```
# import numpy as np
import pandas as pd

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

Cleaning dataset

```
df.drop(df.columns[[0, 1, 2, 3]], axis = 1, inplace = True)
df.head()
```

	Date	Value
0	01-01-2010	0.3
1	01-02-2010	0.0
2	01-03-2010	0.0
3	01-04-2010	0.0
4	01-05-2010	0.0

```
df.columns=["Date","Value"]
df.head()
```

	Date	Value
0	01-01-2010	0.3
1	01-02-2010	0.0
2	01-03-2010	0.0
3	01-04-2010	0.0
4	01-05-2010	0.0

```
df['Date']=pd.to_datetime(df['Date'],  format='%d-%m-%Y')
```

```
df.head()
```

	Date	Value
0	2010-01-01	0.3
1	2010-02-01	0.0
2	2010-03-01	0.0
3	2010-04-01	0.0
4	2010-05-01	0.0

```
df.set_index('Date',inplace=True)
```

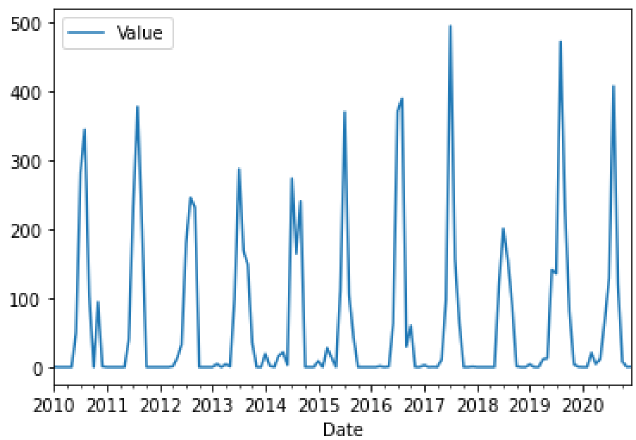
```
df.head()
```

Date	Value
2010-01-01	0.3
2010-02-01	0.0
2010-03-01	0.0
2010-04-01	0.0
2010-05-01	0.0

Visualize the Data

```
df.plot()
```

<AxesSubplot:xlabel='Date'>



Stationarize the series

```
### Testing For Stationarity

from statsmodels.tsa.stattools import adfuller


#Ho: It is non stationary
#H1: It is stationary

def adfuller_test(values):
    result=adfuller(values)
    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 hypothesis. Data has
    else:
        print("weak evidence against null hypothesis, time series has a unit root, indicating it is


adfuller_test(df['Value'])
```

```
ADF Test Statistic : -2.9192061653136365
p-value : 0.04315973392525896
#Lags Used : 11
Number of Observations Used : 120
strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data has no unit
```

```
#Differencing
df['First Difference'] = df['Value'] - df['Value'].shift(1)
```

```
df['Value'].shift(1)
```

```
Date
2010-01-01      NaN
2010-02-01      0.3
2010-03-01      0.0
2010-04-01      0.0
2010-05-01      0.0
              ...
2020-08-01    128.4
2020-09-01    407.0
2020-10-01    121.9
2020-11-01      7.6
2020-12-01      0.2
Name: Value, Length: 132, dtype: float64
```

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

```
df.head(20)
```

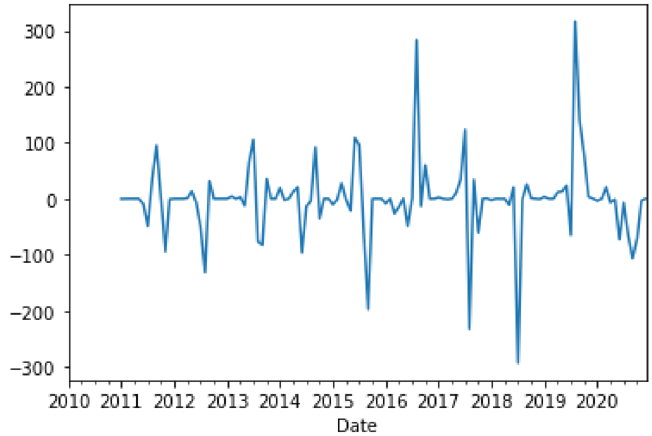
Date	Value	First Difference	Seasonal First Difference
2010-01-01	0.3	NaN	NaN
2010-02-01	0.0	-0.3	NaN
2010-03-01	0.0	0.0	NaN
2010-04-01	0.0	0.0	NaN
2010-05-01	0.0	0.0	NaN
2010-06-01	49.6	49.6	NaN
2010-07-01	280.8	231.2	NaN
2010-08-01	343.8	63.0	NaN
2010-09-01	104.1	-239.7	NaN
2010-10-01	0.0	-104.1	NaN
2010-11-01	94.2	94.2	NaN
2010-12-01	0.9	-93.3	NaN
2011-01-01	0.0	-0.9	-0.3
2011-02-01	0.0	0.0	0.0
2011-03-01	0.0	0.0	0.0
2011-04-01	0.0	0.0	0.0
2011-05-01	0.0	0.0	0.0
2011-06-01	39.7	39.7	-9.9
2011-07-01	232.1	192.4	-48.7
2011-08-01	376.8	144.7	33.0

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

```
ADF Test Statistic : -10.402375786755307
p-value : 1.8947515284875764e-18
#Lags Used : 0
Number of Observations Used : 119
strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data has no unit
```

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

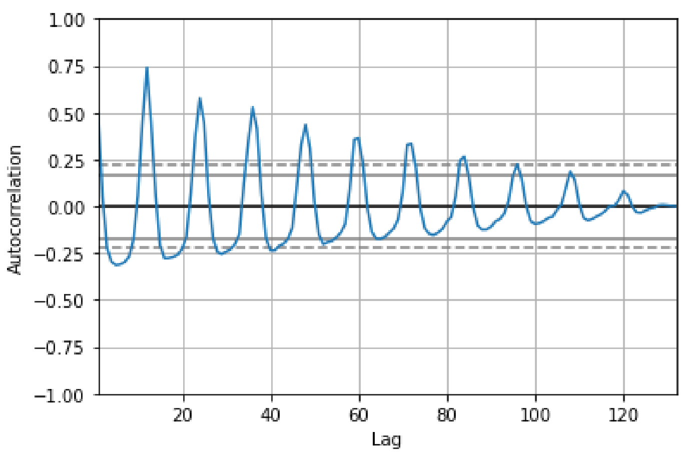
<AxesSubplot:xlabel='Date'>



Auto Regressive Model

An autoregressive model of order p can be written as image.png

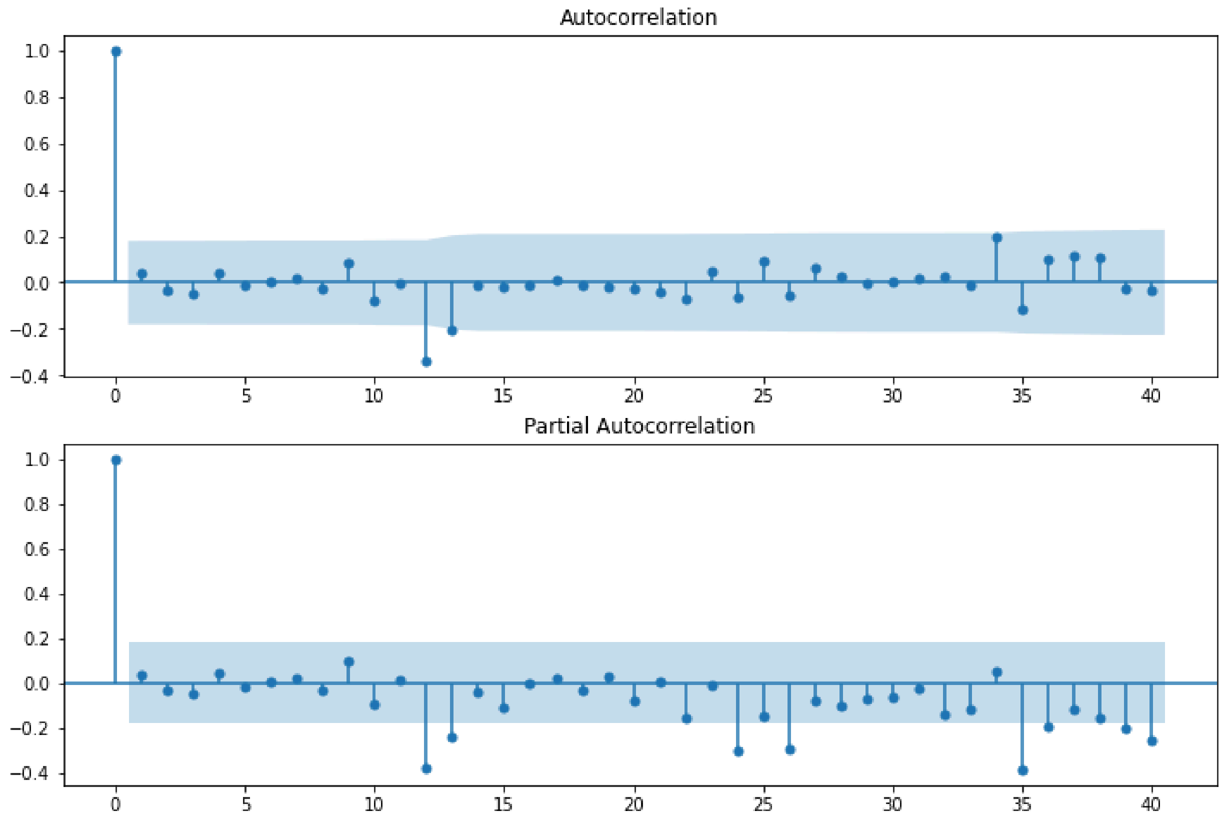
```
from pandas.plotting import autocorrelation_plot
autocorrelation_plot(df['Value'])
plt.show()
```




