Paper SAS5260-2016

A Multistage Modeling Strategy for Demand Forecasting

Pu Wang, Alex Chien, and Yue Li, SAS Institute Inc.

ABSTRACT

Although rapid development of information technologies in the past decade has provided forecasters with both huge amounts of data and massive computing capabilities, these advancements do not necessarily translate into better forecasts. Different industries and products have unique demand patterns. There is not yet a one-size-fits-all forecasting model or technique. A good forecasting model must be tailored to the data in order to capture the salient features and satisfy the business needs. This paper presents a multistage modeling strategy for demand forecasting. The proposed strategy, which has no restrictions on the forecasting techniques or models, provides a general framework for building a forecasting system in multiple stages.

INTRODUCTION

Demand forecasting has played a vital role in the decision-making processes of a variety of businesses and industries. As computing techniques continue their rapid progress, more advanced forecasting models are being developed. Nevertheless, demand forecasting remains a challenging issue, because of the natural difference and variations in data across different industries and products. For example, in the consumer packaged goods (CPG) industry, demand data at the store-SKU level are usually sparse and noisy, making it difficult to extract price and promotional effects. Under these circumstances, applying traditional time series models can be inappropriate and inefficient. This paper proposes a general modeling framework that is flexible in its underlying forecasting techniques and can therefore help tackle these issues.

The paper walks you through a case study step by step to illustrate the proposed method. The data that are used in the paper include four years of monthly product demand data at the SKU level from a large online retailer. The retailer sells more than 300 products with various demand patterns, such as different seasonal patterns, lengths of history, and demand profiles. The most recent 12 months of demand history are used as the holdout period, which is excluded from the model calibration and selection process. The mean absolute percentage error (MAPE) of the test data is computed as the error measure to evaluate the performance of each forecasting model.

Figure 1 shows some plots of demand series; each plot represents a distinct demand pattern. The upper left plot shows a product that is in demand all year round but with no obvious seasonal patterns. This product is referred to as a year-round basic product. In the upper right plot, the demand occurs all year round, along with some seasonal patterns. A product with this type of demand pattern is called a year-round seasonal product. In the lower right plot, the demand occurs only at a particular time of year. To distinguish it from the year-round seasonal product, it is referred to as a highly seasonal product. The lower left plot contains only four demand occurrences in the product's history, indicating that it is a recently introduced product. Without further information, you cannot identify its key features, such as year-round or seasonal demand. This product is called a short product, because the demand series contains a very limited history. To sum up, the demand profiles at the store-SKU level contain a variety of different features. The profiles can be sparse and noisy and can lack information. It is challenging to develop good forecasting models for this store.

