Step 1 : Importing packages and insert chiali dataset csv file

In [36]:

```
import pandas as pd
from pandas import read_csv
from pmdarima.arima import auto_arima
from datetime import datetime
import matplotlib.pyplot as plt

sales_data = read_csv('Chiali_Sales.csv')
sales_data.head(3)
type(sales_data)
```

Out[36]:

```
pandas.core.frame.DataFrame
```

Step 2 : Preparing the data

After reading the dataset, make sure you do not have null values and change the data type of the 'Month' variable to a datetime object. Also, set the index of the dataframe to this variable using the set_index method.

In [37]:

```
sales_data['Month']=pd.to_datetime(sales_data['Month'])
sales_data.set_index('Month',inplace=True)
sales_data.head()
```

Out[37]:

	Chiali sales
Month	
2013-01-01	2815
2013-02-01	2672
2013-03-01	2755
2013-04-01	2721
2013-05-01	2946

Step3: Understanding the pattern

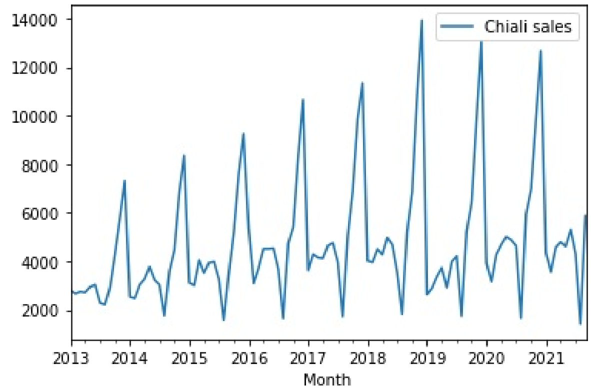
To understand the pattern of the data, we can simply plot using .plot() method.

In [38]:

```
sales_data.plot()
```

Out[38]:

```
<AxesSubplot:xlabel='Month'>
```



Step4: Test for Stationarity

Stationarity is an important concept in time-series and any time-series data should undergo a stationarity test before proceeding with a model. We use the Augmented Dickey-Fuller Test to check whether the data is stationary or not which is available in the ‘pmdarima’ package.

In [39]:

```
#Testing for stationarity
from pmdarima.arima import ADFTest
adf_test = ADFTest(alpha = 0.05)
adf_test.should_diff(sales_data)
```

Out[39]:

```
(0.01, False)
```

From the result Above , we can conclude that the data is non-stationary. Hence, we would need to use the “Integrated (I)” concept, denoted by value ‘d’ in time series to make the data stationary while building the Auto ARIMA model.

Step5: Train and Test split

Split into train and test datasets to build the model on the training dataset and forecast using the test dataset.

In [42]:

```
#Spliting the dataset into train and test


train = sales_data[:85]
test = sales_data[-20:]
```

In [41]:

```
test.head()
```

Out[41]:

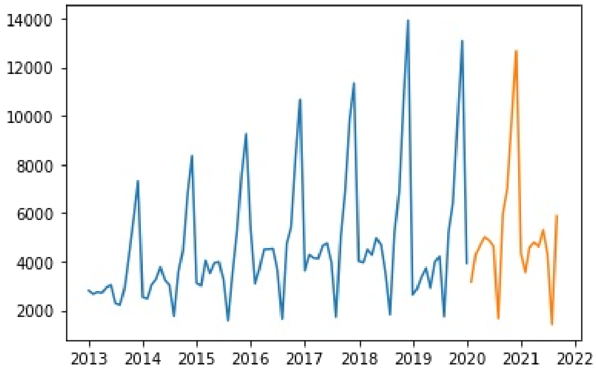
	Chiali sales
Month	
2020-02-01	3162
2020-03-01	4286
2020-04-01	4676
2020-05-01	5010
2020-06-01	4874

In [13]:

```
plt.plot(train)
plt.plot(test)
```

Out[13]:

```
[<matplotlib.lines.Line2D at 0x1a98247fcd0>]
```



<

Step6: Building Auto ARIMA model

Split into train and test datasets to build the model on the training dataset and forecast using the test dataset.

Auto-Regressive (p) -> Number of autoregressive terms. </br> Integrated (d) -> Number of nonseasonal differences needed for stationarity. </br> Moving Average (q) -> Number of lagged forecast errors in the prediction equation.

In the Auto ARIMA model, note that small p,d,q values represent non-seasonal components, and capital P, D, Q represent seasonal components. It works similarly like hyper tuning techniques to find the optimal value of p, d, and q with different combinations and the final values would be determined with the lower AIC, BIC parameters taking into consideration. Here, we are trying with the p, d, q values ranging from 0 to 5 to get better optimal values from the model. There are many other parameters in this model and to know more about the functionality.

In [44]:
```
arima_model =  auto_arima(train,start_p=0, d=1, start_q=0,
                          max_p=5, max_d=5, max_q=5, start_P=0,
                          D=1, start_Q=0, max_P=5, max_D=5,
                          max_Q=5, m=12, seasonal=True,
                          error_action='warn',trace = True,
                          supress_warnings=True,stepwise = True,
                          random_state=20,n_fits = 50 )
```

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=1203.853, Time=0.04 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=1192.025, Time=0.32 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=1176.246, Time=0.80 sec
 ARIMA(0,1,1)(0,1,0)[12]             : AIC=1174.731, Time=0.16 sec
 ARIMA(0,1,1)(1,1,0)[12]             : AIC=1176.034, Time=0.80 sec
 ARIMA(0,1,1)(1,1,1)[12]             : AIC=1176.700, Time=1.78 sec
 ARIMA(1,1,1)(0,1,0)[12]             : AIC=1175.054, Time=0.26 sec
 ARIMA(0,1,2)(0,1,0)[12]             : AIC=1174.769, Time=0.15 sec
 ARIMA(1,1,0)(0,1,0)[12]             : AIC=1194.721, Time=0.04 sec
 ARIMA(1,1,2)(0,1,0)[12]             : AIC=1174.564, Time=0.43 sec
 ARIMA(1,1,2)(1,1,0)[12]             : AIC=inf, Time=1.56 sec
 ARIMA(1,1,2)(0,1,1)[12]             : AIC=inf, Time=1.73 sec
 ARIMA(1,1,2)(1,1,1)[12]             : AIC=1176.645, Time=3.95 sec
 ARIMA(2,1,2)(0,1,0)[12]             : AIC=1176.127, Time=1.24 sec
 ARIMA(1,1,3)(0,1,0)[12]             : AIC=1176.124, Time=2.06 sec
 ARIMA(0,1,3)(0,1,0)[12]             : AIC=1176.458, Time=0.88 sec
 ARIMA(2,1,1)(0,1,0)[12]             : AIC=1176.656, Time=0.31 sec
 ARIMA(2,1,3)(0,1,0)[12]             : AIC=1180.591, Time=1.75 sec
 ARIMA(1,1,2)(0,1,0)[12] intercept   : AIC=inf, Time=0.44 sec

Best model:  ARIMA(1,1,2)(0,1,0)[12]
Total fit time: 18.724 seconds
```

Below is the summary of the model.

In [45]:
```
arima_model.summary()
```

Out[45]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	85
Model:	SARIMAX(1, 1, 2)x(0, 1, [], 12)	**Log Likelihood**	-583.282
Date:	Sun, 05 Sep 2021	**AIC**	1174.564
Time:	19:51:56	**BIC**	1183.670
Sample:	0	**HQIC**	1178.189
	- 85		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.8412	0.152	-5.543	0.000	-1.139	-0.544
ma.L1	0.0513	0.167	0.308	0.758	-0.275	0.378
ma.L2	-0.8673	0.086	-10.134	0.000	-1.035	-0.700
sigma2	5.862e+05	7.03e+04	8.342	0.000	4.48e+05	7.24e+05

Ljung-Box (L1) (Q):	0.05	**Jarque-Bera (JB):**	8.55
Prob(Q):	0.83	**Prob(JB):**	0.01
Heteroskedasticity (H):	2.61	**Skew:**	-0.10
Prob(H) (two-sided):	0.02	**Kurtosis:**	4.68

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

Step7: Forecasting on the test data

Using the trained model which was built in the earlier step to forecast the sales on the test data.

In [46]:

```
prediction = pd.DataFrame(arima_model.predict(n_periods = 20),index=test.index)
prediction.columns = ['predicted_sales']
prediction.head()
```

Out[46]:

Month	**predicted_sales**
2020-02-01	2746.685277
2020-03-01	3247.924303
2020-04-01	3592.488449
2020-05-01	2800.884067
2020-06-01	3841.886933

plt.figure(figsize=(8,5)) plt.plot(train,label="Training") plt.plot(test,label="Test") plt.plot(prediction,label="Predicted") plt.legend(loc = 'upper left') plt.show()

In [18]:

```
from sklearn.metrics import r2_score
test['predicted_sales'] = prediction
r2_score(test['Chiali sales'], test['predicted_sales'])
```

Out[18]:

```
0.8114687712309441
```