Time Series Forecasting Models

Matthew Juan z5259434

Contents

1 Introduction to Time Series'

Time series are found all over the place and are extremely useful to analyse and more importantly, forecast. There are many situations where you want to predict past your last data point. Use cases include:

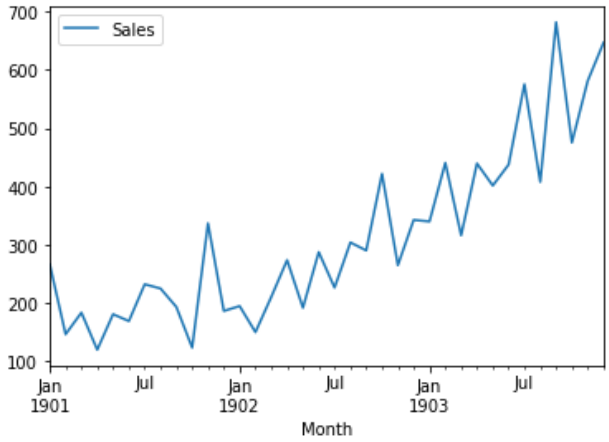
- Traders predicting the stock market so they know when to buy or sell
- Data centre operators identifying when a critical system may go down or is acting irregularly
- Retail store managers knowing how to maximise what stock they order to maximise profit and minimise items that won't sell

Unfortunatly, just like humans, it is impossible to predict the future. However, by analysing the time series, it is possible to formulate statistical properites that can be used to give a rough estimate of where the data is heading. This report will explore the mathematics behind the popular exponential smoothing and ARIMA forecasting models with an accompanying implementation in Python.

2 Non-Seasonal Forecasting Models

There are many types of forecasting models but this paper will look two types. Exponential Smoothing and ARIMA (Auto-Regressive Integrated Moving Average) models. For these non-seasonal models, we will use the Shampoo Sales dataset since there is no clear seasonality to the data.

Month	Sales
1901-01-01	266.0
1901-02-01	145.9
1901-03-01	183.1
1901-04-01	119.3
1901-05-01	180.3
...	...
1903-08-01	407.6
1903-09-01	682.0
1903-10-01	475.3
1903-11-01	581.3
1903-12-01	646.9



2.1 Exponential Smoothing

As the name suggests, Exponential Smoothing models are essentially weighted averages of past observations which decay exponentially over time. This type allows for a straightforward way to forecast for a wide range of time series'. This section of the paper will go into an implementation of simple exponential smoothing, Holt's linear trend method, and the damped trend method.

2.1.1 Simple Exponential Smoothing

This method of forecasting is extremely simple and is useful when there is no clear trend or seasonal pattern in the data. As mentioned, this method uses a smoothing parameter α which exponentially decreases the further in the past the associated observation is. Therefore, for a time series with T data points, we can define such an forecast equation as:

$$\hat{y}_{T+1} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \alpha(1-\alpha)^3 y_{T-3} + ...,$$

where $0 \leq \alpha \leq 1$ and $\hat{y}_{T+1}$ is the first forecasted value. This equation can be simplified by using the predicted value at time T to predict the value at time $T+1$ resulting in

$$\hat{y}_{T+1} = \alpha y_T + (1-\alpha)\hat{y}_T.$$

This equation can also be expressed in component form defined as a level

smoothing equation and forecasting equation:

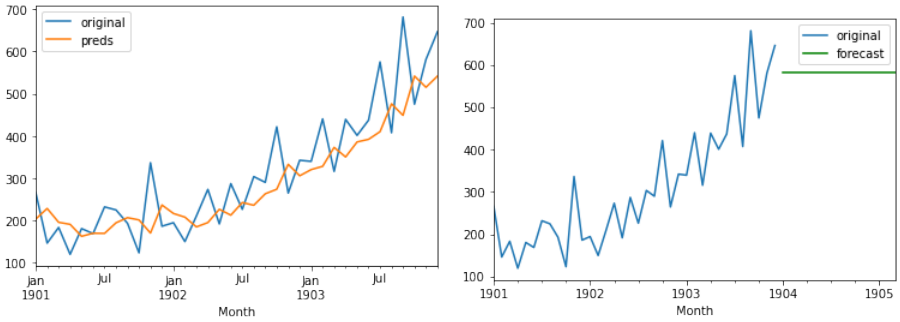
Level equation	$\ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1})$
Forecast equation	$\hat{y}_{t+h} = \ell_t$

By setting $h = 1$, we get the fitted values for the model. The forecasting equation produces a "flat" forecast equal to the last level component since setting $t = T$, we get

$$\hat{y}_{T+h} = \ell_T, \qquad h = 1, 2, ...$$

Let's see how this is applied to the shampoo dataset. Firstly, we need to estimate values for α as well as the initial level ℓ_0. By minimising the SSE (sum of squared errors), the estimated parameters are $\alpha = 0.39938593$ and $\ell_0 = 202.78$. The table below shows the calculations using these parameters.

Time(t)	Observation(y_t)	Level(ℓ_t)	Forecast($\hat{y}_t$)
0		202.78	
1	266.0	228.03	202.78
2	145.9	195.23	228.03
3	183.1	190.38	195.23
4	119.3	161.99	190.38
⋮	⋮	⋮	⋮
33	682.0	541.92	448.78
34	475.3	515.32	541.92
35	581.3	541.67	515.32
36	646.9	583.70	541.67
h			$\hat{y}_{T+h}$
1			583.70
2			583.70
3			583.70
4			583.70
5			583.70



(a) Fitted values

(b) Forecasted values

Figure 1: Fitted and forecasted values using simple exponential smoothing

2.1.2 Trend Methods

This section will show Holt's linear trend method and Gardner & McKenzie's damped trend method which extend simple exponential smoothing by incorporating the trend component. Trend is simply the general upwards or downwards motion of the data. This component helps to forecast in the right direction instead of just a flat line.

