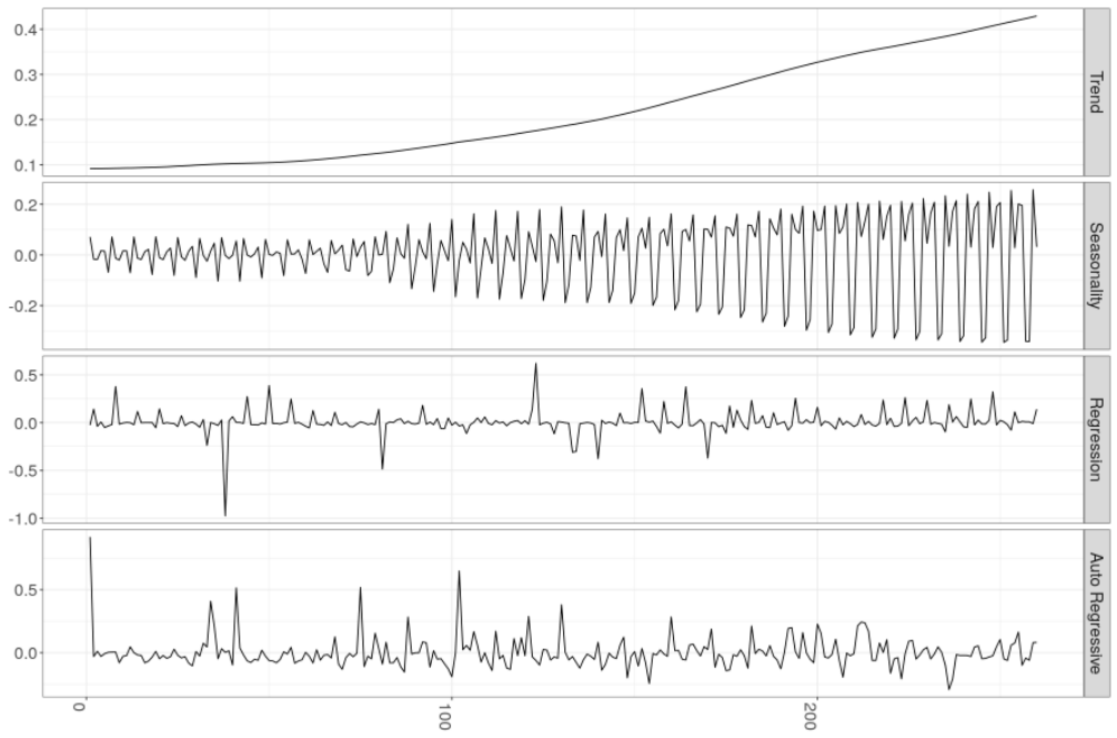


Applying software engineering principles to develop parcel delay forecasting models using tracking data

A study of the models ARIMA, BSTS, and GAM

Master's thesis in Computer science and engineering

ALEX TANG
JOONAS HALMKRONA

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022

Master's thesis 2022

Applying software engineering principles to develop parcel delay forecasting models using tracking data

A study of the models ARIMA, BSTS, and GAM

ALEX TANG
JOONAS HALMKRONA

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2022

Applying software engineering principles to develop parcel delay forecasting models using tracking data
A study of the models ARIMA, BSTS, and GAM
ALEX TANG
JOONAS HALMKRONA

Supervisor: Richard Torkar, Computer Science and Engineering
Advisor: Daniel, Company *X*
Examiner: Robert Feldt, Computer Science and Engineering

Master's Thesis 2022
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Components of the fitted BSTS model over region DEA with service type DHL Parcel Connect.

Typeset in LaTeX
Gothenburg, Sweden 2022

Applying software engineering principles to develop parcel delay forecasting models using tracking data
A study of the models ARIMA, BSTS, and GAM
Alex Tang
Joonas Halmkrona
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

The fast growth of data in the supply chain industry combined with the development of more sophisticated data science tools has amplified the possibilities of actors in this industry to leverage their data. At the same time, industries need to apply best practices for data science programming to ensure software quality. In accordance with the digitalization in the supply chain industry, this thesis aims to explore the possibilities of utilizing tracking event data in parcel delay forecasting. That is, to forecast the share of late packages in various geographical regions. Three different models have been applied for time series analysis where both the predictive ability of the models and the significance of the tracking data is studied. The three models studied in this thesis are Auto-regressive integrated moving average models (ARIMA), Bayesian structural time series (BSTS), and Generalized additive models (GAM). During the development of the models, various tools and techniques were applied to follow software engineering principles. The tools and techniques used was compared to what tools and techniques used currently at Company X.

The research was conducted in collaboration with Company X in a field study that aims to both elaborate on the studied models but also compare practices performed in industry and practices suggested by formal theory research regarding software engineering principles in data science programming. Results show that out of the three models, the best performing model is BSTS and the second best performing model is the GAM. This is expected, since the ARIMA model is, in essence, a sub-model of the two other classes of models. Although the different models have different forecasting capabilities, the quality of the predictions depend on the regions inherent variance in the target variable.

The results also show that there is not a very clear correlation between the features created from the tracking-event data and the target variable although some features stand out more than others. The mixed-effects between variables also seem to have greater predictive ability than the independent contributions of the in-going variables. Furthermore, the results show that Company X makes a distinction between the phases of data exploration and software delivery in data science programming. This distinction is not found during the formal theory research. Although the practices found in the formal theory research are followed by Company X, creating a distinction between the two phases allows different practices to be followed at different stages of a data science project.

To ensure software engineering principles during data science programming, Company X needs to separate data exploration from software delivery and create a clear framework with what tools and techniques should be used at what stage.

Keywords: Time series, ARIMA, BSTS, GAM, software engineering, software engineering principles, supply chain.

Acknowledgements

We would like to first and foremost thank our supervisor Richard Torkar for your guidance and lasting support throughout the thesis. Further, we would like to extend our sincerest gratitude towards company supervisor Daniel who has been of great help during the thesis providing support and guidance throughout the project. Thank you for your patience, your insights, and your knowledge. Vincent, Rigon and Viktor, thank you for helping us during the project and giving us insight and inspiration throughout it.

Alex Tang & Joonas Halmkrona, Gothenburg, June 2022

Contents

List of Figures

List of Tables

1 Introduction

In a digitally changing world, the digitalization of techniques becomes essential for businesses to stay competitive (Saarikko et al., 2020). Industries have started to implement data science tools in their businesses to challenge the market and create innovation. Factors that are essential for an industry to successfully utilize the powers of data science are the availability of large amounts of data and the availability to collect it in a sustainable manner (Sarker, 2021). By studying the data, companies can create forecasting models and understand patterns within their business (Breuker et al., 2016).

One large industry, which has seen a direct benefit of data science tools, is the supply chain industry. In this industry, large amounts of data that can be gathered is generated continuously (Vandeput, 2021). The term supply chain was firstly coined in 1982. It describes a network of organizations and companies which are involved in the various production or delivery processes that create value in the form of products or services to an end consumer (Stadtler and Kilger, 2011). The focus on supply chain management involves the management of the relationships between the involved organizations or companies in a supply chain to achieve a more profitable outcome for all involved parties (Christopher, 2005). Activities in supply chain management include information sharing, inventory management, and network design.

Digitalization and new techniques allow supply chain networks to work differently and to improve performance, despite this only a relatively small amount of companies have utilized the full potential of digital tools (Vandeput, 2021). Cam et al., 2019 states that 53% of the polled manufacturing executives report increased revenues and 61% report decreased costs as a direct consequence of the introduction of data science tools in their supply chain. One application of data science tools is to study delivery delays of commercial outbound parcels. Issues such as small delivery delays at the early stages of the delivery can lead to larger delays at the end, partly due to the bullwhip effect (H. Lee et al., 2004). Therefore, using digital tools can become essential in the future of the industry to understand these effects.

This thesis will be conducted in collaboration with Company X, a cloud-based supply chain company that operates as a delivery manager by connecting stores, warehouses, and shipping locations with carriers, delivery networks, and service providers. The specific case study will involve parcels delivered for Company G's online stores, for whom Company X is the delivery manager. Company G is an international retailer in the clothing industry.

The issue of parcels being late was initially discussed as a classification problem with Company X, where the aim is to predict if a parcel will be classified as delayed or in time, based on its features such as carrier, region, and states in the event chains it has passed through. However, providing a classification algorithm does not give much value to Company G, since knowing that a package will be delayed after it has already been shipped leaves little room for action.

Instead, it is of higher interest for Company X to forecast temporal behavior of aggregated shipments to regions, which give Company G the possibility to plan support for predicted days with an unusually high rate of delivery delays. Hence, time series forecasting methods are suitable for the issue at hand. By modeling and analyzing the ordered sequence of observations and predicting days where the business will report a higher rate of delivery delays than usual, Company G can plan for these days. This thesis will focus on the techniques auto-regressive integrated moving average (ARIMA), Bayesian structural time series (BSTS), and generalized additive models (GAM) to perform time series analysis and to compare the techniques with each other since these models have rarely been applied in the given context. The details of these models will be elaborated upon in the next chapter.

The thesis aims to study whether tracking data can be used as regression variables to help forecast delivery delays using time series. BSTS and GAMs enable the use of regression variables in time series forecasting. BSTS have built in functions for choosing most important regression variables using *spike-and-slap* priors (S. Scott and Varian, 2014), making this an easy model to study impact of adding regression variables. The chosen regression variables are linear. Instead of individual linear predictors, GAM allows modelling of *smooth functions* which has the capability of finding more complex patterns (Hastie and Tibshirani, 1986), such as irregular spikes in the time series by modelling non-linear regression variables. ARIMA is one of the most widely used approaches in time-series forecasting but does not use any regression variables. It is therefore a suitable baseline model for comparison.

While there are many exciting areas of the fast approaching utilization of digital tools in supply chain industries, the fast-growing field faces many challenges simultaneously. One challenge is that data scientists often lack knowledge in engineering principles when developing data science models (Russell et al., 2019). To develop a sustainable framework within data science programming, data scientists should use the help of other research areas, such as software engineering (Wilson et al., 2014; Russell et al., 2019). By applying software engineering methodologies and principles in the development process of data science models, the maintainability and reliability of the models can be improved (Wilson et al., 2014). Since software engineering practices involve systematic and efficient development processes to maintain and improve software robustness (Laranjeiro et al., 2021), these principles should be possible to implement in a data science context as well.

Theoretical software engineering principles are rarely applied in real software development settings (Al-Sarayreh et al., 2021), resulting in gaps between theory and industry. Therefore, with the help of Company X, the validity of various software engineering principles suggested by previous research is analyzed. This is achieved by applying the found principles during model development and by conducting interviews with data scientists at company X. Thus, a comparison can be done between what industry (Company X) perceives as important, and what principles were found to be important during our model development process and theoretical research. Developing the models allow us to mimic tasks that the data scientists at Company X works with.

1.1 Statement of the problem

It is suggested that new, modernized approaches using data science tools can improve predictions within the supply chain network (Vandeput, 2021). As briefly mentioned, the thesis will be conducted together with Company X. Although being a cloud-based software company operating in supply chain businesses, the efforts within data science have only in recent years accelerated for the company. Today, only a handful of data scientists work at Company X. To further develop data science efforts within the company, they wish to investigate the use of time series analysis and the models ARIMA, BSTS, and GAM to forecast delivery delays within their supply chain network.

At the same time, the implementation of data science models require better approaches in order to create reliable and maintainable software (Russell et al., 2019). Since the data science department at Company X is still relatively young, they wish to create a framework for the development process that accommodates software engineering principles.

1.2 Research question

To fulfill the purpose of this thesis, the following research questions will be answered:

1. How does BSTS and GAM models perform, in combination with tracking-event data, in the task of forecasting parcel shipment delays in comparison with a traditional ARIMA model?

2. What software engineering principles are there for data science programming and how can these be implemented in practice?

3. How are software engineering principles applied in data science programming currently at Company X?

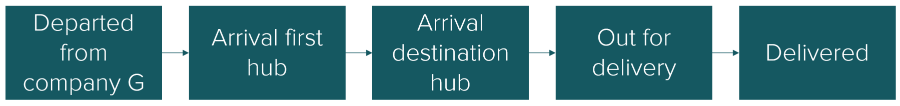


Figure 1.1: Most common chain of events.

1.3 Project case

The specific case within Company X is to study shipments from Company G in Germany. The data consists of shipments from one fixed location to customer locations spread across Germany. Variation between different shipments manifests itself in varying geographic locations, types of carriers performing the shipment, types of services, delivery routes, and so on.

Delivery of a package from an origin point to a final destination can be divided into a series of sequential events. Each registered event data point consists of a timestamp. Examples of these events are departure from the initial location, arrival at intermediate hubs, and in-flight changes to the shipment. The chain of events that a shipment undergoes during its journey from onset to the final destination is also subject for variation. Occasionally, duplicates of events appear during the lifetime of a shipment. In other cases, data on shipment events might also be missing. The duration of time that passes between two sequential events also varies between shipments. This is due to factors such as varying distance between hubs, changing amounts of traffic during different hours of the day, and so on.

The success of a shipment is measured by taking the difference between the estimated time of arrival (ETA) and the actual time of arrival (ATA). The ETA is set as soon as a shipment is confirmed. A successful shipment has minimal difference between ETA and ATA. This thesis will only consider late arrivals as unsuccessful deliveries. Shipments that are delivered earlier than the set ETA is still considered as successful. This difference between ETA and ATA is referred to as delivered on time (DOT) and a shipment is said to be on time if DOT $= 0$. A shipment's DOT $= 0$ if and only if ETA $-$ ATA $<$ 24h. DOT is defined as a binary label to each shipment where either a shipment is late or on time. The most common chain of events for the studied shipments can be found in Figure 1.1.

1.4 Technical environment

The technical workflow of the project is carried out on the Google Cloud Platform (GCP), which is also used by data scientists at Company X. Queries are made in GCP's BigQuery environment to access available data stored in the cloud server. The data science team at Company X working with Company G creates models in `Jupyter` notebooks using both `Python 3` and `R` kernels in GCP's AI workbench. In the notebook, data is accessed by loading saved tables from BigQuery. These tables are then converted into data frames that are subsequently used in the notebooks as input for model creation. All GCP tools available for the data scientists at Company *X* are available for the technical environment in which this thesis is conducted. Extensions such as cloud source repositories, workflow automation, and resource utilization analysis tools are all available.

The programming language chosen for the project is `Python 3` for data preprocessing since this is the language mainly used at Company *X* for its data science projects, and `R` for model creation since `R` has pre-developed libraries for ARIMA, BSTS, and GAM time series. The choice of technical environment is to mimic how data scientists work currently at Company *X*.

2 Theory

This chapter presents the essential theory needed to understand ARIMA, BSTS, and GAM models. Furthermore, software engineering practices in the context of data science are also studied in order to build a framework for software engineering principles in data science projects. This chapter sets the foundation for the results and discussion chapters. The reader is assumed to have knowledge of statistical modeling and software engineering to some extent.

2.1 Delivery time estimation

Despite the potential benefit of data science in logistics and supply chain networks, data science tools have yet to reach their full potential in these fields. Existing supply chain organizations use traditional software algorithms to predict delivery times (Niranjan et al., 2021). The logistics and supply chain management industry is overwhelmed with uncertainties due to the growing generation of structured, semi-structured, and unstructured data (Akbari and Do, 2021). Therefore, data science tools can be powerful due to their computational power and ability to adapt to large-scale data sets and reduce this uncertainty. Today, this data tends in many cases to be underutilized (Niranjan et al., 2021). Data science tools have the potential to improve overall logistics and supply chain performance by identify the driving factors that determine the success of a supply chain network (Makkar et al., 2020).

As carriers deliver record-breaking e-commerce volumes and the risk of delayed shipments increases, more visibility is needed in parcel tracking to better understand where parcels are in the delivery network and also to reduce errors in estimated delivery times (Roberson, 2021). The technology behind parcel tracking has been around for years and has been enhanced over the years to the inclusion of services such as parcel re-routing and scheduling two-hour window deliveries (Roberson, 2021). Furthermore, as E-commerce businesses place a high emphasis on customer-centric services and products, shipment delays have become a critical issue to be tackled as soon as possible.

Akbari and Do, 2021 list a few explanations for the limited adoption of data science tools in the supply chain such as skill deficit in data science competence, cost of adopting data science tools, and lack of high-quality data sets. Shipment delivery time estimation has been researched from the perspective of shipment delay classification by Keung et al., 2021, where various algorithms such as Naive Bayes, Decision tree, ANN, and KNN classifiers were compared for shipment delay classification. In the context of liner shipment classification. Salleh et al., 2017 studied the prediction of arrival punctuality of a vessel at a port of call. Although various models have been used for the prediction of shipment delays, research still needs to be done in the area of parcel shipping delay prediction using time series.

2.2 Time Series

Time series analysis is a useful tool for data science programming practices in contexts involving temporal measurements. The purpose of time series analysis is to extract meaningful statistics and other characteristics of temporal data, which are then used for pattern recognition, forecasting, and other statistical modeling. At the most basic level, time series analysis is concerned with fitting some function of time to time series data and extrapolating it into the future (Brockwell and Davis, 2009).

Definition 1 (Time series) *A time series is a sequence of observations* $(x_t,\ t \in \mathbb{T})$ *with respect to an index set* $\mathbb{T} \subseteq \mathbb{R}$*. A time series model is a specification of the joint distributions (or possibly only the means and covariances) of the sequence of random variables* $(X_t,\ t \in \mathbb{T})$ *from which the sequence of observations* $(x_t,\ t \in \mathbb{T})$ *is assumed to be a realization from (Brockwell and Davis, 2009).*

Therefore, when discussing time series modeling, a selection of a suitable probability model for the observed data as a joint distribution of the sequence of random variables $(X_t,\ t \in \mathbb{T})$ (Brockwell and Davis, 2009) is refered.

Definition 2 (Discrete time series) *A discrete time series* $(x_t,\ t \in \mathbb{T}_0)$ *is one in which the set* $\mathbb{T}_0$ *is discrete (Brockwell and Davis, 2009). A continuous time series is obtained when the data is continuously observed over some time interval, for example* $\mathbb{T} = [0, 1]$.

In reality, only the stochastic process at finite times can be observed. Only discrete models will therefore be considered and for simplicity, group data to their respective calendar day so that discrete set of integers $\mathbb{T} \subseteq \mathbb{Z}_+$, will be observed.

For many economic and social time series, it is typical that its salient characteristics are the trend, which expresses the long-run evolution of the series movement, and a seasonal component that repeats itself more or less over the observed time (Harvey, 1990). These characteristics should be captured by a time series model for the series. Such a model can be formulated in several ways. A starting point is to assume that the series can be decomposed in the following way (Harvey, 1990).

$$Observed\,series = Trend + Seasonal + Irregular \tag{2.1}$$

The irregular component represents nonsystematic movements in the series and is often modeled as Gaussian white noise. Such a process is also known as a *random walk*. A time series may also contain other components and in many cases, the properties of a series are subject to change over time. Therefore, it is beneficial that a model can adapt to changes in the properties of a series. A principal structural time series model is a time series model in which the parameters are allowed to vary over time and one that can be formulated in terms of components that have a direct interpretation. The different components enter into the model in an additive fashion. The stochastic properties of a structural model depend on the full set of disturbances. In time series modeling, it is essential to characterize the properties of these disturbances. (Brockwell and Davis, 2009).

2.2.1 Stationarity

Three conditions must be satisfied for a stochastic process to be stationary, that is (Brockwell and Davis, 2009):

$$\mathbf{E}(\mathbf{y_t}) = \mu \tag{2.2}$$

$$\mathbf{E}[(\mathbf{y_t} - \mu)^{\mathbf{2}}] = \mathbf{Var}(\mathbf{y_t}) = \gamma(\mathbf{0}) \tag{2.3}$$

$$\mathbf{E}[(\mathbf{y_t} - \mu)(\mathbf{y_{t-\tau}} - \mu)] = \gamma(\tau) \tag{2.4}$$

Where $\mathbf{E}(.)$ is the expectation, μ is the mean value, $\mathbf{Var}(.)$ is the variance and $\gamma(.)$ is the autocovariance function. The autocovariance is the covariance between an observation at an arbitrary time and a later occasion where the total difference in time between the two observations is τ. What is essential about a stationary time series is that its properties do not change over time. The mean, the variance, and the autocovariance, all are independent of the time t. Only the relative time difference τ between two observed time points is relevant in the analysis. Stationarity can be achieved by differencing, which is the action of subtracting a previous observation y_{t-1} from the succeeding observation y_t. that is:

$$\texttt{Difference}(t) = y_t - y_{t-1} \tag{2.5}$$

This is done for the whole observed time series. A time series may need to be differenced several times before it becomes stationary (Brockwell and Davis, 2009).

