ARIMA and Seasonal ARIMA

Autoregressive Integrated Moving Averages

The general process for ARIMA models is the following:

- Visualize the Time Series Data
- Make the time series data stationary
- Plot the Correlation and AutoCorrelation Charts
- Construct the ARIMA Model or Seasonal ARIMA based on the data
- Use the model to make predictions

Let's go through these steps!

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

import matplotlib.pyplot as plt
%matplotlib inline
```

In [4]:
```
df=pd.read_csv('perrin-freres-monthly-champagne-.csv')
```

In [5]:
```
df.head()
```

Out [5]:

	Month	Perrin Freres monthly champagne sales millions ?64-?72
0	1964-01	2815.0
1	1964-02	2672.0
2	1964-03	2755.0
3	1964-04	2721.0
4	1964-05	2946.0

In [6]:
```
df.tail()
```

Out [6]:

	Month	Perrin Freres monthly champagne sales millions ?64-?72
102	1972-07	4298.0
103	1972-08	1413.0
104	1972-09	5877.0
105	NaN	NaN
106	Perrin Freres monthly champagne sales millions...	NaN

In [7]:
```
## Cleaning up the data
df.columns=["Month","Sales"]
df.head()
```

Out [7]:

	Month	Sales
0	1964-01	2815.0

	Month	Sales
1	1964-02	2672.0
2	1964-03	2755.0
3	1964-04	2721.0
4	1964-05	2946.0

In [8]:
```
## Drop last 2 rows
df.drop(106,axis=0,inplace=True)
```

In [9]:
```
df.tail()
```

Out [9]:

	Month	Sales
101	1972-06	5312.0
102	1972-07	4298.0
103	1972-08	1413.0
104	1972-09	5877.0
105	NaN	NaN

In [10]:
```
df.drop(105,axis=0,inplace=True)
```

In [11]:
```
df.tail()
```

Out [11]:

	Month	Sales
100	1972-05	4618.0
101	1972-06	5312.0
102	1972-07	4298.0
103	1972-08	1413.0
104	1972-09	5877.0

In [12]:
```
# Convert Month into Datetime
df['Month']=pd.to_datetime(df['Month'])
```

In [13]:
```
df.head()
```

Out [13]:

	Month	Sales
0	1964-01-01	2815.0
1	1964-02-01	2672.0
2	1964-03-01	2755.0
3	1964-04-01	2721.0
4	1964-05-01	2946.0

In [14]:
```
df.set_index('Month',inplace=True)
```

In [15]:
```
df.head()
```

Out [15]:

	Sales
Month	
1964-01-01	2815.0
1964-02-01	2672.0
1964-03-01	2755.0
1964-04-01	2721.0
1964-05-01	2946.0

In [16]:
```
df.describe()
```

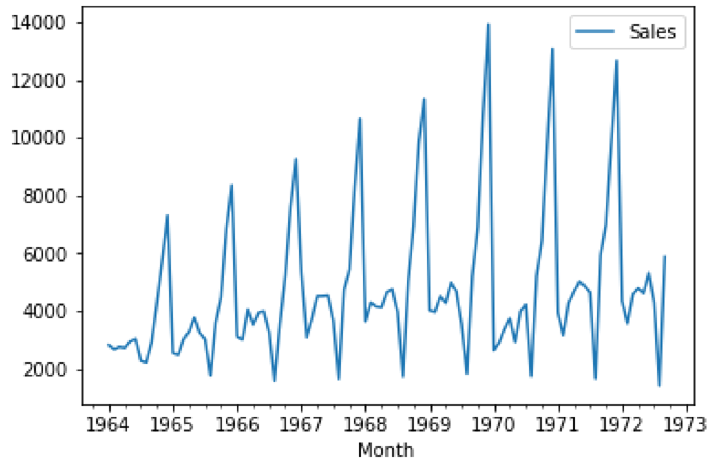
Out [16]:

	Sales
count	105.000000
mean	4761.152381
std	2553.502601
min	1413.000000
25%	3113.000000
50%	4217.000000
75%	5221.000000
max	13916.000000

Step 2: Visualize the Data

In [17]:
```
df.plot()
```

Out [17]: <matplotlib.axes._subplots.AxesSubplot at 0x1d2c881a2e8>



In [18]:
```
### Testing For Stationarity