Holt's Linear Trend Method

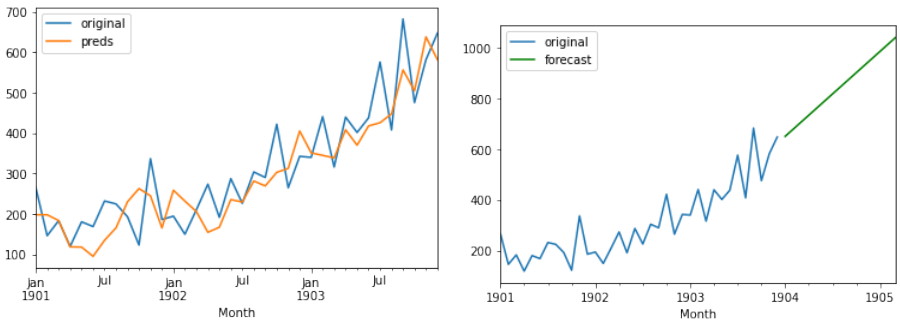
This method now has three equations, a level smoothing equation, a trend smoothing equation, and a forecast equation:

$$\begin{aligned} \text{Forecast equation} \quad & \hat{y}_{t+h|t} = \ell_t + hb_t \\ \text{Level equation} \quad & \ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\ \text{Trend equation} \quad & b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1} \end{aligned}$$

where $0 \leq \alpha, \beta \leq 1$. Similarly, setting $h = 1$ produces the fitted values of the model. By including the trend, forecasts are no longer flat but a linear function of h. As such, the forecasted value at time $T + h$ is equal to the last level plus h multiplied by the last calculated trend value.

There are now two smoothing parameters α and β as well as two initial states ℓ_0 and b_0 which are estimated to $\alpha = 0.48709065, \beta = 0.27116407, \ell_0 = 262.93, b_0 = -18.01$. Giving an initial trend of $y_1 - y_0$ can help the optimizer narrow the search space for this parameter.

Time(t)	Observation(y_t)	Level(ℓ_t)	Slope(b_t)	Forecast($\hat{y}_t$)
0		262.93	-18.01	
1	266.0	255.19	-15.23	244.92
2	145.9	194.14	-27.65	244.92
3	183.1	174.58	-25.46	239.96
4	119.3	134.60	-29.40	166.49
⋮	⋮	⋮	⋮	⋮
33	682.0	583.60	39.51	540.92
34	475.3	551.11	19.99	490.15
35	581.3	576.07	21.33	623.11
36	646.9	621.51	27.87	571.10
h				$\hat{y}_{T+h}$
1				649.38
2				677.25
3				705.12
4				732.99
5				760.86



(a) Fitted values

(b) Forecasted values

Figure 2: Fitted and forecasted values using Holt linear trend method

Damped Trend Method

Holt's Linear method show a constant increasing or decreasing trend which extends infinitely into the future. However, in reality, this method can be quite inaccurate for extended forecasts. Gardner & McKenzie introduced a dampening parameter ϕ, where $0 \leq \phi \leq 1$. This parameter dampens the trend to eventually forecast a flat line after some time.

The component form of this method is:

$$\begin{aligned}
\text{Forecast equation} \quad & \hat{y}_{t+h|t} = \ell_t + (\phi + \phi^2 + ... + \phi^h) b_t \\
\text{Level equation} \quad & \ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + \phi b_{t-1}) \\
\text{Trend equation} \quad & b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)\phi b_{t-1}.
\end{aligned}$$

When $\phi = 1$, this method produces Holt's linear method. In practice, ϕ is normally set between 0.8 and 0.9.

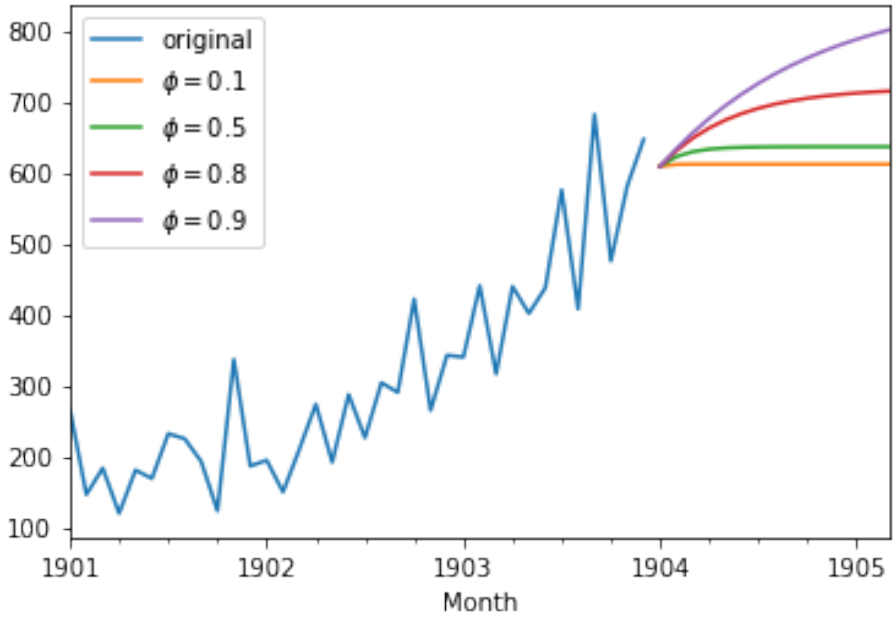


Figure 3: Forecasts for different values of ϕ

2.1.3 Parameter Estimation

Each type of exponential smoothing method needs smoothing parameters and initial values to be chosen. These can be chosen manually but a better option is to get a computer to optimise these value for us. A common method which I implemented is done by minimising the sum of the squared errors (SSE) done quickly through an optimisation library:

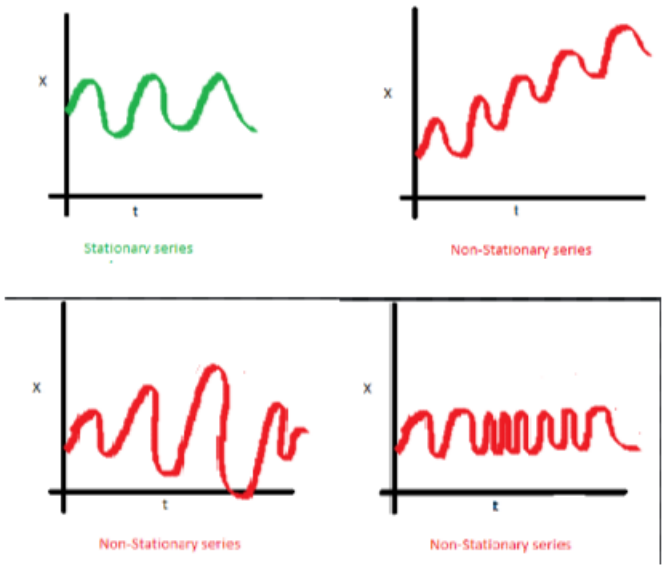
$$SSE = \sum_{i=1}^{T} (y_i - \hat{y}_i)^2 .$$

2.2 ARIMA

ARIMA is an extension of the ARMA or Box-Jenkin model popularised by George E. P. Box and Gwilym Jenkins in 1970. The main difference is that ARIMA simply allows for an extra parameter d which defines how much differencing should be done before fitting the model, whereas ARMA assumes the data is stationary already. ARIMA is actually a combination of three components, an autoregressive model (AR), a moving averages model (MA), and an integrated (I) part denoting the order of differencing. The ultimate goal of ARIMA is to filter the noise and extrapolate the signal wihtin a time series.

2.2.1 Stationarity

ARIMA models work best on data that is "stationary". A stationary time series possesses statistical properties, such as mean, variance, covariance, etc, that remain constant throughout the whole time. Below shows the differences between stationary and non-stationary processes.



2.2.2 How to test for stationarity

To determine whether a time series is stationary or not, the Augmented Dickey Fuller(ADF) test can be performed. ADF performs a hypothesis test and upon observing p-value $p < \alpha$ where α is commonly 0.05, the hypothesis is rejected and the data is considered to be stationary. Performing the ADF test on the original shampoo dataset produces $p = 1.0 > 0.05$ so the data is considered non-stationary.

2.2.3 Methods of making a time series stationary

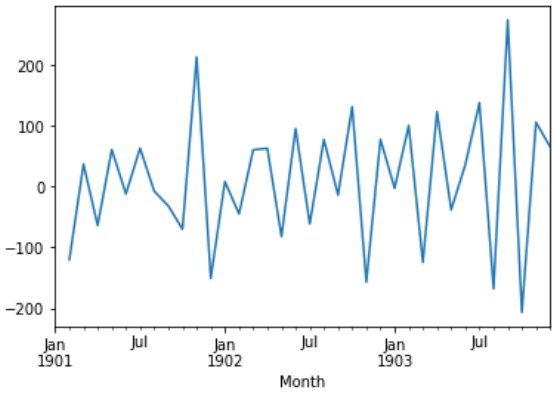
If you have a non-stationary time series, there are a few ways to make it stationary. The shampoo dataset is clearly non stationary because there is a defined upwards trend.

The most common practice to make a time series stationary is differencing. Differencing involves creating a new dataset where the values at time t is equal to the difference between the original value at time t and original value at $t-1$. So if y' is the new dataset and y is the original data, then

$$y'_t = y_t - y_{t-1}.$$

The p-value from the ADF test is about $1.8 \times 10^{-10} < 0.05$ so the data is now stationary when differenced once.

Month	Sales	diff_1
1901-01-01	266.0	NaN
1901-02-01	145.9	-120.1
1901-03-01	183.1	37.2
1901-04-01	119.3	-63.8
1901-05-01	180.3	61.0
...	...	...
1903-08-01	407.6	-167.9
1903-09-01	682.0	274.4
1903-10-01	475.3	-206.7
1903-11-01	581.3	106.0
1903-12-01	646.9	65.6



Sometimes the differenced data will not make the data stationary enough and it may be necessary to difference the data again known as 2nd order differencing. Note, the 2nd order difference is the first difference of the first difference so it comes out to

$$y'_t = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}).$$

There are various other techniques that can be performed and combined with one another to make your data more stationary. These include model fitting, non-linear transformations, or seasonal differencing.

2.2.4 Backshift Notation

Before we jump into defining what an ARIMA model is, there is a useful backshift operator B,

$$By_t = y_{t-1}.$$

The time series values prior to the value at time t are known as "lags". B effectively shifts that data back one time period.So naturally, two applications

of B shifts the data back two time periods.

$$\begin{aligned} B(By_t) &= B^2 y_t \\ &= y_{t-2}. \end{aligned}$$

If we describe how differencing works using this notation it comes out to

$$\begin{aligned} y'_t &= y_t - y_{t-1} \\ &= y_t - By_t \\ &= (1 - B)y_t. \end{aligned}$$

In general, the d'th order of differencing is defined as

$$y'_t = (1 - B)^d y_t.$$

This notation is great as we can use B is an algebraic manner which greatly simplifies some upcoming formulas.

2.2.5 Autoregressive Models

The term *autoregressive* defines that the model uses a linear combination of some previous values and a list of coefficients. Autoregressive models have one parameter p which defines the order of the model. This parameter influences how many previous values are used to predict the next. An AR(1) model is simply defines a model in which the predicted value is equal to some linear combination of the value at the previous time step

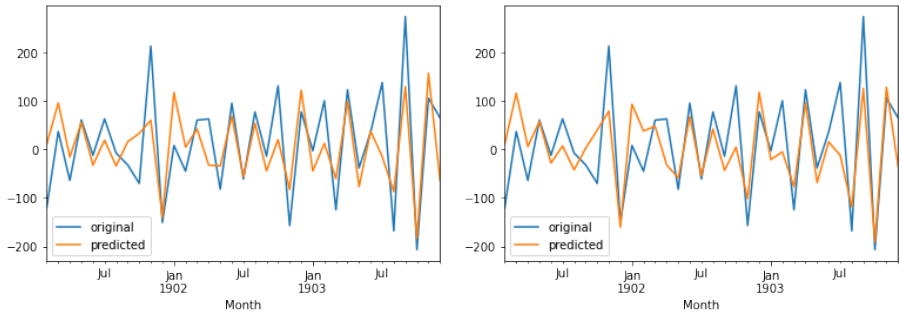
$$y_t = \mu + \phi_1 y_{t-1},$$

where μ is the average period to period change. An AR(2) model uses the previous two timesteps as so on. Thus, an AR(p) model can be generalised to

$$\begin{gathered} y_t = \mu + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} \\ \text{or in equivalent backshift notation} \\ (1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^p) y_t = \mu. \end{gathered}$$

We can see that we are using previous known values as the features to predict the future.

