

★ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



Time Series Analysis and Forecasting with ARIMA in Python



Divyesh Bhatt · [Follow](#)

Published in The ML Classroom

4 min read · Nov 3, 2023



Listen



Share



More

Time series forecasting is a crucial area of machine learning that predicts future points in a series based on past data. It is especially common in economics, weather forecasting, and capacity planning, but it is applicable in many other fields. Today, we'll walk through an example of time series analysis and forecasting using the ARIMA model in Python.

Understanding ARIMA

ARIMA stands for AutoRegressive Integrated Moving Average. It is a class of models that captures a suite of different standard temporal structures in time series data. Before we apply ARIMA, we need to ensure that the series is stationary, which involves checking if its statistical properties such as mean, variance, and autocorrelation are constant over time.

The Data

We are using a dataset that represents the daily total female births in California in 1959, which is commonly used as an example in time series analysis.

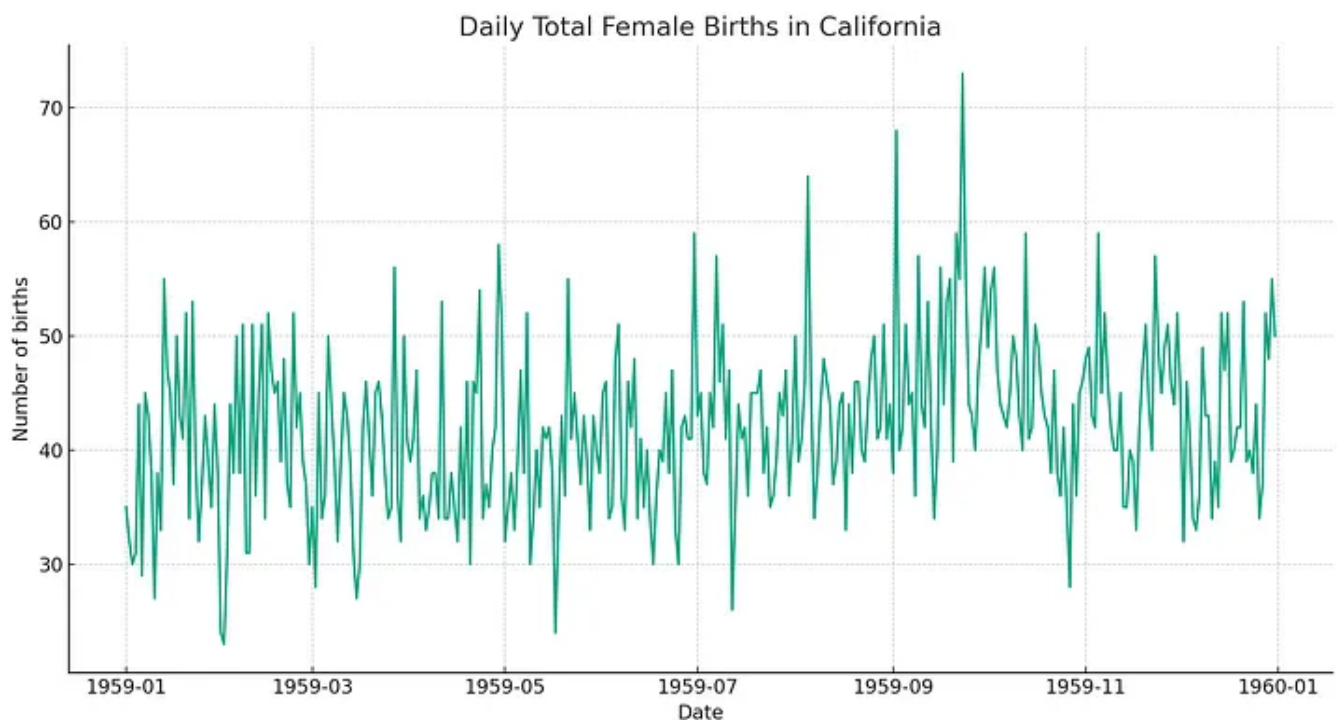
Preliminary Analysis

We start by loading the data and plotting the time series:

```
import pandas as pd
import matplotlib.pyplot as plt
```

```
# Load the dataset
data = pd.read_csv('daily-total-female-births-CA.csv')
data['date'] = pd.to_datetime(data['date'])
data.set_index('date', inplace=True)
# Plot the time series
plt.plot(data['births'])
plt.title('Daily Total Female Births in California')
plt.xlabel('Date')
plt.ylabel('Number of births')
plt.show()
```

The plot helps us visualize the data and provides initial insights into its properties, like trends and seasonality.



The plot helps us visualize the data and provides initial insights into its properties, like trends and seasonality.

Testing for Stationarity

Next, we use the Augmented Dickey-Fuller (ADF) test to check for stationarity:

```
from statsmodels.tsa.stattools import adfuller

adf_test = adfuller(data['births'])
# Output the results
```

```
print('ADF Statistic: %f' % adf_test[0])  
print('p-value: %f' % adf_test[1])
```

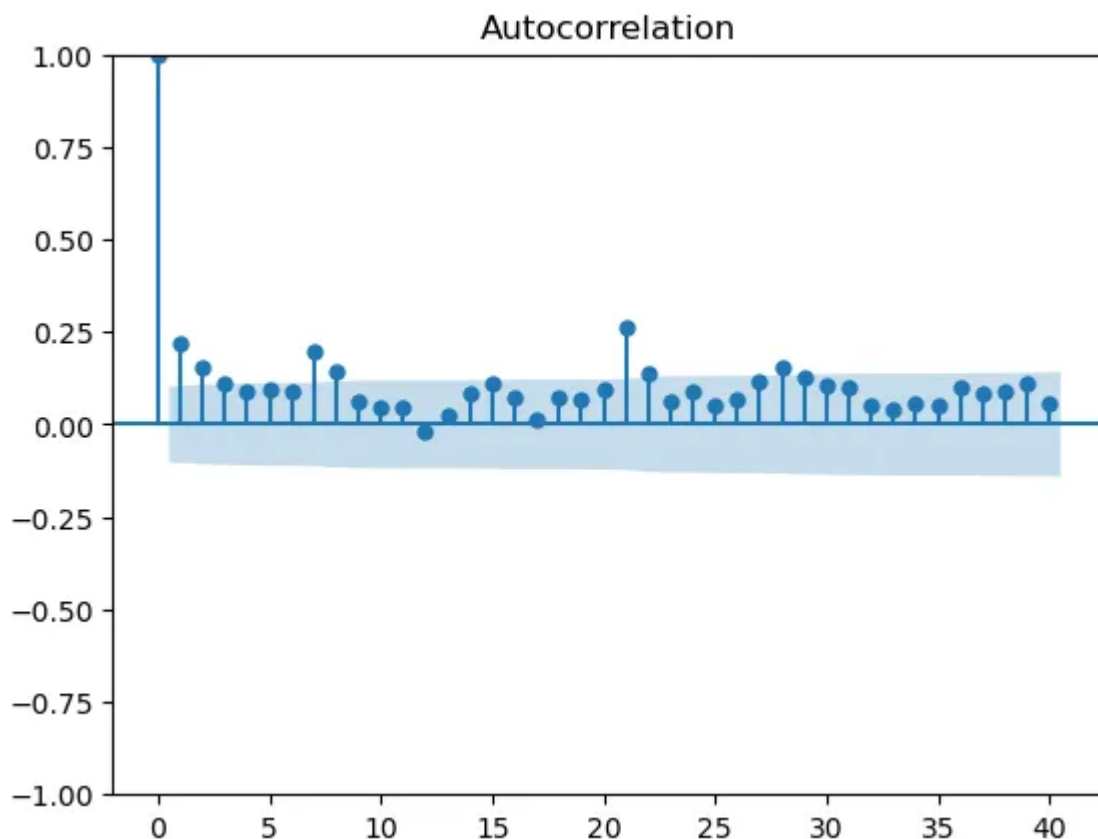
```
ADF Statistic: -4.808291  
p-value: 0.000052
```

A p-value below 0.05 indicates stationarity, and our data meets this criterion, so we do not need to difference it.

Finding ARIMA Parameters

We use the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots to find the ARIMA parameters.

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf  
plot_acf(data['births'], lags=40)  
plot_pacf(data['births'], lags=40)  
plt.show()
```



These plots suggest that an ARIMA(1, 0, 1) model may be a good starting point.

Building the ARIMA Model

```
from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(data['births'], order=(1, 0, 1))
model_fit = model.fit()
```

Training and Forecasting

We train the model on the data and perform a forecast.

```
forecast = model_fit.get_forecast(steps=30)
```

Then we visualize the forecast alongside the historical data.

Model Evaluation

To assess the model, we perform a retrospective forecast.

```
from sklearn.metrics import mean_squared_error

# Split the data into train and test
train_size = int(len(data) * 0.8)
train, test = data[0:train_size], data[train_size:len(data)]

# Fit the ARIMA model on the training dataset
model_train = ARIMA(train['births'], order=(1, 0, 1))
model_train_fit = model_train.fit()

# Forecast on the test dataset
test_forecast = model_train_fit.get_forecast(steps=len(test))
test_forecast_series = pd.Series(test_forecast.predicted_mean, index=test.index)

# Calculate the mean squared error
mse = mean_squared_error(test['births'], test_forecast_series)
rmse = mse**0.5

# Create a plot to compare the forecast with the actual test data
```

Open in app ↗



Search

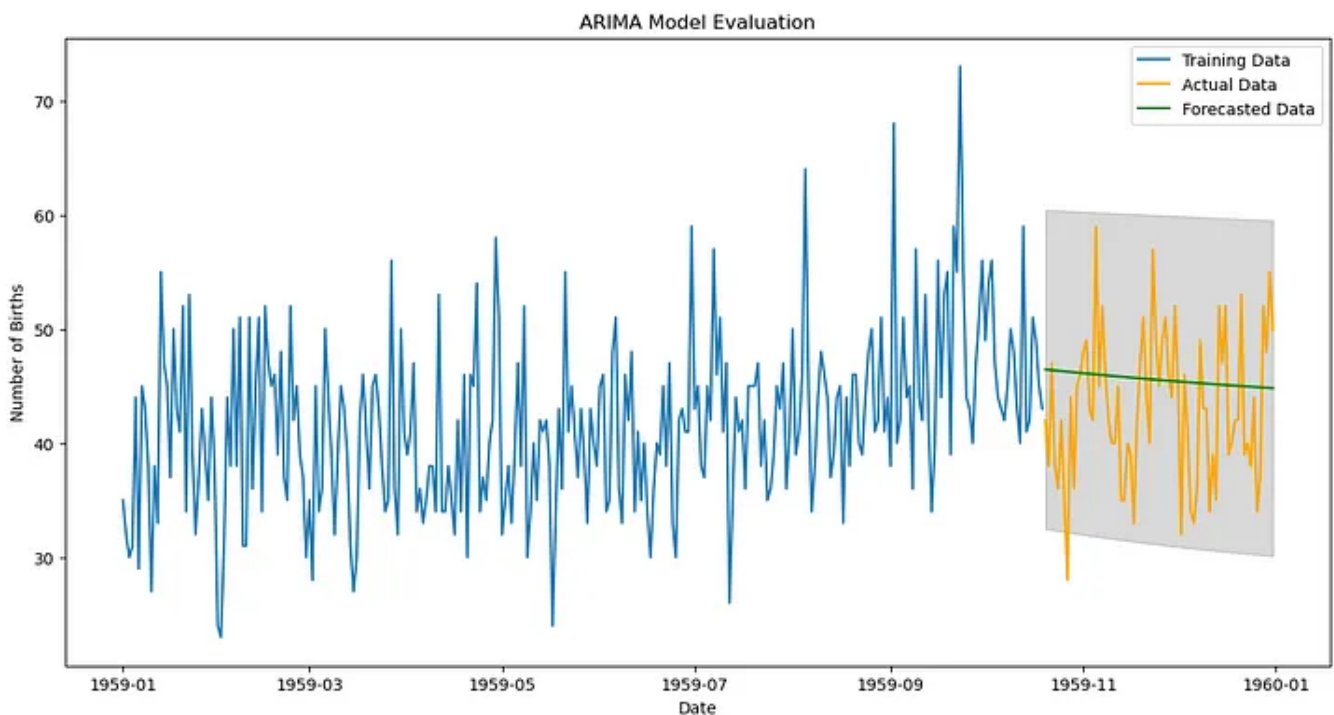


```
plt.plot_between(test_index,
                 test_forecast.conf_int().iloc[:, 0],
                 test_forecast.conf_int().iloc[:, 1],
                 color='k', alpha=.15)

plt.title('ARIMA Model Evaluation')
plt.xlabel('Date')
plt.ylabel('Number of Births')
plt.legend()
plt.show()

print('RMSE:', rmse)
```

RMSE: 6.970853456222879



The retrospective forecast (backtest) compares the forecasted data against the actual data in the test set. The Root Mean Squared Error (RMSE) of the forecast is approximately 6.97 births.

The plot shows the training data, the actual values from the test set (in orange), and the forecasted values (in green), with the 95% confidence interval shown as the shaded area. The model appears to capture the central tendency of the series but does not capture any potential within-sample variability, which is not surprising

given that ARIMA models are often more suited for data with trends or seasonality, which this series does not appear to exhibit.

The RMSE gives an indication of the forecast accuracy, with lower values indicating better fit. Whether this level of error is acceptable depends on the specific context in which the model is being used.

Conclusion

ARIMA models are a powerful tool for time series forecasting. By understanding the underlying patterns in the time series data and using statistical tests and plots to determine the model parameters, we can build a model that provides accurate and reliable forecasts.

This outline can be expanded upon with additional details and insights specific to the dataset and the results of the analysis.

Arima Model



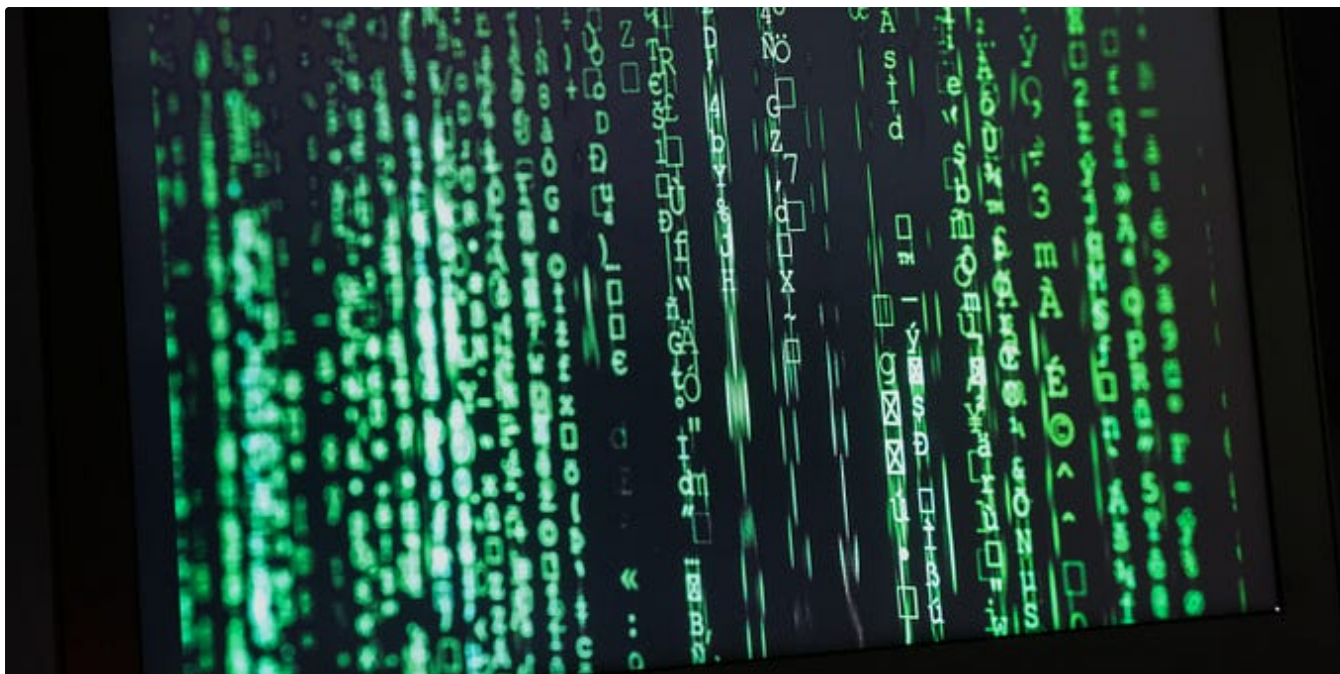
Follow

Written by Divyesh Bhatt

62 Followers · Editor for The ML Classroom

Sr. MLE & AI Enthusiast

More from Divyesh Bhatt and The ML Classroom



Divyesh Bhatt

Understanding the Jacobian Matrix and Determinant in Machine Learning

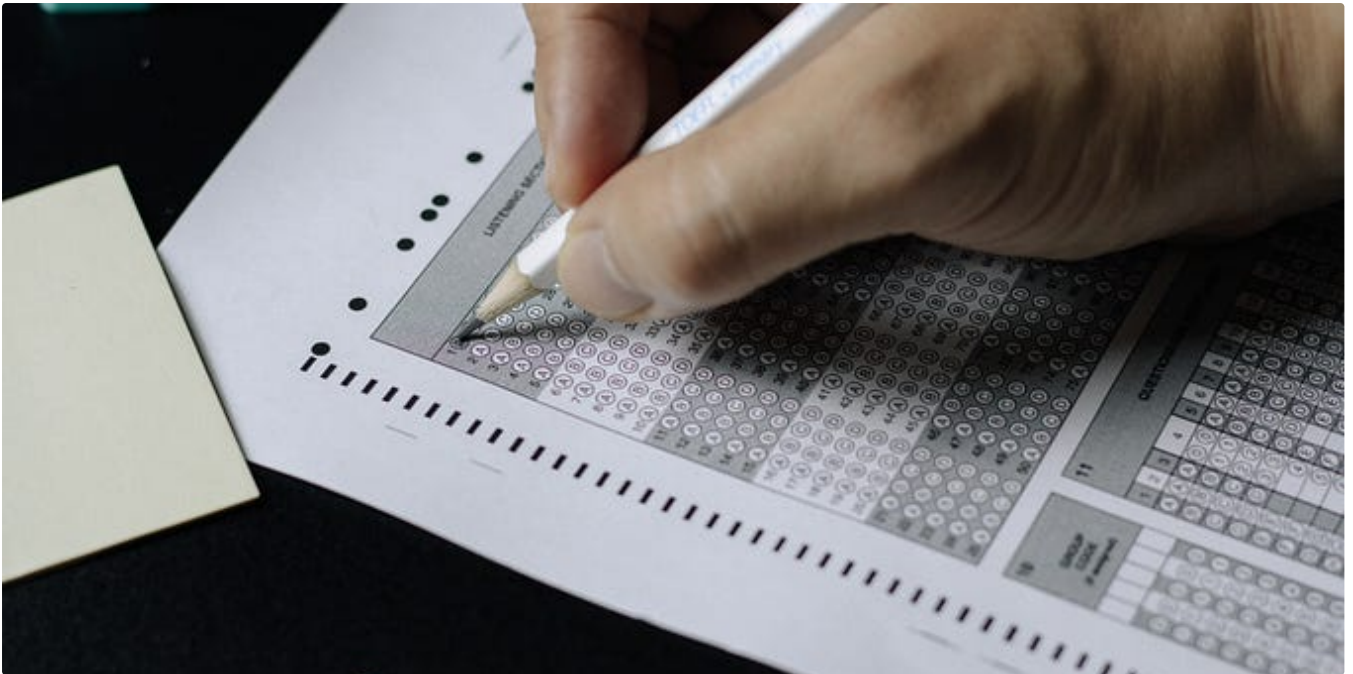
The Jacobian matrix and determinant are fundamental mathematical concepts that play a crucial role in understanding the relationships...

7 min read · Jun 17, 2023



6





Divyesh Bhatt in The ML Classroom

Classification : ROC vs PR Scores

As a machine learning practitioner, you may often come across the terms ROC and PR scores. These scores are used to evaluate the...

3 min read · Mar 22, 2023



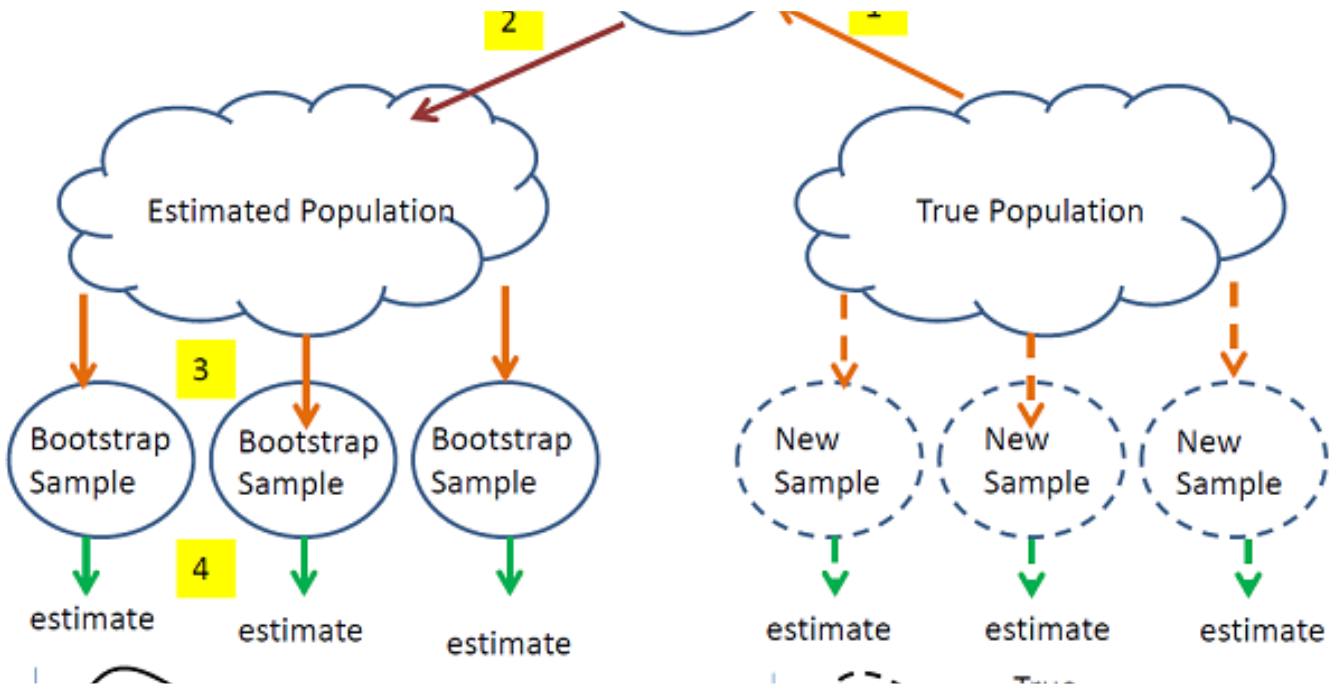
Divyesh Bhatt in The ML Classroom

Functional Programming

A subtle introduction

3 min read · May 17, 2020

117



Divyesh Bhatt

Bootstrap Method In Evaluating ML Models

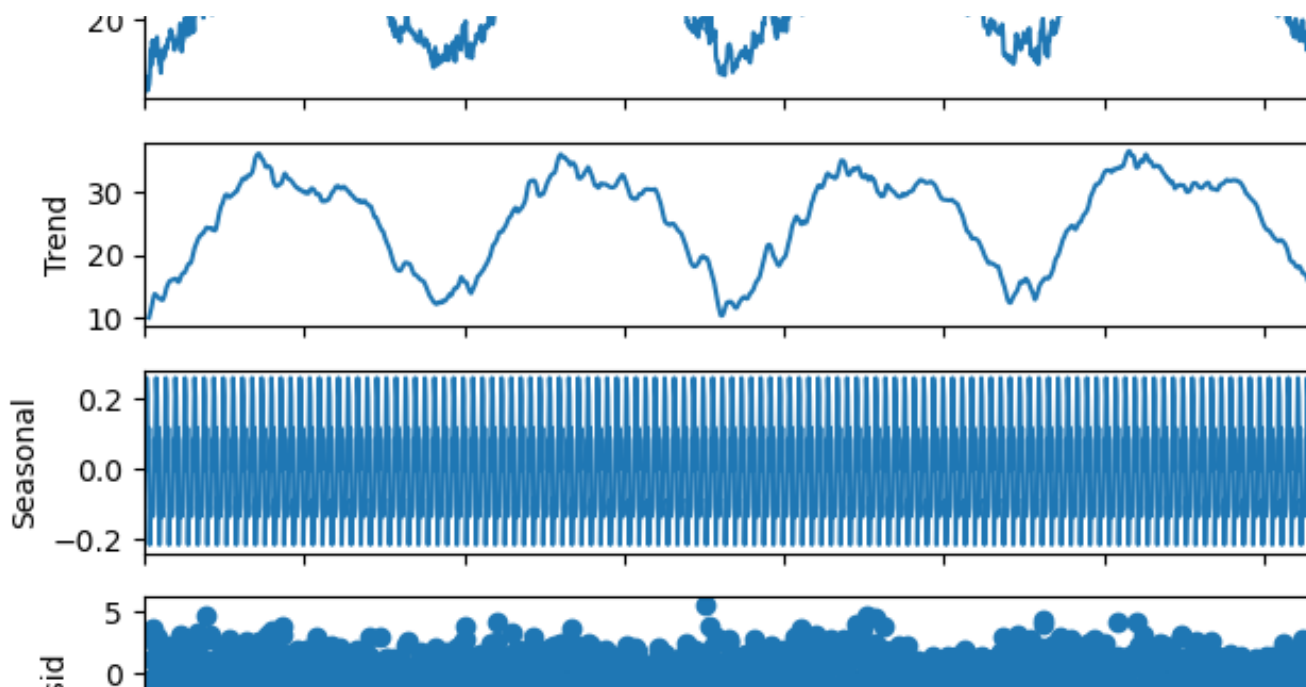
Bootstrap Method in Evaluating Machine Learning Models

2 min read · Apr 17, 2023

1

- See all from Divyesh Bhatt
- See all from The ML Classroom

Recommended from Medium



T Tirtha Mutha

Time Series forecasting using SARIMA in Python

A time series is a series of data points ordered in time. In a time series, time is often the independent variable, and the goal is usually...

5 min read · Nov 10, 2023

 20 



S. Do.

Time series forecasting—ARIMA and SARIMA

Time series forecasting is one of the most useful (and complex) fields of Machine Learning. In this article, second part of the...

7 min read · Dec 11, 2023



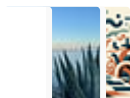
8



1

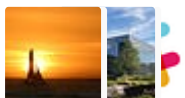


Lists



Staff Picks

597 stories · 803 saves



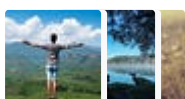
Stories to Help You Level-Up at Work

19 stories · 509 saves



Self-Improvement 101


20 stories · 1454 saves



Productivity 101

20 stories · 1338 saves



 Prof. Frenzel

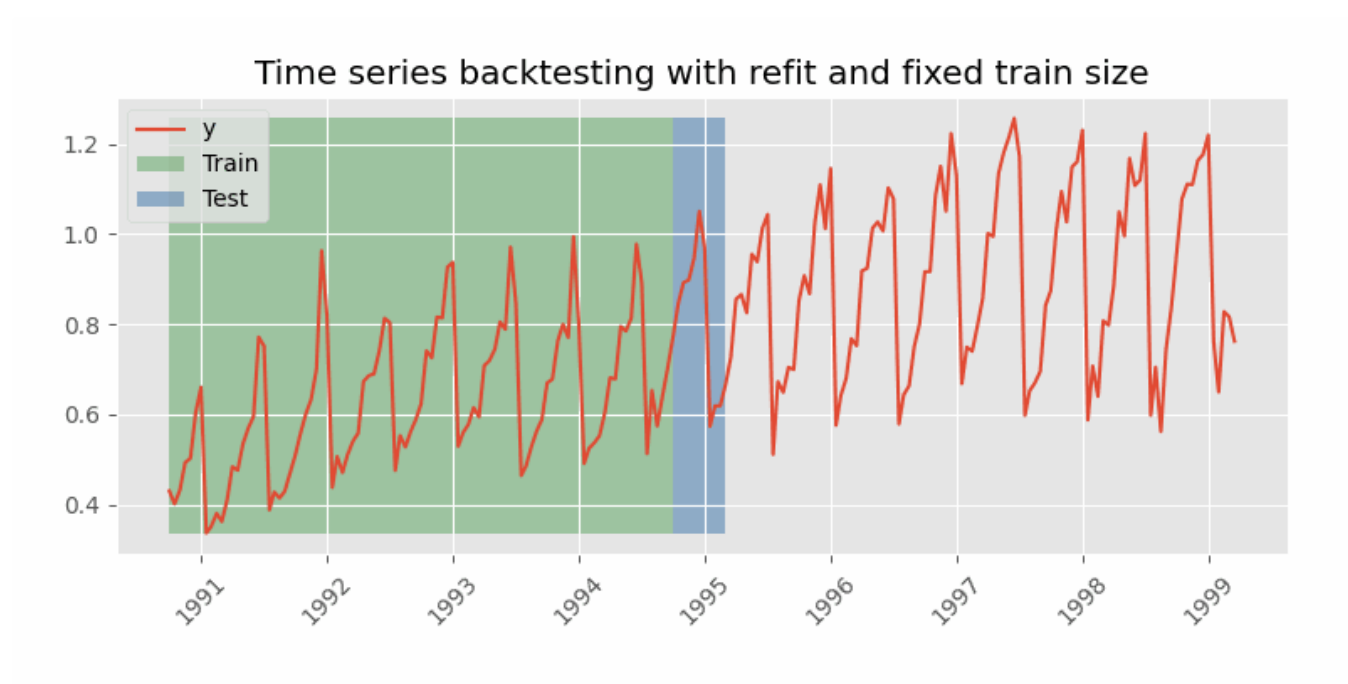
#KB Time-Series Data Part 3

Making a time-series stationary.

10 min read · Oct 1, 2023

 79 



 Everton Gomedé, PhD 

Forecasting Non-Stationary Time Series

Introduction

🌟 · 6 min read · Oct 14, 2023

👏 513 💬 3



 Vasilis Kalyvas in Python in Plain English

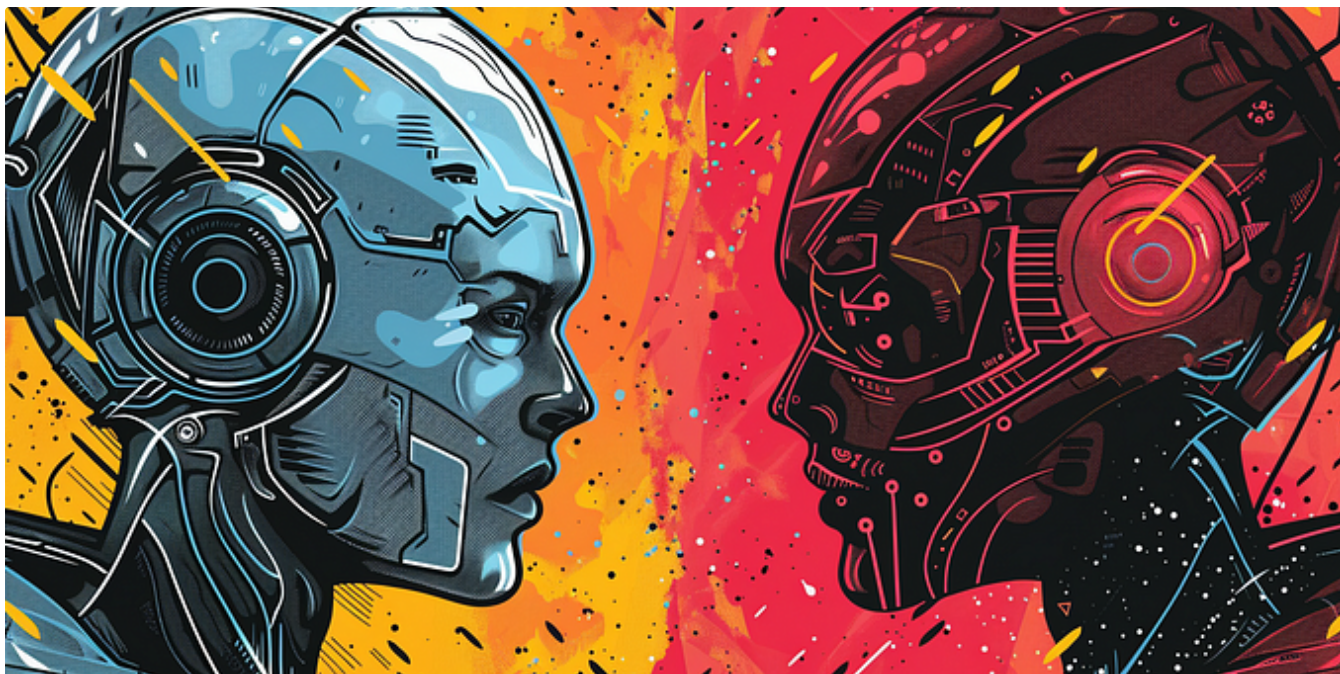
Time Series Episode 0: Familiarize with ARIMA and its parameters

Bonus: Tips and tricks for parameters' selection

🌟 · 10 min read · Nov 17, 2023

👏 156 💬





Mirko Peters in Mirko Peters—Data & Analytics Blog

Comparative Analysis of Time Series Forecasting: ARIMA vs LSTM for Stock Price Prediction

In the ever-evolving landscape of financial markets, the quest for accurate stock price prediction models has never been more rigorous.

★ · 14 min read · 4 days ago



See more recommendations