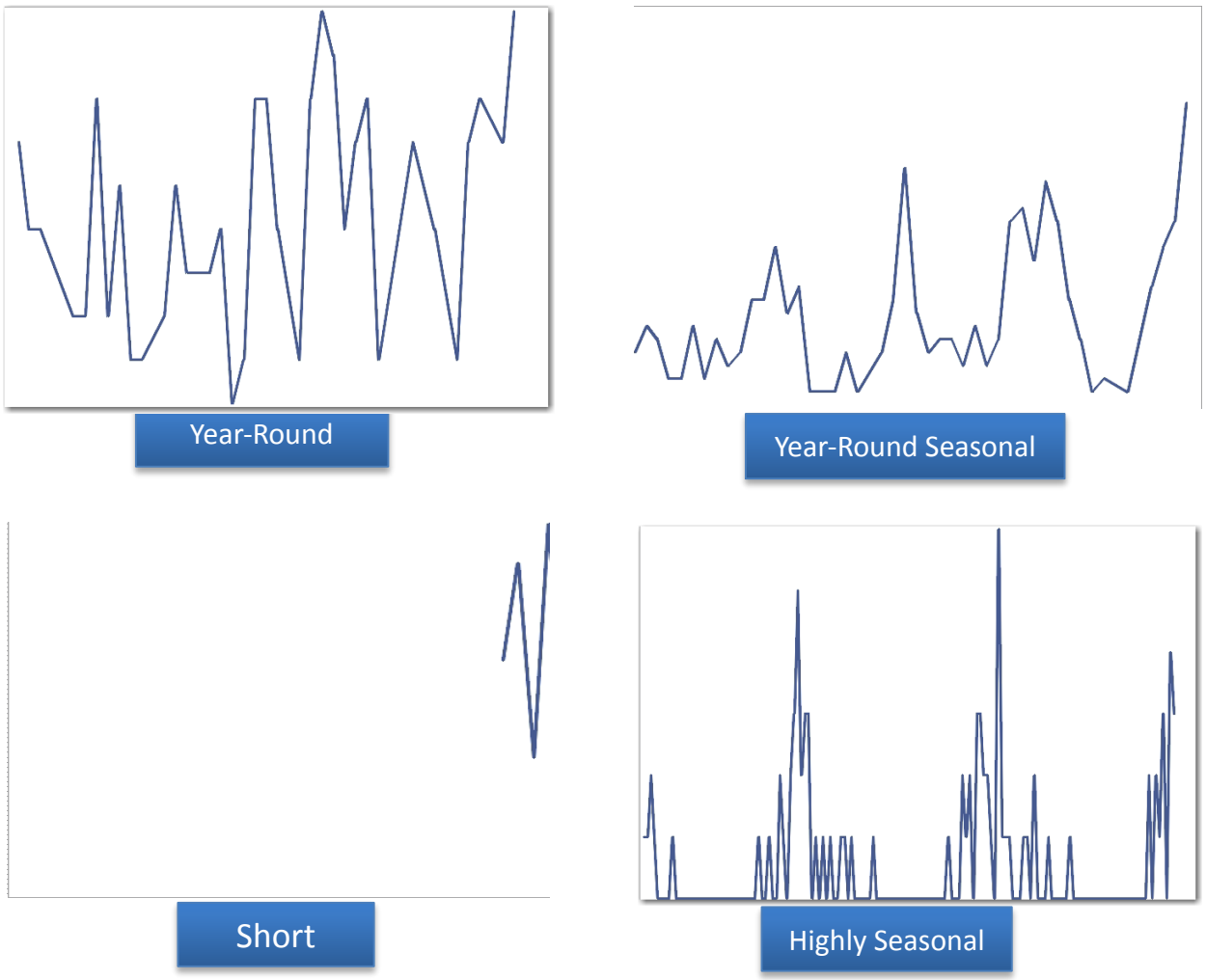


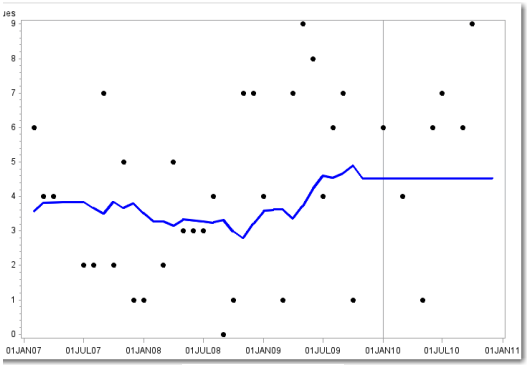
Figure 1. Sample Features of Demand Series

AUTOMATIC TIME SERIES FORECASTING

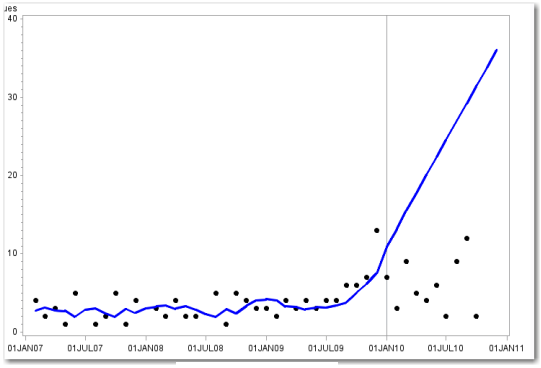
Many traditional techniques are available for generating statistical forecasts for time series, including smoothing models, ARIMA/ARIMAX models, intermittent models, and so on. Procedures have also been developed for automatically testing and selecting time series models that can then be used to generate statistical forecasts for numerous time series. This paper first applies an automatic forecasting procedure to generate forecasts. Because price is usually an important factor that can affect demand, it is used as an X variable in the forecasting model.

FORECAST RESULTS

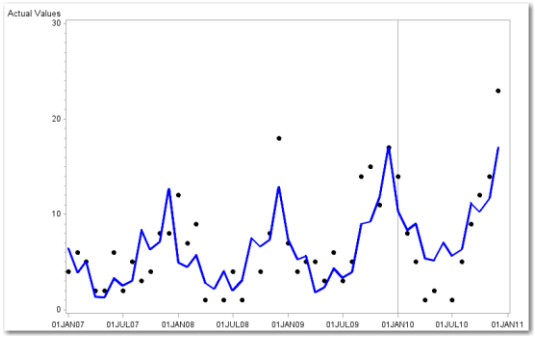
Figure 2 depicts the automatic forecasts for selected series, where black dots are the actual demand and blue lines are the forecasts. The vertical line indicates the start date of the holdout period. This section goes through each plot to check how the automatic forecast performs for each series.



