Chapter 18: Demo Time Series với ARIMA

Giới thiệu

- Dữ liệu chuỗi thời gian là dữ liệu thử nghiệm đã được quan sát tại các thời điểm khác nhau theo thời gian (thường là khoảng cách đều nhau, như một lần mỗi ngày/mỗi tháng/mỗi quý...)
- Ví dụ, dữ liệu bán vé máy bay mỗi ngày là một chuỗi thời gian.
- Tuy nhiên, chỉ vì một loạt các sự kiện có yếu tố thời gian không tự động biến nó thành chuỗi thời gian, chẳng hạn như các ngày xảy ra thảm họa hàng không, với khoảng cách thời gian ngẫu nhiên thì không phải là chuỗi thời gian. Các loại quy trình ngẫu nhiên này được gọi là quá trình điểm (point process).
- Time Series có một số tính năng chính như xu hướng (trend), tính thời vụ (seasonality) và nhiễu (noise).
- Công việc của chúng ta là phân tích các tính năng này của tập dữ liệu time series và sau đó áp dụng mô hình để dự báo trong tương lai

Dự đoán

- “Dự đoán rất khó, đặc biệt là dự báo về tương lai”.
- Dự báo là quá trình đưa ra dự đoán về tương lai, dựa trên dữ liệu trong quá khứ và hiện tại. Một trong những phương pháp phổ biến nhất là sử dụng mô hình **ARIMA** (AutoRegressive Integrated Moving Average)

Thuật toán ARIMA

- Trong mô hình ARIMA có 3 tham số được sử dụng để giúp mô hình hóa các khía cạnh chính của một chuỗi thời gian: seasonality, trend, và noise. Các tham số này được gắn nhãn lần lượt là p, d và q.
- p là tham số kết hợp với khía cạnh tự động hồi quy của mô hình (auto-regressive aspect), kết hợp các giá trị trong quá khứ. Ví dụ: dự báo rằng nếu trời mưa nhiều trong vài ngày qua, có thể cho biết ngày mai trời sẽ mưa.
- d là tham số kết hợp với phần tích hợp của mô hình (integrated part), nó ảnh hưởng đến lượng chênh lệch áp dụng cho một chuỗi thời gian. Ví dụ: dự báo rằng lượng mưa ngày mai sẽ tương tự như lượng mưa ngày hôm nay, nếu lượng mưa hàng ngày tương tự trong vài ngày qua.
- q là tham số liên quan đến phần trung bình động của mô hình (moving average part).
- Nếu mô hình có thành phần theo mùa, ta sử dụng mô hình ARIMA theo mùa (SARIMA). Trong trường hợp đó, ta có một bộ tham số khác: P, D và Q mô tả các liên kết tương tự như p, d và q, nhưng tương ứng với các thành phần theo mùa của mô hình (Seasonal model)

- Chú ý: Ta có thể mắc lỗi phổ biến là ngay lập tức bắt đầu áp dụng các mô hình dự báo ARIMA cho dữ liệu có nhiều yếu tố bên ngoài, ví dụ như giá cổ phiếu hoặc hiệu suất của một đội thể thao. Mặc dù ARIMA có thể là công cụ mạnh mẽ và phù hợp cho chuỗi thời gian liên quan đến

các chủ đề nói trên, nếu ta chỉ sử dụng nó và không tính đến các yếu tố bên ngoài, chẳng hạn như CEO bị sa thải hoặc chấn thương trong nhóm, ta sẽ không có kết quả tốt. Cần ghi nhớ điều này khi ta bắt đầu áp dụng các khái niệm này cho các tập dữ liệu.

