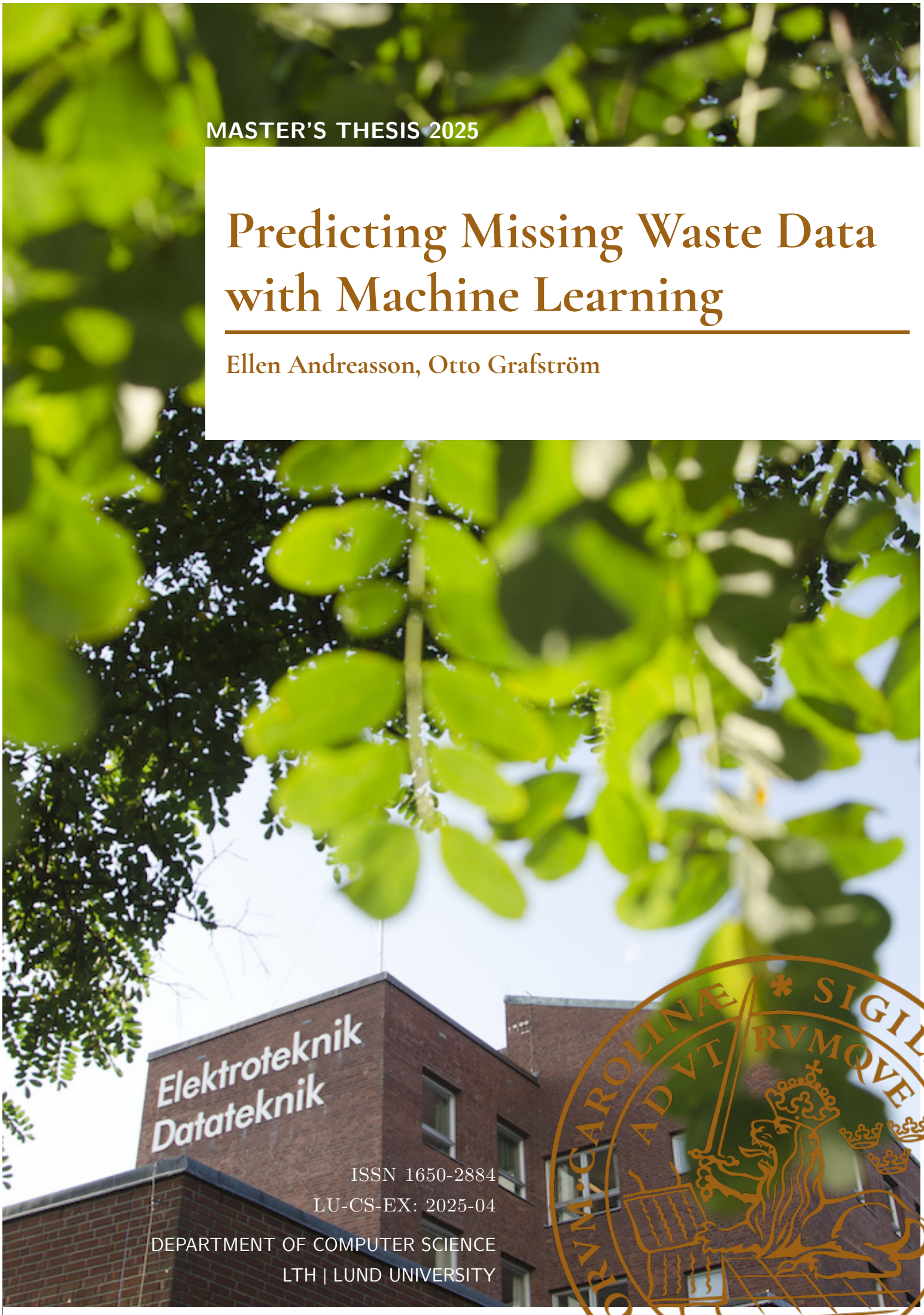


MASTER'S THESIS 2025

Predicting Missing Waste Data with Machine Learning

Ellen Andreasson, Otto Grafström

ISSN 1650-2884
LU-CS-EX: 2025-04

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2025-04

Predicting Missing Waste Data with Machine Learning

Prediktion av förlorad avfallsdata med maskininlärning

Ellen Andreasson, Otto Grafström

Predicting Missing Waste Data with Machine Learning

(A Comparative Study of Time Series Models)

Ellen Andreasson
andreasson.e@hotmail.com

Otto Grafström
ottografstrom@hotmail.com

January 23, 2025

Master's thesis work carried out at Backtick Technologies AB.

Supervisors: Pierre Nugues, pierre.nugues@cs.lth.se
Axel Rahm, axel.rahm@backtick.se
Fredrik Olsson, fredrik@backtick.se

Examiner: Jacek Malec, jacek.malec@cs.lth.se

Abstract

Efficient waste management is a critical challenge for modern societies, with significant environmental and economic implications. The use of sensors to monitor waste container fill levels has proven to be a valuable tool in addressing these challenges. This study focuses on replacing lost sensor data by comparing four predictive modeling approaches: a baseline model, the classical statistical method sARIMAX, a neural network based long short-term memory (LSTM) model, and Chronos-Bolt, a state-of-the-art pre-trained foundation model. The task is divided into two parts: predicting container emptying dates and predicting fill levels at those dates. Among the models, sARIMAX shows the best overall performance, closely followed by Chronos-Bolt. LSTM and the baseline model exhibit significantly lower performance, with LSTM slightly outperforming the baseline. This analysis highlights the continued relevance of classical methods alongside emerging machine learning techniques for reconstructing missing data and supporting efficient waste management practices.

Keywords: Waste management, time series prediction/forecasting, machine learning, sARIMAX, LSTM, Chronos-Bolt

Acknowledgements

We would like to begin by sincerely thanking our supervisors Pierre Nugues, Axel Rahm and Fredrik Olsson. We have greatly appreciated your support and guidance throughout this project. Your insights have been incredibly valuable .

We also wish to thank Olof Blomé and Simon Nilsson at Bintel for all of your help and insight, and of course for providing an interesting thesis topic.

We would also like to thank Michal Stypa for welcoming us to Backtick and for setting us up with Bintel.

Many thanks also go out to everyone at both Backtick and Bintel for giving us a fun, warm and welcoming work environment.

Lastly, we would like to thank Gustav Nauclér and Viktor Larsson for being our master's thesis partners.

Contents

Chapter 1
Introduction

This chapter provides an overview of the background, problem description, and contributions of this thesis, as well as a review of related work. Here, we highlight the importance of effective waste management, the challenges posed by data loss in sensor-monitored waste containers, and the motivation behind this study. Furthermore, we present the predictive techniques evaluated and position this research within the context of existing literature on waste prediction.

1.1 Background

A proper and effective waste collection system is fundamental to a well-functioning society. Monitoring waste levels in containers using internet of things (IoT) sensors serves as a valuable tool in achieving this, by enabling smarter and more informed waste management practices. This type of monitoring technology can provide accurate insights into the quantities and types of garbage collected at various times throughout the week, month, and year. Such detailed information facilitates, e.g., determination of the ideal number of containers for different fractions at collection sites and on-demand emptying, improving overall efficiency. Additionally, having this data opens up the possibility of predicting waste levels on the basis of historical patterns.

Bintel is a Swedish company specializing in IoT sensors and smart waste containers, with approximately 8900 active sensor devices across the country.

1.2 Problem Description

At any given time, around one to two percent of Bintel's sensors are malfunctioning. Since sensors positioned in waste containers operate in challenging environments, data loss is both possible and likely. Issues can arise from sensor-related problems like partial or complete

displacement, or physical damage caused by external forces. Other causes can be network issues, environmental impacts such as fluids or acids penetrating the units, and more.

Several of the services provided by Bintel will see a negative impact if data is corrupted or not collected at all. Examples of services relying on timely data are route and pickup planning, calculation of accumulated waste levels for different fractions, and determination of CO_2–savings. Therefore, a backup solution is required in the event of incorrect data, allowing the system to operate somewhat normally until the underlying issue is resolved. Specifically, the fallback system needs to reliably predict how full a container has been since sensor failure and when it has been emptied during the elapsed period of time.

1.3 Contributions

The objective of this thesis project was to identify a suitable approach to handle the aforementioned problem by assessing various techniques with potential for this task. Using real-world sensor data collected by Bintel, we employed four models to predict the emptying dates and fill levels of waste containers. The investigated techniques comprise a simple baseline model, the well-established seasonal autoregressive integrated moving average model (sARIMAX, Box et al. (2015)), long short-term memory (LSTM, Hochreiter (1997)) models, as well as a state-of-the-art foundation model named Chronos-Bolt (Ansari et al., 2024). Our findings reveal that, among the four models investigated, sARIMAX and Chronos-Bolt significantly outperform the others, with the former demonstrating a slight edge over the latter.

Throughout the project, we collaborated on all experimental work. Regarding the report, co-author Ellen authored Chapters 1 and 7, and Sections 3.1, 3.2, 4.2, 4.3 and 6.1, while co-author Otto authored Chapter 2 and Sections 3.3, 4.1, 4.5, 4.6, 6.2, 6.3 and 6.4. The remaining sections were written as a joint effort.

1.4 Related Work

There is a growing trend toward employing diverse statistical and machine learning (ML) models to tackle the challenges posed by waste management (Namoun et al., 2022). These methods encompass tasks such as waste material classification, vehicle route optimization, and the prediction of waste levels (Abdallah et al., 2020). While some models leverage data obtained from level or image sensors positioned in smart waste containers, others rely solely on economic and sociodemographic variables. With the expansion of IoT, sensor-based approaches have become increasingly common (Namoun et al., 2022).

For the specific task of predicting the amount of generated waste using sequential time series data, statistical modeling is a prevalent approach. For instance, in 2002, Navarro-Esbrí et al. performed a comparative analysis of sARIMA and a nonlinear dynamics-based method for forecasting municipal solid waste (MSW) generation. Similarly, Daramola et al. (2024) utilized ARIMAX, random forest, and XGBoost models, alongside hybrid techniques, to predict MSW generation and recycling based on time series data. Their findings indicated that the hybrid ARIMAX-XGBoost model delivered best performance. Additionally, Fokker et al. (2023) evaluated three statistical algorithms using sensor data from smart containers:

the seasonal naïve benchmark model, a quantile regression model with external variables, and an ensemble model incorporating error, trend, and seasonality with exogenous variables (ETSX). Among these, the ETSX model demonstrated the best predictive accuracy.

In the domain of ML, Abbasi and El Hanandeh (2016) reviewed intelligent systems for predicting monthly waste generation in Queensland, Australia. Their analysis focused on methods such as artificial neural networks (ANN), adaptive neuro-fuzzy inference systems, support vector machines, and k-nearest neighbors. Furthermore, Niu et al. (2021) applied LSTM neural networks to predict MSW generation, addressing the temporal variations and long-term dependencies that limit the performance of traditional ANNs. They obtained results where LSTM models outperformed both ARIMA and conventional ANN approaches. Similarly, Ahmed et al. (2022) assessed a range of ML models, including LSTM, one-dimensional convolutional neural networks, gated recurrent units, and bidirectional LSTM, to forecast public waste bin fullness using sensor-generated time series data. Their findings also highlighted the effectiveness of LSTM in this context.

In recent years, considerable advancements have been witnessed in foundation models (FMs) – large ML models pre-trained on vast datasets, often based on transformer architecture. This has significantly impacted the methodology employed when designing models for time series forecasting. Liang et al. (2024) presented a comprehensive review of recent progress in FMs for time series analysis, detailing their underlying mechanisms, model architectures, pre-training strategies, and adaptation techniques. However, to the best of our knowledge, no studies to date have explored the potential of FMs for predicting MSW – more specifically, the waste level in smart containers. This thesis addresses this knowledge gap and initiates the research in this emerging domain.

