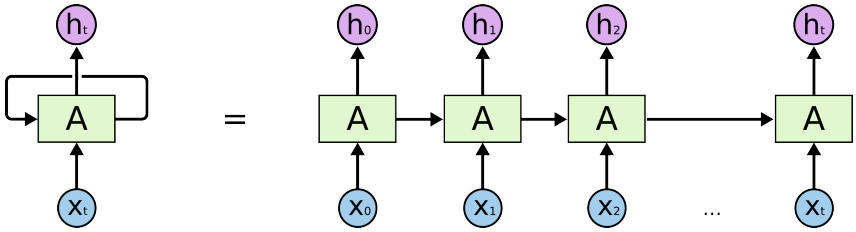
$$x_{\text{ norm }} = \frac{(x - x_{\min})}{(x_{\max} - x_{\min})}, \tag{8}$$

here x represents the value to be normalized, x_{min} and x_{max} represents minimum respective maximum value of the data set.

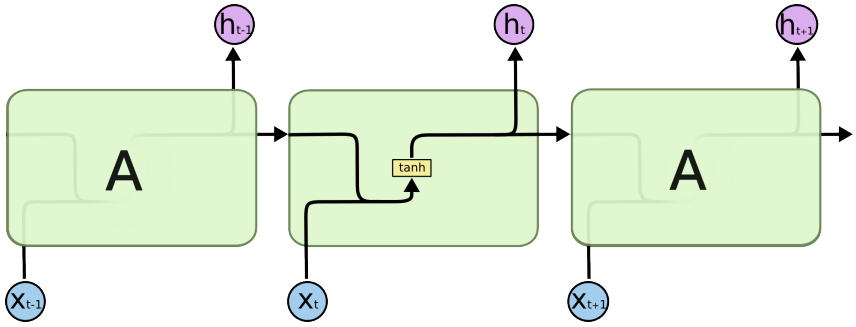
2.3 Recurrent Neural Networks

A recurrent neural network (RNN) is a supervised machine learning model. While being quite similar to artificial neural networks (ANN) they still differ a bit. ANNs usually deal with classification and regression problems, meanwhile, RNN works best with time series. In Fig. 1a a simple example of a RNN chain with one hidden layer is presented. Here x_t is the input data that is being sent into the algorithm and further on an output h_t is received. Such operation is made for each time step of the data and the information achieved in each step is passed further on in time. That leads to one of the main advantages of a RNN, unlike other similar algorithms the output is dependent on the previous sequential elements.

Standard RNNs usually have quite simple structures with only one hidden layer that is built on the *tanh* operation, such model is presented in Fig. 1b.



(a) An example of an unrolled RNN



(b) An example of a standard single layer RNN

Figure 1: Single hidden layer RNN model [4]

However, even if RNNs are designed for working with time series not all of the models work perfectly. The most common problem of an RNN is the vanishing gradient. That problem appears in the models due to their main advantage, the capability of handling long-term dependencies. At the start of a neural network, each loop is assigned a weight, a random number close to zero. During the training of a RNN, the cost function and the error are calculated at each time step, further on the cost function is back-propagated through the network to update the weights and minimize the error. The problem of the model relates to updating weight recurring (wrec) - the weight that connects the hidden layers to themselves. While back-propagating each weight at each time step is multiplied by wrec, that results in multiplying a small number by another small number which in turn leads to a very quick value decrease. A similar problem arises if the wrec is too large. Those exact phenomenons are called vanishing and exploding gradients. The consequences of it are an improperly trained network and bad final results [5].

This problem however has been solved in a RNN algorithm called Long Short-Term Memory (LSTM). Which was the reason why it was chosen for the purposes of this thesis.

2.3.1 LSTM

LSTM network was firstly introduced by Hochreiter & Schmidhuber (1997)[6] and further developed through time. Just as a standard RNN, LSTM has a chain-like structure, but unlike a standard RNN, it has four hidden layers that interact in a specific way. A breakdown of the LSTM hidden layers is shown in Fig. 2.

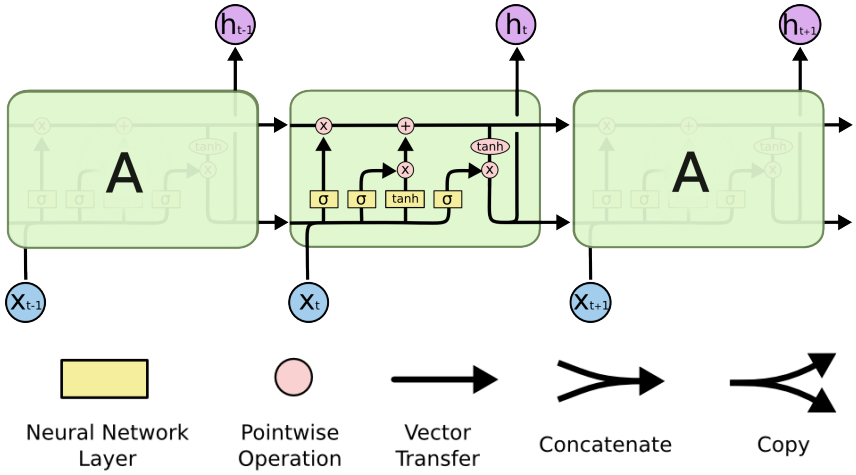


Figure 2: LSTM model with four hidden layers [4]

The core idea of the LSTM network is that it runs through the top layer of the diagram and by using gates decides which information to let through and which information is to be left out. The gates are built from a sigmoid neural net layer and a pointwise multiplication operation.

The first step of LSTM is called the "forget gate layer", it is the first sorting part of the model that decides what information to keep. It compares the previous output h_{t-1} and the current input x_t and outputs a number between **0** and **1** for each number in the cell state C_{t-1}. Here **1** represents "let all information through" and **0** means "let none of the information through". That value to be returned from the "forget gate layer" is calculated by applying the difference to the weight of the layer W, adding the base of the layer b, and multiplying those with the sigmoid function. using:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right). \tag{9}$$

After the first sorting phase the network decides which information to store in the cell state. That is done in two steps, first step is another sigmoid layer that is called the "input gate layer", which decides which values are to be updated. The second is a *tanh* layer that creates a vector of new values, $\tilde{C}_t$, that are added to the state. Those two steps are once again calculated with help of the state weights W and state base b, using:

$$\begin{aligned} i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\ \tilde{C}_t &= \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right). \end{aligned} \tag{10}$$

Further on the networks needs to update the cell state from C_{t-1} to C_t which is done by summarizing the previous steps as:

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t. \tag{11}$$

The final step of an iteration is to decide which output should be sent further on in the chain. The output becomes a filtered version of the cell state. To achieve that another sigmoid layer is

run once again, it decides which parts of the cell state are to be sent to the output. The output of that layer is multiplied by another *tanh* transformation of the cell state as shown below:

$$\begin{aligned} o_t &= \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \\ h_t &= o_t \tanh\left(C_t\right). \end{aligned} \tag{12}$$

After the output h_t is achieved it is sent to the next time step in the network and the loop is made all over again. In that way, the system learns itself and further on can make predictions on future values [4].

