Ex 3: Candy Production

Cho dữ liệu sản xuất kẹo từ 1972 đến 2017 trong tập tin candy_production.csv

- Thực hiện việc dự báo sản xuất sản phẩm sử dụng thuật toán ARIMA
- Cho biết trong 12 tháng sau những năm trên thì giá trị bán sản phẩm như thế nào?

In [1]:
```
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
```

In [2]:
```
# %cd '/content/gdrive/My Drive/LDS6_MachineLearning/practice/Chapter18_ARIMA/'
```

In [3]:
```
import pandas as pd
```

In [4]:
```
data = pd.read_csv("candy_production.csv", index_col=0)
data.head()
```

Out[4]:

observation_date	IPG3113N
1972-01-01	85.6945
1972-02-01	71.8200
1972-03-01	66.0229
1972-04-01	64.5645
1972-05-01	65.0100

In [5]:
```
data.index = pd.to_datetime(data.index)
```

In [6]:
```
data.index.freq = 'MS'
```

In [7]:
```
data.index
```

Out[7]:
```
DatetimeIndex(['1972-01-01', '1972-02-01', '1972-03-01', '1972-04-01',
               '1972-05-01', '1972-06-01', '1972-07-01', '1972-08-01',
               '1972-09-01', '1972-10-01',
               ...
               '2016-11-01', '2016-12-01', '2017-01-01', '2017-02-01',
               '2017-03-01', '2017-04-01', '2017-05-01', '2017-06-01',
               '2017-07-01', '2017-08-01'],
              dtype='datetime64[ns]', name='observation_date', length=548, freq='MS')
```

