STOCKHOLM SCHOOL OF ECONOMICS
Department of Economics
5350 Master's Thesis in Economics
Academic Year 2020-2021

Machine Learning Methods for Forecasting Product Demand

A case study with telecommunications software
in collaboration with Ericsson AB

Meghan Byars (41654)

Abstract

There is a lack of evidence pointing to an optimal method for demand forecasting. This paper joins the collection of studies that forecast demand using a combination of machine learning methods. Demand forecasting literature from economics and supply chain management lead to a selection of machine learning models for this paper: random forest (RF), extreme gradient boosting (XGB), and support vector regression (SVR). These models are developed within a walk-forward validation process, alongside a benchmark seasonal auto-regressive moving average (SARIMA) model, to forecast demand of a telecommunications software product with data from Ericsson AB. A stacked ensemble hybrid model is constructed from the forecasts of the SARIMA and RF models. The SARIMA model, with a root mean square errors (RMSE) of 1.38, was outperformed by the three single machine learning models, which had RMSE between 0.98 and 1.05. The stacked ensemble hybrid model, SARIMA-RF, showed high predictive capabilities, with a RMSE of 0.43.

Keywords: Demand forecasting, machine learning, time series forecasting, random forest, SARIMA
JEL: C22, C44, C55, D12

Supervisor: Rickard Sandberg
Date submitted: May 17, 2021
Date examined: May 26, 2021
Discussant: Ellen Wallin, Eric Axdorph
Examiner: Kelly Ragan

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my academic supervisor, Rickard Sandberg, for his wealth of knowledge and advice. In addition, I am thankful for the guidance from Johan Olsson as my main supervisor at Ericsson who provided a multitude of ideas and interesting discussions throughout the process. Furthermore, I am grateful for Johan Boestad and the Software Supply team at Ericsson for the opportunity and the support I experienced.

1. Introduction

Demand forecasting differs from traditional demand research in economics which puts emphasis on identifying an equation of demand from its determinant factors. Even if demand forecasting provides less insight into the causes of observed demand, forecasting is still a crucial aspect, especially in applied settings. Businesses increasingly employ demand forecasting techniques in order to determine ideal inventory levels, and areas of public concern such as energy or agricultural supply are taking advantage of forecasting methods to determine future demand levels. In these applications, accurate forecasting of demand takes precedence over the relationship between demand and other factors. There is no forecasting method, however, that has been proven to perform optimally for demand forecasting, or forecasting in general (Fildes et al., 2019). Given this, academic literature from several fields apply the use of different modeling techniques to forecast future demand. Studies from economics and supply chain management have a sustained history of using auto-regressive moving average (ARIMA) time series forecasting techniques from Box and Jenkins (1967). Even so, literature from both economics and business are seeing an increase in the use of machine learning methods for forecasting demand.

Machine learning derives from a combination of statistics, mathematics, and computer science and is often harnessed for pattern detection and prediction. This deviates from the movement in economic research towards the emphasis of statistical methods that can determine causation between factors (Athey & Imbens, 2019). Athey and Imbens (2019) point out in their paper, "Machine Learning Methods That Economists Should Know About" that economic research has been slower at adopting machine learning techniques compared to other fields, including statistical research which has largely accepted it. As the usage of machine learning techniques is increasingly present within economics literature, continual exploration of their strengths and weaknesses in economic contexts is crucial.

This study joins those that seek to gain clarity surrounding the potentials of machine learning methods for the use of forecasting demand data. This is explored using download data for telecommunications software products in collaboration with Ericsson AB. The study consolidates literature from several fields in order to inform model choices. These include supply chain management, marketing, industrial organization, agricultural and energy economics, econometrics and statistics. Based on this, the Box and Jenkins (1967) seasonal auto-regressive moving average (SARIMA) forecasting approach is used as a benchmark model. This is complimented by a selection of machine learning

methods, namely random forest, extreme gradient boosting, and support vector machines. In addition, a final stacked machine learning ensemble model is constructed as a hybrid of the Box-Jenkins model with the single machine learning model that shows optimal performance.

The study tests no formal hypothesis as it maintains an exploratory purpose. The goal is a comparison of the chosen modeling approaches for forecasting demand of Ericsson's software. The models are compared by their prediction error, and the benefits and disadvantages of the each are discussed. Particular emphasis rests on the comparison of the benchmark model to the machine learning models, and the hybrid model to the single models.

The paper begins with a literature review based on the conceptual background areas related to the study. This is followed by theoretical background which discusses brief history, theory, and usage of the models and methods in the study. The research goals are then clarified, prior to a discussion regarding the data used in the study. Following this, the methodology is outlined for the study overall and for the different modeling approaches. Then, the results are presented with analysis. After results, some additional checks are outlined prior to a discussion.

2. Conceptual Background

Within this section, conceptual background information is discussed from several literature areas. First, a selection of business theory is discussed briefly which provides greater context to the study. Then, a literature review from business and economics is summarized with a focus on demand forecasting applications and methods. Finally, a summary of takeaways is provided before moving on to *Section 3. Theoretical Background*.

2.1 Supply Chain Management & Product Life Cycle Modeling

This section gives a brief history and overview of concepts related to demand forecasting from business literature. This includes a discussion about supply chain management which places demand forecasting within the larger contextual framework of supply chain management. In addition, product life cycle theory is discussed since it signifies the potential of a life cycle trend of demand after a product is launched. This theory is used in the study when choosing product data.

2.1.1 Supply Chain Management

Supply chain management (SCM) emerged as a field in the 1990's but has not been clearly defined nor given clear boundaries (Tan et al., 2002). The literature utilizes a multi-disciplinary approach, including research from marketing, strategy, operations management, and economics, to study the interconnected global systems ranging from suppliers to customers that global businesses must navigate today (Ellinger et al., 2015). Figure 1 (Meyr et al., 2002) shows a generalized supply chain planning matrix where the vertical axis represents planning intervals and the horizontal axis on the top represents business functions along the supply chain. Each component in Figure 1 represents large areas of research and business practices in their own right. For this study, the areas of demand planning and fulfillment are of particular relevance.

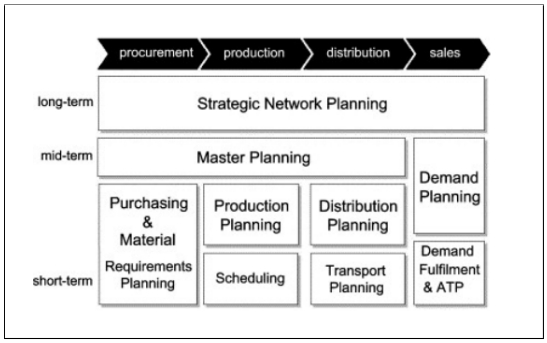

Figure 1: Supply chain management overview by Meyr et al. (2002)

Much of supply chain literature focuses on physical goods which present a commonly-studied phenomena: the bullwhip effect. The effect was coined by Procter & Gamble in the 1990's to refer to the increasing inventory variation seen at the beginning (upstream) of the supply chain as a response from smaller fluctuations in consumer demand at the end (downstream) of the supply chain (Wang & Disney, 2016). The bullwhip effect causes inefficiency and waste in the supply chain, so many studies have sought to understand the phenomena and its causes. Several comprehensive literature reviews have been published on the subject (Geary et al., 2006; Miragliotta, 2006; Towill et al., 2007; Giard & Sali, 2013; Wang & Disney, 2016). Some of this literature overlaps with economic literature through the use of econometric methods to examine the quantitative effect of the bullwhip phenomena. This is especially true within the subset of literature that looks at understanding the role of information-sharing along the supply chain.

Information sharing literature within supply chain literature focuses on modeling sections of the supply chain and examining the effect that information-sharing can have on inventory levels upstream in the supply chain. The idea is that supplier-visibility of point-of-sales data can reduce the effect of demand distortion on supplier inventory levels; therefore, within this literature, demand forecasting is a large focus (Kembro et al., 2014). Several studies model two-level supply chains consisting of a retailer and a supplier. Demand on the retailer level is commonly modeled as a univariate time series process, and the retailer and supplier each have static ordering decision equations with the level of information sharing as a parameter. In these studies, some model demand for products as a simple autoregressive (AR) model (Lee et al., 2000; Raghunathan, 2001), but it is more common to use both AR and moving-average (MA) components (Hsiao & Shieh, 2006; Xu et al., 2010; Cho & Lee, 2013). In addition, many researchers in this space advocate for modeling demand using seasonal ARMA models (i.e. SARMA), due to the common seasonality of product sales (Brown, 2004; Cho & Lee, 2013)

Stadtler (2005) claims that SCM is driven by demand and so demand forecasting is the starting point for understanding the supply chain. In addition, Stadtler claims while addressing Figure 1 from Meyr et al. (2002) that the integration of product life cycle theories into demand forecasting can elevate regular time series forecasting of demand into a more comprehensive demand planning.

2.1.2 Product Life Cycle Modeling

Supply chain management is aided by product life cycle monitoring; therefore, life cycle modeling and prediction can improve supply chain planning (Aitken et al., 2003). Product life cycle (PLC) modeling derives from marketing literature as a method of modeling the demand for a product from its

time entering the market until it is removed from the market. Classical PLC theory emerged in the late 1950's out of a necessity for a framework to account for the trajectory of individual products after market entry, and strategies for when to intervene in a product's trajectory with, e.g. price changes, inventory manufacturing, or discontinuation of the product (Cao & Folan, 2012). PLC theory overall constitutes a bell-shape-like model with stages of the product life cycle such as introduction stage, growth stage, maturity stage, and decline stage. The marketing literature typically measures PLC using revenue or sales as the vertical axis with time as the horizontal axis, as visualized in Figure 2 from Cox (1967), one of the earliest PLC theorists.

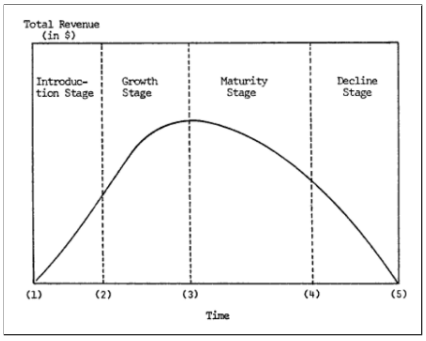

Figure 2: Basic product life cycle model by Cox (1967)

The 1976 paper, "Forget the product life cycle concept!" by Dhalla and Yuspeh proposed a number of issues with PLC theory including its failure to capture the "second lives" seen by some products, the lack of empirical testing of the model, and the fact that the theory implies that the life cycle is an independent variable that companies should adopt their marketing strategy to, instead of a variable that is dependent on marketing actions themselves. In addition, the theory saw little development of quantitative methods for determining the phases within the cycle, meaning its practicality for guiding marketing strategy beyond sales monitoring is low (Wood, 1990; Grantham, 1997; Cao & Folan, 2012). Hayes and Wheelwright (1979) argue that regardless of whether the PLC theory can be applied to every product, it does provide a useful framework for growth and development conceptualization of a new product.

PLC theory, the dominating marketing theory in the 1960's, moved to a "supporting useful role" by the late 1970's (Cao & Folan, 2012). Today, marketing literature readily uses the term *product life cycle management* (PLM), derived from the earlier PLC modeling theory, which now focuses on theories for improving the actions required from businesses during product life cycle stages, instead of focusing on the exact mathematical PLC model for the product. As mentioned in *2.1.1. Supply Chain Management*, the integration of life cycle theories into demand forecasting improves the demand planning

component of SCM (Stadtler, 2005).

2.2 Demand Estimation & Forecasting

This section gives an overview of demand forecasting and estimation techniques across business and economic literature. Some of the methods are discussed in more detail in *Section 3. Theoretical Background*. The section emphasizes recent developments in the literature that show a movement towards using a time series approach for modeling demand. Demand estimation in supply chain management and business literature has shifted primarily to time series forecasting research. This literature applies several forecasting methods to various demand and sales data with fruitful prediction, but there still is a lack of consensus or clear evidence regarding the best models to use for forecasting (Fildes et al., 2019). Within economic literature, traditional demand model systems are still used but there is a growing trend of modeling demand as a time series and employing forecasting techniques. This is seen in regard to demand estimation of consumer goods as well as energy demand.

Within SCM research, there is an abundance of research on modeling and forecasting techniques for the demand of products from customers. Conceptually, in a retail application, demand is more akin to actual sales and several studies forecast demand by building models on past sales data (Ma et al., 2016; Islek & Oguducu, 2015; Akyuz et al., 2017). The Box and Jenkins (1967) approach frequently employed in econometric applications, auto-regressive integrated moving average (ARIMA) model, along with its variations such as seasonal ARIMA (SARIMA) and ARIMA with exogenous factors (ARIMAX), were popular in retail demand literature in the 1970's and 1980's (Bechter & Rutner, 1978; Schmidt, 1979; Geurts & Kelly, 1986). More recently, the use of non-parametric methods from machine learning have become more popular. Alon et al. (2001) claim that neural network (NN) methods outperform ARIMA-based models for retail forecasting when the macroeconomic conditions, such as unemployment, interest rates, or inflation, are unusually high. Additional non-linear methods applied to retail forecasting include random forest regression trees, support vector machines, k-nearest neighbor, and Bayesian P-splines (Ali et al., 2009; Žliobaitė et al., 2012; Lang et al., 2015). Aye et al. (2015) compared 26 time series methods for forecasting South African retail data and concluded that no model outperformed the others in all scenarios but hybrid models which weighted recent data with higher importance than prior data produced better forecasts and responded better to business cycle fluctuations. Fildes et al. (2019) performed extensive literature review on retail forecasting methods and made several observations. First, they argue that most studies in the field do not have much in common methodologically and are overall not very generalizable. Second, they note

the successful results from machine learning methods but claim that the evidence is too limited to conclude the superiority of these methods over traditional ones.

Modeling and forecasting demand as a time series has been popular in SCM literature the past several decades but demand estimation in general has its origin within economic literature. Deaton (1986) describes demand analysis as being in a rare position within economics due to its long history of both theoretical and empirical research. The field of applied econometrics moved away from estimating consumer behavior around the 1980's, favoring instead the estimation of causal or treatment effects with natural and quasi-experiments (Nevo, 2011). Meanwhile, the field of industrial organization (IO) has continued research with demand estimation and is responsible for several recent developments in the study of consumer behavior in general (ibid). Literature within IO centralizes around industries with product differentiation, and mainly, two approaches are taken. Either demand is examined within the product space with demand system models, or demand is examined within the characteristics space with discrete choice models (ibid.). There is, however, a growing amount of applied research in IO and development economics that uses statistical approaches instead of economic models. These new approaches include more traditional econometric time series models as well as modern machine learning models, similar to the trends within the business literature.

Much of demand estimation for consumer goods applies the traditional modeling approaches from IO research. Food demand is a frequently studied concept within agricultural economics, where demand systems are modeled for certain markets or countries. Huang (2000) uses a household demand system with cross-sectional household survey data to model food demand and examine its elasticity within the United States. Similar studies have been performed using data from various countries (Huang, 2000; Kumar et al., 2011; Hoang, 2009; Agbola, 2003). Research in this area is abundant and has become more granular, with plenty of research focusing on demand estimation of certain food consumption industries. Alcohol demand estimation is popular, likely due to its importance for public policy (Gallet, 2007; Meng et al. 2014; Selvanathan, 2004). In addition, meat demand is quite prominent, likely due to the industry's environmental impact (Fiala, 2008; Eales, 1996; Jabarin, 2005; Taljaard et al., 2004). These studies within IO research focus on modeling consumer behavior as well as factors affecting the demand. It is, however, increasingly common to instead employ time-based econometric and machine learning models for demand in these industries. Da Veiga et al. (2014) compare two econometric time-series forecasting models, Holt-Winter (1960) and ARIMA, for demand within the dairy industry in Brazil. Several other studies apply the ARIMA modeling approach to forecast demand for food (Jia et al., 2010; Deshmukh & Paramasivam, 2016; Fattah et al., 2018).

Shukla & Jharkharia (2011) model and forecast the demand for fruits and vegetables using an ARIMA model and Mircetic et al. (2016) develop a seasonal ARIMA model applied to demand in the beverage supply chain.

