Norwegian University
of Life Sciences

Master's Thesis 2024 30 ECTS
Faculty of Science and Technology

Prediction of Bus Dwell Time using Time Series Analysis and Machine Learning

Vegard Molaug
MSc Data Science

Abstract

The reliability of public transportation is strongly dependent on punctual transit trips. Precise bus dwell time (BDT) predictions are important in this regard, as BDT directly influences arrival times and departure times at stops. Since BDT is defined as the duration for which a bus remains stationary at a stop to service passengers, accurate estimates of BDT can enable transit companies to optimize bus scheduling in a larger network of stops and routes. This thesis explored the possibility of modeling BDT as a time series with external predictors. SARIMAX, LSTM and XGBoost were used to predict BDT at selected bus stops on line 20 in the Oslo region. We used data from 19 different stops in the eastward driving direction of line 20. The BDT data at all stops was sampled in 15-minute intervals from 06:00 to 23:00 every day during the first quarter of 2023. In this work, seven predictors were used to model BDT. These seven predictors were either a category of passenger variables or temporal variables. Our results indicated that passenger variables were more important in predicting BDT than temporal variables. There were also significant variations in feature importance between the different stops. The results showed that XGBoost was the best model in our research. It had the lowest prediction error for our selected evaluation metrics across all stops, with cumulated RMSE, MAE and MAPE scores of 99.33, 78.85 and 3.67 respectively. LSTM yielded prediction errors that were similar to those of XGBoost, producing RMSE, MAE and MAPE scores of 100.34, 79.82 and 3.73 respectively. SARIMAX had the highest prediction errors among the models. These results indicated that a time series approach for modeling BDT may not be optimal compared to other methods. This claim is also supported by the fact that the data did not appear to have strong temporal characteristics.

Acknowledgements

I wish to express my sincere thanks to my supervisors, Habib Ullah and Stefan Schrunner, for their invaluable advice and guidance on my thesis.

I also would like to thank my co-supervisor at Ruter, Christian Svendsen Mjøsund. The discussions we had were helpful for steering the thesis in the right direction. I also extend my gratitude to my other colleagues at Ruter for their valuable tips throughout the master's process. I would also like to thank Ruter for supplying the research questions that made this thesis possible.

A special thanks also goes out to my girlfriend, family, and friends who provided valuable support throughout the work of this thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Public transportation is important for sustainable urban development, as it is an eco-friendly alternative to personal vehicles. The role of public transportation also extends beyond sustainability by operating as an efficient and accessible mobility option for all walks of life. Since increased urbanization leads to higher mobility demands, the importance of efficient public transportation systems increases. Efficient operation management can still be difficult in this regard, as the public transportation landscape includes challenges related to optimization problems and customer demands. When surveyed about the defining features of an attractive public transportation system, customers may have slight variations in their individual preferences. Despite this, factors such as punctuality and travel time rank highly across multiple customer demographics [1]. This emphasizes that customers want to get to their destination on time, and that they want to get there as quickly as possible. These customer needs could be violated if a transit vehicle is delayed for parts of a trip.

Traffic delays can be described as a disruption in the traffic flow of transit vehicles, leading to extended travel times beyond the scheduled duration of a trip. The ramifications beyond the increase in travel time could also include missed connections, which can be perceived as an inconvenience to customers. To counteract these undesirable traffic delays, we need to know where and how travel time could be affected in a transit journey. The accumulation of bus travel time during a transit trip occurs primarily in two main segments of a bus journey: Link travel (the journey between two successive stops on a bus route) and at the stops themselves [2]. The duration in which a bus remains stationary at a stop is often known as bus dwell time (BDT), and BDT will be the main focus of the analysis in this thesis.

BDT is a significant component of the travel time spent on a transit trip. For bus trips in Sydney, 15% of the total travel time was reported to consist of BDT [3]. This means that BDT directly impacts overall service punctuality, as the time spent at a stop can influence the arrival time at subsequent stops. BDT variability can also present challenges with bus headway, which is the time between subsequent transit vehicles arriving at a stop. These factors can introduce unreliability into public transportation services, which in turn can negatively impact the customer satisfaction rates of a transit company. Decreasing customer satisfaction rates is undesirable for public transportation companies as it can potentially result in reduced customer retention. This is because customers can switch to another mode of transportation if they feel that public transportation services cannot meet their transportation needs.

Most people want their transportation journey to be easy and accessible, and public transportation companies could provide this by having highly optimized schedules throughout their transit networks. Data is the foundation for efficient schedule optimization, which implies that accurate estimations of traffic flow can help design sufficient transit systems. As BDT is a significant component of travel time, accurate BDT estimates can help public transportation companies become more efficient. By modeling BDT along different combinations of independent variables, it is possible to analyze how different variables impact BDT output.

1.2 Research questions

There have already been several attempts at modeling BDT in the literature. The related work on BDT estimation will be further discussed in Chapter 2. Most of these estimation attempts have been performed through regression variants, probabilistic methods or machine learning. The approach of modeling BDT as a time series problem has been explored in the BDT literature, though only as a univariate time series without external predictors. This leaves the effectiveness of our suggested approach unknown, as we in this work aim to model BDT as a time series with external predictors. A significant share of the approaches performed in the BDT literature also utilize a single model in their work. Since comparisons between results from different papers should be done carefully, not using multiple estimation approaches per paper is a limitation in the existing work. Therefore, the thesis will address the following questions:

1. Which models demonstrate superior performance in predicting BDT values?
2. Which features exhibit the highest predictive power for predicting BDT values?
3. Is it advisable to use a time series with external predictors to predict BDT values?

1.3 Research limitations

Although this research provides insight into BDT modeling, it also has some limitations that must be acknowledged. Firstly, the conducted analysis is limited to the stops on a single bus route in a single direction. Although this helps us to clarify the analysis findings, this narrow focus could also limit the applicability of our results to other bus routes. In addition, our models are primarily trained on temporal characteristics and passenger variables. Beyond these determinants, the literature mentions several categories of BDT determinants that could have been helpful in our work. Regrettably, the scope of the project restricted our capacity to incorporate every determinant that we considered significant for BDT. The study also exclusively uses data from the first quarter of 2023, which means that potential seasonal variations and long-term BDT trends are not adequately captured.

1.4 Industrial partner

This work was developed on behalf of Ruter. Ruter oversees the coordination and planning of public transport services in Akershus and Oslo. The company has a long-term vision of providing sustainable mobility freedom to people within their business area [1]. Mobility freedom means freedom for residents to travel wherever and whenever they want, facilitating accessible travel options for all residents regardless of their circumstances. The goal of mobility freedom is also aligned with sustainability goals for the environment, the society and the customers living in Ruter's market areas. This is due to Norway's commitment to the UN's sustainable development goals, which indirectly also applies to Ruter as a public organization.

To achieve their goals within mobility and sustainability, Ruter has defined data as an important tool for success. As data has become an increasingly important part of the modern era, Ruter has emerged as a frontrunner within public companies for its data-driven business model and AI solutions. Ruter utilizes user data in these solutions to return an improved transit experience back to its users. By embracing the potential of data and technology, Ruter has the potential to shape the future of public transportation in the Oslo region.

Chapter 2

Theory

2.1 Bus dwell time

In the literature, BDT is usually defined as the time a bus takes to load and unload passengers while stationary at a stop [4]. The exact BDT duration is the difference between the time the bus door opens and the time the bus door closes. Although passenger dynamics is often considered a primary influencer of BDT, other factors could also be important for BDT dynamics. Several approaches have been used to estimate BDT and its predictors. Only studies from 2013 and beyond have been considered when reviewing the BDT literature in Section 2.1.1. This is because the older BDT literature mostly utilizes linear regression, which is a limited estimator compared to the methods that have often been used in the past decade.

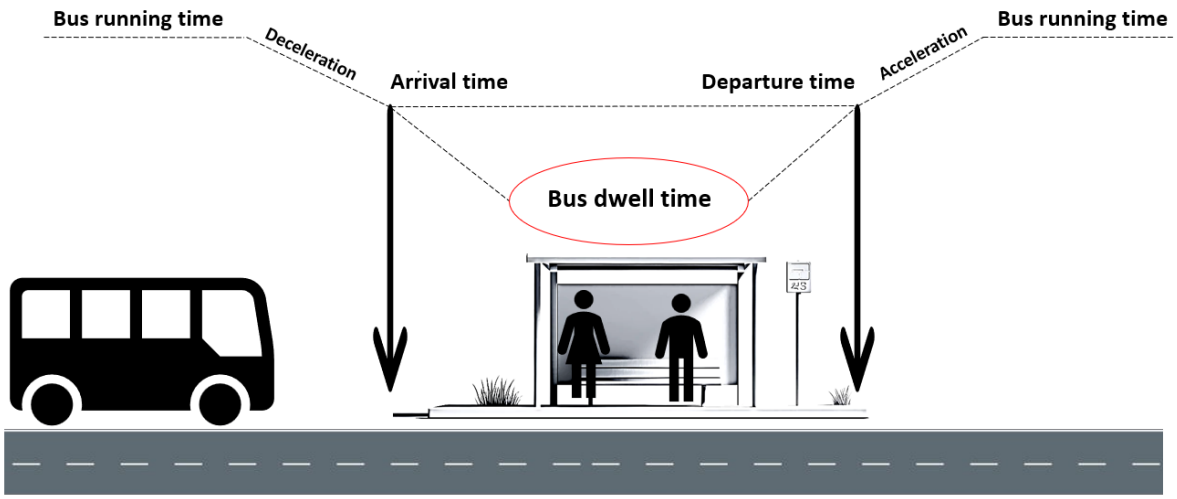

Figure 2.1: Illustration of bus dwell time at a stop. Inspired by [5]

.

2.1.1 Related work

Meng and Qu (2013) proposed a probabilistic approach to deal with the high degree of uncertainty associated with BDT [6]. This uncertainty was attributed to the way buses merge with vehicles in the breakdown lane. The authors demonstrated the effectiveness of their methodology through a case study, in which the impact of advisory signs was highlighted as a tool to reduce BDT and improve bus capacity. Rashidi and Ranjitkar (2015) evaluated four separate methods to estimate BDT as a univariate time series without the inclusion of external predictors. The models they utilized included random walk, exponential smoothing, moving average and autoregressive integrated moving average [5]. Their findings revealed that the performance of these time series models varied significantly between central business districts (CBD) and non-CBD bus stops. They also concluded that the observed BDT in Auckland had different characteristics than the BDT trends in other cities.

Xin and Chen (2016) introduced a dynamic KNN-based model to predict BDT at downstream bus stops using historical GPS data [7]. They suggested that different bus stops need different prediction models, as different stops might have unique characteristics when it comes to passenger dynamics. An average mean absolute error of 2.30 across 30 unique stops suggested that the utilized KNN model was generalizable across a larger network of stops. Mohammad et al. (2017) incorporated a regression model and correlation analysis to analyze BDT at key bus stops in Kuala Lumpur [8]. The results indicated that passenger activity significantly impacts BDT. A constant 4-second delay was also assigned to all onboard passengers due to the stairs of the high-floor buses. Lift usage was also reported to have the greatest impact on BDT by adding 71.32 seconds to the duration of stops.

Isukapati et al. (2017) analyzed the statistical properties of BDT during peak AM hours using Automatic Vehicle Location data from the Pittsburgh Port Authority [4]. The statistical properties of BDT appeared to be similar across multiple seasons for most of the stops, and the daily trends in the BDT distribution tended to be highly individual for each stop. BDT was also described as highly stochastic, indicating that independent variables should be treated as random or stochastic regressors when modeling BDT. Iskupati and colleagues also tried to predict BDT in 2020, where they introduced a Hierarchical Bayesian Framework as an approach [9]. The Bayesian predictive method performed better on the training data than the other models in the study.

Glick and Figlozzi (2017) used multiple regression to measure the impact of traffic conditions and passenger numbers on BDT [10]. By comparing various models with different combinations of independent variables, they concluded that GPS trajectory data could help estimate BDT. Their findings also suggested a nonlinear relationship between pas-

senger numbers and dwell time, suggesting that efficiency could increase with higher passenger volumes. Glick and Figlozzi also tried to model BDT in 2019 with log-linear regression and quantile regression [11]. They emphasized the limitations of linear regression models for BDT data. This is due to the assumption of homoskedasticity, which is often violated in the context of BDT estimation. The authors suggest that log-linear models can improve the robustness of predictions over linear models by stabilizing the variance across the spectrum of predicted BDT. Quantile regression was also concluded to be a robust prediction framework in the presence of heteroskedasticity.

Abad and Fillone (2018) discussed the estimation of BDT using artificial neural networks and linear regression models, with passenger variables as external predictors [12]. The study considered a 3-27-1 network as the optimal network structure based on correlation coefficients. Comparisons between the mentioned models showed equal levels of prediction capability, with both models producing error rates that suggested room for improvement. The authors suggested that including variables beyond passenger numbers could improve predictive accuracy. Aswini (2023) used machine learning models to estimate BDT [13]. Out of the models used, XGBoost demonstrated the best model performance for all chosen metrics. The mean absolute percentage error was above 20% for all the models used, indicating the need for performance improvements in future studies.

2.2 Time series

A time series comprises a collection of data points measured at successive time points [14]. Time series observations are usually equally spaced throughout the length of a record. Arranging these observations chronologically makes it possible to analyze how a given variable changes over time. The order of observations is crucial, as changes to the data order could alter the information of the data. A time series can be univariate or multivariate by modeling either one or more variables over time, respectively [15].

Most of the methods within statistics are meant to be used for experiments in which the data is independent and identically distributed (i.i.d.). Methods that assume i.i.d. data may not be applicable for time series data since sequential observations often will violate the property of independence. This is because sequential records are likely to be correlated with their predecessors and successors. An example of this could be daily temperature measurements. Temperature on consecutive days tends to be correlated over time, so individual temperature measurements cannot be considered independent. Statistical properties such as mean and variance can also vary across different seasons of the year, which means that individual observations do not share the same probability distribution across the time axis.

Time series models are designed to forecast future data points based on the characteristics of the observable data. The characteristics these models aim to capture can be divided into components that play an important role in shaping the overall structure of a time series. These components are seasonalities, cycles, trends, and residuals [16]. Seasonality refers to data patterns that repeat at specific periods within fixed intervals. A cycle occurs when the data fluctuates in a non-fixed period. The trend component indicates the data's direction over a longer period of time, while the residual component captures the unpredictable fluctuations in the data which could be attributed to general randomness.

2.2.1 Statistical foundations of time series analysis

Time series analysis relies on a few key statistical concepts. One of these concepts is autocorrelation, a measure of the linear relationship between a variable and its lagged values. The formula for this measure is denoted as the autocorrelation function (ACF), which can be written as:

$$r_k = \frac{\sum_{t=k+1}^{T}(y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T}(y_t - \bar{y})^2} \tag{2.1}$$

The value r_k denotes the autocorrelation coefficients for observation k, while T represents the entire time series. r_1 denotes the correlation between y_t and y_{t-1}, r_2 denotes the correlation between y_t and y_{t-2}, etc. [16].

Another central statistical concept within time series analysis is stationarity. A stationary time series can be defined as having constant mean, variance and autocorrelation over time [17]. Stationarity is important because many time series models would not perform well with input data that has changing statistical properties over time. Time series with trends or seasonalities would be examples of such non-stationary data. In such cases, differencing can be applied to make the time series stationary. Differencing can make a time series stationary by removing time-dependent structures. The first differencing of a time series X_t is given by the difference between an observation and its previous observation

$$\Delta X_t = X_t - X_{t-1} \tag{2.2}$$

If a time series has a nonlinear or exponential trend, first differencing might not be sufficient to achieve stationarity. In such cases, the differencing operation can be applied more than once to remove the trend component. This is often referred to as higher-order differencing. For seasonal time series, stationarity can be achieved by calculating the

difference between an observation and the preceding observation of the current season [16].

2.2.2 Time series models

The equations and theory described in this subsection are retrieved from [16].

An autoregressive model (AR) is a regression model that uses the past and present values of a time series to make forecasts. The notation AR(p) indicates an AR model of order p, and is defined by the equation:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t \tag{2.3}$$

The AR model is essentially a multiple regression with lagged values of y_t as predictors. ϕ_i are the present values of the variable, while ε_t is white noise. Similarly, a moving average (MA) model utilizes past white noise errors of a time series to make forecasts. This is given by Equation (2.4), where ε_t denotes the error terms and θ_q expresses the model parameters. MA models are often written as MA(q), where q represents the order of lagged error terms.

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}, \tag{2.4}$$

AR and MA models can be combined with a differencing term to make an autoregressive integrated moving average (ARIMA) model. Whereas AR and MA models are applicable exclusively for stationary data, the differencing step within ARIMA models enables the possibility of handling nonstationary data. The parameter notations of ARIMA models are often expressed as (p, d, q), where p represents the autoregressive order, d denotes the degree of differencing needed to make the time series stationary, and q is the order of the moving average component. An ARIMA model can be expressed by Equation (2.5), where y'_t is a differenced series consisting of lagged errors and lagged values of y_t.