2.3 Auto regressive integrated moving average models

A general class of time series models that are also capable of exhibiting non-stationary behavior are called ARIMA(p,d,q) processes. ARIMA is an abbreviation of Auto Regressive Integrated Moving Average (Harvey, 1990). An auto regressive model creates forecasts of the target variable using a linear combination of past observations of the target (Hyndman and Koehler, 2006). Auto indicates that it is a regression of the variable upon itself.

In order to understand the ARIMA(p,d,q) model in greater detail, an explanation to how its individual components behave will be given. First, the auto-regressive part is explained. An auto-regressive model of order p(AR(p)) is shown below:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t \tag{2.6}$$

Here, ε_t denotes an additive noise mechanism. As seen in equation 2.6, y_t is forecasted with lagged values of y_t as predictors. Equation 2.6 is referred to as an AR(p) model meaning an auto-regressive model of order p. The $\phi_1, \ldots, \phi_p$ are the model parameters and c is a constant. Changing $\phi_1, \ldots, \phi_p$ results in different time series patterns. The AR(1) model can be investigated in order to exemplify the auto-regressive model, which is restricted to stationary data (Hyndman and Koehler, 2006):

- $\phi_1 = 0$ gives y_t equivalent to white noise.
- $\phi_1 = 1$, $c = 0$ gives y_t equivalent to a random walk. If $c \neq 0$, y_t is equivalent to a random walk with drift.
- $\phi_1 < 0$ gives a y_t which fluctuates around the mean.

The MA(q) part of ARIMA is the moving average model which has the following mathematical notation (Hyndman and Koehler, 2006):

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_t \tag{2.7}$$

The MA(q) model uses past forecast errors in a conceptual regression model instead of past values of itself. c denotes the mean of the series while the $\theta_1, \ldots, \theta_1$ values are the parameters of the model. The $\varepsilon_t, \ldots, \varepsilon_{t-q}$ values denotes the additive noise mechanism for each temporal observation. q is the order of the MA model.

To create an auto regressive moving average (ARMA) model, the time series needs to be stationary (Hyndman and Koehler, 2006). The I(d) in ARIMA is the integrated part which refers to the number of times it is required to differentiate the studied time series in order to achieve stationarity. When combining differencing with an auto-regressive model and a moving average modelm an ARIMA model that forecasts its target y_t is obtained, with predictors that include both lagged errors and lagged values of y_t.

2.4 Bayesian structural time series models

BSTS are models that mainly consist of three distinct components (S. Scott and Varian, 2014). These components are:

- A "basic structural model". This is the technique used for time series decomposition into different elements, such as trend and seasonality. This is estimated using Kalman filters. (S. Scott and Varian, 2014). Kalman filters are recursive algorithms that produce estimates of the state vector at time t, based on the information available at time t (Harvey, 1990). The state-space equations will be elaborated upon in Subsection 2.4.1. The interested reader is referred to chapter three in Harvey, 1990 for further information regarding the topic of Kalman filters.

- Spike-and-slab method. This is a Bayesian prior that is used to find the most important regressor (independent, predictor) variables in the model.

- Bayesian model averaging. Bayesian model averaging provides a way to account for model uncertainty and is simply a weighted mixture of the probabilities under each model.

The three components, Kalman filter, spike-and-slab regression, and bayesian model averaging tend to work well together. The parameters that must be estimated in the model are the spike-and-slab variables, which determine what variables are included in the model, the regression coefficients for the regression part of the model, and finally the variances of the error terms in the different components of the structural time series (S. Scott and Varian, 2014). In the following subsections, a brief overview of each method is presented.

2.4.1 Structural time series

The practical usefulness of structural time series stems from the fact that they are flexible and modular (Brodersen et al., 2015). The flexibility is derived from the fact that a large class of models, including ARIMA models, can be written in the state space form. Structural time series models are exactly state-space models for time series data. The structural time series models can be defined in terms of the pair of equations.

$$y_t = Z_t^T \alpha_t + \epsilon_t \tag{2.8}$$

$$\alpha_{t+1} = T_t \alpha_t + R_t \eta_t \tag{2.9}$$

where $\epsilon_t \sim \mathcal{N}(0, \sigma_t^2)$ and $\eta_t \sim \mathcal{N}(0, Q_t)$ are independent from the other unknown variables. Equations 2.8 and 2.9 are called the observation equation and the state equation. The latter governs the evolution of the latent d-dimensional state vector α_t over time. The former equation links the observed data y_t to to the state vector α_t. y_t is a scalar observation. Z_t is called the output vector and is d-dimensional. T_t is the state transition matrix and has dimension $d\,x\,d$. R_t is called a control matrix with dimension $d\,x\,q$. ϵ_t and η_t are error terms for the observation y_t and the system R_t.

Structural time series are modular in the sense that it is possible to assemble the model matrices Z_t, T_t, R_t, and Q_t from a library of modular submodels (i.e., the BSTS library by S. Scott and Varian (2014)) to capture important features of the data such as trend and seasonality.

2.4.2 Spike-and-slab

The spike-and-slab method is a variable selection method that can be used for selecting the most “important” variables in a Bayesian setting (S. Scott and Varian, 2014). Let γ be a vector with the same length as the number of regression variables in the data. Let $\gamma_k = 1$ if $\beta_k \neq 0$ and $\gamma_k = 0$ if $\beta_k = 0$. Denote the subset of elements of β where $\beta_k \neq 0$ as β_γ. The spike-and-slab prior can be described as:

$$p(\beta, \gamma, \sigma_\epsilon^2) = p(\beta_\gamma | \gamma, \sigma_\epsilon^2) p(\sigma_\epsilon^2 | \gamma) p(\gamma). \tag{2.10}$$

$p(\gamma)$ is the “spike” since it places positive probability mass at zero. It is often convenient in practice to use an independent Bernoulli prior:

$$\gamma \sim \prod_{k=1}^{K} \pi_k^{\gamma_k} (1 - \pi_k)^{1-\gamma_k}. \tag{2.11}$$

where π can be interpreted as the “expected model size”.

For simplicity, one can choose $\pi_k = \pi \, \forall \, k$. Then, if one expects p predictors, one can simply choose $\pi = p/K$ where K is the number of predictor variables in the model (S. Scott and Varian, 2014). It is also noteworthy that specific variables can be forced to be included by setting $\pi_k = 1$. The thesis will not elaborate further upon the mathematical underpinning of the model since the BSTS software package includes default values for the parameters and it lies outside the scope of the report. For further reading on the topic, the interested reader is referred to S. Scott and Varian, 2014.

2.4.3 Bayesian model averaging

In order to yield forecasts of y_{t+1}, draws from the posterior distribution of the parameters of the model will be performed. The parameters are drawn and combined with the available data to get a forecast for the specific draw. An estimate of the posterior distribution of the forecast y_{t+1} is obtained by repeating these draws many times. This way of producing forecasts are motivated by the statement of S. Scott and Varian (2014):

> [. . .] averaging over an ensemble of models does no worse than using the best single model in the ensemble.

The draws and forecasts are generated automatically by applying the prediction method `predict.bsts()` supplied by the BSTS library by S. Scott and Varian (2014).

2.5 Generalized additive models

To understand GAMs, it can be helpful to first understand *generalized linear models* (GLM). A GLM relates an expected value $\mathbf{E}(\mathbf{Y})$ to linear predictors $X\beta$ with a link function $\mathbf{g}(.)$ which provides a relationship between the linear predictor and the expected value.

in the case of GAMs the response variable depends on a set of smoothing functions $s_k(.)$ (Hastie and Tibshirani, 1986):

$$\sum_{k=1}^{K} s_k(X_i). \tag{2.12}$$

In other words, a GAM produces its predictions by summing over the values generated by the set of smoothing functions $s_k(.)$, where X_i is the set of predictor variables in the data. while traditional linear regression models assume a linear form on the covariate effects, GAMs are capable of modeling the dependence of the target variable in a non-parametric fashion (Hastie and Tibshirani, 1986). The mathematical annotation for GAM can thus be written as:

$$g(\mathbf{E}(\mathbf{Y})) = \alpha + s_1(x_1) + s_2(x_2) + \ldots + s_n(x_n) \tag{2.13}$$

Where Y is the target variable and $\mathbf{E}(\mathbf{Y})$ denotes the expected value of Y. $g(Y)$ is the link function which links expected value of the predictor variables $X_1, X_2, \ldots X_n$. $s_k(.)$ is a set of smoothing functions that are capable of capturing non-linear relationships between the target variable and the predictor variables. The predictor is an additive function which means that it is the sum of the set of fitted smoothing functions, hence the name *Generalized Additive Models*. The smoothing functions $s_k(.)$ are functions that are unspecified and are estimated with *scatterplot smoothers* (Hastie and Tibshirani, 1986). Some examples of scatterplot smoothers that Hastie and Tibshirani, 1986 recount are running mean, running median, running least-square line, kernel estimate, and spline.

2.6 Application areas for ARIMA, BSTS and, GAM

ARIMA models were popularized in 1970 by George Box and Gwilym Jenkins. They are time series analysis models suitable for time series which can be assumed to have a reasonable amount of continuity between the past and present (Stellwagen and Tashman, 2013). ARIMA models have been applied in price prediction of electricity (Alsaedi et al., 2019). Other econometrics areas are also common application areas for ARIMA models since it gives the user an understanding of the mean, autocorrelations and trends of the data. The authors Abu Bakar and Rosbi (2017) also suggest econometrics areas for application such as forecasting market price changes, national house price inflation, and investment strategies for timing purposes. The application of ARIMA models in supply chain networks has been limited. Wang et al. (2021) uses ARIMA models for demand forecasting for Taiwan's semiconductor industry with the purpose of inventory management which can be related to supply chain network management forecasting.

BSTS has been utilized in the context of news sentiment influence on stock prices Ray et al., 2021; For fuel sales forecasting (Lian et al., 2021); Brodersen et al. (2015) used a BSTS model for inferring the causal impact of advertising efforts at Google. Güvercin et al. (2021) applied various time series for predicting, not shipment delivery times, but flight delays using clustered network models based on airports where the models are applied individually to each airport data for predicting flight delays. Others have also studied flight delays (Yiu et al., 2021). The use of BSTS models in the context of predicting shipment delivery delays is limited. One explanation for this is that carrier data is not universally available in contrast to flight data.

GAMs are widely popular and have been applied in several different research areas. In research areas relevant to this study, 206 research articles are published (Engineering (100), Decision Science (73), Economics, Econometrics, and Finance (73)). Some examples of the applications of GAMs in time series analysis are prediction of food sales and beverages in staff canteens and restaurants (Posch et al., 2022), association between meteorological factors, and daily new cases of COVID-19 (Yuan et al., 2021).

2.7 Software engineering principles

Al-Sarayreh et al. (2021) does a systematic mapping and a quantitative literature review of general software engineering principles and finds 30 relevant papers on this subject ranging from the publication year 1969 to 2020. A total of 592 software engineering principles were proposed. Naturally, many of them express similar meanings, but some express contradicting meanings. The authors state that after the literature review of the found 30 papers, more work needs to be done in integrating principles with practices, which in this thesis is expressed as suggesting tools and techniques for applying said principles. The authors also suggest that further work needs to be done in making software engineering principles more available across application domains and other disciplines such as data science programming. Finally, Al-Sarayreh et al., 2021 claims that more work needs to be conducted in applying software engineering principles in real software development environments.

2.7.1 Strategic engineering areas for data science programming

In practice, a data science model is only a minor part of the overall area of data science programming. Despite modeling only being a smaller part of the whole area, it is still essential that a model works properly so that it can be a reliable component of a larger software system. The larger software system also covers tasks such as engineering of data pipelines, monitoring, and logging (Bosch et al., 2021).

In a typical prediction model, integrated into a larger software system, there are four major data science activities (Bosch et al., 2021). These are: assembling of data sets, creating and evolving the machine learning model (model development), training and evaluating the model, and finally deployment. There are engineering challenges that need to be fulfilled for the model to perform reliably in the system (Bosch et al., 2021).

The first challenge is data quality management. Engineering principles must be set to retrieve data of sufficient quality for training and inference. Questions such as handling missing and unbalanced data, and other preparations must be done.

The second strategic area is during design methods and processes. To create an understandable and easy-to-use model it must be repeatable and of high engineering quality.

Thirdly, model performance is a challenging area in the model development process. The performance of the model is measured both in accuracy and quality attributes. Therefore, both problems in training data and lack of support for quality attributes must be addressed.

Deployment and compliance of the model is the final strategic area that must be addressed. Ultimately, deployment of a model in an operational system might require changes in the overall architecture of the system.

2.7.2 The need for software engineering principles in data science programming

When developing software products, many data scientists lack skills in formal and rigorous software engineering principles (Russell et al., 2019). In some cases this results in software with poor maintainability and reliability. Because of this, extra time needs to be spent on tasks such as debugging. To increase the maintainability and reliability of a model, many software engineering methodologies can be used (Wilson et al., 2014).

One large reason for why data scientists rarely follow software engineering principles when programming, is that there is little exchange of ideas between the general software engineering community and the data science community according to Hannay et al. (2009). The reason for this is that data science developers, and “regular” software developers often work with tasks of different nature. Although both professions involve programming, the end result of the programming tasks differ. This leads to surprisingly little collaboration between software engineers and data scientists since the collaboration opportunities are lacking.

Hannay et al. (2009) also claims that there is a lack of formal training for data scientists in the fields of software engineering. The software development aspects of their work are often self-taught. The formal training received is also often given by computer science departments, which teach general software courses that data scientists find less relevant for their domain-specific knowledge. Another aspect is that data scientists might not see the need for formal software development training. This stems from the fact that data science programming is often firstly done on a small scale in an exploratory manner which grows larger in scale with time if the software proves its usefulness in scientific investigations. Before the software proves its usefulness, it is uncertain if it will be a part of a larger end-product. Therefore, the visible need for software engineering practices in data science programming is often shown when it is too late according to the authors.

2.7.3 Software engineering principles in data science programming

Wilson et al. (2014) presents a set of eight software engineering practices that should be adopted when developing data science models. This thesis builds upon the suggested software engineering principles presented by Wilson et al., 2014, but also complements these for domain relevant techniques and tools which help the data scientists at Company X to adopt the given principles.

Consistent coding implies writing code that is both understandable and easily read by other programmers. This is to increase the productivity when coding according to Wilson et al. (2014). A parallel that the authors draw is breaking up code in the same way as scientific papers are broken down into sections and paragraphs to understand the content. Furthermore, the code need consistency in its styling format.

Suggested techniques for *Consistent coding* are breaking up code into easily understood functions, each of which conducts a single task. Variables and functions should be named consistently, distinctive from each other. The name should explain what the function does or what value the variable holds. Deissenbock and Pizka, 2005 suggests that an useful tool for consistent naming is an *identifier dictionary*. By storing information about all identifiers such as their names, the type of the object, and a comprehensive description, the consistency of the code written can be improved.

Another suggestion is to use different types of coding style frameworks to keep a consistent coding style. An example of a coding style framework is the `PEP8` style guide for `Python 3` code (van Rossum et al., 2001). The guide provides rules when writing `Python 3` code such as consistent variable declaration, code length and documentation approaches. The `PEP8` style guide can be implemented with the use of the `Pylint` library which catches bugs and style problems in the `Python 3` code written.

Automating task performances allows the program to unburden the programmer of some tasks. When working with data science models, it can happen several times that a task is performed repeatedly. To save time, these repeatable tasks could be automated instead of performed manually.

Wilson et al., 2014 suggest that recent commands should be saved in a file for re-use. One simple approach to do this is to use command-line tools which have a history option that allows users to re-execute historic commands. Commands can also be saved in a notebook by creating a cell which re-runs a set number of functions and classes.

Another technique for automating task performances is to run time-consuming computer programs at optimal times when the programmer is inactive. GCP allows the user to schedule a notebook's runs by using Vertex AI workbench (Namjoshi, 2021). By scheduling training of models at optimal times and re-training when updates have occurred to the data, the efficiency of a data scientists work can increase. Automation of programs can also be done through GCP airflow tools, where scheduled running sequences can be set to run depending on the results produced by parts of the program GoogleCloud, 2022a. Utilizing this tool decreases the manual labour required by a data scientist.

Incremental coding means working on a software step-wise. A major difference between commercial software development teams and data science programming teams is that data science programming teams tend to not get static requirements from stakeholders (Wilson et al., 2014). The requirements they receive are seldom technical but rather regarding exploration of data. In addition, data scientists often cannot know what to do next before the current version of a model has produced some results. Therefore, it is of great advantage to work in incremental steps with frequent feedback from the stakeholders during data science programming.

When working incrementally, a challenge that can arise is the tracking of changes to a software, especially when working in collaboration with other programmers (Wilson et al., 2014). Having several people changing the same code and with requirements being agile, programmers need to set up a version control system to keep track of the software. The purpose of a version control system is to store historical changes a repository. By doing so, projects can be worked on individually and locally, and the final program is only modified when a programmer commits their changes. Furthermore, version control systems also allow users to resolve conflicts when several people have edited files simultaneously and review each others code. GCP allows the user to use cloud source repositories, which is a tool for version control (GoogleCloud, 2022b). In the cloud source repositories, the users can perform `git` operations and keep track of the history of the developed model. The GCP also allows version control of the data. By using version control on the data, the user can recreate previously produced results of the model.

Furthermore, Wilson et al. (2014) suggests that it is preferable to follow an agile software development methodology over a traditional "waterfall" software development methodology.

Avoiding repetition is essential for software modularity. If changes and updates are made to code which is repeated in multiple places, the risk of errors en inconsistencies increases (Wilson et al., 2014).

The main technique suggested for this principle is writing modular code, according to Wilson et al., 2014. In this way, duplication of code is minimized and bug fixes are implemented faster. With modularized code, the user can also re-use code rather than rewriting it. Modular code can also be achieved by writing modules and libraries which `Jupyter` notebooks are dependent upon rather than writing all of the code in notebook cells. Creating functions can also help the programmer to avoid rewriting code.

By *testing notebooks* data scientists are more likely to produce reliable and maintainable code. Wilson et al. (2014) states that verification of the code and maintaining the validity of the code over time is essential in data science projects.

By writing assertions and unit tests, the validity of the code can be maintained. Data scientists lack the habit of writing tests as a traditional software engineer does, partly because of the lack of coding standards but also because the environment where programming is done differs (Russell et al., 2019). One package which can be utilized in `Jupyter` notebook environments to write tests is `testbook` which allows unit tests to be run on notebooks in separate test files and therefore treating `Jupyter` notebook files as `Python` files (nteract team, 2021). `unittest` is also possible to utilize directly within a `Jupyter` notebook to create unit tests.

The authors Russell et al., 2019 claim that the software engineering principle within data science model development which has been lacking the most in progress in recent years is software testing. The authors show that from the majority of all available comprehensive `R` Archive Network packages, around 70% do not have tests made available with the package. The authors highlight the importance of writing unit tests and acceptance tests during development of `R` packages. Furthermore, the authors also state that unit tests should be written by the developers of the software.

Software optimization allows the software to utilize its resources more efficiently to become more time-efficient. Calculation programs can often be optimized for other, lower-level, languages. Wilson et al., 2014 suggest that if data scientists wish to optimize their code execution with lower-level languages such as `C`, it should be done only after the code works correctly using higher-level languages. This is due to the fact that higher-level languages help the user with program comprehension.

Software documentation can be done in several ways and is essential for a program's understandability. To highlight the importance of documentation in scientific programming, Wilson et al., 2014 draws a parallel to experimental protocols for research methods. In both cases, the purpose of the documentation procedure is to make the artifact reproducible and understandable. The maintenance cost is also lowered with documentation.