Chapter 2
Dataset

In this project, we use a time series dataset acquired from sensor measurements of containers of various types and waste fractions, located in different parts of Sweden. The dataset consists of fill level measurements from 8916 different containers. A fill level of 1 represents a full container and a fill level of 0 represents an empty container. The time spans of the containers' data are different due to the sensors in them being installed at different points in time. Sampling of a container is done hourly, but a data point is only recorded if a large enough difference from the last sample is measured. If the difference is not significant, a data point is instead recorded after eight hours.

The goal of the project consists of the following two kinds of predictions:

1. The emptying dates of a container.
2. The fill level of this container at the time of emptying.

Consequently, data points between emptying dates are not relevant to this project, and are instead assumed to follow a linear growth pattern. Thus, we use a version of the dataset containing only the emptying dates and the corresponding fill levels for our predictions. This emptying dataset was extracted by Bintel using additional information, such as accelerometer data. However, some emptying dates may have been misidentified. To ensure consistency, we use the same date range for validation and testing across all containers. The validation set encompasses four weeks of data, from 2024-08-23 to 2024-09-19, and the test set covers another four weeks, from 2024-09-20 to 2024-10-17. The number of data points in these sets reflects the typical time frame within which a malfunctioning sensor is replaced.

We chose to use four specific containers in our visualizations. These four containers were chosen with the intention of representing a wide range of characteristics:

- **Container A** contains residual waste and represents quickly filled containers with strict emptying schedules. It has 87 data points, sampled between 2023-04-04 and 2024-10-15.

- **Container B** also contains residual waste but represents on-demand emptying of containers that are quickly filled. It has 104 data points, sampled between 2023-05-24 and 2024-10-15.
- **Container C** contains food waste and represents on-demand emptying of food waste containers which are often filled at a slower pace than residual or paper waste containers. Food waste containers are required to be emptied at least every two weeks, as to avoid decay and growth of flies. It has 127 data points, sampled between 2021-12-08 and 2024-10-11.
- **Container D** contains clear glass and represents on-demand emptying of slowly filled containers, such as glass or metal waste containers. It has 35 data points sampled between 2021-12-07 and 2024-09-10.

To evaluate the performance of the different models, we would, ideally, test them on all available containers. However, due to time constraints, we limit the evaluation to a subset of 100 randomly selected containers.

2.1 Exploratory data analysis

The following sections provide a deeper analysis of the dataset. First, we describe and visualize both the *full* data, i.e. all recorded data points, and the emptying frequency for containers A-D. Then, we present some characteristics of the dataset of the 100 selected containers.

Figure 2.1 below illustrates the fill levels over the entire time span for containers A-D, showing the variations of their filling patterns. In Figure 2.2, we see the fill levels over the last 12 weeks of the dataset, and more clearly the containers' filling rate and emptying intervals. Emptying dates are identifiable as sharp drops down to zero.

Figure 2.2 shows that container A has a consistent and regular filling pattern. In contrast, container B exhibits a more irregular pattern with minimal filling during the summer months, as visible in Figure 2.1. This behavior is explained by its location being at a school, where activity significantly decreases during summer. As expected, container C is filled at a slower rate, and container D even more so. We also notice that the filling pattern for container D differs significantly between its first year and the subsequent two years. Additionally, there is a gap in the data for container D between September and November 2022, during which no data points were recorded.

Figure 2.3 illustrates the number of days between consecutive emptying dates for containers A-D. We see that container A is almost always emptied every seven days. Container B is usually emptied more often, with alternating intervals of three and four days during certain periods. The reduced activity during the summer months is evident here as well. Concerning container C, we can see a change in emptying patterns between the first year and the subsequent two years. During the first year, emptyings occurred regularly every seven days, but in the last two years, they became less frequent and more irregular. The filling pattern of Container D is evident here as well; it is filled irregularly its first year and then settles in to a fairly consistent emptying pattern of approximately every four weeks, with a slight trend toward more frequent emptyings during the last year.

The following plots highlight key characteristics of the 100 randomly selected containers. The histogram in Figure 2.4 illustrates the distribution of the mean time interval between

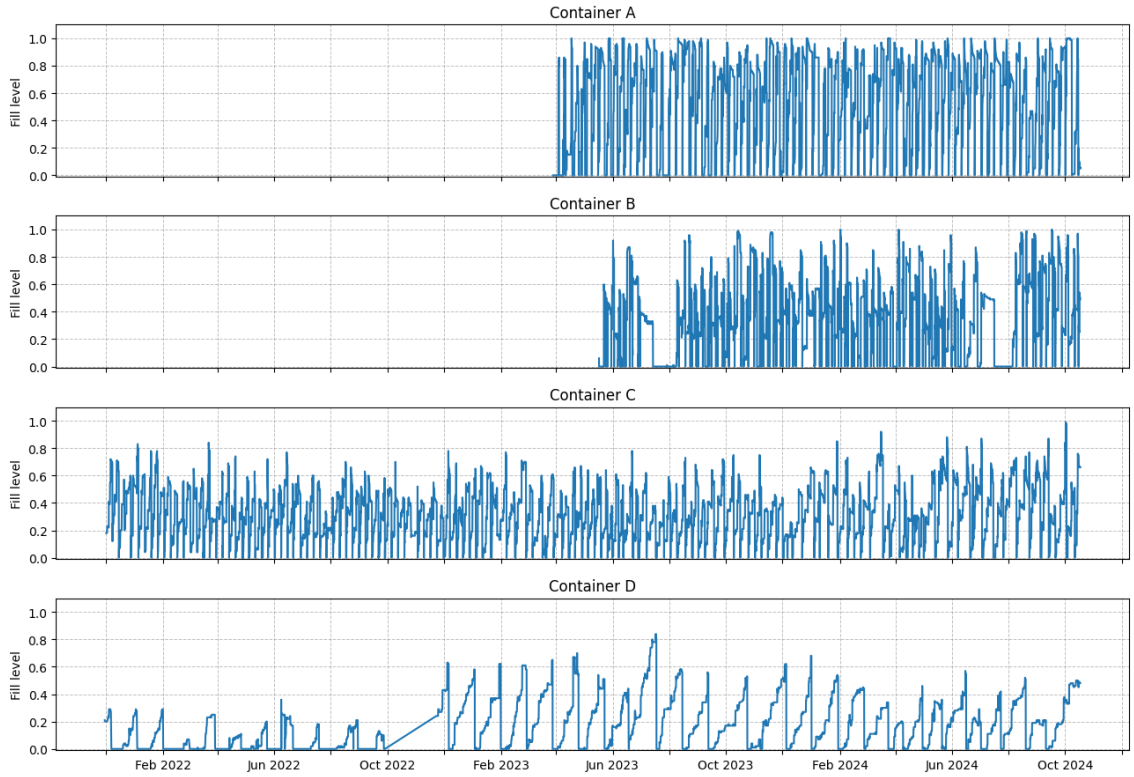


Figure 2.1: All fill level measurements for containers A-D.

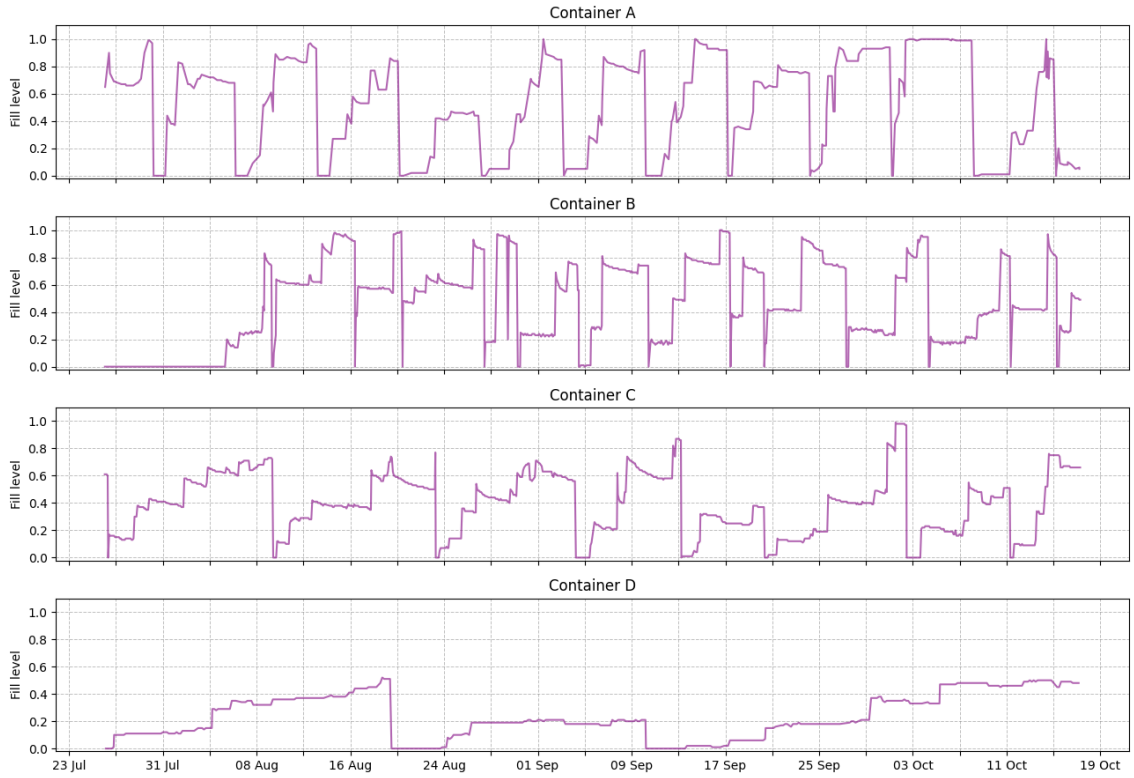


Figure 2.2: The last 12 weeks of fill data for containers A-D.

