Aalto University
School of Science
Master's Programme in Mathematics and Operations Research

Tuukka Sarvi

Predicting product sales in retail store chain

Master's Thesis
Helsinki, October 20, 2020

Supervisor: Prof. Harri Ehtamo
Advisor: M.Sc. Antti Syväniemi

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Tuukka Sarvi

Title Predicting product sales in retail store chain

Degree programme Mathematics and Operations Research

Major Systems and Operations Research **Code of major** SCI3055

Supervisor Prof. Harri Ehtamo

Advisor M.Sc. Antti Syväniemi

Date 20.10.2020 **Number of pages** 63 **Language** English

Abstract

The ability to predict sales of products in different stores as accurately as possible is critical to the survival and growth of a retail chain. Many operational decisions such as pricing, space allocation, assortment planning, ordering and inventory management are related directly to product sales forecast. Increased accuracy of sales predictions enables better strategic, tactical and operational decisions and growth in profits.

In this thesis, three separate sales prediction tasks are examined: the prediction of future monthly sales of 1) existing products in store assortment, 2) products new to a store which have been sold before in other stores and 3) new products. A machine learning framework for sales prediction is developed that includes feature extraction from raw data, model training using cross-validation and hyperparameter optimization. XGBoost, Lasso regression and rule-based benchmark models are implemented. A dataset from Finnish retail store chain is used to assess the accuracy of the model predictions.

Results indicate that the machine learning models are able to make more accurate predictions than the rule-based benchmark models especially in the case of existing products. In the case of new store products, models outperform the benchmark by a small margin. In the case of new products, models outperform the benchmark by a large margin but overall accuracy as measured by relative root mean squared error is modest. XGBoost is more accurate than Lasso regression for existing product sales prediction; in other tasks they perform comparably.

The machine learning framework developed can be used for recurring large-scale sales prediction of existing store products, new store products and new products for retailers that have access to historical monthly sales data. It can incorporate new data sources, such as price, customer and promotions data, and new types of machine learning algorithms.

Keywords retail, sales prediction, machine learning, XGBoost, Lasso, feature engineering

Aalto University
School of Science

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Tuukka Sarvi

Työn nimi Vähittäiskauppaketjun tuotemyynnin ennustaminen

Koulutusohjelma Matematiikka ja operaatiotutkimus

Pääaine Systeemi- ja operaatiotutkimus **Pääaineen koodi** SCI3055

Työn valvoja Prof. Harri Ehtamo

Työn ohjaaja MMM Antti Syväniemi

Päivämäärä 20.10.2020 **Sivumäärä** 63 **Kieli** Englanti

Tiivistelmä

Laadukkaat myyntiennusteet ovat keskeinen tekijä vähittäiskauppaketjun menestyksen ja kasvun kannalta. Useat operatiiviset toiminnot kuten hinnoittelu, tilasuunnittelu, valikoimapäätökset, tilaukset ja varastonhallinta riippuvat suoraan tuotteiden myyntiennusteesta. Tarkka myyntiennuste mahdollistaa parempien päätöksien tekemisen strategisella, taktisella ja operatiivisella tasolla, mikä näkyy suoraan yhtiön tuloksessa.

Tässä diplomityössä käsitellään kolmea erillistä myynnin ennustamistehtävää: kuukausimyynnin ennustaminen 1) kaupan valikoimaan kuuluvien tuotteiden, 2) uusien tuotteiden, joita on myyty muissa kaupoissa, ja 3) uutuustuotteiden tapauksessa. Työssä toteutetaan mallikokonaisuus, joka kattaa piirteiden louhimisen raakadatasta sekä koneoppimismallien ristiinvalidoinnin ja hyperparametrien optimoinnin. Työssä käytetään XGBoost- ja Lasso regressio -koneoppimismalleja sekä sääntöpohjaisia vertailumalleja. Mallien tuottamien ennusteiden tarkkuutta arvioidaan suomalaisen vähittäiskauppaketjun dataa käyttäen.

Tulokset viittaavat siihen, että koneoppimismallit tuottavat tarkempia ennusteita kuin sääntöpohjaiset vertailumallit erityisesti kaupan valikoimaan kuuluvien tuotteiden tapauksessa. Kaupalle uusien tuotteiden tapauksessa koneoppimismallit suoriutuvat vain hieman paremmin kuin vertailumalli. Uusien tuotteiden tapauksessa koneoppimismallit tuottavat selvästi tarkempia ennusteita kuin vertailumalli, mutta kokonaistarkkuus jää vaatimattomalle tasolle. XGBoost on tarkempi kuin Lasso regressio kaupan valikoimaan kuuluvien tuotteiden tapauksessa; muiden ennustetehtävien tapauksessa XGBoost ja Lasso regressio suoriutuvat yhtä hyvin.

Kehitettyä mallikokonaisuutta voidaan käyttää valikoimaan kuuluvien ja uusien tuotteiden myynnin ennustamiseen vähittäiskauppaketjuissa olettaen, että kuukausitason myyntidataa on saatavilla. Malliin voidaan lisätä uusia datalähteitä kuten hinta-, asiakas- ja kampanjadataa sekä uusia koneoppimisalgoritmeja.

Avainsanat vähittäiskauppa, myynnin ennustaminen, koneoppiminen, XGBoost, Lasso, piirteiden louhinta

Acknowledgements

First, I want to express my gratitude to my supervisor Harri Ehtamo and advisor Antti Syväniemi for their guidance, feedback and patience during this project. I want to express my thanks to Houston Analytics, the company this thesis was made for, and my colleagues in there for their continuous support and feedback during this project. It was very interesting to write this thesis as a member of active product development team engaged in practical application of retail analytics. Also, I want to thank Ville Pohjolainen for valuable feedback and comments.

Finally, I wish to thank my family and friends for their support during my studies.

Helsinki, 20.10.2020

Tuukka Sarvi

Contents

1 Introduction

Predicting the sales of different products is a central analytics task in retail business. The ability to accurately predict sales translates directly to actions that can improve the bottom line. Accurate knowledge of future sales potential of products can help in supply chain and inventory management, staffing, assortment planning, space planning, new product launches (Zhang et al., 2016), revenue management, pricing and promotions planning among other things (Fildes et al., 2019). Accurate forecasts are important because retailers want minimize the capital that is tied in stocks and at the same time ensure adequate product availability for customers (Belt, 2017). In the food industry, sales prediction is important due to the short shelf-life of many of the products which can lead to loss of income both due to shortage and surplus (Tsoumakas, 2019). Forecast errors lead directly to poorer service and higher costs (Fildes et al., 2019). All in all, forecasting demand can be considered as one of the crucial capacities of any enterprise (Christou, 2012).

Despite introduction of more advanced methods including econometric modeling and nonlinear methods in the research literature, use of such methods is not common in practice (Fildes et al., 2019). Sales prediction for food items is typically done by managers in a more or less arbitrary manner (Tsoumakas, 2019). Judgemental intervention on top of model-based forecasts remains a significant element in retail forecasting practice. Univariate extrapolative methods such as exponential smoothing and linear regressions are more established than more complex nonlinear methods such as nonlinear regressions, non- or semi-parametric regressions, fuzzy algorithms and machine learning models (Fildes et al., 2019). According to Belt (2017) forecasting is mostly automated but manual corrections and improvements to forecasting are often made.

The scale of the sales prediction task can be very big: physical stores can hold over 200 000 distinct items, Walmart for example has roughly 5000 stores in the United States, meaning that 1 billion unique forecasts need to be prepared in the desired forecast horizon. In order for business to function, these forecasts need to be generated quickly and efficiently (Seaman, 2018). The data hierarchies involved are very large, with tens of thousands unique products in multilevel product hierarchies and thousands of stores. According to Kolassa (2018), even with recent advances

in numerical methods the optimal combination of forecasts in various aggregation levels remains out of reach for realistic retail data hierarchies.

1.1 Research objectives

The objective of this thesis is to investigate the following three research questions:

1. How can future monthly sales of existing products be predicted in a large scale across multiple stores, assuming sales is independent of the availability of other products in store assortment?
2. How can monthly sales of new store products be predicted, assuming these products have been sold before in another store?
3. How can monthly sales of new products be predicted based on sales data from other products?

1.2 Structure of the thesis

The thesis starts with an overview of retail business and typical retailer data. After this, previous research on sales prediction is reviewed covering univariate time series approaches, nonlinear regression, machine learning models and demand estimation models. Then, machine learning pipeline, algorithms and other relevant theoretical concepts are described. This is followed by description of experimental research setup covering the dataset, feature engineering and machine learning models. Results are described and analyzed in the next section. The thesis ends with a discussion of results and directions for future research.

2 Retail business and retailer data

This section provides a short overview of retail as a business and industry and describes some of the key data sources, data types and considerations from perspective of sales prediction.

2.1 Retail business

A retailer is a business that sells products and/or services to consumers for the personal or family use (Levy and Weitz, 2012). However, some retailers also sell directly to businesses. Retailers' role in the supply chain is to link manufacturers to consumers. In addition to selling products in physical stores, retailing encompasses selling services like lodging in a motel and selling products and services on the Internet (Levy and Weitz, 2012). In recent years, online sales have come to account for an increasing share of total retail sales (Fildes et al., 2019).

According to Levy and Weitz (2012), retailers create value to their customers in four ways. Firstly, retailers provide a variety of products in one store so that customers can satisfy most of their purchasing needs in one place. Secondly, retailers buy in large quantities at low prices from manufacturers and sell them to consumers. This enables manufactures to ship more efficiently in larger batch sizes and consumers to purchase in smaller quantities that suit their needs. Thirdly, retailers maintain an accessible inventory of goods for customers who as a result need not to store large quantities of goods themselves. Finally, retailers provide customers the services such as displaying products for customers to see and test, provide possibilities to buy on credit and offer information about the products as a part of customer service.

Retailing is a big industry with top 250 retailer sales totalling US $4.7 trillion in 2018. Biggest retailers include companies like Walmart, Costco and Amazon.com (Deloitte Touche Tohmatsu Limited, 2020). Retailers offering food and other fast-moving consumer goods (FMCG) dominate the top-250 list. Examples of retail institutions include supermarkets, convenience stores, specialist stores, drugstores, warehouse clubs and Internet stores. A retail chain is a company that operates multiple retail units under common ownership and usually has centralized decision-making processes and strategy (Levy and Weitz, 2012).

Both brick-and-mortar and online retailing are high fixed cost, low net margin businesses. Net margins are typically around 2 - 3% of sales and, in general much lower than gross margin, labor cost and the cost of real estate and inventory levels. This means that even small increases in revenue can potentially have a large impact on the bottom line. Thus, if efficient use of data and accurate predictions leads to increased sales, they can make a big difference in terms of company profits (Fisher and Raman, 2018).

2.2 Retailer data

Retailer data warehouse is usually modelled using a so-called star or snowflake schema. At the center of the data model is a fact table which refers to a set of dimension tables which provide additional information concerning the "facts". The facts in the case of retail sales, are sales transactions collected by a point of sale (POS) system which operates at each store cash register recording consumer purchases. Each consumer transaction is recorded at receipt line level. The sales transactions data typically includes information like sales quantity, product selling price, product purchase price, discount, time of transaction, product identifier, store identifier, transaction line identifier, transaction identifier and customer identifier (Kimball and Ross, 2013).

There are at least four hierarchical dimensions that are important for retail demand forecasting: product, supply chain, time and promotion (Kolassa, 2018; Fildes et al., 2019). These dimensions can be represented in multiple different granularities or levels of aggregation: e.g., time dimension (hour, day, week, month, quarter, season, year), product dimension (product, brand, category) and supply chain dimension (store, distribution center, chain) (Fildes et al., 2019). As an example, for replenishment planning at the store level, forecasts in a product (or equivalently stock keeping unit SKU) $\times$ store granularity are needed. In time dimension, forecasts are needed at a temporal aggregate level which corresponds to the delivery schedule (e.g., daily). With regard to promotion dimension, products destined for regular vs. promotion sales are not usually distinguished in replenishment planning and thus a single forecast for combined regular and promotion sales is sufficient (Kolassa, 2018).

Other use cases require forecasts in different aggregation levels. For instance, workforce planning at the store level requires total sales forecasts – i.e., forecast summed

over all products - per location as it is not relevant which particular product a store employee is handling (Kolassa, 2018). Time granularity is usually on an hourly level because worker shifts are planned by the hour. For promotion planning, the object of interest is usually forecasts of the total promotion demand per product but across locations and over the entire promotional period which may span from few days to weeks (Kolassa, 2018). On the other hand, online fashion sales may rely on initial estimate of total seasonal sales that is updated once during a season (Fildes et al., 2019).

Typical dimension tables include store, time, product, customer and promotion. These dimensions can be further aggregated to different granularities. For example, product dimension can be aggregated to brands level or different levels in the category hierarchy and store dimension can be aggregated to chain level. Figure 2.1 illustrates a typical retail sales data model.

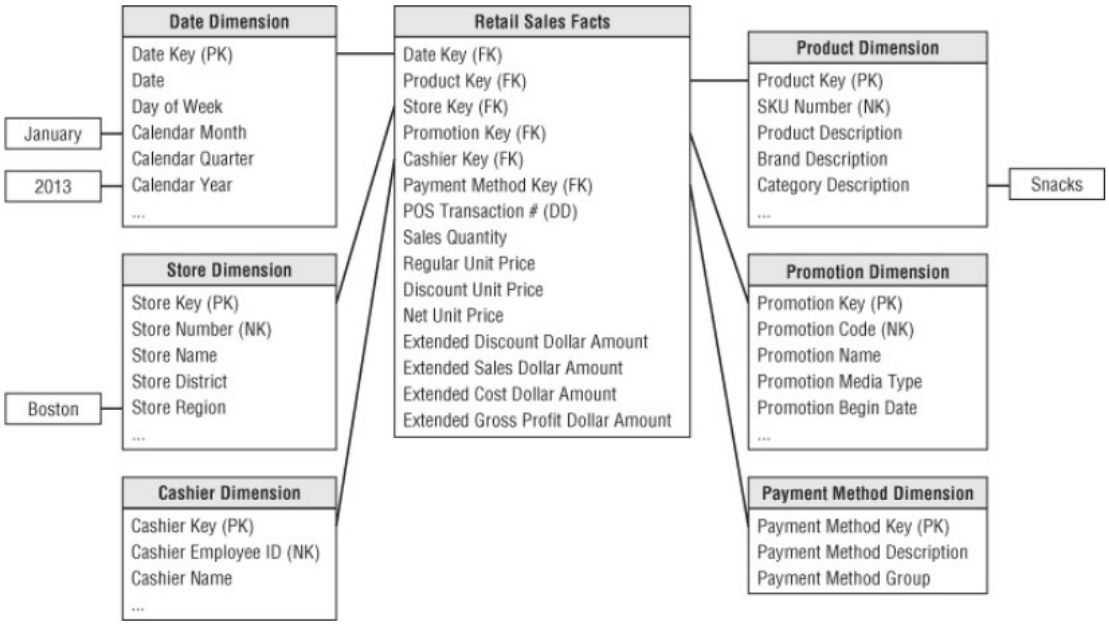

Figure 2.1: Example of retail sales data model. Adapted from Kimball and Ross (2013).

In addition to the data mentioned above, there are several information sources that might be relevant for sales prediction, which are sometimes available. These include pricing information, loyalty card information, marketing data, online product review data, social media data, seasons, special events, holidays, interaction and

cannibalization effects between products in assortments, weather and economic conditions. The set of factors that can affect demand is diverse and thus product-level sales data are typically characterized by high volatility and skewness, multiple seasonal cycles especially when combined with special days (e.g., bank holidays); often large volumes with intermittence, with zero sales being observed frequently at the store level; and high dimensionality in explanatory variable space (Fildes et al., 2019).

Sales forecasts in retail are often based on POS transaction data. However, POS sales are an imperfect observation of the true demand due to demand censoring effect where the actual demand exceeds the available inventory (Fildes et al., 2019). Seaman (2018) makes a distinction between sales and demand forecasts: a sales forecast is a forecast of the actual future sales of an item while demand forecast is a forecast of the future sales of an item if it were to be available for sale. In this terminology, sales forecast accounts for things like sales being less than otherwise possible due to product being out of stock whereas a demand forecast tries to model the underlying potential customer demand. Using this definition, forecasts prepared in this thesis are sales forecasts because data used to train models is based on (possibly censored) POS transaction data.

