CHAPTER 12

Time Series Forecasting

Time series is a series of observations listed in the order of time. The data points in a time series are usually recorded at constant successive time intervals. Time series *analysis* is the process of extracting meaningful non-trivial information and patterns from time series. Time series *forecasting* is the process of predicting the future value of time series data based on past observations and other inputs. Time series forecasting is one of the oldest known predictive analytics techniques. It is widely used in every organizational setting and has deep statistical foundations. An example of a time series is shown in Fig. 12.1. It shows the time series value and period—monthly revenue of a product for a few years.

Up to this point in this book, supervised model building has been about collecting data from several different attributes of a *system* and using these attributes to fit a *function* to predict a desired quantity or target variable. For example, if the system was a housing market, the attributes may have been the price of a house, location, square footage, number of rooms, number of floors, age of the house, and so on. A multiple linear regression model or a neural network model could be built to predict the house price (target) variable given the independent (predictor) variables. Similarly, purchasing managers may use data from several different raw material commodity prices to model the cost of a product. The common thread among these predictive models is that predictors or independent variables that potentially influence a target (house price or product cost) are used to predict that target variable. The objective in time series forecasting is slightly different: to use historical information about a particular quantity to make forecasts about the value of the *same quantity* in the future.

In general, there are two important differences between time series analysis and other supervised predictive models. First, time is an important

Data Science. DOI: https://doi.org/10.1016/B978-0-12-814761-0.00012-5

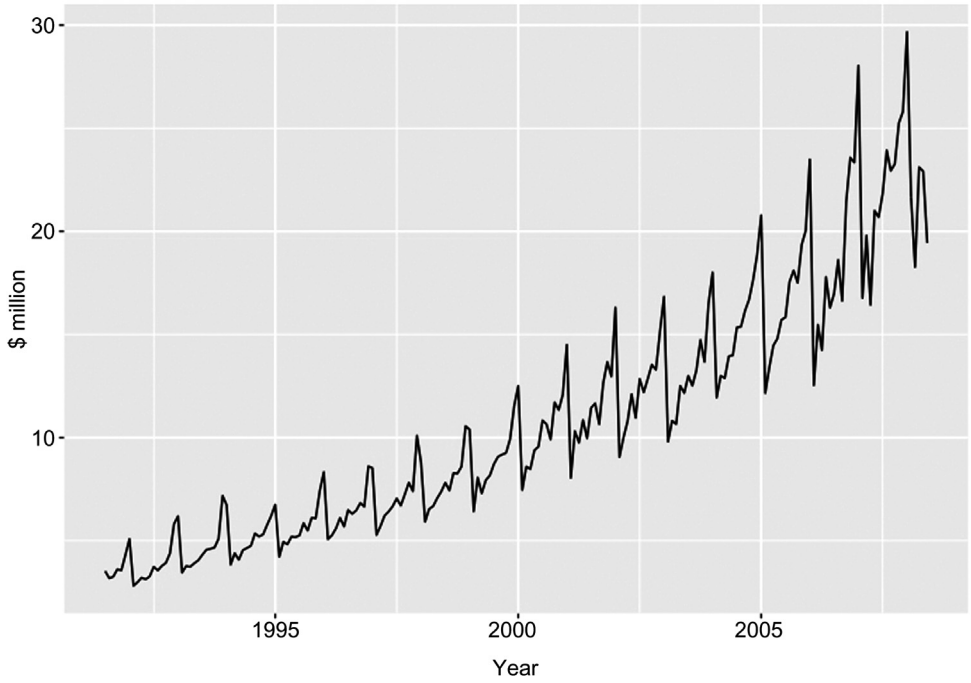


FIGURE 12.1
Time series of monthly antidiabetic drug sales.[1]

predictor in many applications. In time series analysis one is concerned with forecasting a specific variable, given that it is known how this variable has changed over time in the past. In all other predictive models discussed so far, the time component of the data was either ignored or was not available. Such data are known as *cross-sectional* data. Consider the problem of predicting the house price based on location, square footage, number of rooms, number of floors, and age of the house. The predictors are observed in a point in time (like a cross-section of a block of wood in Fig. 12.2) and the price is predicted in the same point in time. However, it is important to take the time variable (length of wood) to predict something like house prices. It fluctuates up and down based on economic conditions, supply and demand, etc., which are all influenced by time.

[1] Total monthly scripts for pharmaceutical products falling under ATC code A10, as recorded by the Australian Health Insurance Commission. July 1991–June 2008 (Hyndman & Athanasopoulos, 2018).

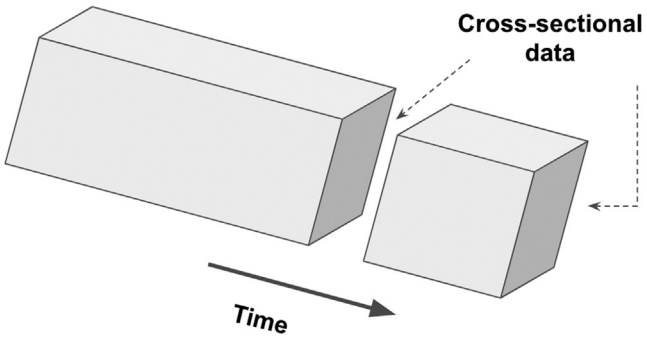


FIGURE 12.2
Cross-sectional data is a subset of time series data.

Second, one may not be interested in or might not even have data for other attributes that could potentially influence the target variable. In other words, independent or predictor variables are not strictly necessary for univariate time series forecasting but are strongly recommended for multivariate time series.

TAXONOMY OF TIME SERIES FORECASTING

The investigation of time series can also be broadly divided into *descriptive* modeling, called time series analysis, and *predictive* modeling, called time series forecasting. Time Series forecasting can be further classified into four broad categories of techniques: Forecasting based on time series decomposition, smoothing based techniques, regression based techniques, and machine learning-based techniques. Fig. 12.3 shows the classification of time series forecasting techniques.

Time series decomposition is the process of deconstructing a time series into the number of constituent components with each representing an underlying phenomenon. Decomposition splits the time series into a trend component, a seasonal component, and a noise component. The trend and seasonality components are predictable (and are called systematic components), whereas, the noise, by definition, is random (and is called the non-systematic component). Before forecasting the time series, it is important to understand and describe the components that make the time series. These individual components can be better forecasted using regression or similar techniques and combined together as an aggregated

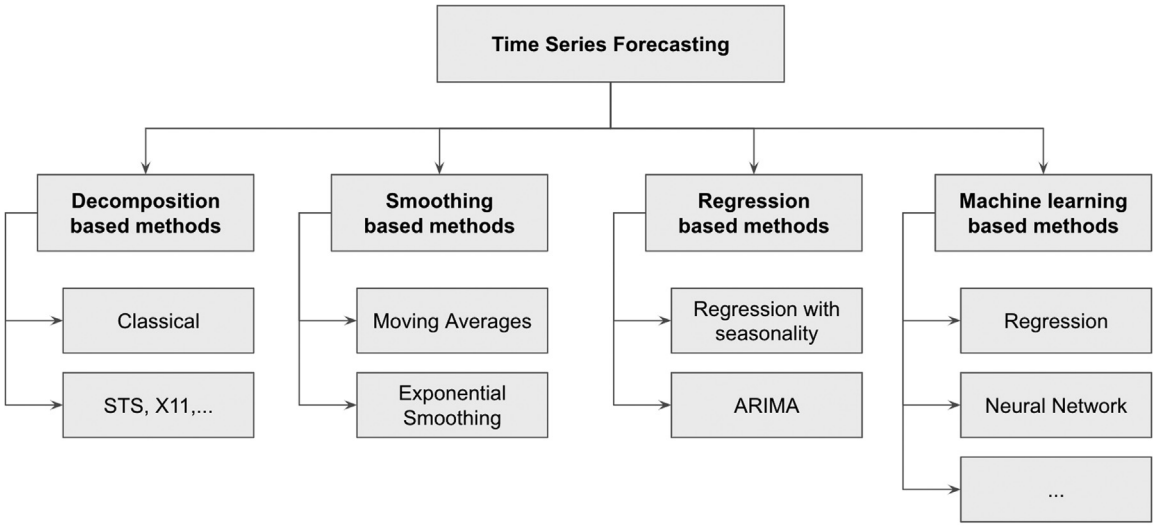


FIGURE 12.3
Taxonomy of time series forecasting techniques.

forecasted time series. This technique is called forecasting with decomposition.

Time series can be thought as past observations informing future predictions. To forecast future data, one can smooth past observations and project it to the future. Such time series forecasting methods are called smoothing based forecasting methods. In smoothing methods, the future value of the time series is the weighted average of past observations.

Regression based forecasting techniques are similar to conventional supervised predictive models, which have independent and dependent variables, but with a twist: the independent variable is now time. The simplest of such methods is of course a linear regression model of the form:

$$y_t = a \times t + b \tag{12.1}$$

where y_t is the value of the target variable at time t. Given a training set, the values of coefficients a and b can be estimated to forecast future y values. Regression based techniques can get pretty complicated with the type of function used to model the relationship between future value and time. Commonly used functions are exponential, polynomial, and power law functions.

Most people are familiar with the trend line function in spreadsheet programs, which offer several different function choices. The regression based time series forecast differs from a regular function-fitting predictive model in the choice of the independent variable. A more sophisticated technique is based on the concept of autocorrelation. Autocorrelation refers to the fact that data from adjacent time periods are correlated in a time series. The most well-known among these techniques is ARIMA, which stands for Auto Regressive Integrated Moving Average.

Any supervised classification or regression predictive models can be used to forecast the time series too, if the time series data are transformed to a particular format with a target label and input variables. This class of techniques are based on supervised machine learning models where the input variables are derived from the time series using a *windowing* technique. The windowing technique transforms a time series to a cross-sectional like dataset where the input variables are lagged data points for an observation. The artificial neural network-based time series forecasting has particular relevance because of its resemblance with the ARIMA technique.

FORECASTING DEMAND OF A PRODUCT

A common application of a time series is to forecast the demand for a product. A manufacturing company makes anti-corrosion wax tapes for use in gas and oil pipelines. The company makes more than a dozen varieties of wax tape products using a handful of assembly lines. The demand for these products varies depending on several factors. For example, routine pipeline maintenance is typically done during warm weather seasons. So, there could be a seasonal spike in the demand. Also, over the last several years, growth in emerging economies has meant that the demand for their products has been growing. Finally, any upcoming changes in pricing, which the company may announce ahead of time, may also trigger stockpiling by their customers, resulting in sudden jumps in demand. So, there can be both trend and seasonality factors, as shown in a sample series Fig. 12.4.

The product manager of the product needs to be able to predict the demand of their products on a monthly, quarterly, and annual basis so that they can plan the production using their limited resources and their department's budget. They make use of the time series forecasting models to predict the potential demand for each of their product lines. By studying the seasonal patterns and growth trends, they can better prepare their production lines. For example, studying seasonality in the sales for the #2 wax tape, which is heavily used in cold climates, reveals that March and April are the months with the highest number of orders placed as customers buy them ahead of the maintenance seasons starting in the summer months. So, the plant manager can dedicate most of their production lines to manufacturing the #2 tape during these months. This insight would not be known unless a time series analysis and forecasting was performed.

(*Continued*)

(Continued)

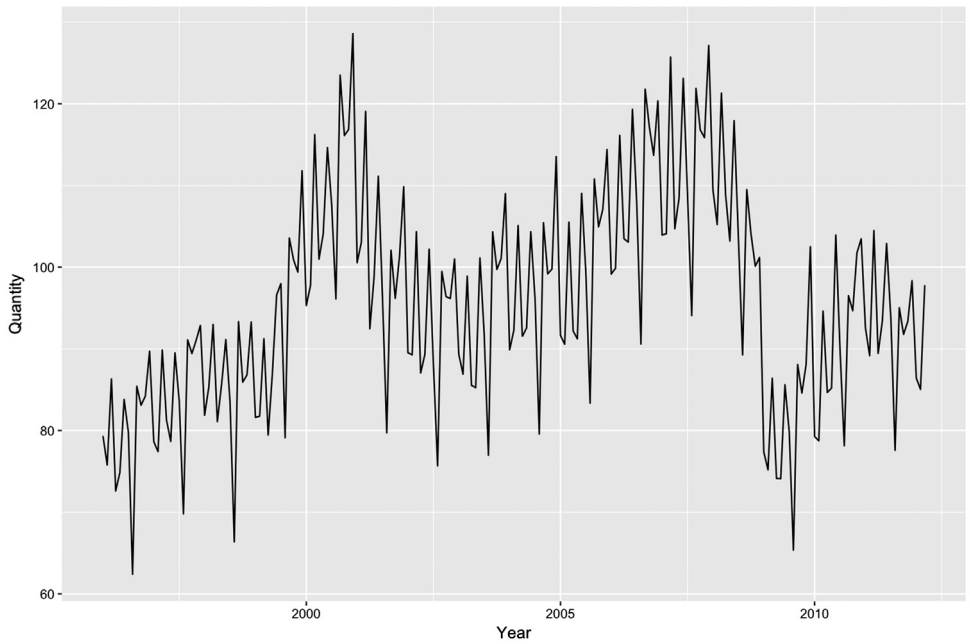


FIGURE 12.4
A time series analysis can reveal trends and seasonal patterns.

12.1 TIME SERIES DECOMPOSITION

Time series data, are univariate, an amalgamation of multiple underlying phenomenon. Consider the example shown in Fig. 12.1. It is a time series of monthly antidiabetic drug sales. A few observations can be made about this time series. Firstly, the overall drug sales is trending upward and the upward trend accelerates in the 2000s. Secondly, there is clear seasonality in the time series of drug sales. In particular, it is a yearly seasonality. There is a spike in drug sales at the start of the year and a dip in every February. This seasonal variation is consistent every year. However, even when accounting for the trend and the seasonal variations there is one more phenomenon that could not be explained. For example, the pattern in 2007 is odd when compared with prior years or 2008.

This unattributable phenomenon can be assumed as noise in the time series. Now that an intuitive understanding of the underlying phenomena of the time series has been developed, the formal definitions of different components that make up a time series can be discussed.

Trend: Trend is the long-term tendency of the data. It represents change from one period to next. If a trend line has been inserted in over a chart in a spreadsheet, it is the regression equation line that represents the trend component of a time series. The trends can be further split into a zero-basis trend and the level in the data. The level in the time series does not change with respect to time while the zero-basis trend does change with the time.

Seasonality: Seasonality is the repetitive behavior during a cycle of time. These are repeated patterns appearing over and over again in the time series. Seasonality can be further split into hourly, daily, weekly, monthly, quarterly, and yearly seasonality. Consider the revenue generated by an online finance portal, like Yahoo Finance or Morningstar. The visits would clearly indicate a daily seasonality with the difference in the number of people accessing the portal or the app during the day time, especially during the market open hours, and the nighttime. Weekdays will have higher traffic (and revenue) than during the weekends. Moreover, the online advertising spend shows quarterly seasonality as the advertisers adjust marketing spend during the end of the quarter. The revenue will also show yearly seasonality where end of the year, after Christmas, shows weakness in revenue because of holidays.

Cycle: Cyclic component represents longer-than-a-year patterns where there is no specific time frames between the cycles. An example here is the economic cycle of booms and crashes. While the booms and crashes exhibit a repeated pattern, the length of a boom period, the length of a recession, the time between subsequent booms and crashes (and even two consecutive crashes—double dip) is uncertain and random, unlike the seasonality components. Fig. 12.1, on the antidiabetic drug sales, shows an accelerated upward trend after the year 2000 which may represent a cyclic component or a non-linear trend. It is hard to conclude anything in this time series because of the limited time frame in the dataset.

Noise: In a time series, anything that is not represented by level, trend, seasonality, or cyclic component is the noise in the time series. The noise component is unpredictable but follows normal distribution in ideal cases. All the time series datasets will have noise. Fig. 12.5 shows the decomposition of quarterly production data, into trend, seasonality, and noise components.

The trend and seasonality are the systematic components of a time series. Systematic components can be forecasted. It is impossible to forecast

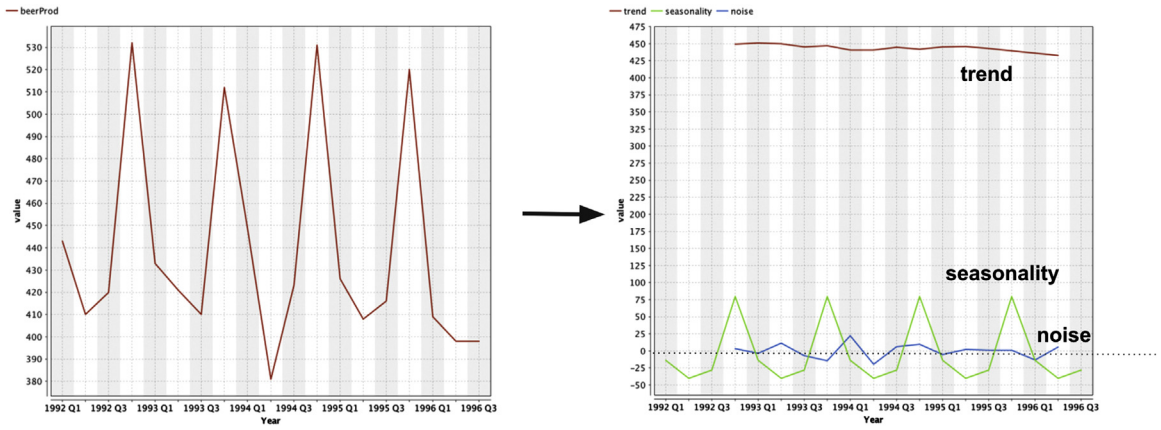


FIGURE 12.5
Decomposition of time series.

noise—the non-systematic component of a time series. In the time series decomposition can be classified into *additive decomposition* and *multiplicative decomposition*, based on the nature of the different components and how they are composed. In an additive decomposition, the components are decomposed in such a way that when they are added together, the original time series can be obtained.