Energy economics has also seen a growing integration of time series modeling. Forecasting demand for energy is a large topic, as accurate energy demand forecasting is essential for energy system planning (Neshat et al., 2018). Intersecting both energy economics and business literature, energy demand forecasting research employs both traditional time series methods, such as ARIMA, and modern machine learning methods. Forecasting energy demand has complexities similar to that of product demand forecasting due to their common nature of calendar effects, non-linearity, non-constant mean and variance, and high volatility (Azadeh et al., 2010; Neshat et al., 2018). Energy demand forecasting literature primarily involves traditional time series methods and regression techniques, often with a single research paper testing a handful of methods. For example, Shah et al. (2019) study short-term electricity demand estimation with a variety of methods including autoregressive moving average (ARMA), as well as non-parametric and vector autoregressive models (VAR). The field has seen an increasing use of machine learning techniques as well. Azadeh et al. (2010) noted the trend of applying machine learning neural network (NN) techniques and hybrid models that contain both a traditional component and a NN component. Li et al. (2018) combined ARMA and a non-parametric gradient boosting model (XGBoost) from machine learning to forecast energy demand and found that the combined model performed better than classical models. Similar results were found by Li and Zhang (2018).

2.3 Summary

Literature from SCM and economics have both shown an increase in time series forecasting approaches to demand. Modeling approaches vary, with ARIMA-based models and machine learning models both being tested and applied for demand estimation. It is particularly interesting that machine learning methods especially vary in the literature. In addition, the success from hybrid machine learning models with ARIMA-based components is a trend seen within these studies. The review indicates that there is potential value in both modeling approaches, and in the pursuit of additional exploration of their potential strengths and weaknesses. The modeling approaches used in the literature are used to inform modeling choices in this study. In addition, PLC theory is used within the data selection process of the study, since the potential existence of the PLC trend over time could relate to demand patterns over time.

3. Theoretical Background

Since demand forecasting literature shows a split preference for ARIMA-based models and machine learning models, this study pursues both. This section gives relevant background information for the models used in the study. The econometric seasonal auto-regressive integrated moving average (SARIMA) is discussed first. Following this, some general concepts from machine learning are discussed with emphasis on supervised machine learning techniques for time series prediction. Then, background information about the chosen machine learning models is discussed. The background information provides the reader relevant knowledge of the history, theory, and usage of the models.

3.1 SARIMA Theoretical Model

The ARMA model, introduced by Box and Jenkins (1976), combines auto-regressive (AR) and moving average (MA) models. A time series, y_t where t is time and y_t are real numbers can be modelled as an ARMA(p, q) process given by:

$$y_t = c + \gamma_1 y_{t-1} + \gamma_2 y_{t-2} + ... + \gamma_p y_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - ... - \theta_q \varepsilon_{t-q} \tag{3.1}$$

where c is a constant, γ_i, $i = 1, 2, ... p$ are coefficients of the auto-regressive component, and θ_j, $j = 1, 2, ... q$ are coefficients for the moving average component of the model. The auto-regressive variables are time-lags of the time series variable, up to p lags, and the moving average variables are time-lags of the error terms in the series, up to q lags.

ARMA processes cannot properly model time series that are not stationary[1] (Box & Jenkins, 1967). If a time series appears non-stationary, it may have a stochastic trend, where differencing the series can transform it into a stationary time series. The ARIMA(p, d, q) model extension allows for this, where d is the number of times the series needs to be differenced to achieve stationarity. If a series follows an ARIMA(p, d, q) model, differencing it d times allows the series to be modelled as ARMA(p, q) (Xu et al., 2010). The difference equation is given by $d(t) = y_t - \mu$ where μ is the mean of the time series. To difference y_t, it is replaced by $r(t) = y_t - d(t)$ in the model given by equation (3.1).

Under a seasonal demand modeling approach, the underlying demand process is assumed to follow SARIMA(p, d, q)(P, D, Q)s; where p, d, and q are the usual paramaters; P, D, and Q are the seasonal parameters and s represents the number of time periods in the seasonal cycle. The seasonal parameters

[1] Stationary time series have time-invariant mean and variance.

work in the same way as the regular parameters, expect they are offset to the previous seasonal cycle, for example the *Pth* seasonal lag of y_t is given by y_{t-s-P} (Cho & Lee, 2013).

3.2 Machine Learning Theory

As mentioned in *Section 2. Conceptual Background*, machine learning methods are frequently used for time series forecasting and increasingly in economics research. Machine learning methods derive from a combination of statistical and computer science research. They offer particular usefulness for prediction, modeling large data, and modeling non-parametric relationships (Athey & Imbens, 2019). In general, machine learning algorithms allow a model to be built through computational learning of the relationship between independent and dependent variables. This is done by fitting a machine learning model onto training data. Machine learning methods use varying algorithms for model fitting. After fitting onto, or learning from, the training data, the model can be used to make informed predictions from new datasets (James et al., 2013; Boehmke & Greenwell, 2020). This process contrasts econometrics which mostly involves researcher-imposed models and tuning (Athey & Imbens, 2019).

A main underlying difference between machine learning and econometrics is that machine learning focuses on optimizing predictions while many econometric applications center on the estimation of model hyperparameters (Mullainathan & Spiess, 2017). Athey and Imbens (2019) claim that leading econometrics sources like Angriske & Pischke (2008) and Wooldridge (2010) focus on the goal of identifying a functional of a joint distribution of the data, where machine learning methods focus on the goal of developing algorithms to predict variables based on other variables (Wu et al., 2018). In fact, researchers often refer to machine learning algorithms as "black boxes" because their results indicate strong detection of relationships between variables but they provide little information about the exact specifications of the relationships (Naimi & Balzer, 2018; Boehmke & Greenwell, 2020).

3.2.1 General Machine Learning Background

Time Series Forecasting as a Supervised Learning Problem

Forecasting of time series variables can easily be framed as a supervised learning problem within machine learning. Supervised learning is one of two main subcategories of machine learning methods, along with unsupervised learning. Supervised learning involves developing models using training data which supplies the model with predictors associated with example values for the target variable of prediction. In comparison, unsupervised learning simply provides data in general, and the

algorithm detects patterns or clusters for example within the dataset (Athey & Imbens, 2019; Kuhn & Johnson, 2019).

A supervised machine learning model takes input and output variables from training data to learn how to estimate an appropriate output value given new data inputs. In this study, input variables are referred to either as inputs, predictors, or features, and output variables are referred to as outputs or target variables. In a time series context, machine learning models can predict the value of the target variable in the future at time *t*, given a subset of past values of the variable. Therefore, the lagged values of the target variable itself act as inputs for each time step, *t*. In this simple reframing of time series data, several machine learning models can be utilized to predict future values of a time series variable.

It is useful to note for practical purposes that framing time series data for a supervised machine learning problem requires transformation of the time series data into a structured dataset where output values and corresponding input values are within a single row. Assuming the data has already been pooled into uniform time steps (e.g. months, quarters, etc.) then time series data will, in each row, have one value for the time series variable for a single time step. The machine learning algorithm, however, requires certain lagged values to be associated with each time step of the output variable. This can easily be done with looping algorithms and the use of a function that gives previous time lags of the main variable, such as the *shift()* function in the Pandas package in Python, or using *lag()* from the Stats package in R.

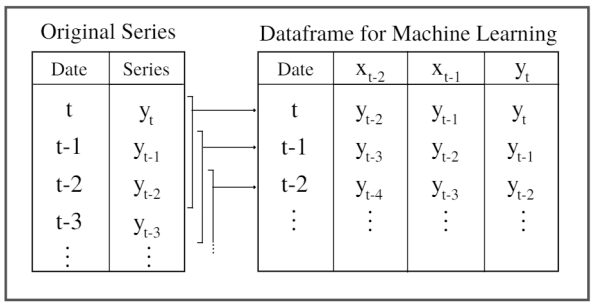

Figure 3: Transformation of time series data into supervised learning problem, author's own.

Overfitting & Underfitting

Overfitting is when a machine learning model fits the training data too well, resulting in poor prediction when applied to new data (Fawagreh et al., 2014). Overfitting is usually caused when the model identifies patterns in the training dataset that are specific to that set, i.e., non-generalizable patterns

(Kuhn & Johnson, 2019). Overfitting is a natural concern since prediction accuracy is usually the goal of machine learning to begin with. Hawkins (2004) describes overfitting as added complexity to a model that does not introduce any gain to performance. Even worse, overfitting often leads to reduced prediction performance (Kuhn & Johnson, 2019; Boehmke & Greenwell, 2020). In contrast, underfitting is when a model does not fit the training data well enough, also resulting in poor prediction performance (ibid.). The goal then is a delicate balance between underfitting and overfitting. The researcher wants the model to fit well enough onto the training data in order to learn adequate relationships within the data so that it can perform successful prediction, but not to fit so well that it cannot generalize its learnings to new data (James et al., 2013; Kuhn & Johnson, 2019).

Loss Functions

Traditional statistics and econometrics typically evaluate models using goodness-of-fit and residual tests (Athey & Imbens, 2019). Brieman et al. (2001) pointed out that these tests are not appropriate for algorithmically developed models. Today, machine learning model assessment is approached from the perspective of loss functions, where the objective of the modeling process is to minimize the model's loss function (Athey & Imbens, 2019; Boehmke & Greenwell, 2020). Loss functions compare a model's predicted values to the actual values. In practice, loss functions are built into the foundation of machine learning methods that are used regularly. Therefore, different machine learning models use different loss functions for their prediction algorithms, although customization is usually possible (Boehmke & Greenwell, 2020).

Feature Selection

In machine learning model development, feature engineering is an important component since machine learning methods allow any input variables. Feature engineering involves the selection of input variables and their preprocessing (Kuhn & Johnson, 2019). Preprocessing includes transforming variables with scaling techniques and handling missing values (ibid). Fortunately, since time series supervised learning involves a constructed dataset derived from the known time series values, it is common to have no missing values and all of the data proportionally scaled. Therefore, mainly the matter of feature selection is important.

In a time series prediction problem, the researcher needs to select which time lags to include as predictors. As mentioned in subsection *3.1. SARIMA Theoretical Model*, lag selection is part of the main model building for ARIMA-based modeling approaches. Unfortunately, machine learning models do

not offer such strict feature selection processes. Most models have built in importance functions, which measure the importance of each feature passed through the model, but these are highly dependent on the training data and model being used. These functions have a tendency to be unstable and should not be considered robust measures of the dependency between predictors and target variables, but instead act as a guide to combine with researcher discretion (Calle & Urrea, 2011; Huynh-Thu et al., 2012).

Machine learning methods for time series forecasting allow a broad array of variables to be used as inputs for the prediction problem. For example, the numbers for the month, ISO week, or day of the week can be input variables, calculated based on the timestamp used for the indexing of the data. These variables are especially relevant in time series data that shows seasonal patterns (Wang & Ramsay, 2008; Huber & Stuckenschmidt, 2020). Technically, machine learning methods offer the ability to add any input variable, but researchers should try to use judgement to determine if the predictor reasonably should be used to predict the outcome variable (Kuhn & Johnson, 2019). Overall, the goal is to use input variables that add predictive power, but concurrently minimize additional variables as too many can lead to high computation time and overfitting that affects the performance of the model on new data (ibid).

3.2.2 Random Forest Model

The random forest model is an ensemble model that was originally developed by Breiman (2001). Ensemble models are a general class of machine learning models that make predictions from a combination of sub-models (Kuncheva & Whitaker, 2003). This subsection begins with a discussion about the base model for random forest, the decision tree model, before discussing random forest as an ensemble of decision trees.

A decision tree is within the class of tree-based models. They are non-parametric and work as an algorithm to partition the feature space [2] into smaller groups. This is done through recursive splitting using splitting rules. Beginning at the top of the tree, also called the root node, with the entire available feature space, the algorithm searches for the best input variable to partition the remaining data into one of two regions such that the sum of squared errors (SSE) between the prediction and the actual output value is minimized. In a continuous prediction case, an inequality separates the two regions. Since the decision tree has several offshoots of nodes, features should be split more than once within the tree (Hastie et al., 2009; Boehmke & Greenwell, 2020). An example of a decision tree

[2] The feature space is the vector space of all input variables.

for continuous data can be seen in Figure 4 where the input variables *industry*, *population*, *wet days*, *temperature* and *wind* are used to predict the target variable *pollution* (Vala, 2019).

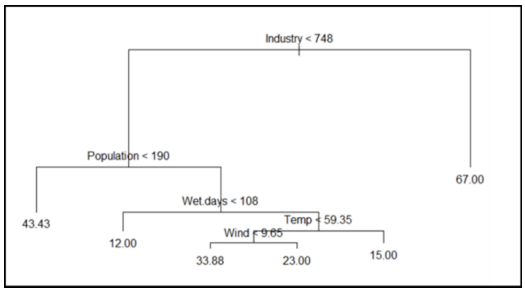

Figure 4: Decision tree visualization by Vala (2019).

Decision trees have the benefits of being non-parametric, requiring little pre-processing of data, and allowing missing values. They do however underperform in prediction accuracy. Deeper trees have the best predictions but are outperformed by most machine learning methods and also suffer from high variance (Boehmke & Greenwell, 2020).

Since the random forest model is an ensemble model of decision trees, it uses the same loss function, the sum of squared errors. Ensemble models improve upon simple learning models, such as decision trees, by combining several weaker models. One main approach to combining submodels is bootstrap aggregation, also referred to as bootstrapping or bagging (Breiman, 2001; Fawagreh et al., 2014). This method was derived by Breiman with the random forest model (2001). Bootstrapping builds each submodel using a randomly drawn sample from the training dataset. This random sampling is with replacement, meaning the samples for different submodels overlap each other. The random forest algorithm builds N decision trees from bootstrapped samples of the training dataset, where N is a hyperparameter for the number of trees to include in the model. Then the random forest algorithm selects its final prediction by taking the mean of the predictions from all N decision trees (Breiman, 2001; Sapp et al., 2014).[3]

The random forest algorithm has several factors that reduce overfitting. Combining predictions from multiple submodels and using the random sampling bootstrapping method adds robustness. In addition, Breiman (2001) included additional randomization into the random forest model within the decision tree building. For random forests applied to discrete data, Breiman added a conditional probability as the splitting decision to add randomness (Breiman, 2001; Fawagreh et al., 2014). Today, in practice, and in the case of continuous prediction instead of classification, additional randomiza-

[3] Note that when random forest is used for classification, the final prediction takes the mode of the predictions from the N decision trees instead of the mean.

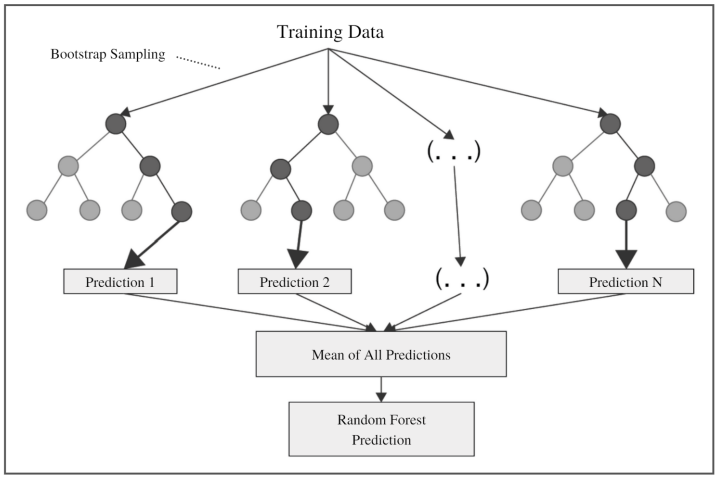

Figure 5: Random forest model visualization, inspired by Chakure (2020).

tion is used in random forest ensembles by setting a hyperparameter for the number of randomly chosen input variables the algorithm evaluates for each splitting decision. The parameter is called *mtry* in R and Python and the "rule of thumb" is to use either the square root, or a third, of the total number of input variables (Boehmke & Greenwell, 2020). Another hyperparameter that reduces the overfitting of random forest is *max depth* which limits the maximum depth of decision trees (ibid).

