TDS Archive · Follow publication

Time Series Data Analysis with sARIMA and Dash

Introducing a Dash web app that guides the analysis of time series datasets, using sARIMA models | Live app | Git Hub

10 min read · May 6, 2023

Gabriele Albini Follow

Introduction

When working with time-series datasets, statistics models such as SARIMA can be a powerful tool to understand the data components: trend, seasonality, and dependency over time.

This article will briefly introduce sARIMA models and then present a web application that guides the user through the steps needed to analyze the data and fit the optimal model to perform predictions. This application has been built using *Plotly Dash* and Python:

- Link to the live app

. . .

1. Theoretical introduction on Sarima Models

1.1 The building blocks of the model

To understand what sARIMA models are, let's first introduce the building blocks of these models.

sARIMA is a composition of different sub-models (i.e. polynomials that we use to represent our time series data) which form the acronym: seasonal (s) autoregressive (AR) integrated (I) moving average (MA):

- **AR**: the autoregressive component, governed by the hyperparameter "p", assumes that the current value at a time "t" can be expressed as a linear combination of the previous "p" values:

$$\textbf{AR of order p:} \quad Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p}$$

AR | Image by author

- **I**: the integrated component is represented by the hyperparameter "d", which is the degree of the differencing transformation applied to the data. *Differencing* is a technique used to remove trend from the data (i.e. make the data stationary with respect to the mean, as we'll see later), which helps the model fit the data as it isolates the trend component (we use d=1 for linear trend, d=2 for quadratic trend, …). Differencing the data with d=1 means working with the difference between consecutive data points:

I differencing of order 1: $Y'_t = Y_t - Y_{t-1}$

I | Image by author

- **MA**: the moving average component, governed by the hyperparameter "q", assumes that the current value at a time "t" can be expressed as a constant term (usually the mean) plus a linear combination of the errors of the previous "q" points:

MA of order q: $Y_t = \mu + \varepsilon_t + \alpha_1 \varepsilon_{t-1} + \alpha_2 \varepsilon_{t-2} + \cdots + \alpha_q \varepsilon_{t-q}$

MA | Image by author

- If we consider the components so far, we get "ARIMA", which is the name of a model family to work with time series data with no seasonality. sARIMA models are a generalization to work with seasonal data with the addition of an **S**-component: the seasonal component, which consists of a new set of AR, I, MA components with a seasonal lag. In other words, once identified a seasonality and defined its lag (represented by the hyperparameter "m" — e.g. m=12 means that every year, on a monthly dataset, we see the same behavior), we create a new set of AR (P), I (D), MA (Q) components, with respect to the seasonal lag (m) (e.g. if D=1 and m=12, this means that we apply a 1-degree differencing to the series, with a lag of 12).

To sum up, the sARIMA model is defined by 7 hyperparameters: 3 for the non-seasonal part of the model, and 4 for the seasonal part. They are indicated as:

> *sARIMA (p,d,q) (P,D,Q)m*

Thanks to the model flexibility, we can "switch off" the components that are not embodied in our data (i.e. if the data doesn't have a trend or doesn't have seasonality, the respective parameters can be set to 0) and still use the same model framework to fit the data.

On the other hand, among sARIMA limitations, we have that these models can capture only 1 seasonality. If a daily dataset has a yearly plus a weekly seasonality, we'll need to choose the strongest one.

1.2 How to choose the model hyperparameters: ACF and PACF

To identify the model hyperparameters, we normally look at the ***autocorrelation*** and ***partial-autocorrelation*** of the time series; since all the above components use past data to model present and future points, we should investigate how past and present data are correlated and define how many past data points we need, to model the present.

For this reason, autocorrelation and partial-autocorrelation are two widely used functions:

- **ACF** (autocorrelation): describes the correlation of the time series, with its lags. All data points are compared to their previous lag 1, lag 2, lag 3, … The resulting correlation is plotted on a histogram. This chart (also called "correlogram") is used to visualize how much information is retained throughout the time series. The ACF helps us in choosing the sARIMA model because:

> *The ACF helps to identify the MA(q) hyperparameter.*

- **PACF** (partial autocorrelation): describes the partial correlation of the time series, with its lags. Differently from the ACF, the PACF shows the correlation between a point X_t and a lag, which is not explained by common correlations with other lags at a lower order. In other words, the PACF isolates the direct correlation between two terms. The PACF helps us in choosing the sARIMA model because:

> *The PACF helps to identify the AR(p) hyperparameter.*

Before using these tools, however, we need to mention that ACF and PACF can only be used on a "**stationary**" time series.

1.3 Stationarity

A (weakly) stationary time series is a time series where:

- The **mean is constant** over time (i.e. the series fluctuates around a horizontal line without positive or negative trends)
- The **variance is constant** over time (i.e. there is no seasonality or change in the deviation from the mean)

Of course not all time series are natively stationary; however, we can transform them to make them stationary. The **most common transformations** used to make a time series stationary are:

- The **natural log:** by applying the log to each data point, we usually manage to make the time series stationary with respect to the *variance*.
- **Differencing:** by differencing a time series, we usually manage to remove the trend and make the time series stationary with respect to the *mean*.

After transforming the time series, we can use two tools to confirm that it is stationary:

- The **Box-Cox** plot: this is a plot of the rolling mean (on the x-axis) vs the rolling standard deviation (on the y-axis) (or the mean vs variance of grouped points). Our data is stationary if we don't observe any particular trends in the chart and we see little variation on both axes.
- The **Augmented Dickey–Fuller** test (ADF): a statistical test in which we try to reject the null hypothesis stating that the time series is non-stationary.

Once a time series is stationary, we can analyze the ACF and PACF patterns, and find the SARIMA model hyperparameters.

. . .

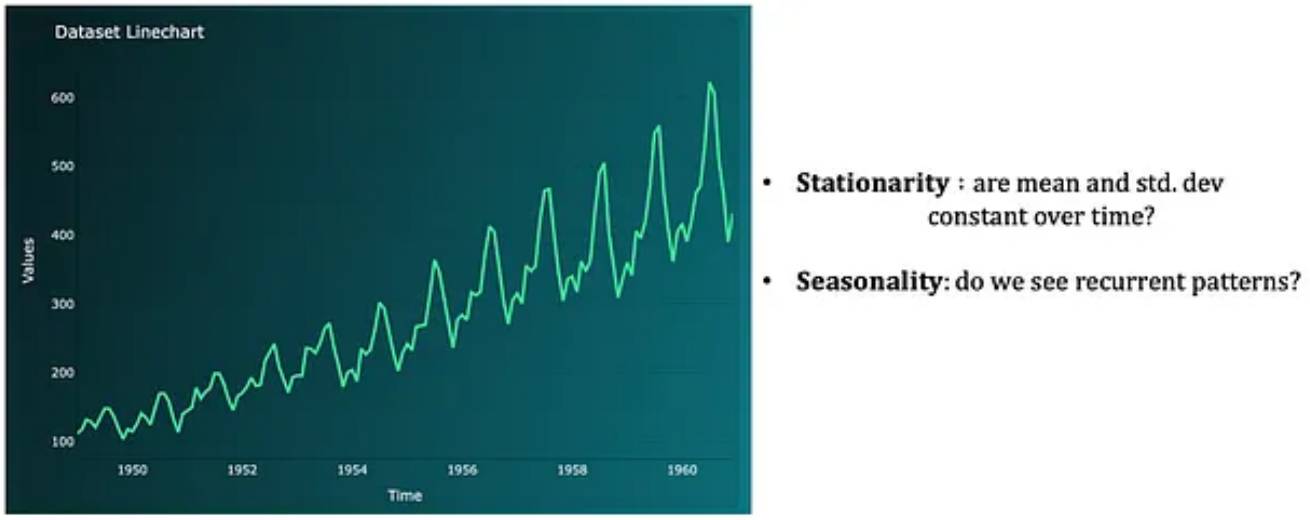
2. A practical template to work with Sarima models

Identifying the sARIMA model that fits our data consist of a series of steps, which we will perform on the AirPassenger dataset (available here).

Each step roughly corresponds to a "page" of the Dash web app.

2.1 Plot your data

Create a line chart of your raw data: some of the features described above can be seen by the naked eye, especially stationarity, and seasonality.



Raw line chart | Image by author

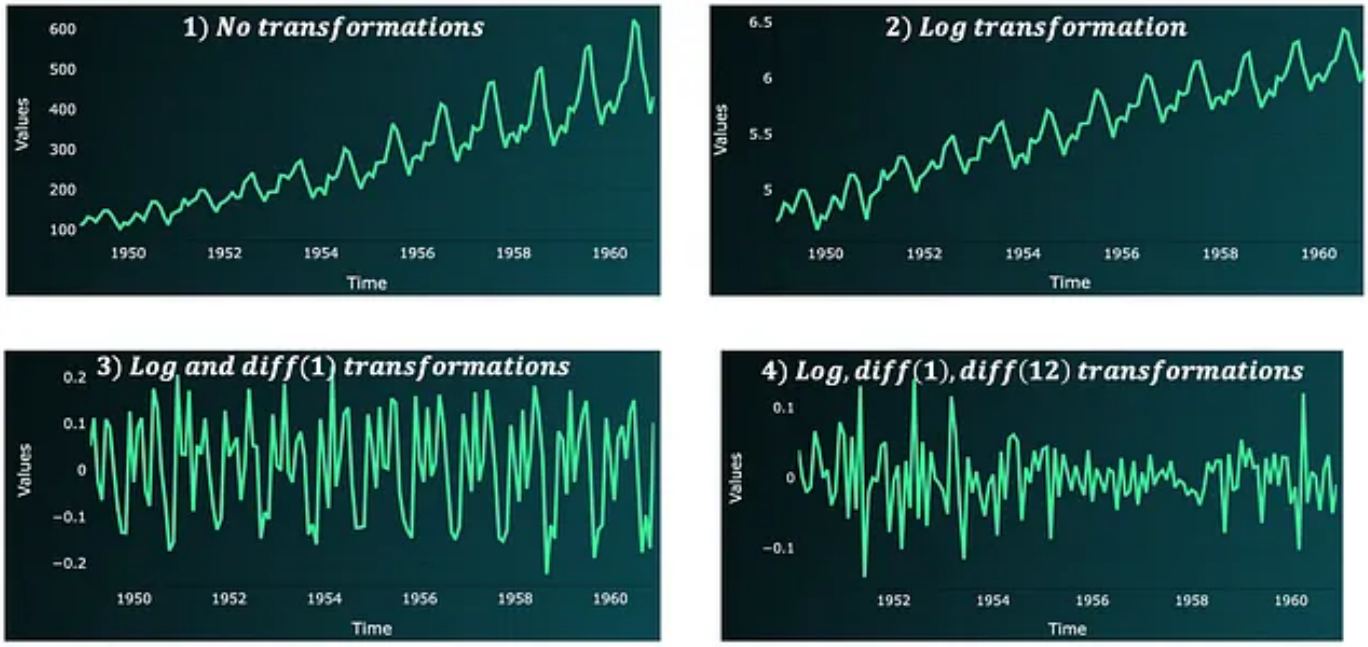
In the above chart, we see a positive linear trend and a recurrent seasonality pattern; considering that we have monthly data, we can assume the seasonality to be yearly (lag 12). The data is not stationary.

2.2 Transform the data to make it stationary

In order to find the model hyperparameters, we need to work with a stationary time series. So, if the data is not stationary, we'll need to transform it:

- Start with the *log transformation*, to make the data stationary with respect to the variance (the log is defined over positive values. So, if the data presents negative or 0 values, add a constant to each datapoint).
- Apply *differencing* to make the data stationary with respect to the mean. Usually start with differencing of order 1 and lag 1. Then, if data is still not stationary, try differencing with respect to the seasonal lag (e.g. 12 if we have monthly data). (Using a reverse order won't make a difference).

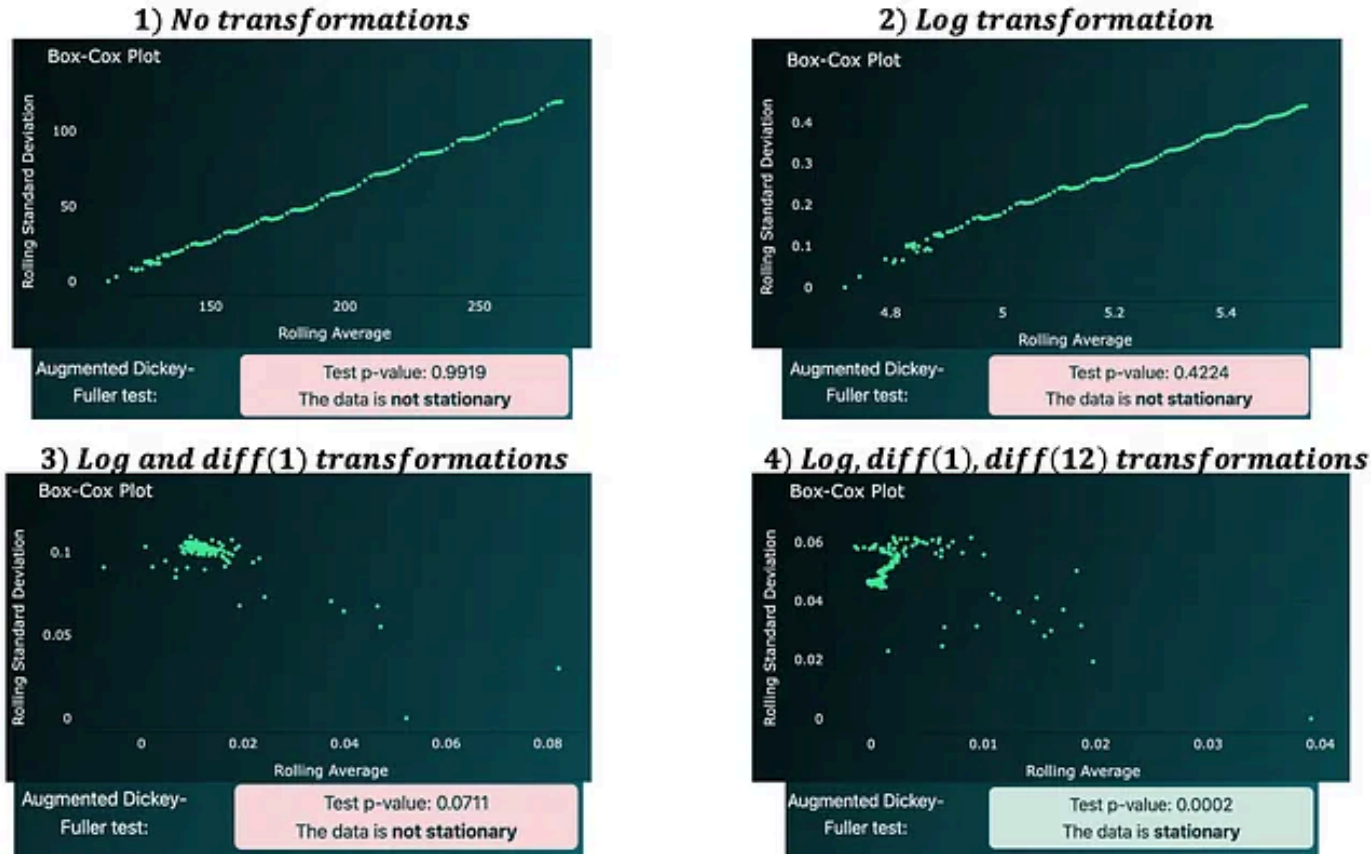
With our dataset, we need to perform the following steps to make it fully stationary:



Stationary transformations | Image by author

After each step, by looking at the ADF test p-value and Box-Cox plot, we see that:

- The Box-Cox plot gets progressively cleaned from any trend and all points get closer and closer.
- The p-value progressively drops. We can finally reject the null hypothesis of the test.



Stationary transformations (2) | Image by author

2.3 Identify suitable model hyperparameters with the ACF and PACF

While transforming the data to stationary, we have already identified 3 parameters:

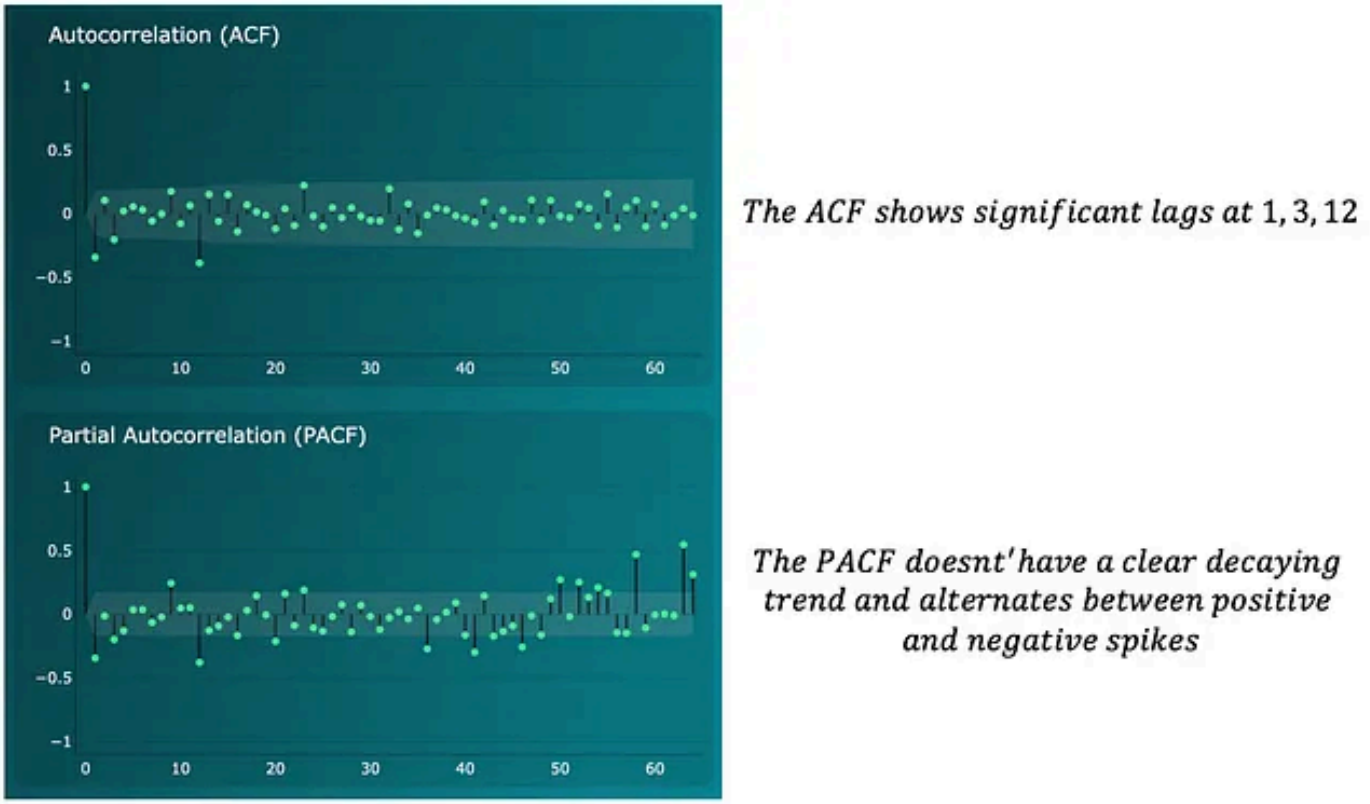
- Since we applied differencing, the model will include differencing components. We applied a differencing of 1 and 12: we can set d=1 and D=1 with m=12 (seasonality of 12).

For the remaining parameters, we can look at the ACF and PACF after the transformations.

In general, we can apply the following *rules*:

- We have an **AR(p) process if:** the PACF has a significant spike at a certain lag "p" (and no significant spikes after) and the ACF decays or shows a sinusoidal behavior (alternating positive, negative spikes).
- We have a **MA(q) process if:** the ACF has a significant spike at a certain lag "q" (and no significant spikes after) and the PACF decays or shows a sinusoidal behavior (alternating positive, negative spikes).
- In the case of **seasonal AR(P) or MA(Q) processes**, we will see that the significant spikes repeat at the seasonal lags.

By looking at our example, we see the following:



ACF and PACF after transformations | Image by author

- The closest rule to the above behavior, suggests some MA(q) process with "q" between 1 and 3; the fact that we still have a significant spike at 12, may also suggest an MA(Q) with Q=1 (since m=12).

We use the ACF and PACF to get a range of hyperparameter values that will form model candidates. We can compare these different model candidates against our data, and pick the top-performing one.

In the example, our model candidates seem to be:

- SARIMA (p,d,q) (P,D,Q)m = (0, 1, 1) (0, 1, 1) 12
- SARIMA (p,d,q) (P,D,Q)m = (0, 1, 3) (0, 1, 1) 12

2.4 Perform a model grid search to identify optimal hyperparameters

Grid search can be used to compare several model candidates against each other: we fit each model to the data and pick the top-performing one.

To set up a grid search we need to:

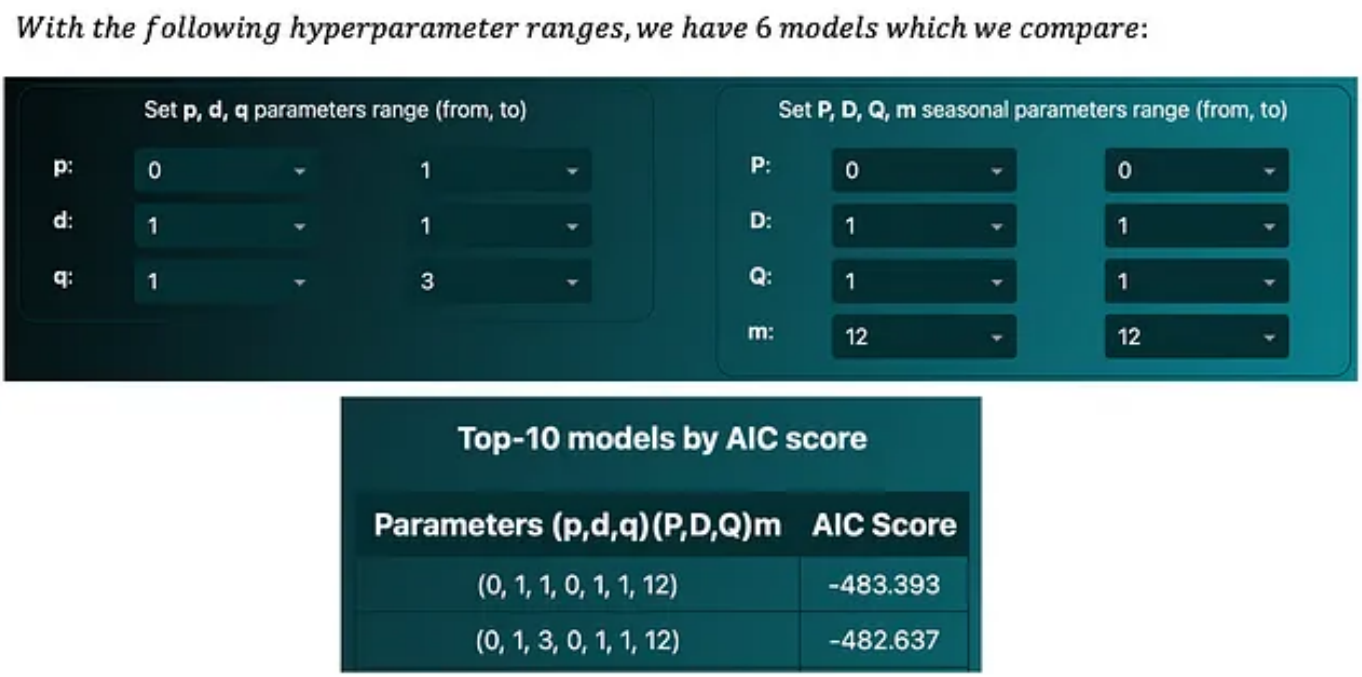
- create a list with all possible combinations of model hyperparameters, given a range of values for each hyperparameter.
- fit each model and measure its performance using a KPI of choice.
- select the hyperparameters looking at the top-performing models.

In our case, we will compare model performances using the **AIC (Akaike information criterion) score**. This KPI formula consists of a trade-off between the fitting error (accuracy) and model complexity. In general, when the complexity is too low, the error is high, because we over-simplify the model fitting task; on the contrary, when complexity is too high, the error is still high due to overfitting. A trade-off between these two will allow us to identify the "top-performing" model.

Practical note: with fitting a sARIMA model, we will need to use the original dataset with the log transformation (if we've applied it), *but we don't want to use the data with differencing transformations.*

We can choose to reserve part of the time series (usually the most recent 20% observations) as a test set.

In our example, based on the below hyperparameter ranges, the best model is:

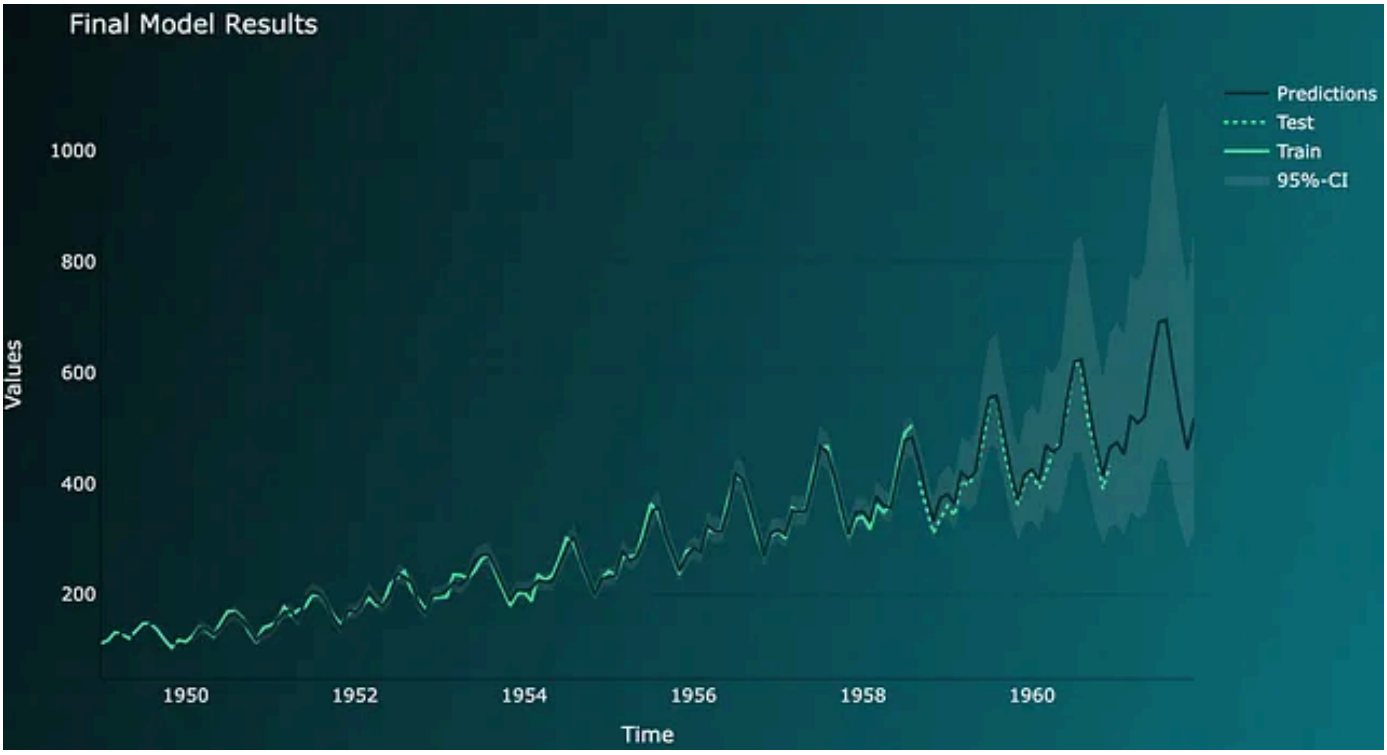


Model grid search | Image by author

> *SARIMA (p,d,q) (P,D,Q)m = (0, 1, 1) (0, 1, 1) 12*

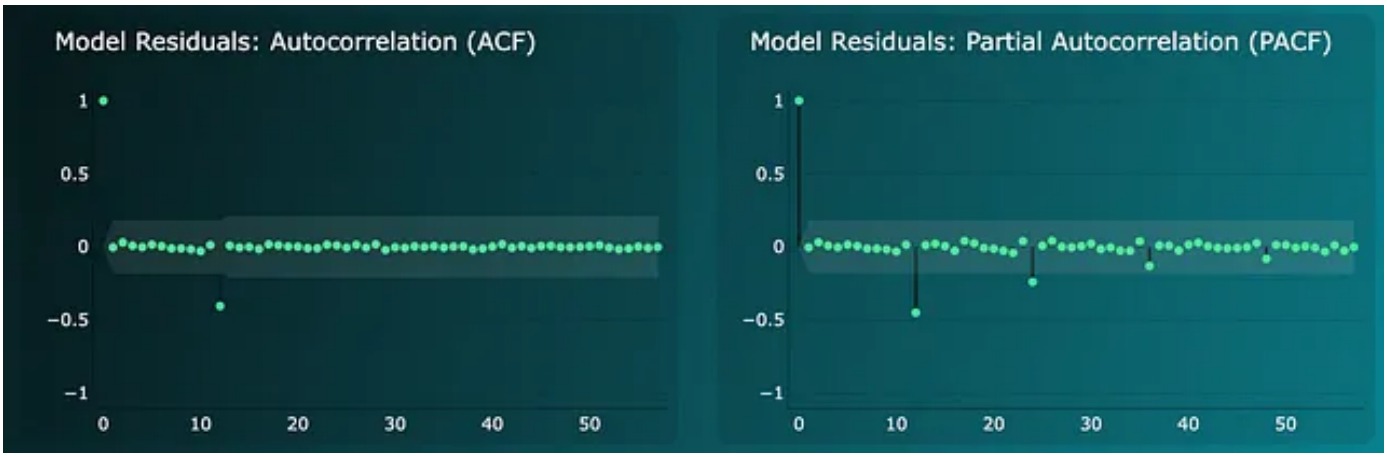
2.5 Final model: fit and predictions

We can finally predict data for train, test, and any future out-of-sample observation. The final plot is:



Final model | Image by author

To confirm that we captured all correlations, we can plot the model residuals ACF and PACF:



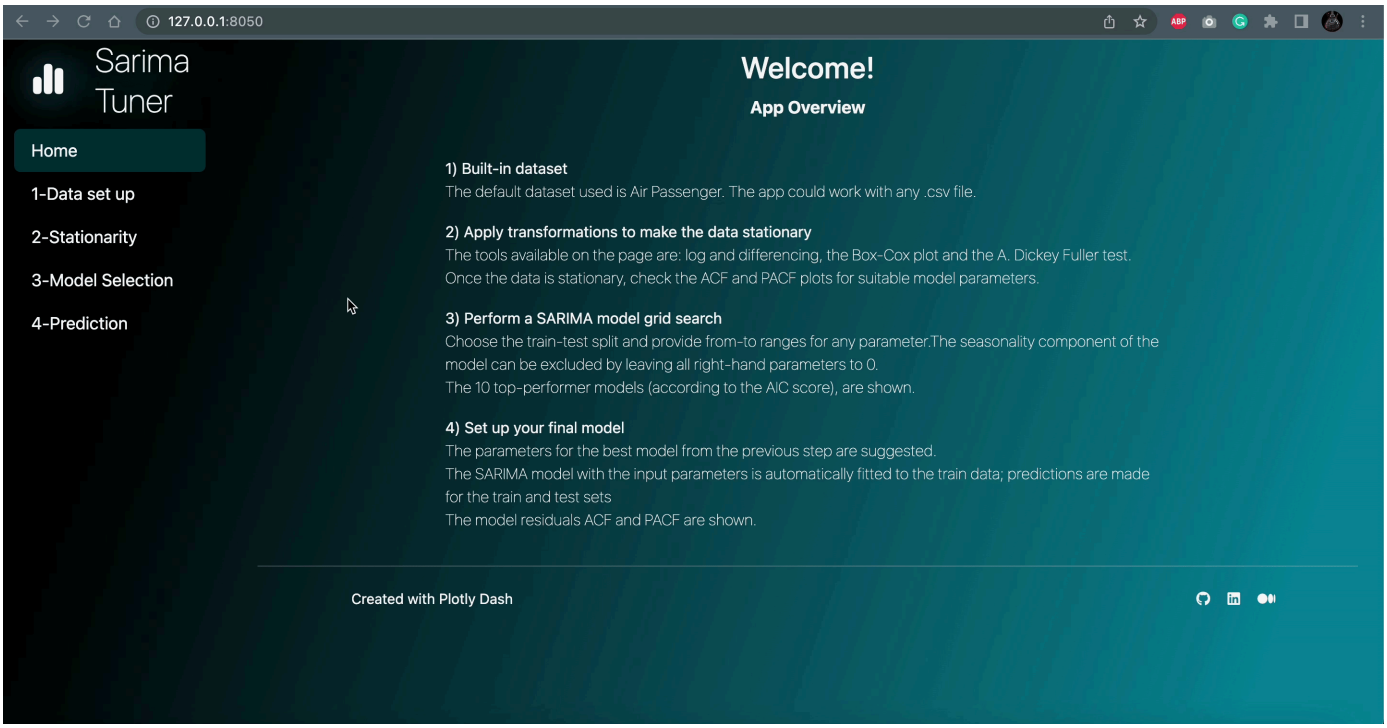
In this case, some signal from the strong seasonality component is still present, but most of the remaining lags have a 0 correlation.

. . .

Conclusion

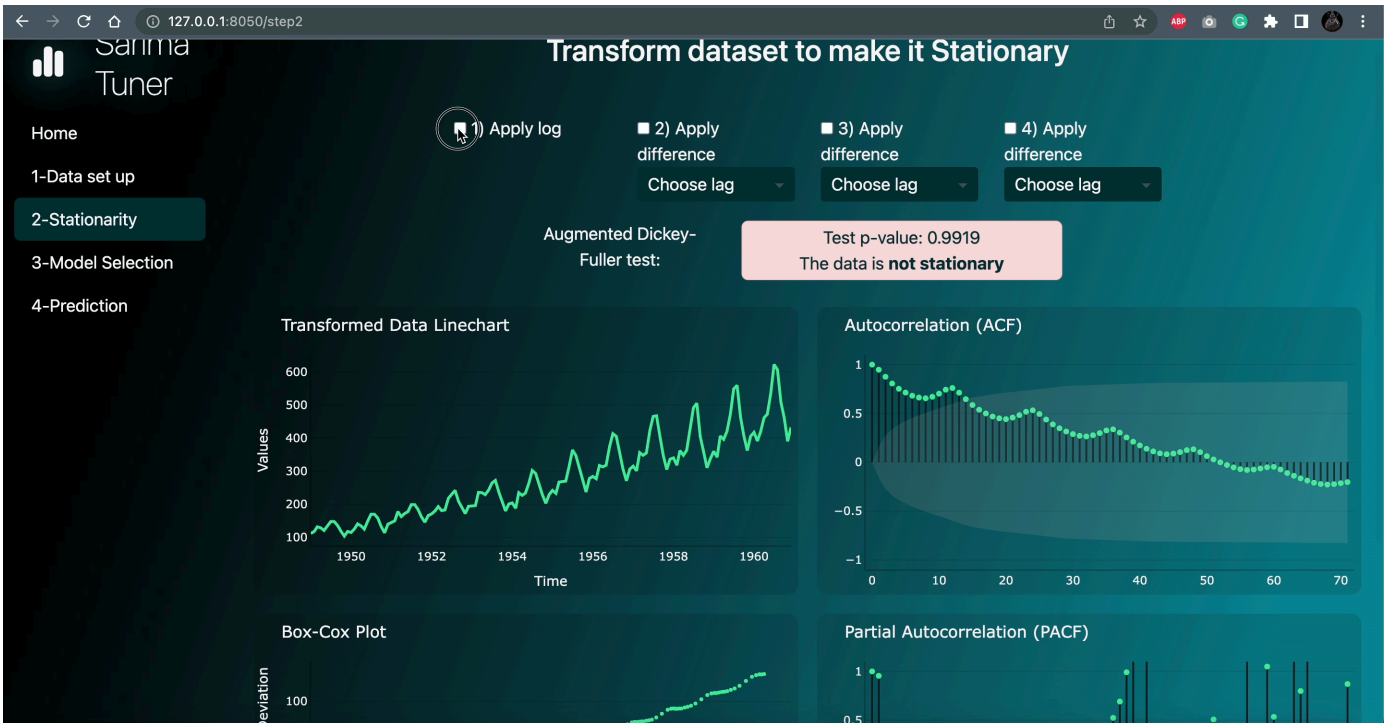
The steps described above should work on any dataset which could be modeled through sARIMA. To recap :

1-Plot & explore your data



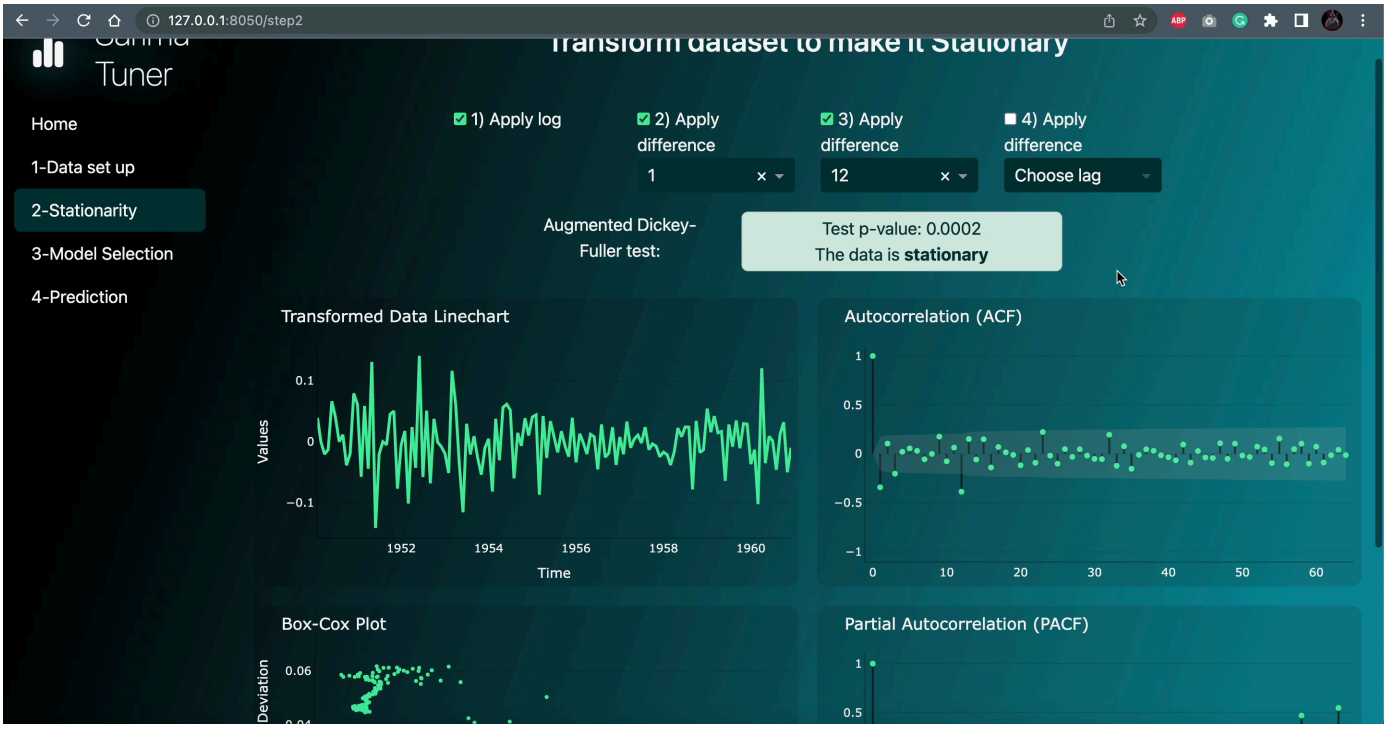
Dash live app | Image by author

2-Apply transformations to make the data stationary (focus on the left-end charts and the ADF test)



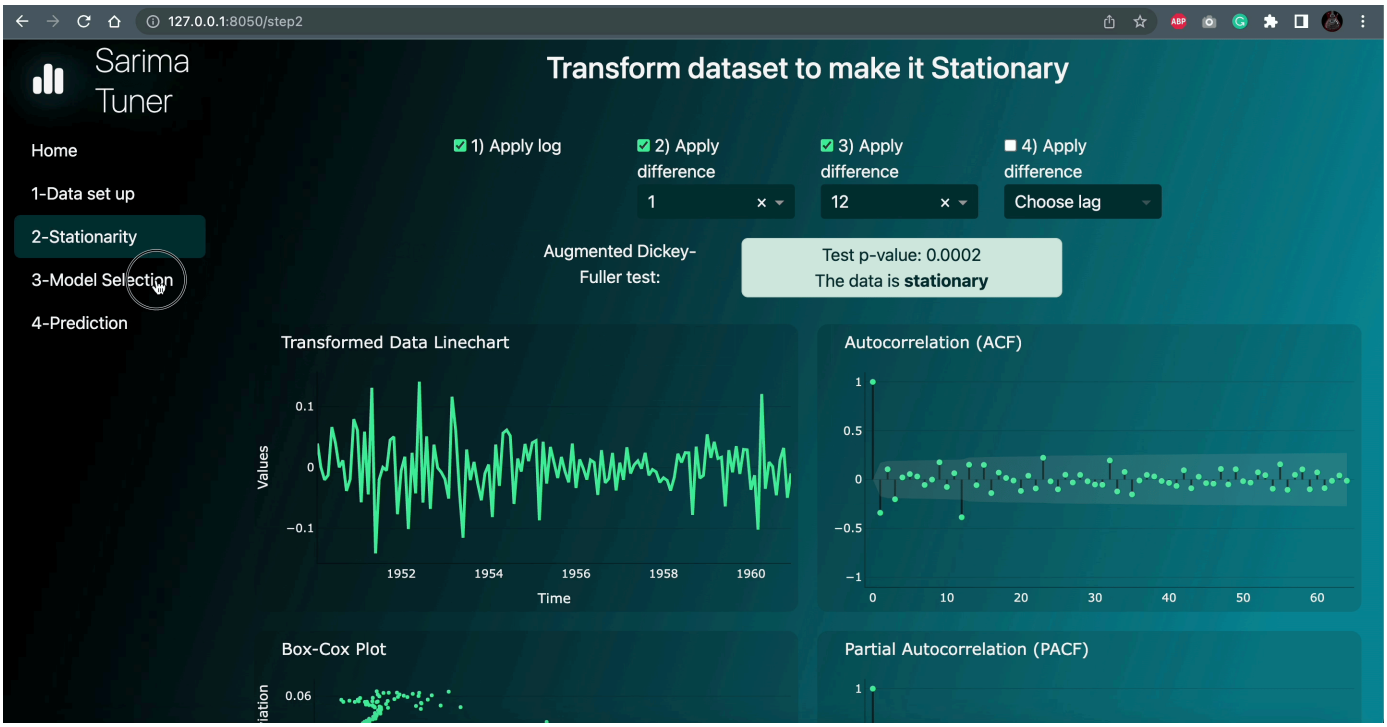
Dash live app | Image by author

3-Identify suitable hyperparameters by looking at the ACF and PACF (right-end charts)



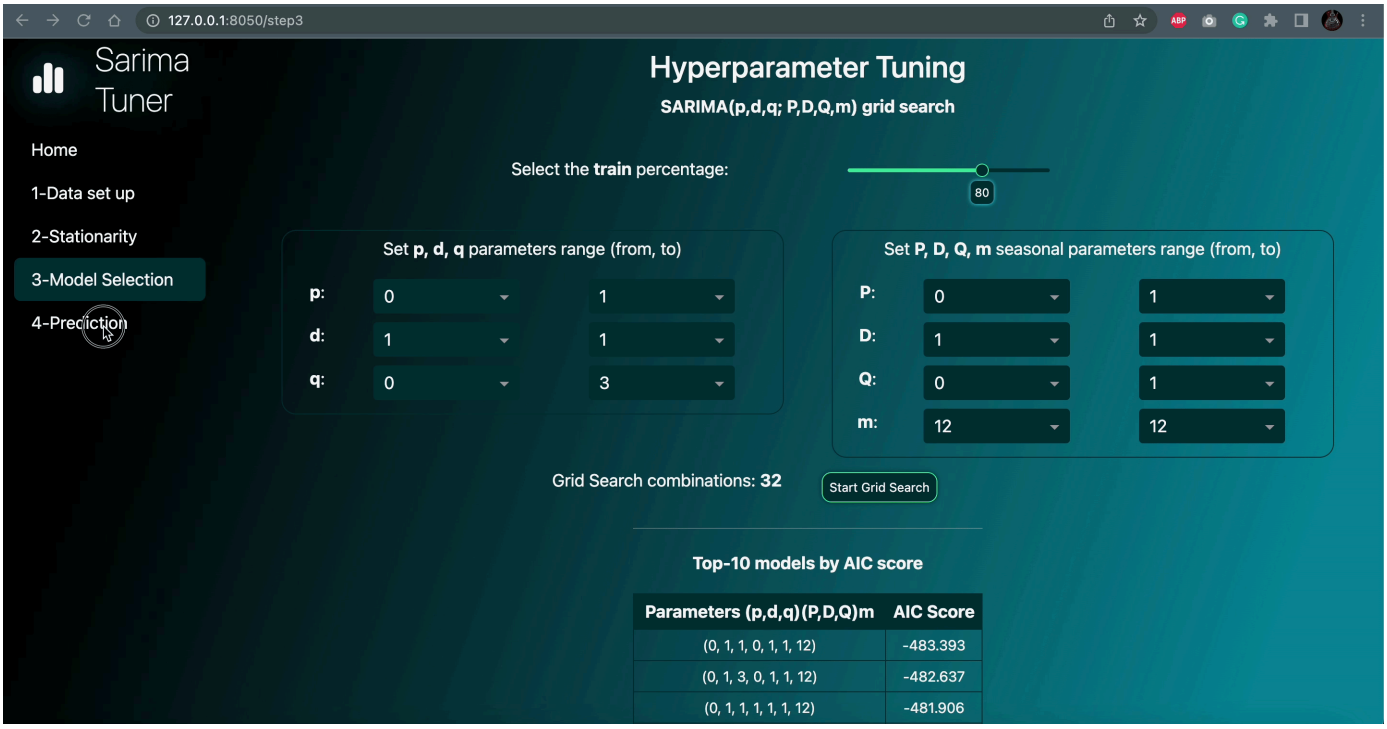
Dash live app | Image by author

4-Perform a grid search to select optimal hyperparameters



Dash live app | Image by author

5-Fit and predict using the best model



Dash live app | Image by author

Download the app locally, upload your own datasets (by replacing the .csv file in the data folder) and try to fit the best model.

Thank you for reading!

Dash Python Sarima Time Series Analysis Plotly

Data Science

Follow

Published in TDS Archive

827K followers · Last published Feb 3, 2025

An archive of data science, data analytics, data engineering, machine learning, and artificial intelligence writing from the former Towards Data Science Medium publication.

Follow

Written by Gabriele Albini

182 followers · 37 following

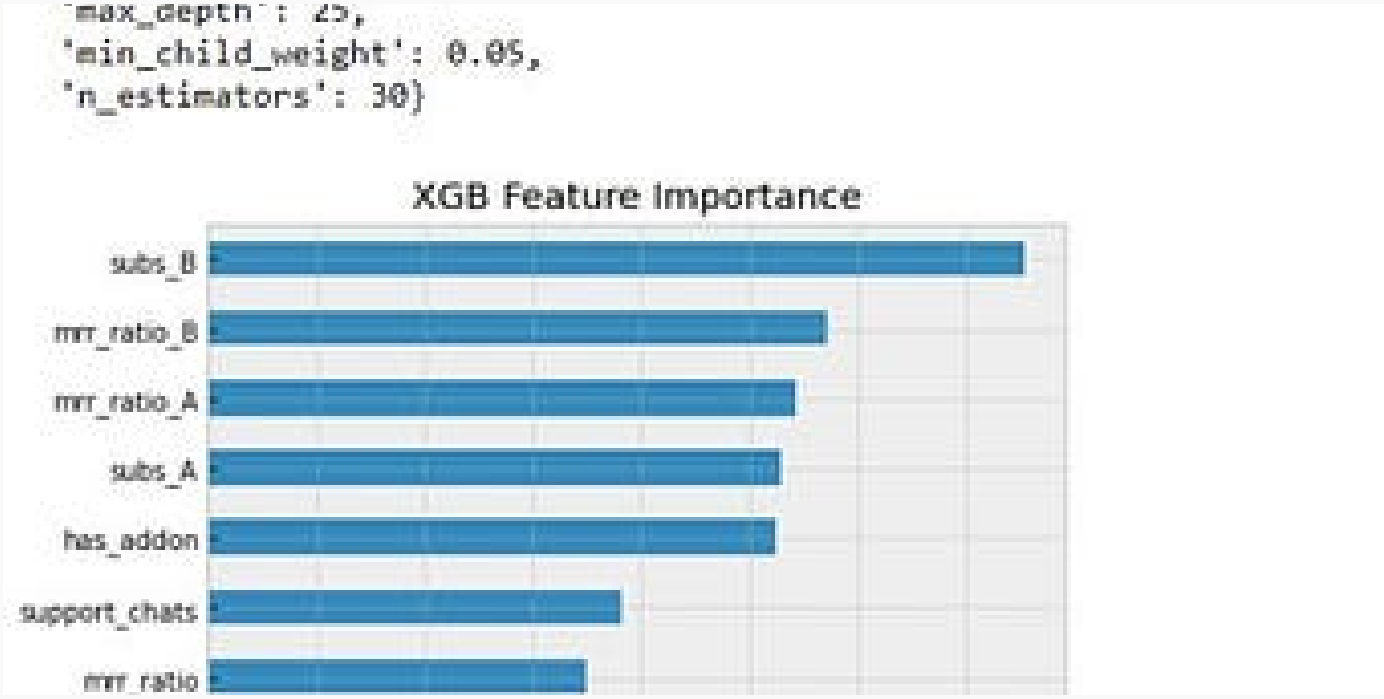
Constant Learner, passionate about data analytics, ML and data visualization. Interested in work, tech, music & guitar

No responses yet

Write a response

What are your thoughts?

More from Gabriele Albini and TDS Archive



In TDS Archive by Gabriele Albini

A Framework for Analyzing Churn

A step-by-step guide to performing a customer churn analysis, using a simulated dataset

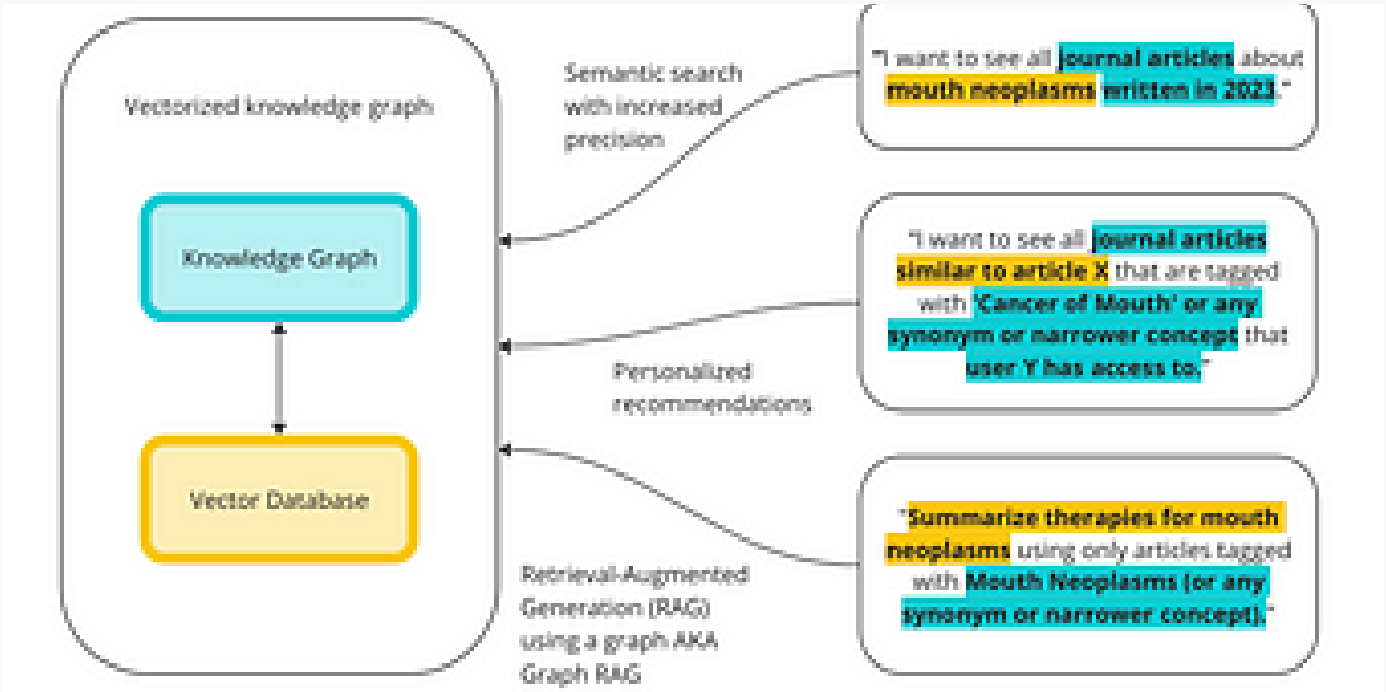
Jan 13, 2023 172 1

In TDS Archive by Thuwarakesh Murallie

How to Build a Knowledge Graph in Minutes (And Make It Enterprise-Ready)

I tried and failed creating one—but it was when LLMs were not a thing!

Jan 13 1.2K 9

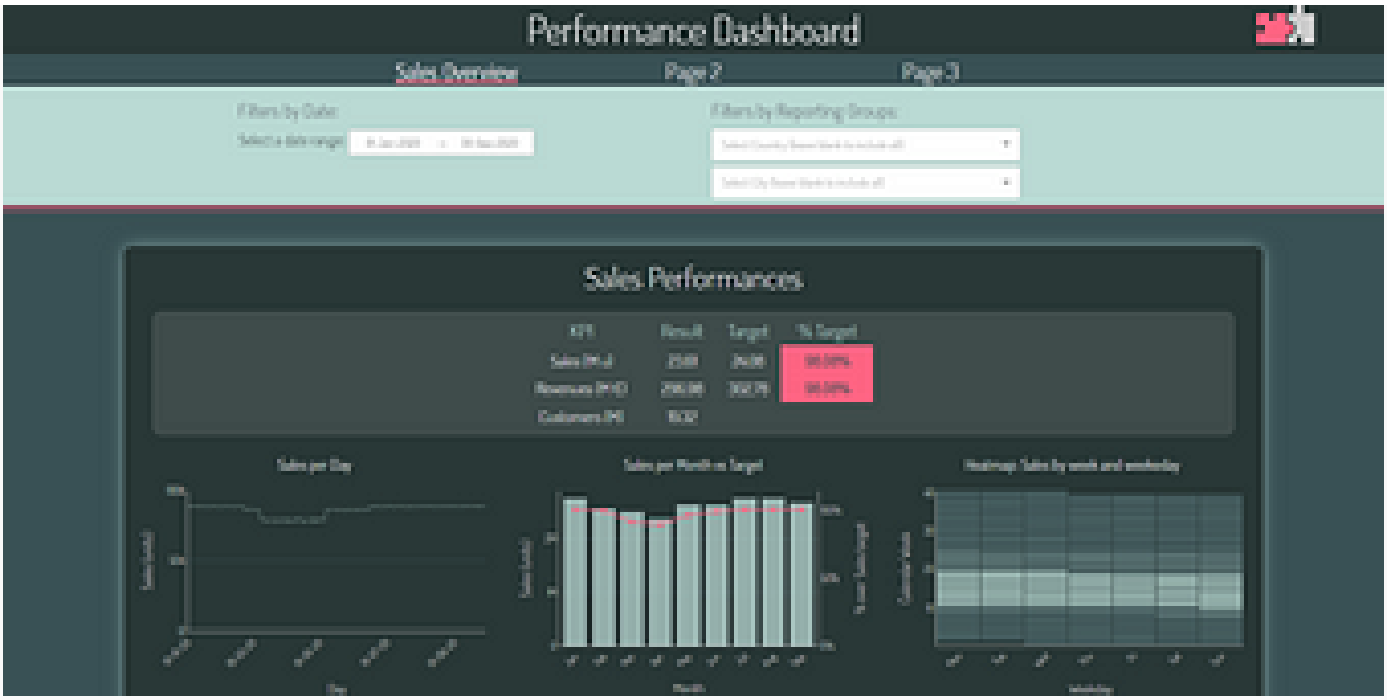


In TDS Archive by Steve Hedden

How to Implement Graph RAG Using Knowledge Graphs and Vector Databases

A Step-by-Step Tutorial on Implementing Retrieval-Augmented Generation (RAG), Semantic Search, and Recommendations

Sep 6, 2024 1.92K 19



In TDS Archive by Gabriele Albini

Create a professional dashboard with Dash and CSS Bootstrap

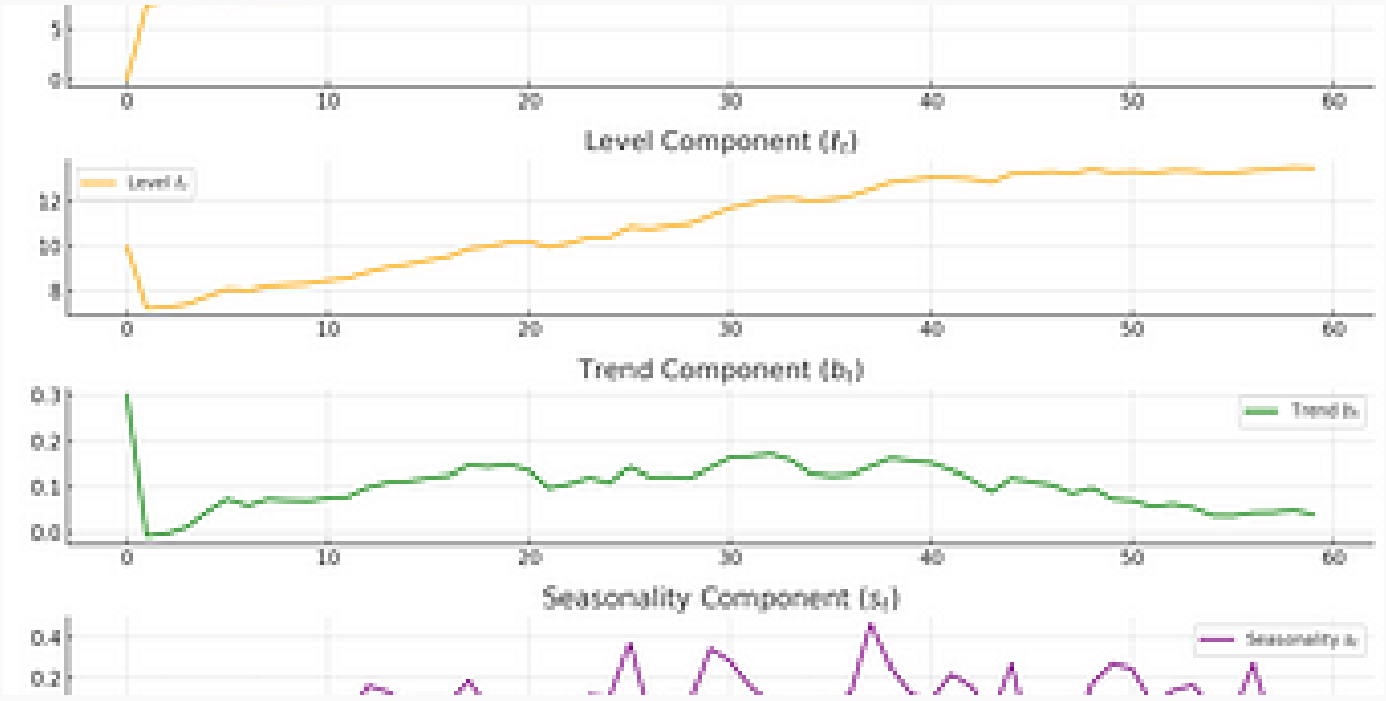
A step-by-step explained example available on Git and Heroku

Nov 1, 2020 314 4

See all from Gabriele Albini

See all from TDS Archive

Recommended from Medium

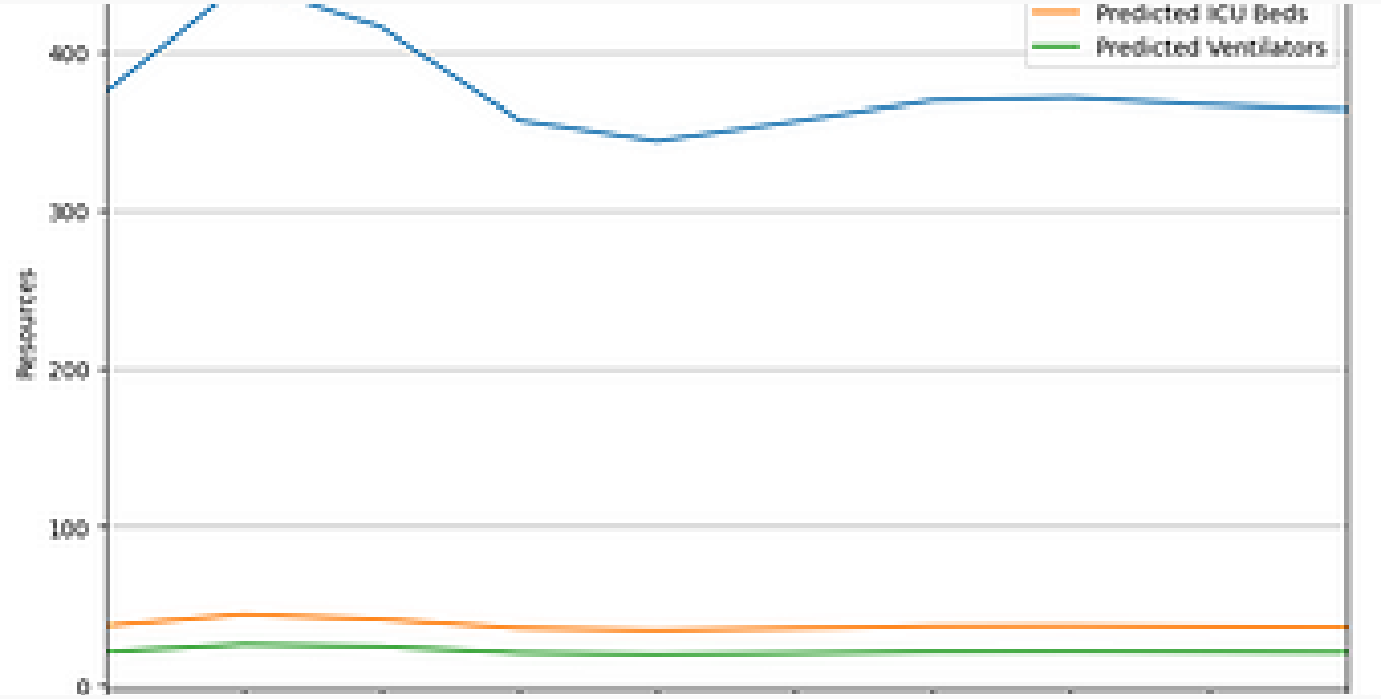


In GoPenAI by Ruth Yang

How ETS and Prophet Really Work: Forecasting Payment Approval Rates with Mathematical Clarity

By Ruth Yang | Data Clarity Series#19

Jul 4 4



ansboyas aitech

Using LSTM is Better For Forecasting OR Not ??

Predicting the future healthcare demands of a city like Chennai—especially in the event of a pandemic—is a complex challenge. The data...

Apr 18 102

In Python in Plain English by Andy McDonald

Add a Navigation Sidebar to Your Multi-Page Plotly Dash App (With Collapsible Menus)

A quick guide to building a flexible, collapsible navigation sidebar using Dash Bootstrap Components.

Jul 7 12

In Towards AI by John Loewen, PhD

Which Python Dashboard Is Better? Dash, Panel And Streamlit Showdown

Prompting GPT-4 for multi-visual interactive dashboard creation

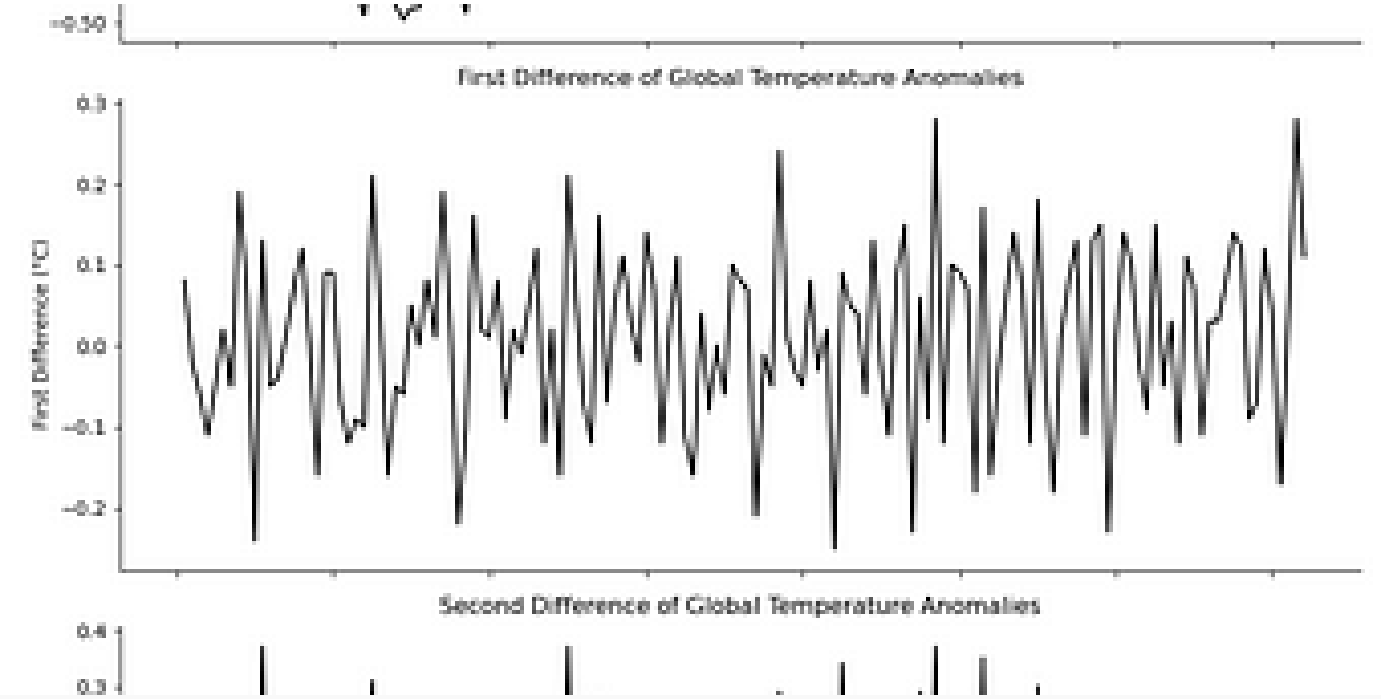
Feb 5 639 14

Dr. Shouke Wei

Visualizing Real-World Data with Python and Flet

Pandas, Matplotlib, Flet, MatplotlibChart, Exchange Rate Dataset

Jul 2 10



Kyle Jones

Transforming Non-Stationary Time Series with Differencing

Some time series data follows clear trends or patterns, making it hard to model accurately. Most statistical methods assume stationarity —...

Feb 17 9 1

See more recommendations