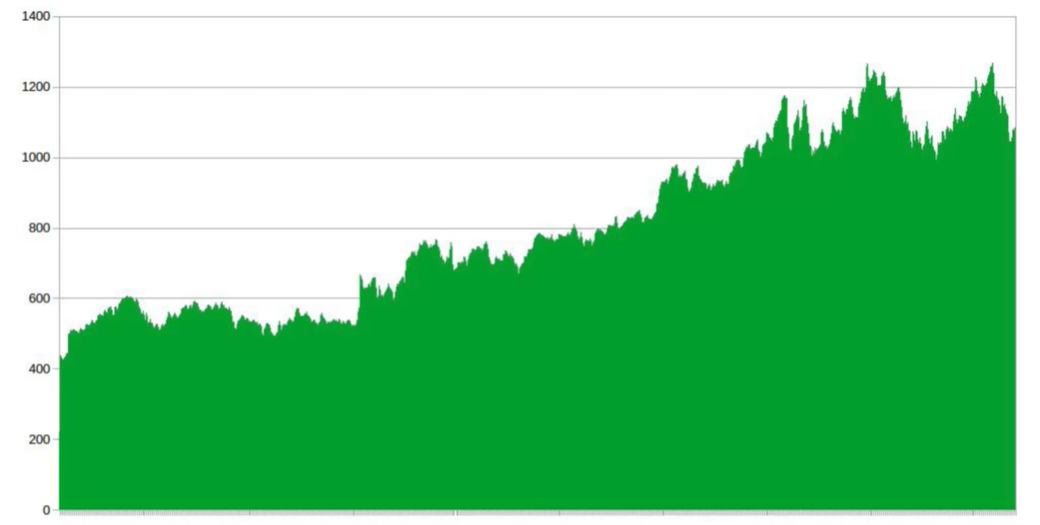
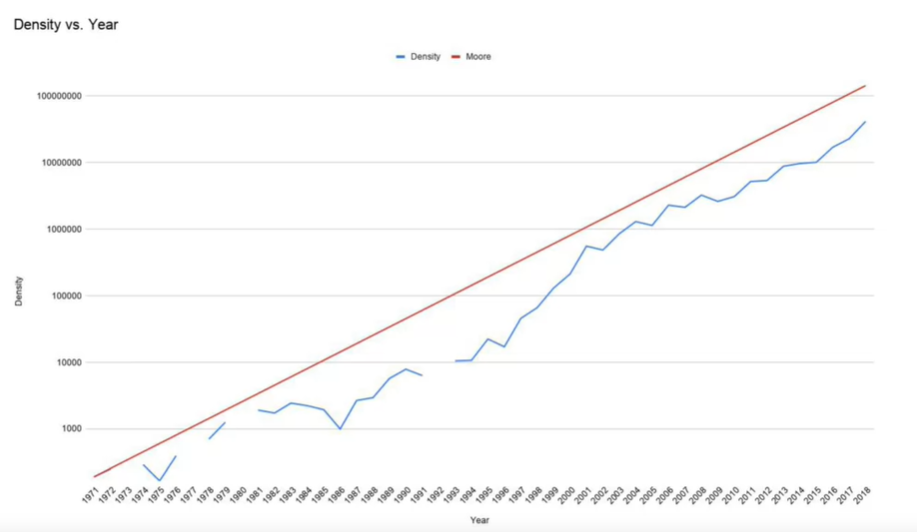
## Example of Time Series :

#### Stock prices :



#### Historical trends :



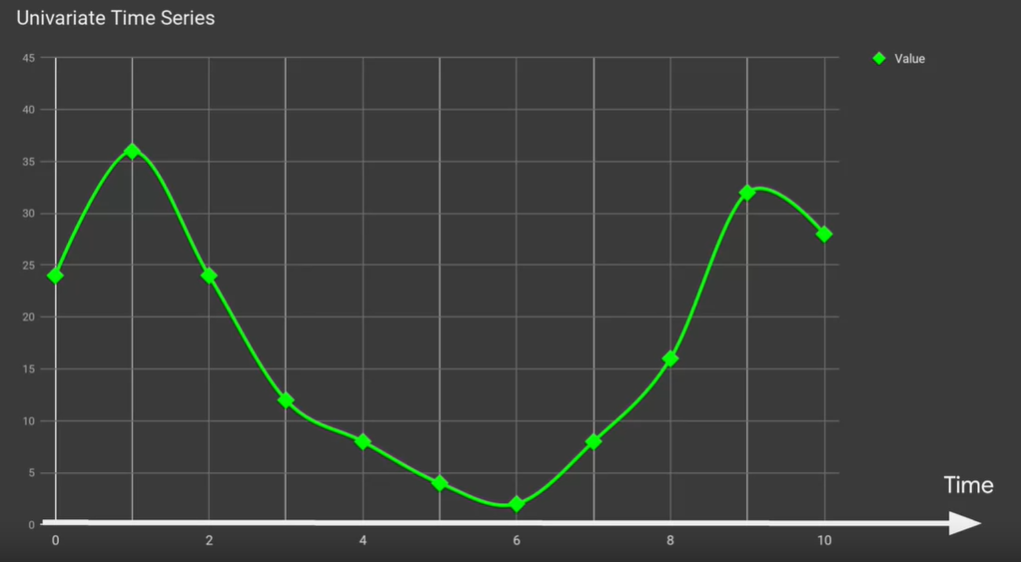
#### Weather forcast :



## Definition of Time Series :

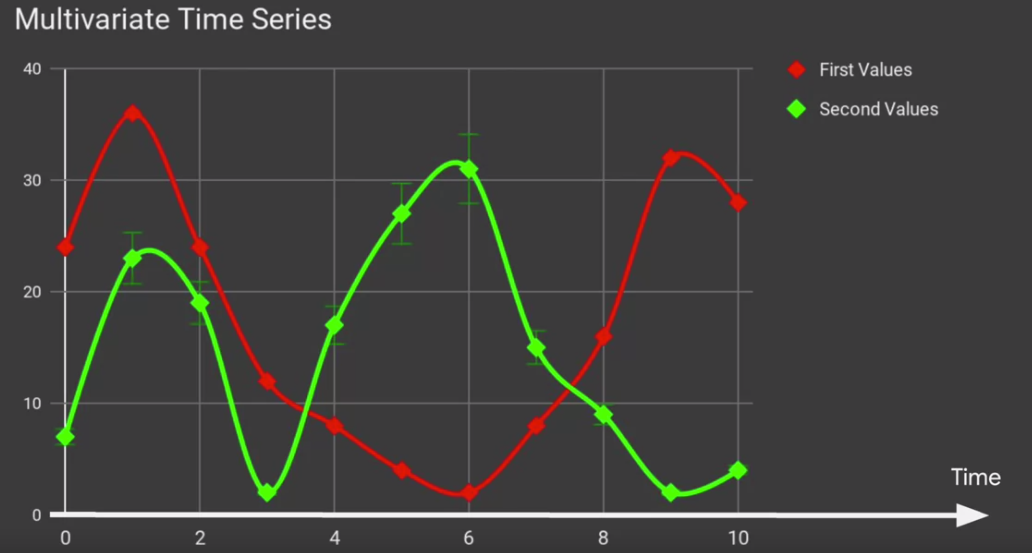
It's typically defined as an ordered sequence of values that are usually equally spaced over time ( every day , year , trimester … ) , the time is always the X-axis .

#### Univariate Time Series :



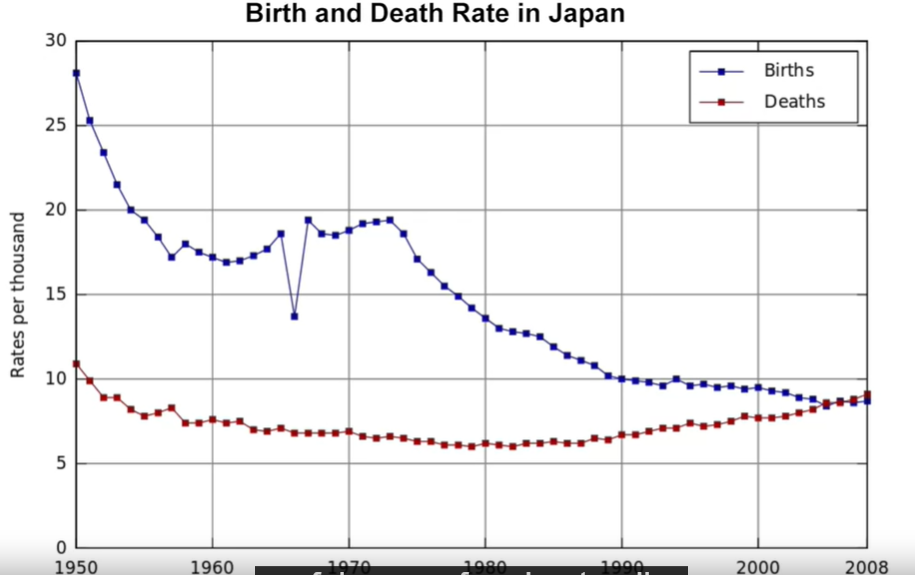
* In the plot we are observing the progression of only one factor

#### Multivariate Time Series :



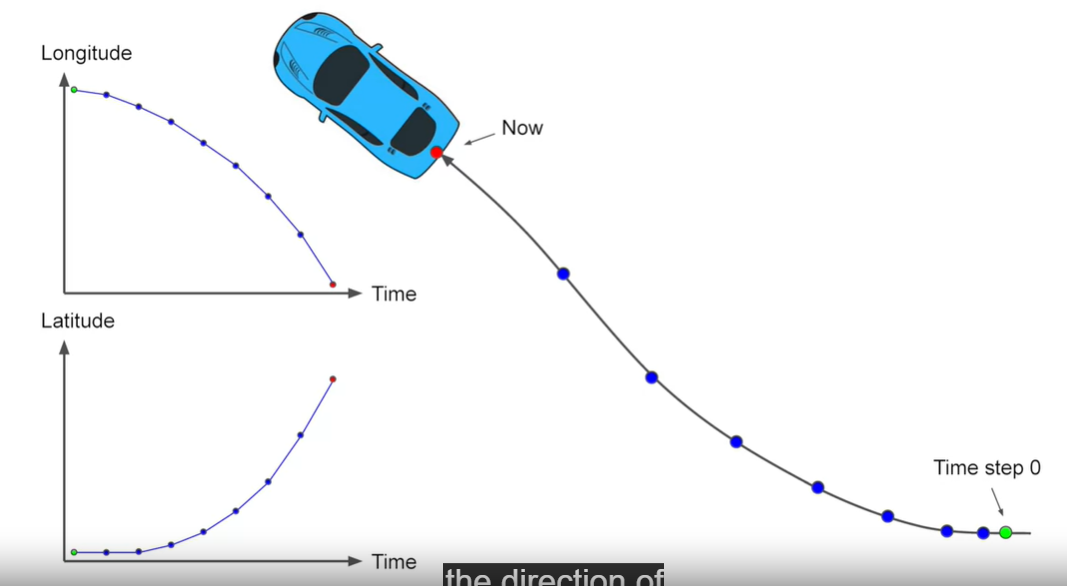
* We have multiple factors to observe at each timestep , This is useful in order to discover the relation ( the correlation ) between the factors

And the Multivariate Time Series of number of deaths/borns in Japan is an interesting example:



* If we don’t plot the two factors in the same chart , We cannot notice that by the end of 2008 , the number of deaths in Japan become superior than the born ones which show the demographic reduction in Japan ( so , We should take a special measurements for that like be opened to the immigration ..etc )

The movement of an object in the map is considered as separated multivariant Time Series , where the first factor is the development for the object Longitude by the time and the second graph is for the Latitude development :

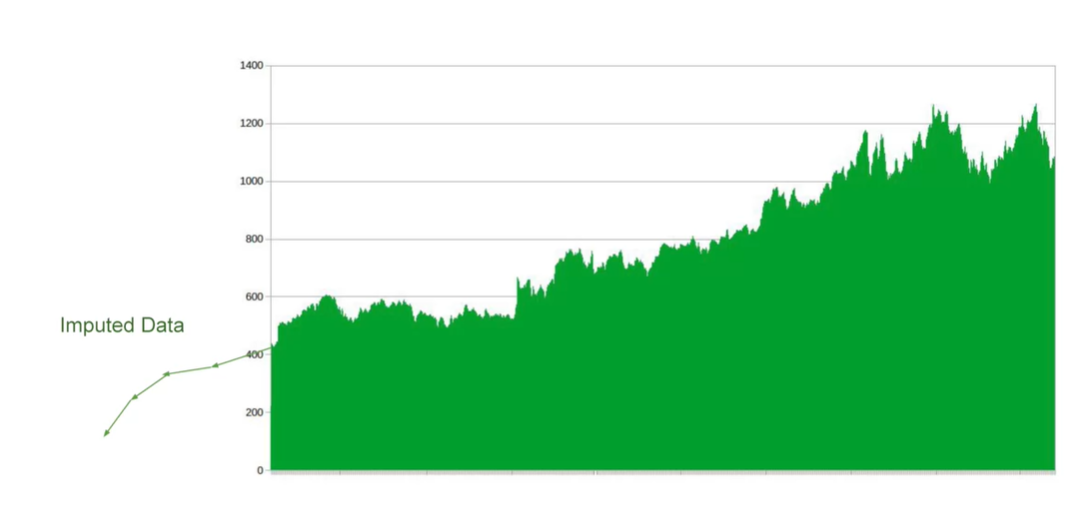


## Machine Learning in Time Series :

### Forecasting:

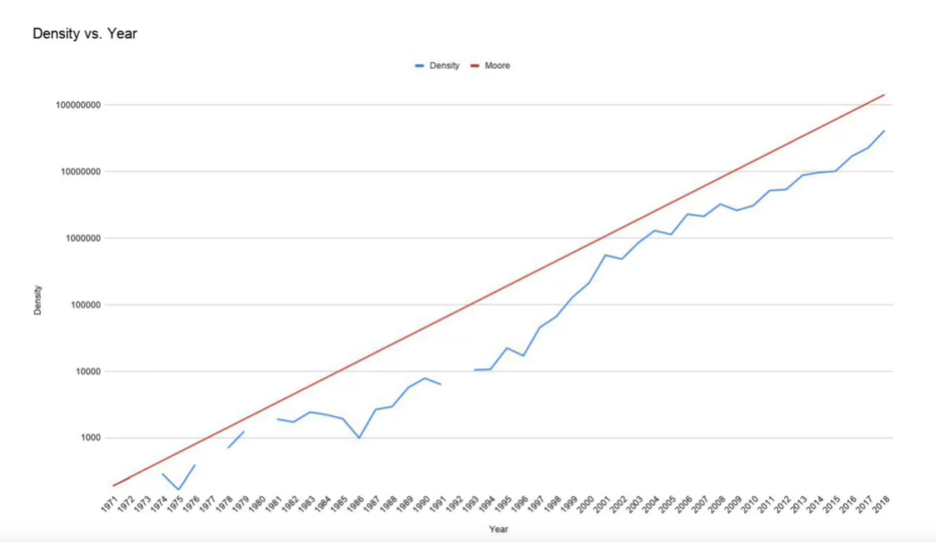


### Imputation of the past:



* It is the opposite of the forecasting : predicting the past data

### Imputation of the holes :



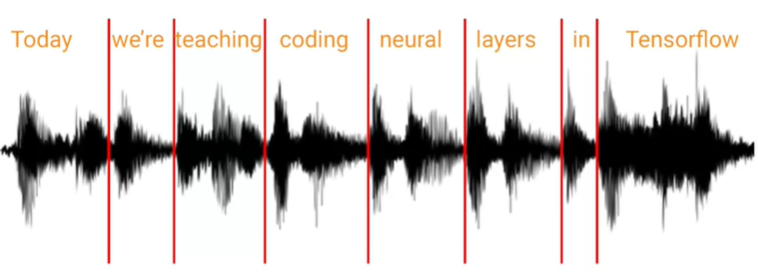
* In the example above , The blue chart has holes which presented not filled data , We can use ML to fit these holes

### Anomalies detection :



* The anomalies in Time series are shown as spikes

### Spot the patterns ( in Audio data for speech recognition )



* By analyzing the sound waves we can spot words in them by doing the split

## Common patterns in the Time Series

### Trends :

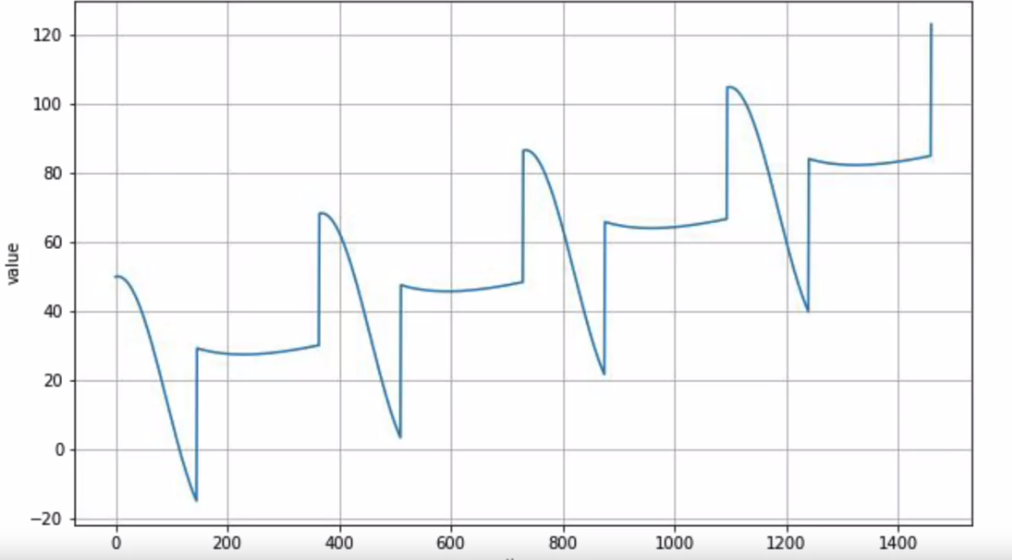
### 

* These two charts follows the Trends pattern , in this pattern the time series follows a specific direction ( upward/downward )

### Seasonality

* In the Seasonality pattern : a specific pattern is repeated in a predictable pattern
* In this example , we see that the same pattern is repeated every week where in each weekend have a decreased values than the values returned to the normal values in the week days … It’s represented the number of visits in an educational website

### Combination of Trends and Seasonality :



### White Noise

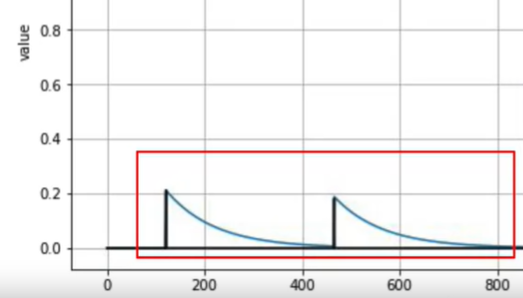


* White Noise are unpredictable data because they are produced by a random data without any clear pattern

### Auto Correlation :

The term autocorrelation refers to the degree of similarity between A) a given time series, and B) a lagged version of itself, over C) successive time intervals. In other words, autocorrelation is intended to measure the relationship between a variable’s present value and any past values that you may have access to.

* Data that follows a predictable shape, even if the scale is different

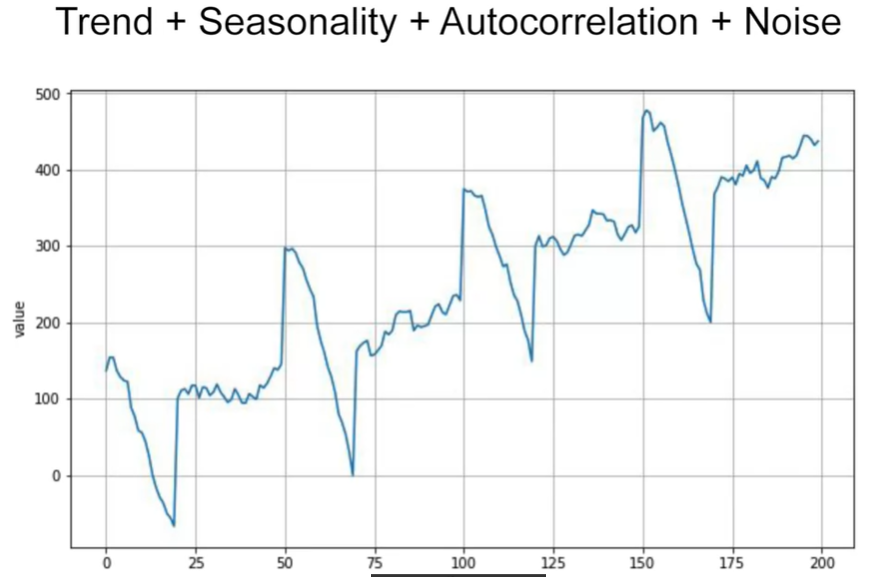


* We can see that these two lags have a similar pattern with a slight change , We can formulate it by :



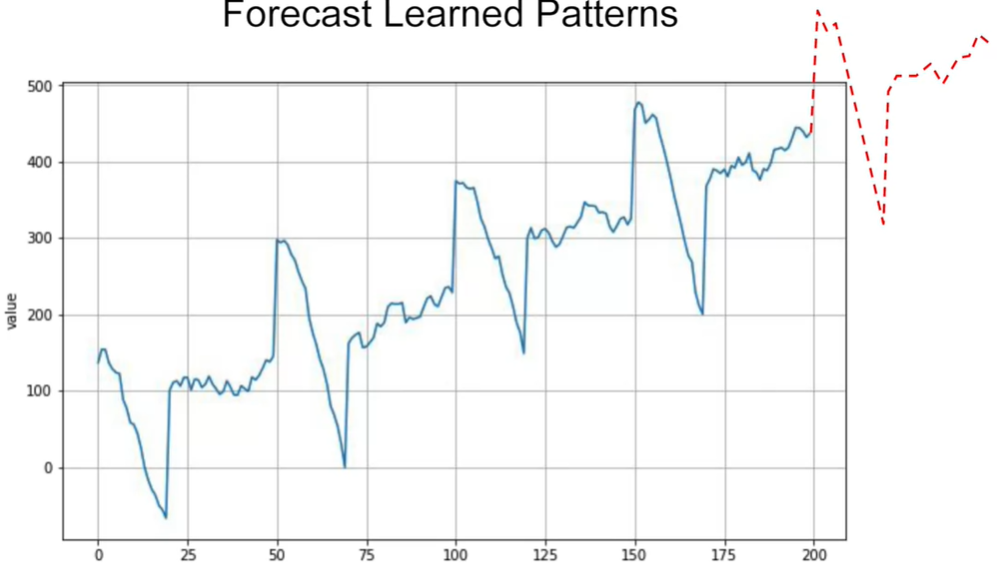
We can see that the graph is autocorrelated with its lagged version ( where the v(t-1) is correlated with v(t) )

### Trends + Seasonality + Noise + Auto Correlation

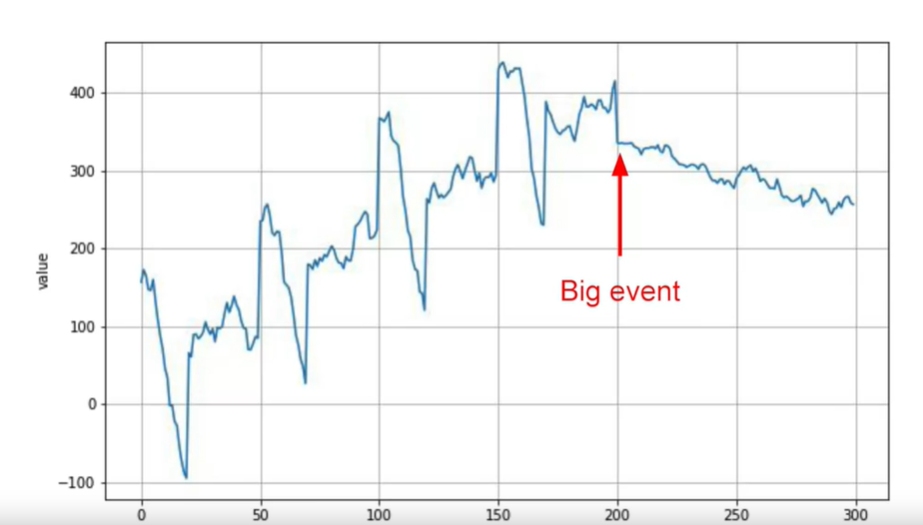


## Forecast Learned patterns:

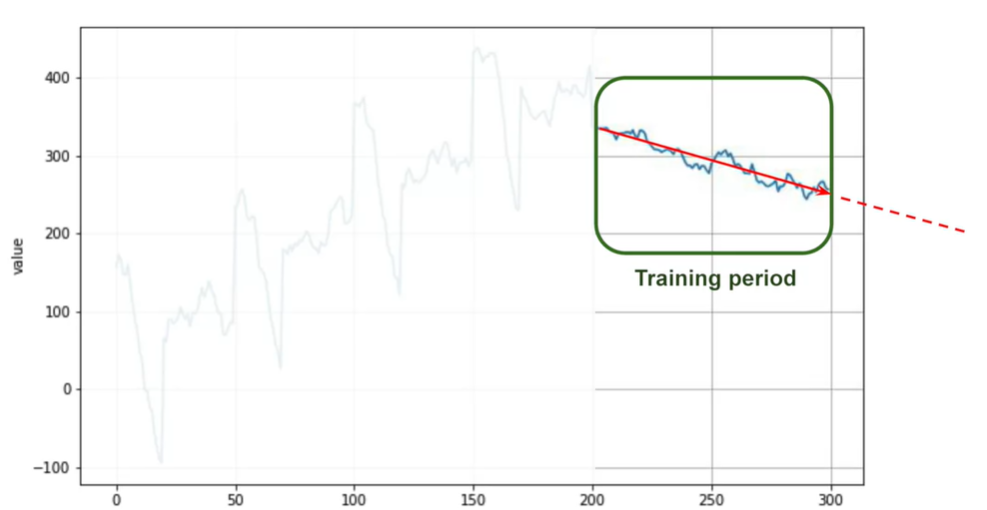
When our model learned the time series pattern , Then he can do its forecasting on respecting the data pattern like the image below :



### The data isn’t always Statinary , It can be No-Stationary Data :

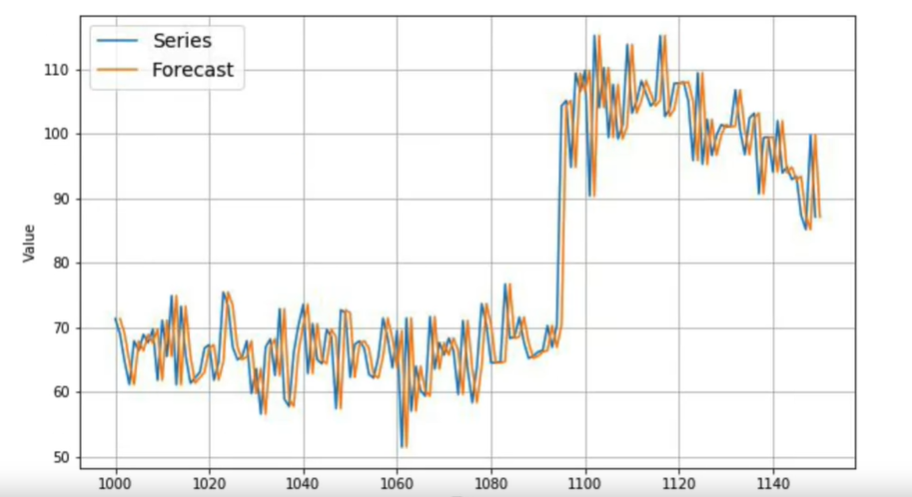


* real life time series are not always that simple. Their behavior can change drastically over time. For example, this time series had a positive trend and a clear seasonality up to time step 200. But then something happened to change its behavior completely. If this were stock, price then maybe it was a big financial crisis or a big scandal or perhaps a disruptive technological breakthrough causing a massive change. After that the time series started to trend downward without any clear seasonality. We'll typically call this a non-stationary time series. To predict on this we could just train for limited period of time. For example, here where I take just the last 100 steps. You'll probably get a better performance than if you had trained on the entire time series. But that's breaking the mold for typical machine, learning where we always assume that more data is better. But for time series forecasting it really depends on the time series. If it's stationary, meaning its behavior does not change over time, then great. The more data you have the better. But if it's not stationary then the optimal time window that you should use for training will vary. Ideally, we would like to be able to take the whole series into account and generate a prediction for what might happen next. As you can see, this isn't always as simple as you might think given a drastic change like the one we see here.
* In this case, we will extract only the data that comes after this big event to do our prediction:

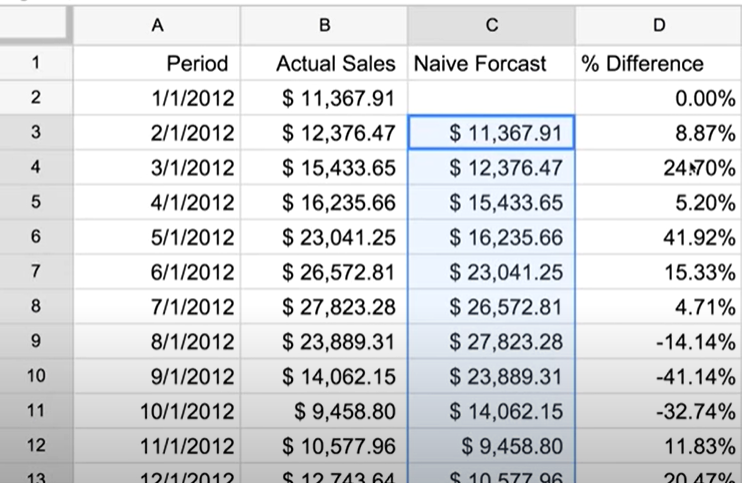


## Train, Validation and Test :

### Naïve Forecasting :



* This approach is called Naïve because there isn’t any learning approach behind , We will just assume that v(t+1) = v(t) ( and that’s why we see that the Forecast chart is shifted by one timestep from the original series .
* Here is a concrete example of how the predicted values are calculated:

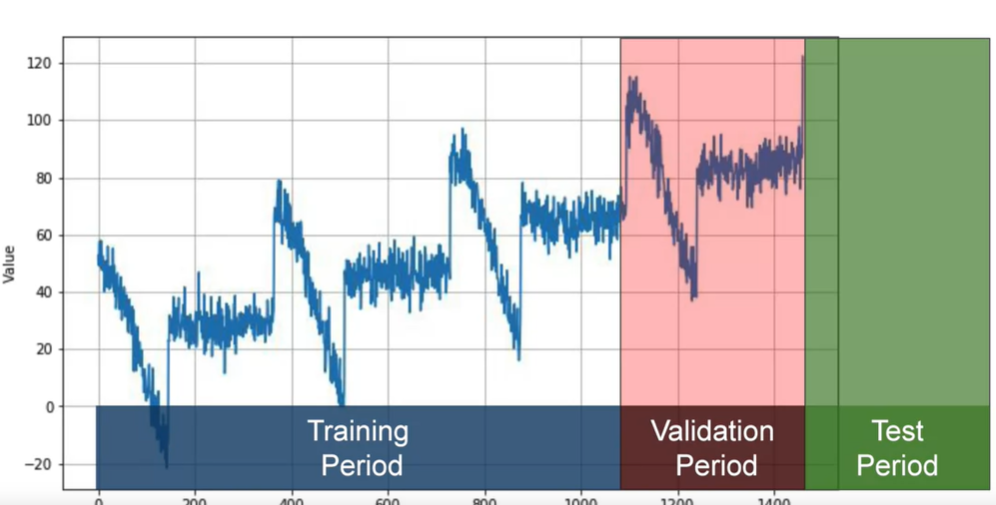


### Splitting the data:

#### Fixed partitioning:

### 

1. We generally split the data into Training period , validation period and Test period : this is the fixed partitioning
2. If the data has seasonality like the above one , we must ensure that each period contain the whole season ( not just a part of it ) for example if the seasonality is a year , we don’t want to have in a period : a year and a half or two years and a half for example
3. We will then train our model in the Training period and then we will evaluate our model in the Validation period, thanks to validation period we can evaluate the accuracy of our model and try to find the most suitable architecture to our problem (working in hyper parameters also).



1. After having a nice validation accuracy, we can merge the Training and the validation periods in a one big period and do our retraining in this merged big period, and we will evaluate the model accuracy finally in the Test period
2. If we got a nice accuracy In the test period also, We can merge the whole data into one big training data, this is important because the data contained in the last timesteps is the most important data: it’s the closest one to the timestep to predict its data. so, it contains the strongest signal to have an optimal prediction
3. And it’s common to see that the time series prediction process skip the first 3 steps and start from the 4th step where we took the whole data as a training period except the latest data which we will take it as a Validation period ( because it’s the closest one to the value to predict ) : exactly like the image above

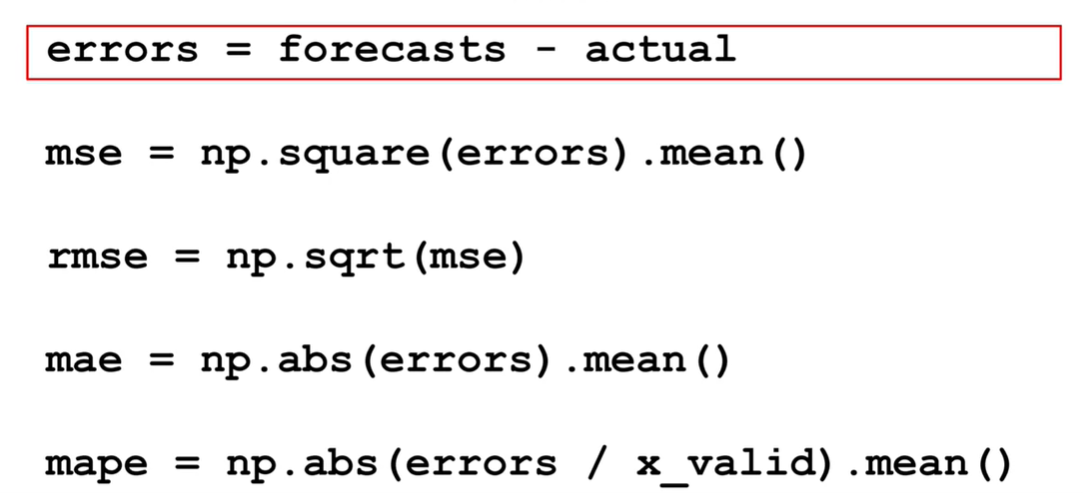
#### Roll-Forward Partitioning:

* In the Roll Forward dimension , We have different Training and validation periods
* In the first iteration, it’s exactly like the fixed partitioning process , we fix a training and validation period : we learn from the training period and try to predict for the validation perido than we compare the generated prediction to the real data in the validation period
* Then, for each iteration : we shift the training period to the right ( to the future ) by a fixed step ( day , week , season , …etc , it depends on the problem ) and we repeat the same process of validation and training period
* So for each further step , we got a bigger period for training period and the Valdiation period keeps shrinking

#### Generally , Roll-forward partitioning is better than the fixed portioning :

* We got multiple examples of the generated prediction for the future ( each one has its own forecast value and its own error )
* By roll-forward partitioning we are mimicking and simulating the deployment scenario where the training period keeps being bigger ( by gathering the real time data ) and using it to predict the future

## The used metrics to evaluate the performance of Time Series prediction

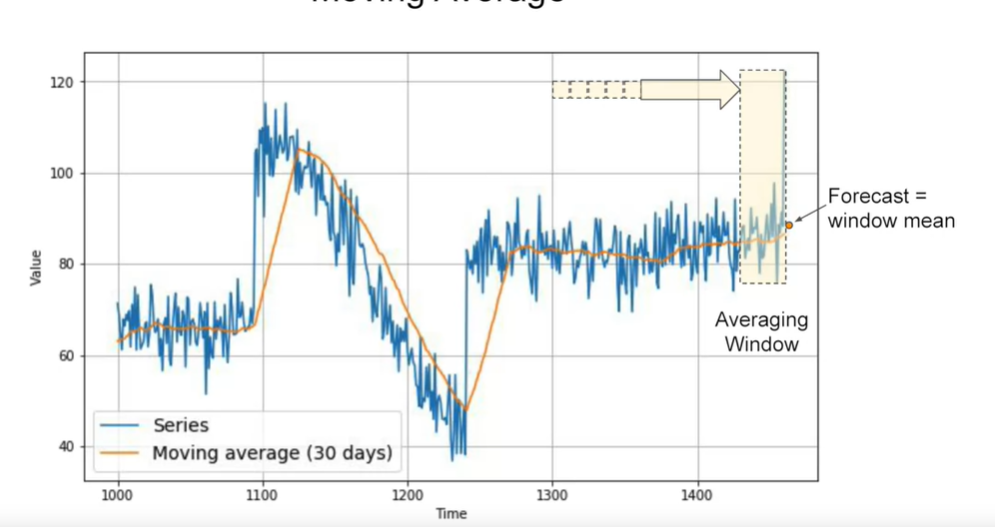


* The most popular ones are RMSE ( Root Mean Squared Error ) and MAE ( Mean Absolute Error )
* RMSE is useful than the MSE because it gives a value in the same scale of the original errors

#### RMSE vs MAE :

* The Square in RMSE is used to penalize the big errors by the square operation while the MAE didn’t give any particular penalization in the error calculating

## Moving Average:



* Instead of considering v(t) as a prediction for v(t+1 ) like we do in Naïve Forecasting , We are going to calculate the mean of values of a fixed period right before the forecast ( it’s the Averaging window in the picture above ), the mean value is our Forecast : the window mean
* If we fixed the period for moving average by 30 Days then :

**V(t+1) = ( V(t) + V(t-1) +…….+ V(t-30) ) / 30**

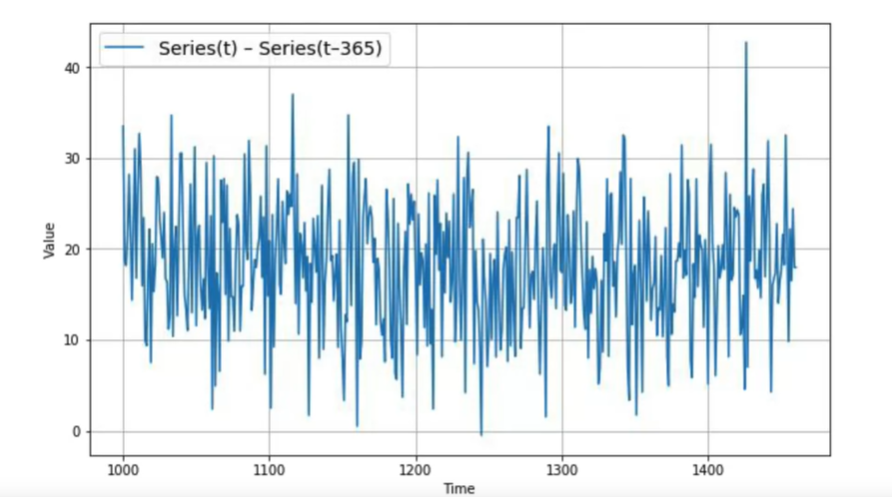
#### Advantages ;

* It eliminates and it’s resistible to the noise in the graph and it simulates nicely the overall graph

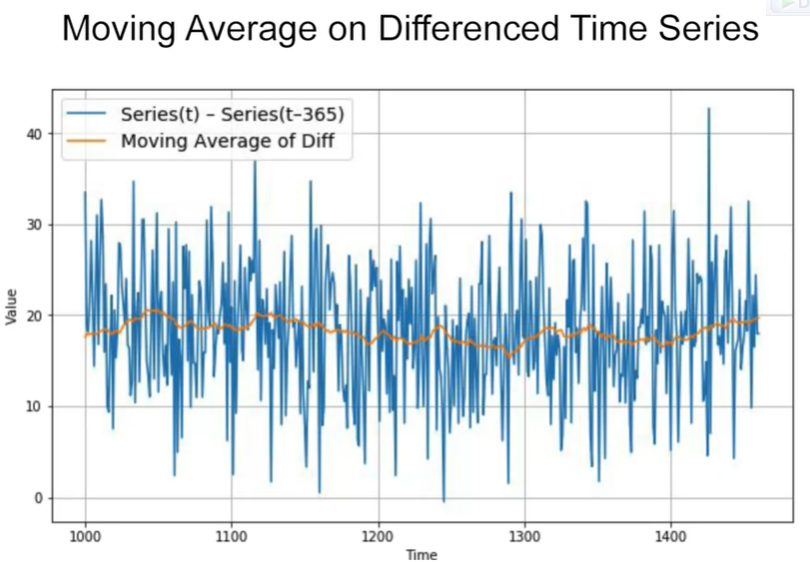
#### Disadvantages :

* It doesn’t anticipate the ‘Trend’ and ‘Seasonality’ in the data and because of that , it can be worse than the Naïve forecasting

### Differencing is the solution to the moving Average Disadvantages :



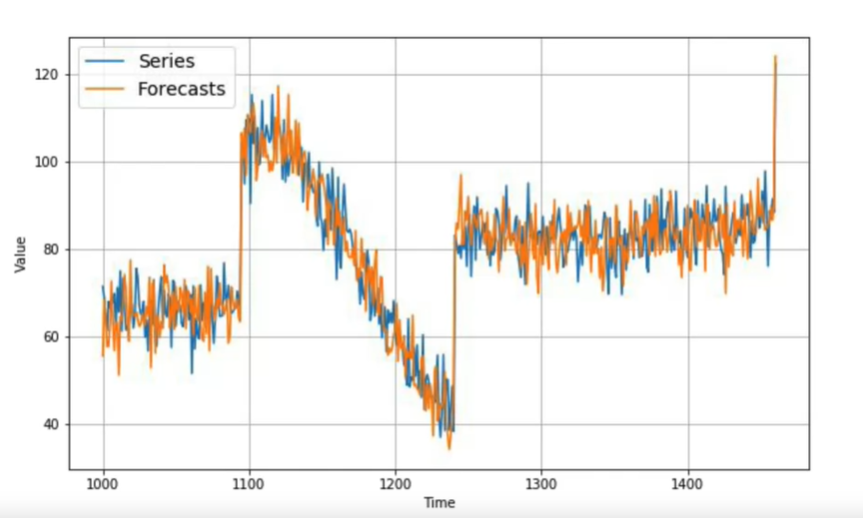
* In order to remove the seasonality and the trend that the Moving average cannot handle: We will analyze the differencing which is calculated by the difference between the actual value and the value of the previous season (in this example the seasonality is 365 days, so in the graph: f(t) = Series(t) – Series (t -365 )

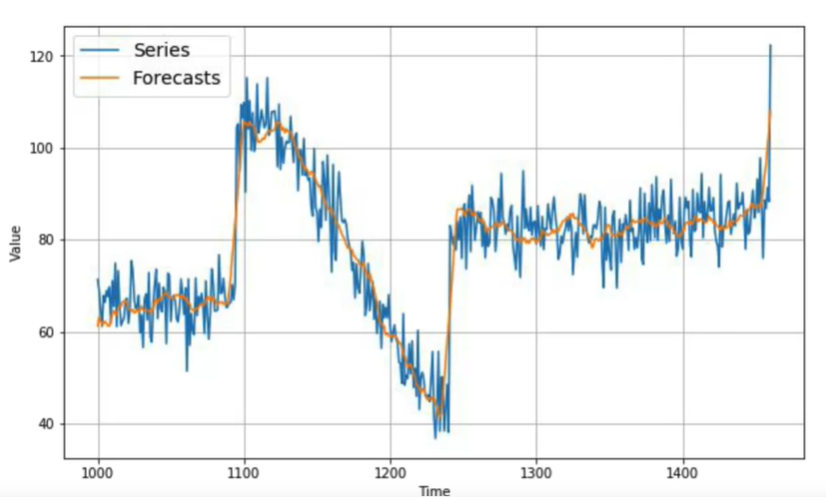


* And then we will apply the moving average technique in the Differenced Time Series to get the mean line
* And finally Here is the formula to get the value to forecast v(t) using v(t-365) and the moving average chart which is intuitive because in Differenced chart we are manipulating the differences instead of real values :



* And finally , that’s we are going to get :



* We notice that we are keeping the trend and the seasonality of the data , But the forecasts are noisy which is expectable since we used the values from the past period which were noisy to calculate the forecasts
* To get rid of the noise , We will apply the moving average technique In the final forecasts :
* The result is a smooth curve , with a nice accuracy , and this chart has the best accuracy compared to the previous techniques
* We can conclude that Simpler approaches can do a perfect work without needing to go to Deep learning architectures