The POS transactions data is typically aggregated to a certain time resolution for sales prediction: hourly, daily, weekly, monthly or yearly. The time resolution depends on the purpose of the sales prediction. Usually, one uses same aggregation level in both training data and the forecasts. For example, in case one needs to produce demand forecasts at SKU $\times$ weekly $\times$ distribution center level, data is aggregated to the SKU $\times$ weekly $\times$ distribution center level and also forecasted at this level (Fildes et al., 2019). Aggregation is done by dimensions so that the aggregated sales table remains at the center of the snowflake schema and one can fetch additional information from the other dimension tables if needed.

When fitting sales prediction models the appropriate extent to which pool data across stores and products needs to be chosen. For example, in order to prepare store $\times$ product level forecasts it is possible to make separate prediction models for each product (separate data pool for each product), each store (separate data pool for each store) or each store $\times$ product combination (separate pool for each store product $\times$ combination), or to make one model for all stores and products (single data pool).

All these data pooling approaches can yield predictions at store × product level. Pooling sales data from many stores and products together increases the amount of data available for training the model, thus enabling the model to capture common patterns across different time series. Another benefit of pooling data (i.e., a making single forecast model that accounts for multiple parts of data) is that the model can be used to forecast the demand for new store × product combinations that are not present in the training data. On the other hand, pooling data and models may result in bias if same model is used for heterogenous parts of data without properly accounting for these parts by using indicator variables for example. According to Fildes et al. (2019), in research literature, there are both, examples of cases where increased pooling results in higher forecast accuracy, and vice versa.

3 Previous research on sales prediction

As outlined in previous section, sales observations are produced by interaction of multiple things. The store chooses prices, product assortment and promotions. In reaction to this, customers make purchasing decisions. Situational factors like season, special events and weather affect customer behaviour as well. In this section, three different categories of sales prediction models are explored: time series models, nonlinear regression and machine learning models, and demand estimation models. Time series, nonlinear regression and machine learning approaches assume that the demand for any product is independent of the availability of other products, i.e., it is assumed that customers would never substitute one product for another. This has been the prevalent approach in the literature at least until 2004 (Strauss et al., 2018). Demand estimation approaches try to model the demand without the so-called independent demand assumption (Strauss et al., 2018) and thus account for the substitution effects. For some retail analytics applications the independent demand assumption is reasonable and for others it is not.

3.1 Time series models

Many traditional time series models have been used for sales prediction since in most cases sales prediction is a time series extrapolation task. Most of these models are univariate, i.e., they use only past sales history as training data. A naive method to make a forecast for sales time series is to assume that the next value of the time series F_{t+1} is same as the last one available y_t (Christou, 2012):

$$F_{t+1} = y_t. \tag{3.1}$$

This forecast would be optimal in terms of accuracy if the associated difference series is a random walk, i.e., an independent and identically distributed random variable with zero mean and constant variance. According to Christou (2012) the last value prediction is rarely useful in practical situations. In this thesis it is used as a benchmark for more complicated sales prediction models as suggested in Seaman (2018).

According to Christou (2012), moving average method is one of the most well-known

and established methods for demand forecasting among supply chain managers. Moving average method predicts the next value of time series as the average of last m observed values:

$$F_{t+1} = \frac{\sum_{i=1}^{m} y_{t-i+1}}{m}. \tag{3.2}$$

The method is attractive because it is very easy to understand and implement. Moving average method can better account for mean value drifts and other stochastic processes governing time series evolution, but it is still simplistic in many ways.

Exponential smoothing is a well-established method in sales forecasting practice (Fildes et al., 2019). Single or simple exponential smoothing incorporates a feedback loop from previous error e_t to the current forecast:

$$F_{t+1} = F_t + ae_t, \tag{3.3}$$

where $a \in [0, 1]$. This can be written in expanded form as

$$F_{t+1} = a\sum_{i=0}^{t-1}(1-a)^i y_{t-i} + (1-a)^t F_1, \tag{3.4}$$

where y_t is the observed demand at time t.

Multiple exponential smoothing methods apply exponential smoothing many times on the initial time series with the idea that multiple smoothing rounds may provide better results than a single application of the method. Exponential smoothing may lag behind the original time series when there are trends in the data, especially for small values of the parameter a (Christou, 2012).

Holt-Winters method is a more complex time series model that can account for trend and seasonal variation. Seasonal component can be modelled as a multiplicative or additive component. The length of the seasons s is assumed to be known. The following system of equations describes the Holt-Winters method with multiplicative

seasonality (Christou, 2012):

$$L_t = a\frac{y_t}{S_{t-s}} + (1-a)(L_{t-1} + b_{t-1}), \tag{3.5}$$

$$b_t = \beta(L_t - L_{t-1}) + (1-\beta)b_{t-1}, \tag{3.6}$$

$$S_t = \gamma\frac{y_t}{L_t} + (1-\gamma)S_{t-s}, \tag{3.7}$$

$$F_{t+h} = (L_t + b_t h)S_{t-s+h}, \tag{3.8}$$

$$t \geq s, h \geq 1, \tag{3.9}$$

$$a, \beta, \gamma \in (0,1), \tag{3.10}$$

where y_t is the observed demand, L_t is seasonally adjusted estimate of the level of demand, S_t is multiplicative seasonality index, b_t is the estimate of a trend in data and F_{t+h} is the forecast for h periods forward. The method is initialized by using $b_{1\leq t\leq s} = y_{t+1} - y_t$, $L_{1\leq t\leq s} = y_t$ and $S_{1\leq t\leq s} = \frac{y_t s}{\sum_{j=1}^{s} y_j}$ for the first s time periods.

In addition to smoothing methods like exponential smoothing and Holt-Winters method, widely used methodology in time series analysis is a decomposition of the time series into its constituent parts: trend, seasonality, cycles and random fluctuations. Decomposition can be done either additively or multiplicatively. Forecasts for the decomposed time series are prepared separately and a final forecast is obtained by combining the forecasts (Christou, 2012).

Autoregressive moving average models (ARMA) are a common class of time series models that are often used for sales prediction. They can be used to model time series that are governed by stochastic processes that are stationary in the weak sense. Weak sense stationary means that the mean and the autocovariance of a stochastic process do not vary in time and autocovariance is finite. ARMA(p,q) -models depict the evolution of time series as function of p past values y_{t-i}, and q past realizations of white noise error terms ϵ_{t-i} (Box et al., 2015):

$$y_t - \phi_1 y_{t-1} - ... - \phi_p y_{t-p} = \epsilon_t - \theta_1 \epsilon_{t-1} - ... - \theta_q \epsilon_{t-q}. \tag{3.11}$$

Sometimes when a stochastic process is not stationary it can be transformed to a stationary series by differencing d times, i.e., by applying $y'_t = y_t - y_{t-1}$, which then yields a class of ARIMA(p,d,q) -models.

In practice, many retailers combine univariate time series prediction with some kind of judgemental adjustment. This is also called base-times-lift approach (Fildes et al., 2019). This usually means a two-step method which begins by generating a baseline forecast from a simple time series model based on training data from which the effects of demand drivers such as past promotions and the weather have been cleansed. This baseline forecast is then adjusted for any upcoming events, which most often means promotions. The adjustments for events are estimated based on the lift effect of the most recent similar event and judgements made by managers (Fildes et al., 2019).

3.2 Nonlinear regression and machine learning models

Sales prediction can be also be defined as a regression or supervised machine learning problem. This category of models includes traditional nonlinear regressions, fuzzy algorithms and machine learning models. According to Pavlyshenko (2019), practice shows that regression approaches can often give better results compared to time series models. However, according to Fildes et al. (2019) evidence for nonlinear and machine learning models generally leading to better forecasting accuracy compared to traditional methods is weak at this point. Pavlyshenko (2019) lists the following limitations of time series approaches for sales forecasting:

- Need to have long historical time series data to capture seasonality. It is possible that there is not much or any historical data for the target variable. This is for example the case when a new product is launched. At the same time, there might be data for a similar product from which inferences about the sales pattern can be made;
- Sales data can have a lot of outliers and missing data. In the case of time series models, outliers must be cleaned, and data interpolated before fitting the model;
- There is a need to take into account a lot of exogenous factors which have impact on sales.

Pavlyshenko (2019) study Random forest models and stacking approaches to forecast sales time series data in the case of two sales datasets. They are able to make good predictions of daily store sales based on small amount of historical time series data

using additional features about the store. Pavlyshenko (2018, 2019) study stacking of multiple machine learning and time series models including ExtraTrees, ARIMA, Random forest, Lasso regression and neural network in order to predict the weekly demand of products in different stores. They find out that using stacking approach improve prediction accuracy compared to single models.

Ali-Vehmas (2016) incorporates weather data to retail sales forecasting. They predict daily sales of products based on time series forecast and a regression model accounting for the weather effects. They find that forecasting accuracy for weather sensitive product categories can be improved by using this method.

In their review, Fildes et al. (2019) list several machine learning and nonlinear regression models used for sales prediction. They note that most published research has found improvements from using nonlinear methods rather than linear regressions but remark that this may well arise from publication bias. The models used for product demand prediction include back-propagation neural networks (Aburto and Weber, 2007; Ainscough and Aronson, 1999), fuzzy neural networks (Kuo, 2001), regression trees (Ali et al., 2009), gray relation analysis and multilayer functional link networks (Chen and Ou, 2011), two-level switching models that select between a simple moving average and k-nearest neighbor or decision trees (Žliobaite et al., 2012), support vector machines (Di Pillo et al., 2016), wavelets (Michis, 2015), fuzzy set approach (da Veiga et al., 2016) and Bayesian P-splines (Lang et al., 2015). Also, extreme learning machines are used in several studies, for example in Wong and Guo (2010). Due to computational limitations, studies with nonlinear regression models often use only small datasets (i.e., datasets with tens of products). Exception to this is Mukherjee et al. (2018), where authors develop a demand density forecasting model with a multi-layered hybrid deep neural network for inventory management in Flipkart.com. They model demand density using mixture of Gaussian distributions and report significant improvement in prediction accuracy.

Giering (2008) models store $\times$ item sales using reduced rank Singular Value Decomposition (SVD). The data they use include daily store level sales broken out by item and retailer-defined customer types, store level demographic data, regional demographics and firmographics. They split the data by customer type, cluster the stores based on a given demographic descriptor and build a SVD model for each cluster separately by minimizing mean absolute error in cross validation. They combine the submodels

with linear regression over demographic types and sum over customer types. They implement a product recommender which recommends new products likely sell well in a store, and an outlier analysis tool which identifies products that sell less in some stores than is to be expected by looking at the overall sales across stores. The model by Giering (2008) is useful for predicting the sales of products that haven't been sold before in a store but have been sold in some other stores. However, it does not account for the time series aspects of sales prediction like seasonality which become important when making predictions many months into the future. Also, a rich dataset about customer types and the distributions of demographic factors for each customer type for each store is required for model to work, which is not usually available.

3.3 Demand estimation methods for sales prediction

Demand estimation models are a class of models that try to account for the substitution between different products, i.e., the effect of store assortment, to realized sales. These models postulate a choice model which describes customer purchase choice given a product assortment. These models are related to revenue management originated in the airline industry in 1970s, which constitutes methods and a theory of management for prices and product availability, in order to maximize profits or revenue (Strauss et al., 2018). The models in this class can be divided into two categories: parametric and non-parametric choice models.

Parametric choice models describe customer choice based on some sequence of parameters $\theta_1, \theta_2, ..., \theta_n$. These models have their roots in random utility theory where it is assumed that consumers associate a certain utility with every choice alternative (i.e., product choice), and decide on the alternative which maximizes his or her utility. Customers associate a certain utility with each product in the set $P \cup \{0\}$ where $\{0\}$ denotes a no-purchase option. The multinominal logit (MNL) model is a common model used in this category. The MNL model assumes that alternative j of a customer has a deterministic component u_j and a random component. The random component is assumed to follow a Gumbel (extreme value of Type I) distribution with mean zero and variance $\frac{\pi^2 \mu}{6}$. Given an assortment of products in the set P, the

probability p_j, with which product j is chosen, is given by:

$$p_j = \frac{e^{\frac{\mu_j}{\mu}}}{\sum_{k \in P} e^{\frac{\mu_j}{\mu}}}, \tag{3.12}$$

where $j \in P \cup \{0\}$. (Kök and Fisher, 2007)

More sophisticated parametric choice models take more complex behaviours into account. The multiclass multinominal logit model extends MNL by allowing for several types of coexisting customer behaviours (Palmer, 2016). Nested logit model gives probability for outcome j as a product of two logit models: the probability of being in the nest, and of being chosen, given that outcome j is in the nest (Palmer, 2016).

Locational choice model assumes that products in a product category differ by a single characteristic that does not affect quality or price. An example of such a characteristic could be product fat-content. Assortment carried by the retailer is represented by a vector of product specifications $(b_1, ..., b_N)$ where N is assortment size and $b_j \in [0, 1]$ denotes the location of product j. Each consumer has an ideal point in $[0, 1]$ and chooses a product that is closest to it (Agrawal and Smith, 2015). Locational choice model was originally developed by Hotelling (1929). Newer papers in this category include Gaur and Honhon (2006).

Exogenous demand models directly specify the demand and substitution behaviour in case the product is not available. There is no underlying consumer utility model that generates the demand behaviour. Every customer is assumed to choose his or her favourite product from the assortment. The varieties of customer behaviour are modelled by a distribution over the products: customer chooses product j with probability p_j, and $\sum_{j \in P \cup \{0\}} p_j = 1$. If customer's favourite product is not available for some reason, she chooses her second favourite with probability δ, and with probability $1 - \delta$ she chooses not to purchase at all. The probability of substituting product j for k is α_{jk}. When a substitute product is not available, consumers repeat the same procedure for the third favourite and so on. (Agrawal and Smith, 2015)

In contrast to parametric choice models, non-parametric choice models make fewer assumptions on customer behaviour and are in this sense more general. Most popular

class of models in this category are generic choice models based on a probability distribution over the set of all possible rankings of products proposed by Farias et al. (2013). Van Ryzin and Vulcano (2015); Palmer (2016); Bertsimas and Mišić (2019) and others build on the model introduced by Farias et al. (2013). Original model by Farias et al. (2013) has factorial number of rankings to be estimated and is intractable for practical cases. The model can consider a subset of possible rankings as done in Bertsimas and Mišić (2019). The model description here follows Bertsimas and Mišić (2019):

There are $\{0, 1, 2, ..., N\}$ products to choose from including no-purchase alternative. Customer always chooses exactly one of the alternatives. Preferences are modelled as K rankings $\sigma_1, ..., \sigma_K$ over the options $\{0, 1, 2, ..., N\}$. Each ranking $\sigma_k : \{0, 1, 2, ..., N\} \to \{0, 1, 2, ..., N\}$ is a bijection that assigns each option rank in $\{0, 1, 2, ..., N\}$. The value $\sigma^k(i)$ indicates the rank of option i such that if $\sigma^k(i) < \sigma^k(j)$ then product i is preferred to j under ranking σ^k. Assumption is that a customer that follows the ranking σ^k, given a set of products $P = \{1, 2, ..., N\}$, chooses option i^* from the set $P \cup \{0\}$ with the lowest rank: $i^* = \arg\min_{i \in P \cup \{0\}} \sigma^k(i)$. Each k can be thought as a customer type or segment and σ^k as the choice behaviour that customers of that type follow. There is a probability distribution $\boldsymbol{\lambda} = (\lambda^1, ..., \lambda^K)$ over the customer types such that λ^k determines the probability that a random customer is of type k, and $\sum_{k=1}^{K} \lambda^k = 1$. Thus, the probability that a random customer chooses product i, given available set of products P is:

$$\Pr(i \mid P) = \sum_{k=1}^{K} \lambda^k I\{i = \underset{i' \in P \cup \{0\}}{\arg\min}\, \sigma^k(i')\}, \tag{3.13}$$

where $I\{.\}$ is an indicator function such that $I\{A\}$ is 1, if A is true, and 0 otherwise.

This ranking-based choice model is general and many other customer behaviour models such as utility-based models can be derived as a special case of it (Bertsimas and Mišić, 2019). The challenge with the ranking-based models is to estimate them based on data with a large number of products. Approaches to overcome this so that the model can be estimated in the case of larger datasets and used for practical applications are proposed in Palmer (2016) and Bertsimas and Mišić (2019). This seems to be the most promising model in the class of demand estimation models that attempt rigorously to account for the substitution effects between products.