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t \tag{2.5}$$

ARIMA models are used primarily for univariate data. In cases where it is desirable to model multivariate data, a seasonal ARIMA (SARIMAX) model with exogenous vari-

ables can suffice. A SARIMAX model integrates seasonal and nonseasonal parameters with exogenous variables to make forecasts. The SARIMAX model has a notation that is usually expressed as SARIMAX$(p, d, q)(P, D, Q)_m$. In addition to using the same non-seasonal parameters of (p, d, q) as the ARIMA model, the SARIMAX model also includes seasonal parameters defined as (P, D, Q). The seasonal parameters behave similarly to their nonseasonal counterparts, where P is seasonal autoregression, D is seasonal differencing and Q is a seasonal moving average. These parameters are applied according to the seasonal period m, which indicates the total number of time units in a single season.

2.2.3 Fourier series

Fourier series is a time series technique that is useful for handling data with long seasonal periods, first demonstrated by the French mathematician Jean-Baptiste Fourier. Fourier demonstrated that any periodic function could be approximated by a series of sine and cosine functions of different frequencies. This allows us to use Fourier terms as external predictors in a time series to model seasonal patterns. The mathematical representation of Fourier terms for a seasonal pattern can be expressed as

$$x_{1,t} = \sin\left(\frac{2\pi t}{m}\right), \qquad x_{2,t} = \cos\left(\frac{2\pi t}{m}\right), \qquad x_{3,t} = \sin\left(\frac{4\pi t}{m}\right),$$
$$x_{4,t} = \cos\left(\frac{4\pi t}{m}\right), \qquad x_{5,t} = \sin\left(\frac{6\pi t}{m}\right), \qquad x_{6,t} = \cos\left(\frac{6\pi t}{m}\right), ...$$

for a predefined K number of Fourier pairs [16]. In practical scenarios, Fourier terms can be more computationally efficient when dealing with long seasonal periods. For cases with shorter seasonal periods, such as yearly data, Fourier terms offer little to no advantage over other seasonal techniques.

2.3 Machine learning

Machine learning is an area within artificial intelligence that enables software programs to identify patterns in data and make subsequent predictions with minimal explicit programming [18]. This learning process is performed by implementing machine learning algorithms with varying complexity and architecture. Machine learning models learn by reducing the discrepancy between predicted outcomes and actual results, which can be achieved through iterative adjustment of parameters within the models. Learning strategies are also executed in other ways, entirely determined by the data structure and the analysis goal at hand.

2.3.1 Supervised learning

Supervised learning involves predicting a response variable based on observed data [19]. This is typically done through a training dataset where each data sample has a label that categorizes the sample into a distinct class. The main objective of supervised learning is to develop models that can identify hidden patterns in the training dataset. Inferences drawn from the training data set can then be utilized to categorize previously unseen data. In addition to classification tasks, supervised learning is also applicable in regression tasks aimed at predicting continuous outcomes.

2.3.2 Unsupervised learning

Unsupervised learning involves the analysis of data without predefined labels [20]. Central to unsupervised learning is the analysis of the underlying structures in the data. This could involve identifying similarities and differences among data points to group them into clusters, also known as cluster analysis. Dimensionality reduction is also a process that is often used within supervised learning. It involves reducing the dimensionality of the data to find a smaller subset that preserves the essential characteristics of the data.

2.3.3 Bias-variance tradeoff

In supervised machine learning, the aim is to create predictive models that can be applied effectively to unseen data. This would infer the creation of models that minimize bias and variance of training samples, as these two terms influence a model's ability to fit the data properly. Bias refers to errors caused by the learning algorithm making simplistic assumptions about the data. Excessive bias in a machine learning model can lead to overlooked correlations between features and target outputs in a given dataset, a condition typically referred to as underfitting. The variance represents the error due to the model's sensitivity to minor variations in the training data. High variance can lead to a scenario where noisy training observations are extensively weighted, which is usually referred to as overfitting. The search for a sufficient tradeoff between bias and variance is a dilemma all machine learning models face, with some models struggling more with one of the error terms than the other. A simple model with few parameters, such as linear regression, could often be susceptible to high bias and low variance. This is because it may deliver consistent performances across datasets, yet fail to capture complex relationships between features. A complex model with many parameters, such as a deep neural network, could conversely exhibit low bias and high variance. This is due to the ability of these models to capture complex relationships, which could also vary greatly between different datasets [21].

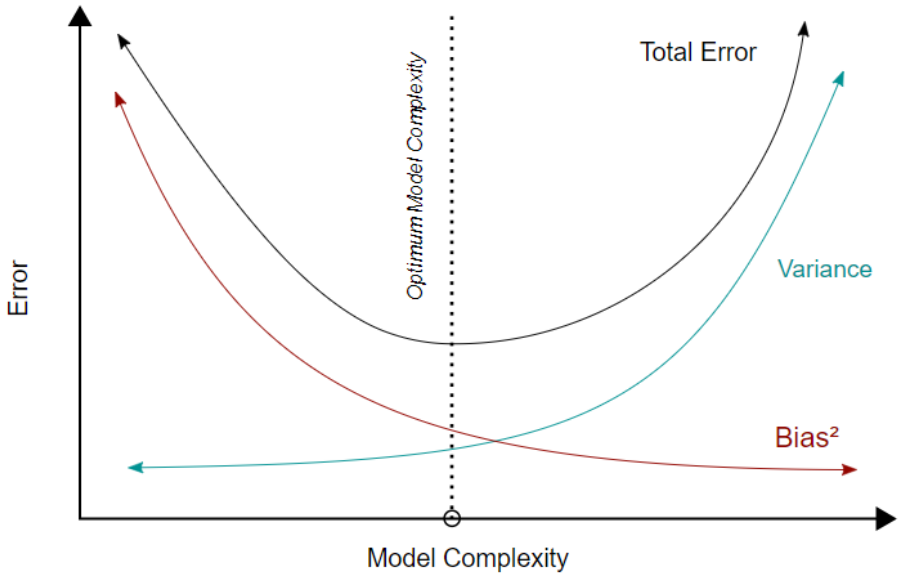

Figure 2.2: Illustration of the bias-variance-tradeoff for model complexity and model error. The illustration is available through the CC0 license, and was retrieved from: **https://commons.wikimedia.org/wiki/File:Bias_and_variance_contributing_to_total_error.svg**

2.3.4 Cost function

The equations and theory presented in this subsection were retrieved from [22].

In supervised machine learning, the model training phase aims to reduce the cost function of a model. A cost function is a quantitative measure between the predictions of a model and the actual observations in the data. In this regard, minimizing the cost function will reduce the prediction errors on a given data set. Optimization algorithms are used to efficiently navigate the parameter space of a model to identify values that correspond to the optimal performance of a model. In the case of a simpler model like Adaline, gradient descent is used as an optimization algorithm to minimize the cost function. The main principle behind gradient descent is to find a local or global cost minimum. This is done by taking iterative steps in the negative direction of the gradient. Traversing in the direction opposite to the gradient of the cost function defined as $J(\mathbf{w})$, gradient descent updates the network weights using the formula:

$$\mathbf{w} := \mathbf{w} + \Delta\mathbf{w} \tag{2.6}$$

Adjustments in the weights $\Delta\mathbf{w}$, results from the product of the negative gradient and

the learning rate η.

$$\Delta \mathbf{w} = -\eta \nabla J(\mathbf{w}) \tag{2.7}$$

The learning rate defines the magnitude of the descending steps along $J(\mathbf{w})$. Next, the partial derivative of the cost function is needed to calculate the gradient of the cost function for every weight w_j, which is defined as:

$$\frac{\partial J}{\partial w_j} = -\sum_i (y^{(i)} - \phi(z^{(i)}))x_j^{(i)} \tag{2.8}$$

In this expression, $y^{(i)}$ represents the actual output values, $\phi(z^{(i)})$ is the predicted output of the model, and $x_j^{(i)}$ are the feature values associated with weight w_j for each observation i. By combining previous formulas, the final update for the weights w_j can be formulated as:

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j} = \sum_i (y^{(i)} - \phi(z^{(i)}))x_j^{(i)} \tag{2.9}$$

2.4 Ensemble learning

Ensemble learning is an approach within machine learning that combines multiple predictive models to produce more robust predictions. The logic of this approach lies in the proposition that whilst single models may be vulnerable to mistakes from noise or bias, a collective of models can mitigate these errors by compensating for one another. These groups of models have their predictions aggregated after the models have been trained on the same data. The ensemble methods relevant to this thesis are described in the following subsections.

2.4.1 Decision trees

A fundamental component of several ensemble learning methods is decision trees. Decision trees extrapolate observations for an item towards conclusions about the item's target value. It is a non-linear classifier that recursively partitions the data space into subsets based on the value of a feature [23]. A decision tree is typically represented as a binary tree with nodes. Every internal node corresponds to a feature test, every branch denotes the test result and each leaf node signifies a class label or output value. The root node determines the initial division of the data into subsets, followed by the subsequent child nodes which divide the data until no further partitioning is necessary.

2.4.2 Boosting

Boosting is an ensemble method designed to convert a collection of weak learners into a strong learner by iteratively focusing on errors made by previous models. These weak learners are typically short decision trees often referred to as stumps [23]. The stumps usually have a small number of splits that yield minimal complexity and reduced learning capacity. The central principle behind boosting is to weigh the training instances differently by prioritizing instances that previous learners misclassified. Each successive model in the boosting sequence focuses on the hardest examples by increasing their weights, whereas correctly classified instances have their weights decreased. This approach allows the ensemble to focus on the most challenging aspects of the training data. The final model is a weighted sum of weak learners, where more accurate classifiers receive a higher weight on predictions.

2.4.3 Gradient boosting

Gradient boosting extends the principles of boosting by using a gradient descent algorithm to minimize the loss function. Optimization is carried out in the function space rather than in the parameter space by fitting new models to the residuals of previous predictions [24]. Each new model is added to the ensemble to address the most significant shortcomings of the current ensemble's performance, which is guided by the gradient. Optimizing the learning rate and the number of trees is important to balance bias and variance.

2.4.4 XGBoost

Extreme Gradient Boosting (XGBoost) models improve upon gradient boosting by introducing more efficient parallel processing, tree-pruning and regularization. XGBoost is highly scalable for large datasets, sparsity-aware for parallel tree learning and explicitly designed to handle missing data during training [25]. The improved scalability is achieved through a weighted quantile sketch that proposes potential split candidates rather than the greedy algorithm used in other tree-based methods. XGBoost often provides state-of-the-art results for multiple classification and regression problems. This has made it one of the most utilized algorithms in the competitive machine learning scene.

2.4.5 Isolation Forest

The Isolation Forest algorithm is an anomaly detection method for identifying outliers in a dataset. Isolation Forest uses a tree-based structure to recursively divide the data to detect outliers, where outliers are expected to have a shorter path length in the trees

than normal data points. The training phase and the scoring phase are the two main phases that comprise the Isolation Forest algorithm. In the training phase, a subsample of size ψ is randomly drawn from the dataset. A binary tree is then built from this subsample by recursively splitting the data using a randomly chosen feature and split value, which is repeated to construct t trees. After the training phase, the path length from the root node to the leaf node is calculated across all trees for each data point. The average path length is calculated and normalized to produce an anomaly score. This score reflects the ease with which the data point can be isolated from the rest of the sample. The anomaly score $s(x, n)$ for a data point x is defined as

$$s(x, n) = 2^{-\frac{E(h(x))}{C(n)}}, \tag{2.10}$$

where $E(h(x))$ represents the expected path length of x across the trees, and $C(n)$ denotes the mean path length for unsuccessful searches in a Binary Search Tree [26]. $C(n)$ is a normalization factor to make the scores comparable for different datasets. The value of s implies different characteristics of the data points. Instances where $s \to 1$ indicates outliers and instances where $s \to 0$ indicates normal data points. A dataset does not contain significant anomalies if $s \approx 0.5$ for all data samples.

2.5 Deep learning

Deep learning is a part of machine learning distinguished by its use of neurons and layers for complex modeling problems. The arrangement of layers and neurons was initially influenced by the way interconnected nerve cells of the brain handle incoming and outgoing signals. The theory and equations presented in Sections 2.5.1 and 2.5.2 were retrieved from [22].

2.5.1 Architecture of neural networks

Numerous varieties of neural networks exists in the deep learning space which are designed for specific applications and problems. In the case of dense neural networks, their architecture often consists of an input layer, one or more hidden layers and an output layer. Raw data is initially passed on to the input layer, where the data is transformed to a format the subsequent layers can use. Next, the data is passed on to the hidden layers which progressively extract more complex data representations as the data moves through the network. The output layer produces a prediction based on the learned features of the previous layers. Figure 2.3 shows an example of a simple artificial neural network with a single hidden layer. The lines connecting the nodes across different layers in Figure 2.3 represent the weights of the connections between neurons. The weights in a neural network determine the influence that a neuron's output will have on

another neuron's activation in the next layer. Each neuron processes the output of its predecessors as a weighted sum wrapped around an activation function.

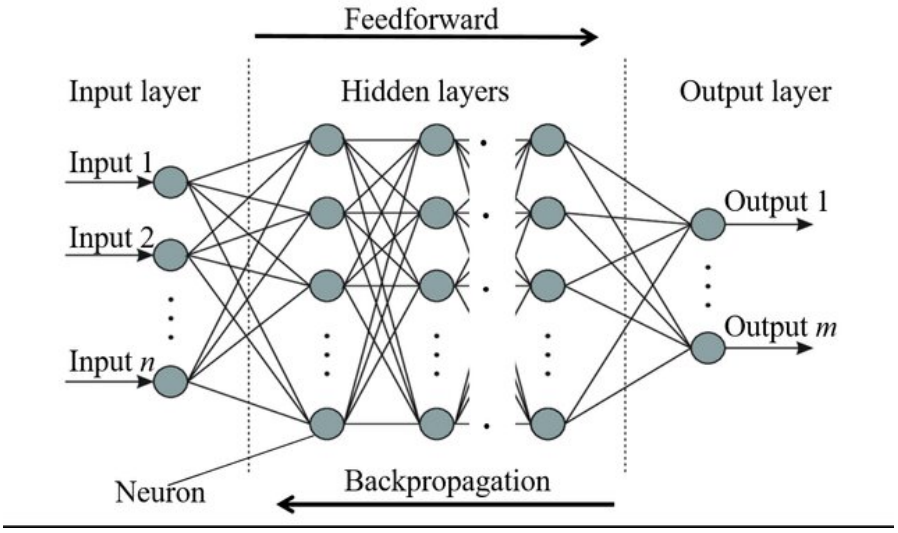

Figure 2.3: Architecture of a dense artificial neural network, available through CC BY 4.0 license. Retrieved from: **https://www.researchgate.net/publication/358145060_A_deep_learning_energy_method_for_hyperelasticity_and_viscoelasticity/figures?lo=1**

2.5.2 Learning process in neural networks

The learning process in a neural network enables the network to adjust its weights and biases to improve its performance over multiple iterations on the dataset. This process starts with forward propagation, where inputs are passed through the network layers to generate outputs that serve as input for the following layers. Forward propagation continues until the output layer produces a prediction. Once the output is obtained through forward propagation, the predicted outputs are evaluated using a loss function. The loss function measures the difference between the actual values and the predictions. Some typical loss functions are cross-entropy loss for classification problems and mean squared error for regression problems. The performance of a model improves as the loss decreases. After a model's loss has been computed, the error rates of a model's parameters are fine-tuned through backpropagation.

Backpropagation is a method of adjusting the weights and biases in a neural network by propagating the error backward through the network layers. The error is then used to compute the gradient of the loss function for each parameter in the network, thereby gradually adjusting these parameters to decrease the model loss. Given a simple neural network with a single hidden layer, the following steps express the overall backpropagation process:

1.

$$\delta^{(out)} = a^{(out)} - y \tag{2.11}$$

The output layer's error term $\delta^{(out)}$ is determined by subtracting the target y from the output layer's activation $a^{(out)}$. This error term is utilized to calculate the gradient of the loss function with respect to the weights.

2.

$$\frac{\partial}{\partial w_{i,j}^{(out)}} J(W) = a_j^{(h)} \cdot \delta_i^{(out)} \tag{2.12}$$

The loss gradient $J(W)$ with respect to the weights between the output layer and the hidden layer and $w_{i,j}^{(out)}$ is computed as the product of the activation of neuron j in the hidden layer $a_j^{(h)}$, and the error term of output neuron i in $\delta_i^{(out)}$.

3.

$$\delta^{(h)} = \delta^{(out)} \cdot (W^{(out)})^T \odot \frac{\partial \phi(a^{(h)})}{\partial a^{(h)}} \tag{2.13}$$

The error term for the hidden layer $\delta^{(h)}$ is the transposed product of the weight matrix of the output layer $(W^{(out)})^T$, and the error term of the output layer $\delta^{(out)}$. This is element-wise multiplied with the derivative of the activation function ϕ applied to the activations of the hidden layer $a^{(h)}$.