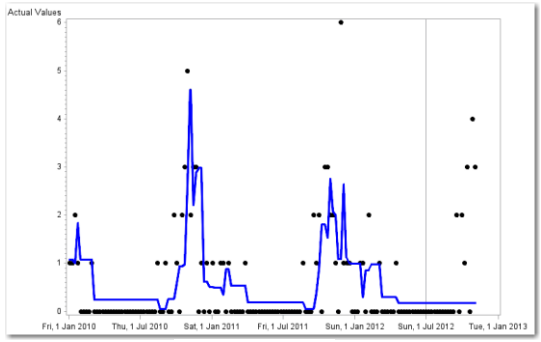
(a)



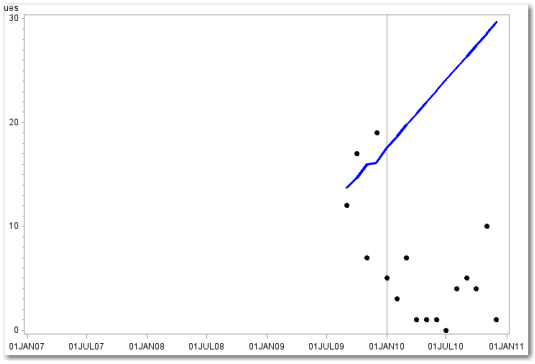
(b)



(c)



(d)



(e)

Black – Actual
Blue – Automatic forecast

Figure 2. Automatic Forecasts for Selected Series

Figure 2(a) and Figure 2(b) plot the automatic forecasts for year-round basic series. The forecast in Figure 2(a) appears reasonable, where the model is the average. However, the model in Figure 2(b) is overforecasting the demand during the holdout period. It is very likely that the system will select a smoothing model that is quite sensitive to the most recent records.

Figure 2(c) plots the automatic forecasts for year-round seasonal series. The model does a good job of capturing the seasonality of the series. It is also able to pick up the upward trend indicated by the actual values. However, it misses the off-season period when the demand is low during the holdout period.

Figure 2(d) plots the automatic forecast for highly seasonal series. The model is able to capture the historical in-season period. However, it does not accurately forecast the zero demand for the off-season period. Furthermore, for the holdout period, it cannot properly forecast future demand when the product is back in season.

Figure 2(e) plots the automatic forecasts for short series. You can observe patterns similar to those in Figure 2(b). The model is too sensitive to the most recent records. It completely overforecasts the values during the holdout period.

TOP-DOWN DISAGGREGATION

Recall that the challenges in demand forecasting include data sparsity, data noisiness, lack of information, and various patterns across multiple series. To improve the forecasting model, you can start with the simple problems: data sparsity and data noisiness. Data aggregation is a straightforward method that solves the sparsity issue and reduces noise. The following steps outline the proposed method, which is illustrated in Figure 3:

1. Apply an automatic forecast and generate a forecast for each series at a low level.
2. With the data arranged in a certain hierarchy, aggregate the data to a higher level of the hierarchy.
3. Apply the automatic forecast again to the aggregated data, and generate a forecast for each aggregated series.
4. Disaggregate the forecast at the aggregated level down to the low level to get the final forecast. The forecast that is generated in step 1 is used as the disaggregation factor.

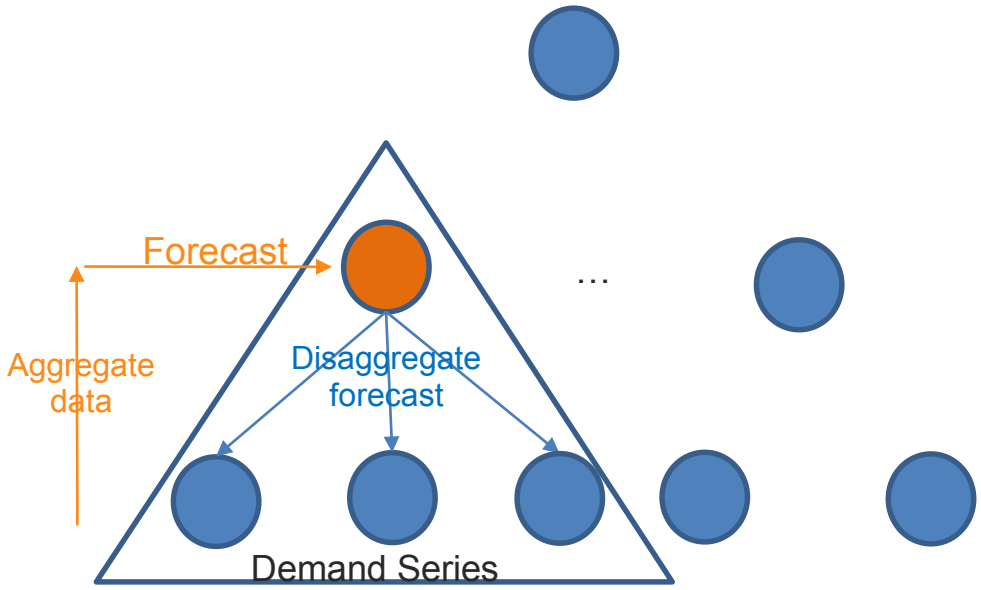
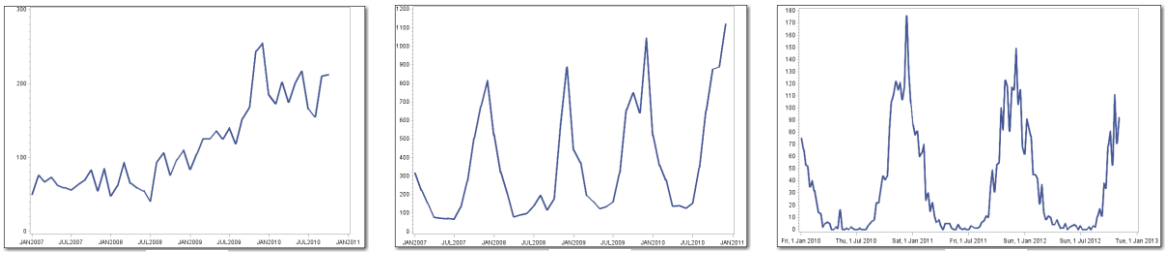