2.3.2 Activation functions

An activation function, sometimes also called a transfer function, in a NN defines transformation of the weighted sum of the input into an output from nodes in a layer. There are many different choices for an activation function and those functions usually differ depending on which type of layer it concerns. In total there are three types of layers: input, hidden, and output layers. It is common to use the same activation function for all hidden layers.

The most common functions for the hidden layers are Sigmoid, Tanh and ReLU. For the purpose of this thesis, the ReLU activation function was chosen as the activation function for both input and hidden layers. ReLU stands for "rectified linear activation function" and its biggest advantage over the other functions is that it is less susceptible to vanishing gradient problem. The function itself is calculated as:

$$\text{ReLU} = max(0.0, x), \tag{13}$$

that implies that if the input value x is negative the function will return 0, otherwise it returns the value itself. Fig. 3 shows and input versus output plot of ReLU activation function.

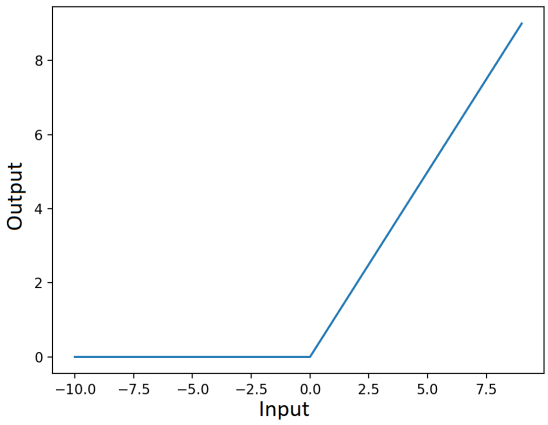


Figure 3: ReLU activation function, showing the input versus output outcome.[7]

As for the output layer there are three other most common activation functions: Linear, Sigmoid and Softmax. For the purposes of this thesis, the linear function was chosen. That function is also usually called no activation since it does not change the weighted sum of the input and it returns the value directly. Fig. 4 shows an input versus output plot of Linear activation function [7].

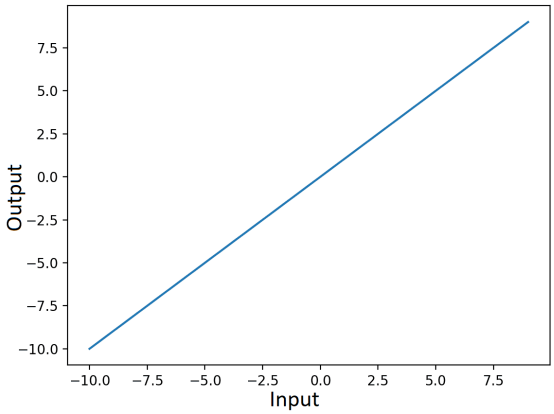


Figure 4: Linear activation function, showing the input versus output outcome[7]

2.4 Prophet

Prophet is open-source software released by Facebook for the solemn purpose of forecasting business time series. The core of the model is based on a decomposition quite similar to the one that was shown in Eq. 1. The biggest difference in the Prophet decomposition is that it also adds the holiday component $h(t)$ to the model. The holiday component corresponds to, for example, extra sales during Christmas or just usual sales during the year. So the final decomposition becomes:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t. \tag{14}$$

The main advantages of the model are such features as the data does not have to be regularly spaced (outlier removal is not needed), fitting the model does not take a long time, and the model has easily interpolated parameters. But let us take a look at how Prophet works with each component of the decomposition to be able to understand its core better [8].

2.4.1 Trend

Prophet covers two different types of trend models: a pairwise linear model and a nonlinear, saturating growth model. Prophet, by default, uses the linear model for the forecasts. For the purpose of this thesis that model was chosen. The model in the most basic form follows:

$$g(t) = kt + m. \tag{15}$$

Here k is the growth rate and m is the offset parameter. However, even if the growth is linear, it is not always constant. Therefore Prophet defines change points to capture that variation. Assuming there are S change points in time that occur at times s_j, where $j = 1, ..., S$, a vector of rate adjustment, $\boldsymbol{\delta} \in R^S$, can be defined. δ_j defines the change at each change point of time s_j. In that case, the total rate at any time t becomes the base rate k plus all the rate changes up to the time t. In vector form it becomes $k + \mathbf{a}(t)^\top \boldsymbol{\delta}$, where $\mathbf{a}(t) \in \{0, 1\}^S$ is defined by:

$$a_j(t) = \begin{cases} 1, & \text{if } t \geq s_j \\ 0, & \text{otherwise .} \end{cases} \tag{16}$$

By applying the change rate to Eq. 15, the final model for the linear trend change becomes:

$$g(t) = \left(k + \mathbf{a}(t)^\top \boldsymbol{\delta}\right) t + \left(m + \mathbf{a}(t)^\top \boldsymbol{\gamma}\right), \tag{17}$$

where γ_j is set to $-s_j \delta_j$ to make the function continuous [8].

2.4.2 Seasonality

As mentioned earlier in Sec. 2.1 time series can have many different seasonalities, everything from yearly to hourly. There can also be several different seasonalities in the same data set, which is called multi-period seasonality. In Prophet those are handled with help of the Fourier series. So an arbitrary smooth seasonal effect can be approximated by a standard Fourier series:

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right), \tag{18}$$

where P is the expected seasonality period (for example $P = 7$ for weekly seasonality).

To fit seasonality a $2N$ parameter $\boldsymbol{\beta} = [a_1, b_1, ..., a_N, b_N]^T$ need to be estimated. For that purpose a matrix of seasonality vectors is constructed for each value of t in both historical and future data. Such matrix in general will look like:

$$X(t) = \left[\cos\left(\frac{2\pi(1)t}{P}\right), \ldots, \sin\left(\frac{2\pi(N)t}{P}\right) \right]. \tag{19}$$

So the final seasonal component becomes:

$$s(t) = X(t)\boldsymbol{\beta}, \tag{20}$$

where $\boldsymbol{\beta} \sim \mathrm{N}\left(0, \sigma^2\right)$ imposes a smooth prior on the seasonality [8].