Thuộc tính và loại của series

- Trend : Tăng hoặc giảm dài hạn trong dữ liệu. Có thể được xem như là một độ dốc - slope (không phải là tuyến tính) gần như đi xuyên qua dữ liệu
- Seasonality : Một chuỗi thời gian được cho là theo mùa khi nó bị ảnh hưởng bởi các yếu tố theo mùa (giờ trong ngày, tuần, tháng, năm, v.v.). Tính thời vụ có thể được quan sát với các mẫu chu kỳ (cyclical patterns) có tần số cố định (fixed frequency).
- Cyclicity : Một chu kỳ xảy ra khi dữ liệu biểu hiện tăng và giảm không có tần số cố định. Những biến động này thường là do điều kiện kinh tế, và thường liên quan đến “business cycle”. Thời gian của những biến động này thường ít nhất là 2 năm.
- Residuals :
  Mỗi chuỗi thời gian có thể được phân tách thành hai phần
  - Forecast: bao gồm một hoặc một số giá trị dự báo (forecasted values)
  - Residuals: sự khác biệt giữa một quan sát (observation) và giá trị được dự đoán của nó ở mỗi time step.
  - Với: Value of series at time t = Predicted value at time t + Residual at time t

Variation (biến thể)

- Một trong những tính năng quan trọng nhất của một time series là variation (biến thể). Biến thể là các mẫu trong time series. Time series có các mẫu lặp lại trong khoảng thời gian đã biết và cố định được được cho là có tính thời vụ (seasonality). Tính thời vụ là thuật ngữ chung cho các biến thể lặp lại định kỳ trong dữ liệu. Các biến thể có 4 loại: theo mùa, theo chu kỳ, xu hướng và biến động không đều (Seasonal, Cyclic, Trend và Irregular fluctuations).
- Biến động theo mùa (Seasonal variation) thường được định nghĩa là biến thể hàng năm trong kỳ, chẳng hạn như doanh số bán áo tắm thấp hơn vào mùa đông và cao hơn vào mùa hè.
- Biến thiên tuần hoàn (Cyclic Variation) là một biến thể xảy ra tại các khoảng thời gian cố định khác, chẳng hạn như biến đổi nhiệt độ hàng ngày.
- Cả hai biến thể theo Seasonal và Cyclic là ví dụ về tính thời vụ trong tập dữ liệu chuỗi thời gian.
- Xu hướng (Trend) là những thay đổi dài hạn ở mức trung bình, liên quan đến số lượng mẫu quan sát.

Chú ý: Các phương thức sử dụng trong ví dụ này sẽ chỉ lấy dữ liệu từ chuỗi thời gian không biến đổi. Điều đó có nghĩa là ta chỉ xem xét mối quan hệ giữa giá trị trục y và thời điểm trục x, không xem xét các yếu tố bên ngoài có thể ảnh hưởng đến chuỗi thời gian.

In [1]:
```
import pandas as pd
```

In [2]:
```
data = pd.read_csv("electric_production.csv",index_col=0)
data.head()
```

Out[2]:

DATE	IPG2211A2N
1939-01-01	3.3842
1939-02-01	3.4100
1939-03-01	3.4875
1939-04-01	3.5133
1939-05-01	3.5133

In [3]:
```
data.index = pd.to_datetime(data.index)
```

In [4]:
```
data.index
```

Out[4]:
```
DatetimeIndex(['1939-01-01', '1939-02-01', '1939-03-01', '1939-04-01',
               '1939-05-01', '1939-06-01', '1939-07-01', '1939-08-01',
               '1939-09-01', '1939-10-01',
               ...
               '2017-11-01', '2017-12-01', '2018-01-01', '2018-02-01',
               '2018-03-01', '2018-04-01', '2018-05-01', '2018-06-01',
               '2018-07-01', '2018-08-01'],
              dtype='datetime64[ns]', name='DATE', length=956, freq=None)
```