B. Lee, 2018 discusses the importance of writing comments as the programmer codes. Error messages should be informative and provide a solution to the error if it occurs. If the developed program operates in a command-line interface, it is beneficial to include a help command for the functions which are created by the programmer. B. Lee, 2018 also suggests using the package `Click` to create command-line interfaces. Comments explaining functions and code snippets should also be written when necessary. Beyond documentation within the program itself, external documentation tools should be used according to B. Lee, 2018. A README file should be included with basic information and a quick-start guide to the developed model. In contrast to the authors Russell, Bennett, and Ghosh (2019), B. Lee, 2018 states that the most underemphasized software engineering principle in data science programming has been software documentation. The authors highlight that software reproducibility suffers from the lack of software documentation.

Software collaboration is the final area of software engineering principles that should be adopted in the domain of data science programming (Wilson et al., 2014). Collaboration during data science programming can eliminate bugs and improve the code's readability. It is also a good way to spread knowledge and best practices around the team.

Techniques such as code reviews before merging and pair programming are brought up as suggestions to implement collaboration when developing data science software. Kalyan et al., 2016 states that utilizing `git` repositories for code review can improve a team's software collaboration efforts by allowing several team members to look at pull requests before merging code. By implementing standards for code review, developers can improve collaboration and ensure that other software engineering principles are followed in a synchronized way within the team Panichella and Zaugg, 2020.

3 Methods

The methods chapter describes the methods followed during the thesis in order to answer the presented research questions.

3.1 Research procedure

The authors Stol and Fitzgerald (2018) highlight eight different research strategies in the domain of software engineering which are shown in Figure 3.1. The research conducted during this thesis is classified as what Stol and Fitzgerald (2018) describes as a field study:

> [...] A field study refers to any research that is conducted in a specific, real-world setting to study a specific software engineering phenomenon.

The specific software engineering phenomenon studied in this thesis is the applicability of ARIMA, BSTS, and GAM in forecasting parcel delays, and the applicability of software engineering principles in data science programming. The study is unobtrusive, meaning that the researchers will not change or control any of the parameters or variables of how Company X currently operates. It is also context specific since the thesis only studies the applicability of the models and principles within Company X.

Since this thesis will evaluate the models ARIMA, BSTS, and GAM in time series forecasting and also evaluate how software engineering principles can be applied in the context of data science programming, the thesis can be categorized into two different research areas, that is, data science and software engineering. The data science area involves model development and evaluation while the software engineering area involves applying and evaluating software engineering principles in a data science context. The field study is conducted by following two parallel processes, which are illustrated in Figure 3.2.

To implement the field study, a literature review is firstly conducted, where the current state of research in both the relevant areas is examined. These areas are ARIMA, BSTS and GAM models and the application of software engineering principles in the domain of data science programming. The results of the literature review is outlined in the theory chapter. In the second step of the field study, the models and applicable software engineering principles are implemented in a real-world setting provided by Company X. Lastly, the performance of the models are evaluated. The utility

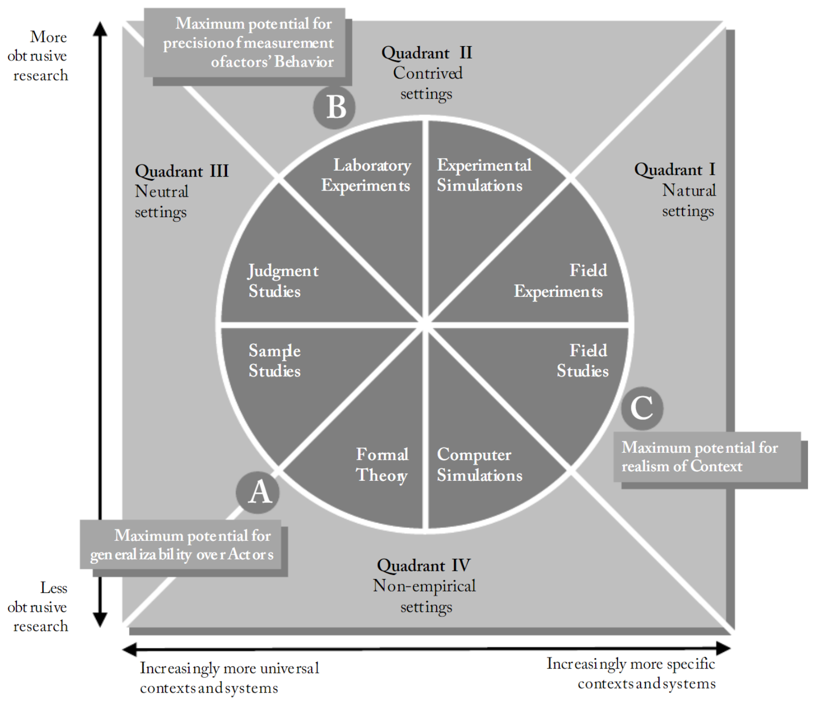
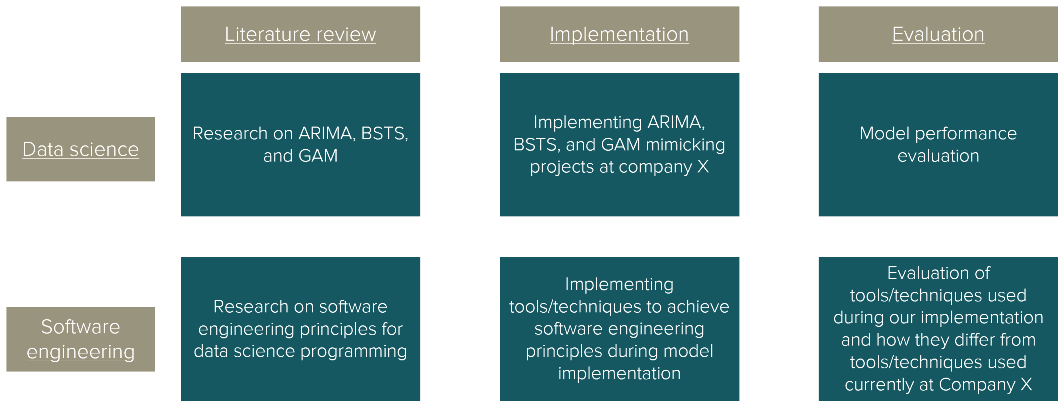

More obtrusive research
Less obtrusive research
Maximum potential for precision of measurement of actors' Behavior
B
Quadrant II Contrived settings
Quadrant III Neutral settings
Quadrant I Natural settings
Laboratory Experiments
Experimental Simulations
Judgment Studies
Field Experiments
Sample Studies
Field Studies
Formal Theory
Computer Simulations
C
Maximum potential for realism of Context
A
Maximum potential for generalizability over Actors
Quadrant IV Non-empirical settings
Increasingly more universal contexts and systems
Increasingly more specific contexts and systems

Figure 3.1: The conducted research and its methodology steps in correlation with the eight research strategies for software engineering by Stol & Fitzgerald, (2018).



	Literature review	Implementation	Evaluation
Data science	Research on ARIMA, BSTS, and GAM	Implementing ARIMA, BSTS, and GAM mimicking projects at company X	Model performance evaluation
Software engineering	Research on software engineering principles for data science programming	Implementing tools/techniques to achieve software engineering principles during model implementation	Evaluation of tools/techniques used during our implementation and how they differ from tools/techniques used currently at Company X

Figure 3.2: The two parallel processes of data science and software engineering conducted during this thesis.

of the found software engineering principles is evaluated by examining differences between practices at Company X and theory. The utility of the software engineering principles is also evaluated by examining the reasoning behind why some principles are followed and why others are not.

3.2 Model development methods

In this section, the key concepts that the devised models are built upon are presented. Before a methodology could be established, an initial data exploration phase was carried out to understand the data at hand. The target variable that is to be predicted is the share of late packages to a specific NUTS1 region for a specific service type. NUTS stands for the Nomenclature of Territorial Units for Statistics and is a geocode standard for hierarchically dividing a country into subdivisions for a statistical purpose (Comission, 2021). The total number of NUTS1 regions in Germany is 16. There are also two service types for each region, service type A and service type B. Therefore there are 32 (16 regions and two service types) potential time series to be studied.

The share of late packages is defined as the total number of late packages divided by the total number of packages shipped during a day. A package is classified as late if its ATA is at least 24 hours greater than its ETA. The data which the study is built upon corresponds to shipment information for the calendar year 2021 in Germany. In this thesis, three models that can forecast the target variable are implemented. The share of late packages is forecasted individually for each NUTS1 region and service combination. The NUTS regions are displayed in Figure 3.3.

3.2.1 ARIMA

Firstly, an ARIMA model is fitted to the various time series. An individual ARIMA model is fit to each service type in each region's time series of observed values of the target variable. As stated previously, the observed values in the time series are the share of late packages for each day. This ARIMA model will be used as a baseline model and the subsequent models will be compared against the performance of the baseline model and each other.

An optimal ARIMA model for each distinct regional time series of late shares is found by applying the `R` function `auto.arima` developed by Hyndman and Khandakar (2008). The function searches for possible models, within the order constraints provided by the user and returns the best model according to either AIC, AICc, or BIC value. The mentioned abbreviations are three different estimates of prediction error. The standard setting used in `auto.arima` package is AIC, which is used during this study as well. Furthermore, the order constraints provided for the `auto.arima` function are the standard values for the function in the study. These are the following:

$$max(p, d, q)x(P, D, Q, s) = (5, 2, 5)x(2, 1, 2, 1) \tag{3.1}$$

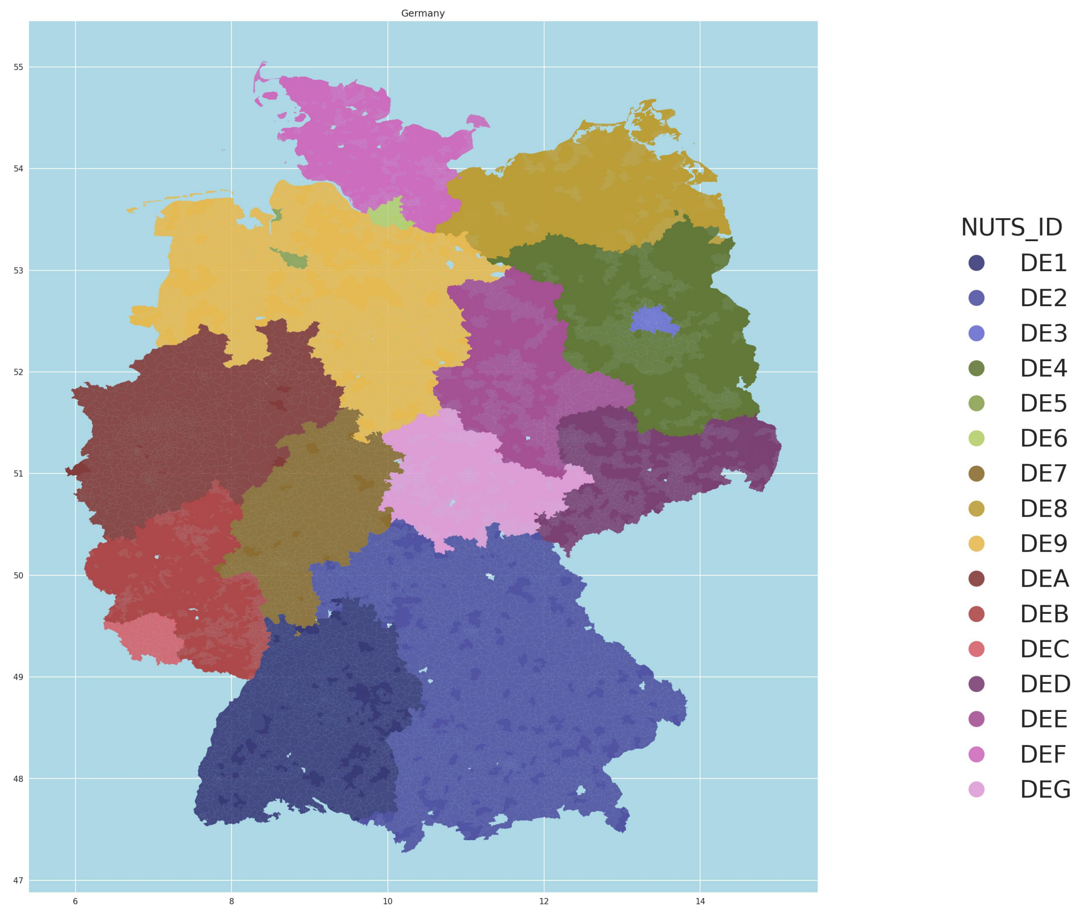


Figure 3.3: NUTS1 regions of Germany.

where the first three parameters (5,2,5) specify the maximum orders for the parameters of the ARIMA function, and the four last parameters (2,1,2,1) specify the maximum order of a potential seasonal component of the model. In other words, the `auto.arima` function searches over a space of models which includes seasonal-ARIMA models. An ARIMA model with a seasonal component is also commonly known as a SARIMA model where the S stands for seasonal, hence the name.

3.2.2 BSTS

A BSTS model is fitted for every service type in every region. The `R` package `bsts` by S. Scott and Varian (2014) will be used for implementation. The BSTS model which is developed during this thesis contains four distinct state-space components. The first component is an automatic auto-regressive component that adds a sparse AR(p) process to the state distribution. The component contributes α_t to the expected value of Y_t, the equation for α_t is defined as:

$$\alpha_t = \phi_1 \alpha_{t-1} + ... + \phi_p \alpha_{t-p} + \epsilon_{t-1}, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2). \tag{3.2}$$

where the component consists of the last p lags of α (S. L. Scott, 2021). The Auto-AR(p) component considers up to p number of lags for the time series models and decides on the optimal amount. The $\phi_i,\ i \in [1, 2, \dots, p]$ are the coefficients for the p lags of the chosen AR(p) model. $\mathcal{N}(0, \sigma^2)$ refers to a normally distributed variable with mean zero and variance σ^2. In other words, ϵ_t is a white noise process. The `bsts` packages chooses the optimal Auto-AR(p) values.

The second component of the BSTS-model is a local-linear trend component. This component adds a local linear trend to the model which assumes that both the mean and trend component of the time series follows random walks. The equations for both the mean (3.3) and the slope (3.4) are the following (S. L. Scott, 2021):

$$\mu_{t+1} = \mu_t + \delta_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2). \tag{3.3}$$

and

$$\delta_{t+1} = \delta_t + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \sigma^2). \tag{3.4}$$

As one can observe, the mean value at an arbitrary time is a sum of the mean value and the slope at the previous time of measurement and an error term. The slope at a time $t+1$ is simply the slope at time t and Gaussian white noise term.

The third component of the BSTS model is a seasonal component. Since Saturdays are removed from all the regional time series (this will be elaborated upon in Subsection 3.3.2), the seasonal component will have a periodicity of six. Denote the number of seasons in the dataset as S. Then, the model can be interpreted as a regression on S dummy variables where the sum of the coefficients is one. In the case of six seasons, the vector γ has dimensions 6-1. The first factor obeys the following mathematical equation (S. L. Scott, 2021):

$$\gamma_{t+1,1} = -\sum_{i=2}^{S} \gamma_{t,i} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2). \tag{3.5}$$

The last component of the BSTS model is a dynamic regression component. This component adds a dynamic regression component to the model in which the coefficients change over time in agreement with a random walk. This dynamic regression component is fit to the features that are engineered in this report.

To yield forecasts with the BSTS model, Bayesian model averaging is performed in the way described in Subsection 2.4.3. From the posterior distribution, 2000 draws are made and the forecasted value is precisely the average value of all the draws.

3.2.3 GAM

The GAMs are fit using the `mgcv` `R` package created by Wood (2011). To capture a seasonal component, a new variable for weekdays is created as by Posch et al. (2022). This variable will be referred to as *wDay* in chapter 4. The weekday variable is a categorical variable that is a number from one to six, since Saturdays are removed, indicating the weekday of the data-point. Then, a smoothing function is fitted to the weekday variable where the basis used for the spline as a cyclic cubic spline is specified. The cyclic cubic spline has a constraint that states that there should be no discontinuity between the endpoints of the spline (Simpson, 2022). It is therefore suitable for fitting a seasonal component.

Similarly, to fit a trend function to the data, the creation of a new variable is required. The trend component requires that the time t is given in numeric values and therefore a variable is created, similarly to Posch et al. (2022), called *time* that maps all the dates in our data set to real integer numbers. A smoothing function is then fit to the time variable to replicate a trend function. This is simply a function of time that is included in the sum of functions that the GAM uses to produce predictions. A smoothing function is also created for all of the regression variables, where the basis used is a Gaussian process.

3.3 Feature engineering

The data that is used as the basis for the regression components of the models is available shipment-specific data and event tracking data from Company X which is based on parcel delivered for Company G. The information that is present in the shipment information data set is summarized in Table 3.1. For event-specific data, the information can be found in Table 3.2. The tables are combined on the key variable *shipmentID* to link a specific shipment with its corresponding events. New features are then created for the times between events by subtracting the event time of a preceding event from its subsequent event. The thesis assumes that every shipment has the chain of events shown in Figure 1.1.

The complete list of features that are engineered from the raw data at hand is summarized in Table 3.3 with their corresponding descriptions. The features that are engineered are based on the following assumptions:

Table 3.1: Shipment information table.

Column name	**Type**	**Description**
ShipmentID	INTEGER	A unique identifier for a shipment.
ShipDateLocal	STRING	shipping date in local time.
ReceiverCity	STRING	Shipping destination city.
ReceiverCountry	STRING	Shipping destination country.
ReceiverZipCode	STRING	Shipping destination ZipCode.
CarrierName	STRING	Name of carrier performing shipment.
ETA	TIMESTAMP	Estimated time of arrival.
ATA	TIMESTAMP	Actual time of arrival at destination.
NUTS3	STRING	NUTS3 region code of shipping destination
DOT	INTEGER	classifying a shipment as on time (0) or late (1).

Table 3.2: Event tracking information Table columns.

Column Name	**Type**	**Description**
TrackingEventGlobalID	STRING	A unique identifier for a tracking event.
ShipmentID	INTEGER	A unique identifier for a shipment.
Description	STRING	Description of event.
Code	STRING	Event description code.
EventTimeUTC	TIMESTAMP	Registration time for event.
EventTimeLocal	STRING	Registration time for event in local time.

1. The variation in the amount of, both finished and unfinished shipments, that are currently burdening the transport network have predictive ability towards the share of late packages in the following time-period.

2. The daily variations in the average times between shipment events have predictive ability towards the share of late packages in the following time-period.

3. The resulting observations of all variables are assumed to be realizations from normal distributed variables.

The viability of the first two assumptions is examined by comparing the predictive ability of the BSTS model and the baseline ARIMA model. By including a spike-and-slab prior in the BSTS model, the significant variables in a predictive sense are found by examining which variables have the strongest inclusion probability in the resulting BSTS models. The viability of the last assumption is discussed in chapter 5.

Reading the Table 3.3 top-down, the first five variables represent shipment volumes in different phases of a shipment during the day preceding the forecast. The last four variables represent the average time between shipment events (TBE) a typical shipment registers during its journey from department to end-location.

After the creation of the predictor variables, the variables are scaled by transforming

Table 3.3: Regression variables with corresponding description.

Variable Name	Description
DepartedFromG	The volume of packages that registered the tracking event DepartedFromG.
ArrivalFirstHub	The volume of packages that registered the tracking event ArrivalFirstHub.
ArrivalDestinationHub	The volume of packages that registerd the tracking event ArrivalDestinationHub.
OutForDelivery	The volume of packages that registered the tracking event OutForDelivery.
Delivered	The volume of packages that registered the tracking event Delivered.
TBE1	The mean of the time duration between tracking events DepartedFromG and ArrivalFirstHub.
TBE2	The mean of the time duration between tracking events ArrivalFirstHub and ArrivalDestinationHub.
TBE3	The mean of the time duration between tracking events ArrivalDestinationHub and OutForDelivery.
TBE4	The mean of the time duration between tracking events OutForDelivery and Delivered.

them into observations from a standard normal distribution. This assumption is supported by the central limit theorem. The theorem states the following: for a sum of independent and identically distributed variables, each having a finite mean μ and a finite non-zero variance σ^2, then, the distribution of the standardized sample mean tends to the standard normal distribution (Montgomery, 2019).