Figure 3. Top-Down Disaggregate Forecast

For the case study, the forecast results for step 1 are already available from the previous section. The next step is to perform data aggregation. Because the data used in the case study have a nice feature—all products in the same category have the same demand pattern—the data are aggregated by category. Figure 4 shows the plots of aggregated series for each category.

Figure 4(a) plots the aggregated series of year-round basic series. Figure 4(b) plots the aggregated series series of year-round seasonal series. Figure 4(c) plots the aggregated series of highly seasonal series. The short series shown in Figure 2(e) actually belongs to the same category as the year-round seasonal series. In other words, it is aggregated as part of the series in Figure 4(b). Hence, you do not see a separate aggregation series for short series. It is easy to identify the good features of the aggregated series: higher volumes, less volatility, stronger signals, and seasonality pattern if any. Therefore, the aggregated series should be easier to forecast than the individual series.



(a) (b) (c)

Figure 4. Aggregated Series

Figure 5 shows the hierarchy-based forecast versus the automatic forecast for each series, where the actual series is black, the automatic forecast is blue, and the hierarchy-based forecast is red. The vertical line indicates the start date of the holdout period. You can observe that the disaggregate forecast improves the forecast for the holdout period for almost all series. Both models' MAPEs for historical fit and holdout forecast are listed in Table 1.