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 956 entries, 1939-01-01 to 2018-08-01
Data columns (total 1 columns):
IPG2211A2N    956 non-null float64
dtypes: float64(1)
memory usage: 14.9 KB
```

In [6]:
```
data.columns = ['Energy Production']
```

In [7]:
```
data.head()
```

Out[7]:

DATE	Energy Production
1939-01-01	3.3842
1939-02-01	3.4100
1939-03-01	3.4875
1939-04-01	3.5133
1939-05-01	3.5133

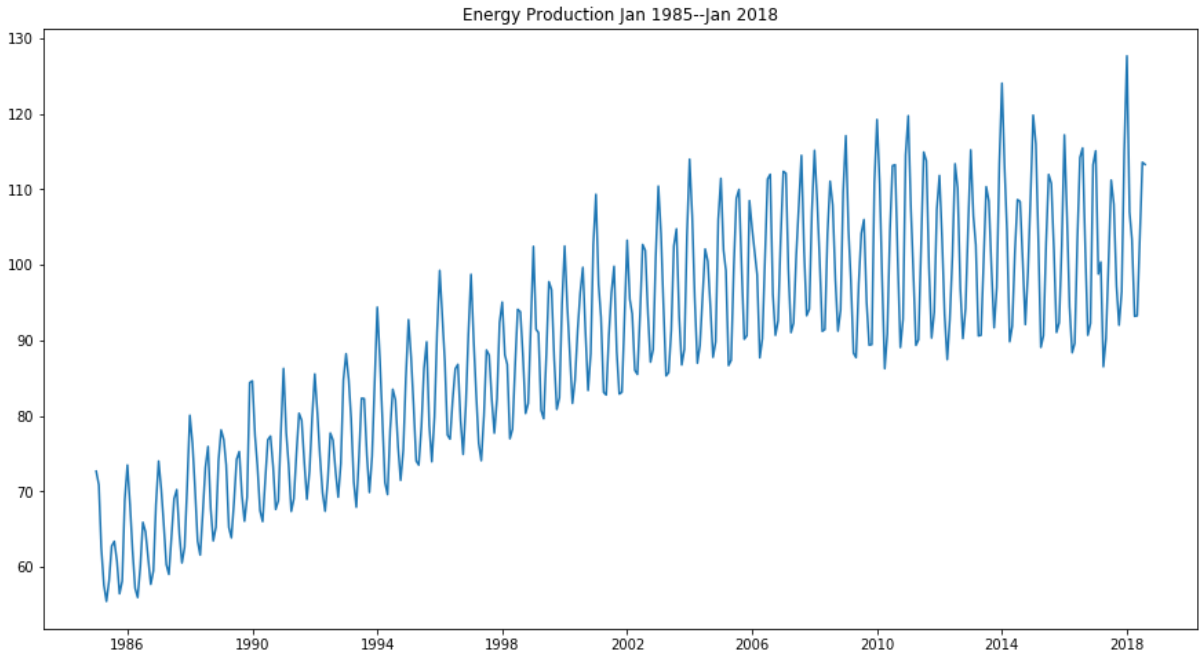
In [8]:
```
import matplotlib.pyplot as plt
```

In [9]:
```
data_1985 = data[data.index.year >=int(1985)]
data_1985.head()
```

Out[9]:

DATE	Energy Production
1985-01-01	72.6803
1985-02-01	70.8479
1985-03-01	62.6166
1985-04-01	57.6106
1985-05-01	55.4467

In [10]:
```
plt.figure(figsize=(15,8))
plt.plot(data_1985)
plt.title("Energy Production Jan 1985--Jan 2018")
plt.show()
```



Decomposition (phân tích):

Một chuỗi thời gian là sự kết hợp của các thành phần sau:

- Trend (chuyển động lên hoặc xuống của đường cong dài hạn (long term)
- Seasonal component (thành phần theo mùa)
- Residuals

Quan sát biểu đồ:

- Biểu đồ trên cho thấy có một xu hướng tăng. Ta có thể sử dụng các mô hình thống kê để thực hiện phân tích chuỗi thời gian này. Sự phân tích chuỗi thời gian là một nhiệm vụ thống kê để

giải mã chuỗi thời gian thành nhiều thành phần, mỗi phần đại diện cho một trong các danh mục mẫu cơ bản. Với các mô hình thống kê, ta có thể thấy xu hướng, theo mùa và các thành phần còn lại của dữ liệu.

- Ta có thể sử dụng mô hình cộng (additive model) khi xu hướng có vẻ như tuyến tính và các trend component dường như không đổi theo thời gian (ví dụ: hàng năm thêm vào 100 đơn vị sản xuất).
- Một mô hình nhân (multiplicative model) thích hợp hơn khi ta tăng (hoặc giảm) ở một tỷ lệ phi tuyến tính (non-linear rate) (ví dụ, mỗi năm nhân đôi số lượng sản xuất).
