Angelica Lo Duca Follow

Sep 9, 2020 · 6 min read · · Listen

How to model a time series through a SARIMA model

A tutorial to model seasonal time series.

Source by OpenClipart-Vectors from Pixabay

In this tutorial I will show you how to model a seasonal time series through a SARIMA model.

Here you can download the Jupyter notebook of the code described in this tutorial.

Getting Started

Convert the dataset into a time series

In this example we will use the number of tourist arrivals to Italy. Data are extracted from the European Statistics: Annual Data on Tourism Industries. Firstly, we import the dataset related to foreign tourists arrivals in Italy from 2012 to 2019 October and then we convert it into a time series.

In order to perform the conversion to time series, two steps are needed:

- the column containing dates must be converted to datetime. This can be done through the function `to_datetime()` , which converts a string into a datetime.
- set the index of the dataframe to the column containing dates. This can be done through the function `set_index()` applied to the dataframe.

```
import pandas as pd

df = pd.read_csv('../sources/IT_tourists_arrivals.csv')
df['date'] = pd.to_datetime(df['date'])
df = df[df['date'] > '2012-01-01']
df.set_index('date', inplace=True)
```

	value
date	
2012-02-01	10468842
2012-03-01	13908950
2012-04-01	18456089
2012-05-01	20294254
2012-06-01	27101300
...	...
2019-05-01	24832942
2019-06-01	34658825
2019-07-01	39123041
2019-08-01	41588218
2019-09-01	30253817

We can get some useful statistics related to the time series through the `describe()` function.

```
df.describe()
```

	value
count	9.200000e+01
mean	2.164878e+07
std	9.780261e+06
min	9.632532e+06
25%	1.291822e+07
50%	1.914685e+07
75%	2.808559e+07
max	4.158822e+07

Preliminary analysis

Plot the time series to check the seasonality

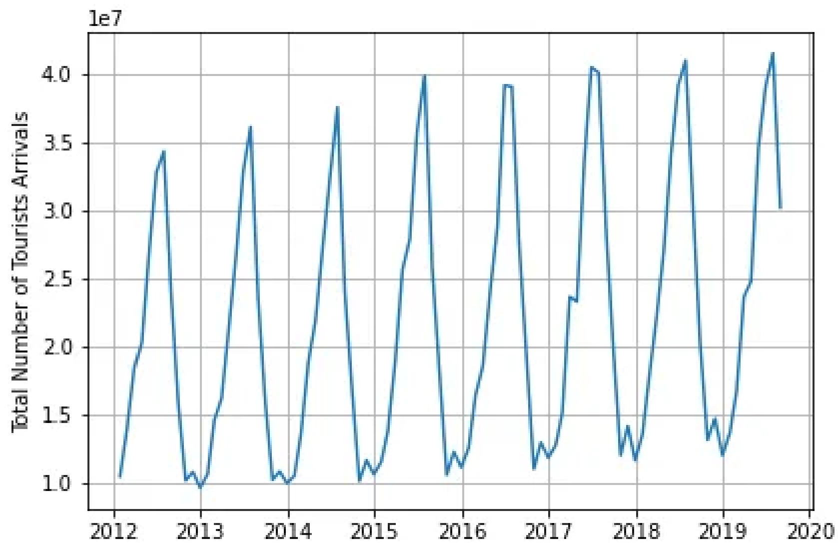
The preliminary analysis involves a visual analysis of the time series, in order to understand its general trend and behaviour. Firstly, we create the time series and we store it in the variable `ts`.

```
ts = df['value']
```

Then, we plot the `ts` trend. We use the `matplotlib` library provided by Python.

```
import matplotlib.pylab as plt
plt.plot(ts)
plt.ylabel('Total Number of Tourists Arrivals')
```

```
plt.grid()
plt.tight_layout()
plt.savefig('plots/IT_tourists_arrivals.png')
plt.show()
```



Calculate the parameters for the model

Tune the model

We build a SARIMA model to represent the time series. SARIMA is an acronym for Seasonal AutoRegressive Integrated Moving Average. It is composed of two models AR and MA. The model is defined by three parameters:

- d = degree of first differencing involved
- p = order of the AR part
- q = order of the moving average part.

The value of p can be determined through the ACF plot, which shows the autocorrelations which measure the relationship between an observation and its previous one. The value of d is the order of integration and can be calculated as the number of transformations needed to make the time series stationary. The value of q can be determined through the PACF plot.

In order determine the value of `d`, we can perform the Dickey-Fuller test, which is able to verify whether a time series is stationary or not. We can use the `adfuller` class, contained in the `statsmodels` library. We define a function, called `test_stationarity()`, which returns True, if the time series is positive, False otherwise.

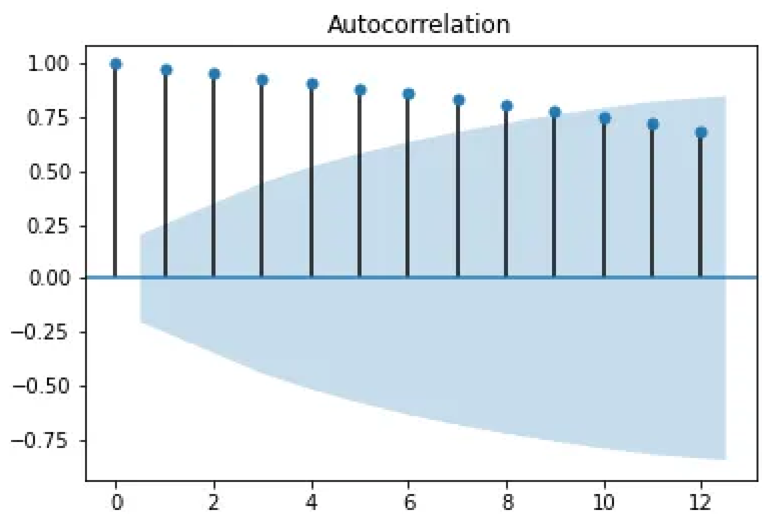
```
from statsmodels.tsa.stattools import adfuller

def test_stationarity(timeseries):

    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-
value','#Lags Used','Number of Observations Used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value

    critical_value = dftest[4]['5%']
    test_statistic = dftest[0]
    alpha = 1e-3
    pvalue = dftest[1]
    if pvalue < alpha and test_statistic < critical_value:  # null
hypothesis: x is non stationary
        print("X is stationary")
        return True
    else:
        print("X is not stationary")
        return False
```

We transform the time series through the `diff()` function as many times as the time series becomes stationary.

```
ts_diff = pd.Series(ts)
d = 0
while test_stationarity(ts_diff) is False:
    ts_diff = ts_diff.diff().dropna()
    d = d + 1
```

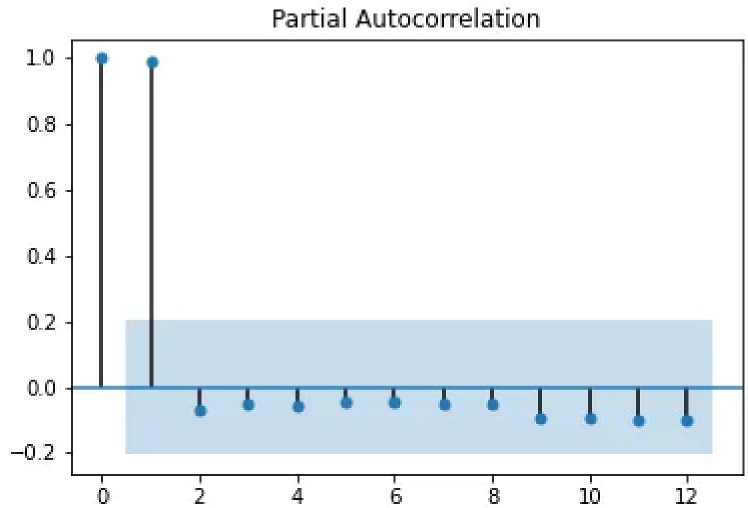
In order to calculate the value of p and q, we can plot the ACF and PACF graphs, respectively. We can use the `plot_acf()` and `plot_pacf()` functions available in the `statsmodels` library. The value of p corresponds to the maximum value in the ACF graph external to the confidence intervals (shown in light blue). In our case, che correct value of p = 9.

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
plot_acf(ts_trend, lags =12)
plt.savefig('plots/acf.png')
plt.show()
```



Similarly, the value of q corresponds to the maximum value in the PACF graph external to the confidence intervals (shown in light blue). In our case, che correct value of q = 1.

```
plot_pacf(ts_trend, lags =12)
plt.savefig('plots/pacf.png')
plt.show()
```



Build the SARIMA model

How to train the SARIMA model

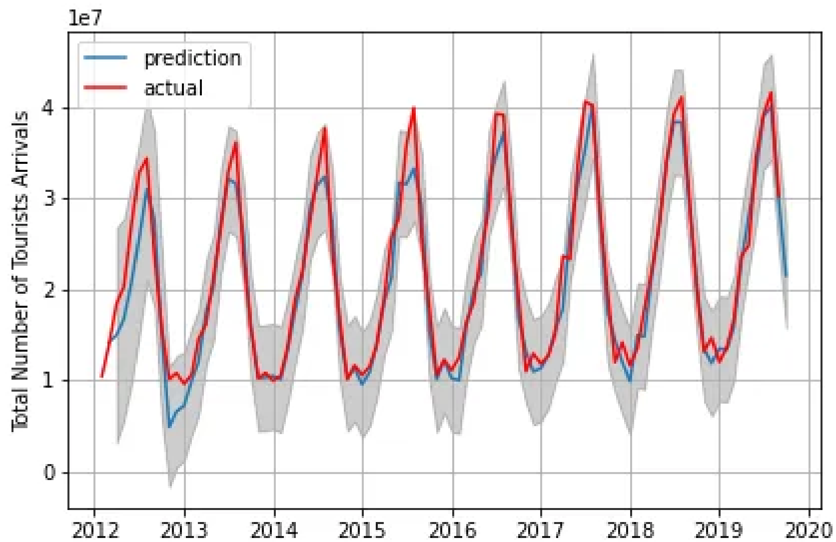
Now we are ready to build the SARIMA model. We can use the `SARIMAX` class provided by the `statsmodels` library. We fit the model and get the prediction through the `get_prediction()` function. We can retrieve also the confidence intervals through the `conf_int()` function.

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

p = 9
q = 1
model = SARIMAX(ts, order=(p,d,q))
model_fit = model.fit(disp=1,solver='powell')

fcast = model_fit.get_prediction(start=1, end=len(ts))
ts_p = fcast.predicted_mean
ts_ci = fcast.conf_int()
```

We plot results.

```
plt.plot(ts_p,label='prediction')
plt.plot(ts,color='red',label='actual')
plt.fill_between(ts_ci.index[1:],
                ts_ci.iloc[1:, 0],
                ts_ci.iloc[1:, 1], color='k', alpha=.2)
```

```
plt.ylabel('Total Number of Tourists Arrivals')
plt.legend()
plt.tight_layout()
plt.grid()
plt.savefig('plots/IT_trend_prediction.png')
plt.show()
```



Calculate some statistics

Check the performance of the model

Finally, we can calculate some statistics to evaluate the performance of the model. We calculate the Pearson's coefficient, through the `pearsonr()` function provided by the `scipy` library.

```
from scipy import stats
stats.pearsonr(ts_trend_p[1:], ts[1:])
```

We also calculate the R squared metrics.

```
residuals = ts - ts_trend_p
ss_res = np.sum(residuals**2)
ss_tot = np.sum((ts-np.mean(ts))**2)
```

```
r_squared = 1 - (ss_res / ss_tot)
r_squared
```

Lesson learned

In this tutorial I have illustrated how to model a time series through a SARIMA model. Summarising, you should follow the following steps:

- convert your data frame into a time series
- calculate the values of p, d and q to tune the SARIMA model
- build the SARIMA model with the calculated p, d and q values
- test the performance of the model.

An improvement of the proposed model could involve the splitting of the time series in two parts: training and test sets. Usually, the training set is used to fit the model while the test set is used to calculate the performance of the model.

Bibliography

- A Visual Guide to Time Series Decomposition Analysis
- An example of time series
- ARIMA Model Python Example — Time Series Forecasting
- ARIMA model to forecast international tourist visit in Bumthang, Bhutan
- Augmented Dickey-Fuller Test in Python
- Auto ARIMA using Pyramid ARIMA Python Package
- Confidence Interval for t-test (difference between means) in Python
- Extracting Seasonality and Trend from Data: Decomposition Using R
- How to Create an ARIMA Model for Time Series Forecasting in Python
- How can I make a time-series stationary?

- How to Remove Trends and Seasonality with a Difference Transform in Python
- How to Tune ARIMA Parameters in Python
- Non-seasonal ARIMA models
- Interpret the partial autocorrelation function (PACF)
- Stationarity testing using the Augmented Dickey-Fuller test
- The signal and the noise
- Time Series Decomposition
- Time Series Forecast : A basic introduction using Python.
- Time Series Analysis in Python — A Comprehensive Guide with Examples
- Time Series Forecast Case Study with Python: Monthly Armed Robberies in Boston
- Time Series prediction using SARIMAX
- Understand ARIMA and tune P, D, Q

Data Analysis Data Science Time Series Forecasting Tutorial Python