The feature selection method within random forest method uses the mean decrease accuracy (MDA) which involves randomly permuting the values of an input variable values and measuring the change in prediction error, using the out-of-bag principle (Breiman, 2001; Calle & Urrea, 2011; Boehmke & Greenwell, 2020). The out-of-bag error involves using the data that is not in a bootstrap sample to measure the error in the model. As mentioned in *3.2.1. General Machine Learning Background*, feature selection processed in machine learning are not statistically stable and the results should be assessed using critical judgement from the researcher. Calle & Urrea (2011) as well as Janitza & Hornung (2018) validated this in the case of random forest's out-of-bag method for assessing feature importance.

3.2.3 Extreme Gradient Boosting Model

Gradient boosting methods are another example of an ensemble method. Instead of the bootstrapping method of submodel combination, they use boosting, another popular subclass of ensemble models (Boehmke & Greenwell, 2020). Within the past 20 years, "boosting" methods have become an important subclass of ensemble methods (Wu et al., 2008). Boosting involves adding an additional submodel sequentially and updating the final model prediction with each iteration. The general

method is called gradient boosting because it follows the algorithm of gradient descent and “boosts” the process with a new weak model at each step (Bühlmann & Hothorn, 2007; Li, 2016). Gradient boosting in general is an algorithmic technique, so unlike the random forest model, it is not restricted to decision tree submodels. Freund & Shapire (1997) first applied boosting to gradient descent algorithms and today there are several variations.

Several gradient boosting models use decision trees as their base, including extreme gradient boosting which is regularly called XGBoost. The XGBoost model, originally developed by Tiqui Chen, can be visualized exactly like the random forest model. The difference lies in the fact that XGBoost is built sequentially while random forests build independent decision trees in parallel (Chen & Guestrin, 2016). In contrast to the random forest method which takes the mean of the estimates of all decision trees at the end of the algorithm, gradient boosting ensembles take the mean after each additional decision tree estimate is made, updating the final estimate in an iterative fashion. In addition, random forest using bootstrapping, random sampling with replacement, where XGBoost performs random sampling without replacement. The loss function for XGBoost applied to continuous variables is the SSE, since this is the loss function used for decision tree models with continuous variables (ibid).

Extreme gradient boosting, and the popular package XGBoost, is commonly praised in data science; in fact, there are articles theorizing why XGBoost wins so many data science competitions ("Tree Boosting", 2017). In addition, it often outperforms random forest within academic studies, some of which were mentioned in Section 2. A few features from XGBoost lend to its predictive power. For one, as trees are added to the model they are built using information the model learned from the previous trees. The newer trees are built giving priority to features that the previous tree ranked as the most important (Boehmke & Greenwell, 2020).

The XGBoost model has the ability to reach accurate predictions with less computation power than the random forest due to its sequential boosting nature, but it also has the potential of this causing overfitting quite quickly (Boehmke & Greenwell, 2020). The learning rate parameter *eta* can help avoid this by setting the scale at which each tree contributes to the model, in the fashion of gradient descent where the learning rate represents the size of the step at each iteration (Chen & Guestrin, 2016). A smaller learning rate can help the model stop before it starts overfitting but comes at the cost of computation time (Boehmke & Greenwell, 2020). In addition, one of XGBoost’s hyperparameters, *gamma* is a regularization parameter for the minimum loss reduction required to make another split. The model also offers several sampling hyperparameters that allow for different random sampling

methods, for example by feature or by tree. XGBoost has many optional hyperparameters, which could explain why it is able to achieve great prediction success in several cases (Chen & Guestrin, 2016).

3.2.4 Support Vector Regression Model

Support vector regression (SVR) is the application of the general support vector machine (SVM) model for continuous variables. Muller et al. (2005) claim that support vector machines show excellent performance for time series prediction. SVM was developed throughout the 1990's, beginning with Vapnik et al. (1992) and continually developed by Vapnik and colleagues[4] at AT&T Bell Laboratories (Smola & Schölkopf, 2004).

Generally, the SVM method maps training data to points in space and finds separating hyperplanes between data points from different classes of the output variable, trying to maximize the width of the margin between classes (Boehmke & Greenwell, 2020). In a binary classification case, this can be easily envisaged, such as the example in Figure 6. The idea is that the separation identifies the decision boundary between prediction decisions. The SVM algorithm decreases computation time by relaxing the requirement of perfect separation by hyperplanes, allowing a "cloud" of separation (ibid).

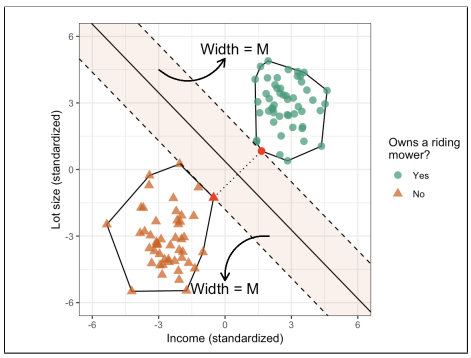

Figure 6: Visualization of a SVM boundary for a binary classification case by Boehmke & Greenwell (2020).

In the case of continuous variables, SVR utilizes an ϵ-insensitive loss function, which, like all machine learning methods, the model attempts to minimize. The ϵ-insensitive loss function is given by

$$L_\epsilon = max(0, |r(X,y)|) - \epsilon \tag{3.2}$$

where ϵ is a threshold parameter set by the researcher on the separating region, $r(X,y)$ is the resid-

[4] (Vapnik et al., 1993; Vapnik & Cortez, 1995; Vapnik et al., 1995; Vapnik et al., 1997)

ual for the prediction associated with the training example with $X = (x_1 + x_2 + ... + x_p)$ as input and y as output where $r(X,y) = y - f(X)$, and $f(X) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$. Data points that satisfy $r(X,y) \pm \epsilon$ form the *support vectors* which define the margin. Points inside the support vectors, within the separating "cloud", are ignored by the algorithm for the learning process (Vapnik & Cortez, 1995, Smola & Schölkopf, 2004). This loss function allows separating regions to be non-strict which generally improves robustness of the model to outliers (Boehmke & Greenwell, 2020).

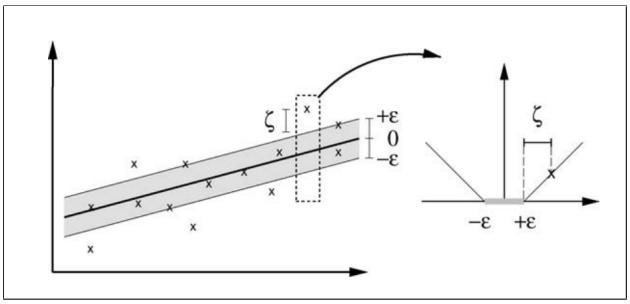

Figure 7: Visualization of an SVR epsilon boundary by Smola & Schölkopf (2002).

This concept is extended to non-linear cases using kernel functions (Smola & Schölkopf, 2004). Then the support vectors are the points where the residuals satisfy $r(X,y) \pm \epsilon$ within the kernel-induced feature space (Boehmke & Greenwell, 2020). The use of kernel functions by SVM is called the "kernel trick," which avoids the explicit mapping required from linear learning algorithms that can be challenging to apply to nonlinear data (Hofmann et al., 2008).

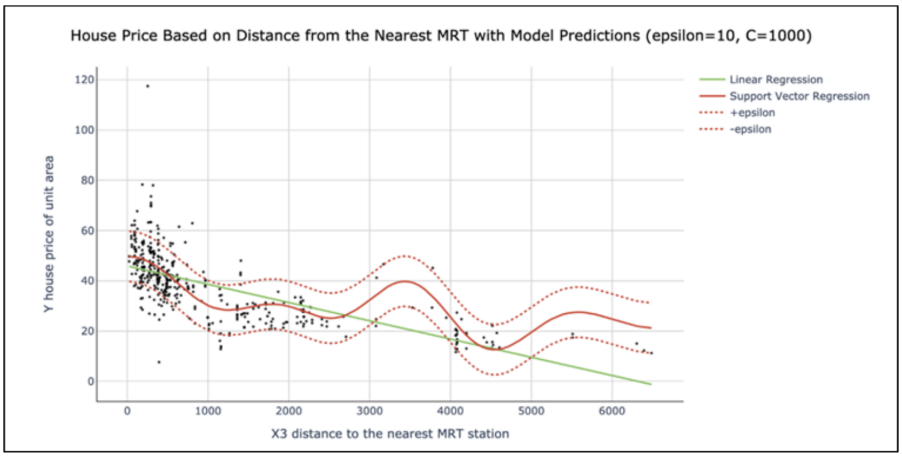

Figure 8: SVR example compared to linear regression from Boehmke & Greenwell (2020).

The kernel trick decreases computation time and allows for a multi-dimensional version of linear computation (Boehmke & Greenwell, 2020). SVM is actually within the class of kernel methods in

statistical learning (Hastie et al., 2009). The kernel function is a dot product of vectors in a high-dimensional feature space. Within the high-dimensional feature space the estimation methods are linear, but the kernel evaluations themselves can be used linearly if the kernel is positive definite, allowing the actual high-dimensional computations to be avoided (ibid). The kernel function can be chosen by the researcher and easily passed as a built-in parameter in the SVM packages in R and Python. Popular examples of kernel functions for classification problems include linear, polynomial, or radial kernels.

The SVM package offers hyperparameters that can optimize the model's prediction. This includes *epsilon* to adjust the size of the separation boundary as described above. In addition *cost* parameter controls how much the separation boundary is allowed to bend. Lower *cost* values result in smoother boundaries and higher *cost* values allow for better prediction (Smola & Schölkopf, 2004).

Unlike random forest and XGBoost, SVM does not have a built-in feature importance calculation. Feature importance is less important for SVM because it still performs well with many input variables (Chen & Li, 2010). Even so, a researcher can manually test how much the SVM performance reacts to a feature being removed.

3.2.5 Stacked Hybrid Model

Stacked ensemble models, or meta-learners, combine different base models to form a single second-layer prediction model. The stacking method can combine models that differ in their prediction algorithm since the individual models are trained prior to the stacked prediction is made (Boehmke & Greenwell, 2020). Wolpert (1992) first developed the stacked model concept, calling it stacked generalization.

A simple model stacking procedure utilizes the bootstrapping concept from random forest. A model can be trained using the random forest algorithm where the dataset used for prediction contains the target variable along with the fitted values from the base models as input variables. Using a random forest algorithm is beneficial because it grows trees concurrently and in a parallel fashion, taking the mean prediction at the end (Naimi & Balzer, 2018).

Stacked models, along with all machine learning models, are "black box" algorithms, i.e., the researcher has no definitive indication of how each input variable contributes to the final prediction (Naimi & Balzer, 2018). The additional layer of stacked modeling naturally adds more ambiguity in this regard. Furthermore, Naimi & Balzer (ibid.) point out that there is no supporting theory that the

estimator developed in a stacked model is consistent or asymptotically normal. Nevertheless, prediction accuracy is found to be improved with stacked models in many case studies, some of which were highlighted in *2.2. Demand Estimation & Forecasting* (ibid.). Some benefits of a stacked model include improved prediction accuracy, reduced risk of overfitting, and minimized parametric assumptions (Naimi & Balzer, 2018). The flexibility of a stacked learner can reduce the risk of bias arising from regression misspecification (Naimi & Balzer, 2018). Furthermore, Van der Laan et al. (2007) showed within the confines of their study that stacked models perform at least as well as their best performing base model.

4. Research Goals

The study is exploratory in nature, with the purpose of exploring the potentials of machine learning methods for forecasting demand in the specific use-case of data from Ericsson's telecommunications software. Due to the established statistical robustness of ARIMA-based models and the seasonal nature of demand data, a SARIMA model is used as a benchmark. Then the outlined machine learning methods are explored for the purpose of an initial analysis of their potential to outperform the SARIMA model in prediction of future demand with this data.

The single machine learning models explored in the study are the bootstrap ensemble method, random forest; the boosting ensemble method, extreme gradient boosting; and the kernel method, support vector regression. These are chosen by the review of relevant demand forecasting literature along with a general goal of selecting simple models with strong performance history for an initial analysis. This goal removed the neural network model from the selection of choices, which was found in several studies in the literature, due to its unsupervised nature. The findings in the literature that showed success of a hybrid model motivate the choice of using a stacked machine learning model with a SARIMA component. The stacked model uses the SARIMA fitted values along with the fitted values of the best-performing machine learning model to build a stacked random forest model. A summary of the chosen models for the study is provided in Figure 9.

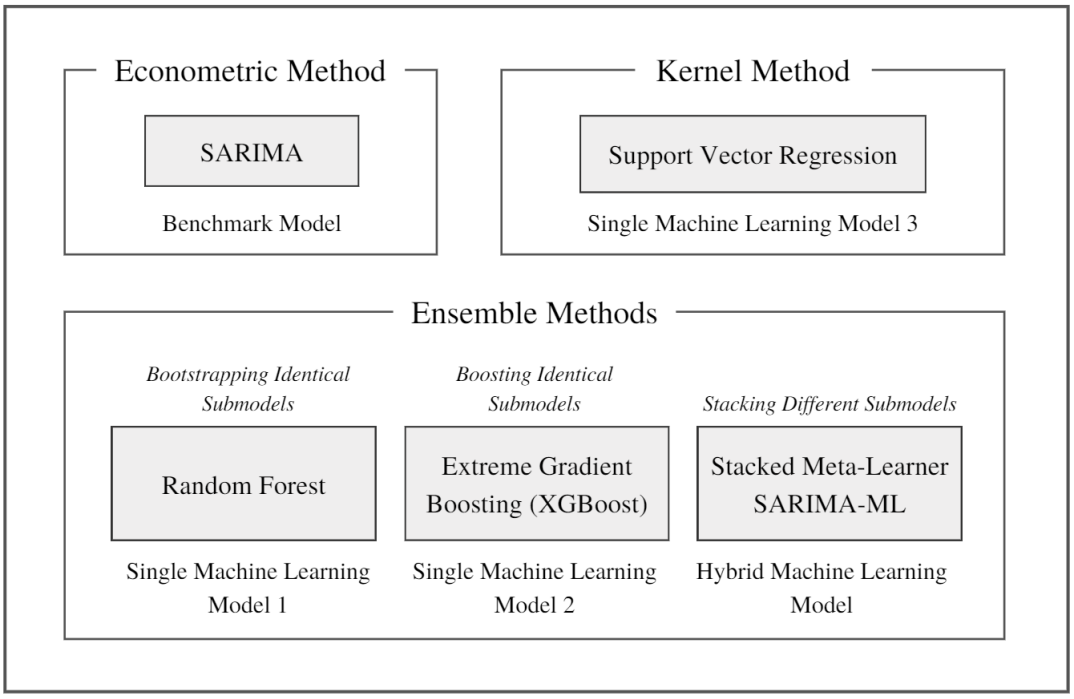

Figure 9: Summary of model choices for the study

Although no specific hypotheses are tested, the literature review guides natural expectations for the exploration. Findings validate in several use cases that SARIMA modeling performs well, and so it is expected to predict demand better than random guessing in this case as well. Furthermore, findings indicate that machine learning methods might perform better than the SARIMA model, although they will likely provide less substantial insight into the relationship between predictor variables and the target variable. In addition, literature indicates that the hybrid stacked model should perform better than single models. The study explores this in the case of this specific data by forecasting a specified period for all models and comparing the error of their predictions.

5. Data

This section first outlines the dataset used for the study. Following this, data handling is discussed prior to data selection. Then, the training, validation, and testing sets for forecasting are discussed.

5.1 Data Sources

Ericsson's Software Supply unit handles the global delivery for all software products Ericsson sells through licenses to customers as well as Ericsson subsidiaries. Ericsson's software offerings relate to telecommunications; therefore, most of the customers are telecom network providers. The software products are likely to be used implicitly by the average consumer through their connection to local networks, but the products are unlikely to be recognized by the average consumer.

All delivered software products are included within Ericsson's Software Gateway system, a cloud-based platform holding the code for software that is uploaded from the product development units within Ericsson, making it available to customers and internal units for download. There is also a License Registrar cloud-based system which contains the information of the licenses sold for software products and packages. Within this system, customer profiles are held with their specific right-to-use entitlements based on sold functionality and capacity.