- Dựa trên biểu đồ trước, ta thấy xu hướng tăng nhẹ với tốc độ hơi khác so với tuyến tính (linenear) (có thể thử nghiệm với cả mô hình cộng (additive model) và mô hình nhân (multiplicative model)).

Function:

statsmodels.tsa.seasonal.seasonal_decompose(x, model='additive', filt=None, freq=None, two_sided=True)

- Phân tích seasonal bằng cách dựa vào trung bình di chuyển

Trong đó:

- x (array): Time series. Nếu là 2d, các individual serries ở trong các column.
- model (str {"additive", "multiplicative"}): loại seasonal component.
- filt (array, tùy chọn): hệ số lọc để lọc ra các seasonal component. Phương pháp trung bình di chuyển, moving average method, được sử dụng trong việc lọc được xác định bởi two_sided.
- freq (int, tùy chọn): Tần số của series. Cần phải sử dụng nếu x không phải là pandas object. Ghi đè chu kỳ mặc định của x nếu x là pandas object với timeseries index.
- two_sided (bool, mặc định): Phương pháp di chuyển trung bình được sử dụng khi lọc. Nếu True (mặc định), một trung bình di chuyển trung tâm được tính bằng filt. Nếu False, các hệ số lọc chỉ dành cho các giá trị trong quá khứ.

Với:

Đây là một naive decomposition. Các phương thức phức tạp hơn nên được ưu tiên hơn.

- Mô hình cộng (additive model): Y[t] = T[t] + S[t] + e[t] (với T: Trend, S: Seasonal, e: Residual)
- Mô hình nhân (multiplicative model): Y[t] = T[t] * S[t] * e[t] (với T: Trend, S: Seasonal, e: Residual)

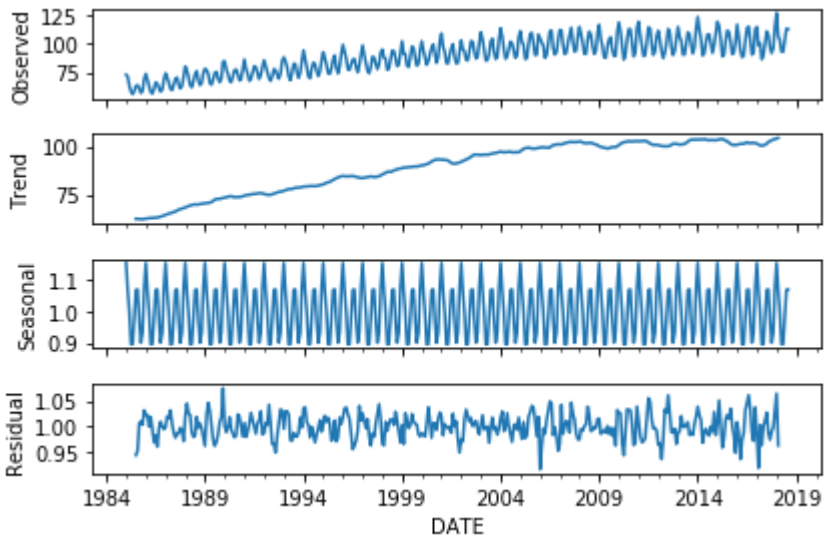
Thành phần seasonal được loại bỏ đầu tiên bằng cách áp dụng bộ lọc chập (convolution) cho dữ liệu. Mức trung bình của series được làm mịn cho từng giai đoạn là seasonal component được trả về.

Kết quả trả về: Một đối tượng với các thuộc tính theo mùa, xu hướng và số dư (seasonal, trend, và resid)

In [11]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
result = seasonal_decompose(data_1985, model='multiplicative')
result
```

Out[11]:
```
<statsmodels.tsa.seasonal.DecomposeResult at 0x25966c8d198>
```

In [12]:
```
result.plot()
plt.show()
```



- Với kết quả trên, ta có thể thấy rõ tính seasonal component của data, và cũng có thể thấy xu hướng dữ liệu ở trên được tách riêng.

- Trend có thể lên hoặc xuống và có thể tuyến tính hoặc phi tuyến tính. Cần phải hiểu tập dữ liệu để biết liệu một khoảng thời gian đáng kể đã trôi qua có thể xác định xu hướng thực tế hay chưa.
- Cũng có thể có biến động bất thường (Irregular fluctuation) là những thay đổi đột ngột ngẫu nhiên và không thể đoán trước

Áp dụng auto_arima để xây dựng mô hình

Cài pip install pyramid-arima

In [13]:
```
from pyramid.arima import auto_arima
```

```
c:\program files\python36\lib\site-packages\pyramid\__init__.py:68: UserWarnin
g:
    The 'pyramid' package will be migrating to a new namespace beginning in
    version 1.0.0: 'pmdarima'. This is due to a package name collision with th
e
    Pyramid web framework. For more information, see Issue #34:

        https://github.com/tgsmith61591/pyramid/issues/34 (https://github.com/
tgsmith61591/pyramid/issues/34)

    The package will subsequently be installable via the name 'pmdarima'; the
    only functional change to the user will be the import name. All imports
    from 'pyramid' will change to 'pmdarima'.

  """, UserWarning)
```

AIC (The Akaike information criterion)

- Là một bộ ước lượng chất lượng tương đối của các mô hình thống kê cho một tập dữ liệu nhất định. Cung cấp một tập hợp các mô hình cho dữ liệu, AIC sẽ ước tính chất lượng của từng mô hình, liên quan đến từng mô hình khác.
- Giá trị AIC cho phép so sánh mô hình phù hợp với dữ liệu và tính đến độ phức tạp của mô hình, vì vậy các mô hình phù hợp hơn trong khi sử dụng ít tính năng hơn sẽ nhận được điểm AIC tốt hơn (thấp hơn) các mô hình tương tự sử dụng nhiều tính năng hơn.
- Thư viện pyramid-arima cho phép ta nhanh chóng thực hiện grid search và tạo ra một model object mà ta có thể fit training data.
- Thư viện này chứa auto_arima function cho phép chúng ta thiết lập một loạt các giá trị p, d, q, P, D và Q và sau đó phù hợp với các mô hình cho tất cả các kết hợp có thể. Sau đó, mô hình sẽ giữ kết hợp có trị AIC tốt nhất.