4.

$$\frac{\partial}{\partial w_{i,j}^{(h)}} J(W) = a_j^{(in)} \cdot \delta_i^{(h)} \tag{2.14}$$

$J(W)$, which is the gradient of the loss function with respect to the weights between the hidden layer and the input layer $w_{i,j}^{(h)}$, is the product of the activation of neuron j in the input layer $a_j^{(in)}$ and the error term of the hidden neuron $\delta_i^{(h)}$.

Steps 1-4 illustrate the backpropagation of errors from the output layer to the input layer, detailing how gradient calculations update the network's weights via gradient descent. Backpropagation is also illustrated in Figure 2.3, where the previously mentioned steps are presented in the context of the architecture of dense neural networks.

2.5.3 Recurrent neural networks

The equations and theory presented in the following subsection are from [27].

Recurrent Neural Networks (RNNs) represent a subfield of artificial neural networks that is specifically designed to process data in sequences or time series. In contrast to conventional dense neural networks, RNNs preserve a type of memory through their internal state to handle sequential data. A basic RNN comprises an input layer, one or more recurrent hidden layers, and an output layer. A simple RNN has an input layer that contains N input units which receive a sequence of vectors over time t. This could be defined as $(..., x_{t-1}, x_t, x_{t+1}, ...)$ where each x_t is a vector $(x_1, x_2, ..., x_N)$. These units are interconnected with the hidden units through a weight matrix WIH. The hidden layer has M hidden units, represented as $h_t = (h_1, h_2, ..., h_M)$, which are linked over time through recurrent connections. A schematic example of a simple RNN model is presented in Figure 2.4.

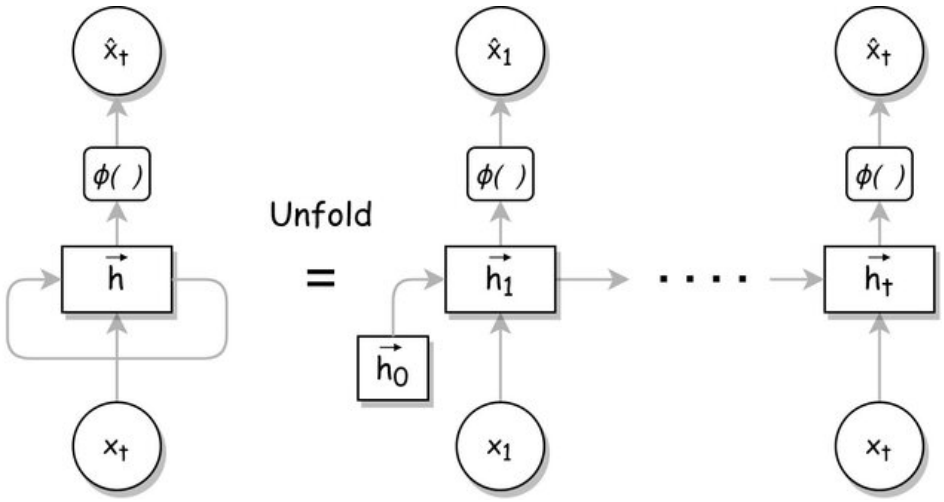

Figure 2.4: Unfolded single layer RNN. Made available via CC BY 4.0 license. Retrieved from: **https://www.researchgate.net/publication/361275024_Data-Driven_Denoising_of_Accelerometer_Signals/figures?lo=1**

The hidden layer h_t encapsulates the memory of the RNN and can be described as

$$\mathbf{h}_t = \mathbf{f}_H(\mathbf{WIH} \cdot \mathbf{x}_t + \mathbf{WHH}\mathbf{h}_{t-1} + \mathbf{b}_h), \tag{2.15}$$

where f_H represents a nonlinear activation function, b_h is the bias for the hidden units, and WHH is the weight matrix for the recurrent connections. The output y_t at each time step t is then computed from the hidden state h_t as

$$\mathbf{y}_t = \mathbf{f}_O(\mathbf{WHO}_t + \mathbf{b}_o), \tag{2.16}$$

where WHO is the weight matrix that connects the hidden layer to the output layer. At each time step, the current hidden state is used to predict the output layer based on the input vector. The hidden state contains information from past network states over multiple timesteps. This information is used to predict the future behavior of the network.

2.5.4 Long short-term memory

The equations and the theory of the preceding subsection are from [27].

Long short-term memory (LSTM) networks are an advanced form of RNN. The LSTM is built to address the limitations of basic RNNs, which can often suffer from vanishing and exploding gradients. These gradient problems can arise when computing the gradient of the loss function during the backpropagation phase. The core of the problem lies in the multiplicative nature of the gradients throughout the network's layers. If the weights in a network are overly small or large, the network's activations could return exceedingly small or large weights over subsequent layers.

The consequence of the vanishing gradient problem is that the weights do not receive adequate updates. On the other hand, the exploding gradient problem results in erratic weights that could deter the training process. LSTM networks can mitigate these issues through their architecture, which consists of multiple gates that regulate the flow of information. The structure of a basic LSTM network is shown in Figure 2.5. A fundamental part of an LSTM network is the LSTM cell which contains an input gate, a forget gate and an output gate. These components control the cell state, which acts as the long-term memory of the network since it contains relevant information from previous time steps. The hidden state functions as the short-term memory of the LSTM and represents the current output of the cell for each timestep.

Each gate in an LSTM cell has weights and biases that impact the cell state and the hidden state. The flow of information starts at the forget gate as it updates the long-term memory of the cell state. It's defined as

$$\mathbf{g}_t^f = \sigma(\mathbf{W}_{Ig^f} x_t + \mathbf{W}_{Hg^f} \mathbf{h}_{t-1} + \mathbf{W}_{g^c g^f} \mathbf{g}^c{}_{t-1} + \mathbf{b}_{g^f}), \tag{2.17}$$

where $\mathbf{W}_{Ig^f}$, $\mathbf{W}_{Hg^f}$ and $\mathbf{W}_{g^c g^f}$ correspond to the weight matrices from the input, previous hidden state and previous cell state with respect to the forget gate. $\mathbf{b}_{g^f}$ is the bias of the input gate. The input gate then determines the values from the input data to be preserved in the cell state. It can be described as

$$\mathbf{g}_t^i = \sigma(\mathbf{W}_{Ig^i} x_t + \mathbf{W}_{Hg^i} \mathbf{h}_{t-1} + \mathbf{W}_{g^c g^i} \mathbf{g}^c{}_{t-1} + \mathbf{b}_{g^i}) \tag{2.18}$$

$\mathbf{W}_{Ig^i}$, $\mathbf{W}_{Hg^i}$ and $\mathbf{W}_{g^c g^i}$ are the weight matrices between the input, the previous hidden state and the previous cell state to the input gate, respectively. $\mathbf{b}_{g^i}$ represents the bias associated with the input gate. The cell gate updates the cell state by integrating the new candidate values moderated by the input gate and the old cell state values influenced by the forget gate:

$$\mathbf{g}_t^c = \mathbf{g}_t^i \tanh(\mathbf{W}_{Ig^c} x_t + \mathbf{W}_{Hg^c} h_{t-1} + \mathbf{b}_{gc}) + \mathbf{g}_t^f \mathbf{g}_{t-1}^c \tag{2.19}$$

$\mathbf{W}_{Ig^c}$ and $\mathbf{W}_{Hg^c}$ are the weight matrices from the input and the previous hidden state to the cell state. The output gate determines which segments of the cell state will affect the hidden state of the LSTM for each timestep. It can be described as

$$\mathbf{g}_t^o = \sigma(\mathbf{W}_{Ig^o} x_t + \mathbf{W}_{Hg^o} \mathbf{h}_{t-1} + \mathbf{W}_{g^c g^o} \mathbf{g}^c{}_t + \mathbf{b}_{g^o}), \tag{2.20}$$

where $\mathbf{W}_{Ig^o}$, $\mathbf{W}_{Hg^o}$ and $\mathbf{W}_{g^c g^o}$ are the weight matrices linking the input, hidden state, and cell state to the output gate, respectively. The hidden state is derived by calculating the activation of the output gate

$$\mathbf{h}_t = \mathbf{g}_t^o \tanh(\mathbf{g}_t^c) \tag{2.21}$$

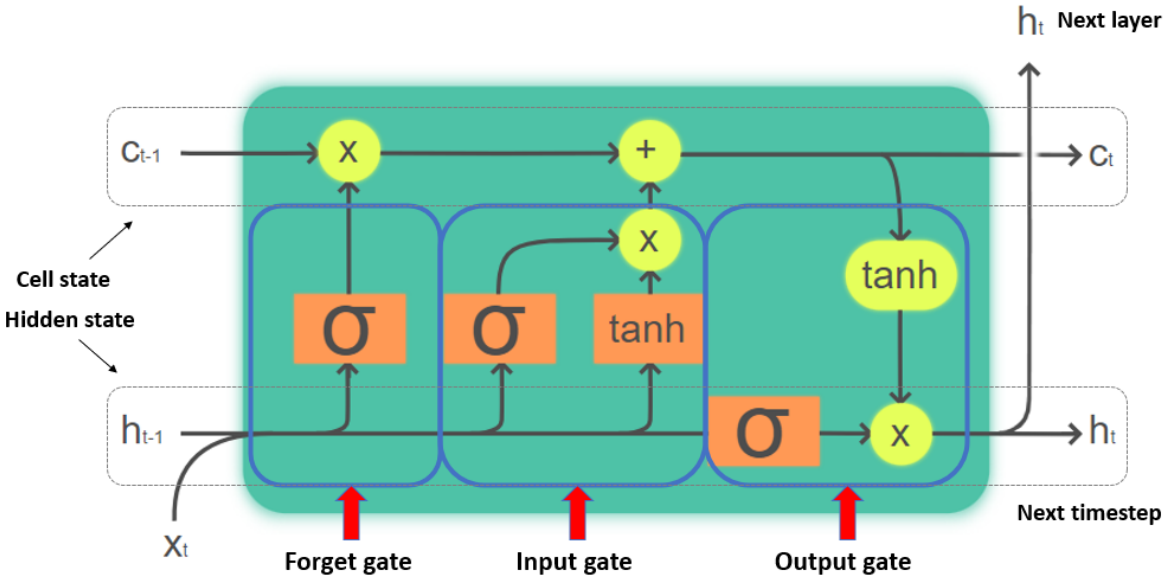

Figure 2.5: Sketch of a LSTM cell. The figure is available through the CC BY 4.0 license and was retrieved from: **https://github.com/guillaume-chevalier/Linear-Attention-Recurrent-Neural-Network/blob/master/inkscape_drawings/lstm-cell.svg**

2.6 Evaluation metrics

Evaluation metrics are used to assess the level of performance in classification and regression problems [28]. These metrics provide a measure through which the performance of the models is quantitatively evaluated. In that sense, a suitable selection of evaluation metrics will aid in obtaining optimal models. Optimal metric selection can depend on the target variable, as a regression problem might benefit from using other metrics than a classification problem. Since this thesis delves into a regression problem, metrics commonly used for continuous target values will be described in the following subsections.

2.6.1 Root mean square error

The root mean square error (RMSE) is the square root of the average square differences between the predicted values $\hat{y}_i$ and the actual values y_i. Equation (2.22) expresses the mathematical representation of RMSE:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{2.22}$$

By squaring the residuals before averaging the differences, the RMSE is sensitive to

larger residuals. Since the RMSE is consistent with the scale of the target variable, it provides an intuitive understanding of the magnitude of the error in the predictions. The consistency of the scale also means that the RMSE should not be compared between datasets with different scales [29].

2.6.2 Mean absolute error

The mean absolute error (MAE) is the average absolute difference between some predicted values $\hat{y}_i$ and some actual values y_i. Mathematically, MAE can be expressed by Equation (2.23):

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{2.23}$$

Implementing absolute values ensures that all errors are treated equally. Outliers are thus penalized to a lesser extent through the MAE formula compared to the RMSE. This linear nature allows MAE to serve as a robust tool for model evaluation [29].

2.6.3 Mean absolute percentage error

The mean absolute percentage error (MAPE) assesses prediction accuracy by expressing the prediction errors as percentages. It is useful for comparing model accuracy across datasets of different magnitudes. MAPE represents the mean of the absolute percentage differences between the predicted values $\hat{y}_i$ and the observed values y_i. The formula for computing MAPE is given by

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \tag{2.24}$$

n represents the total quantity of samples, $\hat{y}_i$ indicates the predicted value for the i^{th} sample, and y_i is the corresponding target value. By converting errors to percentages, MAPE provides a scale-independent accuracy measure that enables comparisons of forecast errors in different data sets or models. MAPE is less suitable for datasets that contain zero or near-zero values due to its mathematical nature [30].

Chapter 3

Methods

3.1 Data source

The data source for this thesis comes from Ruter. Ruter's data warehouse meets the needs of its associated stakeholders by making data available for use within various domains. Some of the measures and dimensions available in the data warehouse include passenger counts, punctuality measures, and vehicle travel speed. These, among other variables, are represented in various tables organized in a star schema system.

The data stored in the data warehouse comes from automatic data collection (ADC) systems installed on Ruter's operating transit vehicles. These ADC systems comprise multiple sensors and measuring equipment that create data of recorded events which are continuously fed to Ruter's data pipeline. The primary components of these ADC systems on Ruter's buses are automatic passenger counting (APC) systems and automatic vehicle location (AVL) systems. APC systems automatically count the number of boarding and alighting passengers at each stop through sensors installed in transit vehicles. AVL systems use real-time GPS data submitted by buses to provide information on traffic data. This enables transit companies to monitor vehicle speed, stop arrival times and BDT. The live feed of data helps Ruter monitor the demand and quality of their services.

The literature emphasizes the importance of evaluating the raw output of APC systems, as these systems can be prone to errors when counting passenger numbers. Pronello and Ruiz evaluated the accuracy of commercial APC systems under real-world conditions [31]. The results revealed that the accuracy of the commercial APC system for boarding and alighting passengers was generally between 50% and 65%, with the individual day accuracies varying and sometimes approaching zero due to counting errors.

3.1.1 Data source types

Ruter currently has two main variations of ADC systems installed on their buses which they refer to as TaaS and SIS. SIS is the older ADC system and is currently being phased out by TaaS on most of Ruter's operative bus lines. Despite this, we deemed the SIS platform data to be more applicable for estimating BDT in the scope of this thesis. The reason for this lies in the differences that these two ADC systems have in their estimation logic of arrival and departure times. SIS measures the arrival and departure times at the time the bus doors open and close, respectively. Instead of considering the opening and closing times at the bus doors, TaaS uses GPS and odometers to estimate when the bus arrives and departs from a stop. These differences in stop estimation logic suggest that data from these two ADC systems probably shouldn't be combined for analytical purposes. As the SIS logic is the closest match to the way BDT is defined in the literature, we decided to prioritize the data from SIS buses in this thesis.

3.2 Dataset

The data used in this work span from 02.01.2023 to 02.04.2023. We used observations at 15-minute intervals for every stop between 06:00 and 23:00 for all the days we included in our sample. Business hours outside the aforementioned time interval were excluded due to the lower passenger activity at these hours. A consequence of this reduced passenger activity is that the bus could end up driving past stops at these hours, as there may be no need to load or unload passengers. In these scenarios, BDT will naturally be equal to 0. Since we decided not to account for cases where the buses drove past stops without servicing passengers, substantial imputation would be required for the data in these low activity hours. Therefore, we chose to incorporate the operating hours that had a reduced need for imputation.

The data consisted exclusively of different stops on bus line 20 in the eastward direction, as seen in Figure 3.1. Line 20 was specifically chosen due to its high passenger activity. A highly frequented bus line will have fewer cases where the need to load or unload passengers is absent, thus providing a lower number of missing BDT instances. The vast majority of operative transit buses on line 20 were also logged on to the SIS platform when the data for this thesis were recorded, which was preferable for our case of modeling BDT due to the reasons described in Section 3.1.1. The extent of possible TaaS buses operating line 20 in the first quarter of 2023 is unknown. These cases were filtered out regardless during the manual screening process described in Section 3.2.2.

We decided to concentrate on only one bus route in a single direction since the time series approach made it difficult to model many stops simultaneously. When the data

were shaped as a time series with external predictors, subsetting the data into separate datasets for each stop became necessary. The rationale behind this is that the time axis is not equally spaced and consistent if data from several stops were mixed into one data set. Splitting data for multiple stops and multiple lines would also introduce additional dimensions to the data, which would have made it difficult to get a clear overview of the results. Four of the stops normally part of line 20 were excluded at the start due to significant instances of errors in passenger values at these stops.

