**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

f (https:*/*/ww*w.*facebook.com/AnalyticsVidhya) |

(https:*/*/twitter.com/analyticsvidhya*)*

8+ (https://plus.google.com/+Analyticsvidhya/posts)

in (https:*//ww*w.linkedin.com/groups*/*Analytics-Vidhya-Learn-everything-about-5057*1*65)

.

Analytics Vidhya (https://www.analyticsvidhya.com) Learn Everything About Analytics

(https:*//*w*ww*.analyticsvidhya.com)

Home (https:*//www.*analy**tiqamah S**

WWW.

Ii I:

**M P Terbaru montirattom*etro*s yan**

**I. L'**

**H IV** UTILINIAI!

**HiliTV:'I TRIMIT** INITI**.' .W*Z*MLULUI**

**TUDI**

**TTTT.**

''

**erhat*i***

**an**

**WRESPOND)**

**UWTLEM*IL*IH**

ii!

Ilivil

bli:

21ri*t*ult

Complete guide to create a Time Series Forecast (with Codes in Python)

MACHINE LEARNING

BUSINESS ANALYTICS (HTTPS:*/*/WWW*.*ANALYTICSVIDHYA.COM/BLOG/CATEGORY/BUSINESS-ANALYTICS, (HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/MACHINE-LEARNINGA PYTHON

(HTTPS://WWW.ANALYTICSVIDHYA.COM/BLOG/CATEGORY/PYTHON-27)

**com*/*sharer.php**?u=http*s://*www*.*analyticsvidhya.com/blo*g*/2016*/***02*/*time-series-forecasting-codes** 20to%20cr**eate%20a%20Time%20Series%20Forecas**t%20(with%20Codes%20in%20Python)) y (https*://***twitter*.*com*/*home?**

%20create%20a%20Time%20Series%20Forecast%20(with%20Codes%20in%20Python):https*://*w**ww.ana**lyticsvidhya.com/blog/2016/02/time

8+ (https*:/*/plus.google.com/share?url=https*://*w**ww.analyticsvidhya.co**m/blog/2016/02*/***time-series-forecasting-codes-python/) *P*** /button/?url=https*://w*ww.analyticsvidhya.com*/*blog/2016/02*/***time-series-forecasting-codes** yticsvidhya.c**om/wp pg&descript**ion-Complete%20guide%20to%20create%20a%20Time%20Series%20Forecast%20(with%20Codes%20in%20Python))

BRIDGE

SCHDOL DE PROFEXONAL STUDIES

NORTHSESTERN

(http://admissions.bridgesom.com/pba-new*/?* utm\_source=AV&utm\_medium=Bannerlnline&utm\_campaign=AVBanner20August)

Introduction

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

***1*/*4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

Time Series (referred as TS from now) is considered to be one of the less known skills in the analytics space (Even I had little clue about it a couple of days back). But as you know our inaugural Mini Hackathon (http:*//*datahack.analyticsvidhya.com/contest/mini-datahack) is based on it, I set myself on a journey to learn the basic steps for solving a Time Series problem and here I am sharing the same with you. These will definitely help you get a decent model in our hackathon today*.*

71

Vidhya Learn Everything About Analytics

COMPLETE GUIDE

TO CREATE A "IME SERIES FORECAST

WITH

(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/guide.jpg)

Before going through this article, I highly recommend reading A Complete Tutorial on Time Series Modeling in R (https:*//ww*w.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series modeling/), which is like a prequel to this article. It focuses on fundamental concepts and is based on R and I will focus on using these concepts in solving a problem end-to-end along with codes in Python. Many resources exist for TS in R but very few are there for Python so I'll be using Python in this article.

Out journey would go through the following steps:

1. What makes Time Series Special? 2. Loading and Handling Time Series in Pandas 3. How to Check Stationarity of a Time Series? 4. How to make a Time Series Stationary? 5. Forecasting a Time Series

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**2*4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

**AMAZING**

**SKILLTEST INR**

**[AUGUST ]**

**FOR DATA SCIENCE**

REGISTER

**28-08-2016**

**ONLINE**

(https://datahack.analyticsvidhya.com/contest/skilltest-r-for-data-science/)

1. What makes Time Series Special?

As the name suggests, TS is a collection of data points collected at **constant time intervals**. These are analyzed to determine the long term trend so as to forecast the future or perform some other form of analysis. But what makes a TS different from say a regular regression problem? There are 2 things:

1. It is **time dependent.** So the basic assumption of a linear regression model that the observations are

independent doesn't hold in this case. 2. Along with an increasing or decreasing trend, most TS have some form **of seasonality trends,** i.e.

variations specific to a particular time frame. For example, if you see the sales of a woolen jacket over time, you will invariably find higher sales in winter seasons.

Because of the inherent properties of a TS, there are various steps involved in analyzing it. These are discussed in detail below. Lets start by loading a TS object in Python. We'll be using the popular AirPassengers data set which can be downloaded here (https://www.analyticsvidhya.com/wp content/uploads/2016/02/AirPassengers.csv*).*

Please note that the aim of this article is to familiarize you with the various techniques used for TS in general. The example considered here is just for illustration and I will focus on coverage a breadth of topics and not making a ver*y* accurate forecast.

2. Loading and Handling Time Series in Pandas

Pandas has dedicated libraries for handling TS objects, particularly the **datatime64Ins**) class which **st**ores time information and allows us to perform some operations really fast. Lets start by firing up the required libraries:

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

***3*/*4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

impor**t pandas as pd import numpy as np**

import m**atplotlib.pylab as plt**

%matplotlib inline

from matplotlib.pylab import rcparams rcParams['figure.figsize'] = 15, 6

Now, *w*e can load the data set and look at some initial rows and data types of the columns

**data = pd.read\_csv*('A*irPassengers.csv')** print data.head()

print '\n Data Types:'

prin**t data.dtypes**

OHNM +

**Month *#*Passengers 1949-01**

112 **1949-02**

**118 1949-03**

132 **1949-04**

129 **1949-05**

**121**

**Data Types: Month**

***#P*assengers dtype: object**

**object int64**

(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/1.-dataload-1.png)

The data contains a particular month and number of passengers travelling in that month. But this is still not read as a TS object as the data types are 'object' and 'int'. In order to read the data as a time series, we have to pass special arguments to the read\_csv command:

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

***4*/*4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

**dateparse = lambda dates: pd.datetime.strptime (dates, '%Y-%m') data = pd.read\_csv*(*'AirPassengers.csv', parse\_dates=**'Month', index\_col='Month',**date\_parser=dateparse)**

**print data.head()**

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

***#*Passengers**

**Month**

**1949-01-01 1949-02-01 1949-03-01 1949-04-01 1949-05-01**

112 **118** 132 **129** 121

(https:*//w*ww.analyticsvidhya.com/wp-content/uploads/2016/02/2.-dataload-2.png)

Let's understand the arguments one by one:

**1. parse\_dates**: This specifies the column which contains the date-time information. As we say above,

the column name is 'Month'. **2. index\_c**ol: A key idea behind using Pandas for TS data is that the index has to be the variable

depicting date-time information. So this argument tells pandas to use the 'Month' column as index. **3. date\_parser**: This specifies a function which converts an input string into datetime variable. Be default

Pandas reads data in format 'YYYY-MM-DD HH:MM:SS. If the data is not in this format, the format has to be manually defined. Something similar to the dataparse function defined here can be used for this purpose.

Now *we* can see that the data has time object as index and #Passengers as the column. We can cross-check the datatype of the index with the following command:

**data.index**

DatetimeIndex(['1949-01-01', '1949-02-01', '1949-03-01', '1949-04-01',

'1949-05-01', '1949-06-01', '1949-07-01', '1949-08-01', '1949-09-01', '1949-10-01',

1960-03-01', '1960-04-01', 1960-05-01', '1960-06-01', '1960-07-01', '1960-08-01', '1960-09-01', '1960-10-01', '1960-11-01', '1960-12-01'], **dtype='datetime 64[ns]', name=u' Month', length=144, freq=None)**

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**5/*4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

(https:*//www.*analyticsvidhya.com/wp-content/uploads/2016/02/3.-index-type.png)

Notice the **dtype='datetimeIns!**' which confirms that it is a datetime object. As a personal preference, I would convert the column into a Series object to prevent referring to columns names e*v*ery time I use the TS. Please feel free to use as a dataframe is that works better for you.

ts = data['*#*Passengers'] ts.head(10)

**Month**

**1949-01-01 112 1949-02-01** 118 **1949-03-01** 132 **1949-04-01**

**129 1949-05-01** 121 **1949-06-01** 135 **1949-0*7*-01 148 1949-08-01 148 1949-09-01 136 1949-10-01** 119 **Name: *#*Passengers, dtype: int64**

(https://*www.*analyticsvidhya.com/wp-content/uploads/2016/02/4.-series.png)

Before going further, I'll discuss some indexing techniques for TS data. Lets start by selecting a particular value in the Series object. This can be done in following 2 ways:

**#1. Specific the index as a string constant:** ts['1949-01-01']

#2. Import the datetime librar**y and use "datet**ime' function: from datetime impor**t datetime ts[datetime (1949,1,1)]**

Both would return the value '112' which can also be confirmed from previous output. Suppose w*e* want all the data upto May 1949. This can be done in 2 ways*:*

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**6/*4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

**#1. Specify the entire range:** ts['1949-01-01':'1949-05-01']

#2. Use ':' if one of the **indices is at ends:** ts[:'1949-05-01']

Both would yield following output:

**Month 1949-01-01** 112 **1949-02-01 118 1949-03-01** 132 **1949-04-01 129 1949-05-01** 121 **Name: *#*Passengers, dtype: int64**

(https:*//ww*w.analyticsvidhya.com/wp-content/uploads/2016/02/5.-index-range.png)

There are 2 things to note here:

1. Unlike numeric indexing, the **end index is included here.** For instance, if we index a list as al:5] then it would return the values at indices - 10,1,2,3,4). But here the index '1949-05-01' was included in the

output. 2 The **indices have to be sorted f**or ranges to work. If you randomly shuffle the index, this won't work.

Consider another instance where you need all the values of the year 1949. This can be done as:

ts['1949']

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

***7/4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

**Month 1949-01-01** 112 **1949-02-01 118 1949-03-01 132 1949-04-01 129** 1949-05-01 **121 1949-06-01 135 1949-07-01 148 1949-08-01 148 1949-09-01 136 1949-10-01 119 1949-11-01 104 1949-12-01 118 Name: *#*Passengers, dtype: int64**

(https:*//ww*w.analyticsvidhya.com/wp-content/uploads/2016/02/6.-index-year.png)

The month part was omitted. Similarly if you all days of a particular month, the day part can be omitted.

Now, lets move onto the analyzing the TS.

3. How to Check Stationarity of a Time Series?

A TS is said to be stationary if its **statistical properties** such as mean, variance remain **constant over time.** But why is it important? Most of the TS models work on the assumption that the TS is stationary. Intuitivel*y,* we can sat that if a TS has a particular behaviour over time, there is a very high probability that it will follow the same in the future. Also, the theories related to stationary series are more mature and easier to implement as compared to non-stationary series.

Stationarity is defined using very strict criterion. Howe*v*er, for practical purposes we can assume the series to be stationary if it has constant statistical properties over time, ie. the following:

1. constant mean 2. constant variance 3. an autocovariance that does not depend on time.

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**8/*4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

I'll skip the details as it is very clearly defined in this article (https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/). Lets move onto the ways of testing stationarity. First and foremost is to simple plot the data and analyze visually. The data can be plotted using following command:

plt.plot(ts)

MWW WWWW

100 L

1949

1951

1953

1953

1955

1957

1959

1959

(https://www.analyticsvidhya.com/wp-content/uploads/2016/02*/7.*-ts.png)

It is clearly evident that there is an **overall increasing trend** in the data along with some seasonal variations. However, it might not always be possible to make such visual inferences (we'll see such cases later). So, more formally, we can check stationarity using the following:

**1. Plotting Rolling Statistics**: We can plot the moving average or moving variance and see if it varies with time. By moving average/variance I mean that at any instant 't', we'll take the average/variance of

the last year, i.e. last 12 months. But again this is more of a visual technique. **2 Dickey-Fuller Test**: This is one of the statistical tests for checking stationarity. Here the null hypothesis

is that the TS is non-stationary. The test results comprise of a **Test Statistic** and some **Critical Values** for difference confidence levels. If the 'Test Statistic' is less than the 'Critical Value', we can reject the null hypothesis and say that the series is stationary. Refer this article (https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/) for details.

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

***9/4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

These concepts might not sound very intuitive at this point. I recommend going through the prequel article. If you're interested in some theoretical statistics, you can refer **Introduction to Time Series and Forecasting by Brockwell and Davis.** The book is a bit stats-heavy, but if you have the skill to read-between-lines, you can understand the concepts and tangentially touch the statistics.

Back to checking stationarity, we'll be using the rolling statistics plots along with Dickey-Fuller test results a lot so I have defined a function which takes a TS as input and generated them for us. Please note that I've plotted standard deviation instead of variance to keep the unit similar to mean.

from **statsmodels.tsa.stattools import adfuller**

**def test\_stationarity(timeseries):**

#Determing rolling s**tatistics**

**rolmean = pd.rolling\_mean(timeseries, window=12*)***

rolstd = pd.rolling\_std(**timeseries, window=12)**

**#Plot rolling statistics:**

orig = plt.plot(**timeseries, c**olor='blue',label='Original')

**mean** = plt.plot(rolmean, color='red', **label='Rolling Mean')**

std = plt.plot(rolstd, color='black', label = 'Rolling Std') plt.legend(loc='best') plt.title('Rolling Mean & Standard Deviation') **plt.show(block=False)**

**#Perform Dickey-Fuller test:**

print 'Results of Dickey-Fuller Test:' **dftest = a**dfuller(**timeseries, autolag=**'AIC')

dfoutput = pd.Series(dftest[0:4], index=['T**est Statist**ic', 'p-value', '#Lags Used', 'Numbe**r of Obser vatio**ns Used'])

**for key, val**ue in dftest[4].items():

dfoutput['Critic**al Value *(*%)'%key] = value**

print dfoutput

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**10/49**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

The code is pretty straight forward. Please feel free to discuss the code in comments if you face challenges in grasping it.

Let's run it for our input series:

**test\_stationa**rity(ts)

Rolling Mean & Standard Deviation

Original Rolling Mean Rolling Std

-

1949

1951

1953

1955

1959

1959

**Results of Dickey-Fuller Test: Test Statistic p-value *#*Lags Used Number of Observations Used Critical Value (5%) Critical Value (1) Critical Value (10%) dtype: float64**

**0.815369 0.991880** 13.000000 **130.000000 -2.884042 -3.481682** -2.578770

(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/1.-dfuller-ts.png)

Though the variation in standard deviation is small, mean is clearly increasing with time and this is not a stationary series. Also, the test statistic is way more than the critical values. Note that the **signed *v*alues should be compared** and not the absolute values.

Next, we'll discuss the techniques that can be used to take this TS towards stationarity.

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**1*1*/49**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

4. How to make a Time Series Stationary?

Though stationarity assumption is taken in many TS models, almost none of practical time series are stationary. So statisticians have figured out ways to make series stationary, which we'll discuss now. Actually, its almost impossible to make a series perfectly stationary, but we try to take it as close as possible.

Lets understand what is making a TS non-stationary. There are 2 major reasons behind non stationaruty of a TS: **1. Trend** - *v*arying mean over time. For eg, in this case we saw that on a*v*erage, the number of passengers was growing over time. **2. Seasonality** - variations at specific time-frames. eg people might have a tendency to buy cars in a particular month because of pay increment or festivals.

The underlying principle is to model or estimate the trend and seasonality in the series and remove those from the series to get a stationary series. Then statistical forecasting techniques can be implemented on this series. The final step would be to convert the forecasted values into the original scale by applying trend and seasonality constraints back.

Note: I'll be discussing a number of methods. Some might work well in this case and others might not. But the idea is to get a hang of all the methods and not focus on just the problem at hand.

Let's start by working on the trend part.

Estimating & Eliminating Trend

One of the first tricks to reduce trend can be **transformation**. For example, in this case we can clearly see that the there is a significant positive trend. So we can apply transformation which penalize higher values more than smaller values. These can be taking a log, square root, cube root, etc. Lets take a **log transform** here for simplicity:

ts\_log = np.log(ts*)*

plt.plot(ts\_log)

**https*://*w*ww.*analyticsvidhya.com*/*blog/2016*/*0*2*/time-series-forecasting-codes-python*/***

**12*1*49**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

mwww www

1949

1951

1953

1955

1957

1959

(https:*//www.*analyticsvidhya.com/wp-content/uploads/2016/02/9.-ts-log.png)

In this simpler case, it is easy to see a forward trend in the data. But its not very intuitive in presence of noise. So we can use some techniques to estimate or model this trend and then remove it from the series. There can be many wa*y*s of doing it and some of most commonly used are:

**1. Aggregation** - taking average for a time period like monthly/weekly averages **2 Smoothing** - taking rolling averages

**3. Polynomial Fitting** - fit a regression model

I will discuss smoothing here and you should try other techniques as well which might work out for other problems. Smoothing refers to taking rolling estimates, i.e. considering the past few instances. There are can be various ways but I will discuss two of those here.

Moving average

In this approach, we take average of 'k' consecutive values depending on the frequency of time series. Here we can take the a*v*erage over the past 1 year, i.e. last 12 values. Pandas has specific functions defined for determining rolling statistics.

**moving\_a**vg = pd.rolling\_mean(ts\_log, 12) plt.plot(ts\_log) plt.plot(moving\_avg, color='red')

**https*://*w*ww.*analyticsvidhya.com*/*blog/2016*/*0*2*/time-series-forecasting-codes-python*/***

**13/*4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

50

MALAYALA*M*

1949

1951

1953

1955

1957

1959

1951

1953

1955 (https:*//*www.analyticsvidhya.com/wp-content/uploads/2016/02/10.-smooth-1.png)

The red line shows the rolling mean. Lets subtract this from the original series. Note that since we are taking average of last 12 values, rolling mean is not defined for first 11 values. This can be observed as:

ts\_log\_moving\_**avg\_diff = ts\_log - moving\_avg ts\_log\_moving\_avg\_diff.head(12)**

**Month** 1949-01-01

**NaN 1949-02-01**

**NaN** 1949-03-01

**NaN 1949-04-01**

**NaN 1949-05-01**

**NaN 1949-06-01**

**NaN 1949-07-01**

**NaN** 1949-08-01

**NaN** 1949-09-01

**NaN** 1949-10-01

**NaN** 1949-11-01

**NaN** 1949-12-01 -0.065494 **Name: *#*Passengers, dtype: float64**

(https://*ww*w.analyticsvidhya.com/wp

content/uploads/2016/02/10.5-missing-rolling.png)

Notice the first 11 being Nan. Lets drop these NaN values and check the plots to test stationarit*y.*

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**1*4/4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

ts\_log\_moving\_avg\_diff.dropna(inplace=True) **test\_st**ationarity(ts\_log\_moving\_avg\_diff)

Rolling Mean & Standard Deviation

Original Rolling Mean Rolling Std

-0.3 L

1950

1952

1954

1956

1958

1960

**Results of Dickey-Fuller Test: Test Statistic p-value *#L*ags Used Number of Observations Used Critical Value (5%) Critical Value (1%) Critical Value (10%) dtype: float64**

**-3.162908**

0.022235 13.000000 119.000000 **-2.886151 -3.486535** -2.**579896**

**الج لنا لا**

(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/2.-dfuller-smooth-1.png)

This looks like a much better series. The rolling values appear to be *v*arying slightly but there is no specific trend. Also, the test statistic is **smaller than the 5% critical values s**o we can sa*y* with 95% confidence that this is a stationary series.