2.4.3 Holiday Effects

Holidays and huge events can provide quite huge disturbances in the data. For example, in Sweden, the biggest spikes fall on Black Friday and Christmas sales. For other models, those disturbances would have been considered outliers and would have been handled separately, however, Prophet can take care of those. It uses a similar method here as when it handles seasonalities. First, a matrix of regressors is constructed:

$$Z(t) = \left[\mathbf{1}\left(t \in D_1\right), \ldots, \mathbf{1}\left(t \in D_L\right) \right], \tag{21}$$

where $i = 1, ..., L$ represents each holiday and D_i is a set of both historical and future dates for each holiday. $\mathbf{1}$ is an indicator function that tells if a time t falls onto a holiday i. Further, each holiday has a parameter κ_i that represents the change in the forecast for that specific holiday. Similarly as with seasonality $\boldsymbol{\kappa} \sim \mathrm{N}\left(0, \nu^2\right)$ is the vector representing all holidays. So the final holiday component becomes:

$$h(t) = Z(t)\boldsymbol{\kappa} \text{ [8].} \tag{22}$$

2.5 ASP

The final prediction model that is going to be presented here is a self-built model that is inspired by the Prophet structure. It considers the data set as a cyclic model that has repetitive weekly and monthly trends (seasonality) and further on catches the general trend of the data with a 3rd degree polynomial. That model can be represented as a decomposition of the time series, similar to the one in Eq. 1 with a slight change:

$$y(t) = g(t) + s_m(t) + s_w(t) + h(t) + \epsilon_t, \tag{23}$$

here $g(t)$ is once again the trend component, $s_m(t)$ and $s_w(t)$ are monthly and weekly seasonal trends respectively and $h(t)$ is the holiday component. Fig. 5 shows an example of each trend separately under a one-year period.

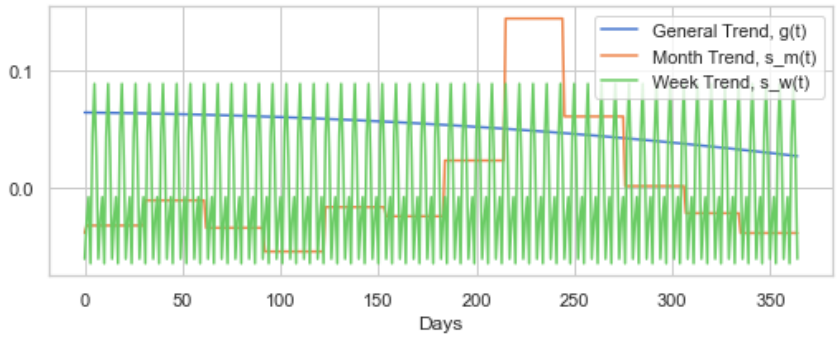


Figure 5: An example of a yearly decomposition from Eq. 23.

The biggest advantage of this model is that it is quite easy to adjust. It is possible to easily change any parameter of the model depending on the data set that is uses.

2.5.1 Trend

The trend component of this model is based on a third-degree polynomial regression. The goal of that method is to find a line that fits the long-term trend of the data in the best way. That implies that a line is to be found of the form:

$$y = a_0 + a_1 x_1 + a_2 x_2^2 + a_3 x_3^3, \tag{24}$$

where $a_{0,1,2,3}$ are the regression coefficients that are to be fitted. That regression line is fitted one year back into the data.

2.5.2 Seasonality

The model takes account for two different seasonal components: weekly and monthly. Both of them are based on the average of previous seasonal trends in the data which is calculated as an arithmetic mean:

$$A = \frac{1}{N} \sum_{i=1}^{N} x_i, \tag{25}$$

here N is the total number of observations and x_i is an observation value.
That procedure helps the model to get rid of the noise in the system and capture well the average trend of the sales.

2.5.3 Holiday Effects

The holiday effects of this model are based on exactly the same principal as the seasonality, however it uses the data achieved from the outliers to identify the increase in sales. Further on, if a holiday is detected the increase is added onto the seasonal component for that specific month.

3 Method

3.1 Programming Tools

To analyze and forecast the data for this thesis the programming language Python was used. All implementations were made in a free and open-source scientific environment Spyder. Spyder environment is written in Python, for Python, and designed by and for scientists [9].

The libraries used were Matplotlib for visualization, Pandas for structuring the data, DateTime and Calendar for performing time operations on the data, NumPy for mathematical computations. Further on for predictions Prophet library, described in Sec. 2.4, was used in one of the implementations. And for the LSTM algorithm Keras, TensorFlow and ScLearn were used.

3.2 Data

All the data used for this thesis was acquired from online sales and was provided to Infobaleen by the client. It represents all sales transactions for the company from January 2019. The data contained a lot of information such as total order amount, items category, article number, and many others. The relevant categories that were picked out for this thesis were: timestamp of the purchase, the total amount paid in that transaction in SEK, availability of the purchased item in storage, and category of the purchased item. All data were stored as pandas data frames for easier management.

3.2.1 Analysing and Pre-processing

The data received was representing each transaction separately. To begin the analysis of the data and find the right models, I chose to work with a total daily revenue. So the first thing to do was to summarize the total revenue and total amount of orders for each day and convert it to time series. In Fig. 6 normalized daily revenue from received data is presented.

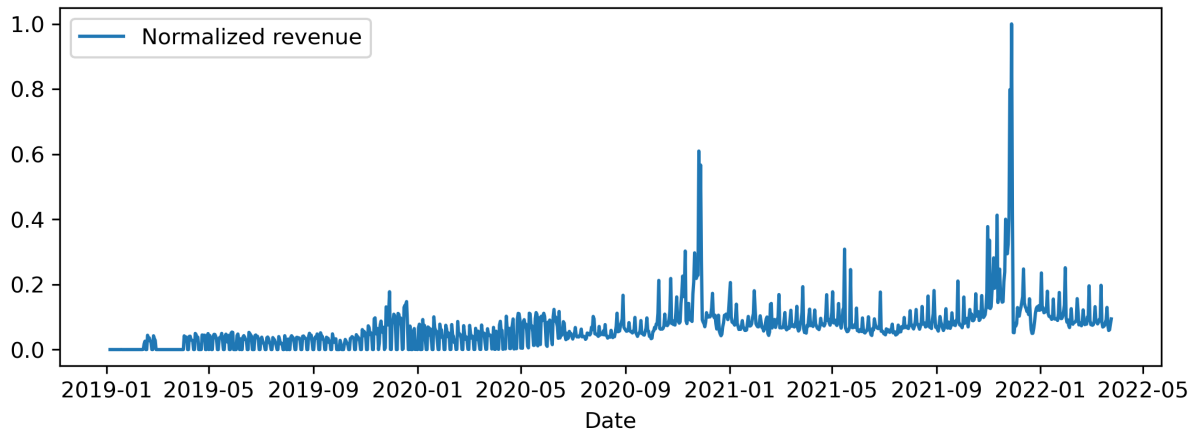


Figure 6: Total daily revenue of sales over past three years. The data is normalised by Eq. 8.