In [8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 548 entries, 1972-01-01 to 2017-08-01
Freq: MS
Data columns (total 1 columns):
IPG3113N    548 non-null float64
dtypes: float64(1)
memory usage: 8.6 KB
```

In [9]:
```
data.columns = ['candy_production']
```

In [10]:
```
data.head()
```

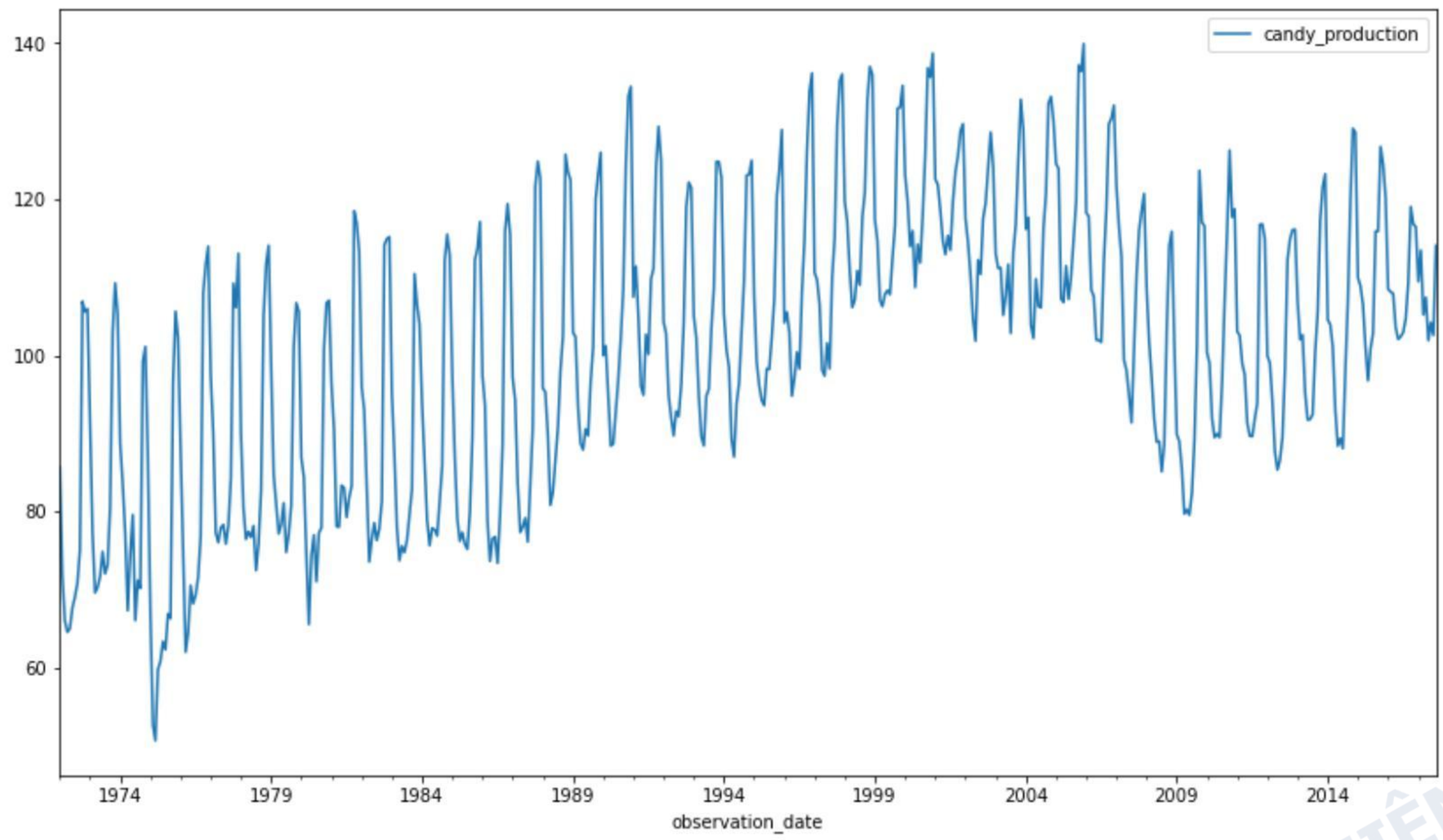
Out[10]:

observation_date	candy_production
1972-01-01	85.6945
1972-02-01	71.8200
1972-03-01	66.0229
1972-04-01	64.5645
1972-05-01	65.0100

In [11]:
```
import matplotlib.pyplot as plt
```

In [12]:
```
data.plot(figsize=(14,8))
```

Out[12]: <AxesSubplot:xlabel='observation_date'>



In [13]:
```
type(data)
```

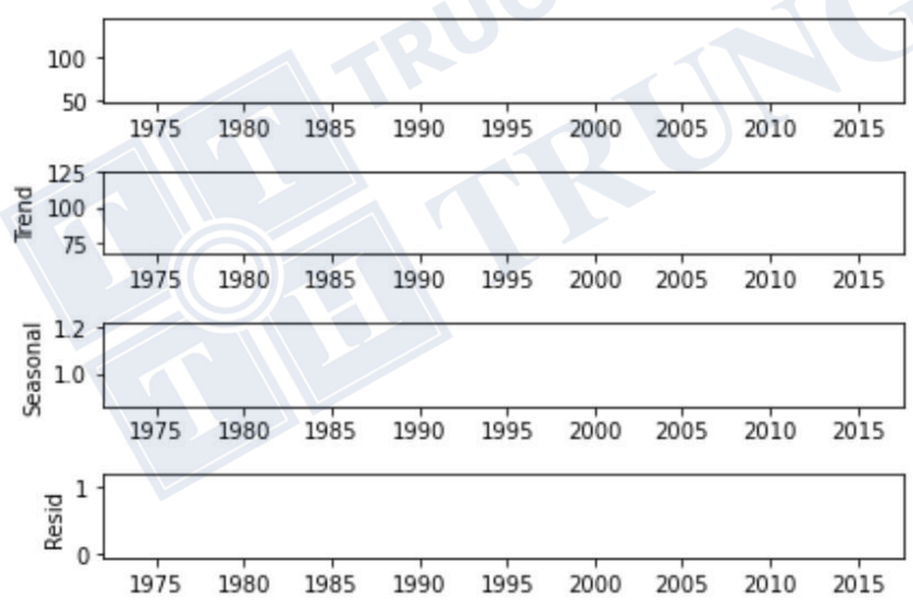
Out[13]: pandas.core.frame.DataFrame

In [14]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
result = seasonal_decompose(x = data, model='multiplicative')
result
```

