Term Paper Challenge

August 12, 2022

Machine Learning for Economists

Supervised by:
Prof. Dr. Tom Zimmermann

Submitted by:
Lennart Bolwin
Justus Töns

1 Introduction

The goal of this challenge is to predict the sold quantity for each product, more precisely stock-keeping units (SKU) of an online retail startup for a given week. To do so, we are provided with the historic sales data of said online retailer per product, as well as details regarding the product's characteristics and details for order-day specific date information.

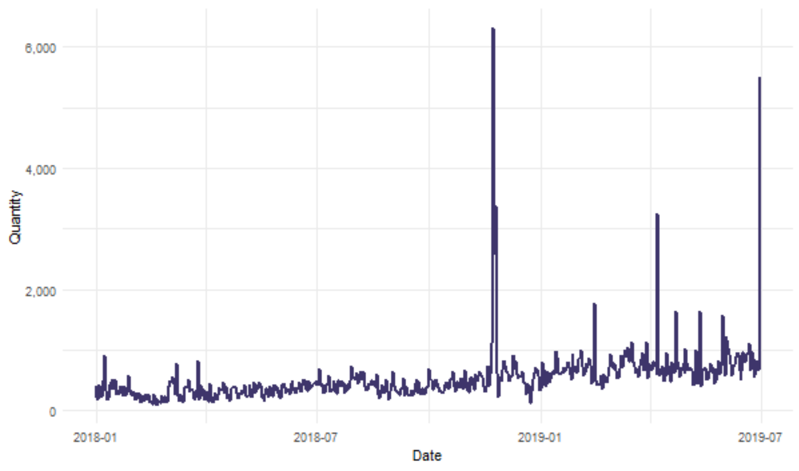
The focus in this short paper is on dataset handling, feature engineering, model selection, tuning and agnostics.

1.1 Initial Data Review and Handling

The given dataset mainly consists of three sources: A file containing the daily time series of sold quantities per SKU from 01-01-2018 until 30-06-2019, a file containing characteristics regarding the individual SKUs, such as the category, type, size and color and a file containing information about public and school holidays. Overall, the dataset consists of a total of $502,320$ rows, as the sales for 920 individual SKU have been observed for 546 days.

To obtain a first impression of the data, we plot the aggregated sold quantity in the given historic period.