TYPE	AUTO_MAPE	DISAGG_MAPE
FIT	144.53%	60.78%
HOLDOUT	94.91%	79.36%

Table 1. MAPE of Disaggregate Forecasts versus Automatic Forecasts

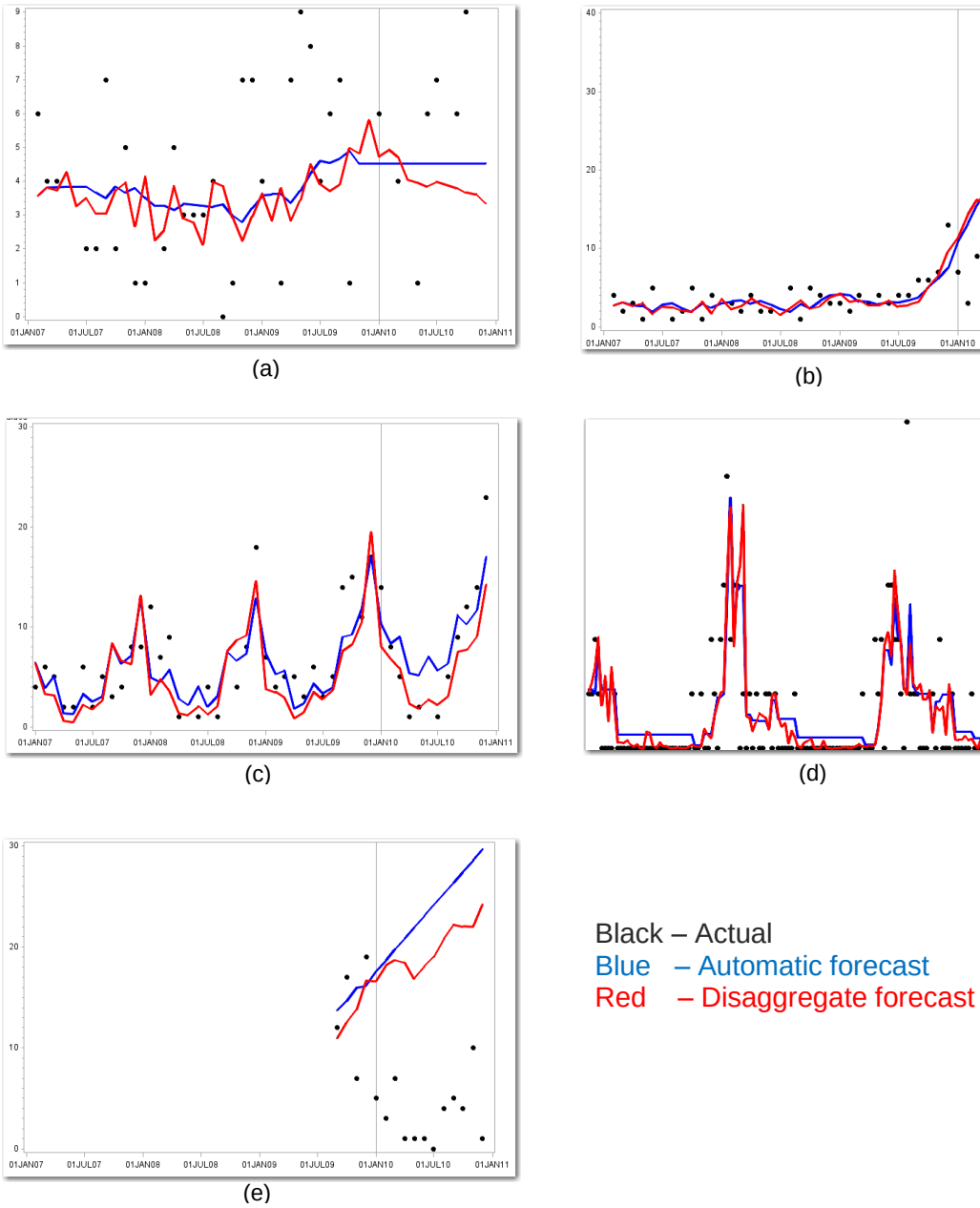


Figure 5. Disaggregate Forecasts versus Automatic Forecasts

In the case study, it is lucky that all products in the same category have similar demand patterns and profiles. In real-world analysis, it is not uncommon that series do not have existing attributions or hierarchies that best fit your modeling purpose. Wang and Li (2015) propose an algorithm to automatically classify and cluster time series based on salient features of the series. You can use the results to create a hierarchy that perfectly fits your modeling purpose.

The disaggregate forecast is able to drag down the overforecast in the holdout period for both year-round basic series (Figure 5b) and short series (Figure 5e), although the forecast is still much higher than the actual demand. For year-round seasonal series (Figure 5c), the disaggregate forecast performs well for the off-season period in the holdout period. However, it underforecasts the last few observations. For the highly seasonal series (Figure 5d), the disaggregate forecast gets the off-season period much closer to

zero. Nevertheless, during the holdout period, it still fails to forecast the demand when the product is back in season.

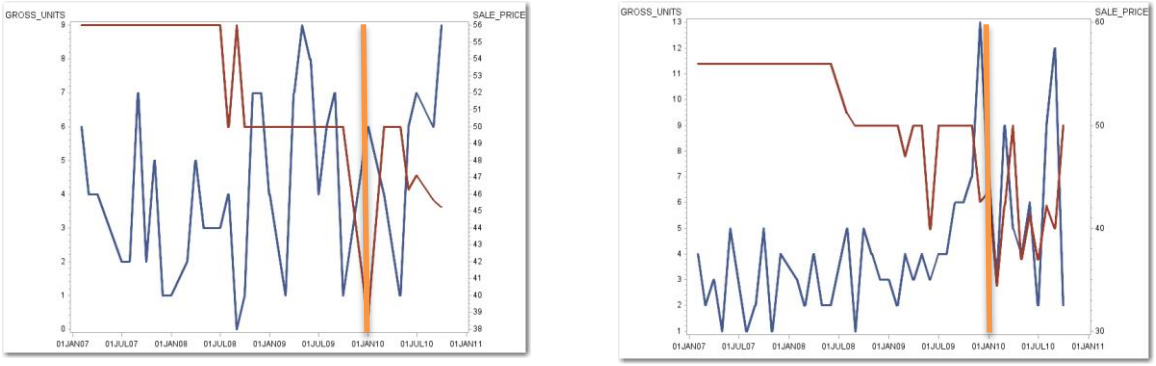
MULTISTAGE MODELING FRAMEWORK

Aggregating data helps you eliminate data sparsity and noisiness when you are forecasting at the aggregated level. Nevertheless, with top-down disaggregation, accuracy of the final forecast is still limited by the low-level forecasts. Additional work needs to be done to improve low-level forecasts. Recall the challenges in building forecasting models: data sparsity, data noisiness, lack of information, and different patterns across multiple series. The first two challenges have been addressed in the previous section. The next sections focus on the last two challenges.