Out[14]: <statsmodels.tsa.seasonal.DecomposeResult at 0x1bbbddc2550>

In [15]:
```
result.plot()
plt.show()
```

```
c:\program files\python36\lib\site-packages\pandas\plotting\_matplotlib\converter.py:245: MatplotlibDeprecationWarning:
The epoch2num function was deprecated in Matplotlib 3.3 and will be removed two minor releases later.
  base = dates.epoch2num(dt.asi8 / 1.0e9)
```



In [16]:
```
#! pip install pmdarima
```

In [17]:
```
from pmdarima import auto_arima
```

In [18]:
```
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)
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,1,1)[12]             : AIC=2955.432, Time=3.13 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=3177.330, Time=0.02 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=3067.080, Time=0.20 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=2969.534, Time=0.40 sec
 ARIMA(2,1,2)(0,1,1)[12]             : AIC=2960.282, Time=1.89 sec
 ARIMA(2,1,2)(1,1,0)[12]             : AIC=inf, Time=2.25 sec
 ARIMA(2,1,2)(2,1,1)[12]             : AIC=2956.153, Time=10.27 sec
 ARIMA(2,1,2)(1,1,2)[12]             : AIC=2956.787, Time=9.57 sec
 ARIMA(2,1,2)(0,1,0)[12]             : AIC=inf, Time=0.66 sec
 ARIMA(2,1,2)(0,1,2)[12]             : AIC=2954.852, Time=7.14 sec
 ARIMA(1,1,2)(0,1,2)[12]             : AIC=2951.313, Time=4.51 sec
 ARIMA(1,1,2)(0,1,1)[12]             : AIC=2957.418, Time=1.60 sec
 ARIMA(1,1,2)(1,1,2)[12]             : AIC=2953.311, Time=6.35 sec
 ARIMA(1,1,2)(1,1,1)[12]             : AIC=2951.707, Time=2.22 sec
 ARIMA(0,1,2)(0,1,2)[12]             : AIC=2958.148, Time=2.05 sec
 ARIMA(1,1,1)(0,1,2)[12]             : AIC=2954.325, Time=2.67 sec
 ARIMA(1,1,3)(0,1,2)[12]             : AIC=2951.129, Time=6.71 sec
 ARIMA(1,1,3)(0,1,1)[12]             : AIC=inf, Time=3.05 sec
 ARIMA(1,1,3)(1,1,2)[12]             : AIC=2953.126, Time=11.51 sec
 ARIMA(1,1,3)(1,1,1)[12]             : AIC=2951.415, Time=3.11 sec
 ARIMA(0,1,3)(0,1,2)[12]             : AIC=2960.055, Time=2.61 sec
 ARIMA(2,1,3)(0,1,2)[12]             : AIC=2949.061, Time=7.01 sec
 ARIMA(2,1,3)(0,1,1)[12]             : AIC=inf, Time=3.19 sec
 ARIMA(2,1,3)(1,1,2)[12]             : AIC=2951.061, Time=11.66 sec
 ARIMA(2,1,3)(1,1,1)[12]             : AIC=2949.457, Time=4.21 sec
 ARIMA(3,1,3)(0,1,2)[12]             : AIC=inf, Time=13.21 sec
 ARIMA(2,1,4)(0,1,2)[12]             : AIC=2952.228, Time=12.31 sec
 ARIMA(1,1,4)(0,1,2)[12]             : AIC=2951.543, Time=7.16 sec
 ARIMA(3,1,2)(0,1,2)[12]             : AIC=2949.561, Time=8.71 sec
 ARIMA(3,1,4)(0,1,2)[12]             : AIC=inf, Time=16.53 sec
 ARIMA(2,1,3)(0,1,2)[12] intercept   : AIC=inf, Time=13.17 sec

Best model:  ARIMA(2,1,3)(0,1,2)[12]
Total fit time: 179.067 seconds
```