Ericsson's Software Gateway system, containing all deliveries of software from Ericsson, is the source of the dataset analyzed in this thesis. The data is referred to as *downloads* in this study, as each delivery is a download of a software product to the system of a customer organization. One row of data corresponds to one software product download. Therefore, for example, a customer downloading a package with three products will produce three lines of data within the dataset. This is another reason why *download* is the chosen term for the study. In addition, it is important to note the nature of time in the Software Gateway data, with timestamps for each download, allowing for time-dynamic analysis of downloads for each software product.

This dataset is conceptually appropriate for demand estimation. Sales data for the software products would be an inferior measure for demand of Ericsson's software products since sales in this case correspond to right-to-use licenses which can be enterprise-wide for the customer organization. In comparison, download data more closely represents the demand for usage of the product. Better insights into true demand could be obtained through granular usage data of the products (e.g., data regarding how often a software product is actually used by a customer), but this data is not available.

The variables available for each delivery in the dataset include the start and finish time of the download, the product number, and the R-State[5]. It also includes the functional designation and product line of the product that was downloaded as well as the customer ID, customer name, and the download method[6]. It is important to note that since customers are organizations, the customer details do not contain any personal data. Furthermore, Ericsson follows all GDPR[7] regulations with data handling.

For the study, the data used is time series data derived from the Software Gateway downloads. A dataset is created that contains counts of deliveries within binned time periods. Daily, weekly, and monthly binned download counts are initially analyzed, but the study uses weekly aggregated data. Although the extra variables that are included in the Software Gateway data are removed from the main testing dataset, they are used within the initial analysis of the data, particularly for examining potential causes of outliers in the binned download data over time.

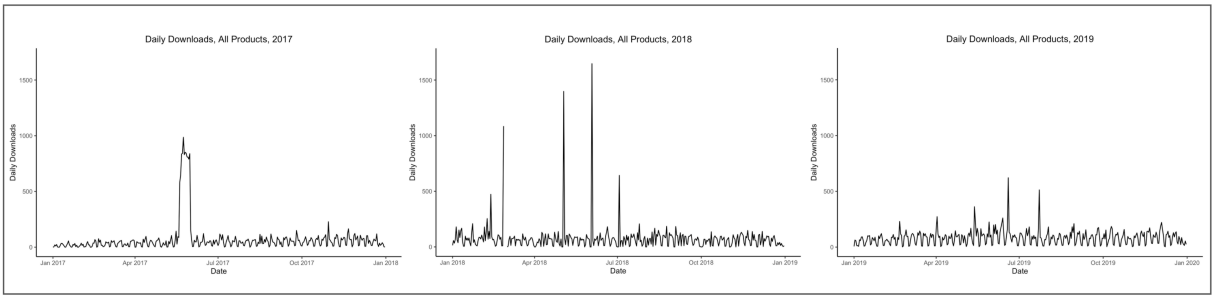

Figure 10: Daily binned downloads of Ericsson's software products, 2017-2019, author's rendering

5.2 Outlier Analysis

Push Downloads

The push download method automatically downloads updates of software products for which a customer has entitlements[8] as soon as they are uploaded into the Software Gateway system. Push downloads from Ericsson are optional and the customer decides whether to use push downloads or not. The push delivery system requires a connection between Ericsson and the customer's database system so many customers choose not to use push delivery due to data sharing preferences.

Push downloads do not conceptually represent the demand for the software products in the same

[5]R-State is an Ericsson code for the version of the product. For example, an update to a software product would cause a new R-State to be assigned.

[6]Download methods include automatic push downloads, downloads from the Software Gateway, downloads from browser, or manual downloads from Ericsson's end

[7]General Data Protection Regulation

[8]Entitlements are determined by license agreements.

way as the rest of the downloads, since the customer does not exercise any choice in the download or its timing. Therefore, push downloads are removed from the dataset for the study. This choice removes some significant outliers from the dataset, since a product update which a customer with push downloads has entitlements to causes several downloads of that product as soon as the update is uploaded.

Download Logs

Within the download method variable, there is a method for Software Gateway pull logs. Data analysis shows that this download method is responsible for outliers in the data. One example is from the download series of the most downloaded product. There is a spike in in downloads on May 29th, 2019 which is double of the second highest download day that month. 95% of the downloads on May 29th have the download method indicating "Software Gateway pull log". These downloads are from a few customer organizations, but all within the same country and the same user code was assigned to each download. This highlights the issue with downloads that have the download method "Software Gateway pull log", as it is likely that this involves a log of past downloads which were manually inserted on May 29th. This is possibly from Ericsson's local subsidiary in the country. Since pull logs also do not represent genuine demand, this download method is also removed from the dataset.

Within-Ericsson Downloads

Downloads are also removed from Ericsson AB. This choice is made because the downloads from Ericsson AB do not conceptually represent demand from customers. It is important to note that Ericsson subsidiaries are not removed from the dataset. The majority of these are the Ericsson organization within different regions, and these subsidiaries act as customers in some ways. Furthermore, their demand for software products serves important for business analysis and also arguably represents customer demand well. Removal of Ericsson AB downloads removes some outliers, since Ericsson uses many of its own software products.

Genuine Demand Spikes

Data analysis also reveals some visual outliers in the data that are not removed from the dataset. There are a few instances of mass downloads from one customer, or a cluster of customers in one region, within a few days. After downloads that use the methods of Push and SWG Pull Status Log have been removed, along with the downlaods from Ericsson AB itself, cases of mass downloads are kept in the data because they represent a genuine demand event; the customer organizations had to

actually trigger these downloads.

Data After Adjustments

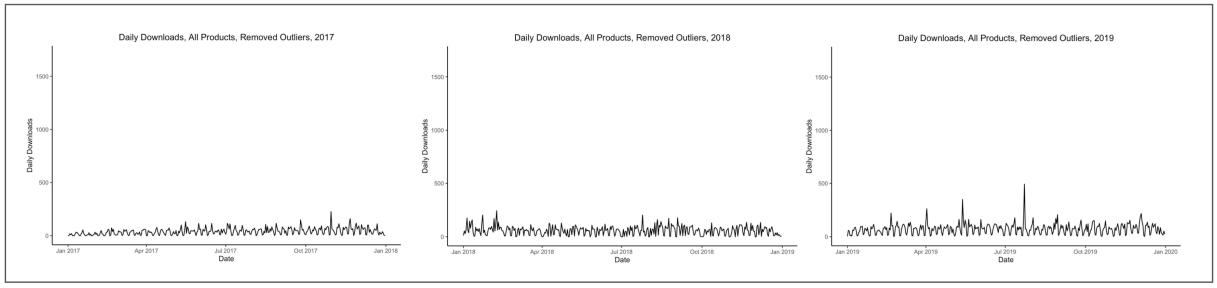

Figure 11: Daily binned downloads of Ericsson's software products with outliers removed, 2017-2019, author's rendering

5.3 Data Selection

After removing the cases above, data selection is still required since the data available in the Software Gateway system is too large to derive cohesive insights from. First, the data is reduced in terms of the time period analyzed. Next, the data is split by product, meaning the downloads for a single product are considered a single time series for analysis. Then, the data is narrowed down into a single product series, based on the timing of product life cycles and availability of data.

The time period is chosen with a preference for recent data. Since download data corresponds to downloads from employees and units within a customer company, the data could likely be affected by working structures and locations. Therefore, 2020 is not selected for this study since the COVID-19 pandemic suddenly shifted the global workforce to at-home work during 2020. This leaves 2019 as the most recent data available at the time of this study. Then, the data from 2017 and 2018 are chosen to provide adequate training data.

Product selection choices began by choosing among the top twenty overall downloaded products within the 2017 through 2019 dataset. Then, the products were narrowed down by selecting those that have downloads throughout the whole time period, since modeling and prediction would be pointless to test if this was not the case. Products which had large gaps in downloads over time were discluded, but the remaining products still had days or weeks with no downloads. Next, keeping product life cycles in mind, the products were narrowed to those that have their date of first upload to the Software Gateway system near the end of 2016. More specifically, products were strictly chosen to have their first upload in October 2016 or later. This choice is based on business practicality. Given the product life cycle concept, it is most valuable to be able to model and forecast demand for a product

over its entire lifecycle, which should begin after launch. This resulted in 6 individual products, out of which one was randomly selected.

This product is used as the main time series in the study. Figure 12 shows the downloads over time for the selected product and Figure 13 shows the square root downloads, which highlights the underlying trend in the series. A time series decomposition of the product series showing the seasonality, trend, and noise can be seen in Figure 14. The trend appears not dissimilar to the beginning of a product life cycle.

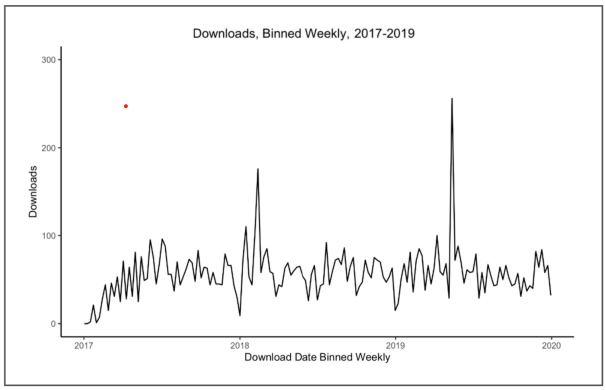

Figure 12: Download data for selected product

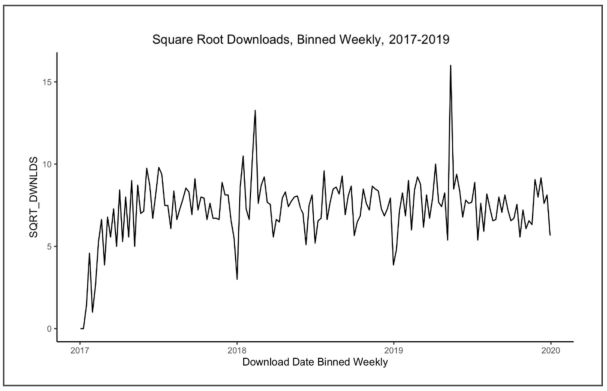

Figure 13: Square root download data for selected product

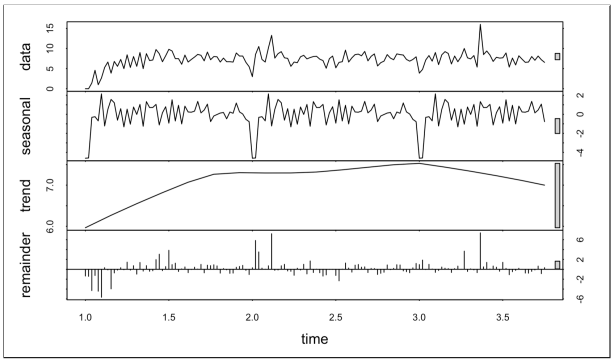

Figure 14: Time series decomposition of the download data for the selected product

5.4 Training, Validation, and Testing Sets

In a supervised machine learning problem with one target variable, a model is first fit onto the training set which contains all predictive variables and the values for the target variable. Before true deployment, the model is set to predict for the period of the test set, where the true values for the target variable are available to the researcher but not known to the model. The model's predictions can then be compared to the true values for the target variable to evaluate the model's performance.

In machine learning applications, it is recommended to have as much training data as possible, but

a general threshold for testing data is maximum 10% of the available data, leaving at least 90% of the data for training (James et al., 2013; Browne, 2000). For this study, a testing period of 12 weeks is chosen, corresponding to October through December of 2019, or the last quarter of 2019. This corresponds to a train-test split of approximately 92% and 8%, respectively. Forecasting data by quarters is a common practice in applied business applications since decisions and planning in many organizations happen quarterly. Choosing only the last quarter of 2019 maximizes the training data by allowing the first three quarters of 2019 to be used for training. In addition, the download data shows a seasonal affect around the end of December each year, so the December period is the most valuable testing period in order to see the ability of each model to detect and predict this seasonality.

In addition to a test set, it is common practice to have a validation process within the model development stage, preceding the final testing stage. A validation dataset can be interpreted similar to a test set, although it is distinctly different in that it is used for model building, i.e. feature selection and hyperparameter tuning. In contrast, the test set is exclusively for model performance evaluation (Boehmke & Greenwell, 2020; Browne, 2000). The general validation-training-testing process is visualized in Figure 15.

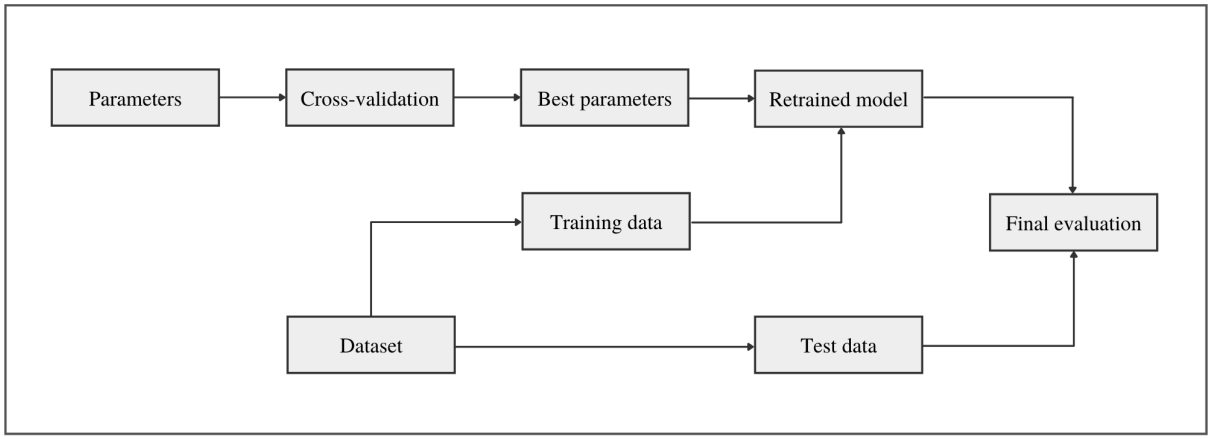

Figure 15: Machine learning model building procedure

The most popular model selection method within data science and academia is cross-validation (Bergmeir et al., 2018; Vanwinckelen & Blockeel, 2012; Zhang & Yang, 2015). With this method, the model first trains on a random subset of the training data and validates on the remaining training data. This is then repeated with different random subsample combinations (Browne, 2000). Hyperparameter tuning is performed during each cross-validation round, or fold, where several parameter combinations are tested and the one with the best model evaluation metric is chosen. There are packages in R and Python which perform cross-validation methods, including k-fold, automatically and

present the results from each fold. This can be either with a built-in cross validation component in the machine learning model package, or using a grid-search algorithm. A popular method is k-fold cross-validation which iterates this process k times. The k-fold cross-validation concept is visualized in Figure 16, where $k = 4$ and the validation and training sets in each training-validation fold are represented by the dark and light sections, respectively.

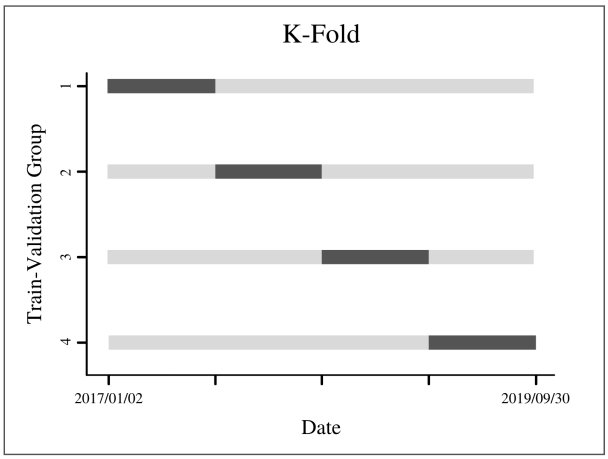

Figure 16: K-fold cross-validation technique, author's rendering inspired by Shrivastava (2020)

The repeated nature of cross validation reduces the risk of overfitting the model, which is always a concern in supervised learning problems (Cook & Ranstam, 2016; Moore, 2001). Cross-validation methods present a way to allow multiple training-validation sets to be used, without requiring a lot of extra data which, in practice, can be hard to attain. In addition, multiple validation steps increase the chance for a robust model since the final model choice is not just based on one round, and this is strengthened in cross-validation techniques involving randomness, such as k-fold (Bergmeir & Benitez, 2012).