By grouping shipments by day, these become daily standardized sample means after applying a standardization where the standard error and mean are estimated by taking the mean and standard error of the whole population. This transformation rests on the assumption that the times-between-events for shipments are independent and identically distributed.

3.3.1 Missing time series data

All regions do not have a complete time series of observations of the target variable for every day in the sequence. Therefore, before any fitting is done to the time series, the missing data has to be accounted for. The process of filling in missing data in a time series is called interpolation (Klosterman, 2019). In the case of time series, temporal methods may be used. The general idea with these methods is that a missing data point has presumably a value in-between adjacent data points. A large set of interpolation methods are available in both `NumPy` and `Pandas`, which are also used in this thesis to fill in missing values in the different regional time series. This is due to the simplicity and convenience of the approach. It is noteworthy that more sophisticated approaches to handling missing data exist where the problem of missing data is approached as a "problem within the problem", that is, to view the missing data as a predictive modeling task. This is referred to as model-based imputation by Klosterman, 2019.

3.3.2 Missing data statistics

Before missing values in the observed time series are filled with interpolation methods, some initial pre-processing is done. After inspection of the different regional time series with the resolution NUTS1, one can observe that in this particular case, the majority of missing time series data are over Saturday deliveries. One explanation for this is that the studied company does not perform deliveries on Saturdays. Another explanation is that the deliveries that are performed on Saturdays are only registered during other days of the week, for example on Sundays or Fridays. The share of missing data in the various observed time series are presented in Table 3.4.

As one can observe, a large share of the missing time-series data occurs on dates that are Saturdays. Therefore, before any interpolation methods are utilized to handle the missing values, Saturdays are removed from the observed time series. The practical usefulness of this procedure is that instead of obtaining $\sim$ 15%artificially interpolated data in the regional time series after filling in missing values, $\sim$ 0.87% of the data is removed in the time series. Now, in the resulting regional time series, only $\sim$ 1.3% of the data has to be filled in by using interpolation methods. In other

Table 3.4: Share of missing time-series data.

	Mean	Standard Deviation
Share of Saturdays in regional time-series data	0.87 %	0.22 %
Share of Saturdays in missing regional time-series data	92.56 %	2.28 %
Total share of missing data when **including** Saturdays	14.67%	0.46%
Total share of missing data when **excluding** Saturdays.	1.29%	0.43%

words, if Saturdays are present in the resulting time series of observations that is to be modeled, these values need to be filled in. If instead Saturdays are removed and the time series only consists of the remaining six weekdays, less values need to be filled in. The argument for implementing this data cleaning step is that the resulting time series models will make better predictions but also be more grounded in reality. This is due to the hypothesis that more artificial data will dilute the information embedded in the actual data.

3.4 Model evaluation methods

Evaluation of data science models can be carried out in several ways. One way to achieve this is by implementing out-of-sample prediction; that is, to forecast values of future days using the trained models to predict beyond the training sample, and comparing the results of the forecasts with actual values from a test set. The performance of different models is compared by firstly calculating some kind of statistical measure, and secondly by comparing the magnitude of said statistical measure. Hyndman and Koehler, 2006 discusses several different statistical measures to use for evaluating the accuracy of time-series forecasts. These are: root mean squared error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE), and mean absolute scaled error (MASE). MAPE is useful when the scales or the unit of measure of the time series vary. Since the measure takes into account the size of the target variable to scale the result, it must be non-zero in every case. Since this is not true in our case, the measure is omitted since it would result in undefined values. This leaves us with three statistical measures for accuracy: the RMSE, the MAE, and the MASE. The quality of the forecasts is evaluated in two differing ways: The first one is to split the data into one training set and one test set for every region. Then, the three different models are fitted to the training set. Next, ten future forecasts are generated and compared with the actual values (in the test set) by using the chosen statistical measures.

The second way of evaluation is by the use of time series cross-validation. The time series cross-validation methodology is summarized in Figure 3.5. The prediction accuracy is evaluated by a procedure sometimes called “evaluation on a rolling

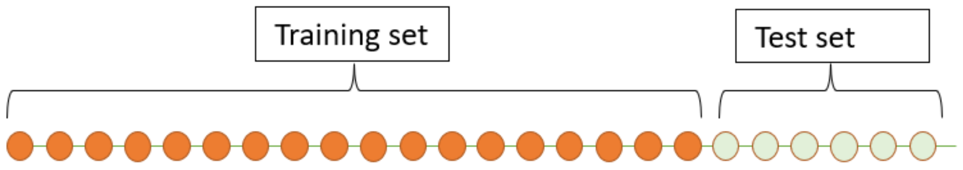


Figure 3.4: Traditional train and test split.

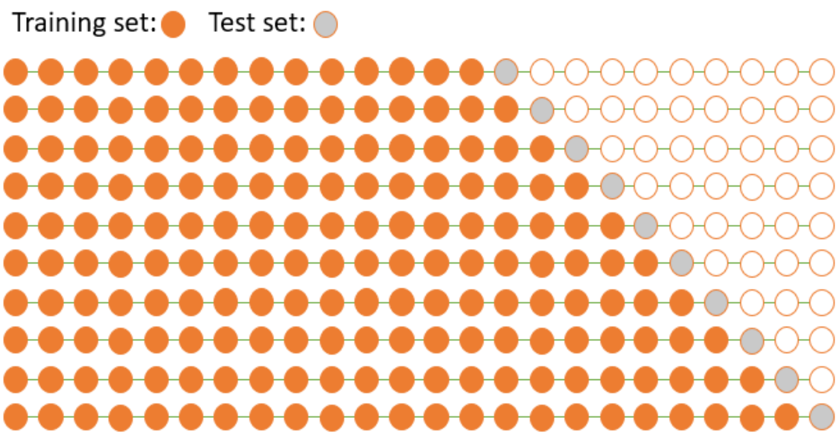


Figure 3.5: Train and test split for cross validation.

forecast origin" (Hyndman, 2022). The method works by averaging over forecasts each with a horizon of size one, where one starts by fitting the model to a part of the data, and then producing a forecast. Next, the forecast "rolls forward" in time by one step and the training set increases its size by one. This procedure is then repeated ten times and finally, the prediction accuracy is averaged over all of the fitted models' forecasts. Since the size of the forecasting horizon is one, The RMSE and MAE will show the same result because the RMSE for a sample of size one is exactly the MAE (eq. 3.6 and eq. 3.7). Since the third statistical measure, MASE is a relative measure, it cannot be used in this case (Hyndman and Khandakar, 2008). Therefore, the statistical measure that will be used for evaluating the cross-validation folds is RMSE/MAE.

3.5 Statistical Measures

The definitions of the three statistical measures used will be elaborated upon in this subsection.

3.5.1 Root mean squared error

Historically, RMSE, has been very popular. This is largely due to its relevance in statistical modeling. However, it has been criticized for its low reliability (Hyndman and Koehler, 2006; Armstrong et al., 1992). Despite the criticism, RMSE will be included as one of the statistical measures here due to its widespread adoption and understandability. The RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y_j})^2}. \tag{3.6}$$

Where $\hat{y_j}$ is the j-th predicted value, y_j is the j-th actual value and n the number of observations.

3.5.2 Mean absolute error

MAE is a statistical measure that also has seen widespread adoption in the domain of statistical modeling (Hyndman and Koehler, 2006). MAE is in concept simpler and has an easier interpretation than the RMSE. In a time-series plot, it is simply the average absolute distance between each actual value and the predicted value at any given time. It is less sensitive to outliers in predictions in contrast to the RMSE, in which outliers tends to increase the RMSE score since the results are squared (Willmott and Matsuura, 2005). The MAE is defined as:

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y_j}|. \tag{3.7}$$

Where $\hat{y_j}$ is the j-th predicted value, y_j is the j-th actual value and n the number of observations.

3.5.3 Mean absolute scaled error

MASE is a statistical measure also discussed by Hyndman and Koehler (2006). The authors considers it the best available measure for forecasting accuracy. Measures based on relative forecasting errors make an attempt to remove the scale of the data by putting the forecasts in relation to some other forecast method that is used as a benchmark. Usually the other forecast method is a naive method. The naive forecast method implemented in this thesis is to simply forecast the value at time t as the actual value at time $t-1$. In other words, the next forecasted value is the current measured value. The MASE values are also easy to interpret. A value that is larger than one performs on average worse than the naive method and conversely a value lesser than one indicates that the model performs better on average than the naive method. A drawback of relative measures is that they can only be computed when several forecasts are made on the same series. Therefore, it is not possible to use MASE to measure out-of-sample prediction error when the forecast horizon is only a single sample (Hyndman and Koehler, 2006). When scaling the errors by the in-sample MAE from the naive method, the scaled error is defined as:

$$q_t = \frac{y_j - \hat{y_j}}{\frac{1}{n-1}\sum_{i=2}^{n}|Y_i - Y_{i-1}|}. \tag{3.8}$$

Where the numerator is the MAE of the forecasted value for given period and the denominator is the MAE of the naive method for given period. The MASE is defined as:

$$MASE = mean(|q_t|). \tag{3.9}$$

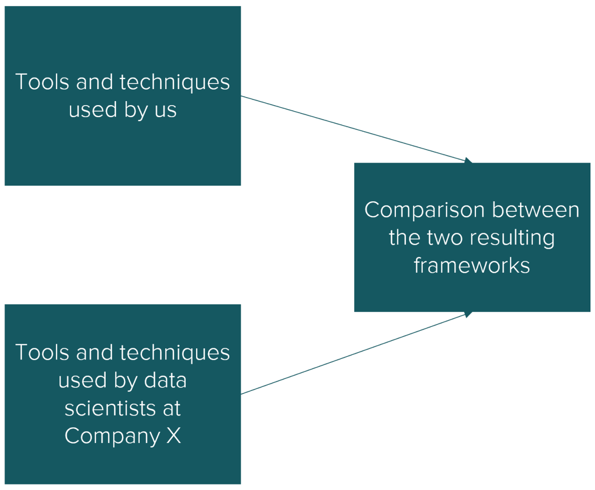

Tools and techniques used by us
Comparison between the two resulting frameworks
Tools and techniques used by data scientists at Company X

Figure 3.6: Comparing tools and techniques used by us with what is used by data scientists at Company X.

3.6 Implementing a software engineering framework

To answer research question 2 and 3, a workflow will be followed which is shown in Figure 3.6.

The first step is to create a framework where tools and techniques utilized during the model development phase are connected to each software engineering principle. This framework will be referred to as framework 1. The purpose for using the tools and techniques is to ensure that the software engineering principles discovered during the literature review are followed. They specific way that these principles are implemented in model development is outlined in the framework and with what tool or technique they are achieved.

The second step is to investigate tools and techniques utilized by the data scientists at Company X. In the same manner as in the previous step, a framework referred to as framework 2 is developed connecting software engineering principles to tools and techniques used by data scientists at Company X. It is important to first develop framework 1 before developing framework 2 since the tools used in framework 1 will be purely based on tools and techniques found in the literature review and these should not be influenced by the current standard practices of data scientists at company X. Framework 2 will be developed by conducting individual interviews with data scientists at Company X.

The last step is to compare framework 1 to framework 2 by discussing similarities and differences. Software engineering principles might be fulfilled differently in the two frameworks. The discussion's main purpose is to find gaps between the theory of this thesis and how Company X currently operates. This step provides a discussion of application possibilities of software engineering principles in a real-world environment. Verifying the application of software engineering principles in real data science projects is lacking in current research according to Al-Sarayreh et al. (2021).

3.6.1 Understanding the current data science development process

As mentioned in the previous section, the current data science development process at Company *X* needs to be mapped out to create framework 2. According to Stol and Fitzgerald, 2018, interviews are a good method to carry through a field study research. Another common technique is field observations. Due to time constraints, field observations will not be used in this study.

The purpose of the interviews is to understand the current model development processes of data science teams at Company *X*. Firstly, an understanding of whether the data scientists at Company *X* are aware of the eight suggested software engineering principles is studied, and if so, what tools or techniques they use to follow said principles.

Furthermore, a basic description of the background of each interview subject will be described by the interviewees to nuance reasoning as to why different data scientists have a different view on software engineering principles or why they use different tools or techniques to achieve software engineering principles.

The interviews will be semi-constructed, meaning that the interview will not be constrained to pre-determined questions. The purpose of the interview questions is to set a guideline for the interview (Bell et al., 2019). The reason why semi-constructed interviews will be used is because the interviews aim to gather opinions and thoughts on proposed software engineering principles. Furthermore, the interviewees will be asked to reflect on these discussing which tools they use to achieve said software engineering principle. Opinions are more easily expressed in a semi-constructed interview in comparison to a constructed interview format (Bell et al., 2019).

The interview subjects were selected by asking all of the eight employees at Company *X* with data scientist as their title, and interviews were held with the available subjects. The study chooses to not interview any data scientist with high seniority since their tasks fall more under project management rather than actual model development. Out of the eight data scientists, six were available for interviews during the time of asking. Since six out of eight data scientists were interviewed at Company X, the chosen sample can be described as saturated for the given context. In other words, all potential interview subjects were requested to participate in an interview.

4 Results

This chapter outlines the results of the thesis. The results of applying ARIMA, BSTS, and, GAM in the given context with model performance statistics are presented along with used tools and techniques when applying software engineering principles. Results of the conducted interviews are also presented.

4.1 Time series models

In the following section, a few illustrative time series are plotted. when creating individual time series for specific pairs of region and service type, nine of the time series are found to only have data for the first two to five months. These are omitted from the study resulting in a total of 23 time series to examine. Every omitted time series is of the service type *A*.

The regional variability for share of late packages for service type *B* is very low and is shown in Table 4.1. The mean for share of late packages is around 0.09 for service type *B*. The regional variance of service type *A* is larger, shown in Table 4.2. The regions DE8, DE9 and, DEA have distinctively higher mean for share of late packages than the rest of the regions with service type *A*. The regions with high mean share of late packages are close to each other, located in the north west parts of Germany.

The printed graphs have the same features. The graphs are cut-off at day 150 and day 270 for illustrative purposes. The models are fitted to the first 260 observations of the various time series. Forecasts are then made for the next consecutive ten days. The red line represents the actual data and the blue dashed line represents the fit of the models. The dotted vertical line indicates where predictions start. Note that the studied data contained values for a year (365 days), beginning in January 2021 and ending in December 2021. The given data is preprocessed by removing Saturdays meaning the input data into our models has around 310 days. Predictions are made on day 260 since the study wanted to avoid making predictions in December, where the data is irregular due to holiday seasons. Instead, predictions are made on dates that correspond to the end of October or the beginning of November.

Table 4.1: Table containing the mean for share of late packages for service type B.

Region	**Mean share of late packages**
DE1	0.0901
DE2	0.0851
DE3	0.0751
DE4	0.0885
DE5	0.0990
DE6	0.0878
DE7	0.0787
DE8	0.0960
DE9	0.0810
DEA	0.0850
DEB	0.0905
DEC	0.0744
DED	0.109
DEE	0.0891
DEF	0.0550
DEG	0.0883

Table 4.2: Table containing the mean for share of late packages for service type A.

Region	**Mean share of late packages**
DE1	0.122
DE7	0.0400
DE8	0.224
DE9	0.191
DEA	0.213
DEB	0.0554
DEF	0.0511

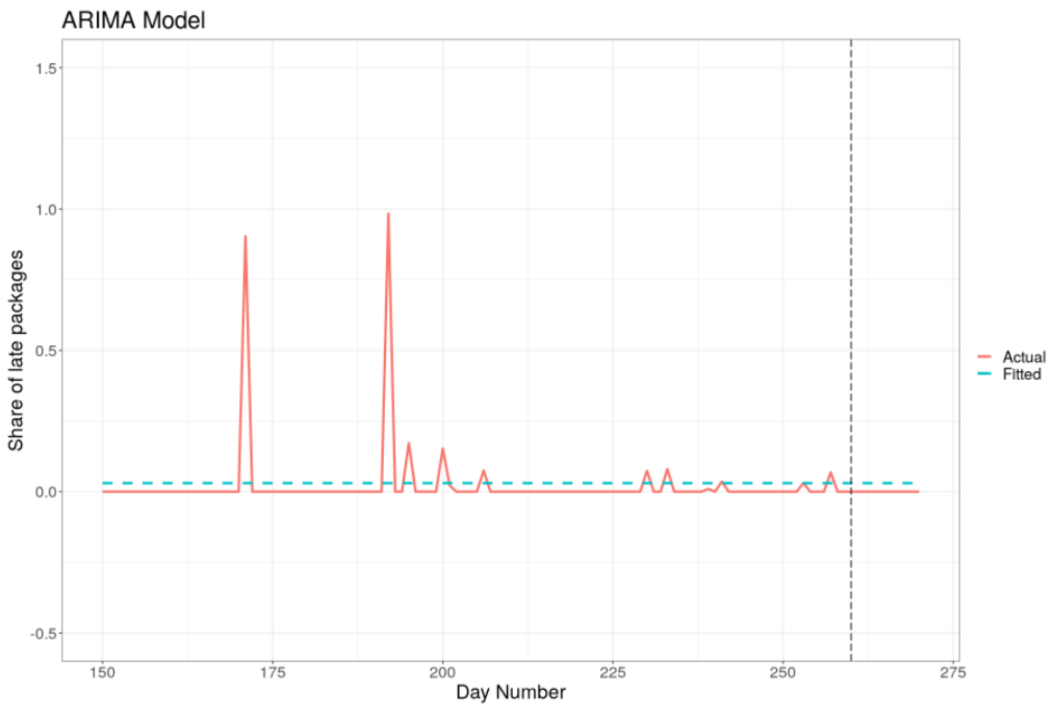


Figure 4.1: ARIMA model results for region DE1 with service type *A*.

4.1.1 Models in Region DE1 with service type A

For many regions, the typical behavior of the time series is a share of late packages around zero with occasional spikes of late packages as seen in Figure 4.1, 4.2 and 4.3.

For these cases, the ARIMA model does not catch any seasonality or trend as seen in Figure 4.1 and only fits a straight line based on the mean of the time series to minimize its predicted errors. The chosen ARIMA model is an ARIMA(0,0,0) with a non-zero mean, meaning that the chosen parameters in the model consist of only a constant mean, putting no weight on the autoregressive part or the moving average part.

Neither does the BSTS model catch the spikes, despite it not fitting a straight line. This shows that the regression variables have an effect on the model but as shown in Figure 4.2, the spikes of the observed time series are never captured by the model.

The same problem arises when fitting the GAM model which shows a poor fit for the spikes as seen in 4.3. The GAM model does not have any significant variables, which in this case is defined as a P-value below 0.05.

The RMSE and MAE score is the best for the BSTS model as seen in Table 4.3. GAM scored better than the ARIMA model with respect to the MAE measure but worse with respect to the RMSE measure.

