# Multivariate timeseries analysis and prediction using missing data

## Table of Contents

# Introduction to Timeseries Concepts

## Time series and its importance

1. Time-Series(TS) is a sequence of measurements on the same variable collected over time.
2. It is a set of observations, each one being recorded at equally spaced time interval.
3. We extrapolate a past trend to predict future such as movement of short-term interest rate, capacity demand in airline and other sectors.

## Multivariate-time series analysis and prediction using incomplete data

4. TS data is collected for multiple variables over the same period of time. For example, a country's unemployment, GDP and inflation data over an interval period.
5. The TS data measures the changes over time. The time column in the data structure is used to sort the data.
6. TS analysis has played major role in areas like fraud detection, spam email filtering, finance, weather, healthcare, space exploration, manufacturing et al.

## How to predict future using TS?

It is difficult to make predictions, especially about the future : NEILS BOHR, Danish physicist

1. Simple deterministic model such as linear extrapolation
2. Complex deep learning approaches

# Multivariate Time Series (MTS)

A Multivariate time series has more than one time series variable. Each variable depends not only on its past values but also has some dependency on other variables. This dependency is used for forecasting future values. Sounds complicated? Let me explain.
Suppose our dataset includes perspiration percent, dew point, wind speed, cloud cover percentage, etc., and the temperature value for the past two years.
In this case, multiple variables must be considered to predict temperature optimally. A series like this would fall under the category of multivariate time series. Below is an illustration of this:

# TS data intervals

The interval depends on the nature of forecasting. For example, for a nation's GDP, we use yearly interval while sales data can be monthly and air-quality index being monitored on hourly basis.

# Challenges with time-series analysis

1. Data is not independent.
1. Data order is important to keep the data structure intact
2. Order is also important as data is list of observations

# Discussion on various models for prediction using missing data

## Vector Autoregression Model

It is a multivariate forecasting algorithm that is used when two or more time series influence each other.
It is considered as an Autoregressive model because, each variable (Time Series) is modelled as a function of the past values, that is the predictors are nothing but the lags (time delayed value) of the series.

in Autoregression models, the time series is modelled as a linear combination of its own lags. That is, the past values of the series are used to forecast the current and future.
A typical AR(p) model equation looks something like this:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_p Y_{t-p} + \epsilon_t$$

where α is the intercept, a constant and β1, β2 till βp are the coefficients of the lags of Y till order p.
Order 'p' means, up to p-lags of Y is used and they are the predictors in the equation.

The ε_{t} is the error, which is considered as white noise.

In the VAR model, each variable is modelled as a **linear combination of past values of itself and the past values of other variables in the system.** Since you have multiple time series that influence each other, it is modeled as a system of equations with one equation per variable (time series).
That is, if you have 5 time series that influence each other, we will have a system of 5 equations.

Well, how is the equation exactly framed?

Let's suppose, you have two variables (Time series) Y1 and Y2, and you need to forecast the values of these variables at time (t).

To calculate Y1(t), VAR will use the past values of both Y1 as well as Y2. Likewise, to compute Y2(t), the past values of both Y1 and Y2 be used.

For example, the system of equations for a VAR(1) model with two time series (variables `Y1` and `Y2`) is as follows:

$$Y_{1,t} = \alpha_1 + \beta_{11,1} Y_{1,t-1} + \beta_{12,1} Y_{2,t-1} + \epsilon_{1,t}$$
$$Y_{2,t} = \alpha_2 + \beta_{21,1} Y_{1,t-1} + \beta_{22,1} Y_{2,t-1} + \epsilon_{2,t}$$

Where, Y{1,t-1} and Y{2,t-1} are the first lag of time series Y1 and Y2 respectively.

The above equation is referred to as a VAR(1) model, because, each equation is of order 1, that is, it

contains up to one lag of each of the predictors (Y1 and Y2).

Since the Y terms in the equations are interrelated, the Y's are considered as endogenous variables, rather than as exogenous predictors.

Likewise, the second order VAR(2) model for two variables would include up to two lags for each variable (Y1 and Y2).

$$Y_{1,t} = \alpha_1 + \beta_{11,1} Y_{1,t-1} + \beta_{12,1} Y_{2,t-1} + \beta_{11,2} Y_{1,t-2} + \beta_{12,2} Y_{2,t-2} + \epsilon_{1,t}$$
$$Y_{2,t} = \alpha_2 + \beta_{21,1} Y_{1,t-1} + \beta_{22,1} Y_{2,t-1} + \beta_{21,2} Y_{1,t-2} + \beta_{22,2} Y_{2,t-2} + \epsilon_{2,t}$$

A second order VAR(2) model with three variables (Y1, Y2 and Y3) :

$$Y_{1,t} = \alpha_1 + \beta_{11,1} Y_{1,t-1} + \beta_{12,1} Y_{2,t-1} + \beta_{13,1} Y_{3,t-1} + \beta_{11,2} Y_{1,t-2} + \beta_{12,2} Y_{2,t-2} + \beta_{13,2} Y_{3,t-2} + \epsilon_{1,t}$$
$$Y_{2,t} = \alpha_2 + \beta_{21,1} Y_{1,t-1} + \beta_{22,1} Y_{2,t-1} + \beta_{23,1} Y_{3,t-1} + \beta_{21,2} Y_{1,t-2} + \beta_{22,2} Y_{2,t-2} + \beta_{23,2} Y_{3,t-2} + \epsilon_{2,t}$$
$$Y_{3,t} = \alpha_3 + \beta_{31,1} Y_{1,t-1} + \beta_{32,1} Y_{2,t-1} + \beta_{33,1} Y_{3,t-1} + \beta_{31,2} Y_{1,t-2} + \beta_{32,2} Y_{2,t-2} + \beta_{33,2} Y_{3,t-2} + \epsilon_{3,t}$$

Each variable is modelled as a **linear combination of past values of itself and the past values of other variables in the system.** Since you have multiple time series that influence each other, it is modeled as a system of equations with one equation per variable (time series).
That is, if you have 5 time series that influence each other, we will have a system of 5 equations.

In a VAR algorithm, each variable is a linear function of the past values of itself and the past values of all the other variables. To explain this in a better manner, I'm going to use a simple visual example:

We have two variables, y1, and y2. We need to forecast the value of these two variables at a time 't' from the given data for past n values. For simplicity, I have considered the lag value to be 1.

| Variable y1 | Variable y2 |
|---|---|
| | |
| $y1_{t-n}$ | $y2_{t-n}$ |
| ... | ... |
| $Y1_{t-2}$ | $Y2_{t-2}$ |
| $Y1_{t-1}$ | $Y2_{t-1}$ |
| $y1_{t}$ | $y2_{t}$ |

| Variable y1 | Variable y2 |
|---|---|
| | |
| $y1_{t-n}$ | $y2_{t-n}$ |
| ... | ... |
| $Y1_{t-2}$ | $Y2_{t-2}$ |
| $Y1_{t-1}$ | $Y2_{t-1}$ |
| $y1_{t}$ | $y2_{t}$ |

To compute y1(t), we will use the past value of y1 and y2. Similarly, to compute y2(t), past values of both y1 and y2 will be used. Below is a simple mathematical way of representing this relation:

$$y_1(t) = a_1 + w_1 11 * y_1(t-1) + w_1 12 * y_2(t-1) + e_1(t-1)$$
$$y_2(t) = a_2 + w_1 21 * y_1(t-1) + w_1 22 * y_2(t-1) + e_2(t-1)$$

Here,

- a1 and a2 are the constant terms,
- w11, w12, w21, and w22 are the coefficients,
- e1 and e2 are the error terms

These equations are similar to the equation of an AR process. Since the AR process is used for univariate time series data, the future values are linear

combinations of their own past values only. Consider the AR(1) process:

$$y(t) = a + w*y(t-1) + e$$

In this case, we have only one variable – y, a constant term – a, an error term – e, and a coefficient – w. In order to accommodate the multiple variable terms in each equation for VAR, we will use vectors. We can write equations (1) and (2) in the following form:

$$\begin{bmatrix} y1(t) \\ y2(t) \end{bmatrix} = \begin{bmatrix} a1 \\ a2 \end{bmatrix} + \begin{bmatrix} w11 & w12 \\ w21 & w22 \end{bmatrix} \begin{bmatrix} y1(t-1) \\ y2(t-1) \end{bmatrix} + \begin{bmatrix} e1(t) \\ e2(t) \end{bmatrix}$$

The two variables are y1 and y2, followed by a constant, a coefficient metric, a lag value, and an error metric. This is the vector equation for a VAR(1) process. For a VAR(2) process, another vector term for time (t-2) will be added to the equation to generalize for p lags:

$$\underset{K\,x\,1}{\begin{bmatrix} y1 \\ y2 \\ \vdots \\ yk \end{bmatrix}} = \underset{K\,x\,1}{\begin{bmatrix} a1 \\ a2 \\ \vdots \\ ak \end{bmatrix}} + \underset{K\,x\,K}{\begin{bmatrix} w11 & \cdots \\ w21 & \vdots \\ \vdots & \\ wk1 & \end{bmatrix}} \underset{K\,x\,1}{\begin{bmatrix} y1(t-1) \\ y2(t-1) \\ \vdots \\ yk(t-1) \end{bmatrix}} + \cdots\cdots \underset{K\,x\,K}{\begin{bmatrix} w'11 & \cdots \\ w'21 & \vdots \\ \vdots & \\ w'k1 & \end{bmatrix}} \underset{K\,x\,1}{\begin{bmatrix} y1(t-p) \\ y2(t-p) \\ \vdots \\ yk(t-p) \end{bmatrix}} + \begin{bmatrix} e1 \\ e2 \\ \vdots \\ ek \end{bmatrix}$$