IMPROVING LOW-LEVEL FORECASTS

To solve the problem of lack of information, a common practice is to borrow information from similar series. You can do this by building models that can pool information across multiple series. As for the problem of different patterns, you can solve this by using custom forecasting. Instead of arbitrarily throwing all series into the automatic-diagnose-and-forecast procedure, you can identify the pattern of each series and find the best model for each type of pattern.

As mentioned earlier, price is used as an X variable while you are building the model. However, price does not seem to affect the final forecasts as expected in all previous models. Figure 6 plots the demand series (blue) overlaid with price (red) for two products, where the orange vertical line indicates the start of the holdout period. Just by checking each individual product, you can see that the demand occurs randomly. First, there are very few price changes. Second, a drop in price does not necessarily lead to an increase in demand. In other words, there is insufficient information to estimate the correlation between demand and price, if there is a correlation.



(a) (b)

Figure 6. Demand Overlaid with Price

For the data used in this case study, the products that belong to the same category have similar demand patterns. By pooling information across the entire category, you can collect more data on price changes, as well as data on how demand responds to those price changes. The correlation between demand and price across all year-round basic products within the same category is –0.5318. This is a positive indicator of the significance of price effects. Therefore, pooling information seems to be a good way to estimate price elasticity or to estimate the effects of any dependent variables in general.

A lot of SAS/STAT® modeling procedures can do information pooling, through a BY statement: for example, the REG, GLM, and GLMSELECT procedures for linear regression; the MIXED procedure for linear regression with random effects; and the GENMOD procedure for generalized linear models. Sample code for information pooling follows:

```
proc glmselect data = glm_data;
data _null_;
   class product_id store_id category month;
   model sales = trend product_id store_id price month month*price;
```

```
    by category;
    output out = glm_out;
run;
```

This code calls PROC GLMSELECT to estimate price elasticity for each month, with information pooled across multiple series within a category. A similar idea is applied in the case study to build models and generate forecasts at a low level.

However, one challenge still remains for forecasting highly seasonal series. This type of forecasting does not help accurately forecast zero demand during the off-season period. It does not help forecast future demand when the product is back in season. To resolve this issue, you can adopt the concept of "custom intervals." The idea is to pull out the off-season period and connect all in-season periods together. The new series is very similar to a series for a year-round seasonal product. Thus, you can use regular forecasting techniques. After generating the forecast for the new series, plug in the off-season periods to generate the final forecasts. Figure 7 shows the forecasts for highly seasonal series in which you use custom intervals. It accurately forecasts zeros for the off-season period. Moreover, it is able to forecast the in-season demand for future periods instead of forecasting a flat line (Figure 2d).

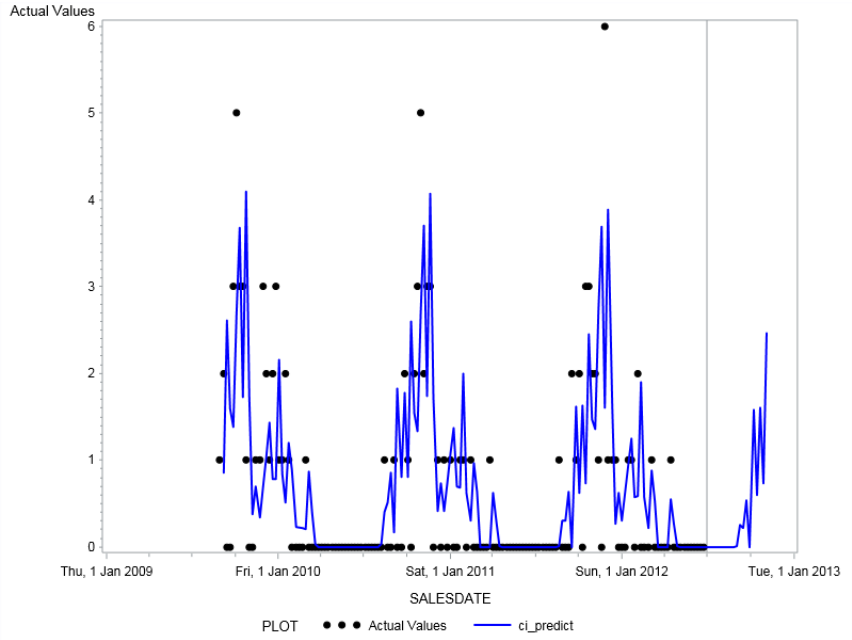


Figure 7. Forecasts for Highly Seasonal Series Using Custom Intervals

TOP-DOWN DISAGGREGATION

Because low-level forecasts are being improved, you can apply top-down disaggregation to generate the final forecasts. This concludes the multistage modeling strategy. The comparisons between multistage forecasts and disaggregate forecasts are shown in Figure 8.