In [19]:
```
print(stepwise_model.aic())
```

```
2949.061078238775
```

In [20]:
```
train = data.loc['1972-01-01':'2009-12-01']
test = data.loc['2009-12-01':]
```

In [21]:
```
test
```

Out[21]:

observation_date	candy_production
2009-12-01	116.5435
2010-01-01	100.3797
2010-02-01	99.0155
2010-03-01	91.9654
2010-04-01	89.4914
...	...
2017-04-01	107.4288
2017-05-01	101.9209
2017-06-01	104.2022
2017-07-01	102.5861
2017-08-01	114.0613

93 rows × 1 columns

In [22]:
```
len(test)
```

Out[22]: 93

In [23]:
```
len(train)
```

Out[23]: 456

In [24]:
```
stepwise_model.fit(train)
```

Out[24]:
```
ARIMA(order=(2, 1, 3), scoring_args={}, seasonal_order=(0, 1, 2, 12),
      suppress_warnings=True, with_intercept=False)
```

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

In [26]:
```
future_forecast
```

Out[26]:
```
array([102.13211266, 100.13213126,  95.28694666,  88.79930746,
        89.64604485,  88.6238308 ,  88.81275702,  96.51020015,
       105.27723551, 121.83806119, 119.38733708, 118.26943919,
       105.1182907 , 101.85950122,  97.03431377,  90.68034887,
        90.81467504,  90.608315  ,  89.27341719,  97.35147057,
       105.69478874, 120.06515434, 119.5472295 , 116.9252781 ,
       104.50967307, 101.16107329,  95.95276167,  90.29731377,
        89.68479953,  89.99870909,  88.55201676,  96.3075675 ,
       105.27373708, 118.95974706, 118.93515196, 116.18425982,
       103.49948883, 100.70505237,  94.87074005,  89.68155903,
        88.92713193,  89.01866278,  88.06388778,  95.24780982,
       104.65334129, 118.18805017, 117.98198654, 115.66670227,
       102.46085171, 100.07926131,  94.08731991,  88.75231714,
        88.38264899,  87.99999587,  87.43212818,  94.45469015,
       103.74534231, 117.61897487, 116.98214476, 115.02855631,
       101.65980279,  99.19007972,  93.49582186,  87.77017075,
        87.73783175,  87.19256096,  86.55957316,  93.8427827 ,
       102.77978515, 116.96731424, 116.16966415, 114.17065258,
       101.02935046,  98.24003579,  92.83724093,  86.95380365,
        86.89280161,  86.54528703,  85.62400177,  93.17728414,
       101.96052959, 116.13356082, 115.50715829, 113.24855232,
       100.35700293,  97.4187466 ,  92.01333256,  86.27752955,
        85.98321321,  85.86621315,  84.80140694,  92.3619398 ,
       101.27183248])
```

In [27]:
```
future_forecast = pd.DataFrame(future_forecast,
                               index = test.index,
                               columns=['Prediction'])
```

