Decomposition in Time Series Data

Abhilasha Chourasia · Follow
Published in Analytics Vidhya · 5 min read · Jan 14, 2020

56

Decomposition in Time Series Data

Photo by Stephen Dawson on Unsplash

Decomposition is a statistical task in which the Time Series data is decomposed into several component or extracting seasonality, trend from a series data. These components are defined as follows:

- Level: The average value in the series.
- Trend: The increasing or decreasing value in the series.
- Seasonality: The repeating short-term cycle in the series.
- Noise: The random variation in the series.

Time series data are combination of these components. All the series have Level and noise. The trend and seasonality components are optional.

In time series data, these components are either additively or multiplicatively combined.

Additive Model:

Additive Model are the one where the variance of data doesn't change over different values of the time series. The systematic component is the arithmetic sum of the individual effects of the predictors.

Additive model is linear and the trend line here is a straight line and seasonality has same frequency and amplitude (height and width of the cycle respectively).

$\mathcal{Y}(t)=S\text{t} + T\text{ t} + \text{Rt}$

Where St = seasonal component,

Tt = trend-cycle component and

Rt = remainder component

Multiplicative Model:

Multiplicative Model are the one where as the data increases, so does the seasonal pattern or the variance increases. Here the trend and seasonal components are multiplied and then added to the error component.

Multiplicative model is non-linear, such as quadratic or exponential and the trend is a curved line and seasonality has an increasing or decreasing frequency and amplitude over time.

$\mathcal{Y}(t)=S\text{t} \times T\text{t} \times \text{Rt}$

Classical Decomposition

Decomposition is used for time series analysis. And the result can be used to inform forecasting model as per the problem. It gives us brief knowledge about the forecasting problem in terms of model complexity and how best to captures these components in the model.

The time series data may have multiplicative or additive components. There can be increase trend followed by decreasing or can be of non-repeating cycle with repeating seasonality components.

Decomposition helps us to better analyze the data and exploring different ways to solve the problem.

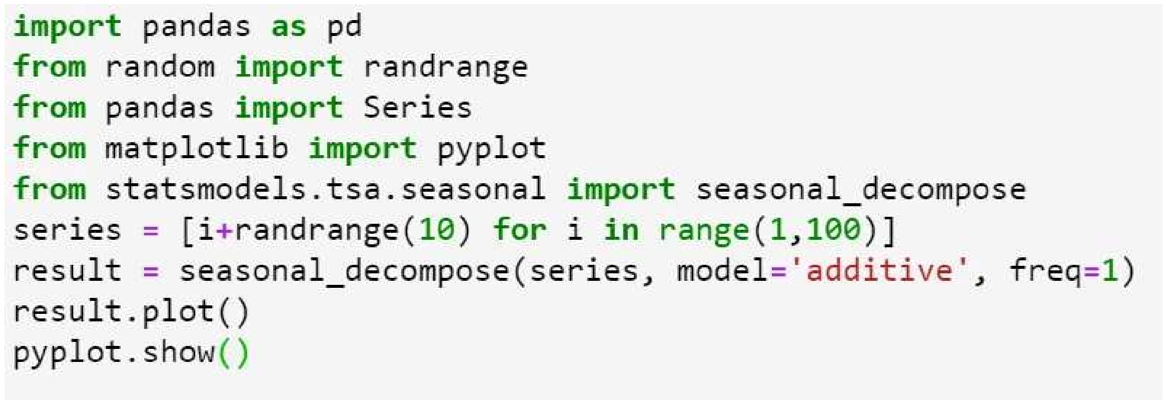
Python provide statsmodels library which helps to decompose a series data into its components. The function used is seasonal_decompose(). This function requires model to be specified as “Additive” or “Multiplicative”.

The output of the function is trend and seasonal series stored in an array. The residuals are the one when trend and seasonal components are removed from the data. Also, the original which is observed data is stored.

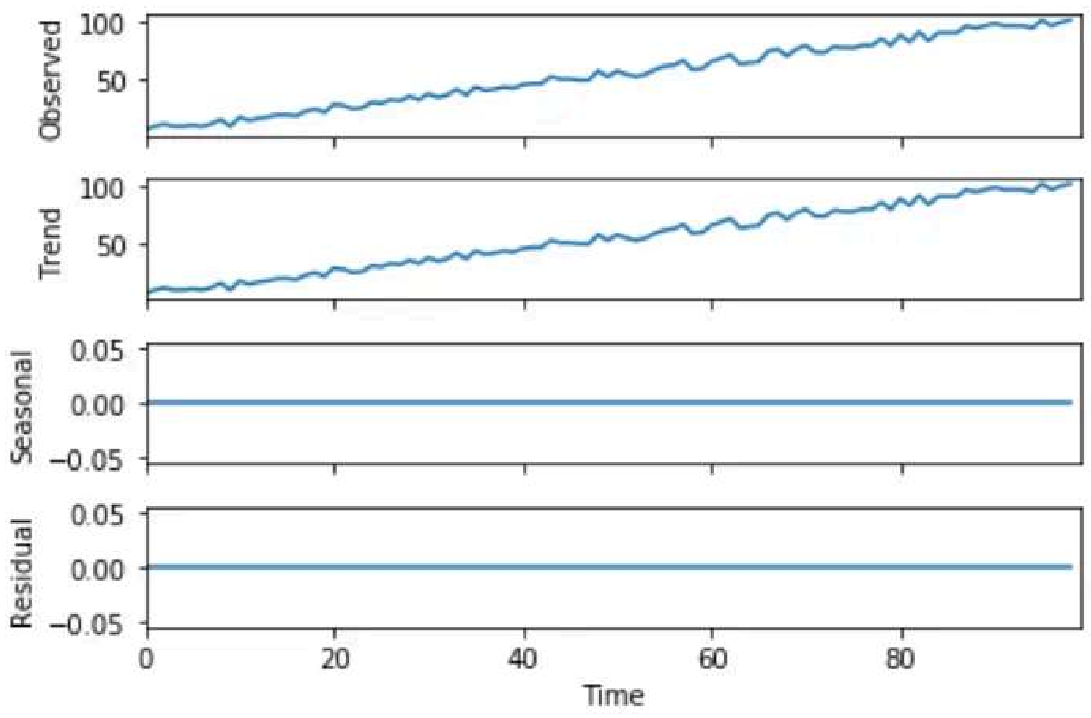
Additive Decomposition

Let’s see how Additive Decomposition works using the function seasonal_decompose().

Let’s generate number between 1 to 100 with some random noise. This series comprised of linearly increasing trend. The result obtained after applying additive decomposition is:



```
import pandas as pd
from random import randrange
from pandas import Series
from matplotlib import pyplot
from statsmodels.tsa.seasonal import seasonal_decompose
series = [i+randrange(10) for i in range(1,100)]
result = seasonal_decompose(series, model='additive', freq=1)
result.plot()
pyplot.show()
```



From the graph plotted we can see that the entire series is taken as series and there's no seasonality.

As we can see that there is no residual as well that means decomposition was not able to separate noise that we added to our series from the linear trend. It uses classical or Naïve method for decomposition.

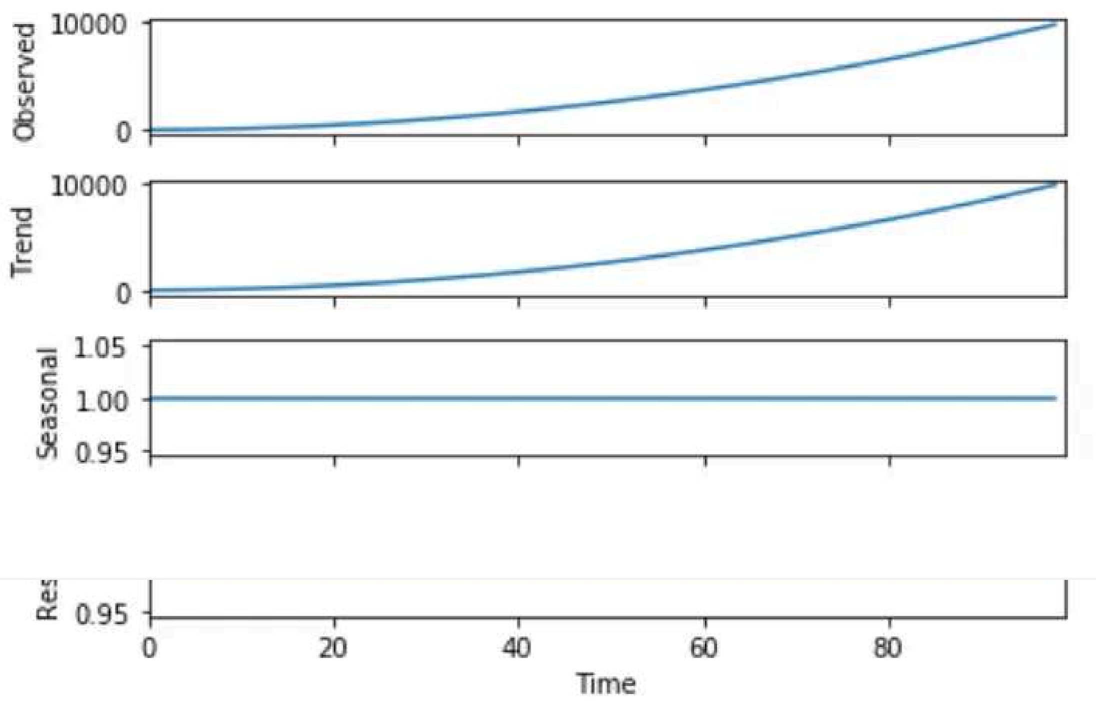
As Naïve Decomposition is a simple method, there are more advanced decompositions available such as Seasonal and Trend decomposition using Loess (STL) decomposition.

Multiplicative Decomposition

Let’s see how Multiplicative Decomposition works using the function seasonal_decompose ().

Let’s generate square of the time between 1 to 100 with some random noise. The result obtained after applying multiplicative decomposition is:

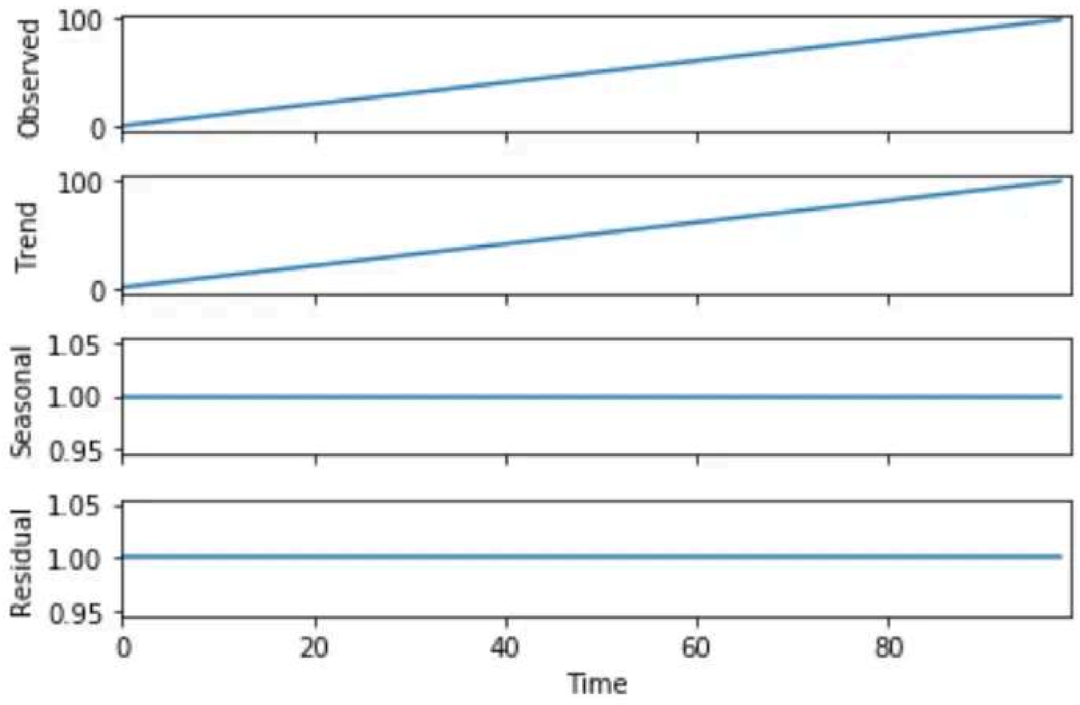
```
from pandas import Series
from matplotlib import pyplot
from statsmodels.tsa.seasonal import seasonal_decompose
series = [i**2.0 for i in range(1,100)]
result = seasonal_decompose(series, model='multiplicative', freq=1)
result.plot()
pyplot.show()
```



We can see from the graph that the trend line is quadratic. Exponential changes can be transformed to linear by data transformation.

Let's see the how the graph changes when the data is transformed:

```
from pandas import Series
from matplotlib import pyplot
import math
from statsmodels.tsa.seasonal import seasonal_decompose
series = [math.sqrt(i**2.0) for i in range(1,100)]
result = seasonal_decompose(series, model='multiplicative',freq=1)
result.plot()
pyplot.show()
```



From the above graph, we can see how the graph changes from quadratic to linear by taking square root of the data.

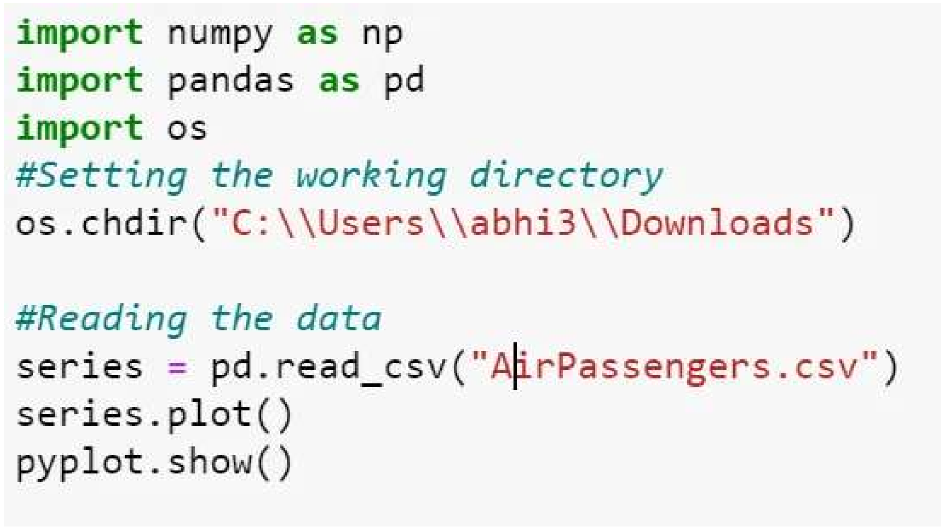
Decomposition on Time Series dataset

Data set Description:

Dataset: Airline Passenger Dataset

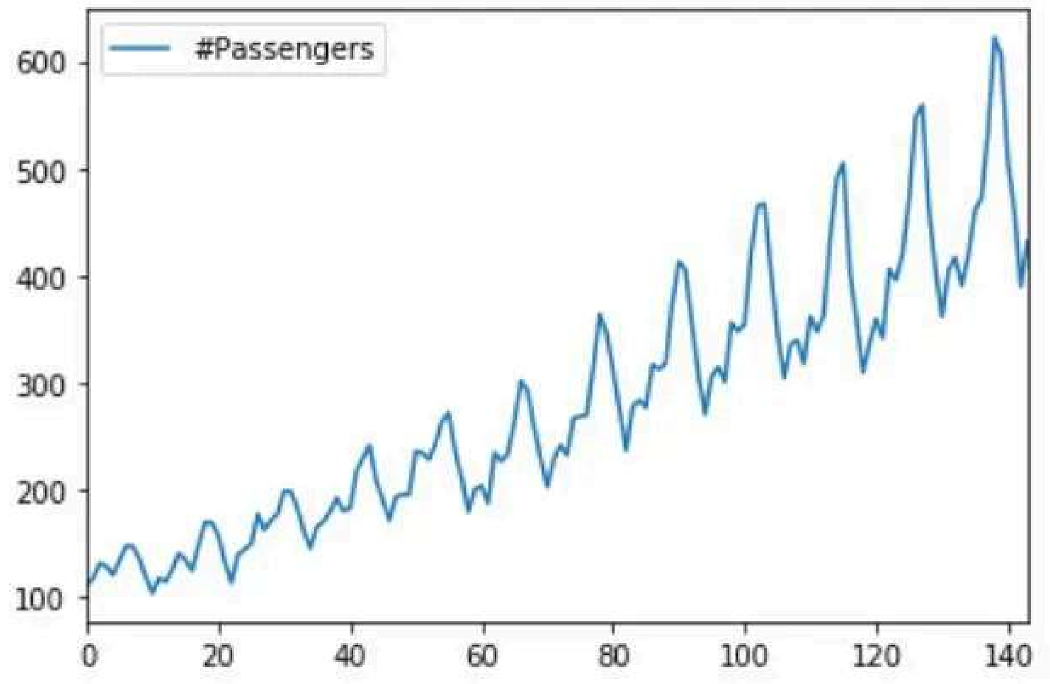
Rows X Columns = 144 X 2

Columns : Month, #Passengers



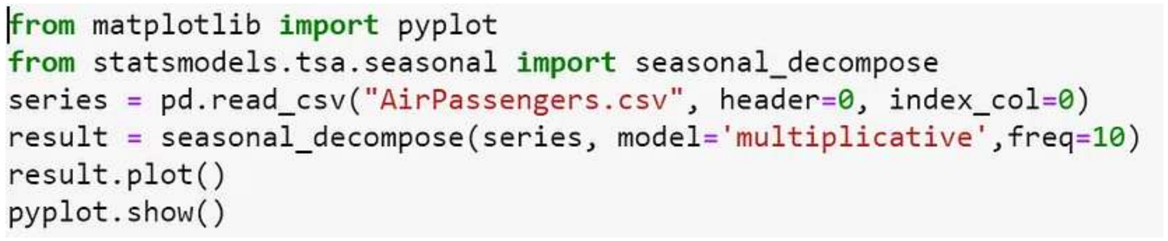
```
import numpy as np
import pandas as pd
import os
#Setting the working directory
os.chdir("C:\\Users\\abhi3\\Downloads")

#Reading the data
series = pd.read_csv("AirPassengers.csv")
series.plot()
pyplot.show()
```

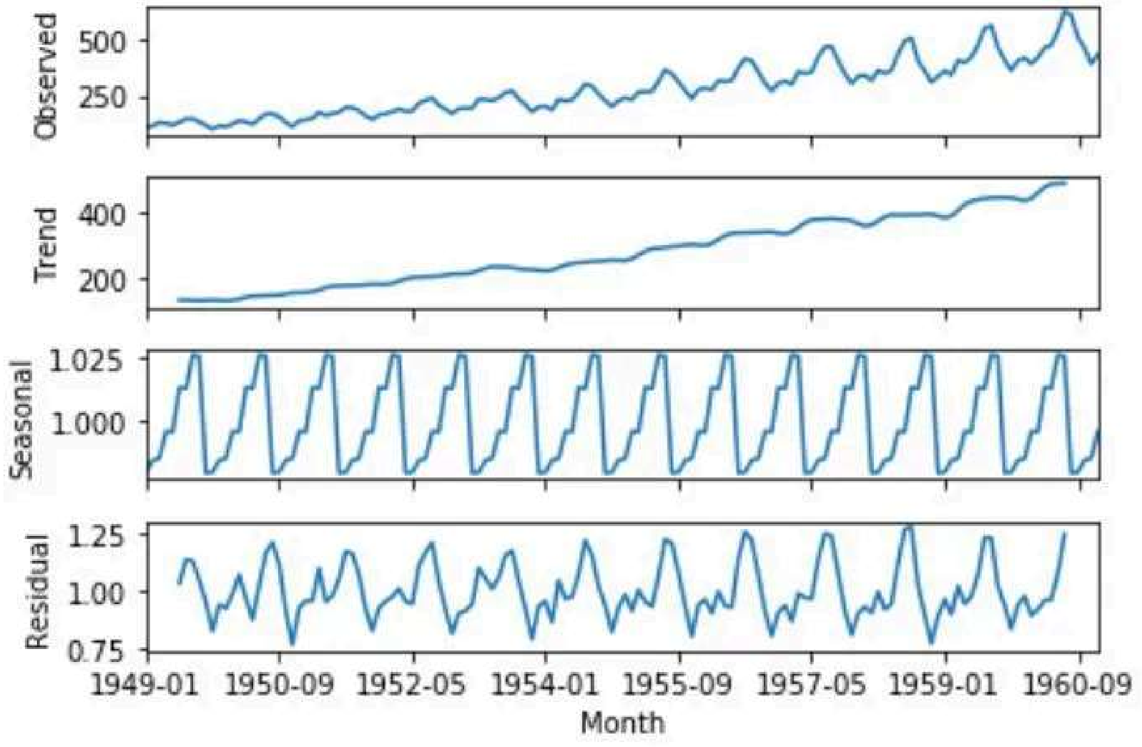


We can see the graph have seasonal and trend as well. Decomposing this gives us clear picture of all the component in the series.

Let's use Multiplicative Decomposition and look at the resultant graphs.



```
from matplotlib import pyplot
from statsmodels.tsa.seasonal import seasonal_decompose
series = pd.read_csv("AirPassengers.csv", header=0, index_col=0)
result = seasonal_decompose(series, model='multiplicative',freq=10)
result.plot()
pyplot.show()
```



We can clearly see from the graph that the trend and seasonality from the series have some pattern even the residual shows high variance in the early and late years.

SEATS Decomposition

Here “SEATS” stand for “Seasonal Extraction in ARIMA Time Series”. Here the series data is either quarterly or monthly.

Advantage of SEATS

- Model-based
- Smooth trend estimate
- Allows estimates at end points
- Allows changing seasonality
- Developed for economic data

STL Decomposition

Here “STL” stands for “Seasonal and Trend decomposition using Loess”. It is more robust and versatile method used for decomposition. Loess is a method for estimating Non-Linear relationship.

Advantage of STL

- Can handle any type of seasonality
- Seasonal component allowed to change over time, and rate of change controlled by user.
- Smoothness of trend-cycle also controlled by user.
- Robust to outliers Not trading day or calendar adjustments.
- Only additive.
- Take logs to get multiplicative decomposition.
- Use Box-Cox transformations to get other decompositions.

Forecasting with Decomposition

We forecast seasonal component and seasonally adjusted component (trend and residual together) separately to forecast a decomposed series data.

It is assumed that seasonal component changes very slowly over time or is unchanging, so the forecast can be done by taking last year data of the estimated component.

In simple words we use Naïve method for seasonal component.

Any non-seasonal forecasting methods like random walk with drift model or Holt's method or non-seasonal ARIMA method can be used for forecasting seasonally adjusted component.

Summary

From the above, we learned:

- Why Decomposition of Time series data is important.
- Various methods of decomposing time series data.
- How can we forecast a time series data with decomposition.

*academic work at Praxis Business School,Bangalore.

Machine Learning

Published in Analytics Vidhya

70K Followers · Last published Oct 15, 2024

Follow

Analytics Vidhya is a community of Generative AI and Data Science professionals. We are building the next-gen data science ecosystem https://www.analyticsvidhya.com

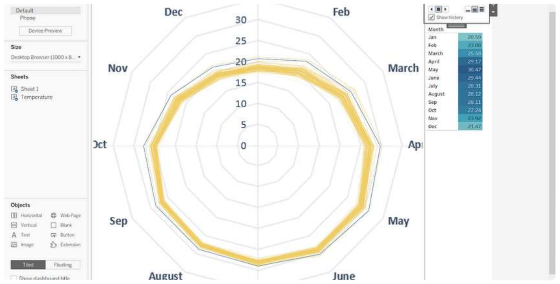
Written by Abhilasha Chourasia

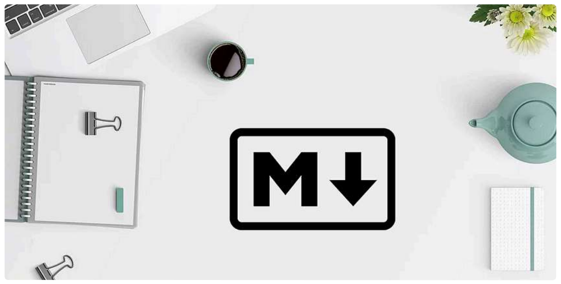
9 Followers · 4 Following

Follow

Student at Praxis Business School(Data Science), Bangalore, India

More from Abhilasha Chourasia and Analytics Vidhya





Abhilasha Chourasia

An Analysis of the All India Temperature, 1901–2017

The modern world has progressed so far and so fast. Us humans and our curiosity have...

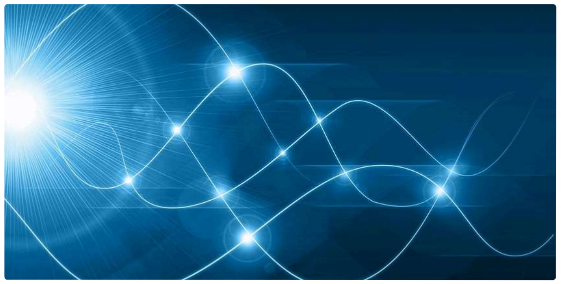
Oct 26, 2019

In Analytics Vidhya by Hannan Satopay

The Ultimate Markdown Guide (for Jupyter Notebook)

An in-depth guide for Markdown syntax usage for Jupyter Notebook

Nov 18, 2019 2.4K 13



In Analytics Vidhya by Leland Roberts

Understanding the Mel Spectrogram

(and Other Topics in Signal Processing)

Mar 6, 2020 2.2K 27

In Analytics Vidhya by Nadeem

Encoders-Decoders, Sequence to Sequence Architecture.

Understanding Encoders-Decoders, Sequence to Sequence Architecture in Deep...

Mar 11, 2021 138 3

See all from Abhilasha Chourasia

See all from Analytics Vidhya

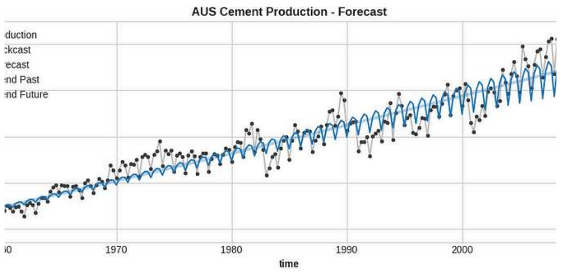
Recommended from Medium

Irina (Xinli) Yu, Ph.D.

Mastering Time Series Forecasting with ARIMA Models

Time series forecasting is a critical component in many domains, including...

Jun 7 2



In Towards Data Science by Florin Andrei

Modeling variable seasonal features with the Fourier transform

Improve time series forecast performance with a technique from signal processing

Oct 12, 2023 325 1

Lists

Predictive Modeling w/ Python

20 stories · 1700 saves

Practical Guides to Machine Learning

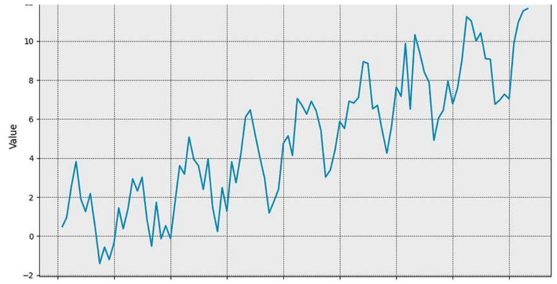
10 stories · 2068 saves

Natural Language Processing

1842 stories · 1466 saves

The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 518 saves

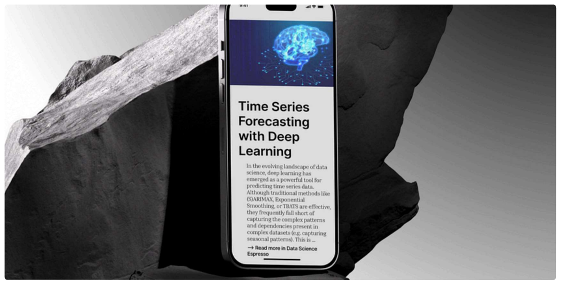


In Stackademic by Ganesh Bajaj

Time Series Analysis: Interpretation of ACF and PACF…

Autocorrelation (ACF) and Partial Autocorrelation (PACF) plots are powerful…

Sep 24 50



In Python in Plain English by Sarah Lea

Beginner’s Guide to Advanced Techniques for Time Series…

In the evolving landscape of data science, deep learning has emerged as a powerful to…

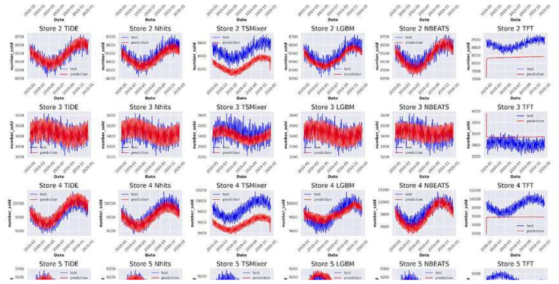
Jul 22 44

Dhiraj K

Embeddings: Applying Them to Tabular and Time Series Models

Imagine a retail store that wants to predict how much a customer is likely to spend next...

4d ago 50



Esther Cifuentes

Comparing Time Series Algorithms

Evaluating Leading Time Series Algorithm with Darts.

Oct 16 367 7

See more recommendations