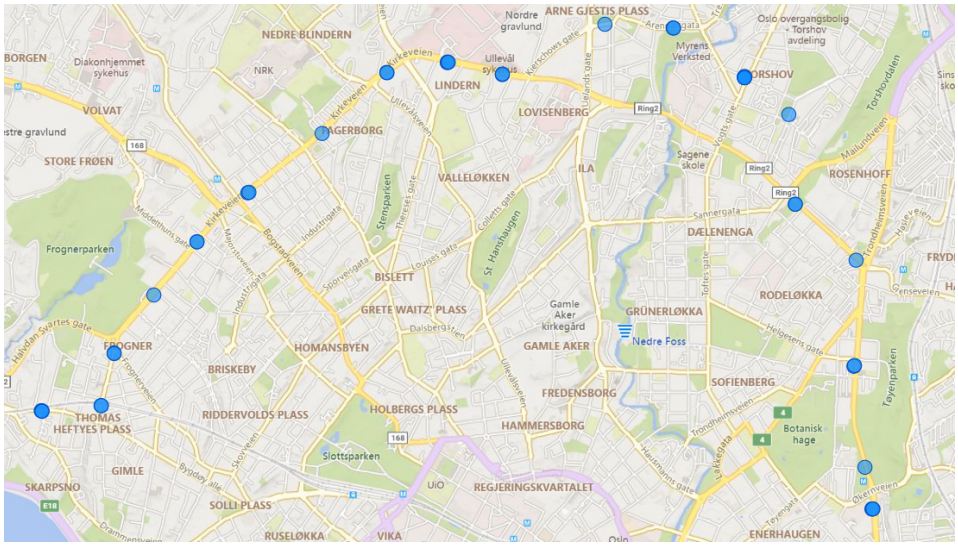

Figure 3.1: Map over line 20 with the stops that were used in this thesis. Created in Power BI through Ruter's GPS data.

3.2.1 Variables

The selected dataset contained variables that we considered helpful in predicting BDT. The dimensions and measures in the semantic layers were extensive, leading to a prioritization choice of the most important independent variables. In this sense, the variables associated with passenger activity (onboard, boarding, alighting, fill rate) were chosen because the existing literature identifies them as significant determinants of BDT. All temporal variables (working day, morning rush, afternoon rush) were incorporated into the dataset as dummy variables. The rush hours were divided into two variables due to the belief that commutes could affect different stops differently depending on the time of day. The working day variable was also included to investigate if there was a difference between BDT during the weekdays and weekends. The variables presented in Table 3.1 were used in every model we utilized in this thesis.

Variable	Description
Onboard	Number of passengers onboard the bus after passenger boarding and alighting has been completed.
Boarding	Number of passengers boarding at a given stop.
Alighting	Number of passengers alighting at a given stop.
Fill rate	The number of onboard passengers divided by the capacity of the bus.
Bus dwell time	Engineered feature. Derived from subtracting departure time from arrival time, returning a duration time in seconds. The target variable for this study.
Working day	Flag variable indicating whether a given observation occurred on a working day or not.
Morning rush	Flag variable indicating whether a given observation occurred during the morning rush hours, defined as the time interval from 07:00 to 09:00 on working days.
Afternoon rush	Flag variable indicating whether a given observation occurred during the afternoon rush hours, defined as the time interval from 15:00 to 17:00 on working days.

Table 3.1: The variables that were used in the datasets of this work.

3.3 Research design

The following subsections describe the overall research design implemented in this thesis, which is also illustrated in Figure 3.2.

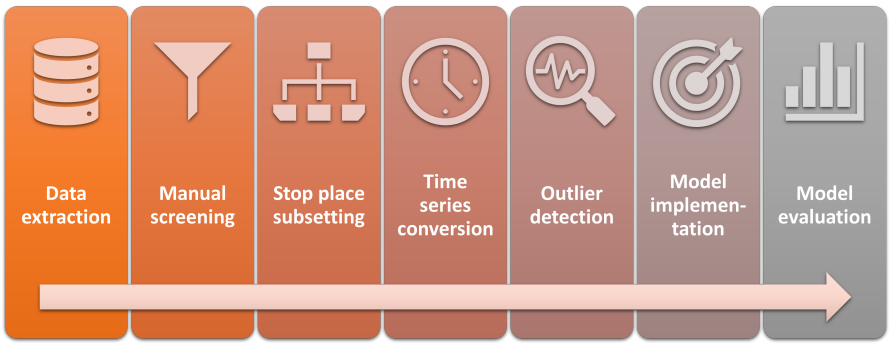

Figure 3.2: Illustration of thesis workflow.

3.3.1 Data extraction

The data was accessed through a semantic layer of Ruter's data warehouse. This semantic layer was organized as a data model in Power BI, which worked as a good solution for dynamic access to the data we needed for this study. After selecting the variables that we desired for BDT estimation, the data was stored locally.

3.3.2 Manual screening

Since ADC data is erroneous by nature, manual data screening was necessary to remove significant outliers before the modeling phase began. The screening process was performed by filtering the data through flag variables that were available in the semantic layer. These flag variables allowed us to exclude obvious outliers from the experiment data. All passenger variables were filtered to be in the range of 0 - 120. The lower bound of 0 was implemented because negative passenger values would sometimes occur in the raw data, which would not be possible in a real-life scenario. The upper bound of 120 was implemented due to the predefined maximum capacity of the operative buses that Ruter uses. As for the other variables in the dataset, the following filters were applied:

- **Arrival time uncertainty:** Flag variable indicating if there is uncertainty about the recorded arrival times at the stops. If the measured arrival time at the platform falls outside the measured arrival and time for the stops' predefined geofence, the row is flagged as uncertain. Data with uncertain arrival times were removed from the experiment data.
- **Departure time uncertainty:** Similar flag variable as the arrival time uncertainty, designed with the same logic. Data with uncertainties in departure time measurements were removed from the experiment data.
- **Journey assignment uncertainty:** Flag variable indicating if a vehicle has been logged onto another trip than the one it actually drove. Data flagged with a value of zero indicated journey assignment uncertainty and was subsequently removed from the experiment data.

3.3.3 Stop place subsetting

After manual screening was performed to account for obvious outliers, the data was divided into individual datasets based on the different instances of stops. This process made each dataset a continuous time series with BDT as the target variable. After subsetting the data, each dataset had a size of 6188 rows and 7 columns. This thesis uses the words data set and stop interchangeably to describe the BDT dynamics for a particular stop, since each stop was separated into its own data set.

The data needed to be divided into different subsets for several reasons. If the data had been sorted chronologically by stop order and time, the time series would not be continuous across the whole time axis. Time jumps back and forth would occur since a given trip $n + 1$ would start before a trip n had finished all its stops, resulting in a disjointed time axis. There were also significant differences between stops for BDT variance homogeneity. These large differences between stops made us divide the data

into different groups of stops. The authors of [5] also indicated that stop data should be separated when working with BDT estimation. Since we had data for 19 different stops, the stop place subsetting led to us having 19 individual datasets to work with for our work. Table 3.2 displays the stops with which we worked in this thesis. The stop index assigns the driving order of the stops during a typical bus ride. The stop index begins with 2 because the initial two stops of line 20 were excluded from this work. Similarly, the last two stops that normally are part of line 20 were also excluded from the data.

Stop index	Stop place
2	Olav Kyrres plass
3	Odins gate
4	Frogner plass
5	Vigelandsparken
6	Frogner stadion
7	Majorstuen
8	Marienlyst
9	Vestre Aker kirke
10	Ullevål sykehus
11	Fayes gate
12	Sagene
13	Arendalsgata
14	Torshov
15	Torshovparken
16	Københavngata
17	Carl Berners plass
18	Helgesens gate
19	Tøyenparken
20	Tøyen

Table 3.2: The modeled stops of our work with their associated stop index.

3.3.4 Time series conversion

The data extracted from Ruters' data warehouse did not natively have a time series structure. Conversion of the data to have a time series structure was therefore necessary, which first started with indexing the data chronologically. We opted not to use an axis of date and time in our work, contrary to the norm within time series modeling. This is because a traditional time series approach would have resulted in an irregular time series axis for our data. The irregularity would occur since the granularity of our high-frequency data would not allow for a traditional time axis due to the omitted nighttime values. We opted to use an axis of ascending integer values as a workaround for our irregular time axis, which is displayed in Figure 3.3. This gave the data sequential context while still effectively dealing with the natural gaps in the data.

As manual filters had been applied to the data initially, most of the time series datasets had some missing gaps in their respective time axis. The missing data in these gaps were imputed with forward fill, except for the observations between 06:00 and 08:00. The missing time gaps in these morning hours needed to be imputed with backward fill, since nighttime data was omitted for this thesis.

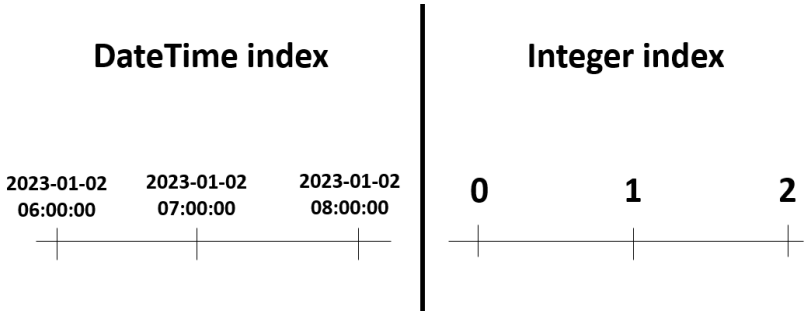

Figure 3.3: Arbitary values displaying the difference between a typical time series axis, and the sequential axis we used in our work.

3.3.5 Outlier detection

The isolation forest algorithm was implemented to detect outliers in each time series dataset. The algorithm was used primarily on the target variable, as erroneous passenger variables were handled by the manual screening step described in Section 3.2.2. Listing 3.1 shows how the Isolation Forest was applied to the individual datasets. We used linear interpolation to address these detected outliers. We decided to use imputation as an outlier strategy for multiple reasons. As sensor data can be unreliable, outliers are expected for data generated by ADC systems. There were also certain edge cases that we did not want to include in the model, in which BDT values might be artificially elevated. Examples of such edge cases could be cases where a lift operation was necessary to load or unload passengers with restricted mobility capabilities. We preferred to treat potential cases like this as outliers. Since we were dealing with time series data, we could not simply remove potential outliers as this would make our time axis irregular.

```
1 for df in dataframes:
2     isolation_forest = IsolationForest(n_estimators=100, ↩
          random_state=42)
3     df['Outlier'] = isolation_forest.fit_predict(df[['Bus ↩
          dwell time']])
4     df['Outlier'] = df['Outlier'] == -1
5     df.loc[df['Outlier'], 'Bus dwell time'] = np.nan
6     df['Bus dwell time'].interpolate(method='linear', ↩
          inplace=True)
```

Listing 3.1: Utilized code for detecting outliers.

3.3.6 Model implementation

The models we used in our work shared similarities in the implementation phase. Since the data for each stop was separated into different datasets, the model workflow started by iterating over the dataset folder to read all 19 datasets. After storing all individual datasets in a list, we split all individual datasets into training sets and test sets. The split between training and test data was 10 and 3 weeks for all datasets, respectively. The last common step for all model implementations included data standardization, as some features had different magnitudes for their measurement scales. *Scikit-learn's StandardScaler* was used for this purpose. We applied *StandardScaler* to all 19 datasets separately, where only predictors were scaled. Scaling the data would also help the LSTM converge faster.

SARIMAX

All functions related to the SARIMAX models were implemented using the *Pmdarima* library, designed to provide time series functionality in a similar interface to that of *scikit-learn*. We used the *Pipeline* method to streamline time series preprocessing and model selection. The pipeline encapsulates these two steps into a single object that can be used to fit the data and make predictions. *Pmdarima's auto.arima* was selected as the estimator in our pipeline. It simplified the model selection process by searching across a defined range of ARIMA model parameters and selecting the model that best suited the data according to a given information criterion. In our case, AIC was used as the information criterion for the models. By modeling the different stops in our data separately, unique SARIMAX models were utilized for each present dataset.

Since the dataset contained observations in 15-minute intervals from 06:00 to 23:00, the seasonal period was equal to 68. This is a substantially longer seasonal period than other typical time series problems, where a seasonal length longer than 24 is rarely seen. The complex and long seasonality in our case was difficult to handle for a standard SARIMAX model, which meant that we needed to apply some specific preprocessing steps to make *auto.arima* able to run without timing out. In this regard, the model parameter that specifies seasonality was set to false. This meant that the seasonal parameters (P, D, Q) were automatically set to zero for all models that were searched with *auto.arima*. As an alternative approach to preserve information about the seasonal patterns in the data, we used Fourier terms in our pipeline to account for the seasonality. Ten Fourier pairs were used in our *auto.arima* models, where these Fourier pairs were added as additional predictors for the models.

LSTM

As a preprocessing step before applying the LSTM models to the datasets, we transformed the datasets into sequences that served as input for the LSTM models. Each sequence contained sequence lengths of a specified number that allowed the LSTM models to use preceding values to predict future values. We ended up using a sequence length of 17 for our models after some experimentation. The sequence length was also used to prepare the time series in a suitable format for the LSTM models. This was necessary since LSTM models need 3D tensors to process data sequences over time.

```
model = Sequential([
    LSTM(80, activation='relu', return_sequences=True,
             input_shape(n_steps_in, n_features)),
    LSTM(40, activation='relu'),
    Dense(1)])
model.compile(optimizer=Adam(learning_rate=0.00009),
              loss='mean_squared_error')
model.fit(X_train_seq, y_train_seq, epochs=50, batch_size=68,
          validation_data=(X_test_seq, y_test_seq),
callbacks=[EarlyStopping(patience=7)])
```

Listing 3.2: Utilized LSTM model parameters.

To find the best setup for our models, we experimented with different numbers of neurons and layers. This process was more exploratory as we tried various combinations to see which one worked best with our data. We aimed to find a balance where the model was complex enough to capture the patterns in the data, yet not so complex that it became too focused on the training data. After experimenting with different combinations of layers and model parameters, we settled on using the model architecture depicted in Listing 3.2. This model setup was used in all our datasets. The initial layer of the model was configured with 80 neurons and employed the rectified linear unit (ReLU) activation function. It also returned full sequences to the subsequent LSTM layer, which ensured that the hidden state of all the time steps was passed on. The input shape of this layer was determined by the number of time steps and the number of features considered in each sequence. Following the first LSTM layer was a second LSTM layer comprising 40 neurons, in addition to the ReLU activation function. The final layer translates the feature representation learned from the preceding LSTM layers into a continuous prediction. The models were configured using the Adam optimizer with a learning rate of 0.00009. This learning rate seemed to have a decent trade-off between learning speed and model optimization. Since this work is a regression problem, MSE was used as a loss function. During training, the model was fed sequences of BDT data along with its corresponding predictors. *EarlyStopping* callback functioned as a

regularization mechanism since it stopped the training process if the validation loss did not improve over 7 consecutive epochs. This measure mitigated the risk of overfitting.

XGBoost

The XGBoost models were imported from the *xgboost* package in Python. Since we are dealing with a continuous target variable in this work, we utilized the *scikit-learn* wrapper *XGBRegressor* to make predictions. Unlike SARIMAX and LSTM, XGBoost could not take advantage of the sequential dependencies in the data. In this sense, lagged variables could have been added to give the XGBoost models some information about previous time steps. We decided not to implement lagged variables since we wanted to investigate whether the sequential models had an advantage since the data was modeled sequentially. To obtain optimal hyperparameters for our XGBoost models, we utilized *scitkit-learn's RandomizedSearchCV* module. Due to our approach of splitting the data into 19 datasets, *RandomizedSearchCV* became appealing because of its computational efficiency for high-dimensional data. The parameter distribution provided to *RandomizedSearchCV* can be viewed in Table 3.3.

Parameter	**Range**
n_estimators	100 - 500
learning_rate	0.01 - 0.05
subsample	0.3 - 0.7
max_depth	3 - 9
min_child_weight	1 - 6

Table 3.3: Parameter distribution provided to RandomizedSearchCV.

Table 3.4 displays the XGBoost parameters yielded by *RandomizedSearchCV*. These parameters were used by every XGBoost model that was applied to the individual datasets. By applying the same model parameters for every dataset, we believed that the feature importance output from XGBoost would be more representative at each stop. A possible drawback to this approach is that a given preset combination of parameters might not be the best suited to every stop, as some stops could have benefited from using other model parameters.

Table 3.4: XGBoost model parameters.

Parameter	Value
objective	reg:squarederror
colsample_bytree	0.7337
learning_rate	0.02345
subsample	0.3491
max_depth	3
min_child_weight	2
n_estimators	331
n_iter	25
cv	5
random_state	1

3.3.7 Model evaluation

The evaluation of the models in this thesis was done through MAPE, MAE, and RMSE for the predictions in the test datasets. These metrics were chosen for their ability to highlight forecast accuracy and error magnitude in time series problems. A comparative analysis was performed using the selected evaluation metrics to evaluate the performance of the SARIMAX, LSTM, and XGBoost models compared to a baseline model. This comparative analysis also allowed us to gauge how the level of performance varied between different stops. The models were also individually evaluated to gauge how the independent variables influenced BDT and how the individual data characteristics differed between the various stops.

3.4 Model selection

Since we decided to use a time series approach for this thesis, we needed to implement models suitable for sequential data. SARIMAX and LSTM models were selected in this regard, as they both excel at modeling time series data. By selecting these two specific models, we were able to assess the performance of a conventional statistical model in contrast to a recurrent neural network. SARIMAX was chosen as the main time series model in this thesis because we initially hypothesized that there would be an hourly seasonal pattern in the data. SARIMAX is also the default estimator used in *auto.arima*, which aligned well with our preference. Regarding our primary neural network model, we opted for LSTM for its ability to handle long-term dependencies in sequential data. This ability was desirable for our case, since we had a long seasonal period in the data.

The XGBoost was also utilized to investigate how a general-purpose decision tree model would fare against the aforementioned sequential models. By including XGBoost in our

thesis, we could investigate whether BDT should be modeled as a time series problem, or whether it should be estimated by models that assume i.i.d. data. Since we also wanted to investigate how the chosen independent variables affected BDT in this thesis, the XGBoost's feature importance also made it a desirable model choice. A baseline model was also implemented to provide some context to the performance levels of the main models in the thesis. The baseline model used the average BDT value in the training sets to make predictions.

3.5 Software specifications

Python was used as the main programming language for this thesis. The version specifications of the software and libraries used are highlighted in Table 3.5. More information on the use of AI tools in this thesis can be found in Appendix A.

Module	Version
Python	3.11.4
ChatGPT	4.0
Pmdarima	2.0.4
Keras	2.15.0
Scikit-learn	1.4.0
Power BI	2.128.1177.0
Matplotlib	3.7.2
NumPy	1.25.1

Table 3.5: The versions of the modules utilized in this project.

Chapter 4

Results

4.1 Data exploration

Figure 4.1 is a line chart that shows the average passenger values per stop, organized in ascending order by stop index. The chart shows that boardings and alightings grow and decline at a similar rate as BDT for most stops. This would indicate a relation between BDT and the number of people boarding and alighting the bus. Although Figure 4.1 does not show an obvious correspondence between onboard passengers and BDT, these two variables are still likely to have a relationship to some degree. Figure 4.1 also shows the large variability of BDT between different stops in our data.

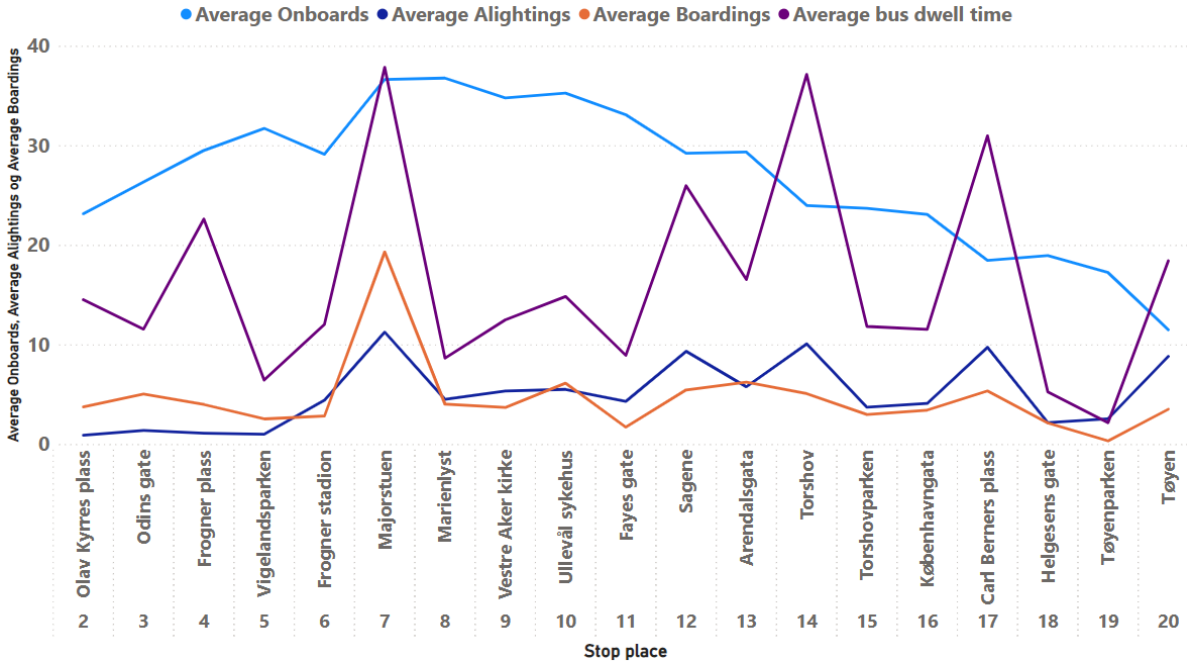

Figure 4.1: Average values of passenger variables and BDT per stop place.

The grouped bar chart in Figure 4.2 displays average BDT values for every temporal variable in our datasets. We can observe that the BDT values are largely similar within the different temporal categories. When comparing the morning rush and the afternoon rush, we can see that the BDT values are slightly higher during the morning rush for most of the stops in the data. The bar chart data suggest little to no difference between BDT in the afternoon rush and hours outside the defined rush hours. Similarly, the differences between BDT during the working days and the weekends appear to be minimal.

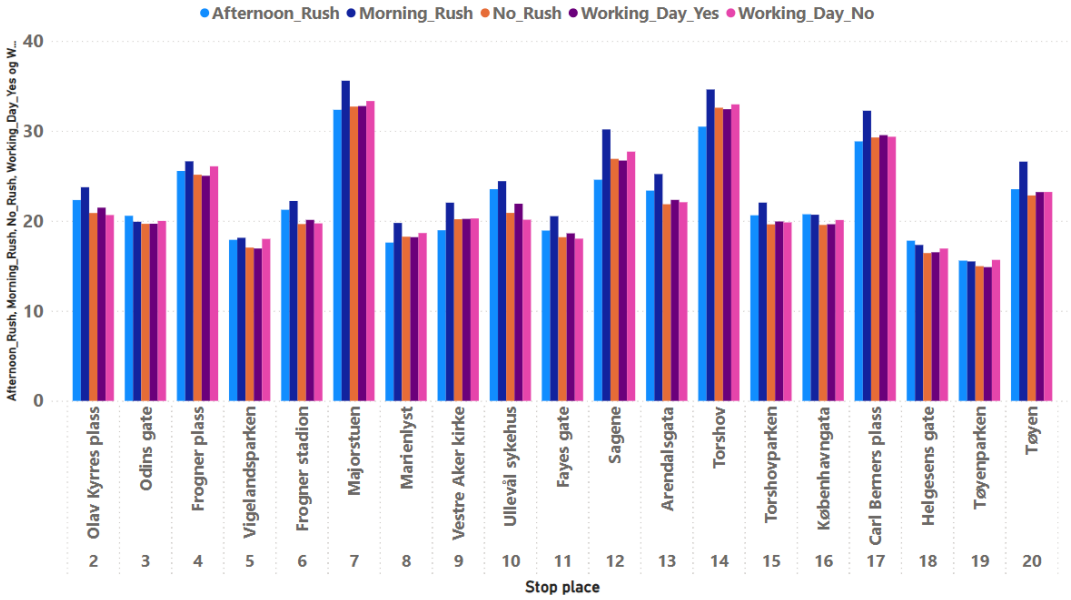

Figure 4.2: Average values of BDT for every time-related category per stop place.

The ACF and PACF in Figures B.1 and B.2 do not indicate systematic patterns in the autocorrelation values. ACF and PACF values fluctuate randomly within the significance bounds for most stops, suggesting a lack of serial correlation between different lags. This suggests that when examining BDT as a time series, previous values do not have a significant relationship with future values within the observed time lags. In addition, spikes that appear outside the significance bounds rarely exceed magnitudes of 0.05. Relative to the significance intervals between -0.025 and 0.025, we can note that any autocorrelation patterns that may be present are weak.

The correlation matrix in Figure 4.3 was made before the data was subsetted, which means that the values in the correlation matrix show the average correlation between variables across all stops. This implies that there might be some local variations in the different stops that are not explicitly shown in Figure 4.3. We can observe a perfect correlation between the number of onboard passengers and the bus fill rate. This is expected as these two variables essentially display the same data. We included both of these variables in our work as we expected that not all buses operating on line 20 would have the same size, which in those cases we could have expected to see a larger difference

between fill rate and the number of onboard passengers. However, the correlation matrix indicates that the buses in our datasets had largely the same capacities.

Notable correlation values of 0.47 and 0.46 between BDT and the number of passengers boarding and alighting can be observed. With BDT also having a correlation value of 0.21 with fill rate and onboard passengers, we can infer that passenger variables have the highest correlation with BDT in our dataset. This can be attributed to increased passenger numbers leading to more people getting on and off the bus, which in turn affects the time the bus remains stationary at a stop. Except for the correlation between onboard passengers and morning rush, the temporal variables included in the dataset exhibited a low correlation with BDT and passenger variables. This outcome could be partially attributed to the temporal variables being encoded as dummy variables. Since dummy variables are binary, they have restricted variability in their possible values. This could lead to lower correlation values since the variation range within the variable is limited. Regardless of this small uncertainty that might be present for the correlation values of the temporal variables, Figure 4.3 still indicates that the variables related to passengers impact BDT more than the other variables in our dataset.

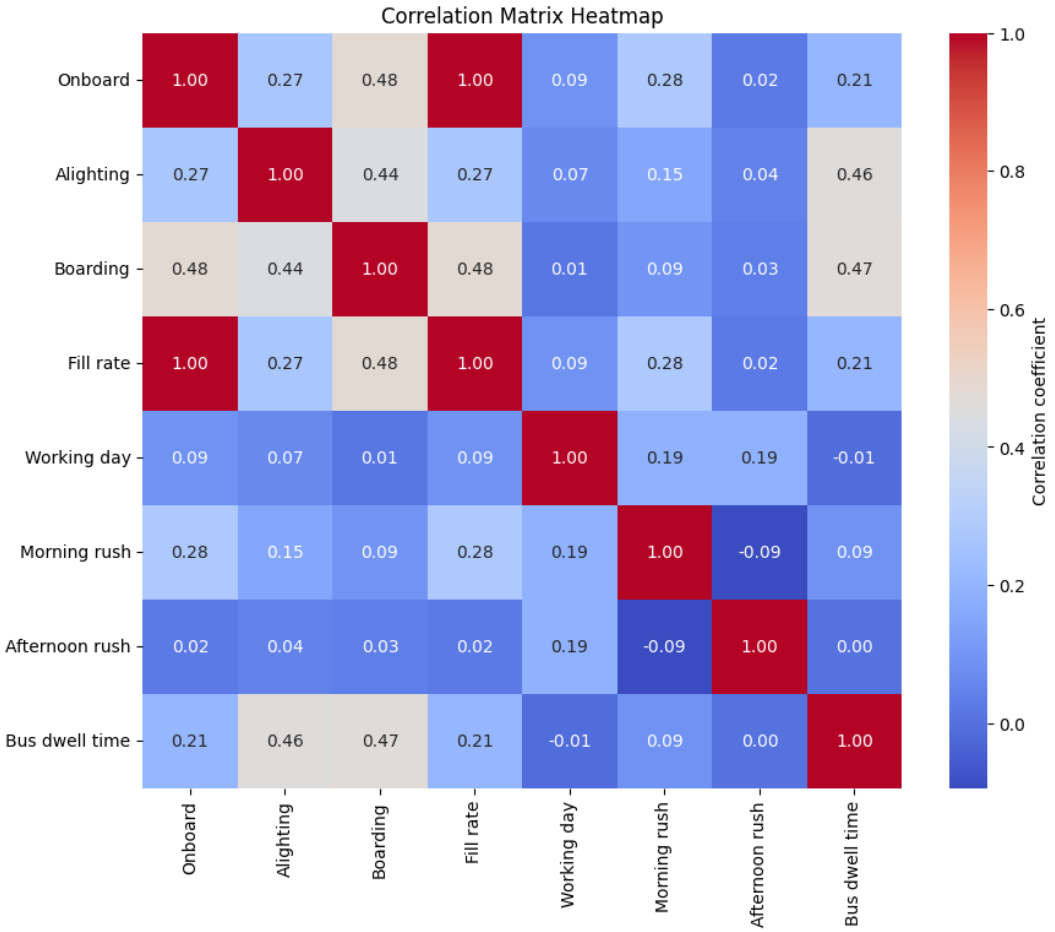

Figure 4.3: Correlation matrix for the variables used in this work

Figure 4.4 illustrates the outliers detected after the Isolation Forest algorithm was applied iteratively to each dataset. The percentage of outliers ranged from 2.96% to 6.03% for the different stops. This variation in the outlier magnitude could indicate differences in BDT dynamics between different stops. Since we decided to impute the BDT outliers instead of ignoring them, there was a small variation in the degree of imputation between the datasets, as seen in Figure 4.4. The predominance of nonoutlier data suggests that BDT data follow expected patterns in most cases.

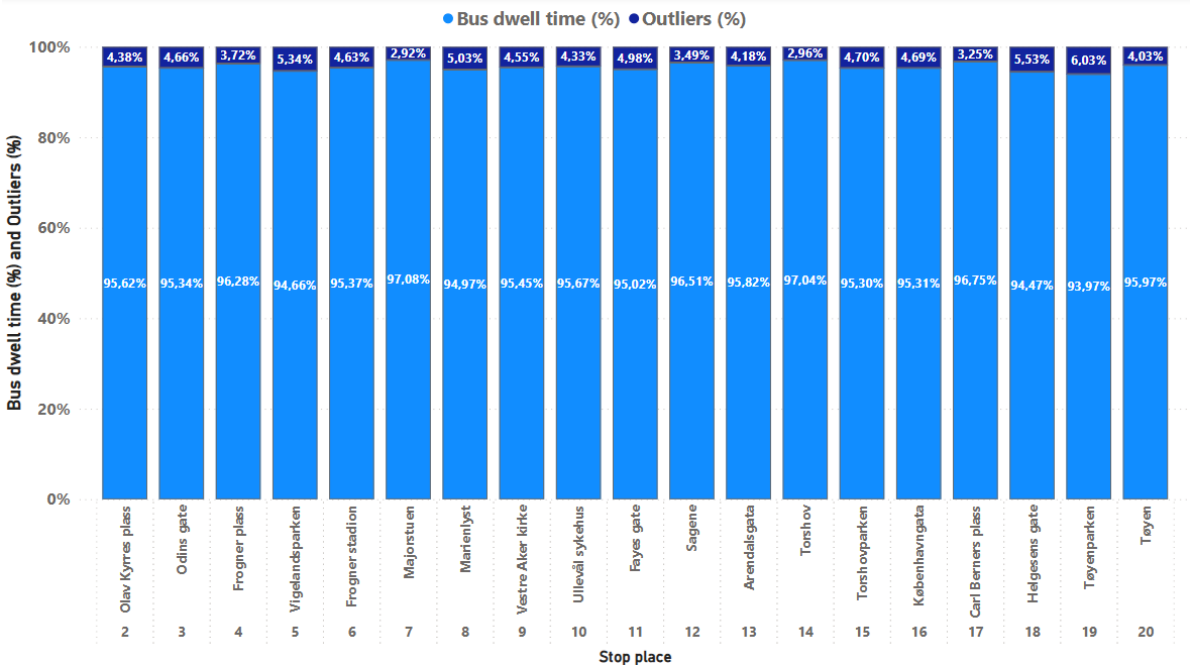

Figure 4.4: Detected outliers as a share of BDT data per stop place.

4.2 SARIMAX

Table 4.1 shows the variable coefficients and data characteristics extracted using the *summary*-function provided from *auto.arima*. The coefficients in Table 4.1 indicate the impact of each variable on BDT. We can observe mostly positive coefficients for the alightings and boardings variables. This could indicate that an increase in these variables is associated with prolonged BDT, which also coincides with the findings of Figure 4.3. The coefficient values for the onboard variable are predominantly negative, which may imply that higher occupancy levels could accelerate the boarding process and lead to reduced BDT. The magnitudes of the coefficients for the temporal variables appear to be less influential in predicting BDT than the passenger variables. This is reflected in the magnitudes of the coefficients present in Table 4.1.

The p-values derived from the hypothesis tests for heteroskedasticity and autocorrelation add additional information about the characteristics of the datasets. Fayes gate and

Sagene appear to have non-constant variance in their residuals, as their heteroskedasticity test probabilities are close to zero. This can lead to inefficiencies in the model estimators and biased standard errors. There seems to be no substantial heteroskedasticity in the data beyond Sagene and Fayes Gate. The column showcasing the test probability for autocorrelation at each stop implies that the models have captured most of the temporal dependencies within the data. Some stops have values close to the commonly used significance level of 0.05, implying that we cannot safely say that the models have captured all the temporal dependencies in these datasets.