In [14]:
```
stepwise_model = auto_arima(data, start_p=2, start_q=2,
                            max_p=5, max_q=5, m=12,
                            start_P=1, seasonal=True,
                            d=1, D=1, trace=True,
                            error_action='ignore',
                            suppress_warnings=True,
                            stepwise=True)
```

```
Fit ARIMA: order=(2, 1, 2) seasonal_order=(1, 1, 1, 12); AIC=3729.018, BIC=376
7.811, Fit time=11.552 seconds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(0, 1, 0, 12); AIC=4238.962, BIC=424
8.660, Fit time=0.092 seconds
Fit ARIMA: order=(1, 1, 0) seasonal_order=(1, 1, 0, 12); AIC=4058.517, BIC=407
7.914, Fit time=1.256 seconds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(0, 1, 1, 12); AIC=3859.889, BIC=387
9.286, Fit time=1.904 seconds
Fit ARIMA: order=(2, 1, 2) seasonal_order=(0, 1, 1, 12); AIC=3729.631, BIC=376
3.574, Fit time=8.028 seconds
Fit ARIMA: order=(2, 1, 2) seasonal_order=(2, 1, 1, 12); AIC=3707.297, BIC=375
0.938, Fit time=37.189 seconds
Fit ARIMA: order=(2, 1, 2) seasonal_order=(2, 1, 0, 12); AIC=3776.649, BIC=381
5.442, Fit time=22.306 seconds
Fit ARIMA: order=(2, 1, 2) seasonal_order=(2, 1, 2, 12); AIC=3698.512, BIC=374
7.003, Fit time=38.462 seconds
Fit ARIMA: order=(1, 1, 2) seasonal_order=(2, 1, 2, 12); AIC=3699.508, BIC=374
3.149, Fit time=34.245 seconds
Fit ARIMA: order=(3, 1, 2) seasonal_order=(2, 1, 2, 12); AIC=nan, BIC=nan, Fit
time=nan seconds
Fit ARIMA: order=(2, 1, 1) seasonal_order=(2, 1, 2, 12); AIC=3701.329, BIC=374
4.970, Fit time=31.525 seconds
Fit ARIMA: order=(2, 1, 3) seasonal_order=(2, 1, 2, 12); AIC=3700.293, BIC=375
3.633, Fit time=48.572 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(2, 1, 2, 12); AIC=3700.580, BIC=373
9.372, Fit time=38.443 seconds
Fit ARIMA: order=(3, 1, 3) seasonal_order=(2, 1, 2, 12); AIC=3698.976, BIC=375
7.165, Fit time=57.966 seconds
Fit ARIMA: order=(2, 1, 2) seasonal_order=(1, 1, 2, 12); AIC=3723.193, BIC=376
6.834, Fit time=38.391 seconds
Total fit time: 369.939 seconds
```

In [15]:
```
print(stepwise_model.aic())
```

```
3698.5121729262005
```

In [16]:
```
train = data.loc['1985-01-01':'2016-12-01']
test = data.loc['2015-01-01':]
```

In [17]:
```
test.head()
```

Out[17]:

DATE	Energy Production
2015-01-01	119.8260
2015-02-01	116.0253
2015-03-01	103.9265
2015-04-01	89.0847
2015-05-01	90.6408

In [18]:
```
len(test)
```

Out[18]: 44

Bước 2: Fit mô hình

In [19]:
```
stepwise_model.fit(train)
```

Out[19]:
```
ARIMA(callback=None, disp=0, maxiter=50, method=None, order=(2, 1, 2),
   out_of_sample_size=0, scoring='mse', scoring_args={},
   seasonal_order=(2, 1, 2, 12), solver='lbfgs', start_params=None,
   suppress_warnings=True, transparams=True, trend='c')
```