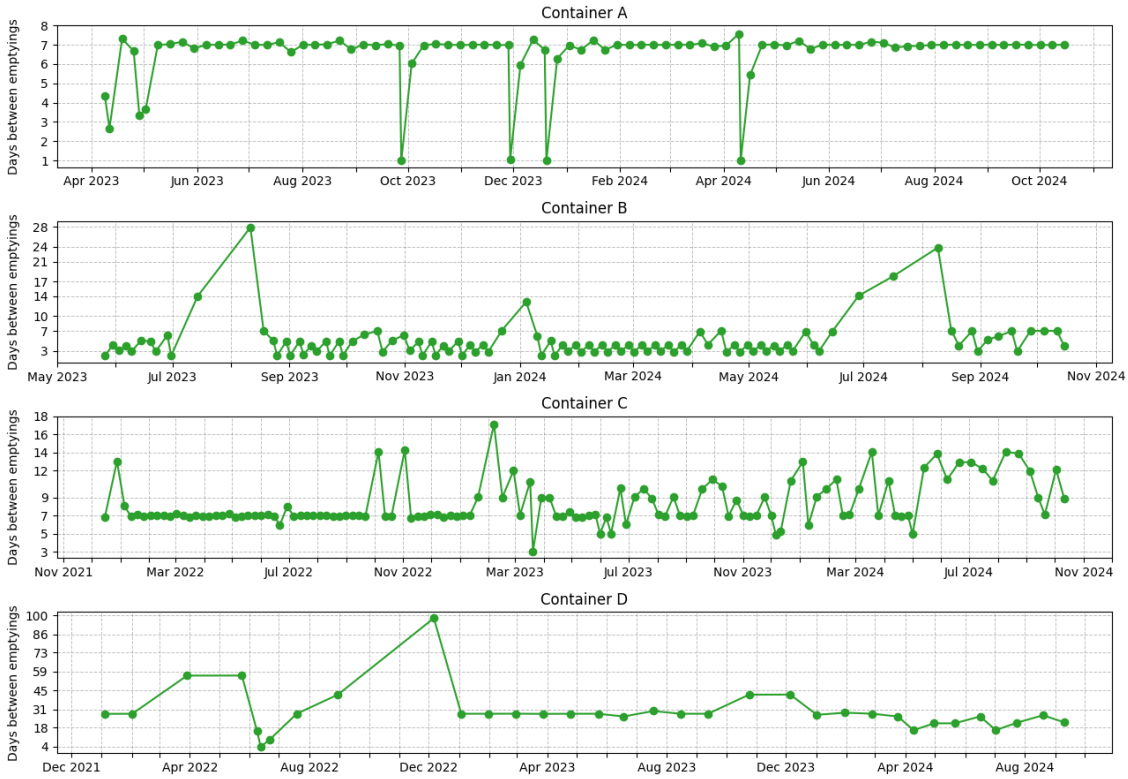


Figure 2.3: Number of days between consecutive emptyings for containers A-D, during their complete time spans.

emptying dates for these containers. This mean will be called **period** in this report and it is calculated over the 12 weeks immediately preceding the start of the test set for each data series. If a container has one or no emptyings within this 12-week window, the calculation extends further back in time since at least two emptyings are required to calculate such a time difference. Figure 2.5 illustrates the relationship between a container's mean fill level at the time of emptying and its period. The linear regression line suggests a weak negative correlation between these two metrics. Figure 2.6 depicts the distribution of the total number of data points of the containers. Figure 2.7 shows the distribution of different waste fractions.

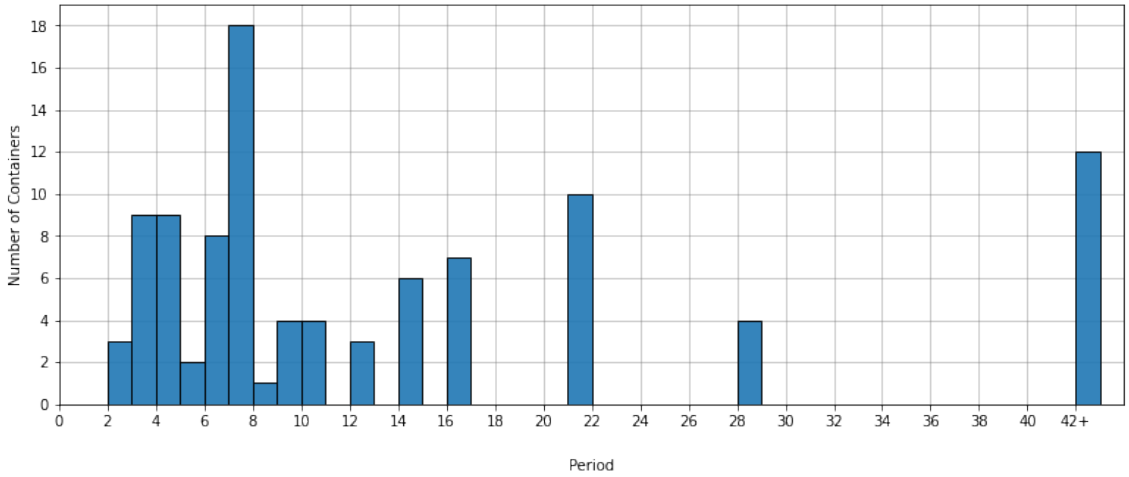


Figure 2.4: Histogram of the periods of the 100 containers.

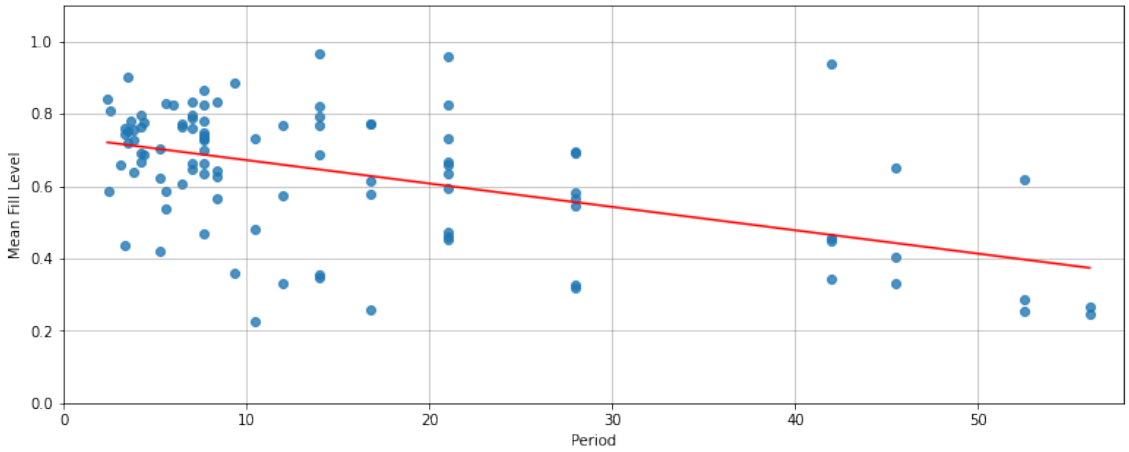


Figure 2.5: Mean fill level at emptying dates by period.

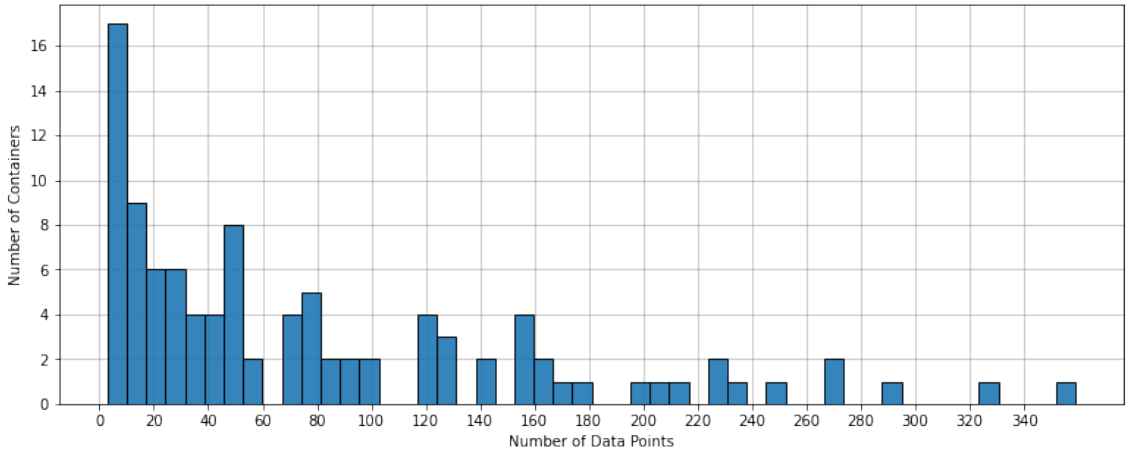


Figure 2.6: Histogram of the number of data points of the 100 containers.

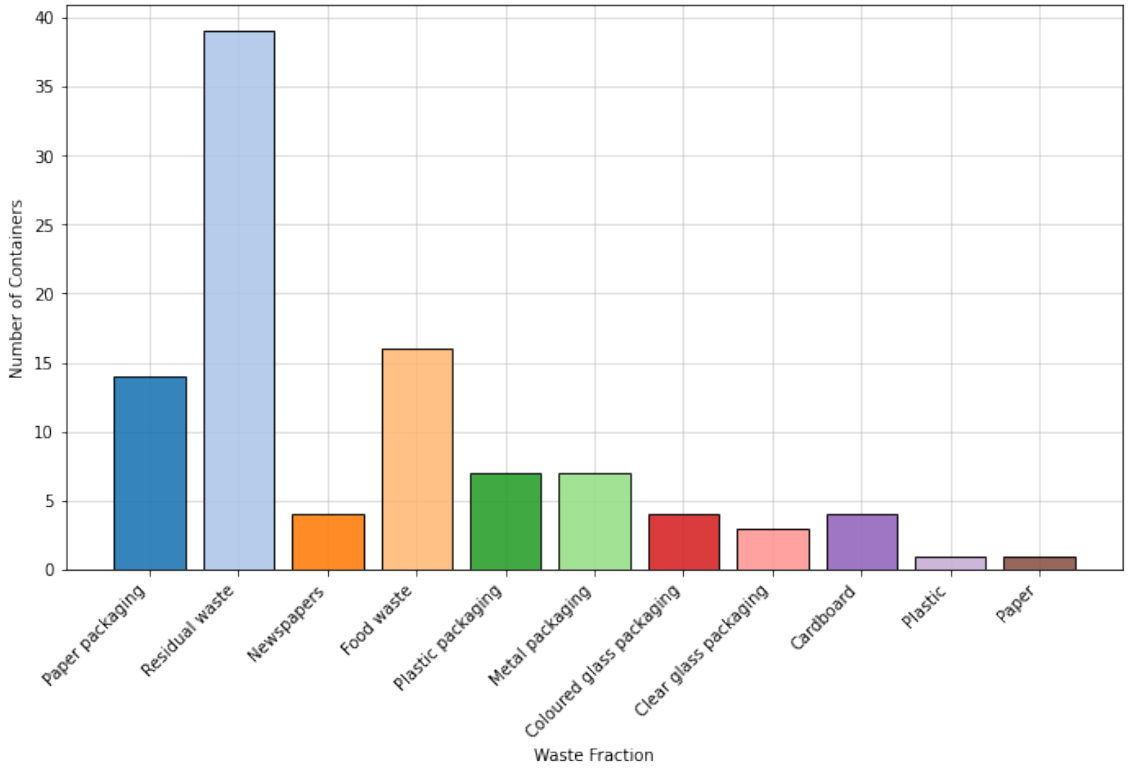


Figure 2.7: Distribution of waste fractions.

2.2 Exogenous variables

To aid the various models in their predictions we incorporated several exogenous variables. The methodology for selecting which ones of these variables to use is explained in detail in Section 4.2. The following variables were derived from a time series' own data through feature engineering:

- **Days Since Emptying**

  This variable tracks the number of days since a container was last emptied. It was designed to help the models identify the pattern that fill levels tend to increase as more days pass since the previous emptying.

- **Weekday**

  Many containers follow specific emptying schedules tied to particular weekdays. By representing weekdays as numeric values between 0 and 6, this variable enables the models to detect such patterns.

- **Day of Year**

  This variable assigns each data point a value between 1 and 366, corresponding to the day of the year. It might help the models capture yearly seasonality patterns.

In addition, exogenous variables utilized were the emptying dates and fill levels of other containers.

Chapter 3
Theoretical background

Given our findings from studying related research, we decided to focus our work on three models that appeared to hold particular promise for our objective. This chapter outlines the theoretical foundations underlying these techniques.