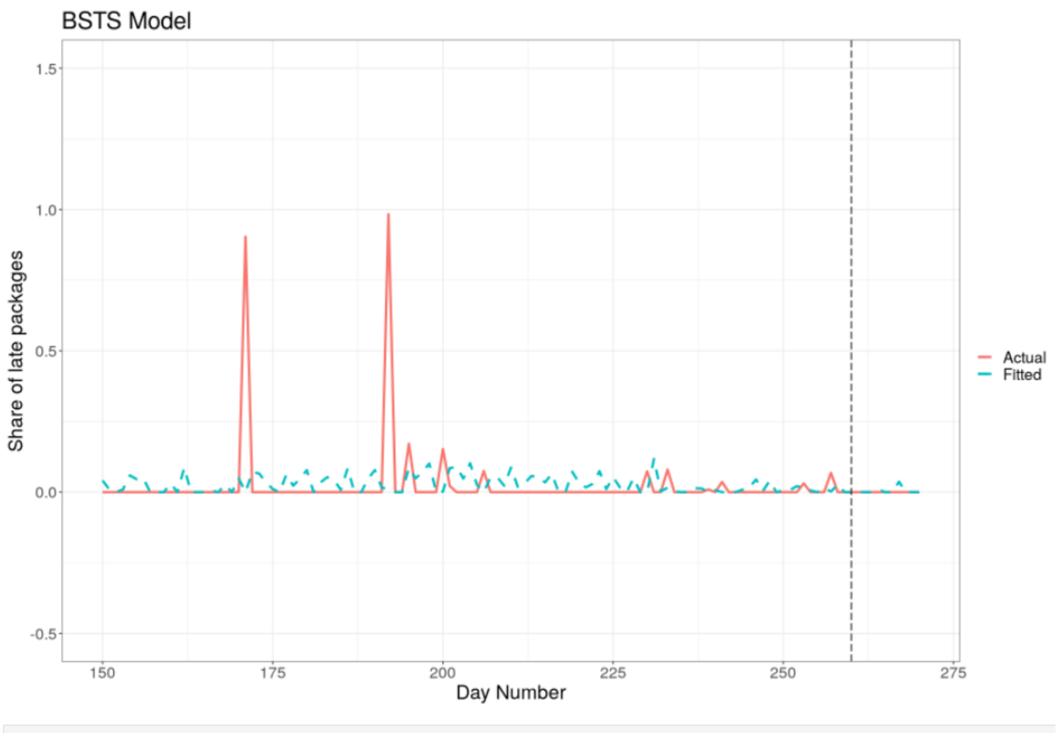


Figure 4.2: BSTS model results for region DE1 with service type A.

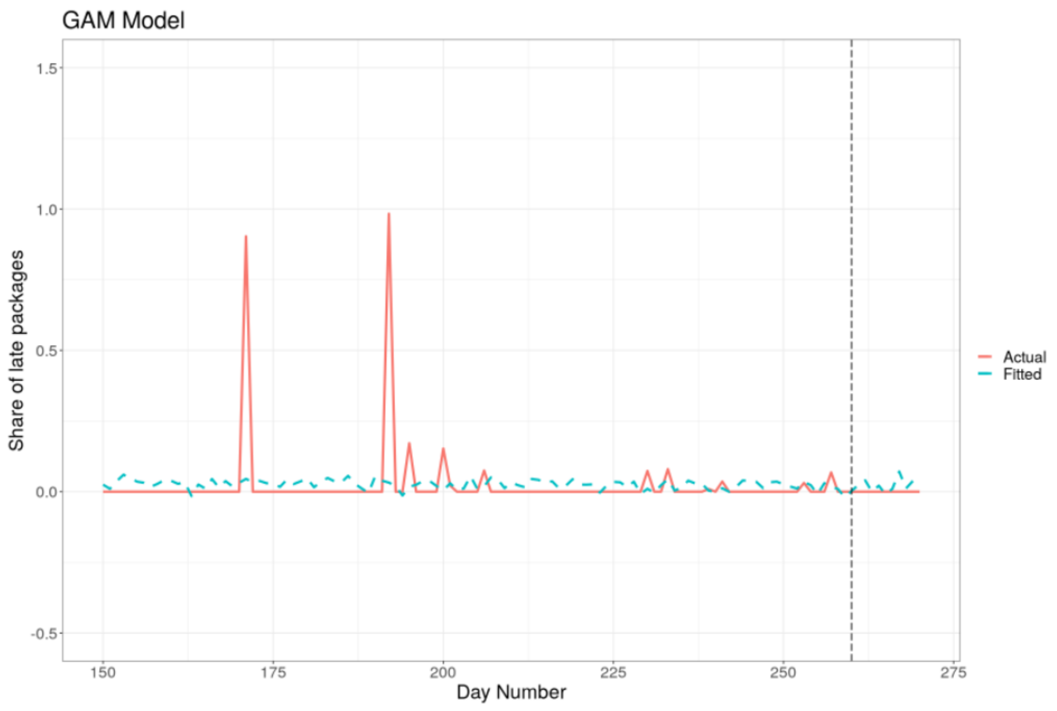


Figure 4.3: GAM model results for region DE1 with service type A.

Table 4.3: Table containing the the statistical measurements for region DE1 with service type A.

Region DE1 with service type Service type A			
Model	RMSE	MAE	MASE
ARIMA	0.0302	0.0302	inf
BSTS	0.001	0.001	inf
GAM	0.034	0.026	inf

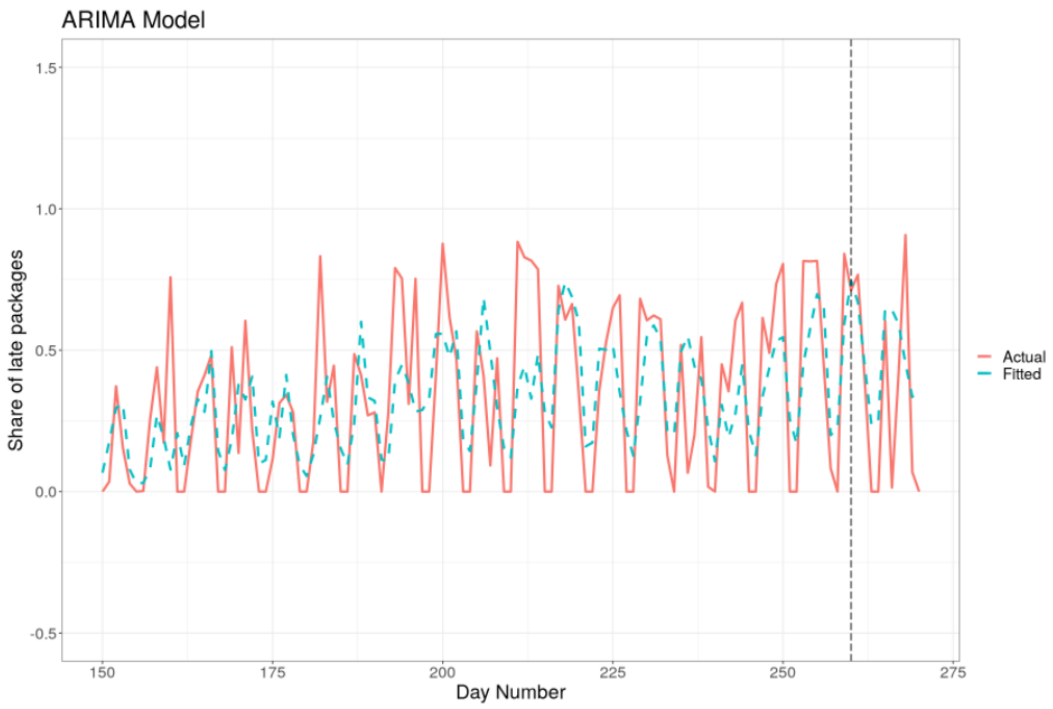
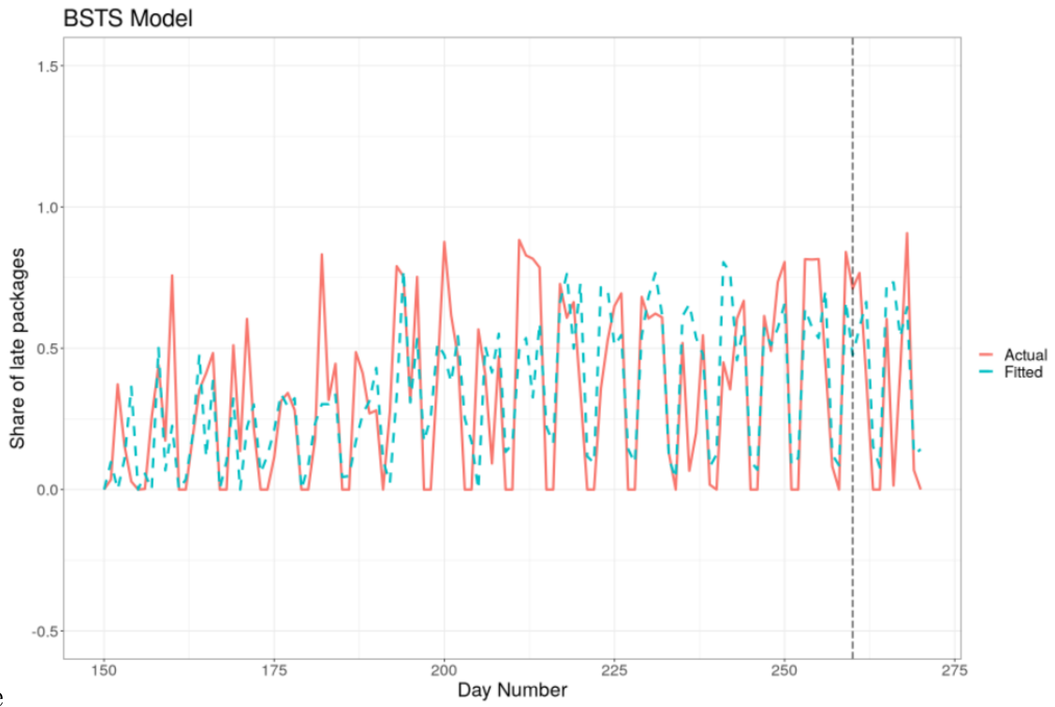


Figure 4.4: ARIMA model results for region DEA with service type A.

4.1.2 Models in region DEA with service type A

In a few regions, the share of late packages is consistently high across the whole observed period. An example of a region that displays this kind of behavior is DEA as seen in Figures 4.4, 4.5 and, 4.7. It is clear that these time series are better-behaved for modeling purposes. They display a seasonal component and in general more variation in the data. Since these time series contain more variation, the models are capable of fitting this variation in greater quality.

The chosen ARIMA model for this pair of region and service type is ARIMA(5,1,1). When observing the plot of the ARIMA model (Figure 4.4), one can see that the model captures the seasonality in the data, but the model has a clear upward going trend despite the data being differenced once in the chosen model. Therefore, the ARIMA model predicts increasingly larger values further along the time-axis.



e

Figure 4.5: BSTS model results for region DEA with service type *A*.

The BSTS model obtains the best fit to the time series in this case in terms of MAE, RMSE, and MASE (Table 4.4). The BSTS model also has an upward trend component (Figure 4.6), but in contrast to the ARIMA model, it also combines a dynamic regression component when producing its predictions. This results in predictions that are more reliable and accurate. The seasonal component seems to compensate for the upward direction in the trend component and therefore, the BSTS model is capable of producing predictions that are alternating between lower values and higher values in contrast to the ARIMA model which produces consistently high values in its predictions.

The four different components of the BSTS model that are combined to produce the predictions are displayed in Figure 4.6. As mentioned, the trend component has an upward trend with respect to the time axis. The seasonal component has a periodicity of 6, which corresponds to the six different days of the week that are included in the time series (since Saturdays are omitted). It is dynamic and therefore changes over time. It starts with a rather small amplitude, which then grows with time, similar to the actual time series. The Regression component displays the state contribution of the regression part of the fitted model. The Auto regressive component displays an AR(3) process which the `Auto.Ar` function has decided upon. This process takes into account the value of the target variable from the three preceding days when producing its contribution to the final predicted value.

A visualization of the GAM's fit to the time series is displayed in Figure 4.7. As one can observe, the GAM captures a seasonal component of the data together with an upward trend. However, since the seasonal component is static and not changing

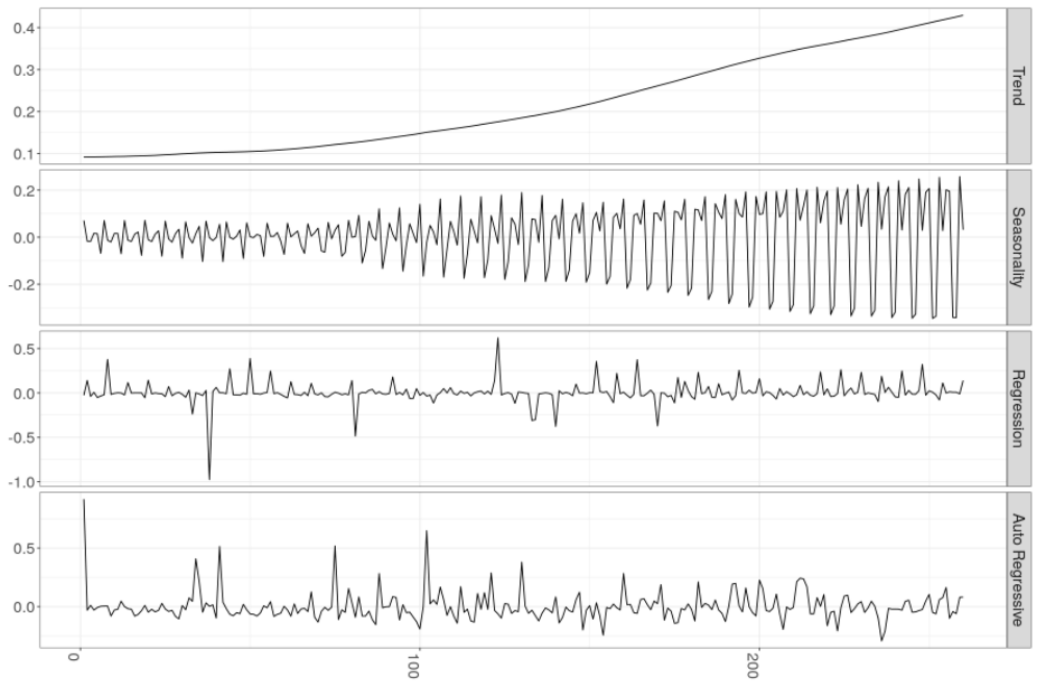


Figure 4.6: Components of the fitted BSTS model over region DEA with service type *A*.

Table 4.4: Table containing the the statistical measurements for region DEA with service type *A*.

Region DEA with service type Service type *A*			
Model	RMSE	MAE	MASE
ARIMA	0.311	0.256	0.612
BSTS	0.226	0.199	0.477
GAM	0.283	0.258	0.618

as in the case of the BSTS model, it fails to predict high and low values with the correct amplitude. Instead, it fits a constant seasonal component that results in underwhelming predictions for both high and low values. The additional smoothing functions that are fit to the predictor variables seem to be of little to no help in aiding the GAM to produce accurate predictions. Only the *wDay* variable and *time* variable representing a seasonal and trend component where assigned a significant P-value.

4.1.3 Models in Region DE7 with service type B

Figure 4.8, 4.9 and, 4.10 illustrates the models created for region DE7 with service type *B*.

As for the case with region DE1 with service type *A*, the ARIMA model is more or less a straight line. This time, the chosen model is an ARIMA(2,0,0) with a non-zero

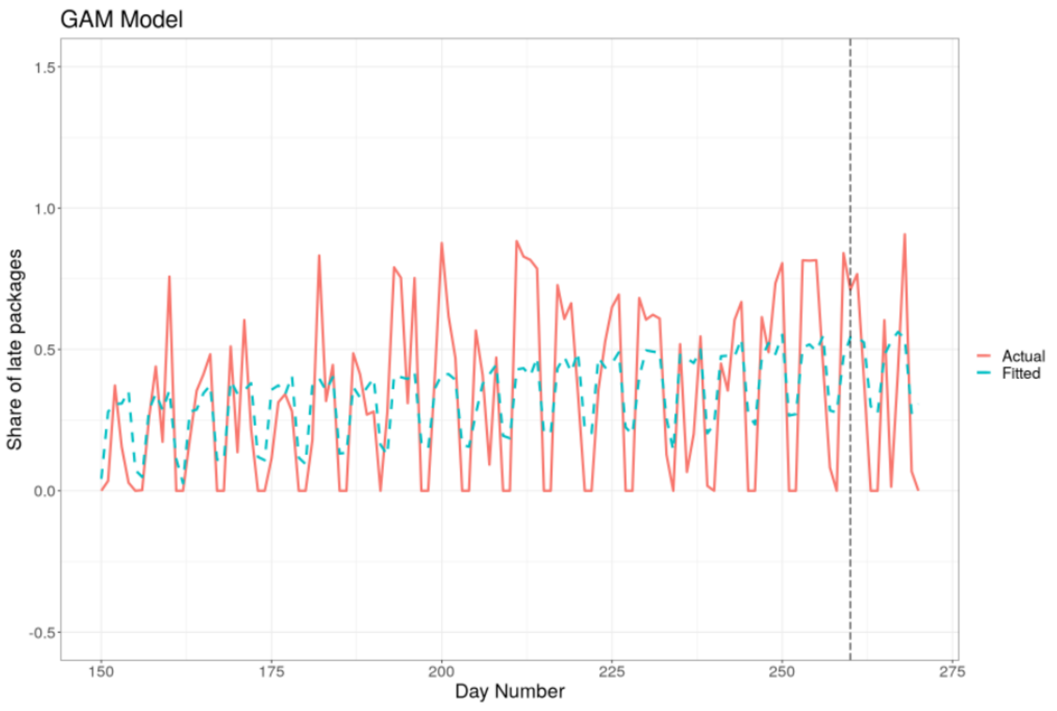


Figure 4.7: GAM model results for region DEA with service type *A*.

Table 4.5: Table containing the the statistical measurements for region DE7 with service type DPDf Priority.

Region DE7 with service type Service type *B*			
Model	RMSE	MAE	MASE
ARIMA	0.018	0.014	0.715
BSTS	0.013	0.010	0.499
GAM	0.020	0.016	0.816

mean, meaning it contains an Auto regressive part and a mean value. As seen in Figure 4.8, the fitted line is not completely straight and dips at around day 170. Although it chooses an ARIMA(2,0,0) model, its predictions are the worst out of the studied models for this region and service type.

The BSTS model on the other hand manages to predict a spike at around day 265 as the actual data shows in Figure 4.9.The region DE7 with service type DPD Parcel Connect illustrates that the BSTS model might have the best predictive ability of irregular spikes due to its regression variables.

The GAM model also performs poorly for the given region and service type shown in Figure 4.10. As seen in Table 4.5, the GAM model performs the worst for the given region service type.

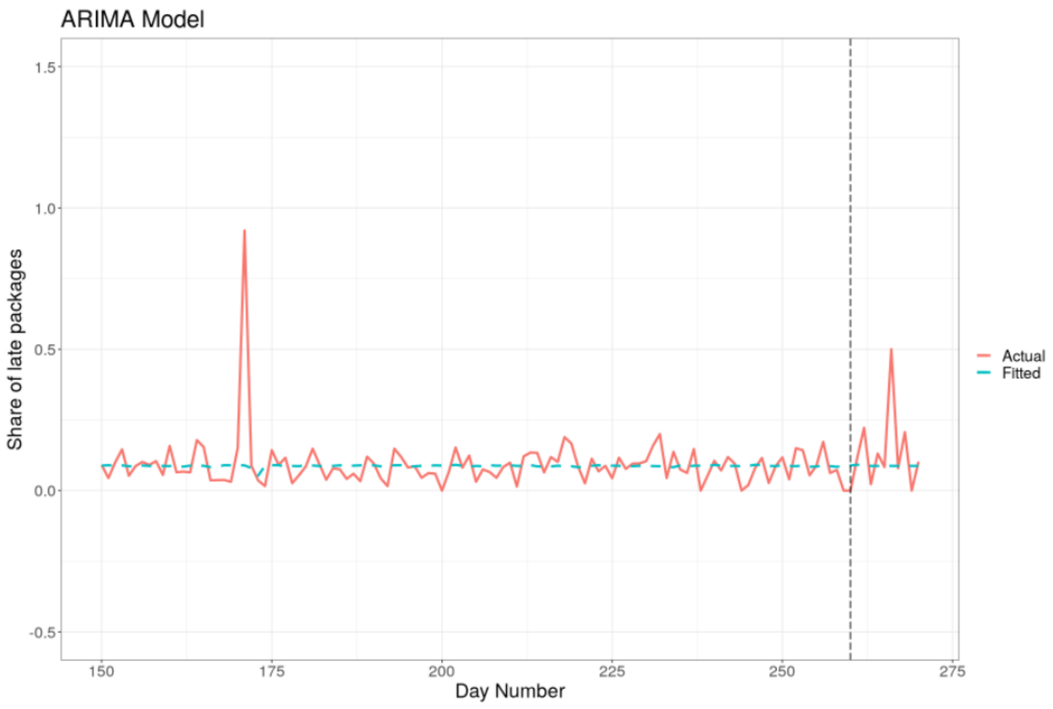


Figure 4.8: ARIMA model results for region DE7 with service type *B*.

