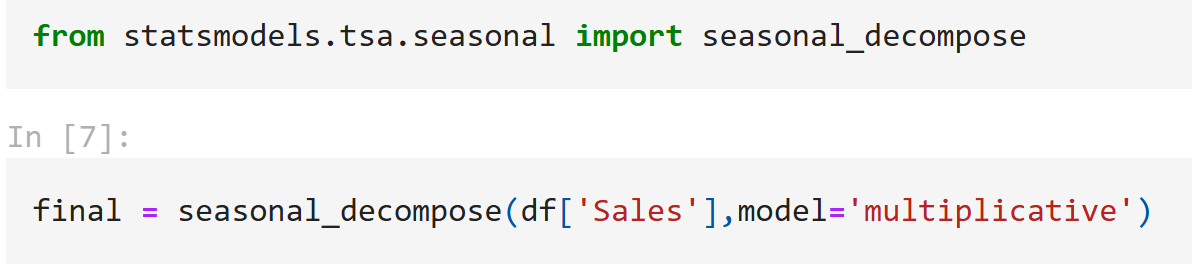
**Học phần Phân tích dữ liệu chuỗi thời gian**

**Bài thực hành 1.**

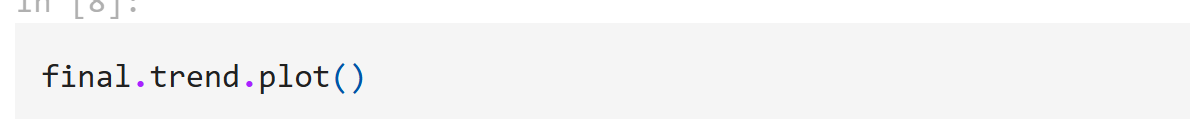
**Bài 1: Xử lý dữ liệu**

1/Phân tích dữ liệu: Cho Bộ dữ liệu **bán hàng**. Đây là dữ liệu chuỗi thời gian đơn biến nghĩa là nó chỉ có một biến. Cột chỉ mục được chỉ định là cột Tháng của bộ dữ liệu

2/ Khảo sát dữ liệu bằng hàm **seasonal\_decompose**



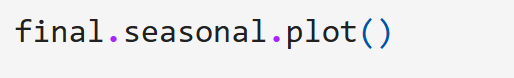
3/ Xác định xu hướng, đọc giải thích phân tích kết quả



A graph showing the growth of the year

Description automatically generated

* Bắt đầu từ tháng 7 2021 dữ liệu có xu hướng tăng theo thời gian.



A graph with blue lines

Description automatically generated

* Dữ liệu có tính mùa vụ theo, có thể chia thành nhiều chu kỳ và có tần suất cố định. VD có thể theo mỗi năm, hoặc mỗi tháng

**Bài 2: Xây dựng mô hình ARIMA, SARIMA cho bộ dữ liệu Điện năng**

ARIMA là viết tắt của 'Đường trung bình động tích hợp hồi quy tự động' được sử dụng trên dữ liệu chuỗi thời gian và cung cấp thông tin chi tiết về các giá trị trong quá khứ như độ trễ và lỗi dự báo có thể được sử dụng để dự báo các giá trị trong tương lai.

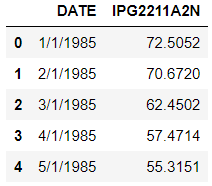
Các bước thực hiện **triển khai mô hình ARIMA để hỗ trợ dự báo/dự đoán** như sau:

1. Phân tích dữ liệu chuỗi thời gian bằng cách vẽ biểu đồ hoặc trực quan hóa dữ liệu đó.
2. Chuyển đổi dữ liệu chuỗi thời gian sang dạng dừng.
3. Vẽ biểu đồ ACF và PACF.
4. Xây dựng mô hình ARIMA.
5. Đưa ra dự đoán bằng cách sử dụng mô hình đã tạo.

**Triển khai sử dụng Python**

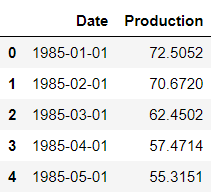
Yêu cầu Các gói cơ bản như NumPy và pandas để xử lý dữ liệu được nhập. Để trực quan hóa, matplotlib được sử dụng. Và để triển khai mô hình ARIMA, statsmodel được nhập.

**→ Phân tích dữ liệu**

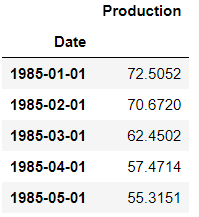
Dữ liệu được sử dụng là dữ liệu theo mùa về sản lượng điện.

