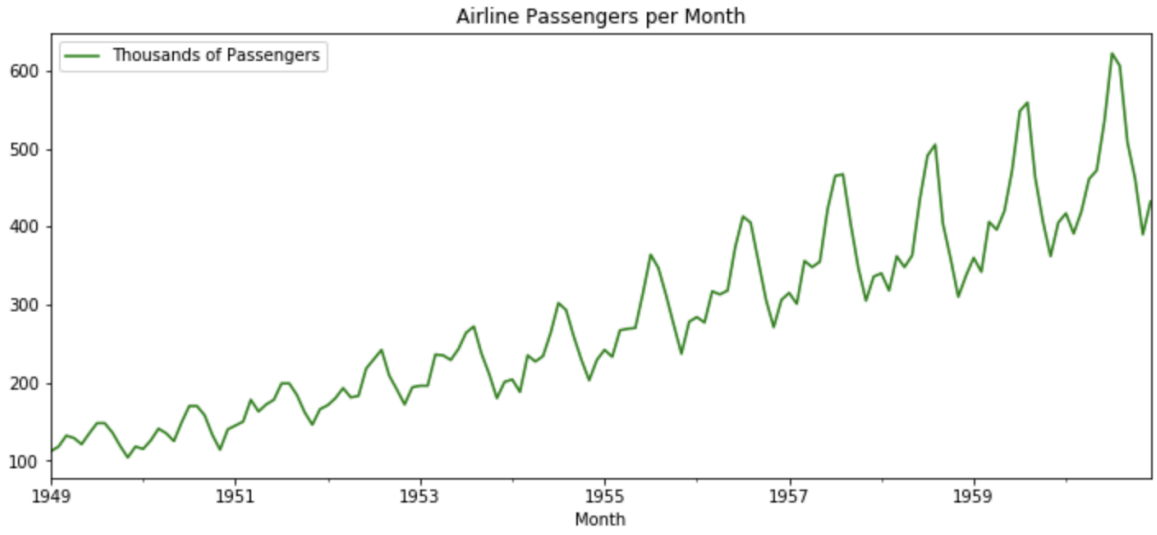
Python | ARIMA Model for Time Series Forecasting

Last Updated : 19 Feb, 2020

A **Time Series** is defined as a series of data points indexed in time order. The time order can be daily, monthly, or even yearly. Given below is an example of a Time Series that illustrates the number of passengers of an airline per month from the year 1949 to 1960.



Time Series Forecasting

Time Series forecasting is the process of using a statistical model to predict future values of a time series based on past results.

Some Use Cases

- To predict the number of incoming or churning customers.
- To explaining seasonal patterns in sales.
- To detect unusual events and estimate the magnitude of their effect.
- To Estimate the effect of a newly launched product on number of sold units.

Components of a Time Series:

- **Trend:**The trend shows a general direction of the time series data over a long period of time. A trend can be increasing(upward), decreasing(downward), or horizontal(stationary).
- **Seasonality:**The seasonality component exhibits a trend that repeats with respect to timing, direction, and magnitude. Some examples include an increase in water consumption in summer due to hot weather conditions, or an increase in the number of airline passengers during holidays each year.
- **Cyclical Component:** These are the trends with no set repetition over a particular period of time. A cycle refers to the period of ups and downs, booms and slums of a time series, mostly observed in business cycles. These cycles do not exhibit a seasonal variation but generally occur over a time period of 3 to 12 years depending on the nature of the time series.
- **Irregular Variation:** These are the fluctuations in the time series data which become evident when trend and cyclical variations are removed. These variations are unpredictable, erratic, and may or may not be random.
- **ETS Decomposition**
  ETS Decomposition is used to separate different components of a time series. The term ETS stands for Error, Trend, and Seasonality.