3.1 sARIMAX

The autoregressive integrated moving average model (ARIMA) is a prevalent statistical model within the field of short-term time series forecasting (Vagropoulos et al., 2016). It aims to capture the trend and cyclical patterns in historical data, by expressing each data point as a linear combination of previously observed values (lags) and errors. This is done by combining three components:

- **Autoregression** (AR)

  This component models the dependency between an observed data point (Y_t) and its lags (Y_{t-p}). The relationship is represented by the following equation, where c is a constant, α is the correlation coefficient, p is the number of lags, and ϵ_t is the residual error:

  $$Y_t = c + \alpha_1 Y_{t-1} + \cdots + \alpha_p Y_{t-p} + \epsilon_t$$

- **Differencing/Integration** (I)

  For the ARIMA model to perform accurate predictions, the historical data must be stationary. When the raw data does not exhibit stationarity, this component transforms the data by replacing each value with the difference between consecutive observations, as shown in the expression below:

  $$Y_t^{'} = Y_t - Y_{t-1}$$

- **Moving Average** (MA)

  The third component captures the relationship between an observation and a lagged residual from a moving average of previous observations, enabling adjustments to the current value based on past prediction errors. With β representing the dependency coefficient and q denoting the number of lags, this is expressed as:

$$Y_t = c + \beta_1 \epsilon_{t-1} + \cdots + \beta_q \epsilon_{t-q}$$

In the presence of seasonality in the time series — defined as regularly occurring patterns or variations at fixed intervals — an additional, seasonal component can be incorporated into the ARIMA model, resulting in a *seasonal* ARIMA (sARIMA). This extension allows the model to account for periodic fluctuations that repeat over a consistent time frame, enhancing its ability to capture both trend and seasonal dynamics. It accounts for correlations between observed values and their seasonal lags, such as a lag of seven for daily data with weekly patterns, for example. A seasonal model may also require AR and/or MA terms, as well as seasonal differencing.

In addition to its seasonal extension, the ARIMA model can be further enhanced by incorporating independent, exogenous variables with which the variable of interest correlates. A model of this type is denoted "ARIMAX". The exogenous regressors are integrated into the model to capture the influence of external drivers on the time series, allowing for more accurate and nuanced prediction. However, the model requires that this data is available for all past dates as well as for future ones subject to prediction (Box et al., 2015).

The integration of both seasonal components and exogenous variables within an ARIMA model is referred to as "sARIMAX".

3.1.1 Selection of model parameters

The notation of a sARIMA model is given as $(p, d, q) \times (P, D, Q)_S$, where p and q represent the order of the autoregressive and moving average terms for the non-seasonal component, respectively. Similarly, P, and Q denote the corresponding terms for the seasonal component, with S indicating the length of the seasonal cycle. The parameters d and D specify the degree of non-seasonal and seasonal differencing. These seven parameters must be determined before fitting the model to the data. However, each dataset has its own optimal order of parameters (Box et al., 2015).

The Python library *pmdarima* contains an auto-ARIMA function which automatically finds the best $(p, d, q) \times (P, D, Q)$ combination based on some constraints and an evaluation metric, typically the Akaike information criterion (AIC). To achieve this, it employs either a grid search or a step-wise algorithm [1]. The latter is presented by Hyndman and Khandakar (2008) and is recommended for its efficiency and more intelligent search strategy. The seasonality (S) must be known a priori and given to the auto-ARIMA. The method we used to estimate the seasonality of each container is detailed in Section 4.3.

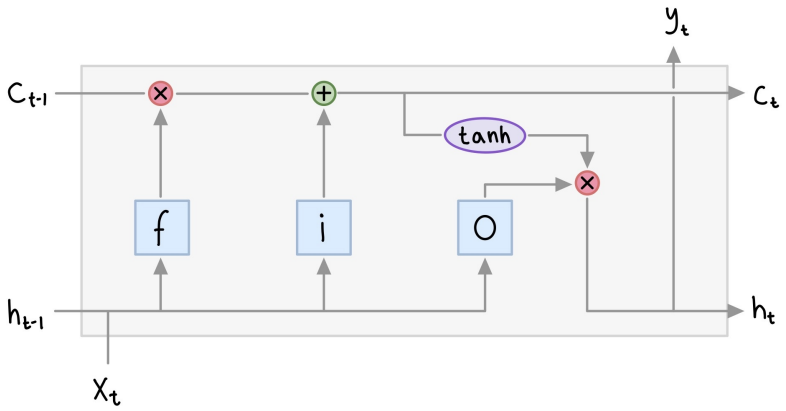
[1] `https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima`

3.2 LSTM

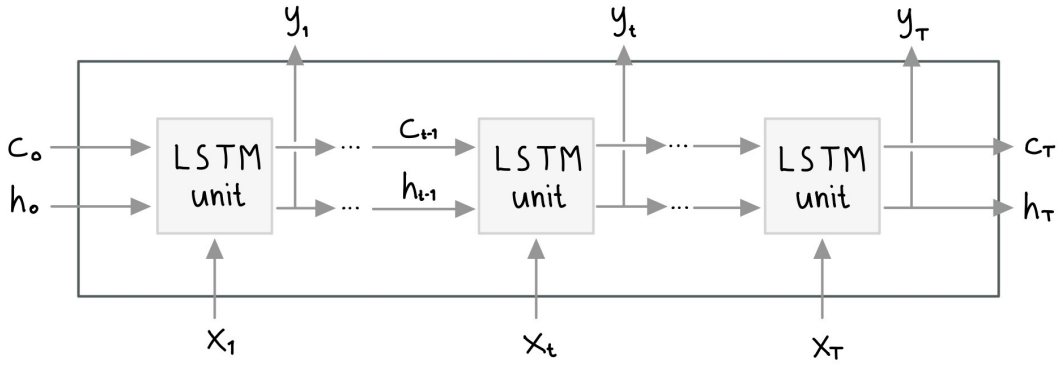
Long short-term memory (LSTM) networks (Hochreiter, 1997) are a specialized type of artificial recurrent neural networks (RNN). They are frequently employed in time series forecasting due to their ability to process sequential data while preserving the importance of temporal order. Unlike many other neural network architectures, RNNs are capable of retaining insights from previously processed steps in a sequence by maintaining a hidden state. However, when dealing with long sequences, regular RNNs suffer from a gradual loss of information from earlier inputs. This hinders their ability to learn long-term dependencies. The LSTM architecture overcomes this limitation by incorporating mechanisms that enable the retention of older information, allowing predictions at later time steps to be influenced by early insights.

These mechanisms mainly consist of memory cells and three types of gates: input gates, forget gates, and output gates, in addition to the hidden state present in traditional RNNs. The gates regulate the flow of information by determining what is retained in the memory cell, what is outputted at the current time step, and what is passed on to the next hidden state at the subsequent time step. Figures 3.1a and 3.1b display the architecture and data flow of a single LSTM unit and an entire LSTM layer, respectively.

In Figure 3.1a, x_t denotes the input vector provided to the unit at time step t. h_{t-1} repre-



(a) The architecture and data flow of an LSTM unit at time step t.



(b) Data flow in an LSTM layer with multiple time steps.

Figure 3.1: The LSTM architecture.

sents the hidden state vector outputted from the previous time step, while h_t is the updated vector with additional information from the current time step. c_{t-1} and c_t denote the previous and present cell states, respectively, and keep track of long-term information. x_t, c_{t-1} and h_{t-1} are processed by the gates before a vector y_t is outputted from the current unit, representing the predictions given the input.

The equations for the three gates are as follows:

- **Forget gate**

$$f(t) = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f),$$

  where σ is a sigmoid function, W_f is the weight matrix, b_f is the bias vector and $[h_{t-1}, x_t]$ is the concatenation between the two vectors.

- **Input gate**

$$i(t) = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \times tanh(W_c \cdot [h_{t-1}, x_t] + b_c),$$

  where W_i and W_c are the weight matrices and b_i and b_c are the bias vectors.

- **Output gate**

$$o(t) = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o),$$

  where W_o is the weight matrix and b_o is the bias vector.

3.3 Chronos-Bolt

Chronos-Bolt models are improved variations of the CHRONOS models (Ansari et al., 2024), with the primary difference being their underlying architecture: CHRONOS is based on the original T5 models (Raffel et al., 2020), whereas Chronos-Bolt is based on the improved T5-Efficient models (Tay et al., 2021). Since no paper on Chronos-Bolt is currently available, this section will describe the structure of CHRONOS, as it is largely analogous to that of Chronos-Bolt.

CHRONOS leverages the shared sequential nature of natural language and time series, by adapting large language models (LLMs) for the time series domain. LLMs are foundation models, pretrained on extensive datasets, which enables them to identify complex patterns and be applied to a wide variety of tasks. These models process sequences of text tokens and predict the next token in the sequence. Most LLMs utilize the transformer architecture, introduced by Vaswani et al. (2017), which is an attention-based model originally designed for language translation. The attention mechanism allows transformers to efficiently capture both long-term and short-term dependencies, making them well-suited for sequential data tasks.

As mentioned earlier, CHRONOS utilizes the architecture of the encoder-decoder transformer model T5, which is closely based on the original transformer, illustrated in Figure 3.2. The encoder transforms an input into sequences of continuous representations, which are passed to the decoder. The decoder processes these sequences and generates an output in an autoregressive manner. The number of layers in the encoder and decoder stacks, and the resulting number of model parameters, vary between different sizes of T5 models. We used a

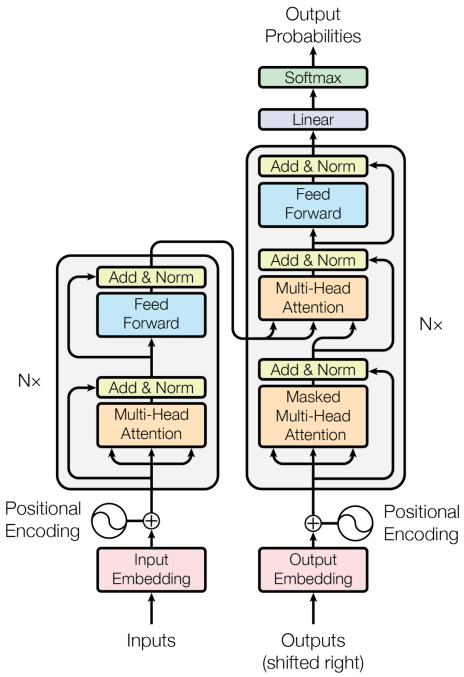


Figure 3.2: Illustration of the original Transformer architecture. The left block represents the encoder, and the right block represents the decoder.

Chronos-Bolt version based on T5-Efficent-BASE, which has 12 layers in each of the encoder and decoder stack, and a total of 220 million parameters [2, 3].

The main change from LLMs made to make CHRONOS suitable for time series predictions pertains to the tokenization of the input data. Whereas an LLM has a limited vocabulary of tokens, a time series model must be able to handle numerical real values from an unbounded, often continuous regime. Thus, CHRONOS needs to map time series values to a finite set of tokens, which it does through the use of two methods – scaling for normalizing the values, and quantization for mapping the values to discrete tokens. These methods are illustrated in the left panel of Figure 3.3.

Scaling: The purpose of scaling is to transform values into a range suitable for the next part of tokenization – the quantization. In CHRONOS, scaling is done using mean normalization, i.e., dividing each value by the mean of the training data s.

Quantization: The function of quantization is to map the scaled time series values to discrete tokens of a finite vocabulary. The tokens of this vocabulary are defined by uniformly selecting a set of bin centers on the real line in the range $[-15s, 15s]$, and then dividing the real line into bins by selecting bin edges between these centers. The first and last bins are assigned $-\infty$ and ∞ as their respective lower and upper edges. A time series is then tokenized by assigning each scaled value to its corresponding bin. CHRONOS uses 4096 bins.