Time series = Trend + Seasonality + Noise

In the case of multiplicative decomposition the components are decomposed in the such a way that when they are multiplied together, the original time series can be derived back.

Time series = Trend × Seasonality × Noise

Both additive and multiplicative time series decompostions can be represented by these equations:

$$y_t = T_t + S_t + E_t \tag{12.2}$$

$$y_t = T_t \times S_t \times E_t \tag{12.3}$$

where T_t, S_t, and E_t are trend, seasonal, and error components respectively. The original time series y_t is just an additive or multiplicative combination of components. If the magnitude of the seasonal fluctuation or the variation in trend changes with the level of the time series, then multiplicative time series decomposition is the better model. The antidiabetic drug sales time series from Fig. 12.1 shows increasing seasonal fluctuation and exponential rise in the long-term trend. Hence, a multiplicative model is more appropriate. Fig. 12.6 shows both the additive and multiplicative decomposition of the

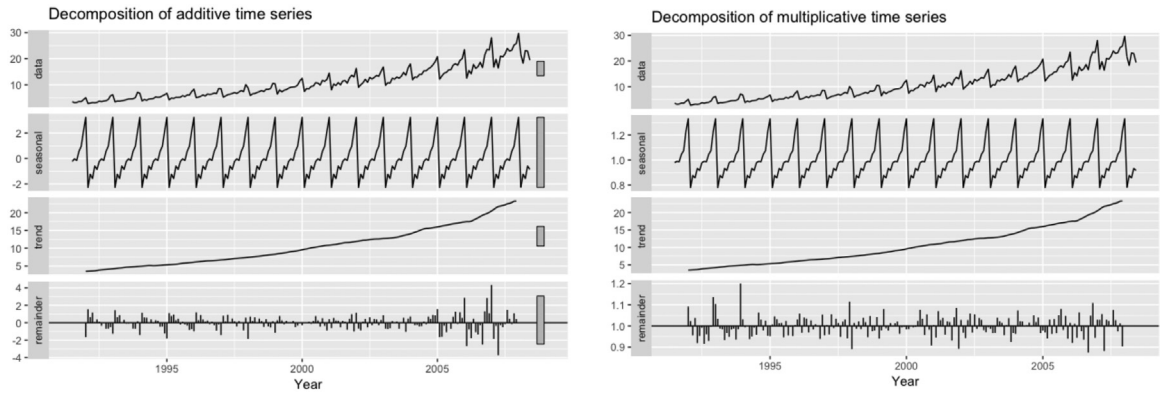


FIGURE 12.6
Additive and multiplicative decomposition of time series.

antidiabetic drug sales time series. Note the scale of the seasonal component is different for both decompositions. The noise in the additive decomposition is higher post 2005, as it ineffectively struggles to represent increases in the seasonal variation and trend changes. The noise is much smaller in the multiplicative decomposition.

To decompose the time series data to its individual components, there are a few different techniques available. In this section, some common approaches to describe time series data will be reviewed and how the knowledge can be used for forecasting time series to future time periods.

12.1.1 Classical Decomposition

The classical decomposition technique is simple, intuitive, and serves as a baseline of all other advanced decomposition methods. Suppose the time series has yearly seasonality with monthly data as shown in Fig. 12.1. m represents seasonal period, which is 12 for monthly data with yearly seasonality. The classical decomposition technique first estimates the trend component by calculating the long term (say 12 month) moving average. The trend component is removed from the time series to get remaining seasonal and noise components. The seasonal component can be estimated by average Jan, Feb, Mar, ..., variance in the remaining series. Once the trend and seasonal components are removed, what is left is noise. The algorithm for classic additive decomposition is:

1. Estimate the trend T_t: If m is even, calculate $2 \times m$ moving average (m-MA and then a 2-MA); if m is odd, calculate m-moving average. A moving average is the average of last m data points.

2. Calculate detrended series: Calculate $y_t - T_t$ for each data point in the series.
3. Estimate the seasonal component S_t: average $(y_t - T_t)$ for each m period. For example, calculate the average of all January values of $(y_t - T_t)$ and repeat for all the months. Normalize seasonal value in such a way that mean is zero.
4. Calculate the noise component E_t: $E_t = (y_t - T_t - S_t)$ for each data point in the series.

Multiplicative decomposition is similar to additive decomposition: Replace subtraction with division in the algorithm described. An alternative approach is to convert multiplicative decomposition into additive by applying the logarithmic function on both the sides of Eq. (12.3).

12.1.2 How to Implement

The decomposition algorithm is quite simple, and can be implemented in a spreadsheet. A practical example of a time series additive decomposition using RapidMiner will be briefly described. The process shown in Fig. 12.7 has simple data pre-processing operators to estimate the trend, seasonal, and error components. The data used in this process is the quarterly Australian beer production dataset[2] which can be found on the companion website www.IntroDataScience.com. Steps to convert the time series into the components are:

- *Trend*: Since the data show four quarter seasonality, a four period moving average operator and a two-period moving average operator is used to estimate the trend component.
- *Seasonality*: The *Generate attribute* operator is used to calculate the detrended series by finding the difference between the time series and the 2×4 month moving average values. The same operator also extracts quarter value (Q1, Q2, Q3, Q4) from the Year (e.g., 1996 Q1) attribute. The detrended series has seasonality and noise. To calculate the seasonality, the quarter values have to be averaged using the *Aggregate* operator. This gives seasonality values for one year. The *Join* operator is used to repeat the seasonal component for every Q1, Q2, Q3 and Q4 records.
- *Noise*: Noise is calculated by the difference of the time series with the combination of trend and seasonality.

The result of the process is shown in Fig. 12.8. The initial time series is decomposed into its components. Note that the noise is randomly

[2] Total quarterly beer production in Australia from 1956 to 2010. Australian Bureau of Statistics. Cat. 8301.0.55.001 (Hyndman & Athanasopoulos, 2018).

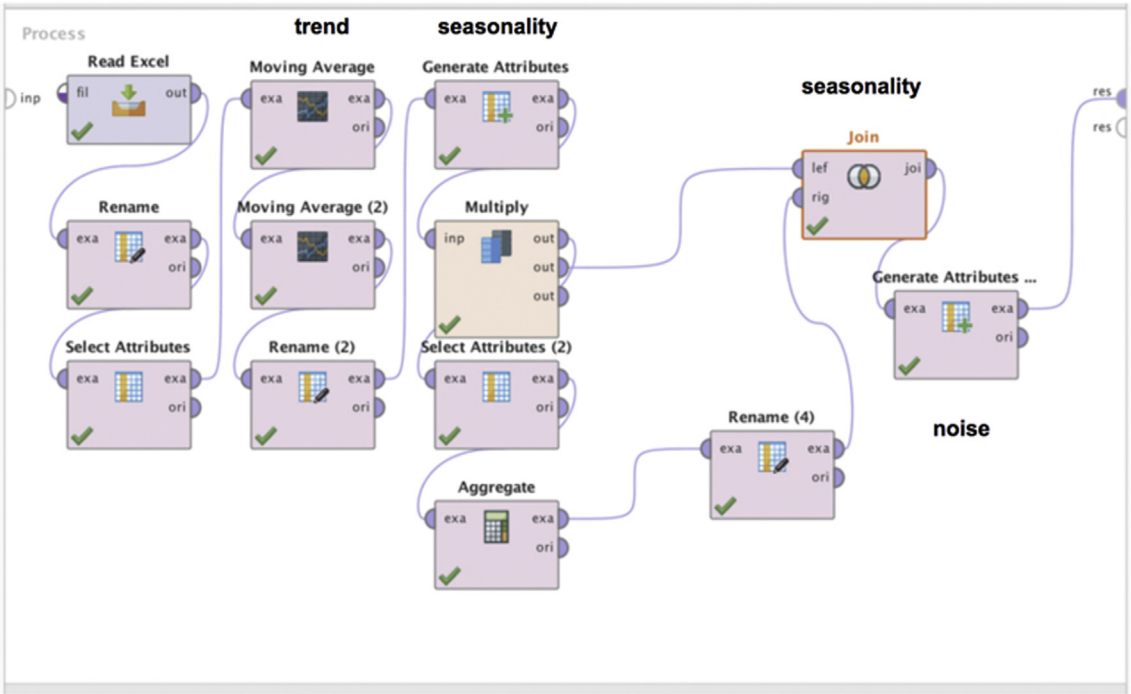


FIGURE 12.7
Process for time series decomposition.

distributed with the mean as zero. Even though the classical decomposition is straight forward, it has some serious limitations. Classical decomposition assumes the occurrence of the same seasonal pattern throughout the entire time series. That is too constrictive of an assumption for practical use cases. As the $2 \times m$ moving average is used, the first $m/2$ and the last $m/2$ data points are not used in the modeling. Moreover, classical decomposition is inept at handling anomalies in the data.

There are quite a range of advanced time series decomposition techniques. STL (Seasonal and Trend decomposition using Loess), SEATS (Seasonal Extraction in ARIMA Time Series), and X11 (from US Census Bureau) are some of the advanced decomposition techniques. All these methods have additional steps to deal with the limitations of the classical decomposition technique, particularly to deal with the change in the seasonality, decomposing quarterly, monthly, weekly, and daily seasonality, and handling changes in the trend.

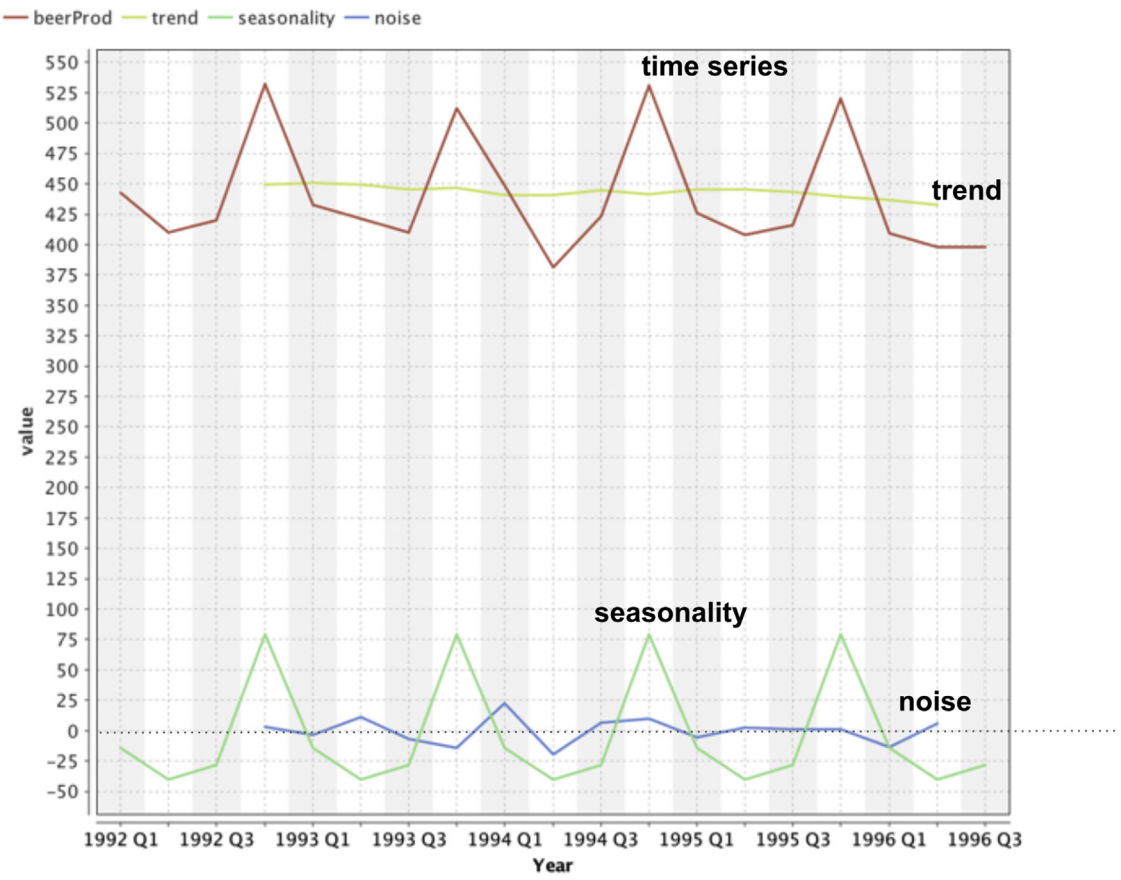


FIGURE 12.8
Time series and decomposed data.

Forecasting Using Decomposed Data

While decomposing the time series is used to describe and increase the understanding of time series data, it is useful for forecasting as well. The idea is to breakdown the time series into its parts, forecast the parts, and put it back together for forecasted future time series values. Why? Forecasting the time series through their components is a much easier task.

$$\hat{y}_t = \hat{S}_t + \hat{T}_t \tag{12.4}$$

It is assumed that the seasonal component of the time series does not change. Hence, the forecast of the seasonal component is the same as the values extracted from the time series. The time series data without

the seasonal component is called *seasonally adjusted* time series. It contains just the trend component and noise in the data. The seasonally adjusted time series can be forecasted by relatively easier methods: linear or polynomial regression, Holt's method, or ARIMA. For example, the trend components in Figs. 12.6 and 12.8 can be extended using linear regression. Noise is normally distributed with the mean as zero. Hence, it is not forecasted. The time series forecast for the future value is the sum of the seasonal forecast and the seasonally adjusted forecast of the trend component.

12.2 SMOOTHING BASED METHODS

In the smoothing based approaches, an observation is a function of past few observations. It is helpful to start out with a basic notation system for time series in order to understand the different smoothing methodologies.

- *Time periods:* $t = 1, 2, 3, \ldots, n$. Time periods can be seconds, days, weeks, months, or years depending on the problem.
- *Data series:* Observation corresponding to each time period above: y_1, y_2, y_3, $\ldots$ y_n.
- *Forecasts*: F_{n+h} is the forecast for the h^{th} time period following n. Usually $h = 1$, the next time period following the last data point. However h can be greater than 1. h is called the *horizon*.
- *Forecast errors*: $e_t = y_t - F_t$ for any given time, t.

In order to explain the different methods, a simple time series data function will be used, $Y(t)$. Y is the *observed* value of the time series at any time t. Furthermore, the observations are made at a constant time interval. If in the case of intermittent data, one can assume that an interpolation scheme is applied to obtain equally spaced (in time) data points.

12.2.1 Simple Forecasting Methods

Naïve Method

Probably the simplest forecasting "model." Here one simply assumes that F_{n+1}, the forecast for the next period in the series, is given by the last data point of the series, y_n

$$F_{n+1} = y_n \tag{12.5}$$

Seasonal Naive Method

If the time series is seasonal, one can forecast better than the naive point estimate. The forecast can assume the value as the previous value of the same

season. For example, the next January revenue can be assumed as the last known January revenue.

$$F_{n+1} = y_{n-s} \tag{12.6}$$

where s is the seasonal period. In the case of monthly data with yearly seasonality, seasonal period is 12.

Average Method

Moving up a level, one could compute the next data point as an average of all the data points in the series. In other words, this model calculates the forecasted value, F_{n+1}, as:

$$F_{n+1} = \text{Average}(y_n, y_{n-1}, y_{n-2}, \ldots, y_1) \tag{12.7}$$

Suppose one has monthly data from January 2010 to December 2010 and they want to predict the next January 2011 value, they would simply average the values from January 2010 to December 2010.