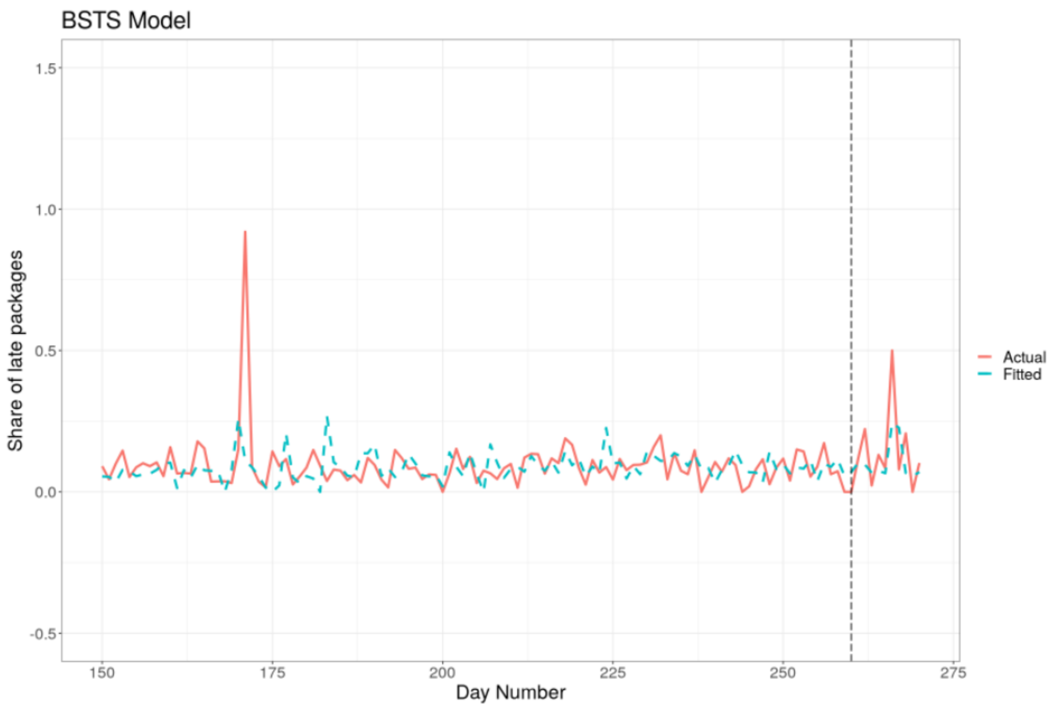


Figure 4.9: BSTS model results for region DE7 with service type *B*.

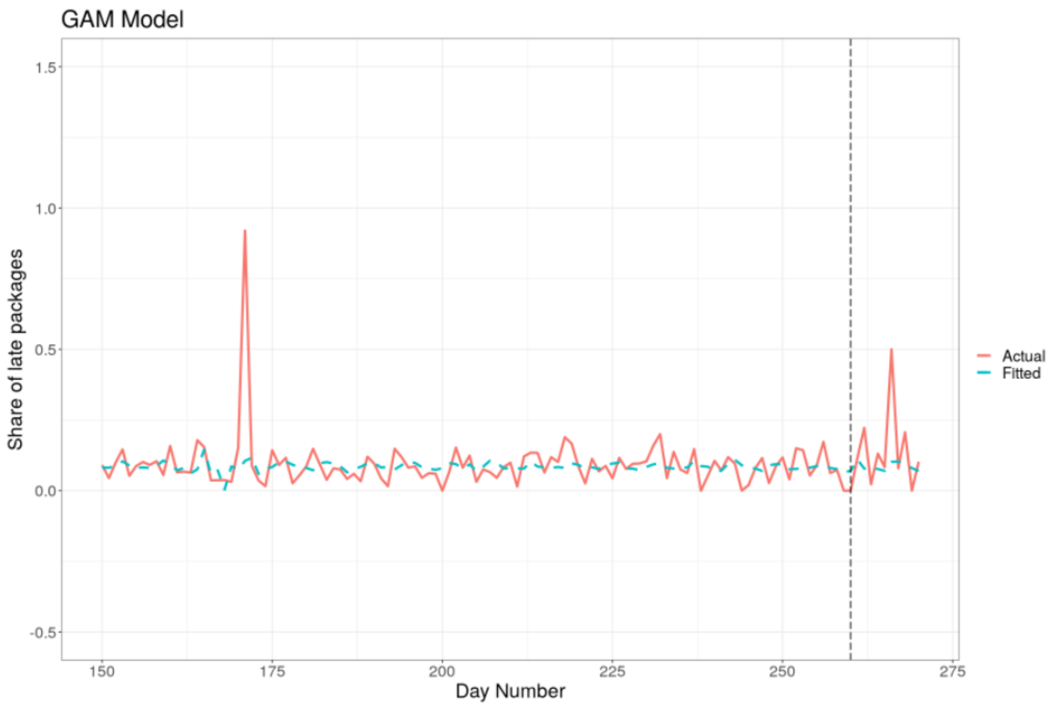


Figure 4.10: GAM model results for region DE7 with service type B.

4.2 Model evaluation

The three different models are evaluated, as previously stated, by comparing the three different statistical measurements RMSE, MAE, and MASE. The results of the fitted models and their accuracy are summarized in Table 4.6. These tables show the statistical measures including all service types, resulting in a count of 23 regions with service type time series. There are a total of 16 NUTS1 regions and two service types, service type A and service type B. This means that nine time series regions with specific service types are excluded because these time series resulted in incomplete data, meaning data were only recorded for short periods. The result is based on a ten-day rolling forecast of future days values, which are compared to actual observations from the test set. As seen in Table 4.6, the statistical measurements all show that BSTS is the best performing model, followed by GAM and lastly ARIMA.

The results can be granulated further by dividing the data into service types. Table 4.7 shows the statistical measurements for the service type A. This resulted in a count of 7 time series regions. As seen in Table 4.7, GAM gets marginally the best RMSE score, followed by BSTS and lastly ARIMA. For the measurements MAE and MASE BSTS is the best performing model, followed by GAM and lastly ARIMA.

The results for the service type B are displayed in Table 4.8. The results are based on measurements of 16 time series. All NUTS1 regions had complete time series data for the service type B. As seen in Table 4.8, the statistical measurements RMSE, MAE, and MASE all show that BSTS is the best performing model, followed by ARIMA and in last place GAM.

Table 4.6: Result of statistical measurement for all service types combined.

All service types				
Statistical measurement		ARIMA	BSTS	GAM
RMSE	Mean (across regions)	0.069	0.064	0.067
	Aggregated (all regions)	1.590	1.473	1.549
MAE	Mean (across regions)	0.055	0.045	0.054
	Aggregated (all regions)	1.268	1.034	1.247
MASE	Mean (across regions)	0.896	0.573	0.851
	Aggregated (all regions)	18.823	12.037	17.880

Table 4.7: Result of statistical measurement for service type *A*.

Service type *A*				
Statistical measurement		ARIMA	BSTS	GAM
RMSE	Mean (across regions)	0.152	0.139	0.136
	Aggregated (all regions)	1.064	0.974	0.954
MAE	Mean (across regions)	0.126	0.096	0.113
	Aggregated (all regions)	0.879	0.671	0.789
MASE	Mean (across regions)	1.746	0.533	0.953
	Aggregated (all regions)	8.731	2.664	4.763

Another, more rigorous, way of measuring the performance of the three models is by studying the cross validation score of the different models and service types shown in Table 4.9. The results of the cross validation show that for service type *B*, BSTS is the best performing model followed by ARIMA and lastly GAM. For the service type DHL Service Connect, BSTS is once again the best performing model, followed by GAM and lastly ARIMA.

Table 4.8: Result of statistical measurement for service type *B*.

Service type *B*				
Statistical measurement		ARIMA	BSTS	GAM
RMSE	Mean (across regions)	0.033	0.031	0.037
	Aggregated (all regions)	0.526	0.499	0.595
MAE	Mean (across regions)	0.024	0.023	0.029
	Aggregated (all regions)	0.388	0.362	0.458
MASE	Mean (across regions)	0.631	0.586	0.820
	Aggregated (all regions)	10.092	9.373	13.117

Table 4.9: The aggregated cross validation (CV) score for all service types, service type A and service type B.

CV score results			
Model	All service types	Service type B	Service type A
ARIMA	12.884	3.872	9.012
BSTS	10.226	3.490	6.737
GAM	12.553	4.530	8.024

Table 4.10: A Table showing the 23 chosen ARIMA models.

	Chosen model	Number of times chosen
1	ARIMA(2,0,0) with non-zero mean	13.000
2	ARIMA(0,0,0) with non-zero mean	4.000
3	ARIMA(1,0,0) with non-zero mean	2.000
4	ARIMA(5,1,0)	2.000
5	ARIMA(1,1,1)	1.000
6	ARIMA(5,1,1)	1.000

4.3 Feature importance

Table 4.10 shows the different chosen ARIMA models for our different time series. As seen in the table, ARIMA(2,0,0) is chosen for half of the time series. Generally, the `auto.arima()` method chooses to put weight on the auto-regressive part of the model rather than the moving average part for our given case. The less informative ARIMA models, that is, the ones with only auto regressive components e.g. ARIMA(0,0,0) and ARIMA(2,0,0) are chosen in regions where the variability in the target variable is low. The models with a greater number of parameters are chosen in cases where the regional time series exhibit greater variance (e.g. ARIMA(5,1,1)).

In Table 4.11, features chosen by the BSTS models are summarized and assigned a score. The score is simply the mean inclusion probability times the number of times the variables have been chosen by the various BSTS models. The Count represents the number of models which has the given feature in its top five most important features. The variable that gets the highest score is $ArrivalFirstHub$, that is, the volume of packages that registered the event $ArrivalFirstHub$ the previous day. In places 2-5 the study finds mixed effects between the various TBE variables (e.g. $TBE2 : TBE3 : TBE4$ corresponds to the product of the three variables). The mixed-effects variables indicate that when the mean time between several events are high or low simultaneously, they together display some correlation with the target variable. This indicates that the TBE variables are more informative when they all demonstrate a deviation from the mean value simultaneously, but that alone, the variables might not be as informative.

To quantify the significance level of the features created for the GAM models in

Table 4.11: Regression variables for the created BSTS models.

	Feature	Inc.probability	Count	Score
1	ArrivalFirstHub	0.987	5.000	4.933
2	TBE2:TBE3:TBE4	0.641	6.000	3.844
3	TBE3	0.395	9.000	3.557
4	TBE1:TBE2:TBE4	0.353	10.000	3.534
5	TBE1:TBE2:TBE3	0.504	7.000	3.530
6	OutForDelivery	0.349	9.000	3.141
7	TBE1:TBE2:TBE3:TBE4	0.410	7.000	2.869
8	TBE3:TBE4	0.675	4.000	2.698
9	Delivered	0.366	7.000	2.563
10	TBE2:TBE4	0.629	4.000	2.517
11	TBE2:TBE3	0.481	5.000	2.403
12	TBE4	0.479	5.000	2.394
13	TBE1:TBE3:TBE4	0.348	6.000	2.087
14	TBE1	0.695	3.000	2.084
15	ArrivalDestinationHub	0.227	8.000	1.815
16	DepartedFromG	0.538	3.000	1.615
17	TBE1:TBE3	0.261	6.000	1.566
18	TBE1:TBE4	0.256	5.000	1.282
19	TBE1:TBE2	0.425	3.000	1.276
20	TBE2	0.304	3.000	0.913

the report, The significant variables in the models are observed and analyzed. In Table 4.12 The significant variables in all GAMs are summarized. The Count column displays in how many models a feature is assigned a significant p-value. The variable *wDay* has significance *** (P-value < 0.001) in four of the fitted time series. This is the corresponding seasonal component in the fitted GAMs. The variable *time* in Table 4.12 corresponds to the trend component in the GAMs and has a significance level *** in three regional time series. The predictor variable that is chosen the largest number of times as significant in the GAMs is $TBE2$, which is chosen by three GAMs as significant, once for every significance level. $TBE2$ is the mean time between events $ArrivalFirstHub$ and *ArrivalDestinationHub*. Other predictor variables that are chosen in the various GAMs are $TBE3$, $ArrivalFirstHub$, $TBE1$, and $TBE2$.

4.4 Applying software engineering principles in data science programming

A framework for scientific programming has been brought forward during this thesis which connects software engineering principles to tools and techniques possible to implement in practice. The framework is based on the eight principles presented by Wilson et al. (2014). In Table 4.13 one can observe the tools and techniques implemented during the model development phase of this thesis. Table 4.14 shows

Table 4.12: Table containing variable significance scores in fitted GAMs.
*: P-value < 0.05, **: P-value < 0.01, ***: P-value < 0.001.

Variable	Significance	Count
s(wDay)	***	4
s(TBE2)	*, **, ***	3
s(time)	***	3
s(TBE3)	***	2
s(ArrivalFirstHub)	*	1
s(TBE1)	*	1

the tools and techniques used by data scientists at Company X.

4.4.1 Created software

Before creating the final software, a data exploration phase was conducted to understand the data. The created files for the data exploration were temporary. Code snippets were moved to the final software if found relevant. The final software consists of three different software files that were created to analyze ARIMA, BSTS, and GAM models and their performance in forecasting shipment delays for Company G. The three files created were:

- Time series data handling module in `Python 3`.
  - Creates time series data objects with built in methods to analyze and preprocess data.

- ARIMA, BSTS and, GAM models module in `R`.
  - Creates ARIMA, BSTS and GAM models with the help of the `R` packages `bsts`, `mgcv`, and `forecast` which includes the `auto.arima()` function.
  - Visualizes results.

- `Jupyter` notebook.
  - Retrieves and reads BigQuery data.
  - Calls on the created modules to produce results.
  - Analyze resulting forecasting models.

Figure 4.11 shows the workflow of extracting raw data from Company X's database to create and analyze the models.The reasoning behind creating a `Python` module for handling and preprocessing data instead of creating these features directly in the notebook is to achieve software modularity. A standardized package was created which could also potentially be reused by Company X in preprocessing data for time series analysis in other contexts.

A module in `R` was also created which runs the various `R` packages to create ARIMA, BSTS and GAM models. This module was created in `R` simply due to `R` having more resources for the studied models. Model visualization was also chosen to be done in

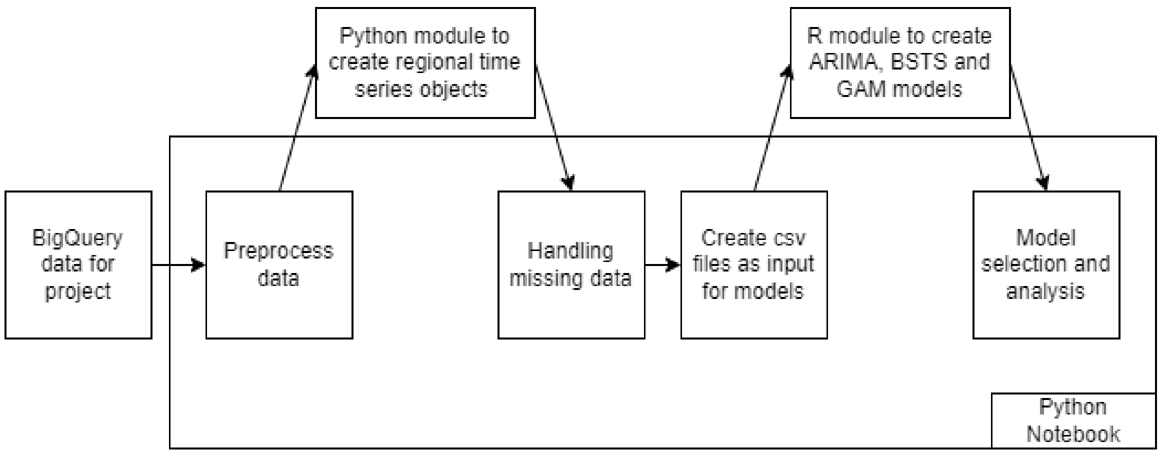

Python module to create regional time series objects
R module to create ARIMA, BSTS and GAM models
BigQuery data for project
Preprocess data
Handling missing data
Create csv files as input for models
Model selection and analysis
Python Notebook

Figure 4.11: Flowchart of created software.

the `R` module. The `Jupyter` notebook created aims to produce a logical workflow calling on desired modules when needed and producing final results. It acts as a bridge between the different modules created and Company X's internal systems.

4.5 Software engineering principles followed in this thesis when developing the data science models

Table 4.13 summarizes the tools and techniques which were utilized by the thesis writers when developing data science models. An in-depth result of what tools and techniques were implemented and which were not used follows in the subsections below.

4.5.1 Consistent coding

Consistent coding was mainly achieved by learning and implementing the `PEP8` framework when writing `Python` code. Furthermore, the package `Pylint` was also utilized in order to help the thesis writers to follow the `PEP8` guide by finding bugs and style problems in `Python` code. Consistent coding tools and techniques were only used when developing the final model. These tools and techniques were never applied during data exploration since it became too tedious to apply these to temporary code.

4.5.2 Automate task performances

The theory study suggested that a tool in the Google cloud platform which could be utilized is Vertex AI workbench for training models in scheduled time. This tool was never found to be useful for us. The reason for this is that the developed models were not time consuming enough when trained and there was never a need to regularly update the input data to retrain the models. Static data is studied. The techniques

Table 4.13: Connecting software engineering principles with tools which was implemented to achieve them during model development by the thesis writers.

Approach by thesis writers	
SE Principle	**Tool/Technique**
Consistent Coding	Using `PEP8` style guide framework `Pylint` tool
Automate Task Performances	Creating classes and functions
Incremental Coding	Using `Git` repositories
Avoiding Repetition	Modular coding Creating classes and functions
Testing Notebooks	Unit testing Assertion tests in line
Software Optimization	-
Software Documentation	Writing docstring for classes and functions using `PEP8` style guide framework Creating a readme file Commenting in line
Software Collaboration	Code review before merging in `Git` Pair programming

of regularly creating classes and functions were followed on the other hand since it was early found to be useful to create reusable functions and classes. Many of the functions and classes which were created during data exploration could be directly moved into the final software if found relevant. The created functions and classes followed the `PEP8` style guide.

4.5.3 Incremental coding

`Git` repositories were set up early during the project to achieve incremental coding principles. `Git` repositories were developed both for the data exploration phase and during the final software development. During data exploration, `Git` repositories were found useful for the task of saving old results and findings since both the notebook code cells and their respective output could be saved in a version control system. During the final software development, `Git` repositories were particularly useful to save checkpoints in the case when errors occurred. Stages of the development could be reloaded and bugs could be tracked. No particular principles of agile software development was implemented during the project.

4.5.4 Avoiding repetition

The software engineering principle of avoiding repetition was achieved by focusing on creating modular code by creating modules and functions. Furthermore, two modules was developed during the thesis which can be reused by Company X in similar analysis contexts. This can be useful for both software exploration stages and final software delivery stages in the future.

4.5.5 Testing Notebooks

Testing the code written during the project was done in several ways. Assertion tests were written in line to make sure that input objects were of the correct type for functions. Unit tests were also written to ensure that during development, functions and classes were not broken when code was changed. Implementation of unit tests in the notebook was achieved with the `unittest` package in `Python`. Unit tests were developed to test input data during the data exploration phase as well. These tests could be directly moved into the final software. Other test cases were further developed during final software development which ensured that the created software worked as intended throughout the development stage.

4.5.6 Software Optimization

Since the project did not require consistently updating the input data or increased time-performance, there was no need to optimize the written code in a lower-level language. If the project would move into the software deployment stage, optimizing the code would be more relevant.

4.5.7 Software Documentation

To ensure software documentation, `docstrings` were added to each class and function created in `Python`. It was also suggested by the `PEP8` style guide to write `docstrings` for functions and classes. The format of the `docstrings` follows the `PEP8` style guide suggestions. Furthermore, comments in-line were also added to explain algorithms which could be difficult to understand at a first glance. The comments also followed `PEP8` style guide's suggestions, only writing comments if the programmer suspects that the code written is difficult to understand. Finally, a `readme` file was written to document how to run and set up the created software. Software documentation tools and techniques were only used during the final model development, except for small comments in line during data exploration.