4 Machine learning framework for sales prediction

This section frames sales prediction as machine learning problem and introduces concepts that are relevant for later data analysis. First, the whole machine learning pipeline is outlined, and most relevant steps are described. Secondly, the machine learning algorithms used in this thesis are described. Lastly, few relevant data analysis techniques that are used in the analysis are reviewed including latent semantic analysis, Shapley additive explanations and one-hot encoding.

4.1 Machine learning pipeline

Machine learning pipeline as a concept encompasses the whole process leading from raw data to value adding information and insights. Crucial parts of the machine learning pipeline are raw data, features and modelling. Raw data are observations of some real-world phenomena that usually come with some noise. Mathematical modelling is a way of deriving useful insights from data. A feature is a numeric representation of the raw data collected. There are many ways to transform the raw data into a set of numeric features. Best set of features depend on the available data, modelling task and the class of mathematical models used (Zheng and Casari, 2018). Figure 4.1 outlines the parts of the machine learning pipeline.

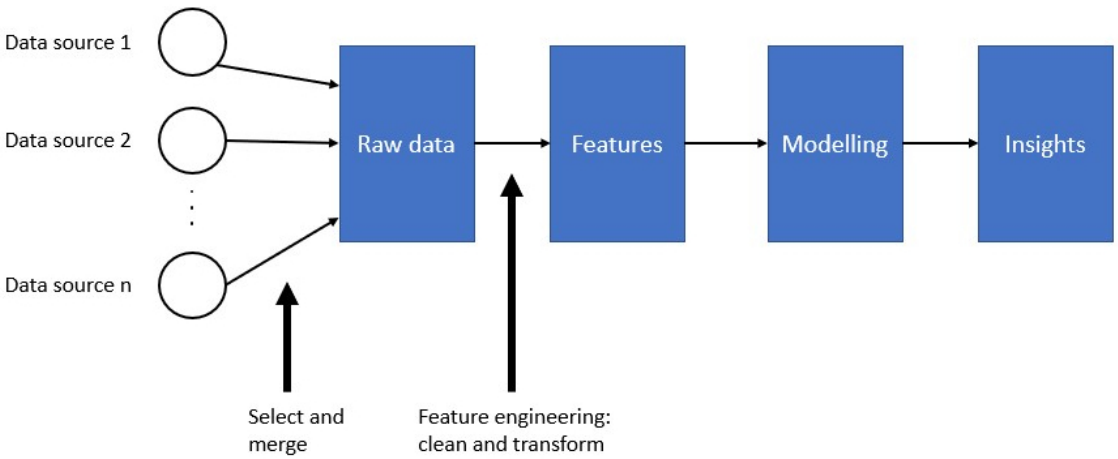

Figure 4.1: Machine learning pipeline. Adapted from Zheng and Casari (2018).

Features and models are between raw data and the desired insights from the modelling task. When constructing a machine learning workflow, one chooses not only the model but also the features. The effect is bidirectional: the choice of one affects the other. Good features make the subsequent modelling step easier while bad features might make it impossible to get good results. Conversely, a feature representation of data that works for some models might not work for other models. (Zheng and Casari, 2018)

4.1.1 Feature engineering

Feature engineering is the act of extracting features from raw data and transforming them into formats suitable for the use of the actual machine learning or statistical algorithm (Zheng and Casari, 2018). Feature engineering is an important step in the machine learning pipeline because well defined features can ease the difficulty of modelling and therefore enable the overall pipeline to output better results. Machine learning practitioners agree that the vast majority of time in building a machine learning pipeline is spent on feature engineering and data cleaning work (Zheng and Casari, 2018).

4.1.2 Machine learning model

Machine learning can be divided into supervised and unsupervised learning. In supervised learning, the focus is on making accurate predictions whereas in unsupervised learning the aim is to find compact descriptions of data (Barber, 2012). In both supervised and unsupervised learning, one is looking to find methods that generalise well to previously unseen data. Supervised learning is further divided in two categories based on the nature of the output: if the output is continuous, the problem is called regression problem; if output consists of a discrete number of classes, the problem is called classification problem (Barber, 2012).

Sales prediction falls under the class of regression problems in supervised learning. We want to learn the relationship between a set of inputs $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ...\boldsymbol{x}_n\}$ and outputs $\{y_1, y_2, ..., y_n\}$. Inputs can also be called predictors, independent variables, explanatory variables or features. Outputs are also called responses or dependent variables (Friedman et al., 2009). The supervised learning task can be understood as a problem of function approximation (Friedman, 2001; Friedman et al., 2009). The

goal is to obtain an estimate or an approximation of the function $F(\boldsymbol{x})$ mapping $\boldsymbol{x}$ into y, $y = F(\boldsymbol{x})$, that minimises the expected value of some specified loss function $L(F(\boldsymbol{x}), y)$ over the joint distiribution of all $(\boldsymbol{x}, y)$ -values:

$$F^* = \arg\min_F \mathop{\mathbb{E}}_{\boldsymbol{x},y} L(F(\boldsymbol{x}), y) = \arg\min_F \mathop{\mathbb{E}}_{\boldsymbol{x}} \Big[\mathop{\mathbb{E}}_{y} L(F(\boldsymbol{x}), y) \;\Big|\; \boldsymbol{x} \Big]. \tag{4.1}$$

The expected value of loss function $R(F) = \mathbb{E}_{\boldsymbol{x},y} L(F(\boldsymbol{x}), y)$ is called the risk of model F. The loss function $L(F(\boldsymbol{x}), y)$ is a measure of error between prediction $F(\boldsymbol{x})$ and actual value y.

The real joint distribution of $(\boldsymbol{x}, y)$ -values is usually not known. In practice, the function is estimated based on data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$ $(|\mathcal{D}| = n, \boldsymbol{x} \in \mathbb{R}^m, y_i \in \mathbb{R})$ which is a sample from joint distribution $(\boldsymbol{x}, y)$. This is called empirical risk $\hat{R}(f)$ minimization (Barber, 2012):

$$\hat{F}^* = \arg\min_F \hat{R}(f) = \arg\min_F \frac{1}{n} \sum_{i=1}^{n} L(F(\boldsymbol{x_i}), y_i). \tag{4.2}$$

In sales prediction, output y is sales quantity or sales in a currency in a given time interval. Depending on the business problem and the available data, time interval can be hours, days, weeks, months or years. The predictor variables $\boldsymbol{x}$ can be, for example, past realizations of the output, product or store characteristics, marketing data, weather data or any data that might have predictive power with respect to future sales.

There are several ways to quantify the error, i.e., choose the evaluation metric in sales prediction. This corresponds to choosing a measure for empirical risk and a loss function. Tsoumakas (2019) gives the following list of evaluation metrics assuming $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$ denotes mean of the output variable:

- Mean squared error (MSE): $\frac{1}{n} \sum_{i=1}^{n} (y_i - F(\boldsymbol{x}_i))^2$
- Root mean squared error (RMSE): $\sqrt{MSE}$
- Relative root mean squared error (RRMSE): $\sqrt{\frac{\sum_{i=1}^{n} (y_i - F(\boldsymbol{x}_i))^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}}$
- Mean absolute error (MAE): $\frac{1}{n} \sum_{i=1}^{n} |y_i - F(\boldsymbol{x}_i))|$

– Mean absolute percentage error (MAPE): $\frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - F(\boldsymbol{x}_i)}{y_i}\right|$

– Mean absolute scaled error (MASE): $\frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - F(\boldsymbol{x}_i)}{MAE(baseline)}\right|$

Seaman (2018) mentions weighted average of the product-wise mean absolute percentage errors (WMAPE) as a useful metric as this allows for clear prioritization of items. For example, a weighing based on dollar or unit sales will put more emphasis on the forecast accuracy for higher selling items than on the lower selling items. Hyndman and Koehler (2006) recommend mean scaled error (MASE) as a measure for comparing forecast accuracy across multiple different time series.

4.1.3 Model selection and assessment

The overall goal of modelling step is to assess the performance of different models in order to choose the best one. Best model is the one that has the best generalisation performance, i.e., the ability to make accurate predictions on previously unseen data (Friedman et al., 2009). A model usually has tuning parameters or hyperparameters α that vary the complexity and characteristics of the model so that the fitted model $\hat{F}_\alpha(\boldsymbol{x})$ depends both on the selection of the hyperparameters α and the dataset used to fit the model (Friedman et al., 2009).

In a data-rich situation, best approach is to divide the full dataset $\mathcal{D}$, into training $\mathcal{D}_{train}$, validation $\mathcal{D}_{validate}$, and testing $\mathcal{D}_{test}$ datasets for model training and selection. Training set is used to fit the models; the validation set is used to estimate prediction error for model selection and hyperparameter tuning; the testing set is used for assessment of the generalization error of the final chosen model (Friedman et al., 2009). Usually, training set comprises most of the observations available.

The model F is fitted to the training data $\mathcal{D}_{train}$ for several values of the hyperparameters α, and for each α one finds an estimate of the expected value (or some other desired characterisation of the distribution) of error, or loss $L(\hat{F}_\alpha(\boldsymbol{x}), y)$ in the validation set:

$$Err_{validate} = \mathbb{E}\, L(\hat{F}_\alpha(\boldsymbol{x}), y) = \frac{1}{n_{validate}} \sum_{i \in \mathcal{D}_{validate}} L(\hat{F}_\alpha(\boldsymbol{x}_i), y_i), \tag{4.3}$$

where $Err_{validate}$ is called validation error or estimated extra-sample error (Friedman

et al., 2009). The values of hyperparameters α are chosen so that the validation error is minimized. This process is also called hyperparameter tuning or optimization. There are several ways to try to achieve this, e.g., random search, grid search, or Bayesian optimization (Balandat et al., 2019) in hyperparameter space. After this, using the found hyperparameters α, the model is fitted using both the training and validation data $\mathcal{D}_{train} \cup \mathcal{D}_{validate}$. The generalisation ability of the model is assessed by making predictions for the previously unseen data in the testing set $\mathcal{D}_{test}$ and calculating the error (Barber, 2012).

In cross-validation, the dataset excluding testing data $\mathcal{D} \setminus \mathcal{D}_{test}$ is partitioned multiple times. Each partition produces a different training set $\mathcal{D}^i_{train}$, validation set $\mathcal{D}^i_{validate}$, fitted model and associated validation error $Err^i_{validate}$. Error of hyperparameters α is the average of the validation errors over all partitions i. The best overall hyperparameters α are the ones with the minimal average validation error. After cross-validation has been performed for the desired set of hyperparameters, the model can be retrained using the whole data $\mathcal{D} \setminus \mathcal{D}_{test}$ and best hyperparameters α, and the generalisation ability assessed by calculating the error in the testing set.

In sales prediction task, the data is based on time series. This means that the cross-validation partitions need to take into account the time at different data points. At time t, only data generated up to that point is available. Therefore, unlike in standard k-fold cross-validation where observations are assigned to folds randomly, in time series cross-validation (Hyndman and Athanasopoulos, 2018) data is split so that in each split training set consists of observations $\{(\boldsymbol{x}_i, y_i) \mid t_i < t_s\}$ up to time t_s, and testing set consits of observations $\{(\boldsymbol{x}_i, y_i) \mid t_i = t_s\}$ at t_s. Hence, no future observations are used in constructing the forecast. Different values of t_s yield different splits that can be used in cross-validation. Figure 4.2 illustrates the principle.

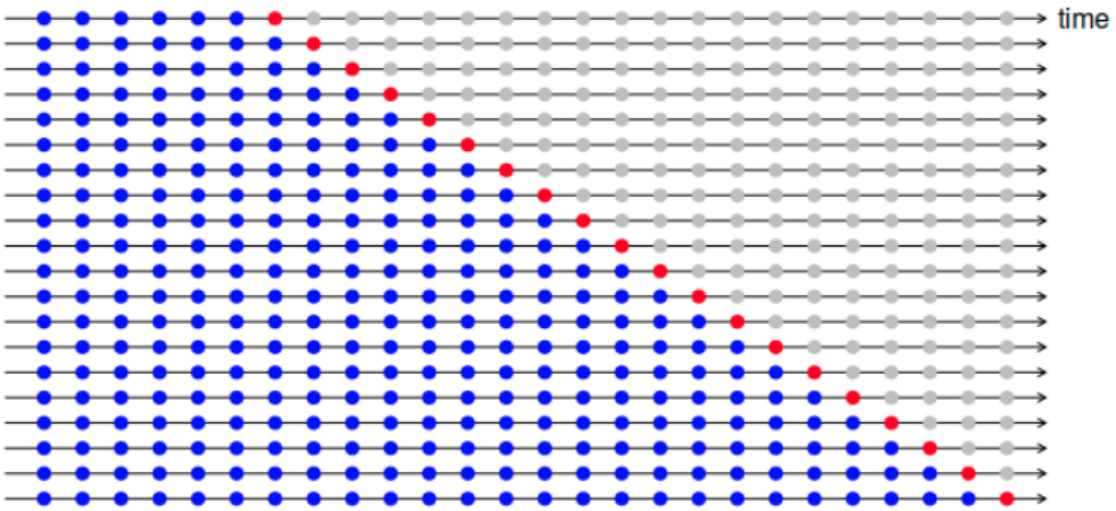

Figure 4.2: Splits in time series cross-validation. Blue and red dots indicate times included in training set and testing set in different splits (Hyndman and Athanasopoulos, 2018).

4.2 Machine learning algorithms

In this thesis, Lasso regression, extreme gradient boosting (XGBoost) and rule-based benchmark models are used for sales prediction. This section expounds the theory behind Lasso regression and XGBoost models. Regression trees are also described because the XGBoost algorithm uses them as base learners.

4.2.1 Lasso regression

Lasso regression is a linear regression method where regression coefficients are shrunk by imposing a penalty on their size (Friedman et al., 2009). Lasso estimate is defined by:

$$\begin{aligned} \hat{\beta}^{lasso} &= \arg\min_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{m} x_{ij}\beta_j \right)^2 \\ &\text{subject to} \sum_{j=1}^{m} |\beta_j| \leq t. \end{aligned} \tag{4.4}$$

The equivalent Lagrangian form is (Friedman et al., 2009):

$$\hat{\beta}^{lasso} = \arg\min_{\beta} \left(\frac{1}{2} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{m} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{m} |\beta_j| \right). \tag{4.5}$$

Thus, Lasso regression is a linear model with an added regularization term. Lasso estimate solves the minimization of least-squares with penalty proportional to the L1-norm of the coefficient vector. The nature of the optimization constraint, depending on the degree of regularisation chosen by setting λ, can cause some of the coefficients to be exactly zero. This can be viewed as automatic feature selection based on regularization, so that the coefficients for variables which are less important predictors are set to zero, and they are not present in the model. Also, this means Lasso regression is easily interpretable: significant predictors can be easily separated from non-significant predictors because the latter have zero coefficients in the estimated model.

4.2.2 Regression trees

Tree-based methods partition the feature space into a set of rectangles, and then fit a simple model, for example a constant, in each one. They are conceptually simple but can be powerful. Treatment here follows Friedman et al. (2009). Data consists of n observations of m inputs $\boldsymbol{x}_i = (x_{i1}, x_{i2}, ..., x_{im})$ and responses y_i. Given a partition of the input space into T regions $R_1, R_2, ..., R_T$, where the response is modelled as a constant w_l in each region, then:

$$F(x) = \sum_{l=1}^{T} w_l I(x \in R_l), \tag{4.6}$$

where I is the indicator function. It can be shown that the best fit $\hat{w}_l$ is the average of responses in a region: $\hat{w}_l = \text{mean}(\{y_i \mid \boldsymbol{x}_i \in R_l\})$.

A greedy algorithm proceeds with all the data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ splits along each dimension $j \in \{1, .., m\}$ of the inputs $\boldsymbol{x}_i$, and finds the best splitting variable x_j and the best split. A split with split point s for variable j, is defined by the half-planes: $R_1(j, s) = \{\boldsymbol{x} \in \mathbb{R}^m \mid x_j \leq s\}$ and $R_2(j, s) = \{\boldsymbol{x} \in \mathbb{R}^m \mid x_j > s\}$. Best splitting

variable and split point is solved from the minimization problem:

$$\min_{j,s}\left(\min_{w_1}\sum_{i|\boldsymbol{x}_i\in R_1(j,s)}(y_i-w_1)^2+\min_{w_2}\sum_{i|\boldsymbol{x}_i\in R_2(j,s)}(y_i-w_2)^2\right). \tag{4.7}$$

For any (j,s) the inner minimization is solved by taking averages: $\hat{w}_1=\text{mean}(\{y_i \mid \boldsymbol{x}_i\in R_1(j,s)\})$ and $\hat{w}_2=\text{mean}(\{y_i \mid \boldsymbol{x}_i\in R_2(j,s)\})$. Determination of the split point can be done very quickly for each splitting variable j. Thus, by scanning through all input dimensions, the determination of best pair (j,s) is feasible.

After finding the best split, data is partitioned into two regions and the process is repeated for each region until tree is grown large enough. Tree size is a tuning parameter governing the model's complexity. A common approach is to initially grow a large tree such that some nodes contain only a small number of observations and then use pruning, for example cost-complexity pruning see, e.g., Breiman et al. (1984), to simplify the tree. Pruning means successively collapsing nodes, i.e., assigning a common average for all observations in the subtree starting from a node, in a fashion that reduces tree complexity but doesn't increase the sum-of-squares or some other error measure too much.

4.2.3 Extreme gradient boosting (XGBoost)

Boosting is a way to combine the outputs of many "weak" learners such as trees to produce a powerful ensemble or a committee of models. Treatment here follows mostly Chen and Guestrin (2016), a paper where extreme gradient boosting (XGBoost) algorithm was originally proposed. Given a dataset with n observations and m features $\mathcal{D}=\{(\boldsymbol{x}_i,y_i)\}$ $(|\mathcal{D}|=n,\boldsymbol{x}\in\mathbb{R}^m,y_i\in\mathbb{R})$, a tree ensemble model uses K additive trees to predict the output,

$$\hat{y}_i=F(\boldsymbol{x}_i)=\sum_{k=1}^{K}f_k(\boldsymbol{x}_i),\quad f_k\in\mathcal{F}, \tag{4.8}$$

where $\mathcal{F}=\{f(\boldsymbol{x})=w_{q(\boldsymbol{x})}\}(q:\mathbb{R}^m\to T,\boldsymbol{w}\in\mathbb{R}^T)$ is the space of regression trees. Here q represents the structure of each tree that maps an observation to corresponding leaf index, $\boldsymbol{w}$ denotes the leaf weights and T is the number of leaves in the tree. Tree

$f(\boldsymbol{x})$ can be expressed using the same notation as in equation (4.6):

$$f(\boldsymbol{x}) = w_{q(\boldsymbol{x})} = \sum_{l=1}^{T} w_l I(q(\boldsymbol{x}) = l) = \sum_{l=1}^{T} w_l I(x \in R_l). \tag{4.9}$$

For a given observation, the decision rules in the trees given by q, classify it into a leaf. The final prediction is given by summing up the predictions from different trees. Unlike in traditional gradient tree boosting, XGBoost uses a regularized objective to learn the model:

$$\mathcal{L}(F) = \sum_i L(\hat{y}_i, y_i) + \sum_k \Omega(f_k), \tag{4.10}$$

where $\Omega(f) = \gamma T + \frac{1}{2}\lambda \|\boldsymbol{w}\|^2$ is the regularization term that penalizes the complexity of the model. The regularization term helps to smooth the final learnt weights and to avoid overfitting.

The tree ensemble model in equation (4.10) includes functions as parameters and cannot be optimized using traditional optimization methods in Euclidian space. Instead, the boosting model is trained in an additive manner. Let $\hat{y}_i^{(t)}$ be the prediction of the i-th instance at the t-th iteration. At iteration t, a new tree f_t that most improves the model according to equation (4.10) is added:

$$\min_{f_t} \mathcal{L}^{(t)} = \sum_{i=1}^{n} L(\hat{y}_i^{(t-1)} + f_t(\boldsymbol{x}_i), y_i) + \Omega(f_t). \tag{4.11}$$

Chen and Guestrin (2016) use a second order Taylor series expansion to approximate the objective function. Define $I_j = \{i \mid q(\boldsymbol{x}_i) = j\}$ as the instance set of leaf j. Using the approximation, the objective function can be written as:

$$\min_{f_t} \tilde{\mathcal{L}}^{(t)} = \sum_{j=1}^{T} \Big(G_j w_j + \frac{1}{2}(H_j + \lambda) w_j^2 \Big) + \gamma T, \tag{4.12}$$

where $G_j = \sum_{i \in I_j} \partial_{\hat{y}^{(t-1)}} L(\hat{y}_i^{(t-1)}, y_i)$ and $H_j = \sum_{i \in I_j} \partial^2_{\hat{y}^{(t-1)}} L(\hat{y}_i^{(t-1)}, y_i)$ are the sums of first and second order gradient statistics for instance set at leaf j.

Given a fixed leaf structure $q(\boldsymbol{x})$, optimal weight w_j^* for leaf j is:

$$w_j^* = -\frac{G_j}{H_j + \lambda}. \tag{4.13}$$

Optimal value of objective function is

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda} + \gamma T. \tag{4.14}$$

Normally it is not possible to enumerate all possible tree structures q. A greedy algorithm that starts from a single leaf and iteratively adds branches is used instead. Let I_L and I_R be the instance sets of left and right nodes after the split and $I = I_L \cup I_R$. Loss reduction (gain) after the split is:

$$gain = \tilde{\mathcal{L}}_{split} = \frac{1}{2}\left(\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} + \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right) - \gamma. \tag{4.15}$$

Best split is found by enumerating over all possible splits on all the features (exact greedy algorithm), or approximately, by only considering some split points based on the distributions of features. XGBoost also accounts for sparsity caused by missing values in the data by assigning a default direction in each tree node. When a value is missing for a feature in the data the instance is classified into the default direction. Default directions are learned from the data. Algorithm 1 is the split finding algorithm.

Algorithm 1: Sparsity-aware split finding

Input: I, instance set of current leaf node

Input: $I_k = \{i \in I \mid x_{ik} \neq missing\}$

Initialize $gain \leftarrow 0, G \leftarrow \sum_{i \in I} g_i$, $H \leftarrow \sum_{i \in I} h_i$.

for $k = 1$ to m **do**

$G_L \leftarrow 0$, $H_L \leftarrow 0$

for j in sorted(I_k, ascent order by x_{jk}) **do**

$G_L \leftarrow G_L + g_j$, $H_L \leftarrow H_L + h_j$

$G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$

$gain \leftarrow max(gain, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$

end for

$G_R \leftarrow 0$, $H_R \leftarrow 0$

for j in sorted(I_k, descent order by x_{jk}) **do**

$G_R \leftarrow G_R + g_j$, $H_R \leftarrow H_R + h_j$

$G_L \leftarrow G - G_R$, $H_L \leftarrow H - H_R$

$gain \leftarrow max(gain, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$

end for

end for

Output: Split and default directions with max gain

An exact greedy algorithm for growing trees successively adds new leaves to the tree until a stopping condition is met. At each iteration, the algorithm tries to split each existing leaf at each feature at the best split point and finds the split associated with maximum gain. Stopping condition for the algorithm can be, for example, defined by maximum tree depth. To learn the full ensemble, new trees are added and gradient statistics updated iteratively until the desired number of base learners has been reached.

Define depth $d(l)$ at leaf l as the number of splits made in the tree required to reach that leaf, and I_l as the instance set at leaf l. Exact tree growing algorithm for XGBoost, and the XGBoost algorithm for learning the full ensemble adapted from Chen and Guestrin (2016) are summarised in Algorithms 2 and 3.

Algorithm 2: Tree growing algorithm for XGBoost

Input: Gradient statistics of loss function at previous iteration of the ensemble $g_i = \partial_{\hat{y}^{(t-1)}} L(\hat{y}_i^{(t-1)}, y_i)$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} L(\hat{y}_i^{(t-1)}, y_i)$

Initialize tree q with one leaf: $I_1 = \{1, ..., n\}$, $T = 1$.

while $\max_{1 \leq l \leq T} d(i) < d_{max}$ **do**

for $l = 1$ to T **do**

Find best split at leaf l using algorithm 1.

end for

Update tree structure q so that new split is made at the leaf l, feature k and split point x_{ik} with the maximum $gain$.

end while

Calculate optimal leaf weights associated for tree q: $w_l = -\frac{G_l}{H_l + \lambda}$ for $l = 1, ..., T$.

Output: Tree $f(\boldsymbol{x}) = w_{q(\boldsymbol{x})}$.

Algorithm 3: XGBoost algorithm

Input: Training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$ $(|\mathcal{D}| = n, \boldsymbol{x} \in \mathbb{R}^m, y_i \in \mathbb{R})$

Input: Learning rate η

Initialize $F_0(\boldsymbol{x}_i) = \hat{\theta}_0 = \arg\min_\theta \sum_{i=1}^N L(\theta, y_i)$.

for $t = 1$ to K **do**

Calculate $g_i = \partial_F L(F, y_i)|_{F=F_{t-1}(\boldsymbol{x}_i)}$ and $h_i = \partial^2_F L(F, y_i)|_{F=F_{t-1}(\boldsymbol{x}_i)}$.

Grow tree $f_t(\boldsymbol{x}_i)$ using algorithm 2.

Set $F_t(\boldsymbol{x}_i) = F_{t-1}(\boldsymbol{x}_i) + \eta f_t(\boldsymbol{x}_i)$.

end for

Output: Tree ensemble $F_K(\boldsymbol{x}_i)$.

In addition to the parameters referenced above, there are several other parameters available in the XGBoost implementation. For instance, to control overfitting, it has been found useful to allow a new tree to only be split by a subset of features. Parameter colsample_bytree $\in (0, 1]$ is the subsample ratio of features when constructing each tree and colsample_bylevel $\in (0, 1]$ is the subsample ratio of features for each level in the tree. In the former case, subsampling occurs once for every tree constructed, and in the latter once for every new depth level reached in a tree. Also, XGBoost allows for early stopping which is a method often used in machine learning to control overfitting. When using early stopping, new trees are added to the ensemble until the performance of the ensemble stops improving as measured by prediction accuracy

in an out-of-sample evaluation set.

In summary, the key features of XGBoost algorithm that distinguish it from traditional gradient boosting algorithms and that have enabled it to achieve exceptional performance in many problems involving structural data are: regularization to penalize complex models, use of second order gradient statistics instead of first order statistics, algorithmic handling of missing data, weighted quantile sketch (a novel and efficient approximate algorithm for split finding) and parallelized implementation combined with efficient caching of gradient statistics.

4.3 Other theoretical concepts

This section describes latent semantic analysis, Shapley additive explanations and one-hot encoding, which are data analysis techniques used as a part of the experimental setup in this study. Latent semantic analysis is used to derive features for new product sales prediction, Shapley additive explanations are used for evaluating predictor importance of different models, and one-hot encoding is applied to represent categorical variables in numerical format.

4.3.1 Latent semantic analysis for textual data

Latent semantic analysis (LSA) is an unsupervised technique for deriving implicit representation of text semantics based on observed co-occurrence of words (Aggarwal and Zhai, 2012). In this technique, textual data is first represented in a numerical format as a term-document matrix. Singular value decomposition (SVD) is then used to reduce the dimensionality of the term-document representation. It can be applied to textual data to create low dimensional numerical representation of that data.

Given text dataset comprising of m terms (vocabulary) and n documents, a value a_{ij} in term-document matrix $\boldsymbol{A}$ represents the weight of the term i in document j. A term can denote a single word (also called unigram) or sequence of n words (n-gram). There are multiple ways to quantify weight. Simplest way is to count the number of occurrences of term i in a document j: tf_{ij}. In this case, a value in term-document matrix $a_{ij} = \mathrm{tf}_{ij}$ refers to number of times term i occurs in document j.

Another common way to quantify term weight is tf-idf term weighting. Tf means term-frequency while tf-idf means term-frequency times inverse document-frequency:

$$\text{tf-idf}_{ij} = \text{tf}_{ij}\text{idf}_t, \tag{4.16}$$

where $\text{idf}_t = log(\frac{n}{1+\text{df}_t})+1$, n is the number of documents in corpus, and $\text{df}_t = \sum_{j=1}^{n} \text{tf}_{ij}$, is document frequency (Anandarajan et al., 2019).

Document frequency measures frequency of terms across all documents. Inverse document-frequency measures the rarity of terms so that rarer items are assigned a higher weight. Tf-idf is high when a term occurs many times in few documents and is low when a term occurs in most documents. Tf-idf weighting allows one to put more weight to rarer and more interesting terms and less weight to common words (e.g., "the" or "is" in English) that carry little meaningful information.

After construction of term-document matrix $\boldsymbol{A}$, singular value decomposition (SVD) is applied to the matrix to represent it as a product of three matrices. Every matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ can be represented as a product of three matrices:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\intercal, \tag{4.17}$$

where $\boldsymbol{U} \in \mathbb{R}^{m \times m}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ are orthogonal matrices and $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times n}$ is diagonal and non-negative. The diagonal entries $\sigma_i = \boldsymbol{\Sigma}_{ii}$ of $\boldsymbol{\Sigma}$ are known as singular values. The number of non-zero singular values is equal to rank r of $\boldsymbol{A}$. Diagonal entries in $\boldsymbol{\Sigma}$ are ordered in descending order so that $\sigma_1 \geq \sigma_2 ... \geq \sigma_r$. (Manning et al., 2010)

Singular value decomposition can be used to construct rank-k approximation of a matrix by replacing $r - k$ smallest singular values with zeros on the diagonal of $\boldsymbol{\Sigma}$:

$$\boldsymbol{A}_k = \boldsymbol{U}\boldsymbol{\Sigma}_k\boldsymbol{V}^\intercal = \boldsymbol{U}'_k\boldsymbol{\Sigma}'_k\boldsymbol{V}'^\intercal_k, \tag{4.18}$$

where $\boldsymbol{\Sigma}_k$ is $\boldsymbol{\Sigma}$ with $r - k$ smallest singular values replaced with zeros, $\boldsymbol{\Sigma}'_k \in \mathbb{R}^{k \times k}$ is the square submatrix of $\boldsymbol{\Sigma}_k$, $\boldsymbol{U}'_k \in \mathbb{R}^{m \times k}$, and $\boldsymbol{V}'_k \in \mathbb{R}^{n \times k}$. Because $r - k$ smallest singular values are set to zero, the $r - k$ columns in $\boldsymbol{U}$ and $\boldsymbol{V}$ do not affect the result and can be removed. Form on the right is also known as truncated SVD. It can be shown that $\boldsymbol{A}_k$ is the best rank-k approximation of $\boldsymbol{A}$ as measured by Froebenius norm (Manning et al., 2010).

In latent semantic analysis the term-document matrix $\boldsymbol{A}$ is approximated by truncated SVD: $\boldsymbol{A}_k = \boldsymbol{U}'_k \boldsymbol{\Sigma}'_k \boldsymbol{V}'^{\mathsf{T}}_k$. Hence, a document $\boldsymbol{x}$ can be represented in the k dimensional space by using the transformation (Manning et al., 2010):

$$\boldsymbol{x}_k = (\boldsymbol{\Sigma}'_k)^{-1} \boldsymbol{U}'^{\mathsf{T}}_k \boldsymbol{x}. \tag{4.19}$$

4.3.2 Shapley additive explanations

Shapley additive explanations (SHAP) is an unified approach for interpreting model predictions (Lundberg and Lee, 2017; Lundberg et al., 2020). It is based on a solution concept in cooperative game theory called Shapley value (Shapley, 1953). SHAP assigns each feature an importance value for a prediction. It works across different types of machine learning algorithms and it has been shown to be the only possible method in the class of additive feature attribution methods that simultaneously satisfies three important properties: local accuracy, consistency and missingness (Lundberg and Lee, 2017). Shapley value $\phi_i(F, \boldsymbol{x})$ gives the importance or effect of feature i for the prediction of model F given input $\boldsymbol{x}$. Shapley values are calculated by introducing each feature, one at a time, into a conditional expectation function of the model's output, and attributing the change produced at each step to the feature that was introduced. Conditional expectation of the model's output is (Lundberg et al., 2020),

$$F_{\boldsymbol{x}}(S) = \mathbb{E}(F(X) \mid \mathrm{do}(X_S = \boldsymbol{x}_s)), \tag{4.20}$$

where S is the set of features we are conditioning on, X is a random variable including all features, X_S is random variable for features in set S, $\boldsymbol{x}_s$ are the values of features in S from input $\boldsymbol{x}$ and do is Pearl's do-operator (Pearl, 2009), which in this case gives conditional expectation given an intervention $X_s = \boldsymbol{x}_s$ is made.