Howe*v*er, a drawback in this particular approach is that the time-period has to be strictly defined. In this case we can take yearly averages but in complex situations like forecasting a stock price, its difficult to come up with a number. So we take a 'weighted moving average' where more recent values are given a higher weight. There can be many technique for assigning weights. A popular one **is exponentially weighted moving average w**here weights are assigned to all the previous values

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

***15/49***

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

with a decay factor. Find details here (http://pandas.pydata.org/pandas docs/stable/computation.html#exponentially-weighted-moment-functions). This can be implemented in Pandas as:

**expwighted\_a*v*g = pd.ewma(ts**\_log, halflife=12*)* plt.plot(ts\_log) plt.plot(expwighted\_avg, color='red')

ta

hammut

1949

1951

1953

1957

1959

(https:*//ww*w.analyticsvidhya.com/wp-content/uploads/2016/02/12.-smooth-2.png)

Note that here the parameter 'halflife' is used to define the amount of exponential decay. This is just an assumption here and would depend largely on the business domain. Other parameters like span and center of mass can also be used to define decay which are discussed in the link shared above. Now, let's remove this from series and check stationarity:

**ts\_log\_ewma\_diff = ts\_log - expwighted\_avg**

**test\_stationarity(ts\_log\_ewma**\_diff)

**https*://www.*analyticsvidhya.com*/*b*l*og*/*2016*/02/*time-series-forecasting-codes-python/**

**16/*4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

Rolling Mean & Standard Deviation

Original Rolling Mean Rolling Std

-0 1949

1951

1953

1955

1957

1959

Results of Dickey-Fuller Test: **Test Statistic** p-value ***#*La*g*s Used Number of Observations Used** Critical Value (5%) **Critical Value (1%)** Critical Value (10%) **dtype: float64**

-3.601262

0.005737 13.000000 130.000000 -2.884042 -3.481682 -2.578*77*0

(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/3.-dfuller-smooth-2.png)

This TS has even lesser variations in mean and standard deviation in magnitude. Also, the test statistic **is smaller than the 1% critical value,** which is better than the previous case. Note that in this case there will be no missing values as all values from starting are given weights. So it'll work even with no previous values.

Eliminating Trend and Seasonality

The simple trend reduction techniques discussed before don't work in all cases, particularly the ones with high seasonality. Lets discuss two ways of removing trend and seasonality:

**1. Differencing** - taking the differece with a particular time lag **2. Decomposition** - modeling both trend and seasonality and removing them from the model.

Differencing

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

*1*7/*4*9

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

One of the most common methods of dealing with both trend and seasonality is differencing. In this technique, we take the difference of the observation at a particular instant with that at the previous instant. This mostly works well in improving stationarity. First order differencing can be done in Pandas as

ts\_log\_diff = ts\_log - ts\_log.shift() plt.plot(ts\_log\_diff)

1951

1953

1955

1957

1959

1959

(https:*//*www.analyticsvidhya.com/wp-content/uploads/2016/02/14.-ts-diff.png)

This appears to have reduced trend considerably. Lets verify using our plots:

ts\_log\_diff.dropna(inplace=True)

**test\_stat**ionarity(ts\_log\_diff)

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**18/*4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

Rolling Mean & Standard Deviation

Original Rolling Mean Rolling Std

-03

1951

1953

1955

1957

**1959**

**Results of Dickey-Fuller Test: Test Statistic p-value *#*Lags Used Number of Observations Used Critical Value (5%) Critical Value (18) Critical Value (108) dtype: float64**

-2*.*717131

0.071121 **14.000000** 128.000000 **-2.884398 -3.482501** -2.5**78960**

(https:*//www.*analyticsvidhya.com/wp-content/uploads/2016/02/4.-dfuller-diff.png)

W*e* can see that the mean and std variations have small variations with time. Also, the Dickey-Fuller test statistic is **less than the 10% critical value**, thus the TS is stationary with 90% confidence. We can also take second or third order differences which might get even better results in certain applications. I leave it to you to try them out.

Decomposing In this approach, both trend and seasonality are modeled separately and the remaining part of the series is returned. I'll skip the statistics and come to the results:

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**19/*4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

**from statsmodels.tsa.seasonal import seasonal\_decompose decomposition = seasonal\_decompose(ts\_log)**

trend = decomposition.trend **seasonal = decomposition.seasonal**

**residual = decomposition.resid**

**plt. subplot(411)**

plt.plot(ts\_log, label='Original')

**plt.legend(loc='best')**

**pl**t.subplot(412)

plt.plot(trend, label='Trend') **plt.legend(loc='best')** plt.subplot(413) **plt.plot(seasonal, label='Season**ality') plt.legend(loc='best') **plt. subplot*(*414*)*** plt.plot(re**sidual, label='Residuals')**

**plt.legend(loc='best')**

plt. tight\_layout()

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

5.0

1949

1951

1953

1955

1957

1959

unninn NOBODNO

1950

1951

1952

1953

1954

1955

1956

1957

1958

1959

1960

0.3 02

0.1

0.0 -0.1 -0.2

1949

1951

1953

1955

1957

1959

0.10 0.05 0.00

-0.05*V*

-0.10 -0.15

mmmmmmm

1952 1953 1954 1955 1956 1957

1950

1950

1951

1951

1952

1953

1954

1956

1957

1959

1958

1959

1959

1960

(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/16.-decompose.png)

**https*://*w*ww.*analyticsvidhya.com*/*blog/2016*/*0*2*/time-series-forecasting-codes-python*/***

**20*/4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

Here we can see that the trend, seasonality are separated out from data and we can model the residuals. Lets check stationarity of residuals:

**ts\_log\_decompose = residual**

ts\_log\_decompose.dropna(inplace=True) **test\_stationarity(ts\_log\_decompose)**

Rolling Mean & Standard Deviation

0.05

-0.05

-0.10

Original Rolling Mean Rolling Std

—

-0.15

1950

1951

1952

1953

1954

1955

E

195

1956

1957

1958

1959

1950

**Results of Dickey-Fuller Test: Test Statistic**

**-6.332387e+00 p-value**

**2.885059e-08 *#*Lags Used**

**9.000000e+00 Number of Observations Used 1.220000e+02 Critical Value (5%)**

**-2.885538e+00 Critical Value (1%)**

-3.**485122e+00 Critical Value (10%)**

-2.5**79569e+00 dtype: float64**

**NO H NMN**

(https:*//*www.analyticsvidhya.com/wp-content/uploads/2016/02/5.-dfuller-decompose.png)

The Dickey-Fuller test statistic is significantl**y lower than the 1% critical value.** So this TS is very close to stationary. *Y*ou can try advanced decomposition techniques as well which can generate better results. Also, you should note that converting the residuals into original values for future data in not very intuitive in this case.

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**2*1/4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

5. Forecasting a Time Series

*W*e saw different techniques and all of them worked reasonably well for making the TS stationary. Lets make model on the TS after differencing as it is a very popular technique. Also, its relatively easier to add noise and seasonality back into predicted residuals in this case. Having performed the trend and seasonality estimation techniques, there can be two situations:

**1. A strictly stationary series w**ith no dependence among the values. This is the easy case wherein we

can model the residuals as white noise. But this is very rare. 2. A series with significant **dependence among values.** In this case we need to use some statistical

models like ARIMA to forecast the data.

Let me give you a brief introduction to ARIMA. I won't go into the technical details but you should understand these concepts in detail if you wish to apply them more effectively. ARIMA stands for **Auto-Regressive Integrated Moving Averages.** The ARIMA forecasting for a stationary time series is nothing but a linear (like a linear regression) equation. The predictors depend on the parameters (p.d,q) of the ARIMA model:

**1. Number of AR (Auto-Regressive) terms (p**): AR terms are just lags of dependent variable. For instance

if p is 5, the predictors for x(t) will be x(t-1)....x(t-5). **2 Number of MA (Moving A*v*erage) terms (q)**: MA terms are lagged forecast errors in prediction

equation. For instance if q is 5, the predictors for x(t) will be e(t-1)....e(t-5) where e(i) is the difference between the moving average at ith instant and actual value. **3. Number of Differences (d**): These are the number of nonseasonal differences, i.e. in this case we took

the first order difference. So either we can pass that variable and put d=o or pass the original variable and put d=1. Both will generate same results.

An importance concern here is how to determine the value of 'p' and 'q! We use two plots to determine these numbers. Lets discuss them first.

**1. Autocorrelation Function (ACF**): It is a measure of the correlation between the the TS with a lagged

version of itself. For instance at lag 5. ACF would compare series at time instant 't1.m'tz' with series at

instant 't1-5...'t2-5' (t1-5 and t2 being end points). **2. Partial Autocorrelation Function (PACF**): This measures the correlation between the TS with a lagged

version of itself but after eliminating the variations already explained by the intervening comparisons. Eg at lag 5, it will check the correlation but remove the effects already explained by lags 1 to 4.