Moving Average Smoothing

The obvious problem with a simple average is figuring out how many points to use in the average calculation. As the observational data grows (as n increases), should one still use all the n time periods to compute the next forecast? To overcome this problem, one can select a window of the last "k" periods to calculate the average, and as the actual data grows over time, one can always take the last k samples to average, that is, n, $n - 1$, ..., $n - k + 1$. In other words, the window for averaging keeps moving forward and, thus, returns a moving average. Suppose in the simple example of the window $k = 3$; then to predict the January 2021 data, a three-month average would be taken using the last three months. When the actual data from January comes in, the February 2021 value is forecasted using January 2021 (n), December 2020 ($n - 1$) and November 2020 ($n - 3 + 1$ or $n - 2$). This model will result in problems when there is seasonality in the data (e.g., in December for retail or in January for healthcare insurance), which can skew the average. Moving average smoothens the seasonal information in the time series.

$$F_{n+1} = \text{Average}(y_n, y_{n-1}, \ldots, y_{n-k}) \tag{12.8}$$

Weighted Moving Average Smoothing

For some cases, the most recent value could have more influence than some of the earlier values. Most exponential growth occurs due to this simple effect. The forecast for the next period is given by the model:

$$F_{n+1} = (a \times y_n + b \times y_{n-1} + c \times y_{n-k})/(a + b + c) \tag{12.9}$$

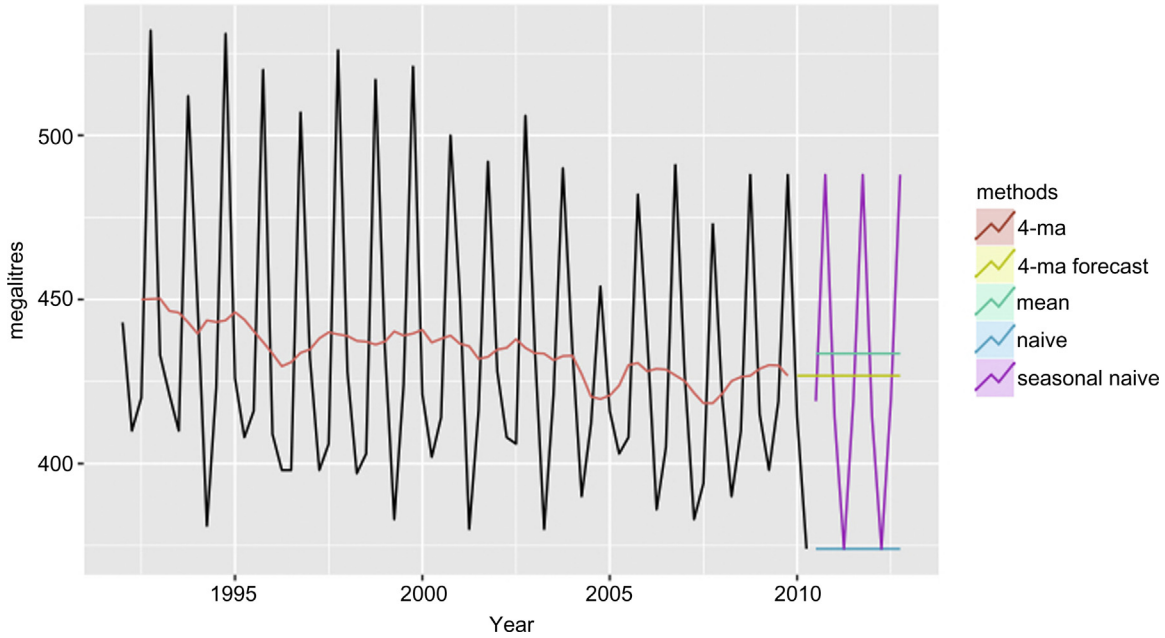


FIGURE 12.9
Comparing one-step-ahead forecasts for basic smoothing methods.

where typically $a > b > c$. Fig. 12.9 compares the forecast results for the simple time series introduced earlier. Note that all of the mentioned methods are able to make only one-step-ahead forecasts due to the nature of their formulation. The coefficients a, b, and c may be arbitrary, but are usually based on some previous knowledge of the time series.

12.2.2 Exponential Smoothing

Exponential smoothing is the weighted average of the past data, with the recent data points given more weight than earlier data points. The weights decay exponentially towards the earlier data points, hence, the name. The exponential smoothing is given by the equation 12.10.

$$F_{n+1} = \alpha y_n + \alpha(1-\alpha)y_{n-1} + \alpha(1-\alpha)^2 y_{n-2} + \cdots. \tag{12.10}$$

α is generally between 0 and 1. Note that $\alpha = 1$ returns the naïve forecast of Eq. (12.5). As seen in the charts in Fig. 12.10, using a higher α results in putting more weight on actual values and the resulting curve is closer to the actual curve, but using a lower α results in putting more emphasis on

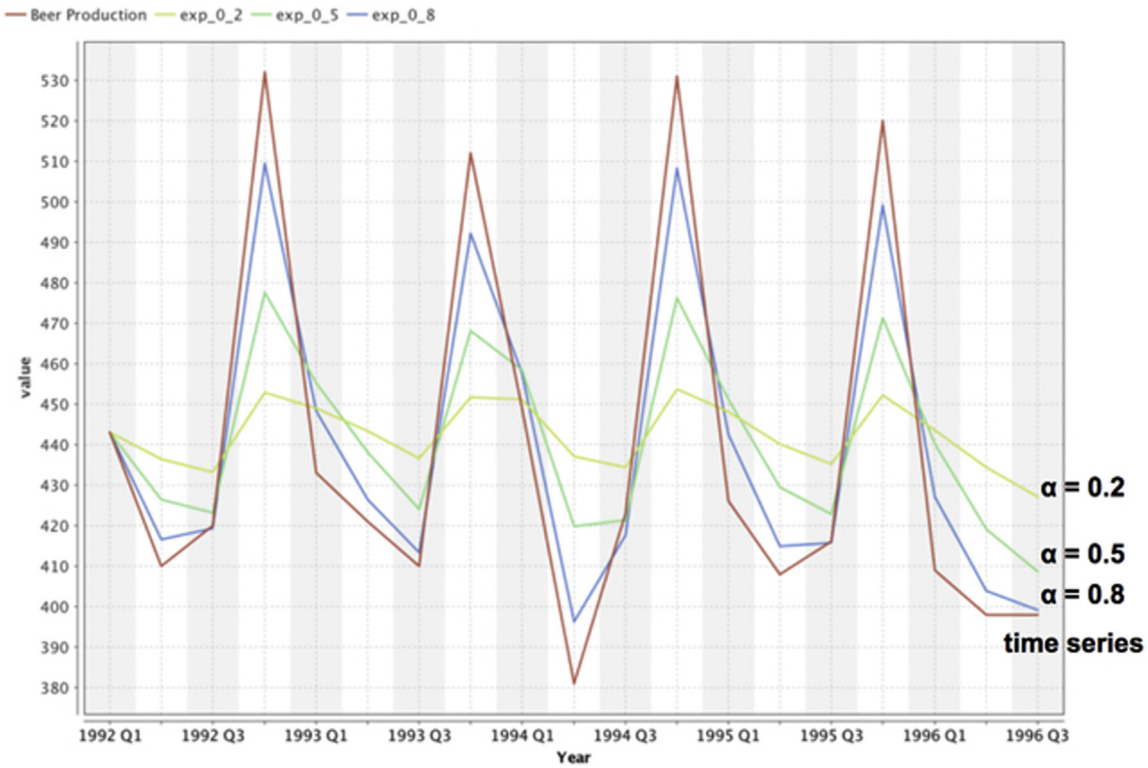


FIGURE 12.10
Fitted time series—exponential smoothing with different α levels.

previously forecasted values and results in a smoother but less accurate fit. Typical values for α range from 0.2 to 0.4 in practice.

To forecast the future values using exponential smoothing, Eq. (12.10) can be rewritten as:

$$F_{n+1} = \alpha \times y_n + (1 - \alpha) \times F_n \tag{12.11}$$

Eq. (12.11) is more useful because it involves both actual value y_n and forecasted value F_n. A higher α value gives an exact fit and a lower value gives a smoother fit. Going back to the monthly example, if one wanted to make the February 2011 forecast using not only the actual January 2011 value but also the previously *forecasted* January 2011 value, the new forecast would have "learnt" the data better. This is the concept behind basic exponential smoothing (Brown, 1956).

This simple exponential smoothing is the basis for a number of common smoothing based forecasting methods. However, the model is suited only for time series *without clear trend or seasonality*. The smoothing model has only one parameter, α, and can help smooth the data in a time series so that it is easy to extrapolate and make forecasts. Like many methods discussed earlier, the forecast will be flat that is, there is no trend or seasonality factored in. Only the level is forecasted. Also, if Eq. (12.10) were examined, one would see that forecasts cannot be made more than one-step ahead, because to make a forecast for step $(n + 1)$, the data for the previous step, n, is needed. It is not possible to make forecasts several steps ahead, that is, $(n + h)$, using the methods described (where it was simply assumed that $F_{n+h} = F_{n+1}$), where h is the horizon. This obviously has limited utility. For making longer horizon forecasts, that is, where $h \gg 1$, the trend and seasonality information also needs to be considered. Once trend and seasonality are captured, one can forecast the value for any time in the future, not just the values for one step ahead.

Holt's Two-Parameter Exponential Smoothing

Anyone who has used a spreadsheet for creating trend lines on scatter plots intuitively knows what a trend means. A trend is an averaged long-term tendency of a time series. The simplified exponential smoothing model described earlier is not particularly effective at capturing trends. An extension of this technique, called Holt's two-parameter exponential smoothing, is needed to accomplish this.

Recall that exponential smoothing [Eq. (12.10)] simply calculates the average value of the time series at $n + 1$. If the series also has a trend, then an average slope of the series needs to be estimated as well. This is what Holt's two-parameter smoothing does by means of another parameter, β. A smoothing equation similar to Eq. (12.10) is constructed for the average trend at $n + 1$. With two parameters, α and β, any time series with a trend can be modeled and, therefore, forecasted. The forecast can be expressed as a sum of these two components, average value or "level" of the series, L_n, and trend, T_n, recursively as:

$$F_{n+1} = L_n + T_n \tag{12.12}$$

where,

$$L_n = \alpha \times y_n + (1 - \alpha) \times (L_{n-1} + T_{n-1}) \text{ and } T_n = \beta \times (L_n - L_{n-1}) + (1 - \beta) \times T_{n-1} \tag{12.13}$$

To make future a forecast over an horizon, one can modify Equation 12.12 to:

$$F_{n+h} = L_n + h \times T_n \tag{12.14}$$

The values of the parameter can be estimated based on the best fit with the training (past) data.

Holt-Winters' Three-Parameter Exponential Smoothing

When a time series contains seasonality in addition to a trend, yet another parameter, γ, will be needed to estimate the seasonal component of the time series (Winters, 1960). The estimates for value (or level) are now adjusted by a seasonal index, which is computed with a third equation that includes γ. (Shmueli, 2011; Hyndman, 2014; Box, 2008).

$$F_{t+h} = (L_t + {}_hT_t)\, S_{t+h-p} \tag{12.15}$$

$$L_t = \alpha y_t / S_{t-p} + (1-\alpha)(L_{t-1} + T_{t-1}) \tag{12.16}$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \tag{12.17}$$

$$S_t = \gamma\left(y_t / L_t\right) + (1-\gamma)\, S_{t-p} \tag{12.18}$$

where p is the seasonality period. One can estimate the value of the parameters α, β, and γ from the fitting of the smoothing equation with the training data.

12.2.3 How to Implement

Holt-Winters' three parameter smoothing provides a good framework to forecast time series data with level, trend, and seasonality, as long as the seasonal period is well defined. One can implement the time series forecasting model in RapidMiner using R extension. R[3] is a powerful, widely used statistical tool and this use case warrants the integration of RapidMiner with R functionality. The key operator in this process is *Execute R* which accepts the data as the input, processes the data, and outputs the data frame from R.

The data used in the process is the Australian Beer Production time series dataset used in prior time series process. After a bit of data pre-processing, like renaming the attributes and selecting the production data, R operator is invoked for Holt and Holt Winters' forecasting function (Fig. 12.11).

The *Execute R* operator executes the R script entered in the operator. The R script for the Holt-Winters' forecasting is shown below. It uses the forecast package library and invokes the forecasting function holt(inp, $h = 10$) for Holts and hw(inp, $h = 10$) for Holt-Winters' forecasting. The data frame (dataset in R) is returned to RapidMiner process.

[3] https://www.r-project.org/.

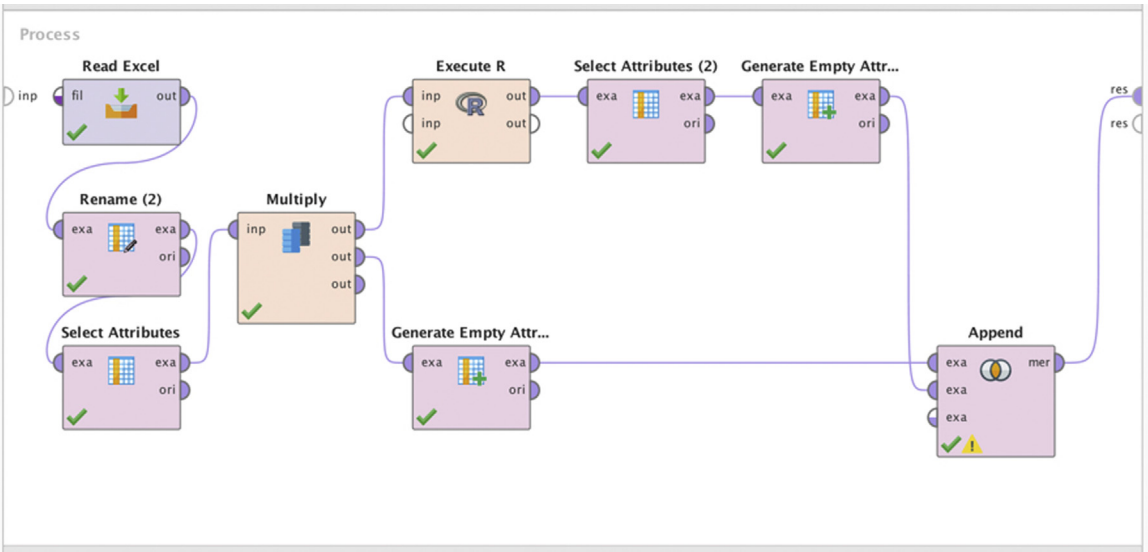


FIGURE 12.11
Holts and Holt-Winters' forecasting using R.

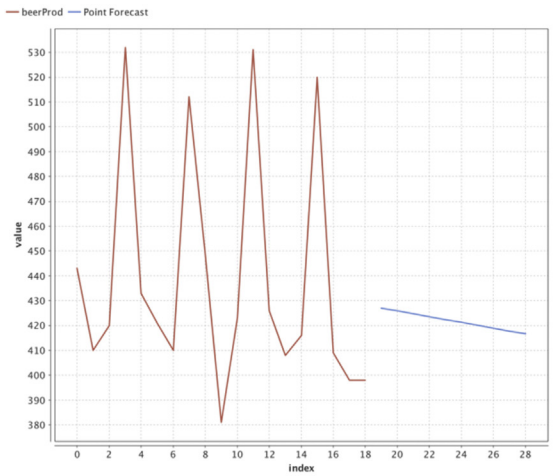
R Script for Holt-Winters' Forecasting

```
rm_main = function(data)
{
library(forecast)
  inp <- ts(data, freq = 4)
    y <- holt(inp, h = 10)   # or hw(inp, h = 10) for Holt Winters' smoothing
    df <- as.data.frame(y)
    return(list(df))
}
```

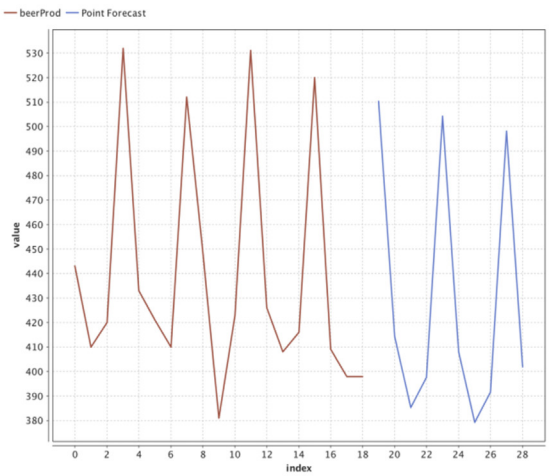
The output from the Execute R operator has point forecasts and the interval forecasts. For this implementation, the point forecast is selected using *Select Attributes* operator. The forecasted dataset is appended with the original dataset for better visualization using *Generate Attribute* and *Append* operators. Fig. 12.12 shows the forecasted result of both Holt's and Holt-Winters' exponential smoothing based forecast. Note that in Holt's forecast, only trend and level are forecasted and in Holt Winters' forecasting both trend and seasonality are forecasted.

12.3 REGRESSION BASED METHODS

In the regression based methods, the variable *time* is the predictor or independent variable and the time series value is the dependent



Holt's Two-Parameter Smoothing



Holt-Winters' Three-Parameter Smoothing

FIGURE 12.12
Holt's and Holt-Winters' smoothing.

variable. Regression based methods are generally preferable when the time series appears to have a global pattern. The idea is that the model parameters will be able to capture these patterns and, thus, enable one to make predictions for any step ahead in the future under the assumption that this pattern is going to be repeated. For a time series with local patterns instead of a global pattern, using a regression based approach requires one to specify how and when the patterns change, which is difficult. For such a series, smoothing approaches work best because these methods usually rely on extrapolating the most recent local pattern as seen earlier.

Fig. 12.13 shows two time series: Fig. 12.13A shows antidiabetic drug revenue and Fig. 12.13B shows the economy class passenger volume in Sydney-Melbourne[4] route. A regression based forecasting method would work well for the antidiabetic drug revenue series because it has a global pattern. However the passenger volume series shows no clear start or end for any patterns. It is preferable to use smoothing based methods to attempt to forecast this second series.

[4] Ansett Airlines (defunct). The dip shows the industrial dispute in 1989. FPP2 package in R.

FIGURE 12.13

Global and local patterns. (A) Antidiabetic drug revenue - global pattern and (B) Airline economy class passenger volume between Sydney and Melbourne - local pattern.

12.3.1 Regression

The simplest of the regression based approaches for analyzing a time series is using linear regression. As mentioned in the introduction to the chapter, one assumes the time period is the independent variable and attempts to predict the time series value using this. For the Australian beer dataset used so far, the chart in Fig. 12.14 shows a linear regression fit. As can be seen, the linear regression model is able to capture the long-term tendency of the series, but it does a poor job of fitting the data.