```
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
```

```
import statsmodels.api as sm

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=ax2)
```



ARIMA prediction plot

```
# arima pdq order
#p=1, d=1, q=0 or 1
from statsmodels.tsa.arima_model import ARIMA


import warnings

warnings.filterwarnings('ignore')

model=ARIMA(df['Value'],order=(1,1,0))
model_fit=model.fit()


model_fit.summary()
```

ARIMA Model Results

Dep. Variable:	D.Value	**No. Observations:**	131
Model:	ARIMA(1, 1, 0)	**Log Likelihood**	-805.555
Method:	css-mle	**S.D. of innovations**	113.332
Date:	Wed, 03 Nov 2021	**AIC**	1617.110
Time:	09:32:10	**BIC**	1625.736
Sample:	02-01-2010 - 12-01-2020	**HQIC**	1620.615

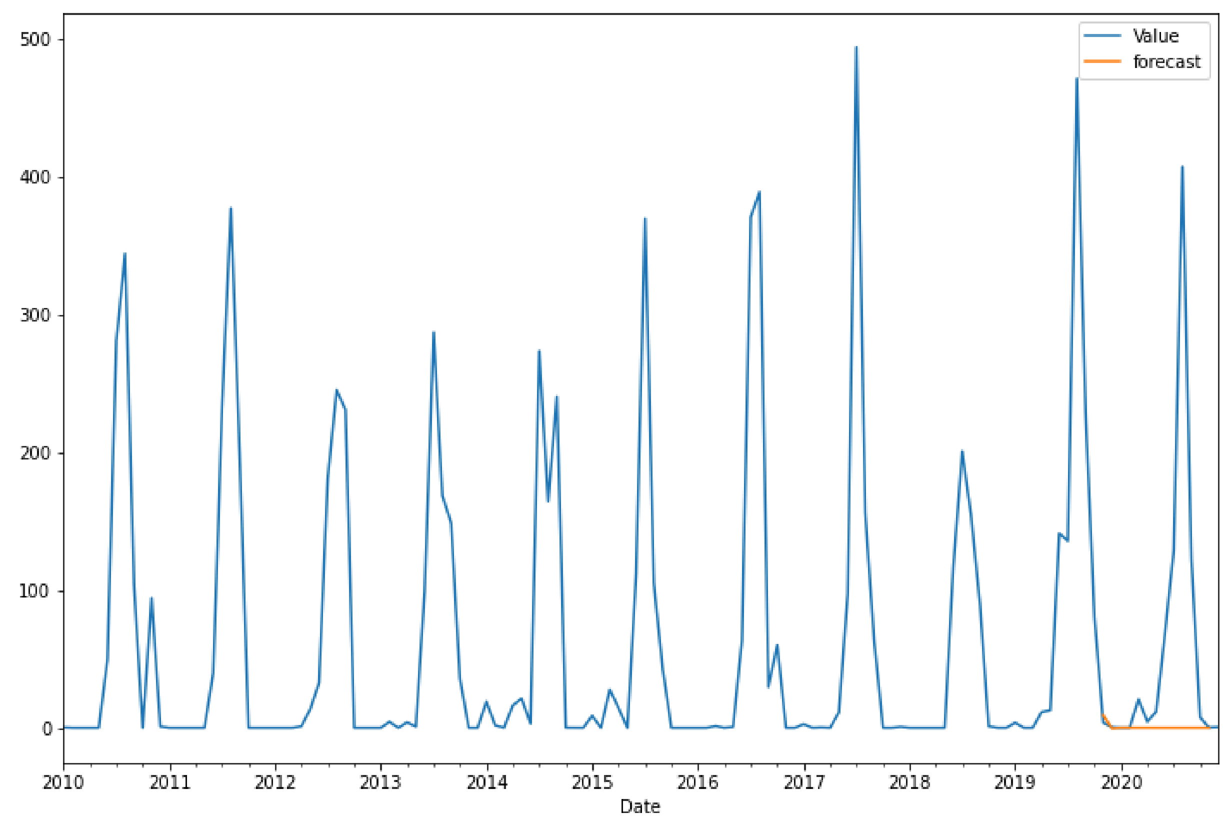
	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	0.0016	9.334	0.000	1.000	-18.292	18.295
ar.L1.D.Value	-0.0614	0.087	-0.706	0.480	-0.232	0.109

Roots

	Real	**Imaginary**	**Modulus**	**Frequency**
AR.1	-16.2978	+0.0000j	16.2978	0.5000