In [28]:
```
from sklearn.metrics import mean_absolute_error
```

In [29]:
```
mean_absolute_error(test['candy_production'], future_forecast)
```

Out[29]:
```
8.524769812375583
```

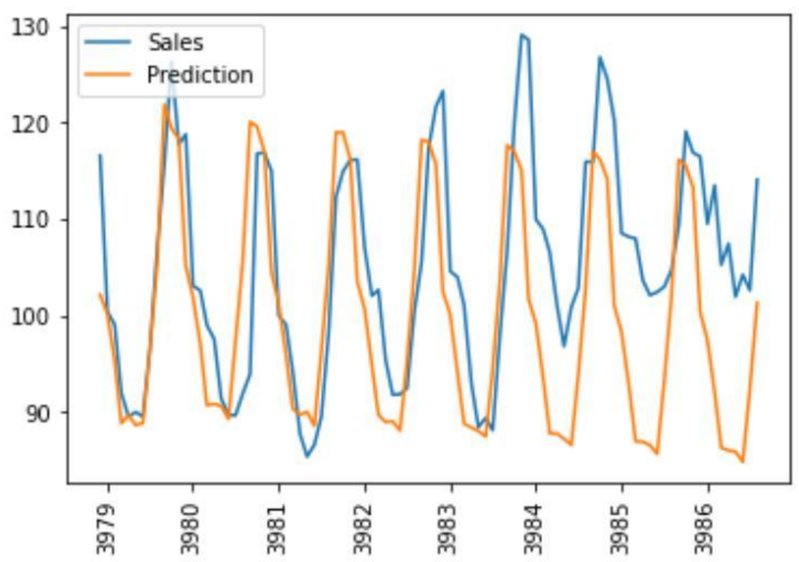
In [30]:
```
future_forecast = pd.DataFrame(future_forecast,index = test.index,columns=['Prediction'])
df_merge = test.join(future_forecast)
df_merge.head()
```

Out[30]:

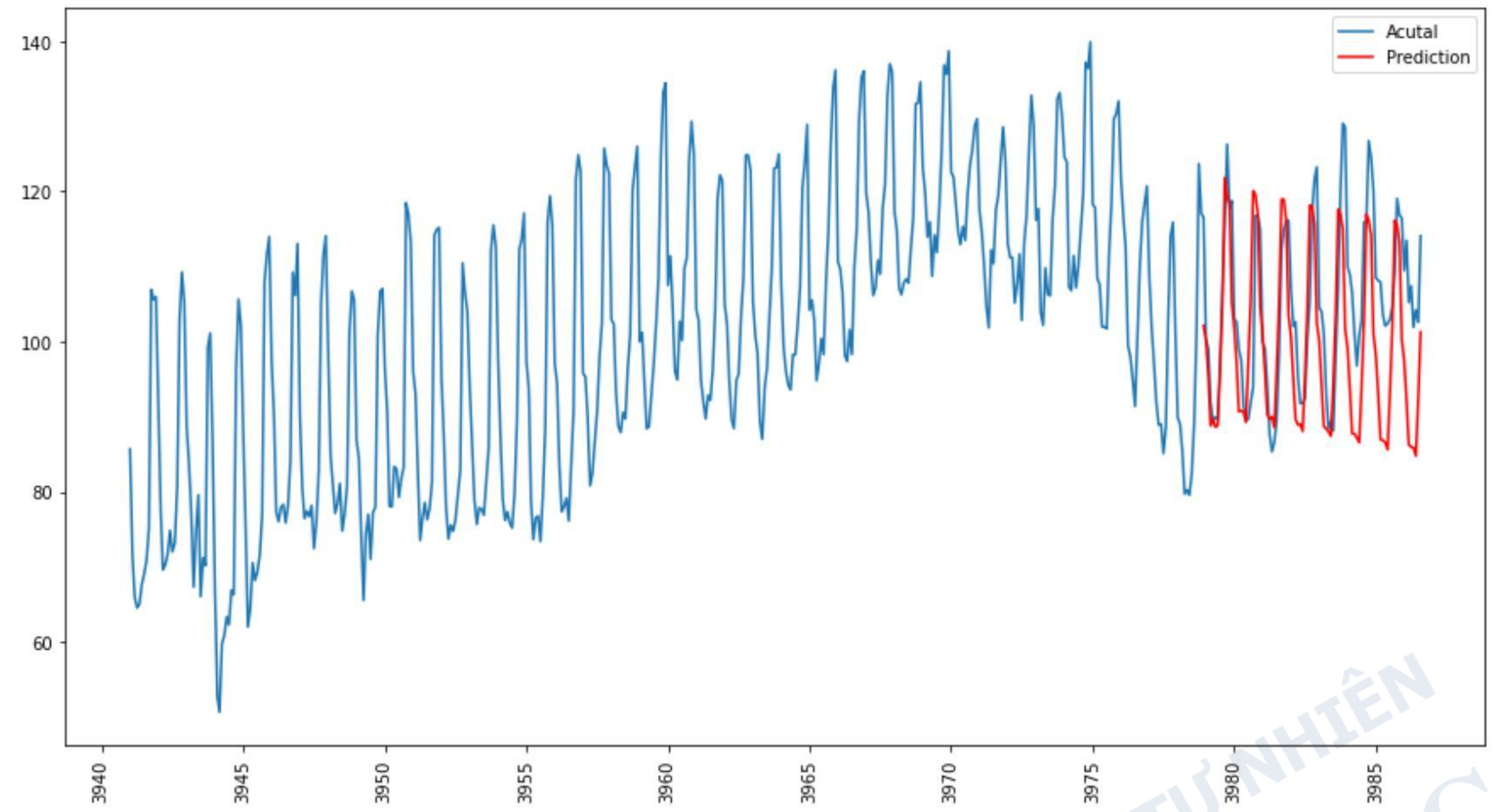
observation_date	candy_production	Prediction
2009-12-01	116.5435	102.132113
2010-01-01	100.3797	100.132131
2010-02-01	99.0155	95.286947
2010-03-01	91.9654	88.799307
2010-04-01	89.4914	89.646045

In [31]:
```
plt.plot(test, label='Sales')
plt.plot(future_forecast, label='Prediction')
plt.xticks(rotation='vertical')
plt.legend()
plt.show()
```

```
c:\program files\python36\lib\site-packages\pandas\plotting\_matplotlib\converter.py:245: MatplotlibDeprecationWarning:
The epoch2num function was deprecated in Matplotlib 3.3 and will be removed two minor releases later.
  base = dates.epoch2num(dt.asi8 / 1.0e9)
```



In [32]:
```
plt.figure(figsize=(15,8))
plt.plot(data, label='Acutal')
plt.plot(future_forecast, label='Prediction', color='red')
plt.xticks(rotation='vertical')
plt.legend()
plt.show()
```



In [33]:
```
# Dự đoán 12 tháng sau
```

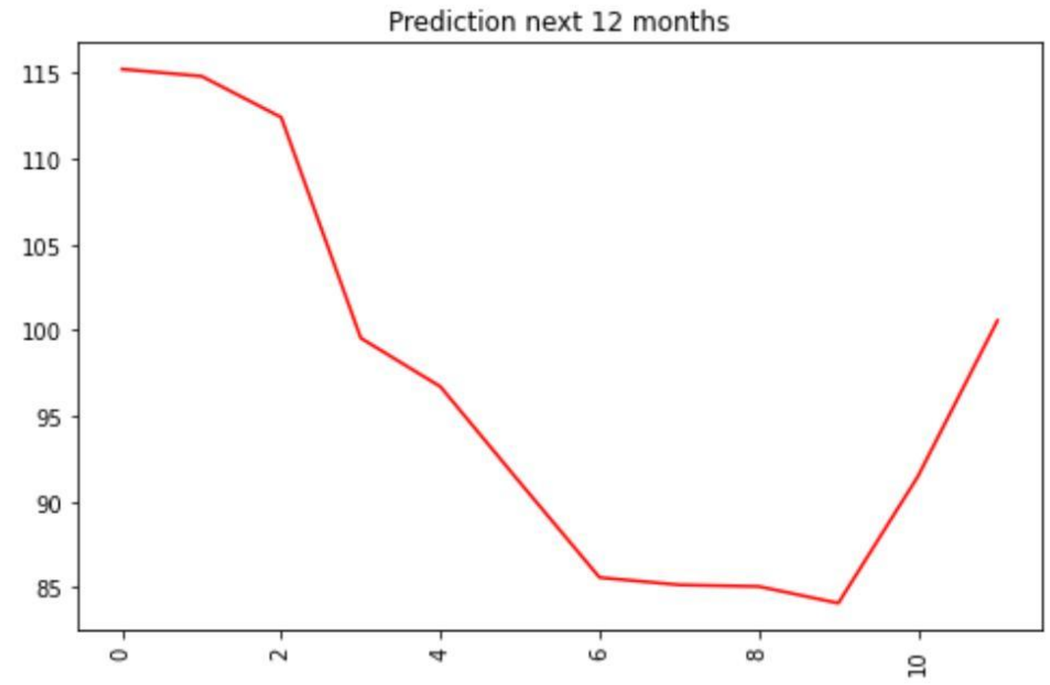
In [34]:
```
future_forecast = stepwise_model.predict(n_periods=len(test)+12)
future_forecast
```