Sophisticated polynomial functions can be used to improve the fit. Polynomial regression is similar to linear regression except that higher-degree functions of the independent variable are used (squares and cubes on the time variable). Since the global trend here is straight decline, it is difficult to argue that the cubic polynomial does a significantly better job. However in either of these cases, one is not limited to a one-step-ahead forecast of the simple smoothing methods.

12.3.2 Regression With Seasonality

The linear regression trend-fitting model can be significantly improved by simply accounting for seasonality. This is done by introducing *seasonal dummy variables* for each period (quarter), of the series, which triggers either 1 or 0 in the attribute values as seen in Fig. 12.15. In this example, four new attributes are added to the original dataset to indicate which quarter the record belongs to. The attribute value of (Quarter = Q1) is turned 1 if the quarter is Q1 and so on.

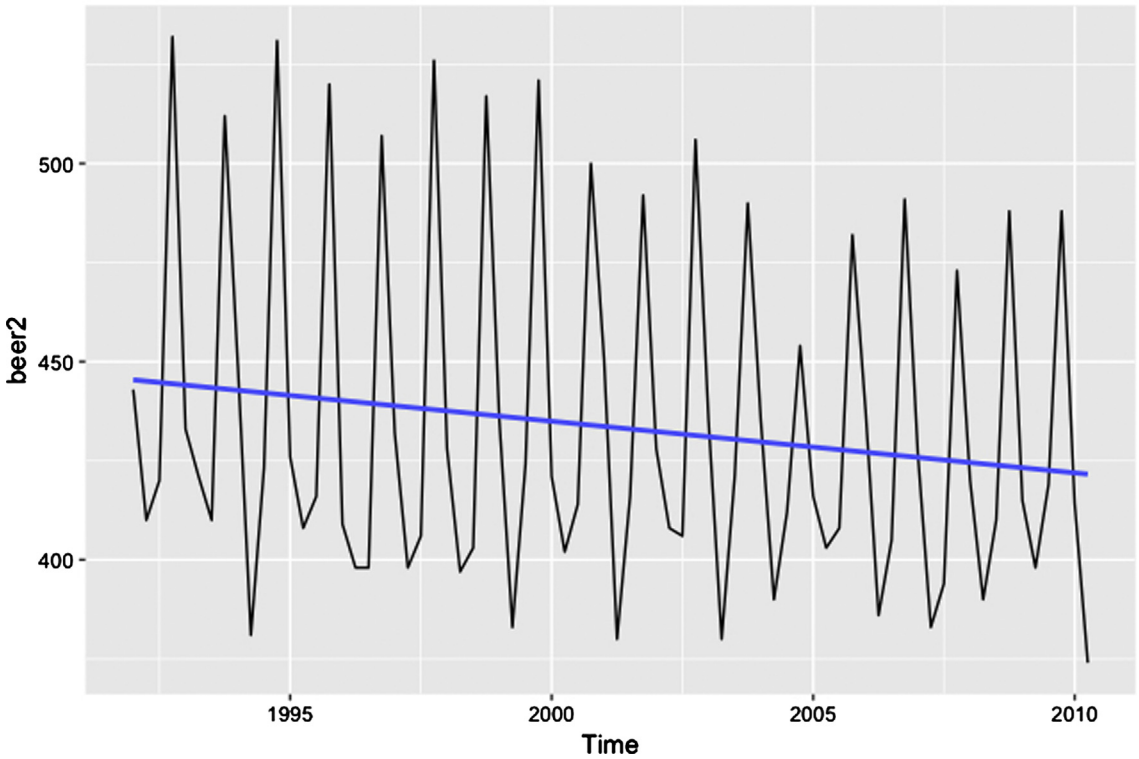


FIGURE 12.14
Linear regression model.

Just this trivial addition to the predictors of the linear regression model can yield a surprisingly good fit in most datasets with clear seasonality. Although the model equation may appear a bit complicated, in reality it is just a linear regression model with four variables: the time period and four dummy variables for each quarter of a year. The independent variable *time* captures the level and the long-term trend. The four dummy seasonal variables capture the seasonality. This regression equation can be used for predicting any future value beyond $n + 1$, and thus, has significantly more utility than the simpler counterparts in the smoothing side.

$$\begin{aligned}\text{Forecast} = {} & 442.189 - 1.132 \times \text{time} - 27.268 \times (\text{Quarter} = \text{Q2}) \\ & - 16.336 \times (\text{Quarter} = \text{Q3}) + 92.882 \times (\text{Quarter} = \text{Q4})\end{aligned}$$

There is of course no reason to use linear regression alone to capture both trend and seasonality. More sophisticated models can easily be built using polynomial equations along with the sine and cosine function to model seasonality.

Year	time	Quarter = Q1	Quarter = Q2	Quarter = Q3	Quarter = Q4	beerProd
1992 Q1	1	1	0	0	0	443
1992 Q2	2	0	1	0	0	410
1992 Q3	3	0	0	1	0	420
1992 Q4	4	0	0	0	1	532
1993 Q1	5	1	0	0	0	433
1993 Q2	6	0	1	0	0	421
1993 Q3	7	0	0	1	0	410
1993 Q4	8	0	0	0	1	512
1994 Q1	9	1	0	0	0	449
1994 Q2	10	0	1	0	0	381

FIGURE 12.15
Seasonal attributes.

How to implement

The implementation process for seasonal linear regression is similar to the linear regression model discussed in Chapter 5, Regression Methods. The additional step is to set up data pre-processing to add seasonal dummy variables. Fig. 12.16 shows the complete RapidMiner process for seasonal linear regression for the Australian beer production dataset. The steps in building the forecasting model are:

1. *Add time attribute*: After the dataset is read, a new attribute "time" is generated to indicate the consecutive time period of each example record. First record "1992 Q1" is tagged as 1 and the next as 2, and so on. This is essential in all regression based forecasting because *time* is the independent variable and will be part of the regression equation. It is important to sort the dataset before adding the sequential time attribute.
2. *Extract seasonal attribute*: The next step is to extract "Q1" from the "1992 Q1" attribute. It can be achieved with Generate attribute and cut () function to extract Q1 from the text. If the time series has weekly or monthly seasonality, corresponding identifiers like weekday or month id have to be extracted.
3. *Generation of independent seasonal attributes*: The quarter attribute has to be pivoted to "Is Q1", "Is Q2", . . . attributes. The *Nominal to Numerical* conversion operator is used to generate these attributes. The value of the attribute is either 1 or 0, depending on the record's seasonal period.

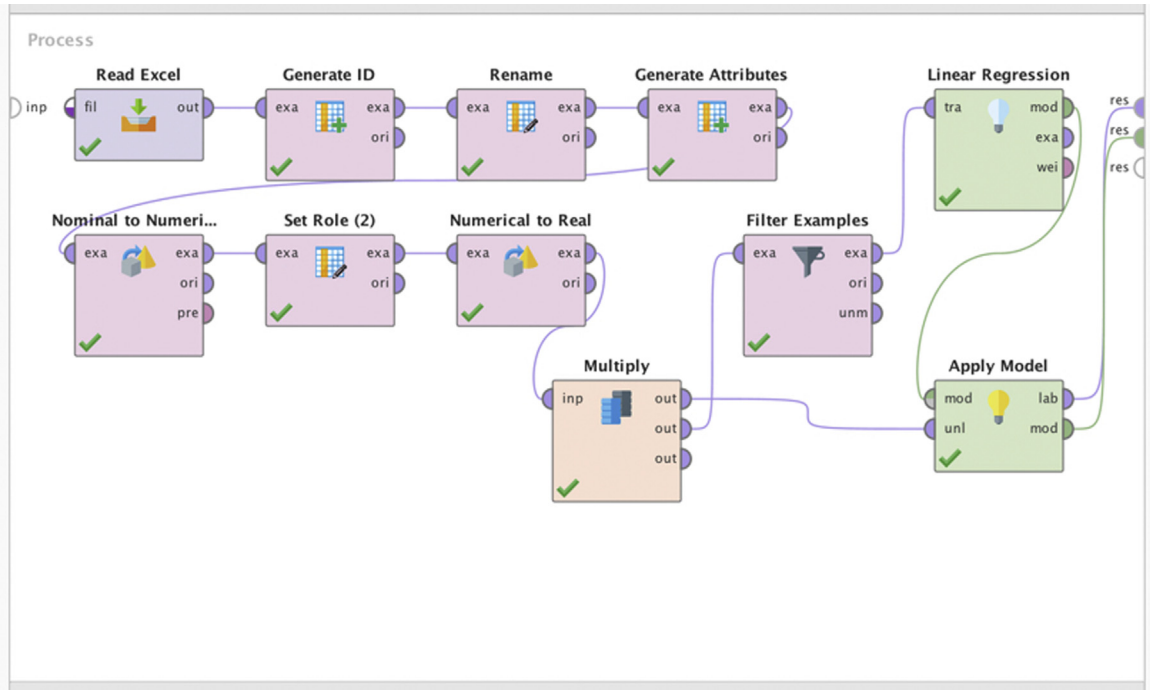


FIGURE 12.16
Process for seasonal linear regression.

4. *Modeling*: A linear regression model is used to fit the equation with the training data. A polynomial regression learner can be used for a better fit.
5. *Forecasting*: The dataset also contains the placeholders for future periods. The *Apply model* operator is used to visualize the fit of the model for the training period and to forecast the future horizon.

The process can be saved and executed. The result window shows the time series of both actual and forecasted data. As shown in Fig. 12.17, the seasonal linear regression forecasting has done a good job of fitting the model for the past data and projected the seasonality for the future horizon. Contrast this result with Fig. 12.14 where the linear regression model captures only the level and long-term trend.

12.3.3 Autoregressive Integrated Moving Average

ARIMA stands for Autoregressive Integrated Moving Average model and is one of the most popular models for time series forecasting. The ARIMA methodology originally developed by Box and Jenkins in the 1970s

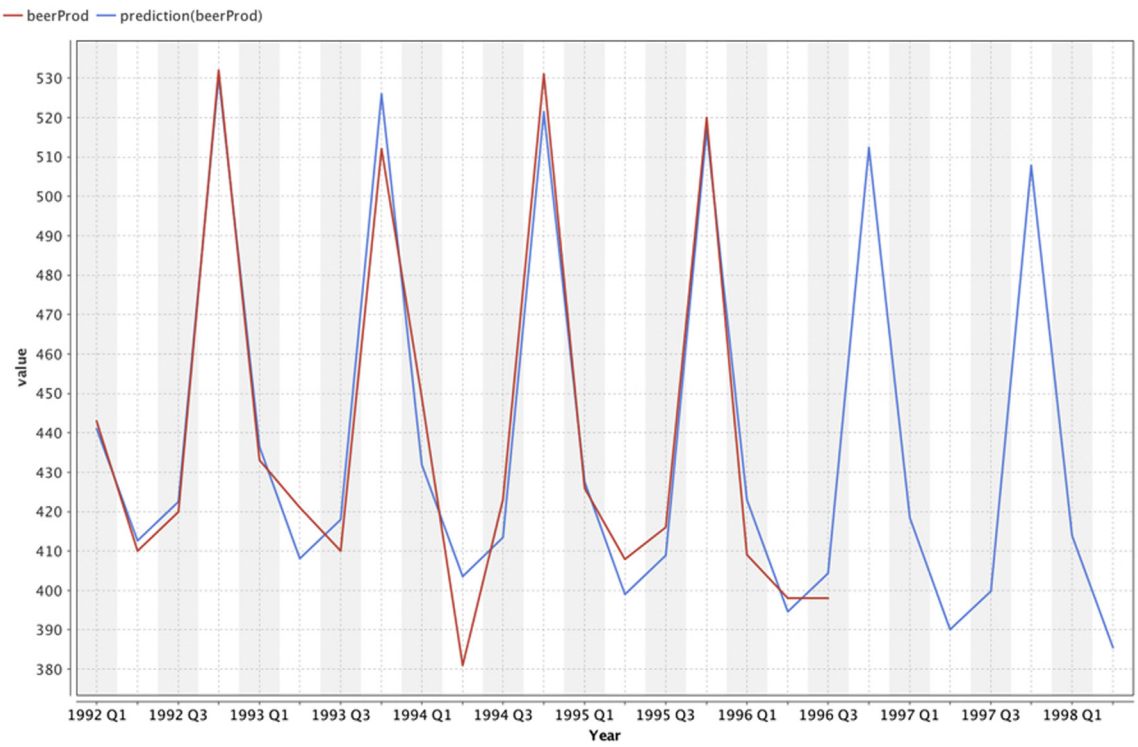


FIGURE 12.17
Forecasting using seasonal linear regression.

(Box, 1970). Even though the steps of training an ARIMA model are more involved, the implementation is relatively straightforward using statistical packages that support ARIMA functions. In this section, the concepts of autocorrelation, autoregression, stationary data, differentiation, and moving average of error are introduced, which are used to increase the understanding on time series and serve as the building blocks of ARIMA models.

Autocorrelation

Correlation measures how two variables are dependent on each other or if they have a linear relationship with each other. Consider the time series shown in Fig. 12.18. The second column "prod" shows the data for the simple time series. In the third column, data are lagged by one step. 1992 Q1 data is shown in 1992 Q2. This new series of values is termed a "1-lag" series. There are an additional 2-lag, 3-lag, . . ., *n*-lag series in the dataset. Notice that there is a strong correlation between the original time series "prod" and 4-lag "prod-4." They tend to move together. This phenomenon is

Year	prod	prod-1	prod-2	prod-3	prod-4	prod-5	prod-6
1992 Q1	443	?	?	?	?	?	?
1992 Q2	410	443	?	?	?	?	?
1992 Q3	420	410	443	?	?	?	?
1992 Q4	532	420	410	443	?	?	?
1993 Q1	433	532	420	410	443	?	?
1993 Q2	421	433	532	420	410	443	?
1993 Q3	410	421	433	532	420	410	443
1993 Q4	512	410	421	433	532	420	410
1994 Q1	449	512	410	421	433	532	420
1994 Q2	381	449	512	410	421	433	532
1994 Q3	423	381	449	512	410	421	433
1994 Q4	531	423	381	449	512	410	421
1995 Q1	426	531	423	381	449	512	410

FIGURE 12.18
Lag series and autocorrelation.

called autocorrelation, where the time series is correlated with its own data points, with a lag.

As in a multivariate correlation matrix (Chapter 3: Data Exploration), one can measure the strength of correlation between the original time series and all the lag series. The plot of the resultant correlation matrix is called an *Autocorrelation Function* (ACF) chart. The ACF chart is used to study all the available seasonality in the time series. From Fig. 12.19 it can be concluded that the time series is correlated with the 4th, 8th, and 12th lagged quarter due to the yearly seasonality. It is also evident than Q1 is negatively correlated with Q2 and Q3.

Autoregressive Models

Autoregressive models are regression models applied on lag series generated using the original time series. Recall in multiple linear regression, the output is a linear combination of multiple input variables. In the case of autoregression models, the output is the future data point and it can be expressed as a linear combination for past p data points. p is the lag window. The autoregressive model can be denoted as the equation:

$$y_t = l + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \cdots + \alpha_p y_{t-p} + e \tag{12.19}$$

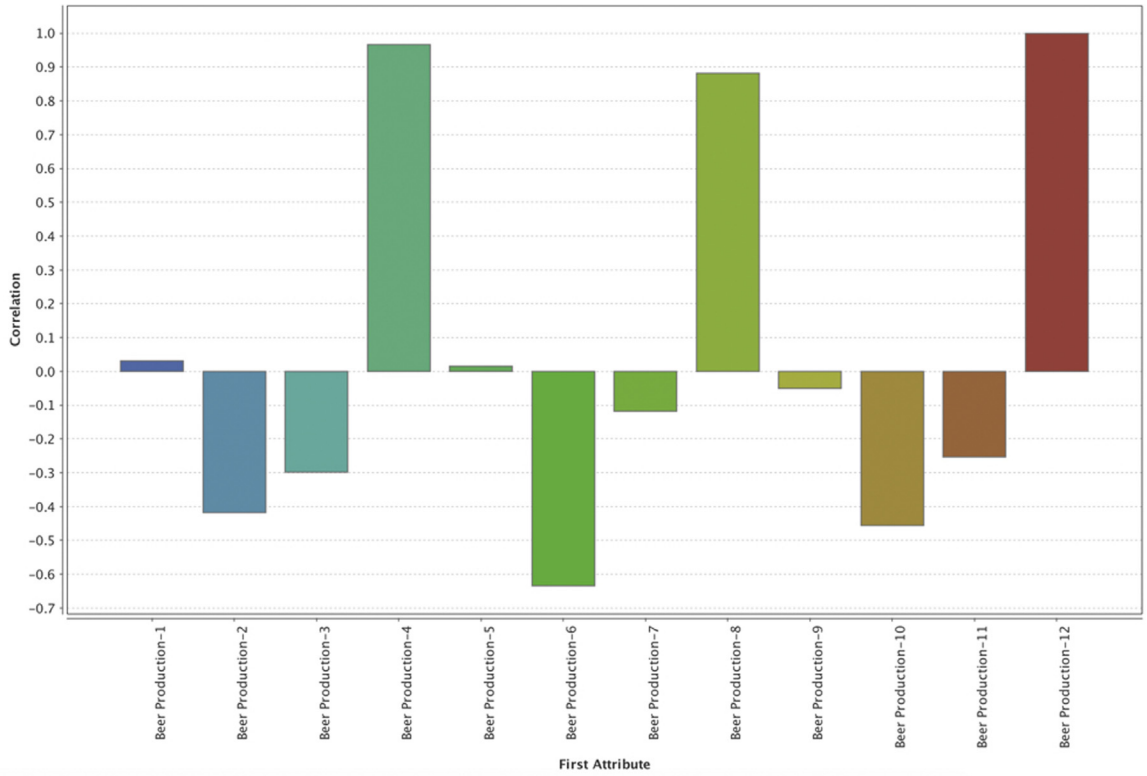


FIGURE 12.19
ACF chart. *ACF*, Autocorrelation Function.

where, l is the level in the dataset and e is the noise. α are the coefficients that need to be learned from the data. This can be referred to as an autoregressive model with p lags or an **AR(p)** model. In an AR(p) model, lag series is a new predictor used to fit the dependent variable, which is still the original series value, Y_t.