Shapley additive explanation (SHAP) values are defined as

$$\phi_i(F, \boldsymbol{x}) = \sum_{R \in \mathcal{R}} \frac{1}{m!} \left(F_{\boldsymbol{x}}(S_i^R \cup i) - F_{\boldsymbol{x}}(S_i^R) \right), \tag{4.21}$$

where $\mathcal{R}$ is the set of all feature orderings, S_i^R is the set of all features that come before feature i in ordering R, m is the number of input features for the model and $F_{\boldsymbol{x}}$ is a conditional expectation function of the model's output (Lundberg et al.,

2020).

4.3.3 One-hot encoding

One-hot encoding is a bit-wise representation of categorical variable. For a categorical variable C with k mutually exclusive categories $c_1, c_2, ..., c_k$ one-hot encoded representation of C is a vector $\boldsymbol{e} \in \mathbb{R}^k$ such that (Zheng and Casari, 2018):

$$e_i = \begin{cases} 1, & \text{if } C = c_i \\ 0, & \text{otherwise.} \end{cases} \tag{4.22}$$

5 Experimental setup

This section describes the experimental research setup used in the thesis. First, the dataset used is described. Then, sales prediction task with reference to the data in available is formulated as a machine learning problem. Next, feature engineering and the features extracted from the raw data in the case of each research question are described. Finally, the setup for fitting the machine learning models is specified.

5.1 Dataset

This study uses a dataset from a retail store chain operating in Finland. Dataset comprises of point of sales transactions and related dimension tables (e.g., products and stores). Since the goal in all research questions is to predict monthly sales of products in different stores, data is aggregated to monthly level. The aggregated dataset contains monthly sales quantities for products sold in 20 stores between 01/2016 and 10/2018 (34 months). There are 12458 products belonging to three-level product category hierarchy with 6, 32 and 99 categories in each level. Brand information for products is known: there are 292 product brands. The dataset has in total 588958 rows and consists of sales observations y of products p in stores s in different months t. The raw data can be represented by the set $\mathcal{D} = \{((s_i, p_i, t_i), y_i)\}_1^n$, where $n = 588958$. Each row represents a unique store, product and month - combination.

The scope of the dataset is minimal in terms of available data: it comprises only monthly sales figures of products in different stores at different points in time together with product category and brand information. For example, product price, customer and promotions data is not included. The dataset is intentionally limited to this minimal scope so that the prediction accuracy of the machine learning pipeline and models can be tested with data that is available in almost all cases of retail monthly sales prediction.

For the purposes of the analysis, the dataset is further divided to training and testing datasets. Training dataset contains the months 01/2016 – 08/2018 (32 months) and the testing dataset the months 9-10/2018 (2 months). When training the models, first 6 months of data are left out of training set since for these rows, most of the

lagged features extracted from the data are not available.

Rows with negative sales quantity are dropped from the dataset. Negative sales are related to product returns. Products returns is a separate topic for modelling and prediction and is left out of scope here. Focus is on predicting gross sales figures.

5.2 Problem formulation

In general form, the problem can be summarized as the task to learn the function from set cartesian product of set of stores S, set of products P and set of months $T \subset \mathbb{Z}_0^+$ to set of sales $Y = \mathbb{R}_0^+$,

$$\begin{aligned} F : S \times P \times T &\rightarrow Y \\ F(s, p, t) &= y \end{aligned} \tag{5.1}$$

using the observations $\mathcal{D} = \{((s_i, p_i, t_i), y_i) \mid s_i \in S, p_i \in P, t_i \in T\}_{i=1}^n$.

Before applying the machine learning algorithms, features are generated from raw data and the resulting dataset is represented as a feature matrix $\boldsymbol{X}$ where each row $i = 1, ..., n$ represents an observation and each column $j = 1, ..., m$ represents a feature:

$$\boldsymbol{X} = \begin{array}{c} \begin{matrix} f_1 & f_2 & f_3 & ... & f_m \end{matrix} \\ \begin{bmatrix} x_{11} & x_{12} & x_{13} & ... & x_{1m} \\ x_{21} & x_{22} & x_{23} & ... & x_{2m} \\ x_{31} & x_{32} & x_{33} & ... & x_{3m} \\ ... & ... & ... & ... & ... \\ x_{n1} & x_{n2} & x_{n3} & ... & x_{nm} \end{bmatrix} \end{array} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix}. \tag{5.2}$$

The data for the target variable y consists of sales observations $\{y_i\}_{i=1}^n$. Each observation i is associated with a time t_i. If prediction is made one month into the future, the data available for making predictions consists of data collected up to time $t_i - 1$. If prediction is made h months into the future, data available consists of data collected up to time $t_i - h$. This means the feature values for predicting the same observation y_i depend on the forecast horizon h.

To represent the features which depend on the forecast horizon $h = 1, 2, ..., q$, each observation in the data is represented q times, once for each forecast horizon, in a

feature matrix defined as follows:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{1} & \boldsymbol{X}_1 \\ \boldsymbol{2} & \boldsymbol{X}_2 \\ \vdots & \vdots \\ \boldsymbol{q} & \boldsymbol{X}_q \end{bmatrix}, \tag{5.3}$$

where $\boldsymbol{X} \in \mathbb{R}^{nq \times (m+1)}$ is the feature matrix for all forecast horizons, $\boldsymbol{1}$, $\boldsymbol{2}, ..., \boldsymbol{q} \in \mathbb{R}^n$ are constant vectors of length n which denote the forecast horizon and $\boldsymbol{X}_h \in \mathbb{R}^{n \times m}$ is a feature matrix for forecast horizon h.

The machine learning problem can be formulated as a function approximation problem using the notation from section 4.1.2:

$$\hat{F} = \arg\min_F \frac{1}{nq} \sum_{i=1}^{nq} L(F(\boldsymbol{x}_i), y_i), \tag{5.4}$$

where L is the loss function, y_i is the target variable sales and $\boldsymbol{x}_i$ are the feature data associated with i:th observation which corresponds to a single row in feature matrix $\boldsymbol{X}$ in equation (5.3). Target varible does not depend on forecast horizon and thus $y_i = y_{i+nh} \, \forall \, i = 1, ..., n, h = 1, ..., q-1$. In this representation, each data row is associated with a unique store, product, month and forecast horizon -combination.

5.3 Feature engineering

Constructing a good feature representation of the raw data is perhaps the most important single step in the machine learning workflow as noted among others by Zheng and Casari (2018). This section describes the features created from the raw data.

A simple feature denoting time (i.e., month number) is derived by ordering the months in the dataset in ascending order and enumerating them. The month number starts at zero at the first month in the dataset. A feature that indicates the calendar month (1 - 12) is also derived. In addition, categorical variables denoting the store, product, brand and category levels 1, 2 and 3 are constructed. In some cases, these are inputted to models as raw integers and in others, they are transformed to categorical

dummies using one-hot encoding.

Features denoting the first and last month the product had a transaction in any of the stores and the first and last month in the product had a transaction in the store associated with the data row are derived. Furthermore, related features denoting the number of months between first and last transactions and features denoting the number of months elapsed between current month and first and last transactions are derived.

A feature (row type) denoting the type of the prediction task corresponding to each row is also derived. Every data row is associated with a row type that is either old store product, new store product or new product. These categories corresponds to a product that has been sold before in that store (research question 1), a product that has not been sold before in the store but has been sold in other stores (research question 2) and a product that has not been sold before in any of the stores (research question 3). This feature is used to separate the data for different prediction tasks so that they can be handled separately. Secondly, it can be used to create aggregate features that use only data corresponding to a specific prediction task. For instance, only sales data from previous new product introductions can be used when creating lagged or aggregate features for predicting new product sales.

Multiple features of lagged aggregated monthly sales at different lags and aggregation levels are derived. An aggregate function (for example mean, median, min or max) takes a finite set of real numbers as an argument and returns a real number. Let A be an aggregate function defined over set of finite sets of real numbers $\{Z \mid Z = \{z_1, z_2, ..., z_r\}, z_i \in \mathbb{R}, r \in \mathbb{N}\}$:

$$A : \{Z \mid Z = \{z_1, z_2, ..., z_r\}, z_i \in \mathbb{R}, r \in \mathbb{N}\} \rightarrow \mathbb{R}. \tag{5.5}$$

As described before, the raw data is duplicated so that each row in the raw data is represented q times, once for each forecast horizon. The forecast horizon h_i associated with a row determines which part of the past sales data is available for creating aggregate features. If prediction is made h_i months into the future, data collected up to time $t_i - h_i$ is available. Because available data depends on forecast horizon, aggregated features depend on forecast horizon as well. Define $I = \{1, ..., nq\}$ as the instance set of all observations in the duplicated dataset. Data available for

observation i having forecast horizon h_i can represented by defining an instance set $I(i) = \{j \mid t_j \leq t_i - h_i, j \in I\}$. It contains the indices of observations that are available for making aggregate features for observation i. Aggregate feature of sales observations can now be defined as:

$$A\Big(\{y_j \mid j \in J(i) \cap I(i)\}\Big), \tag{5.6}$$

where $A(Z)$ is an aggregation function, $I(i)$ is instance set of available observations and $J(i) \subset I$ is an instance set that contains indices of observations belonging to specific aggregation level. Both instance set of available observations $I(i)$ and instance set of observations that belong to an aggregation level $J(i)$ vary based on observation i.

For example, by choosing $J(i) = \{j \mid s_j = s_i, j \in I\}$ and $A = \text{mean}$, a feature denoting historical average sales in the same store can be created:

$$\begin{aligned} \text{mean}\Big(\{y_j \mid j \in J(i) \cap I(i)\}\Big) &= \text{mean}\Big(\{y_j \mid s_j = s_i, t_j \leq t_i - h_i, j \in I\}\Big) \\ &= \frac{\sum_{k \in \{j \mid s_j = s_i, t_j \leq t_i - h_i, j \in I\}} y_k}{|\{j \mid s_j = s_i, t_j \leq t_i - h_i, j \in I\}|}. \end{aligned} \tag{5.7}$$

In this thesis, different lagged aggregate features are created. A lagged aggregate sales feature requires that only data from l months back is used, i.e., only sales observations such that $t_j = t_i - l$ are used. Lagged aggregate sales feature in general form can be denoted as:

$$A\Big(\{y_j \mid t_j = t_i - l, j \in J(i) \cap I(i)\}\Big). \tag{5.8}$$

In addition to choosing lags l, constructing different types lagged aggregate sales features comes down determining the correct subset of observations $J(i)$ to aggregate over for each observation i. Lagged aggregate sales per store uses data only from the store corresponding to observation i:

$$A\Big(\{y_j \mid s_j = s_i, t_j = t_i - l, j \in I(i)\}\Big). \tag{5.9}$$

Similarly, lagged aggregate sales per product uses data only from the product

corresponding to observation i:

$$A\Big(\{y_j \mid p_j = p_i, t_j = t_i - l, j \in I(i)\}\Big). \tag{5.10}$$

Aggregation can depend on multiple conditions. For example, lagged aggregate store category sales uses only data from the store and the product category corresponding to observation i:

$$A\Big(\{y_j \mid s_j = s_i, p_j \in C_i, t_j = t_i - l, j \in I(i)\}\Big), \tag{5.11}$$

where $C_i \subset P$ is a product category such that $p_i \in C_i$.

Lagged aggregate store product sales uses data only from the store and product associated with the observation i:

$$A\Big(\{y_j \mid s_j = s_i, p_j = p_i, t_j = t_i - l, j \in I(i)\}\Big) = g_L(i, l). \tag{5.12}$$

Because each observation in raw data is uniquely determined by a combination of store, product and month, this aggregation level corresponds to a single observation in the past data or to an empty set, and the feature is referred to simply as lagged sales $g_L(i, l)$. All aggregation functions used in this thesis (only arithmetic mean is used) have the property $A(\{y\}) = y$. Thus,

$$g_L(i,l) = \begin{cases} y_j, & \text{if } \exists j \in I(i) : s_j = s_i, p_j = p_i, t_j = t_i - l \\ undefined, & \text{otherwise.} \end{cases} \tag{5.13}$$

It is common that this feature - this also applies to aggregate features but to a lesser degree - is undefined because there are months in which some products do not have any sales. In the data, undefined values are represented by null entries.

A variation of lagged aggregated sales named last known lagged aggregate sales is used also. Last known lagged aggregate sales with lag l is defined:

$$A\Big(\{y_j \mid t_j = t_i - l_{min}(i, l), j \in J(i) \cap I(i)\}\Big), \tag{5.14}$$

where $l_{min}(i, l) = \min\{l' \mid l' \geq l, t_j = t_i - l', l' \in \mathbb{N}, j \in J(i) \cap I(i)\}$ is the smallest

lag greater than or equal l for which there is data available. In other words, last known aggregate sales is the most recent lagged aggregate sales feature available in the data. To clarify, the extended form of the set-builder notation is used here, i.e.,

$$\{\Psi(j,l') \mid \Phi(j,l')\} = \{l' \mid l' \geq l, t_j = t_i - l', l' \in \mathbb{N}, j \in J(i) \cap I(i)\}, \tag{5.15}$$

where $\Psi(j,l') = l'$ and $\Phi(j,l')$ is a predicate returning either true or false depending on whether the proposition is true for given j and l'.

Last known lagged aggregate sales features are less sparse than lagged aggregate sales features since if a lagged feature value is missing for some month, the most recent known value from history is used instead. Last known values can convey more or less the same information as lagged features but in a less sparse format. However, some of the time structure is lost since the last known value with lag $t-l$ can represent a value from any of the months $t-l$, $t-l-1$,...,1, depending on which is the first one available.

For predicting sales of new store products (research question 2), features denoting last known lagged aggregate sales of same product in other stores are derived:

$$A\Big(\{y_j \mid s_j = s_k, p_j = p_i, t_j = t_i - l_{min}(i,k,l), j \in I(i)\}\Big) = g_{LK}(i,k,l), \tag{5.16}$$

where $s_k \neq s_i$ is a different store than the store associated with observation i and $l_{min}(i,k,l) = \min\{l' \mid l' \geq l, s_j = s_k, p_j = p_i, t_j = t_i - l', l' \in \mathbb{N}, j \in I(i)\}$ is the smallest lag greater than or equal l for which there is data available. These features are referred to as last known lagged sales from other stores $g_{LK}(i,k,l)$. The attribute "aggregate" can be dropped from the name as a raw data row is uniquely determined by a combination of store, product and month, and hence the aggregation level in the above equation corresponds to a single past sales observation or to an empty set.

Last known lagged sales features are used in the case of research question 2 instead of lagged sales features because, with the former, almost the same information can be represented with a smaller number of features. It is quite common that lagged sales features have missing values, and thus more lags would need to be used for lagged sales features to capture the same information as in the case of last known lagged sales features. These features are made for all other stores in the data - 19 stores

in this case. Therefore, representing lagged product sales in other stores results in $19r$ features, where r is the number of lags. In order to limit the overall size of the dataset given to models and avoid the curse of dimensionality, it is important to use a small number of features per store.

Latent semantic analysis is performed on the product names to create numerical features from text data. These features are used in research question 3. Given that each product name is a document and each word is a term, a term-document matrix with tf-idf weights is created. The term-document matrix is then truncated to 100 term features using SVD as outlined in section 4.3.1. As a result, each product name can be represented by a 100-dimensional numerical vector. Components of this vector are used as features.

Missing values in the features are replaced by zeros before the data is given as input to the models. In the case of XGBoost it is not necessary to replace missing values with zeros because the XGBoost algorithm can handle missing values algorithmically. It was found out that retaining missing values gives worse results, and thus they are replaced with zeros also in the input data given to XGBoost. Tables 5.1, 5.2 and 5.3 show the features extracted from the raw data in the case of research questions 1, 2 and 3 respectively. In total 565, 519 and 476 features are extracted in data preparation steps for research questions 1, 2 and 3.

Table 5.1: Features extracted from raw data in the case of research question 1. OH indicates that feature was one-hot encoded.