From the figure above one can clearly identify some outliers in the data as well as an upwards going trend and seasonality. Those observations are the first step towards finding a suitable model for the predictions.

The next step in pre-processing the data was to remove outliers. At this step, I also decided to remove the first year of the data, since, as Fig. 6 shows us, there is a lot of incomplete data in that time period. As it can also be seen in Fig. 6 the biggest outliers in the data were centered around

November and December, which corresponds to Black-Friday that usually accrues around the end of November and Christmas sales that accrue around the end of December. Those values needed to be removed from the data set to be able to predict an average day more accurately. The procedure for that removal is described in Sec. 2.2.1.

3.2.2 Separate Data Sets

I started my analysis on the complete data set, however, the goal of this project was to make predictions on separate data sets as well. So I separated the data set into 24 smaller ones. That separation was based on 6 different product categories and 4 different customer groups. The customer groups were based on the number of purchases made from the store by that customer earlier. Those were as follows: 1st purchase, 2nd purchase, 3rd or 4th purchase and 5th and higher. Further on those data sets were handled just as the total set with revenue summarizing and outlier removal.

3.3 Time Series Forecasting

To pick, adjust and test the forecasting models I chose to use the total revenue of the sales first. To try the different algorithms it was important to separate the data set into two parts: training and test. Initially, it was done in a proportion of 80% to 20%. Later on those 20% were used to evaluate the models with help of MSE presented in Eq. 5, MAE presented in Eq. 6, and MAPE presented in Eq. 7. After the models were evaluated the test data set was changed to consist only of the time period of the requested prediction.

The choice of the first two models, LSTM and Prophet, was based on my pilot study. My decision was mostly based on "Forecasting: Principles and Practice" [10] and "Food Industry Sales Prediction" [2]. I chose not to try models such as AR, MA, ARMA, ARIMA, or SARIMA since those models require clear patterns and strong trends in the data which was not the case for the data I was working with. Further, the ASP model was developed continuously over the time of this thesis project.

3.3.1 LSTM

The first model I chose to implement was LSTM described in Sec. 2.3.1. Here several parameters were tested out. The first one was a look-back span that defines how many time steps backward the algorithm will use in order to predict the current time step. Since the clearest seasonality was weekly I chose a look-back of 7 days. Other parameters that needed to be adjusted were: the number of layers, number of neurons in each layer, number of epochs for the system to learn on, and batch size. The final parameter that needed to be set was the loss calculation function which I chose to be MSE described in Eq. 5. The activation functions that I used were ReLU for all the layers except the output layer where I used linear activation, both of which are explained in Sec. 2.3.2.

3.3.2 Prophet

The second model to be tested was Prophet described in Sec. 2.4. For that model, no outlier removal was needed. That depends on the Prophet's ability to handle holiday effects.
The model itself was quite straightforward in implementation. Here a linear trend was used for the model, the chosen seasonality was weekly. And finally, the holiday component was consisting of Black Week and Christmas Holidays.

3.3.3 ASP

The third model that was implemented is a model I constructed myself. For that purpose, I created two different time series, one with the revenue data without outliers and one that included outliers only. Based on the first data set I calculated the trend and seasonality of the model. The second data set was used to implement holiday effects. That was done by finding the mean value of each month for the outliers data set. Further, if the mean value of that month was higher than the mean value of all months, that month would be counted as a month that has holiday trade in it. In that case, the total value of that month will be increased proportionally.

3.4 Error evaluation

To evaluate the results I used MSE, MAE and MAPE described in Sec. 2.2.2. However due to secrecy I will only present the results of the MAPE further on in this thesis.

In the beginning of this thesis a benchmark model was chosen. That model was used as an indicator of performance for the implemented algorithms. That model considers the past month of sales backward and by taking an average of those sales predicts each day.

3.5 Stock Capacity

For predictions of stock capacity, I used the number of orders as the forecasted variable. I picked out 20 of the highest sold items by the company during the previous two years. I wanted to use LSTM for all of the predictions of stock capacity, however, some of the data sets were lacking continuity in the data which caused LSTM to result in worse predictions. So I decided to divide those items into two categories. The first category was containing items that have a continuous record of sales and sold pretty much every day. The second category was items that were sold not as often, but still in large amounts.

For predictions of the items in the first group, I chose the LSTM algorithm. And for the second group the ASP model algorithm was more suitable. The function makes a prediction of the upcoming sales in terms of "times the item was ordered" for the upcoming month. After achieving the total amount of orders that are expected it compares it with the current stock balance. If the stock balance is larger the function says that the current stock will last longer than a month, otherwise, it calculates exactly how many days will the current stock last based on the daily predictions.

4 Result

The goal of this thesis was to find the best suitable model for predictions of the provided data. In this section, I will present the results that I achieved from the studied algorithms. I will compare the performance of all the models to each other and to the benchmark model that had a MAPE of **47.0%**.

The results from the separate data sets are going to be presented as tables. The separation was made depending on item group(I.G.) and customer group(C). Customer groups were separated depending on the number of purchases that a customer has done at that store before. For example, C.3-4 means that it is the customer's 3rd or 4th purchase from the store.
The results of predictions for weeks in stock are going to be presented as a table with a prediction for each item compared to its actual stock value.

4.1 LSTM

As hyperparameters that were used in the LSTM model, I chose to use 3 layers (1 hidden layer, 1 input layer and 1 output layer), 100 learning epochs, look-back of 7 days, and batch size of 6. That provided me with the final model that could be trained and tested. The resulting MAPE for LSTM ended up being **17.5%** for the test values. In Fig. 7 the training and testing results of the whole data set are presented. Here the blue line represents the original data, the orange line represents training results on the data set and the green line represents the prediction results of the model.