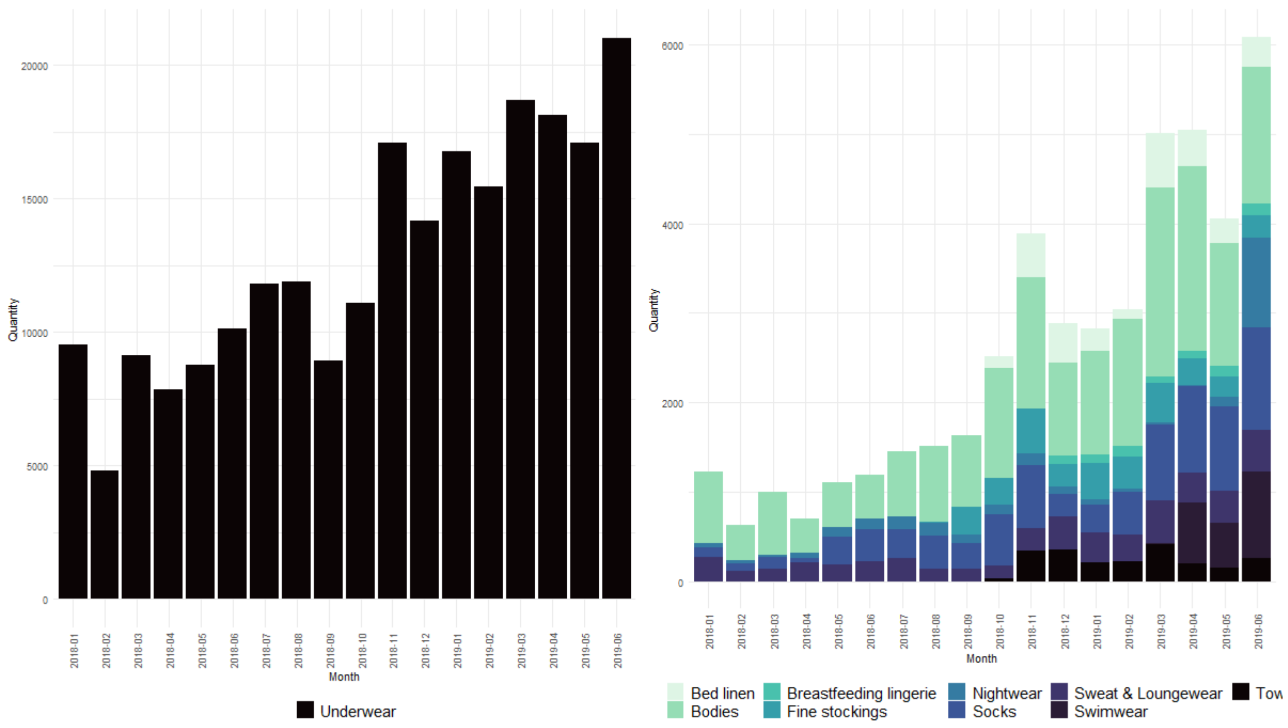
Figure 1: Timeseries: Total Items Sold



It becomes apparent that the sold quantity increases slowly but steadily over the past months and especially in 2019 remains on an overall higher level. Furthermore, there seem to be certain outliers which dominate the overall picture. For example, the most successful day in terms of products sold was the 23rd of November 2018 with more than 6,000 units sold. After a brief online research it became clear, that this was due to the *Black Friday* holiday, which is known for special sales offers, especially in the online retail market. The treatment of these outliers will be addressed later in this document.

Let's take a look at the data a little more closely and include the categories the sold-items are grouped by. By far the most important category is Underwear, which is responsible for more than 83% of the overall items sold and which dominates the timeseries. In order not to lose sight of the remaining products and types, in the following the graphs showing the sold quantities by month are separated by category.

Figure 2: Sales Per Category and Month



The monthly aggregared data confirms the previously noted positive trend and furthermore indicates, that although different in magnitude, the trend is similar for all categories. Another peculiarity to mention is that the data series on a single SKU level is quite sparse for many products. In fact, 722 of the 920 overall SKUs did not sell at all in more than 75% of the days in the historic period (i.e. quantity = 0).

1.2 Feature Engineering and Handling

Since we ran the data analysis on our local laptops, the immense amount of data caused severe computation issues. It turned out, that this curse of dimensionality was not only due to the amount of included explanatory variables, but arose mainly from the number of observations in our data frame.[1] Therefore, we strove for a strategy to reduce the rows that each statistical learning algorithm had to deal with. First of all, we noticed that 224 of the 920 SKU have not been sold at all during the entire data period. Thus, we put these observations aside and assigned a forecast of zero for the forecasting period, which is the first week of July. Besides the advantages of faster computation times, this pre-filtering strategy prevented our models from being unreasonably inert.

In the next step, we took a closer look at the outliers. Since in this case the outliers are actual sold items and not a results of measurement errors or the like, we decided to keep them in the time series, but reduce them in magnitude. To do so, we defined a given datapoint as an outlier, if the quantity sold for a single SKU on a day was larger than the mean plus three times the standard

[1]Eventually, have reduced the variables contained in the data set to 47.

deviation, conditionally an order took place for this particular SKU, i.e.

$$replace(x, x >= (mean(x[x! = 0]) + 3 * sd(x[x! = 0])), mean(x[x! = 0]) + 3 * sd(x[x! = 0]))$$

If an outlier was identified, it was replaced by the constraint itself, serving as upper truncation limit.

In order to ensure maximum use of information, we furthermore used various feature engineering techniques. First, we dissected the date itself to get potentially useful information like the week of the month, the quarter of the year, and more. Second, we computed autoregressive (AR: up to order 6) and rolling average (MA: up to order 12) features in order to account for the time series structure of the data.