Feature name	Explanation
store	store identifier
product	product identifier
brand	brand identifier (OH)
c1	category level 1 identifier
c2	category level 2 identifier (OH)
c3	category level 3 identifier
t	incremental month number starting from 0 (01/2016)
m	number of calendar month (1-12) (OH)
store_product_sales lags 1-6, 12	1,2,3,4,5,6 and 12 months lagged sales
mean_product_sales lags 1-6, 12	1,2,3,4,5,6 and 12 months lagged mean product sales
mean_store_c1_sales lags 1-3	1,2 and 3 months lagged mean store category level 1 sales
mean_store_c2_sales lags 1-3	1,2 and 3 months lagged mean store category level 2 sales
mean_store_c3_sales lags 1-3	1,2 and 3 months lagged mean store category level 3 sales
mean_store_brand_sales lags 1-3	1,2 and 3 months lagged mean store brand sales
mean_store_c3_brand_sales lags 1-3	1,2 and 3 month lagged mean store brand category level 3 sales
t_first_sales	first sales month for product in any store
t_first_sales_store	first sales month for product in same store
t_last_sales	last sales month for product in any store
t_last_sales_store	last sales month for product in same store
months_between_first_and_last_sales	months between first and last product sales in any store (OH)
months_between_first_and_last_sales_store	months between first and last product sales in same store (OH)
months_since_first_sales	months since first product sales in any store (OH)
months_since_first_sales_store	months since first product sales in same store (OH)
months_since_sales	months since last product sales in any store (OH)
months_since_sales_store	months since last product sales in same store (OH)

Table 5.2: Features extracted from raw data in the case of research question 2. OH indicates that feature was one-hot encoded.

Feature name	Explanation
store	store identifier
product	product identifier
brand	brand identifier (OH)
c1	category level 1 identifier
c2	category level 2 identifier (OH)
c3	category level 3 identifier
t	incremental month number starting from 0 (01/2016)
m	number of calendar month (1-12) (OH)
mean_product_sales lags 1-6, 12	1,2,3,4,5,6 and 12 months lagged mean product sales
mean_store_c1_sales lags 1-3	1,2 and 3 months lagged mean store category level 1 sales
mean_store_c2_sales lags 1-3	1,2 and 3 months lagged mean store category level 2 sales
mean_store_c3_sales lags 1-3	1,2 and 3 months lagged mean store category level 3 sales
mean_store_brand_sales lags 1-3	1,2 and 3 months lagged mean store brand sales
mean_store_c3_brand_sales lags 1-3	1,2 and 3 month lagged mean store brand category level 3 sales
t_first_sales	first sales month for product in any store
t_last_sales	last sales month for product in any store
months_between_first_and_last_sales	months between first and last product sales in any store (OH)
months_since_first_sales	months since first product sales in any store (OH)
months_since_sales	months since last product sales in any store (OH)
product_sales lags 1-3 for 20 stores	1, 2 and 3 months last known lagged sales in other stores (one feature per each lag and store combination)

Table 5.3: Features extracted from raw data in the case of research question 3. OH indicates that feature was one-hot encoded.

Feature name	Explanation
store	store identifier (OH)
brand	brand identifier (OH)
c1	category level 1 identifier
c2	category level 2 identifier (OH)
c3	category level 3 identifier
t	incremental month number starting from 0 (01/2016)
m	number of calendar month (1-12) (OH)
mean_store_c1_sales lags 1-3	1,2 and 3 months lagged mean store category level 1 sales
mean_store_c2_sales lags 1-3	1,2 and 3 months lagged mean store category level 2 sales
mean_store_c3_sales lags 1-3	1,2 and 3 months lagged mean store category level 3 sales
mean_store_brand_sales lags 1-3	1,2 and 3 months lagged mean store brand sales
mean_store_c3_brand_sales lags 1-3	1,2 and 3 month lagged mean store brand category level 3 sales
lsa	features 1-100 from latent semantic analysis of product names

5.4 Machine learning models

For each of the research questions, three types of models are studied: a benchmark model, Lasso regression and XGBoost. The benchmark model is a last value prediction in the case of research question 1. It predicts the previous available sales quantity for a product in store. For research question 2, the benchmark model predicts the average of last sales observations from other stores. In the case of research question 3, the benchmark model is the category level 3 average sales.

Lasso regression is chosen as one of the models because it is a simple linear model with feature selection and regularization embedded in it. Linear models have shown good performance in sales forecasting (Fildes et al., 2019). If a linear model can achieve good accuracy, it is usually the best choice because if its high degree of transparency and explainability. XGBoost is chosen as the second model because it is one of the best performing machine learning models in the case of structural (i.e., tabular) data in recent years. This is evidenced by its great success in many machine learning contests, such as those in Kaggle, one of the world's largest platforms for data science competitions (Lardinois et al., 2017). For example, among the 29 challenge winning solutions published at Kaggle's blog during 2015, 17 solutions used XGBoost (Chen and Guestrin, 2016). The benchmark models are included in the experimental setup so that it is possible to assess the degree to which machine learning -based models can create added value in terms of prediction accuracy.

Lasso regression, XGBoost and benchmark models are trained using feature engineered data. Lasso regression model is trained using the Python's scikit-learn package (Pedregosa et al., 2011). XGBoost model is trained using xgboost-package (Chen and Guestrin, 2016). Squared error is used as loss function $L(\hat{F}(\boldsymbol{x}), y) = (y - \hat{F}(\boldsymbol{x}))^2$ and mean squared error as an objective function or a measure of empirical risk:

$$\hat{R}(F) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{F}(\boldsymbol{x}_i))^2. \tag{5.17}$$

Evaluation metric is relative root mean squared error (RRMSE):

$$\text{RRMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{F}(\boldsymbol{x}_i))^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}}. \tag{5.18}$$

The evaluation metric RRMSE is minimized when mean squared error is minimized.

Data is pooled by and separate models are trained for prediction horizons $h = 1, 2, ..., 6$ months in the case of research questions 1 and 3. Thus, six models are trained in the case of these research questions. For research question 2, data is pooled by both prediction horizon and store. This means separate models are built for each prediction horizon and store. Thus, $6 \times 20 = 120$ models are trained. In the case of this research question, it is convenient to make a separate model for each store. The models can only utilize lagged product-level sales data from other stores because product-level sales data from the store prediction is made for is not available, due to the fact that the product has not been sold there. In addition, a store-specific model can attempt to learn which other stores' data is most predictive for that particular store.

In the case of research questions 1 and 2 models are fitted and evaluated using full data including all sales: existing products, new store products and new products. For research question 3, models are trained only using data from new product sales.

Models are fitted using the following combined hyperparameter optimization and cross-validation setup. Training dataset (months 1 – 32) is divided into three folds with first fold having months 1 - 29 as training set and month 30 as validation set, second fold having months 1 – 30 as training set and month 31 as validation set, and third fold having months 1 – 31 as training set and month 32 as validation set. In each iteration of the hyperparameter optimization, a model is fitted with given hyperparameters in the training sets, predictions are made for the months in the validation sets and errors are calculated using the evaluation metric. In the case of XGBoost, early stopping is also used. This means more trees are added to the ensemble until the evaluation metric in the validation set stops improving. Because there are three folds, in each iteration of the hyperparameter optimization, models are fitted three times and prediction is made for unseen data of the validation sets. The evaluation error for the hyperparameters is calculated as the average of the evaluation error in the three validation sets.

Hyperparameter optimization is performed using Ax- and BoTorch- packages (Balandat et al., 2019). The hyperparameter search space for Lasso regression is $\alpha \in [0, 5]$. Hyperparameter search space for XGBoost is: $\eta \in [0.01, 0.8]$, $\gamma \in [0.02, 100]$,

$d_{max} \in [4, 15]$, colsample_bytree $\in [0.4, 1]$ and colsample_bylevel $\in [0.4, 1]$. Hyperparameter search is performed for 20 iterations for Lasso regression and for 60 iterations for XGBoost. After this, the models are fitted using the whole training data and the best hyperparameter combination found.

Models are trained and evaluated using Amazon Web Services EC2 instances with 32 cores and 128 GB of memory. Depending on the research question, it took between 1 and 8 hours to extract the features and train the models using the cross-validation and hyperparameter optimization setup.

6 Results

For reference, the three research questions chosen for the thesis are:

1. How can future monthly sales of existing products be predicted across multiple stores, assuming sales is independent of the availability of other products in store assortment?
2. How can monthly sales of new store products be predicted, assuming these products have been sold before in another store?
3. How can monthly sales of new products be predicted based on sales data from other products?

Results for each research question are presented and analysed in separate subsections.

6.1 Research question 1

Figure 6.1 shows the relative root mean squared error for the three models as a function of prediction horizon, last 12 month store sales and last 12 month category level 2 sales for existing product sales prediction. Few categories with small number of sales transactions have constant sales for all products in all stores. They are omitted from the per category RRMSE graph because RRMSE is undefined if there is no variation in sales.

From the first graph in Figure 6.1 it can be seen that overall, XGBoost gives the most accurate predictions, followed by Lasso regression and last value benchmark. Prediction accuracy weakens somewhat as predictions are made further into the future. RRMSE values are well below 1 which indicates good performance.

The second graph suggests that the performance of the models in terms of accuracy is good and fairly uniform across stores. However, there appears to be a rough trend downwards in the graph. Hence, it seems that as store size increases, and thus amount of data increases, predictions tend to be on average more accurate. Order of the models in terms of accuracy is the same as observed in first graph: XGBoost is best followed by Lasso regression and then last value benchmark.

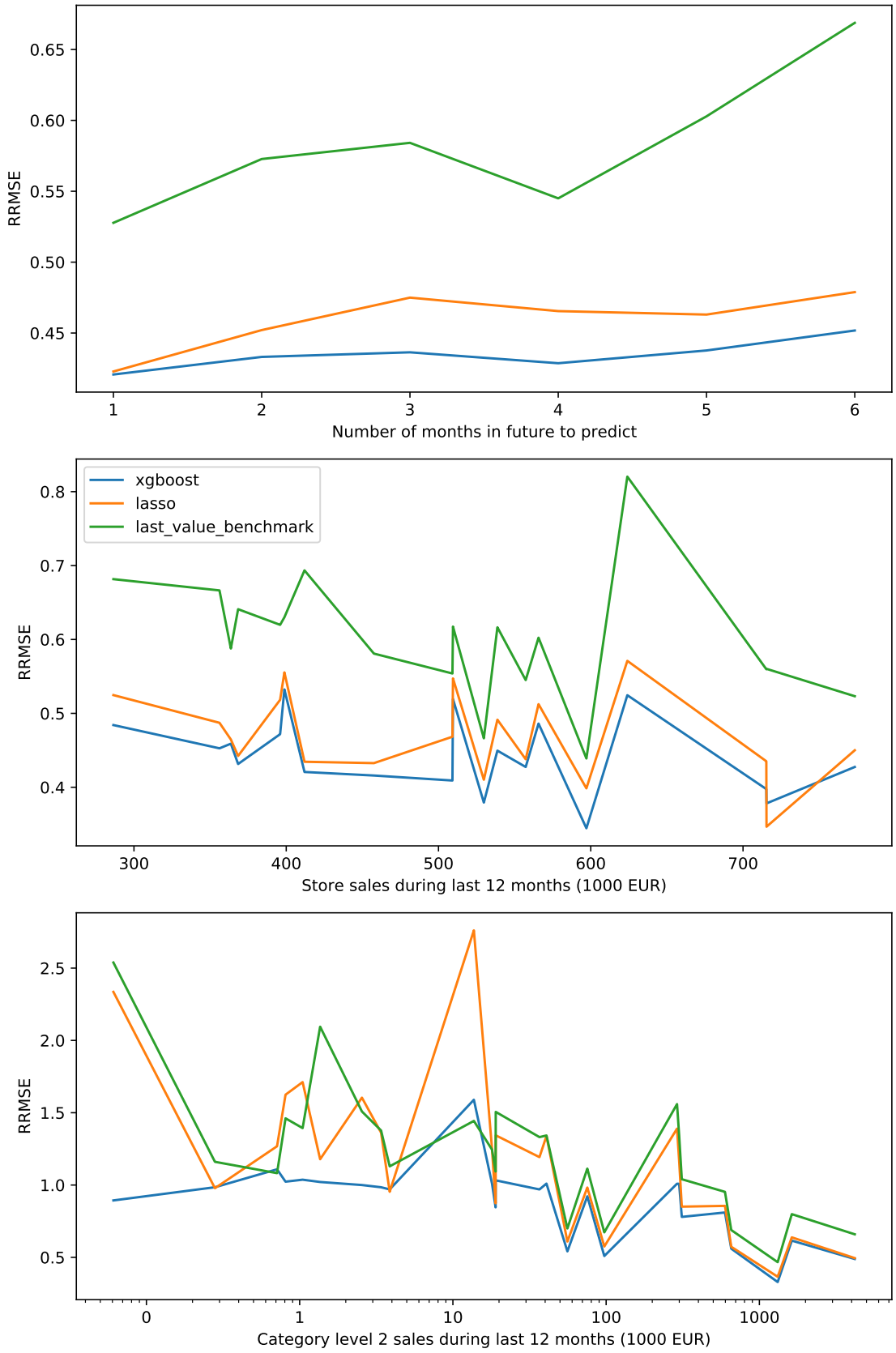

Figure 6.1: Relative root mean squared error (RRMSE) of predictions of existing store products by prediction horizon, last 12 month store sales and last 12 month category level 2 sales for XGBoost, Lasso regression and benchmark model.

The third graph shows sales as a function of category level 2 sales. The x-axis scale is logarithmic since categories vary greatly in size as measured by total sales in last 12 months. Prediction accuracy gets better as category size increases, which is in line with larger categories having more data to learn from. It is also notable that there seems to be more variation or noise in the graph for smaller categories. This is likely caused by smaller size of the testing dataset for smaller categories.

All in all, in the case sales prediction for existing products, the machine learning models give clear added value over rule-based benchmark decreasing RRMSE between 0.1 - 0.15 points depending on the prediction horizon.

Figure 6.2 shows the feature importance for top 20 predictors for XGBoost and Lasso regression models in the case of research question 1.

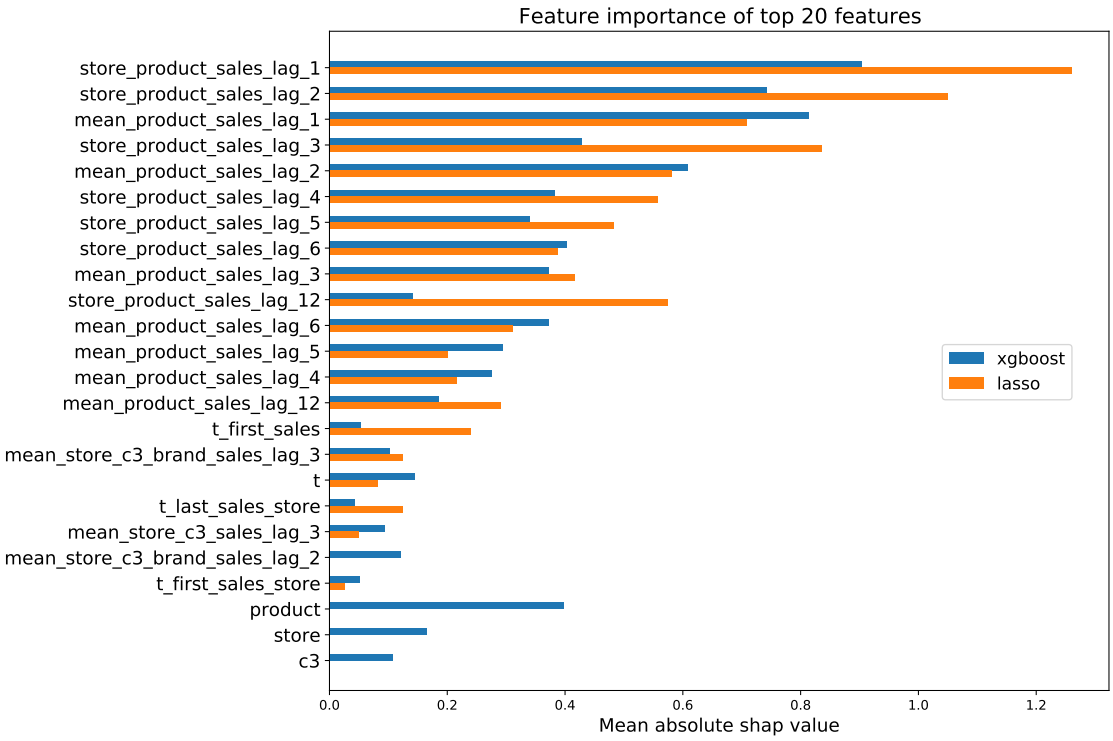

Figure 6.2: Feature importance as measured by mean absolute SHAP value for 20 most important predictors for XGBoost and Lasso regression in the case of existing store product sales prediction. Features among the top 20 predictors of either model are included.