from statsmodels.tsa.stattools import adfuller
```

In [19]:
```
test_result=adfuller(df['Sales'])
```

In [20]:
```python
#Ho: It is non stationary
#H1: It is stationary

def adfuller_test(sales):
    result=adfuller(sales)
    labels = ['ADF Test Statistic','p-value','#Lags Used','Number of Observations Used'
    for value,label in zip(result,labels):
        print(label+' : '+str(value) )
    if result[1] <= 0.05:
        print("strong evidence against the null hypothesis(Ho), reject the null hypothe
    else:
        print("weak evidence against null hypothesis, time series has a unit root, indi
```

In [21]:
```python
adfuller_test(df['Sales'])
```

```
ADF Test Statistic : -1.8335930563276297
p-value : 0.3639157716602417
#Lags Used : 11
Number of Observations Used : 93
weak evidence against null hypothesis, time series has a unit root, indicating it is non-stationary
```

Differencing

In [22]:
```python
df['Sales First Difference'] = df['Sales'] - df['Sales'].shift(1)
```

In [212]:
```python
df['Sales'].shift(1)
```

Out [212]:
```
Month
1964-01-01       NaN
1964-02-01    2815.0
1964-03-01    2672.0
1964-04-01    2755.0
1964-05-01    2721.0
1964-06-01    2946.0
1964-07-01    3036.0
1964-08-01    2282.0
1964-09-01    2212.0
1964-10-01    2922.0
1964-11-01    4301.0
1964-12-01    5764.0
1965-01-01    7312.0
1965-02-01    2541.0
1965-03-01    2475.0
1965-04-01    3031.0
1965-05-01    3266.0
1965-06-01    3776.0
1965-07-01    3230.0
1965-08-01    3028.0
1965-09-01    1759.0
1965-10-01    3595.0
1965-11-01    4474.0
1965-12-01    6838.0
1966-01-01    8357.0
1966-02-01    3113.0
1966-03-01    3006.0
1966-04-01    4047.0
1966-05-01    3523.0
1966-06-01    3937.0
                 ...
1970-04-01    3370.0
1970-05-01    3740.0
1970-06-01    2927.0
1970-07-01    3986.0
1970-08-01    4217.0
```

```
1970-09-01     1738.0
1970-10-01     5221.0
1970-11-01     6424.0
1970-12-01     9842.0
1971-01-01    13076.0
1971-02-01     3934.0
1971-03-01     3162.0
1971-04-01     4286.0
1971-05-01     4676.0
1971-06-01     5010.0
1971-07-01     4874.0
1971-08-01     4633.0
1971-09-01     1659.0
1971-10-01     5951.0
1971-11-01     6981.0
1971-12-01     9851.0
1972-01-01    12670.0
1972-02-01     4348.0
1972-03-01     3564.0
1972-04-01     4577.0
1972-05-01     4788.0
1972-06-01     4618.0
1972-07-01     5312.0
1972-08-01     4298.0
1972-09-01     1413.0
Name: Sales, Length: 105, dtype: float64
```

In [188]:

```
df['Seasonal First Difference']=df['Sales']-df['Sales'].shift(12)
```

In [190]:

```
df.head(14)
```

Out [190]:

Month	Sales	Sales First Difference	forecast	Seasonal First Difference
1964-01-01	2815.0	NaN	NaN	NaN
1964-02-01	2672.0	-143.0	NaN	NaN
1964-03-01	2755.0	83.0	NaN	NaN
1964-04-01	2721.0	-34.0	NaN	NaN
1964-05-01	2946.0	225.0	NaN	NaN
1964-06-01	3036.0	90.0	NaN	NaN
1964-07-01	2282.0	-754.0	NaN	NaN
1964-08-01	2212.0	-70.0	NaN	NaN
1964-09-01	2922.0	710.0	NaN	NaN
1964-10-01	4301.0	1379.0	NaN	NaN
1964-11-01	5764.0	1463.0	NaN	NaN
1964-12-01	7312.0	1548.0	NaN	NaN
1965-01-01	2541.0	-4771.0	NaN	-274.0
1965-02-01	2475.0	-66.0	NaN	-197.0

In [192]:

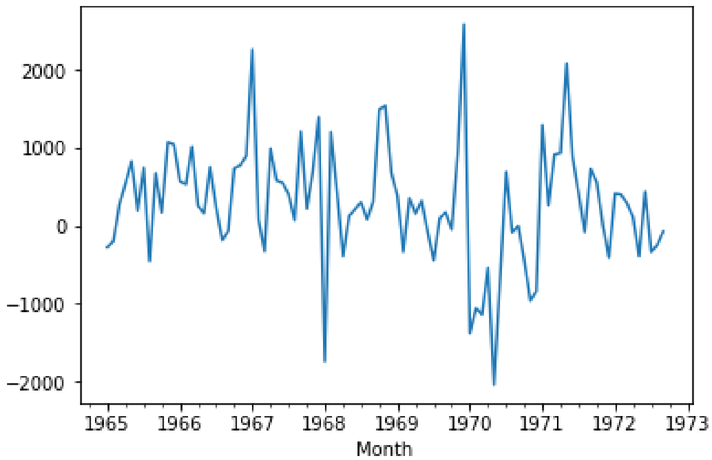
```
## Again test dickey fuller test
adfuller_test(df['Seasonal First Difference'].dropna())
```

```
ADF Test Statistic : -7.626619157213163
p-value : 2.060579696813685e-11
#Lags Used : 0
Number of Observations Used : 92
strong evidence against the null hypothesis, reject the null hypothesis. Data has no unit root and is stationary
```

In [193]:

```
df['Seasonal First Difference'].plot()
```

Out [193]: <matplotlib.axes._subplots.AxesSubplot at 0x1d2da817e80>



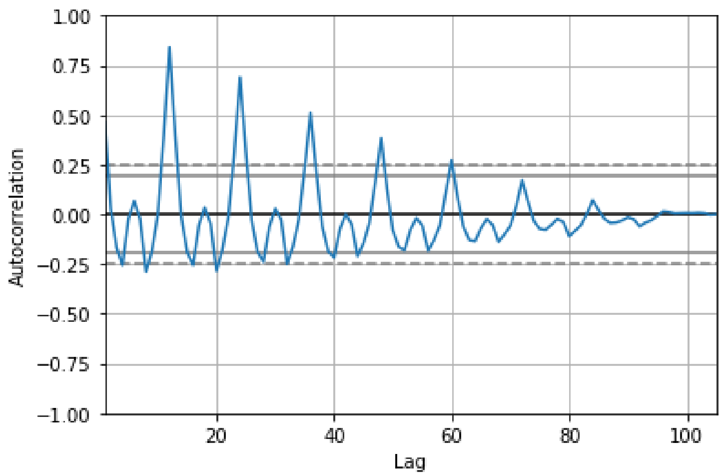
Auto Regressive Model

image.png

In [54]:

```
from pandas.tools.plotting import autocorrelation_plot
autocorrelation_plot(df['Sales'])
plt.show()
```