Let's see how this works for the first differenced data.



(a) AR(1) (b) AR(2)

Figure 4: Two types of AR models. AR(1) with estimated $\phi_1 = -0.70868438$. AR(2) with estimated $\phi_1 = -0.9237595, \phi_2 = -0.26627682$.

Observe that there are subtle changes between the two. If we kept on increasing the AR term, our model would fit this data extremely well. In machine learning, this is commonly known as overfitting and future forecasts can be very inaccurate.

2.2.6 Moving Average Models

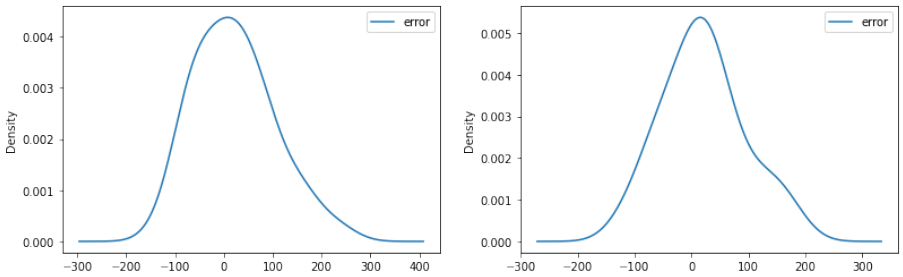
Moving Average (MA) models istead use the errors of past forecasts to predict. An MA model instead has a single parameter q so an MA(q) model can be written as

$$y_t = \mu + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + ... + \theta_q \varepsilon_{t-q}$$

or in equivalent backshift notation

$$y_t = \mu + (\theta_1 B + \theta_2 B^2 + ... + \theta_q B^q)\varepsilon_t.$$

If your data is fully stationary, the error terms produced should form white noise in the form of a normal distribution with 0 mean and θ^2 variance.



(a) MA(1) (b) MA(2)

Figure 5: Error terms for MA(1) with estimated $\theta_1 = -0.58305195$ and MA(2) with estimated $\theta_1 = -0.71206191, \theta_2 = 0.28793809$.

2.2.7 Putting It All Together

When we combine differencing, an AR, and an MA model, we obtain an ARIMA model. ARIMA has 3 parameters and can be expressed as ARIMA(p, d, q).

- p is the order of autoregressive component
- d is the order of differencing to be performed on the data
- q is the order of moving averages component

It's model equation is as follows

$$y'_t = \mu + \phi_1 y'_{t-1} + ... + \phi_p y'_{t-p} + \theta_1 \varepsilon'_{t-1} + ... + \theta_q \varepsilon'_{t-q}$$

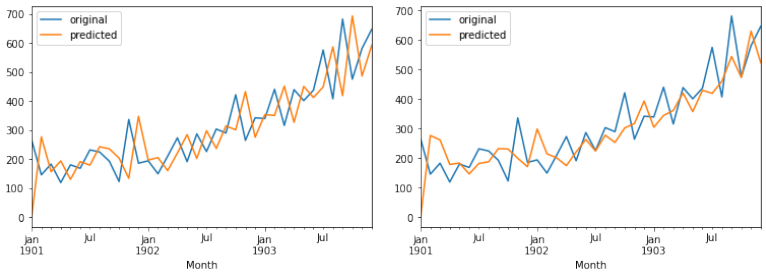
or in equivalent backshift notation

$$\underset{\uparrow}{(1 - \phi_1 B - \cdots - \phi_p B^p)} \quad \underset{\uparrow}{(1 - B)^d y_t} \quad = \quad \underset{\uparrow}{\mu + (\theta_1 B + \cdots + \theta_q B^q)\varepsilon_t}$$

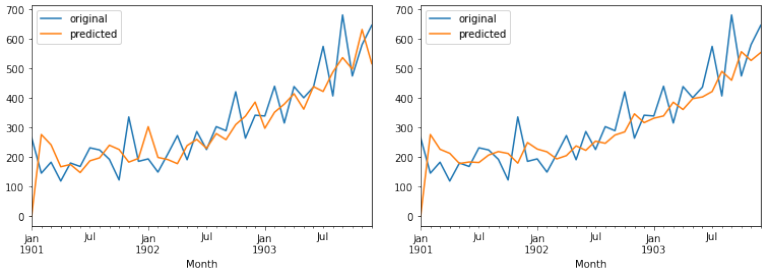
$$\text{AR}(p) \qquad d \text{ differences} \qquad \text{MA}(q)$$

where again y'_t is the differenced data. Here you can see how both AR and MA models are incorporated into the full model and why the backshift operator is very useful for its definition.

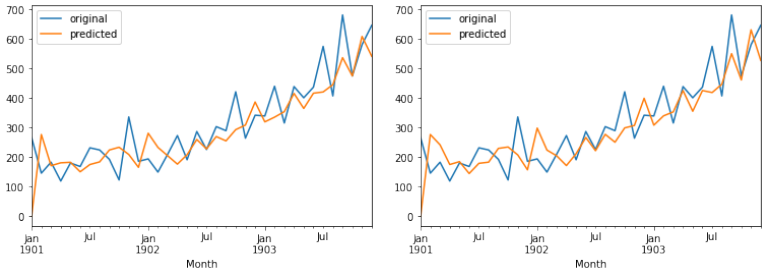
We can see effects of different values for the parameters on how they fit the first differenced data.