The tokenized time series is then fed into the transformer architecture, which generates probabilistic forecasts by autoregressively sampling multiple future trajectories from a pre-

[2] `https://huggingface.co/amazon/chronos-bolt-base`
[3] `https://huggingface.co/google/t5-efficient-base`

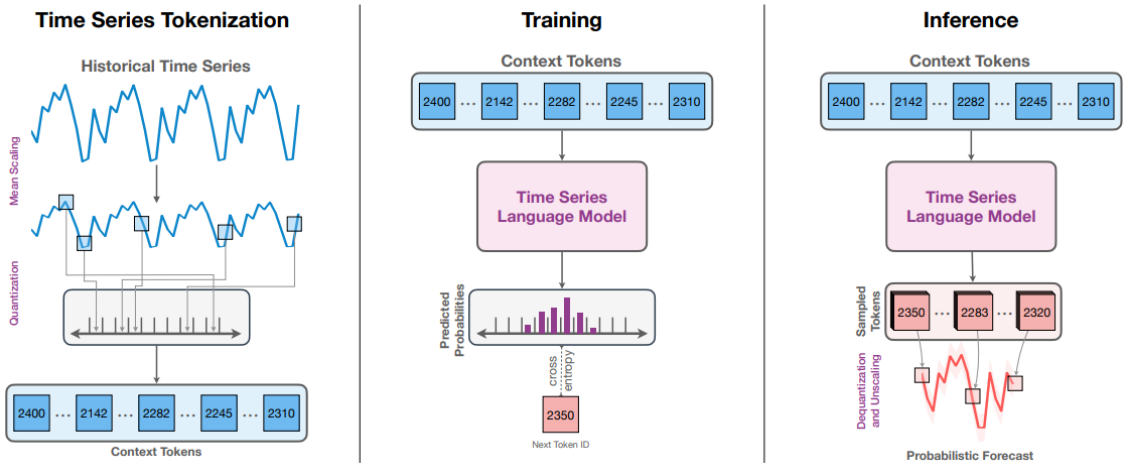


Figure 3.3: Illustration of the CHRONOS process. The left panel demonstrates tokenization through scaling and quantization. The central panel depicts the training procedure, and the right panel illustrates the inference process.

dicted token distribution. During training of the model, the predictions are compared to target values using cross-entropy loss. In inference, the predicted tokens are mapped back to real values by using the corresponding bin center values, and then unscaled by multiplying with s. Figure 3.3 provides an overview of the CHRONOS model process.

LLMs rely on being trained on large amounts of different data. CHRONOS was trained on 28 different datasets of time series of varying scale, length, and sampling frequency. The datasets were from a wide range of domains, including energy, transport, healthcare, retail, web, weather, and finance. Sampling frequencies ranged from every 5 minutes to yearly. The total number of time series was 890 thousand, totaling 84 billion values. This is a small training set compared to the vast datasets used in training LLMs, which is due to the fact that the availability of different public time series is quite limited compared to the language domain.

The CHRONOS models offer flexibility in their application: They can be fine-tuned through additional training on new time series data or used for zero-shot predictions, allowing them to quickly generate predictions on previously unseen data.

Chapter 4
Method

This chapter outlines the methodological framework of the experiments we carried out. First, the data preprocessing is detailed, including the selection of relevant exogenous variables. This is followed by an analysis of seasonal patterns within the data. Subsequently, the implementation of four prediction approaches – a benchmark model, sARIMAX, LSTM, and Chronos-Bolt models – is delineated. The chapter concludes with post-processing steps and a detailed explanation of the evaluation strategy employed to assess the model performance.

4.1 Data preprocessing

Different models have varying requirements and capabilities for handling data inputs, making data preprocessing a necessary step. Additionally, preprocessing was carried out to optimize model performance. Below, we detail the preprocessing steps applied for our experiments.

For all models, the data for each container was divided into two separate time series:

1. **A binary time series** where emptying dates are assigned a value of 1 and dates without an emptying are assigned a value of 0.

2. **A sawtooth time series** created by setting the value of the data point after each emptying to 0 and then linearly interpolating between these zeros and the actual fill levels of the emptying dates.

Figure 4.1 illustrates these two time series for container B, alongside the source data from which they were extracted. Both of these time series are created so as to be regular with a daily resolution, i.e. they have exactly one data point per day.

The test set for date prediction consists of the final 28 data points (i.e. days) of the binary time series. For fill level prediction, we instead used the last 28 data points of the sawtooth time series. Section 4.6 provides details of the evaluation process. For the sARIMAX, LSTM

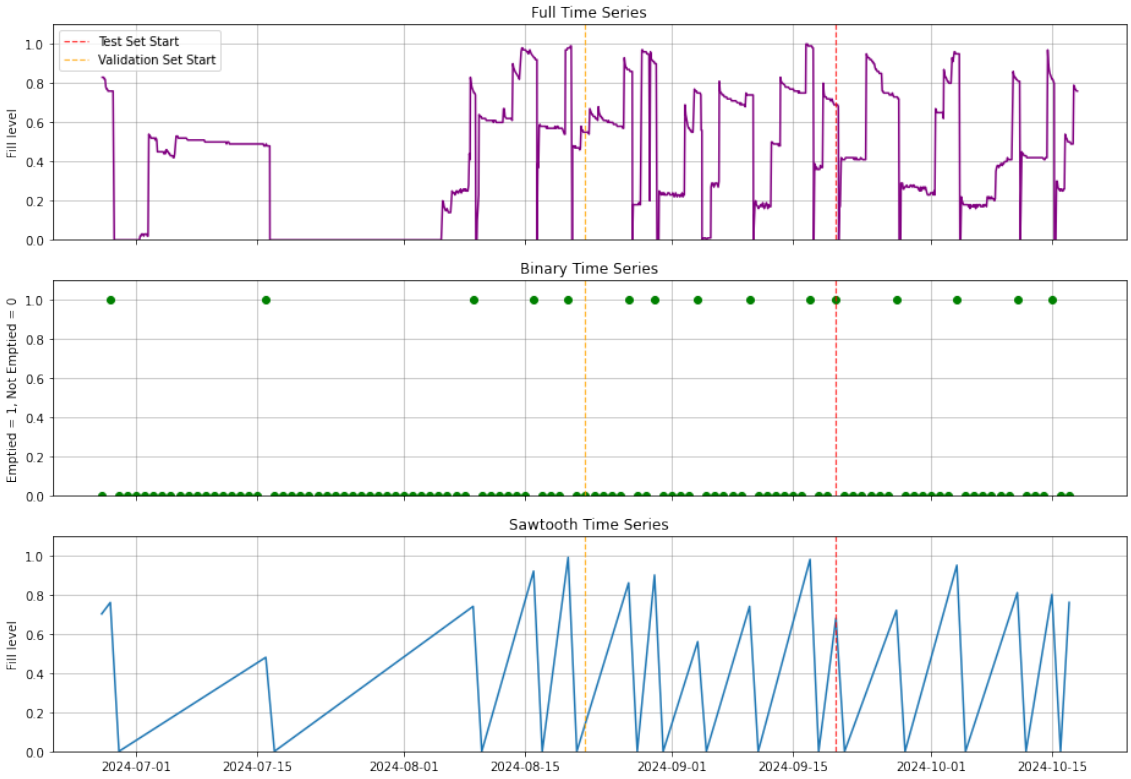


Figure 4.1: Full time series, binary time series and sawtooth time series from the last 16 weeks of the data of container B.

and Chronos-Bolt models, we also used validation sets consisting of the 28 data points immediately preceding each test set. The training set for each container consists of all data remaining after excluding the test and, where applicable, validation data. The vertical lines in Figure 4.1 show these data splits.

Baseline: We did not preprocess the data for the baseline model beyond the steps outlined above.

sARIMAX: In addition to the preprocessing described above, we selected exogenous variables (see the details in Section 4.2) and determined the seasonality (see Section 4.3) for the sARIMAX model.

LSTM: Since LSTM models process sequential data, we divided the binary and sawtooth time series into sequences with a length equal to twice the seasonality of the time series. Each such sequence was assigned a target value corresponding to the data point one step after the sequence's final value. The sequences were generated using a rolling window approach. The same sequencing approach was applied to the exogenous variables.

Chronos-Bolt: Beyond the preprocessing described above, only the selection of exogenous variables was done for the Chronos-Bolt models.

4.2 Selection of exogenous variables

To improve the performance for the sARIMAX, LSTM and Chronos-Bolt models, we incorporated additional data beyond the data specific to individual containers. This supplementary data includes variables such as the weekday, the number of days since the previous emptying, and data from other containers, i.e. exogenous containers (see Section 2.2). When the date range of an exogenous container did not match the date range of the target container, the exogenous time series was either trimmed or padded to match the target container's range. Padding was done by adding zeros for any missing dates.

In order to select suitable exogenous variables for each container, we computed Pearson correlation coefficients (Pearson and Galton, 1895) both between each external feature and the target variable (emptying date or fill level) and between the features themselves. We then selected variables iteratively, prioritizing those with the highest correlation to the target while excluding features that correlated strongly with already selected variables.

In the training of models for predicting fill levels, we complemented the exogenous variables selected through the correlation analysis with each container's own binary time series. For the binary time series, the data points in the validation and test ranges were replaced with values predicted by the binary time series prediction model. For the LSTM model, we also supplied the "*days since emptying*" variable.

4.3 Estimation of seasonality

When constructing a sARIMAX model, the seasonality must be explicitly specified. Thus, it had to be estimated for each individual container. A majority of the containers monitored by Bintel exhibit some form of weekly emptying pattern. Hence, we concluded that most containers would benefit from a seasonality set to a multiple of seven. Therefore, we defined the seasonality of a container as its period (explained in Section 2.1) rounded to the nearest multiple of seven, based on the proportionality of its deviation from the nearest multiples of seven. However, due to memory management limitations, we had to cap the maximum seasonality at 42 days (six weeks).

As mentioned earlier, we also used the seasonality to determine an appropriate sequence length for the LSTM model.

4.4 Implementation

In this section, we describe the implementation of the four different models we used. For all models, we employed two separate prediction processes: one for predicting the emptying dates and another for the fill levels. Below, we detail the specific implementation steps for each model.

4.4.1 Baseline

As a benchmark model, we predicted emptying dates by iteratively adding a container's period to the last observed emptying date. For each predicted emptying date, the fill level was

set to the mean of the observed fill levels at emptying dates in the training data.

4.4.2 sARIMAX

For the sARIMAX model, we first tuned the prominence of the peak identifier (Section 4.5) using the validation data. We used the stepwise auto_arima function from the pmdarima module to identify the optimal model parameters $(p, d, q) \times (P, D, Q)$ for each container. The tuned prominences were then used for the binary time series predictions, again using the auto_arima function to optimize model parameters.

After predicting the emptying dates, we forecasted the fill levels at these dates.

4.4.3 LSTM

Following the evaluation of sARIMAX, we investigated the performance of LSTM networks. Predictions were made using recursive multi-step forecasting, i.e. we predicted one time step at a time and then used that prediction as input for the subsequent time steps.

Due to the limited amount of data available for each container, we implemented the networks with a single LSTM layer in addition to the input and output layers (see Figure 3.1 for a visual representation of the network architecture). To mitigate overfitting, we applied a dropout rate of 0.1. Both models used the AdamW optimizer, but with different loss functions: *binary cross-entropy* for predicting emptying dates and *mean squared error* (MSE) for forecasting fill levels. We used a learning rate of 0.005 for both models.

In order to identify suitable thresholds for classifying the binary predictions (Section 4.5), we conducted an initial grid search using data from containers A-D. The grid search explored various combinations of thresholds, batch sizes, and the number of LSTM units. The results indicated that the optimal threshold for a container correlated with its emptying frequency. Consequently, we adopted distinct threshold values for each of the seasonality categories.

Ideally, a larger grid search could have been performed across the 100 containers to determine the optimal threshold for each container individually. However, constraints in time and computational resources necessitated this simplified approach.