The ACF and PACF plots for the TS after differencing can be plotted as:

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

***2214*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

**#ACF a**nd PACF plots: **from statsmodels.tsa.stattools** import acf, pacf

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

**lag\_acf** = acf(ts\_log\_diff, n**lags=20) lag\_pacf = pa**cf(ts\_log\_diff, nlags=20, method='ols')

#Plot ACF: plt.subplot(121) plt.plot(lag\_acf)

**plt.axhline (y=0, linestyle**='--',color='gray')

**plt.axhline** (y=-1.96*/*np. sqrt(len(ts\_log\_diff)), linestyle='--',color='gray')

**plt.ax**hline (y=1.96*/*np.sqrt(len(ts\_log\_diff)),linestyle='--', color='gray') plt.title*(*'Autocorrelation Function')

#Plot PACF:

plt.subplot(122) plt.plot(lag\_pact) plt.axhline (y=0, linestyle='--',color='gray') plt.axhline(y=-1.96*/*np.sqrt(len(ts\_log\_diff)), linestyle='--',color="gray') plt.axhline(y=1*.96/*np.sqrt(len(ts\_log\_diff)), linestyle='--',color='gray') plt.title('Partial Autocorrelation Function') plt.tight\_layout()

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**2*3/*49**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

Autocorrelation Function

Partial Autocorrelation Function

=

07

*19*

20

ut

(https:/*/w*ww.analyticsvidhya.com/wp-content/uploads/2016/02/6.-acf-pcf-final.png)

In this plot, the two dotted lines on either sides of o are the confidence interevals. These can be used to determine the 'p' and 'q' values as:

**1. p** -The lag value where the PACF chart crosses the upper confidence interval for the first time. If you

notice closely, in this case p=2. 2q - The lag value where the ACF chart crosses the upper confidence interval for the first time. If you

notice closely, in this case q=2.

Now, lets make 3 different ARIMA models considering individual as well as combined effects. I will also print the RSS for each. Please note that here RSS is for the values of residuals and not actual series.

We need to load the ARIMA model first:

from **statsmodels.tsa.arima\_model import ARIMA**

The p,d,q values can be specified using the order argument of ARIMA which take a tuple (p.d,q). Let model the 3 cases:

AR Model

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

***24/*49**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

model = ARIMA(ts\_log, order=(2, 1, 0*)*) results\_AR = model.fit(disp=-1)

plt.plot(ts\_log\_diff)

plt.plot(re**sults\_AR. fittedvalues, color**='red') plt.title('RSS: %.4f'% sum((results\_AR.fittedvalues-ts\_log\_diff)\*\*2*)*)

RSS: 1.5023

-0.3 L

1951

1953

1955

1957

1959

(https:*//w*ww.analyticsvidhya.com/wp-content/uploads/2016/02/18.-model-AR.png)

**MA Model**

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

**model = ARIM**A(ts\_log, order=(0, 1, 2)) **resu**lts\_MA = model.fit(disp=-1) plt.plot(ts\_log\_diff) plt.plot(results\_MA. f**ittedvalues, color**='red')

plt.title('RSS: %.4f'% sum((results\_MA. fit**tedvalues-t**s\_log\_diff)\*\**2*))

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**25*/4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

RSS: 1.4721

-01

1951

1953

1957

1959

(https:*//*www.analyticsvidhya.com/wp-content/uploads/2016/02/19.-model-MA.png)

Combined Model

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

**model = ARIMA(ts**\_log, order=*(2*, 1, 2)) results\_ARIMA = model.fit(disp=-1)

plt.plot(ts\_log\_diff) plt.plot(results\_ARIMA. fi**ttedvalues,** color='red') plt.title('RSS: %.4f'% sum((results\_ARIMA. fittedvalues-ts\_log\_diff]\*\**2*))

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**26/*4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

RSS: 1.0292

-03

1951

1953

1955

1957

1959

(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/20.-model-both.png)

Here we can see that the AR and MA models have almost the same RSS but combined is significantly better. Now*, we* are left with 1 last step, i.e. taking these values back to the original scale.

Taking it back to original scale

Since the combined model gave best result, lets scale it back to the original values and see how well it performs there. First step would be to store the predicted results as a separate series and observe

predictions\_ARIMA\_diff = pd.Series(results\_ARIMA. fittedvalues, copy=True) print predictions\_ARIMA\_diff.head()

I

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

**Month**

**1949-02-01 0.009580 1949-03-01 0.0*1*7491 1949-04-01 0.02*7*670 1949-05-01 -0.004521 1949-06-01** -0.023889 **dtype: float64**

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

***27/49***

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/21.-check-output.png)

Notice that these start from '1949-02-01' and not the first month. Why? This is because we took a lag by 1 and first element doesn't have anything before it to subtract from. The way to convert the

differencing to log scale is to add these differences consecutively to the base number. An eas*y way t*o do it is to first determine the cumulative sum at index and then add it to the base number. The cumulative sum can be found as:

predictions\_ARIMA\_diff\_cumsum = predictions\_ARIMA\_diff.cumsum() **print predictions\_ARIMA\_d**iff\_cumsum.head*(*)

**Month**

**1949-**02-01 0.009580 **1949-03-01 0.027071 1949-0*4*-01 0.054742 1949-05-01 0.050221 1949-06-01 0.026331 dtype: float64**

(https:*//ww*w.analyticsvidhya.com/wp-content/uploads/2016/02/22.-cumsum.png)

You can quickly do some back of mind calculations using previous output to check if these are correct. Next w*e'v*e to add them to base number. For this lets create a series with all values as base number and add the differences to it. This can be done as:

**predictions\_ARIMA\_log = pd.Series(ts\_log.ix[0], index=ts\_log.index)**

predictions\_ARIMA\_log = predictions\_ARIMA\_log.add(predictions\_ARIMA\_diff\_cumsum, **fill\_value=0)**

**predicti**ons\_ARIMA\_log.head()

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**28*/4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

**Month**

**1949-01-01 4.718499 1949-02-01 *4*.728079 1949-03-01 4.7*4*5570 1949-04-01 4*.773*241 1949-05-01 4.768720 dtype: float64**

(https://www.analyticsvidhya.com/wp-content/uploads/2016/02/23.-add-cumsum.png)

Here the first element is base number itself and from thereon the values cumulatively added. Last step is to take the exponent and compare with the original series.

predictions\_ARIMA = np.exp(predictions\_ARIMA\_log)

plt.plot(ts)

plt.plot(predictions\_ARIMA) plt.title('RMSE: %.4f'% np.sqrt(sum((predictions\_ARIMA-ts)\*\**2)/*len(ts)))

RMSE: 90.1047

1951

1953

1955

1957

1959

(https:*//w*ww.analyticsvidhya.com/wp-content/uploads/2016/02/24.-final-plot.png)

Finally we ha*v*e a forecast at the original scale. Not a very good forecast I would say but you got the idea right? Now, I leave it upto you to refine the methodology further and make a better solution.

**https*://*w*ww.*analyticsvidhya.com*/*blog/2016*/*0*2*/time-series-forecasting-codes-python*/***

**29*/4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

End Notes

Through this article I have tried to give you a standard approach for solving time series problem. This couldn't have come at a better time as today is our Mini DataHack (http://datahack.analyticsvidhya.com/contest/mini-datahack) which will challenge you to solve a similar problem. *We'v*e covered concepts of stationarity, how to take a time series closer to stationarity and finally forecasting the residuals. It was a long journey and I skipped some statistical details which I encourage you to refer using the suggested material. If you don't want to copy-paste, you can download the iPython notebook with all the codes from my GitHub (https://github.com/aarshayj/Analytics\_Vidhya/tree/master/Articles) repository.

I hope this article will help you achieve a good first solution today. All the best guys!

Did you like the article? How helpful was it in the hackathon today? Somethings bothering you which you wish to discuss further? Please feel free to post a comment and I'll be more than happy to discuss.

**Share this:**

in (https*://w*ww.analyticsvidhya.com/blog/2016/02*/*time-series-forecasting-codes-python/?share=linkedin&nb=1)

481

f (https*://www*.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/?share=facebook&nb=1)

G+ (https*://www*.analyticsvidhya.com/blog/2016/02*/*time-series-forecasting-codes-python/?share=google-plus-1&nb=1)

(https*://ww*w.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/?share=twitter&nb=1)

♡ (https*://*www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/?share=pocket&nb=1)

(https*://www.*analyticsvidhya.com/blog*/*2016/02/time-series-forecasting-codes-python/?share=reddit&nb=1)

**RELATED**

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**30*/*49**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

Bishwarun

Python

*V*opani

Python

Analytics Vidhya Vopani Data Hack

Mark Lady Data Hack Hackathon Python Last ManStandingswarup

Analytics VidhyaPython \*Bishwarum Mark Landry lackathon

VopaniPython Mark Lord Hackathon

Bishwarup Data Hack Packathon

LILY Bishwarup

irHackathon

Hackathon

TV

Vidya

ast Man Standing

Analytics Vida

(https*://*www.analyticsvidhya.com/ blog/2016/06/winners-mini datahack-time-series-approach codes-solutions/) Winners of Mini Data Hack (Time Series) - Approach, Codes and Solutions (https*://*www.analyticsvidhya.com/ blog/2016/06/winners-mini datahack-time-series-approach codes-solutions*/*) In "Business Analytics"

(https*://*www.analyticsvidhya.com*/* blog/2016/02/secrets-winners signature-hackathon-last-man standing/) Mini DataHack and the tactics of the three "Last Man Standing"! (https*://*www.analyticsvidhya.com/ blog/2016/02/secrets-winners signature-hackathon-last-man standing/) In "Business Analytics"

(https*://*www.analyticsvidhya.com*/* blog/2015/02/exploration-time series-data-r/) Exploration of Time Series Data in R (https*://w*ww.analyticsvidhya.com*/* blog*/2*015*/*02*/*exploration-time series-data-r*/*) In "Business Analytics"

**TAGS: A**RIMA (HTTPS*://*WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/ARIMA*/*), FORECASTING ANALYTICS (HTTPS*://W*WW*.A*NALYTICSVIDHYA.COM*/*BLOG/TAG*/*FORECASTING-ANALYTICS/), PANDAS (HTTPS*://*WWW*.A*NALYTICSVIDHYA.COM/BLOG/TAG*/*PANDAS/), TIME **SERI**ES (HTTPS*://*WWW.ANALYTICSVIDHYA.COM/BLOG/TAG/TIME-SERIES/), **TIME SERIES ANALYS**IS (HTTPS*://*WWW*.*ANALYTICSVIDHYA.COM/BLOG/TAG/**TIME SERIES-ANALYS**IS/), TIME SERIES FORECASTING (HTTPS*://****W*WW.ANA**LYTICSVIDHYA.COM*/*BLOG/T*A*G*/*TIME-SERIES-FORECASTING*/*)

Previous Article

Next Article Mini DataHack and the tactics of the **What I learnt about Time Series Analysis**

**three "Last Man** Standing”!

in 3 hour Mini Dat**aHack? (https*://www.*analyticsvidhya**.com*/*blog/2016*/0*2/**secretfhttps:*//*www.analyticsvidhya**.com/blog*/20*16*/02/*hand

**winners-signature-hackathon-last-man**

**learn-time-series-3-hours-mini-datahack*/*) standing)**

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

***31/*49**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

(https:*//*www*.*analyticsvidhya.com/blog/author/aarshay*/)*

Author Aarshay Jain (https*://*w*w*w.analyticsvidhya.com/blog/author/aarshay*/*) Aarshay is a ML enthusiast, pursuing MS in Data Science at Columbia University, graduating in Dec 2017. He is currently exploring the various ML techniques and writes articles for AV to share his

knowledge with the community.

(mailto:aarshayjain@gmail.com) in (https://in.linkedin.com/in/aarshayjain)

og (https://github.com/aarshayj) S (aarshay)

38 COMMENTS

**Dr.D.K.Samue248S*://*WWW*.*ANA**LYTICSVIDHYA.COM*/*BLOG/2016/02*/*TIME-SERIES-FORECASTING CODES-PYTHON/?REPLYTOCOM-105271#RESPOND)

CICI I

TITLE

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

***3*2*4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

**FEBRUARY 6, 2**016 AT 10:02 AM (HTTPS*://*WWW*.*ANALYTICSVIDHYA.COM/BLOG/2016/02/TIME-SERIES-FORECASTING-CODES PYTHON/#COMMENT-105271)

Real thanks for a post which met my need

**Aarshay JaktpeAVISTPS*://W*WW.ANALYTICSVIDHYA.COM*/*BLOG*/*2016/02*/*TIME-SERIES-FORECASTING CODES-PYTHON/?REPLYTOCOM-105272#RESPOND)** FEBRUARY 6, 2016 AT 10:04 AM (HTTPS*://*WWW.ANALYTICSVIDHYA.COM/BLOG*/*2016/02/TIME-SERIES-FORECASTING-CODES

PYTHON/#CO**MMENT-1052*7*2)**

I'm glad you liked it

**satish say SREPLY (HTTPS*://W*WW.ANALYTICSVIDHYA.COM*/*BLOG*/*2016*/*02*/*TIME-SERIES-FORECASTING CODES-PYTHON*/*?REPLYTOCOM-105327#RESPOND)** FEBRUARY 7, 2016 AT 3:17 PM (HTTPS*://*WWW*.*AN*A*LYTICSVIDHYA.COM*/*BLOG*/*20*1*6*/*02*/*TIME-SERIES-FORECASTING-CODES PYTHON/#COMMENT-10532*7*)

Thanks for great explanation related to timeseries. What is difference between holtwinters and arima forcast?

**shan says: REPLY (HTTPS:*//*WWW.ANA**LYTICSVIDHYA.COM*/*BLOG*/*2016*/*02*/*TIM**E-SERIES-FORECASTIN**G CODES-PYTHON/?REPLYTOCOM-105357#RESPOND) FEBRUARY 8, 2016 AT 6:09 AM (HTTPS*://*WWW.ANALYTICSVIDHYA.COM/BLOG*/*2016/02*/*TIME-SERIES-FORECASTING-CODES PYTHON/#C**OMMENT**-105357)

Holtwinters is double exponential smoothening method. ARIMA, forecasts by identifying p,d,q component of a series. Hope it helps.

**Aarshay *J*akubsaMAPS*://*WWW.ANA**LYTICSVIDHYA.COM/BLOG/2016/0*2/*TIM**E-SERIES FO**RECASTING-CODES-PYTHO**N/?REPLYTOCOM-105358#RESPOND)** FEBRUARY 8, 2016 AT 6:51 AM (HTTPS*://*WWW*.*ANALYTICSVIDHYA.COM/BLOG/2016/02*/*TIME-SERIES-FORECASTING-CODES PYTHON/#COMMENT-105358)

To add to Shan, Holtwinters uses a weighted average of past values while ARIMA uses both past values and past errors. You can find more details here: https:*/*/www*.*google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&*v*ed=0ahUKEwiUgKXCX fKAhXlc44KHTTPDYUQFggjMAE&url=http%3A%2F%2Fw*ww*.ons.gov.uk%2Fons%2Fguide method%2Fukcemga%2Fukcemga-publications%2Fpublications%2Farchive%2Ffrom-holt-winters to-arima-modelling-measuring-the-impact-on-forecasting-errors-for-components-of-quarterly estimates-of-public-service-output.pdf&usg=AFQjCNGmYzfVB -gdss4LKTGW4WZgBC\_w&sig2=9pnseABIC\_40XC2KnWmHNw&cad=rja

**https*://*w*ww.*analyticsvidhya.com*/*blog/2016*/*0*2*/time-series-forecasting-codes-python*/***

**33/49**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

(https:*//*www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0ahUKEwiUgKXCX fKAhXIC44KHTTPDYUQFggjMAE&url=http%3A%2F%2Fww*w.*ons.gov.uk%2Fons%2Fguide method%2Fukcemga%2Fukcemga-publications%2Fpublications%2Farchive%2Ffrom-holt-winters to-arima-modelling--measuring-the-impact-on-forecasting-errors-for-components-of-quarterly estimates-of-public-service-output.pdf&usg=AFQjCNGmYzfVB -gdss4LKTGW4W*V*ZgBC\_w&sig2=SpnseABIC\_40XC2KnWmHNw&cad=rja)

**Il Says: RE**PLY (HTTPS*://*WWW.ANALYTICSVIDHYA.COM/BLOG/2016*/*02/TIME-SERIES-FORECASTING-CODES-PYTHON/?REPLYTOCOM-109273#RESPOND) APRIL 11, 2016 AT 12:59 AM (HTTPS*://*WWW.ANALYTICSVIDHYA.COM/BLOG*/*2016/02/TIME-SERIES-FORECASTING-CODES PYTHON/#COMMENT-109273)

Holt winters (at least the additive model) is a special case of arima model (a seasonal arima model). That would be an arimalp,d,q)P,D,Q) where the second parentheses contains the seasonal

effects. I would additionally recommend checking out any of Rob Hyndman's work on arima modeling, I find it to be very accessible.

**Shan SayS-R**EPLY (HTTPS*://*WWW.ANALYTICSVIDHYA.COM/BLOG/2016*/*02/TIME-SERIES FORECASTING-CODES-PYTHON/?REPLYTOCOM-105332#RESPOND) FEBRUARY 7, 2016 AT 6:12 PM (HTTPS*://W*WW*.*ANALYTICSVIDHYA.COM/BLOG/2016/02/TIME-SERIES-FORECASTING-CODES PYTHON*/*#COMMENT-105332)

Hi..

Thanks. for an informative article. I am eager to know on followings :

a) how can we identify what should be nlags value to test with lag\_acf = acf(ts\_log\_diff, nlags=20) lag\_pacf = pacf(ts\_log\_diff, nlags=20, method='ols')

b) How can we forecast for future time points (say 12 time points ahead). Can we use followings still ? predictions\_ARIMA\_log = pd.Series(ts\_log.ix[o], index=ts\_log.index) predictions\_ARIMA\_log = predictions\_ARIMA\_log.add(predictions\_ARIMA\_diff\_cumsum, fill\_value=0) ts\_log is not available for future points.

c) In one of the article (A Complete Tutorial on Time Series Modeling in R.) referred by you,

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

***34/*49**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

while performing adf says adf.test(diff(log(AirPassengers)), alternative="stationary", k=0) What is k, and how can we identify the value of k while performing the test..

7 Analytics Vidhya

sunt

, 1, 1),seasonal = list(order = c(0, 1, 1), period = 12)))

AVCasino

**INTRODUCTION**

CF PACF plots. al = list(order = c(0, 1, 1) easonal parameter and how to identify it.

**PROB*A*BILITY**

bove.. Thanks in anticipation.

**2016, Aug 10th - 31st**

**ONLINE**

**REGISTER NOW**

**HIYA*.*COM*/*BLOG/2016*/*02/TIME-SERIES-FORECASTING-COD**ES-PYTH**ON/?REPLYTOCOM-105359#RESPOND)** http*s:/*/datbronkyanøystiasuid**hwaApps*.*applere*n***tCSVIDHYA.COM/BLOG/2016*/*0*2/*TIME-SERIES-FORECASTING-CODES **Casino-intppchentipcon Erradorab**ility

Hi Shan,

Thanks for reaching out. Please find my responses below:

a) So the 'nlags' doesn't affect the output values. I just specifies how many values to display. So you can start with a small number and if you don't find the crossing point within that, you can increase maximum upto the number of observations in data.

b) ARIMA has a specific function for forecasting values. The 'results\_ARIMA' variable here is of the type 'ARIMAresults' which has a 'predict' function. You can check the details as - http://statsmodels.sourceforge.net/de*v*el/generated/statsmodels.tsa.arima\_model.ARMAResults. (http://statsmodels.sourceforge.net/devel/generated/statsmodels.tsa.arima\_model.ARMAResults **Please feel free t**o get back to me in case you face challenges in implementing this. You can also start a thread in the discussion forum which will allow more freedom of expression while discussing

c) I'm not much experienced with R so let me read the code syntax. I'll get back to you on this.

Cheers!

**LLL**

**Aarshay Jaktpe ASPS***://*WWW*.*ANALYTICSVIDHYA.COM*/*BLOG/2016*/*02/TIME-SERIES-FORECASTING CODES-PYTHON/?REPLYTOCOM-105373#RESPOND)

TIL II

**L**

**4**

**https*://*w*ww.*analyticsvidhya.com*/*blog/2016*/*0*2*/time-series-forecasting-codes-python*/***

***3*5*/49***

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

FEBRUARY 8, 2016 AT 12:32 PM (HTTPS*://*WWW*.*AN*A*LYTICSVIDHYA.COM*/*BLOG/2*01*6/02*/*TIME-SERIES-FORECASTING-CODES PYTHON/#COMMENT-105373)

Hi Shan,

7 Analytics Vidhya

er to better te discussion thread for your query *'*c': Lets continue the

e reading this and interested in exploring further, please check

AVCasino

**INTRODUCTION**

n/t/seasonal-parameter-in-arima-and-adf-test/7385/1 m/t/seasonal-parameter-in-arima-and-adf-test/7385/1)

**PROB*A*BILITY**

**2016, Aug 10th - 31st**

**anceddataanalytics.net*/***2016*/*02/08/distill**ed-news-316*/*) says: ONLINE**

**REGISTER NOW**

IDHYA.COM/BLOG*/*2016/02*/*TIME-SERIES-FORECASTING-CODES-PYTHON/#C**OMMENT 15$$$$*/*/**datahack.analyticsvidhya.com/contest/av CassiaeniteedyetinentDoc relateazititine Series Forecast (with Codes in Python) Time Series (referred as TS from now) is considered to be one of the less known skills in the analytics space (Even I had little clue about it a couple of days back). But as you know our inaugural Mini Hackathon is based on it, I set myself on a journey to learn the basic steps for solving a Time Series problem and here I am sharing the same with you. These will definitely help you get a decent model in our hackathon today. [...]

TL

**What I learnt about Time Series Analysis in 3 hour Mini Data Hack? (http:*/*/www.analyticsvidhya.com/blog/2016*/*02*/*hand-learn-time-series-3-hours-mini-datahack/) says:** FEBRUARY 8, 2016 AT 8:00 PM (HTTPS*://*WWW*.A*NALYTICSVIDHYA.COM/BLOG*/2*016*/*0*2/*TIME-SERIES-FORECASTING-CODES-PYTH**ON/#COMMENT**

**105386)** ... Since it was already decided that the problem would be about Time Series - I made sure I was well equipped with knowledge and packages about Time Series, Infact, I even wrote a guide to Time Series in Python. [...]

**Complete guide to create a Time Series Forecast (http*://*big-data-fr.com*/*complete-guide-to-create-a-time-series forecast-with-codes-in-python/) says:** FEBRUARY 13, 2016 AT 1:14 AM (HTTPS*://W*WW*.*ANALYTICSVIDHYA.COM/BLOG/2016/02*/*TI**ME-SERIES**-FORECASTING-CODES-PYTHON/#C**OMMENT** 105598) ... Read tutorial By Aarshay Jain Source: analyticsvidhya.com [...]

**amitsethia AY HTTPS*://*WWW***.*ANALYTICSVIDHYA.COM/BLOG*/*2016/0*2/*TIME-SERIES-FORECASTING-CODES-PYTHON/?REPLY**TOCOM-105619#RESPOND)** FEBRUARY 13, 2016 AT 12:23 PM (HTTPS*://*WWW*.A*NALYTICSVIDHYA.COM/BLOG/2016*/0*2*/*TIME-SERIES-FORECASTING-CODES PYTHO**N/#COMMENT**-105619)

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**36*/*49**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

Thanks Aarshay for this write up. It is also recommended to not to go for combined models as p & qused together will nullify their impact on the model, hence, it is either a moving average or auto correlation along with differences, but here combined model has given the best results. Can you please correct my understanding around combined models.

7 Analytics Vidhya

AVCasino

IYA.COM*/*BLOG/2016*/*02*/***TIME-SERIES-FORECASTING-CODE**S-PYTHON/?REPLYTOC**OM-105623#RESPOND)** VW*.*ANALYTICSVIDHYA.COM/BLOG*/*2016*/*02*/*TIME-SERIES-FORECASTING-CODES

**INTRODUCTION**

**PROBABILITY**

pt be combined. It's actually appears counter intuitive because if puld not exist in the first place. Can you throw some light on why ut the effect of one another?

**2016 , Aug 10th - 31st**

**ONLINE**

REGISTER NOW

(https:*//*datahackanalyticsvidhya.com/contest/av

***QUA*NALYTICSVIDHYA.COM*/*BLOG/2016*/*02*/*TIME-SERIES-FORECASTING CODES-PYTHON*/*?REPLYTOCOM-106400#RESPOND)** casino-introduction =probability

FEBRUARY 29, 2016 AT 10:58 PM (HTTPS*://W*WW.ANALYTICSVIDHYA.COM/BLOG*/*2016/02*/*TIME-SERIES-FORECASTING-CODES PYTHON/#COMMENT-106400)

Hi! The article is the best available on Time Series with Python with great external links too for those who want to understand the stat behind also. I would like to request to please extend this article to predict out-of-sample data range also with different models to depict the better ones as you did for eliminating trend (taking rolling average and ewma).

That will make it all fully fledged time-series article. Thanks in advance.

**Aarshay Jatibar "S*://W*WW.ANA**LYTICSVIDHYA.COM/BLOG*/*2016*/*02*/*TIM**E-SERIES-FORECASTING-CODES-PYTHON*/*?REPLYTOCOM-106425#RESPOND) MAR**CH 1, 2016 AT 8:46 AM (HTTPS*://*WWW*.A*NALYTICSVIDHYA.COM/BLOG*/*2016/02*/*TIME-SERIES-FORECASTING-CODES PYTHON/#COMMENT-106425)

Hi Ayush! Thanks for your valuable feedback. Yes I think that component is necessary. But instead of extending this article, I'll probably write a separate post taking another case study. I'm a bit crunched for bandwidth but you can expect it sometime in this month. Stay tuned!

**Michael says:**

**https*://*w*ww.*analyticsvidhya.com*/*blog/2016*/*0*2*/time-series-forecasting-codes-python*/***

***3*7*/4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

**MARCH 13, 20GPAT (HIMBSPMMWMANON*Y*MCWWDNWACOWBIJEDHD/&/ODALMISTAKESFORECASTINGLOBRSPSHORPORSLATAEDADES267#RESPOND)** PYTHON/#COMMENT-10*7*26*7*)

Thanks for the excellent article. I have 2 clarifications **Din the Estimatin**g & Eliminatina. Trend step, I have negative numbers. Could you please tell me

7 Analytics Vidhya" V orbereitet med dett\_ y. Log and Sart returns NAN?

ecompose,nlags-10) while executing specifies nlags not defined.

AVCasino

**INTRODUCTION**

**PROB*A*BILITY**

**HYA.COM/BLOG*/*2016*/*02*/*TIME-SERIES-FORECASTING-CODES-PYTHON*/*?REPLYTOCOM-107270#RESPOND)**

NALYTICSVIDHYA.COM/BLOG*/*2016*/*02*/*TIME-SERIES-FORECASTING-CODES

**2016, Aug 10th - 31st**

**ONLINE**

**REGISTER NOW**

(https:*//*datahack.analyticsvidhya.com/contest/av casino-introierstofor**treacokab**idity / Regarding your queries:

1. You can try scaling up your values and then applying transformations. Also, you might want to check if log transformation is actually required in your case. You can try a cube root as well. 2. Please remove the nlags argument and then run the code. I've updated the code above as well.

**will welch $AYBXHTTPS:*//*WWW.ANAL**YTICSVIDHYA.COM/BLOG*/*2016*/02/*TI**ME-SERIES**-FORECASTING-CODES-PYTHON*/*?REPLYTOCOM-107299#RESPOND) MARCH 14, 2016 AT 1:34 AM (HTTPS*://*WWW.ANALYTICSVIDHYA.COM/BLOG*/*2016*/02/*TIME-SERIES-FORECASTING-CODES PYTHON/#COMMENT-107299)

Nice article, you rarely see this range of models discussed in one place and in such a hands-on wa*y.*

For anyone doing seasonal decomposition in Python, I'd like to shamelessly plug my seasonal package (PyPl or https://github.com/welch/seasonal (https://github.com/welch/seasonal)) in addition to statsmodels seasonal\_decompose. `seasonal offers some richer and more robust detrending possibilities, and will also estimate your model's periodicity for you (convenient in a dev-ops setting with thousands of streams at hand). It also includes a robust periodogram for visualizing the periodicities in your data.

**https*://www.*analyticsvidhya.com*/*b*l*og*/*2016*/02/*time-series-forecasting-codes-python/**

**38/*4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

**Aarshay Jain VSTPS*://*WW*W.*ANAL**YTICSVIDHYA.COM/BLOG/2016*/02/*T**IME-SERIES FORECASTING-COD**ES-PYTHON*/*?REPLYTOCOM-107313#RESPOND) **MARC**H 14, 2016 AT 5:58 AM (HTTPS*://W*WW*.*ANALYTICSVIDHYA.COM/BLOG*/*2016/02*/*TIME-SERIES-FORECASTING-CODES

PYTHON/#COMMENT-107313)

**Thanks Will for sharing *v*our** library. It'll be helpful for e*v*er*y*one.

7 Analytics Vidhya

AVCasino

36\*## (http*://*w**ww.36dsj.com*/*archives*/*44065) says:** HYA.COM*/*BLOG*/*2016*/*02/TIME-SERIES-FORECASTING-CODES-PYTHON/#COMMENT

**INTRODUCTION**

Series Forecast (with Codes in Python) [...]

**PROB*A*BILITY**

HYA.COM*/*BLOG/2016/02/TIME-SERIES-FORECASTING CODES-PYTHON/?REPLYTOCOM-108176#RESPOND) ANALYTICSVIDHYA.COM*/*BLOG/20*1*6/02/TIME-SERIES-FORECASTING-CODES

**2016, Aug 10th - 31st**

**ONLINE**

**REGISTER NOW**

le dftestlo:4)? (https:*//*datahack.analyticsvidhya.com/contest/av casino-introduction-to-probability/

**Jan Peta P**PS:/*/*WWW*.*ANALYTICSVIDHYA.COM*/*BLOG/2016/*02/*TIME-**SERIES FORECASTING-C**ODES-PYTHON/?REPLYTOCO**M-108224#RES**POND) MARCH 25, 2016 AT 6:26 AM (HTTPS*://W*WW*.*ANALYTICSVIDHYA.COM/BLOG/2016/02/TIME-SERIES-FORECASTING-CODES PYTHON/#COMMENT-10**8224)**

the adfuller function returns a list with many values. I'm picking the first 4 using (0:4). I've used the 5th value separately. You might want to print the dftest variable and you'll know*.*

**Lbert saySREPLY (HTTPS*://WW*W*.*ANALYTICSVIDHYA.COM/BLOG*/*2016*/*02*/*TIME-SERIES-FORECASTING-CODES-PYTHON/?REPLYTOCOM-109243#RESPOND)**

| (HTTPS*://****WW*W.ANALYTICSV**IDHYA.COM/BLOG/2016*/*02*/*TIME-SERIES-F**ORECASTING-CODES** PYTHON/#COMMENT-109243)

Can we use this method for decimal data? Why the program gave me an error of “ValueError: You must specify a freq or x must be a pandas object with a timeseries index"?

**Aars**

**PENALTPS*://*WWW.**ANALYTICSVIDHYA.COM/BLOG/2016*/*02*/*TIME-SERIES-FORECASTING-CODES-PYTHON/?REPLYTOCOM-10**9280#RES**POND) APRIL 11, 2016 AT 6:59 AM (HTTPS*://*WWW.ANALYTICSVIDHYA.COM*/*BLOG/2016/02/TIME-SERIES-FORECASTING-CODES PYTHON/#COMMENT-109280)

I don't think it is a decimal error. Please check whether your index is a timeseries object.

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**39*/4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

**Cloga (httREPLIWW'sologa.info isaVSIYA.COM*/*BLOG/201*6/*0*2/*TIME-SERIES-FORECASTING-CODES-PYTHON*/*?REPLYTOCOM-109445#RESPOND)** APRIL 14, 2016 AT 9:54 AM (HTTPS*://W*WW.ANALYTICSVIDHYA.COM*/*BLOG*/*2016/0*2/*TIME-SERIES-FORECASTING-CODES PYTHON**/#COMMENT-109445)**

Hi Aarsbav lain

7 Analytics Vidhya

AVCasino

**INTRODUCTION**

og as sampel, ie, model = ARIMA(ts\_log, order=(2, 1, 2)), but when

as diff value: predictions\_ARIMA\_diff = ues, copy=True), is results\_ARIMA.fittedvalues return log value or

**PROBABILITY**

**2016, Aug 10th - 31st**

**ONLINE**

**REGISTER NOW https://darahatak dapat PALAWANANALYTICSKIAHLA CONUBLOG*/*2016/02/TIME-SERIES-FORECASTING-C**ODES-PYTHON/?REPLYTOCOM-109454#RESPOND)

intAPRIL 14, 2016 AT 10:53 AM:HTTPS*://*WW*W.*ANALYTICSVIDHYA.COM*/*BLOG*/*2016*/*02*/*TIME-SERIES-FORECASTING-CODES

PYTHON/#COMMENT-109454)

I

UNITI

actually while calling ARIMA I have set order = (2,1,2). Here the middle argument 1 means that ARIMA will automatically take a difference of 1 while making predictions.

**REPONS K ANALAISEWEHYA.C**OM*/*BLOG/2016/02/TIME-SERIES-FORECASTING-CODES-PYTHON/?REPLYTOCOM=109489#RESPOND) APRIL 15, 2016 AT 2:31 AM (HTTPS*://*WWW.ANALYTICSVIDHYA.COM/BLOG/2016*/*02*/*TIME-SERIES-FORECASTING-CODES PYTHON/#COMMENT-109489)

Got it thank you!

**Ayodeji VINAPHAYOT*I*MDAGANAY ATSVIDHYA.COM/BLOG*/*2016/02*/*TIME-SERIES-FORECASTING-CODES-PYTHON*/*?REPLYTOCOM-109660#RESPOND)** APRIL 19, 2016 AT 12:**44 AM** (HTTPS*://W*WW.ANALYTICSVIDHYA.COM/BLOG/2016*/*02*/*TIME-SERIES-FORECASTING-CODES PYTHON/#COMMENT-109660)

What an excellent article on Time Series, more grease to your elbow. But the question is, is this method a package of analyzing Time Series related data or what? And can't we do the same on SPSS and have the same simple method as this? How*ev*er, I have to commend you a lot for this wonderful presentation. God will continue to increase your knowledge.

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**40*/4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

**Aarshay Jaji sayfus*://WWW.*ANALYTICSVIDHYA.COM*/*BLOG*/*201*6/*0*2/*TIME-SERIES-FORECASTING-CODES-PYTHON*/*?REPLYTOCOM-109664#RESPOND)** APRIL 19, 2016 AT 5:09 AM (HTTPS*://W*WW*.*ANALYTICSVIDHYA.COM/BLOG*/*2016*/*02*/*TIME-SERIES-FORECASTING-CODES PYTHON/#**COMMEN**T-109664)

**Thanks A*v*odail**

7 Analytics Vidhya

**dute**, I didn't get your questions - what do you mean "is this method

es related data"? Please elaborate.

**es r**elated data"? Please elaborate.

**AV C asino**

INTRODUCTION

IYA.**COM*/*BLOG*/*2016*/*02/TIME-SERIES-FORECASTING-C**ODES-PYTHON**/?REPLYTOCOM-109830#RESPOND)** NALYTICSVIDHYA.COM*/*BLOG*/*2016*/*02*/*TIME-SERIES-FORECASTING-CODES

**PROB*A*BILITY**

**2016, Aug 10th - 31st**

**ONLINE**

**REGISTER NOW**

de with different dataset and then an error shown up like this: (https:*//*datahack.analyticsvidhya.com/contest/av casino-intrpeluettarort@clorabajlitates to unpack

please give an advice

thank you

**Aarshay JalaXSTPS*://W*WW*.*ANA**LYTICSVIDHYA.COM*/*BLOG*/*2016*/*02*/*TIME-SERIES-FORECASTING-CODES-PYTHON/?REPLYTOCOM-J10007#RESPOND) APRIL 25, 2016 AT 5:50 PM (HTTPS*:/*/WWW.AN*A*LYTICSVIDHYA.COM/BLOG/2016*/*0*2/*TIME-SERIES-FORECASTING-CODES PYTHON*/*#COMMENT-110007)

please share the code..

**Dennis Say SE**PLY (HTTPS*://*WWW.ANALYTICSVIDHYA.COM*/*BLOG/2016/02*/*TIME-SERIES FORECASTING-CODES-PYTHON/?REPLYTOCOM-111**506#RESP**OND) **M*AY 27*, 2**016 AT 1:41 AM (HTTPS*://*WWW.ANALYTICSVIDHYA.COM/BLOG*/2*016/02*/*TIME-SERIES-FORECASTING-CODES PYTHON/#COMMENT-111506)

Andrew, your are probably passing in a dataframe instead of a series, in the code Aarshay wrote up for the dftest. Specifically here: dftest = adfuller(timeseries.unstack(), autolag='AIC') note the .unstack() that I added - transforming the df into a series – when I also encountered the same error.

**https*://*w*ww.*analyticsvidhya.com*/*blog/2016*/*0*2*/time-series-forecasting-codes-python*/***

***41/4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

**bawasir ki dawaifhttp*wwt*aikerholaisrombarskahershawasikisdawanbalmboBAY SATHON/?REPLYTOCOM-110515#RESPOND)**

**M*A*Y 6, 2016 AT 1**2*:*47 PM (HTTPS*://*WWW*.*AN*A*LYTICSVIDHYA.COM/BLOG/2016/02*/*TIME-SERIES-FORECASTING-CODES

PYTHON/#COMMENT-110515)

***G*reat blennast thanks for sha**ring this post.

7 Analytics Vidhya V unter de tota rajherbals.com/arshoher-bawasir-ki-dawa.html)

AVCasino

**INTRODUCTION**

HYA.COM*/*BLOG/*2*016/02*/*TIME-SERIES-FORECASTING-CODES-PYTHON/?REPLYTOCOM-110**533#RES**POND) LYTICSVIDHYA.COM/BLOG*/2*016/0*2*/TIME-SERIES-FORECASTING-CODES

**PROB*A*BILITY**

**2016, Aug 10th - 31st**

**ONLINE**

**REGISTER NOW** (https://**darakharshamartAVASSAN MILANALYTICSVIDMATEOMYBLOG/2016*/*0*2/*TIME-SERIES FORECASTING-CODESPYTHON**/PREPLYTOCOM-11435#RESPOND) casino-in**tMAX.25t2016 -1373B MSHE**PS*)//*W*W*W*.A*NALYTICSVIDHYA.COM*/*BLOG*/*2016*/*02*/*TIME-SERIES-FORECASTING-CODES

PYTHON/#COMMENT-111435)

In Dicket-fuller test my came Results of Dickey-Fuller Test: Test Statistic -2.287864 p-value 0.175912 *#*Lags Used 11.000000 Number of Observations Used 215.000000

Critical Value (1%) -3.461136 Critical Value (10%) -2.573986 Critical Value (5%) -2.*8*75079 dtype: float64 My p value is so less? Its means my data is not normal ox not not suited to this model?