Traditional cross-validation methods, such as k-fold, do not preserve time-ordering of the data, meaning they are not ideal for time series problems. Time dependent data theoretically violates the underlying assumptions in regular cross-validation techniques, namely the assumption that data observations are independent (Bergmeir et al., 2018; Bergmeir & Benitez, 2012). Since time series data is transformed for supervised machine learning so that lagged observations are included as predictors for each target variable value, the data is inherently serially-dependent. On top of violating the underlying theory of the cross-validation method, the use of standard cross-validation methods is not practical. In deployment of a model in a real scenario, the prediction period will truly be in the future, with no possibility of feeding the model data beyond the prediction period. Therefore, if a model is trained using overlapping time periods, it is trained with data sampling that cannot be recreated in

deployment, which is naturally ill-advised (ibid.).

Since cross-validation results in more robust model selection on average (Bergmeir & Benitez, 2012), researchers have aimed to find a way to apply cross-validation techniques to time-dependent data. Several researchers advocate for block forms of cross-validation, where data is split into time blocks and the validation set is chosen one block ahead in time for each iteration of training (Snijders, T. A., 1988; Racine, 2000; Burman, 1989; Jong, 1988). There are variations of this, but the general consensus for time series data is forward-moving validation sets through time. This general concept is often called walk-forward validation (ibid). These methods of data-splitting preserve the time-ordering of data which is crucial. The element of randomness is removed from this form of validation, but the iterations of multiple training and validation sets in general contribute to more robust model training compared to a single static training set (Roberts et al., 2017; Bergmeir & Benitez, 2012).

Using an expanding-window method of model validation, the study uses four different training- validation folds. The validation window is chosen to have a length of 12 weeks of data, so that it is the same size as the testing set. Using this method, the validation set remains the same size in each fold, while the training data increases in size with subsequent folds (Schnaubelt, 2019; Muzumdar et al., 2020). The resulting training-validation folds are displayed in Figure 17, where the validation and training sets in each training-validation fold are represented by the dark and light sections, respectively.

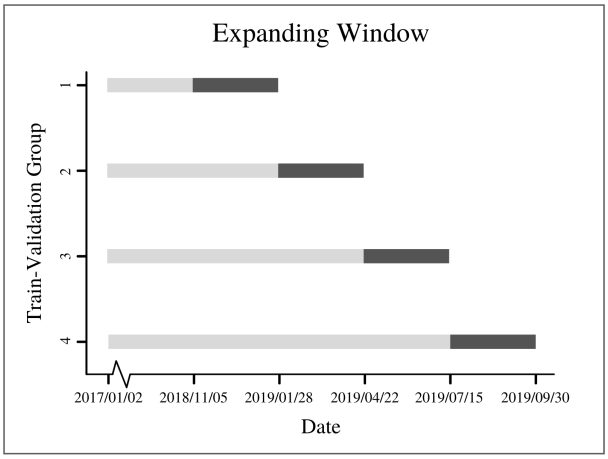

Figure 17: Training-validation groups used in the study, based on the expanding-window technique, author's rendering inspired by Shrivastava (2020)

6. Methodology

This section begins with an outline of the general methodology used to compare the prediction performance of chosen models for the selected demand data. Then, the specific methodology for the SARIMA model is outlined. Following this, the specific methodology for the machine learning model development is discussed.

6.1 General Procedure

Each of the forecasting models that are compared in the study have their own modeling "best practices." The study follows each methodology within the confines of a larger methodology in the study. The larger methodology involves using the expanding-window model-building and validation process as described in *5.4.Training, Validation, and Testing Sets*, and comparing the models' final prediction performances from the testing period to the true test set using statistical error computations.

Within each training-validation fold, each model is tuned and assessed for performance, as outlined in *5.4. Training, Validation, and Testing Sets*. For the machine learning methods, hyperparameters are tuned, meaning that several combinations of possible hyperparameters are tested and the combination producing the lowest error measure on the validation set is chosen. In addition, input variable selection occurs within each fold, using each model's standard method for feature selection in combination with logical discretion. In subsections *6.2. Econometric Model Development* and *6.3. Machine Learning Model Development*, the model-building procedure that occurs within the training-validation folds are discussed in more detail for the SARIMA and machine learning models, respectively. The best parameter and input variable choices are stored for each validation round. If the choices are different between rounds, preference is given to choices that were selected more often and associated with later validation rounds since these were selected with the use of larger training sets.

For the SARIMA model, such iterative validation is not as common, but some academic studies do borrow this machine learning principle for ARIMA-based modeling (Krishna et al., 2015; Alberg & Last, 2018). This study uses the expanding-window forward validation process for the SARIMA model as well in order to follow a general procedure. It is expected that the SARIMA model selected will be the same within each validation step because the parameters p and q are not as sensitive to different datasets as some of the hyperparameters in machine learning models. However, if it arises that different SARIMA models are selected in different folds, it could indicate that there is something wrong

with the model specification.

The final versions of each of the single models are selected based on their four expanding-window validation rounds. Then they are trained using the entire training dataset before forecasting the 12 periods of the testing dataset. The predictions are then compared to the actual download values in the testing period using the root mean square error (RMSE). RMSE is chosen based on review of literature and data science sources which indicated that the RMSE is used most often in practice for assessing machine learning predictions. Furthermore, Chai & Draxler (2014) advocate for the use of RMSE in academia, especially when residuals are expected to be non-uniform. Another common error evaluation metric, mean absolute percentage error (MAPE), is not chosen since its denominator is the true value so it is sensitive to true values that are close to zero which occurs often in the software product downloads over time (Flores, 1986; Lewison, 2020). The RMSE is given by Equation 6.1 where P is the number of time steps in the forecast period.

$$RMSE = \sqrt{\frac{1}{P}\sum_{i=1}^{P}(Predicted_i - Actual_i)^2} \tag{6.1}$$

The process is repeated for the stacked model. Model development for this model differs in that the input variables are the fitted values from the two submodels. The two submodels are the SARIMA model and whichever single machine learning model results in the lowest prediction error. The stacked model then builds a random forest model using these inputs within each expanding-window validation round where parameter tuning occurs according to the random forest parameters.

For the entirety of the study, the demand series chosen in Section 5 is transformed to its square root series. This is chosen based on the general practices in time series analysis (Brockwell & Davis, 2016). It is common to transform a time series to either log or square root (ibid.). Since the time series contains zero values, the log transformation would result in undefined values, so square root transformation is chosen instead.

6.2 Econometric Model Development

Box-Jenkins Methodology

The modeling approach first follows Box-Jenkins methodology (1967), which uses the autocorrelation and partial autocorrelation functions, ACF and PACF, for the series to identify the parameters p and q.

This process requires the series to be stationary, which can approximately be estimated from examining whether the series appears mean-reverting. If it does not appear stationary, but the differenced series does, the series may be difference stationary. The Augmented-Dickey-Fuller (ADF) test can be used to check this more concretely, where the null hypothesis is that the series is difference stationary against the hypothesis that it is stationary. If the ADF indicates that the series is stationary, then the ACF and PACF can be computed using the original series. If not, then the series can be differenced, and if this series appears stationary it can be used for the rest of the procedure (Brockwell & Davis, 2016).

The next step in Box-Jenkins method is to examine the ACF and PACF. Figure 18 displays the model decision criteria based on ACF and PACF examination. Recalling the theory for ARIMA models discussed in *Section 3.1*, an ARIMA model with $d > 0$ can be converted to an ARMA(p, q) model by differencing the series d times. Therefore, the ACF and PACF identify p and q in ARIMA(p,d,q) and d is determined by how many times the series was differenced to achieve stationarity.

Shape of ACF	Indicated Model
Exponential series decaying to 0	Auto Regressive (AR) model. PACF to be used to identify the order of the model
Alternative positive and negative spikes, decaying to 0	Auto Regressive (AR) model. PACF to be used to identify the order of the model
One or more spikes in series, the remaining are all 0	Moving Average (MA) model, identify order where plot becomes 0
A long decay after a few lags	Mixed AR & MA model
Total series is 0 or nearly 0	Data is random

Figure 18: Selection criteria for the Box-Jenkins approach, inspired by Chatterjee (2018)

After p and q are determined, model diagnostic checks are performed. First, a Wald test checks that the variables selected by p and q shows statistical significance in the model. Next, the error terms should appear as an approximate white-noise process with mean zero. The Ljung-Box test is the standard test for assessing the error terms. Passing the white-noise test and significance test, the model is accepted. If these tests are not successful, the researcher returns to further analysis, searching for previously unidentified factors in the model such as trends or co-integration.

Information Criteria Selection

The Box-Jenkins approach is still respected and used today, but researchers advocate for stricter cri-

teria before final models are chosen. In particular, the most respected selection criteria today are Akaike and Bayesian information criteria, or AIC and BIC, respectively (Brockwell & Davis, 2016). Under this selection approach, different combinations of p and q are tested in ARIMA models iteratively, and their AIC and BIC scores are recorded. Then the model producing the lowest AIC and BIC scores are selected (Kuha, 2004). Both criteria often select the same model. There is an automatic ARIMA function available in R, *auto.arima*, that selects the best ARIMA(p,d,q) based on the AIC and BIC values. This function also works for SARIMA. Since the study uses weekly data and has an annual seasonal pattern, the seasonal period is set to 52.

Overall SARIMA Procedure

Both methods are used in the study, as the Box-Jenkins procedure provides useful model diagnostics. The Box-Jenkins procedure is performed without seasonality, but just as an initial analysis of the series, and the final model selection is based on the model selected using the second procedure. If the parameters chosen for p, d, and q differ from each method, additional analysis is performed to determine if some wrong assumptions were made.[9]

Predictions are calculated using a prepared forecast function in R, using the selected model which is fitted onto the training data, and setting the forecast period to 12 periods, corresponding to the test set. The RMSE of the forecasts compared to the actual download frequencies is computed and recorded for comparison to other models.

6.3 Machine Learning Model Development

Within each training-validation round, the models are fit with 8 auto-regressive time lags as well as the date-based variables *ISO week*, *month*, and *year*. The models' respective feature importance functions are used to reduce input variables within a training round. In the case of the SVR model which does not have a feature importance method, researcher discretion in combination with the results from the feature selection of other models is used.

Recalling the discussions from *Section 3. Theoretical Background*, the hyperparameters in Tables 1 through 3 are used in each model for possible values. A grid-search algorithm is constructed where all possible parameter combinations are tested and the combination resulting in the lowest RMSE for the model is given. This process is usually recommended to be performed using cross-validation,

[9] For example if the series was trend stationary instead of difference stationary it should be detrended instead of differenced, which is discussed in Section 8.

but due to the same reasoning for avoiding cross-validation for data splitting, it is avoided here too. Instead, the entire training set available within the current training-validation group is used for the grid-search algorithm.

The choices for initial parameters come from a few sources. The insights from the literature review in *Section 3. Theoretical Background* provide some indications, particularly for the random forest parameters, the learning rate (*eta*) for XGBoost, and the possible kernel options for SVR. The remaining choices are based on consolidation of additional sources (Hastie et al., 2009; James et al., 2013; Banerjee, 2020). The stacked ensemble method uses the same initial parameters for grid-search as the regular random forest.

ntrees	300	500	1000
mtry	p	$\sqrt{p}$	$p/3$
maxdepth	3	4	5

Table 1: Random Forest Parameters for Validation

ntrees	300	500	1000
eta	0.2	0.3	0.4
maxdepth	5	6	7
gamma	0	0.4	0.8

Table 2: XGBoost Parameters for Validation

kernel	"linear"	"polynomial"	"radial"
epsilon	0.2	0.3	0.4
cost	2	4	6

Table 3: SVR Parameters for Validation

7. Results & Analysis

This section begins with the results obtained during the model development process of each model. Then the prediction results from each model are outlined. Following this, an analysis of the results is provided.

7.1 Model Development

7.1.1 SARIMA Model

The SARIMA model was identical in each training-validation round. This is as expected, since ARIMA-based modeling procedures do not require multi-step validation datasets. Only the final model details are described here, as the results are identical.

Beginning with the Box-Jenkins model selection process as an initial analysis of the time series, the ADF test indicated that the square root download series is difference stationary. Differencing the series passes visual inspection which can be seen in Figure 26 in the Appendix. The ACF and PACF for the differenced series indicate an auto-regressive model with $p = 3$, displayed in Figures 19 and 20. The fitting of an ARIMA(*3, 1, 0*) model to the data passes the Wald test and Ljung-Box tests, which are provided in the Appendix in Figure **??**. These preliminary results indicate that three auto-regressive download lags of the differenced series may be a good model, without yet considering seasonality.

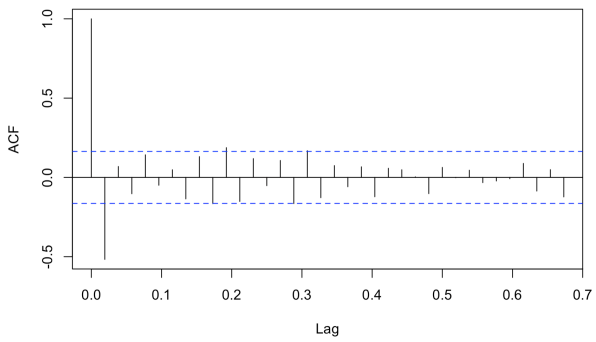

Figure 19: Autocorrelation plot

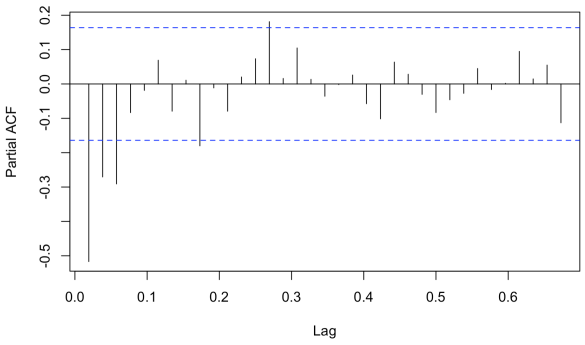

Figure 20: Partial autocorrelation plot

Following this, the results from the *auto.arima* package in R with the seasonal period set to 52 selected the model SARIMA(3,1,0)(0,1,0)52. The parameter *D=1* validates the annual seasonality that can be implied by visual inspection of the series. The lack of auto-regressive or moving average seasonal lags indicates that the values of the series just prior to last season's value are not significantly correlated with the current value. This model also passes a Wald coefficient test which is provided in Table 4 and

residual testing.

	Dependent variable:
	sqrt_series
ar1	-0.700^{***}
	(0.095)
ar2	-0.574^{***}
	(0.105)
ar3	-0.450^{***}
	(0.096)
Observations	91
Log Likelihood	-201.652
σ^2	4.865
Akaike Inf. Crit.	411.303
Note:	*p<0.1; **p<0.05; ***p<0.01

Table 4: Coefficient test for the ARIMA(3,1,0)(0,1,0)52 model selected during final test

The fact that the Box-Jenkins initial analysis and the final model selection procedure both identified the same significant number of lagged variables is reassuring. The selection of 3 auto-regressive lags implies that the demand for one week is dependent on the three previous weeks demand levels. This can be interpreted that demand for any given approximate monthly period is serially correlated. The selection of 0 for both moving average parameters indicates that the series is not significantly dependent on previous error terms.

7.1.2 Random Forest Model

For the random forest model, feature selection for lag variables differed substantially within each training-validation round. As mentioned in *Section 3. Theoretical Background*, the variable importance functions in machine learning models are not robust and should not be considered scientific. The 4th and 5th download lags were ranked as top importance in several of the folds. The first 5 lags were selected for the final model since it does not conceptually make sense that the 4th and 5th lags would hold importance and the first 3 would not. The date-based variables, *ISO week*, *month*, and *year* were consistent, with *ISO week* ranking with very high importance in each round and the others ranking very low. Conceptually, it seems logical that the *ISO week* variable would hold predictive power since there is an annual seasonal pattern and the data is binned weekly. Therefore, the *ISO*

week variable introduces the value from one season prior, acting similarly to a seasonal differencing parameter, $D = 1$, that was detected from the SARIMA modeling. Therefore, *ISO week* was kept in the final model. Feature importance computations from the 4th validation fold are found in the appendix in *Figure 27*.