4.5.8 Software Collaboration

During the project, the two thesis writers worked simultaneously on the code. When working on different sections of the code, reviews of pull requests were required before merging code. The purpose of creating pull requests was often for the sole reason of reviewing each other's code rather than solving conflicts. On more difficult algorithms such as building the models, pair programming was implemented to ensure that both the thesis writers understood the code and that the code worked properly. Collaboration tools and techniques were found useful both during data exploration and final software development.

4.6 Software engineering principles followed by data scientists at Company X

Since software engineering principles applied by data scientists can differ due to the employees background and tasks at Company X, this section begins with a background description of the interview subjects. Thereafter the tools and techniques which the interview subjects brought up will be presented together with opinions brought forward during each interview on how software engineering principles should be applied and are currently applied.

4.6.1 Background of the interview subjects

Interview subject 1 (IS1) has worked at Company X for around eight months as a data scientist with educational background from statistics and previous working experience at a technology company working within data science. The project which IS1 works within has internal stakeholders at Company X and the time frame for each project can vary, mainly having a long time frame of 4-6 months. Some tasks are shorter and only have a time frame of 1 to 2 weeks. IS1 often works alone in the projects, but collaborates with colleges if the projects need other expertise.

Interview subject 2 (IS2) works with explorative data science projects within Company X. Projects often start when an external customer comes with a request of a task or data which the data scientists at Company X tries to solve. Projects can also start when internal stakeholders at Company X finds a problem which they require data scientists to solve. Some projects start but never finish since the resulting business value is viewed as too low.

Interview subject 3 (IS3) has a long background as a software developer at Company X before transitioning to data science tasks. IS3 also has experience in data engineering. IS3 works with both internal and external stakeholders in data science projects which often span for a couple of months in length. The final delivered product is often a result and/or a working module or tool.

Interview subject 4 (IS4) works with an external stakeholder at Company X in a project together with interview subject 5 (IS5) and interview subject 6 (IS6). The external stakeholder for these interview subjects is Company G. The interviews with IS4, IS5, and IS6 were held individually to receive subjective opinions. With the stakeholder, IS4, IS5, and IS6 are building a proof of concept when developing data science models. The project has been ongoing for six months. The data science tasks are often exploratory in the beginning, but the end goal is to deliver a software tool to the external stakeholder. When the interview was held, IS4, IS5, and IS6 were still in the explorative phase of the project and had yet to begin developing the end product for the external stakeholders. During the explorative phase, the programming is often done individually in the team of IS4, IS5, and IS6 in `Jupyter` notebooks where results are shared after finished calculations.

Table 4.14: Connecting software engineering principles with tools and techniques used currently at Company *X* during data science programming

Approach by data scientists at Company X	
SE principle	**Tool/Technique**
Consistent Coding	`PEP` standard/`PEP8` (IS1, IS3) Specific writing rules for final models within the team (IS4, IS5, IS6) Using linting tools when writing code (IS2, IS3, IS5)
Automate Task Performances	Writing modules and libraries (IS1, IS3, IS4) Functional programming (IS5, IS6) Writing python files rather than notebooks (IS3) GCP airflow tool (IS3, IS4)
Incremental Coding	Using `Git` repositories (IS1, IS3, IS5, IS6)
Avoiding Repetition	Writing modules and libraries (IS1, IS3, IS6) Writing modular code (IS1, IS2, IS3, IS4, IS5, IS6)
Testing Notebooks	Unit testing internally in notebook (IS1, IS5, IS6) External test files for created modules (IS1) Assertions in line (IS1, IS4) Testing data (IS1, IS4, IS5, IS6) Manual testing (IS3)
Software Optimization	Refactors code before deployment (IS1, IS2, IS3, IS4, IS5, IS6)
Software Documentation	Writing `docstrings` for libraries and modules using the `PEP8` framework (IS1, IS3) Commenting in line (IS1, IS3, IS4, IS5) Creating a readme file (IS2, IS3) Markdown documentation in notebook (IS4, IS5, IS6) Literate programming (IS5)
Software Collaboration	Code review (IS3, IS4, IS5, IS6) Pair programming (IS3, IS4, IS5, IS6)

4.6.2 Consistent Coding

Consistent coding is a software engineering principle followed by all of the interview subjects but in different ways. Specific writing rules are brought up by IS1, IS3, IS4, IS5 and IS6. IS2 and IS5 do not have any specific styling rules but use linting tools to ensure that easier styling errors are avoided. IS4, IS5 and IS6 uses an internal styling framework.

All of the interview subjects highlight that for explorative studies, consistent coding rules are not followed as meticulously as for final model development. IS2 claims that consistency in code writing for data scientists within the company might be trivialized due to time constraint. According to IS5, data science code is more isolated in comparison with traditional software development, meaning that for data science projects, there is less need to have consistent coding standards, mainly because the code is less likely integrated in other systems and viewed by other developers.

4.6.3 Automate Task Performances

IS1, IS3 and IS4 all bring up writing modules and functions rather than doing "ad hoc" programming. IS3 also specifies that notebooks are rarely used by him whilst the other interview subjects mainly program in notebooks. The interview subjects also agree that modules and functions are mainly built when developing a product for deployment and not during explorative programming phases. IS3 and IS4 have experience of automatic GCP airflow tools as well, which were mainly used for feature engineering. IS5 and IS6 also bring up functional programming for the purpose of writing tests easier and automating tasks with functions.

4.6.4 Incremental Coding

Incremental coding by utilizing a `Git` repository is done by IS1, IS3, IS5 and IS6. IS1 claims that the main reason for utilizing `Git` repositories is for version control whilst IS3 uses `Git` repositories for both version control and the ability to conduct code review. IS4 says that `Git` repositories are not used at the moment for their team since the code is very small at the moment and can be changed very quickly. IS4 adds that there have been recent requirements from other departments and more senior employees that data science teams also use `Git` repositories accessible by the employees at the company for knowledge sharing.

As interviews with IS5 and IS6 were held one week after the interview with IS4, recent `Git` repositories have been set up for the project team. IS5 says that the reason for using `Git` repositories is to track historical changes in the software. At the same time, IS5 discusses that keeping historical versions of the software is more difficult for `Jupyter` notebooks since the output of each cell is also saved, but these are dependent on the input data which will change over time when data is updated. IS6 says that the ambition with the `Git` repositories is to set up code review standards for the final model development phase.

IS2 claims that no version control is necessary during an explorative phase of a project and is rarely needed in general by the interview subject. IS2 says that the reason he avoids using `Git` repositories is that it can create friction when working and it has the risk of becoming unnecessarily time-consuming. He also adds that old data and results are obsolete, and therefore there exists no need for version control in a data science context.

When discussing agile software development as a development technique in a data science context with the interview subjects, the respondents agree that it is too difficult to implement. According to IS5, the reason for this is that it is very difficult to divide tasks in data science projects in the same way as in more "classical" software engineering projects. The tasks are more abstract in nature and not as easily defined as traditional software engineering tasks. IS2 adds that efficiency within a data science team may decrease if agile software development principles were to be implemented.

4.6.5 Avoiding Repetition

The technique of writing modular code is followed by every interview subject. IS1, IS3, and IS6 also emphasize writing modules and libraries rather than only programming in notebooks. IS2 does not put any emphasis on writing classes and functions regularly but does so if he finds that the methods are frequently occurring in the same project. IS5 and IS6 also bring up functional programming which can be used to avoid repetition and make the code more understandable.

4.6.6 Testing Notebooks

IS1 uses a lot of different test suites, both in line and external files. The reason why IS1 writes tests is to "feel more safe" when developing and feels that the results are more reliable in a data science context. IS4 does not currently write external tests but writes assertion tests in line of the notebooks and tests the input data. The goal is to write more rigorous tests when the team starts the development of a deliverable product. IS5 and IS6 writes unit tests inside their notebooks. This becomes easier by implementing the functional programming technique.

IS4, IS5 and IS6 all bring up the importance of testing the data to make sure that the input data is as expected and therefore can be used as input in their models. IS3 mainly uses manual tests and claims that writing tests becomes superfluous since the stakeholder is often only interested in a numerical result rather than understanding the process of receiving the result. There is small value in writing tests on code with single runs. IS3 also states that he wished that there would be more incitement for writing tests in data science projects since he finds it valuable to ensure that the code written is robust and can be used in multiple contexts. IS2 sees little value in writing tests in data science context.

4.6.7 Software Optimization

All of the interview subjects describe two different phases of programming in data science projects. Firstly an explorative phase needs to be conducted, secondly, the final tool or software needs to be delivered. Only the second phase is deliverable, and therefore also refactored for optimization before deployment or integration into some other system. IS2 and IS5 also bring up the fact that module such as `Numpy` are used when developing data science software in `Python`, which utilizes optimization possibilities in `C`.

4.6.8 Software Documentation

Software documentation is done very differently among the interview subjects. IS1 and IS3 document their code by writing `docstrings` for the developed libraries and modules, following the `PEP8` framework. For explorative programming, IS1, IS3, IS4, and IS5 emphasize the importance of writing comments in line while IS2 says that this is not as necessary in data science software since the functionality of the code should be easier to understand. Instead, IS2 highlights readme files when developing the final software, which is a documentation method shared by IS3.

Within the project team of IS4, IS5, and IS6, they utilize `markdown` cells in notebooks to document their software. By using markdown cells, the team can explain calculation methods and expected input and output for their developed models in a literate way. Literate programming is also a technique that was brought up by IS5, where documentation of code is done along with the development of software. The purpose is to be able to read code almost like a book, where each operation and algorithm is explained step-wise.

4.6.9 Software collaboration

Code reviews were utilized by every interview subject except for IS1 and IS2, who both state that this can be improved within Company X. They still feel that code review is important for maintaining a coding standard within the company but receives little opportunity for it. IS3 says that code review is put as a standard in projects which he has worked in by requiring a review before pushing code in `Git`.

IS4, IS5 and IS6 also claim that code review is important and that they achieve this by sharing notebooks within the team. They also have an ambition of setting up a `Git` repository which requires code review for the final model development. Code review is less utilized during explorative phase where they instead share and discuss results of the explorative programming. Pair programming is also utilized when possible by IS3, IS4, IS5 and IS6.

5
Discussion

5.1 Time series models

In this section, the results of the three proposed models are discussed. As seen in the result section, the model that has the best performance (in terms of prediction accuracy) out of the three models is the BSTS model. In all but one of the compared cases, the BSTS model has the best RMSE, MAE and MASE. Only when studying the time series of delays for the service type B, the study finds that the mean RMSE is better for the GAM. In the other cases, the BSTS model has the best performance.

One explanation for why the BSTS model out-performs the GAMs is possibly that the BSTS model uses mixed-effect variables and does not look at only independent variables in contrast to the GAM. A second explanation is that the BSTS is dynamic; the coefficients for its variables change over time and it can therefore adapt to changes in the observed time series. When the amplitude of the time series changes, the BSTS can adjust its coefficients accordingly. The GAM model assumes that the behavior of the time series is constant over time, and fits the coefficients for its functions accordingly, i.e. it minimizes the total error across the whole observed time series.

The ARIMA models perform in all the studied cases worse than the corresponding GAM and BSTS models. This is expected since the ARIMA models are in essence subsets of the two other classes of models. The BSTS models include both trend, seasonal and auto-regressive components, together with a dynamical regression component. The GAM model also utilizes regression variables in addition to correlation, trend, and seasonality. This indicates that the regression variables have *some* predictability toward the target variable.

Observing the MASE score, one can see that the BSTS model makes significantly better predictions in comparison with a naive model. As mentioned in Subsection 3.5.3, a MASE value lower than one means that the model performs on average better than a naive model, and a value larger than one means that the forecasts are worse on average. The BSTS model has a mean MASE score of ~0.57 (Table 4.6) across all regions, which means that it on average scores just over half the prediction error in comparison with a naive model. The GAMs and ARIMA models in contrast only reach a MASE score of ~0.85 and ~0.89 on average. This means that the two classes of models are not capable of producing significantly better predictions than a naive model.

By observing the accuracy scores, the GAM and ARIMA model do not produce sufficiently reliable forecasts to be entrusted in the given context. However, by observing Figure 4.5 together with its accuracy scores, one can argue that it could be used in a real life setting when analyzing future behaviour of the target variable. As seen in the figure, it predicts dips and spikes that are also observed in the actual data.

When studying the service type A the MASE is $\sim$0.53 for the BSTS model (Table 4.7), whereas the GAM and ARIMA model reach a MASE of $\sim$0.95 and $\sim$1.75 respectively. For these time series, in particular, the BSTS model performs significantly better than the other two and a naive model.

For service type B, GAM and ARIMA obtain slightly better accuracy scores (Table 4.8), but the BSTS models still obtain the best performance. An explanation for why the two other classes of models (GAM and ARIMA) perform better on these time series is that they display a lot less variability in the time series, as opposed to their counterpart time series of service type A which exhibit more variation.

5.1.1 Generalizability of findings

The results of the modeling practices in the report are specific to the studied case. The regression variables used in the report consist of data that is owned by Company X. It is also case-specific. It is therefore not as straightforward as to apply the modeling practices in this report to other similar cases since similar data is required. The models are also tailored specifically to the observed series of late shares for the different regions and shipment services. Therefore, the models created cannot be directly re-applied to new cases. For Company X however, time series analysis with ARIMA, BSTS, and GAM can be reapplied in other projects with the help of the modules created during the thesis.

Generalizability rather lies in the comparison between our three classes of models. The ability to dynamically change the values of coefficients in irregular time series such as the ones considered in this thesis is valuable for forecasting purposes. Another finding is that the inclusion of predictor variables based on tracking events displays improved prediction accuracy when producing time series forecasts of the late share of packages.

The performance of the models also seems to depend on the appearance of the time series. When the time series of late shares display small variance and only consists of sporadic spikes in the target variable, as in the time series for region DE1 displayed in Figure 4.1, the models performs poorly. In contrast, when the time series possess high variability, the models perform better. This means that the modeling results are dependent on the presence of variation in the observed time series.

5.1.2 Feature engineering and importance scores

The importance of the engineered features are summarized in both Table 4.11 and Table 4.12. For the GAMs, the smoothing functions that are assigned a significant score (P-value lesser than 0.05), are displayed in Table 4.12. Both the variables created for capturing a trend component and a seasonal component, *time* and *wDay* respectively, display high significance the most number of times. Time series which has significant smoothing functions are the ones that display the largest variability on its target variable. In time series that don't display obvious trends or seasonal patterns, the importance of the smoothing functions is not as significant.

Out of the variables engineered from the tracking-event data, the ones that are deemed as significant are $TBE1$, $TBE2$, $TBE3$ and, $ArrivalFirstHub$. In general, the variables that explain the mean time between tracking events, seem to be the most informative in the GAMs. However in some pairs of region and service type, no variables are chosen as significant. This is not a surprising result since in these regions, the target variable does not display much deviation from the value zero (e.g. region DE1 displayed in Figure 4.3). Naturally, it is hard for a variable to correlate with a target variable that is constant for most of the time. In the time series where a lot of variation is observed, a few variables are deemed as important. These variables are the ones listed in Table 4.12. It is worth mentioning that no mixed effects between the variables have been analyzed in the case of GAMs.

In Table 4.11, the feature importance results for the fitted BSTS models are summarized. The features have been assigned a score, that is, as previously mentioned the mean inclusion probability multiplied with the number of times a feature has been selected by the various models. The variable that obtains the highest score is $ArrivalFirstHub$. This variable was also selected by the GAMs in one case. The BSTS model also assigns higher significance scores to mixed effects between the TBE variables. In essence, this indicates that when duration between many events are high simultaneously, shipments to the studied region are more likely to be late. Several TBE variables have an inclusion probability that is on average below 0.5. This means that when drawing from the posterior distribution, they are only included in less than half of the 2000 iterations. The only variable that has a high mean inclusion probability (0.987) is $ArrivalFirstHub$.

5.1.3 Utility of modeling results for Company X

The modeling part of the report has resulted in findings regarding different time series modeling techniques and when they are appropriate, and information regarding the usefulness of certain variables. One takeaway is that for ARIMA, BSTS, and, GAM to work properly on different time series, the time series need to display a sufficient amount of variance. By comparing for example time series such as the two displayed in Figure 4.2 and Figure 4.5, one can observe that in the latter case, the models are capable of capturing some characteristics of the underlying observed series, whereas in the former case this is not possible. Other target variables such as mean, maximum or median delay of shipments could be more informative for less varying time series.

When a regional time series of late share of packages display enough variance it is possible, at least *to some extent*, to predict the behaviour of the underlying time series. However, the predictability is far from perfect or by any means exhaustive for drawing sound conclusions about the future behavior of the target variable. Instead, the best models examined in this report can give an *indication* about future behavior which, combined with further analytics and knowledge about the environment, can help an agent to draw more informed guesses about the future values of the target variable.

Share of late packages is a relative measurement. Its impact on the delivery capabilities for Company G depends on the volume of packages delivered during the given day. Therefore the utility of the developed models will increase if the volume of packages are forecasted simultaneously.

5.1.4 Future research in the modeling domain

When it comes to future research in the time series modeling domain, the study recommends that researchers continue on the path of including regression components in the input data when modeling time series of delays. Researchers are recommended in particular to study the mixed effects of the variables representing time-between-events and the $ArrivalFirstHub$ variable. Further dividing larger regions into smaller time series is also recommended. However, a warning is given to not divide the data into too granular regions since this might make the variation in the target variable limited. The regions DE8, DE9, and, DEA of service type A should be further investigated by Company X since these showed distinctively higher mean share of late packages. Furthermore, different pre-processing approaches to the data might yield better results. One suggestion is to handle missing data as a “problem within the problem” as discussed in Subsection 3.3.1.

In this report, only variables that are created on the basis of tracking data has been taken into account. Further research can look into expanding the set of variables to also include variables that represent weather data in the regions and data representing traffic congestion (e.g. from Google live traffic data). Furthermore, the data was limited to one year. Therefore only weekly seasonality was studied. By enlarging the temporal measurement of the study, several types of seasonality can be found such as monthly and yearly seasonality. Holiday effects could also be captured.

5.2 Software engineering principles in data science programming

The theory show numerous tools and techniques applicable to realize software engineering principles in the context of data science programming. Furthermore, the results show the tools and techniques applicable for Company X’s working environment summarized in Table 4.13. Similar to the interview subjects, the study also found a distinction between data exploration and final software delivery. All software engineering principles were realized to some degree by applying the tools and techniques shown in the results in either the software exploration phase or the final software delivery phase, except for the software engineering principle software optimization. This is due to the project scope not involving the strategic area model deployment. If the project would move into the software deployment stage, software optimization tools and techniques would be of higher relevance.

Some tools and techniques brought up in the theory were not implemented. For consistent coding, an identifier dictionary was never used. This seemed redundant since the PEP8 framework was already implemented and the tool pylint was used. Adding another tool for consistent coding would not bring value and would only be time consuming. GCP's airflow tool was never utilized since the project scope and size never required re-training of time consuming models. This could be done manually instead.

5.2.1 Software engineering principles followed by Company X

The results show that different interview subject view different software engineering principle as important. This is dependent on what background the interview subject has. For instance, interview subjects who worked alone for the most part rather than in teams did not follow the principle software collaboration, although they did mention the importance of this software engineering principle. Furthermore, the choice of coding environment was also varying between interview subjects. Some interview subjects preferred to only code in notebooks where literate programming could be implemented and in this way, follow the principle of software documentation. Some only wrote `Python` files to easier create libraries and modules to automate task performances and to avoid repetition.

There is also little consistency in what tools and techniques the data scientists at Company X use to follow the software engineering principles. The differences in opinions on software testing are mainly found due to the data scientists' background and tasks at hand where data scientists who have previously worked with software engineering tasks value software testing principles more than those with pure mathematical and data science background. Software documentation is also implemented very differently among the data scientists with some utilizing external documentation tools such as `readme` files and some only documenting inside the code itself. Although documentation is done by all data scientists at Company X, the different methods used can create friction.

5.2.2 Data exploration and final software delivery

An important finding from the interviews, is the difference between the data exploration phase and an eventual final software delivery phase of data science projects.