Most of the top predictors are same in both models. Lagged sales of the same product in same store is most predictive followed by lagged average sales of the same product

across all stores. The nearest lags have most predictive power. Also 12-month lagged sales feature is important indicating that sales have some seasonality that depends on calendar month, and this seasonality is captured by the models. A couple of other features like month number and month number of first sales transaction are also among top predictors. XGBoost uses more predictors than Lasso as is expected. The L1-norm regularization used in Lasso regression leads to most of the features to have zero coefficients.

6.2 Research question 2

Figure 6.3 shows the relative root mean squared error for the three models as a function of prediction horizon, last 12 month store sales and last 12 month category level 2 sales for new store product sales prediction. Few categories with small number of transactions have constant sales for all products in all stores also in this case. They are omitted from the per category RRMSE graph.

From the first graph in Figure 6.3 it is seen that XGBoost and Lasso regression have roughly the same overall accuracy and perform slightly better than the benchmark model. The benchmark model in this case is the average sales of the product in the other stores. RRMSE in this case is much higher, slightly below 1, than for existing store product sales prediction. By using the set of features in Table 5.2, the same level of prediction accuracy as for existing products cannot be reached. This indicates that it is harder to predict sales of new store products than existing store products based on the available data.

Second graph shows that performance across stores is fairly uniform. Accuracy for large stores might be on average a bit better than for small stores. It seems that in the case of larger stores XGBoost and Lasso regression more consistently outperform the benchmark. However, there is a lot of noise in the graph due to the limited size of testing data, and thus these observations are tentative.

Based on the third graph it seems that last value benchmark performs best for small categories on average whereas for large categories XGBoost and Lasso perform best. Similarly as for stores, increased data availability due to larger category size appears to lead to better performance for both XGBoost and Lasso regression. There is considerable variation: some categories seem harder to predict than others. More

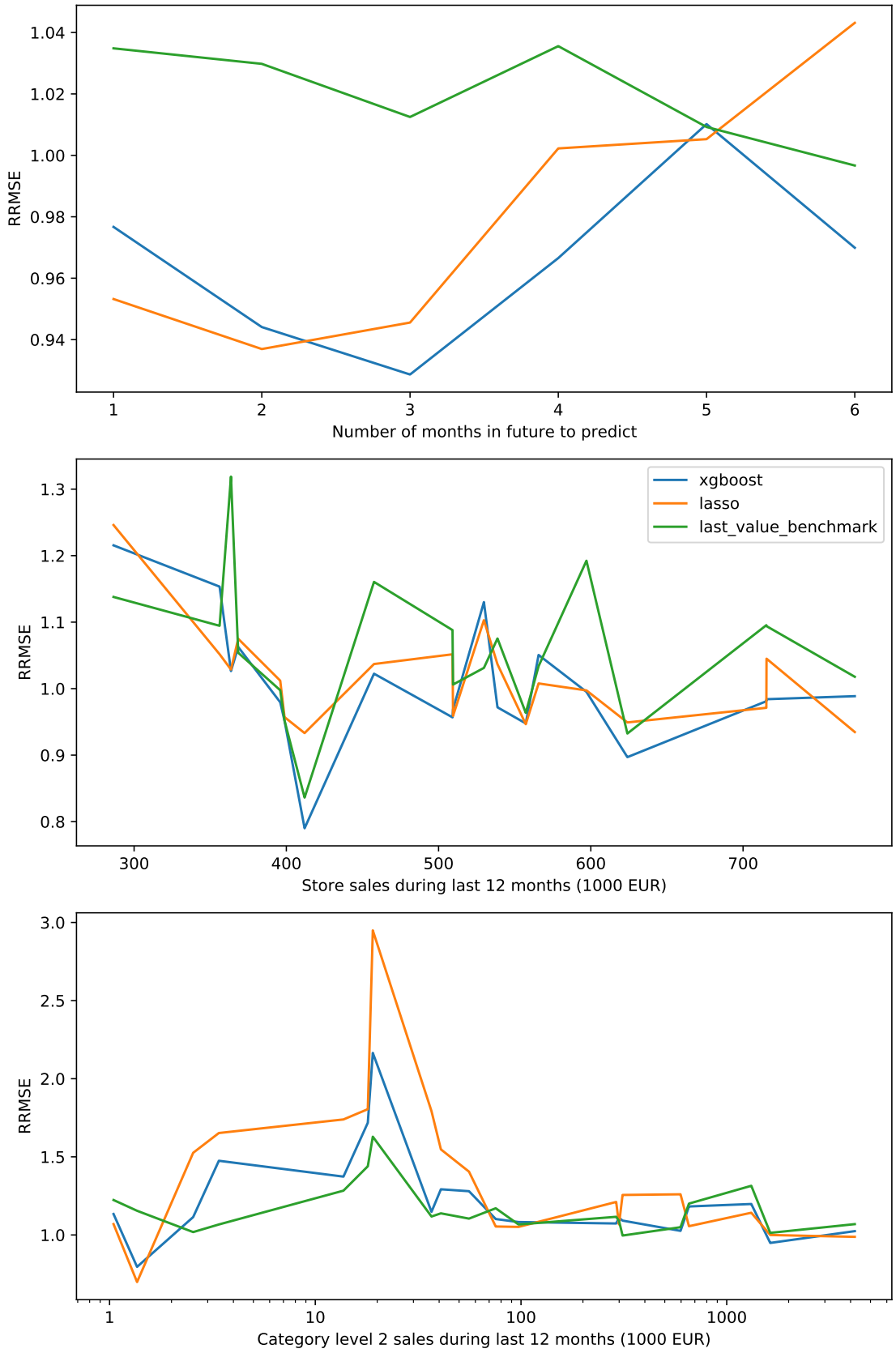

Figure 6.3: Relative root mean squared error (RRMSE) of predictions of new store products by prediction horizon, last 12 month store sales and last 12 month category level 2 sales for XGBoost, Lasso regression and benchmark model.

experiments and a larger testing dataset would be needed to see if the observed differences are due to sales in some categories being inherently more difficult to predict or due to random noise.

Overall, XGBoost and Lasso regression give between 0 - 0.1 points lower RRMSE than benchmark. They provide added value but less than in the case of existing sales prediction. It is notable that RRMSE is only slightly below 1 for this research question. RRMSE=1 can be achieved by the average of new store product sales in the testing set. Getting below 1 in terms of RRMSE is an important additional benchmark because constant prediction can produce performance of almost RRMSE=1. However, RRMSE=1 cannot be achieved exactly since the constant prediction needs to be made based on training data, not testing data. Nevertheless, the fact that RRMSE is only slightly below 1 indicates that the machine learning models give less added value than in the case of research question 1.

Figure 6.4 shows the feature importance for top 20 predictors for XGBoost and Lasso regression models in the case of research question 2.

As in research question 1, most top predictors for XGBoost and Lasso are the same. Lagged average product and store category sales and last known lagged sales from other stores are most important predictors. The nearest lags seem to hold most predictive power on average. In this case, also the product and the category levels are important predictors. It seems that lagged average sales features are more important for XGBoost than for Lasso regression. For Lasso regression, last known lagged sales are more important in comparison. Store number 107 sales are most predictive. This is the largest store in the dataset and thus it likely has the most data available.

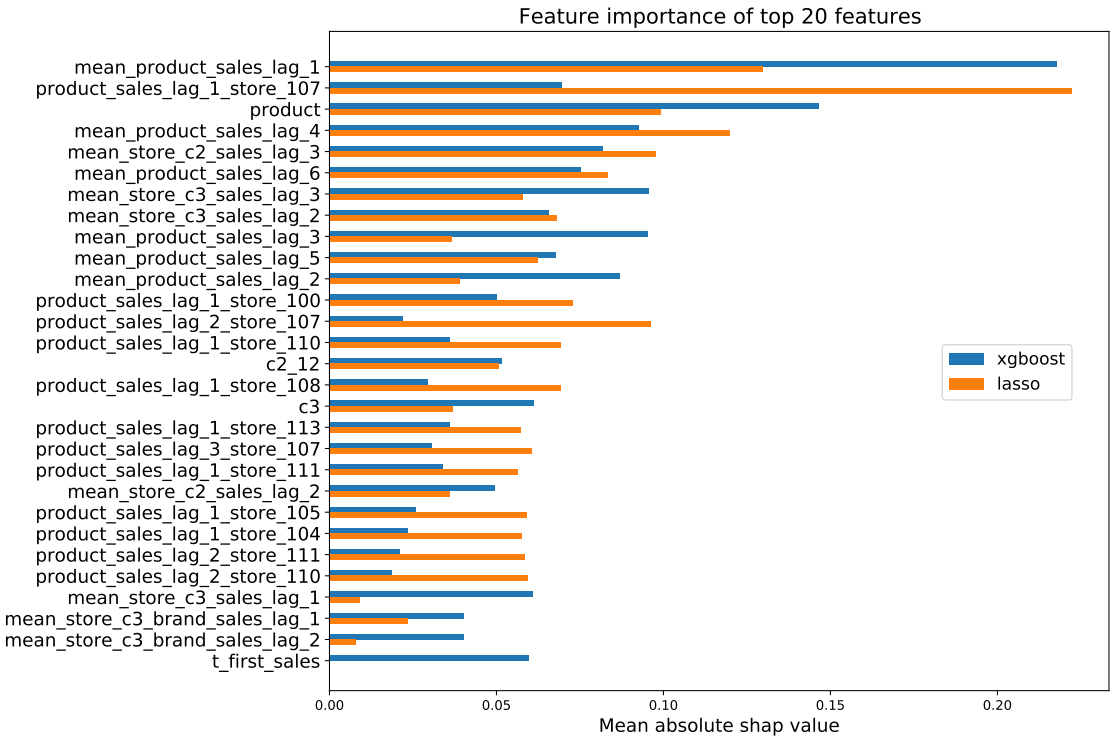

Figure 6.4: Feature importance as measured by mean absolute SHAP value for 20 most important predictors for XGBoost and Lasso regression in the case of new store product sales prediction. Features among the top 20 predictors of either model are included.

6.3 Research question 3

Figure 6.5 shows the relative root mean squared error for the three models as a function of prediction horizon, last 12 month store sales and last 12 month category level 2 sales in the case of new product sales prediction. Also in this case, few categories with small number of transactions have constant sales for all products in all stores and they are left out from the per category RRMSE graph.

The first graph in Figure 6.5 shows that XGBoost and Lasso regression are more accurate than category average sales prediction by a considerable margin. Category level average was calculated based on all data and thus didn't take into account the fact that new product sales are likely lower on average than sales of existing products. XGBoost and Lasso have similar levels of accuracy except for the for forecast horizon 6 months where XGBoost performs worse than Lasso. XGBoost is slightly more accurate than Lasso regression when predicting 1 - 3 months into the future; Lasso

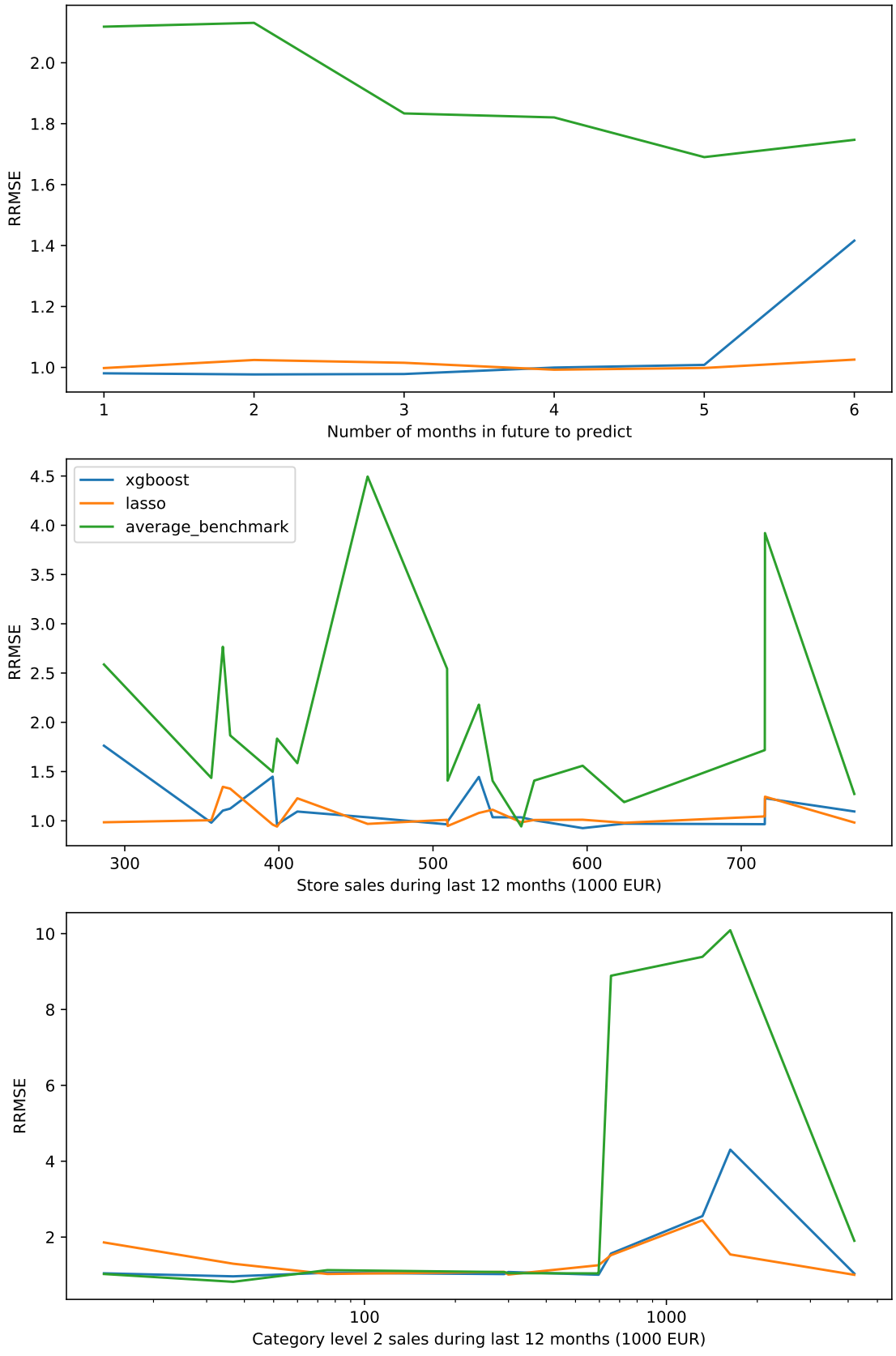

Figure 6.5: Relative root mean squared error (RRMSE) of predictions of new products by prediction horizon, last 12 month store sales and last 12 month category level 2 sales for XGBoost, Lasso regression and benchmark model.

is more accurate when predicting 4 - 6 months into the future. Relative root mean squared error hovers around and slightly above 1 which indicates worse accuracy than in the case of research questions 1 and 2. For clarification, because the exact RRMSE figures are not as clearly visible from the y-axis in this case due to wider y-axis scale, RRMSE for XGBoost for prediction horizons between 1 and 3 months is 0.98 and then rises to 1 and above it, and RRMSE for Lasso regression is in the interval 0.99 - 1.03 for all prediction horizons. The fact that RRMSE is higher in this case than in the case of research questions 1 and 2 seems reasonable as predicting the sales of altogether new products is likely harder than predicting sales of existing products or products that have been sold in other stores.

The second graph indicates that the accuracy is fairly uniform across different store sizes except for category average benchmark which has clearly worse accuracy for some stores. The third graph indicates that performance is also reasonably uniform across category size except for large categories. For large categories the accuracies of all models are lower. This could be explained by the fact that the large categories have more diverse set of products, and thus also the sales levels of new products vary more, which in turn makes the prediction task harder for the models. Also, both the training and testing datasets in the case of new products are considerably smaller than in case of research questions 1 and 2. In the dataset, it is quite rare that a new product is introduced. It might be that the amount of data is insufficient for models to properly learn the factors that account for the variance in sales.