Note that the autoregressive parameters, which are simply the lagged values of the dependent variables themselves, and the respective moving averages, which are the mean values of the autoregressive parameters, are perfectly co-linear for successive periods. For instance, it holds that $MA_2 = \frac{1}{2}(AR_1 + AR_2)$ and $MA_3 = \frac{1}{3}(AR_1 + AR_2 + AR_3)$. Thus, when we performed models that require the inversion of the X-matrix, columns that are a convex combinations of other columns drop out and do not influence the final forecast. This finding caused us to shift the autoregressive parameters and the rolling averages to time windows with little overlap.

We further discussed to one hot encode all characteristics of the product, such as size and color, but found that the resulting, vastly increased dataset strongly affected our computation time and we therefore discarded the idea.

Subsequently, we thought about including further external data, which might enhance the model and decided to include economic as well as weather data. As economic data we included the following monthly data from the ifo Insitute as regular and lagged (AR1) features: ifo Business Climate, ifo Business Situation, ifo Business Expectation and ifo Business Cycle. As weather data, we included the daily sunshine duration and rainfall amount in Germany. The external data was standardized before it was added to the overall dataset.

1.3 Computational Setup and Cross Validation

In a next step, we utilized the fact that the individual products were stratified into 12 categories and within the categories into 94 types (after removing the unsold SKUs, 72 types and 10 categories remained)[2]. This stratification gave rise to one of our most crucial assumptions: As we could not train models on the entire data for computational reasons, we assumed that there was no inter-product relationship with respect to category and type and trained individual model specifications for each of the 72 category-type combinations[3]. Next, we chose for each combination the model specification with the lowest validation RMSE, which provided us with 72 data-driven model recommendations. This filleting strategy does not come without cost: As each of our 72 model specifications had to deal with one specific category-type combination only, we did not allow the models to learn statistical patterns of other combinations. Yet it offered the advantage of

[2]We found one category-type-combination that consisted of a single SKU (Towels / Washing Glove) which caused errors for some of our models. For the sake of simplicity, we merged the individual SKU to the Towels Washing Cloth group

[3]In fact, our first approach was to train individual models for each of the 10 categories only. However, this procedure turned out not to be computationally feasible for the underwear-category which is responsible for roughly 60% of the data

an increased flexibility on the category-type level, since each combination is granted access to its individual and most suitable model specifications. Proceeding as described implicitly assumes that the increased flexibility outweighs the absence of inter-product relations. In a perfect world with access to unlimited GPU and RAM, one would conduct both approaches and let the validation RMSE decide which one is superior.

Speaking of the validation RMSE guides us to one of the essential steps in each machine learning journey: Performance evaluation of machine learning models is the integral part to identify both the best model among many (model assessment) and the best combination of hyperparameter values within a given model (model selection). Yet, to prevent overfitting, it is important, that performance evaluation is done for new data as it is easy to inflate a fit statistic for the training data by choosing a model with maximum flexibility. Oftentimes resampling models like cross validation (CV), the bootstrapping or the jackknife method are used to robustly estimate a performance metric like RMSE. k-fold CV, for instance, involves the random splitting of the original data into k equally sized and non-overlapping folds. Then, in each iteration, one fold serves as validation set and the algorithm is fit on the remaining $k-1$ folds. This procedure allows to estimate k performance metrics where the final CV metric is obtained by averaging the k quantities. The repeated drawing of samples ensures an optimal utilization of information and reduces the impact of individual observations on the final model which fosters the model-robustness.

For k-fold CV to work, it is important to have serially unrelated data since independent of its specific position in the data, each of the k folds serves as validation set. In the context of time series data, this serial-independence constraint is not fulfilled, and the CV-algorithm requires adjustment. Time series-CV pays attention to the data structure by training and validating the models on adjacent observations only. In our case, each model is trained on 53 weeks of observations that are sorted according to the date and is evaluated on the first month after the training period. This window of training and validation observations is shifted through the entire time span of the data. As we have 18 months of data, we have a total of 5 folds to evaluate the model performances, with the first ranging from February 2018[4] until February 2019 and the last from June 2018 until June 2019. To implement a version of time series-CV, we rely on the R-package caret and its trainControl function.

2 Used Models and Hyperparameter Tuning

We pursued the target of including models of different statistical origins, ranging from parametric to non-parametric, from simple to complex, and from regularizable to non-regularizable. Our set of employed models consist of the following models:

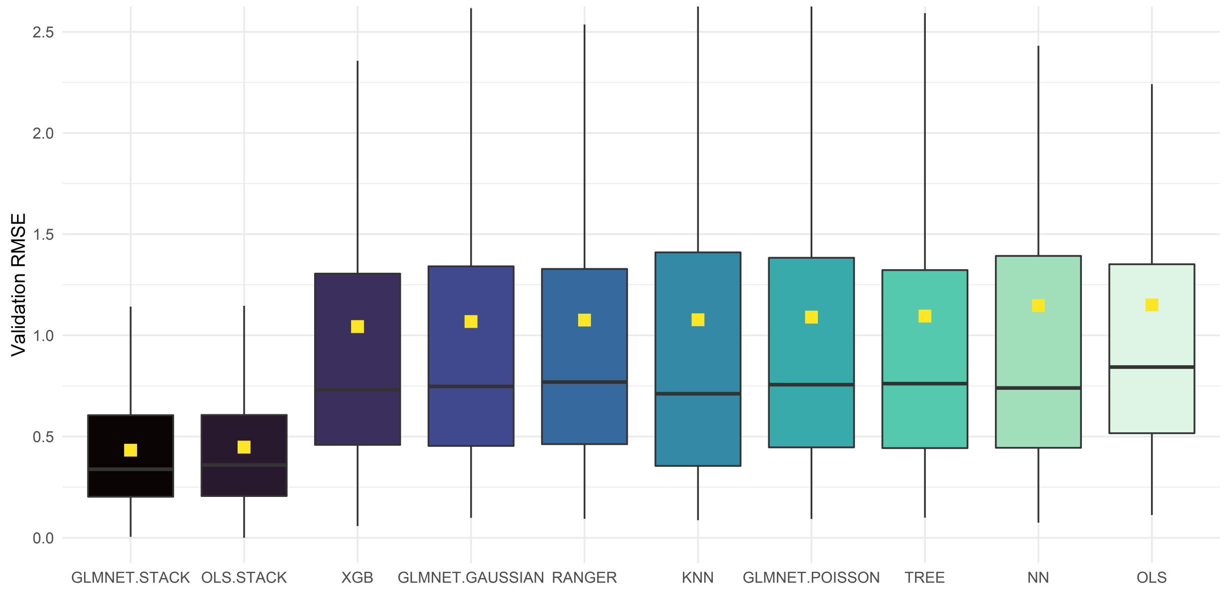
- OLS [OLS]: A simple, parametric benchmark-model with no regularization opportunities.
- Elastic Net (Gaussian) [GLMNET.GAUSSIAN]: A simple OLS model augmented by a regularization term that helps to prevent overfitting. The tuning parameters determine the share of Lasso/Ridge and how the amount of covariates negatively impacts the quantity to be minimized.

[4]We excluded the first month due to missing values in the AR and MA specifications

- Elastic Net (Poisson) [GLMNET.POISSON]: Similar to the Gaussian Net but assumes that the response variable is Poisson-distributed. This assumption seems reasonable as the sold quantity is an integer.
- K Nearest Neighbors [KNN]: A simple, non-parametric model with only one tuneable hyperparameter given by the number of neighbors to consider.
- Regression Tree [TREE]: A simple non-parametric model with virtually no regularization opportunities.
- Random Forest [RANGER]: A complex, non-parametric model that requires special attention to hyperparameter tuning to prevent overfitting.
- Extreme Gradient Boosting [XGB]: An even more complex, non-parametric model that overfits easily and thus requires careful hyperparameter tuning.
- Neural Network [NN]: Probably the most complex model with the largest amount of tuning parameters.

We tuned the models using the tuneLength-function and time series cross validation in caret. tuneLength is an alternative to specifying a tune grid manually and performs random parameter search. Depending on the complexity of the specific models, we defined tuneLength-values between 5 and 20. After all 8 models were trained for our 72 category-type combinations, we also performed two versions of model-stacking. That is, unregulized stacking by OLS as well as regulized stacking using an Gaussian elastic net. The following figure shows the validation RMSE for each of the models mentioned above. The medians and means depicted below were calculated on the basis of all models (10) for all types (72) and all cross-validation iterations (5), in total $3,600$.

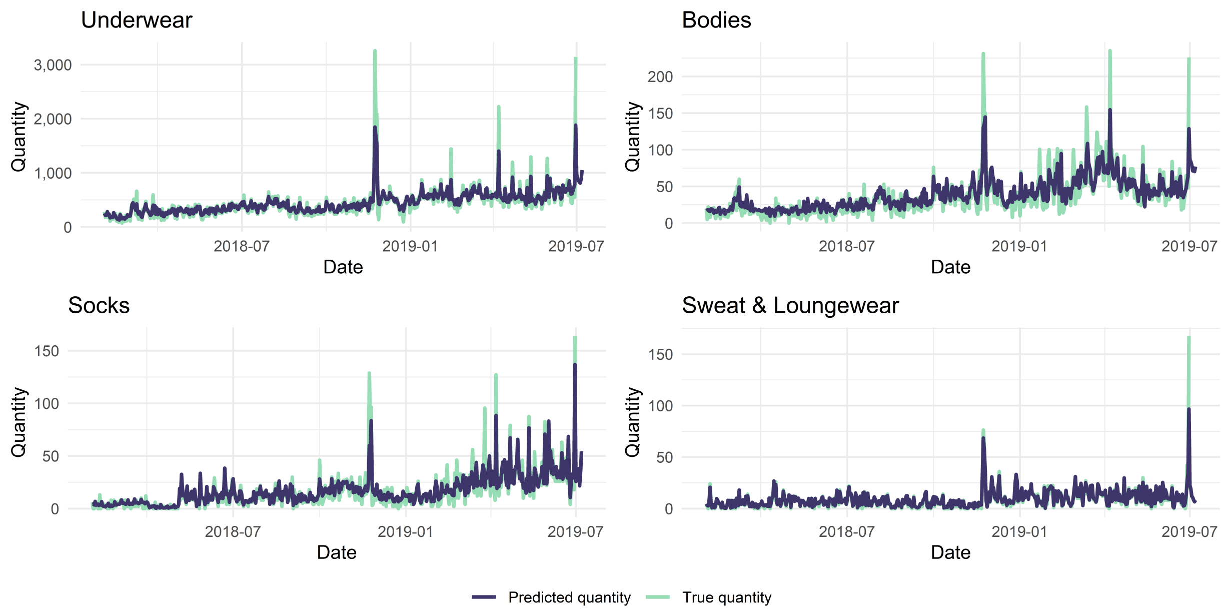
Figure 3: Validation RMSE by Model



The Boxplot indicates, that the stacked models perfom best regarding the validation RMSE (yellow square). However, we found that for some of the types we forecasted, the forecast seemed

to overshoot despite a low validation error. Therefore, we have added another validation step that determines whether or not the prediction is feasible using the model selection based on the lowest RMSE. We defined a forecast to be feasible, if the cumulated forecasted sales in the first week of July did not exceed twice the maximum of the weekly cumulated sales of the previous four weeks. If the forecast is defined as unfeasible the next best model in terms of validation error is used and feasibility is checked again, such that in the final forecast the best feasible model is used. Without this validation step, we would use for 61/72 types the elastic net stacked model, for 10/72 types the ols stacked model and for the remaining 1/72 type a neural network. After the feasibility validation step, we saw a much more heterogeneous distribution: For 34/72 types we used the elastic net stacked model, for 8/72 types the KNN Model, for 7/72 the TREE Model, for 6/72 types the NN Model, for 5/72 the Gaussian elastic net model, for 5/72 the OLS stacked model, for 4/72 the OLS model, for 2/72 the Poisson elastic net model and finally for the remaining 1/72 type the XGB model.
This model selection yields the forecasts in Figure 4 exemplary for the 4 most important (in terms of items sold) categories.

Figure 4: Forecast: Top 4 Categories

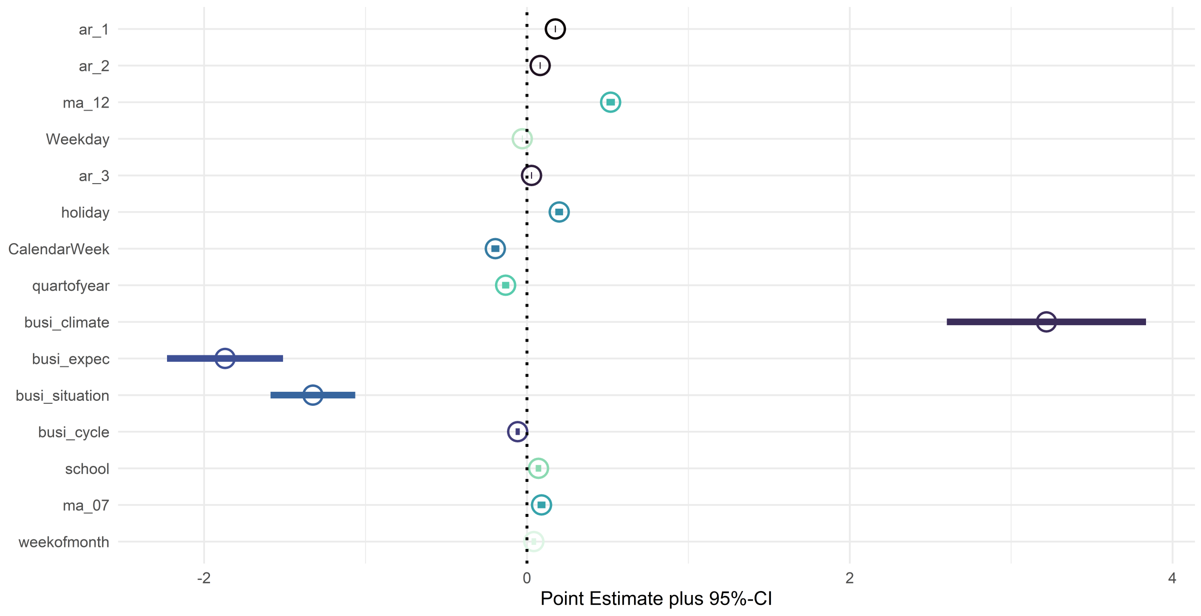


3 Model Agnostics

The interpretability of machine learning models is becoming increasingly important as ML more and more often answers impactful and far-reaching questions ranging from autonomous driving to medicine. Though it can be argued that the sales figures of an online retailer are not equally important, we want to shed some light into the used black box models. Model-Agnostic methods can be applied at both the aggregated (global) and the individual level (local). Oftentimes, these models are computationally demanding, especially if they involve the re-estimation of potential outcomes for a variety of covariate-values. For the sake of simplicity and because we employed 72 individual model-hyperparameter combinations, we pursued a global surrogate approach. That is, we trained an interpretable model (OLS) to the predictions of our black box model using the same covariates. Our surrogate model explains 51.1% of the variation in the predicted quantities – a decent statistic – especially if one takes the simplicity and the strict model assumptions of OLS into consideration. Figure 5 displays the marginal effects of our covariates for the OLS model sorted descendingly according to the absolute value of the t-statistic as well as the 95%-confidence interval.

We observe that on average, the most important covariates in order to explain the sold quantity of today is the sold quantity of yesterday and the day before yesterday proving the time structure relevance of the dataset. On an all-other-things-equal-basis, these covariates contribute with 17.6% (ar_1) and 8.2% (ar_2). Further we see, that less is sold at the beginning of the week (neg. Weekday-coefficient), more during holiday times and significantly more if the business climate as reported by the ifo institute is positive.

Figure 5: Marginal Effects of Surrogate Model


ar_1
ar_2
ma_12
Weekday
ar_3
holiday
CalendarWeek
quartofyear
busi_climate
busi_expec
busi_situation
busi_cycle
school
ma_07
weekofmonth
Point Estimate plus 95%-CI