The chosen parameters after tuning in all four folds are in Table 5. The final model will grew 1000 parellel decision trees, and at each node's splitting decision was fed $\sqrt{6}$ random features out of which to decide the best feature for splitting (based on the random forest loss function, SSE). Furthermore, the maximum depth of all decision trees is set to 3 nodes.

ntrees	*mtry*	*maxdepth*
1000	$\sqrt{6}$	3

Table 5: Random forest tuned hyperparameters

7.1.3 XGBoost Model

Similar to the random forest model, feature selection was not clear for the XGBoost model. The first 5 lags were identified as important in all of the validation folds, although in different orders. The 6th lag was identified as important in two of four validation rounds, but since this lag is far from the predicted value and did not show consistent importance ranking, it was not included. Therefore, 5 lags were chosen for the final model. The importance function results from the fourth training-validation can be seen in Figure 28 in the Appendix.

The chosen parameters after tuning in all four rounds are in Table 6. Similarly to the random forest model, 1000 decision trees are used in the final XGBoost model. The maximum depth of decision trees was chosen at 6 nodes. The learning rate, *eta*, and across-tree regularization parameter, *gamma*, were chosen at 0.4 and 0. *Eta* was chosen just above its "rule of thumb" value of 0.3 and the *gamma* choice of 0 means that no regularization was chosen, which is the default value.

nrounds	*eta*	*gamma*	*maxdepth*
1000	0.4	0	6

Table 6: XGBoost tuned hyperparameters

7.1.4 Support Vector Regression Model

Since the random forest and XGBoost models showed similar feature importance rankings, 5 lags and *ISO week* were also chosen for the final SVR model. The chosen parameters after tuning in all four rounds are in Table 7. The kernel function that fit the model best was polynomial. This is not

surprising since the product's life cycle trend appears polynomial. The *epsilon* parameter for the ϵ-insensitive loss function is chosen as 0.3 and *cost* are chosen as 4.

kernel	*epsilon*	*cost*
polynomial	0.3	4

Table 7: SVR tuned hyperparameters

7.2 Prediction Results

The RMSE from each model's predictions during the testing period are provided in Table 8. The predictions plotted with the actual series values are shown in Figures 21-25. Overall, the machine learning models outperformed the SARIMA model. The random forest and support vector regression models performed slightly better than the XGBoost model, but ultimately, the hybrid model performed far better than all of the models.

	SARIMA	**RF**	**XGB**	**SVR**	**Meta**
RMSE	1.3801	0.9896	1.0563	0.9977	0.4363

Table 8: RMSE from model testing

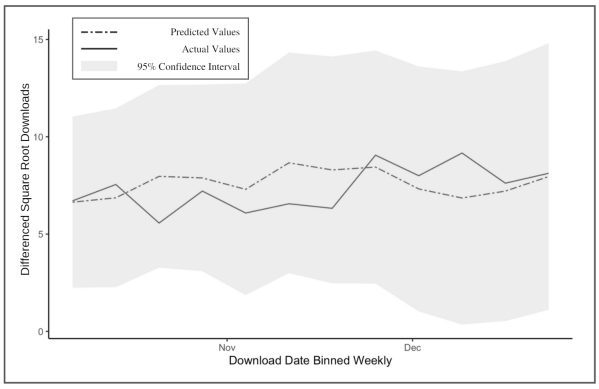

Figure 21: Predictions from SARIMA Model

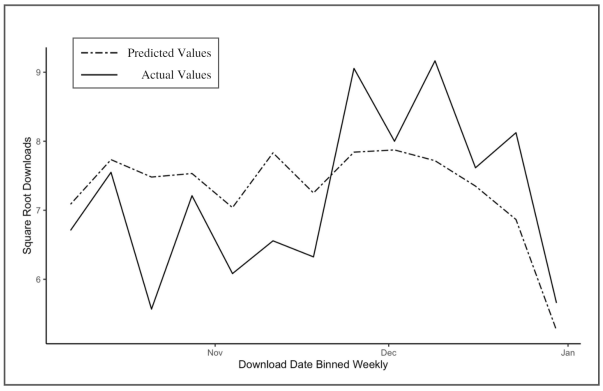

Figure 22: Predictions from random forest model

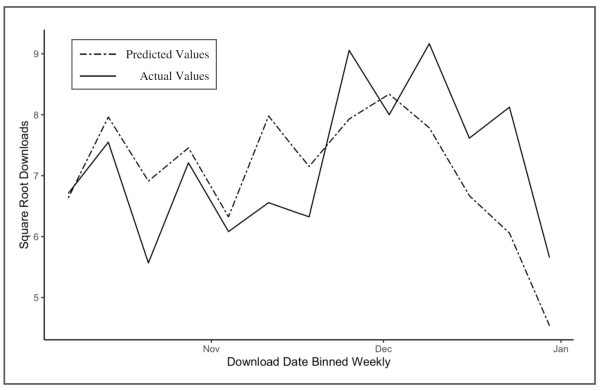

Figure 23: Predictions from XGBoost model

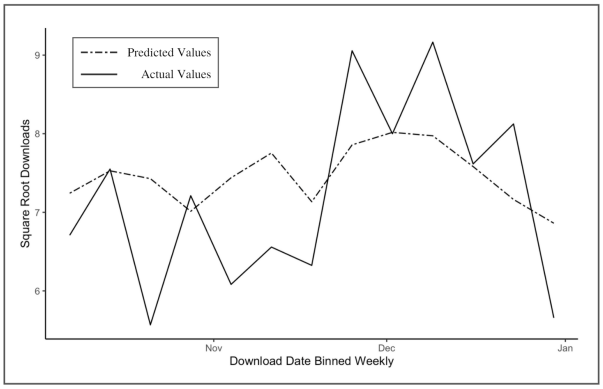

Figure 24: Predictions from SVR model

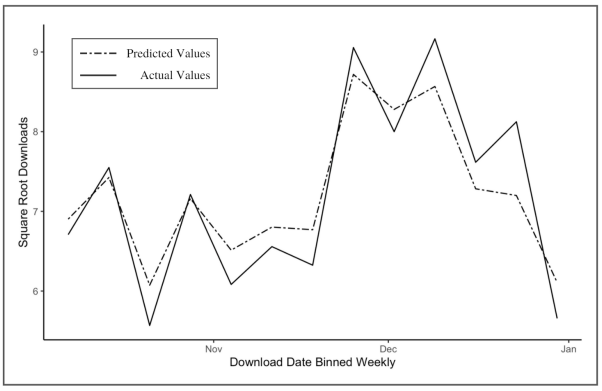

Figure 25: Predictions from stacked SARIMA-RF model

7.3 Analysis of Results

The results suggest that in the case of this data, machine learning methods provide better prediction capabilities compared to the more traditional SARIMA approach. The most significant result was that a stacked ensemble hybrid model made from both the SARIMA and random forest submodels performed substantially better than all single models. Overall, the results are not misaligned with expectations based on theory and past research, except for the subpar performance of XGBoost in comparison to the other single machine learning models.

The single machine learning models (random forest, XGBoost, and SVR) performing better than the SARIMA model is not surprising given the findings from previous studies. The flexibility of machine learning methods generally allows intricate relationship detection between inputs and outputs, as long as the models are not too overfit to the training data (Kuhn & Johnson, 2019). The success of these results indicate that overfitting of the models was at least adequately avoided to provide prediction improvement in comparison to the more traditional SARIMA model. The expanding-window validation technique, which was chosen for the purpose of avoiding overfitting in light of time series data, may have been adequate at fulfilling its role.

It is surprising that the XGBoost model performed slightly worse than the random forest and SVR models. As mentioned, XGBoost has been the winning model in several data science competitions and has also seen success in academic studies, even in direct comparison to random forest. From a theoretical perspective, as described in *Section 3*, XGBoost is an ensemble method using boosting of decision trees that are optimized based on its previous learning, making it more prone to overfitting compared to random forest. Several XGBoost hyperparameters offer the ability to offset the chance of this overfitting, therefore, a reasonable cause of the under-performance of the XGBoost model in the

study is poor hyperparameter tuning. A downside of the grid-search method for parameter tuning is that the tested parameters are chosen by the researcher using "rule of thumb". There is potential that the ideal parameters were never entered into the grid-search algorithm to begin with, resulting in unoptimized parameters. Furthermore, the *gamma* parameter depends on the combination of training data and other parameter choices, so if another parameter was mischosen, the effect could have been compounded by the choice of *gamma* (Chen & Guestrin, 2016).

The stacked hybrid model outperforming the others is in line with findings from literature as discussed in previous sections. The reduction in error from using the hybrid approach was substantial. The results perhaps validate what is emphasized in the literature; diversity in machine learning model building can prevent overfitting (Kuhn & Johnson, 2019). Using the predictions from the two models that had the most differing RMSE results as the inputs for the stacked model may have contributed to the level of success that the hybrid model achieved.

The results across all models validated the idea of seasonality in the demand for Ericsson's software products. The identification of $D = 1$ in the SARIMA model with no identification of significant auto-regressive lags ($P = 0$) could imply that the seasonality might be quite granular, i.e. the value of the series exactly one year ago is important for prediction but the values slightly before that are not statistically important. This was further validated from the machine learning models which identified *ISO week* as an important predictor but not *month*.

The model building procedures also validated what literature often highlights about the trade-off of traditional econometric approaches to machine learning approaches. Particularly in the matter of feature selection, or the selection of significant input variables. The SARIMA model detected that only three auto-regressive lags are significant but the machine learning methods detected predictive importance for the fourth and fifth lags. This exemplifies the flexibility of pattern detection that machine learning methods offer compared to econometric methods, but the reminder should remain that the relationships the machine learning methods detect should not be considered statistically significant or proven in any way (Naimi & Balzer, 2018). Ultimately, this highlights the downside of machine learning methods for both academic research and business value: at the end of the day, not much can be said about the relationships between the variables.

8. Additional Checks

SARIMA Model

Recalling the theory for ARIMA models discussed in *Section 3.1*, an ARIMA model with $d > 0$ can be converted to an ARMA(p, q) model by differencing the series d times. As an additional check of the original SARIMA(3, 1, 0)(0, 1, 0)52 model, the modeling process was repeated instead with the differenced time series. Theoretically, the expectation is that the AIC and BIC selected model should have the same parameters for the AR and MA components (p, q) , as well as seasonal parameters (P, D, Q), but that the parameter d should be zero. This was confirmed with the data, indicating additional evidence of proper model specification. If this check was not successful, there would have been indication of improper model selection and further analysis would be advised. The data was assumed difference stationary from the ADF test, but instead could be trend stationary. This could be tested with a Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test which compliments the ADF unit root test by testing trend stationarity against difference stationarity (Kwiatkowski et al., 1992). If the series was indeed trend stationary, it would need to be detrended before the modeling process instead of differenced (Brockwell & Davis, 2016).

Single Machine Learning Models

Each model was run with the differenced download series to check robustness. Expectations are that the models would perform similarly with or without differenced data, as stationarity is not a requirement for these models. Generally machine learning models do not require that the data to be differenced, but some sources claim that random forest and XGBoost models are not great at extrapolating time trends (Loh et al., 2007). Since the data was differenced for the SARIMA model, it is useful to check whether feeding the machine learning models the non-differenced data was an unfair assessment in comparison to SARIMA. The RMSE from the predictions on differenced data were very similar. The ability of the machine learning methods to perform similarly with the original data and differenced data is valuable because they can be utilized with less data preprocessing, and more importantly, they offer the potential to predict future trends relateively well, which in practice holds great value.

Stacked Model

The stacked model was also run with differenced data for the same reasoning. Instead of similar re-

sults, this caused significantly worse results. Researchers claim that stacked models perform best when the base models' predictions are less correlated (Boehmke & Greenwell, 2020). This could explain this response, since the the original model contained the SARIMA predictions for the stationary series and random forest predictions for the original squareroot series, which was determined to have stochastic trend. The robustness check with differenced data resulted in the two input variables being based on the stationary series, likely increasing their correlation. Reflecting on this further, the random forest predictions could possibly act for the model as a measure of the "trend" over time, while the SARIMA predictions could capture the mean-reverting nature in the absence of trend. Since both submodels had seasonality included (the ISO week variable for the random forest), they likely shared the responsibility in capturing the seasonal effects.

In addition, the model was tested by removing the confidence interval estimates that derived from the SARIMA prediction. Conceptually, the confidence intervals could be valuable predictors for the model, as they are bound relatively closely around the true value. This was already found in the SARIMA forecasting for this specific data. The stacked model could likely learn an approximate relationship between the lower and upper confidence intervals and then use the confidence intervals to improve the final predictions. There is no theory to support this, or research which suggests this. Therefore, the stacked model was replicated with the confidence interval estimates removed from the data. The results were worse, with a RMSE of approximately 0.51, providing evidence in support of the reasoning above.

9. Discussion, Limitations & Future Research

An overarching limitation of the study is that the models are only tested on one series. This is often the case in time series studies, which typically involve macroeconomic series that do not have the same granularity as products. Even so, only one product was tested for one time period in the study, meaning that the results should be explored further in future research by replication and extension.

In addition, the modeling process in the study gave precedence to a "fair assessment" of the models by following a general methodology, which may have resulted in some of the models being chosen under unoptimized specifications. The general methodology for the machine learning algorithms involved grid-search algorithms for choosing hyperparameters from preset "rule of thumb" values. These researcher choices involved selection of possible hyperparameter values along with choices of hyperparameters themselves. These "rule of thumb" choices may not have contained the optimum values, and may be responsible for the final results instead of the models' capabilities themselves. In practice, a data scientist or researcher using only one model could build the model in an agile fashion, returning back to original parameter choices after validation rounds to further optimize the model. Since the study was comparing models, this was avoided so as to give a "fair" assessment of each. This highlights a downside to the flexibility of machine learning methods. Their customizability provides potential for very well-built models but can also make it difficult for a researcher to find the optimal specifications. This is where the use of the highly praised cross-validation can provide great value. Completing parameter grid-search algorithms with cross-validation is the recommended approach for model building (Krstajic et al., 2016). This means cross-validation for parameter tuning within the greater confines of the cross-validation for training-validation sets. Although time series data invalidates cross-validation's underlying theory, many data scientists and even academic researchers still apply cross-validation for time series data (Bergmeir & Benítez, 2012). Therefore, a clear limitation in this study is using the sub-optimal manual grid-search without cross-validation for hyperparameter tuning. Perhaps following the studies that use cross-validation for time series data, using it within the grid-search process inside the larger confines of the expanding-window validation could have resulted in better hyperparameter tuning and different final results.

Another limitation to machine learning methods is that they do not offer statistically stable confidence intervals for predictions (Shrestha & Solomatine, 2006). Confidence intervals for forecasting hold important value for business applications, especially in supply chain forecasting for physical goods (Dalrymple, 1987). Generally, businesses with physical goods would prefer to make inventory

decisions based on the confidence intervals of demand forecasts to ensure adequate inventory. This is less important in this study where the product being analyzed has unlimited inventory, but is an important consideration for examining the use of these methods for the case of demand for physical goods.

The study findings imply that machine learning methods may provide value for demand forecasting efforts of Ericsson's Software Supply unit, and that a stacked model, particularly a hybrid model with ARIMA and machine learning bases may prove the most accurate upon further research. The findings also imply that demand for these products is not random noise and could perhaps be interpretable as a product life cycle. This was previously unknown to Ericsson's Software Supply unit and this study could be a starting point for validation of product life cycles on additional software products.

Extending from this point, Ericsson could replicate the study on data from additional products. The process could be replicated with different initial parameters in the machine learning models as well. After additional validation, the process could be implemented and tested on a rolling basis, allowing forecasting across longer time periods. Furthermore, Ericsson could extend this analysis by adding additional variables to the modeling process, i.e. exogenous variables for a SARIMAX model and additional factors within the machine learning models. Examples of additional variables could be a dummy variable indicating when new updates of a product are uploaded to the Software Gateway, and dummy variables indicating new sales licenses for large customer organizations associated with the specific software products on the license.