The pooling method provides a better estimate of price elasticity, which greatly improves the forecasts for year-round basic series because it no longer overforecasts for future periods; see Figure 8(b). This method also enhances the forecasts for year-round basic series, especially for the last few records during the holdout period, where it used to underforecast; see Figure 8(c). Now check the short series in Figure 8(e), where magic happens. No seasonal pattern is shown in the limited history record. However, multistage models generate forecasts with seasonal patterns for the future period. Recall that the short series belongs to the same category as the year-round seasonal series. Therefore, it borrows the seasonal information from other series.

The forecasts for highly seasonal series are shown in Figure 8(d). Because the low-level forecasts are improved by using custom intervals, it is not surprising to see that multistage modeling generates forecasts that are much more reasonable than the disaggregate forecasts.

Table 2 presents MAPEs of the forecasts from each forecasting model, showing continuous improvement in forecasting accuracy.

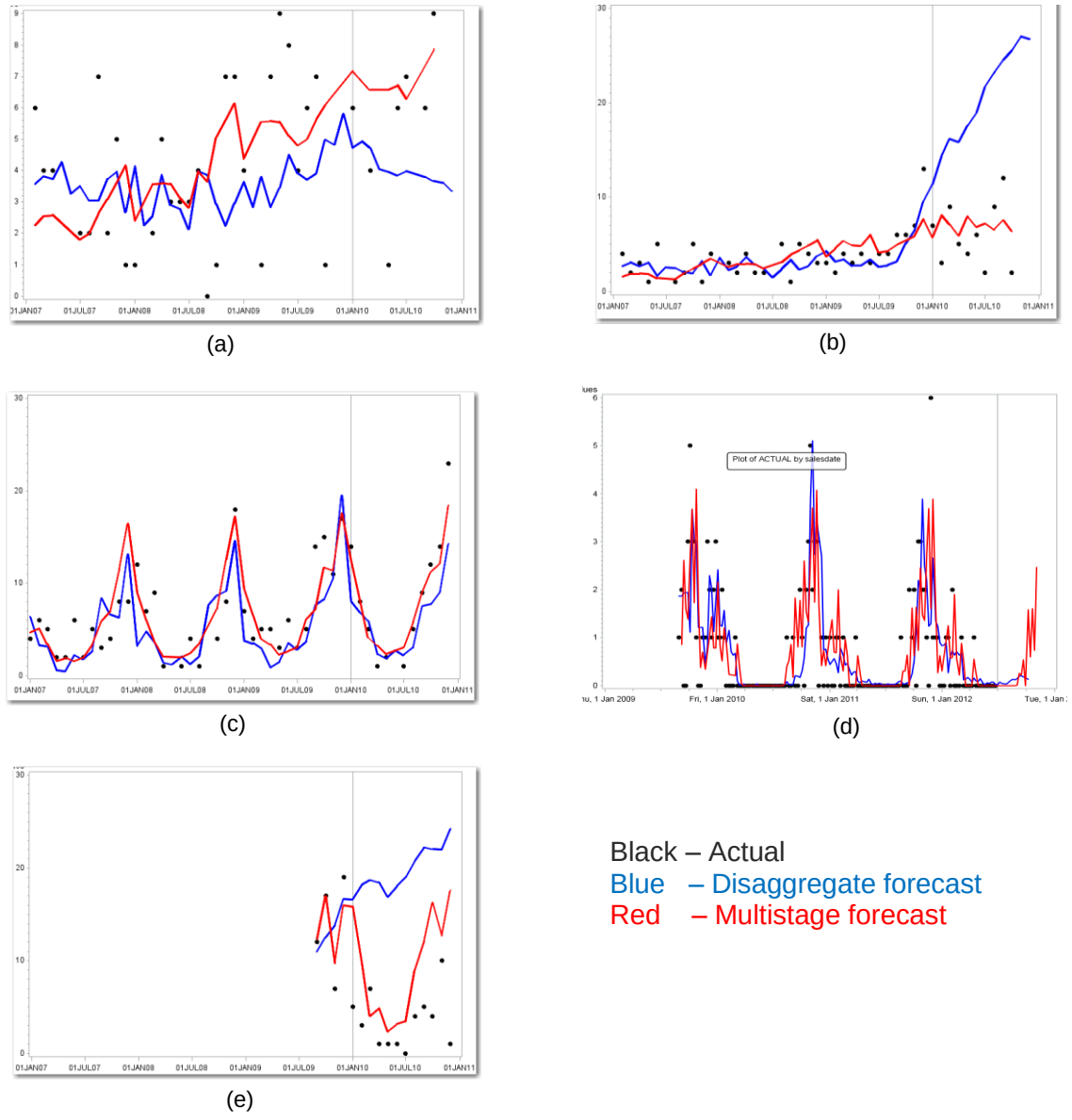


Figure 8. Multistage Forecasts versus Disaggregate Forecasts

TYPE	AUTO_MAPE	DISAGG_MAPE	MS_MAPE
FIT	144.53%	60.78%	22.22%
HOLDOUT	94.91%	79.36%	46.07%

Table 2. MAPEs of Forecasts

MULTISTAGE MODELING WORKFLOW

To sum up, here is the proposed two-stage multistage modeling workflow:

- Stage 1
  - Aggregate the data.
  - Forecast (time series model / regression model, and so on).
- Stage 2
  - Pool the information across series; apply custom models to generate a low-level forecast for each individual time series.
  - Perform top-down disaggregation to generate the final forecast.

DISCUSSION

The idea of multistage modeling is to use the hierarchy structure, pooling and borrowing information across multiple series. The algorithm that is used in the case study is just one application of the multistage modeling concept. Multistage modeling can be more flexible in real-world applications. The following sections present two more possible uses of the multistage modeling framework.

MULTILEVEL FEATURE EXTRACTION

In multilevel feature extraction, you extract features (estimate parameters) at a high level and use the extracted features or parameters as an adjustment at a low level. The following steps illustrate the workflow:

1. Aggregate the data.
2. Estimate the seasonality at the aggregate level.
3. Remove the seasonality at a low level.
4. Build a forecasting model and generate a forecast.
5. Apply seasonality to a predicted series to generate the final forecast.

