Time Series Modeling

- Time Series Modeling - Overview
  - 1.1. Time Series Model: Definition
  - 1.2. Applications
  - 1.3. Time Series: Need
  - 1.4. Time Series Pattern Types
  - 1.5. Trend
  - 1.6. Seasonal
  - 1.7. Cyclic
  - 1.8. Irregular
  - 1.9. White Noise
  - 1.10. Stationarity
  - 1.11. Non Stationary Series
  - 1.12. Stationarity Check
    - 1.12.1. Rolling Statistics (Visual Stationarity Check)
    - 1.12.2. Dickey Fuller test (Statistical Stationarity Check)
  - 1.13. Removal of Non-Stationarity
    - 1.13.1. Differencing
    - 1.13.2. Decomposition
  - 1.14. Demo: Air Passengers - I

Time Series Modeling - Overview

Time Series Model: Definition

Time Series Models are useful models, when we have weekly contested data.

Series of monthly sales data

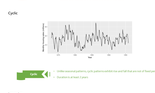
Time series is a sequence of observations over successive intervals of time.

A time series defined by the random variable Y_t, $t = 1, 2, \ldots, n$ and Y_t is a function of t.

$$Y = f(t)$$

Applications

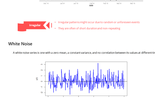
Time Series: Need



Time Series Pattern Types

- Trend
- Seasonal
- Cyclic
- Irregular

Trend



Seasonal

Cyclic

Irregular

White Noise

A stationary time series is one with a zero mean, a constant variance, and no correlation between its values at different times.

Random walk
Random walk
Random walk

The results of Dickey-Fuller test show that the null hypothesis is rejected at the 5% level of significance. The p-value is 0.025800, which is less than the critical value for the test statistic at the 5% level.

Practice Project: Beer Production - I

Problem Statement: The Beer production dataset provides a time series data for monthly beer production in Australia. For the period ranging from 1956 to 1995.

Objective: Check for the stationarity of your data using Rolling Statistics and Dickey Fuller test. If stationarity is present, remove it using differencing or factor.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
from pandas.tseries.offsets import *
%matplotlib inline
beer = pd.read_csv('/home/dataset/beer-production-in-australia.csv')
beer.head()
```

	Month	Monthly beer production in Australia
0	1956-01	93.0
1	1956-02	96.0
2	1956-03	95.0
3	1956-04	75.0
4	1956-05	62.0

```
beer.shape
beer.dtypes
beer.rename(inplace=True, columns={'Monthly beer production in Australia': 'Beer'})
beer['Month'] = pd.to_datetime(beer['Month'])
beer.set_index('Month', inplace=True)
ts = beer['Beer']
plot = ts.plot(figsize=(15,5))
```

```
Beer    int64
dtype: object
Beer, dtype: int64
Month, dtype: int64
```

```
def test_stationarity(timeseries):
    #Determing rolling statistics
    rolmean = timeseries.rolling(12).mean()
    rolstd = timeseries.rolling(12).std()
    #Plot rolling statistics:
    orig = plot(timeseries, color='blue',label='Original')
    mean = plot(rolmean, color='red', label='Rolling Mean')
    std = plot(rolstd, color='black', label = 'Rolling Std')
    plot(legend=True,loc='upper left')
    show(block=False)
    #Perform Dickey-Fuller test:
    print('Results of Dickey-Fuller Test:')
    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 (%d)'%key] = value
    print(dfoutput)
```

```
test_stationarity(ts)
```

The p-value = 0.0 is smaller than the critical value of 0.05, so we reject the null hypothesis and conclude that the time series is stationary.

For the test to be true, we need to have at least one of the following assumptions:

- The mean of the series should not be a function of time rather should be a constant.
- The variance should not be a function of time.
- The covariance should be independent of time.

Various Time Series Models

```
ts_log = np.log(ts)
plot = ts_log.plot(figsize=(15,5))
```

```
moving_avg = ts_log.rolling(12).mean()
plot = moving_avg.plot(figsize=(15,5))
```

```
ts_log_moving_avg = ts_log - moving_avg
ts_log_moving_avg.head(12)
```

```
Month
1956-01    NaN
1956-02    NaN
1956-03    NaN
1956-04    NaN
1956-05    NaN
1956-06    NaN
1956-07    NaN
1956-08    NaN
1956-09    NaN
1956-10    NaN
1956-11    NaN
1956-12    0.0
Beer, dtype: int64
```

```
test_stationarity(ts_log_moving_avg)
```

The p-value = 0.0 is smaller than the critical value of 0.05, so we reject the null hypothesis and conclude that the time series is stationary.

Auto Regressive (AR) Model

```
ts_log_moving_avg = ts_log - moving_avg
ts_log_moving_avg.dropna(inplace=True)
test_stationarity(ts_log_moving_avg)
```

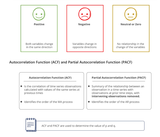
```
expwighted_avg = ts_log.ewm(halflife=12).mean()
plot = expwighted_avg.plot(figsize=(15,5))
```

```
ts_log_ewm = ts_log - expwighted_avg
test_stationarity(ts_log_ewm)
```

```
ts_log_diff = ts_log - ts_log.shift()
plot = ts_log_diff.plot(figsize=(15,5))
```

```
ts_log_diff.dropna(inplace=True)
test_stationarity(ts_log_diff)
```

```
decomposition = seasonal_decompose(ts_log, freq=12)
trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.residual
plot = plot(trend, color='blue',label='Trend')
plot = plot(seasonal, color='red', label='Seasonality')
plot = plot(residual, color='black', label = 'Residuals')
plot(legend=True,loc='upper left')
show(block=False)
```




```
ts_log_decompose = residual
ts_log_decompose.dropna(inplace=True)
test_stationarity(ts_log_decompose)
```

The p-value = 0.0 is smaller than the critical value of 0.05, so we reject the null hypothesis and conclude that the time series is stationary.

Moving Average (MA) Model

NIA Model vs SIA smoothing

ARIMA Model

ARIMA Model

Correlation

Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF)

Characteristics of ACF and PACF

Steps in Time Series Forecasting

Demo: Air Passengers - II

Practice Project: Beer Production - II

Date, 4.4.1.1.1
400
350
300
250
200
150
100
50
0
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000