Due to the lack of consensus for modeling approaches to demand forecasting, exploration of models in different demand contexts presents a useful contribution towards gaining more clarity. Particular value could be interpreted from the fact that the demand series used for this study was from a non-physical good. There was no indication from literature that demand for telecommunications software would in fact show any time-dependent relationship that could be modeled, so the results provide an initial indication that a regular demand perspective can be applied in similar ways to these types of goods.

The lack of consensus regarding modeling also extends to the use of machine learning for prediction in general. As mentioned in the study, there are no strict recommendations for which models outperform which in general. Since machine learning is growing in the economics literature in general, it is important for research to continue exploring the strengths and limitations of machine learning models. The results of this study provide additional evidence in the grander scheme of required research

into this question. Future studies should continue applying different models to different time series to attempt to shed light on the limitations of different methods in different use cases.

10. Conclusion

The study provides additional evidence to the demand forecasting literature which spans several fields. It contributes to the collection of studies which employ machine learning methods for demand modeling and also to those that compare ARIMA and machine learning methods. This study compares several methods for the purpose of demand forecasting. Five individual models are developed within an expanding-window validation process. These include a benchmark SARIMA model and several machine learning models. Random forest, XGBoost, and support vector regression are each developed. Following this a stacked ensemble model is created using the SARIMA and random forest predictions.

Overall, the machine learning models showed higher forecast accuracy compared to the SARIMA model. The forecast from the SARIMA model had a RMSE of 1.38 while the single machine learning methods had RMSE between 0.98 and 1.05. Out of the single machine learning models, random forest performs best, followed by support vector regression. The most valuable contribution to this research is the success of the hybrid modeling approach which had a substantially lower forecast RMSE of 0.43. The results overall indicate that machine learning methods present good possibilities for demand forecasting, and in the case of software demand. In addition, they indicate that further exploration in this application should place emphasis on hybrid meta-learning models.

The SARIMA results provide evidence in support of the seasonal approach to demand modeling in the case of software series analyzed. The SARIMA process identified seasonality within the demand series and selected three autoregressive lags. The resulting model was SARIMA$(3, 1, 0)(0, 1, 0)52$, with weekly aggregated data and an annual seasonal period. The machine learning models detected some predictive power for the first 5 lags, as well as a variable for the ISO week, aligning with the annual seasonality detected by the SARIMA model. The difference in lag selection highlights the trade off between ARIMA-based and machine learning models.

Machine learning methods have the potential for substantially higher accuracy in forecasts compared to the ARIMA-based models, and this was validated in the case of this data. A downside of these machine learning methods, however, is their inability to provide clarity about the time series being forecasted. The importance measures of predictor variables associated with machine learning techniques are statistically unstable, so even after seeing error reduction in the model from selecting "important" features, the researcher still cannot say to what extent these variables are connected to

the target variable (Naimi & Balzer, 2018).

This was seen in the study, with the two machine learning "importance" rankings giving different results in different iterations. In contrast, SARIMA models are built from statistical foundations, and allow for regression coefficients to be estimated as well as reliable statistical tests to determine their significance. In this case, statistical backing identified that only the first three auto-regressive lags are significant, while the machine learning models detected importance for the fourth and fifth lags. The matter of the relationship between input and output variables may not be a significant concern in a time series context where the predictors are largely made from previously observed values. It does, however, become important when the researcher wants to include additional factors into the forecasting process, such as exogenous variables.

The predictive accuracy of the hybrid SARIMA-random forest model in the study is a promising result in light of this trade off. Not only did the model present potential for more accurate prediction, it presents the potential for combining the statistical clarity that the SARIMA model offers with the predictive power of "black box" machine learning models. As mentioned in this study, previous research has already found promising results from ARIMA-machine learning hybrid models. The findings of this study further validate these previous findings in a new context of software demand.

11. References

Afonja, T. (2018, July 13). Kernel Functions - Towards Data Science. Medium. https://towardsdatascience.com/kernel-function-6f1d2be6091

Agbola, F. W. (2003). Estimation of food demand patterns in South Africa based on a survey of households. Journal of Agricultural and Applied Economics, 35(1379-2016-113556), 663-670.

Ahmed, N. K., Atiya, A. F., Gayar, N. E., & El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. Econometric Reviews, 29(5-6), 594-621.

Aitken, J., Childerhouse, P., & Towill, D. (2003). The impact of product life cycle on supply chain strategy. International Journal of Production Economics, 85(2), 127–140.

Akaike, H. (1998). Information theory and an extension of the maximum likelihood principle. In Selected papers of hirotugu akaike (pp. 199-213). Springer, New York, NY.

Akyuz, A. O., Uysal, M., Bulbul, B. A., & Uysal, M. O. (2017, July). Ensemble approach for time series analysis in demand forecasting: Ensemble learning. In 2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA) (pp. 7-12). IEEE.

Alberg, D., & Last, M. (2018). Short-term load forecasting in smart meters with sliding window-based ARIMA algorithms. Vietnam Journal of Computer Science, 5(3), 241-249.

Ali, Ö. G., Sayın, S., Van Woensel, T., & Fransoo, J. (2009). SKU demand forecasting in the presence of promotions. Expert Systems with Applications, 36(10), 12340-12348.

Alon, I., Qi, M., & Sadowski, R. J. (2001). Forecasting aggregate retail sales:: a comparison of artificial neural networks and traditional methods. Journal of retailing and consumer services, 8(3), 147-156.

Athey, S., & Imbens, G. (2019). Machine Learning Methods That Economists Should Know About. Annual Review of Economics, 11(1), 685–725.

Au, K. F., Choi, T. M., & Yu, Y. (2008). Fashion retail forecasting by evolutionary neural networks. International Journal of Production Economics, 114(2), 615-630.

Aye, G. C., Balcilar, M., Gupta, R., & Majumdar, A. (2015). Forecasting aggregate retail sales: The case of South Africa. International Journal of Production Economics, 160, 66-79.

Azadeh, A., Asadzadeh, S., & Ghanbari, A. (2010). An adaptive network-based fuzzy inference system for short-term natural gas demand estimation: Uncertain and complex environments. Energy Policy, 38(3), 1529–1536.

Bajari, P., Nekipelov, D., Ryan, S. P., & Yang, M. (2015). Machine learning methods for demand estimation. American Economic Review, 105(5), 481-85.

Banerjee, P. (2020, July). A Guide on XGBoost hyperparameters tuning. Kaggle. https://www.kaggle.com/prashant111/a-guide-on-xgboost-hyperparameters-tuning

Bechter, D. M., & Rutner, J. L. (1978). Forecasting with statistical models and a case study of retail sales. Economic Review, 63(Mar), 3-11.

Bergmeir, C., & Benítez, J. (2012). On the use of cross-validation for time series predictor evaluation. Information Sciences, 191, 192–213.

Bergmeir, C., Hyndman, R. J., & Koo, B. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. Computational Statistics & Data Analysis, 120, 70-83

Boehmke, B., & Greenwell, B. (2020). Hands-On Machine Learning with R (1st ed.). Chapman & Hall.

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (1976). Time series analysis: forecasting and control. John Wiley & Sons.

Boysen, O. (2012). A food demand system estimation for Uganda.

Breiman, Leo. 1996b. “Stacked Regressions.” Machine Learning 24 (1). Springer: 49–64.

Brockwell, P. J., Davis, R. A. (2016). Introduction to time series and forecasting. springer

Brown, R. G. (2004). Smoothing, forecasting and prediction of discrete time series. Courier Corporation.

Browne, M. (2000). Cross-Validation Methods. Journal of Mathematical Psychology, 44(1), 108–132.

Burman, P. (1989). A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods. Biometrika, 76(3), 503-514.

Calle, M. L., & Urrea, V. (2011). Letter to the editor: stability of random forest importance measures. Briefings in bioinformatics, 12(1), 86-89

Cao, H., & Folan, P. (2012). Product life cycle: the evolution of a paradigm and literature review from 1950-2009. Production Planning & Control, 23(8), 641–662.

Cerqueira, V., Torgo, L., & Mozetic, I. (2020). Evaluating time series forecasting models: an empirical study on performance estimation methods. Machine Learning, 109(11), 1997–.

Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. Geoscientific model development, 7(3), 1247-1250.

Chakure, A. (2020, November 6). Random Forest Regression - The Startup. Medium. https://medium.com/swlh/random-forest-and-its-implementation-71824ced454f

Chatterjee, S. (2018, February 5). Time Series Analysis Using ARIMA Model In R. DataScience+. https://datascienceplus.com/time-series-analysis-using-arima-model-in-r/

Chen, F., & Li, F. (2010). Combination of feature selection approaches with SVM in credit scoring. Expert Systems with Applications, 37(7), 4902–4909.

Chen, T. (2014). Introduction to boosted trees. University of Washington Computer Science, 22(2014), 115.

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794)

Cho, D., & Lee, Y. (2013). The value of information sharing in a supply chain with a seasonal demand process. Computers & Industrial Engineering, 65(1), 97–108.

Cook, J. A., & Ranstam, J. (2016). Overfitting. Journal of British Surgery, 103(13), 1814-1814.

Cryer, J. D., & Chan, K. S. (2008). Time series analysis: with applications in R. Springer Science & Business Media.

Da Veiga, C. P., Da Veiga, C. R. P., Catapan, A., Tortato, U., & Da Silva, W. V. (2014). Demand forecasting in food retail: A comparison between the Holt-Winters and ARIMA models. WSEAS transactions on business and economics, 11(1), 608-614.

Dalrymple, D. J. (1987). Sales forecasting practices: Results from a United States survey. International journal of Forecasting, 3(3-4), 379-391.

Deane-Mayer, Zachary A., and Jared E. Knowles. 2016. CaretEnsemble: Ensembles of Caret Models. https://CRAN.R-project.org/package=caretEnsemble.

Deaton, A. (1986). Chapter 30 Demand analysis. Handbook of Econometrics, 3, 1767–1839.

Deshmukh, S. S., & Paramasivam, R. (2016). Forecasting of milk production in India with ARIMA and VAR time series models. Asian Journal of Dairy & Food Research, 35(1), 17-22.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A., & Vapnik, V. (1997). Support vector regression machines. Advances in neural information processing systems, 9, 155-161.

Eales, J. (1996). A symmetric approach to Canadian meat demand estimation. Journal of Agricultural and Resource Economics, 368-380.

Ellinger, A., Saenz, M., & Koufteros, X. (2015). Literature Reviews in Supply Chain Management and Logistics. (12th ed.). Emerald Publishing Limited.

Fattah, J., Ezzine, L., Aman, Z., El Moussami, H., & Lachhab, A. (2018). Forecasting of demand using ARIMA model. International Journal of Engineering Business Management, 10, 1847979018808673

Fawagreh, K., Gaber, M. M., & Elyan, E. (2014). Random forests: from early developments to recent advancements. Systems Science & Control Engineering: An Open Access Journal, 2(1), 602-609.

Fiala, N. (2008). Meeting the demand: an estimation of potential future greenhouse gas emissions from meat production. Ecological economics, 67(3), 412-419.

Fildes, R., Ma, S., & Kolassa, S. (2019). Retail forecasting: Research and practice. International Journal of Forecasting.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences, 55(1), 119-139.

Gallet, C. A. (2007). The demand for alcohol: a meta-analysis of elasticities. Australian Journal of Agricultural and Resource Economics, 51(2), 121-135.

Geary, S., Disney, S. M., & Towill, D. R. (2006). On bullwhip in supply chains—historical review, present practice and expected future impact. International Journal of Production Economics, 101(1), 2-18.

Geurts, M. D. and I. B. Ibrahim (1975), "Comparing the Box-Jenkins Approach with the Exponentially Smoothed Forecasting Model with an Application to Hawaii Tourists," Journal of Marketing Research, 12 (May), 182-7.

Geurts, M. D., & Kelly, J. P. (1986). Forecasting retail sales using alternative models. International Journal of Forecasting, 2(3), 261-272.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. Machine learning, 63(1), 3-42.

Giard, V., & Sali, M. (2013). The bullwhip effect in supply chains: a study of contingent and incomplete literature. International Journal of Production Research, 51(13), 3880-3893.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

Hawkins, D. M. (2004). The problem of overfitting. Journal of Chemical Information and Computer Sciences, 44(1), 1–12.

Hoang, L. V. (2009). Estimation of food demand from household survey data in Vietnam. Depocen. Work Pap. Ser, 12, 26.

Hofmann, T., Schölkopf, B., & Smola, A. J. (2008). Kernel methods in machine learning. The annals of statistics, 1171-1220

Hossain, M., Ahmed, S., & Uddin, M. (2021). Impact of weather on COVID-19 transmission in south Asian countries: An application of the ARIMAX model. The Science of the Total Environment, 761, 143315–143315.

Hsiao, J., & Shieh, C. (2006). Evaluating the value of information sharing in a supply chain using an ARIMA model. International Journal of Advanced Manufacturing Technology, 27(5), 604–609.

Huang, K. S., & Lin, B. H. (2000). Estimation of food demand and nutrient elasticities from household survey data (No. 1488-2016-123635).

Huber, J., & Stuckenschmidt, H. (2020). Daily retail demand forecasting using machine learning with emphasis on calendric special days. International Journal of Forecasting, 36(4), 1420-1438.

Huynh-Thu, V. A., Saeys, Y., Wehenkel, L., & Geurts, P. (2012). Statistical interpretation of machine learning-based feature importance scores for biomarker discovery. Bioinformatics, 28(13), 1766-1774.

Intihar, M., Kramberger, T., & Dragan, D. (2017). Container Throughput Forecasting Using Dynamic Factor Analysis and ARIMAX Model. Promet, 29(5), 529–542.

Islek, I., & Oguducu, S. G. (2015, June). A retail demand forecasting model based on data mining techniques. In 2015 IEEE 24th International Symposium on Industrial Electronics (ISIE) (pp. 55-60). IEEE.

Ismail Shah, Hasnain Iftikhar, Sajid Ali, & Depeng Wang. (2019). Short-Term Electricity Demand Forecasting Using Components Estimation Technique. Energies (Basel), 12(13), 2532–.

Jabarin, A. S. (2005). Estimation of meat demand system in Jordan: an almost ideal demand system. International Journal of consumer studies, 29(3), 232-238.

Jain, A., Menon, M. N., & Chandra, S. (2015). Sales forecasting for retail chains.

James A. Johnson, & Ernest H. Oksanen. (1977). Estimation of Demand for Alcoholic Beverages in Canada From Pooled Time Series and Cross Sections. The Review of Economics and Statistics, 59(1), 113–118.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 18). New York: springer.

Janitza, S., & Hornung, R. (2018). On the overestimation of random forest's out-of-bag error. PloS one, 13(8), e0201904

Jia, J. S., Zhao, J. Z., Deng, H. B., & Duan, J. (2010). Ecological footprint simulation and prediction by ARIMA model—a case study in Henan Province of China. Ecological Indicators, 10(2), 538-544

Jong, P. D. (1988). A cross-validation filter for time series models. Biometrika, 75(3), 594-600.

Kapoor, S., Madhok, P., & Wu, S. (1981). Modeling and Forecasting Sales Data by Time Series Analysis. Journal of Marketing Research, 18(1), 94-100.

Kembro, J., Selviaridis, K., & Näslund, D. (2014). Theoretical perspectives on information sharing in supply chains: a systematic literature review and conceptual framework. Supply chain management: An international journal.

Krishna, V. B., Iyer, R. K., & Sanders, W. H. (2015, October). ARIMA-based modeling and validation of consumption readings in power grids. In International Conference on Critical Information Infrastructures Security (pp. 199-210). Springer, Cham.

Krstajic, D., Buturovic, L. J., Leahy, D. E., & Thomas, S. (2014). Cross-validation pitfalls when selecting and assessing regression and classification models. Journal of cheminformatics, 6(1), 1-15.

Kuha, J. (2004). AIC and BIC: Comparisons of Assumptions and Performance. Sociological Methods & Research, 33(2), 188–229.