```
df['forecast']=model_fit.predict(start=118,end=130,dynamic=True)
df[['Value','forecast']].plot(figsize=(12,8))
```

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

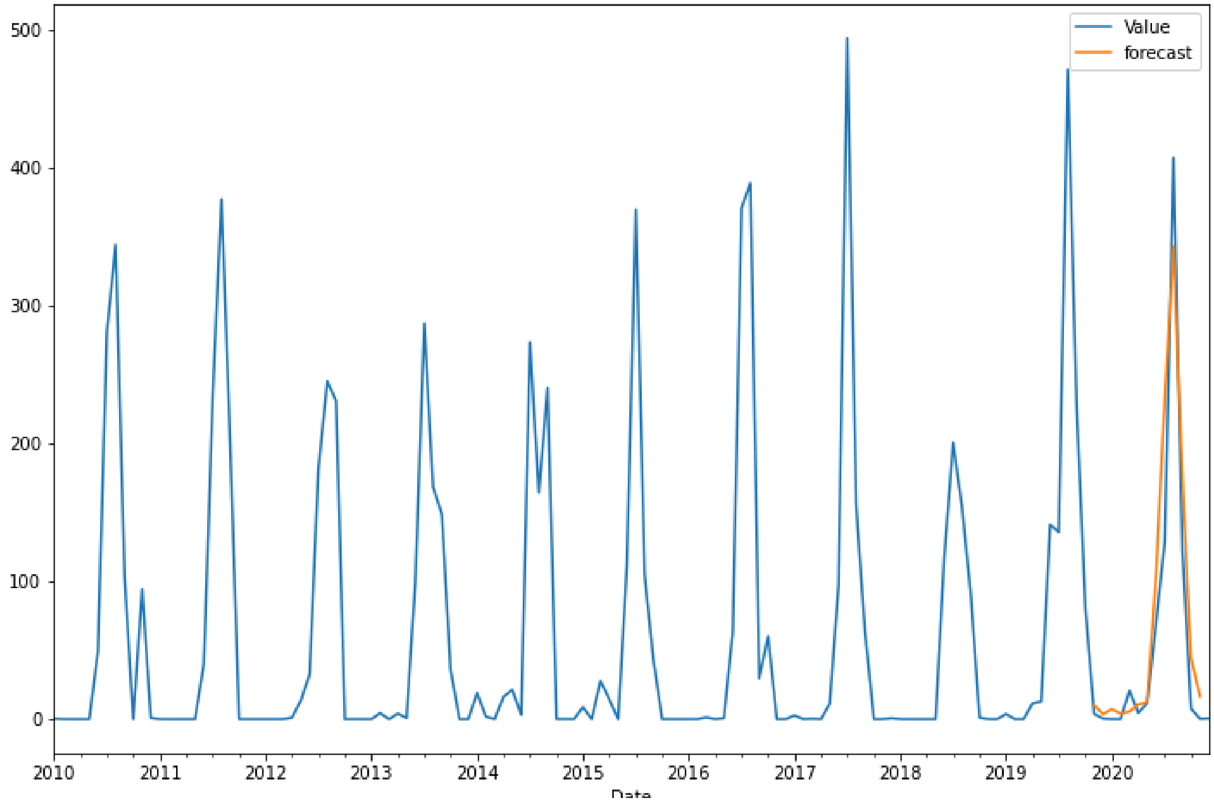


SARIMA prediction plot

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

```
df['forecast']=results.predict(start=118,end=130,dynamic=True)
df[['Value','forecast']].plot(figsize=(12,8))
```

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




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

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

```
future_datest_df.tail()
```

	Value	First Difference	Seasonal First Difference	forecast
2022-07-01	NaN	NaN	NaN	NaN
2022-08-01	NaN	NaN	NaN	NaN
2022-09-01	NaN	NaN	NaN	NaN