The interview subjects often distinguishes on how they work during these two different phases, putting less emphasis on software engineering principles during the data exploration phase. The difference was also experienced by us, finding a contrast between work done during data exploration, and work done to develop the final model.

The data exploration phase is defined by the interview subjects as the initial step in a data science project where the programmers need to spend time to learn the data at hand and test different methods to achieve results. The delivery from a data exploration phase is often some type of data or some mathematical results. In some cases, data science projects require the team to move into a software delivery phase where the delivery is a software.

All software engineering principles are less emphasized by the interview subjects if the project phase is data exploration. An argument for this is that during data exploration, there is no guarantee that the software which is developed will be used repeatedly. Therefore, the data exploration phase is often done quickly and software engineering principles are sacrificed for time. The data exploration phase is often done by individual data scientists as well where only the finding of the experiment is presented to the team. For final software delivery there is more emphasis on software engineering principles since there are often more data scientists working simultaneously on a software and the software is often larger. Differentiating a data science project into data exploration and software delivery is something which the theory research did not emphasize upon.

5.2.3 How Company X applies software engineering principles compared to the thesis's development process

The tools and techniques which were studied during the project are currently being used by at least one interview subject showing the industrial relevance of tools and techniques found in the theory. The interview subjects used several different tools and techniques as well which the study did not apply. This is due to the fact that some techniques are just different ways of reaching the same goal.

The main difference between how Company *X* works and the working procedure implemented by us is that Company *X* creates a larger distinction in the need of applying software engineering principles in the data exploration phase in comparison with the final software delivery phase. As shown by our working procedure though, there can be many benefits of trying to apply some of the software engineering principles in the data exploration phase as well. For instance, by always writing modular code, created functions and classes during data exploration were directly moved into the final software. `Git` repositories were useful to keep track of previous analysis during data exploration and to save previous findings. During the data exploration phase, test cases were often written to test the input data. These test cases could also be directly moved into the final software. The results do show though, that it is more difficult to apply all software engineering principles during data exploration in comparison with the final software delivery. Software principles such as consistent coding and software documentation were sacrificed for time.

5.2.4 Recommendations for Company X

By clearly defining the two different phases data exploration and final software delivery for data science programming internally at Company X, relevant software engineering principles can more easily be applied for specific phase. Although the findings from this thesis's development work is similar to Company X's opinions in the case of software engineering principles being more relevant in final software delivery rather than data exploration, the thesis's development process still finds some tools and techniques for software engineering principles to be relevant during data exploration. As shown by the thesis's development process, code can easily be moved to the final software delivery phase if they are written as functions and classes. Incremental coding should also be applied during data exploration to save old results of experiments. Test cases used during data exploration can also be moved into the final software.

Furthermore, Company X lacks a synchronized framework for what tools and techniques their data scientists apply in order to follow software engineering principles. Although consistent coding, documentation and avoiding repetition is done by all data scientists, these principles are realized with different methods. By having an internal framework for what coding style they use and whether they write `docstrings` or `readme` files for documentation for example, tools and techniques used by the company become more consistent and friction is reduced for their data scientists.

5.2.5 Future research for software engineering principles in data science programming

The conducted thesis has shown that it is possible to apply software engineering principles in data science programming with the brought up tools and techniques. Future work should try to measure how well software engineering principles are followed with applied tools and techniques by studying the software's quality. By adding a measurable metric, more informative decisions on what tools and techniques should be used can be made.

Another area which can be further explored is a broader or different project scope. The strategic area software deployment is never studied. By exploring tools and techniques to achieve software engineering principles applicable in software deployment, more software engineering principles can be followed and more tools and techniques become relevant.

Furthermore, the case which this thesis has been conducted in is specific to one company. More research needs to be done in different fields to give a broader understanding of how the industry applies software engineering principles in data science programming. The studied environment is GCP, programming done in

`Python` and `R`. By expanding this scope, findings can become more generalizable.

Finally, research within software engineering principles for data science programming should try to define data science programming in not only strategic areas, but also the two different phases of data exploration and software delivery. By defining the two different phases and their differences, it becomes easier to reason for applying tools and techniques for software engineering principles. Different tools and techniques are suitable in different phases of data science programming.

5.3 Threats to validity

As shown in the methods chapter, the conducted thesis is in the research strategy area of a field study. The thesis's obstructiveness is relatively low, and the context is specific to Company X. Although the realism of the thesis is high, the threat to validity is that the conducted research becomes context-dependent.

There are also limitations to the data on which the thesis is conducted. The studied temporal information is not the most recent data. The data can change over time, and the insights found might differ for more recent data. The thesis can show that BSTS is better than GAM and ARIMA in the specific context but cannot prove that this model always performs the best since the parameters used for the models differ and are specific for the context. The specific ways that the models are applied might also not be optimal, and therefore the results are also bound by the capabilities of the thesis writers. The created models were dependent on packages available in R created by other developers than us. This dependency on other developers' knowledge and correctness also threatens validity.

It is also worth discussing if the assumption of normality of the different observations of the TBE variables holds. It is assumed that the mean TBE is a realization of a sum of independent identically distributed observations (that is, the TBE for all packages during a day) and therefore the observations of mean times are normally distributed according to the central limit theorem. This assumption is somewhat weak since several packages might be transported with the same vehicle to the next hub as an example. In this case, the TBE for these variables is not independent. The same argument can be made for the variables describing volumes of packages completing steps in their journey to their end region.

There are also threats to validity when comparing the suggested tools and techniques for software engineering principles from the theory and tools and techniques used by Company X. Since the thesis writers had contact with employees from Company X during the project, their thoughts and opinions might have influenced us. The comparison can therefore become biased. Furthermore, the conducted interviews could also become biased since the interviewer and the interviewee could be affected during the interview process. If a neutral participant conducted the interview instead, the participant's opinions would become less influenced by each other.

6 Conclusion

In the task of predicting shipment delays from Company G, the BSTS model displays the best forecasting capability out of the compared models. In terms of the three statistical measures RMSE, MAE, and MASE across the two different service types, the BSTS model obtains the best scores. The only exception is when comparing the average RMSE score for service type B. In this case, the GAM model obtains a marginally lower RMSE score, however the BSTS scores better in terms of the two other statistical measures.

Furthermore, all regional time series for both services do not display sufficient variation in the target variable to make them suitable for time series analysis. Consequentially, meaningful conclusions can only be drawn from time series that exhibit a larger amount of variation in its target variable. In these cases, the BSTS models have the proven ability to take advantage of the regression variables when producing forecasts. None of the variables show a significant correlation in every region considered. This is expected since as earlier stated, some regions show very little to no variation in the target variable. Additionally, mixed effects between the variables also score better than their independent contributions. These findings should be taken into account when working with the data in future projects.

During the research procedure the software engineering principles of consistent coding, automate task performances, incremental coding, avoiding repetition, testing notebooks, software documentation and software collaboration were found to be relevant. Tools and techniques applicable to follow these software engineering principles are shown in Table 4.13 and 4.14. The results also show that industrial applications of the software engineering principles are similar to what is suggested from the theory.

However, the results shows that in an industrial setting, data science programming is divided into two fundamentally different phases, data exploration, and final software development. The industrial setting places different emphasis on the importance of software engineering principles in the different development phases. The thesis suggests future research to more clearly define the two different phases of data exploration and final software development. Moreover, there is no clear consistency in how software engineering principles are followed at Company X. The thesis also suggest Company X to consider the importance of creating a clearly outlined framework for the application of software engineering principles in both the data exploration phase and the final software development phase.

Bibliography

Abu Bakar, N., & Rosbi, S. (2017). Autoregressive integrated moving average (arima) model for forecasting cryptocurrency exchange rate in high volatility environment: A new insight of bitcoin transaction. *International Journal of Advanced Engineering Research and Scinece*, *4*. https://doi.org/10.22161/ijaers.4.11.20

Akbari, M., & Do, T. (2021). A systematic review of machine learning in logistics and supply chain management: Current trends and future directions. *Benchmarking An International Journal*, *28*, 2977–3005. https://doi.org/10.1108/BIJ-10-2020-0514

Alsaedi, Y., Tularam, A., & Wong, V. (2019). Application of arima modelling for the forecasting of solar, wind, spot and options electricity prices: The australian national electricity market. *International Journal of Energy Economics and Policy*, *9*, 263–272. https://doi.org/10.32479/ijeep.7785

Al-Sarayreh, K. T., Meridji, K., & Abran, A. (2021). Software engineering principles: A systematic mapping study and a quantitative literature review. *Engineering Science and Technology, an International Journal*, *24*(3), 768–781. https://doi.org/https://doi.org/10.1016/j.jestch.2020.11.005

Armstrong, J. S., Collopy, F., Armstrong, J. S., & Collopy, F. (1992). Error measures for generalizing about forecasting methods: Empirical comparisons. *International Journal of Forecasting*, 69–80.

Bell, E., Bryman, A., & Harley, B. (2019). *Dbusiness research methods (5th ed.)* Oxford University Press.

Bosch, J., Olsson, H., & Crnkovic, I. (2021). Engineering ai systems: A research agenda. https://doi.org/10.4018/978-1-7998-5101-1.ch001

Breuker, D., Matzner, M., Delfmann, P., & Becker, J. (2016). Comprehensible predictive models for business processes. *MIS Quarterly*, *40*, 2016. https://doi.org/10.25300/MISQ/2016/40.4.10

Brockwell, P., & Davis, R. (2009). *Time series: Theory and methods*. Springer. https://books.google.se/books?id=%5C_DcYu%5C_EhVzUC

Brodersen, K. H., Gallusser, F., Koehler, J., Remy, N., & Scott, S. (2015). Inferring causal impact using bayesian structural time-series models. *The Annals of Applied Statistics*, *9*, 247–274. https://doi.org/10.1214/14-AOAS788

Cam, A., Chui, M., & Hall, B. (2019). Global ai survey: Ai proves its worth, but few scale impact (M.
bibinitperiod Company, Ed.) [[Online; fethed 13-march-2022]]. https://www.mckinsey.com/featured-insights/artificial-intelligence/global-ai-survey-ai-proves-its-worth-but-few-scale-impact

Christopher, M. (2005). *Creating value-adding networks logistics and supply chain management*. Prentice Hall.

Comission, E. (2021). Nuts - nomenclature of territorial units for statistics. https://ec.europa.eu/eurostat/web/nuts/background

Deissenbock, F., & Pizka, M. (2005). Concise and consistent naming [software system identifier naming], 97–106. https://doi.org/10.1109/WPC.2005.14

GoogleCloud. (2022a). *Cloud composer overview* (Documentation). https://cloud.google.com/composer/docs/concepts/overview

GoogleCloud. (2022b). Create a code repository in cloud source repositories. https://cloud.google.com/source-repositories/docs/create-code-repository

Güvercin, M., Ferhatosmanoglu, N., & Gedik, B. (2021). Forecasting flight delays using clustered models based on airport networks. *IEEE Transactions on Intelligent Transportation Systems*, *22*(5), 3179–3189. https://doi.org/10.1109/TITS.2020.2990960

Hannay, J. E., MacLeod, C., Singer, J., Langtangen, H. P., Pfahl, D., & Wilson, G. (2009). How do scientists develop and use scientific software? *2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, 1–8. https://doi.org/10.1109/SECSE.2009.5069155

Harvey, A. C. (1990). *Forecasting, structural time series models and the kalman filter*. Cambridge University Press. https://doi.org/10.1017/CBO9781107049994

Hastie, T., & Tibshirani, R. (1986). Generalized Additive Models. *Statistical Science*, *1*(3), 297–310. https://doi.org/10.1214/ss/1177013604

Hyndman, R. J. (2022). Cross-validation for time series. https://robjhyndman.com/hyndsight/tscv/

Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software*, *27*(3), 1–22. https://doi.org/10.18637/jss.v027.i03

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, *22*(4), 679–688. https://doi.org/https://doi.org/10.1016/j.ijforecast.2006.03.001

Kalyan, A., Chiam, M., Sun, J., & Manoharan, S. (2016). A collaborative code review platform for github. *2016 21st International Conference on Engineering of Complex Computer Systems (ICECCS)*, 191–196. https://doi.org/10.1109/ICECCS.2016.032

Keung, K. L., Lee, C., & Yiu, Y. (2021). A machine learning predictive model for shipment delay and demand forecasting for warehouses and sales data, 1010–1014. https://doi.org/10.1109/IEEM50564.2021.9672946

Klosterman, S. (2019). *Data science projects with python: A case study approach to successful data science projects using python, pandas, and scikit-learn*. Packt Publishing. https://books.google.se/books?id=XzCWDwAAQBAJ

Laranjeiro, N., Agnelo, J., & Bernardino, J. (2021). A systematic review on software robustness assessment. *ACM Computing Surveys*, *54*, 1–65. https://doi.org/10.1145/3448977

Lee, B. (2018). Ten simple rules for documenting scientific software. *PLOS Computational Biology*, *14*, e1006561. https://doi.org/10.1371/journal.pcbi.1006561

Lee, H., Padmanabhan, V., & Whang, S. (2004). Information distortion in a supply chain: The bullwhip effect. *Management Science*, *43*, 546–558. https://doi.org/10.1287/mnsc.1040.0266

Lian, H., Liu, B., & Li, P. (2021). A fuel sales forecast method based on variational bayesian structural time series. *J. High Speed Networks*, *27*, 45–66.

Makkar, S., G.Naga Rama Devi, D., & Solanki, V. (2020). Applications of machine learning techniques in supply chain optimization. https://doi.org/10.1007/978-981-13-8461-5_98

Montgomery, D. C. (2019). *Applied statistics and probability for engineers, 7th australia and new zealand edition* (7th ed.). John Wiley & Sons, Incorporated.

Namjoshi, N. (2021). *Scale your data science workflows with the vertex ai workbench notebook executor* (G. Cloud, Ed.; tech. rep.). https://cloud.google.com/blog/products/ai-machine-learning/schedule-and-execute-notebooks-with-vertex-ai-workbench

Niranjan, K., Narayana, K., & Rao, M. (2021). Role of artifical intelligence in logistics and supply chain, 1–3. https://doi.org/10.1109/ICCCI50826.2021.9402625

nteract team. (2021). *Testbook documentation release 0.4.2* (tech. rep.). https://testbook.readthedocs.io/_/downloads/en/stable/pdf/

Panichella, S., & Zaugg, N. (2020). An empirical investigation of relevant changes and automation needs in modern code review. *Empirical Software Engineering*, *25*, 1–40. https://doi.org/10.1007/s10664-020-09870-3

Posch, K., Truden, C., Hungerländer, P., & Pilz, J. (2022). A bayesian approach for predicting food and beverage sales in staff canteens and restaurants. *International Journal of Forecasting*, *38*(1), 321–338. https://doi.org/https://doi.org/10.1016/j.ijforecast.2021.06.001

Ray, P., Ganguli, B., & Chakrabarti, A. (2021). A hybrid approach of bayesian structural time series with lstm to identify the influence of news sentiment on short-term forecasting of stock price. *IEEE Transactions on Computational Social Systems*, *PP*, 1–10. https://doi.org/10.1109/TCSS.2021.3073964

Roberson, C. M. (2021). Tpm21: Parcel shippers need better visibility to avoid delays. https://www.joc.com/node/3672176

Russell, S., Bennett, T., & Ghosh, D. (2019). Software engineering principles to improve quality and performance of r software. *PeerJ Computer Science*, *5*, e175. https://doi.org/10.7717/peerj-cs.175

Saarikko, T., Westergren, U., & Blomquist, T. (2020). Digital transformation: Five recommendations for the digitally conscious firm. *Business Horizons*, *63*, 823–839. https://doi.org/https://doi.org/10.1016/j.bushor.2020.07.005

Salleh, N. H. M., Riahi, R., Yang, Z., & Wang, J. (2017). Predicting a containership's arrival punctuality in liner operations by using a fuzzy rule-based bayesian network (frbbn). *The Asian Journal of Shipping and Logistics*, *33*, 95–104. https://doi.org/10.1016/j.ajsl.2017.06.007

Sarker, I. (2021). Data science and analytics: An overview from data-driven smart computing, decision-making and applications perspective. *SN Computer Science*, *2*. https://doi.org/10.1007/s42979-021-00765-8

Scott, S., & Varian, H. (2014). Predicting the present with Bayesian structural time series. *Int. J. of Mathematical Modelling and Numerical Optimisation*, *5*, 4–23. https://doi.org/10.1504/IJMMNO.2014.059942

Scott, S. L. (2021). *Bayesian structural time series*. https://cran.rstudio.com/web/packages/bsts/bsts.pdf

Simpson, G. (2022). Modelling seasonal data with gams. https://fromthebottomoftheheap.net/2014/05/09/modelling-seasonal-data-with-gam/

Stadtler, H., & Kilger, C. (2011). *Supply chain management and advanced planning: Concepts, models, software, and case studies*. Springer.

Stellwagen, E., & Tashman, L. (2013). Arima: The models of box and jenkins. *Foresight: Int. J. Appl. Forecast.*, 28–33.

Stol, K.-J., & Fitzgerald, B. (2018). The abc of software engineering research. *ACM Transactions on Software Engineering and Methodology*, *27*, 1–51. https://doi.org/10.1145/3241743

Vandeput, N. (2021). *Data science for supply chain forecasting*. De Gruyter.

van Rossum, G., Warsaw, B., & Coghlan, N. (2001). *Style guide for Python code* (PEP No. 8). https://www.python.org/dev/peps/pep-0008/

Wang, C.-C., Chien, C.-H., & Trappey, A. (2021). On the application of arima and lstm to predict order demand based on short lead time and on-time delivery requirements. *Processes*, *9*, 1157. https://doi.org/10.3390/pr9071157

Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate Research*, *30*(1), 79–82. https://doi.org/10.3354/cr030079

Wilson, G., Aruliah, D., Brown, C. T., Chue Hong, N., Davis, M., Guy, R., Haddock, S., Huff, K., Mitchell, I., Plumbley, M., Waugh, B., White, E., & Wilson, P. (2014). Best practices for scientific computing. *PLoS Biology*, *12*, e1001745. https://doi.org/10.1371/journal.pbio.1001745

Wood, S. N. (2011). Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models.

Yiu, C. Y., Ng, K. K. H., Kwok, K. C., Tung Lee, W., & Mo, H. T. (2021). Flight delay predictions and the study of its causal factors using machine learning algorithms. *2021 IEEE 3rd International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, 179–183. https://doi.org/10.1109/ICCASIT53235.2021.9633571

Yuan, J., Wu, Y., Jing, W., Liu, J., Du, M., Wang, Y., & Liu, M. (2021). Association between meteorological factors and daily new cases of covid-19 in 188 countries: A time series analysis. *Science of The Total Environment*, *780*, 146538. https://doi.org/https://doi.org/10.1016/j.scitotenv.2021.146538

A
Interview questions

1. Could you describe your current role at company X?
   (a) Which are the main responsibilities you have in your role at company X?
   (b) Could you provide examples from your daily work that characterise the main responsibilities at company X?
   (c) Which stakeholders (internal and/or external to company X) do you collaborate and interact with to fulfil these responsibilities?
2. Could you describe your work within data science at company X?
   (a) What time frame do you have when working with data science projects?
   (b) How many people work simultaneously in a team of the data science projects?
   (c) What is the goal of the models you build? Is it for exploratory purposes or is it to deliver a specific tool to the stakeholders?
3. During the thesis work, we have found a framework of 8 software engineering principles for scientific programming. These are: Consistent Coding, Automate task performances, incremental coding, avoiding repetition, testing notebooks, software optimization, software documentation and software collaboration. Are you familiar with these?
4. When working with data science projects, do you try to implement any of the 8 software engineering principles and how do you do it?
   (a) What are the main challenges to implement said software engineering principles?
   (b) Could you describe the techniques/tools which are utilized to implement said software engineering principles?
   (c) What are the reasoning behind implementing the mentioned software engineering principles in your work? Is it a standard way of working or do you have any purpose of working in a certain way?
5. Out of the 8 software engineering principles which you do not implement, do you see any specific pros/cons with these?
   (a) What are the main reasoning behind not implementing these?
   (b) Can you see a future purpose of implementing more software engineering principles?