**chuyển đổi thành chỉ mục DateTime.**

df['Date'] **=** pd.to\_datetime(df['Date'])   
 df.head()

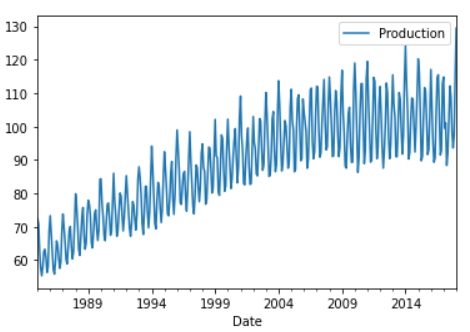


df.set\_index('Ngày',inplace **=True** )   
df.head()



**→ Trực quan hóa dữ liệu**

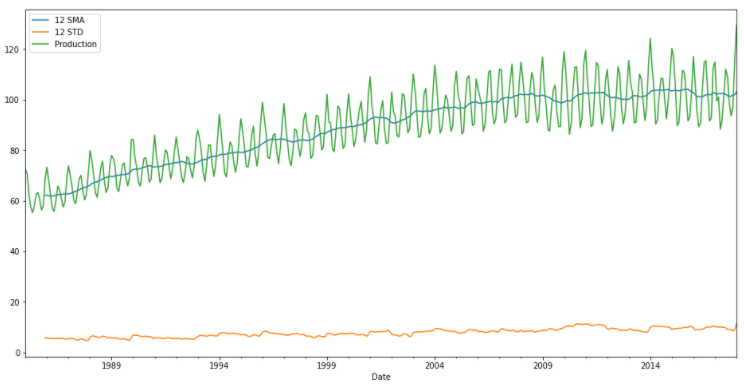
df.plot()



Như có thể thấy từ biểu đồ, đây là dữ liệu theo mùa vì nó có tính thời vụ và cũng có xu hướng tăng.

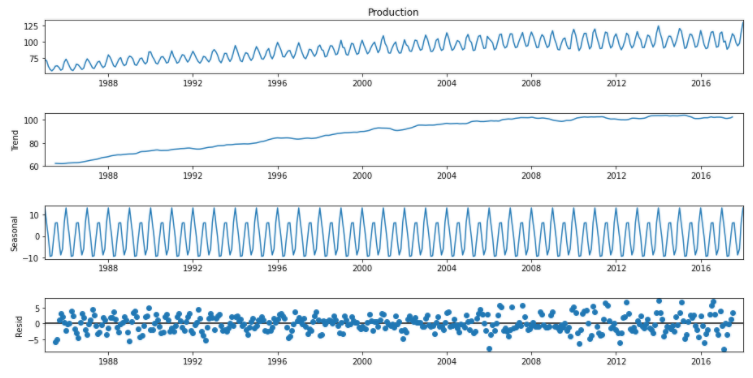
Bây giờ hãy **so sánh đường trung bình động đơn giản 12 tháng** cùng với chuỗi để xác định xu hướng. Độ lệch chuẩn cũng được vẽ để xem có bất kỳ phương sai nào hay không.

df[product].rolling(12).mean().plot(nhãn **=** '12 SMA',figsize **=** (16,8))   
 df[‘ product '].rolling(12).std().plot(nhãn **=** '12 STD')   
 df['product '].plot()   
 plt.legend()



Như có thể thấy từ biểu đồ trên, độ lệch chuẩn không thay đổi nhiều nên không có nhiều phương sai.

**→ Phân tích dữ liệu chuỗi thời gian thành xu hướng, tính theo mùa và các thành phần còn lại.**



Có thể xem riêng từng lỗi xu hướng, lỗi theo mùa và lỗi còn lại tại đây.

A screenshot of a graph

Description automatically generated

* Ở biểu đồ TREND có thể thấy rằng dữ liệu có xu hướng tăng theo thời gian, rõ rệt nhất là ở giai đoạn từ trước năm 1988 đến năm 2008, sau đó bắt đầu có xu hướng ổn định dần, không còn thay đổi quá nhiều qua thời gian nữa.
* Ở biểu đồ SEASON nhận thấy rằng dữ liệu có tính mùa vụ vì hình dạng biểu đồ lặp lại giống nhau qua từng giai đoạn, có thể là theo mỗi năm,…

**→ Chuyển đổi dữ liệu sang dạng tĩnh**

Đầu tiên, dữ liệu được kiểm tra bằng cách sử dụng kiểm tra **Dickey-Fuller** để kiểm tra xem dữ liệu có ở dạng dừng hay không, sau đó thay đổi dạng của dữ liệu.