Stationary Data

In a time series with trends or seasonality, the value is affected by time (hence, one would be interested in this subject). A time series is called *stationary* when the value of time series is not dependent on time. For instance, random white noise is a stationary time series. Daily temperature at a location is not stationary as there will be a seasonal trend and it is affected by time. Meanwhile, the noise component of a time series is stationary. Stationary time series do not have any means of being forecasted as they are completely random. Fig. 12.20A is an example of nonstationary data because

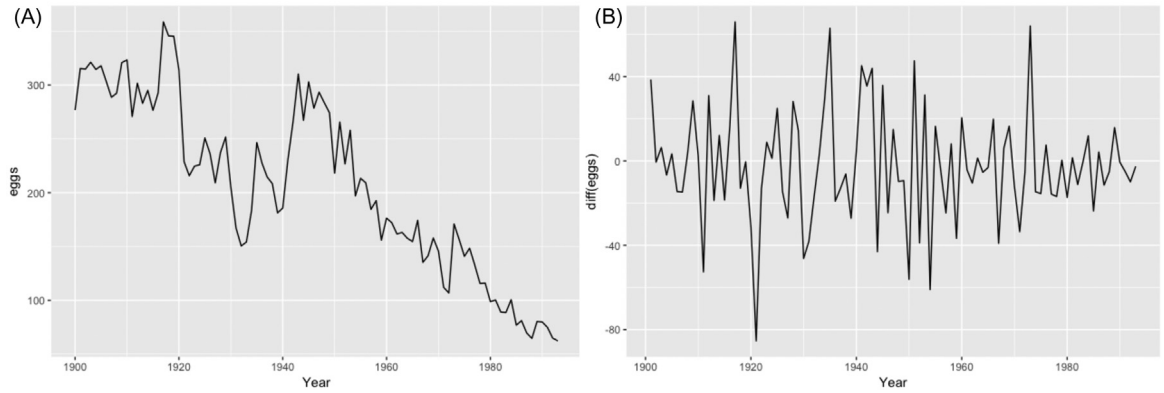


FIGURE 12.20
(A) Non-stationary Time series and (B) Stationary time series.

of the presence of both trend and seasonality and Fig. 12.20B is an example of stationary data because there is no clear trend or seasonality.

Differencing

A non-stationary time series can be converted to a stationary time series through a technique called differencing. Differencing series is the change between consecutive data points in the series.

$$y'_t = y_t - y_{t-1} \tag{12.20}$$

This is called first order differencing. Fig. 12.20 shows a time series and a first order differenced time series. In some cases, just differencing once will still yield a nonstationary time series. In that case a second order differencing is required. Second order differencing is the change between two consecutive data points in a first order differenced time series. To generalize, differencing of order **d** is used to convert nonstationary time series to stationary time series.

Seasonal differencing is the change between the same period in two different seasons. Assume a season has period, *m*.

$$y'_t = y_t - y_{t-m} \tag{12.21}$$

This is similar to the Year-over-Year metric used commonly in business financial reports. It is also called as m-lag first order differencing. Fig. 12.21 shows the seasonal differencing of the Australian Beer production dataset and the seasonal first order differencing of the same series with the seasonal lag as 4—to factor in the number of quarters in a year.

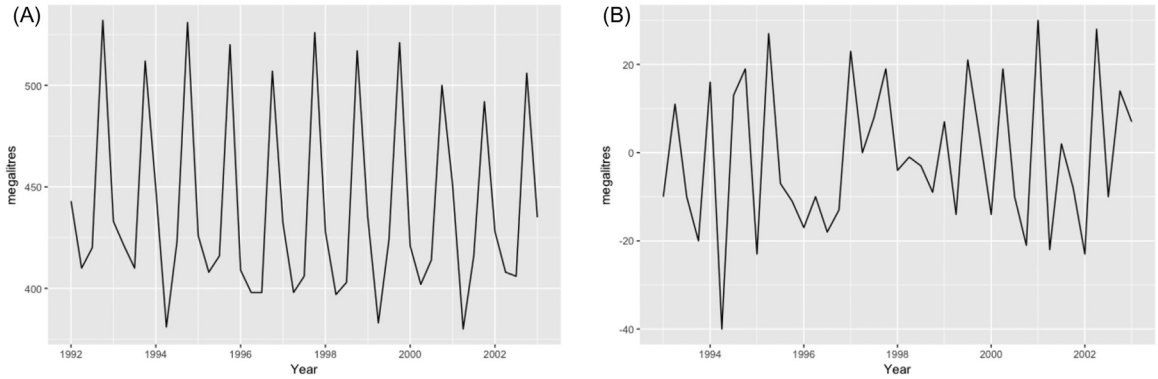


FIGURE 12.21
(A) Time series (B) Seasonal differencing of the time series.

Moving Average of Error

In addition to creating a regression of actual past "p" values as shown in Eq. (12.19), one can also create a regression equation involving forecast errors of past data and use it as a predictor. Consider this equation with:

$$y_t = I + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q} \tag{12.22}$$

where e_i is the forecast error of data point i. This makes sense for the past data points but not for data point t because it is still being forecasted. Hence, e_t is assumed as white noise. The regression equation for y_t can be understood as the weighted (θ) moving average of past q forecast errors. This is called Moving Average with q lags model or **MA(q)**.

Autoregressive Integrated Moving Average

The Autoregressive Integrated Moving Average (ARIMA) model is a combination of the differenced autoregressive model with the moving average model. It is expressed as:

$$y'_t = I + \alpha_1 y'_{t-1} + \alpha_2 y'_{t-2} + \cdots + \alpha_p y'_{t-p} + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q} \tag{12.23}$$

The AR part of ARIMA shows that the time series is regressed on its own past data. The MA part of ARIMA indicates that the forecast error is a linear combination of past respective errors. The I part of ARIMA shows that the data values have been replaced with differenced values of d order to obtain stationary data, which is the requirement of the ARIMA model approach. Why would one need to get to this level of complexity? The ARIMA model is effective in fitting past data with this combination approach and help forecast future points in a time series.

Eq. (12.23) shows the predictors are the lagged p data points for the autoregressive part and the lagged q errors are for the moving average part, which

are all differenced. The prediction is the differenced y_t in the d^{th} order. This is called the ARIMA(p,d,q) model. Estimating the coefficients α and θ for a given p,d,q is what ARIMA does when it learns from the training data in a time series. Specifying p,d,q can be tricky (and a key limitation) but one can tryout different combinations and evaluate the performance of the model. Once the ARIMA model is specified with the value of p,d,q, the coefficients of Eq. (12.23) need to be estimated. The most common way to estimate is through the Maximum Likelihood Estimation. It is similar to the Least Square Estimation for the regression equation, except MLE finds the coefficients of the model in such a way that it maximizes the chances of finding the actual data.

ARIMA is a generalized model. Some of the models discussed in this chapter are special cases of an ARIMA model. For example,

- ARIMA (0,1,0) is expressed as $y_t = y_{t-1} + e$. It is the naive model with error, which is called the Random walk model.
- ARIMA (0,1,0) is expressed as $y_t = y_{t-1} + e + c$. It is a random walk model with a constant trend. It is called random walk with drift.
- ARIMA (0,0,0) is $y_t = e$ or white noise
- ARIMA (p,0,0) is the autoregressive model

How to Implement

The dataset used to build an ARIMA trainer is the Australian beer production dataset. The task is to forecast the production for ten more quarters. RapidMiner provides an extension (Time Series Extension) for time series operators to describe and predict time series. ARIMA modeling and the forecasting are implemented by *ARIMA trainer* to build the model and *Apply forecast* to apply the model and forecast for ten more quarters. The process is shown in Fig. 12.22

Step 1: The Australian beer production dataset has two attributes: Year and beer production value in megaliters. The role of Year should be set as ID and the only attribute in the dataset should be the time series value. A sequential time period is generated and added to the dataset as an anchor for the charts.

Step 2: The *ARIMA trainer* operator can be found under Extension > Time Series > Forecast > ARIMA. The operator has these parameters:

- p: order of autoregression part. It is set as 1
- d: degree of differencing. Set as 0
- q: order of the moving average of the error part. Set as 1
- criterion: criterion to minimize for coefficient selection. Set as aic.

As a start, the model parameters are specified as ARIMA(1,0,1). One can further optimize the value of p,d,q using the *Optimize Parameters* operator where

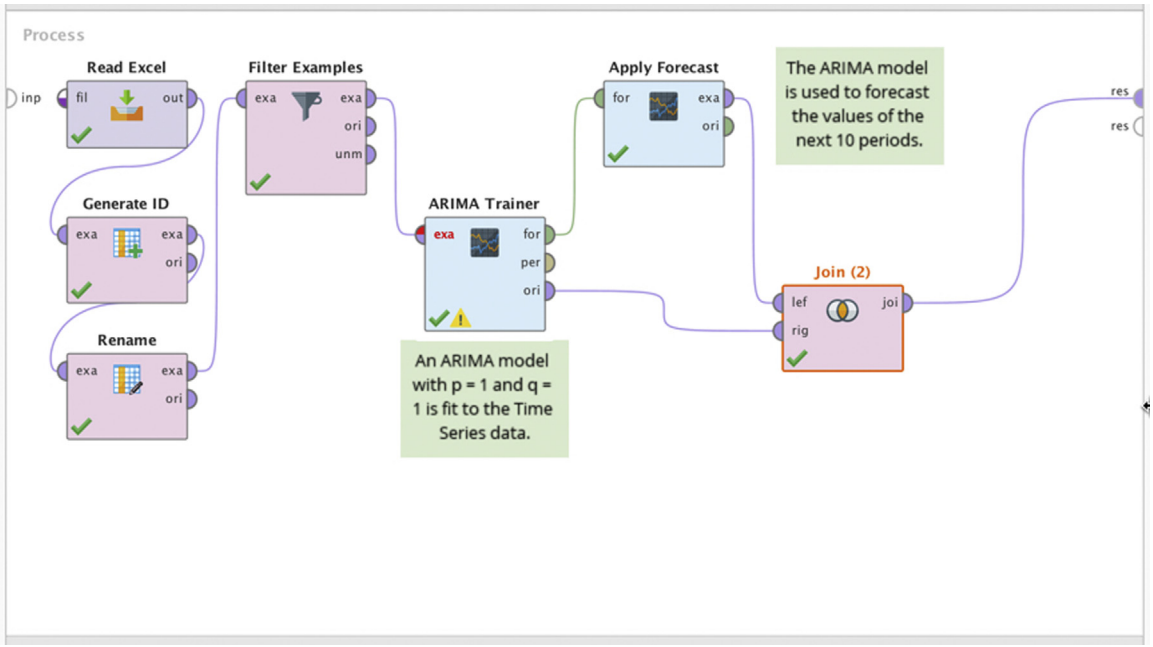


FIGURE 12.22
Process to implement ARIMA model. *ARIMA*, Autoregressive Integrated Moving Average.

all the combination of *p,d,q* are tried out to select the most optimal combination.

Step 3: The *Apply the forecast* operator takes the input of the forecasted model and applies the model for future horizon. Forecast horizon is set as 10 to predict 10 more data points at the end of the time series. The forecasted data is then joined with the original dataset using the *Join* operator so the forecasted data can be visualized along with the original dataset (Fig. 12.23).

The process shown in Fig. 12.22 can be saved and run. The result window has the output of the *Join* operator. The time series chart provides the plot of time series, both the original and forecasted value. The model has done a decent job of capturing the level and trend in the data, but the forecasted value does not have any seasonality, unlike the actual data. This is because seasonality was not expressed in the model. The ARIMA model discussed thus far does not take into account of the seasonality component. One would have a few additional steps to complete in order to incorporate the seasonality in the ARIMA modeling.

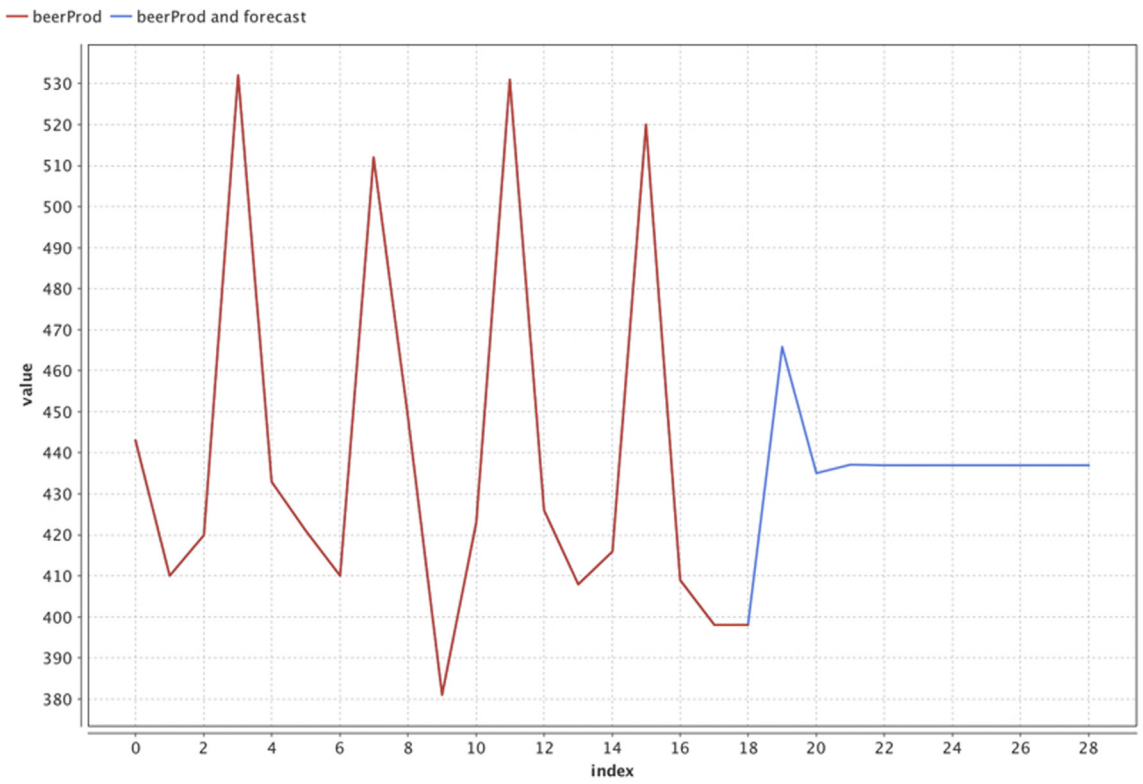


FIGURE 12.23
Forecast using ARIMA. *ARIMA*, Autoregressive Integrated Moving Average.

12.3.4 Seasonal ARIMA

The ARIMA model can be further enhanced to take into account of the seasonality in the time series. Seasonal ARIMA is expressed by the notion ARIMA$(p,d,q)(P,D,Q)_m$ where

p is the order of nonseasonal autoregression
d is the degree of differencing
q is the order of nonseasonal moving average of the error
P is the order of the seasonal autoregression
D is the degree of seasonal differencing
Q is the order of seasonal moving average of the error
m is the number of observations in the year (for yearly seasonality)

In terms of the equation, the seasonal part of ARIMA is similar to the terms used in the nonseasonal part, except that it is *backshifted m times*, where m is

the seasonal period. For example, the differencing is performed not with the consecutive data points but with the data points with m lags. If one has quarterly seasonality in monthly grain, each data point is differenced with the same month last quarter.

How to Implement

To implement seasonal ARIMA, *Execute R* operator from the *R* extension for RapidMiner is used. The RapidMiner process shown in Fig. 12.24 looks similar to the process built for the Holt-Winters' smoothing model. The difference is in the R code inside the *Execute R* operator. The process has a data preparation step before feeding data to *R* and a post process to combine the forecast and original data for visualization. The Australian beer production dataset is used for this process.

The R code inside the *Execute R* operator is shown here. It uses the Forecast package in R to activate the arima() function. In this processes the auto.arima() function was used, which automatically selects the best parameters for $(p,d,q)(P,D,Q)_m$. The optimal parameters for the Beer production dataset is ARIMA$(1,0,0)(1,1,0)_4$. The seasonal ARIMA model is used to forecast the future 12 data points using the forecast() function.

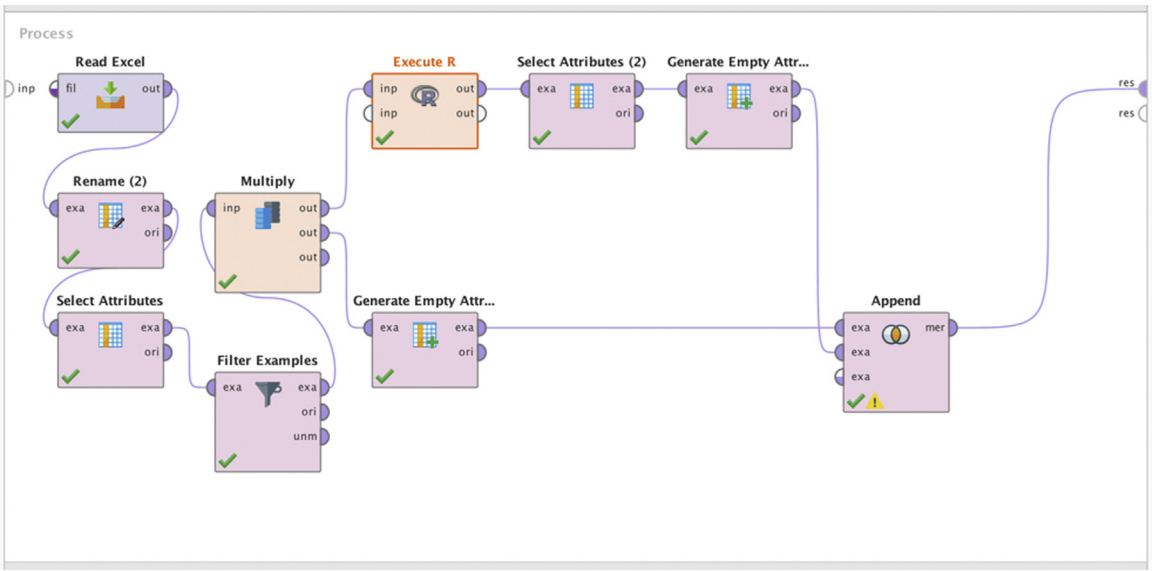


FIGURE 12.24

Seasonal ARIMA process using R. *ARIMA*, Autoregressive Integrated Moving Average.

```
rm_main = function(data)
{
library(forecast)
  inp <- ts(data, freq = 4)
  y <- auto.arima(inp)
    df <- as.data.frame(forecast(y, h = 12))
    return(list(df))
}
```

The process can be saved and executed. The forecast output is shown in Fig. 12.25. The model seems to have accounted for both the trend and seasonality in the time series. ARIMA is one of the most used forecasting techniques in businesses today. It has strong underpinning in statistics, has been field tested for decades, and is effective in modeling seasonal and

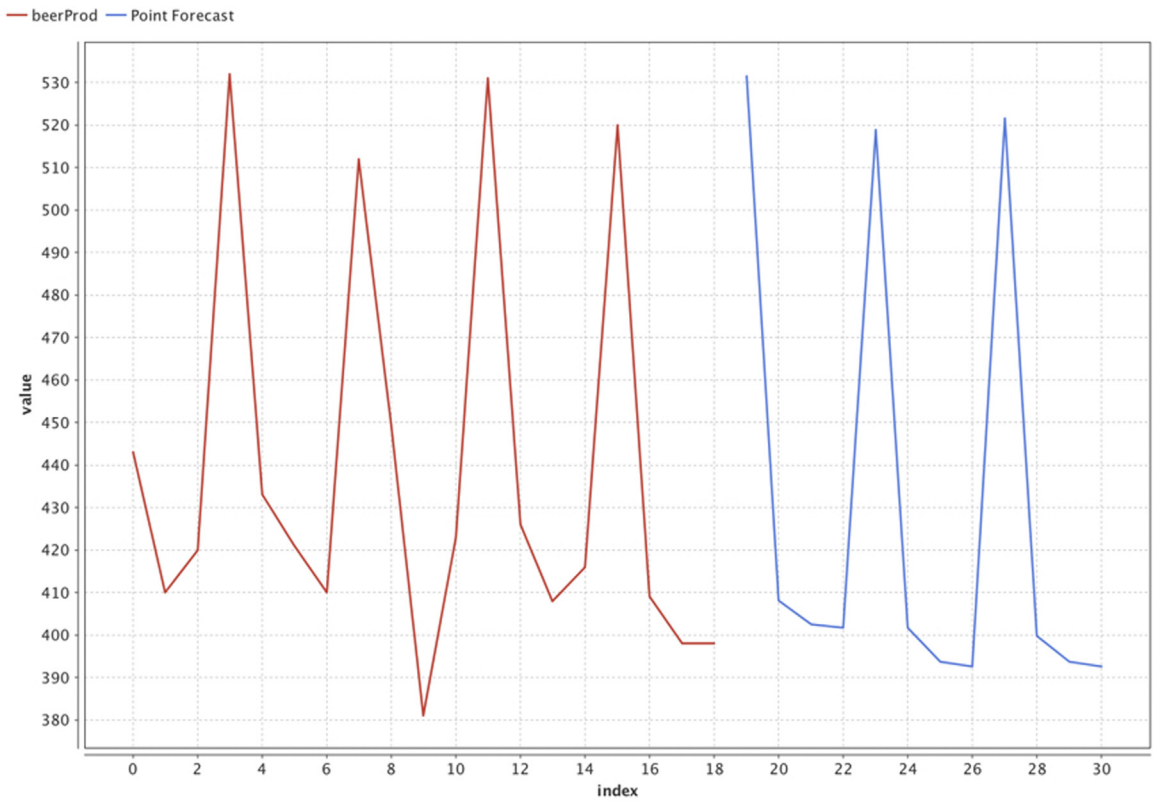


FIGURE 12.25
Forecast using seasonal ARIMA. *ARIMA*, Autoregressive Integrated Moving Average.

nonseasonal time series. Specifying the ARIMA modeling parameters beforehand might seem arbitrary. However, one can test the fit of the model for many combination or use meta functions like auto.arima() for the *Optimize parameters* operators. There is one more class of time series forecasting that has gained contemporary popularity—Machine learning–based time series forecasting.

12.4 MACHINE LEARNING METHODS

A time series is a unique dataset where the information used to predict future data points can be extracted from past data points. A subset of the past known data points can be used as inputs to an inferred *model* to compute the future data point as an output. Fig. 12.26 shows the general framework of the approach, which is similar to supervised learning techniques. Standard

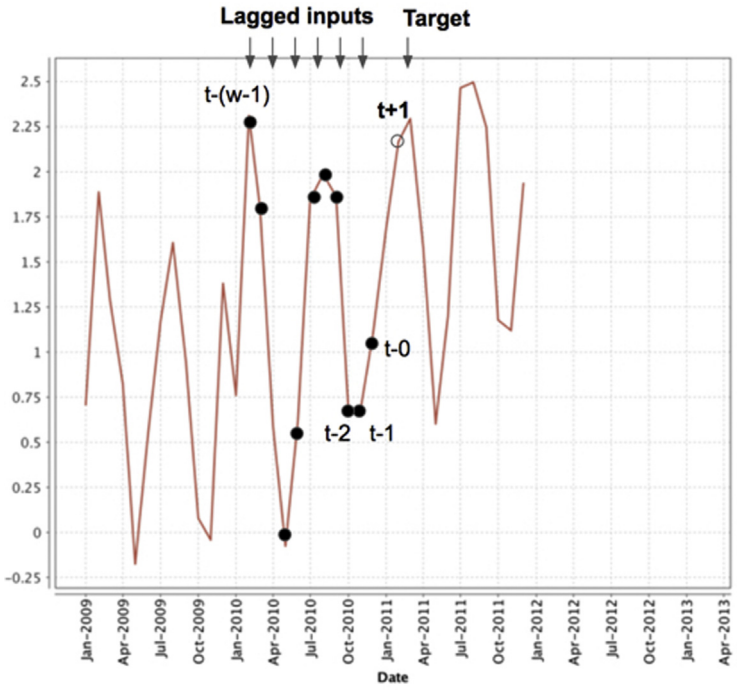


FIGURE 12.26
Lagged inputs and target.

machine learning techniques are used to build a model based on the inferred relationship between input (past data) and target (future data).

In order to use the supervised learners on time series data, the series is transformed into cross-sectional data using a technique called *windowing*. This technique defines a set of consecutive time series data as a window, where the latest record forms the target while other series data points, which are lagged compared to the target, form the input variables (Fig. 12.27). As consecutive windows are defined, the same data point in the series may function as the target for one cross-sectional window and the input variable for other cross-sectional windows. Once a sufficient number of windows are extracted from the dataset, a supervised model can be learned based on the inferred relationship between the lagged input variables and the target variable. This is similar to an autoregressive model where past p data points are used to predict the next data point. Any of the supervised learners discussed in the Classification or Regression chapter can be applied to learn and predict the target variable—the next time step in the series. The inferred model can be used to predict a future time series data point based on the last window of the time series. This gives visibility into one future data point. The new predicted data point can be used to define a new window and predict one more data point into the future. This subroutine can be repeated until all future predictions are made.

12.4.1 Windowing

The purpose of windowing is to transform the time series data into a generic machine learning input dataset. Fig. 12.28 shows a sample windowing and cross-sectional data extraction from the time series dataset.

The characteristics of the windows and the cross-sectional data extractions are specified by the parameters of the windowing process. The following parameters of windowing allow for changing the size of the windows, the

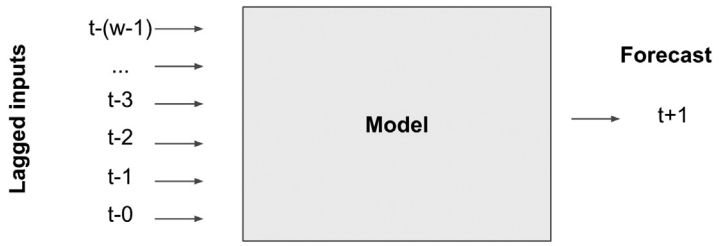
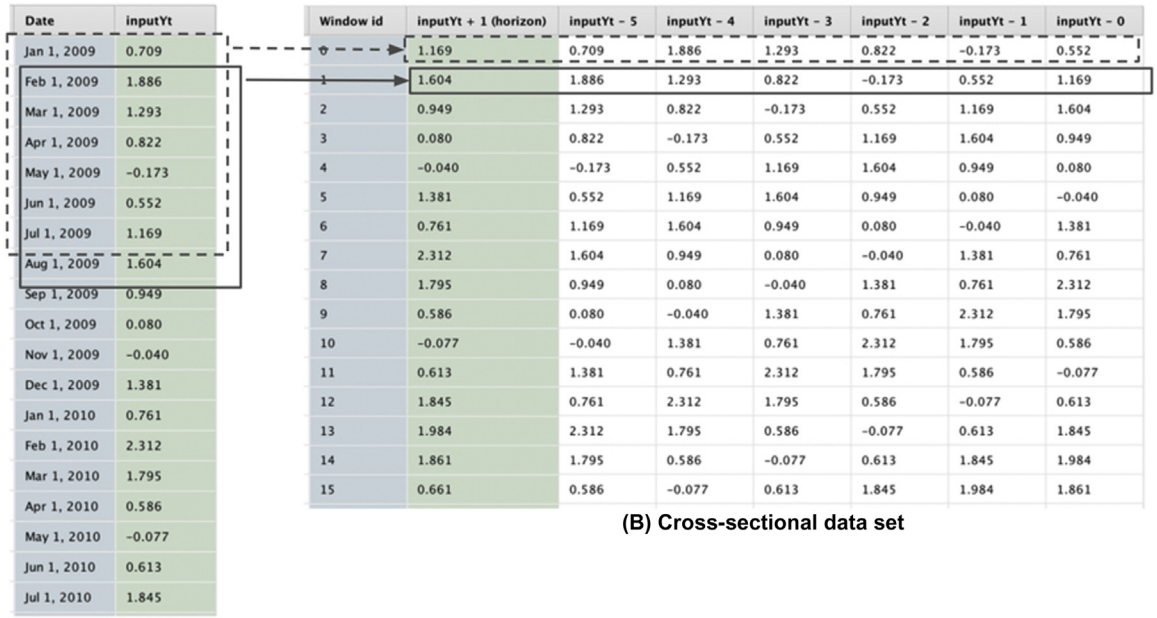


FIGURE 12.27
Machine learning model for time series.



Date	inputYt
Jan 1, 2009	0.709
Feb 1, 2009	1.886
Mar 1, 2009	1.293
Apr 1, 2009	0.822
May 1, 2009	-0.173
Jun 1, 2009	0.552
Jul 1, 2009	1.169
Aug 1, 2009	1.604
Sep 1, 2009	0.949
Oct 1, 2009	0.080
Nov 1, 2009	-0.040
Dec 1, 2009	1.381
Jan 1, 2010	0.761
Feb 1, 2010	2.312
Mar 1, 2010	1.795
Apr 1, 2010	0.586
May 1, 2010	-0.077
Jun 1, 2010	0.613
Jul 1, 2010	1.845

(A) Time series data set

Window id	inputYt + 1 (horizon)	inputYt - 5	inputYt - 4	inputYt - 3	inputYt - 2	inputYt - 1	inputYt - 0
0	1.169	0.709	1.886	1.293	0.822	-0.173	0.552
1	1.604	1.886	1.293	0.822	-0.173	0.552	1.169
2	0.949	1.293	0.822	-0.173	0.552	1.169	1.604
3	0.080	0.822	-0.173	0.552	1.169	1.604	0.949
4	-0.040	-0.173	0.552	1.169	1.604	0.949	0.080
5	1.381	0.552	1.169	1.604	0.949	0.080	-0.040
6	0.761	1.169	1.604	0.949	0.080	-0.040	1.381
7	2.312	1.604	0.949	0.080	-0.040	1.381	0.761
8	1.795	0.949	0.080	-0.040	1.381	0.761	2.312
9	0.586	0.080	-0.040	1.381	0.761	2.312	1.795
10	-0.077	-0.040	1.381	0.761	2.312	1.795	0.586
11	0.613	1.381	0.761	2.312	1.795	0.586	-0.077
12	1.845	0.761	2.312	1.795	0.586	-0.077	0.613
13	1.984	2.312	1.795	0.586	-0.077	0.613	1.845
14	1.861	1.795	0.586	-0.077	0.613	1.845	1.984
15	0.661	0.586	-0.077	0.613	1.845	1.984	1.861

(B) Cross-sectional data set

FIGURE 12.28

Windowing process. (A) original time series and (B) cross-sectional data set with consecutive windows.

overlap between consecutive windows, and the prediction horizon which is used for forecasting.

1. *Window Size*: Number of lag points in one window excluding the target data point.
2. *Step*: Number of data points between the first value of the two consecutive windows. If the step is 1, maximum number of windows can be extracted from the time series dataset.
3. *Horizon width*: The prediction horizon controls how many records in the time series end up as the target variable. The common value for the horizon width is 1.
4. *Skip*: Offset between the window and horizon. If the skip is zero, the consecutive data point(s) from the window is used for horizon.

In Fig. 12.28, the window size is 6, step is 1, horizon width is 1, and skip is 0.

Thus, the series data are now converted into a generic cross-sectional dataset that can be predicted with learning algorithms like regression, neural networks, or support vector machines. Once the windowing process is done, then the real power of machine learning algorithms can be brought to bear on a time series dataset.

Model Training

Consider the time series dataset shown in Fig. 12.28A. The dataset refers to historical monthly profits from a product, from January 2009 to June 2012. Suppose the objective in this exercise is to develop profitability forecasts for the next 12 months. A linear regression model can be used to fit the cross-sectional dataset shown in Fig. 12.28B using the technique described in Chapter 5, Regression Methods. The model will be:

$$\begin{aligned}\text{input } Y_{t+1}(\text{label}) = {} & 0.493 \times \text{input } Y_{t-5} + 0.258 \times \text{input } Y_{t-4} + 0.107 \times \text{input } Y_{t-3} \\ & - 0.098 \times \text{input } Y_{t-2} - 0.073 \times \text{input } Y_{t-1} + 0.329 \times \text{input } Y_{t-0} + 0.135\end{aligned}$$

Training the model is quite straightforward. The inferred relationship between a data point in the time series with the previous six data points is established. In other words, if one knows six consecutive data points in a time series, they can use the model to predict the seventh unseen data point. Since a new data point has been forecasted, it can be used along with the five preceding data points to predict one more data point and so on. That's time series forecasting one data point at a time!

How to Implement

Implementing a time series forecasting process using supervised learning is similar to a classification or regression process. The distinguishing step in time series forecasting is the conversion of a time series dataset to a cross-sectional dataset and stacking the forecast one data point at a time. The RapidMiner process is shown in Fig. 12.29. It uses operators from the Time Series extension. Although the process looks complicated, it consists of three functional blocks: (1) conversion to cross-sectional data, (2) training an machine learning model, and (3) forecasting one data point at a time in a loop. The dataset used in the process is the Product profit[5] dataset (the dataset can be downloaded from www.IntroDataScience.com) shown in Fig. 12.28. The time series has two attributes: Date and Input Y_t.

Step 1: Set Up Windowing

The process window in Fig. 12.29 shows the necessary operators for windowing. The time series dataset has a date column, and this must be treated with special care. The operator must be informed that one of the columns in the dataset is a date and should be considered as an "id." This is accomplished with the *Set Role* operator. If the input data has multiple time series, *Select Attributes* operator can be used to select the one to be forecasted. In this case, only a one value series is used and strictly speaking this operator is not needed. However, to make the process generic it has been included and the

[5] Available for download from www.IntroDataScience.com.