The rightmost column of Table 4.1 denotes the most optimal combination of parameters discovered by *auto.arima* for every dataset. After a stepwise search through various parameter combinations, a model setup of SARIMAX(5,1,0)(0,0,0)0 had the lowest AIC for 14 of the 19 datasets present. This suggests an autoregressive pattern in the BDT data at most stops, where knowledge of previous values will aid in forecasting future values. The differencing term that appeared in most model configurations suggests some slight trends in the data that could have been eliminated to achieve stationarity.

Table 4.1: SARIMAX coefficients and data characteristics per stop place. Variable names are abbreviated. P(H) and P(A) indicate the p-values from hypothesis tests of heteroscedacity and autocorrelation, respectively.

Stop place	Alight.	Onboard.	Board.	Fill.	A.Rush	M.Rush	W.Day	P(H)	P(A)	Model params.
Arendalsgata	0.94	−6.87	1.44	7.54	0.21	0.16	−0.53	0.3	0.06	(5,1,0)(0,0,0)0
Carl Berners plass	1.83	−2.30	1.50	2.49	0.12	−0.03	0.05	0.4	0.15	(5,1,0)(0,0,0)0
Fayes gate	0.96	−9.90	0.88	10.42	0.04	0.06	−0.01	0.01	0.16	(5,1,0)(0,0,0)0
Frogner plass	0.42	−4.20	0.96	3.69	0.47	0.02	−0.23	0.5	0.18	(5,1,0)(0,0,0)0
Frogner stadion	1.12	−5.13	0.99	5.57	−0.06	−0.12	−0.24	0.7	0.17	(5,1,0)(0,0,0)0
Helgesens gate	0.61	−3.46	1.12	3.50	0.08	0.07	−0.29	0.9	0.99	(5,1,1)(0,0,0)0
Københavngata	0.70	−1.14	0.82	1.58	0.09	0.15	−0.38	0.9	0.49	(5,1,0)(0,0,0)0
Majorstuen	1.68	−29.33	2.80	29.46	−0.07	−0.05	−0.59	0.6	0.12	(4,1,1)(0,0,0)0
Marienlyst	1.13	−9.12	1.35	9.51	−0.09	−0.04	−0.43	0.4	0.94	(5,1,0)(0,0,0)0
Odins gate	−0.09	−5.65	0.99	6.35	−0.25	−0.13	−0.48	0.6	0.08	(5,1,0)(0,0,0)0
Olav Kyrres plass	0.27	−14.61	1.34	14.99	0.33	0.05	0.05	0.5	0.99	(3,0,0)(0,0,0)0
Sagene	1.42	−19.86	1.56	19.61	0.12	−0.47	−0.62	0.01	0.98	(1,0,0)(0,0,0)0
Torshov	1.41	3.31	1.50	−3.26	0.02	−0.49	−0.37	0.4	0.18	(5,1,0)(0,0,0)0
Torshovparken	1.15	−6.94	1.18	7.04	0.28	0.01	−0.31	0.7	0.13	(5,1,0)(0,0,0)0
Tøyen	1.27	−4.23	1.82	4.37	0.46	−0.02	0.02	0.7	0.19	(5,1,0)(0,0,0)0
Tøyenparken	0.78	−1.18	0.30	1.43	0.09	0.00	−0.36	0.6	0.17	(5,1,0)(0,0,0)0
Ullevål sykehus	1.56	−12.47	1.84	12.82	0.26	0.06	0.53	0.5	0.08	(5,1,0)(0,0,0)0
Vestre Aker kirke	0.79	−6.07	1.12	6.63	−0.04	−0.22	0.03	0.3	0.15	(5,1,0)(0,0,0)0
Vigelandsparken	0.50	−6.80	1.32	7.22	−0.02	−0.07	−0.29	0.1	0.94	(4,1,1)(0,0,0)0

4.3 LSTM

We can visually gauge the performance of the LSTM models in Figure 4.5, which shows the standardized values for loss and validation loss at the training phase of each stop. The distinctions between loss and validation loss reflect the model's ability to generalize beyond unseen data. Most stops exhibit a declining trend in both loss and validation loss

as the number of epochs increases, indicating that these specific models are learning and improving their prediction capabilities over time. Fluctuations for loss and validation loss curves are not volatile across epochs for most stops. This smooth loss decrease indicates an appropriately chosen learning rate for the models. The shape of the graphs in the line charts suggests that the learning dynamics vary across different stops. At stops like Frogner plass and Torshov, the validation loss decreases negligibly across epochs. This implies that the models did not adequately learn the dynamics in the data at these stops. A significant portion of the stops have a sharp decrease in validation loss between epoch five and epoch ten, which may imply that the model setup is suitable for capturing the underlying patterns in the data at these locations. It is also interesting to note that certain stops exhibit a validation loss that is lower than the training loss. This could indicate that these test datasets are less complex or exhibit less variability than the training datasets.

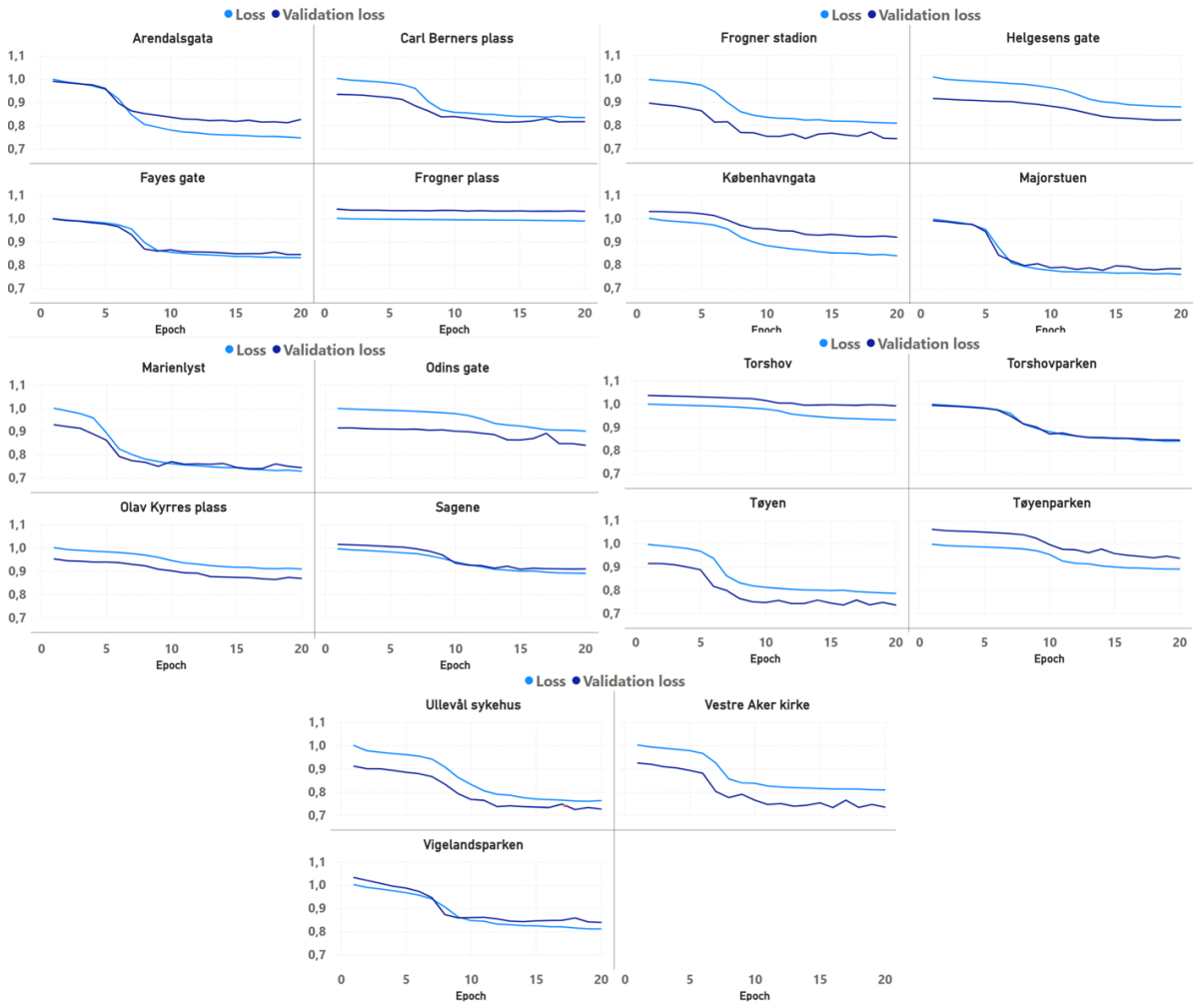

Figure 4.5: Loss and validation loss per epoch at stop places.

4.4 XGBoost

Figure 4.6 shows feature importance per stop place for the XGBoost models utilized in our work. The line chart allows us to gauge the usefulness of our variables for the different stops in the data. We can infer that the passenger variables (boardings, alightings, onboards, fill rate) appear to be more helpful in predicting BDT than the temporal variables (working day, rush hours). The XGBoost models have contrasting variability of feature importance scores across different stops. This variability underlines the influence of local conditions at each stop. Figure 4.6 also suggests that the XGBoost models can handle local dynamics at individual bus stops, as the line plots dynamically change across the axis of stops. This adaptability could be beneficial for the deployment of predictive models in real-world scenarios where each stop may present unique characteristics.

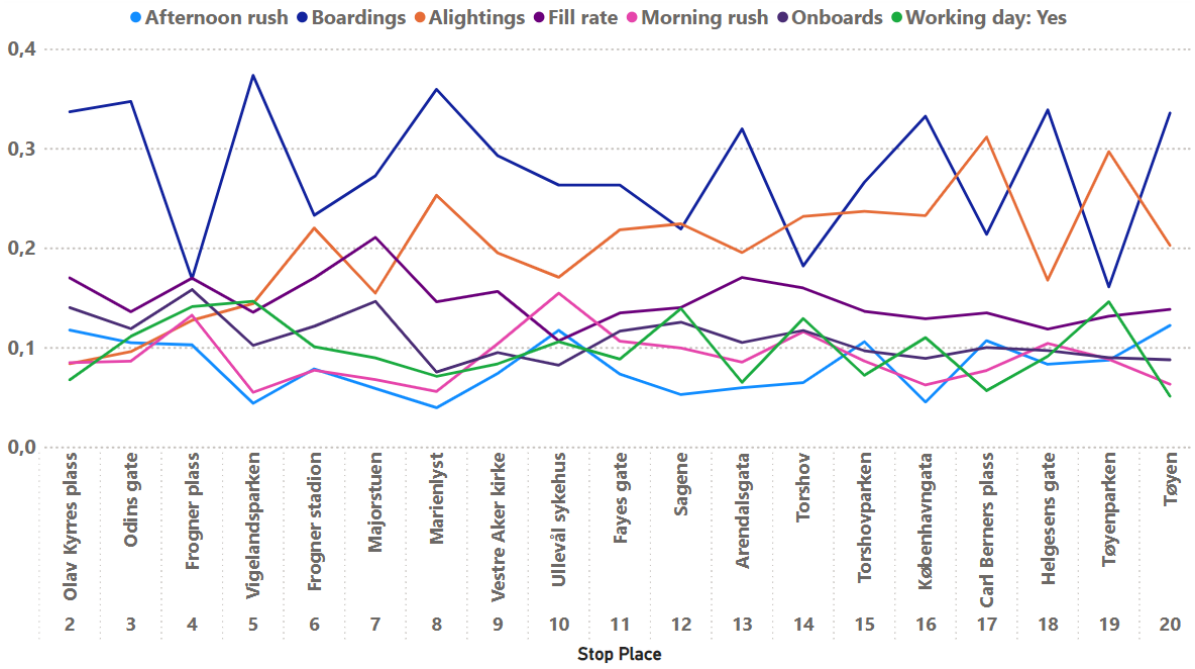

Figure 4.6: XGBoost feature importance per stop place.

4.5 Model performance

The RMSE scores in Figure 4.7 show similar results for LSTM and XGBoost at all stops. Table 4.2 also shows similar RMSE scores for the two models when presented as a cumulative sum across all stops, with XGBoost exhibiting slightly lower overall error than LSTM. The SARIMAX models outperformed the baseline for all stops except Odins gate and Frogner plass. Although the SARIMAX models matched the RMSE scores of LSTM and XGBoost at some stops, we can also observe that SARIMAX has a higher error score in Table 4.2 compared to LSTM and XGBoost. The cumulative RMSE score

for SARIMAX is somewhat skewed by a suboptimal performance at Odins gate, as seen in Figure 4.7. This score also indicates that SARIMAX has room for improvement in this case, as its total RMSE across all stops is only 0.58 lower than the baseline score.

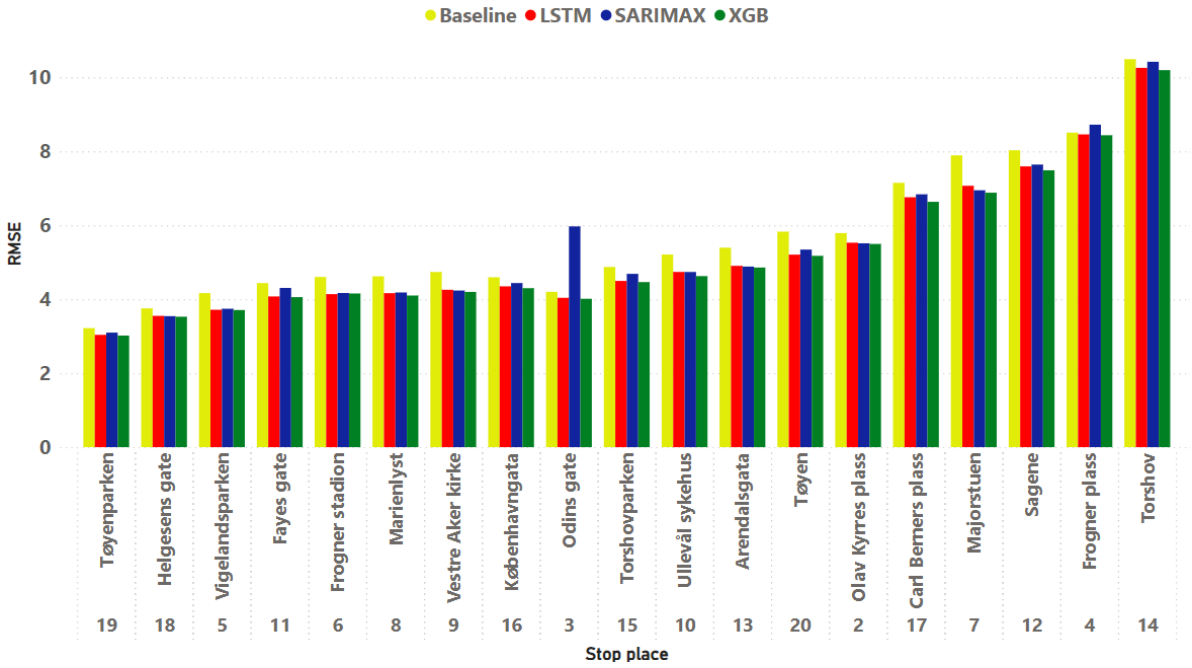

Figure 4.7: RMSE score for every model at all stop places

Table 4.2: Cummulated sum of evaluation metrics for all stop places.

Evaluation Metric	Baseline	SARIMAX	LSTM	XGB
RMSE	107.50	106.92	100.34	99.33
MAE	86.59	84.10	79.82	78.85
MAPE	4.06	3.88	3.73	3.67

The MAE scores in Figure 4.8 further support the conclusion that LSTM and XGBoost offer more precise predictions of BDT. All figures in this section were sorted ascendingly by metric score per stop, which yielded a nearly identical sorting order for Figure 4.7 and Figure 4.8. This helps us to establish which of the stops the models struggled the most with predicting BDT. The x-axis of the figures in this section also contains stop index identifiers, which show the stop order for a standard journey on line 20. By comparing the metric score with the stop index number, we can assess if there is a relationship between these two values and BDT. In Figure 4.7 and Figure 4.8, the models had the lowest error on Tøyenparken and Helgens gate. Their respective stop indices reveal that Tøyenparken and Helgesens gate appear later on the journey of line 20, indicating that a stop's geographical location could be an important factor in BDT estimation. Apart from these two instances, no other notable patterns emerge in the data when comparing stop indices with metric scores.