With these thresholds, we evaluated all 100 containers on their validation data using different combinations of batch sizes and number of LSTM units. The optimal hyperparameter configurations were then used to generate the final predictions.

4.4.4 Chronos-Bolt

The last model we evaluated was the Chronos-Bolt foundation model. Specifically, we used the "Base" version (which is the largest Bolt-version), with a covariate regressor for incorporating exogenous variables. Our evaluation included both zero-shot and fine-tuned model variants, using AutoGluon's TimeSeriesPredictor framework for regression-based output.

Similar to the sARIMAX approach, we began by tuning the prominence of the peak identifier (Section 4.5), using the validation data. The tuned prominence values were then used to predicted the binary time series, followed by forecasting the fill levels.

4.5 Data post-processing

For the binary predictions of the sARIMAX, LSTM and Chronos-Bolt models the outputs were not inherently binary. Both sARIMAX and LSTM generated probability outputs, while Chronos-Bolt used regression to produce continuous values. As a result, post-processing was necessary to classify these outputs into binary values, where 1 represents emptying dates and 0 represents dates without an emptying. The classification approach differed between models due to variations in their prediction mechanisms: the LSTM model employed recursive multi-step forecasting, whereas sARIMAX and Chronos-Bolt generated complete prediction series.

LSTM binary classification: The LSTM model employed thresholds to classify each recursive prediction into a binary value. Predictions greater than or equal to the specified threshold were classified as 1, while predictions below the threshold were classified as 0. Each classified value was then used as input for predicting the subsequent time steps. We specified the thresholds based on an analysis of containers A-D.

sARIMAX and Chronos-Bolt binary classification: The sARIMAX and Chronos-Bolt models produced complete 28-step emptying date predictions. To classify these outputs as 1 or 0, we implemented a "peak identification" algorithm using the find_peaks function from scipy.signal. We used the "prominence" parameter to evaluate each data point. Prominence is a measure of how much a data point stands out relative to its surrounding values. As mentioned in Sections 4.4.2 and 4.4.4, we tuned the prominence parameter on the validation data before being applied to the test data. The tuning was performed individually for each container, testing prominence values in the range of 0.1 to 0.5.

After obtaining predictions for the emptying dates and corresponding fill levels, we combined the predictions to construct a sawtooth time series, as outlined in Section 4.1. The emptying date predictions and the sawtooth series were then evaluated, as described in Section 4.6.

These predictions were then combined to construct a sawtooth time series, as outlined in Section 4.1. The resulting emptying date predictions and sawtooth series were subsequently evaluated, as described in Section 4.6.

4.6 Evaluation strategy

Since we predict two different sets of values – the emptying dates and the fill levels at emptying – we need two ways of evaluation. Evaluating how well a series of predicted emptying dates fits the observed emptying dates requires an evaluation strategy that can handle too many or too few predictions. Therefore, we implemented an evaluation measure using the Hungarian algorithm (Kuhn, 1955).

The Hungarian algorithm allows us to match predictions with observations so that the total prediction error is minimized. The prediction error of a match is the absolute time difference in days. When too many or too few emptying dates are predicted and therefore cannot be matched, they are given a penalty error value equal to the mean time interval

between emptying dates, i.e., the period. The date error value is the mean absolute error (MAE) of the matches and penalties, divided by the period.

We cap the largest possible error of a match at twice the penalty error, since a prediction that bad can be seen as missing a prediction at one date and adding an extra prediction at another date. In addition, there are a few special cases that need to be handled. They are the following:

- There are no emptying dates in the test data and no emptyings are predicted. Since this is a correct prediction it gets a date error of 0.00.
- There are no emptying dates in the test data but one or more emptyings are predicted. In this case, we calculate the date error, E_D, as follows:

$$E_D = \frac{P}{P+1},$$

  where P is the number of predictions made.

- There are emptying dates in the test data but no emptyings are predicted. In this instance, the date error is calculated as:

$$E_D = \frac{R}{R+1},$$

  where R is the number of observed emptying dates.

We calculate the fill level error as the MAE between the predicted sawtooth series and the test data sawtooth series. To account for the fact that MAEs often are smaller for containers of lower fill levels, we then divide the MAE by the mean value of the training data's fill levels at emptying dates.

We also calculate a total error, as the sum of the date error and the fill level error.

Chapter 5
Results

In this chapter, we present the results of our experiments. It includes the overall evaluation scores for the four models tested across the 100 containers, along with detailed plots and evaluation scores for containers A-D. In the results tables below, E_D is the date error, E_F is the fill level error, and E_T is the total error.

Table 5.1 gives an overview of the results from evaluating our models on the 100 containers. We can see that the lowest mean E_T is obtained with sARIMAX, closely followed by both of the Chronos-Bolt versions. Notably, the LSTM model performs only slightly better than our baseline model.

Table 5.1: Summary of the results of all the models.

Model	Mean E_D	Mean E_F	Mean E_T
Baseline	0.40	0.37	0.77
sARIMAX	**0.20**	0.25	**0.45**
LSTM	0.42	0.32	0.74
Zero-shot Chronos-Bolt	0.24	**0.22**	0.47
Fine-tuned Chronos-Bolt	0.25	0.23	0.48

5.1 Baseline

Table 5.2 presents the evaluation scores for the baseline model applied to the 100 containers. Of these, 50 containers exhibited weekly seasonality, making it the largest seasonality group by a significant margin, while only four displayed monthly seasonality. No containers showed a 35-day seasonality which is why that group is excluded from the tables. The best mean scores were obtained for the latter group, with a mean E_T of 0.50. In contrast, containers emptied every three weeks proved to be the most challenging to predict, with an E_T of 0.95. Across the 100 containers, the model received a mean E_T of 0.77.

Table 5.3 and Figure 5.1 provide a detailed overview of the predictions for containers A-D. Due to the model's prediction approach (see Section 4.4.1), it is best suited for handling data with regular emptying and fill patterns the best. This is evident in the results for container A, which exhibits these characteristics.

Table 5.2: Evaluation results for the **baseline** model across the 100 containers.

Seasonality (days)	**No. containers**	Mean E_D	Mean E_F	Mean E_T
7	50	0.37	0.40	0.76
14	17	0.38	0.35	0.73
21	17	0.56	0.39	0.95
28	4	0.23	0.27	0.50
42+	12	0.36	0.32	0.68
Total	100	0.40	0.37	0.77

Table 5.3: Evaluation results for the **baseline** model on containers A-D.

Container	**Period** (days)	**Seasonality** (days)	E_D	E_F	E_T
A	7.0	7	0.00	0.06	0.06
B	8.4	7	0.50	0.48	0.97
C	12.0	14	0.58	0.66	1.24
D	21.0	21	0.50	0.63	1.13

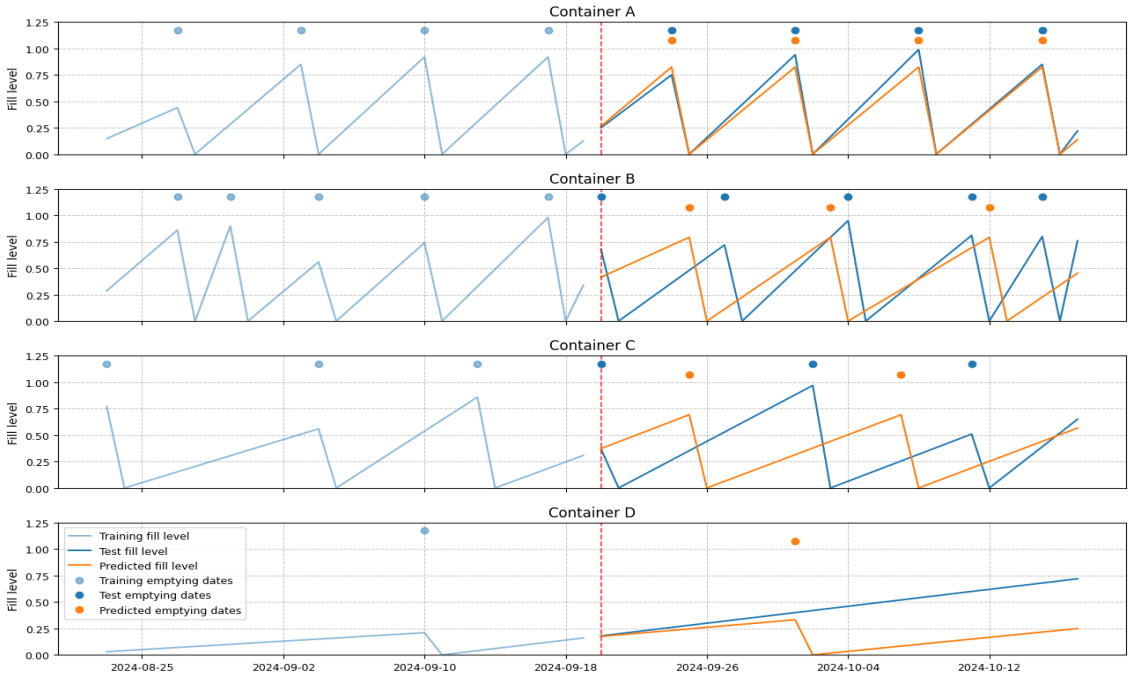


Figure 5.1: Prediction results for container A - D when using **baseline**.

5.2 sARIMAX

In Table 5.4, we can see that the sARIMAX performed best overall on waste containers with a four-week seasonality, closely followed by those with a weekly pattern.

As shown in Table 5.5 and Figure 5.2, the sARIMAX model successfully predicted the emptying dates for three of the four containers. For container B, it correctly identified three of the five actual emptyings, resulting in an E_D of **0.40**. Consequently, this led to a higher E_T than for the other three containers.

Table 5.4: Evaluation results for the **sARIMAX** model across the 100 containers.

Seasonality (days)	**No. containers**	**Mean** E_D	**Mean** E_F	**Mean** E_T
7	50	0.16	0.21	0.37
14	17	0.22	0.26	0.48
21	17	0.28	0.31	0.59
28	4	0.13	0.21	0.33
42+	12	0.28	0.34	0.62
Total	100	0.20	0.25	0.45

Table 5.5: Evaluation results for the **sARIMAX** model on containers A-D.

Container	**Period** (days)	**Seasonality** (days)	E_D	E_F	E_T
A	7.0	7	0.00	0.05	0.05
B	8.4	7	0.40	0.37	0.77
C	12.0	14	0.00	0.20	0.20
D	21.0	21.0	0.00	0.40	0.40

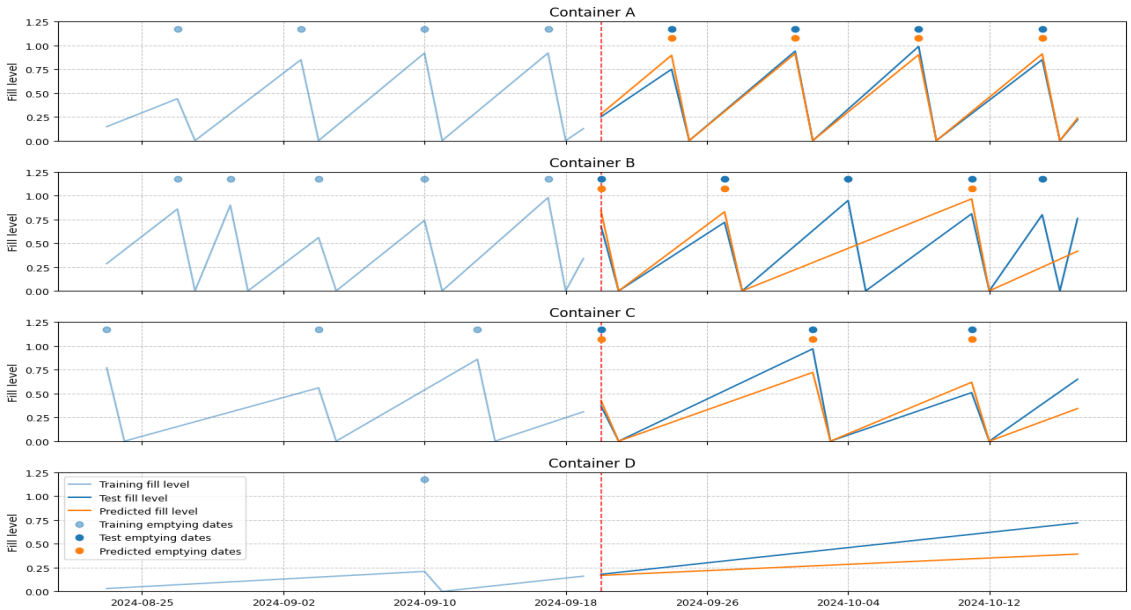


Figure 5.2: Prediction results for container A-D when using **sARIMAX**.