The above equation represents a VAR(p) process with variables y1, y2 …yk. The same can be written as:

$$\begin{bmatrix} y \end{bmatrix} = \begin{bmatrix} a1 \end{bmatrix} + \begin{bmatrix} w1 \end{bmatrix} \begin{bmatrix} y1(t-1) \end{bmatrix} + \cdots\cdots \begin{bmatrix} wp \end{bmatrix} \begin{bmatrix} y1(t-p) \end{bmatrix} + \begin{bmatrix} e \end{bmatrix}$$

$$y(t) = a + w_1 * y(t-1) + .... + w_p * y(t-p) + \varepsilon t$$

The term $\varepsilon_t$ in the equation represents multivariate vector white noise. For a multivariate time series, $\varepsilon_t$ should be a continuous random vector that satisfies the following conditions:

1. $E(\varepsilon_t) = 0$

   The expected value for the error vector is 0

2. $E(\varepsilon_{t1}, \varepsilon_{t2}') = \sigma_{12}$

   The expected value of $\varepsilon_t$ and $\varepsilon_t'$ is the standard deviation of the series.

The procedure to build a VAR model involves the following steps:

1. Analyze the time series characteristics
2. Test for causation amongst the time series
3. Test for stationarity
4. Transform the series to make it stationary, if needed
5. Find optimal order (p)
6. Prepare training and test datasets
7. Train the model
8. Roll back the transformations, if any.
9. Evaluate the model using test set
10. Forecast to future

# Why do we need VAR models?

**VAR is able to understand and use the relationship between several variables.**

This is useful for describing the dynamic behaviour of the data and also provides better forecasting results. Additionally, implementing VAR is as simple as using any other univariate technique (which you will see in the last section).

The extensive usage of VAR models in finance, econometrics, and macroeconomics can be attributed to

their ability to provide a framework for achieving significant modeling objectives. With VAR models, it is possible to elucidate the values of endogenous variables by considering their previously observed values.

### Granger's Causality Test

Granger's causality test can be used to identify the relationship between variables prior to model building. This is important because if there is no relationship between variables, they can be excluded and modelled separately. Conversely, if a relationship exists, the variables must be considered in the modelling phase.

The test in mathematics yields a p-value for the variables. If the p-value exceeds 0.05, the null hypothesis must be accepted. Conversely, if the p-value is less than 0.05, the null hypothesis must be rejected.

# ARIMA model

AR - Auto Regression. I - Integrated. MA - Moving average. Auto-Regressive Integrated Moving Average (ARIMA) is a time series model that identifies hidden patterns in time series values and makes predictions. For example, an ARIMA model can predict future stock prices after analyzing previous stock prices.
Also, an ARIMA model assumes that the time series data is stationary. Before implementing the ARIMA model, we will remove the non-stationarity components in the time series.

A non-stationary time series is a series whose properties change over time. A non-stationary time series has trends and seasonality components. Removing the non-stationarity in a time series will make it stationary and apply the ARIMA model. The properties of time series that should remain constant are variance and mean. Allowing these properties to remain constant will remove the trend and seasonal components. We remove non-stationarity in a time series through differencing.

The differencing technique subtracts the present time series values from the past time series values. We may have to repeat the process of differencing multiple times until we output a stationary time series.

AR - Auto Regression. I - Integrated. MA - Moving average.

They have the following functionalities:

- Auto Regression sub-model - This sub-model uses past values to make future predictions.
- Integrated sub-model - This sub-model performs differencing to remove any non-stationarity in the time series.
- Moving Average sub-model. - It uses past errors to make a prediction.

These sub-models are parameters of the overall ARIMA model. We initialize the parameters using unique notations as follows:

- p: It is the order of the Auto Regression (AR) sub-model. It refers to the number of past values that the model uses to make predictions.
- d: It is the number of differencing done to remove non-stationary components.
- q: It is the order of the Moving Average (MA) sub-model. It refers to the number of past errors that an ARIMA Model can have when making predictions.

## What will we do here

We will build a multivariate time series model with Auto ARIMA and explore how the Auto ARIMA model works and how it automatically finds the best parameters of an ARIMA model. We will then implement the Auto ARIMA model.

# Why do we use Auto ARIMA?

Before we build an ARIMA model, we pass the p,d, and q values. We use statistical plots and techniques to find the optimal values of these parameters.

We also use statistical plots such as Partial Autocorrelation Function plots and AutoCorrelation Function plot.

The process of using statistical plots is usually hectic and time-consuming. Many people have difficulties interpreting these plots to find the optimal parameter values. Wrong interpretation leads to people not getting the best/optimal p,d, and q values. It affects the ARIMA model's overall performance.

Auto ARIMA automatically generates the optimal parameter values (p,d, and q). The generated values are the best, and the model will give accurate forecast results.

Auto ARIMA simplifies the process of building a time series model using the ARIMA model. Now we know how an ARIMA works and how Auto ARIMA applies its concepts. We will start exploring the time series dataset.

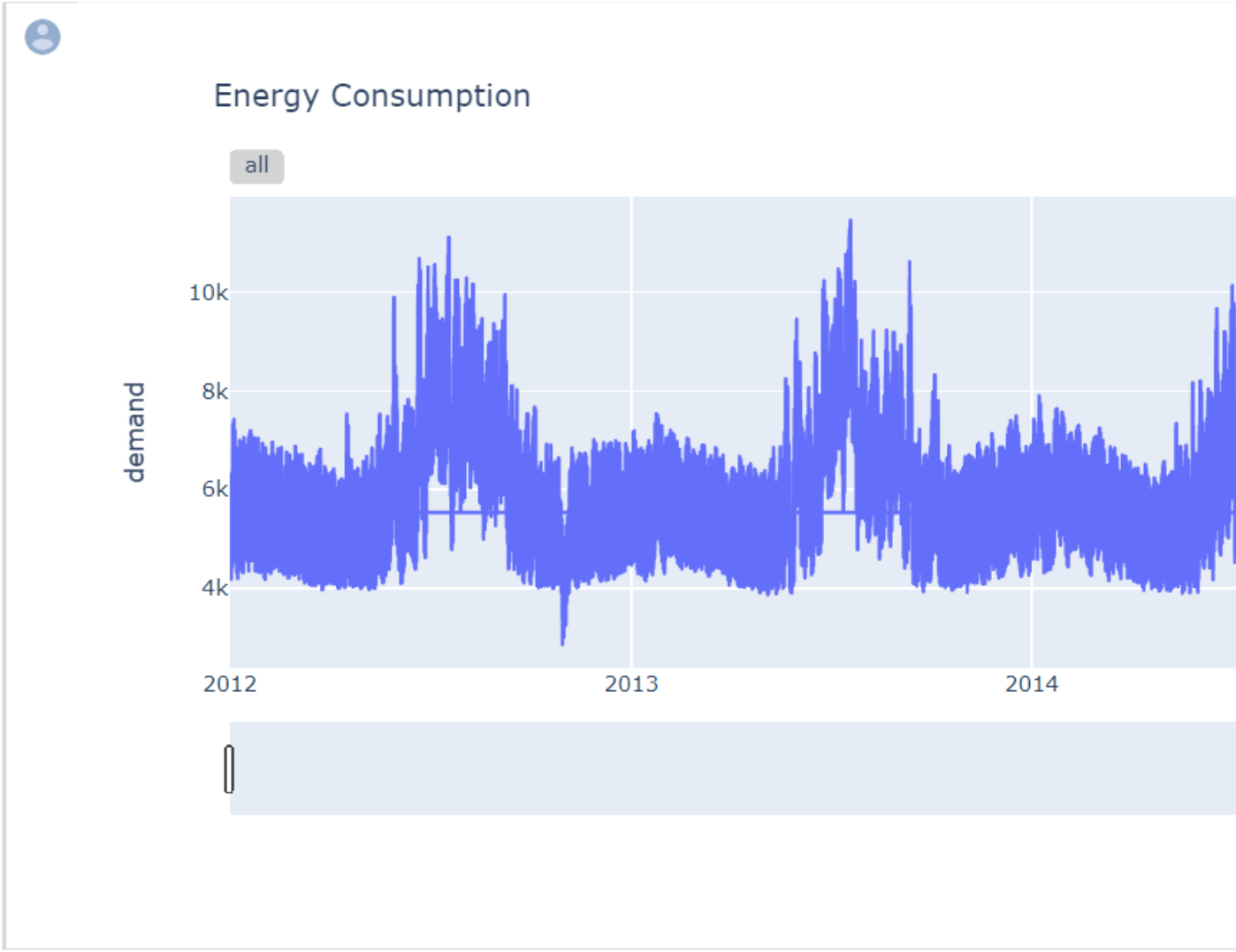
Data Set - https://drive.google.com/file/d/1l5MhAnlBYdp5Dk7EvcxzfGJFpvrwbbuw/view
The dataset shows the energy demand from 2012 to 2017 recorded in an hourly interval.

| | timeStamp | demand | precip | temp |
|---|---|---|---|---|
| 0 | 2012-01-01 00:00:00 | 4937.5 | 0.0 | 46.13 |
| 1 | 2012-01-01 01:00:00 | 4752.1 | 0.0 | 45.89 |
| 2 | 2012-01-01 02:00:00 | 4542.6 | 0.0 | 45.04 |
| 3 | 2012-01-01 03:00:00 | 4357.7 | 0.0 | 45.03 |
| 4 | 2012-01-01 04:00:00 | 4275.5 | 0.0 | 42.61 |

From this output, we have the timeStamp, demand, precip, and temp columns. The columns are the variables that will build the time series model.

The time series is multivariate since it has three-time dependent variables (demand, precip, and temp). They have the following functions:

- The timestamp column shows the time of recording.
- The demand column shows the hourly energy consumption.
- The precip and temp columns correlate with the demand column.




From the output above, the dataset has seasonality (repetitive cycles). Since the dataset has seasonality, we can say it is non-stationary. But still, we need to perform a statistical check using the Augmented Dickey-Fuller (ADF) test to assess stationarity in our dataset. The test is more accurate.

## Augmented Dickey Fuller test (ADF Test)

It is a common statistical test used to test whether a given Time series is stationary or not. It is one of the most commonly used statistical test when it comes to analyzing the stationary of a series.
There is a hypothesis testing involved with a null and alternate hypothesis and as a result a test statistic is computed and p-values get reported.

It is from the test statistic and the p-value, you can make an

inference as to whether a given series is stationary or not.

A Dickey-Fuller test is a unit root test that tests the null hypothesis that α=1 in the following model equation. alpha is the coefficient of the first lag on Y.

Null Hypothesis (H0): alpha=1

$$y_t = c + \beta t + \alpha y_{t-1} + \phi \Delta Y_{t-1} + e_t$$

where,

- y(t-1) = lag 1 of time series
- delta Y(t-1) = first difference of the series at time (t-1)

The ADF test expands the Dickey-Fuller test equation to include high order regressive process in the model.

$$y_t = c + \beta t + \alpha y_{t-1} + \phi_1 \Delta Y_{t-1} + \phi_2 \Delta Y_{t-2} .. + \phi_p \Delta Y_{t-p} + e_t$$

we have only added more differencing terms, while the rest of the equation remains the same. This adds more thoroughness to the test.

The null hypothesis however is still the same as the Dickey Fuller test.

A key point to remember here is: Since the null hypothesis assumes the presence of unit root, that is α=1, the p-value obtained should be less than the significance level (say 0.05) in order to reject the null hypothesis. Thereby, inferring that the series is stationary.

The `statsmodel` package provides a reliable implementation of the ADF test via the `adfuller()` function in `statsmodels.tsa.stattools`. It returns the following outputs:

1. The p-value
2. The value of the test statistic
3. Number of lags considered for the test
4. The critical value cutoffs.

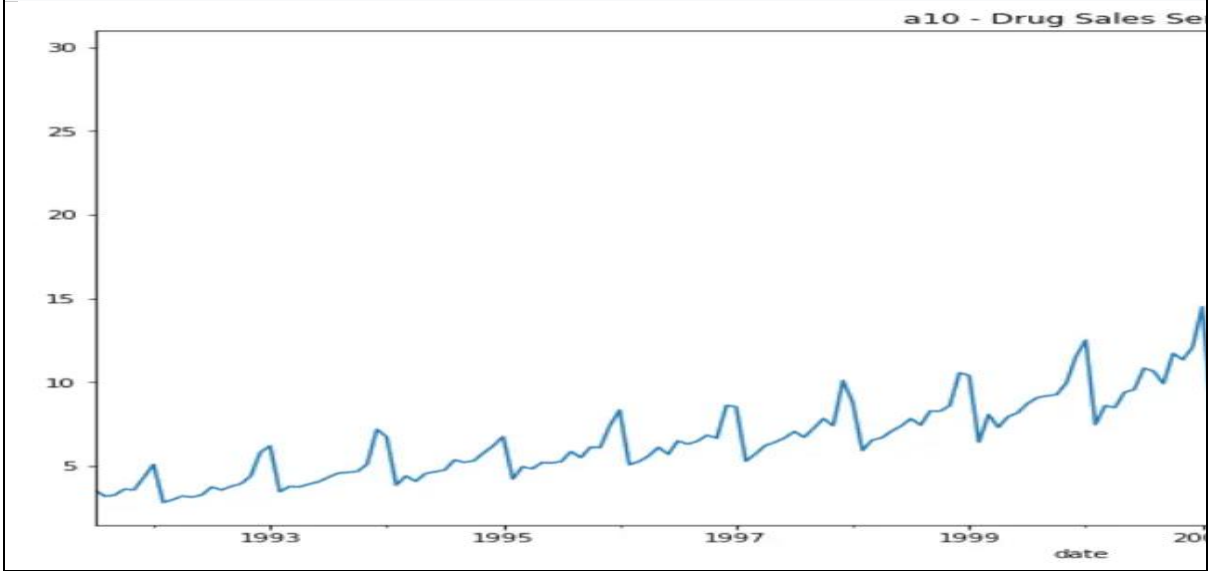
When the test statistic is lower than the critical value shown, you reject the null hypothesis and infer that the time series is stationary.

Alright, let's run the ADF test on the `a10` dataset from the `fpp` package from R. This dataset counts the total monthly scripts for pharmaceutical products falling under ATC code A10. The original source of this dataset is the Australian Health Insurance Commission.

As see earlier, the null hypothesis of the test is the presence of unit root, that is, the series is non-stationary.

```
# Setup and Import data
from statsmodels.tsa.stattools import adfuller
import pandas as pd
import numpy as np
%matplotlib inline

url =
'https://raw.githubusercontent.com/selva86/datasets/master/a1
0.csv'
df = pd.read_csv(url, parse_dates=['date'], index_col='date')
series = df.loc[:, 'value'].values
df.plot(figsize=(14,8), legend=None, title='a10 - Drug Sales
Series');
```




```
# ADF Test
result =
adfuller(series,
autolag='AIC')
print(f'ADF
Statistic:
{result[0]}')
print(f'n_lags:
{result[1]}')
print(f'p-value:
{result[1]}')
for key, value in
result[4].items():
    print('Critial
Values:')
    print(f'   {key},
{value}')
```

```
ADF Statistic:
3.1451856893067296
n_lags: 1.0
p-value: 1.0
Critial Values:
   1%, -
3.465620397124192
Critial Values:
   5%, -
2.8770397560752436
Critial Values:
   10%, -
2.5750324547306476
```

The p-value is obtained is greater than significance level of 0.05 and the ADF statistic is higher than any of the

| | | |
|---|---|---|
| | | critical values.<br><br>Clearly, there is no reason to reject the null hypothesis. So, the time series is in fact non-stationary. |

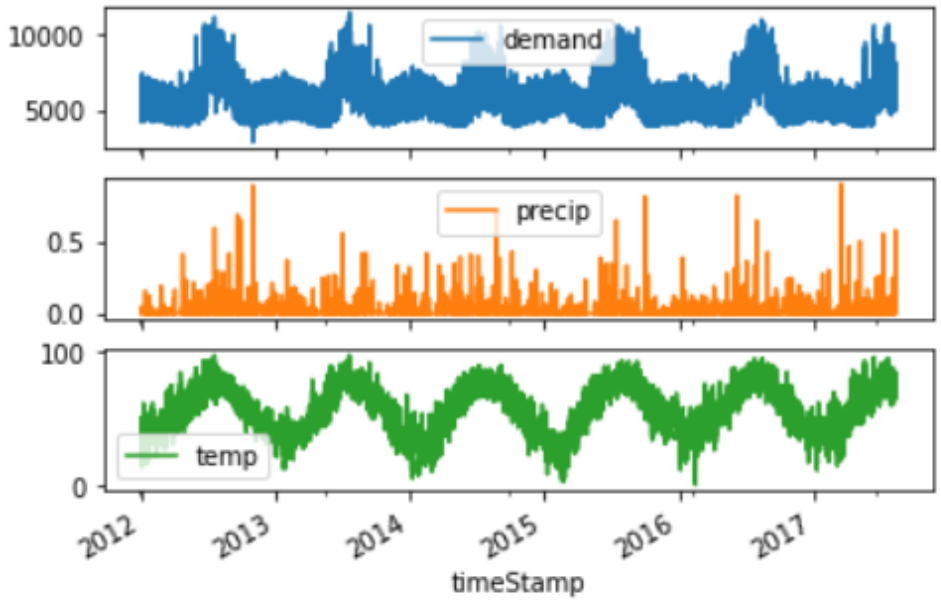
## Implementing the Auto ARIMA model

If we find the dataset is non-stationary after the ADF test, we will have to perform differencing to make it stationary. Auto ARIMA performs differencing automatically.

We implement the Auto ARIMA model using the pmdarima time-series library. This library provides the auto_arima() function that automatically generates the optimal parameter values.

The subplots will show the time-dependent variables in the dataset. We will visualize the demand, precip, and temp columns.

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f691fd2a7
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f691fce0e
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f691fd2a3
      dtype=object)
```




## Dataset Cleanup

We need to check for missing values in the dataset. Missing values affects the model and leads to inaccurate forecast results.

## Dataset resampling

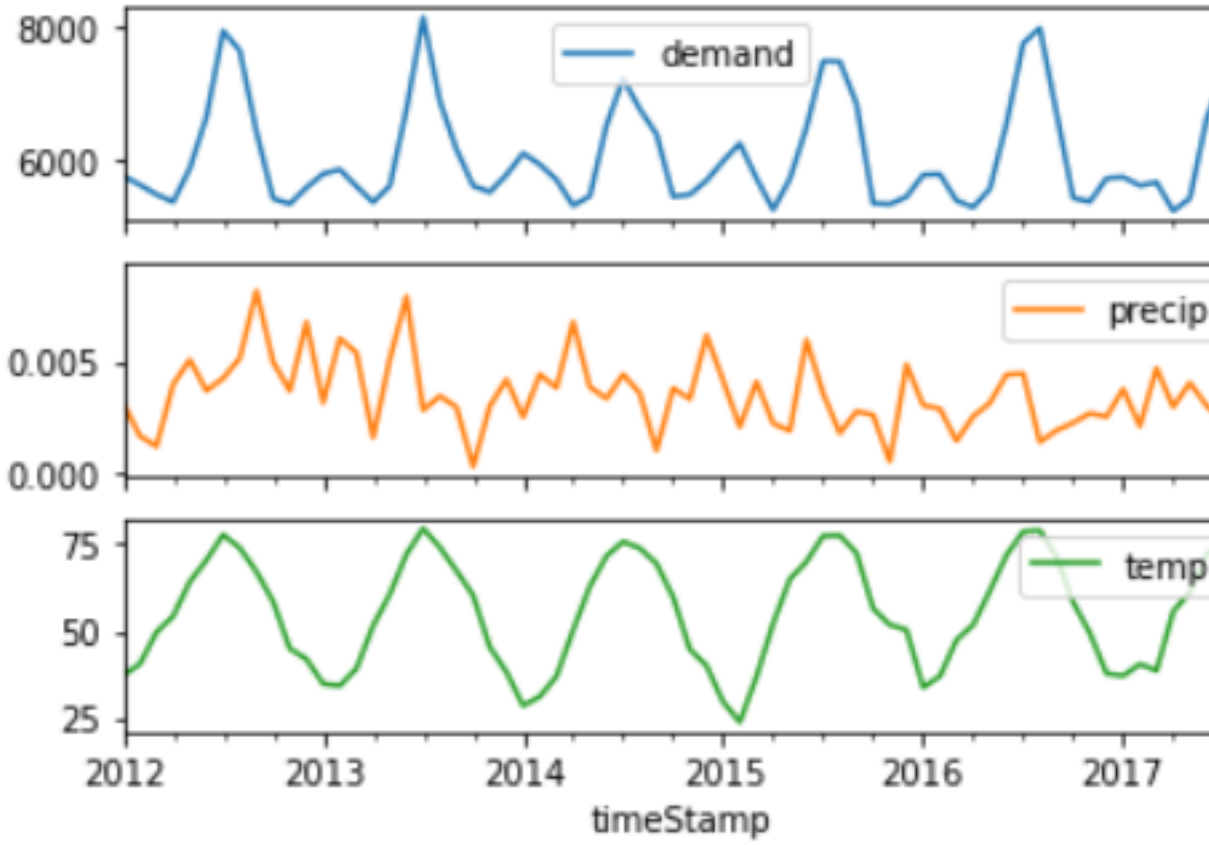
The time series has many data points that may be difficult to analyze and visualize. We need to resample the time by compressing and aggregating it to monthly intervals. We will have fewer data points that are easier to analyze.

The resample() method will aggregate all the data points in the time series and change them to monthly intervals.

| timeStamp | demand | precip | temp |
|---|---|---|---|
| **2012-01-31** | 5757.495833 | 0.003116 | 37.174288 |
| **2012-02-29** | 5622.331609 | 0.001650 | 40.448046 |
| **2012-03-31** | 5479.919198 | 0.001179 | 49.607917 |
| **2012-04-30** | 5368.945833 | 0.003959 | 54.255903 |
| **2012-05-31** | 5867.896102 | 0.005064 | 64.188078 |
| **...** | ... | ... | ... |
| **2017-04-30** | 5233.477382 | 0.002942 | 55.621764 |
| **2017-05-31** | 5421.773060 | 0.004031 | 61.115457 |
| **2017-06-30** | 6597.990346 | 0.003008 | 71.962625 |
| **2017-07-31** | 7306.861511 | 0.002266 | 76.380363 |
| **2017-08-31** | 6711.707542 | 0.008961 | 73.730258 |

68 rows × 3 columns

```
[19]
array([<matplotlib.axes._subplots.AxesSubplot ob
       <matplotlib.axes._subplots.AxesSubplot ob
       <matplotlib.axes._subplots.AxesSubplot ob
      dtype=object)
```




## Splitting the TS Series

We split the time series dataset into a training data frame and a test data frame.
The code selects the data points from 2012-01-31 to 2017-04-30 for model training.
The data points from 2017-04-30 are for model testing.

## Fitting the auto Arima model

Fitting the Auto ARIMA model to the train data frame will enable the model to learn from the time-series dataset. The final model will make future predictions.

```
model.fit(train['demand'])
```

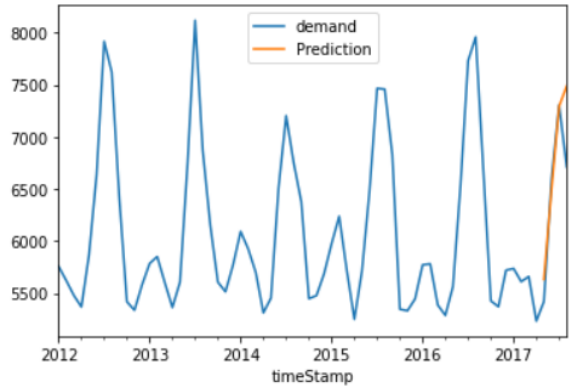
# Using auto Arima model for predictions

The Auto ARIMA model will predict using the test data frame. It will also forecast/predict the unseen future time series values.

```
forecast=model.predict(n_periods=4, return_conf_int=True)
```

| timeStamp | Prediction |
|---|---|
| 2017-05-31 | 5633.941335 |
| 2017-06-30 | 6492.114538 |
| 2017-07-31 | 7294.397892 |
| 2017-08-31 | 7488.320679 |

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f68fe165850>
```



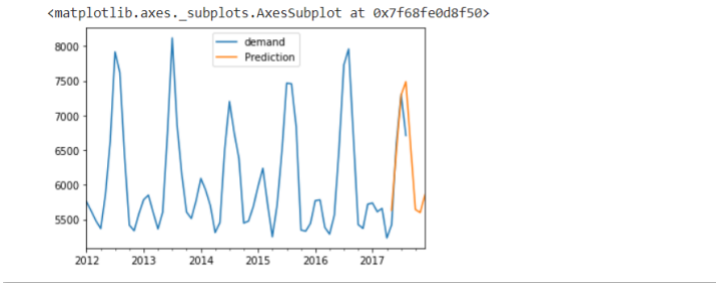

From the line chart above:

- The blue line is the actual energy demand.
- The orange line is the predicted energy demand.

The Auto ARIMA model has performed well and has made accurate predictions. The blue and orange lines are close to each other.

We can now use this model to predict unseen future values.

## Predicting and plotting the unseen future time series values




From the line chart above:

- The blue line represents the actual energy demand.
- The orange line represents the predicted energy demand.

The orange line also shows the unseen future predictions. The Auto ARIMA model has performed well since the orange line maintains the general pattern.

# Difference between VAR and Auto-Arima

The primary difference is ARIMA models are unidirectional, where, the predictors influence the Y and not vice-versa.
Whereas, Vector Auto Regression (VAR) is bi-directional. That is, the variables influence each other.

# Why DL approach has performed not so well for univariate time series forecasting compared to naive and classical forecasting methods?

Deep learning (DL) approaches have shown remarkable success in various fields, including computer vision, natural language

processing, and speech recognition. However, when it comes to univariate time series forecasting, DL models have not always outperformed classical and naive forecasting methods.

There are several reasons why DL approaches may not perform as well for univariate time series forecasting:

Lack of data: DL models require a large amount of data to train effectively. In some cases, univariate time series data may not be abundant, making it challenging for DL models to learn patterns and make accurate predictions.

Overfitting: DL models are prone to overfitting when the training data is not representative of the test data. This is especially problematic when working with time series data, as there may be hidden patterns and trends that are not captured in the training data.

Complexity: DL models can be very complex and have many parameters to tune. This can make it challenging to find the optimal architecture and hyperparameters for a particular time series problem.

Interpretability: DL models can be challenging to interpret, making it difficult to understand why they are making certain predictions. This can be a significant drawback in some applications, where interpretability is crucial.

Performance metric: DL models are often evaluated using mean squared error (MSE) or mean absolute error (MAE). While these metrics are useful for measuring the accuracy of point forecasts, they may not be the best metrics for evaluating the overall performance of a time series forecasting model.

In contrast, classical and naive forecasting methods, such as ARIMA and exponential smoothing, have been developed specifically for time series forecasting and have been shown to perform well in many cases. These methods are often simpler to implement and interpret than DL models and can be effective when working with smaller datasets. Additionally, classical and naive methods often have well-established performance metrics, such as AIC or BIC, that can be used to evaluate the overall performance of a model.

Overall, DL approaches can be a powerful tool for time series forecasting, but they may not always outperform classical and naive methods, especially when working with small datasets or when interpretability