Kuhn, M., & Johnson, K. (2019). Feature engineering and selection: A practical approach for predictive models. CRC Press

Kumar, P., Kumar, A., Parappurathu, S., & Raju, S. S. (2011). Estimation of demand elasticity for food commodities in India. Agricultural Economics Research Review, 24(347-2016-16882), 1-14.

Kuncheva, L. I., & Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning, 51(2), 181–207.

Kwiatkowski, D., Phillips, P. C., Schmidt, P., & Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?. Journal of econometrics, 54(1-3), 159-178.

Laan, Mark J. van der, Eric C. Polley, and Alan E. Hubbard. 2003. “Super Learner.” Statistical Applications in Genetics and Molecular Biology 6 (1).

Lang, S., Steiner, W. J., Weber, A., & Wechselberger, P. (2015). Accommodating heterogeneity and nonlinearity in price effects for predicting brand sales and profits. European Journal of Operational Research, 246(1), 232-241.

LeDell, Erin, Stephanie Sapp, Mark van der Laan, and Maintainer Erin LeDell. 2014. Package "Subsemble". https://CRAN.R-project.org/package=subsemble.

Lee, H. L., So, K. C., & Tang, C. S. (2000). The value of information sharing in a two-level supply chain. Management science, 46(5), 626-643.

Li, C. (2016). A gentle introduction to gradient boosting.
http://www.ccs.neu.edu/home/vip/teach/MLcourse/4_ boosting/slides/gradient_boosting.Pdf

Li, C., Zheng, X., Yang, Z., & Kuang, L. (2018). Predicting short-term electricity demand by combining the advantages of arma and xgboost in fog computing environment. Wireless Communications and Mobile Computing, 2018.

Li, P., & Zhang, J. S. (2018). A new hybrid method for China's energy supply security forecasting based on arima and xgboost. Energies, 11(7), 1687.

Loh, W. Y., Chen, C. W., & Zheng, W. (2007). Extrapolation errors in linear model trees. ACM Transactions on Knowledge Discovery from Data (TKDD), 1(2), 6-es.

Ma, S., Fildes, R., & Huang, T. (2016). Demand forecasting with high dimensional data: The case of SKU retail sales forecasting with intra-and inter-category promotional information. European Journal of Operational Research, 249(1), 245-257.

Meade, P., & Rabelo, L. (2004). The technology adoption life cycle attractor: Understanding the dynamics of high-tech markets. Technological Forecasting & Social Change, 71(7), 667–684.

Meinshausen, N., & Ridgeway, G. (2006). Quantile regression forests. Journal of Machine Learning Research, 7(6).

Meng, Y., Brennan, A., Purshouse, R., Hill-McManus, D., Angus, C., Holmes, J., & Meier, P. S. (2014). Estimation of own and cross price elasticities of alcohol demand in the UK—A pseudo-panel approach using the Living Costs and Food Survey 2001–2009. Journal of health economics, 34, 96-103.

Meyr H., Wagner M., Rohde J. (2002) Structure of Advanced Planning Systems. In: Stadtler H., Kilger C. (eds) Supply Chain Management and Advanced Planning. Springer, Berlin, Heidelberg.

Michelle Grantham, L. (1997). The validity of the product life cycle in the high-tech industry. Marketing Intelligence & Planning, 15(1), 4–10.

Miragliotta, G. (2006). Layers and mechanisms: A new taxonomy for the bullwhip effect. International Journal of production economics, 104(2), 365-381.

Mircetic, D., Nikolicic, S., Maslaric, M., Ralevic, N., & Debelic, B. (2016). Development of S-ARIMA model for forecasting demand in a beverage supply chain. Open Engineering, 6(1).

Moore, A. W. (2001). Cross-validation for detecting and preventing overfitting. School of Computer Science Carneigie Mellon University.

Müller, K. R., Smola, A. J., Rätsch, G., Schölkopf, B., Kohlmorgen, J., & Vapnik, V. (1997). Predicting time series with support vector machines. In International Conference on Artificial Neural Networks (pp. 999-1004). Springer, Berlin, Heidelberg

Muzumdar, A., Modi, C., & Vyjayanthi, C. (2020, October). An Efficient Regional Short-Term Load Forecasting Model for Smart Grid Energy Management. In IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society (pp. 2089-2094). IEEE.

Naimi, A., & Balzer, L. (2018). Stacked generalization: an introduction to super learning. European Journal of Epidemiology, 33(5), 459–464.

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. Frontiers in neurorobotics, 7, 21.

Neshat, N., Hadian, H., & Behzad, M. (2018). Nonlinear ARIMAX model for long –term sectoral demand forecasting. Management Science Letters, 8(6), 581–592.

Nevo, A. (2011). Empirical Models of Consumer Behavior. Annual Review of Economics, 3(1), 51–75.

Pannakkong, W., Sriboonchitta, S., & Huynh, V. (2018). An Ensemble Model of Arima and Ann with Restricted Boltzmann Machine Based on Decomposition of Discrete Wavelet Transform for Time Series Forecasting. Journal of Systems Science and Systems Engineering, 27(5), 690–708.

Patrick Bajari, & C. Lanier Benkard. (2005). Demand Estimation with Heterogeneous Consumers and Unobserved Product Characteristics: A Hedonic Approach. The Journal of Political Economy, 113(6), 1239–1276.

Patrick Bajari, Denis Nekipelov, Stephen P. Ryan, & Miaoyu Yang. (2015). Machine Learning Methods for Demand Estimation. The American Economic Review, 105(5), 481–485.

Pavlyshenko, B. (2019). Machine-Learning Models for Sales Time Series Forecasting. Data (Basel), 4(1), 15–.

Pektaş, A., & Kerem Cigizoglu, H. (2013). ANN hybrid model versus ARIMA and ARIMAX models of runoff coefficient. Journal of Hydrology (Amsterdam), 500, 21–36.

Peter Bühlmann, & Torsten Hothorn. (2007). Boosting Algorithms: Regularization, Prediction and Model Fitting. Statistical Science, 22(4), 477–505.

Pindyck, R. S. and D. L. Rubenfeld (1976), Econometric Models and Economic Forecasts. New York, McGraw- Hill Book Company. Polley, Eric, Erin LeDell, Chris Kennedy, and Mark van der Laan. 2019. SuperLearner: Super Learner Prediction. https://CRAN.R-project.org/package=SuperLearner.

R. Samsudin, A. Shabri and P. Saad, 2010. A Comparison of Time Series Forecasting using Support Vector Machine and Artificial Neural Network Model. Journal of Applied Sciences, 10: 950-958.

Racine, J. (2000). Consistent cross-validatory model-selection for dependent data: hv-block cross-validation. Journal of econometrics, 99(1), 39-61.

Raghunathan, S. (2001). Information sharing in a supply chain: A note on its value when demand is nonstationary. Management science, 47(4), 605-610.

Roberts, D. R., Bahn, V., Ciuti, S., Boyce, M. S., Elith, J., Guillera-Arroita, G., ... & Dormann, C. F. (2017). Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography, 40(8), 913-929.

Sahin, E. K. (2020). Assessing the predictive capability of ensemble tree methods for landslide susceptibility mapping using XGBoost, gradient boosting machine, and random forest. SN Applied Sciences, 2(7), 1-17

Salisu, A. A., & Ayinde, T. O. (2016). Modeling energy demand: Some emerging issues. Renewable and Sustainable Energy Reviews, 54, 1470-1480.

Samsudin, R., Saad, P., & Shabri, A. (2011). River flow time series using least squares support vector machines. Hydrology and Earth System Sciences, 15(6), 1835–1852.

Sapp, S., van der Laan, M. J., & Canny, J. (2014). Subsemble: an ensemble method for combining subset-specific algorithm fits. Journal of applied statistics, 41(6), 1247-1259.

Schmidt, J. R. (1979). Forecasting state retail sales: econometric vs. time series models. The Annals of Regional Science, 13(3), 91-101.

Schnaubelt, M. (2019). A comparison of machine learning model validation schemes for non-stationary time series data (No. 11/2019). FAU Discussion Papers in Economics.

Selvanathan, E. A., & Selvanathan, S. (2004). Economic and demographic factors in Australian alcohol demand. Applied Economics, 36(21), 2405-2417.

Sendhil Mullainathan, & Jann Spiess. (2017). Machine Learning: An Applied Econometric Approach. The Journal of Economic Perspectives, 31(2), 87–106.

Shi, X., Wong, Y. D., Li, M. Z. F., Palanisamy, C., & Chai, C. (2019). A feature learning approach based on XGBoost for driving assessment and risk prediction. Accident Analysis & Prevention, 129, 170-179

Shrestha, D. L., Solomatine, D. P. (2006). Machine learning approaches for estimation of prediction interval for the model output. Neural Networks, 19(2), 225-235.

Shrivastava, S. (2020, January 17). Cross Validation in Time Series - Soumya Shrivastava. Medium. https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4

Shukla, M., & Jharkharia, S. (2011). ARIMA models to forecast demand in fresh supply chains. International Journal of Operational Research, 11(1), 1-18.

Smola, A., & Schölkopf, B. (2004). A tutorial on support vector regression. Statistics and Computing, 14(3), 199–222.

Snijders, T. A. (1988). On cross-validation for predictor evaluation in time series. In On model uncertainty and its statistical implications (pp. 56-69). Springer, Berlin, Heidelberg.

Stadtler, H. (2005). Supply chain management and advanced planning—basics, overview and challenges. European Journal of Operational Research, 163(3), 575–588.

Taljaard, P. R., Alemu, Z. G., & Van Schalkwyk, H. D. (2004). The demand for meat in South Africa: an almost ideal estimation. Agrekon, 43(4), 430-443.

Tan, K. C., Lyman, S. B., & Wisner, J. D. (2002). Supply chain management: a strategic perspective. International journal of operations & production management.

Team, R. C. (2013). R: A language and environment for statistical computing. https://cran.microsoft.com/snapshot/2014-09-08/web/packages/dplR/vignettes/xdate-dplR.pdf

Towill, D. R., Zhou, L., & Disney, S. M. (2007). Reducing the bullwhip effect: Looking through the appropriate lens. International Journal of Production Economics, 108(1-2), 444-453.

Tree Boosting With XGBoost – Why Does XGBoost Win "Every" Machine Learning Competition? (2017, October 22). Synced. https://syncedreview.com/2017/10/22/tree-boosting-with-xgboost-why-does-xgboost-win-every-machine-learning-competition/

Vala, K. (2019, March 18). Tree-Based Methods: Regression Trees - Towards Data Science. Medium. https://towardsdatascience.com/tree-based-methods-regression-trees-4ee5d8db9fe9

Van der Laan MJ, Rose S. Targeted learning: causal inference for observational and experimental data. New York: Springer; 2011

Van der Laan, M. J., Polley, E. C., & Hubbard, A. E. (2007). Super learner. Statistical applications in genetics and molecular biology, 6(1).

Van Nguyen, T., Zhou, L., Chong, A., Li, B., & Pu, X. (2020). Predicting customer demand for remanufactured products: A data-mining approach. European Journal of Operational Research, 281(3), 543–558.

Vanwinckelen, G., & Blockeel, H. (2012). On estimating model accuracy with repeated cross-validation. In BeneLearn 2012: Proceedings of the 21st Belgian-Dutch conference on machine learning (pp. 39-44).

Wang, A. J., & Ramsay, B. (1998). A neural network based estimator for electricity spot-pricing with particular reference to weekend and public holidays. Neurocomputing, 23(1-3), 47-57.

Wang, X., & Disney, S. (2016). The bullwhip effect: Progress, trends and directions. European Journal of Operational Research, 250(3), 691–701.

Wei, W. W. (2006). Time series analysis. In The Oxford Handbook of Quantitative Methods in Psychology: Vol. 2.

William E. Cox. (1967). Product Life Cycles as Marketing Models. The Journal of Business (Chicago, Ill.), 40(4), 375–384.

Wolpert, David H. 1992. “Stacked Generalization.” Neural Networks 5: 241–59. Wood, L. (1990). The end of the product life cycle. Journal of Marketing Management, 6(2), 145–.

Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Yu, P., Steinberg, D. (2008). Top 10 algorithms in data mining. Knowledge and Information Systems, 14(1), 1–37.

Xu, S., Chan, H. K., & Zhang, T. (2019). Forecasting the demand of the aviation industry using hybrid time series SARIMA-SVR approach. Transportation Research Part E: Logistics and Transportation Review, 122, 169-180.

Xu, X., Qi, Y., & Hua, Z. (2010). Forecasting demand of commodities after natural disasters. Expert Systems with Applications, 37(6), 4313–4317.

Yu, Z., Yan, H., & Cheng, T. C. E. (2002). Modelling the benefits of information sharing-based partnerships in a two-level supply chain. Journal of the Operational Research Society, 53(4), 436-446.

Zhang, G. (2003). Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing (Amsterdam), 50, 159–175.

Zhang, Y., & Yang, Y. (2015). Cross-validation for selecting a model selection procedure. Journal of Econometrics, 187(1), 95-112.

Žliobaitė, I., Bakker, J., & Pechenizkiy, M. (2012). Beating the baseline prediction in food sales: How intelligent an intelligent predictor is?. Expert Systems with Applications, 39(1), 806-815.

12. Appendix

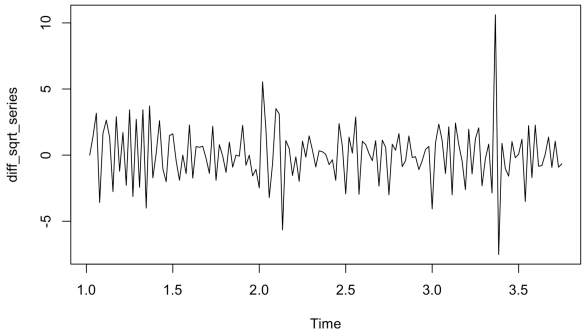

Figure 26: Differenced time series

	Dependent variable:
	sqrt_series
ar1	-0.658^{***}
	(0.078)
ar2	-0.553^{***}
	(0.084)
ar3	-0.358^{***}
	(0.079)
Observations	143
Log Likelihood	-206.988
σ^2	1.059
Note:	*p<0.1; **p<0.05; ***p<0.01

Table 9: Coefficient test from Box-Jenkins methodology ARIMA model on entire training set

Ljung-Box Test	
Q	26.723
df	26
p-value	0.424
Model df	3
Total lags used	29

Table 10: Ljung-Box residual test from Box-Jenkins methodology ARIMA model on entire training set

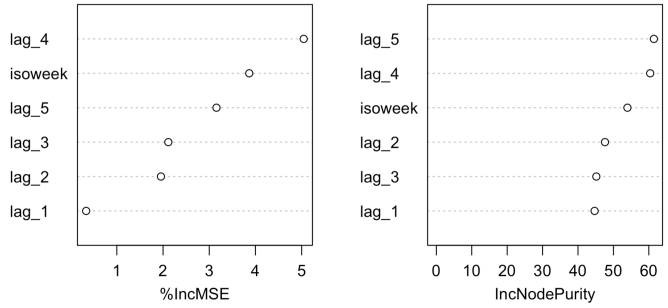

Figure 27: Feature importance from random forest, 4th validation fold

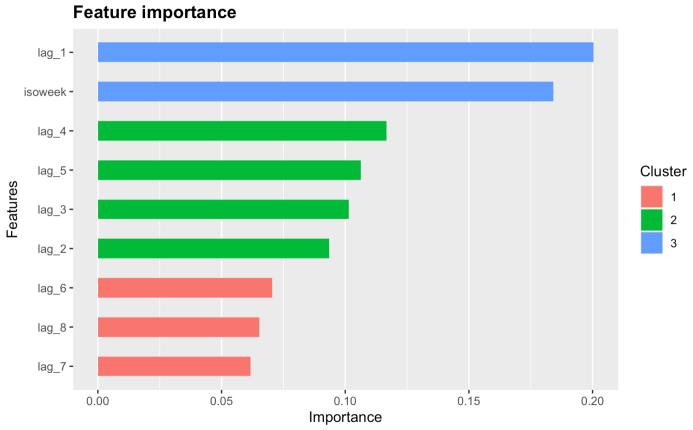

Figure 28: Feature importance from XGBoost, 4th validation fold