As a whole, it appears that also in new product sales prediction, XGBoost and Lasso regression give some added value over rule-based benchmark and constant prediction, but perhaps less so than in the case of research questions 1 and 2. For sales prediction up to 3 months into the future, XGBoost still gives lower than 1 RRMSE which indicates that the model is able to produce added value compared to the constant sales prediction which is calculated based on new product sales only. For prediction up to 4 - 6 months into the future, RRMSE is around or above 1, which indicates less or no added value is produced.

Figure 6.6 shows the feature importance for top 20 predictors for XGBoost and Lasso regression models in the case of research question 3.

There is a bigger difference between the top predictors of XGBoost and Lasso

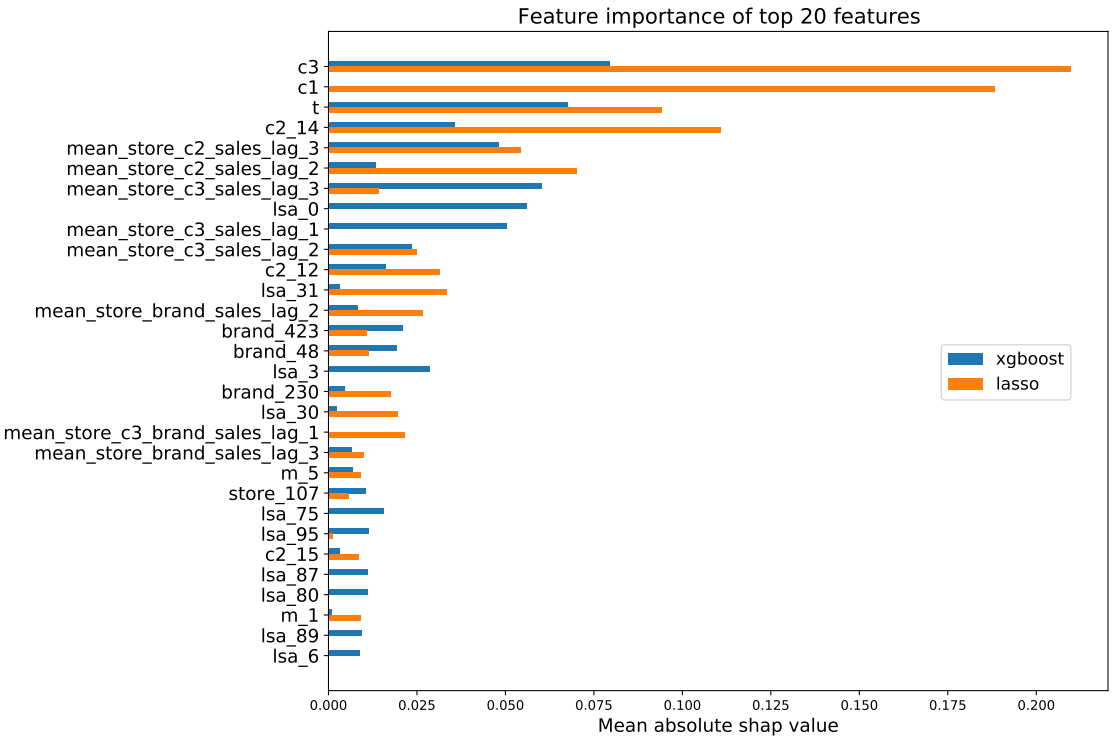

Figure 6.6: Feature importance as measured by mean absolute SHAP value for 20 most important predictors for XGBoost and Lasso regression in the case of new product sales prediction. Features among the top 20 predictors of either model are included.

regression in the case of research question 3 than in the case of research questions 1 and 2. Some features that are important for Lasso regression are not important for XGBoost, and vice versa. Overall, category level information, month number and lagged average store category sales have the most predictive power. In this case, the nearest lags are not most predictive. Some of the latent semantic analysis features are important especially for XGBoost. Brand information is somewhat important as well, which is demonstrated by the fact that one-hot encoded brand features and lagged average brand sales features are among the top predictors.

7 Conclusions

This study investigates three research questions pertaining to prediction of future monthly sales of products in stores of a retail chain. First research question concerns prediction of future sales of existing products already in the assortment of a store. Second deals with sales prediction for such new products that are not included in the store's assortment, but have been sold before in other stores. Third research question concerns sales prediction for altogether new products from which there is no prior sales data.

For sales prediction of existing products in stores, XGBoost gives the best overall performance. Both XGBoost and Lasso regression are better than the benchmark last value prediction by a clear margin. Relative root mean squared errors are clearly below 1 which indicates good performance. All in all, in the case of sales prediction for existing products the machine learning models give clear added value over the rule-based benchmark model.

For sales prediction of new store products, which have been sold before in other stores, XGBoost and Lasso regression are somewhat more accurate than the rule-based benchmark. They provide added value, but less than in the case of existing product sales prediction.

In the case of sales prediction for new products, both models clearly outperform benchmark which predicts lowest level product category average sales. However, RRMSE is only slightly below 1. It seems that also in this case machine learning models give some added value over the rule-based benchmark and constant prediction, but less so than in the case of research questions 1 and 2.

In addition to providing more accurate predictions, the developed sales prediction framework can provide value due to its automatization and large-scale deployment capability. Unlike traditional univariate models, which usually are limited to sales prediction for existing products, the model framework can predict sales for existing store products, new store products and new products. Moreover, missing values and short time series for store products, which are common in the case of retail sales data, are not a problem because the models learn from multiple aggregated features: e.g., if granular product-level data is unavailable, category-level data is used instead.

Using the pipeline, the recurring prediction of sales of tens of thousands of products in hundreds of stores can be automated and implemented in scale.

The machine learning framework built in this thesis is scalable and extendable based on additional data. It is scalable across customers because the same framework can be used for most retailers that have access to historical monthly sales data. It is computationally scalable because the framework is implemented inside a containerized environment, which means that it can be run on any modern computer or cluster of computers. Any number of models can be trained in parallel using cloud service providers such as Amazon Web Services or Google Cloud Platform which provide access to large amounts of computing resources. Furthermore, the framework is easily extendable, should additional types of data be available (e.g., price, customer or promotions data), as these data can be incorporated into the framework by deriving new features while rest of the framework stays the same. It is probable that for example price data, given that future prices are known as well, would provide considerable boost to the accuracy of predictions because sales depend crucially on product prices. In addition, the framework could be used for other purposes than sales prediction. For example, it could be used to derive estimates of product price elasticities of demand by predicting sales based on multiple future price points.

Several additional experiments could be performed to improve and further assess the sales prediction models based on the current experimental setup and dataset. Additional aggregate features could be extracted, such as features that account separately for the sales from existing products, new store products or new products. This can be achieved by aggregating by the feature row type in addition to other variables when deriving lagged aggregate features. In addition to mean features, other types of lagged aggregated features can be used. It would be interesting to see if using aggregate features based on median, maximum, minimum or standard deviation can improve the accuracies. Also, aggregate features that are not based on specific lags could be derived. One potentially useful feature is historical sales aggregated based on calendar month. Using this feature, historical average sales in same calendar month as the prediction is made for can be used directly as a predictor. Moreover, it would be interesting to assess the degree to which the models are able to capture relevant seasonal patterns. In the case of existing product sales prediction, 12 months lagged sales was among the top predictors, which indicates that some

seasonal patterns are captured by the models. Capturing seasonality automatically without explicitly determining which products have strongly seasonal sales would be very useful.

Going forward, it would be interesting to assess the prediction accuracy by using data from other retailers, perhaps with a totally different type of retailer such as online retailer or supermarket with broad selection of product categories. In addition, it would be useful to incorporate additional data sources such as price, customer, marketing and promotions data to see if they improve performance. Moreover, one potential extension to the current setup would be to adapt the modeling framework for prediction of sales that is aggregated to a different level than monthly (e.g., daily or weekly).

In this study, Lasso regression and XGBoost models are used. Lasso regression is used as it is a simple linear model with feature selection and regularization embedded in it. XGBoost is used because it is one of the top performing algorithms in the case of structural data in recent years. However, it is possible to try other types of models and assess if they perform better. Most interesting model types to try are perhaps neural networks and ensemble models. Neural networks have in general performed very well in many problems, and ensembling could improve predictions by combining strengths of different types of models. Furthermore, data pooling is another aspect of the models which could be experimented with. There are many alternative ways to pool the data. For instance, separate models could be fitted for each product, for each product category, or even for each calendar month. Because of the way the framework is implemented, training separate models for arbitrary parts of data is straightforward.

Relative root mean squared error is used as an evaluation metric in this thesis. This metric was chosen because it is a variation of the common approach to minimize sum of squared error. Secondly, it is a measure that is relative to the total variation in the target variable, thus enabling comparisons between results from different research questions. Having said that, other evaluation metrics could be better in some cases. For example, mean absolute error (MAE) is a rather commonly used evaluation metric in sales prediction. A relative version of MAE is mean absolute scaled error (MASE), which is recommended by Hyndman and Koehler (2006). However, the XGBoost algorithm requires that the loss function is at least twice continuously

differentiable, which is a criterion MAE does not fulfill. Nevertheless, XGBoost can be used if Huber loss, a continously differentiable approximation of absolute error loss, is used instead (Friedman et al., 2009).

In this study, a central assumption behind the prediction models was that the sales of products are independent of availability of other products in that location (i.e., the independent demand assumption). This assumption is not true in many cases due to substitution, complementarity and cannibalization effects. Thus, the predictions produced by these models might be off the mark if changes are made to store product assortment. More complex models such as non-parametric choice models and other types of demand estimation models could be used to capture these effects. Alternatively, substitution and complementarity could be modelled separately after sales potential has been forecasted. In any case, depending on the intended use of the predictions, the independent demand assumption may be reasonable or not.

References

Aburto, L. and Weber, R. (2007). Improved supply chain management based on hybrid demand forecasts. *Applied Soft Computing*, 7(1):136–144.

Aggarwal, C. C. and Zhai, C. (2012). *Mining text data*. Springer Science & Business Media.

Agrawal, N. and Smith, S. A., editors (2015). *Retail Supply Chain Management: Quantitative Models and Empirical Studies*, volume 223. Springer US, 2nd edition.

Ainscough, T. L. and Aronson, J. E. (1999). An empirical investigation and comparison of neural networks and regression for scanner data analysis. *Journal of Retailing and Consumer Services*, 6(4):205–217.

Ali, Ö. G., Sayin, S., Van Woensel, T., and Fransoo, J. (2009). SKU demand forecasting in the presence of promotions. *Expert Systems with Applications*, 36(10):12340–12348.

Ali-Vehmas, A. (2016). Incorporating weather forecasts into retail sales forecasting. Master's thesis, Aalto Universtity.

Anandarajan, M., Hill, C., and Nolan, T. (2019). *Practical Text Analytics. Maximizing the Value of Text Data*. Springer, 1st edition.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. (2019). BoTorch: Programmable Bayesian Optimization in PyTorch. *arXiv preprint arXiv:1910.06403*.

Barber, D. (2012). *Bayesian Reasoning and Machine Learning*. Cambridge University Press.

Belt, T. (2017). When is forecast accuracy important in the retail industry? Effect of key product parameters. Master's thesis, Aalto Universtity.

Bertsimas, D. and Mišić, V. V. (2019). Exact first-choice product line optimization. *Operations Research*, 67(3):651–670.

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

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Chen, F. L. and Ou, T. Y. (2011). Sales forecasting system based on Gray extreme learning machine with Taguchi method in retail industry. *Expert Systems with Applications*, 38(3):1336–1345.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM.

Christou, I. T. (2012). *Quantitative Methods in Supply Chain Management*. Springer.

da Veiga, C. P., da Veiga, C. R. P., Puchalski, W., dos Santos Coelho, L., and Tortato, U. (2016). Demand forecasting based on natural computing approaches applied to the foodstuff retail segment. *Journal of Retailing and Consumer Services*, 31:174–181.

Deloitte Touche Tohmatsu Limited (2020). *Global Powers of Retailing 2020*.

Di Pillo, G., Latorre, V., Lucidi, S., and Procacci, E. (2016). An application of support vector machines to sales forecasting under promotions. *4OR*, 14(3):309–325.

Farias, V. F., Jagabathula, S., and Shah, D. (2013). A Nonparametric Approach to Modeling Choice with Limited Data. *Management Science*, 59(2):305–322.

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

Fisher, M. and Raman, A. (2018). Using Data and Big Data in Retailing. *Production and Operations Management*, 27(9):1665–1669.

Friedman, J. (2001). Greedy Function Approximation : A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232.

Friedman, J., Hastie, T., and Tibshirani, R. (2009). *The elements of statistical learning*. Springer, New York, 2 edition.

Gaur, V. and Honhon, D. (2006). Assortment planning and inventory decisions under a locational choice model. *Management Science*, 52(10):1528–1543.

Giering, M. (2008). Retail sales prediction and item recommendations using customer demographics at store level. *ACM SIGKDD Explorations Newsletter*, 10(2):84–89.

Hotelling, H. (1929). Stability in Competition. *The Economic Journal*, 39(153):41–57.

Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.

Hyndman, R. J. and Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688.

Kimball, R. and Ross, M. (2013). *The Data Warehouse Toolkit, The Definitive Guide to Dimensional Modeling*. John Wiley & Sons.

Kök, A. G. and Fisher, M. L. (2007). Demand estimation and assortment optimization under substitution: Methodology and application. *Operations Research*, 55(6):1001–1021.

Kolassa, S. (2018). Commentary on Retail Forecasting. *International Journal of Forecasting*, 34(4):830–831.

Kuo, R. J. (2001). A sales forecasting system based on fuzzy neural network with initial weights generated by genetic algorithm. *European Journal of Operational Research*, 129(3):496–517.

Lang, S., Steiner, W. J., Weber, A., and 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.

Lardinois, F., Mannes, J., and Lynley, M. (2017). Google is acquiring data science community Kaggle. `https://techcrunch.com/2017/03/07/google-is-acquiring-data-science-community-kaggle/`. Accessed on 19.10.2020.

Levy, M. and Weitz, B. A. (2012). *Retailing management*. McGraw-Hill, New York, 8th edition.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., and Lee, S.-I. (2020). From local explanations to

global understanding with explainable AI for trees. *Nature Machine Intelligence*, 2(1):2522–5839.

Lundberg, S. M. and Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc.

Manning, C., Raghavan, P., and Schütze, H. (2010). Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103.

Michis, A. A. (2015). A wavelet smoothing method to improve conditional sales forecasting. *Journal of the Operational Research Society*, 66(5):832–844.

Mukherjee, S., Shankar, D., Ghosh, A., Tathawadekar, N., Kompalli, P., Sarawagi, S., and Chaudhury, K. (2018). Armdn: Associative and recurrent mixture density networks for eretail demand forecasting. *arXiv preprint arXiv:1803.03800*.

Palmer, H. (2016). Large-Scale Assortment Optimization. Master's thesis, École Polytechnique de Montréal.

Pavlyshenko, B. (2018). Using Stacking Approaches for Machine Learning Models. In *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, pages 255–258. IEEE.

Pavlyshenko, B. M. (2019). Machine-Learning Models for Sales Time Series Forecasting. *Data*, 4(1).

Pearl, J. (2009). *Causality*. Cambridge University Press, 2nd edition.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Seaman, B. (2018). Considerations of a retail forecasting practitioner. *International Journal of Forecasting*, 34(4):822–829.

Shapley, L. S. (1953). A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317.

Strauss, A. K., Klein, R., and Steinhardt, C. (2018). A review of choice-based revenue management: Theory and methods. *European Journal of Operational Research*, 271(2):375–387.

Tsoumakas, G. (2019). A survey of machine learning techniques for food sales prediction. *Artificial Intelligence Review*, 52(1):441–447.

Van Ryzin, G. and Vulcano, G. (2015). A market discovery algorithm to estimate a general class of nonparametric choice models. *Management Science*, 61(2):281–300.

Wong, W. K. and Guo, Z. X. (2010). A hybrid intelligent model for medium-term sales forecasting in fashion retail supply chains using extreme learning machine and harmony search algorithm. *International Journal of Production Economics*, 128(2):614–624.

Zhang, X., Pei, J., and Ye, X. (2016). Demographic transformation and clustering of transactional data for sales prediction of convenience stores. *Proceedings of 2016 IEEE International Conference on Cloud Computing and Big Data Analysis*, pages 102–108.

Zheng, A. and Casari, A. (2018). *Feature engineering for machine learning: principles and techniques for data scientists*. O'Reilly Media, Inc.

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