Out[34]:
```
array([102.13211266, 100.13213126,  95.28694666,  88.79930746,
        89.64604485,  88.6238308 ,  88.81275702,  96.51020015,
       105.27723551, 121.83806119, 119.38733708, 118.26943919,
       105.1182907 , 101.85950122,  97.03431377,  90.68034887,
        90.81467504,  90.608315  ,  89.27341719,  97.35147057,
       105.69478874, 120.06515434, 119.5472295 , 116.9252781 ,
       104.50967307, 101.16107329,  95.95276167,  90.29731377,
        89.68479953,  89.99870909,  88.55201676,  96.3075675 ,
       105.27373708, 118.95974706, 118.93515196, 116.18425982,
       103.49948883, 100.70505237,  94.87074005,  89.68155903,
        88.92713193,  89.01866278,  88.06388778,  95.24780982,
       104.65334129, 118.18805017, 117.98198654, 115.66670227,
       102.46085171, 100.07926131,  94.08731991,  88.75231714,
        88.38264899,  87.99999587,  87.43212818,  94.45469015,
       103.74534231, 117.61897487, 116.98214476, 115.02855631,
       101.65980279,  99.19007972,  93.49582186,  87.77017075,
        87.73783175,  87.19256096,  86.55957316,  93.8427827 ,
       102.77978515, 116.96731424, 116.16966415, 114.17065258,
       101.02935046,  98.24003579,  92.83724093,  86.95380365,
        86.89280161,  86.54528703,  85.62400177,  93.17728414,
       101.96052959, 116.13356082, 115.50715829, 113.24855232,
       100.35700293,  97.4187466 ,  92.01333256,  86.27752955,
        85.98321321,  85.86621315,  84.80140694,  92.3619398 ,
       101.27183248, 115.23556918, 114.82152429, 112.42526771,
        99.54907873,  96.71886061,  91.12606792,  85.58553617,
        85.15975584,  85.06468986,  84.09146229,  91.48457837,
       100.57370712])
```

In [39]:
```
future_forecast[len(test):]
```

Out[39]:
```
array([115.23556918, 114.82152429, 112.42526771,  99.54907873,
        96.71886061,  91.12606792,  85.58553617,  85.15975584,
        85.06468986,  84.09146229,  91.48457837, 100.57370712])
```

In [35]:
```
plt.figure(figsize=(8,5))
plt.plot(future_forecast[len(test):], color='red')
plt.xticks(rotation='vertical')
plt.title("Prediction next 12 months")
plt.show()
```



In [36]:
```
import numpy as np
```

In [37]:
```
plt.plot(np.arange(len(test)+12), future_forecast, color='green')
plt.plot(np.arange(len(test), len(test)+12, 1),future_forecast[len(test):], color='red')
plt.xticks(rotation='vertical')
plt.title("Prediction next 12 months")
plt.show()
```