Dickey-Fuller có giả thuyết không rằng có một gốc đơn vị và đây là một chuỗi thời gian không dừng. Giả thuyết thay thế là không có gốc đơn vị và chuỗi là dừng.

Chúng ta sẽ kiểm tra với sự trợ giúp của tham số **'p'.** Nếu p nhỏ (**p <= 0,05**), chúng ta bác bỏ giả thuyết không, nếu không thì không bác bỏ giả thuyết không.

Từ gói statsmodels, hàm kiểm tra **dickey-fuller**. Nó trả về một bộ bao gồm các giá trị: **adf, pvalue, usedlag, nobs, critical values, icbest và restore.**

from statsmodels.tsa.stattools

**import**  adfuller

Sau đó, hàm này được gọi trên cột sản xuất của tập dữ liệu.

fuller\_test **=** adfuller(df['product'])

fuller\_test(-2.256990350047239,

* Bây giờ giá trị p được in ra và sử dụng giá trị p để xác định xem dữ liệu có dừng hay không.

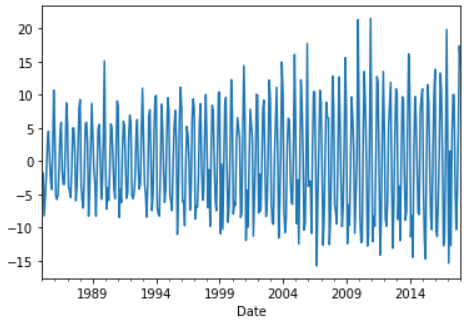
**def** test\_p\_value(data):   
 fuller\_test **=** adfuller(data)   
 print('Giá trị P: ',fuller\_test[1])   
 **if** fuller\_test[1] **<=** 0.05:   
 print('Bác bỏ giả thuyết null, dữ liệu là dừng')   
 **else** :   
 print('Không bác bỏ giả thuyết null, dữ liệu là không dừng')

test\_p\_value(df['product'])Giá trị P: 0,18621469116586975

#Không bác bỏ giả thuyết không, dữ liệu không dừng

Vì dữ liệu **không dừng nên phải thực hiện phép tính chênh lệch**. Chênh lệch là sự thay đổi của chuỗi thời gian từ giai đoạn này sang giai đoạn khác. Chênh lệch đầu tiên, chênh lệch thứ hai và chênh lệch theo mùa được tính toán và đối với mỗi chênh lệch, giá trị p được kiểm tra.

df['First\_diff'] **=** df['product'] **-** df['product'].shift(1)   
df['First\_diff'].plot()

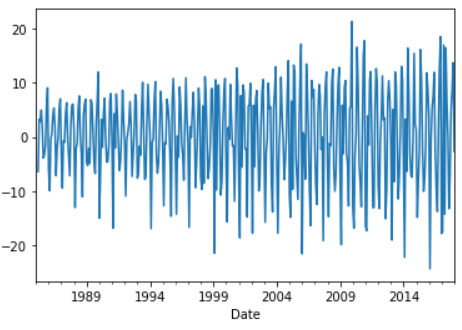


test\_p\_value(df['First\_diff'].dropna())

#Giá trị P: 4.0777865655398996e-10   
#Bác bỏ giả thuyết không, dữ liệu là dừng

Trong phép tính chênh lệch đầu tiên, chúng ta có được dữ liệu ở dạng dừng. Trong trường hợp cần phép tính chênh lệch thứ hai thì sẽ thực hiện như sau.

df['Second\_diff'] **=** df['First\_diff'] **-** df['First\_diff'].shift(1)   
df['Second\_diff'].plot()

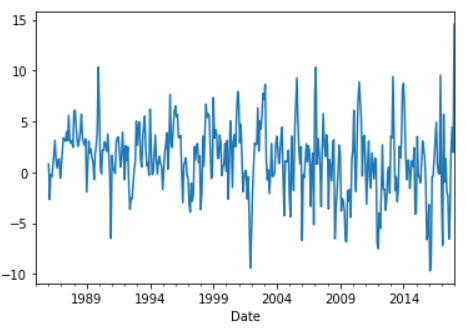


test\_p\_value(df['Second\_diff'].dropna())

#Giá trị P: 4.1836937480000375e-17   
#Bác bỏ giả thuyết không, dữ liệu là dừng

Sự khác biệt theo mùa cũng có thể được hiểu như sau. Sự thay đổi sẽ diễn ra trong toàn bộ một mùa là '12'.

df['Seasonal\_diff'] **=** df['product'] **-** df['product'].shift(12)   
df['Seasonal\_diff'].plot()



test\_p\_value(df['Seasonal\_diff'].dropna())

#Giá trị P: 8.812644938089026e-07   
#Bác bỏ giả thuyết không, dữ liệu là dừng

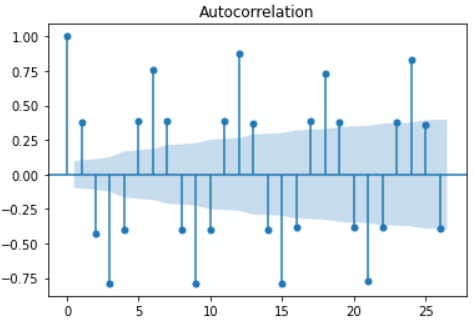
**→ Vẽ đồ thị ACF và PACF**

#Từ gói statsmodels, các hàm vẽ đồ thị ACF và PACF được nhập vào.

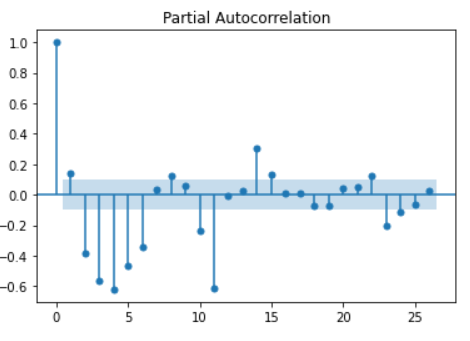
**From**  statsmodels.graphics.tsaplots

Import plot\_acf, plot\_pacf

first\_diff **=** plot\_acf(df['First\_diff'].dropna())

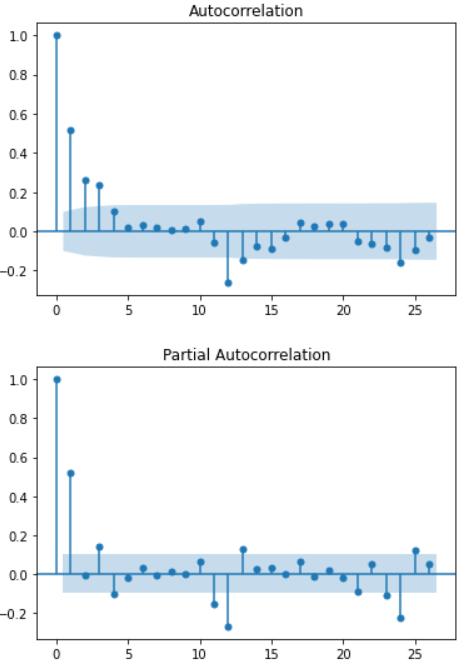


sec\_diff **=** plot\_pacf(df['Second\_diff'].dropna())



Bây giờ, các biểu đồ ACF và PACF cuối cùng sẽ được vẽ và sử dụng sau này.

p1 **=** plot\_acf(df['Seasonal\_diff'].dropna())   
 p2 **=** plot\_pacf(df['Seasonal\_diff'].dropna())



**→ Xây dựng mô hình ARIMA**

#Đối với dữ liệu không theo mùa, mô hình ARIMA có thể được nhập từ mô-đun #statsmodels.

**From**  statsmodels.tsa.arima.model

**import** ARIMA

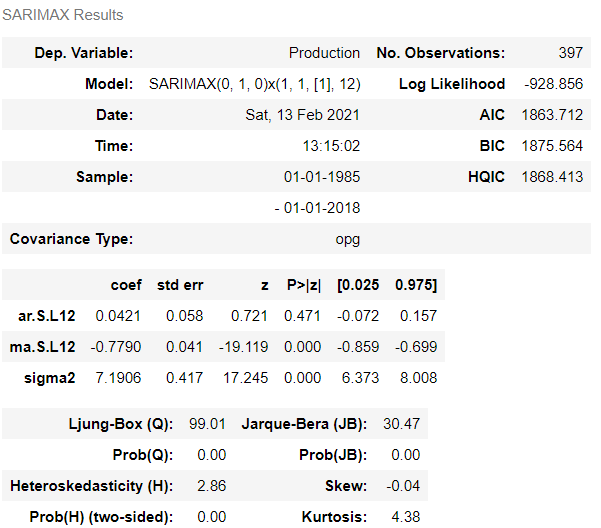
Đối với dữ liệu theo mùa, mô hình ARIMA theo mùa có thể được nhập từ mô-đun statsmodels. Dữ liệu được sử dụng ở đây là dữ liệu theo mùa nên mô hình ARIMA theo mùa được sử dụng. Các đối số được sử dụng là cột sản xuất, đơn hàng và đơn hàng theo mùa. thứ tự: order (p,d,q) của mô hình cho số lượng tham số AR, sự khác biệt và tham số MA.

Thứ tự theo mùa: Thứ tự (P,D,Q,s) của thành phần theo mùa của mô hình đối với các tham số AR, sự khác biệt, tham số MA và tính chu kỳ.

model **=** sm.tsa.statespace.SARIMAX(df['Production'],order**=**(0,1,0),seasonal\_order**=**(1,1,1,12))

Once the model is created, the next thing to do is to fit the model.

results **=** model.fit()  
results.summary()

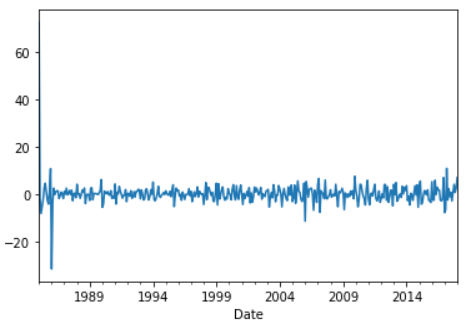


To know about the residuals values or error, the ‘resid’ method can be called on the results.

results.residDate  
1985-01-01 72.505200  
1985-02-01 -1.833200  
1985-03-01 -8.221800  
1985-04-01 -4.978800  
1985-05-01 -2.156300  
 ...   
2017-09-01 0.529985  
2017-10-01 4.057874  
2017-11-01 0.690663  
2017-12-01 2.477697  
2018-01-01 6.953533  
Length: 397, dtype: float64

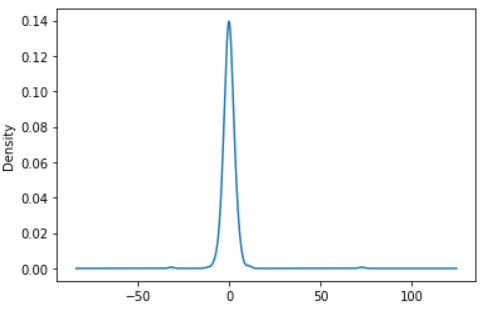
#The plot of the residual points can be created.

>>> results.resid.plot()



#The distribution of the errors can be seen by plotting the KDE. And as seen from the plot below, the errors are distributed around 0 which is good.

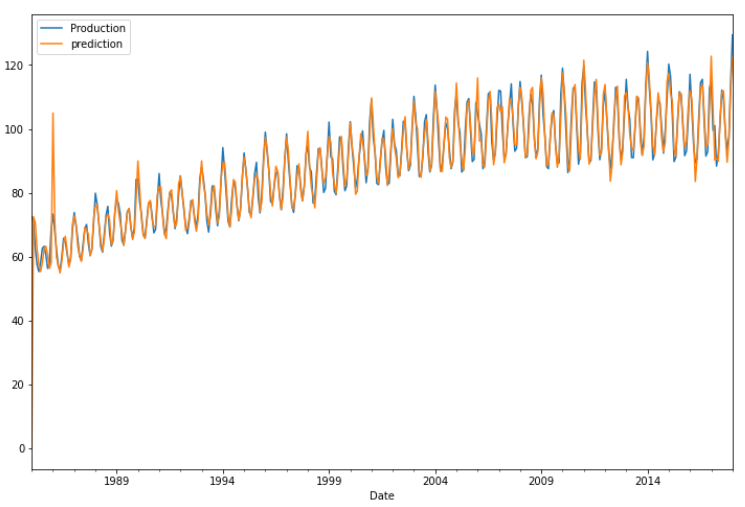
results.resid.plot(kind**=**'kde')



**→ Predicting or forecasting**

Bằng cách dự đoán các giá trị, hiệu suất của mô hình có thể được xác định. Đầu tiên, chúng ta có thể xem xét cách nó dự đoán dữ liệu hiện tại và sau đó chuyển sang dự đoán dữ liệu trong tương lai.

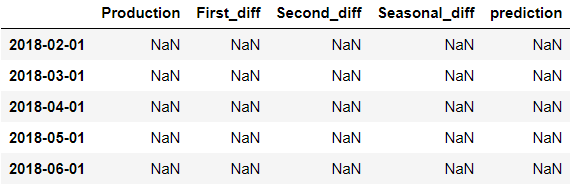
df['prediction'] **=** results.predict()  
df[['Production','prediction']].plot(figsize**=**(12,8))



Như đã thấy từ sơ đồ trên, mô hình thực hiện tốt công việc dự đoán dữ liệu hiện tại. Bây giờ để dự đoán cho tương lai, chúng ta có thể thêm nhiều tháng hơn vào tập dữ liệu với các giá trị null và dự đoán cho nó. Điều này có thể được thực hiện bằng pandas. Chỉ số cuối cùng được lấy là ngày cuối cùng và một độ lệch tháng được thêm vào đó bắt đầu từ 1 và tăng lên đến 24.

**from** pandas.tseries.offsets **import** DateOffset  
extra\_dates **=** [df.index[**-**1] **+** DateOffset(months**=**m) **for** m **in** range (1,24)]>>> extra\_dates  
[Timestamp('2018-02-01 00:00:00'),  
 Timestamp('2018-03-01 00:00:00'),  
 Timestamp('2018-04-01 00:00:00'),  
 Timestamp('2018-05-01 00:00:00'),  
 Timestamp('2018-06-01 00:00:00'),  
 Timestamp('2018-07-01 00:00:00'),  
 Timestamp('2018-08-01 00:00:00'),  
 Timestamp('2018-09-01 00:00:00'),  
 Timestamp('2018-10-01 00:00:00'),  
 Timestamp('2018-11-01 00:00:00'),  
 Timestamp('2018-12-01 00:00:00'),  
 Timestamp('2019-01-01 00:00:00'),  
 Timestamp('2019-02-01 00:00:00'),  
 Timestamp('2019-03-01 00:00:00'),  
 Timestamp('2019-04-01 00:00:00'),  
 Timestamp('2019-05-01 00:00:00'),  
 Timestamp('2019-06-01 00:00:00'),  
 Timestamp('2019-07-01 00:00:00'),  
 Timestamp('2019-08-01 00:00:00'),  
 Timestamp('2019-09-01 00:00:00'),  
 Timestamp('2019-10-01 00:00:00'),  
 Timestamp('2019-11-01 00:00:00'),  
 Timestamp('2019-12-01 00:00:00')]

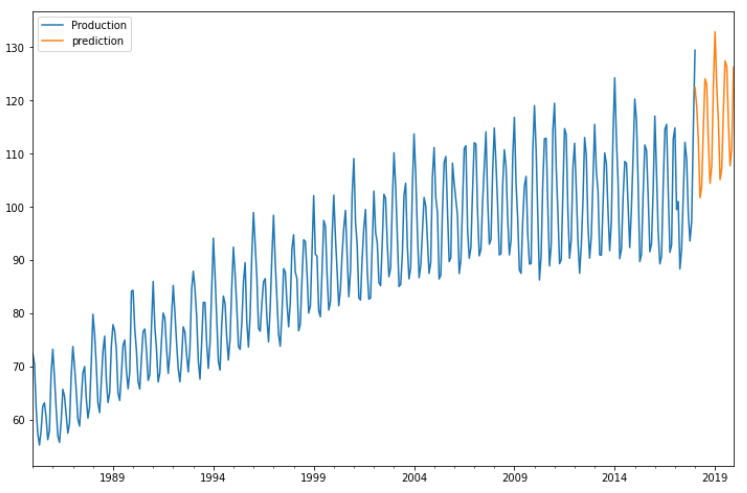
forecast\_df **=** pd.DataFrame(index**=**extra\_dates,columns**=**df.columns)  
 forecast\_df.head()



final\_df **=** pd.concat([df,forecast\_df])

**Dự đoán giá trị cho các điểm dữ liệu cuối bằng cách truyền đối số bắt đầu và kết thúc**

final\_df['prediction'] **=** results.predict(start**=**396, end**=**430)  
final\_df[['Production','prediction']].plot(figsize**=**(12,8))



Bài 3: Áp dụng các mô hình dự báo phù hợp cho dữ liệu cổ phiếu ô tô stock.csv