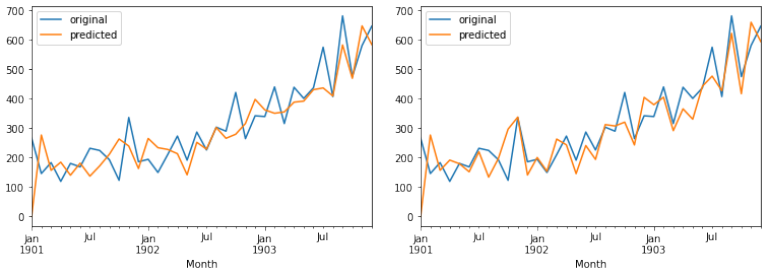
(a) ARIMA$(0, 1, 0)$ (b) ARIMA$(1, 1, 1)$



(c) ARIMA$(1, 1, 0)$ (d) ARIMA$(0, 1, 1)$



(e) ARIMA$(2, 1, 1)$ (f) ARIMA$(1, 1, 2)$



(g) ARIMA$(5, 1, 2)$ (h) ARIMA$(2, 1, 5)$

Figure 6: Fits for different values of p and q

2.2.8 ACF and PACF

Picking the right parameter values is dataset specific but there are auto_arima packages in Python and R which can search for optimal values based on your data. Observing and analysing the autocorrelation function (ACF) and partial autocorrelation function (PACF) plots often also give a good indication of what values of p and q you should choose. ACF is a numerical representation of how correlated a time series is to a lagged version of itself. Conversely, PACF represents the amount of correlation between a time series and a lagged version of itself after adjusting for all other terms of a shorter lag.

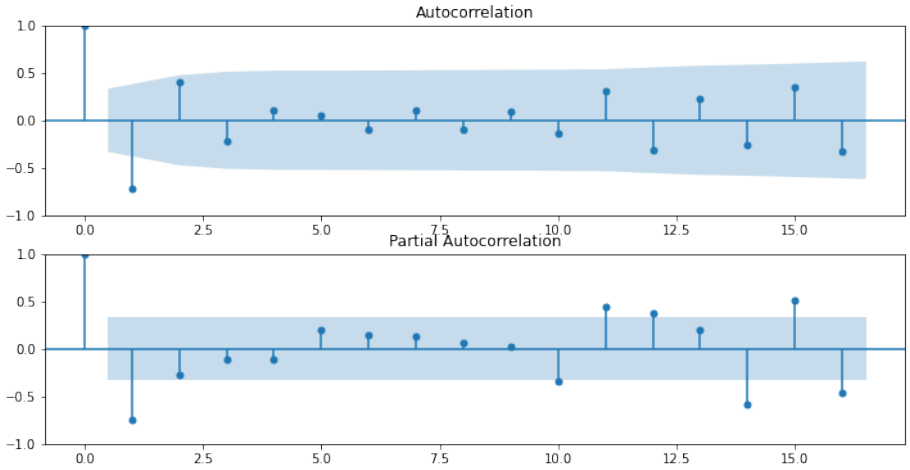


Figure 7: ACF and PACF plot for first differenced data

Obviously a time series is perfectly correlated to itself lagged zero times. Here, ACF helps determine the MA order and PACF for the AR order. We are looking for large drops or the earliest lag somewhere within the blue zone which represents the 95% confidence level. So for the shampoo dataset, a value of $p = 1$ or 2 and $q = 1$ or 2 would be suitable.

2.2.9 Coefficient Estimation

Estimating the coefficients for an ARIMA model is an extremely important but often a tricky task as there needs to be a balance between having a good fit for the data while not overfitting. For my implementation, similar to exponential smoothing, both the AR and MA terms were optimised to minimise the sum of squared errors. To avoid overfitting the data, I set constraints on the coefficients namely,

- $-1 \leq \phi_p \leq 1$
- $\phi_i + \phi_{i+1} \leq 1 \quad \text{for } i = 1, 2, ..., p-1$

- $\phi_{i+1} - \phi_i \leq 1 \quad \text{for } i = 1, 2, ..., p-1$

The same constraints are put for $\theta_1, ..., \theta_q$. In practice, ARIMA packages in Python and R use Maximum Likelihood Estimation to estimate the coefficients which often invoves much more complicated computations. There are also more complicated constraints, especially as the values of p and q increase, which these packages handle that account for the invertability and stationarity constraints.

2.2.10 Forecasting

So far, we have shown how to fit an ARIMA model on a stationary time series so let's see how this model forecasts future data points. It is quite simple and involves 3 steps:

1. Expand the model's ARIMA backshift equation and group by the powers of B
2. Convert to equation with only y_t's and move all terms that isn't y_t to the right side of the equation
3. Replace t with $T+1$ is the number of future time steps and update time series with predicted value

Let's try this with an example using an ARIMA(2,1,2) model as an example. In backshift notation, the model is written as

$$(1 - \phi_1 B - \phi_2 B^2)(1 - B)y_t = \mu + (\theta_1 B + \theta_2 B^2)\varepsilon_t.$$

Using step 1, we expand and group:

$$\begin{aligned}(1 - B - \phi_1 B + \phi_1 B^2 - \phi_2 B^2 + \phi_2 B^3)y_t &= \mu + (\theta_1 B + \theta_2 B^2)\varepsilon_t \\ [1 - (1 + \phi_1)B - (\phi_2 - \phi_1)B^2 + \phi_2 B^3]y_t &= \mu + (\theta_1 B + \theta_2 B^2)\varepsilon_t.\end{aligned}$$

With second step, we convert giving

$$y_t - (1 + \phi_1)y_{t-1} - (\phi_2 - \phi_1)y_{t-2} + \phi_2 y_{t-3} = \mu + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2},$$

then moving terms

$$y_t = \mu + (1 + \phi_1)y_{t-1} + (\phi_2 - \phi_1)y_{t-2} - \phi_2 y_{t-3} + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2}.$$

Third step is to replace t with $T+1$:

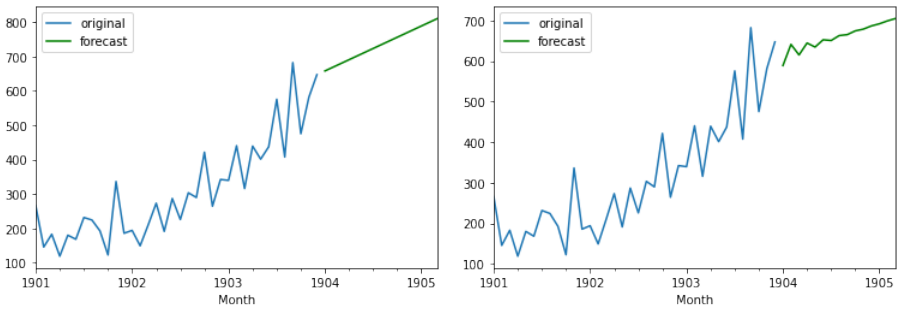
$$y_{T+1} = \mu + (1 + \phi_1)y_T + (\phi_2 - \phi_1)y_{T-1} - \phi_2 y_{T-2} + \theta_1 \varepsilon_{T-1} + \theta_2 \varepsilon_{T-2}.$$

To predict further into the future say at time $T+2$, then just repeat step 3 which produces

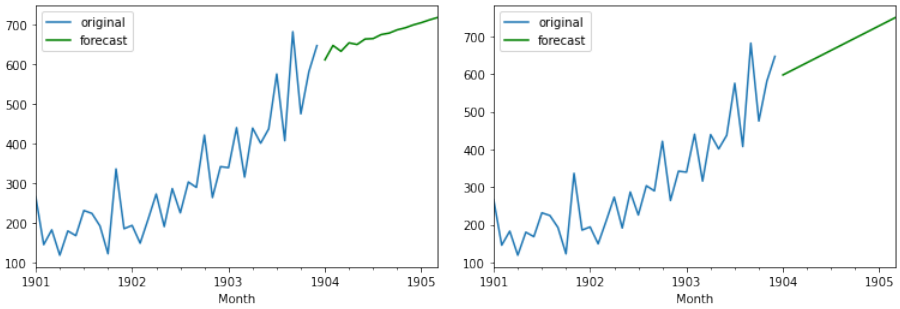
$$y_{T+2} = \mu + (1 + \phi_1)y_{T+1} + (\phi_2 - \phi_1)y_T - \phi_2 y_{T-1} + \theta_2 \varepsilon_{T-1}.$$

Observe that there should be a $\theta_1 \varepsilon_T$ term but since we set the time series value at T to be equal to the predicted value, this error term evaluates to 0. We are essentially taking our predicted value as the truth which is why all the error terms eventually become 0. Hence, if you predict $h > q$ future time steps, all MA terms disappear.

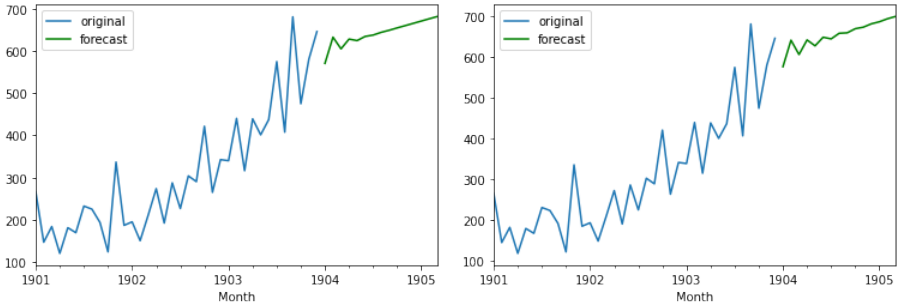
Looking at the below plots, while the ARIMA(5,1,2) forecasts looks the most realistic, it is still quite noisy so the ARIMA(1,1,1) or ARIMA(1,1,2) might prove to be a better forecasting model.



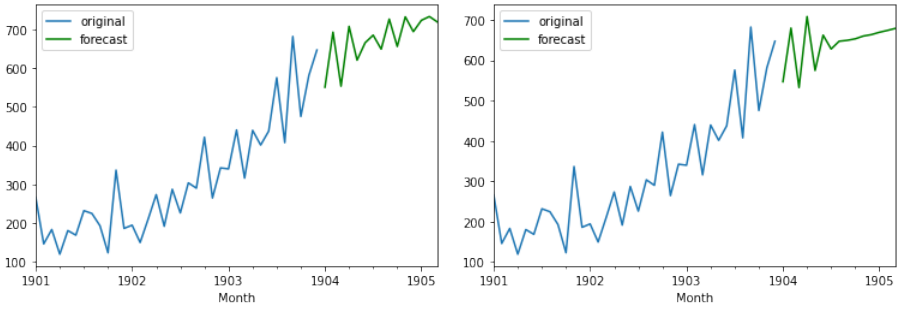
(a) ARIMA$(0, 1, 0)$ (b) ARIMA$(1, 1, 1)$



(c) ARIMA$(1, 1, 0)$ (d) ARIMA$(0, 1, 1)$



(e) ARIMA$(2, 1, 1)$ (f) ARIMA$(1, 1, 2)$



(g) ARIMA$(5, 1, 2)$ (h) ARIMA$(2, 1, 5)$

Figure 8: Forecasting for 15 future time steps from different ARIMA models. (c) and (d) are equivalent to an AR(1) and MA(1) with 1 order of differencing.

3 Seasonal models

Many time series data in the real work exhibit some form of seasonality. This includes higher network traffic during peak hours of the day or increased spending during Christmas. As such, there is a defined frequency of seasonality commonly denoted as m. For example $m = 4$ could represent quarterly data and $m = 12$ for monthly data. Here we will explore how seasonal ARIMA (SARIMA) and the Holt-Winters' seasonal method works and accounts for the seasonal component of the time series. For these examples, we will use the monthly milk dataset so $m = 12$. This dataset is great since there is a clear seasonal pattern and has an upward trend.

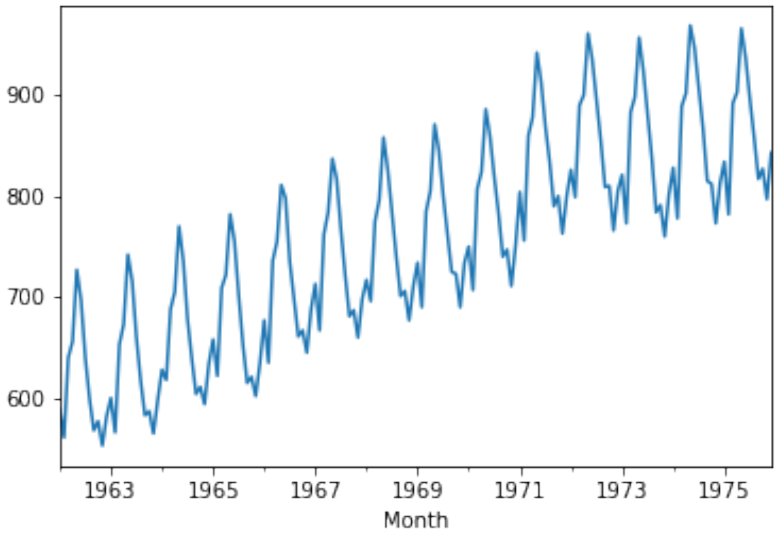


Figure 9: Milk dataset

3.1 Holt-Winters' Seasonal Method

Holt and Winters improved Holt's linear method to capture seasonality. This method includes a level equation, trend equation, seasonal equation, and forecasting equation. It introduces a new smoothing parameter γ and m initial seasonal values that have to be estimated. This method comes in two flavours, an additive and multiplicative method. Which one you pick is dependent on your data. The additive method is preferred when seasonal variations and trend change at a constant rate. Peaks and valleys in your data are roughly of the same magnitude. In contrast, multiplicative is preferred when the changes are proportional to the level of the series.

3.1.1 Additive Method

$$\begin{aligned}
\text{Level equation} \quad & \ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
\text{Trend equation} \quad & b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1} \\
\text{Seasonal equation} \quad & s_t = \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m} \\
\text{Forecast equation} \quad & \hat{y}_{t+h} = \ell_t + hb_t + s_{t+h-m(k+1)},
\end{aligned}$$

where k is defined as $\lfloor \frac{h-1}{m} \rfloor$. This guarantees that that every forecast uses the seasonal component from the most recently observed season.

3.1.2 Multiplicative Method

$$\begin{aligned}
\text{Level equation} \quad & \ell_t = \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
\text{Trend equation} \quad & b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1} \\
\text{Seasonal equation} \quad & s_t = \gamma\frac{y_t}{(\ell_{t-1} + b_{t-1})} + (1-\gamma)s_{t-m} \\
\text{Forecast equation} \quad & \hat{y}_{t+h} = (\ell_t + hb_t)s_{t+h-m(k+1)},
\end{aligned}$$

3.1.3 Initial Values for Trend and Seasonal Components

Similar to the non-seasonal methods, values can be estimated for the initial trend and initial seasonal components to speed up the optimisation process.

Initial Trend

For Holt Linear, we simply used the first two data points for the initial trend. Since we have seasonal data, we can make a better estimate by taking the average trend between the first two seasons:

$$b_0 = \frac{1}{m}\left(\frac{y_{m+1} - y_1}{m} + \frac{y_{m+2} - y_2}{m} + ... + \frac{y_{m+m} - y_m}{m}\right)$$

Initial Seasonal Components

Calculating the initial seasonal components is done in a relatively similar manner. First we compute the averages A_n for each of the N years,

$$A_n = \frac{\sum_{i=1}^{m} y_i}{m}, \qquad n = 1, 2, ..., N.$$

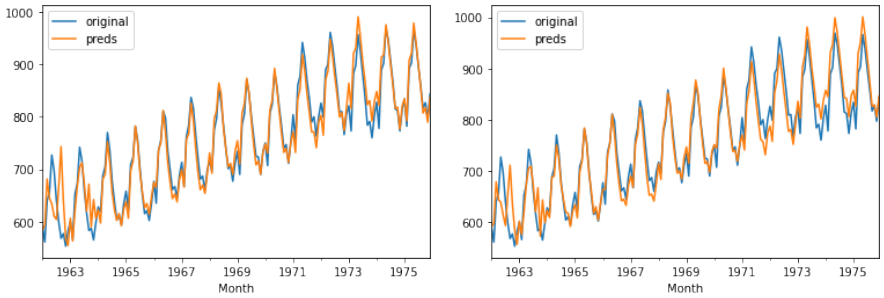
The next step depends on whether the additive or multiplicative method is used. For additive, we sum up the differences between the time associated data points in each season with their respective yearly average and finally divide by N,

$$\begin{aligned} s_1 &= \frac{1}{N}\left[(y_1 - A_1) + (y_{m+1} - A_2) + ... + (y_{m(N-1)+1} - A_N)\right] \\ s_2 &= \frac{1}{N}\left[(y_2 - A_1) + (y_{m+2} - A_2) + ... + (y_{m(N-1)+2} - A_N)\right] \\ &\vdots \end{aligned}$$

For multiplicative, instead of taking the difference, we divide the data point by the respective average, i.e. $\frac{y_1}{A_1}$.

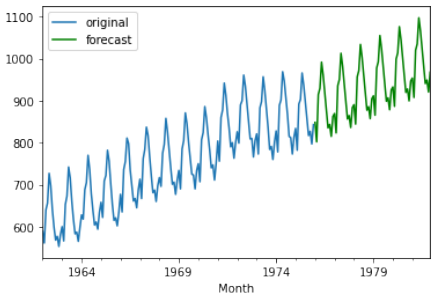
3.1.4 Example

Applying both methods to the milk dataset, the smoothing parameters for additive came out to $\alpha = 0.057, \beta = 0, \gamma = 0.951$ and the multiplicative came out to $\alpha = 0, \beta = 0.097, \gamma = 0.878$.

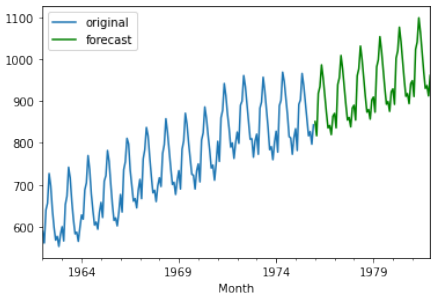


(a) Additive fit

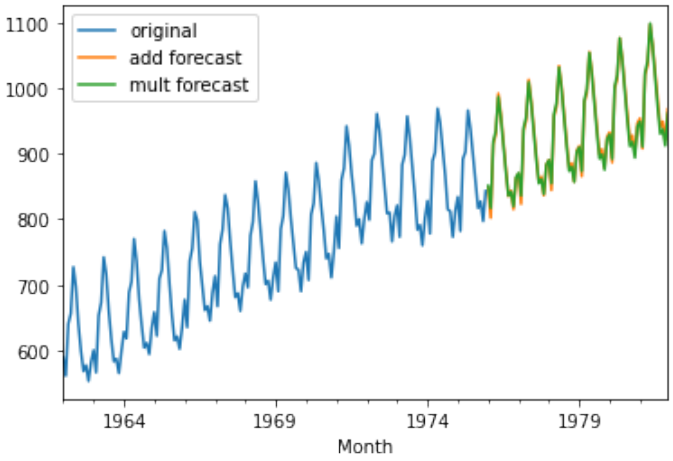
(b) Multiplicative fit



(c) Additive forecast



(d) Multiplicative forecast



(e) Comparitive forecast

Figure 10: Fits and forecasts for Holt-Winters' method

Looking at the comparitive forecast, the differences between additive and multiplicative become clear. The difference between the peaks and dips in the multiplicative forecast become larger each season on whereas it stays constant in the additive case.

3.2 Seasonal ARIMA

If non-seasonal ARIMA wasn't complicated enough, there exists a seasonal extension which introduces 3 new parameters. A seasonal ARIMA model has two components, non-seasonal and seasonal, which are both structurally the same. Each have their own AR, MA and order of differencing parameters. Combined with non-seasonal differencing, taking a seasonal difference can in some instances help make the data more stationary. Seasonal differencing is simply $y'_t = y_t - y_{t-m}$.

Seasonal ARIMA models can be written as

$$ARIMA(p,d,q) \times (P,D,Q)_m$$

where P, D, and Q are the seasonal AR (SAR) terms, order of seasonal differencing, and seasonal MA (SMA) terms respectively with m being the period.

Using backshift notation, the entire equation can be written as:

$$(1 - \phi_1 B - ... - \phi_p B^p)(1 - \Phi_1 B - ...\Phi_P B^{mP})(1-B)^d(1-B^m)^D y_t = \\ \mu + (1 + \theta_1 B + ... + \theta_q B^q)(1 + \Theta_1 B + ...\Theta_Q B^{mQ})\varepsilon_t.$$

In practice, picking the right values for the parameters is very dataset specific. Similarly to ARIMA, the same auto_arima packages can be used to optimise the parameters for SARIMA models.

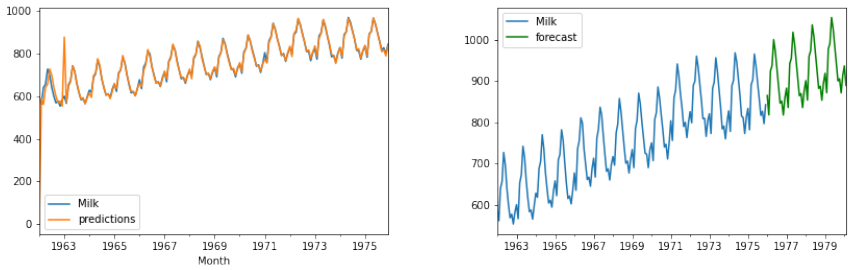


Figure 11: SARIMA$(1,1,1) \times (0,1,1)_{12}$ fit and forecasts

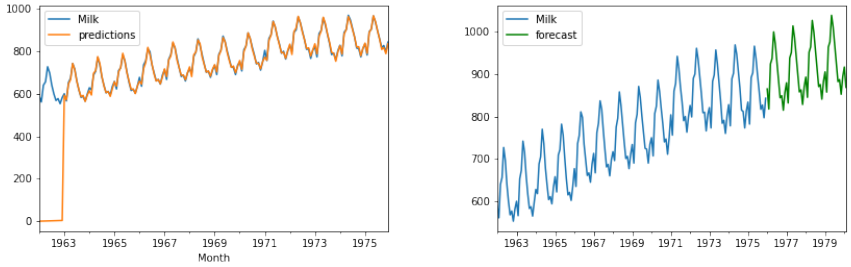


Figure 12: SARIMA$(2,0,0) \times (0,1,1)_{12}$ fit and forecasts

4 Conclusion

We covered two of the most popular time series forecasting methods in exponential smoothing and ARIMA models as well as implementing them in Python. As previously mentioned, accurate forecasting is practically impossible, especially when you are only given a time and value. Recurrent and convolutional neural networks have been employed to deal with extra factors but they require a lot of computational power to operate.

Choosing which model to pick is dependent on the type of data you want to forecast as they use different approaches. Both are extremely robust and powerful models for time series forecasting. In my experience, for large values of $m > 50$, SARIMA models can take a considerable more amount of time to fit than the Holt-Winters' model for practically the same results.

5 Reflection

I came across this area during my previous internship where I simply just used the built-in packages without really knowing how any of it worked. The fact that the major project was very open ended got me curious and motivated my choice of this topic. During my initial research, I found out how few manual implementations of these models were, especially ARIMA. A lot just used the

same external libraries with very little explanation. Slowly, after reading more scholarly papers and textbooks, I was able to get a solid fundamental understanding of the mathematics and processes which drive these models and started on the implementation. Throughout the implementation stage, I got caught up trying to get the same results as the external libraries but no matter how many different approaches I tried, I could never fully accomplish it and eventually settled that I was "close enough". Overall, this was a fun challenge to work on and will be very useful to me sometime in the future.

References

[1] Robert Nau. *Statistical forecasting: notes on regression and time series analysis*. URL: `https://people.duke.edu/~rnau/411home.htm`.

[2] Grisha Trubetskoy. *Holt-Winters Forecasting for Dummies - Part III*. 2016. URL: `https://grisha.org/blog/2016/02/17/triple-exponential-smoothing-forecasting-part-iii/`.

[3] Hyndman R.J. & Athanasopoulos G. *Forecasting: Principles and Practice (2nd edition)*. OTexts, 2018.