```
C:\Users\krish.naik\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:2:
FutureWarning: 'pandas.tools.plotting.autocorrelation_plot' is deprecated, import
'pandas.plotting.autocorrelation_plot' instead.
```



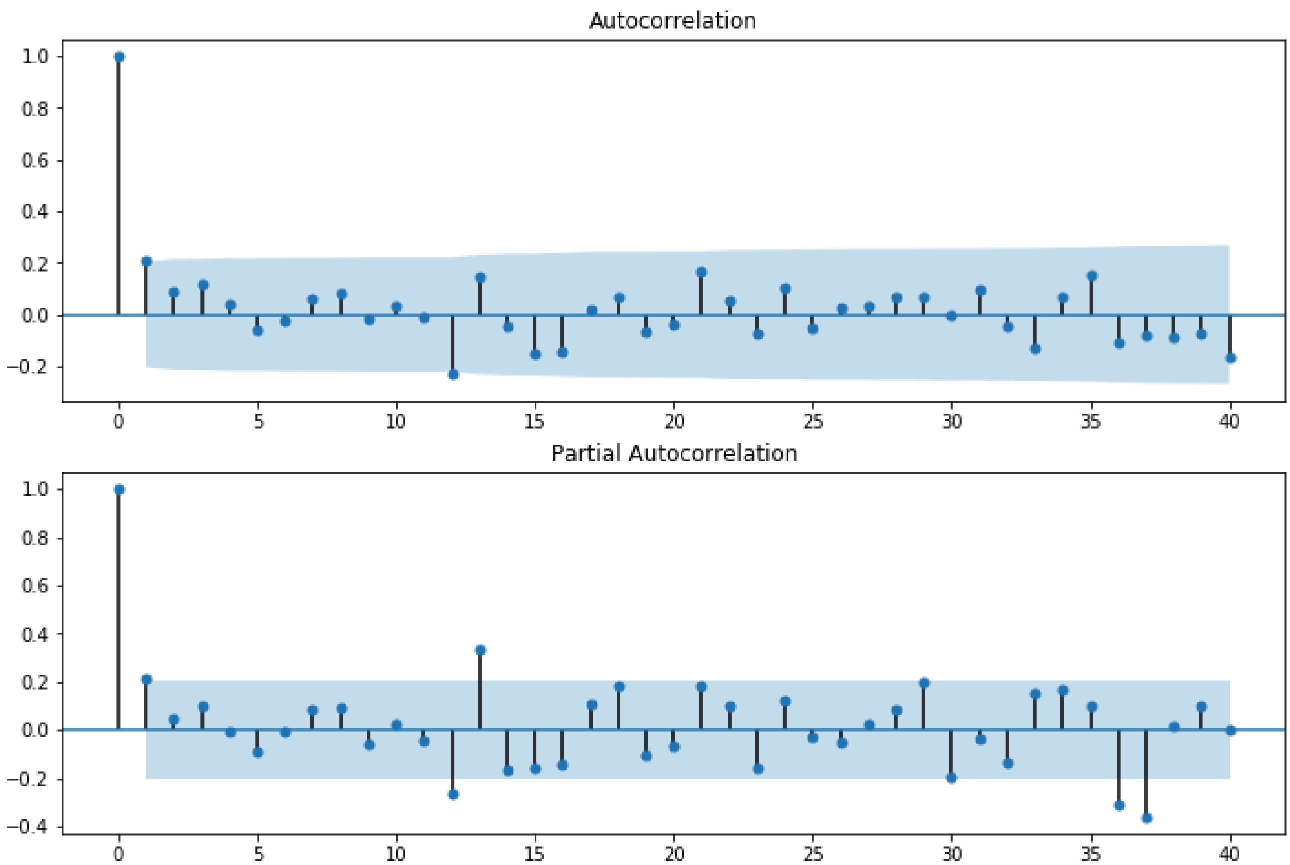
Final Thoughts on Autocorrelation and Partial Autocorrelation

- Identification of an AR model is often best done with the PACF.
    - For an AR model, the theoretical PACF “shuts off” past the order of the model. The phrase “shuts off” means that in theory the partial autocorrelations are equal to 0 beyond that point. Put another way, the number of non-zero partial autocorrelations gives the order of the AR model. By the “order of the model” we mean the most extreme lag of x that is used as a predictor.
- Identification of an MA model is often best done with the ACF rather than the PACF.
    - For an MA model, the theoretical PACF does not shut off, but instead tapers toward 0 in some manner. A clearer pattern for an MA model is in the ACF. The ACF will have non-zero autocorrelations only at lags involved in the model.

      p,d,q p AR model lags d differencing q MA lags

In [26]:
```
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
```

In [203]:
```
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(df['Seasonal First Difference'].iloc[13:],lags=40,ax=ax1
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(df['Seasonal First Difference'].iloc[13:],lags=40,ax=ax
```



In [115]:
```
# For non-seasonal data
#p=1, d=1, q=0 or 1
from statsmodels.tsa.arima_model import ARIMA
```

In [176]:
```
model=ARIMA(df['Sales'],order=(1,1,1))
model_fit=model.fit()
```

```
C:\Users\krish.naik\AppData\Local\Continuum\anaconda3\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:171: ValueWarning: No frequency information was provided, so
inferred frequency MS will be used.
  % freq, ValueWarning)
C:\Users\krish.naik\AppData\Local\Continuum\anaconda3\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:171: ValueWarning: No frequency information was provided, so
inferred frequency MS will be used.
  % freq, ValueWarning)
```

In [177]:
```
model_fit.summary()
```

Out [177]:

ARIMA Model Results

Dep. Variable:	D.Sales	**No. Observations:**	104
Model:	ARIMA(1, 1, 1)	**Log Likelihood**	-951.126
Method:	css-mle	**S.D. of innovations**	2227.262
Date:	Wed, 18 Mar 2020	**AIC**	1910.251

Time:	13:40:32	**BIC**	1920.829
Sample:	02-01-1964	**HQIC**	1914.536
	- 09-01-1972		

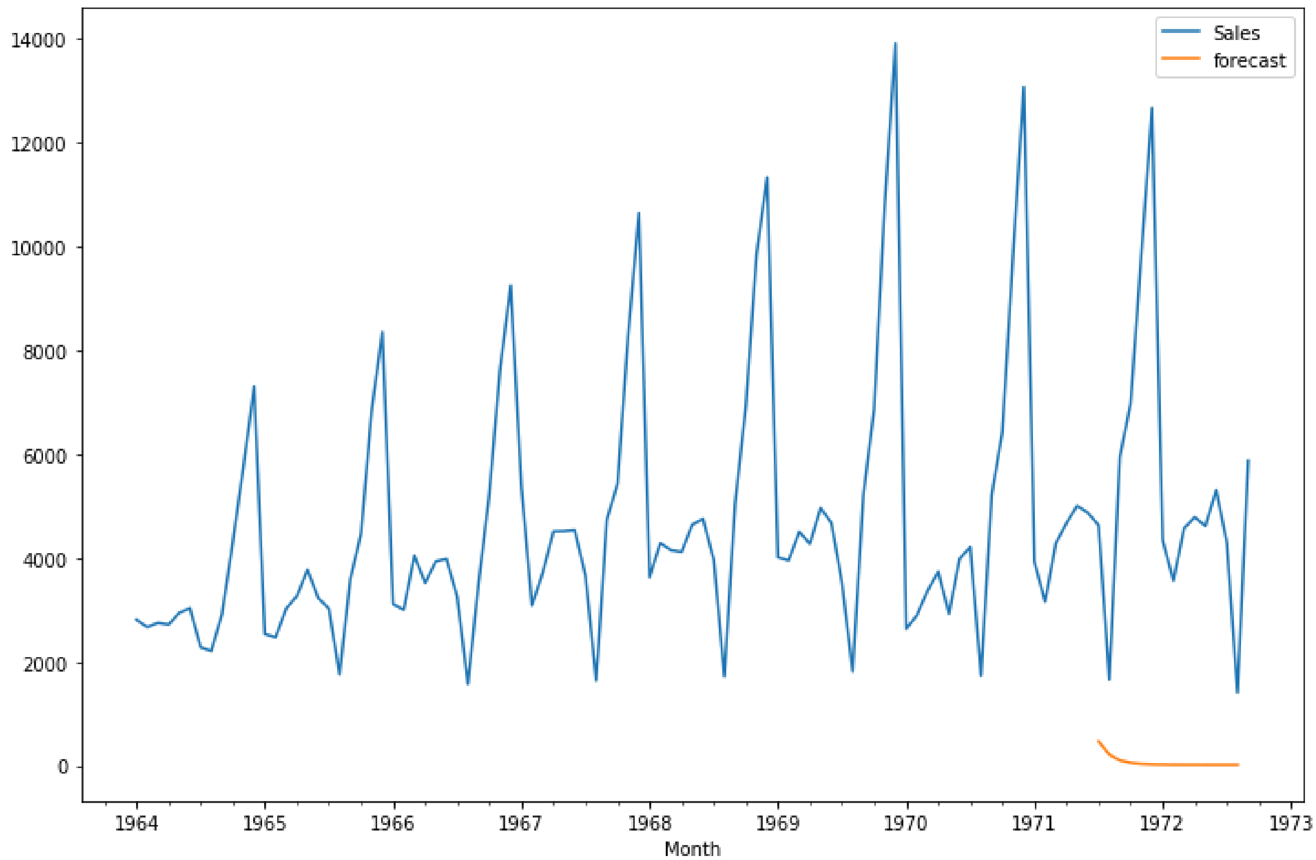
	coef	std err	z	P>|z|	[0.025	0.975]
const	22.7838	12.405	1.837	0.069	-1.530	47.098
ar.L1.D.Sales	0.4343	0.089	4.866	0.000	0.259	0.609
ma.L1.D.Sales	-1.0000	0.026	-38.503	0.000	-1.051	-0.949

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	2.3023	+0.0000j	2.3023	0.0000
MA.1	1.0000	+0.0000j	1.0000	0.0000

In [178]:
```
df['forecast']=model_fit.predict(start=90,end=103,dynamic=True)
df[['Sales','forecast']].plot(figsize=(12,8))
```

Out [178]: <matplotlib.axes._subplots.AxesSubplot at 0x1d2d9f1fda0>



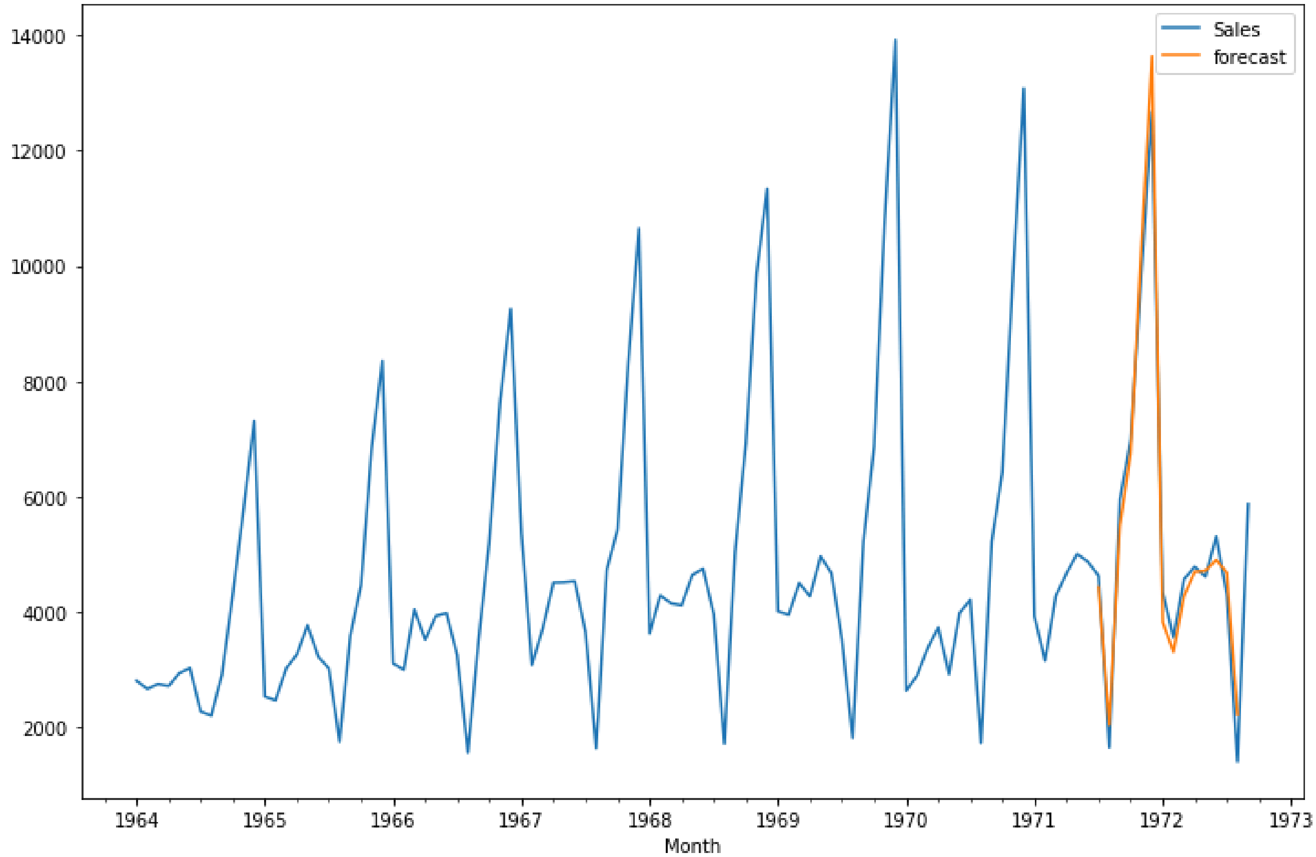
In [113]:
```
import statsmodels.api as sm
```

In [204]:
```
model=sm.tsa.statespace.SARIMAX(df['Sales'],order=(1, 1, 1),seasonal_order=(1,1,1,12))
results=model.fit()
```

```
C:\Users\krish.naik\AppData\Local\Continuum\anaconda3\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:171: ValueWarning: No frequency information was provided, so
inferred frequency MS will be used.
  % freq, ValueWarning)
```

In [205]:
```
df['forecast']=results.predict(start=90,end=103,dynamic=True)
df[['Sales','forecast']].plot(figsize=(12,8))
```

Out [205]: <matplotlib.axes._subplots.AxesSubplot at 0x1d2db70e278>



In [206]:
```
from pandas.tseries.offsets import DateOffset
future_dates=[df.index[-1]+ DateOffset(months=x)for x in range(0,24)]
```

In [207]:
```
future_datest_df=pd.DataFrame(index=future_dates[1:],columns=df.columns)
```

In [208]:
```
future_datest_df.tail()
```

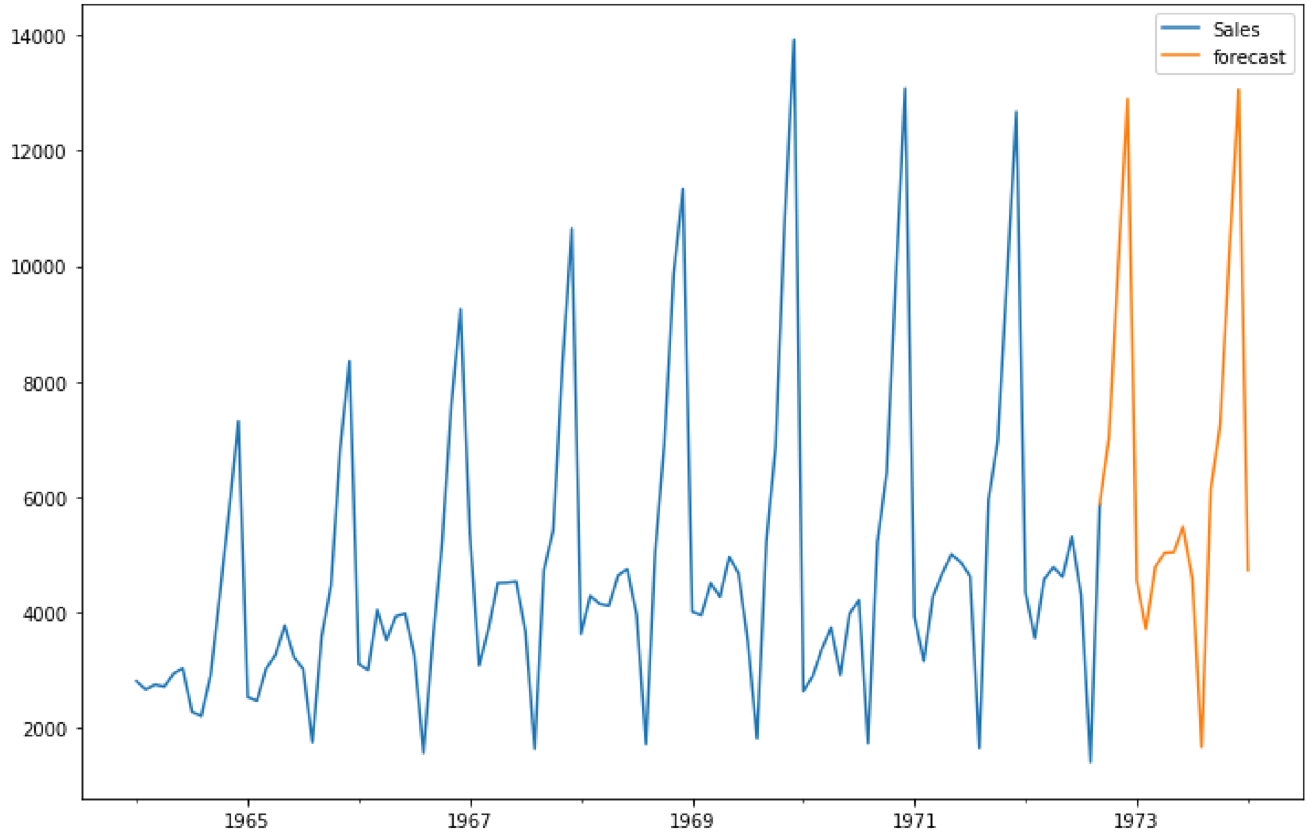
Out [208]:

	Sales	Sales First Difference	forecast	Seasonal First Difference
1974-04-01	NaN	NaN	NaN	NaN
1974-05-01	NaN	NaN	NaN	NaN
1974-06-01	NaN	NaN	NaN	NaN
1974-07-01	NaN	NaN	NaN	NaN
1974-08-01	NaN	NaN	NaN	NaN

In [209]:
```
future_df=pd.concat([df,future_datest_df])
```

In [201]:
```
future_df['forecast'] = results.predict(start = 104, end = 120, dynamic= True)
future_df[['Sales', 'forecast']].plot(figsize=(12, 8))
```

Out [201]: <matplotlib.axes._subplots.AxesSubplot at 0x1d2daee5048>



Sales
forecast
14000
12000
10000
8000
6000
4000
2000
1965
1967
1969
1971
1973

In [ ]: