Published in Towards Data Science

Shweta Follow

Jul 27, 2021 · 13 min read · Listen

Introduction to Time Series Forecasting

Part 1: Average and Smoothing Models

Time Series is a unique field. It is a Science in itself. Experts quote 'A good forecast is a blessing while a wrong forecast can prove to be dangerous'. This article aims to introduce the basic concepts of time series and briefly discusses the popular methods used to forecast time series data.

A time series data is the data on a response variable Y(t) observed at different points in time t. Data on the variable is collected at regular intervals and in a chronological order. **Anything that is observed sequentially over time is time series.**

For e.g. data collected on the sale of smartphones over several time intervals, the GDP of a nation each year, electricity production every year/month etc. are all examples of time series data.

The aim of forecasting time series data is to understand how the sequence of observations will continue in the future.

A time series data will have one or more than one of these following components:

1. **Trend Component** — It is the consistent upward or downward movement of the data over the entire time span. The trend can be both linear and non linear
2. **Seasonality Component** — It is the repetitive upward or downward fluctuation from the trend that occurs within a calendar year at fixed intervals. It always has a fixed and a known frequency.
3. **Cyclical Component** — Cyclical fluctuations occur due to macro economic factors like recession. Here, the interval between the repetition is more than a few years. Periodicity of cyclical fluctuation is not fixed. The average length of cyclical pattern is longer than the average length of the seasonal pattern.
4. **Irregular Fluctuation (Also called White Noise)** — It is the uncorrelated random component of the time series data. If the time series data only has White Noise as a component, it cannot be used for prediction. This is because the series has observations which are identically and independently distributed with a mean of 0 and a constant variance.

The choice of the forecasting model will depend on the component/s present in the time series. The time series forecasting models can be broadly classified into Simple Models (Mean Model, Linear Trend Model, Random Walk Model) , Average and

Let us understand these underlying models before we deep dive into complex models like smoothing and ARIMA.

Simple Models of Forecasting :

Mean Model :

For a time series that is independently and identically distributed(**i.i.d — there is no trend and all observations have th** 147 | 1 **ty distribution and are independent from each other**), the forecast at time t+1 is given by the mean of the historical data till time t. This mean value minimizes the mean squared error and is also an unbiased predictor.

If we predict long into the the future, the forecast will be a horizontal line or the mean. This is because the model assumes that all future observations will be drawn from the same distribution . Consider a series X with mean 45. Therefore, as per the mean model, the forecast of X for all future periods should be 45. We know that this is an unrealistic assumption unless X is a set of independent random samples from a population that does not change with time.

Linear Trend Model :

A linear trend model is a special case of simple regression model in which the independent variable is time t. It is used in time series where the mean is gradually increasing over time i.e. there is a constant trend. In such cases, instead of using the horizontal line or the mean model to forecast the future value, a sloping line is fit to the data. The linear trend model tries to find the slope and intercept that gives the best fit to the historical data.

Random Walk Model:

In a Random Walk Model, the value of time series X at y(t+1) is equal to y(t) plus a random noise.

At t=2, X2 = X1+Z2 . But X1 = Z1, therefore, X2 = Z1 + Z2

If we perform this operation far into the future, we get X(t) = Z1+Z2+…Z(t).

Thus, the forecast value at time t is the sum of the white noise till time t.

How do we interpret this?

The model assumes that in each period the variable takes a random step away from its previous value and the steps are independently and identically distributed in size i.e. the change in the values of the series at time t and t-1 is completely random and has a zero mean.

Random Walk patterns are seen in stock prices. The price movement is not random , however the day to day change in prices is random due to which it is impossible to forecast the price for the next day.

Another variation of Random Walk is a **Random Walk with Drift**. Here, the series take a random step away from its last recorded position with steps that has a non zero mean i.e. y(t) = y(t-1) + alpha, where alpha is the drift parameter.

Let us now get into actual time series forecasting.

We will use the data on the Industrial Production — Utilities to understand the concept of time series forecasting better. The data used can be sourced from this link : **https://fred.stlouisfed.org/series/IPG2211A2N** . The data set gives the monthly industrial production of all gas and electrical utilities in the United States from 1940 to 2020.

Let's first start by importing important libraries in Python :

```
# Importing Libraries
import pandas as pd
import numpy as np
```


```
from numpy import mean
from sklearn.metrics import mean_squared_error
import math
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima_model import ARIMA
import statsmodels.api as sm
import pmdarima as pm
from statsmodels.tsa.api import ExponentialSmoothing
from matplotlib import pyplot
import warnings
import itertools
```

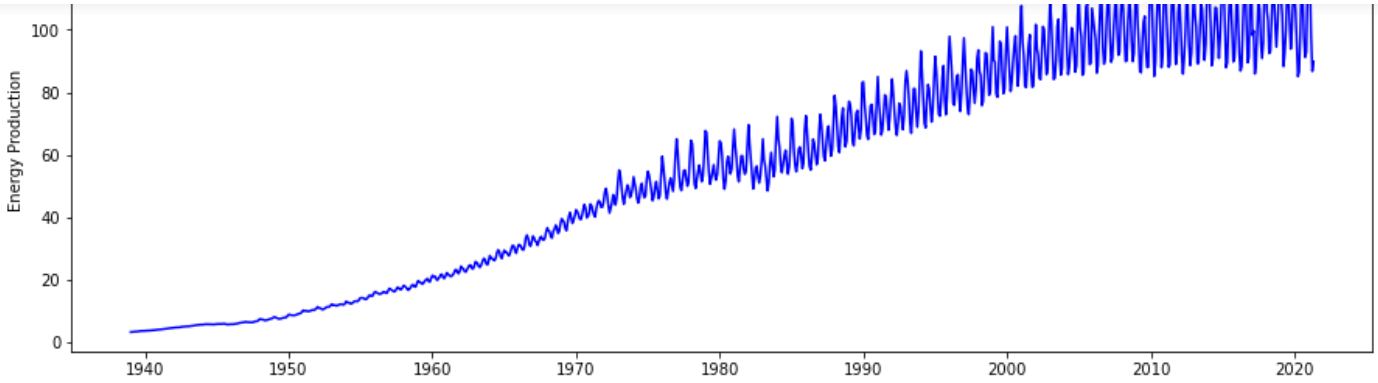
Let us also look at the initial observations. There are two columns : Date and Production column named as 'IPG2211A2N'.

```
df.head()

   DATE            IPG2211A2N
0  1939-01-01      3.3298
1  1939-02-01      3.3552
2  1939-03-01      3.4315
3  1939-04-01      3.4569
4  1939-05-01      3.4569
```

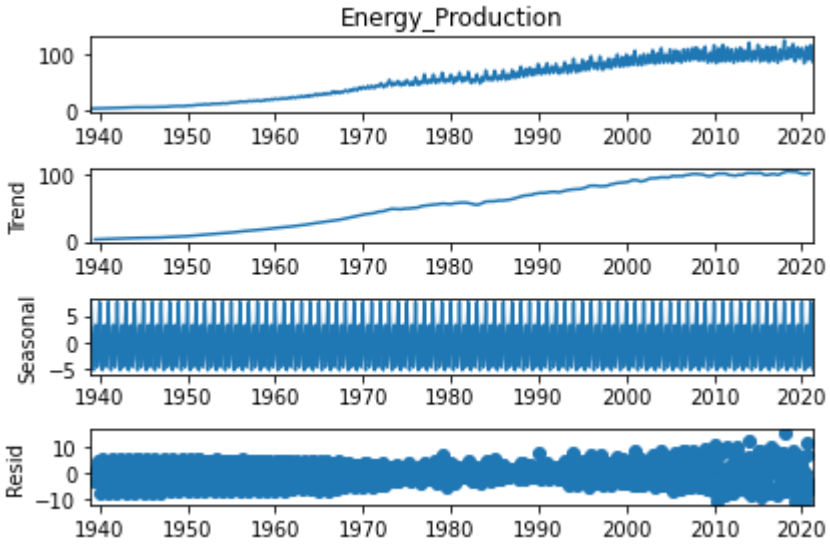
We will change the name of the column 'IPG2211A2N' to 'Energy_Production'. The date is in the object format. We will change it to datetime. There are in all 989 observations with no missing data.

Let's plot the time series.

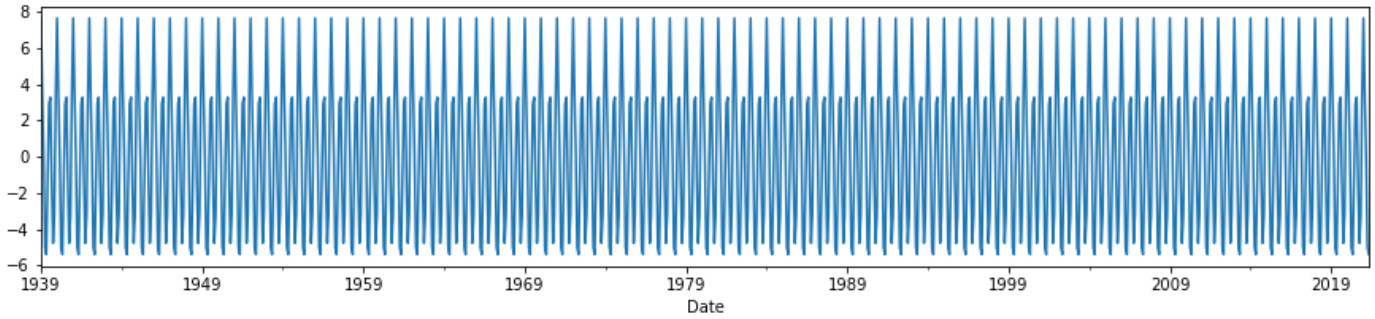
We can observe that there is both trend and seasonality. We can decompose the time series data into its individual components using the function '**decompose**' in Python.

```
y_decompose = seasonal_decompose(df['Energy_Production'], model =
'additive', freq = 12)
y_decompose_plot = y_decompose.plot()
```



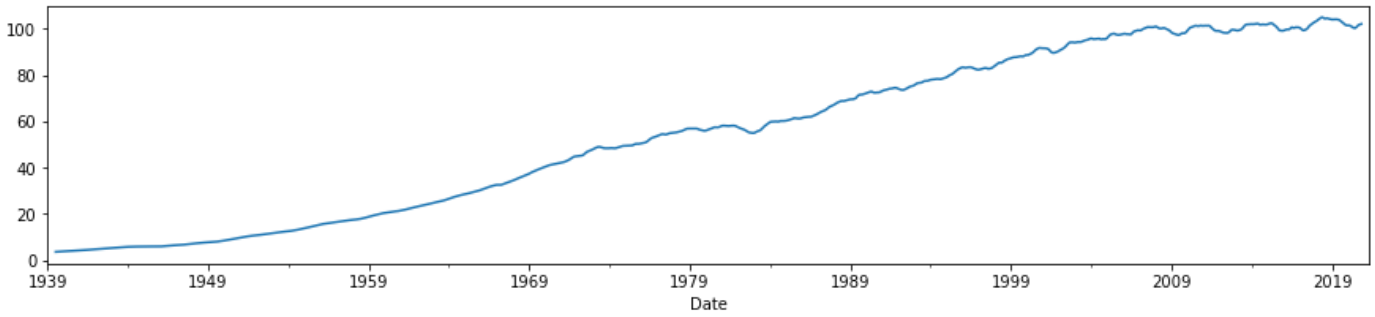
On using this function, we get four different plots. These include the overall visual plot of the series, the trend component, the seasonal component and the residual. One can also look at each time series component separately.

Seasonal Component of Time Series



Trend Component of Time Series

```
plt.figure(figsize=(15,3))
y_decompose.trend.plot();
```



If we observe these individual plots, we can infer that **both the trend and seasonal components are present in the time series and it is additive.**

What does this mean?

Additive time series is one in which the magnitude of trend and seasonality does not increase with time. They remain fairly constant. **Multiplicative time series** is one in which the magnitude of trend and seasonality increases as time period increases.

We will first use simple methods of forecasting, check the error metrics (RMSE and MAPE) and then use more complex forecasting measures like SARIMA.

components.

SMA is one of the simplest forecasting method that forecasts the future value of a time series data using average of the past N observations. Here, N is the hyperparameter. The basic assumption of averaging models is that the series has a slow varying mean. Thus, we take a moving average to estimate the current value of the mean and then use this as a forecast for the future. **If the series is more or less stable, a lower value of N can be taken. If the series is very volatile, a higher value of N should be taken. The value of N needs to be explored to find the best fit model.**

The formula for Moving Average Method is given as :

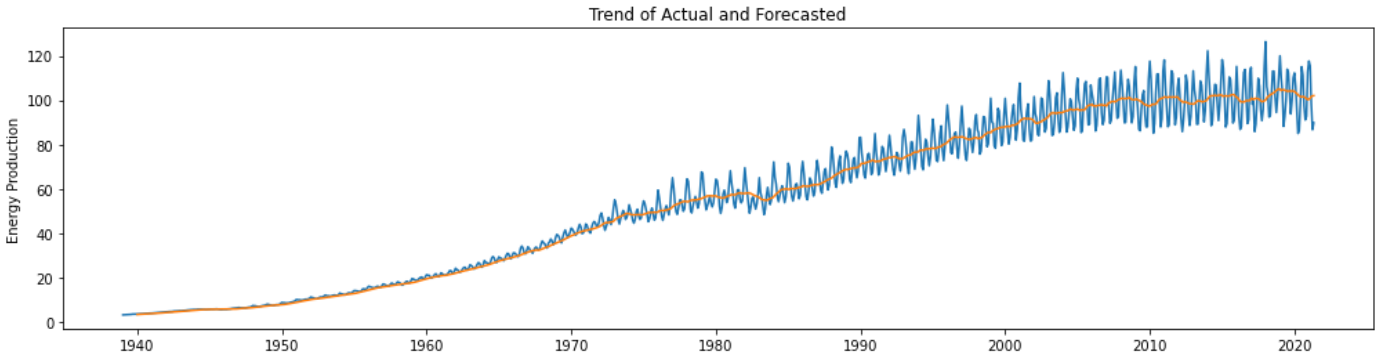
$$SMA = \frac{A_1 + A_2 + ... + A_n}{n}$$

The given time series is highly seasonal and also has a strong trend. SMA method of forecasting will not work here. However, we will still go ahead with it to understand why it is not the best model.

We will take a 12 month moving average as we are looking at monthly data and the pattern repeats itself every year. We will then plot the actual and predicted trend and observe how close/distant the predicted value is from the actual time series. The below python code .rolling (window=12) takes the hyper parameter N.

```
df1 = df.copy()
df1['Moving Avg_12'] =
df1['Energy_Production'].rolling(window=12).mean().shift(1)
```


```
plt.title('Trend of Actual and Forecasted')
plt.plot(df1[['Energy_Production','Moving Avg_12']]);
```



Clearly the Moving Average Method gives the average trend. It does not reflect the peak and troughs of the actual data. Hence, in this case we cannot predict the production. We will check the forecast accuracy using metrics like RMSE (Root Mean Square Error) and MAPE (Mean Absolute Percentage Error).

For the beginners, **RMSE is the square root of the average of the squared errors.** It is given by the formula :

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

The other accuracy metric used is **Mean Absolute Percentage Error. It is the average of absolute percentage error**. It is easily interpretable as it expresses the average error in percentage terms. The formula for MAPE is given as :

$$MAPE = \frac{\sum \frac{|A-F|}{A} \times 100}{N}$$

latest 5 year data

```
# Function for MAPE
def get_mape(actual, predicted):
    return np.round(np.mean(np.abs((actual-predicted) /
actual))*100,2)

get_mape(df1['Energy_Production'][928:].values, df1['Moving Avg_12']
[928:].values)
```

8.48

```
# Calculate RMSE
from sklearn.metrics import mean_squared_error
np.sqrt(mean_squared_error(df1['Energy_Production'][928:].values,
df1['Moving Avg_12'][928:].values))
```

10.158

Thus, the forecast accuracy metrics for 12 month moving average method is : RMSE = 10.15 , MAPE = 8.48

This can definitely be improved further using more advanced methods.

Method 2 : Exponential Smoothing Method

The drawback of Simple Moving Average Method is that it gives equal weight to all the observations. Intuitively the most recent observation should be given more weight that the earlier observations. The **Exponential Smoothing Method** removes this limitation by **assigning differential weights to the past observations. Here, the weights assigned to past data decline exponentially with the most recent observations assigned higher weights. The rate at which the weights decreases is controlled by a hyperparameter also called the 'smoothing constant' .**

the mean squared errors.

Exponential Smoothing Methods are of three types :

1. **Single Exponential Smoothing**
2. **Double Exponential Smoothing and**
3. **Triple Exponential Smoothing or Holt Winters Method**

Let's cover each one of them in detail.

Single Exponential Smoothing :

This method addresses only the level component of the time series. It uses a hyper parameter alpha which is called a smoothing constant, the value of which lies between 0 and 1. Since only one smoothing constant is used, it is called Single Exponential Smoothing.

Here the forecast at time t is given as **Ft = α * y(t-1) + (1-α)*F(t-1)**

Double Exponential Smoothing :

This addresses both the level(l) and trend (b) component of the time series. Thus, **two smoothing constants are used** i.e. alpha for the level component and beta for the trend component. The equations are as follows:

$l(t) = \alpha * y(t) + (1-\alpha)*(l(t-1)+b(t-1))$ —— **Level** l
$b(t) = \beta * (l(t) - l(t-1)) + (1-\beta)* b(t-1)$ —— **Trend** b
$y(t+1) = l(t) + b(t)$ —— **Forecast**

Models with low values of beta assume that the trend changes very slowly over time while models with larger value of beta assume that the trend is changing very rapidly.

Single Exponential Smoothing method assumes the time series to be relatively stable with

exponential smoothing as well.

We will thus use Triple Exponential Smoothing, also known as Holt Winter Model. It takes into account level, trend and seasonal components.

Triple Exponential Smoothing / Holt Winter's Method :

In this method, we apply smoothing to seasonal component in addition to level and trend components. The smoothing is applied across seasons. That is, the fourth component of one season is smoothed against the fourth component of the previous season, fourth component of two seasons ago and so on. **Thus, there will be four equations — one each for level, trend, seasonality and the final equation with all the individual components.**

The equations vary given the model is additive or multiplicative. Honestly, one needs to put in significant effort to understand the math behind these equations. In case you want to go into more depth on the intuition behind the equations, you can refer to this link : https://grisha.org/blog/2016/02/17/triple-exponential-smoothing-forecasting-part-iii/

The four equations of the additive Holt Winter's Method is given as :

$$\text{(Level) } L_t = \alpha * (Y_t - S_{t-s}) + (1 - \alpha) * (L_{t-1} + b_{t-1})$$

$$\text{(Trend) } b_t = \beta * (L_t - L_{t-1}) + (1 - \beta) * b_{t-1}$$

$$\text{(Seasonal) } S_t = \gamma * (Y_t - L_t) + (1 - \gamma) * S_{t-s}$$

$$\text{(Forecast for period } m\text{) } F_{t+m} = L_t + m*b_t + S_{t+m-s}$$

Here s is the season length i.e. the number of data points after which a new season begins.

Lets see how to code the Triple Exponential Smoothing in Python. We will use the train data to model

```
[:928].values), seasonal_periods=12 ,trend='add',
seasonal='add',).fit()

fit1.summary()
```

The model output gives the optimal values of smoothing parameters alpha, beta and gamma.

Exponential Smoothing Model Results

Dep. Variable:	endog	No. Observations:	928
Model:	Exponential Smoothing	SSE:	2846.672
Optimized:	True	AIC:	1072.171
Trend:	Additive	BIC:	1149.500
Seasonal:	Additive	AICC:	1072.924
Seasonal Periods:	12	Date:	Wed, 21 Jul 2021
Box-Cox:	False	Time:	18:40:08
Box-Cox Coeff:	None		

	coeff	code
smoothing_level	**0.4085441**	**alpha**
smoothing_slope	**5.4582e-17**	**beta**
smoothing_seasonal	**0.3517790**	**gamma**
initial_level	58.340136	l.0
initial_slope	0.1026049	b.0
initial_seasons.0	-55.146903	s.0
initial_seasons.1	-55.161188	s.1
initial_seasons.2	-55.142231	s.2
initial_seasons.3	-55.137545	s.3
initial_seasons.4	-55.138415	s.4
initial_seasons.5	-55.097211	s.5
initial_seasons.6	-55.101285	s.6
initial_seasons.7	-55.096024	s.7
initial_seasons.8	-55.072809	s.8
initial_seasons.9	-55.106835	s.9
initial_seasons.10	-55.102856	s.10
initial_seasons.11	-55.147951	s.11

```
train_data = df1['Energy_Production'][:928]
test_data = df1['Energy_Production'][928:]
y_hat_avg = test_data.copy()
y_hat_avg['Holt_Winter'] = fit1.forecast(len(test_data))

rms = math.sqrt(mean_squared_error(test_data, y_hat_avg.Holt_Winter))
print(rms)
```

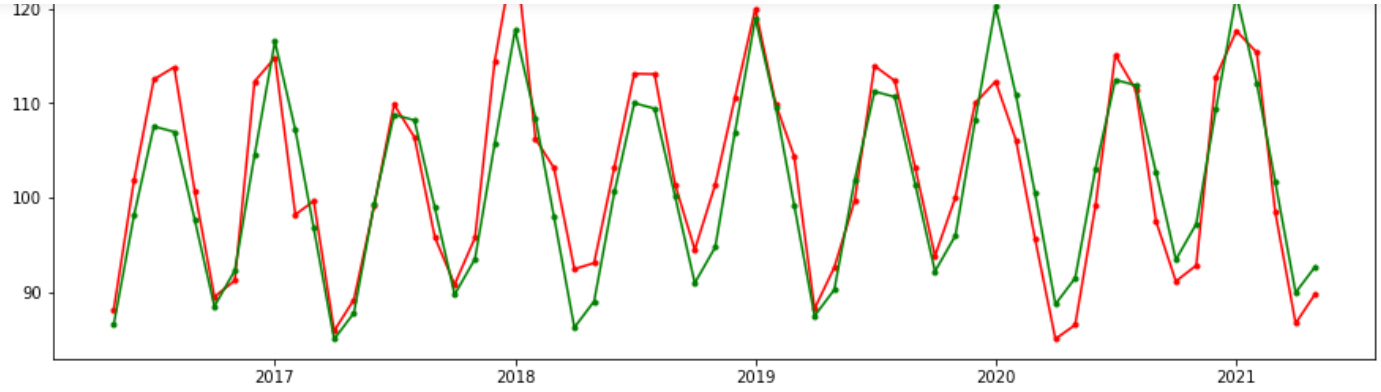
3.99167142334534447

```
get_mape(test_data, y_hat_avg.Holt_Winter)
```

3.27

Let us plot the actual and predicted series.

```
fig = plt.figure(figsize=(15,5));
future, = plt.plot(test_data.index, test_data, 'r.-', label='Actual
Output');
predicted_future, = plt.plot(test_data.index, y_hat_avg.Holt_Winter,
'g.-', label='Forecasted Output');
plt.legend(handles=[future, predicted_future]);
plt.title('Actual Production vs. Predicted Output');
```

We can see that the forecasted output closely follows the actual output. Much better performance as compared to Simple Moving Average method.

The forecast accuracy of Triple Exponential Smoothing Method is : RMSE of 3.99 and MAPE of 3.27. We can try more advanced techniques to optimize the metrics further.

This brings us to the end of the first part.

I will sum up the key points covered in this article:

1. In a time series data, the dependent variable is Y(t), observed at different points of time t.
2. A number of techniques like simple models, average and smoothing models, linear models and ARIMA models are used for forecasting time series data.
3. Metrics like MAPE and RMSE are more frequently used to evaluate the accuracy of the forecasting model.
4. Techniques like Simple and Weighted Moving Average are one of the simplest forecasting methods, however they are not suitable for data that has high seasonality and trend.
5. Smoothing techniques offer improvement over the moving average method. Here, past observations are assigned differential weights.

situations. Thus, there is merit in developing a number of models using different techniques before selecting the final model.

In Part 2, we will use more complex methods like ARIMA and its extension , SARIMA or seasonal ARIMA and also understand the terms like stationarity in detail.

Hope you found this article useful. I am open to any questions or suggestions.