Bước 3: Dự đoán kết quả

In [20]:
```
future_forecast = stepwise_model.predict(n_periods=len(test))
```

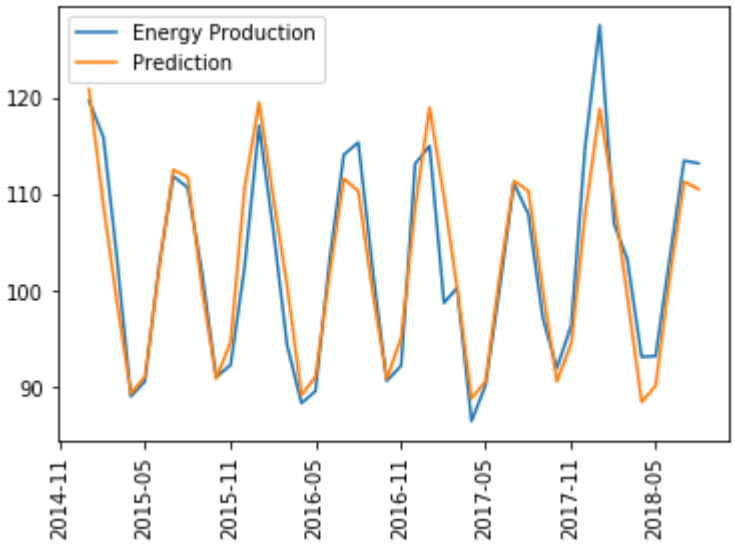
In [21]:
```
future_forecast
```

Out[21]:
```
array([121.02093743, 108.96541585,  99.25005531,  89.36263872,
        91.15240549, 102.58023544, 112.64660083, 111.8612734 ,
       100.16103254,  90.92434851,  94.73861988, 110.74551708,
       119.63790847, 109.64710782, 100.63934302,  89.21472724,
        91.14381236, 102.04387384, 111.72315674, 110.41734412,
        99.83636462,  90.89462894,  95.18341575, 108.79151397,
       119.09329406, 109.77761568, 100.34402343,  88.85838068,
        90.65731515, 101.87039198, 111.48665134, 110.42950596,
       100.05382187,  90.64954818,  94.59006641, 108.17197239,
       118.98115851, 109.48192681,  99.59711228,  88.49917059,
        90.17854912, 101.71774161, 111.40038319, 110.62959389])
```

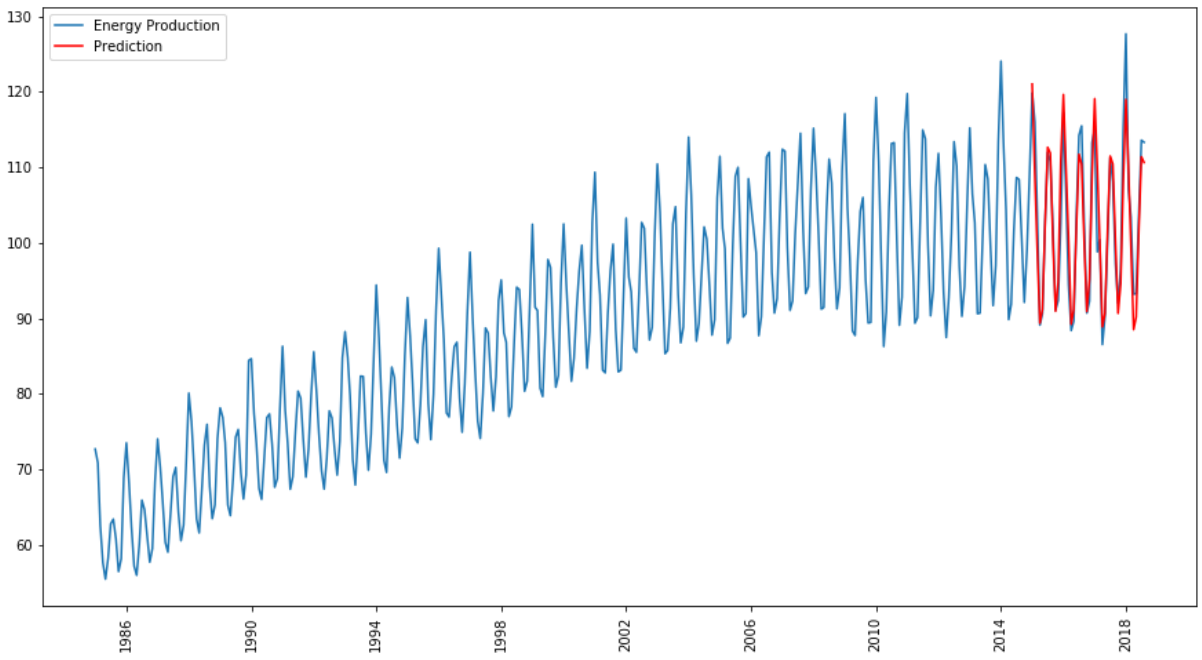
In [22]:
```
future_forecast = pd.DataFrame(future_forecast,index = test.index,columns=['Predic
```

Bước 4: Trực quan hóa dữ liệu

In [23]:
```
plt.plot(test, label='Energy Production')
plt.plot(future_forecast, label='Prediction')
plt.xticks(rotation='vertical')
plt.legend()
plt.show()
```



In [24]:
```
plt.figure(figsize=(15,8))
plt.plot(data_1985, label='Energy Production')
plt.plot(future_forecast, label='Prediction', color='red')
plt.xticks(rotation='vertical')
plt.legend()
plt.show()
```