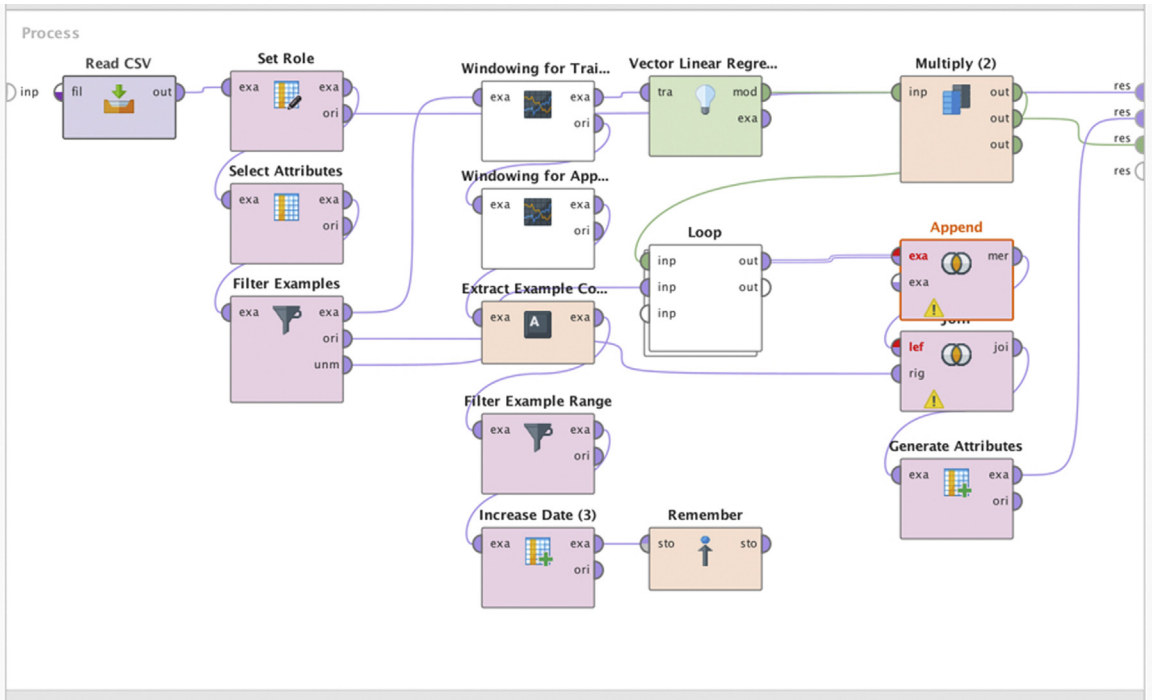


FIGURE 12.29
Process for time series forecasting using machine learning.

column labeled "inputYt" has been selected. Optionally, one may want to use the *Filter Examples* operator to remove any data points that may have missing values. The central operator for this step is the *Windowing* operator in the Time series extension. The main parameters for the *Windowing* operator are:

1. *Window size:* Determines how many "attributes" are created for the cross-sectional data. Each row of the original time series within the window size will become a new attribute. In this example, $w = 6$ was chosen.
2. *Step size:* Determines how to advance the window. $s = 1$ was used.
3. *Horizon width:* Determines how far out to make the forecast. If the window size is 6 and the horizon is 1, then the seventh row of the original time series becomes the first sample for the "label" variable. $h = 1$ was used.

Fig. 12.28 shows the original data and the transformed output from the windowing process. The window operator adds six new attributes named input Y_{t-5} through input Y_{t-0}.

Step 2: Train the Model

When training any supervised model using this data, the attributes labeled input Y_{t-5} through input Y_{t-0} form the independent variables. In this case, *linear regression* is used to fit the dependent variable called label, using the independent variables input Y_{t-5} through input Y_{t-0}. The *Vector Linear Regression* operator is being used to infer the relationship between six dependent variables and the dependent variable. The model output for the dataset is:

$$\text{label} = 0.493 \times \text{input } Y_{t-5} + 0.258 \times \text{input } Y_{t-4} + 0.107 \times \text{input } Y_{t-3} - 0.098 \times \text{input } Y_{t-2} - 0.073 \times \text{input } Y_{t-1} + 0.329 \times \text{input } Y_{t-0} + 0.135$$

Step 3: Generate the Forecast in a Loop

Once the model fitting is done, the next step is to start the forecasting process. Note that given this configuration of the window size and horizon, one can now only make the forecast for the next step. In the example, the last row of the transformed dataset corresponds to December 2011. The independent variables are values from June–November 2011 and the target or label variable is December 2011. The regression equation is be used to predict December 2011 value. The same regression equation is also used for predicting January 2012 value. All one needs to do is insert the values from July–December into the regression equation to generate the January 2012 forecast. Next, a new row of data needs to be generated that would run from August–*January* to predict February 2012 using the regression equation. All the (actual) data from August–December is available as well as the predicted value for January. Once the predicted February value is obtained, there is nothing preventing the actual data from September–December plus the predicted January and February values from being used to forecast March 2012.

To implement this in RapidMiner process, one would need to break this up into two separate parts. First, take the last forecasted row (in this case, December 2011), drop the current value of input Y_{t-5} (current value is 1.201), rename input Y_{t-4} to input Y_{t-5}, rename input Y_{t-3} to input Y_{t-4}, rename input Y_{t-2} to input Y_{t-3}, rename input Y_{t-1} to input Y_{t-2}, rename input Y_{t-0} to input Y_{t-1}, and finally rename predicted *label* (current value is 1.934) to input Y_{t-0}. With this new row of data, the regression model can be applied to predict the next date in the series: January 2012. Fig. 12.30 shows the sample steps. Next, this entire process need to be put inside a *Loop* operator that will allow these steps to be repeatedly run for as many future periods as needed.

The *Loop* operator will contain all the mechanisms for accomplishing the renaming and, of course, to perform the forecasting (Fig. 12.31). Set the *iterations* in the *Loop* operator to the number of future months to forecast

Date	prediction(l...	inputYt-5	inputYt-4	inputYt-3	inputYt-2	inputYt-1	inputYt-0
Dec 1, 2011...	1.694	1.201	2.466	2.497	2.245	1.179	1.119
Jan 1, 2012 ...	2.597	2.466	2.497	2.245	1.179	1.119	1.694
Feb 1, 2012...	2.693	2.497	2.245	1.179	1.119	1.694	2.597
Mar 1, 2012...	2.196	2.245	1.179	1.119	1.694	2.597	2.693
Apr 1, 2012...	1.457	1.179	1.119	1.694	2.597	2.693	2.196
May 1, 201...	1.457	1.119	1.694	2.597	2.693	2.196	1.457
Jun 1, 2012 ...	2.087	1.694	2.597	2.693	2.196	1.457	1.457
Jul 1, 2012 ...	2.784	2.597	2.693	2.196	1.457	1.457	2.087
Aug 1, 2012...	2.807	2.693	2.196	1.457	1.457	2.087	2.784
Sep 1, 2012...	2.265	2.196	1.457	1.457	2.087	2.784	2.807
Oct 1, 2012...	1.720	1.457	1.457	2.087	2.784	2.807	2.265
Nov 1, 2012...	1.816	1.457	2.087	2.784	2.807	2.265	1.720
Dec 1, 2012...	2.433	2.087	2.784	2.807	2.265	1.720	1.816
Jan 1, 2013 ...	2.974	2.784	2.807	2.265	1.720	1.816	2.433
Feb 1, 2013...	2.911	2.807	2.265	1.720	1.816	2.433	2.974

FIGURE 12.30
Forecasting one step ahead.

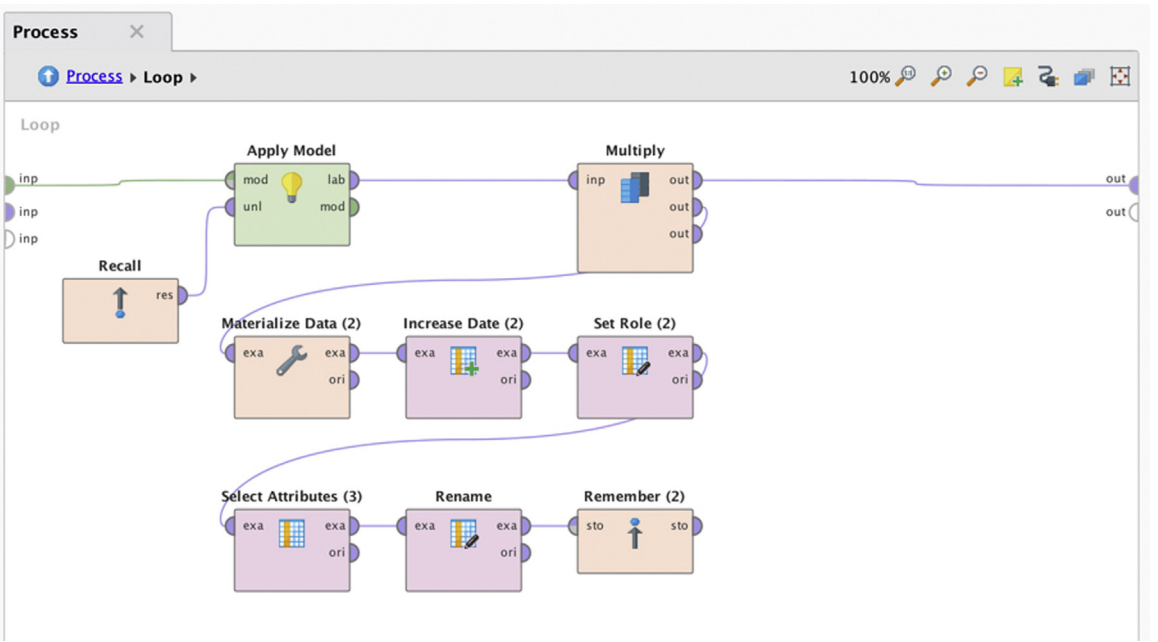


FIGURE 12.31
Looping subroutine to forecast one data point at a time.

(horizon). In this case, this is defined by a process variable called *futureMonths* whose value can be changed by the user before process execution. It is also possible to capture the Loop counts in a macro if the *set iteration macro* box is checked. A *macro* in RapidMiner is nothing but a process variable that can be called by other operators in the process. When *set iteration macro* is checked and a name is provided in the *macro name* box, a variable will be created with that name whose value will be updated each time, one loop is completed. An initial value for this macro is set by the *macro start value* option. Loops may be terminated by specifying a *timeout*, which is enabled by checking the *limit time* box. A macro variable can be used by any other operator by using the format %{macro name} in place of a numeric value.

Before the looping is started, the last forecasted row needs to be stored in a separate data structure. This is accomplished by a new *Windowing* operator and the macro titled *Extract Example Set*. The *Filter Example* operator simply deletes all rows of the transformed dataset except the last forecasted row. Finally the *Remember* operator stores this in memory and allows one to "recall" the stored value once inside the loop.

The loop parameter *iterations* will determine the number of times the inner process is repeated. Fig. 12.31 shows that during each iteration, the model is applied on the last forecasted row, and bookkeeping operations are performed to prepare application of the model to forecast the next month. This includes incrementing the month (date) by one, changing the role of the predicted label to that of a regular attribute, and finally renaming all the attributes. The newly renamed data are stored and then recalled before the next iteration begins.

The output of this process is shown in Fig. 12.32 as an overlay on top of the actual data. As seen, the simple linear regression model seems to adequately capture both the trend and seasonality of the underlying data. The *Linear Regression* operator of *Step 2: Train the model* can be quickly swapped to a *Support Vector Machine* operator and its performance tested without having to do any other programming or process modifications.

12.4.2 Neural Network Autoregressive

The windowing process allows time series dataset to be transformed to a cross-sectional horizontal dataset that is conducive to learners to create a forecasting model. That includes artificial neural networks to forecast the time series. Consider a simple ANN architecture with six inputs, one hidden layer with five nodes and an output, as shown in Fig. 12.33. It is multilayer feed forward network that maps nonlinear functions.

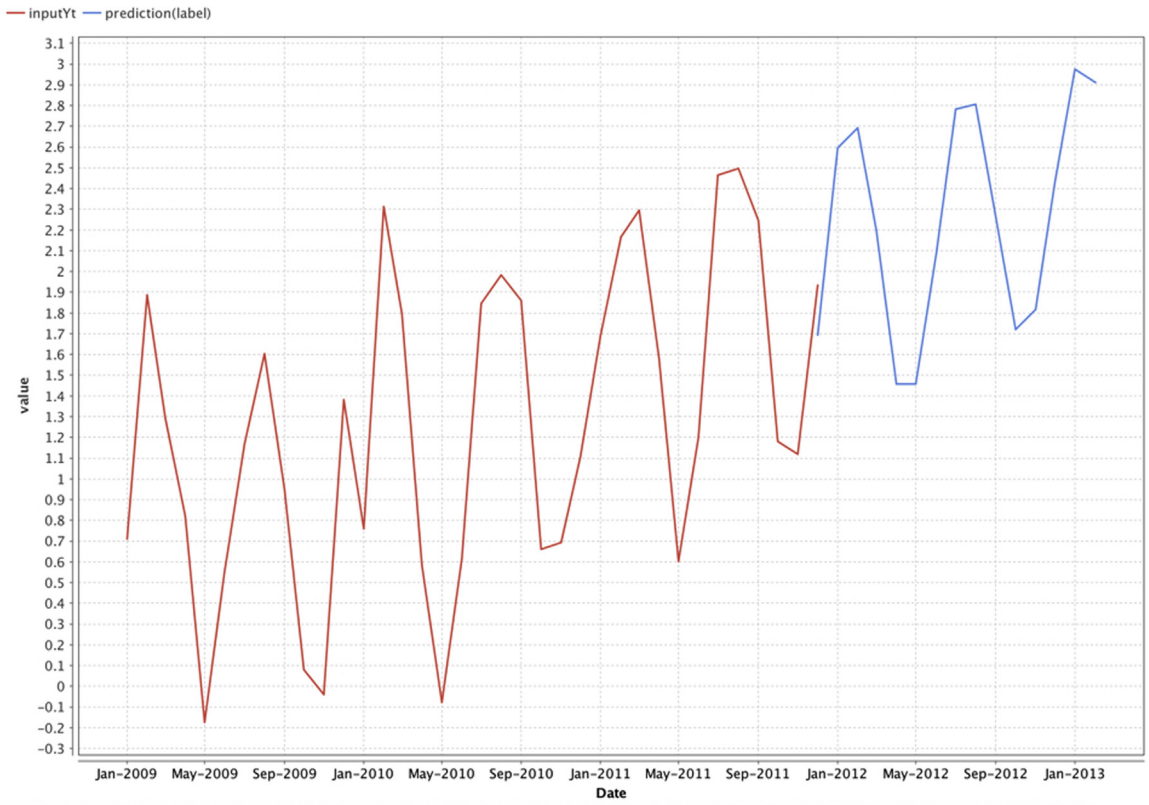


FIGURE 12.32
Forecasted time series.

Time series data, which is transformed into a cross-sectional dataset, can be fed into a neural network as inputs to predict the output. The weights for the links can be estimated with the training dataset, with lagged inputs and label, to build the neural network model. A feed forward neural network with one hidden layer, in the context of Time series is denoted as Neural Network Autoregressive—NNAR$(p,P,k)_m$ where p is the number of lagged inputs (order of the autoregressive model), k is the number of nodes in the hidden layer, P is the auto regressive part of the seasonal component and m is the seasonal period. The neural network NNAR$(p,P,k)_m$ has a particular relevance when it comes to time series forecasting. A NNAR$(p,P,k)_m$ functions *similarly* to a seasonal ARIMA $(p,0,0)$ $(P,0,0)_m$ model (Hyndman & Athanasopoulos, 2018).

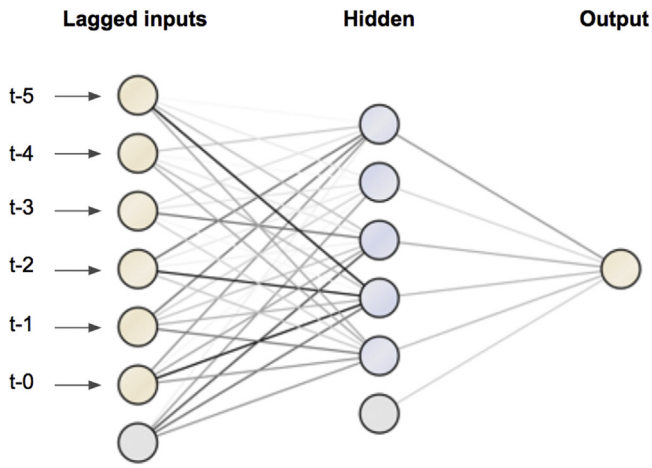


FIGURE 12.33
Neural network autoregressive model.

How to Implement

The implementation of NNAR is the same as the previous Windowing process shown in Fig. 12.29 with the *Neural Net* operator replacing the *Vector Linear Regression* modeling operator. The neural network operator has a parameter called *Hidden layer* where one can specify the number of hidden layers and nodes within each layer. For the NNAR model, the number of hidden layers is 1 and the number of nodes is one less than the number of the input nodes (Hyndman & Athanasopoulos, 2018). Fig. 12.34 shows the Rapidminer process for NNAR. This process has six lagged inputs and the number of nodes in the hidden layer is 5.

The forecasting is done one step ahead. The *Loop* operator takes the latest output and applies it to the input for the next iteration. The process can be saved and executed. The time series forecast is shown in Fig. 12.35. It can be observed that the NNAR forecast is different from the forecast achieved with the linear regression model.

An important point about any time series forecasting is that one should not place too much emphasis on the point forecasts. A complex quantity like sales demand for a product is influenced by too many factors and to claim that any forecasting will predict the exact value of demand three months in advance is unrealistic. However, what is far more valuable is the fact that recent undulations in the demand can be effectively captured and predicted.

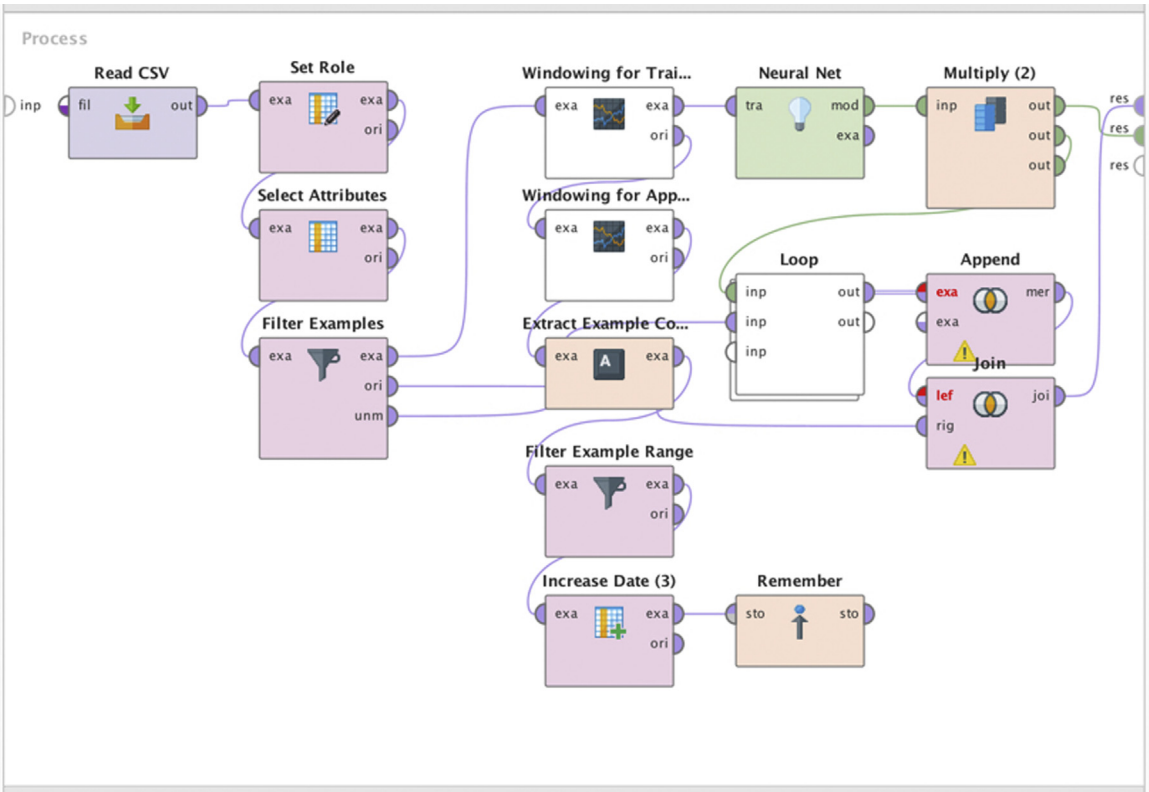


FIGURE 12.34
Process for neural network autoregressive model.

12.5 PERFORMANCE EVALUATION

How does one know if the time series forecasting model provides accurate forecasts? forecast accuracy metrics have to be created to compare the performance of one model with another and to compare the models across different use cases. For example, the performance of a revenue forecasting model can be contrasted with other models built by different techniques or different parameters; and even to compare with models to forecast the customer complaints for a product.

12.5.1 Validation Dataset

One way to measure accuracy is to measure the actual value post-fact and compare against the forecast and calculate the error. However, one would

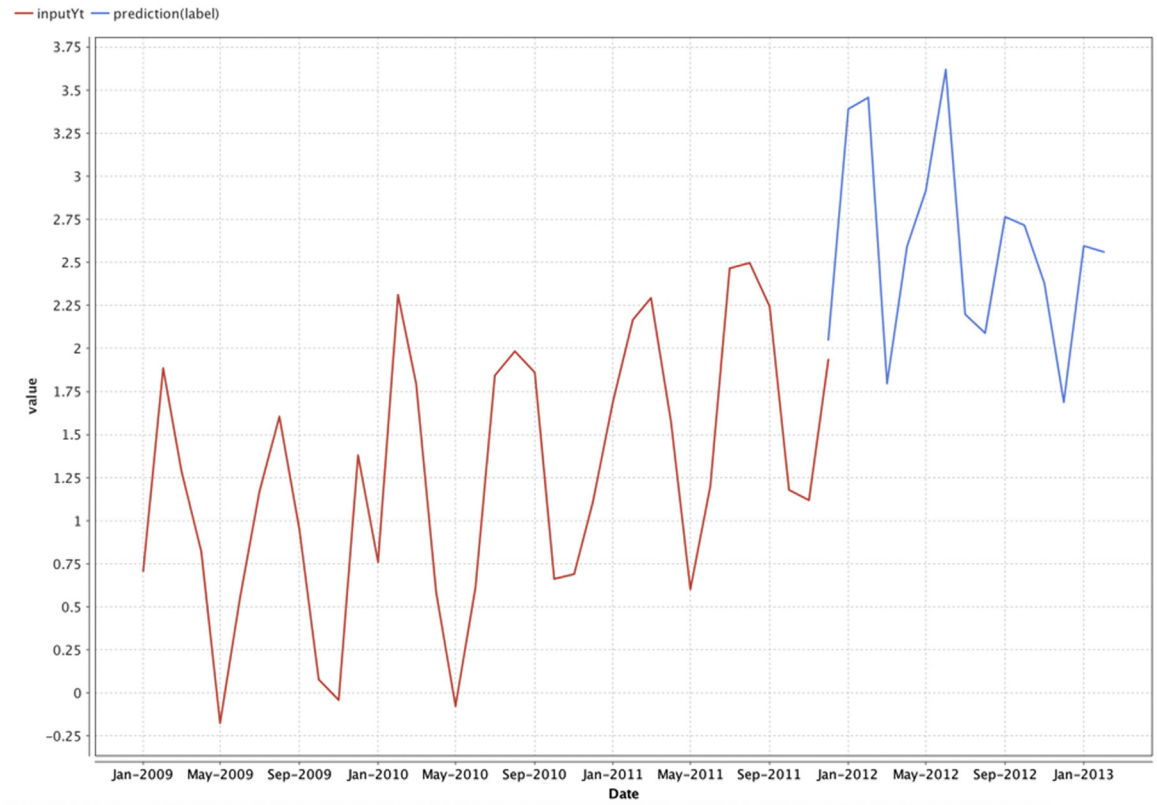


FIGURE 12.35
Forecast using neural network autoregression.

have to wait for the time to pass and the actual data to be measured. Residues are calculated when a forecast model is fitted on training data. By all means, the model might have overfitted the training data and perform poorly with unseen future forecasts. For this reason, training residuals are not a good way to measure how the forecast models are going to perform in the real world. Recall that in the case of the supervised learners, a set of known data is reserved for model validation. Similarly, some data points can be reserved in the time series as a *validation dataset* just to test the accuracy of the model. The training process uses data by restricting it to a point and the rest of the later time series is used for validation as shown in Fig. 12.36. As the model has not seen the validation dataset, deviation of actuals from the forecast is the forecast error of the model. The forecast error is the difference

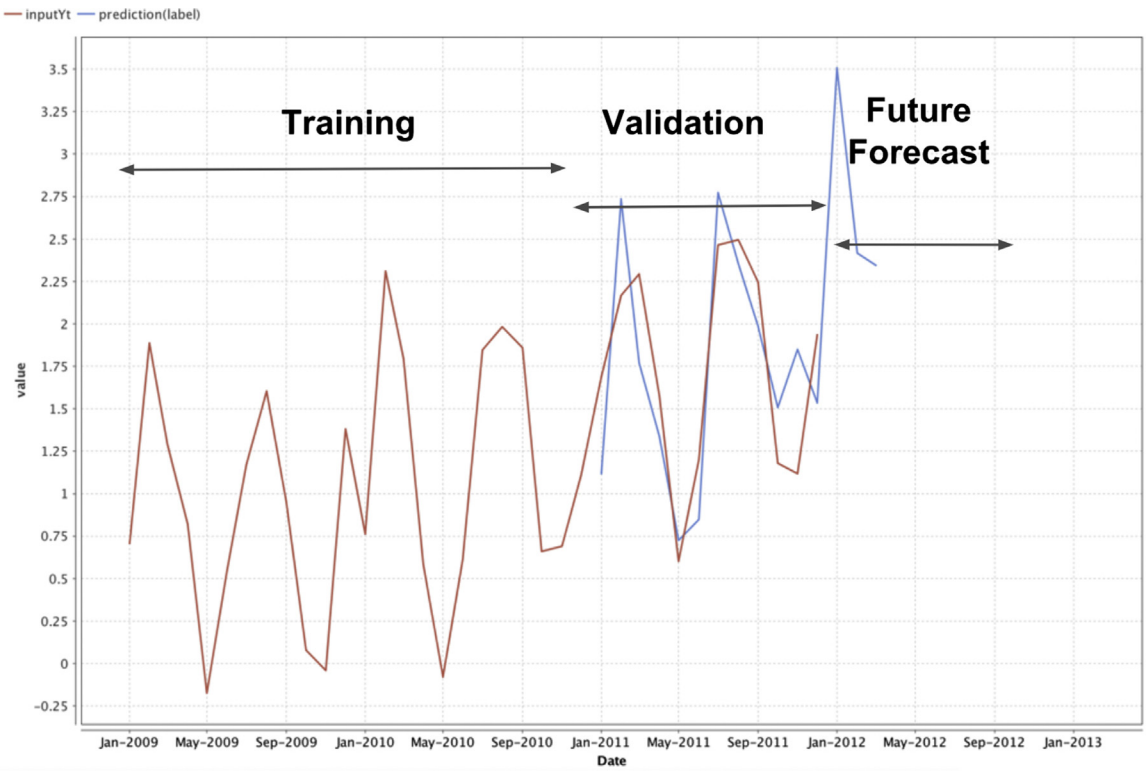


FIGURE 12.36
Validation dataset.

between the actual value $\mathbf{y_i}$ and the forecasted value $\hat{\mathbf{y}}_\mathbf{i}$. The error or the residue for the i^{th} data point is given by Eq. (12.24)

$$e_i = y_i - \hat{y}_i \tag{12.24}$$

The forecast error shown in Eq. (12.24) is scale dependent. The error measured for each of the data points and can be aggregated to one metric to indicate the error of the forecasting model. Some of the commonly used forecast accuracy aggregate metrics are:

Mean Absolute Error

The error of the individual data point may be positive or negative and may cancel each other out. To derive the overall forecast for the model, calculate the absolute error to aggregate all the residuals and average it.

$$\text{Mean absolute error} = \text{mean}(|e_i|) \tag{12.25}$$

MAE is a simple metric and it is scale dependent. It is convenient to communicate the error of the revenue forecasting model as, for example, ± $900,000 per day.

Root Mean Squared Error

In some cases it is advantageous to penalize the individual point error with higher residues. Even though two models have the same MAE, one might have consistent error and the other might have low errors for some points and high error for other points. RMSE penalizes the latter.

$$\text{Root mean squared error} = \sqrt{\text{mean}\,(e^2)} \tag{12.26}$$

RMSE is scale dependent and is used in situations where penalizing high relative residue is necessary. On the other hand, it is slightly difficult to understand the RMSE for a stand-alone model.

Mean Absolute Percentage Error

Percentage error of a data point is $p_{i=}100\ \frac{e_i}{y_i}$. It is a scale independent error that can be aggregated to form mean absolute percentage error.

$$\text{Mean absolute percentage error} = \text{mean}\left(\left|p_i\right|\right) \tag{12.27}$$

MAPE is useful to compare against multiple models across the different forecasting applications. For example, the quarterly revenue forecast, measured in USD, for a car brand might be ± 5% and the forecast for world-wide car demand, measured in quantity, might be ± 3%. The firm's ability to forecast the car demand is higher than the revenue forecast for one brand. Even though MAPE is easy to understand and scale independent, MAPE has a significant limitation when it applies to intermittent data where zero values are possible in actual time series. For example, profit or defects in a product. Zero value in the time series yields an infinite error rate (if the forecast is nonzero) and skews the result. MAPE is also meaningless when the zero point is not defined or arbitrarily defined, as in non-kelvin temperature scales.

Mean Absolute Scaled Error

MASE is scale independent and overcomes the key limitations of MAPE by comparing the forecast values against a naive forecast. Naive forecast is a simple forecast where the next data point has the same value as the previous data point (Hyndman & Koehler, 2006). Scaled error is defined as:

$$\text{MASE} = \frac{\sum_{i=1}^{T} |e|}{\frac{T}{T-1}\sum_{i=2}^{T} |\overline{y}_i - \overline{y}_{i-1}|} \tag{12.28}$$

T is the total number of data points. Scaled error is less than one if the forecast is better than naive forecast and greater than one if it is worse than naive

forecast. One would want a scaled error much less than one for a good forecasting model.

12.5.2 Sliding Window Validation

Sliding window validation is a process of backtesting time series models built through machine learning based methods. The whole cross-sectional dataset is divided into different training *windows* by specifying the window width. A model is trained using a training window and applied on the testing window to compute the performance for the first run. For the next run, the training window is slid to new set of training records and the process is repeated until all the training windows are used. By this technique, an average performance metric can be calculated across the entire dataset. The performance metric derived through sliding window validation is generally more robust than split validation technique.

12.6 CONCLUSION

Time series forecasting remains one of the cornerstones of data science techniques. It is one of the most widely used analytical applications in business and organizations. All organizations are forward looking and want to plan for the future. Time series forecasting, thus, forms a lynchpin to look into the *most probable* future and plan accordingly. Time series forecasting, like any other data science technique, has a diverse set of techniques and methods. This chapter covered the most important techniques that have practical relevance in a business setting.

Univariate time series forecasting treats prediction, essentially, as a single-variable problem, whereas, multivariate time series may use many time-concurred value series for prediction. If one has a series of points spaced over time, conventional forecasting uses smoothing and averaging to *predict* where the next few points will likely be. However, for complex systems such as the economy or demand of a product, point forecasts are unreliable because these systems are functions of hundreds if not thousands of variables. What is more valuable or useful is the ability to predict trends, rather than point forecasts. Trends can be predicted with greater confidence and reliability (i.e., Are the quantities going to trend up or down?), rather than the values or levels of these quantities. For this reason, using an ensemble of different modeling schemes such as artificial neural networks or support vector machines or polynomial regression can sometimes give highly accurate trend forecasts. If the time series is not highly volatile (and, therefore, more predictable), time series forecasting can help understand the underlying structure of the variability better. In such cases, trends or seasonal components have a stronger signature than random components do.

12.6.1 Forecasting Best Practices

While the science part of time series forecasting is covered in this chapter, there is a bit of *art* in getting robust forecasts from the time series models. Here is a list of suggested practices to build a robust forecasting model.

1. *Understand the metric*: Investigate how the time series metric is derived. Is the metric influenced by other metrics or phenomenon that can be better candidates for the forecasting? For example, instead of forecasting profit, both revenue and cost can be forecasted, and profit can be calculated. This is particularly suitable when profit margins are low and can go back and forth between positive and negative values (loss).
2. *Plot the time series*: A simple time series line chart reveals a wealth of information about the metric being investigated. Does the time series have a seasonal pattern? Long-term trends? Are the seasonality and trend linear or exponential? Is the series stationary? If the trend is exponential, can one derive log() series? Aggregate daily data to weeks and months to see the normalized trends.
3. *Is it forecastable*: Check if the time series is forecastable using stationary checks.
4. *Decompose*: Identify trends and seasonality using decomposition methods. These techniques show how the time series can be split into multiple meaningful components.
5. *Try them all*: Try several different methods mentioned in the forecasting taxonomy in Fig. 12.3, after splitting, training, and validation samples. For each method:
   a. Perform residual checks using MAE or MAPE metric.
   b. Evaluate forecasts using the validation period.
   c. Select the best performing method and parameters using optimization functions.
   d. Update the model using the full dataset (training + validation) for future forecasts.
6. *Maintain the models*: Review models on a regular basis. Time series forecast models have a limited shelf life. Apart from feeding the latest data to the model, the model should be refreshed to make it relevant for the latest data. Building a model daily is not uncommon.

References

Box, G. A. (1970). *Time series analysis: Forecasting and control*. San Francisco, CA: Holding Day.

Box, G. J. (2008). *Time series analysis: Forecasting and control*. Wiley Series in Probability and Statistics.

Brown, R. G. (1956). *Exponential smoothing for predicting demand*. Cambridge, MA: Arthur D. Little.

Hyndman R.A. (2014). *Forecasting: Principles and practice*. <Otexts.org>.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice*. 2nd edition. <Otexts.org>.

Hyndman, Rob J., & Koehler, Anne B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting, 22*, 679–688.

Shmueli G. (2011). *Practical time series forecasting: A hands on guide*. <statistics.com>.

Winters, P. (1960). Forecasting sales by exponentially weighted moving averages. *Management Science, 6*(3), 324–342.