5.3 LSTM

With the LSTM model, the distribution of containers across seasonality groups differed slightly compared to the other approaches. This discrepancy arose because, unlike sARIMAX and Chronos-Bolt, the validation data was excluded when estimating seasonality for each waste container. This ensured that information from the data used to monitor the training process and prevent overfitting was not already incorporated.

Overall, the LSTM model struggled to identify patterns in the data, despite its purported ability to do so. As a result, it yielded relatively high errors, as shown in Table 5.6.

For containers A-D however, the LSTM performed significantly better than for the 100 containers – particularly when predicting their emptying dates, as seen in Table 5.7 and Figure 5.3.

Table 5.6: Evaluation results for the **LSTM** model across the 100 containers.

Seasonality (days)	**No. containers**	Mean E_D	Mean E_F	Mean E_T
7	51	0.35	0.28	0.63
14	15	0.51	0.35	0.85
21	15	0.58	0.40	0.97
28	7	0.48	0.40	0.89
42+	12	0.39	0.34	0.73
Total	100	0.42	0.32	0.74

Table 5.7: Evaluation results for the **LSTM** model on containers A-D.

Subset	**Period** (days)	**Seasonality** (days)	E_D	E_F	E_T
A	7.0	7	0.00	0.09	0.09
B	9.3	14	0.40	0.31	0.71
C	14.0	14	0.00	0.20	0.20
D	21.0	21	0.00	0.41	0.41

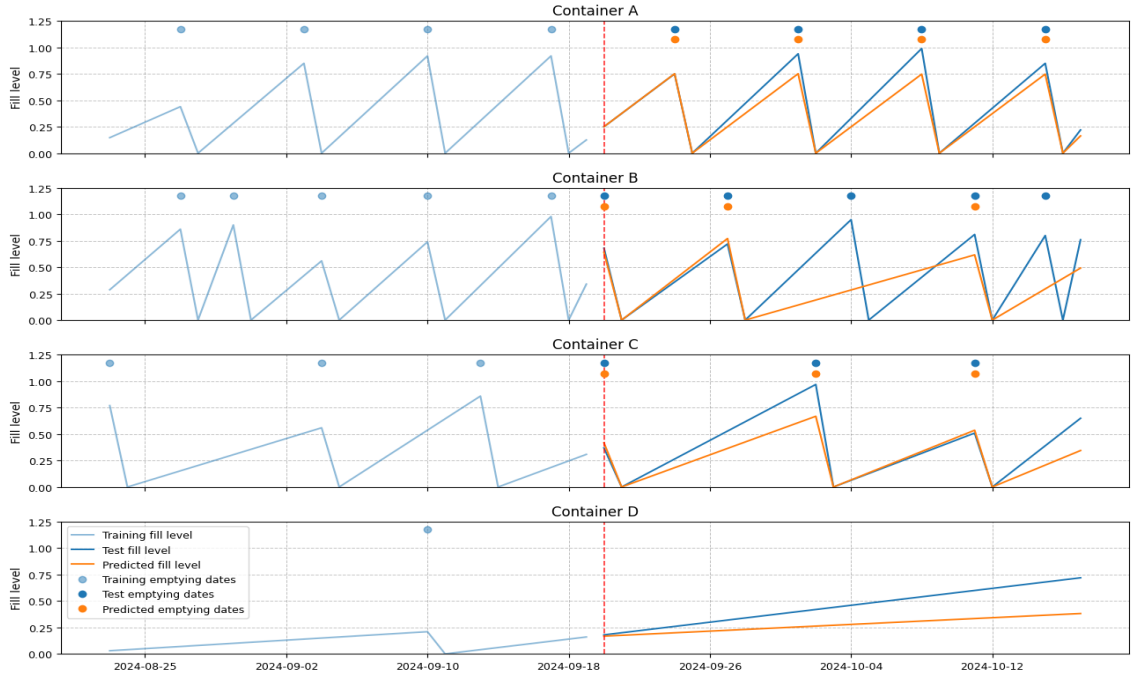


Figure 5.3: Prediction results for containers A-D when using **LSTM**.

5.4 Chronos-Bolt

Tables 5.8 and 5.9 present the evaluation results for the zero-shot and fine-tuned Chronos-Bolt models across the 100 containers. We see small performance differences between the two approaches, with the zero-shot version achieving slightly lower errors. The results, as a whole, highlight the Chronos-Bolt models' strong performance in handling data from containers with a weekly seasonality.

Figure 5.4 and Table 5.10 showcase the predictions and corresponding evaluation scores for containers A-D generated by the zero-shot Chronos-Bolt model.

Table 5.8: Evaluation results for the **zero-shot Chronos-bolt** model across the 100 containers.

Seasonality (days)	**No. containers**	**Mean** E_D	**Mean** E_F	**Mean** E_T
7	50	0.15	0.16	0.31
14	17	0.37	0.26	0.63
21	17	0.36	0.31	0.67
28	4	0.38	0.35	0.72
42+	12	0.28	0.27	0.55
Total	100	0.24	0.22	0.47

Table 5.9: Evaluation results for the **fine-tuned Chronos-bolt** model across the 100 containers.

Seasonality (days)	**No. containers**	Mean E_D	Mean E_F	Mean E_T
7	50	0.17	0.17	0.34
14	17	0.33	0.26	0.58
21	17	0.37	0.31	0.67
28	4	0.37	0.33	0.70
42+	12	0.28	0.32	0.60
Total	100	0.25	0.23	0.48

Table 5.10: Evaluation results for the **zero-shot Chronos-bolt** model on containers A-D.

Subset	**Period** (days)	**Seasonality** (days)	E_D	E_F	E_T
A	7.0	7	0.00	0.06	0.06
B	8.4	7	0.40	0.32	0.72
C	12.0	14	0.00	0.24	0.24
D	21.0	21	0.00	0.53	0.53

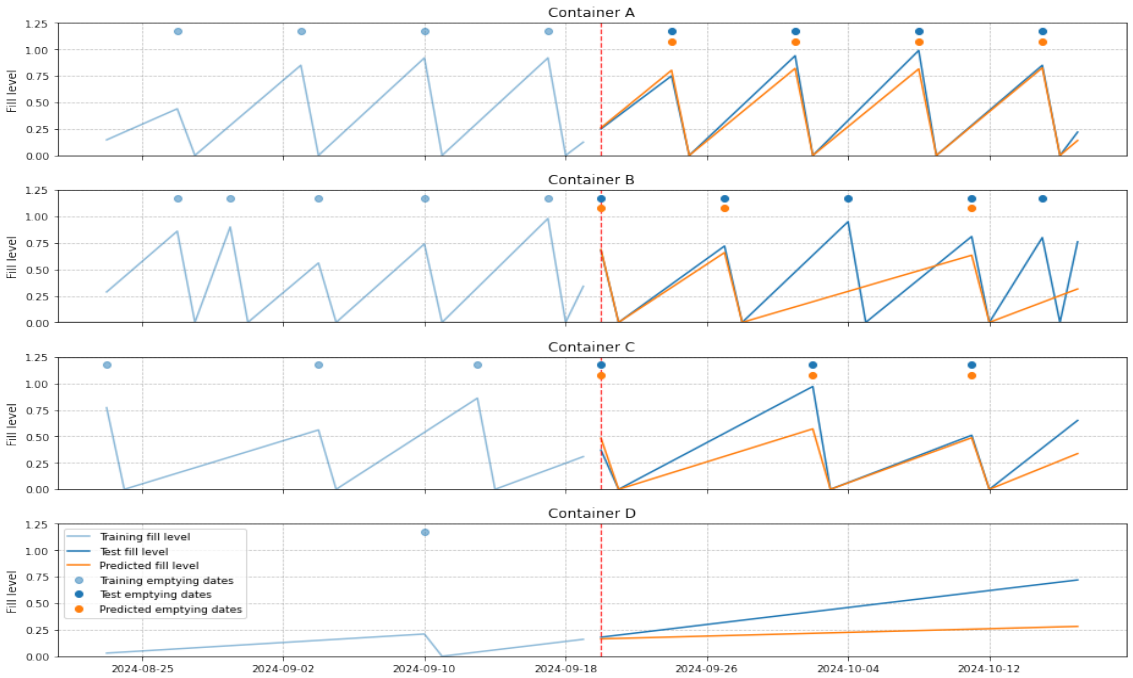


Figure 5.4: Prediction results for containers A-D when using **zero-shot Chronos-Bolt**.

Chapter 6
Discussion

In this chapter, we discuss our results and reflect on the project as a whole. Initially, general observations are addressed, including limitations, difficulties, and alternative approaches. Next, we delve into factors influencing the performance of each model and propose potential solutions and improvements.

6.1 General observations

The main takeaway from our results is the significant improvements over our benchmark model, thus, justifying the use of more advanced models for prediction of waste data. Specifically, our results suggest the use of sARIMAX or Chronos-Bolt for this dataset. In addition, we have made the following observations regarding this project.

Firstly, the amount of collected waste can vary significantly throughout the year. For instance, larger quantities of waste are expected during periods such as Christmas, Black Friday, and other major holidays or events. Hence, such waste patterns could hypothetically be identified and learned by predictive models. Bintel's sensors are designed to operate for a minimum of seven years based on their battery capacity. However, the harsh operating conditions in which they function often reduce their lifespan. As a result, the time series data collected by Bintel spans a maximum period of seven years, though in many cases the actual duration is shorter. This limitation significantly restricts the models' capability to identify and incorporate annual fluctuation patterns into their predictions.

Secondly, due to the limited time frame of this thesis, we chose to evaluate our models on a subset of only 100 monitored containers out of Bintel's approximately 8900. A larger project scope, encompassing *all* available containers, would likely have provided more comprehensive and nuanced results.

Additionally, the fill level prediction error of each model is currently influenced by the accuracy of the emptying date predictions; inaccurate emptying date predictions frequently result in amplified fill level errors. An alternative evaluation strategy could involve using

observed emptying dates instead of predicted ones to construct the sawtooth used in the fill level evaluation. This approach would ensure a more equitable assessment of the fill level errors. Furthermore, separating the two types of predictions would have facilitated the identification of model combinations that could minimize the total error. For example, one might combine sARIMAX for emptying date predictions with Chronos-Bolt for fill level predictions. However, due to time constraints and our primary focus on total error, we did not conduct experiments with this approach.

6.2 sARIMAX

While sARIMAX delivered the lowest total error among our models, there remains room for potential improvement.