The concept is applied to customer data, where six months (two quarters) of data are held out. The error statistics using a disaggregate model and a multistage model are listed in Table 3, including MAPE for historical fit as well as MAPE for both quarters in the holdout period. Apparently, the multistage model performs much better than the disaggregate model in both historical fit and future forecasting. Furthermore, there is a huge increase in the MAPE from the first quarter to the second quarter when you use the disaggregate model. In contrast, the multistage model provides a much more stable forecast for future periods.

TYPE	DISAGG_MAPE	MS_MAPE
FIT	6.81%	8.88%
Q1	81.40%	40.56%
Q2	308.94%	55.89%

Table 3. Error Statistics Using Different Models

THREE-STAGE MODELING

Multistage modeling is not restricted to the two-level framework. You can build models at different levels of aggregated data for different purposes of feature extraction and parameter estimation. So the word "multistage" can refer to three or more stages—however many stages best fit your data analysis. The following is one example of three-stage modeling:

1. Stage 1: Estimate the trend, seasonality, and holiday effects at a selected level of the hierarchy. The level of aggregation is usually at a relatively high level in the hierarchy. For example, the level could be selected as the climate zone level for a location hierarchy or the department level for a product hierarchy.

2. Stage 2: Estimate the price elasticity and promotion effects at a selected level of the hierarchy. For this stage, the level of aggregation is the one where the company makes pricing decisions. For example, the corresponding level can be the region level for a location hierarchy or the category level for a product hierarchy.

3. Combine parameters from Stage 1 and Stage 2, and generate a forecast for the aggregated level (same level as aggregation level in Stage 2) of the data.

4. Stage 3: Build custom forecasting models and generate forecasts at a low level. Disaggregate the high-level forecast to get the final forecast.

CONCLUSION

This paper goes through a case study step by step to show how the forecasting model is improved. It proposes a multistage modeling framework for demand forecasting based on best practices. The key idea is to borrow information across multiple series or inherit features from aggregate data. In addition to the two-stage modeling used in the case study, two more types of implementations are provided: multilevel feature extraction and three-stage modeling. The proposed framework shows great improvement in forecasting accuracy compared to that of traditional time series models.

REFERENCE

Wang, P., and Li, Y. (2015). "Automatic Time Series Classification and Clustering." Thirty-Fifth International Symposium of Forecasting, sponsored by the International Institute of Forecasters.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

Pu Wang
SAS Campus Drive
Cary, NC 27513
980-938-1811
Pu.Wang@sas.com

Alex Chien
SAS Campus Drive
Cary, NC 27513
919-531-9236
Alex.Chien@sas.com

Yue Li
SAS Campus Drive
Cary, NC 27513
949-517-9342
Yue.Li@sas.com