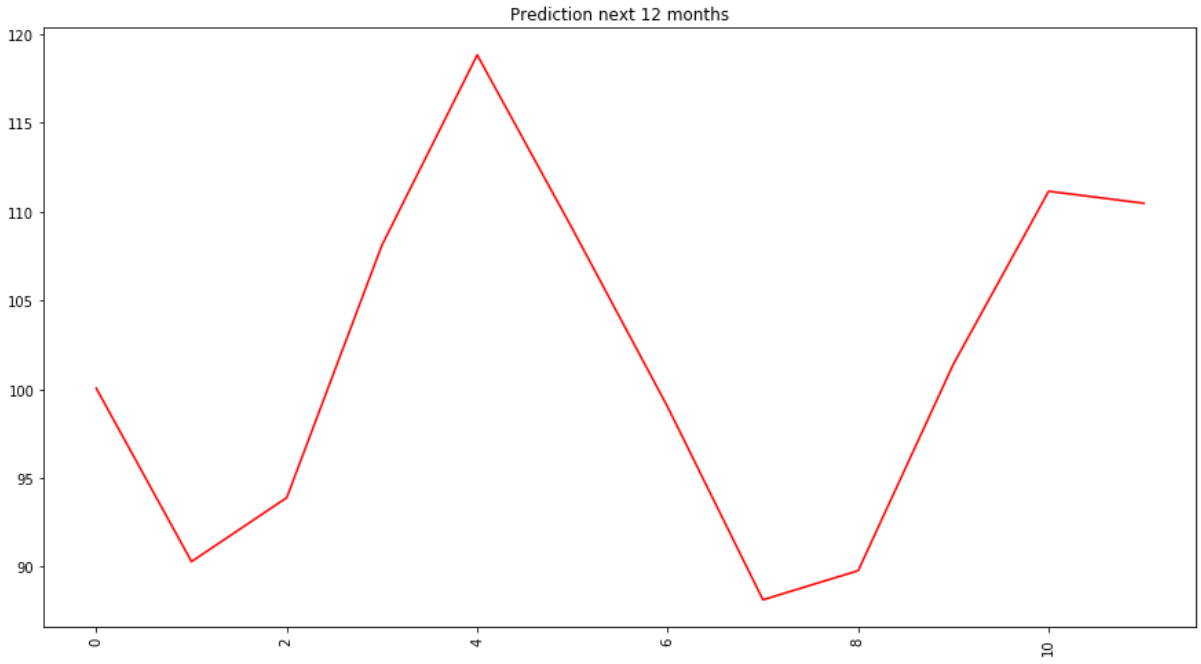


Dự đoán 12 tháng tiếp theo

In [25]:
```
future_forecast = stepwise_model.predict(n_periods=len(test)+12)
future_forecast
```

Out[25]:
```
array([121.02093743, 108.96541585,  99.25005531,  89.36263872,
        91.15240549, 102.58023544, 112.64660083, 111.8612734 ,
       100.16103254,  90.92434851,  94.73861988, 110.74551708,
       119.63790847, 109.64710782, 100.63934302,  89.21472724,
        91.14381236, 102.04387384, 111.72315674, 110.41734412,
        99.83636462,  90.89462894,  95.18341575, 108.79151397,
       119.09329406, 109.77761568, 100.34402343,  88.85838068,
        90.65731515, 101.87039198, 111.48665134, 110.42950596,
       100.05382187,  90.64954818,  94.59006641, 108.17197239,
       118.98115851, 109.48192681,  99.59711228,  88.49917059,
        90.17854912, 101.71774161, 111.40038319, 110.62959389,
       100.05827356,  90.29611652,  93.89578844, 108.12580355,
       118.83579264, 109.03648227,  99.00111142,  88.14316308,
        89.78421857, 101.42313235, 111.15783398, 110.48135914])
```

In [26]:
```
plt.figure(figsize=(15,8))
plt.plot(future_forecast[len(test):], color='red')
plt.xticks(rotation='vertical')
plt.title("Prediction next 12 months")
plt.show()
```



In [ ]:
```

```