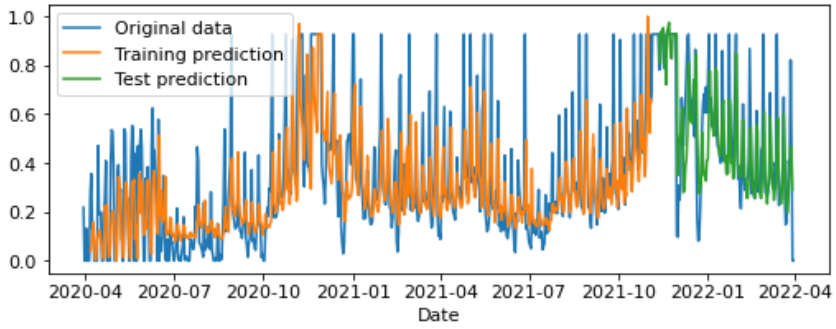


Figure 7: Training of LSTM model.

In Fig. 8 the forecast for one month of daily revenue is shown. Here the blue line represents the actual values of the revenue, the orange line represents LSTM predictions and the green line represents the forecast of the benchmark model. The gray area represents the MAPE of the model. All data is once again normalized.

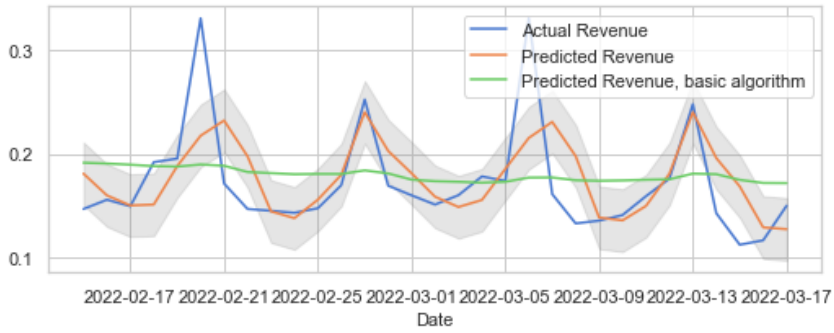


Figure 8: Normalized prediction of total revenue by the LSTM model for a time span of one month. The grey area represents the MAPE of the model.

The next prediction was made on the separated data sets. In this prediction, the hyperparameters used were the same as earlier. The results of that prediction are shown in Tab. 1. Here a normalized sum of monthly revenue for each data set is presented for both the actual value of the revenue and the predicted value. The results are normalized towards the same minimum and maximum values for a better comparison.

Table 1: Normalized prediction of Revenue by LSTM model for a time span of one month for separated data sets.

	Actual				Predicted			
	C.1	C.2	C.3-4	C.5-	C.1	C.2	C.3-4	C.5-
I.G.1	0.486	0.174	0.087	0.128	0.509	0.177	0.087	0.134
I.G.2	0.938	0.331	0.147	0.176	0.994	0.333	0.126	0.117
I.G.3	0.263	0.058	0.005	0.008	0.254	0.054	0.004	0.006
I.G.4	0.188	0.050	0.011	0.010	0.201	0.049	0.008	0.006
I.G.5	0.910	0.289	0.115	0.108	0.782	0.282	0.113	0.110
I.G.6	0.220	0.073	0.026	0.043	0.287	0.074	0.024	0.044

Here I chose to only present the results of one-month predictions as an average benchmark. The results for one week and one-quarter predictions can be found in A.1.1.

4.1.1 Holiday test

The model was also tested on the data without outlier removal. That helped the model catch the holiday effects in the data, but resulted in a MAPE of **36%**.

4.2 Prophet

For the Prophet model I chose a Fourier transformation, shown in Eq. 18, of 5th order for a period of one week to set the seasonality. The resulting MAPE of the model ended up being **31.1%**. In Fig. 9 Prophet prediction of revenue for one month is presented. Here, similarly, as for the LSTM model, the blue line represents the actual revenue, the orange line represents Prophet prediction and the green line represents predictions from the benchmark model. The grey area once again shows the MAPE range calculated from the mean value of the data.

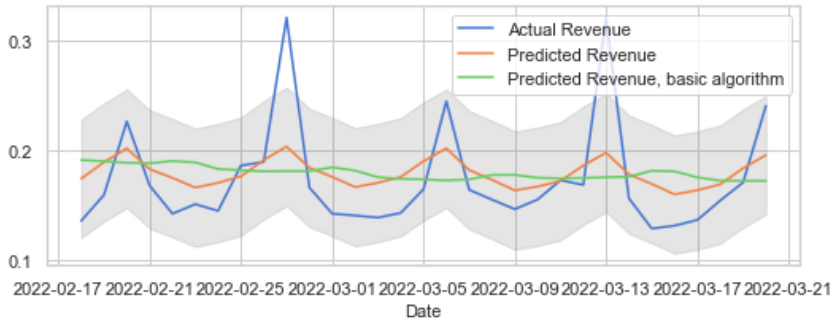


Figure 9: Normalized prediction of total revenue by the Prophet model for a time span of one month. The grey area represents the MAPE of the model.

Similarly, as for the LSTM model, a prediction for the separated data sets was done. The results of that prediction can be found in Tab. 2. Here the normalized sum of the revenues for each data set is presented.

Table 2: Normalized prediction of Revenue by Prophet model for a time span of one month for separated data sets.

	Actual				Predicted			
	C.1	C.2	C.3-4	C.5-	C.1	C.2	C.3-4	C.5-
I.G.1	0.501	0.174	0.084	0.134	0.522	0.197	0.122	0.171
I.G.2	0.955	0.341	0.150	0.186	1.000	0.369	0.187	0.236
I.G.3	0.260	0.053	0.000	0.005	0.236	0.051	0.004	0.006
I.G.4	0.193	0.046	0.005	0.007	0.223	0.058	0.012	0.008
I.G.5	0.919	0.289	0.111	0.112	0.730	0.249	0.160	0.152
I.G.6	0.215	0.067	0.022	0.043	0.293	0.096	0.043	0.052

The results of weekly and quarterly forecasts for both total revenue and separated data sets can be found in A.1.2.

4.3 ASP

For the final model the results are presented in the same way as for the earlier two models. The MAPE for this model is **25.9%**. In Fig. 10 the normalized predicted revenue for the ASP model over one month period is presented. Here the results are once again compared to the benchmark model.

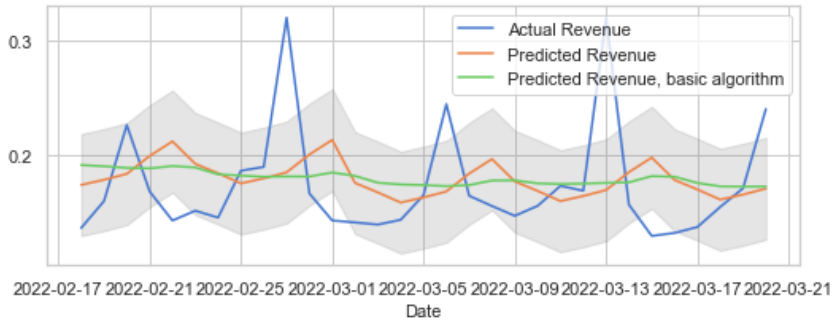


Figure 10: Normalized prediction of total revenue by the ASP model for a time span of one month. The grey area represents the MAPE of the model.

For this model, the results for a month's prediction on the separated data sets are presented in Tab. 3. The structure of this table is the same as the ones for LSTM and Prophet models.

Table 3: Normalized prediction of Revenue by ASP model for a time span of one month for separated data sets.

	Actual				Predicted			
	C.1	C.2	C.3-4	C.5-	C.1	C.2	C.3-4	C.5-
I.G.1	0.465	0.168	0.083	0.126	0.557	0.193	0.087	0.105
I.G.2	0.881	0.316	0.145	0.167	1.000	0.329	0.136	0.129
I.G.3	0.213	0.039	0.000	0.004	0.229	0.044	0.005	0.003
I.G.4	0.197	0.049	0.010	0.007	0.198	0.050	0.014	0.007
I.G.5	0.796	0.251	0.114	0.109	0.651	0.212	0.100	0.078
I.G.6	0.204	0.066	0.026	0.039	0.284	0.083	0.035	0.042

The results of weekly and quarterly forecasts for both total revenue and separated data sets can be found in A.1.3.

4.4 Weeks in stock

For predictions of stock capacity, both LSTM and ASP models were used. Out of 20 items to be forecasted the separation of items ended up being half by the LSTM model and half by the ASP model. Tab. 4 shows the results of the forecast for one month. The prediction is built on the amount of each specific item to be sold during that time period. The results are presented similarly to the revenue predictions for the separate data frames, but here the total order number is summed up instead. The left side of the table presents the LSTM forecast for the first ten items compared to the actual value. The right side of the table presents the forecast of the second ten items predicted with help of the ASP model. All values were normalized.

Table 4: Normalized values of forecast for weeks in stock for one month for 20 different items. Predicted with LSTM and ASP model.

LSTM			ASP model		
	Actual	Predicted		Actual	Predicted
Item1	0.521	0.482	Item11	0.546	0.340
Item2	0.433	0.386	Item12	0.311	0.467
Item3	0.170	0.194	Item13	0.329	0.283
Item4	0.252	0.228	Item14	0.196	0.309
Item5	0.000	0.049	Item15	0.287	0.443
Item6	0.437	0.531	Item16	0.290	0.242
Item7	0.282	0.261	Item17	0.213	0.028
Item8	0.329	0.296	Item18	0.320	0.291
Item9	0.212	0.222	Item19	0.319	0.332
Item10	0.155	0.199	Item20	0.209	0.202

5 Discussion

Forecasting is a very broad subject in which it is impossible to find perfection. The predicted results can always be improved further and further, but sadly they will never be completely perfect.

5.1 Results

In this thesis, I investigated three different models for forecasting a data set provided by the client to Infobaleen. All three of those models performed better than the model that was chosen as benchmark and had MAPE of 47.0%. The best performing model was LSTM which delivered MAPE of 17.5%, the second-best performing model was the ASP model with 25.9% and the worst of those three was Prophet with 31.1%. LSTM performed more than twice as well as the already existing model, which means that by using that model the client can drastically reduce their losses by better optimization of transportation and storage, which will lead to better overall profits.

However, the results of this thesis can be interpreted in different ways. Since the goal never was a day-by-day prediction, but a forecast for a longer period all of the models performed quite similarly. That can be explained by that the ASP model and Prophet caught the mean value of the predictions, meanwhile LSTM made better predictions on a day by day level which led to a lower MAPE.

The results of the algorithms will also improve with time by themselves. In this thesis I ended up using only approximately 2 years of data which is actually not as much as it sounds. For example the Prophet and the ASP model had only two unique Decembers to calculate the seasonality for that month on. A longer range of data will definitely improve the predictions for all three models, which will be achieved over time as more data is collected.

5.2 Weaknesses and improvement

Even though the results are quite successful there is still room for improvement in all of the models.

The LSTM model performs best on the bigger data sets, otherwise, it gets the vanishing gradient problem and the results end up at zero. It also does not take into account holiday sales which means that the values for those periods need to be adjusted manually. I made a test of running the model without outlier removal, which helped the model catch the holiday trends, but worsened the resulting MAPE to 36%. However, that could be a solution to use for the specific events that are expected to occur.

As for the Prophet model, it seems to struggle to catch the small variations in the data over time. That partly depends on the non-stationary trends in the used time series, but also on the fact that Prophet is building on the model that removes all the noise from the system. I think a big boost for prophet would be to add a depending variable for the forecast. For example, if we take sales of an outdoor market, the number of customers visiting the market will be higher if the weather outside is good. So the dependent variable for that case would be the weather.

The ASP model has some improvement points as well. From the results of my testing, I could clearly see that the model is not sensitive to noise in the data which is really good for a model like that. This model is built quite straightforwardly, which makes it easy to adjust and play with different parameters, leading to a huge potential for improvement. Further on this model, just as

Prophet, can be adjusted to work with dependent variables which could also improve the results.

Overall I did expect the LSTM model to perform the best, however, I thought that Prophet would perform better than the ASP model since the second one is a simpler version of the first. But it was a positive indicator to see the improved results in comparison to the benchmark model. It both helped me to realise that I actually managed to archive the goals of my thesis and also provide the client with an improved model.

5.3 Further work

Apart from the points for improvement that were already mentioned, there is even more possible work to be done on the final algorithm. In this thesis I only worked with one specific data set and all the models presented were adjusted after it.

Infobaleen is working with many different clients that are also interested in future predictions and the data they store may differ quite a bit from the one I worked with. That brings the need to adjust the algorithms for each specific data set. An increasing amount of clients using such an algorithm would also require that the adjustment can be automated since it can take quite a while to find the right parameters for each model. Other customers may also have data that is not compatible with the models that were chosen in this thesis, so it will require time to test other models as well.

The customers often want to know how the results were achieved, and since they don't always have the knowledge required to understand, for example, a RNN algorithm, that could be quite a challenge. I did my data set breakdown in this thesis for the purpose to be able to show on a deeper level where the total predictions come from, however, it could be a goal for the future work to make an even deeper breakdown. That could be done by finding other algorithms that for example perform better on subcategory item level or even on item level.

.

6 Conclusion

Even though future predictions are an interesting topic for many companies a lot of them use simplistic and inaccurate solutions for forecasting. Sometimes such a forecast can do more harm than good and the company may end up with an understocked supply. At the same time, an accurate prediction can make wonders for companies planning and organizing.

During this thesis, I researched the topic of time series predictions and picked out three models. Those models can be improved even further, but they already provide a better and clearer result in comparison to the benchmark model. The best performing model showed to be the LSTM, that is why I would recommend to the client to use that particular model. Hopefully, those results will provide the client with a clearer understanding of general predictions and help them improve their sales and reduce losses.

In conclusion, I would like to say that I am satisfied with the achieved results. I hope that the resulting algorithm will be a useful tool for Infobaleen's data analysis in the future and that it will further on be used also for other clients that may be interested in predictions and forecasts of their data.

References

[1] Adam Hayes. What Is a Time Series? https://www.investopedia.com/terms/t/timeseries.asp, 24/04/2021.

[2] Maja Lindström. Food Industry Sales Prediction, 09/06/2021.

[3] Bei Sun Chunhua Yang. Modeling, optimization, and control of zinc hydrometallurgical purification process, 2021.

[4] Christopher Olah. Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 27/08/2015.

[5] SuperDataScience Team. Recurrent neural networks (rnn) - the vanishing gradient problem. https://www.superdatascience.com/blogs/recurrent-neural-networks-rnn-the-vanishing-gradient-problem, 23/08/2018.

[6] J. Schmidhuber S. Hochreiter.

[7] Jason Brownlee. How to choose an activation function for deep learning. https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/#:~:text=for%20Output%20Layers-,Activation%20Functions,a%20layer%20of%20the%20network., 2021.

[8] Letham B. Taylor SJ. Forecasting at scale. https://doi.org/10.7287/peerj.preprints.3190v2, 2017.

[9] Spyder. https://www.spyder-ide.org/.

[10] G. Athanasopoulos RJ. Hyndman. Forecasting: Principles and practice (2nd ed), 2016.

A Appendix

A.1 Weekly and quarterly predictions

A.1.1 LSTM

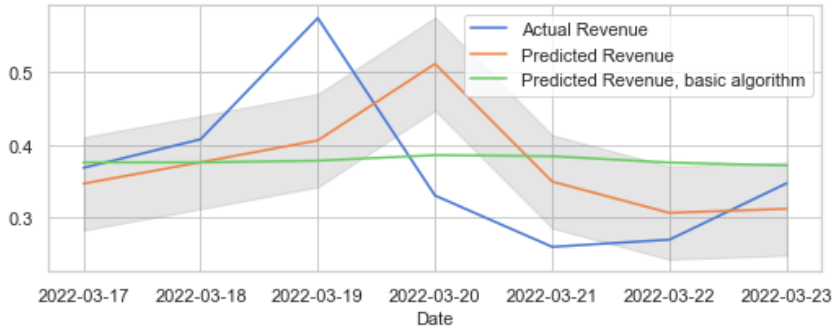


Figure 11: Prediction of total revenue by the LSTM model for a time span of one week. The grey area represents the MAPE of the model.

Table 5: Normalized prediction of Revenue by LSTM model for a time span of one week for separated data sets.

	Actual				Predicted			
	C.1	C.2	C.3-4	C.5-	C.1	C.2	C.3-4	C.5-
I.G.1	0.458	0.184	0.094	0.125	0.607	0.179	0.091	0.115
I.G.2	0.906	0.319	0.157	0.172	1.000	0.342	0.143	0.156
I.G.3	0.222	0.047	0.010	0.008	0.238	0.046	0.002	0.006
I.G.4	0.151	0.050	0.009	0.024	0.206	0.051	0.000	0.005
I.G.5	0.844	0.272	0.108	0.103	0.825	0.242	0.081	0.059
I.G.6	0.211	0.066	0.036	0.048	0.239	0.073	0.019	0.038

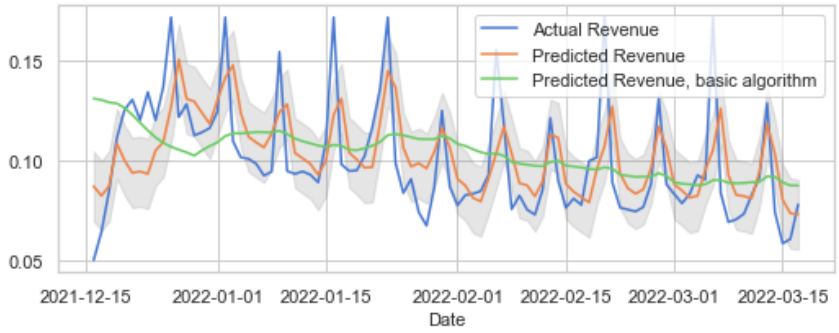


Figure 12: Prediction of total revenue by the LSTM model for a time span of one quarter of a year. The grey area represents the MAPE of the model.

Table 6: Normalized prediction of Revenue by LSTM model for a time span of one quarter for separated data sets.

	Actual				Predicted			
	C.1	C.2	C.3-4	C.5-	C.1	C.2	C.3-4	C.5-
I.G.1	0.480	0.176	0.091	0.129	0.476	0.156	0.092	0.128
I.G.2	0.982	0.356	0.160	0.179	1.000	0.319	0.161	0.165
I.G.3	0.222	0.042	0.002	0.004	0.217	0.041	0.000	0.005
I.G.4	0.209	0.055	0.011	0.008	0.205	0.048	0.006	0.004
I.G.5	0.837	0.277	0.120	0.113	0.771	0.267	0.112	0.109
I.G.6	0.221	0.074	0.030	0.042	0.218	0.070	0.025	0.038

A.1.2 Prophet

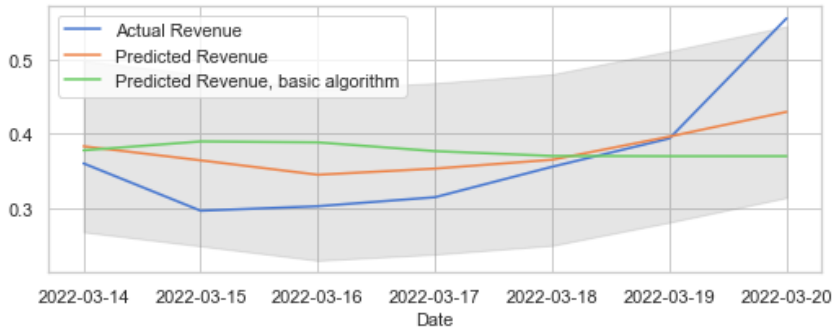


Figure 13: Prediction of total revenue by the Prophet model for a time span of one week. The grey area represents the MAPE of the model.

Table 7: Normalized prediction of Revenue by Prophet model for a time span of one week for separated data sets.

	Actual				Predicted			
	C.1	C.2	C.3-4	C.5-	C.1	C.2	C.3-4	C.5-
I.G.1	0.474	0.081	0.081	0.139	0.512	0.194	0.109	0.150
I.G.2	0.941	0.316	0.132	0.169	1.000	0.367	0.181	0.205
I.G.3	0.229	0.051	0.003	0.006	0.258	0.055	0.008	0.010
I.G.4	0.190	0.036	0.000	0.000	0.212	0.061	0.016	0.013
I.G.5	0.828	0.255	0.078	0.094	0.774	0.254	0.157	0.153
I.G.6	0.198	0.056	0.022	0.046	0.266	0.096	0.045	0.055

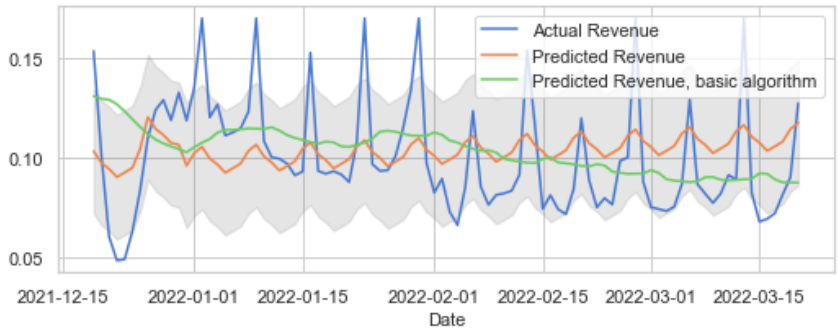


Figure 14: Prediction of total revenue by the Prophet model for a time span of one quarter of a year. The grey area represents the MAPE of the model.

Table 8: Normalized prediction of Revenue by Prophet model for a time span of one quarter for separated data sets.

	Actual				Predicted			
	C.1	C.2	C.3-4	C.5-	C.1	C.2	C.3-4	C.5-
I.G.1	0.362	0.133	0.068	0.098	0.515	0.176	0.085	0.107
I.G.2	0.758	0.278	0.125	0.140	1.000	0.349	0.145	0.153
I.G.3	0.166	0.031	0.000	0.002	0.216	0.048	0.003	0.003
I.G.4	0.157	0.039	0.006	0.004	0.169	0.047	0.006	0.004
I.G.5	0.627	0.206	0.088	0.086	0.721	0.238	0.096	0.088
I.G.6	0.163	0.055	0.021	0.031	0.246	0.078	0.026	0.028

A.1.3 ASP

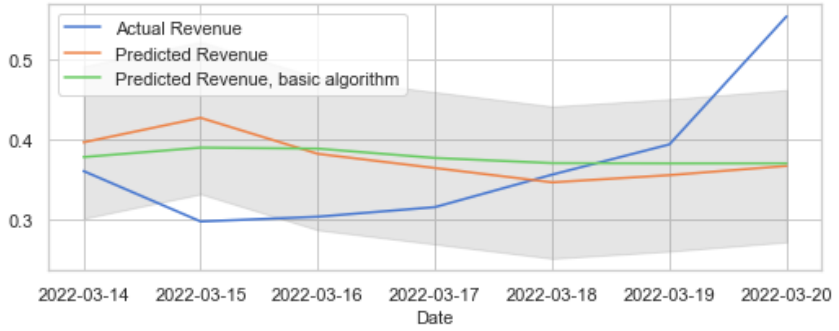


Figure 15: Prediction of total revenue by the ASP model for a time span of one week. The grey area represents the MAPE of the model.

Table 9: Normalized prediction of Revenue by ASP model for a time span of one week for separated data sets.

	Actual				Predicted			
	C.1	C.2	C.3-4	C.5-	C.1	C.2	C.3-4	C.5-
I.G.1	0.441	0.154	0.076	0.119	0.54	0.188	0.086	0.110
I.G.2	0.828	0.294	0.131	0.160	1.000	0.332	0.141	0.140
I.G.3	0.226	0.045	0.000	0.003	0.243	0.049	0.007	0.009
I.G.4	0.172	0.041	0.008	0.005	0.191	0.046	0.017	0.012
I.G.5	0.810	0.260	0.109	0.104	0.693	0.224	0.115	0.099
I.G.6	0.190	0.060	0.019	0.035	0.271	0.082	0.037	0.049

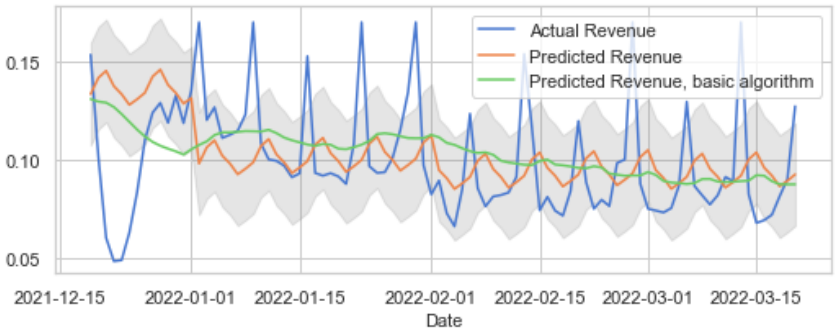


Figure 16: Prediction of total revenue by the ASP model for a time span of one quarter of a year. The grey area represents the MAPE of the model.

Table 10: Normalized prediction of Revenue by ASP model for a time span of one quarter for separated data sets.

	Actual				Predicted			
	C.1	C.2	C.3-4	C.5-	C.1	C.2	C.3-4	C.5-
I.G.1	0.462	0.168	0.085	0.116	0.537	0.189	0.084	0.095
I.G.2	1.000	0.364	0.161	0.171	0.913	0.325	0.129	0.114
I.G.3	0.216	0.045	0.005	0.007	0.219	0.048	0.004	0.000
I.G.4	0.188	0.051	0.010	0.009	0.201	0.057	0.012	0.004
I.G.5	0.706	0.236	0.103	0.098	0.554	0.199	0.080	0.062
I.G.6	0.219	0.072	0.030	0.039	0.265	0.083	0.031	0.031