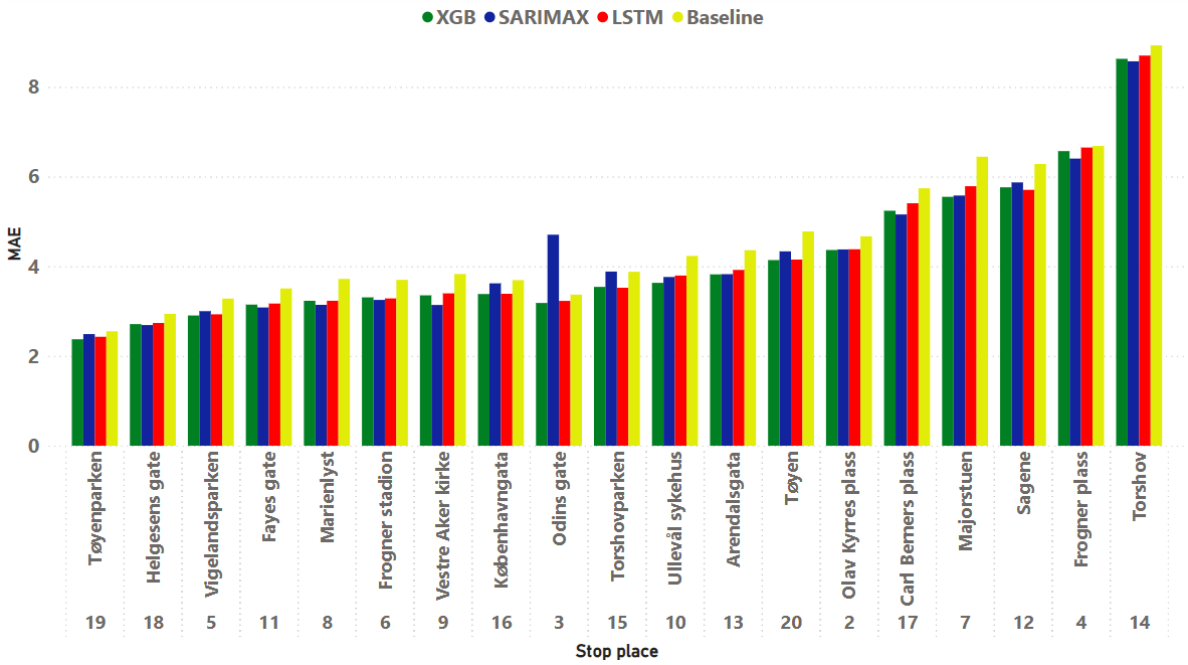

Figure 4.8: MAE score for every model at all stop places.

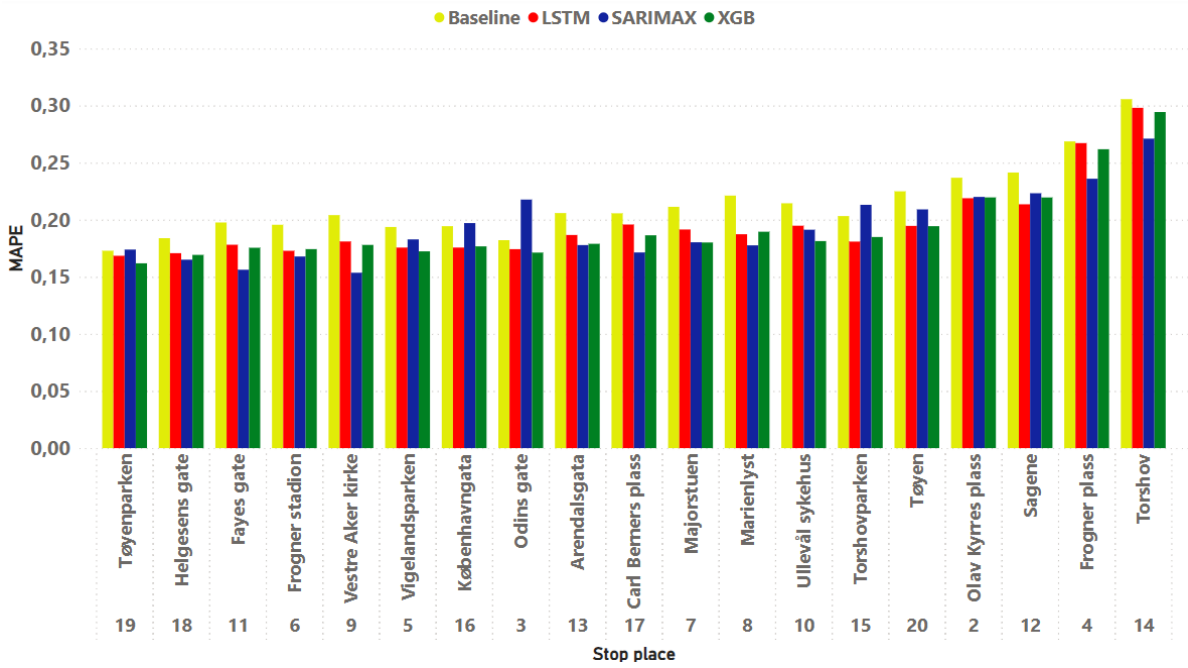

Figure 4.9: MAPE score for every model at all stop places.

Figure 4.9 offers a perspective on the relative prediction errors through MAPE. Although Table 4.2 showed a higher cumulative MAPE score at all stops for SARIMAX, we can observe that SARIMAX had the lowest MAPE score at nine of the stops in the data. This could indicate that SARIMAX was less conservative in its predictions when compared to the MAPE of the other models. The varying performance of the models at different stops suggests that a one-size-fits-all approach may not be sufficient for BDT

prediction across different stops.

Chapter 5

Discussion

5.1 Model differences

The three predictive models utilized in this thesis have demonstrated a contrasting suitability for BDT prediction. LSTM and XGBoost exhibited lower prediction errors than SARIMAX, as seen in Table 4.2 and in the figures in Section 4.5. The different complexities of the models could explain this outcome, as LSTM and XGBoost are nonlinear models. In contrast, SARIMAX is a linear model that assumes linear relationships between predictors [32]. The authors of [33] mentioned that BDT has a nonlinear residual part, which implies that SARIMAX's linear nature might not be sufficient to estimate BDT properly. [5] is important comparison material for our work, as it is the related work that is most similar to this thesis through its utilization of a univariate time series approach. The average MAPE for the best model in [5] was 12.8% in non-central business districts, and 32.7% for stops in central business districts. Seeing that our MAPE results ranged from 16% - 29% for our best model, the results in our work appear to be similar to comparable results from the literature.

LSTM and XGBoost also handled the varying BDT dynamics at different stops by delivering consistent performance across all datasets. Since there are many stops in a transit network, a sufficient BDT estimation model must be able to adapt to the varying dynamics at different stops. In terms of practicality, XGBoost had some advantages over the other models. It had a shorter training time than the LSTM and a substantially shorter training time than SARIMAX. The respective training durations for the models were 28 minutes for XGBoost, 39 minutes for LSTM and 135 minutes for SARIMAX. The slow training time for the SARIMAX models is probably due to the long seasonal period of our data, a drawback of the SARIMAX model which has been previously discussed in [34]. The workaround we used for the seasonality in the SARIMAX models

was to utilize Fourier pairs as additional predictors, which did not seem to give it an advantage in predictive capability compared to the other models. Differences in training time might also suggest that XGBoost is more scalable for larger datasets. LSTM could have benefited from using more training data, since all datasets individually had 6188 rows of data due to the stop place subsetting described in Section 3.3.3. Through its built-in feature importance, XGBoost also had an excellent framework for discerning which variables impacted the BDT predictions the most. SARIMAX were also helpful in terms of model interpretability through the coefficients provided by *auto.arima*. Due to the use of a stacked model architecture, extracting interpretability from the LSTM models proved to be challenging.

5.2 Variable differences

Our findings indicated that passenger variables impacted BDT predictions more than temporal variables. This makes sense since the number of boarding and alighting passengers is directly linked to the operational aspects that occur while a bus is stationary at a stop. The number of alighting and boarding passengers were the most influential predictors for all but five stops, as per Figure 4.6. Since the stops in Figure 4.6 were sorted chronologically, the lines in the chart effectively form a time series of how the importance of a variable evolves between stops. In this regard, we can observe a slight upward trend for the predictive influence of alighting passengers on BDT as a bus progresses through a sequence of stops. This indicates that alighting passengers may be a more influential predictor of BDT in the later stops of a bus trip sequence. Assuming the average number of stops a passenger remains on the bus is more than two stops, it makes sense that alighting passengers have a reduced predictive influence on BDT in the earlier stages of a bus ride. The negative coefficient values in Table 4.1 for the number of passengers onboard suggest that there could be a nonlinear relationship between the number of passengers already on the bus and BDT. This could indicate that BDT cases with more passengers onboard might experience faster passenger service, as passengers may board and alight faster when the bus is crowded to avoid delays.

Figure 4.6 also indicates that the number of boarding passengers is the most influential determinant of BDT in our dataset, as it has the highest feature importance at all but two stops. By cross-referencing stops with high feature importance scores for boarding passengers with passenger counts from the same stops, we cannot safely say that boarding is a more important BDT determinant at stops with more boarding passengers. This might imply that additional variables beyond the ones we chose in this work influence the BDT dynamics at certain stops. Examples of such underlying variables could be the service frequency at stops, or that passenger behavior differs from stop to stop due to

varying passenger demographics between stops. The temporal dummy variables, such as rush hours and working days, appeared less significant in predicting BDT. We can infer this from Figure 4.6. This indication is also shown indirectly in 4.2, as there seem to be small differences between BDT for the different temporal variables. There are also some local variations between the stops for the temporal variables, which could reveal something about the area surrounding a stop. For example, Ullevål sykehus has morning rush as its third most important variable as per Figure 4.6. This indicates that the healthcare workers who commute to the hospital near the Ullevål sykehus stop have a strong impact on BDT in the morning hours here. Another example is Tøyenparken, where working day is the third most important feature. This could indicate that this is a stop with significant differences in ridership dynamics between weekends and weekdays.

5.3 Stop place differences

Chapter 4 highlighted the differences in prediction error for the models in this work. To explore these differences, we looked at the two stops with the highest and lowest errors across the selected evaluation metrics. This was Torshov and Tøyenparken, respectively. After analyzing the thesis results with map illustrations of the stops, certain differences emerged between the stops that could explain why some stops were harder to predict for the models than others.

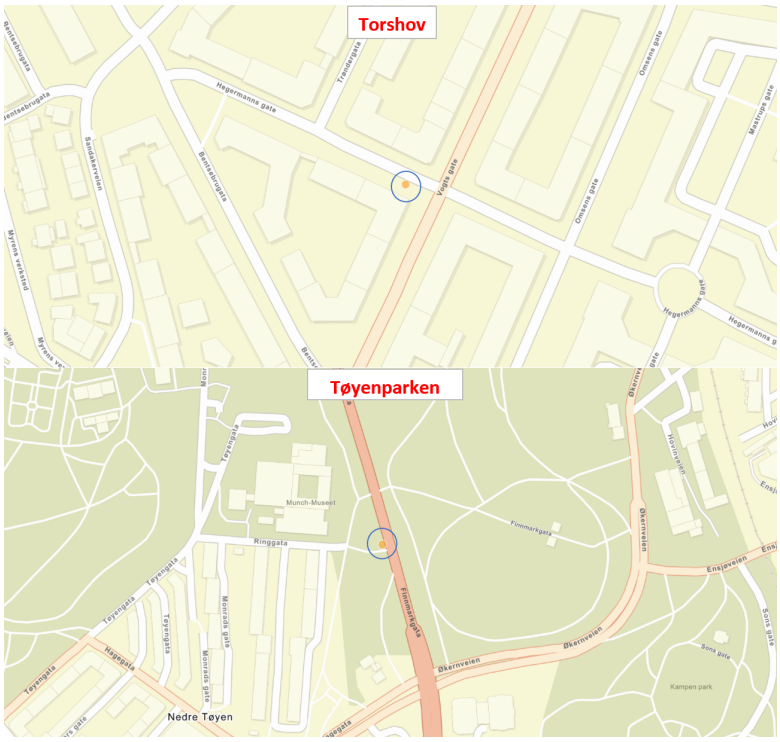

Figure 5.1: Map photos of the stops Torshov and Tøyenparken, marked with blue circles. The images were designed in ArcGIS for Power BI, using data from Ruter.

The surrounding infrastructure at a stop appears to be important in predicting BDT. After examining the Torshov stop closer, we found out that the stop is located right next to an intersection. Traffic signals at this intersection could thus prolong the BDT at the Torshov stop in certain cases. This is evident as the bus might need to wait for a green light before resuming its journey, thereby prolonging waiting time at the stop. This introduces an additional layer of complexity to the BDT patterns occurring at Torshov, which could be challenging for prediction models that typically assume consistent behavior for present variables. Contrary to the situation at the Torshov stop, the stop at Tøyenparken is positioned far away from intersections and traffic lights. We believe that the environment around Tøyenparken could lead to more stable BDT patterns, which would help achieve lower prediction errors. The impact of stop location is also highlighted in the literature, where the BDT at stops near intersections was 4.2 to 5.0 s slower than the BDT at stops on the far side of intersections [35].

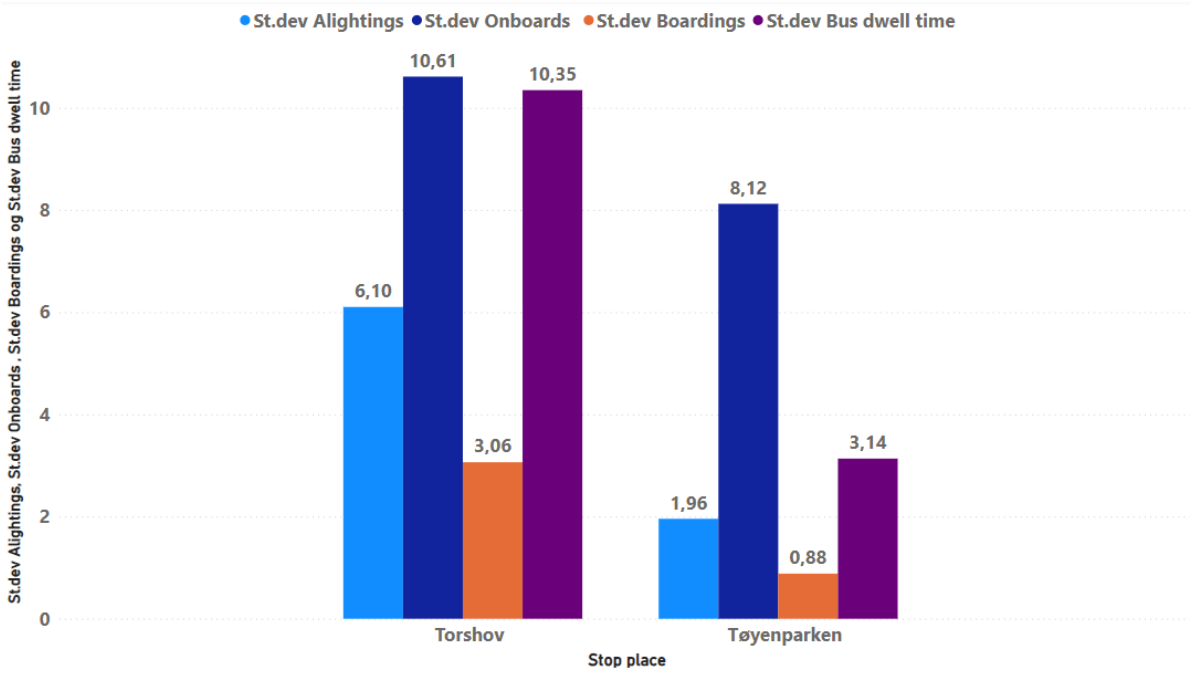

Figure 5.2: Comparions between Torshov and Tøyenparken of standard deviations per category.

Figure 5.2 shows the standard deviation for selected passenger variables and BDT in Torshov and Tøyenparken. We can observe that the variability of BDT is significantly higher in Torshov than in Tøyenparken, further supporting the previously stated hypothesis that the surrounding infrastructure is an important factor for predicting BDT. The variability of BDT can also be accounted for by the specific passenger dynamics at the respective stops, as busier stops tend to have higher average BDT than less trafficked stops. This difference in BDT between stops can also be due to the location of the stop. The Tøyenparken stop has fewer buildings in its local surroundings than the Tor-

shov stop, which has more apartment buildings nearby. These factors show that stops can have individual geographical and infrastructural characteristics in their surrounding area, resulting in passenger dynamics that could be unique from stop to stop.

The impact of BDT variations between districts was also discussed in [5]. The authors stated that the average BDT was relatively longer in the afternoon rush compared to the morning rush in central business districts (CBD), while the BDT at non-CBD stops did not differ during the times of the day. Although we did not explicitly categorize stops as central and non-central in our work, we can still safely say that the findings in [5] did not match our findings for the temporal characteristics of stops. This can be observed in Figure 4.2, where the average BDT is higher during the morning rush hours than during other periods of the day. The discrepancies between our results and those reported in the literature are consistent with observations from the related work of this thesis, where we noted that BDT trends vary significantly between different cities. The authors of [5] also implied that the variation in BDT at CBD stops could be due to the variability of passenger variables, which is something we highlighted in the previous paragraph and in Figure 5.2.