As mentioned in Section 4.3, memory management issues restricted us to a maximum seasonality of 42 days. In Table 5.4, we can see that sARIMAX struggled with containers of seasonality 42 days or more. Extending the seasonality parameter could potentially enhance these results by enabling the use of more suitable seasonalities for such containers.

When possible, we used an interval of 12 weeks for estimating the seasonality of a container. However, the fill characteristics of a container might change during such a time period, leading to skewed results. For instance, the last 12 weeks of container B's training data include a summer hiatus. This led us to calculate a seasonality that did not quite align with the most recent emptying patterns. A potentially more effective approach would be to use a validation set to determine the optimal interval for seasonality estimation.

The seasonal component of sARIMAX can significantly enhance model performance when applied to containers with a strong seasonal pattern. Conversely, this component can negatively impact performance when dealing with containers that lack a consistent seasonal pattern. sARIMAX's reliance on fixed seasonality restricts its ability to adapt to evolving container characteristics. While a regular ARIMAX model could be a potential alternative for containers with weak or absent seasonality, our preliminary experiments revealed even worse performance using ARIMAX. However, a possible solution could be to use other models, for example Chronos-Bolt, for such containers.

6.3 LSTM

Interestingly, while multiple related studies have reported strong performance using LSTM networks, our results showed only a marginal improvement over the baseline model.

One possible explanation for this underperformance is the relatively short time series used for training. Neural networks, such as LSTM, excel at uncovering complex patterns in larger datasets, and our limited dataset may have constrained its capacity to do so. However, waste container characteristics evolve over time, as exemplified by container D, so longer data series do not necessarily guarantee more distinct patterns.

Another potential explanation of the suboptimal performance is that we, due to time constraints, performed a rather small grid search, which restricted the selection of optimal hyperparameters.

Moreover, developing a global LSTM model trained on data from all containers, rather

than individual models for each container, might have improved its generalizability and performance. However, the time required to implement and train such a model exceeded the scope of this study.

6.4 Chronos-Bolt

The small performance loss of the fine-tuned version compared to the zero-shot version suggests that the fine-tuning may have resulted in slight overfitting.

Similar to the LSTM, the performance of Chronos-Bolt could potentially benefit from developing it as a global model. However, the strong performance of the zero-shot version suggests that incorporating additional container data may lead to marginal improvements only.

While this project did not primarily focus on the time efficiency of the different models, it is worth mentioning that the zero-shot Chronos-Bolt model showed the most potential in this regard. In preliminary studies, it delivered performance comparable to sARIMAX in a fraction of the time.

Chapter 7
Conclusion

In this Master's thesis, we evaluated four predictive techniques – baseline, sARIMAX, LSTM and Chronos-Bolt – with the purpose of predicting emptying dates and fill levels of waste containers. We carried out experiments using time series data generated by IoT sensor measurements collected by Bintel. The motivation behind this study was to propose a robust solution that Bintel could adopt to address data loss caused by sensor failures, which currently impacts several of their operational services negatively.

Our findings indicate that, among the four techniques, sARIMAX and Chronos-Bolt are the most suitable models for this task. sARIMAX achieved the lowest mean total error (0.45), closely followed by zero-shot Chronos-Bolt (0.47). Despite its slight advantage in prediction accuracy, sARIMAX's performance was influenced by its assumptions of fixed seasonality. This makes Chronos-Bolt a competitive alternative due to its adaptability and computational efficiency.

Interestingly, although multiple studies on waste prediction have reported strong performance from LSTM networks for similar time series tasks, our implementation of LSTM yielded poor results. It achieved a mean total error of 0.74, only slightly better than the baseline model (0.77). This underperformance can possibly be attributed to limited training data, suboptimal hyperparameter tuning due to time constraints, and the usage of separate models for each container rather than a unified global model.

7.1 Future work

To address the challenges identified in this thesis, future work could include using a validation set to determine the optimal interval for seasonality estimation. This could make sARIMAX models more adaptable to changing patterns in the data. Additionally, developing global models trained on data from all containers, rather than individual models, could improve generalizability and the ability to capture shared patterns. Also, evaluating the chosen techniques on all 8900 monitored containers, instead of only 100, would yield more comprehen-

sive and general results. Another potential improvement involves modifying the evaluation approach by separating emptying date predictions from fill level predictions. This approach would enable optimizing models for each task. The best models for each task could then be combined, potentially resulting in better overall performance.

Furthermore, our study only covers a fraction of the techniques and models available for these types of prediction. For instance, foundation models represent a rapidly evolving field, with over 60 such model s developed specifically for time series tasks in just the past two years (Liang et al., 2024). Due to time constraints, we were only able to investigate one of these models, leaving many promising techniques unexplored. Beyond foundation models, the broader fields of machine learning and deep learning include numerous methods with significant potential. In our research, we identified several promising approaches for future studies, including XGBoost and ETSX models.

References

Abbasi, M. and El Hanandeh, A. (2016). Forecasting municipal solid waste generation using artificial intelligence modelling approaches. *Waste management*, 56:13–22.

Abdallah, M., Talib, M. A., Feroz, S., Nasir, Q., Abdalla, H., and Mahfood, B. (2020). Artificial intelligence applications in solid waste management: A systematic research review. *Waste Management*, 109:231–246.

Ahmed, S., Mubarak, S., Du, J. T., and Wibowo, S. (2022). Forecasting the status of municipal waste in smart bins using deep learning. *International Journal of Environmental Research and Public Health*, 19(24):16798.

Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S. S., Pineda Arango, S., Kapoor, S., Zschiegner, J., Maddix, D. C., Mahoney, M. W., Torkkola, K., Gordon Wilson, A., Bohlke-Schneider, M., and Wang, Y. (2024). Chronos: Learning the language of time series. *Transactions on Machine Learning Research*.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Daramola, I. T., Garg, A., and Mehrotra, D. (2024). Multivariate time series forecasting of municipal solid waste. In *2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, pages 1–5. IEEE.

Fokker, E., Koch, T., and Dugundji, E. R. (2023). Short-term time series forecasting for multisite municipal solid waste management. *Procedia Computer Science*, 220:170–179.

Hochreiter, S. (1997). Long short-term memory. *Neural Computation MIT-Press*.

Hyndman, R. J. and Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software*, 27:1–22.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

Liang, Y., Wen, H., Nie, Y., Jiang, Y., Jin, M., Song, D., Pan, S., and Wen, Q. (2024). Foundation models for time series analysis: A tutorial and survey. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pages 6555–6565.

Namoun, A., Tufail, A., Khan, M. Y., Alrehaili, A., Syed, T. A., and BenRhouma, O. (2022). Solid waste generation and disposal using machine learning approaches: a survey of solutions and challenges. *Sustainability*, 14(20):13578.

Navarro-Esbrí, J., Diamadopoulos, E., and Ginestar, D. (2002). Time series analysis and forecasting techniques for municipal solid waste management. *Resources, Conservation and Recycling*, 35(3):201–214.

Niu, D., Wu, F., Dai, S., He, S., and Wu, B. (2021). Detection of long-term effect in forecasting municipal solid waste using a long short-term memory neural network. *Journal of Cleaner Production*, 290:125187.

Pearson, K. and Galton, F. (1895). Vii. note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58(347-352):240–242.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Tay, Y., Dehghani, M., Rao, J., Fedus, W., Abnar, S., Chung, H. W., Narang, S., Yogatama, D., Vaswani, A., and Metzler, D. (2021). Scale efficiently: Insights from pre-training and fine-tuning transformers. *arXiv preprint arXiv:2109.10686*.

Vagropoulos, S. I., Chouliaras, G. I., Kardakos, E. G., Simoglou, C. K., and Bakirtzis, A. G. (2016). Comparison of sarimax, sarima, modified sarima and ann-based models for short-term pv generation forecasting. In *2016 IEEE International Energy Conference (ENERGYCON)*, page 1–6. IEEE.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.(nips), 2017. *arXiv preprint arXiv:1706.03762*, 10:S0140525X16001837.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2025-01-30

EXAMENSARBETE Predicting Missing Waste Data With Machine Learning
STUDENT Ellen Andreasson & Otto Grafström
HANDLEDARE Pierre Nugues (LTH), Axel Rahm & Fredrik Olsson (Backtick Technologies)
EXAMINATOR Jacek Malec (LTH)

Prediktion av förlorad avfallsdata med hjälp av maskinlärning

POPULÄRVETENSKAPLIG SAMMANFATTNING **Ellen Andreasson & Otto Grafström**

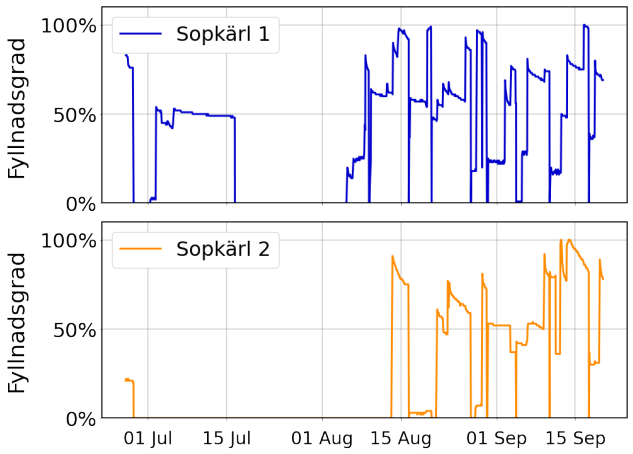
Effektiv sophantering är en viktig utmaning i dagens samhälle. I detta arbete testar vi olika maskininlärningsmodellers förmåga att förutsäga fyllnadsmönster hos sopkärl.

Tänk dig ett samhälle där sopbilar kör helt i onödan till halvfulla kärl eller missar överfyllda sopstationer. Effektiv avfallshantering är avgörande för att undvika onödiga transporter, spara resurser och minska vår påverkan på miljön. I takt med att fler använder smarta soptunnor med sensorer som mäter fyllnadsnivåer, öppnas nya möjligheter för att skapa en mer hållbar sophantering. Men vad händer när dessa sensorer går sönder och data går förlorad? Här tar vårt arbete vid.

I vår studie undersökte vi hur förlorad sensordata kan återskapas med hjälp av smarta modeller. Vi testade fyra olika modeller: en enkel referensmodell, den statistiska modellen sARIMAX, ett LSTM-baserat neuralt nätverk och den toppmoderna, avancerade modellen Chronos-Bolt. Målet var att förutse två saker: när ett sopkärl töms och hur fullt det är vid tömningstillfället. Resultaten visade att sARIMAX var den mest träffsäkra modellen, tätt följd av Chronos-Bolt. LSTM-modellen, som ofta lyfts fram som effektiv för tidsserieprediktioner, lyckades däremot inte lika väl på vårt dataset.

Många soptunnor visade sig ha oregelbundna fyllnadsmönster som var svåra att förutsäga. För att förbättra modellernas prediktioner för ett kärl utnyttjade vi istället data från andra soptunnor med liknande fyllnadsmönster, oavsett om de stod i samma soprum eller tvärs över landet.

Överst i figuren ser vi ett exempel på ett kärl med ett oregelbundet fyllnadsmönster. Det står vid en skola och är därför tomt under större delen av sommarlovet. Figuren visar också data för sopkärl 2 som användes för att underlätta för modellerna i deras förutsägelser av sopkärl 1.



Vår studie visar hur maskininlärning kan bli en kraftfull verktygslåda för framtidens avfallshantering. Genom att använda intelligenta algoritmer kan vi inte bara förutse och åtgärda förlorad data, utan också skapa smartare och miljövänligare system som bidrar till ett mer hållbart samhälle.