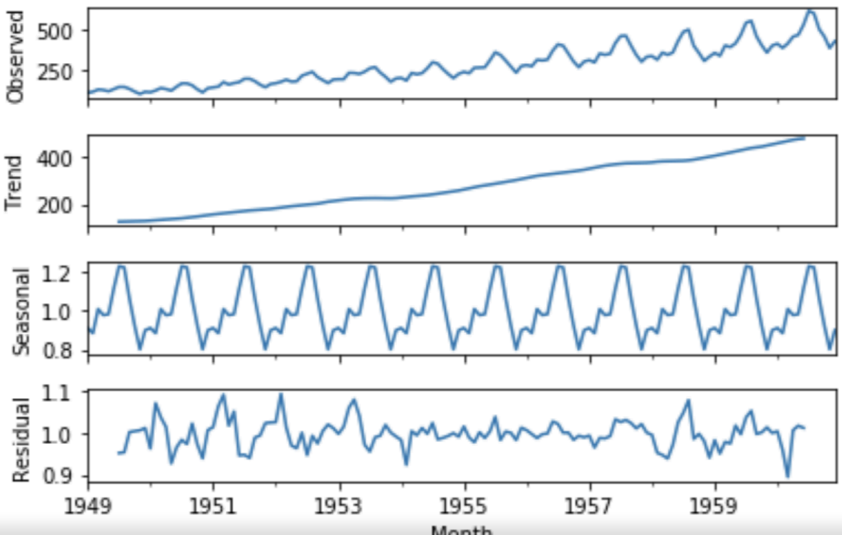
Code: ETS Decomposition of Airline Passengers Dataset:

```
# Importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pylot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
```

```
                        parse_dates = True)

# Print the first five rows of the dataset
airline.head()

# ETS Decomposition
result = seasonal_decompose(airline['# Passengers'],
                            model ='multiplicative')

# ETS plot
result.plot()
```

Output:

Month	#Passengers
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121



ARIMA Model for Time Series Forecasting

ARIMA stands for autoregressive integrated moving average model and is specified by three order parameters: *(p, d, q).*

- **AR(*p*) Autoregression** – a regression model that utilizes the dependent relationship between a current observation and observations over a previous period.An auto regressive (*AR(p)*) component refers to the use of past values in the regression equation for the time series.
- **I(*d*) Integration** – uses differencing of observations (subtracting an observation from observation at the previous time step) in order to make the time series stationary. Differencing involves the subtraction of the current values of a series with its previous values d number of times.
- **MA(*q*) Moving Average** – a model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations. A moving average component depicts the error of the model as a combination of previous error terms. The order *q* represents the number of terms to be included in the model.

Types of ARIMA Model

- **ARIMA:**Non-seasonal Autoregressive Integrated Moving Averages
- **SARIMA:**Seasonal ARIMA
- **SARIMAX:**Seasonal ARIMA with exogenous variables

Pyramid Auto-ARIMA

The **'auto_arima'** function from the **'pmdarima'** library helps us to identify the most optimal parameters for an ARIMA model and returns a fitted ARIMA model.

Code : Parameter Analysis for the ARIMA model

```
<div id="highlighter_178447" class="syntaxhighlighter nogutter  "><table border="0
```

Output:

Statespace Model Results

Dep. Variable:	y	**No. Observations:**	144
Model:	SARIMAX(0, 1, 1)x(2, 1, 1, 12)	**Log Likelihood**	-501.921
Date:	Fri, 31 Jan 2020	**AIC**	1015.842
Time:	18:07:17	**BIC**	1033.093
Sample:	0	**HQIC**	1022.852
	- 144		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.0004	0.033	0.011	0.991	-0.064	0.064
ma.L1	-0.4259	0.068	-6.223	0.000	-0.560	-0.292
ar.S.L12	0.6682	0.160	4.189	0.000	0.356	0.981
ar.S.L24	0.3302	0.096	3.447	0.001	0.142	0.518
ma.S.L12	-0.9734	1.188	-0.820	0.412	-3.301	1.354
sigma2	111.0288	108.931	1.019	0.308	-102.472	324.530

Ljung-Box (Q):	52.94	**Jarque-Bera (JB):**	7.41
Prob(Q):	0.08	**Prob(JB):**	0.02
Heteroskedasticity (H):	2.82	**Skew:**	0.10
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.15

Code : Fit ARIMA Model to AirPassengers dataset

```
start = len(train)
end = len(train) + len(test) - 1

# Predictions for one-year against the test set
predictions = result.predict(start, end,
                             typ = 'levels').rename("Predictions")

# plot predictions and actual values
predictions.plot(legend = True)
test['# Passengers'].plot(legend = True)
```

Output:

Statespace Model Results

Dep. Variable:	#Passengers	**No. Observations:**	132
Model:	SARIMAX(0, 1, 1)x(2, 1, 1, 12)	**Log Likelihood**	-443.013
Date:	Fri, 31 Jan 2020	**AIC**	896.026
Time:	18:24:59	**BIC**	909.922
Sample:	01-01-1949	**HQIC**	901.669
	- 12-01-1959		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	-0.2983	0.077	-3.870	0.000	-0.449	-0.147
ar.S.L12	0.7097	0.236	3.005	0.003	0.247	1.173
ar.S.L24	0.2895	0.100	2.884	0.004	0.093	0.486
ma.S.L12	-0.9803	2.266	-0.433	0.665	-5.422	3.461
sigma2	88.2925	180.492	0.489	0.625	-265.465	442.050

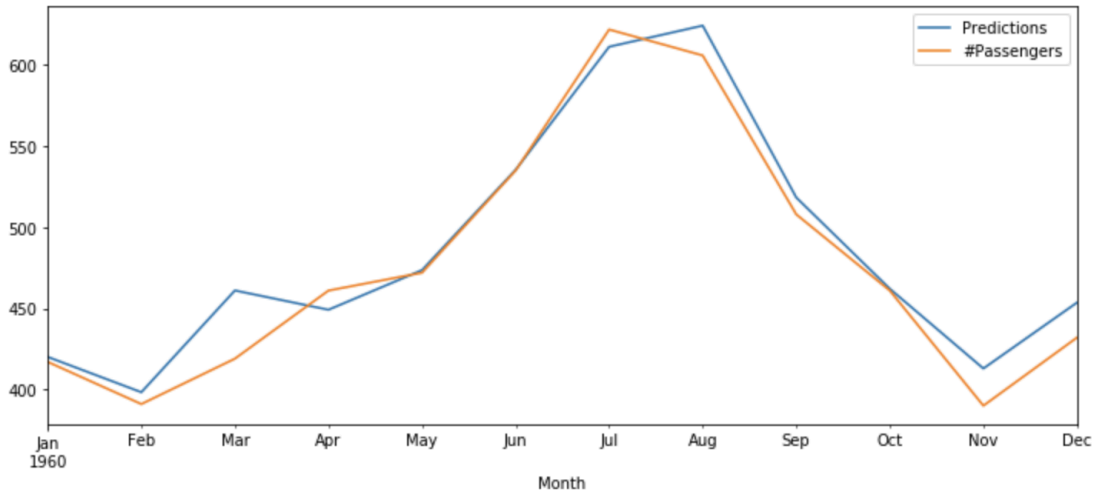
Ljung-Box (Q):	38.27	**Jarque-Bera (JB):**	0.00
Prob(Q):	0.55	**Prob(JB):**	1.00
Heteroskedasticity (H):	1.62	**Skew:**	-0.00
Prob(H) (two-sided):	0.13	**Kurtosis:**	2.99

Warnings:

```
# Load specific evaluation tools
from sklearn.metrics import mean_squared_error
from statsmodels.tools.eval_measures import rmse

# Calculate root mean squared error
rmse(test["# Passengers"], predictions)

# Calculate mean squared error
mean_squared_error(test["# Passengers"], predictions)
```

Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c21ac3050>
```



Code : Evaluate the model using MSE and RMSE

```
# Train the model on the full dataset
model = model = SARIMAX(airline['# Passengers'],
                        order = (0, 1, 1),
                        seasonal_order =(2, 1, 1, 12))
result = model.fit()

# Forecast for the next 3 years
forecast = result.predict(start = len(airline),
                          end = (len(airline)-1) + 3 * 12,
                          typ = 'levels').rename('Forecast')

# Plot the forecast values
airline['# Passengers'].plot(figsize = (12, 5), legend = True)
```

```
rmse(test["#Passengers"],predictions)
```

```
17.145543874593976
```

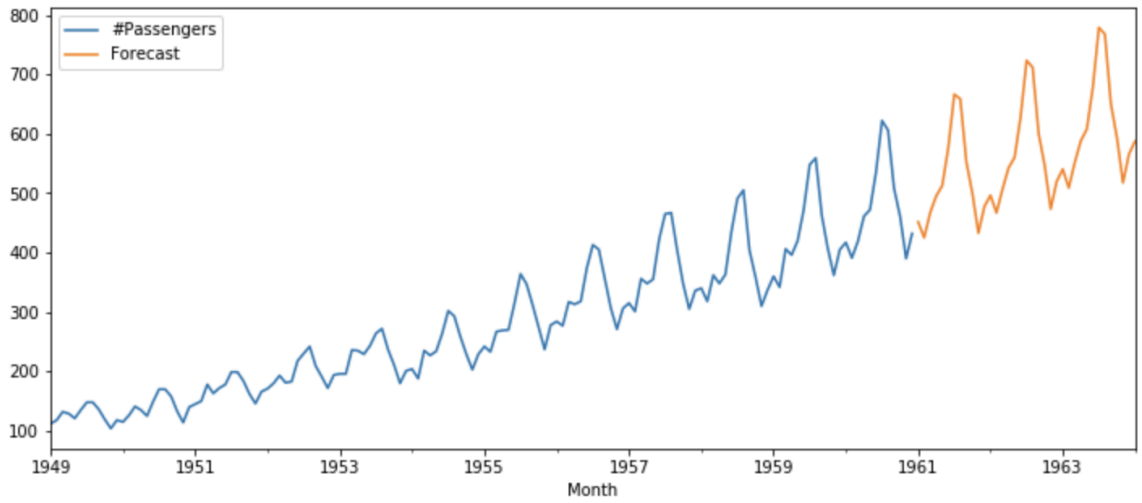
```
mean_squared_error(test["#Passengers"],predictions)
```

```
293.96967475562707
```

Code : Forecast using ARIMA Model

Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c20715950>
```



Comment More info

Next Article

Multiclass classification using scikit-learn

Similar Reads

Machine Learning Algorithms

Machine learning algorithms are computational models that allow computers to understand patterns and forecast or make judgments based on data without explicit programming. These algorithms form the...

14 min read

Top 15 Machine Learning Algorithms Every Data Scientist Should Know in 2024

Machine Learning (ML) Algorithms are the backbone of everything from Netflix recommendations to fraud detection in financial institutions. These algorithms form the core of intelligent systems, empowering...

15 min read

Linear Model Regression

Linear Model Classification

Regularization

K-Nearest Neighbors (KNN)

Support Vector Machines

ML | Stochastic Gradient Descent (SGD)

Gradient Descent is an iterative optimization process that searches for an objective function's optimum value (Minimum/Maximum). It is one of the most used methods for changing a model's parameters in order to redu...

10 min read

Decision Tree

Generative Model

Time Series Forecasting

Components of Time Series Data

Time series data is a sequence of data points recorded or collected at regular time intervals. It is a type of data that tracks the evolution of a variable over time, such as sales, stock prices, temperature, etc. The...

11 min read

AutoCorrelation

Autocorrelation is a fundamental concept in time series analysis. Autocorrelation is a statistical concept that assesses the degree of correlation between the values of variable at different time points. The article aims t...

10 min read

How to Check if Time Series Data is Stationary with Python?

Time series data are generally characterized by their temporal nature. This temporal nature adds a trend or seasonality to the data that makes it compatible for time series analysis and forecasting. Time-series data is...

8 min read

How to Perform an Augmented Dickey-Fuller Test in R

Augmented Dickey-Fuller Test: It is a common test in statistics and is used to check whether a given time series is at rest. A given time series can be called stationary or at rest if it doesn't have any trend and depict...

3 min read

How to calculate MOVING AVERAGE in a Pandas DataFrame?

Calculating the moving average in a Pandas DataFrame is used for smoothing time series data and identifying trends. The moving average, also known as the rolling mean, helps reduce noise and highlight...

7 min read

Exponential Smoothing in R Programming

The Exponential Smoothing is a technique for smoothing data of time series using an exponential window function. It is a rule of the thumb method. Unlike simple moving average, over time the exponential function...

15+ min read

Python | ARIMA Model for Time Series Forecasting

Supervised Dimensionality Reduction Technique

Metrics for Classification & Regression Algorithms

Cross Validation Technique

Optimization Technique

Clustering

Association Rule Mining

Anomaly Detection

Dimensionality Reduction Technique

Model-Based Methods

Model-Free Methods

Asynchronous Advantage Actor Critic (A3C) algorithm

The Asynchronous Advantage Actor Critic (A3C) algorithm is one of the newest algorithms to be developed under the field of Deep Reinforcement Learning Algorithms. This algorithm was developed by Google's...

3 min read

Article Tags : AI-ML-DS Machine Learning AI-ML-DS With Python

Practice Tags : Machine Learning

Corporate & Communications Address:- A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305) | Registered Address:- K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305