5.4 Resarch considerations

The decision to model BDT estimation as a time series with external predictors was a central part of this thesis. This decision was based on the assumption that BDT had temporal characteristics which could be utilized through time series analysis. In this regard, we needed to implement models suitable for time series data such as LSTM and SARIMAX. Contrary to models that assume i.i.d. data, LSTM and SARIMAX required extra preprocessing steps that added additional levels of complexity to the implementation process. The performance of XGBoost in our work challenged the premise that BDT should be modeled as a time series problem. Although XGBoost does not account for temporal dependencies in the data, it still slightly outperformed LSTM and SARIMAX on the time series data. This could indicate that methods beyond time series analysis could be most appropriate for BDT estimation since temporal data modeling adds extra complexity to the preprocessing phase. The low degree of autocorrelation at most stops indicates a process closer to white noise, where each observation is predominantly a random fluctuation that is independent of past values. This is another characteristic that questions the effectiveness of modeling BDT data as a time series problem.

The datasets used in this thesis consist of data from the first quarter of 2023. It is therefore unknown whether our models could generalize well across the other quarters of the year. The consideration of long-term seasonality of BDT is mentioned in [4], where the

author states that it is reasonable to join data for the same season (or month) across years. This implies that the findings of this work are also applicable for the first quarter of other years, as long as the transit schedules and route structure remain consistent. We could also have chosen different research decisions that probably would have improved the reproducibility of the results for future work. This was in part due to the decision to use data from the SIS platform, which is slowly being completely replaced by the TaaS platform. Since the TaaS platform has a different logic for measuring stop and arrival times than the SIS platform has, an alternative approach utilizing TaaS data would need alternative preprocessing steps to generate appropriate results. It is also possible that the research results would be more reproducible with an alternative strategy for handling outliers. The reason for this can be attributed to the proportion of the detected outliers in our work varying greatly from stop to stop, which was subsequentially imputed without much supervision. This could have slightly skewed the results since stops with a larger outlier proportion consequently had a larger imputation degree, thus introducing a larger proportion of easier observations for the models to work with.

The variables used in this thesis were selected conservatively. This notion came from our research goal of starting small and adding new elements to the experiments as we proceeded. In this sense, we began by using passenger variables as predictors before eventually including work day and rush hour variables in the experiments. We would have liked to examine the influence that other variables could also have had on BDT. As discussed in Section 5.2, we believe that data on local infrastructure and nearby buildings around the stop could have improved BDT estimates. It would also have been interesting to investigate how the demographics of passengers in the stopping districts and the bus headway at the stops would have affected BDT. However, including these variables was unfortunately beyond the scope of the thesis.

Chapter 6

Conclusions

6.1 Conclusion

This thesis explored the possibility of modeling BDT predictions as a time series with external predictors. XGBoost, LSTM and SARIMAX were iteratively applied to subsets of data at selected stops on line 20 from the first quarter of 2023. The cumulated RMSE, MAE and MAPE scores indicated that XGBoost and LSTM delivered more precise BDT predictions than SARIMAX. XGBoost and LSTM produced inseparable performances for all stops in our data, with XGBoost having slightly lower cumulative error across the stops for our selected evaluation metrics. The potential nonlinear relationships between passenger variables and BDT were highlighted to explain why XGBoost and LSTM demonstrated superior predictive capabilities.

Passenger counts and temporal variables were used as independent variables alongside BDT. Our results showed that passenger variables had a bigger influence on BDT than temporal characteristics, with significant variability in feature importance between different stops. The number of boarding and alighting passengers seemed to be the most important variables for predicting BDT among the passenger variables.

The approach of modeling BDT as a time series with external predictors was partially effective. The data expressed a limited presence of typical time series features such as autocorrelation, trends and seasonalities. This challenged the initial belief of BDT having some noteworthy temporal characteristics that could be sufficiently modeled through time series methods. Subsequently, SARIMAX also demonstrated a limited ability to forecast BDT by only having a slightly lower error than the baseline model at all stops in our work. Although LSTM had a lower cumulative error than SARIMAX, the non-temporal XGBoost method ultimately yielded the lowest prediction error in our analysis. We therefore cannot conclude that it is advisable to approach the prediction

of BDT values as a time series with external predictors.

6.2 Future work

The scope and applicability of our findings could be extended beyond what was done in this thesis. Since this study focused on a single bus route, the generalizability of our results could be improved by extending the data source to include multiple routes. This would introduce additional variability by having more unique stop characteristics to work with, which would help solidify BDT trends across stops to a greater degree. It would also be interesting to explore whether the estimation of dwell time for other modes of transportation would result in alternative outcomes, given that trams and metros are also an important part of urban transportation systems.

Future studies on BDT estimation could also benefit from using an alternative pool of independent variables. Factors such as external infrastructure and passenger demographics in relation to stop area were discussed as points of interest in this regard. Since we did not include data on these factors in our work, it would be interesting for future studies to investigate these variables further.

Recommendations for future studies would also be to utilize more data for model training. This could help to underline implications about the temporal characteristics BDT has beyond an extended period of the year. Adding more data could also limit the applicability of a time series approach, as the risk of irregularities in the time axis would increase with more data points. Seeing as this thesis did not provide evidence that a time series approach was beneficial for BDT estimation, using an alternative for future BDT estimation tasks would probably be appropriate.

References

[1] Ruter, "Målbilde for bærekraftig bevegelsesfrihet," Ruter, Technical Report, 2020, [Online]. Available: https://ruter.no/globalassets/dokumenter/ruterrapporter/malbilde-barekraftig-bevegelsesfrihet-2022.pdf.

[2] S. A. Soroush Rashidi and P. Ranjitkar, Estimating bus dwell time: A review of the literature, *Transport Reviews*, vol. 43 (1): 32–61, 2023, DOI: 10.1080/01441647.2021.2023692.

[3] A. Tirachini, Estimation of travel time and the benefits of upgrading the fare payment technology in urban bus services, *Transportation Research Part C: Emerging Technologies*, vol. 30: 239–256, 2013, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X11001598.

[4] I. Isukapati, H. Rudová, G. Barlow, and S. Smith, Analysis of trends in data on transit bus dwell times, *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2619: 64–74, Jan. 2017, DOI: 10.3141/2619-07.

[5] S. Rashidi and P. Ranjitkar, Estimation of bus dwell time using univariate time series models, *Journal of Advanced Transportation*, vol. 49 (1): 139–152, 2015, DOI: https://doi.org/10.1002/atr.1271.

[6] Q. Meng and X. Qu, Bus dwell time estimation at bus bays: A probabilistic approach, *Transportation Research Part C-emerging Technologies*, 2013, DOI: 10.1016/j.trc.2013.08.007.

[7] J. Xin and S. Chen, Bus dwell time prediction based on knn, *Procedia Engineering*, 2016, DOI: 10.1016/j.proeng.2016.01.260.

[8] S. Mohammad, H. Moosavi, A. Ismail, and A. Balali, Evaluating bus dwell time at key stops using automatic data collection systems, *Ite Journal-institute of Transportation Engineers*, 2017, [Online]. Available: https://www.researchgate.net/publication/324131852_Evaluating_bus_dwell_time_at_key_stops_using_automatic_data_collection_systems.

[9] I. K. Isukapati, C. Igoe, E. Bronstein, V. Parimi, and S. F. Smith, Hierarchical bayesian framework for bus dwell time prediction, *IEEE Transactions on Intelligent Transportation Systems*, 2020, DOI: 10.1109/tits.2020.2979390.

[10] T. B. Glick and M. A. Figliozzi, Measuring the determinants of bus dwell time: New insights and potential biases: *Transportation Research Record*, 2017, DOI: 10.3141/2647-13.

[11] T. B. Glick and M. A. Figliozzi, Analysis and application of log-linear and quantile regression models to predict bus dwell times: *Transportation Research Record*, 2019, DOI: 10.1177/0361198119848701.

[12] R. P. Abad and A. Fillone, Estimating bus dwell time using artificial neural networks, May 2018, DOI: 10.11175/easts.12.1266.

[13] B. Aswini, R. M. Savithramma, and R. Sumathi, Bus dwell time forecasting using machine learning models, *2023 7th International Conference on Trends in Electronics and Informatics (ICOEI)*, 2023, DOI: 10.1109/icoei56765.2023.10125868.

[14] J. E. Monogan, Time series analysis, in *Political Analysis Using R*. Cham: Springer International Publishing, 2015: 157–186, DOI: 10.1007/978-3-319-23446-5_9.

[15] G. Spadon, S. Hong, B. Brandoli, J. Rodrigues Jr, and J. Sun, Pay attention to evolution: Time series forecasting with deep graph-evolution learning, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, Apr. 2021, DOI: 10.1109/TPAMI.2021.3076155.

[16] R. Hyndman and G. Athanasopoulos, Forecasting: Principles and practice, in. 2021, [Online]. Available: https://otexts.com/fpp3/.

[17] National Institute of Standards and Technology, *Stationarity*, https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc442.htm, 2023.

[18] I. El Naqa and M. J. Murphy, What is machine learning? In *Machine Learning in Radiation Oncology: Theory and Applications*, I. El Naqa, R. Li, and M. J. Murphy, Eds. Cham: Springer International Publishing, 2015: 3–11, DOI: 10.1007/978-3-319-18305-3_1.

[19] Supervised learning, in *Encyclopedia of Machine Learning and Data Mining*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2017: 1213–1214, DOI: 10.1007/978-1-4899-7687-1_803.

[20] Unsupervised learning, in *Encyclopedia of Machine Learning and Data Mining*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2017: 1304–1304, DOI: 10.1007/978-1-4899-7687-1_976.

[21] S. Doroudi and S. A. Rastegar, The bias–variance tradeoff in cognitive science, *Cognitive Science*, vol. 47 (1): e13241, 2023, DOI: https://doi.org/10.1111/cogs.13241.

[22] S. Raschka and V. Mirjalili, *Python Machine Learning*, Second. Packt Publishing Ltd., Sep. 2017.

[23] L. Rokach, Decision forest: Twenty years of research, *Information Fusion*, vol. 27: 111–125, 2016, DOI: https://doi.org/10.1016/j.inffus.2015.06.005.

[24] J. H. Friedman, Greedy function approximation: A gradient boosting machine., *The Annals of Statistics*, vol. 29 (5): 1189–1232, 2001, DOI: 10.1214/aos/1013203451.

[25] T. Chen and C. Guestrin, Xgboost: A scalable tree boosting system, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, San Francisco, California, USA: Association for Computing Machinery, 2016: 785–794, DOI: 10.1145/2939672.2939785.

[26] Y. Chabchoub, M. U. Togbe, A. Boly, and R. Chiky, An in-depth study and improvement of isolation forest, *IEEE Access*, vol. 10: 10219–10237, 2022, DOI: 10.1109/ACCESS.2022.3144425.

[27] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, *Recent advances in recurrent neural networks*, 2018, [Online]. Available: https://doi.org/10.48550/arXiv.1801.01078.

[28] H. M. and S. M.N, A review on evaluation metrics for data classification evaluations, *International Journal of Data Mining & Knowledge Management Process*, vol. 5: 01–11, 2015, [Online]. Available: https://api.semanticscholar.org/CorpusID:61877559.

[29] T. O. Hodson, Root-mean-square error (rmse) or mean absolute error (mae): When to use them or not, *Geoscientific Model Development*, vol. 15 (14): 5481–5487, 2022, DOI: 10.5194/gmd-15-5481-2022.

[30] A. de Myttenaere, B. Golden, B. Le Grand, and F. Rossi, Mean absolute percentage error for regression models, *Neurocomputing*, vol. 192: 38–48, 2016, Advances in artificial neural networks, machine learning and computational intelligence, DOI: https://doi.org/10.1016/j.neucom.2015.12.114.

[31] C. Pronello and X. R. Garzón Ruiz, Evaluating the performance of video-based automated passenger counting systems in real-world conditions: A comparative study, *Sensors*, vol. 23 (18), 2023, DOI: 10.3390/s23187719.

[32] F. R. Alharbi and D. Csala, A seasonal autoregressive integrated moving average with exogenous factors (sarimax) forecasting model-based time series approach, *Inventions*, vol. 7 (4), 2022, DOI: 10.3390/inventions7040094.

[33] J. Ding, M. Yang, Y. Cao, and S. L. Kong, Dwell time prediction of bus rapid transit using arima-svm hybrid model, in *Sustainable Cities Development and Environment Protection IV*, ser. Applied Mechanics and Materials, vol. 587, Trans Tech Publications Ltd, Aug. 2014: 1993–1997, DOI: 10.4028/www.scientific.net/AMM.587-589.1993.

[34] R. J. Hyndman, *Forecasting with long seasonal periods*, Sep. 2010, [Online]. Available: https://robjhyndman.com/hyndsight/longseasonality/.

[35] E. I. Diab and A. M. El-Geneidy, The farside story: Measuring the benefits of bus stop location on transit performance, *Transportation Research Record*, vol. 2538 (1): 1–10, 2015, DOI: 10.3141/2538-01.

Appendix A

Statement of AI usage

AI has been used in this thesis in accordance with the guidelines provided by the Faculty of Science and Technology at NMBU. The main AI tool used within this thesis includes ChatGPT 4.0. I am aware that I am responsible for all content in this master's thesis, including the parts where ChatGPT was used as a tool. ChatGPT was used as an aid to debug errors in the Python code from the model implementation phase. ChatGPT was also used for brainstorming to get ideas about how certain topics could be explored further. No AI-generated text was directly inserted into the sections of this thesis. The content provided by ChatGPT was cross-referenced with research papers and academic books to ensure the generated information was factual.

Appendix B

Plots

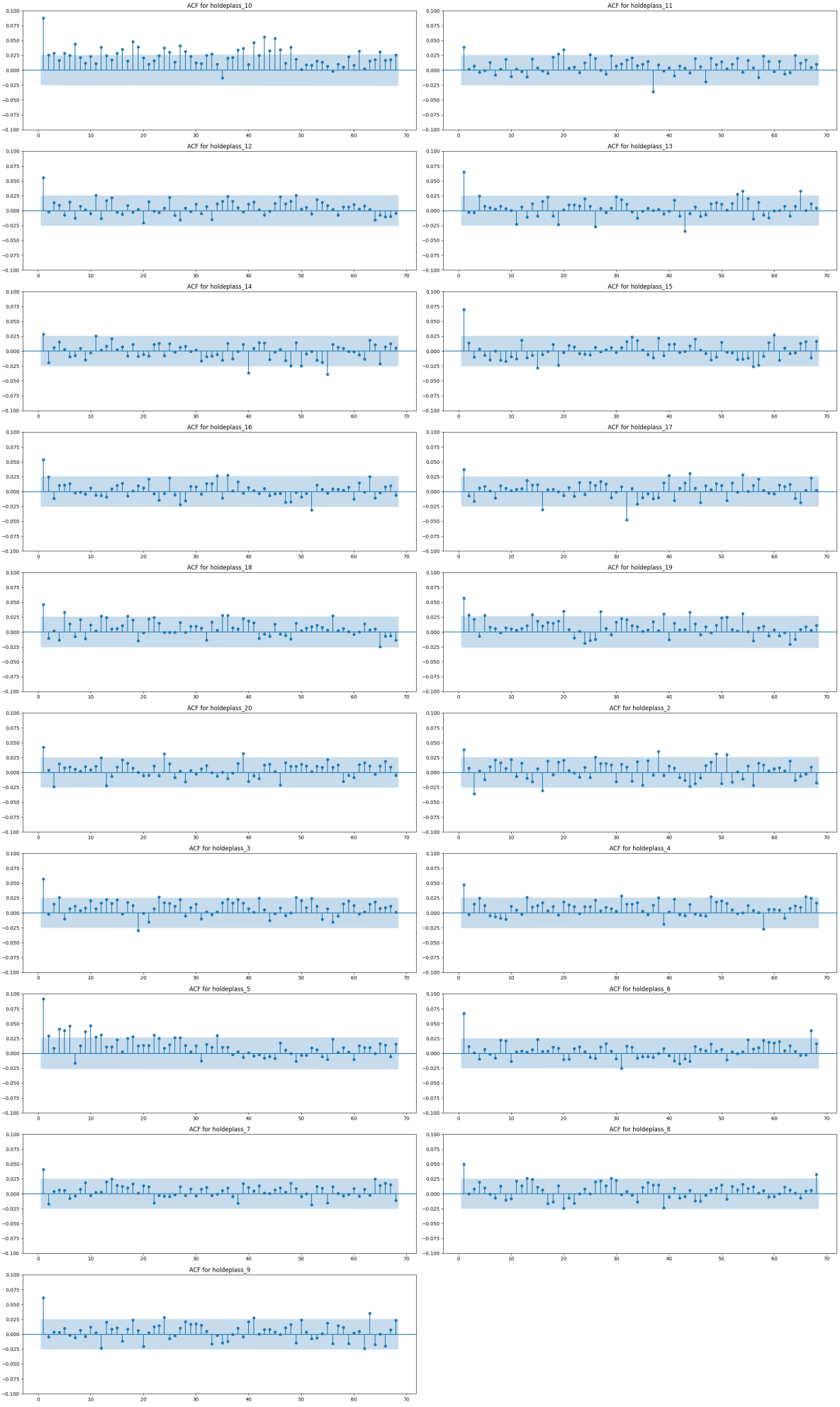

Figure B.1: ACF plots for every stop in the data

Figure B.2: PACF plots for every stop in the data

Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway