An introduction to Time Series Forecasting using a case study of predicting sales of an Ecuadorian store chain

Nicklas Prochazka s87375@BHT-BERLIN.DE
Data Science
Fachbereich VI
Berliner Hochschule für Technik

Abstract

Time series forecasting is a vital task in many industries, as it allows for the prediction of future trends and patterns. This paper represents a concise guide to time series analysis and forecasting, including data preprocessing, feature generation, Exploratory Data Analysis and creating basic models using Exponential Smoothing and SARIMA(X). The models and their hyperparameters are evaluated using a self-built and published Python package. After a theoretical introduction, a case study using a retail store sales dataset from Kaggle is performed. By following the steps outlined in this paper, people concerned with such problems can effectively analyze time series data and make informed decisions for their respective business.

Keywords: Time Series, Forecasting, EDA, Parallel Hyperparameter Optimization, CV, Store Sales, Exponential Smoothing, SARIMAX, Kaggle, Favorita

1. Introduction

Time series (TS) analysis and forecasting is a powerful analytical tool that can provide valuable insights into the future performance of businesses. For store owners, the ability to accurately predict sales trends and anticipate market changes is essential for maintaining profitability and competitiveness. However, the process of time series forecasting can be complex and challenging, requiring expertise in data analysis, statistical modeling, and machine learning.

In this paper, the basics of TS analysis and forecasting are explained and showcased using a detailed case study of predicting store sales data from the Favorita retail store chain in Ecuador. The analysis encompasses a range of techniques, including data preprocessing, feature generation, Exploratory Data Analysis (EDA), parallel Hyperparameter Optimization using Cross-validation (CV; using a self-written and published Python package), and model selection using popular approaches such as Exponential Smoothing (ExpSm) and SARIMA(X) (trying both univariate and multivariate approaches). These are then used to forecast approximately two weeks.

By following this methodology and applying the principles and practices of TS analysis and forecasting, store owners can gain deeper insights into their sales data and make informed decisions about inventory, pricing, and marketing strategies. This paper serves as an introductory, yet comprehensive, guide to the tools and techniques of TS analysis and forecasting and offers practical advice on how to apply them to real-world business problems.

2. Time Series Analysis and Forecasting: Theory and Methodology

TS analysis is a prelude to TS forecasting. It uses statistical techniques to evaluate how time-dependent data changes over time. As such, it involves the analysis of data points that are taken at (normally) regular intervals over time, and the aim is to understand the underlying patterns, trends, and relationships in the data. With this understanding, an appropriate model is generated to predict/forecast future data points. This can be done using a univariate (i.e. only using past values of a single series) or a multivariate (i.e. using auxiliary data, namely explanatory variables in the form of additional time series) approach (Chatfield, 2000).

The following steps provide a general framework for conducting a time series analysis and forecast:

1. **Data Preparation**
   The first step in TS analysis is to collect and prepare the data. This involves checking for data quality, removing any missing values, and ensuring that the data is in the correct format. As with any ML task, it is also important to split the data into training (, validation) and test sets to evaluate the performance of the models. However, in TS analysis there exists an additional caveat. Since TS data is inherently sequential, the data cannot be randomly split into training and test set as it could lead to data leakage (i.e. using the past to predict the future). Instead, the data must split in a chronological order, with the training set consisting of the earlier observations and the test set consisting of the more recent observations. The same goes when using Cross-validation (CV). Here, a technique called *rolling window* CV is used, where the model is trained on a fixed window of the data and tested on the next window, which is then repeated on the following windows, essentially splitting the data into several non-overlapping buckets.

2. **Exploratory Data Analysis (EDA)**
   EDA is an important step in TS analysis as it provides insights into the underlying patterns and trends in the data. It includes (Hyndman and Athanasopoulos, 2018):

   - Visualizing the data using several TS plots.
   - Analyzing the correlation between a TS and itself if lagged by a certain time period. This is done inspecting the autocorrelation function (ACF) and the partial autocorrelation function (PACF). The former measures the correlation between a TS and all of its lagged values, while the latter measures the correlation between a TS and its lagged values, while controlling for the effects of all shorter lags (i.e. using only the lags up to k as explanatory variables).
   - Decomposing the TS to identify trends, seasonality, and other patterns.
   - Testing for stationarity, i.e. checking if statistical properties such as mean, variance, and autocorrelation structure remain constant over time. One popular test is the Augmented Dickey-Fuller (ADF), taking a slightly different form depending on constant and trend (or their absence). The H_0 says: *TS does possess a unit root*, i.e. is not stationary. It estimates a lag length, which can be done computationally by minimizing the AIC until the last lag is statistically significant (e.g. $p < 0.05$). If the TS is not stationary, techniques/methods such as differencing, ExpSm or SARIMA can be used to achieve (weak) stationarity.

3. **Feature Generation**
   This step involves creating new features from the original TS data or adding explanatory variables to it. These features can include time-specific (one-hot) encoded features, such as *day of the week* and *month of the year*, or lagged values of variables as well as external data such as economic factors. This step is not always necessary, mainly for enriching multivariate methods, sometimes it is helpful for visualization (e.g. for showing distribution throughout the week/year etc.)[1].

1. This means it can sometimes be useful to do (parts of) this step before EDA.

4. **Model Selection and Fitting**
There are many different models suited for TS forecasting which can be divided into parametric (e.g. ExpSm, AR(I)MA and similar ones) and non-parametric models (e.g. Neural Networks, Regression). Many can be used either in a univariate or multivariate manner (Gautam and Singh, 2020). Two will be used and thus introduced here:

   - **Exponential Smoothing** ExpSm is a popular method/model that is used to smooth as well as predict future values of a TS. It is a simple, yet effective, method that involves averaging the previous values of the TS to generate predictions for the future. While a simple Moving Average (MA) weighs all past observations equally, ExpSm decreases their weight the further they lie in the past. There are several types, namely Simple, Double, and Triple Exponential Smoothing (also known as Holt-Winters). The first one helps deal with noise (see Equation 1), the second one additionally helps deal with a trend in the data (see Equation 2) and the third one additionally helps deal with seasonality (see Equation 3). This is done by introducing the smoothing parameters α, β and γ respectively. The forecast uses all variables estimated previously (see Equation 4) (Hyndman and Athanasopoulos, 2018).

$$l_t = \alpha y_t + (1-\alpha)\hat{y}_{t-1} \tag{1}$$
$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{2}$$
$$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m} \tag{3}$$
$$\hat{y}_{t+h} = l_t + hb_t + s_{t+h-m} \tag{4}$$

   where:

y	= observed value
$\hat{y}$	= predicted value
t	= an index denoting a time period
l	= level/noise component
b	= trend factor
s	= seasonal index
m	= seasonal period
h	= forecast at m periods ahead
h	= number of steps to forecast ahead
α, β, γ	= smoothing parameters to be estimated

   - **SARIMA(X)** Seasonal Autoregressive Integrated Moving Average models are used for forecasting. It is a univariate method (like ExpSm) that includes the same components as ARIMA models, namely autoregression (AR), differencing (I), and Moving Average (MA), but also incorporates an additional seasonal component (S).

     SARIMA models use, additionally to the three regular parameters (lowercase) used in ARIMA models, their seasonal parts (uppercase; with seasonal periods accounted for):

$$\underbrace{(p,d,q)}_{\substack{\uparrow \\ \text{Non-seasonal part} \\ \text{of the model}}} \quad \underbrace{(P,D,Q)_s}_{\substack{\uparrow \\ \text{Seasonal part} \\ \text{of the model}}} \tag{5}$$

where:

p/P = number of autoregressive terms
d/D = order of differencing
q/Q = number of moving average terms
s = seasonal periods (e.g. *7* for weekly data)

Adding these to the ARIMA formula yields (with the same notation using lowercase symbols for the non-seasonal and uppercase symbols for the seasonal part):

$$\phi_p(B)\Phi_P(B^s)\Delta^d\Delta_s^D y_t = \theta_q(B)\Theta_Q(B^s)\epsilon_t \tag{6}$$

where:

ϕ = autoregressive operator
B = backshift operator (lag)
∇ = differencing operator
θ = moving average operator
t = time
y = time series (values)
ϵ = (white noise) error term

The (hyper-)parameters in Equation 5 have to be estimated using ACF and PACF (plots) and/or Grid Search over all (feasible) combinations.

SARIMAX (SARIMA with Exogenous Variables) is a multivariate method that can include other predictors/covariates in addition to the time series itself (which can improve the forecasting accuracy). It adds a vector of exogenous variables x to Equation 6.

5. **Hyperparameter Optimization**
Hyperparameter Optimization is, as with almost all ML tasks, an important step in TS forecasting. TS models often have a lot of hyperparameters which can be tuned (in contrast to model parameters, which are normally determined by the model during the training process). This is done either by inspection using mathematical/graphical tools or more often by creating a combination of all possible/a range of hyperparameters and running the training process with these, evaluating them against a validation set. Hyperparameters often try to deal with (if necessary) noise, trend and seasonality, i.e. help prevent over- or underfitting (Meisenbacher et al., 2022).

6. **Model Evaluation**
A similar process used for evaluating hyperparameters is used when gauging overall model performance and comparing different models amongst each other. This is done all throughout the process of developing ML models. First, a baseline should be established or found, against

which more sophisticated models can be compared. Since the target value in TS analysis/-forecasting is continuous, metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) or Root Mean Square Log Error (RMSLE) are used. The MAE gives no particular weight to far-off predictions, the RMSE does that (by squaring them), the RMSLE is based on the percentage error rather than the absolute one the other two use (Saxena, 2019). The model is trained using training data and evaluated against held-out validation data (potentially using CV).

7. **Forecasting**
   The final step is to do the actual forecasting, i.e. predict future data. This can either be a single-step or multi-step prediction. If it is done against a test set, the result should be evaluated as shown above.

3. Analysis and forecast of store sales: implementation

3.1 Introduction to the data and their features

The provided data are daily sales of the individual stores, split by product family, of Favorita, an Ecuadorian chain of general stores (among other business areas), selling e.g. products of daily use and automobile parts. They span a time of about 4.5 years, from 2013 to mid-2017, with about 3 million data points. Other auxiliary data are the count of articles on promotion, holidays and the oil price. Further features can be engineered from the dates, i.e. day of the week, day of the month, day of the year, month etc. The goal is to forecast the sales for 15 days at a time.

3.2 Exploratory Data Analysis

After loading the data, several plots were used to visualize trends, patterns/seasonality, peculiarities, i.e. summarize their main characteristics.

Figure 1 shows the average sales broken down into different time features. What is immediately noticeable is that sales are particularly high on weekends and in December. This might hint a certain seasonality.

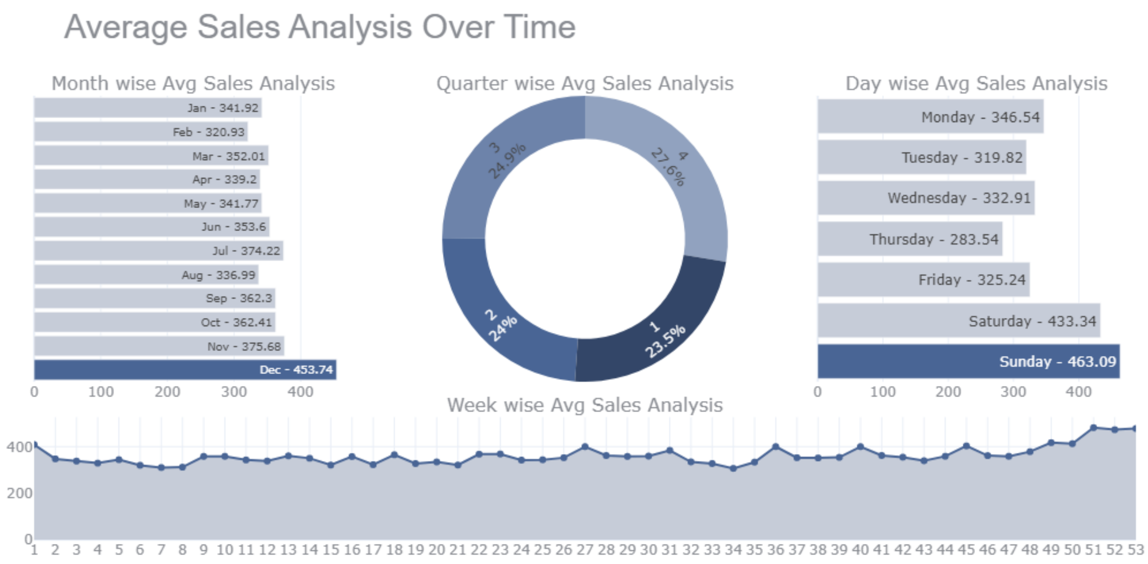


Figure 1: Quarterly, monthly, weekly and daily analysis of aggregated store *sales*

The seasonality plot of the Seasonal and Trend decomposition using Loess (STL) in Figure 2 and the ACF plot of Figure 3 confirm this. A clear weekly pattern can be recognized, as well as a weaker monthly and yearly pattern (showing more clearly in Figure 8 and Figure 7). The correct/most important seasonality to choose thus seems to be a *weekly* one. This also manifests itself in the residual plots of Figure 2 and Figure 7, with the weekly one exhibiting a more uniform pattern and generally smaller values (i.e. explaining the underlying patterns better). There is also a slight upward trend over time, meaning the average number of sales increases.

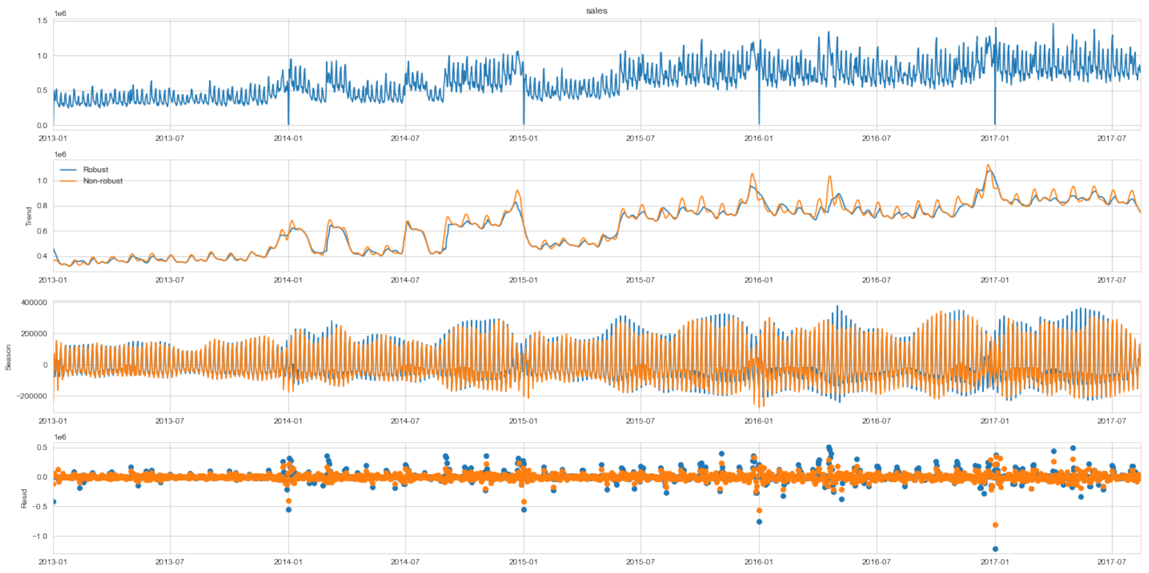


Figure 2: STL (decomposition) of *sales* (*period=7*)

The data is highly correlated with itself, as seen in the ACF and PACF plots (see Figure 3) with the highest spikes at (multiples of) lags 1, 6, 7 and 8 in the ACF plot. It stays statistically relevant until about 200 data points (days) in. When controlling for intermediate lags, spikes are highest at lags 1, 3, 6 and 7 with negative correlation at lags 8 and 9. This also suggests a weekly pattern.

These observations also get confirmed by a p-value of 0.09 in the ADF, meaning the TS is not completely stationary (H_0: *TS has unit root* cannot be rejected). This non-stationarity should be dealt with sufficiently by the two chosen prediction models, though. The remaining residuals are mostly white noise, with some heavier outliers around the first day of the year (see last plot in Figure 2 and first plot in Figure 3).

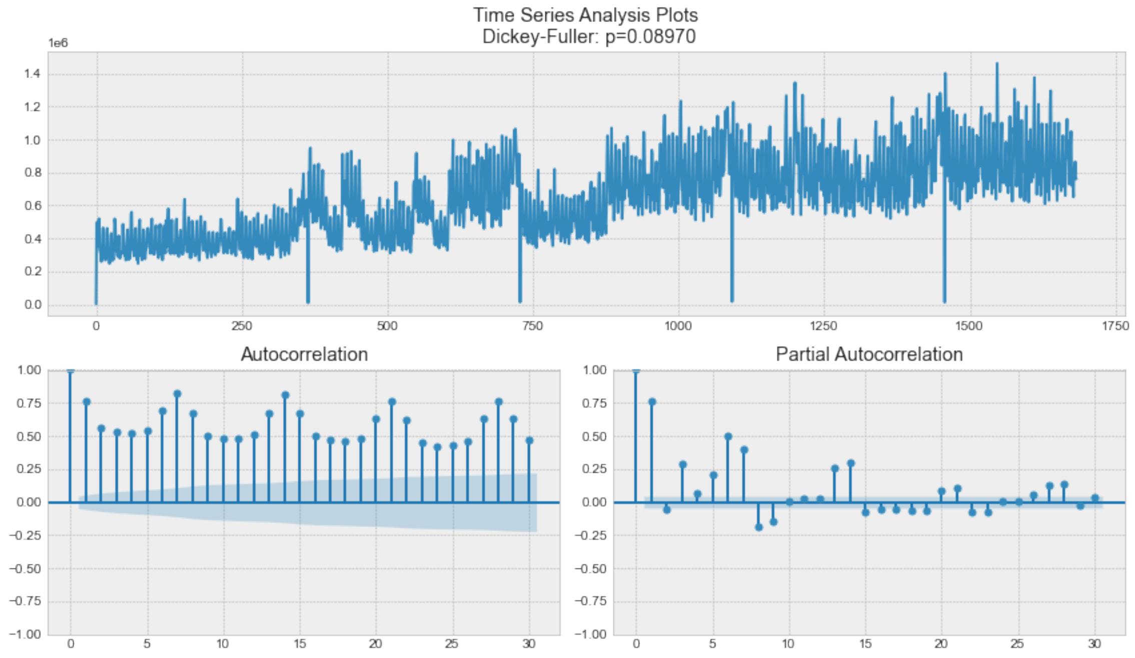


Figure 3: Results of an ADF test and ACF/PACF plots of *sales*

The correlation between sales of different stores is strong (except for a few outliers; overall=0.8047; see Figure 10), between sales of different product families moderate/low (overall=0.4241; see Figure 11), between sales and how many articles are on promotion moderate/strong (overall=0.5749), between sales and oil price moderately negative (-0.5008).

3.3 Model Selection, Hyperparameter Optimization and Evaluation

ExpSm and SARIMA(X) models were chosen due to their simplicity and explainability, which fit the intent of this paper to be a concise introduction to TS analysis/forecasting. Following this line, all sales across all stores and product families will be analyzed/modelled together. This would be a top-down approach (assuming results were to be disaggregated according to the proportions of individual product families/stores in relation to the whole sales; also not done here). For a more comprehensive and precise analysis and forecast, all product families could be analyzed separately (due to having a rather low correlation), maybe even all stores additionally (not as necessary due to higher correlation), and then aggregated. This would be a bottom-up approach (Soto-Ferrari et al., 2019). This is often not feasible for larger data sets (in this case 33 product families and 54 stores would lead to 1782 separate models).

Hyperparameter Optimization is done using (parallelized) Grid Search with CV.

Different hyperparameter combinations and models are compared and evaluated using the MAE, RMSE and RMSLE.

3.3.1 Exponential Smoothing

The first method chosen is a smoothing method which can also be used for forecasting (see Figure 4). As a result of the data having a clear trend and seasonality (as shown in subsection 3.2), it is apparent that Triple ExpSm (Holt-Winters) must be used. The package used is *statsmodels*.

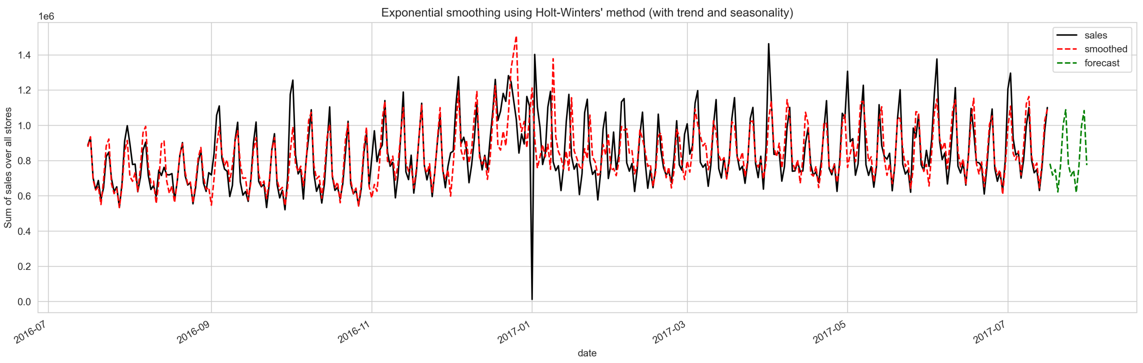


Figure 4: ExpSm used to smooth the data and forecast a 15 period window

Available hyperparameters to tune mainly deal with trend, seasonality, bias and variance.[2] The parameters and their optimal values are:

- *trend*: *additive*,
- *seasonal*: *multiplicative*,
- *damped_trend*: *True*,
- *seasonal_periods*: *7* (manually set because known beforehand),
- *use_boxcox*: *True*,
- *remove_bias*: *True*.

The model with these parameters yields an average RMSE of around <u>108.0k</u> (CV).

3.3.2 SARIMA(X)

First, the univariate version is used. Available hyperparameters to tune are explained in Equation 5, some can already be gauged looking at the ACF and PACF (see Figure 3), q/Q and p/P should be 1 because they have the first lag at 1. The parameters and their optimal values are:

- *p/P*: *1/1*,
- *d/D*: *1/1*,
- *q/Q*: *1/1*,
- *s*: *7* (manually set because known beforehand),

The model with these parameters yields an average RMSE of around <u>97.7k</u> (CV), i.e. lower than the ExpSm model above (which is to be expected given the more sophisticated nature).

For the multivariate version, additional (exogenous) features are needed. These can either be other TS with continuous values for the same time frame (like count of items on promotion) or one-hot encoded features, either extracted from the *date* (like *day_of_the_week*, *is_weekend* or similar) or

2. Further explained at: https://www.statsmodels.org/dev/generated/statsmodels.tsa.holtwinters.ExponentialSmoothing.html.

providing external information (like *holidays*). Those get fed into the model (one model uses all and one only those not extracted from the *date*).

The former model yields an average RMSE of around 88.9k (CV), i.e. lower than the univariate SARIMA model, which, again, is to be expected given the extra predictive power of additional features.

4. Results and discussion

In total, four models (one ExpSm, one SARIMA, two SARIMAX) were trained and their hyperparameters optimized. The results of their predictions can be seen in Figure 5. The metrics measured against the held out test data can be seen in Table 1. While the SARIMA without exogenous features has a lower MAE, the RMSE and RMSLE of the SARIMAX is better. The test set does have a certain peculiarity, namely the unusually low sales on the second weekend. The reason for this seems to be the vicinity to a national holiday (being shortly before or after a holiday was only sometimes accounted for). Therefore, the metrics observed with CV might be more indicative of actual model performance. There, the difference in performance between models was even bigger, which does suggest SARIMAX is the best model.

Table 1: Error metrics of all models' predictions compared against the test set.

Model	MAE	RMSE	RMSLE
Exponential Smoothing	92287.5	125555.8	0.1386
SARIMA	84461.0	116867.5	0.1302
SARIMAX	89452.3	110509.4	0.1205
SARIMAX (no time features)	87837.6	117312.6	0.1321

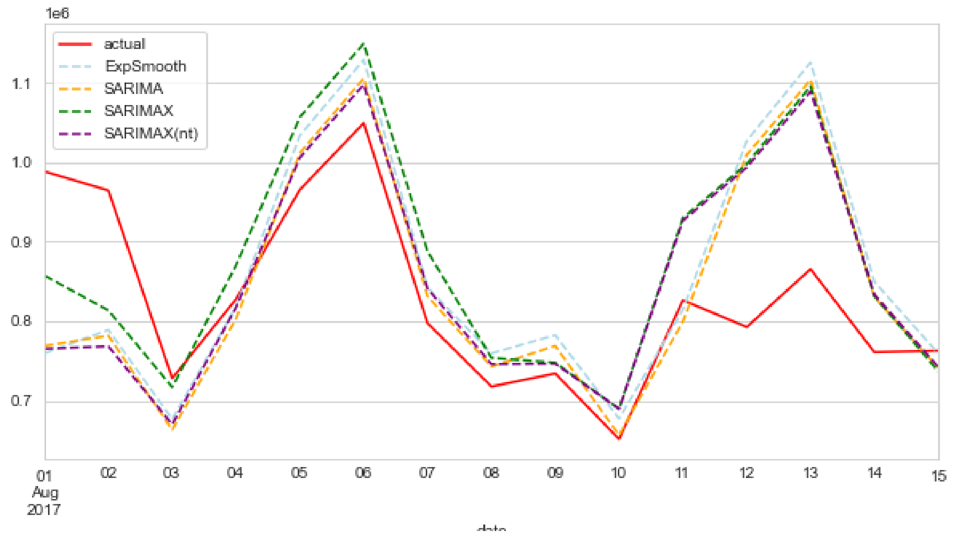


Figure 5: Predictions (15-step forecast) of all models compared against the test set.

As mentioned before, a basic approach in aggregating all sales and predicting them together, without disaggregating afterwards, using basic methods, was chosen. This allows to get a general understanding and continue from there.

As best performance was not the focal point, all models are lacking predictive power in comparison to the top results on Kaggle. Possible extensions to them could be: using a bottom-up approach, collecting more (potentially) meaningful exogenous features, using more sophisticated methods like Prophet or (LSTM) Neural Networks.

5. Creating a Python package for parallel Hyperparameter Optimization using CV for TS

It often makes sense to use parallel programming (i.e. leveraging multiple processors/cores) when optimizing hyperparameters since it is computationally intensive, especially with CV. There exist libraries to do parallel optimization like Ray Tune[3], though they are not easily adaptable to all models and none were found to be suited (well) for optimizing hyperparameters of TS models, particularly for ExpSm and SARIMA(X). Furthermore, parallelization in Jupyter Notebooks is often cumbersome due to most parallelization packages requiring the function run in parallel to be imported from outside the Notebook.

All of this meant it was easier to construct something own. This led to the creation of a small convenience module which takes the data and a combination of all specified hyperparameters (among others) as input and then splits the data using CV (with regard to the special constraints for TS) and evaluates each split, using the RMSE. Where useful, this process can be repeated for each split to deliver more accurate results (e.g. in the case of ExpSm, SARIMA(X) contains no stochastic components/is deterministic). The output consists of the (sorted) results for all combinations of hyperparameters tested. So far, ExpSm and SARIMA(X) are supported, but the module can be easily extended to use other methods.

It can also be used for model selection by providing a more comprehensive approach/more accurate metrics due to CV (and multiple simulations) than just using a single validation set.

The package looks as follows:

```python
def optimize_hyperparams(hyperparams: list, data, func: str, n_steps=1, n_splits=10,
    runs_per_split=1):
    tscv = TimeSeriesSplit(n_splits = n_splits, test_size=n_steps)

    rmse_split = list()
    for train_index, test_index in tscv.split(data):
        cv_train, cv_test = data.iloc[train_index], data.iloc[test_index]
        rmse = list()
        for i in range(0, runs_per_split):
            try:
                preds = globals()[func](hyperparams, cv_train, n_steps, cv_test)
            except:
                print(hyperparams)
                traceback.print_exc()
                return

            true_values = cv_test.iloc[:,0].values
            rmse.append(sqrt(mean_squared_error(true_values, preds)))

        rmse_split.append(np.mean(rmse))

    rmse_all = dict()
    rmse_all[str(hyperparams)] = round(np.mean(rmse_split), 2)
    return rmse_all
```

3. https://docs.ray.io/en/latest/tune/index.html

It is used as follows:

```python
from ts_hyperparam_opt import parallel_hyperparameter_optimization as pho

if __name__ == '__main__':
    freeze_support()
    results_exp_smoothing = pho.sort_results(process_map(partial(pho.
    optimize_hyperparams, data=df_sales_train, func="exp_smoothing", n_steps=15,
    runs_per_split=10), params_exp_smoothing))
```

Producing output looking like this:

```
[{"{'trend': 'add', 'damped_trend': True, 'seasonal': 'mul', 'periods': 7, '
    use_boxcox': True, 'remove_bias': True}": 108003.45},
 {"{'trend': None, 'damped_trend': False, 'seasonal': 'mul', 'periods': 7, '
    use_boxcox': True, 'remove_bias': True}": 108611.14},
 {"{'trend': 'mul', 'damped_trend': True, 'seasonal': 'mul', 'periods': 7, '
    use_boxcox': True, 'remove_bias': False}": 108993.4}, [...] ]
```

The package has been published on GitHub (https://github.com/nick2202/ts-hyperparam-opt) and PyPI (https://pypi.org/project/ts-hyperparam-opt/) and can be installed using

```
$ pip install ts-hyperparam-opt
```

6. Conclusion

TS analysis and forecasting is a crucial task in many industries, including retail, finance, and health-care The goal of this paper was to give an introduction to and cover all the basic steps of TS analysis and forecasting and to apply it to a practical example. This included data preparation, feature generation, EDA and modeling and predicting using ExpSm and SARIMA(X). We also discussed the importance of Hyperparameter Optimization, Grid Search and CV. To help with the last tasks, a Python package was developed which can help deal with these common problems. The owner of a (small) store should be better able to predict sales for upcoming periods using the information and employing the techniques presented here.

References

Chris Chatfield. *Time-series forecasting*. CRC press, 2000.

Anjali Gautam and Vrijendra Singh. Parametric versus non-parametric time series forecasting methods: A review. *Journal of Engineering Science and Technology Review*, 13:165–171, 06 2020. doi: 10.25103/jestr.133.18.

Robin John Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Australia, 2nd edition, 2018.

Stefan Meisenbacher, Marian Turowski, Kaleb Phipps, Martin Rätz, Dirk Mueller, V. Hagenmeyer, and Ralf Mikut. *Review of automated time series forecasting pipelines*. 2022.

Sharoon Saxena. What's the difference between rmse and rmsle?, 2019. `https://medium.com/analytics-vidhya/root-mean-square-log-error-rmse-vs-rmlse-935c6cc1802a`, Last accessed on 25.03.2023.

Milton Soto-Ferrari, Odette Chams-Anturi, Juan Pablo Escorcia Caballero, Namra Hussain, and Muhammad Khan. *Evaluation of Bottom-Up and Top-Down Strategies for Aggregated Forecasts: State Space Models and ARIMA Applications*, pages 413–427. 09 2019. doi: 10.1007/978-3-030-31140-7_26.

Appendix

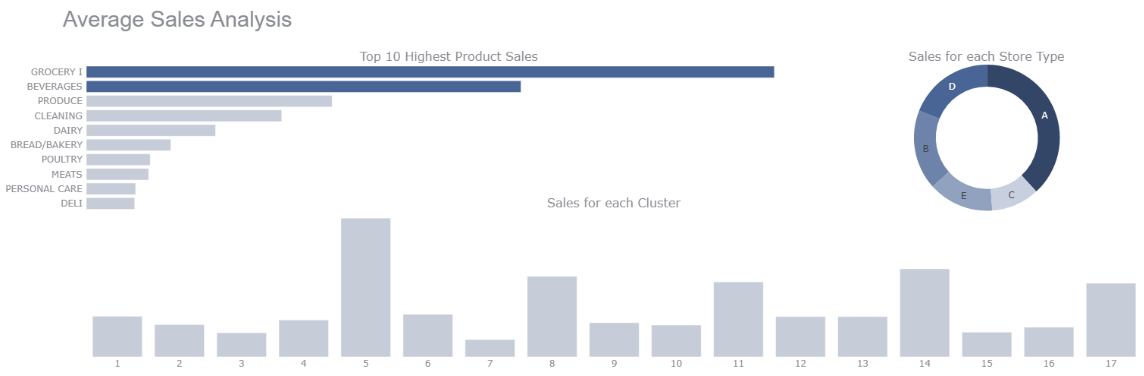


Figure 6: Analysis of average sales for product families, store types and clusters.

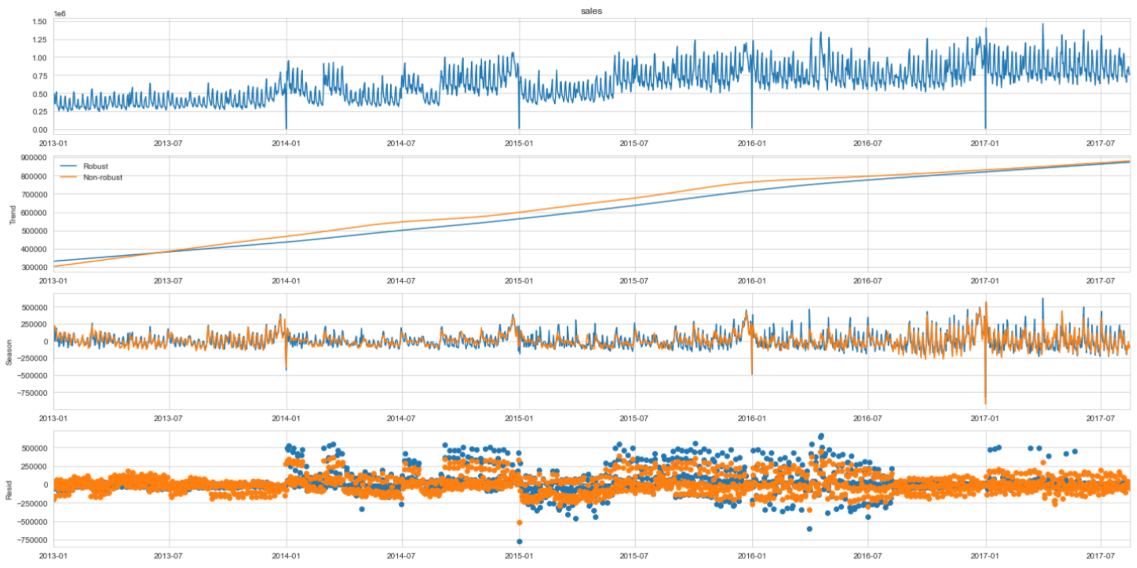
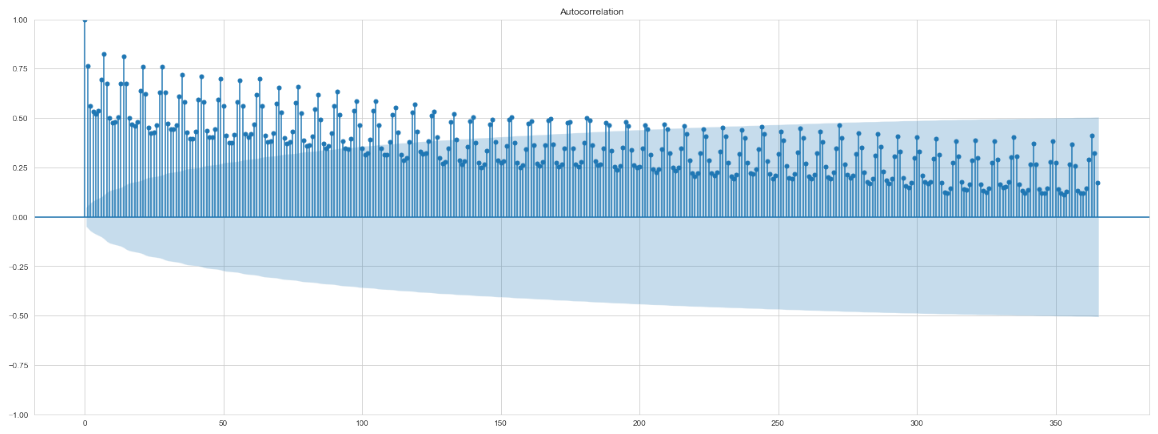


Figure 7: STL decomposition of sales (*period=365*).



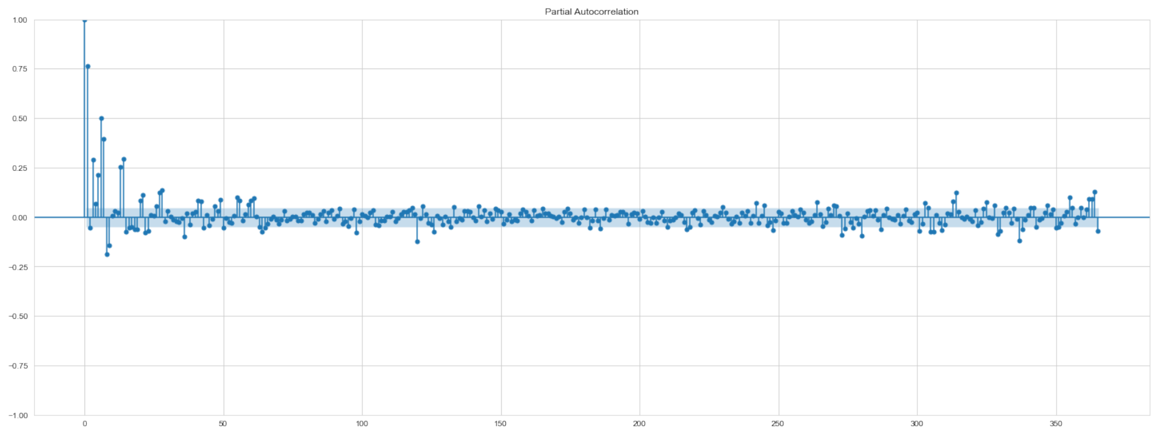


Figure 8: ACF and PACF for *lag=365*.

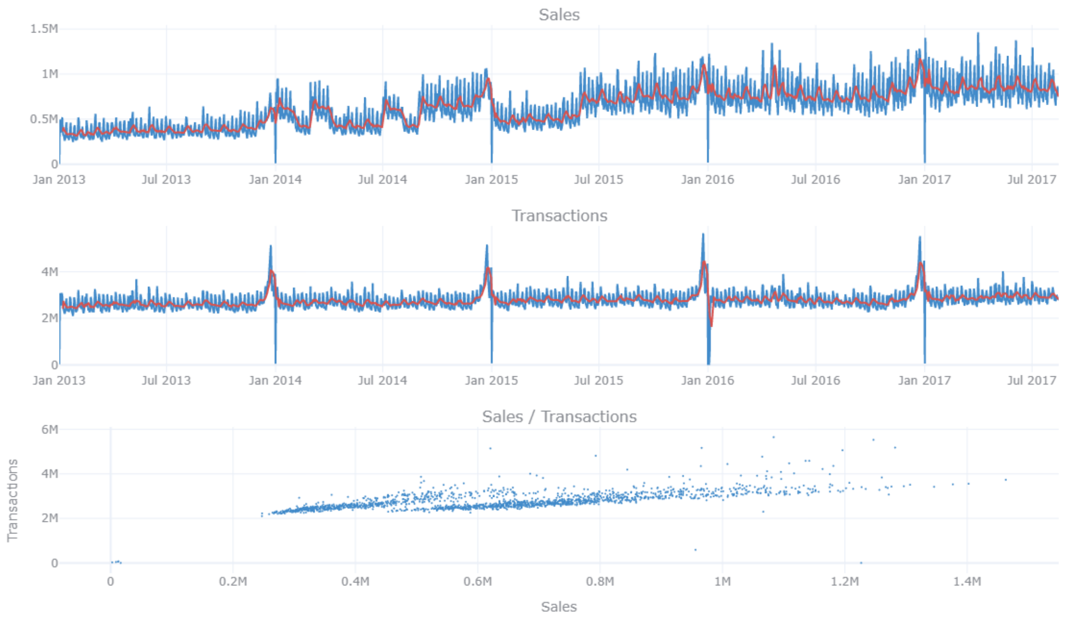


Figure 9: Comparison of sales and transactions.

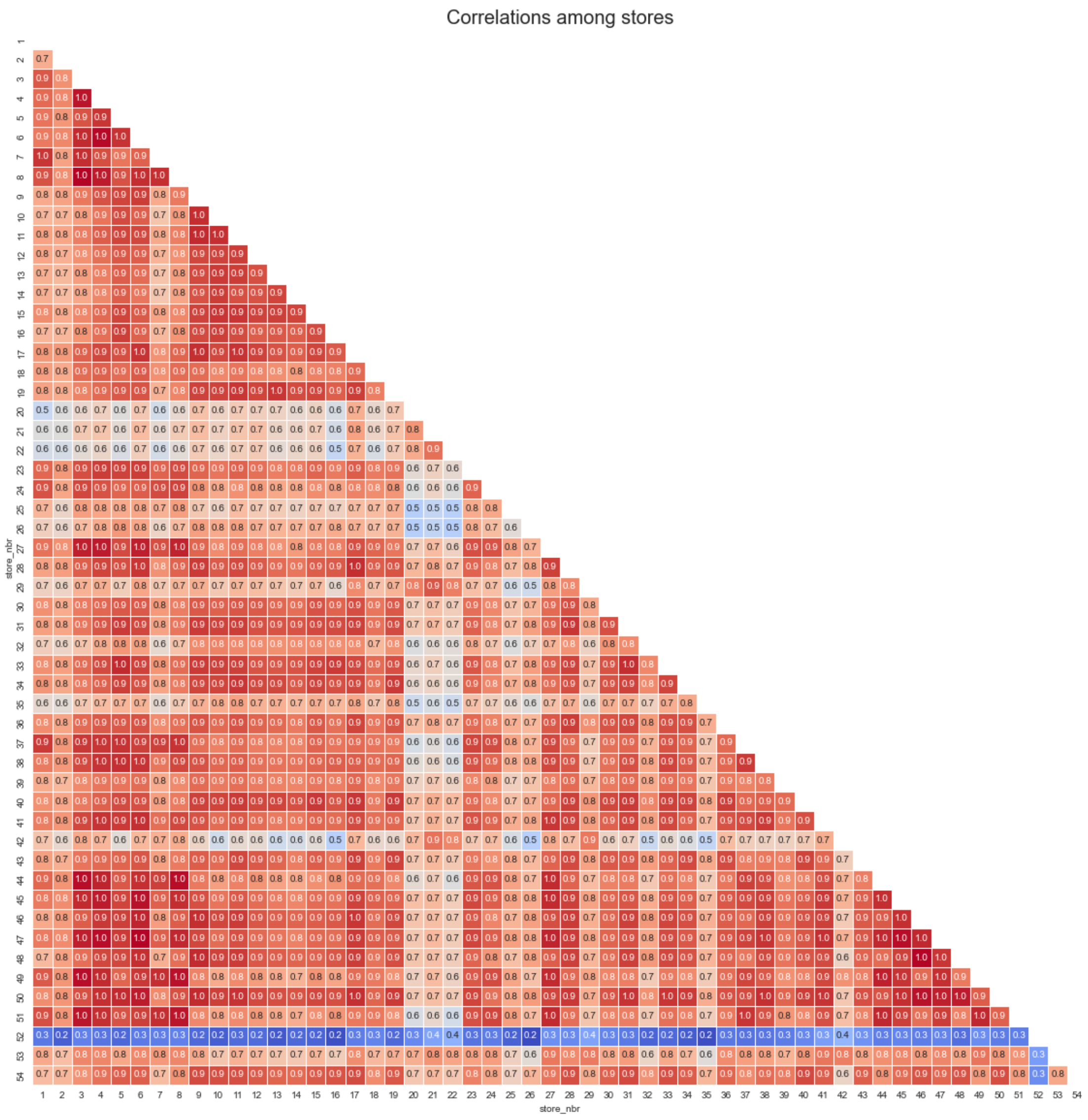


Figure 10: Correlation between sales of different stores.

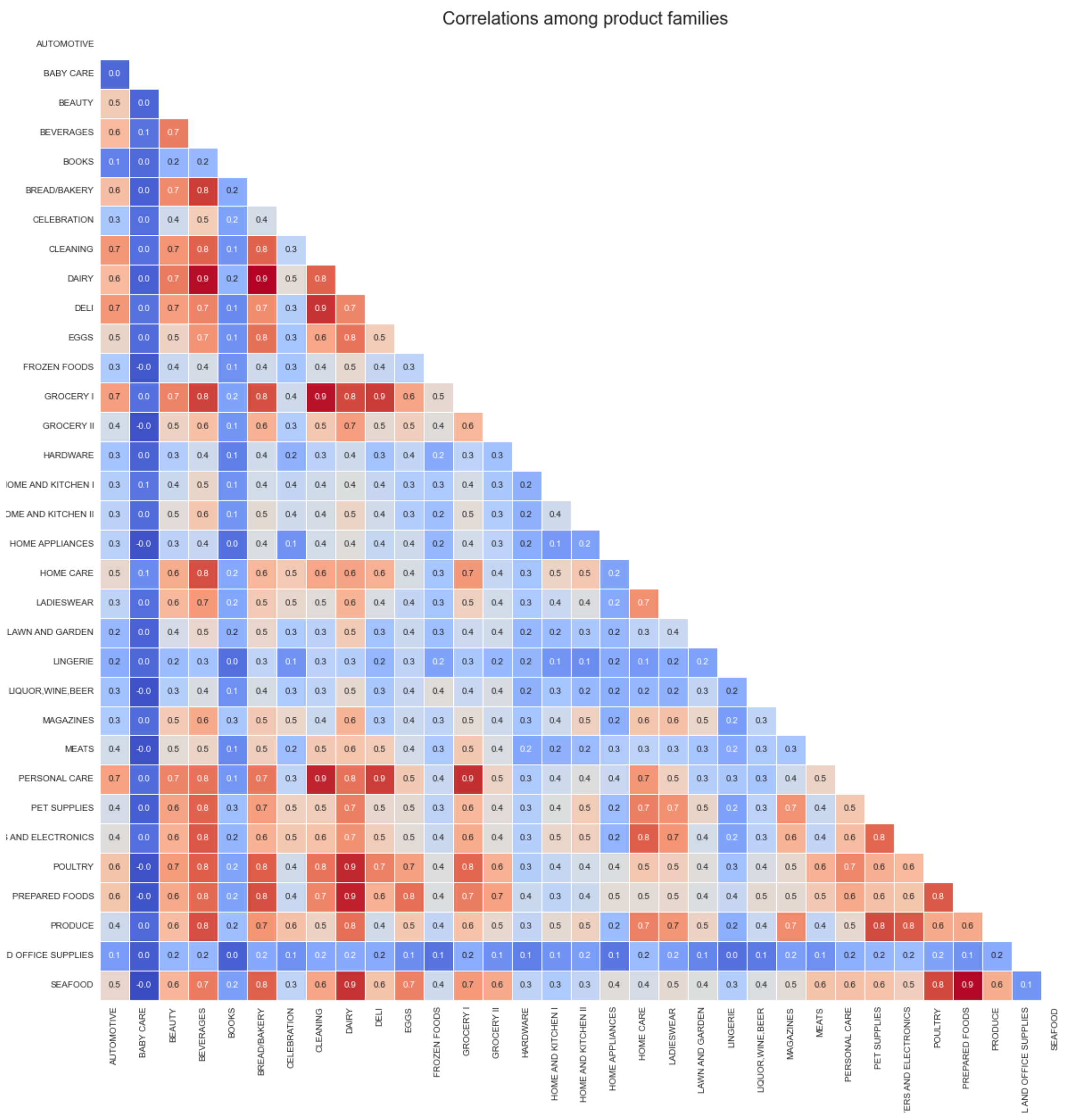


Figure 11: Correlation between sales of different product families.