**Jie Says: REPLY (HTTPS*:*/*/*WWW.ANALYTICSVIDHYA.COM/BLO**G/2016/02/TIME-SERIES-FO**RECASTING-CODES-PYTHON/?REPLYTOCOM-112372#RESPOND)** JUNE 19, 2016 AT 3:59 AM (HTTPS*://W*WW*.*ANALYTICSVIDHYA.COM/BLOG/2016/02/TIME-SERIES-FORECASTING-CODES PYTHO**N/#COMMENT-1123*7*2)**

really liked this post. Thank you very much for sharing.

**LEAVE A REPLY**

**https*://*w*ww.*analyticsvidhya.com*/*blog/2016*/*0*2*/time-series-forecasting-codes-python*/***

***4*2*/4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

Connect with:

f (https:*//www.*analyticsvidhya.com/wp-login.php? action=wordpress\_social\_authenticate&mode=login&provider=Facebook&redirect\_to=https%3A%2F%2Fw*ww.*a

7 Analytics Vidhya

untibus

AVCasino

**INTRODUCTION**

**PROB*A*BILITY**

**2016, Aug 10th - 31st**

**ONLINE**

**REGISTER NOW**

(https:*//*datahack.analyticsvidhya.com/contest/av casino-introduction-to-probability

Name (required)

Email (required)

Website

**SUBMIT COMMENT**

TOP AV USERS

**Rank Name**

**Points**

SRK (https:*/*/datahack.analyticsvidhya.com/user/profile/SRK)

5378

2

u

Aa*y*ushmnit (https:*/*/datahack.analyticsvidhya.com/user/profile/aa*y*ushmnit)

4818

**https*://www.*analyticsvidhya.com*/*b*l*og*/*2016*/02/*time-series-forecasting-codes-python/**

**43*/4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

4407

Nalin Pasricha (https://datahack.analyticsvidhya.com/user/profile/Nalin)

Vopani (https:*//*datahack analyticsvidhya.com*/*user*/*profile/Rohan Rao)

7 Analytics Vidhya

4353

analyticsvidhya.com/user/profile/binga)

337*1*

AVCasino

**INTRODUCTION**

More Rankings (http://datahack.analyticsvidhya.com/users)

**PROB*A*BILITY**

**2016, Aug 10th - 31st**

**ONLINE**

**REGISTER NOW**

(https:

W BtbhabN!

(http:*//*www*.*greatlearning.in/great-lakes-pgpba?

ntesta*v* casino-introc**hureedASİO-Srobab**ility/).

APPLY NOW

*in Chennal, Gurgaon, Bengaluru, Pune* & *Hyderabad* utm\_source=avm&utm\_medium=avmbanner&utm\_campaign=pgpba)

POPULAR POSTS

*– A* Complete Tutorial to Learn Data Science with Python from Scratch

(https:*//www.*analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python scratch-2/) Essentials of Machine Learning Algorithms (with Python and R Codes)

(https:*//*www.analyticsvidhya.com/blog/2015/08/common-machine-learning-algorithms/) – 7 Types of Regression Techniques you should know!

(https:*//*www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/) – A Complete Tutorial on Tree Based Modeling from Scratch (in R & Python)

(https:*//www.*analyticsvidhya.com/blog/2016*/*04/complete-tutorial-tree-based-modeling-scratch-in python/) A Complete Tutorial on Time Series Modeling in R (https:*//www*.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/)

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

***444*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

Complete guide to create a Time Series Forecast (with Codes in Python)

(https*://*www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/) – Beginner's guide to Web Scraping in Python (using BeautifulSoup)

(https:*//*www.analyticsvidhya.com/blog/2015/10/beginner-guide-web-scraping-beautiful-soup

7 Analytics Vidhya

dI learn? (https:*//w*ww.analyticsvidhya.com/blog/2014/03/sas

AVCasino

**INTRODUCTION**

**PROB*A*BILITY**

**2016, Aug 10th - 31st**

**ONLINE**

REGISTER NOW Kht**Me: Kunahazian Analytiesy.chmadsreakav**at Marketing Conclave 2014 casino-introduction-to-probability,

(http:*/*/imarticus.org/sas-online)

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**45*/4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

RECENT POSTS

7 Analytics Vidhya

un betala om/blog/2016/08/deep-learning-path/)

A Complete Guide on

ted with

Deep Learning in l**idhya.com*/*blo*g****/*2016/08/deep-learnin**g-path)**

A

V

C

asino

**INTRODUCTION**

**PROBABILITY**

om/blog/2016/08/full-solution-skill-test-on-r-for-data

**2016 , Aug 10th - 31st**

Data Science (https*://*www.analyticsvidhya.com/blog*/*2016*/*08/full **ONLINE**

REGISTER NOW

cience/) (https:*/*/datahagkanakyticevidhyamam/contest/av casino-introduction-to-probability

(https:*/*/www.analyticsvidhya.com/blog/2016/08/solutions-for-skilltest-in-statistics

*r*evealed)

Solutions for Skilltest in Statistic**s Revealed** (https*://****w*ww.analyticsvidhya.c**om/blog*/*2016*/*08/solution**s-for-skilltest-in-statistics-revealed/)** KUNAL JAIN , AUGUST 27, 2016

(https:*//ww*w.analyticsvidhya.com/blog/2016/08/10-youtube-videos-explaining-the-real

world-applications-of-internet-of-things-iot/

10Real World Applications of Internet of Things (IoT) - Explained in Videos (https*://www.*analyticsvidhya.com/blo*g/*2016*/*08/10-youtube-**videos-explaining-the-real-world applications-of-internet-of-**things-iot*/*) **SWATI KAS**HYAP, AUGUST 26, 2016

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**46*/4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

LE*DVANCER*

EDUVENTURES

CERTIFIED BUSINESS

7 Analytics Vidhya

(http:*//ww*w*.*edvancer.in/course/cbap?

AVCasino

**INTRODUCTION**

**PROB*A*BILITY**

\_campaign=A*V*adsnonfc&utm\_content=cbapavad)

**2016, Aug 10th - 31st**

**ONLINE**

**REGISTER NOW**

f

18,957

(https:*/*/datanackanalyticsvidhya.com/contest/av casino-introduction-to-probability)

6,110

FOLLOWERS (http://www.twitter.com/analyticsvidhya)

**8+ 1,2*76***

FOLLOWERS (https://plus.google.com/+Analyticsvidhya)

FOLLOWERS (http://www.facebook.com/Analyticsvidhya)

a Email

SUBSCRIBE (http://feedburner.google.com/fb/a/mailverify? uri-analyticsvidhya)

ABOUT US

For those of you, who are wondering what is "Analytics Vidhya", "Analytics" can be defined as the science of extracting insights from raw data. The spectrum of analytics starts from capturing data and evolves into using insights / trends from this data to make informed decisions. Read More (http:/*/*w*w*w.analyticsvidhya.com/about-me/)

STAY CONNECTED

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

*47/4*9

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

y 6,110

f 18,957 FOLLOWERS

FOLLOWERS (http://www.twitter.com/analyticsvidhya)

(http://www.facebook.com/Analyticsvidhya) **1*9*7*0***

7 Analytics Vidhya

**.) Email**

SUBSCRIBE (https Aus.google.com/analytiesviahya) (https://feedburner.google.com/fb/a/mailverify?

L*ATES*T POSTS

uri=analyticsvidhya) INTRODUCTION

**PRORADILET**Vdhya.com/blog/2016*/*08/deep-learning-path/) Get**ting Started with**

A Complete Guide on

Deep Learning in Python (https*//www.*analyticsvidhya.com*/*blo*g/*2016**/08*/*deep-learning path/)**

F*AIZA*N SHAIKH. AUGUST 31, 2016 **ONLINE**

**REGISTER NOW**

***I*hti**ne: *Il*datahack.analyticsvidhya.com/contest/a*v*

CHEBSbi*b.*amaynesvidhya.com/blog/2016/08/full-solution skill-test-on-r-for-data

science)

Full Solution - Skilltest on R for Data Science (https*://w*w*w.*analyticsvidhya.com*/*blo*g/2*016*/*08/full solution-skill-test-on-r-for-**data-science/)** KUNAL JAIN , AUGUST 28, 2016

(https:*//*w*w*w.analyticsvidhya.com/blog/2016/08/solutions-for-skilltest-in-statistics

**revealed/)**

**Solutions** for Skilltest in Stati**stics Revealed** (https*://*www.analyticsvidhya.com*/*blo*g/2*016/08/solutions-for-skillt**est-in-statistics-revealed*/*)** KUNAL JAIN, AUGUST *2*7, 2016

(https:*/*/www.analyticsvidhya.com/blog/2016/08/10-youtube-videos-explaining-the-real

world-applications-of-internet-of-things-iot/)

10Real World Applications of Internet of Things (IoT) - Explained in Videos **(https*://www.*analyticsvidhya.**com/blo*g/*2016*/*08/10-youtube-**videos-explaining-the-real-world** applications-of-internet-of-things-iot*/*) S*W*ATI KASHYAP , AUGUST 26, 2016

**https*://*w*ww.*analyticsvidhya.com*/*blog/2016*/*0*2*/time-series-forecasting-codes-python*/***

**48/*4*9**

**31.8.2016**

**Complete guide to create a Time Series Forecast (with Codes in Python)**

QUICK LINKS

Home (https:*/*/*www.*analyticsvidhya.com*/*) About Us (https://www.analyticsvidhya.com/about-me/) O*ur*team (htipswww.*anal*yticsvidhya.com/about me team

bebe

AVCasino

(https://

**INTRODUCTION** privac policy/) *Re*fund Polic*y* (h*ttp*s://wwwa PROLAZILITY fund policy*/)* Te*rm*s of Use (https So*n Man*alyticsvidhya.com/terms/)

**2016, Aug 10th - 31st**

**ONLINE**

REGISTER NOW (https:7 datahack.analyticsvidhya.com/contest/av casino-introduction-to-probability

© Copyright 2016 Analytics Vidhya

**https*://ww*w*.*analyticsvidhya.co*m/*blog*/*2016*/02/*time-series-forecasting-codes-python/**

**49*/4*9**